Agenda:

• phpMyAdmin
• MySQLi

phpMyAdmin

• Before you can put your data into a table, that table should exist.
• A table, however, cannot exist on its own; it should belong to a certain database.
• A database belongs to a user account of an application called Database Management System (DBMS)
A DBMS can have one or more user accounts.

A user account in a DBMS can have one or more databases.

A DBMS is a software for creating and managing (create, retrieve and update data) databases.

phpMyAdmin

- A DBMS can have one or more user accounts
- A user account in a DBMS can have one or more databases
- A DBMS is a software for creating and managing (create, retrieve and update data) databases.

phpMyAdmin

- phpMyAdmin is a tool that enables you to easily interact with MySQL DBMS (often referred to as MySQL database)
- Through phpMyAdmin, interaction to MySQL is through a Web browser.
- To access phpMyAdmin in your local machine you type: `localhost/phpmyadmin` in the address bar of a Web browser

phpMyAdmin

- By default, a root user account is used
- This account does not have a password and hence you will not asked to type a password
- Once you are in this account, you can create a database and a table
- Once you have created a table, you can change its structure, insert, delete, update etc
Introduction to MySQLi

- MySQLi is an extension to MySQL
- MySQLi stands for MySQL improved
- MySQLi offers many benefits such as;
  - An Object-Oriented interface
  - Support prepared statements which are useful for security purposes

We mainly use `mysql` class to establish a connection with a database and thereafter manipulate it

- `new` operator is used to create a new instance/object of `mysqli` class
  - Thus, the newly created instance inherit all attributes and functions (methods) defined in the `mysqli` class

The resultant instance/object, which is stored in a memory somewhere, can be accessed through the assigned variable (in this case `$conn`)
  - This variable is called reference variable
Introduction to MySQLi

• To access an attribute or a method of an object, object operator (->) is used.

• This is done by typing the name of a reference variable followed by the object operator and name of an attribute or a method.

Establishing a DB Connection

• To insert or retrieve data from a database, you first need to establish a connection to the database.

• To establish a connection, we need to know four things:
  – Host (where a DBMS is hosted)
  – Username
  – Password
  – Database name

Establishing a DB Connection

• We are going to use mysqli class to create an object.

• This object is created by specifying host, username, password and database name.
Establishing DB Connection

```php
<?php
    $host = "localhost";
    $username = "root";
    $pwd = "";
    $db = "mazoezi";
    $conn = new mysqli($host, $username, $pwd, $db);
?>

* Assumption here is that a database called "mazoezi" exists in the MySQL DBMS
```

Inserting Data in a Table

- Once a connection is established, data in database tables can be inserted, retrieved, deleted or updated
- We can insert data into a table in two ways;
  - Using old traditional way
  - Using prepared statements

Using Old Traditional Way

```php
<?php
    $conn = new mysqli('localhost', 'root', '', 'csexercise');
    $fname = $_POST['fname'];
    $sname = $_POST['sname'];
    $query = "INSERT INTO student values ('$fname', '$sname')";
    $conn->query($query);
    $conn->close();
?>
```

* Assumption here is that the data comes from an HTML form that has two textboxes called 'fname' and 'sname'
Using Old Traditional Way

• First you need to define your insert statement and store it in a variable as a string (line 6)

• Then use query method (line 7) to execute the query
  – The query method takes string as an argument (in this case the insert statement)
  – The query method, in this case, returns true if the query is executed without an error and false if otherwise

Using Old Traditional Way

• Thereafter use close method (line 8) of mysqli instance to close the database connection

• This method, however, is prone to SQL injection and hence not secure

Using Prepared Statements

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$conn = new mysqli($servername, $username, $password);
if ($conn->connect_error) {
    die("Connection failed: ", $conn->connect_error);
}

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $fname = $POST['fname'];
    $lname = $POST['lname'];
    $query = "INSERT INTO student VALUES(?,?)";
    $stmt = $conn->prepare($query);
    $stmt->bind_param("ss", $fname, $lname);
    $stmt->execute();
    $stmt->close();
    $conn->close();
}
?>
```

*Assumption here is that the data comes from an HTML form that has two textboxes called ‘fname’ and ‘lname’
Using Prepared Statements

• First you need to define your insert statement and store it in a variable as a string (line 6)

• You then need to use a prepare method (or function) of mysqli instance (or object) to prepare the statement (line 7)
  – The prepare method takes string as an argument (in this case the insert statement)
  – The prepare method returns an object (thus a reference variable needs to be used)

Using Prepared Statements

• Thereafter, use bind_param method (line 8) of the object created in line 7 to bind arguments (i.e. values) with parameters (question marks in the insert statement)
  – The first argument of the bind_param method specify data types of the arguments
  – In this case the data type of the two arguments is string

Using Prepared Statements

• Then, use execute method (line 9) of the object created in line 7 to execute the prepared statement

• Then use close method (line 10) of the object created in line 7 to close the prepared statement

• Thereafter use close method (line 11) of mysqli instance to close the database connection
Retrieving Data from a Table

- First you need to define your select statement and store it in a variable as a string (line 4).
- You then need to use a `query` method of `mysqli` instance to execute the query (line 5).
  - The `query` method takes string as an argument (in this case the select statement).
  - The `query` method, in this case, returns an object (thus a reference variable needs to be used).

```
<?php
    $conn = new mysqli('localhost', 'root', '', 'ce06exercise');
    $sql = "select * from student";
    $stmt = $conn->query($sql);
    while ($row = $stmt->fetch_assoc()) {
        echo $row['firstname'] . ": " . $row['surname'] . '<br />
    }
    $conn->close();
```

Retrieving Data from a Table

- The object created in line 5 contains all rows (with their attributes) from the table.
- We want to access each row and display its attributes.
- To do this, we need to use loop and a `fetch_assoc` method (line 6) of the object created in line 5.
Retrieving Data from a Table

- Since the target table might be empty, we use a while loop to ensure we are not attempting to display an empty row.
- The `fetch_assoc` method returns a row of a table from the object created in line 5.
- Thus, the returned row (which is an array of attributes) needs to be stored in order to display its attributes (thus a variable needs to be used).

Retrieving Data from a Table

- In each loop, items of a returned array are displayed by referring to the names of the items in the array (line 7).
  - In this case, names of the items are names of attributes as specified in the table.
- Thereafter, use the `close` method (line 10) of `mysqli` instance to close the database connection.