KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO MÔN HỌC**

**ĐỀ TÀI:**

**PHÁT TRIỂN ỨNG DỤNG**

**WEB VỚI MÃ NGUỒN MỞ**

|  |  |
| --- | --- |
| *Giảng viên hướng dẫn:*  Nguyễn Ngọc Đan Thanh | *Nhóm sinh viên thực hiện:*  Trần Khánh Duy - 110121150  Lê Thị Nguyễn Xuyên - 110121174  Lê Trực Tín - 110121137  Thạch Minh Thắng -  Mã Lớp DA21TTC |

***Trà Vinh, tháng năm 2024***

**LỜI CẢM ƠN**

*Trà Vinh, ngày tháng năm*

Nhóm thực hiện:

Trần Khánh Duy - 110121150

Lê Thị Nguyễn Xuyên - 110121174

Lê Trực Tín – 110121137

Thạch Minh Thắng -

**MỤC LỤC**

**MỤC LỤC HÌNH**

**NHẬN XÉT CỦA NGƯỜI HƯỚNG DẪN**

*Trà Vinh, ngày tháng năm*

Giảng viên hướng dẫn

**NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

**BẢNG PHÂN CÔNG CÔNG VIỆC**

# 

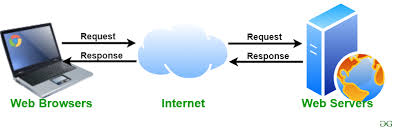
# CHƯƠNG 1: GIỚI THIỆU

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## . Tổng quan về Webserver

### 2.1.1 Webserver là gì?

Web server (máy chủ web) là một phần mềm hoặc phần cứng chịu trách nhiệm nhận, xử lý, và phản hồi các yêu cầu từ trình duyệt web của người dùng. Nó lưu trữ các tài nguyên của website như mã HTML, CSS, JavaScript, hình ảnh, và các tệp khác, đồng thời phân phối chúng qua mạng Internet hoặc mạng nội bộ khi người dùng gửi yêu cầu (request).



Ví dụ: Khi bạn truy cập vào một website bằng cách nhập URL trên trình duyệt, trình duyệt gửi yêu cầu HTTP/HTTPS đến web server, và web server sẽ trả về nội dung trang web tương ứng.

* + 1. **Cách hoạt động của Web Server.**

Web server đóng vai trò trung gian giữa người dùng (client) và máy chủ bằng cách nhận các yêu cầu từ trình duyệt và trả về nội dung tương ứng. Dưới đây là chi tiết về cách hoạt động của web server.

1. **Trình duyệt (Client) gửi yêu cầu đến Web Server**

Khi người dùng nhập URL vào thanh địa chỉ của trình duyệt (ví dụ: www.example.com), trình duyệt sẽ gửi yêu cầu HTTP hoặc HTTPS đến web server.

Yêu cầu có thể là:

* Tải trang web HTML.
* Chạy tập lệnh PHP.
* Lấy dữ liệu từ API (thường trả về JSON).

1. **Web Server Xử Lý Yêu Cầu**

Web server sẽ kiểm tra đường dẫn và tệp được yêu cầu có tồn tại không.

**Nếu là tệp tĩnh** (HTML, CSS, hình ảnh):

* Web server chỉ cần tìm tệp và gửi về trình duyệt.

**Nếu là mã động** (PHP, Python, hoặc ứng dụng web khác):

* Web server sẽ gọi trình thông dịch PHP interpreter hoặc chạy các ứng dụng tương ứng để xử lý mã.
* Ví dụ: Với PHP, máy chủ sẽ chạy đoạn mã PHP và trả về kết quả đã được biên dịch thành HTML cho trình duyệt.

1. **Gửi Phản Hồi (Response) Đến Trình Duyệt**

Sau khi xử lý, web server sẽ gửi lại **phản hồi HTTP** (HTTP Response) về trình duyệt.

* + 1. **XAMPP – Web Server Phổ biến cho Môi trường Phát triển Local**

XAMPP là một gói phần mềm mã nguồn mở được thiết kế để tạo môi trường máy chủ web trên máy tính cá nhân, giúp lập trình viên phát triển và kiểm thử các ứng dụng web cục bộ (local) mà không cần phải triển khai trực tiếp trên máy chủ thực tế.



**X** – Đa nền tảng (Cross-Platform), hỗ trợ **Windows, Linux, macOS**.

**A** – Apache, máy chủ web phổ biến nhất.

**M** – MariaDB (hoặc trước đây là MySQL), hệ quản trị cơ sở dữ liệu.

**P** – PHP, ngôn ngữ lập trình phía server.

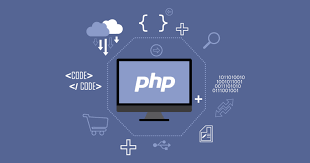
**P** – Perl, ngôn ngữ lập trình đa năng (ít dùng hơn so với PHP trong phát triển web).

* 1. **Tổng quan về PHP**

PHP (viết tắt của PHP: Hypertext Preprocessor) là một ngôn ngữ lập trình phía server được sử dụng rộng rãi để phát triển web động. Ban đầu, PHP được phát triển bởi Rasmus Lerdorf vào năm 1994 và đã nhanh chóng trở thành một trong những công cụ phổ biến nhất để xây dựng các trang web và ứng dụng web.

### 2.2.1 PHP là gì ?

### PHP là một ngôn ngữ mã nguồn mở, được thiết kế đặc biệt để phát triển các ứng dụng web có tính tương tác cao. Nó chạy trên máy chủ và trả về kết quả đã được xử lý dưới dạng HTML hoặc JSON cho trình duyệt của người dùng.



PHP chủ yếu được sử dụng cho:

* Xử lý các biểu mẫu (forms).
* Kết nối và tương tác với cơ sở dữ liệu (MySQL, MariaDB, PostgreSQL).
* Quản lý phiên làm việc (session management).
* Tạo và thao tác tệp trên máy chủ.
* Phát triển hệ thống CMS (Content Management System), như WordPress.

### 2.2.2 Đặc điểm và Tính năng của PHP

PHP là một ngôn ngữ lập trình mạnh mẽ và phổ biến với nhiều đặc điểm nổi bật. Những tính năng này giúp PHP trở thành lựa chọn hàng đầu cho các nhà phát triển web, từ các dự án cá nhân đến các hệ thống lớn như CMS và thương mại điện tử.

**Mã nguồn mở và miễn phí**

PHP được phát hành dưới giấy phép mã nguồn mở, cho phép bất kỳ ai cũng có thể tải xuống, sử dụng và sửa đổi mà không mất phí. Điều này giúp giảm đáng kể chi phí phát triển dự án và cho phép lập trình viên hoặc doanh nghiệp tùy chỉnh mã nguồn để phù hợp với nhu cầu cụ thể của họ. PHP còn được hỗ trợ bởi một cộng đồng lớn, thường xuyên cập nhật và cải thiện hiệu năng, bảo mật.

**Dễ Học và Sử Dụng**

PHP có cú pháp đơn giản và dễ tiếp cận, đặc biệt đối với những người đã quen thuộc với HTML, CSS hoặc JavaScript. Cấu trúc mã của PHP không yêu cầu người học phải có kiến thức sâu về lập trình để bắt đầu, điều này giúp sinh viên và lập trình viên mới nhanh chóng xây dựng được các ứng dụng web cơ bản. Ngoài ra, PHP hỗ trợ rất nhiều tài liệu học tập và thư viện mã nguồn mở, giúp quá trình học trở nên dễ dàng hơn.

**Đa Nền Tảng (Cross-Platform)**

PHP có khả năng hoạt động tốt trên nhiều hệ điều hành như Windows, Linux và macOS, khiến nó trở thành công cụ linh hoạt cho mọi loại môi trường phát triển. Bên cạnh đó, PHP dễ dàng tích hợp với các máy chủ web phổ biến như Apache hoặc Nginx mà không cần cấu hình phức tạp. Điều này mang lại sự thuận tiện cho các lập trình viên, giúp họ triển khai ứng dụng trên nhiều nền tảng khác nhau.

**Hỗ Trợ Nhiều Cơ Sở Dữ Liệu**

PHP có khả năng kết nối với nhiều hệ quản trị cơ sở dữ liệu khác nhau, mang lại sự linh hoạt trong lưu trữ và quản lý dữ liệu. Các cơ sở dữ liệu phổ biến mà PHP hỗ trợ bao gồm:

* MySQL/MariaDB: Hệ quản trị cơ sở dữ liệu quan hệ phổ biến nhất cho các ứng dụng web.
* PostgreSQL: Cơ sở dữ liệu mạnh mẽ và hỗ trợ nhiều tính năng phức tạp hơn MySQL.
* SQLite: Cơ sở dữ liệu nhỏ gọn, phù hợp cho các ứng dụng nhỏ hoặc thử nghiệm.
* MongoDB: Cơ sở dữ liệu NoSQL, thích hợp cho các ứng dụng có dữ liệu phi cấu trúc.

