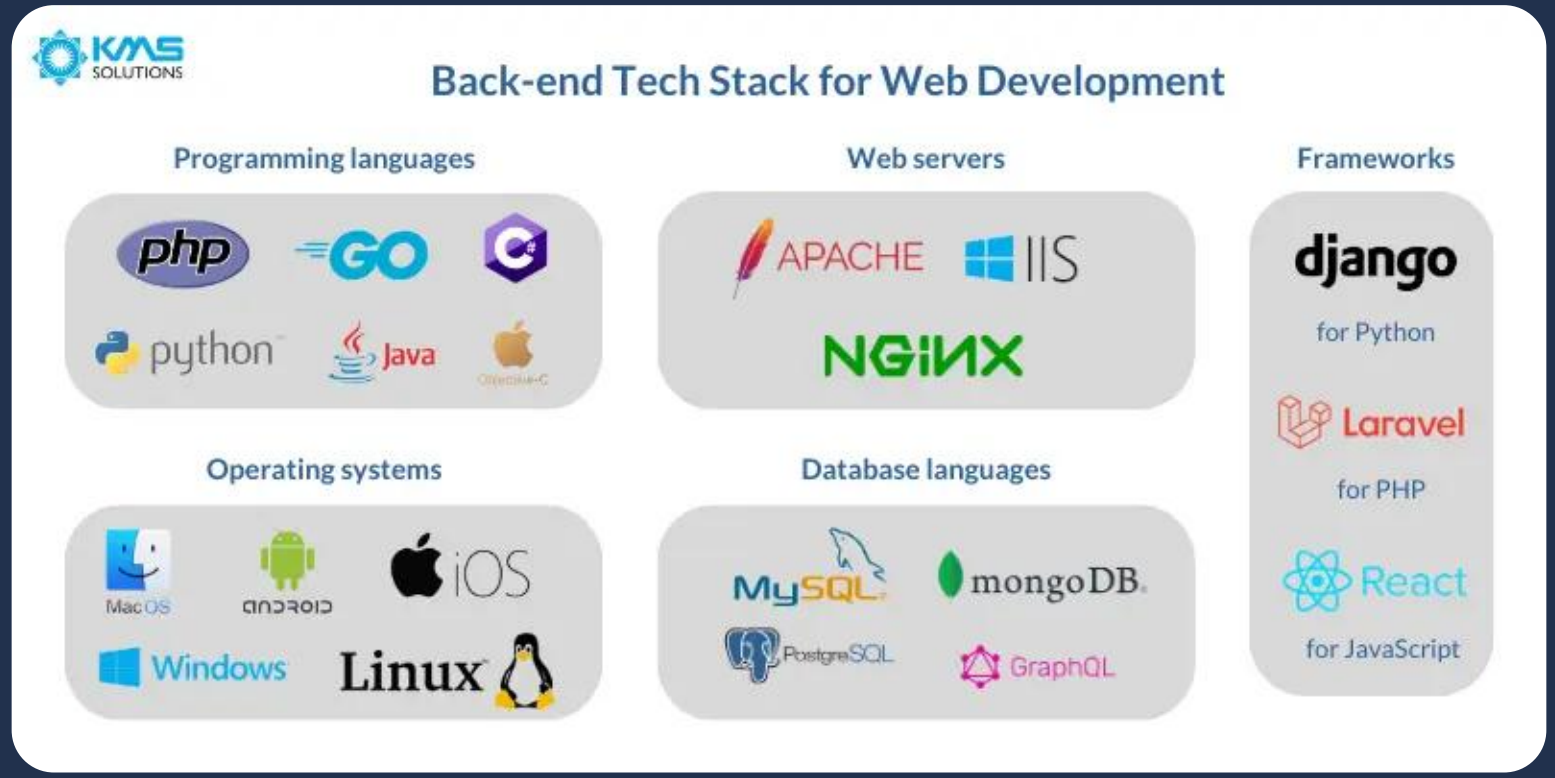
CSE485 – Công nghệ Web

dungkt@tlu.edu.vn

QUI ĐỊNH

- Lớp học KHÔNG DÙNG ĐIỆN THOẠI
- Lớp học KHÔNG MẶC QUẦN ĐÙI đến lớp
- Lớp học CẦN MANG LAPTOP
- Lớp học KHÔNG ĐI MUỘN (*chậm nhất sau 5 phút không vào lớp*)
- Lớp học CẦN LÀM BÀI TẬP ĐẦY ĐỦ, NỘP ĐÚNG HẠN
- Lớp học BỊ PHẠT THEO NHÓM nếu ngồi cùng BÀN với thành viên vi phạm các QUI ĐỊNH
- Máy tính cần dán biển tên để định dang gọi tên SV



Bài 5. PHP OOP (Lập trình hướng đối tượng)

NỘI DUNG

1. Giới thiệu về Class và Object
2. Thuộc tính của lớp
3. Phương thức của lớp
4. Phương thức khởi tạo của lớp
5. Public và private
6. Phương thức getter và setter
7. Các thuộc tính và phương thức static
8. Hằng số trong lớp
9. Kế thừa
10. Override method
11. Protected
12. Ứng dụng PHP OOP và PDO

NỘI DUNG

12. Ứng dụng PHP OOP và PDO
13. Xác thực người dùng
14. Site Admin
15. Phân trang
16. Upload file
17. Mối quan hệ giữa các Bảng
18. Javascript và CSS
19. PHP send email
20. File cấu hình và xử lý lỗi

1. GIỚI THIỆU VỀ CLASS và OBJECT

- **Lập trình hướng đối tượng (OOP)** là một phương pháp lập trình mô phỏng thế giới thực bằng cách tổ chức mã thành các đơn vị gọi là "đối tượng". Mỗi đối tượng kết hợp dữ liệu (thuộc tính) và các hành động (phương thức) hoạt động trên dữ liệu đó.

Đặc điểm	Lập trình hướng đối tượng (OOP)	Lập trình hướng thủ tục
Tổ chức	Mã được tổ chức thành các đối tượng.	Mã được tổ chức thành các hàm và thủ tục.
Cách tiếp cận	Tập trung vào dữ liệu và các hành động liên quan đến dữ liệu.	Tập trung vào các bước thực hiện để giải quyết vấn đề.
Tính đóng gói	Cao	Thấp
Tính kế thừa	Có	Không
Tính đa hình	Có	Không
Khả năng tái sử dụng	Cao	Thấp
Độ phức tạp	Phức tạp hơn ban đầu, nhưng dễ bảo trì và mở rộng về sau	Đơn giản ban đầu, nhưng khó bảo trì và mở rộng về sau

1. GIỚI THIỆU VỀ CLASS và OBJECT

- **Khái niệm về Class**
  - Class là một bản thiết kế, một khuôn mẫu cho các đối tượng. Nó định nghĩa các thuộc tính (đặc điểm) và phương thức (hành vi) mà các đối tượng thuộc class đó sẽ có.
  - Ví dụ: Class "Con người" có thể có các thuộc tính như tên, tuổi, giới tính và các phương thức như ăn(), ngủ(), làm việc().
- **Khái niệm về Object**
  - Object là một thể hiện cụ thể của một class. Mỗi object có các giá trị riêng cho các thuộc tính của class.
  - Ví dụ: "Nguyễn Văn A" là một object của class "Con người" với tên = "Nguyễn Văn A", tuổi = 20, giới tính = "Nam".

```php
<?php
class HinhChuNhat {
  // Thuộc tính
  public $chieuDai;
  public $chieuRong;

  // Phương thức tính diện tích
  public function tinhDienTich() {
    return $this->chieuDai * $this->chieuRong;
  }
}

// Tạo object
$hinhChuNhat = new HinhChuNhat();

// Gán giá trị cho thuộc tính
$hinhChuNhat->chieuDai = 5;
$hinhChuNhat->chieuRong = 3;

// Gọi phương thức
$dienTich = $hinhChuNhat->tinhDienTich();

// Hiển thị kết quả
echo "Diện tích hình chữ nhật là: " . $dienTich;
// Output: 15
?>
```

2. THUỘC TÍNH CỦA LỚP

- **Thuộc tính** là những biến được định nghĩa bên trong một class, dùng để lưu trữ dữ liệu (đặc điểm) của các đối tượng thuộc lớp đó.
- Trong PHP, thuộc tính được khai báo với từ khóa var, public, private hoặc protected theo sau là tên thuộc tính.
  - **public**: Thuộc tính public có thể được truy cập từ bất kỳ đâu, cả bên trong lẫn bên ngoài class.
  - **private**: Thuộc tính private chỉ có thể được truy cập từ bên trong class.
  - **protected**: Thuộc tính protected có thể được truy cập từ bên trong class và từ các class con (kế thừa từ class này).
  - **var**: Từ khóa var tương đương với public, nhưng không được khuyến khích sử dụng trong các phiên bản PHP mới.

```php
<?php
class HinhChuNhat {
  public $chieuDai; // Thuộc tính public
  private $chieuRong; // Thuộc tính private
  protected $mauSac; // Thuộc tính protected
}
?>
```

