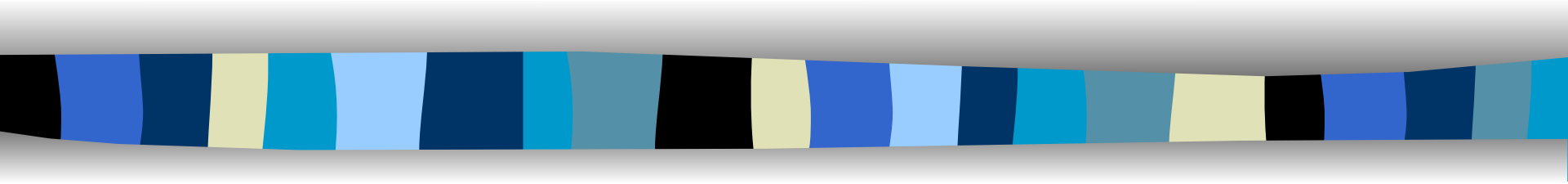
PHP and MYSQL



CSCI 3000
Web Programming

Dr. Luis A. Cueva-Parra

Connecting MySQL with PHP

- Since PHP 5, we can connect MySQL using:
  - MySQLi extension, or
  - PDO (PHP Data Objects)
- MySQLi works with SQL databases
- PDO works with 12 different databases
- PDO more flexible if we change databases
- Both are object-oriented, but MySQLi also offers a procedural API.
- Both support Prepared Statements. Prepared Statements protect from SQL injection

Open a Connection to MySQL

- Using MySQLi Object-Oriented:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error); }
echo "Connected successfully";    ?>
```

Open a Connection to MySQL

- Using MySQLi Procedural:

```
<?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";    ?>
```

Open a Connection to MySQL

- Using PDO:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
try {   $conn = new
PDO("mysql:host=$servername;dbname=myDB",
$username, $password);
 // set the PDO error mode to exception
 $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
}
```

Open a Connection to MySQL

- Using PDO (continued):

```
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }
?>
```

Close a Connection to MySQL

- Using MySQLi Object-Oriented:

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

- Using MySQLi Procedural:

```
<?php
 mysqli_close($conn);     ?>
```

- Using PDO:

```
<?php
 $conn = null;     ?>
```

Create a MySQL Database

- Using MySQLi Object-Oriented:

```
<?php
// Create and test connection
// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

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

Create a MySQL Database

- Using MySQLi Procedural:

```
<?php
// Create and test connection
// 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);  ?>
```

Create a MySQL Database

- Using PDO:

```
<?php
// Define variables
try {  $conn = new PDO("mysql:host=$servername", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "CREATE DATABASE myDBPDO";
    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Database created successfully<br>";   }
```

Create a MySQL Database

- Using PDO: (continued)

```
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

Create a MySQL Table

- Use the CREATE TABLE statement:

```
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
)
```

- It creates a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date"

Create a MySQL Table

- Optional **attributes** for each column:
- NOT NULL - Each row must contain a value, null values are not allowed.
- DEFAULT value - Set a default value that is added when no other value is passed.
- UNSIGNED - For number types, limits the stored data to positive numbers and zero.
- AUTO INCREMENT - by 1 each time a new record is added.
- PRIMARY KEY - Uniquely identify the rows in a table. Column ID number, with AUTO_INCREMENT.

Create a MySQL Table

- Using MySQLi Object-Oriented: (1/3)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);  }
```

Create a MySQL Table

- Using MySQLi Object-Oriented: (2/3)

```
// 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
)";
```

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

Create a MySQL Table

- Using MySQLi Object-Oriented: (3/3)

```
if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

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

Create a MySQL Table

- Using MySQLi Procedural: (1/3)

```
<?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()); }
```

Create a MySQL Table

- Using MySQLi Procedural: (2/3)

```
// 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
)";
```

Create a MySQL Table

- Using MySQLi Procedural: (3/3)

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

mysqli_close($conn);
?>
```

Create a MySQL Table

- Using PDO: (1/3)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";
try {
    $conn = new
PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
```

Create a MySQL Table

- Using PDO: (2/3)

```
// 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
  )";
  // use exec() because no results are returned
  $conn->exec($sql);
  echo "Table MyGuests created successfully";       }
```

Create a MySQL Table

- Using PDO: (3/3)

```
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

Insert Data Into MySQL

- Syntax rules:
  - 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.
- Use the INSERT INTO statement to add new records:

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

Insert data into MySQL Table

- Using MySQLi Object-Oriented: (1/2)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);  }
```

Insert data into MySQL Table

- Using MySQLi Object-Oriented: (2/2)

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

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

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

Insert data into MySQL Table

- Using MySQLi Procedural: (1/2)

```
<?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()); }
```

Insert data into MySQL Table

- Using MySQLi Procedural: (2/2)

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

mysqli_close($conn);
?>
```

Insert data into MySQL Table

- Using PDO: (1/2)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";
try {  $conn = new
PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $sql = "INSERT INTO MyGuests (firstname, lastname,
email)
```