Sự đa dạng này giúp lập trình viên lựa chọn công cụ phù hợp nhất với yêu cầu của từng dự án.

**Tương Thích với HTML và JavaScript**

PHP có thể được nhúng trực tiếp vào mã HTML, điều này cho phép lập trình viên tạo ra các trang web động một cách dễ dàng. Ví dụ, PHP có thể hiển thị nội dung khác nhau tùy vào dữ liệu được người dùng nhập vào. Bên cạnh đó, PHP hoạt động tốt với JavaScript và AJAX – các công nghệ hỗ trợ giao tiếp không đồng bộ giữa trình duyệt và máy chủ, mang đến trải nghiệm mượt mà cho người dùng mà không cần tải lại toàn bộ trang.

**Xử Lý Dữ Liệu và Session**

Một trong những tính năng quan trọng của PHP là quản lý session và cookie, cho phép duy trì trạng thái người dùng trong suốt phiên làm việc. Điều này rất cần thiết trong các trang web yêu cầu đăng nhập hoặc lưu trữ thông tin giỏ hàng trong thương mại điện tử. Ngoài ra, PHP cũng hỗ trợ quản lý dữ liệu POST và GET từ biểu mẫu web, giúp lập trình viên dễ dàng xử lý thông tin người dùng và phản hồi tương tác.

PHP cũng cho phép mã hóa cookie và session để đảm bảo an toàn, giảm thiểu nguy cơ bị tấn công đánh cắp dữ liệu. Khả năng này rất hữu ích cho các hệ thống web có yêu cầu bảo mật cao như cổng thông tin người dùng, ngân hàng trực tuyến, hoặc dịch vụ thương mại điện tử.

### 2.2.3 Ứng Dụng của PHP trong Thực Tế

PHP là một ngôn ngữ lập trình phổ biến và có mặt trong nhiều lĩnh vực phát triển web. Nhờ tính linh hoạt, dễ sử dụng và khả năng mở rộng cao, PHP đã và đang được sử dụng rộng rãi trong nhiều loại hình dự án thực tế, từ các trang web doanh nghiệp, hệ thống quản trị nội dung đến các nền tảng thương mại điện tử và API.

**Xây Dựng Website và Ứng Dụng Web**

PHP là lựa chọn phổ biến cho việc phát triển các trang web doanh nghiệp, cổng thông tin, và hệ thống quản lý tài nguyên. Các trang web động được xây dựng bằng PHP có khả năng xử lý dữ liệu, tương tác với người dùng và kết nối với cơ sở dữ liệu một cách hiệu quả.

* Doanh nghiệp: PHP được dùng để xây dựng các trang web giới thiệu doanh nghiệp với các tính năng tương tác như biểu mẫu liên hệ, đặt hàng trực tuyến hoặc hệ thống đăng nhập cho nhân viên.
* Cổng thông tin (Portal): Nhiều tổ chức, trường học và chính phủ sử dụng PHP để phát triển các cổng thông tin nhằm quản lý dữ liệu người dùng và cung cấp thông tin công cộng.
* Ứng dụng web nội bộ: PHP cũng được sử dụng để tạo các hệ thống quản lý nội bộ, như phần mềm quản lý nhân sự (HRM) hoặc quản lý tài liệu (DMS).

Với các framework như Laravel và CodeIgniter, việc phát triển các ứng dụng web phức tạp trở nên dễ dàng hơn nhờ các tính năng hỗ trợ mạnh mẽ.

**Quản lý Nội dung (CMS)**

Nhiều hệ thống CMS nổi tiếng được viết bằng PHP, cho phép người dùng dễ dàng tạo và quản lý nội dung trang web mà không cần kiến thức sâu về lập trình.

* WordPress: Là hệ thống quản trị nội dung (CMS) phổ biến nhất thế giới, chiếm hơn 40% số lượng trang web hiện nay. WordPress cho phép người dùng tạo các blog, trang bán hàng và website doanh nghiệp với nhiều tùy chọn mở rộng thông qua các plugin và chủ đề.
* Drupal: Một CMS mã nguồn mở mạnh mẽ, phù hợp cho việc phát triển các trang web lớn và phức tạp, như cổng thông tin doanh nghiệp và hệ thống quản lý học tập.
* Joomla: Một hệ thống quản lý nội dung khác với tính năng linh hoạt, hỗ trợ đa ngôn ngữ và có khả năng mở rộng với các module. Joomla được sử dụng trong nhiều dự án như diễn đàn cộng đồng và trang tin tức.

**Thương Mại Điện Tử (E-commerce)**

PHP đóng vai trò quan trọng trong việc xây dựng các nền tảng thương mại điện tử, giúp doanh nghiệp quản lý sản phẩm, đơn hàng và khách hàng một cách hiệu quả.

* Magento: Một giải pháp thương mại điện tử mã nguồn mở, được thiết kế để phục vụ các doanh nghiệp vừa và lớn. Magento cung cấp nhiều tính năng quản lý bán hàng, giỏ hàng và thanh toán trực tuyến, đồng thời cho phép tùy chỉnh theo nhu cầu của từng doanh nghiệp.
* WooCommerce: Plugin bán hàng nổi tiếng được tích hợp trên WordPress, giúp các trang web WordPress dễ dàng chuyển đổi thành cửa hàng trực tuyến. WooCommerce phù hợp với cả những doanh nghiệp nhỏ và cửa hàng cá nhân nhờ tính dễ sử dụng và tích hợp nhiều tiện ích mở rộng.

### 2.2.4 Ưu và nhược điểm của PHP

|  |  |
| --- | --- |
| Ưu điểm | Nhược điểm |
| * Tốc độ xử lý nhanh * Cộng đồng lớn * Khả năng mở rộng * Bảo mật tốt * Hỗ trợ OOP | * Hiệu năng không cao bằng một số ngôn ngữ khác * Mã nguồn có thể khó bảo trì nếu không tổ chức tốt. * Yêu cầu bảo mật cao |

* 1. **Làm quen với các khái niệm PHP cơ bản**
     1. **PHP Tags**

Mã PHP được bắt đầu bằng <?php và kết thúc bằng ?>. Nó tương tự như tất cả các thẻ HTML vì tất cả chúng đều bắt đầu bằng biểu tượng (<) và kết thúc bằng biểu tượng (>). Các ký hiệu <?php và ?> được gọi là thẻ PHP.

<html>

<body>

<?php

echo "Nội dung động từ PHP!";

?>

</body>

</html>

Các thẻ PHP giúp cho máy chủ web hiểu nơi mã PHP bắt đầu và kết thúc, văn bản giữa các thẻ này được hiểu là PHP. Còn lại bất kỳ văn bản nào bên ngoài các thẻ này đều được coi là HTML bình thường.

* + 1. **PHP Keywords**

Trong PHP, keywords (từ khóa) là những từ được định nghĩa sẵn và có ý nghĩa đặc biệt trong ngôn ngữ lập trình. Chúng không thể được sử dụng làm tên biến, hàm, hoặc lớp. Dưới đây là danh sách các từ khóa phổ biến trong PHP cùng với mô tả ngắn gọn về chức năng của chúng:

|  |  |
| --- | --- |
| **abstract**: Được sử dụng để khai báo một lớp hoặc phương thức trừu tượng.  **and**: Tương đương với toán tử logic &&.  **array**: Được sử dụng để định nghĩa mảng.  **as**: Sử dụng trong vòng lặp foreach để chỉ định tên biến cho mỗi phần tử.  **break**: Dùng để thoát khỏi vòng lặp hoặc switch.  **const**: Khai báo một hằng số trong lớp.  **class**: Khai báo một lớp trong PHP.  **echo**: Được sử dụng để in ra một chuỗi.  **else**: Sử dụng trong câu lệnh điều kiện if để chỉ định mã sẽ thực thi nếu điều kiện không đúng.  **elseif**: Kết hợp với if để kiểm tra nhiều điều kiện.  **empty**: Kiểm tra xem một biến có rỗng hay không.  **exit**: Kết thúc một kịch bản PHP.  **extends**: Sử dụng để kế thừa từ một lớp khác. | **function**: Khai báo một hàm.  **global**: Để sử dụng biến toàn cục trong một hàm.  **if**: Bắt đầu một cấu trúc điều kiện.  **for**: Bắt đầu một vòng lặp for.  **implements**: Sử dụng để thực hiện một giao diện.  **include**, **require**: Để chèn một file PHP khác vào file hiện tại.  **interface**: Khai báo một giao diện.  **isset**: Kiểm tra xem một biến có được khai báo và khác NULL hay không.  **list**: Dùng để gán giá trị của mảng vào nhiều biến.  **new**: Tạo một thể hiện mới của lớp.  **private**, **protected**, **public**: Định nghĩa mức độ truy cập cho các thuộc tính và phương thức trong lớp.  **unset**: Xóa một biến. |

* + 1. **Biến ($)**

Biến là các đối tượng được sử dụng để lưu trữ dữ liệu. Biến có thể chứa các loại dữ liệu khác nhau, như số nguyên, số thực, chuỗi, mảng, đối tượng, và nhiều hơn nữa.