2. THUỘC TÍNH CỦA LỚP

- **Gán giá trị cho thuộc tính**: Có hai cách phổ biến để gán giá trị cho thuộc tính:
  - **Gán giá trị trực tiếp**: Đối với thuộc tính public, ta có thể gán giá trị trực tiếp từ bên ngoài class.
  - **Gán giá trị trong phương thức**: Đối với thuộc tính private hoặc protected, ta cần gán giá trị thông qua các phương thức public (ví dụ: phương thức khởi tạo hoặc các phương thức setter).

```php
<?php
class HinhChuNhat {
  public $chieuDai;
  private $chieuRong;

  // Phương thức khởi tạo
  public function __construct($chieuDai, $chieuRong) {
    $this->chieuDai = $chieuDai;
    $this->chieuRong = $chieuRong;
  }

  // Phương thức tính diện tích
  public function tinhDienTich() {
    return $this->chieuDai * $this->chieuRong;
  }

  // Getter cho chiều rộng
  public function getChieuRong() {
    return $this->chieuRong;
  }

  // Setter cho chiều rộng
  public function setChieuRong($chieuRong) {
    $this->chieuRong = $chieuRong;
  }
}
```

2. THUỘC TÍNH CỦA LỚP

- **Gán giá trị cho thuộc tính**: Có hai cách phổ biến để gán giá trị cho thuộc tính:
  - **Gán giá trị trực tiếp**: Đối với thuộc tính public, ta có thể gán giá trị trực tiếp từ bên ngoài class.
  - **Gán giá trị trong phương thức**: Đối với thuộc tính private hoặc protected, ta cần gán giá trị thông qua các phương thức public (ví dụ: phương thức khởi tạo hoặc các phương thức setter).

```php
// Tạo object và gán giá trị cho thuộc tính
$hinhChuNhat = new HinhChuNhat(5, 3);

// Truy cập thuộc tính public
echo "Chiều dài: " . $hinhChuNhat->chieuDai; // Output: 5

// Truy cập thuộc tính private thông qua getter
echo "Chiều rộng: " . $hinhChuNhat->getChieuRong(); //
Output: 3

// Thay đổi giá trị thuộc tính private thông qua setter
$hinhChuNhat->setChieuRong(4);

// Gọi phương thức tính diện tích
$dienTich = $hinhChuNhat->tinhDienTich();

// Hiển thị kết quả
echo "Diện tích hình chữ nhật là: " . $dienTich; // Output:
20
?>
```

3. PHƯƠNG THỨC CỦA LỚP

- Tương tự như thuộc tính, phương thức cũng được khai báo với từ khóa public, private hoặc protected theo sau là tên phương thức và cặp dấu ngoặc đơn (). Bên trong dấu ngoặc đơn là danh sách các tham số (nếu có).
  - **public**: Phương thức public có thể được gọi từ bất kỳ đâu, cả bên trong lẫn bên ngoài class.
  - **private**: Phương thức private chỉ có thể được gọi từ bên trong class.
  - **protected**: Phương thức protected có thể được gọi từ bên trong class và từ các class con (kế thừa từ class này).

```php
<?php
class HinhChuNhat {
  // ... (thuộc tính)

  public function tinhDienTich() {
    // ...
  }

  private function kiemTraHopLe() {
    // ...
  }

  protected function tinhChuVi() {
    // ...
  }
}
?>
```

3. PHƯƠNG THỨC CỦA LỚP

- Truy cập thuộc tính trong phương thức: Để truy cập thuộc tính của đối tượng hiện tại bên trong phương thức, ta sử dụng **$this->ten_thuoc_tinh**.

```php
<?php
class HinhChuNhat {
  public $chieuDai;
  private $chieuRong;
  public function tinhDienTich() {
    return $this->chieuDai * $this->chieuRong; // Truy cập thuộc tính
  }
}
?>
```

- Phương thức có thể nhận các tham số đầu vào và trả về một giá trị.

```php
<?php
class HinhChuNhat {
  // ...
  public function thayDoiKichThuoc($chieuDaiMoi, $chieuRongMoi) { // Nhận tham số
    $this->chieuDai = $chieuDaiMoi;
    $this->chieuRong = $chieuRongMoi;
  }
  public function tinhChuVi() {
    $chuVi = 2 * ($this->chieuDai + $this->chieuRong);
    return $chuVi; // Trả về giá trị
  }
}
?>
```

4. PHƯƠNG THỨC KHỞI TẠO CỦA LỚP

- **Phương thức khởi tạo (constructor)** là một phương thức đặc biệt trong class có tên là **__construct()**. Nó được gọi tự động mỗi khi một đối tượng (object) mới được tạo ra từ class đó.
- Mục đích:
  - Khởi tạo giá trị ban đầu cho các thuộc tính của đối tượng. Điều này giúp đảm bảo rằng đối tượng được tạo ra ở trạng thái hợp lệ và sẵn sàng để sử dụng.
  - Kết nối đến cơ sở dữ liệu.
  - Mở file.
  - Khởi tạo các biến phụ thuộc.

```php
<?php
class HinhChuNhat {
  public $chieuDai;
  private $chieuRong;

  // Phương thức khởi tạo
  public function __construct($chieuDai, $chieuRong) {
    $this->chieuDai = $chieuDai;
    $this->chieuRong = $chieuRong;
  }

  // Phương thức tính diện tích
  public function tinhDienTich() {
    return $this->chieuDai * $this->chieuRong;
  }
}

// Tạo object và truyền giá trị cho thuộc tính
$hinhChuNhat = new HinhChuNhat(5, 3);

// Gọi phương thức tính diện tích
$dienTich = $hinhChuNhat->tinhDienTich();

// Hiển thị kết quả
echo "Diện tích hình chữ nhật là: " . $dienTich; // Output:
15
?>
```

5. PUBLIC và PRIVATE

- **public**
  - Thuộc tính và phương thức public có thể được truy cập từ bất kỳ đâu, cả bên trong lẫn bên ngoài class.
  - Khi một thuộc tính hoặc phương thức được khai báo là public, nó có thể được truy cập trực tiếp bằng cách sử dụng toán tử **->**.

```php
<?php
class HinhChuNhat {
  public $chieuDai;

  public function tinhDienTich() {
    // ...
  }
}

$hinhChuNhat = new HinhChuNhat();
$hinhChuNhat->chieuDai = 5; // Truy cập thuộc tính public từ bên ngoài class
$hinhChuNhat->tinhDienTich(); // Gọi phương thức public từ bên ngoài class
?>
```

5. PUBLIC và PRIVATE

- **private**
  - Thuộc tính và phương thức private chỉ có thể được truy cập từ bên trong class.
  - Điều này có nghĩa là không thể truy cập trực tiếp các thuộc tính hoặc phương thức private từ bên ngoài class.

```php
<?php
class HinhChuNhat {
  private $chieuRong;

  private function kiemTraHopLe() {
    // ...
  }

  public function tinhDienTich() {
    // Có thể truy cập $this->chieuRong và $this->kiemTraHopLe()
    // từ bên trong class
  }
}

$hinhChuNhat = new HinhChuNhat();
// $hinhChuNhat->chieuRong = 3; // Lỗi: Không thể truy cập thuộc tính private từ bên ngoài class
// $hinhChuNhat->kiemTraHopLe(); // Lỗi: Không thể gọi phương thức private từ bên ngoài class
?>
```

5. PUBLIC và PRIVATE

```php
$hinhChuNhat = new HinhChuNhat(5, 3);

// Truy cập thuộc tính private thông qua getter
echo "Chiều dài: " . $hinhChuNhat->getChieuDai(); // Output: 5

// Thay đổi giá trị thuộc tính private thông qua setter
$hinhChuNhat->setChieuRong(4);
?>
```

```php
<?php
class HinhChuNhat {
  private $chieuDai;
  private $chieuRong;

  // Phương thức khởi tạo
  public function __construct($chieuDai, $chieuRong) {
    $this->chieuDai = $chieuDai;
    $this->chieuRong = $chieuRong;
  }

  // Phương thức tính diện tích
  public function tinhDienTich() {
    return $this->chieuDai * $this->chieuRong;
  }

  // Getter cho chiều dài
  public function getChieuDai() {
    return $this->chieuDai;
  }

  // Setter cho chiều dài
  public function setChieuDai($chieuDai) {
    $this->chieuDai = $chieuDai;
  }

  // Getter cho chiều rộng
  public function getChieuRong() {
    return $this->chieuRong;
  }

  // Setter cho chiều rộng
  public function setChieuRong($chieuRong) {
    $this->chieuRong = $chieuRong;
  }
}
```