Insert data into MySQL Table

- Using PDO: (2/2)

```
    VALUES ('John', 'Doe', 'john@example.com')";
    // use exec() because no results are returned
    $conn->exec($sql);
    echo "New record created successfully";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }
$conn = null;
?>
```

Get ID of The Last Inserted Record

- If we perform INSERT or UPDATE on a table with an AUTO_INCREMENT field, we can get the ID of the last inserted/updated record immediately.
- MySQLi Object-Oriented

```
if ($conn->query($sql) === TRUE) {
    $last_id = $conn->insert_id;
    echo "New record created successfully. Last inserted ID is: " . $last_id;
} else ....
```

Get ID of The Last Inserted Record

- MySQLi Procedural

```
if (mysqli_query($conn, $sql)) {
    $last_id = mysqli_insert_id($conn);
    echo "New record created successfully. Last inserted ID is: " . $last_id;
} else ...
```

Get ID of The Last Inserted Record

- MySQLi PDO

```
    // use exec() because no results are returned
    $conn->exec($sql);
    $last_id = $conn->lastInsertId();
    echo "New record created successfully. Last inserted ID is: " . $last_id;
    }
 ...
```

Get ID of The Last Inserted Record

- If we perform INSERT or UPDATE on a table with an AUTO_INCREMENT field, we can get the ID of the last inserted/updated record immediately.
- MySQLi Object-Oriented

```
if ($conn->query($sql) === TRUE) {
    $last_id = $conn->insert_id;
    echo "New record created successfully. Last inserted ID is: " . $last_id;
} else ....
```

Insert Multiple Records Into MySQL

- MySQLi Object-Oriented

```
$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');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";
if ($conn->multi_query($sql) === TRUE) {
    echo "New records created successfully";
} else ...
```

Insert Multiple Records Into MySQL

- MySQLi Procedural

```
$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');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";
if (mysqli_multi_query($conn, $sql)) {
    echo "New records created successfully";
} else
```

Insert Multiple Records Into MySQL

- MySQLi PDO (1/2)

```
    // begin the transaction
     $conn->beginTransaction();
     // our SQL statements
     $conn->exec("INSERT INTO MyGuests (firstname,
lastname, email)
     VALUES ('John', 'Doe', 'john@example.com')");
     $conn->exec("INSERT INTO MyGuests (firstname,
lastname, email)
     VALUES ('Mary', 'Moe', 'mary@example.com')");
      // commit the transaction
     $conn->commit();
     echo "New records created successfully";    }
```

Insert Multiple Records Into MySQL

- MySQLi PDO (2/2)

```
    // catch(PDOException $e)
     {
     // roll back the transaction if something failed
     $conn->rollback();
     echo "Error: " . $e->getMessage();
     }

$conn = null;
?>
```

PHP Prepared Statements

- A prepared statement is a compiled template for executing SQL statements with different input values repeatedly with high efficiency.
- Process:
  - **Prepare:** An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?")

    Example:

    ```
    INSERT INTO MyGuests VALUES(?, ?, ?)
    ```

PHP Prepared Statements

- Process (continued):
  - **Compilation/Optimization:** The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it.
  - **Execution:** The application binds the values to the parameters, and the database executes the statement.

PHP Prepared Statements

- Prepared statements advantages:
  - Reduce parsing time.
  - Bound parameters traffic is reduced compared with the traffic of the complete query.
  - Increase security. It is useful against SQL injections.

PHP Prepared Statements

- Using MySQLi Object-Oriented: (1/4)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);  }
```

PHP Prepared Statements

- Using MySQLi Object-Oriented: (2/4)

```
// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();
```

PHP Prepared Statements

- Using MySQLi Object-Oriented: (3/4)

```
$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();
```

PHP Prepared Statements

- Using MySQLi Object-Oriented: (4/4)

```
echo "New records created successfully";

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

PHP Prepared Statements

- The first argument in the **bind_param()** function contains a string with characters of the following types:

  i - integer

  d - double

  s - string

  b - BLOB

- We must have one of these for each parameter value.
- By telling MySQL what type of data to expect, we minimize the risk of SQL injections.

Select Data From a MySQL Database

- The SELECT statement is used to select data from one or more tables.

  SELECT column_name(s) FROM table_name

- We can use the * character to select ALL columns from a table:

  SELECT * FROM table_name

Delete Data From a MySQL Table

- The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
WHERE some_column = some_value
```

- The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

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

Update Data in MySQL Table

- The UPDATE statement is used to update existing records in a table:

  ```
  UPDATE table_name
  SET column1=value, column2=value2, ...
  WHERE some_column=some_value
  ```

- Example:

  ```
  $sql = "UPDATE MyGuests SET
  lastname='Doe' WHERE id=2";
  ```

Limit Data Selections From MySQL Database

- MySQL provides a LIMIT clause that is used to specify the number of records to return.
- Select all records from 1 - 30 (inclusive) from a table called "Orders"

```
$sql = "SELECT * FROM Orders LIMIT 30";
```

- Return only 10 records, start on record 16 (OFFSET 15)

```
$sql = "SELECT * FROM Orders LIMIT 10 OFFSET 15";
```