**Khai báo biến**

Biến trong PHP được khai báo bằng cách sử dụng ký tự “$” trước tên biến. Tên biến có thể chứa chữ cái, số và dấu gạch dưới, nhưng không thể bắt đầu bằng số.

$tenBien = giaTri;

//Exp

$ten = "Nguyễn Văn A";

$tuoi = 25;

**Quy tắc đặt tên biến**

* Tên biến phải bắt đầu bằng ký tự chữ cái hoặc dấu gạch dưới (\_).
* Tên biến không được bắt đầu bằng số.
* Tên biến phân biệt chữ hoa và chữ thường (ví dụ: $Ten và $ten là hai biến khác nhau).
* Không sử dụng từ khóa hoặc ký tự đặc biệt (ngoại trừ dấu gạch dưới).

**Kiểu dữ liệu trong biến**

PHP là một ngôn ngữ dynamically typed (kiểu dữ liệu động), có nghĩa là bạn không cần chỉ định kiểu dữ liệu khi khai báo biến. PHP sẽ tự động xác định kiểu dữ liệu dựa trên giá trị mà biến nhận.

**Chuỗi (String)**: Dùng để lưu trữ chuỗi ký tự.

$ten = "Nguyễn Văn A";

**Số nguyên (Integer)**: Dùng để lưu trữ số nguyên.

$tuoi = 25;

**Số thực (Float)**: Dùng để lưu trữ số thực (số thập phân).

$diem = 8.5;

**Boolean**: Dùng để lưu trữ giá trị đúng hoặc sai.

$duyet = true;

**Mảng (Array)**: Dùng để lưu trữ một tập hợp các giá trị.

$mang = array(1, 2, 3);

**Đối tượng (Object)**: Dùng để lưu trữ một thể hiện của một lớp.

class Nguoi {

public $ten;

public $tuoi;

}

$nguoi = new Nguoi();

$nguoi->ten = "Nguyễn Văn A";

**Thay đổi kiểu, giá trị biến**

PHP không yêu cầu bạn phải khai báo kiểu dữ liệu rõ ràng cho biến khi bạn khởi tạo nó. PHP tự động suy luận kiểu dựa trên giá trị của biến tại thời điểm được khai báo. Điều này làm cho PHP linh hoạt hơn nhưng cũng có thể dẫn đến các lỗi không mong muốn do việc chuyển đổi kiểu dữ liệu tự động.

<?php

$number = 10; // Đây là một số nguyên (int)

$number = "10"; // Biến được chuyển thành chuỗi (string) mà không có lỗi

?>

**Biến toàn cục và biến cục bộ**

Biến cục bộ là biến được khai báo bên trong một hàm. Nó chỉ có thể được truy cập và sử dụng trong hàm đó. Khi hàm kết thúc, biến cục bộ sẽ không còn tồn tại và không thể truy cập từ bên ngoài.

function tinhTong($a, $b) {

$tong = $a + $b; // biến cục bộ

return $tong;

}

echo tinhTong(5, 10); // In ra: 15

// echo $tong; // Lỗi: biến cục bộ không thể truy cập bên ngoài hàm

Biến toàn cục là biến được khai báo bên ngoài bất kỳ hàm nào. Nó có thể được truy cập và sử dụng từ bất kỳ đâu trong mã, bao gồm cả bên trong các hàm. Để sử dụng biến toàn cục trong một hàm, bạn cần khai báo từ khóa **global.**

$bienToanCuc = "Hello";

function ham() {

global $bienToanCuc; // Sử dụng biến toàn cục

echo $bienToanCuc;

}

ham(); // In ra: Hello

**Ép kiểu**

Type Casting ( ép kiểu ) trong PHP là quá trình chuyển đổi một biến từ kiểu dữ liệu này sang kiểu dữ liệu khác một cách rõ ràng. Có thể ép kiểu một biến sang kiểu dữ liệu khác bằng cách đặt kiểu dữ liệu mong muốn vào trong ngoặc đơn trước biến.

$var = "123"; // var đang mang giá trị kiểu string

// Ép chuỗi thành số nguyên

$number = (int) $var; // var đã mang giá trị kiểu integer

**Biến Biến (Variable Variables)**

PHP cho phép bạn sử dụng tên biến động. Điều này có nghĩa là bạn có thể sử dụng giá trị của một biến để tạo ra tên biến khác.

$tenBien = "bien1";

$$tenBien = "Giá trị của biến 1";

echo $bien1; // In ra: Giá trị của biến 1

**Nối chuỗi**

Trong PHP bạn có thể nối (kết hợp) hai hoặc nhiều chuỗi thành một chuỗi duy nhất bằng cách sử dụng dấu “.” giữa các chuỗi cần nối.

<?php

$firstName = "Ho";

$lastName = "Ten";

// Nối chuỗi

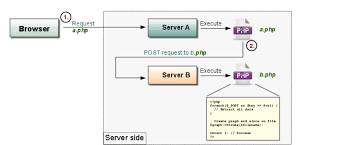
$fullName = $firstName . " " . $lastName;

echo $fullName; // Kết quả: Ho Ten

?>

**Truy cập biến trong form**

Trong PHP, để truy cập các biến từ form HTML (các dữ liệu mà người dùng nhập vào form), bạn có thể sử dụng các biến siêu toàn cục như $\_GET, $\_POST, hoặc $\_REQUEST, tùy thuộc vào phương thức gửi dữ liệu từ form.



* **Phương thức POST**

Nếu form được gửi bằng phương thức POST, bạn sử dụng mảng $\_POST để lấy dữ liệu. Dữ liệu từ form sẽ được gửi ẩn trong phần body của yêu cầu HTTP và không hiển thị trong URL.

<form method="**POST**" action="submit.php">

Tên: <input type="text" name="name"><br>

Tuổi: <input type="number" name="age"><br>

<input type="submit" value="Gửi">

</form>

//Xử lý lấy dữ liệu gửi từ form

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "**POST**") {

$name = **$\_POST**['name']; // Lấy giá trị của input có name="name"

$age = **$\_POST**['age']; // Lấy giá trị của input có name="age"

echo "Tên của bạn là: " . **htmlspecialchars**($name) . "<br>";

echo "Tuổi của bạn là: " . (int)$age;

}

?>

* **Phương thức GET**

Nếu form được gửi bằng phương thức GET, bạn sẽ sử dụng mảng $\_GET để lấy dữ liệu. Dữ liệu từ form sẽ được đính kèm vào URL dưới dạng chuỗi truy vấn (query string).

Cách thức truy cập dữ liệu từ form sử dụng POST và GET trong PHP là giống nhau về mặt cú pháp. Cả hai đều truy cập dữ liệu từ form bằng cách sử dụng tên trường (field\_name) mà bạn đã định nghĩa trong form HTML, chỉ khác nhau ở cách dữ liệu được gửi từ trình duyệt đến server:

* Với GET, dữ liệu được đính kèm vào URL.
* Với POST, dữ liệu được gửi trong body của yêu cầu HTTP.

**Vậy nên**, bạn cần phân định rõ khi nào nên sử dụng POST và GET.

* Sử dụng GET khi bạn muốn lấy dữ liệu từ server mà không làm thay đổi trạng thái của nó, ví dụ như khi tìm kiếm hoặc truy vấn dữ liệu.
* Sử dụng POST khi bạn muốn gửi dữ liệu nhạy cảm hoặc khi thực hiện các thao tác làm thay đổi dữ liệu trên server, như khi gửi form đăng ký, đăng nhập, hoặc tải lên file.
* **Phương thức REQUEST**

Có thể sử dụng mảng $\_REQUEST để truy cập biến từ cả hai phương thức GET và POST. Tuy nhiên, trong thực tế, nên ưu tiên sử dụng $\_POST hoặc $\_GET để rõ ràng hơn về nguồn gốc của dữ liệu.