6. PHƯƠNG THỨC GETTER VÀ SETTER

- **Getter** và **setter** là các phương thức public được sử dụng để truy cập và thay đổi giá trị của các thuộc tính private trong một class. Chúng đóng vai trò quan trọng trong việc bảo vệ dữ liệu và kiểm soát truy cập vào các thuộc tính của đối tượng.
- **Getter**
  - Getter là một phương thức public được sử dụng để lấy giá trị của một thuộc tính private.
  - Tên của getter thường bắt đầu bằng get theo sau là tên của thuộc tính (viết hoa chữ cái đầu tiên).
  - Getter không nhận tham số đầu vào và trả về giá trị của thuộc tính.

```php
<?php
public function getChieuDai() {
  return $this->chieuDai;
}
```

- **Setter**
  - Setter là một phương thức public được sử dụng để thiết lập giá trị cho một thuộc tính private.
  - Tên của setter thường bắt đầu bằng set theo sau là tên của thuộc tính (viết hoa chữ cái đầu tiên).
  - Setter nhận một tham số đầu vào là giá trị mới cho thuộc tính và thường không trả về giá trị (hoặc trả về $this để cho phép chaining method).

```php
<?php
public function setChieuDai($chieuDai) {
  $this->chieuDai = $chieuDai;
  // hoặc: return $this;
}
```

6. PHƯƠNG THỨC GETTER VÀ SETTER

- **Kiểm soát truy cập dữ liệu:** Sử dụng getter và setter giúp kiểm soát cách thức truy cập và thay đổi dữ liệu của các thuộc tính private.
  - **Kiểm tra tính hợp lệ của dữ liệu**: Trước khi gán giá trị mới cho thuộc tính, setter có thể kiểm tra xem giá trị đó có hợp lệ hay không.
  - **Thực hiện các hành động bổ sung**: Setter có thể thực hiện các hành động bổ sung khi thay đổi giá trị của thuộc tính, ví dụ như ghi log, cập nhật cơ sở dữ liệu,...
  - **Tính toán giá trị**: Getter có thể tính toán và trả về giá trị dựa trên các thuộc tính khác của đối tượng.

```php
<?php
// Getter cho chiều dài
public function getChieuDai() {
  return $this->chieuDai;
}

// Setter cho chiều dài
public function setChieuDai($chieuDai) {
  if ($chieuDai > 0) {
    $this->chieuDai = $chieuDai;
  } else {
    echo "Chiều dài phải lớn hơn 0.";
  }
}

// Getter cho chiều rộng
public function getChieuRong() {
  return $this->chieuRong;
}

// Setter cho chiều rộng
public function setChieuRong($chieuRong) {
  if ($chieuRong > 0) {
    $this->chieuRong = $chieuRong;
  } else {
    echo "Chiều rộng phải lớn hơn 0.";
  }
}
```

7. CÁC THUỘC TÍNH VÀ PHƯƠNG THỨC STATIC

- **Từ khóa static** được sử dụng để khai báo thuộc tính và phương thức tĩnh. Khác với các thuộc tính và phương thức thông thường (thuộc về đối tượng), thuộc tính và phương thức tĩnh thuộc về class, không thuộc về bất kỳ đối tượng cụ thể nào.
  - **Thuộc tính static**: Là biến thuộc về class, được chia sẻ bởi tất cả các đối tượng của class đó. Thay đổi giá trị của thuộc tính static sẽ ảnh hưởng đến tất cả các đối tượng.
  - **Phương thức static**: Là hàm thuộc về class, có thể được gọi mà không cần tạo đối tượng từ class. Phương thức static chỉ có thể truy cập các thuộc tính static và các phương thức static khác trong cùng class.
- **Truy cập**:

```
TenClass::$ten_thuoc_tinh; // Truy cập thuộc tính static
TenClass::ten_phuong_thuc(); // Gọi phương thức static
```

7. CÁC THUỘC TÍNH VÀ PHƯƠNG THỨC STATIC

```php
<?php
class HinhChuNhat {
  private $chieuDai;
  private $chieuRong;
  public static $soLuongHinhChuNhat = 0; // Thuộc tính static

  // Phương thức khởi tạo
  public function __construct($chieuDai, $chieuRong) {
    $this->chieuDai = $chieuDai;
    $this->chieuRong = $chieuRong;
    self::$soLuongHinhChuNhat++; // Tăng số lượng hình chữ nhật
  }

  // ... (các phương thức khác)

  // Phương thức static để lấy số lượng hình chữ nhật
  public static function getSoLuongHinhChuNhat() {
    return self::$soLuongHinhChuNhat;
  }
}
```

```php
// Tạo các object
$hinhChuNhat1 = new HinhChuNhat(5, 3);
$hinhChuNhat2 = new HinhChuNhat(4, 2);

// Truy cập thuộc tính static
echo "Số lượng hình chữ nhật: " .
HinhChuNhat::$soLuongHinhChuNhat; // Output: 2

// Gọi phương thức static
echo "Số lượng hình chữ nhật: " .
HinhChuNhat::getSoLuongHinhChuNhat(); // Output: 2
?>
```

8. HẰNG SỐ TRONG LỚP

- **Hằng số** là một giá trị cố định được định nghĩa trong class và không thể thay đổi sau khi được khai báo. Trong PHP, hằng số được khai báo bằng từ khóa **const**.
- Khai báo hằng số:
  - Cú pháp: const TEN_HANG_SO = gia_tri;
  - TEN_HANG_SO thường được viết in hoa để phân biệt với biến.gia_tri phải là một giá trị literal (giá trị cụ thể) như số, chuỗi hoặc null.
  - Không thể gán biểu thức hoặc biến cho hằng số.

```php
<?php
class HinhChuNhat {
  private $chieuDai;
  private $chieuRong;

  const PI = 3.14159; // Khai báo hằng số PI

  // Phương thức khởi tạo
  public function __construct($chieuDai, $chieuRong) {
    $this->chieuDai = $chieuDai;
    $this->chieuRong = $chieuRong;
  }

  // Phương thức tính diện tích
  public function tinhDienTich() {
    return $this->chieuDai * $this->chieuRong;
  }

  // Phương thức tính chu vi hình tròn ngoại tiếp
  public function tinhChuViHinhTronNgoaiTiep() {
    $duongKinh = sqrt(pow($this->chieuDai, 2) + pow($this-
>chieuRong, 2));
    return self::PI * $duongKinh; // Sử dụng hằng số PI
  }
}

// Tạo object
$hinhChuNhat = new HinhChuNhat(5, 3);

// Gọi phương thức
$chuVi = $hinhChuNhat->tinhChuViHinhTronNgoaiTiep();

// Hiển thị kết quả
echo "Chu vi hình tròn ngoại tiếp là: " . $chuVi;
?>
```

9. KẾ THỪA

- **Kế thừa** là một trong những tính chất quan trọng nhất của lập trình hướng đối tượng (OOP)
  - Cho phép bạn tạo ra một class mới (class con) dựa trên một class đã có (class cha).
  - Class con sẽ kế thừa tất cả các thuộc tính và phương thức của class cha, đồng thời có thể mở rộng thêm các thuộc tính và phương thức mới hoặc thay đổi hành vi của các phương thức đã kế thừa.
- **Khái niệm**
  - Class cha (superclass/parent class): Là class được kế thừa.
  - Class con (subclass/child class): Là class mới được tạo ra dựa trên class cha.Kế thừa tạo ra một mối quan hệ "is-a" giữa class con và class cha.
  - Ví dụ, nếu class "Hình vuông" kế thừa từ class "Hình chữ nhật", ta có thể nói "Hình vuông là một loại hình chữ nhật".
- **Mục đích**
  - Tái sử dụng mã: Tránh lặp lại mã bằng cách sử dụng lại các thuộc tính và phương thức đã được định nghĩa trong class cha.
  - Mở rộng chức năng: Class con có thể thêm các thuộc tính và phương thức mới để mở rộng chức năng của class cha.
  - Tạo cấu trúc phân cấp: Kế thừa giúp tạo ra một cấu trúc phân cấp giữa các class, phản ánh mối quan hệ giữa các đối tượng trong thế giới thực.

9. KẾ THỪA

- **Từ khóa:**

```php
<?php
class ClassCon extends ClassCha {
  // Khai báo thuộc tính và phương thức của class con
}
?>
```

```php
<?php
class HinhChuNhat {
  protected $chieuDai;
  protected $chieuRong;

  public function __construct($chieuDai, $chieuRong) {
    $this->chieuDai = $chieuDai;
    $this->chieuRong = $chieuRong;
  }

  public function tinhDienTich() {
    return $this->chieuDai * $this->chieuRong;
  }
}
?>
```

```php
<?php
class HinhVuong extends HinhChuNhat {
  public function __construct($canh) {
    parent::__construct($canh, $canh); // Gọi phương thức khởi tạo của class cha
  }
}

// Tạo object hình vuông
$hinhVuong = new HinhVuong(5);

// Gọi phương thức tính diện tích (kế thừa từ class HinhChuNhat)
$dienTich = $hinhVuong->tinhDienTich();

// Hiển thị kết quả
echo "Diện tích hình vuông là: " . $dienTich; // Output: 25
?>
```

9. KẾ THỪA

- **Lưu ý:**
  - Class con chỉ kế thừa các thuộc tính và phương thức public và protected của class cha.
  - Class con có thể ghi đè (override) các phương thức của class cha để thay đổi hành vi.
  - PHP không hỗ trợ đa kế thừa (một class kế thừa từ nhiều class cha).

