WEB PROGRAMMING AND APPLICATIONS

(503073)

WEEK 8

Prepared by Mai Van Manh

1. XAMPP: Please see this video at SAKAI.

2. PHP My Admin: Please see this video and this video at SAKAI.

3. CONNECT TO MySQL

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.

Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

3.1. MySQLi Object-Oriented:

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

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

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

3.2 PDO:

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

   try {
      $conn = new PDO("mysql:host=$servername;dbname=teb_database", $username, $password);
      // set the PDO error mode to exception
      $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
      echo "Connected successfully";
   }
   catch(PDOException $e)
   {
      echo "Connection failed: " . $e->getMessage();
}
?>
```

4. INSERT DATA

4.1 MySQLi:

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

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

      $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();
?>
```

4.2 PDO:

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

      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);
         $sql = "INSERT INTO MyGuests (firstname, lastname, email)
         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;
?>
```

5. LOAD DATA

5.1 MySQLi:

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

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

    $sql = "SELECT id, firstname, lastname FROM MyGuests";
    $result = $conn->query($sql);

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

5.2 PDO:

```
<?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);
  }

  $sql = "SELECT id, firstname, lastname FROM MyGuests";
  $result = $conn->query($sql);
```

```
   if ($result->num_rows > 0) {
      echo "<table><tr><th>ID</th><th>Name</th></tr>";
      // output data of each row
      while($row = $result->fetch_assoc()) {
         echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]."
".$row["lastname"]."</td></tr>";
      }
      echo "</table>";
   } else {
      echo "0 results";
   }
   $conn->close();
?>
```

6. DELETE DATA

6.1 MySQLi:

```
<?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);
   }

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

   if ($conn->query($sql) === TRUE) {
      echo "Record deleted successfully";
   } else {
      echo "Error deleting record: " . $conn->error;
   }

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

6.1 PDO:

```
<?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);

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

      // use exec() because no results are returned
      $conn->exec($sql);
```

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

    $conn = null;
?>
```

7. UPDATE DATA

7.1 MySQLi:

```
<?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);
    }

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

    if ($conn->query($sql) === TRUE) {
        echo "Record updated successfully";
    } else {
        echo "Error updating record: " . $conn->error;
    }

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

7.2 PDO:

```
<?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);

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

        // Prepare statement
        $stmt = $conn->prepare($sql);

        // execute the query
        $stmt->execute();

        // echo a message to say the UPDATE succeeded
        echo $stmt->rowCount() . " records UPDATED successfully";
    }
```

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

   $conn = null;
?>
```

8. PREPARED STATEMENTS AND BOUND PARAMETERS

A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency. Prepared statements basically work like this:

- 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(?, ?, ?).
- The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it.
- Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values.

Compared to executing SQL statements directly, prepared statements have three main advantages:

1. Prepared statements reduces parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
2. Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query
3. Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

MySQLi:

```
<?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);
  }

  // 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();

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

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

  echo "New records created successfully";

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

PDO:

```
<?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);

    // prepare sql and bind parameters
    $stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
    VALUES (:firstname, :lastname, :email)");
    $stmt->bindParam(':firstname', $firstname);
    $stmt->bindParam(':lastname', $lastname);
    $stmt->bindParam(':email', $email);

    // insert a row
    $firstname = "John";
    $lastname = "Doe";
    $email = "john@example.com";
    $stmt->execute();
    // insert another row
    $firstname = "Mary";
    $lastname = "Moe";
    $email = "mary@example.com";
    $stmt->execute();

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

    echo "New records created successfully";
  }
  catch(PDOException $e)
  {
    echo "Error: " . $e->getMessage();
  }
  $conn = null;
?>
```

9. CREATE MYSQL TABLES

MySQLi:

```
<?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);
  }

  // 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
  )";

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

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

PDO:

```
<?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);

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

  $conn = null;
?>
```

10. GET ID OF THE LAST INSERTED RECORD

MySQLi:

```
<?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);
  }

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

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

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

PDO:

```
<?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);
    $sql = "INSERT INTO MyGuests (firstname, lastname, email)
    VALUES ('John', 'Doe', 'john@example.com')";
    // 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;
  }
  catch(PDOException $e)
  {
    echo $sql . "<br>" . $e->getMessage();
  }

  $conn = null;
?>
```

REFERENCES

[1]. W3School – PHP: https://www.w3schools.com/php