* + 1. **Câu lệnh điều kiện**

Câu lệnh điều kiện được sử dụng để thực hiện các thao tác dựa trên một điều kiện nào đó. PHP cung cấp nhiều loại câu lệnh điều kiện để kiểm tra các điều kiện và điều khiển luồng chương trình. Dưới đây là các loại câu lệnh điều kiện phổ biến trong PHP:

**Câu lệnh if**

Câu lệnh if kiểm tra một điều kiện và chỉ thực thi khối lệnh nếu điều kiện đó đúng (true).

//Cú pháp

if (điều\_kiện) {

// Khối lệnh được thực thi nếu điều kiện đúng

}

//ví dụ

$tuoi = 20;

if ($tuoi >= 18) {

echo "Bạn đã đủ tuổi trưởng thành.";

}

**Câu lệnh if…else**

Câu lệnh if...else cho phép bạn thực thi một khối lệnh nếu điều kiện đúng, và một khối lệnh khác nếu điều kiện sai.

//Cú pháp

if (điều\_kiện) {

// Khối lệnh nếu điều kiện đúng

} else {

// Khối lệnh nếu điều kiện sai

}

//ví dụ

$diem = 7;

if ($diem >= 5) {

echo "Bạn đã đậu.";

} else {

echo "Bạn đã rớt.";

}

**Câu Lệnh if...elseif...else**

Câu lệnh này cho phép kiểm tra nhiều điều kiện khác nhau. Nếu điều kiện đầu tiên sai, nó sẽ kiểm tra các điều kiện elseif tiếp theo, và nếu tất cả đều sai, nó sẽ thực hiện khối lệnh else.

//cú pháp

if (điều\_kiện\_1) {

// Khối lệnh nếu điều kiện\_1 đúng

} elseif (điều\_kiện\_2) {

// Khối lệnh nếu điều kiện\_2 đúng

} else {

// Khối lệnh nếu tất cả điều kiện đều sai

}

//ví dụ

$diem = 8;

if ($diem >= 9) {

echo "Xuất sắc";

} elseif ($diem >= 7) {

echo "Khá";

} else {

echo "Trung bình";

}

**Câu Lệnh switch**

Câu lệnh switch được sử dụng để so sánh một biến hoặc biểu thức với nhiều giá trị khác nhau và thực thi khối lệnh tương ứng.

|  |  |
| --- | --- |
| //cú pháp  switch (biểu\_thức) {  case giá\_trị\_1:  // Khối lệnh nếu biểu\_thức == giá\_trị\_1  break;  case giá\_trị\_2:  // Khối lệnh nếu biểu\_thức == giá\_trị\_2  break;  default:  // Khối lệnh nếu không có giá trị nào khớp  } | //ví dụ  $thang = 5;  switch ($thang) {  case 1:  echo "Tháng Một";  break;  case 2:  echo "Tháng Hai";  break;  case 5:  echo "Tháng Năm";  break;  default:  echo "Không tìm thấy tháng.";  } |

# CHƯƠNG 3: THỰC HÀNH LẬP TRÌNH WEB VỚI PHP

## 3.1 TUẦN 1

Ở tuần này, nội dung thực hành là xây dựng một trang web cho phép tải lên hình ảnh từ ổ đĩa máy tính và hiển thị ra trang web một danh sách các “sản phẩm” cùng với thông tin, hình ảnh đã tải lên.

### 3.1.1 Cấu trúc thư mục

### |----CSDL(baitapso1)

### | |--sanpham(id\_sp[tự động tăng], ten\_sp, gia\_sp, mota\_sp, id\_img)

### | |--hinhanh(id\_img[tự động tăng], img\_data)

### |----img //tập tin chứa các hình ảnh khi đã thêm vào.

### |----connect.php //trang kết nối cơ sở dữ liệu MySQL

### |----sanpham.php //trang giao diện sẽ hiển thị danh sách các sản phẩm

### |----themsanpham.php

### 3.1.2 connect.php

Ở tập tin này sẽ tiến hành kết nối cơ sở dữ liệu.

//Cú pháp

$connection = mysqli\_connect(host, username, password, database);

if (!$connection) {

die("Kết nối thất bại: " . mysqli\_connect\_error());

} else {

echo "Kết nối thành công!";

}

//Ví dụ

<?php

$host = "localhost"; // Tên máy chủ

$username = "root"; // Tên người dùng MySQL

$password = ""; // Mật khẩu (nếu có)

$database = "your\_database"; // Tên cơ sở dữ liệu

// Kết nối

$connection = mysqli\_connect($host, $username, $password, $database);

// Kiểm tra kết nối

if (!$connection) {

die("Kết nối thất bại: " . mysqli\_connect\_error());

} else {

echo "Kết nối thành công!";

}

?>

* **Khai báo các thông tin kết nối**

$host = "localhost"; // Tên máy chủ

$username = "root"; // Tên người dùng MySQL

$password = ""; // Mật khẩu (nếu có)

$database = "your\_database"; // Tên cơ sở dữ liệu

**$host**: Tên máy chủ của cơ sở dữ liệu. Trong môi trường phát triển, localhost thường được sử dụng để chỉ máy chủ nội bộ.

**$username**: Tên người dùng MySQL. Mặc định thường là "root" trong các máy chủ cục bộ như XAMPP, WAMP.

**$password**: Mật khẩu của tài khoản MySQL. Trong môi trường cục bộ (localhost), mật khẩu thường để trống ("").

**$database**: Tên cơ sở dữ liệu bạn muốn kết nối.

* **Kết nối đến cơ sở dữ liệu bằng MySQL**

$connection = mysqli\_connect($host, $username, $password, $database);

**mysqli\_connect()**: Hàm này dùng để tạo kết nối đến cơ sở dữ liệu MySQL. Nó nhận 4 tham số: host, username, password, database.

* **Kiểm tra kết nối thành công hay thất bại**

if (!$connection) {

die("Kết nối thất bại: " . mysqli\_connect\_error());

} else {

echo "Kết nối thành công!";

}

**if (!$connection)**: Kiểm tra xem kết nối có thành công không. Dấu ! là phủ định, có nghĩa là nếu kết nối trả về false thì sẽ thực hiện khối lệnh bên trong if.

**mysqli\_connect\_error()**: Hàm này trả về thông báo lỗi nếu kết nối thất bại.

**die():** Dừng chương trình và in ra thông báo lỗi.

### 3.1.3 themsanpham.php

Ở tập tin này sẽ tiến hành xây dựng giao diện cũng như xử lý form để cho phép thêm vào một “sản phẩm” bao gồm Tên sản phẩm, Ảnh sản phẩm, Giá, Thương hiệu, Mô tả vào CSDL.

**3.1.3.1 HTML form**

<form method="POST" enctype="multipart/form-data">

<label for="ten\_sp">Tên Sản Phẩm:</label><br>

<input type="text" name="ten\_sp" required><br><br>

<label for="image">Ảnh Sản Phẩm:</label><br>

<input type="file" name="image"><br><br>

<label for="gia\_sp">Giá:</label><br>

<input type="number" name="gia\_sp" required><br>

<label for="hieu\_sp">Thương Hiệu:</label><br>

<input type="text" name="hieu\_sp" required><br><br>

<label for="mota\_sp">Mô Tả:</label><br>

<input type="text" name="mota\_sp" required><br><br>

<button name="them\_sp" type="submit">Thêm</button>

</form>

Form có phương thức gửi dữ liệu là POST và thuộc tính enctype có giá trị là "multipart/form-data". Thuộc tính này xác định cách dữ liệu từ form sẽ được mã hóa khi gửi tới server và khi form chứa các trường cho phép người dùng tải lên tệp (ví dụ: ảnh, video, tài liệu). Nó cho phép gửi cả văn bản và tệp dưới dạng các phần riêng biệt (multipart), mà không mã hóa chúng vào dạng URL.

Nội dung trong form là các input nhập thông tin sản phẩm với các trường cần thiết như tên sản phẩm, giá, thương hiệu, mô tả, và ảnh. Các trường được đặt tên <name> tương ứng để xử lý dữ liệu.