10. GHI ĐÈ PHƯƠNG THỨC

- **Ghi đè phương thức (method overriding)** là khả năng của class con định nghĩa lại một phương thức đã được kế thừa từ class cha. Điều này cho phép class con cung cấp một phiên bản thực thi khác cho phương thức đó, phù hợp với đặc điểm riêng của mình.
  - Để ghi đè phương thức, class con cần khai báo lại phương thức với cùng tên, cùng số lượng và kiểu dữ liệu của tham số, và cùng kiểu dữ liệu trả về như phương thức trong class cha.
- Lưu ý:
  - Phạm vi truy cập của phương thức trong class con phải bằng hoặc rộng hơn phạm vi truy cập của phương thức tương ứng trong class cha (ví dụ: nếu phương thức trong class cha là protected thì phương thức trong class con có thể là protected hoặc public, nhưng không thể là private).

10. GHI ĐÈ PHƯƠNG THỨC

```php
<?php
class HinhChuNhat {
  protected $chieuDai;
  protected $chieuRong;

  public function __construct($chieuDai, $chieuRong) {
    $this->chieuDai = $chieuDai;
    $this->chieuRong = $chieuRong;
  }

  public function tinhDienTich() {
    return $this->chieuDai * $this->chieuRong; 1
  }
}
?>
```

```php
<?php
class HinhVuong extends HinhChuNhat {
  public function __construct($canh) {
    parent::__construct($canh, $canh);
  }

  // Ghi đè phương thức tinhDienTich()
  public function tinhDienTich() {
    return $this->chieuDai * $this->chieuDai; // Hoặc return
pow($this->chieuDai, 2);
  }
}

// Tạo object hình vuông
$hinhVuong = new HinhVuong(5);

// Gọi phương thức tinhDienTich() (đã được ghi đè)
$dienTich = $hinhVuong->tinhDienTich();

// Hiển thị kết quả
echo "Diện tích hình vuông là: " . $dienTich; // Output: 25
?>
```

11. TỪ KHÓA PROTECTED

- **Từ khóa protected** được sử dụng để khai báo thuộc tính và phương thức với phạm vi truy cập "bảo vệ".
  - **Truy cập từ bên trong class**: Thuộc tính và phương thức protected có thể được truy cập từ bên trong class mà chúng được định nghĩa, giống như private.
  - **Truy cập từ class con**: Điểm khác biệt so với private là thuộc tính và phương thức protected cũng có thể được truy cập từ các class con (class kế thừa từ class hiện tại).
  - **Tính đóng gói**: protected vẫn góp phần vào tính đóng gói của OOP, vì nó hạn chế truy cập trực tiếp từ bên ngoài class, nhưng đồng thời cho phép class con kế thừa và sử dụng các thành phần này.
- **Khi nào nên sử dụng protected?**
  - Khi bạn muốn các thuộc tính hoặc phương thức được kế thừa bởi các class con, nhưng vẫn muốn bảo vệ chúng khỏi truy cập trực tiếp từ bên ngoài.
  - Khi bạn muốn cung cấp một mức độ linh hoạt nhất định cho các class con, cho phép chúng thay đổi hoặc mở rộng hành vi của class cha.

11. TỪ KHÓA PROTECTED

```php
<?php
class HinhChuNhat {
  protected $chieuDai; // Thay đổi từ private thành
protected
  protected $chieuRong; // Thay đổi từ private thành
protected

  public function __construct($chieuDai, $chieuRong) {
    $this->chieuDai = $chieuDai;
    $this->chieuRong = $chieuRong;
  }

  public function tinhDienTich() {
    return $this->chieuDai * $this->chieuRong; 1
  }
}
?>
```

```php
<?php
class HinhVuong extends HinhChuNhat {
  public function __construct($canh) {
    parent::__construct($canh, $canh);
  }

  public function tinhChuVi() {
    return 4 * $this->chieuDai; // Truy cập thuộc tính
protected từ class con
  }
}

$hinhVuong = new HinhVuong(5);
echo "Chu vi hình vuông là: " . $hinhVuong->tinhChuVi(); //
Output: 20
?>
```

12. ỨNG DỤNG PHP OOP VÀ PDO - MYSQL

- **Tạo class kết nối CSDL với PDO**:
  - Tạo một class mới (ví dụ: Database).
  - Khai báo các thuộc tính private để lưu trữ thông tin kết nối (host, database, username, password).
  - Tạo phương thức khởi tạo (__construct()) để nhận thông tin kết nối và tạo đối tượng PDO.
  - Tạo phương thức getConnection() để trả về đối tượng PDO.
  - Xử lý ngoại lệ (exception) trong trường hợp kết nối thất bại.

```php
<?php
class Database {
  private $host = "localhost";
  private $database_name = "mydatabase";
  private $username = "username";
  private $password = "password";
  public $conn;

  public function getConnection(){
    $this->conn = null;
    try{
      $this->conn = new PDO("mysql:host=" . $this->host .
";dbname=" . $this->database_name, $this->username, $this-
>password);
      $this->conn->exec("set names utf8");
    }catch(PDOException $exception){
      echo "Database could not be connected: " . $exception-
>getMessage();
    }
    return $this->conn;
  }
}
?>
```

12. ỨNG DỤNG PHP OOP VÀ PDO - MYSQL

- **Kết nối CSDL sử dụng class Database**:
  - Tạo đối tượng từ class Database.
  - Gọi phương thức getConnection() để lấy đối tượng PDO.
  - Sử dụng đối tượng PDO để thực hiện các truy vấn CSDL.

```php
<?php
require_once 'Database.php'; // Include file Database.php

$database = new Database();
$db = $database->getConnection();

// Sử dụng $db để thực hiện truy vấn
$query = "SELECT * FROM articles";
$stmt = $db->prepare($query);
$stmt->execute();
// ...
?>
```

12. ỨNG DỤNG PHP OOP VÀ PDO - MYSQL

- **Xử lý lỗi kết nối:**
  - Sử dụng khối try...catch để bắt ngoại lệ PDOException.
  - Hiển thị thông báo lỗi thân thiện cho người dùng.
  - Ghi log lỗi để debug.
  - Có thể sử dụng các kỹ thuật khác như chuyển hướng đến trang lỗi, gửi email thông báo,....
- **Câu lệnh prepare SQL**
  - Chuẩn bị câu truy vấn SQL với các placeholder.
  - Gán giá trị cho các placeholder.
  - Thực thi câu truy vấn.

```php
<?php
// ... (Kết nối CSDL)

$query = "SELECT * FROM articles WHERE id = :id";
$stmt = $db->prepare($query);
$stmt->bindParam(':id', $articleId);
$stmt->execute();
// ...
?>
```

12. ỨNG DỤNG PHP OOP VÀ PDO - MYSQL

- **Tạo class Article**:
  - Khai báo các thuộc tính của class tương ứng với các cột trong bảng articles.
  - Tạo các phương thức để thực hiện các thao tác CRUD (Create, Read, Update, Delete) với bảng articles:
    - create(): Thêm bài viết mới.
    - read(): Lấy danh sách bài viết.
    - getById(): Lấy bài viết theo ID.
    - update(): Cập nhật bài viết.
    - delete(): Xóa bài viết.

```php
<?php
class Article {
  // ... (Khai báo thuộc tính)

  public function create(){
    // ... (Thêm bài viết mới)
  }

  public function read(){
    // ... (Lấy danh sách bài viết)
  }

  public function getById($id){
    // ... (Lấy bài viết theo ID)
  }

  public function update(){
    // ... (Cập nhật bài viết)
  }

  public function delete(){
    // ... (Xóa bài viết)
  }
}
?>
```

12. ỨNG DỤNG PHP OOP VÀ PDO - MYSQL

- **Thay đổi kiểu trả về của method getById:**
  - Mục tiêu: Hướng dẫn thay đổi kiểu dữ liệu trả về của phương thức getById() trong class Article.
  - Nội dung:
    - Thay vì trả về một mảng, có thể trả về một đối tượng Article.
    - Lợi ích: Tăng tính nhất quán và dễ sử dụng.

```php
<?php
// ... (Trong class Article)

public function getById($id){
  // ... (Truy vấn CSDL)

  if ($row) {
    $article = new Article();
    $article->id = $row['id'];
    $article->title = $row['title'];
    // ...
    return $article; // Trả về đối tượng Article
  } else {
    return null;
  }
}
?>
```

