Week 10 – COMP1230

Lab Exercise: PHP OOP Fundamentals

Exercise 1: Defining a Class with a Constructor

Objective:

Create a PHP class named **Book** with a constructor to initialize book properties.

Task:

Define a class with private properties for **title, author,** and **isbn.** The constructor should take parameters for each of these properties and initialize them.

Hints:

- Use **private $property_name;** to define a property.
- The constructor is a function named **__construct** within the class.

Exercise 2: Accessor Methods

Objective:

Implement getter and setter methods to interact with private properties.

Task:

Add **getTitle, setTitle**, and similar accessor methods for **author** and **isbn.**

Hints:

- Getter for a property **title** is **getTitle** and setter is **setTitle.**
- Validate input in setter methods before assigning properties. Make sure the title is string and not empty.

Exercise 3: Static Properties and Methods

Objective:

Understand and use static properties and methods.

Task:

Add a static property **count** to track the number of books created and a static method **getCount** to retrieve this count.

Hints:

- Use **private static $count = 0;** to declare the static property.
- Increment the count inside the constructor.
- Use **public static function getCount()** to define the static method.

Exercise 4: Constants

Objective:

Define and use a class constant for book cover type.

Task:

Define a constant **BOOK_COVER_TYPE** within the Book class and output its value.

Hints:

- Use **const HARD_COVER = 1, SOFT_COVER = 2;** inside the class to define the constant.
- Access both constant via **Book::** out side class

Exercise 5: Magic Methods for Properties (__set and __get)

Objective:

Implement magic methods to handle property access and setting.

Task:

Write the **__set** and **__get** magic methods to manage inaccessible properties.

Hints:

- The **__set** method has two parameters: the property name and the value to set.
- The **__get** method has one parameter: the property name.

Exercise 6: Destructor and Cloning

Objective:

Understand the use of destructors and the cloning process.

Task:

Implement the **__destruct** method and demonstrate object cloning.

Hints:

- The **__destruct** method does not take parameters and is used to perform cleanup.
- Use the **clone** keyword to clone an object.

Exercise 7: Implementing Deep Copy through Cloning

Objective:

Understand the difference between shallow and deep copies and how to implement deep copying of object properties in PHP.

Task:

In this exercise, you'll enhance the **Book** class with a **Publisher** object property and implement the **__clone** method to perform a deep copy of the **Book** object, ensuring that the **Publisher** object is also cloned.

Hints:

- Define a new **Publisher** class with some **properties**.
- Add a **Publisher** object as a property in the **Book** class.
- In the **__clone** method, check if the **Publisher** property is an object and clone it.

How to access Cpanel

- https://username.gblearn.com/cpanel
- GBLearn username and password.

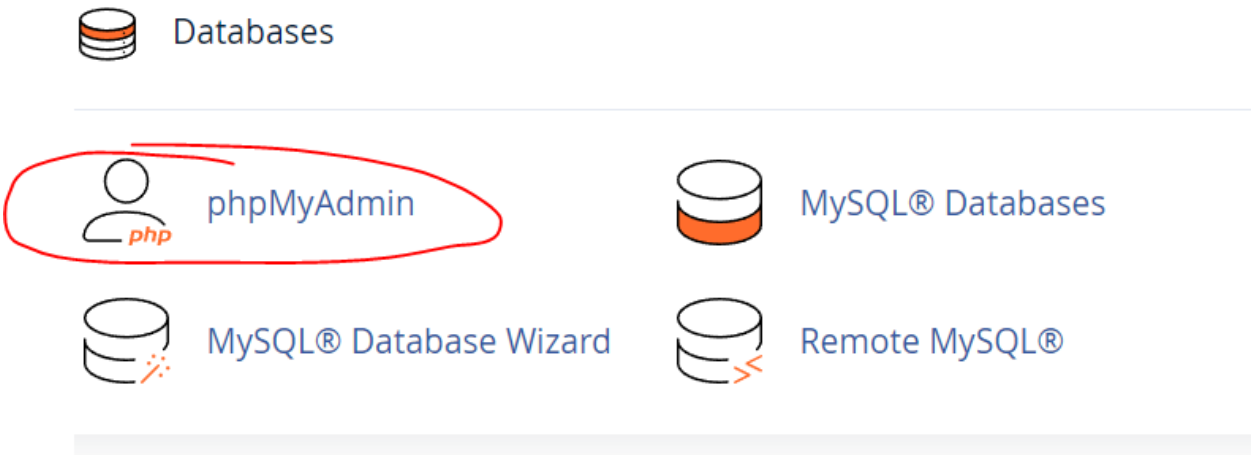
How to access PHPMyAdmin:

Please note you cannot create database via phpMyAdmin. You must create the database on Cpanel / MySQL Databases options.

How to Create MySQL Databases on CPanel

https://youtu.be/nVnGRhW5LY8?si=DoFZN-Uk7F403GtK

1. Login to cpanel
2. Look for database section
3. Click on phpMyAdmin icon



Accessing PHPMyAdmin on XAMPP:

1. **Start XAMPP:**
    - Open the XAMPP Control Panel application on your computer.
    - Start the Apache and MySQL modules by clicking the 'Start' buttons next to each module.
2. **Open PHPMyAdmin:**
    - Open a web browser and navigate to `http://localhost/phpmyadmin`. This should open the PHPMyAdmin interface, which is a web-based tool to manage MySQL databases.
    - If you have changed the default ports during installation, you may need to access it via `http://localhost:PORT/phpmyadmin`, replacing `PORT` with the specific port number you configured.

Creating a Database in PHPMyAdmin:

3. **Navigate to the Databases Tab:**
    - On the PHPMyAdmin home page, click on the 'Databases' tab at the top.
2. **Create Database:**
    - Enter the name of the new database in the 'Create database' input field.
    - Choose the appropriate collation (character set and collation) if necessary. If you're unsure, you can leave it as the default setting.
    - Click the 'Create' button.

Creating a User in PHPMyAdmin:

1. **Navigate to the User Accounts Tab:**
    - Click on the 'User accounts' tab at the top of the PHPMyAdmin interface.
2. **Add User Account:**

- Scroll down to the 'Add user account' section.
- Enter the desired username in the 'User name' field.
- Choose the host where this user will connect from. For local users, you can choose 'localhost'.
- Enter a password for the user and confirm it.
- Optionally, you can generate a password using the 'Generate' button.

Assigning User Privileges:

3. **Assign Global Privileges (Optional):**
    - If you wish to assign global privileges to the user, you can do so in the 'Global privileges' section. Be cautious with global privileges as they apply to all databases.
    - Click 'Check All' to select all privileges, or select individual privileges as needed.
    - Click the 'Go' button at the bottom of the page to apply these privileges.
4. **Assign Database-Specific Privileges:**
    - Scroll down to the 'Database-specific privileges' section.
    - Select the database you created from the 'Add privileges on the following database' dropdown.
    - After selecting the database, a new page will open where you can specify the privileges for this user on the chosen database.
    - Select the privileges you want to assign. For full access, click 'Check All'.
    - Click the 'Go' button at the bottom of the page to apply these privileges.

Finalizing:

- After creating the user and assigning privileges, you can start using the database with the new user account. Make sure to remember or securely store the username and password you've created.

Notes:

- PHPMyAdmin is a powerful tool, and with great power comes great responsibility. Be careful when granting privileges, especially on a production server or where security is a concern.
- The look and options of PHPMyAdmin might slightly differ based on the version of PHPMyAdmin that comes with your XAMPP package.

Remember that these instructions are for managing a local development environment. Managing a live server environment would require additional security measures and is generally not managed through PHPMyAdmin.