**3.1.3.2 Xử lý dữ liệu bằng PHP**

<?php

include './connect.php'; // Kết nối với cơ sở dữ liệu

if (isset($\_POST['them\_sp'])) {

$ten\_sp = $\_POST['ten\_sp'];

$gia\_sp = $\_POST['gia\_sp'];

$mota\_sp = $\_POST['mota\_sp'];

$hieu\_sp = $\_POST['hieu\_sp'];

$image = $\_FILES['image']['name'];

$image\_tmp = $\_FILES['image']['tmp\_name'];

// Chèn dữ liệu ảnh vào bảng 'hinhanh'

$sql\_image = "INSERT INTO hinhanh (img\_data) VALUES ('$image')";

mysqli\_query($connect, $sql\_image);

$image\_id = mysqli\_insert\_id($connect);

// Chèn dữ liệu sản phẩm vào bảng 'sanpham'

$sql\_product = "INSERT INTO sanpham (ten\_sp, gia\_sp, hieu\_sp, mota\_sp, id\_img)

VALUES ('$ten\_sp', '$gia\_sp', '$hieu\_sp', '$mota\_sp', '$image\_id')";

mysqli\_query($connect, $sql\_product);

move\_uploaded\_file($image\_tmp, 'img/' . $image);

header('Location: sanpham.php');

}

?>

**3.1.3.3 Kiểm tra xem người dùng có gửi biểu mẫu không**

if (isset($\_POST['them\_sp'])) {

}

**isset($\_POST['them\_sp'])**: Kiểm tra xem nút "Thêm" trong biểu mẫu đã được nhấn chưa. Nếu biểu mẫu đã được gửi, khối mã bên trong **if** sẽ được thực thi.

**3.1.3.4 Nhận dữ liệu từ form**

$ten\_sp = $\_POST['ten\_sp'];

$gia\_sp = $\_POST['gia\_sp'];

$mota\_sp = $\_POST['mota\_sp'];

$hieu\_sp = $\_POST['hieu\_sp'];

Ở đây các dữ liệu với '*name*' trong <input> sẽ được lấy ra bằng phương thức $\_POST và gán vào biến với tên tương ứng. Lúc này các biến $ten\_sp, $gia\_sp….. sẽ lưu trữ giá trị nhập vào từ form.

**3.1.3.5 Xử lý file ảnh tải lên**

$image = $\_FILES['image']['name'];

$image\_tmp = $\_FILES['image']['tmp\_name'];

**$\_FILES** là một biến toàn cục chứa dữ liệu file tải lên. Hai thuộc tính chính được sử dụng:

* **$\_FILES['image']['name']**: Lấy tên file ảnh mà người dùng tải lên và gán vào biến $image.
* **$\_FILES['image']['tmp\_name']**: Đường dẫn tạm thời nơi file được lưu trên máy chủ sau khi tải lên và gán vào biến $image\_tmp (có nghĩa là biến tạm).

**3.1.3.6 Chèn dữ liệu ảnh vào bảng hinhanh**

$sql\_image = "INSERT INTO hinhanh (img\_data) VALUES ('$image')";

mysqli\_query($connect, $sql\_image);

Câu lệnh SQL: Chèn tên ảnh vào bảng hinhanh trong cơ sở dữ liệu.

**mysqli\_query()**: Lệnh thực thi câu lệnh SQL với kết nối cơ sở dữ liệu.

**3.1.3.7 Lấy id của ảnh vừa được thêm vào CSDL**

$image\_id = mysqli\_insert\_id($connect);

**mysqli\_insert\_id($connect)**: Lấy ID tự động tăng của bản ghi ảnh vừa được thêm vào bảng hinhanh. ( ID này sẽ được dùng làm khóa ngoại trong bảng sanpham)

**3.1.3.8 Chèn dữ liệu sản phẩm vào bảng sanpham**

$sql\_product = "INSERT INTO sanpham (ten\_sp, gia\_sp, hieu\_sp, mota\_sp, id\_img)

VALUES ('$ten\_sp', '$gia\_sp', '$hieu\_sp', '$mota\_sp', '$image\_id')";

mysqli\_query($connect, $sql\_product);

Câu lệnh SQL: Chèn dữ liệu sản phẩm vào bảng sanpham với các thông tin: ten\_sp, gia\_sp, hieu\_sp, mota\_sp, id\_img (ID ảnh – khóa ngoại) với dữ liệu thêm vào là các $biến đã được gán giá trị tương ứng ở câu lệnh **Nhận dữ liệu từ form.**

**3.1.3.9 Di chuyển file ảnh từ thư mục tạm vào thư mục img/**

move\_uploaded\_file($image\_tmp, 'img/' . $image);

**move\_uploaded\_file()**: Di chuyển ảnh từ thư mục tạm thời $image\_tmp vào thư mục **img/** trên máy chủ, với tên file từ biến $image được giữ nguyên.

**3.1.3.10 Chuyển hướng người dùng**

header('Location: sanpham.php');

**header()**: Chuyển hướng người dùng sang trang **sanpham.php** sau khi sản phẩm được thêm thành công.

### 3.1.4 sanpham.php

Đây là trang mà chúng ta sẽ xây dựng giao diện chính để hiển thị danh sách các sản phẩm từ CSDL.

<body>

<div>

        <h2>Danh Sách Sản Phẩm</h2>

        <table>

            <thead>

            <tr>

                <th>STT</th>

                <th>Tên Sản Phẩm</th>

                <th>Ảnh Sản Phẩm</th>

                <th>Giá</th>

                <th>Thương Hiệu</th>

                <th>Mô Tả</th>

            </tr>

            </thead>

            <tbody>

            <?php

            include './connect.php';

            $sql\_products = "SELECT \* FROM sanpham inner join hinhanh on sanpham.id\_img = hinhanh.id\_img";

            $query\_products = mysqli\_query($connect,$sql\_products);

                $i = 1;

                    while($row = mysqli\_fetch\_assoc($query\_products)){ ?>

                    <tr>

                        <td><?php echo $i++; ?></td>

                        <td><?php echo $row['ten\_sp']; ?></td>

                        <td>

                            <img *style* ="width: 120px; height: auto;" *src*="img/<?php echo $row['img\_data']; ?>">

                        </td>

                        <td *style*="color:red"><?php echo number\_format($row['gia\_sp']),'',''.' VND' ;?></td>

                        <td><?php echo $row['hieu\_sp']; ?></td>

                        <td><?php echo $row['mota\_sp']; ?></td>

                    </tr>

                <?php  }?>

            </tbody>

        </table>

    </div>

</body>

* + - 1. **Tạo giao diện hiển thị danh sách sản phẩm**

<h2>Danh Sách Sản Phẩm</h2>

<table>

<thead>

<tr>

<th>STT</th>

<th>Tên Sản Phẩm</th>

<th>Ảnh Sản Phẩm</th>

<th>Giá</th>

<th>Thương Hiệu</th>

<th>Mô Tả</th>

</tr>

</thead>

Sử dụng thẻ <table> để tạo thành một hàng các tên cột chứa dữ liệu tương ứng.

* + - 1. **Kết nối CSDL và truy vấn dữ liệu sản phẩm**

include './connect.php'; // Kết nối với cơ sở dữ liệu thông qua file connect.php

$sql\_products = "SELECT \* FROM sanpham

INNER JOIN hinhanh

ON sanpham.id\_img = hinhanh.id\_img";

$query\_products = mysqli\_query($connect, $sql\_products);

**$sql\_products**: Lệnh SQL dùng để lấy tất cả dữ liệu từ bảng sanpham và hinhanh bằng cách JOIN hai bảng thông qua cột chung **id\_img** (khóa ngoại từ bảng sanpham liên kết đến khóa chính trong bảng hinhanh).