12. ỨNG DỤNG PHP OOP VÀ PDO - MYSQL

- **Sửa bài viết bằng PDO**:
  - Mục tiêu: Hướng dẫn sử dụng PDO để cập nhật dữ liệu bài viết.
  - Nội dung:
    - Sử dụng câu lệnh UPDATE trong SQL.
    - Sử dụng prepared statements để ngăn chặn SQL injection.
    - Xử lý dữ liệu đầu vào từ form.
    - Cập nhật dữ liệu trong CSDL.

```php
<?php
// ... (Trong class Article)

public function update(){
  $query = "UPDATE articles SET title = :title, content = :content
WHERE id = :id";
  $stmt = $this->conn->prepare($query);

  // Gán giá trị cho các placeholder
  $stmt->bindParam(':title', $this->title);
  $stmt->bindParam(':content', $this->content);
  $stmt->bindParam(':id', $this->id);

  // Thực thi câu truy vấn
  if($stmt->execute()){
    return true;
  }
  return false;
}
?>
```

12. ỨNG DỤNG PHP OOP VÀ PDO - MYSQL

- **Đưa validate vào Article class:**
  - Mục tiêu: Hướng dẫn thêm các quy tắc validate dữ liệu vào class Article.
  - Nội dung:
    - Kiểm tra tính hợp lệ của dữ liệu trước khi thêm hoặc cập nhật bài viết.
    - Ví dụ: Kiểm tra độ dài tiêu đề, nội dung, định dạng dữ liệu,...
    - Hiển thị thông báo lỗi nếu dữ liệu không hợp lệ.

```php
<?php
// ... (Trong class Article)

public function create(){
  // Validate dữ liệu
  if (empty($this->title) || strlen($this->title) > 255) {
    // Hiển thị thông báo lỗi
    return false;
  }
  // ... (Thêm bài viết)
}
?>
```

12. ỨNG DỤNG PHP OOP VÀ PDO - MYSQL

- **Xóa bài viết bằng PDO**:
  - Mục tiêu: Hướng dẫn sử dụng PDO để xóa dữ liệu bài viết.
  - Nội dung:
    - Sử dụng câu lệnh DELETE trong SQL.
    - Sử dụng prepared statements để ngăn chặn SQL injection.
    - Xác nhận xóa trước khi thực hiện.

```php
<?php
// ... (Trong class Article)

public function delete(){
  $query = "DELETE FROM articles WHERE id = :id";
  $stmt = $this->conn->prepare($query);

  // Gán giá trị cho placeholder
  $stmt->bindParam(':id', $this->id);

  // Thực thi câu truy vấn
  if($stmt->execute()){
    return true;
  }
  return false;
}
?>
```

12. ỨNG DỤNG PHP OOP VÀ PDO - MYSQL

- **Thêm bài viết bằng PDO**:
  - Mục tiêu: Hướng dẫn sử dụng PDO để thêm dữ liệu bài viết mới.
  - Nội dung:
    - Sử dụng câu lệnh INSERT trong SQL.
    - Sử dụng prepared statements để ngăn chặn SQL injection.
    - Xử lý dữ liệu đầu vào từ form.
    - Thêm dữ liệu vào CSDL.

```php
<?php
// ... (Trong class Article)

public function create(){
  // ... (Validate dữ liệu)

  $query = "INSERT INTO articles SET title = :title, content
= :content";
  $stmt = $this->conn->prepare($query);

  // Gán giá trị cho các placeholder
  $stmt->bindParam(':title', $this->title);
  $stmt->bindParam(':content', $this->content);

  // Thực thi câu truy vấn
  if($stmt->execute()){
    return true;
  }
  return false;
}
?>
```

13. XÁC THỰC NGƯỜI DÙNG

- **Tạo class và Table User**:
  - Thiết kế bảng users với các trường cần thiết (id, username, password, email,...).
  - Tạo class User với các thuộc tính tương ứng với các cột trong bảng users.
  - Tạo các phương thức trong class User để thực hiện các thao tác với bảng users, ví dụ:
    - getByUsername(): Lấy thông tin người dùng theo username.
    - create(): Tạo tài khoản người dùng mới.
    - update(): Cập nhật thông tin người dùng.

```sql
// Tạo bảng users
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(255)
);
```

```php
<?php
// Tạo class User
class User {
    private $conn;

    public function __construct($db){
        $this->conn = $db;
    }

    public function getByUsername($username){
        $query = "SELECT * FROM users WHERE username =
:username";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(':username', $username);
        $stmt->execute();
        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    // ... (Các phương thức khác)
}
```

13. XÁC THỰC NGƯỜI DÙNG

- **Xác thực người dùng từ thông tin trong CSDL:**
  - Lấy thông tin đăng nhập từ form (username, password).
  - Sử dụng class User để lấy thông tin người dùng từ CSDL dựa trên username.
  - So sánh mật khẩu nhập vào với mật khẩu đã được mã hóa trong CSDL (sử dụng password_verify()).
  - Nếu xác thực thành công, lưu thông tin người dùng vào session.

```php
<?php
require_once 'User.php';

// ... (Kết nối CSDL)

$user = new User($db);

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $username = $_POST['username'];
    $password = $_POST['password'];

    $userData = $user->getByUsername($username);

    if ($userData && password_verify($password,
$userData['password'])) {
        // Xác thực thành công
        $_SESSION['user_id'] = $userData['id'];
        // ... (Lưu thêm thông tin)
    } else {
        // Xác thực thất bại
    }
}
?>
```

13. XÁC THỰC NGƯỜI DÙNG

- **Hash Password:**
  - Giải thích về tầm quan trọng của việc mã hóa mật khẩu.
  - Cú pháp và cách sử dụng hàm password_hash().
  - Các thuật toán mã hóa được hỗ trợ.

```php
<?php
$password = "mysecretpassword";
$hashedPassword = password_hash($password,
PASSWORD_DEFAULT);
echo $hashedPassword; // Output:
$2y$10$... (chuỗi hash)
```

- **Lưu Hash password vào CSDL**
  - Sử dụng phương thức create() trong class User để thêm người dùng mới.
  - Mã hóa mật khẩu trước khi lưu vào CSDL.

```php
<?php
// ... (Trong class User)

public function create(){
    // ... (Validate dữ liệu)

    $this->password = password_hash($this->password,
PASSWORD_DEFAULT);

    $query = "INSERT INTO users SET username = :username,
password = :password, email = :email";
    // ... (Thực thi câu truy vấn)
}
?>
```

13. XÁC THỰC NGƯỜI DÙNG

- **Chuyển auth include thành dạng class:**
  - Tạo một class mới (ví dụ: Auth) để quản lý các chức năng xác thực.
  - Chuyển các hàm xác thực từ file include vào class Auth.
  - Tạo các phương thức trong class Auth để thực hiện các chức năng như:
    - login(): Xác thực người dùng và lưu session.
    - logout(): Xóa session.
    - check(): Kiểm tra xem người dùng đã đăng nhập hay chưa.

```php
<?php
class Auth {
    private $conn;

    public function __construct($db){
        $this->conn = $db;
    }

    public function login($username, $password){
        // ... (Xác thực người dùng)
    }

    public function logout(){
        // ... (Xóa session)
    }

    public function check(){
        // ... (Kiểm tra session)
    }
}
?>
```

13. XÁC THỰC NGƯỜI DÙNG

- **Class Autoload**:
  - Autoloading cho phép tự động include các file class khi cần thiết, mà không cần phải sử dụng require hoặc include thủ công.
  - Cú pháp và cách sử dụng hàm spl_autoload_register().
  - Tạo hàm autoload để tự động include các file class.

```php
<?php
spl_autoload_register(function ($class_name) {
    include $class_name . '.php';
});

$user = new User($db); // Không cần require 'User.php'
?>
```

13. XÁC THỰC NGƯỜI DÙNG

- **Return value từ include file:**
  - Sử dụng return trong file include để trả về giá trị.
  - Gán giá trị trả về cho một biến khi include file.

```php
<?php
// File included.php
return "Hello world!";

// File main.php
$message = include 'included.php';
echo $message; // Output: Hello world!
?>
```

13. XÁC THỰC NGƯỜI DÙNG

- **Chuyển các code liên quan authen vào class Auth:**
  - Di chuyển tất cả các code liên quan đến xác thực (đăng nhập, đăng xuất, kiểm tra session,...) vào class Auth.
  - Tạo các phương thức public để truy cập các chức năng xác thực từ bên ngoài.

```php
<?php
class Auth {
    // ... (Các thuộc tính và phương thức)

    public function isLoggedIn(){
        // ... (Kiểm tra session)
    }

    public function getCurrentUser(){
        // ... (Lấy thông tin người dùng từ session)
    }

    // ... (Các phương thức khác)
}
?>
```

14. SITE ADMIN

- **Thêm page index admin:**
  - Tạo file index.php trong thư mục admin.
  - Kiểm tra xem người dùng đã đăng nhập và có quyền admin hay chưa (sử dụng class Auth).
  - Nếu chưa đăng nhập hoặc không có quyền admin, chuyển hướng đến trang đăng nhập.
  - Hiển thị giao diện trang quản trị, bao gồm các liên kết đến các chức năng quản lý (bài viết, người dùng, ...).

```php
<?php
// admin/index.php

require_once '../Auth.php';

$auth = new Auth($db); // Giả sử $db là đối tượng PDO

if (!$auth->isLoggedIn() || !$auth->getCurrentUser()['role']
=== 'admin') {
    header('Location: ../login.php');
    exit;
}

?>

<!DOCTYPE html>
<html>
<head>
  <title>Trang quản trị</title>
</head>
<body>
  <h1>Xin chào, <?php echo $auth-
>getCurrentUser()['username']; ?></h1>
  <ul>
    <li><a href="articles.php">Quản lý bài viết</a></li>
    <li><a href="users.php">Quản lý người dùng</a></li>
    </ul>
</body>
</html>
```