**mysqli\_query()**: Thực thi lệnh SQL trên kết nối cơ sở dữ liệu và lưu kết quả vào biến $query\_products.

* + - 1. **Xử lý hiển thị từng sản phẩm trong bảng**

$i = 1;

while ($row = mysqli\_fetch\_assoc($query\_products)) { ?>

<tr>

<td><?php echo $i++; ?></td>

<td><?php echo $row['ten\_sp']; ?></td>

<td>

<img style="width: 120px; height: auto;" src="img/<?php echo $row['img\_data']; ?>">

</td>

<td style="color:red"><?php echo number\_format($row['gia\_sp'], 0, '', ',') . ' VND'; ?></td>

<td><?php echo $row['hieu\_sp']; ?></td>

<td><?php echo $row['mota\_sp']; ?></td>

</tr>

<?php } ?>

* **$i = 1;** Khởi tạo biến đếm số thứ tự (STT) cho mỗi sản phẩm với giá trị ban đầu là 1.

while ($row = mysqli\_fetch\_assoc($query\_products)) {

// Sử dụng $row để truy cập thông tin sản phẩm

}

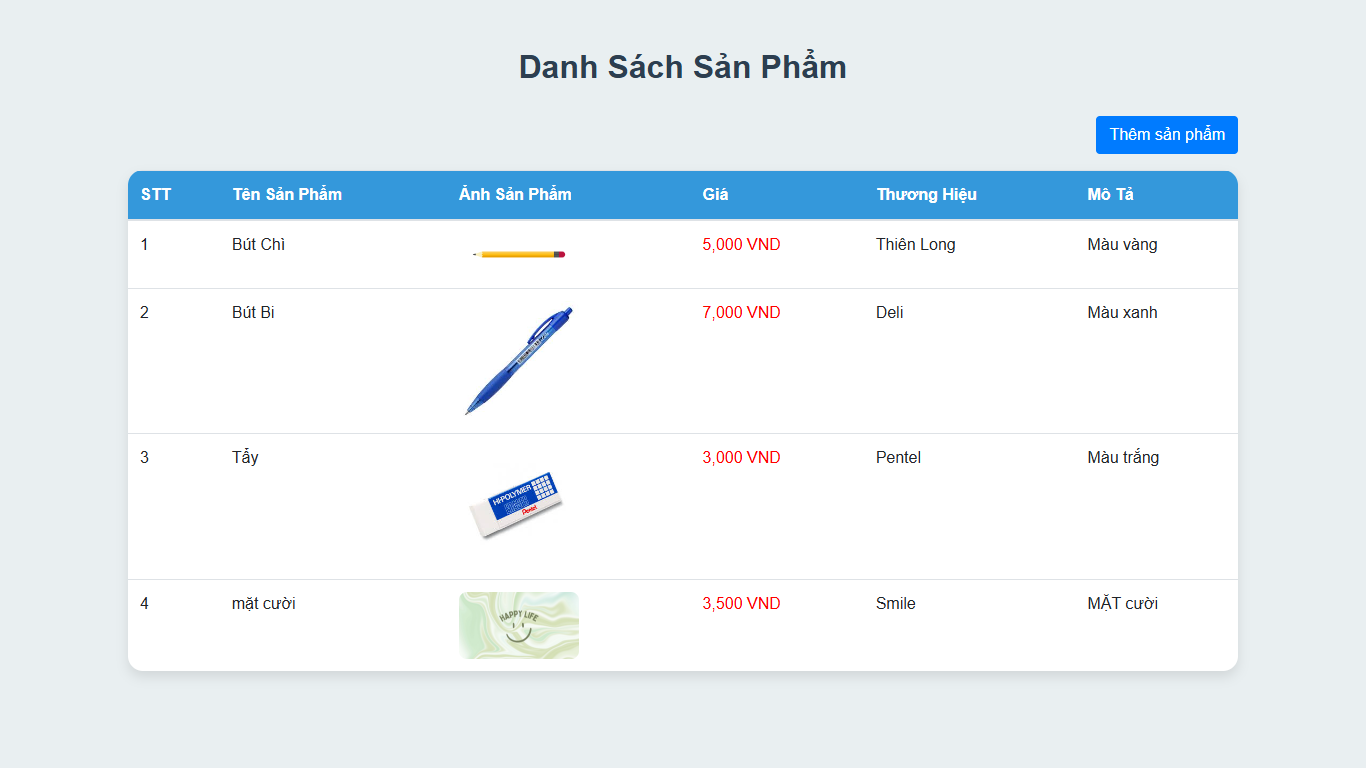
* **mysqli\_fetch\_assoc()**: Đây là một hàm trong PHP được sử dụng để lấy một hàng dữ liệu từ kết quả của truy vấn cơ sở dữ liệu **($query\_products**) dưới dạng một mảng kết hợp và gán vào biến **$row.**
* **while**: Sử dụng vòng lặp lấy từng dòng dữ liệu từ kết quả truy vấn dưới dạng mảng kết hợp được gán vào biến **$row** và hiển thị từng dòng trong bảng.

**Trong mỗi dòng dữ liệu của <td>:**

1. Cột STT: Hiển thị số thứ tự và tăng dần thông qua biến $i++.
2. Cột Tên Sản Phẩm: Lấy từ *ten\_sp* trong CSDL.
3. Cột Ảnh Sản Phẩm: Hiển thị ảnh từ thư mục img/ với tên ảnh từ cột *img\_data* trong cơ sở dữ liệu.
4. Cột Giá: Hiển thị giá lấy từ *gia\_sp* và sử dụng number\_format() để định dạng số (thêm dấu phân cách hàng nghìn).
5. Cột Thương Hiệu: Lấy từ *hieu\_sp*.
6. Cột Mô Tả: Lấy từ *mota\_sp*.

### 3.1.5 Kết quả đạt được

Qua các đoạn code trên, bạn có thể sử dụng thêm các câu lệnh định dạng cho giao diện trang web trông thân thiện, đẹp mắt hơn bằng cách sử dụng các thư viện Bootstrap, CSS…..cùng với các câu lệnh kiểm tra điều kiện trong việc xử lý khi nhập nội dung từ form, khi thêm dữ liệu vào CSDL để có được một dự án hoàn hảo.



## 3.2 TUẦN 2

Ở tuần này, sẽ xây dựng trang web để quản lý thông tin sinh viên, lớp học. Trong đó sẽ bao gồm các chức năng Đăng nhập/Đăng ký, thêm-sửa-xóa thông tin sinh viên, lớp học.

Vì các chức năng và giao diện của Quản lý Lớp học và Sinh viên gần như là giống nhau chỉ khác về dữ liệu nên phần này sẽ lấy Quản lý Sinh viên là chính.

* + 1. **Cấu trúc thư mục**

/project-root

│

├── /public

│ ├── index.php # Trang chủ

│ ├── login.php # Trang đăng nhập

│ ├── logout.php # Đăng xuất người dùng

│ ├── register.php # Trang đăng ký người dùng

│

├── /config # Chứa các tệp kết nối và cấu hình

│ └── connect.php # Kết nối cơ sở dữ liệu

│

│

├── /controllers # Xử lý logic

│ ├── xuly\_dangky.php # Xử lý đăng ký tài khoản

│ ├── xuly\_dangnhap.php # Xử lý đăng nhập

│ ├── xuly\_themlop.php # Xử lý thêm lớp

│ ├── xuly\_themsinhvien.php # Xử lý thêm sinh viên

│ ├── xuly\_suasinhvien.php # Xử lý sửa thông tin sinh viên

│ ├── xoa\_lop.php # Xử lý xóa lớp

│ └── xoasinhvien.php # Xử lý xóa sinh viên

│

├── /views # Giao diện người dùng (HTML/PHP hiển thị)

│ ├── chitiet\_lophoc.php # Trang chi tiết lớp học

│ ├── chitiet\_lophoc\_user.php # Trang chi tiết lớp học cho user

│ ├── lophoc.php # Trang danh sách lớp học

│ ├── lophoc\_user.php # Trang danh sách lớp học cho user

│ ├── sinhvien.php # Trang danh sách sinh viên

│ ├── themlop.php # Form thêm lớp

│ ├── themsinhvien.php # Form thêm sinh viên

│ ├── sua\_lop.php # Form sửa lớp

│ └── sua\_sinhvien.php # Form sửa sinh viên

* + 1. **Kết nối CSDL**

<?php

    $servername = "localhost";

    $username = "root";

    $password = "";

    $dbname = "ql\_sinhvien";

    $conn = new *mysqli*($servername, $username, $password, $dbname);

    //todo Kiểm tra kết nối

    if ($conn->connect\_error) {

        die("Connection failed: " . $conn->connect\_error);

    }

?>

Cũng như bài ở Tuần 1 thì cách kết nối CSDL cơ bản là như nhau, chỉ khác tên CSDL cần sử dụng.

* + 1. **SESSION**

Trong bài này xây dựng các chức năng Đăng nhập và Đăng ký nên sẽ tìm hiểu về khái niệm session.

Session trong PHP là một cơ chế được sử dụng để lưu trữ dữ liệu tạm thời trên máy chủ trong suốt thời gian người dùng truy cập website. Nó cho phép bạn duy trì trạng thái người dùng giữa các trang vì HTTP là giao thức stateless (không lưu trạng thái).