14. SITE ADMIN

- **Hiển thị các bài viết dưới dạng Table:**
  - Sử dụng class Article để lấy danh sách bài viết từ CSDL.
  - Sử dụng vòng lặp để duyệt qua danh sách bài viết và hiển thị trong bảng HTML.
  - Thêm các cột cho các thông tin cần thiết (tiêu đề, ngày tạo, tác giả, ...).
  - Thêm các liên kết để chỉnh sửa và xóa bài viết.

```php
<?php
// admin/articles.php

// ... (Kết nối CSDL và kiểm tra quyền admin)

$article = new Article($db);
$articles = $article->read();
?>

<table>
  <thead>
    <tr>
      <th>Tiêu đề</th>
      <th>Ngày tạo</th>
      <th>Tác giả</th>
      <th>Hành động</th>
    </tr>
  </thead>
  <tbody>
    <?php foreach ($articles as $article): ?>
      <tr>
        <td><?php echo $article['title']; ?></td>
        <td><?php echo $article['created_at']; ?></td>
        <td><?php echo $article['author']; ?></td>
        <td>
          <a href="edit_article.php?id=<?php echo
$article['id']; ?>">Sửa</a> |
          <a href="delete_article.php?id=<?php echo
$article['id']; ?>">Xóa</a>
        </td>
      </tr>
    <?php endforeach; ?>
  </tbody>
</table>
```

14. SITE ADMIN

- **Thêm các đường link giữa các trang:**
  - Thêm liên kết đến trang chủ quản trị (admin/index.php).
  - Thêm liên kết đến các trang chức năng khác (quản lý bài viết, người dùng, ...).
  - Có thể sử dụng menu hoặc breadcrumb để điều hướng.

```
<?php
<a href="index.php">Trang chủ</a> |
<a href="articles.php">Quản lý bài viết</a> |
<a href="users.php">Quản lý người dùng</a>
```

14. SITE ADMIN

- **Chuyển edit và delete vào admin:**
  - Tạo các trang edit_article.php và delete_article.php trong thư mục admin.
  - Kiểm tra quyền admin trước khi cho phép chỉnh sửa hoặc xóa bài viết.
  - Sử dụng class Article để thực hiện các thao tác cập nhật và xóa dữ liệu.

```php
<?php
// admin/edit_article.php

// ... (Kết nối CSDL và kiểm tra quyền admin)

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $article = new Article($db);
  $article->id = $_POST['id'];
  $article->title = $_POST['title'];
  // ...
  if ($article->update()) {
      // Chuyển hướng đến trang danh sách bài viết
  }
}
?>
```

14. SITE ADMIN

- **Chuyển new article vào admin:**
  - Tạo trang new_article.php trong thư mục admin.
  - Kiểm tra quyền admin trước khi cho phép thêm bài viết mới.
  - Sử dụng class Article để thêm dữ liệu vào CSDL.

```php
<?php
// admin/new_article.php

// ... (Kết nối CSDL và kiểm tra quyền admin)

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $article = new Article($db);
  $article->title = $_POST['title'];
  $article->content = $_POST['content'];
  // ...
  if ($article->create()) {
      // Chuyển hướng đến trang danh sách bài viết
  }
}
?>
```

15. PHÂN TRANG

- **Câu lệnh SQL dùng để phân trang:**
  - Phân trang là kỹ thuật chia dữ liệu thành nhiều trang nhỏ để hiển thị, giúp cải thiện hiệu suất và trải nghiệm người dùng.
  - Cú pháp câu lệnh LIMIT và OFFSET trong MySQL:
    - LIMIT: Giới hạn số lượng bản ghi trả về.
    - OFFSET: Bỏ qua một số lượng bản ghi nhất định.

```sql
SELECT * FROM articles LIMIT 10 OFFSET 10;
```

15. PHÂN TRANG

- **Phương thức getPage:**
  - Phương thức getPage() nhận hai tham số:
    - page: Số trang.
    - limit: Số lượng bản ghi trên mỗi trang.
  - Tính toán offset dựa trên page và limit.
  - Sử dụng câu lệnh LIMIT và OFFSET để lấy dữ liệu từ CSDL.

```php
<?php
// ... (Trong class Article)

public function getPage($page, $limit) {
    $offset = ($page - 1) * $limit;
    $query = "SELECT * FROM articles LIMIT :limit OFFSET
:offset";
    $stmt = $this->conn->prepare($query);
    $stmt->bindParam(':limit', $limit, PDO::PARAM_INT);
    $stmt->bindParam(':offset', $offset, PDO::PARAM_INT);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
?>
```

15. PHÂN TRANG

- **Tính limit và offset từ số trang:**
  - limit: Là số lượng bản ghi hiển thị trên mỗi trang (thường là một giá trị cố định).
  - offset: Là số lượng bản ghi cần bỏ qua trước khi lấy dữ liệu cho trang hiện tại.
  - Công thức tính offset: offset = (page - 1) * limit
- **Lấy số trang từ query string**
  - Query string là phần sau dấu ? trong URL, chứa các cặp key-value.
  - Sử dụng biến toàn cục $_GET để lấy giá trị từ query string.
  - Ví dụ: index.php?page=2 => $_GET['page'] = 2

```php
$page = isset($_GET['page']) ? $_GET['page'] : 1;
```

- Validate page number
  - Kiểm tra xem số trang có phải là số nguyên dương hay không.
  - Nếu không hợp lệ, có thể gán giá trị mặc định là 1.

```php
<?php
$page = isset($_GET['page']) ? (int)$_GET['page'] : 1;
if ($page <= 0) {
    $page = 1;
}
```

15. PHÂN TRANG

- **Thêm link Previos và Next:**
  - Tạo các liên kết HTML với href trỏ đến trang trước và trang sau.
  - Tính toán số trang trước và trang sau dựa trên số trang hiện tại.
  - Vô hiệu hóa liên kết "Previous" nếu đang ở trang đầu tiên, và vô hiệu hóa liên kết "Next" nếu đang ở trang cuối cùng.

```php
<?php
 if ($page > 1): ?>
  <a href="?page=<?php echo $page - 1; ?>">Previous</a>
<?php endif; ?>

<?php if ($page < $totalPages): ?>
  <a href="?page=<?php echo $page + 1; ?>">Next</a>
<?php endif; ?>
```

15. PHÂN TRANG

- **Lấy số bài viết từ CSDL**:
  - Sử dụng câu lệnh SELECT COUNT(*) để đếm số lượng bản ghi trong bảng articles.
  - Tính toán số trang bằng cách chia tổng số bài viết cho số lượng bản ghi trên mỗi trang (limit).

```php
<?php
$query = "SELECT COUNT(*) FROM articles";
$stmt = $this->conn->prepare($query);
$stmt->execute();
$totalArticles = $stmt->fetchColumn();
$totalPages = ceil($totalArticles / $limit);
```

15. PHÂN TRANG

- **Thêm tính năng phân trang cho Admin:**
  - Sử dụng phương thức getPage() trong class Article để lấy dữ liệu cho trang hiện tại.
  - Hiển thị danh sách bài viết dưới dạng bảng HTML.
  - Thêm các liên kết "Previous" và "Next" để điều hướng giữa các trang.
  - Hiển thị thông tin phân trang (ví dụ: "Trang 1 trên 5").

```php
<?php
// admin/articles.php

// ... (Kết nối CSDL và kiểm tra quyền admin)

$limit = 10;
$page = isset($_GET['page']) ? (int)$_GET['page'] : 1;
if ($page <= 0) {
    $page = 1;
}

$article = new Article($db);
$articles = $article->getPage($page, $limit);

// ... (Hiển thị danh sách bài viết)

// Lấy tổng số bài viết và tính toán số trang
$totalArticles = $article->getTotalArticles();
$totalPages = ceil($totalArticles / $limit);

// Hiển thị phân trang
echo "Trang $page trên $totalPages";

// ... (Thêm liên kết Previous và Next)
```

16. UPLOAD FILE

- **Tạo form upload file:**
  - Sử dụng thẻ <form> với enctype="multipart/form-data" và method="POST".
  - Sử dụng thẻ <input type="file" name="fileToUpload"> để tạo trường chọn file.
  - Thêm các trường khác nếu cần (ví dụ: tiêu đề, mô tả).
  - Thêm nút submit để gửi form.

```
<form action="upload.php" method="post"
enctype="multipart/form-data">
  Chọn file để upload:
  <input type="file" name="fileToUpload" id="fileToUpload">
  <input type="submit" value="Upload Image" name="submit">
</form>
```