**Cách thức hoạt động của Session:**

1. Khi người dùng truy cập vào trang web, PHP sẽ tạo ra một Session ID duy nhất cho người dùng đó.
2. Session ID này được gửi tới trình duyệt của người dùng và thường được lưu trữ dưới dạng cookie.
3. Dữ liệu của session (ví dụ: thông tin đăng nhập, giỏ hàng, v.v.) được lưu trên máy chủ, liên kết với session ID.
4. Khi người dùng di chuyển giữa các trang, session ID được gửi lại cho máy chủ, từ đó máy chủ nhận ra người dùng và phục hồi trạng thái trước đó.

**Lưu ý**: Session phải được bắt đầu ở đầu mỗi tệp PHP bằng lệnh session\_start(). Nếu không, bạn sẽ không thể truy cập vào dữ liệu của session.

* + 1. **Chức năng Đăng ký**

<form *action*="xuly\_dangky.php" *method*="post">

<div>

<label *for*="username">Tên đăng nhập:</label>

<input *type*="text" *id*="username" *placeholder*="Nhập tên đăng nhập" *name*="username" *required*>

</div>

<div>

<label *for*="password">Mật khẩu:</label>

<input *type*="password" *id*="password" *placeholder*="Nhập mật khẩu" *name*="password" *required*>

</div>

<div>

<label *for*="role">Vai trò:</label>

<select *id*="role" *name*="role" *required*>

<option *value*="user">Sinh viên</option>

</select>

</div>

<button *type*="submit">Đăng Ký</button>

<a *href*="login.php"*>* Đã có tài khoản? Đăng nhập</a>

</form>

Xây dựng giao diện Đăng ký cùng với form có phương thức gửi dữ liệu là POST cho phép người dùng nhập vào Tên người dùng và Mật khẩu cùng với đó là vai trò mặc định là ‘user’, với admin thì sẽ được tạo riêng trong CSDL. Đặt ‘*name*’ tương ứng để dễ dàng xử lý dữ liệu được lấy ra.

**Xử lý đăng ký**

if (isset($\_POST['username']) && isset($\_POST['password']) && isset($\_POST['role'])) {

. . . .

} else {

    echo "Vui lòng điền đầy đủ thông tin!";

}

Sử dụng câu lệnh if để đảm bảo lấy được đầy đủ dữ liệu từ form. Nếu lấy được đủ dữ liệu câu lệnh bên trong if sẽ thực thi:

$username = $\_POST['username'];

$password = **password\_hash**($\_POST['password'], PASSWORD\_DEFAULT); // Mã hóa mật khẩu bằng bảng băm.

$role = $\_POST['role'];

Lấy dữ liệu từ input bằng phương thức POST và gán với tên tương ứng.

// Kiểm tra xem tài khoản đã tồn tại chưa

$sql\_check = "SELECT \* FROM users WHERE username = '$username'";

$result\_check = $conn->query($sql\_check);

if ($result\_check->num\_rows > 0) {

// Thiết lập thông báo vào biến phiên và chuyển hướng về trang đăng ký

$\_SESSION['register\_error'] = "Tài khoản đã tồn tại. Vui lòng chọn tên khác!";

header("Location: register.php");

exit(); // Dừng script

} else {

// Thêm tài khoản mới vào cơ sở dữ liệu

$sql\_insert = "INSERT INTO users (username, password, role) VALUES ('$username', '$password', '$role')";

if ($conn->query($sql\_insert) === TRUE) {

// Thiết lập thông báo thành công

$\_SESSION['register\_success'] = "Đăng ký thành công! Vui lòng đăng nhập.";

// Chuyển hướng đến trang login

header("Location: login.php");

exit(); // Dừng script

} else {

echo "Lỗi khi đăng ký: " . $conn->error;

}

Kiểm tra xem tài khoản đã tồn tại chưa bằng cách truy vấn CSDL bảng user với điều kiện là username = $username và gán vào $result\_check.

Thực hiện câu lệnh if với điều kiện ($result\_check->num\_rows > 0) có nghĩa là nếu tồn tại 1 dòng dữ liệu thì tài khoản đó đã có trong CSDL, gán lỗi đăng ký vào biến $\_SESSION và trả về giao diện đăng ký. Ngược lại nếu thỏa điều kiện thì tiến hành thêm dữ liệu vào CSDL và cho phép đến giao diện Đăng nhập.

* + 1. **Chức năng Đăng nhập**

<form *action*="xuly\_dangnhap.php" *method*="post">

<div>

<label *for*="username">Tên đăng nhập:</label>

<input *type*="text" *id*="username" *placeholder*="Nhập tên đăng nhập" *name*="username" *required*>

</div>

<div>

<label *for*="password">Mật khẩu:</label>

<input *type*="password" *id*="password" *placeholder*="Nhập mật khẩu" *name*="password" *required*>

</div>

<button *type*="submit">Đăng Nhập</button>

<a *href*="register.php">Chưa có tài khoản? Đăng ký</a>

</form>

</div>

Xây dựng giao diện chức năng đăng nhập cùng với form cho phép người dùng nhập vào thông tin Tên tài khoản và mật khẩu.

**Xử lý đăng nhập**

**Truy vấn kiểm tra tài khoản**

$sql = "SELECT \* FROM users WHERE username = '$username'";

$result = $conn->query($sql);

Câu lệnh SQL được sử dụng để tìm kiếm người dùng trong bảng users dựa trên username mà người dùng nhập vào.

**$conn->query($sql)**: Thực thi câu truy vấn trên cơ sở dữ liệu và lưu kết quả vào biến $result.

**Kiểm tra xem tài khoản có tồn tại hay không**

if ($result->num\_rows > 0) {

$user = $result->fetch\_assoc();

**$result->num\_rows > 0**: Nếu tìm thấy một tài khoản có tên người dùng trùng khớp, đoạn mã sẽ lấy thông tin người dùng dưới dạng **mảng liên kết** bằng **fetch\_assoc()** và lưu vào biến $user.

**Kiểm tra mật khẩu**

if (password\_verify($password, $user['password'])) {

**password\_verify()**: So sánh mật khẩu người dùng nhập vào (biến $password) với mật khẩu đã mã hóa trong cơ sở dữ liệu ($user['password']). Nếu mật khẩu chính xác, hệ thống sẽ tiếp tục thực hiện các bước tiếp theo.

**Thiết lập Session và chuyển hướng theo vai trò**

$\_SESSION['username'] = $user['username'];

$\_SESSION['role'] = $user['role'];

**Session** là nơi lưu trữ thông tin người dùng trên **máy chủ**.

Dữ liệu được lưu bao gồm: Tên người dùng và vai trò (role).

**Phân loại vai trò và chuyển hướng:**

if ($user['role'] == 'admin') {

header("Location: lophoc.php");

} else {

header("Location: lophoc\_user.php");

}

* Nếu vai trò là "admin", người dùng sẽ được chuyển hướng đến trang quản lý lớp học (lophoc.php).
* Nếu vai trò là "user", người dùng sẽ được chuyển đến trang chỉ có quyền xem thông tin lớp học (lophoc\_user.php).
  + 1. **Chức năng đăng xuất**

<?php

session\_start();

// Xóa tất cả các biến phiên

$\_SESSION = [];

// Hủy phiên

session\_destroy();

// Chuyển hướng đến trang đăng nhập

header("Location: login.php");

exit();

?>

**session\_start():** Bắt đầu hoặc tiếp tục một phiên làm việc (session) hiện tại. Lệnh này cần thiết để truy cập các biến phiên đã được lưu trữ.

Gán biến **$\_SESSION** thành một mảng rỗng, nó giúp xóa toàn bộ dữ liệu đã được lưu trong phiên (như tên người dùng, vai trò, thông báo lỗi, v.v.). Tuy nhiên, phiên vẫn còn tồn tại trên máy chủ cho đến khi được hủy hoàn toàn.

**session\_destroy()**: Hủy phiên làm việc hiện tại, giải phóng tài nguyên trên máy chủ. Sau khi gọi hàm này, tất cả dữ liệu của phiên sẽ bị xóa vĩnh viễn.

**header("Location: login.php")**: Chuyển hướng người dùng đến trang đăng nhập sau khi đăng xuất thành công.

**exit()**: Dừng thực thi script ngay lập tức để đảm bảo không có đoạn mã nào khác chạy sau khi chuyển hướng.

**Lưu ý:** Bạn cần sử dụng một câu lệnh sau trong tất cả các trang để đảm bảo việc người dùng chưa đăng nhập, khác vai trò sẽ không vào được các trang đó.