16. UPLOAD FILE

- **Xử lý lỗi upload file:**
  - Sử dụng biến $_FILES['fileToUpload']['error'] để kiểm tra mã lỗi.
  - Hiển thị thông báo lỗi tương ứng với từng mã lỗi.
  - Tham khảo danh sách mã lỗi upload trong tài liệu PHP.

```php
<?php
if ($_FILES["fileToUpload"]["error"] > 0) {
  echo "Error: " . $_FILES["fileToUpload"]["error"] .
"<br>";
}
```

16. UPLOAD FILE

- **Giới hạn kích thước upload file trên server:**
  - Sửa đổi file php.ini:
    - upload_max_filesize: Giới hạn kích thước tối đa của một file
    - upload.post_max_size: Giới hạn kích thước tối đa của toàn bộ dữ liệu gửi lên trong một request POST.
  - Khởi động lại web server sau khi thay đổi php.ini
- **Giới hạn kích thước upload file trong form**
  - Sử dụng thuộc tính MAX_FILE_SIZE trong thẻ <input type="file">.
  - Lưu ý: Đây chỉ là giới hạn client-side, vẫn cần kiểm tra kích thước file trên server.

```
<input type="hidden" name="MAX_FILE_SIZE" value="3000000">
<input type="file" name="fileToUpload" id="fileToUpload">
```

16. UPLOAD FILE

- **Giới hạn loại tập tin khi upload**:
  - Kiểm tra phần mở rộng của file ($_FILES['fileToUpload']['type’]).
  - Cho phép upload chỉ những loại file mong muốn (ví dụ: ảnh, văn bản).
  - Có thể sử dụng MIME type để kiểm tra loại file chính xác hơn.

```php
<?php
$allowed = array("jpg" => "image/jpeg", "png" => "image/png", "gif" => "image/gif");
$filename = $_FILES["fileToUpload"]["name"];
$filetype = $_FILES["fileToUpload"]["type"];
$ext = pathinfo($filename, PATHINFO_EXTENSION);
if(!array_key_exists($ext, $allowed)) die("Error: Please select a valid file
format.");
```

16. UPLOAD FILE

- **Chuyển file vào thư mục upload:**
  - Sử dụng hàm move_uploaded_file() để di chuyển file.
  - Xác định thư mục đích để lưu trữ file.
  - Kiểm tra xem việc di chuyển file có thành công hay không.

```php
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
  echo "The file ". htmlspecialchars( basename(
$_FILES["fileToUpload"]["name"])). " has been uploaded.";
} else {
  echo "Sorry, there was an error uploading your file.";
}
```

16. UPLOAD FILE

- **Không ghi đè trong trường hợp trùng tên:**
  - Kiểm tra xem file đã tồn tại trong thư mục đích hay chưa.
  - Nếu tồn tại, đổi tên file upload hoặc hiển thị thông báo lỗi.
  - Có thể sử dụng kỹ thuật tạo tên file duy nhất (ví dụ: thêm timestamp).

```php
<?php
$target_file = $target_dir .
basename($_FILES["fileToUpload"]["name"]);
$i = 1;
while (file_exists($target_file)) {
  $target_file = $target_dir . "(" . $i . ")" .
basename($_FILES["fileToUpload"]["name"]);
  $i++;
}
```

16. UPLOAD FILE

- **Thêm tên file upload vào CSDL**:
  - Thêm một cột vào bảng trong CSDL để lưu trữ tên file.
  - Sau khi upload file thành công, sử dụng câu lệnh INSERT để thêm tên file vào CSDL.

```php
<?php
// ... (Kết nối CSDL)

$filename = basename($_FILES["fileToUpload"]["name"]);

$query = "INSERT INTO articles (title, content, filename)
VALUES (:title, :content, :filename)";
$stmt = $this->conn->prepare($query);
$stmt->bindParam(':title', $title);
$stmt->bindParam(':content', $content);
$stmt->bindParam(':filename', $filename);
$stmt->execute();
```

16. UPLOAD FILE

- **Xóa file khi không cần thiết**:
  - Sử dụng hàm unlink() để xóa file.
  - Xóa file khi xóa bản ghi tương ứng trong CSDL.
  - Có thể sử dụng cron job để tự động xóa các file cũ.

```php
<?php
$filename = $_POST['filename'];
$filepath = "uploads/" . $filename;
if (file_exists($filepath)) {
  unlink($filepath);
}

// ... (Xóa bản ghi trong CSDL)
```

- **Lưu ý chung**:
  - Cần kiểm tra kỹ mã nguồn và validate dữ liệu đầu vào để tránh lỗi và lỗ hổng bảo mật.
  - Nên sử dụng thư viện xử lý ảnh để thay đổi kích thước, định dạng ảnh trước khi lưu trữ.
  - Có thể sử dụng dịch vụ lưu trữ đám mây (ví dụ: Amazon S3) để lưu trữ file.

17. MỐI QUAN HỆ GIỮA CÁC BẢNG

- **Mối quan hệ 1-1 giữa hai bảng**
  - Khái niệm: Mỗi bản ghi trong bảng này chỉ liên kết với duy nhất một bản ghi trong bảng kia và ngược lại.
  - Ví dụ: Mối quan hệ giữa bảng users và bảng profiles, mỗi người dùng chỉ có một hồ sơ và mỗi hồ sơ chỉ thuộc về một người dùng.
  - Triển khai: Thường sử dụng khóa ngoại (foreign key) để liên kết hai bảng. Có thể đặt ràng buộc UNIQUE cho khóa ngoại để đảm bảo tính duy nhất.

```sql
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) UNIQUE NOT NULL
);

CREATE TABLE profiles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT UNIQUE,
    bio TEXT,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

17. MỐI QUAN HỆ GIỮA CÁC BẢNG

- **Mối quan hệ 1-nhiều giữa hai bảng**
  - Khái niệm: Một bản ghi trong bảng này có thể liên kết với nhiều bản ghi trong bảng kia, nhưng một bản ghi trong bảng kia chỉ liên kết với duy nhất một bản ghi trong bảng này.
  - Ví dụ: Mối quan hệ giữa bảng categories và bảng articles, một danh mục có thể chứa nhiều bài viết, nhưng một bài viết chỉ thuộc về một danh mục.
  - Triển khai: Sử dụng khóa ngoại trong bảng "nhiều" để liên kết với khóa chính trong bảng "một".

```sql
CREATE TABLE categories (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);

CREATE TABLE articles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES categories(id)
);
```

17. MỐI QUAN HỆ GIỮA CÁC BẢNG

- **Mối quan hệ nhiều-nhiều giữa hai bảng**
  - **Khái niệm**: Nhiều bản ghi trong bảng này có thể liên kết với nhiều bản ghi trong bảng kia.
  - **Ví dụ**: Mối quan hệ giữa bảng articles và bảng tags, một bài viết có thể có nhiều thẻ tag, và một thẻ tag có thể được gắn vào nhiều bài viết.
  - **Triển khai**: Sử dụng một bảng trung gian (bảng liên kết) để liên kết hai bảng. Bảng trung gian chứa khóa ngoại của cả hai bảng.

```sql
CREATE TABLE articles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL
);

CREATE TABLE tags (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);

CREATE TABLE article_tag (
    article_id INT,
    tag_id INT,
    PRIMARY KEY (article_id, tag_id),
    FOREIGN KEY (article_id) REFERENCES articles(id),
    FOREIGN KEY (tag_id) REFERENCES tags(id)
);
```

17. MỐI QUAN HỆ GIỮA CÁC BẢNG

- **Hiển thị category lên bài viết**
  - Khi lấy dữ liệu bài viết, sử dụng câu lệnh JOIN để lấy thêm thông tin danh mục từ bảng categories.
  - Hiển thị tên danh mục trên trang chi tiết bài viết.

```php
<?php
$query = "SELECT a.*, c.name AS category_name
FROM articles a
JOIN categories c ON a.category_id = c.id
WHERE a.id = :id";
// ... (Thực thi truy vấn và hiển thị dữ liệu)
echo "Danh mục: " . $article['category_name'];
```

17. MỐI QUAN HỆ GIỮA CÁC BẢNG

- **Tạo phương thức lấy danh sách category**
  - Thực hiện truy vấn SELECT để lấy tất cả bản ghi từ bảng categories.
  - Trả về danh sách danh mục dưới dạng mảng.

```php
<?php
// Trong class Category
public function getCategories() {
    $query = "SELECT * FROM categories";
    $stmt = $this->conn->prepare($query);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
?>
```

17. MỐI QUAN HỆ GIỮA CÁC BẢNG

- **Tạo form để edit category của bài viết**
  - Tạo form HTML với các checkbox hoặc dropdown list để chọn danh mục.
  - Lấy danh sách danh mục từ CSDL (sử dụng phương thức getCategories()).
  - Hiển thị các danh mục hiện tại của bài viết dưới dạng checked hoặc selected.

```php
<?php
// ... (Lấy danh sách danh mục)
$categories = $category->getCategories();

// ... (Lấy danh sách danh mục của bài viết hiện tại)
$articleCategories = $article->getCategories($articleId);

?>

<form action="update_article.php" method="post">
  <label for="categories">Danh mục:</label><br>
  <?php foreach ($categories as $category): ?>
    <input type="checkbox" name="categories[]" value="<?php echo $category['id']; ?>"
           <?php if (in_array($category['id'], $articleCategories)): ?> checked <?php endif; ?>>
    <?php echo $category['name']; ?><br>
  <?php endforeach; ?>
  <input type="submit" value="Cập nhật">
</form>
```

17. MỐI QUAN HỆ GIỮA CÁC BẢNG

- **Thêm các category vào CSDL**
  - Lấy danh sách danh mục đã chọn từ form.
  - Sử dụng câu lệnh INSERT để thêm dữ liệu vào bảng liên kết article_tag.

```php
<?php
// ... (Lấy danh sách danh mục đã chọn)
$selectedCategories = $_POST['categories'];

// ... (Xóa các danh mục cũ của bài viết)

foreach ($selectedCategories as $categoryId) {
    $query = "INSERT INTO article_tag (article_id,
category_id) VALUES (:article_id, :category_id)";
    $stmt = $this->conn->prepare($query);
    $stmt->bindParam(':article_id', $articleId);
    $stmt->bindParam(':category_id', $categoryId);
    $stmt->execute();
}
```

17. MỐI QUAN HỆ GIỮA CÁC BẢNG

- **Thực thi truy vấn INSERT một lần duy nhất**
  - Tạo một câu lệnh INSERT với nhiều values.
  - Sử dụng prepared statements với nhiều placeholder.

```php
<?php
$query = "INSERT INTO article_tag (article_id, category_id)
VALUES ";
$values = [];
$placeholders = [];
foreach ($selectedCategories as $i => $categoryId) {
    $placeholders[] = "(:article_id$i, :category_id$i)";
    $values[":article_id$i"] = $articleId;
    $values[":category_id$i"] = $categoryId;
}
$query .= implode(", ", $placeholders);
$stmt = $this->conn->prepare($query);
$stmt->execute($values);
```

17. MỐI QUAN HỆ GIỮA CÁC BẢNG

- **Xóa các category bị uncheck**
  - Lấy danh sách danh mục hiện tại của bài viết từ CSDL.
  - So sánh với danh sách danh mục đã chọn từ form.
  - Xóa các danh mục không còn được chọn bằng câu lệnh DELETE.

```php
<?php
// ... (Lấy danh sách danh mục hiện tại của bài viết)
$articleCategories = $article->getCategories($articleId);

// ... (Lấy danh sách danh mục đã chọn từ form)
$selectedCategories = $_POST['categories'];

// ... (Tìm các danh mục cần xóa)
$categoriesToDelete = array_diff($articleCategories,
$selectedCategories);

foreach ($categoriesToDelete as $categoryId) {
    $query = "DELETE FROM article_tag WHERE article_id =
:article_id AND category_id = :category_id";
    // ... (Thực thi truy vấn)
}
```

18. PHP Send EMAIL

- **Tạo Contact Form**
  - Sử dụng thẻ <form> với method="POST" và action="contact.php" (hoặc tên file xử lý form của bạn).
  - Tạo các trường input cho thông tin người dùng:
    - name: Tên người gửi
    - email: Email người gửi
    - subject: Tiêu đề tin nhắn
    - message: Nội dung tin nhắn
  - Sử dụng thẻ <textarea> cho trường message để người dùng có thể nhập nhiều dòng.
  - Thêm nút submit để gửi form.

```
<form action="contact.php" method="post">
  <label for="name">Họ và tên:</label><br>
  <input type="text" id="name" name="name" required><br><br>

  <label for="email">Email:</label><br>
  <input type="email" id="email" name="email"
required><br><br>

  <label for="subject">Tiêu đề:</label><br>
  <input type="text" id="subject" name="subject"
required><br><br>

  <label for="message">Nội dung:</label><br>
  <textarea id="message" name="message" rows="5"
required></textarea><br><br>

  <input type="submit" value="Gửi">
</form>
```

18. PHP Send EMAIL

- **Validate Contact Form**
  - Kiểm tra xem các trường bắt buộc đã được điền hay chưa.
  - Kiểm tra định dạng email.
  - Có thể kiểm tra độ dài của các trường input.
  - Hiển thị thông báo lỗi nếu dữ liệu không hợp lệ.

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = $_POST["name"];
  $email = $_POST["email"];
  $subject = $_POST["subject"];
  $message = $_POST["message"];

  // Kiểm tra các trường bắt buộc
  if (empty($name) || empty($email) || empty($subject) ||
empty($message)) {
    echo "Vui lòng điền đầy đủ thông tin.";
    exit;
  }

  // Kiểm tra định dạng email
  if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Email không hợp lệ.";
    exit;
  }

  // ... (Xử lý gửi email)
}
?>
```

18. PHP Send EMAIL

- **Gửi email bằng PHPMailer**
  - PHPMailer là một thư viện PHP phổ biến để gửi email.
  - Cài đặt PHPMailer: Sử dụng Composer để cài đặt hoặc tải thư viện thủ công
  - Cấu hình PHPMailer: Thiết lập các thông số kết nối đến Mail Server (host, port, username, password).
  - Gửi email: Sử dụng các phương thức của PHPMailer để gửi email (setFrom, addAddress, Subject, Body, send).

```php
<?php
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;

require 'vendor/autoload.php'; // Include thư viện PHPMailer

$mail = new PHPMailer(true);
```

18. PHP Send EMAIL

```
try {
    //Server settings
    $mail->SMTPDebug = SMTP::DEBUG_OFF;                      //Enable verbose debug output
    $mail->isSMTP();                                            //Send using SMTP
    $mail->Host       = 'smtp.example.com';                     //Set the SMTP server to send through
    $mail->SMTPAuth   = true;                                   //Enable SMTP authentication
    $mail->Username   = 'user@example.com';                     //SMTP username
    $mail->Password   = 'password';                               //SMTP password
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;         //Enable TLS encryption;
`PHPMailer::ENCRYPTION_SMTPS` encouraged
    $mail->Port       = 587;                                    //TCP port to connect to, use 465 for
`PHPMailer::ENCRYPTION_SMTPS` above

    //Recipients
    $mail->setFrom('from@example.com', 'Mailer');
    $mail->addAddress('joe@example.net', 'Joe User');     //Add a recipient

    //Content
    $mail->isHTML(true);                                  //Set email format to HTML
    $mail->Subject = 'Here is the subject';
    $mail->Body    = 'This is the HTML message body <b>in bold!</b>';
    $mail->AltBody = 'This is the body in plain text for non-HTML mail clients';

    $mail->send();
    echo 'Message has been sent';
} catch (Exception $e) {
    echo "Message could not be sent. Mailer Error: {$mail->ErrorInfo}";
}
?>
```

18. PHP Send EMAIL

- **Gửi email từ Contact Form**
  - Lấy dữ liệu từ form liên hệ.
  - Sử dụng PHPMailer để gửi email với nội dung từ form.
  - Hiển thị thông báo thành công hoặc lỗi cho người dùng.

```php
<?php
// ... (Validate Contact Form)

// ... (Cấu hình PHPMailer)

//Recipients
$mail->setFrom($email, $name);
$mail->addAddress('your_email@example.com'); // Email của
bạn

//Content
$mail->isHTML(true);
$mail->Subject = $subject;
$mail->Body    = $message;

if($mail->send()) {
  echo "Cảm ơn bạn đã liên hệ!";
} else {
  echo "Có lỗi xảy ra khi gửi email.";
}
?>
```

19. FILE CẤU HÌNH

- **Tách các thông số thành file cấu hình**
  - Lợi ích của việc tách file cấu hình:
    - Dễ dàng thay đổi cấu hình mà không cần sửa code.
    - Nâng cao bảo mật, tránh để lộ thông tin nhạy cảm trong code.
  - Tạo file cấu hình (ví dụ config.php) và lưu trữ các thông số dưới dạng mảng hoặc biến.
  - Include file cấu hình vào các file PHP khác khi cần sử dụng.

```php
// config.php
<?php
return [
    'db_host' => 'localhost',
    'db_name' => 'mydatabase',
    'db_user' => 'username',
    'db_pass' => 'password',
    'api_key' => 'your_api_key',
];
?>

// index.php
<?php
$config = require 'config.php';
$db_host = $config['db_host'];
// ...
?>
```

19. FILE CẤU HÌNH

- **Giới hạn truy cập file cấu hình**
  - Đặt file cấu hình ngoài thư mục web root.
  - Sử dụng .htaccess để chặn truy cập trực tiếp đến file cấu hình.

```
# .htaccess
<Files "config.php">
  Order Allow,Deny
  Deny from all
</Files>
```

Download the code for the PHP & MySQL book and try examples out online

“Câu hỏi & Thảo luận”

THE END!