<?php

session\_start(); // Bắt đầu session

// Kiểm tra nếu người dùng chưa đăng nhập

if (***!isset***($\_SESSION['username'])) {

    // Nếu chưa đăng nhập, chuyển hướng về trang login

    header("Location: login.php");

    exit(); // Dừng script

}

if ($\_SESSION['role'] **!==** '***admin*** hoặc ***user*** theo bạn phân quyền của trang') {

  echo "Bạn không có quyền truy cập trang này.";

  exit();

}

?>

* + 1. **Chức năng Thêm**

<form action="xuly\_themsinhvien.php" method="post">

<div>

<label for="masv">Mã sinh viên:</label>

<input type="text" id="masv" placeholder="Nhập mã sinh viên" name="txtMaSV" required>

</div>

<div>

<label for="holot">Họ lót:</label>

<input type="text" id="holot" placeholder="Nhập họ lót" name="txtHoLot" required>

</div>

<div>

<label for="tensv">Tên sinh viên:</label>

<input type="text" id="tensv" placeholder="Nhập tên sinh viên" name="txtTenSV" required>

</div>

<div>

<label for="ngaysinh">Ngày sinh:</label>

<input type="date" id="ngaysinh" name="txtNgaySinh" required max="2006-12-31">

</div>

<div>

<label for="gioitinh">Giới tính:</label>

<select id="gioitinh" name="txtGioiTinh" required>

<option value="Nam">Nam</option>

<option value="Nữ">Nữ</option>

</select>

</div>

<div>

<label for="malop">Lớp học:</label>

<select id="malop" name="txtMaLop" required>

**<?php**

**$sql = "SELECT maLop, tenLop FROM lophoc";**

**$result = $conn->query($sql);**

**while ($row = $result->fetch\_assoc()) {**

**echo "<option value='" . $row['maLop'] . "'>" . $row['maLop'] . " - " . $row['tenLop'] . "</option>";**

**}**

**?>**

</select>

</div>

<div>

<button type="submit">Thêm</button>

<a href="sinhvien.php">Quay lại</a>

</div>

</form>

Sử dụng form để xây dựng lên giao diện cho phép người dùng nhập vào các thông tin của Sinh viên cần thêm vào CSDL, đặc biệt với thông tin Lớp học sẽ cho chọn các lớp đã có sẵn trong CSDL bằng cách thực hiện câu truy vấn với bảng lophoc sau đó dùng vòng lặp while và hàm fetch\_assoc để duyệt qua từng dòng các dữ liệu. In ra các tùy chọn mà dữ liệu đã duyệt qua kết hợp với nối chuỗi ‘.’ .

**Xử lý chức năng thêm**

if (isset($\_POST['them\_sv'])) {

$maSV = $\_POST['txtMaSV'];

$hoLot = $\_POST['txtHoLot'];

$tenSV = $\_POST['txtTenSV'];

$ngaySinh = $\_POST['txtNgaySinh'];

$gioiTinh = $\_POST['txtGioiTinh'];

$maLop = $\_POST['txtMaLop'];

Lưu dữ liệu người dùng nhập từ form vào các biến tương ứng trong PHP để sử dụng trong truy vấn SQL.

$sql = "INSERT INTO sinhvien (maSV, hoLot, tenSV, ngaySinh, gioiTinh, maLop)

VALUES ('$maSV', '$hoLot', '$tenSV', '$ngaySinh', '$gioiTinh', '$maLop')";

Câu lệnh SQL dùng để thêm một hàng dữ liệu mới vào bảng sinhvien với các dữ liệu tương ứng được biến lưu trữ.

if ($conn->query($sql) === TRUE) {

echo "<script>alert('Thêm sinh viên thành công!'); window.location='sinhvien.php';</script>";

} else {

echo "Lỗi khi thêm sinh viên: " . $conn->error;

}

**$conn->query($sql)**: Thực thi câu lệnh SQL trên cơ sở dữ liệu.

Nếu câu lệnh thành công: Hiển thị thông báo thành công bằng JavaScript alert và chuyển hướng người dùng đến trang sinhvien.php.

Nếu câu lệnh thất bại: In ra thông báo lỗi, kèm theo chi tiết lỗi từ cơ sở dữ liệu ($conn->error).

* + 1. **Chức năng sửa**

Với chức năng sửa thì giao diện thì vẫn giống như khi Thêm vào, chỉ khác ở chỗ dữ liệu nào bạn không cho phép sửa hay nhập vào thì sẽ thay đổi từ khóa trong thẻ <input> thành *readonly(chỉ cho phép đọc)* thay vì *required.*

Cùng với đó là khi xử lý sửa cũng giống như khi xử lý Thêm dữ liệu vào, chỉ khác ở câu lệnh SQL cụ thể như sau:

$sql = "UPDATE sinhvien SET hoLot = '$hoLot', tenSV = '$tenSV', ngaySinh = '$ngaySinh', gioiTinh = '$gioiTinh', maLop = '$maLop' WHERE maSV = '$maSV'";

* + 1. **Chức năng xóa**

<?php

include\_once("connect.php");

if (isset($\_GET['maSV'])) {

$maSV = $\_GET['maSV'];

// Thực hiện truy vấn xóa

$sql = "DELETE FROM sinhvien WHERE maSV = '$maSV'";

if ($conn->query($sql) === TRUE) {

echo "<script>alert('Xóa sinh viên thành công!'); window.location='sinhvien.php';</script>";

} else {

echo "Lỗi khi xóa sinh viên: " . $conn->error;

}

}

$conn->close();

?>

**Lấy mã sinh viên được truyền qua URL**

if (isset($\_GET['maSV'])) {

$maSV = $\_GET['maSV'];

isset($\_GET['maSV']): Kiểm tra xem tham số maSV có tồn tại trong URL hay không. Sau đó lấy giá trị maSV từ tham số GET trong URL và lưu vào biến $maSV.

**Thực hiện truy vấn xóa**

$sql = "DELETE FROM sinhvien WHERE maSV = '$maSV'";

if ($conn->query($sql) === TRUE) {

echo "<script>alert('Xóa sinh viên thành công!'); window.location='sinhvien.php';</script>";

} else {

echo "Lỗi khi xóa sinh viên: " . $conn->error;

}

Câu lệnh SQL: DELETE Xóa bản ghi của sinh viên với điều kiện maSV bằng với giá trị được truyền vào ($maSV).

$conn->query($sql): Thực thi câu lệnh SQL.

Nếu thành công, hiển thị thông báo bằng JavaScript alert và chuyển hướng người dùng về trang sinhvien.php. Nếu thất bại, in ra thông báo lỗi kèm theo thông tin lỗi từ cơ sở dữ liệu ($conn->error).

* + 1. **Giao diện Quản lý Sinh viên**

<?php

$sql = "SELECT sinhvien.\*, lophoc.tenLop FROM sinhvien INNER JOIN lophoc ON sinhvien.maLop = lophoc.maLop";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

echo "<table>";

echo "<thead><tr><th>Mã SV</th><th>Họ Lót</th><th>Tên SV</th><th>Ngày Sinh</th><th>Giới Tính</th><th>Lớp</th><th>Sửa</th><th>Xóa</th></tr></thead>";

echo "<tbody>";

while ($row = $result->fetch\_assoc()) {

echo "<tr>";

echo "<td>" . $row["maSV"] . "</td>";

echo "<td>" . $row["hoLot"] . "</td>";

echo "<td>" . $row["tenSV"] . "</td>";

echo "<td>" . $row["ngaySinh"] . "</td>";

echo "<td>" . $row["gioiTinh"] . "</td>";

echo "<td>" . $row["tenLop"] . "</td>";

echo "<td><a href='suasinhvien.php?maSV=" . $row["maSV"] . "' title='Sửa'></a></td>";

echo "<td><a onclick=\"return confirm('Có thực sự muốn xóa không?')\" href='xoasinhvien.php?maSV=" . $row["maSV"] . "' title='Xóa'></a></td>";

echo "</tr>";

}

echo "</tbody></table>";

} else {

echo "<p>Không có sinh viên nào trong danh sách.</p>";

}

$conn->close();

?>

$sql = "SELECT sinhvien.\*, lophoc.tenLop FROM sinhvien

INNER JOIN lophoc ON sinhvien.maLop = lophoc.maLop";

$result = $conn->query($sql);

Thực hiện truy vấn lấy toàn bộ thông tin sinh viên cùng với tên lớp từ bảng lophoc và lưu vào biến $result.

while ($row = $result->fetch\_assoc()) {

echo "<tr>";

echo "<td>" . $row["maSV"] . "</td>";

echo "<td>" . $row["hoLot"] . "</td>";

echo "<td>" . $row["tenSV"] . "</td>";

echo "<td>" . $row["ngaySinh"] . "</td>";

echo "<td>" . $row["gioiTinh"] . "</td>";

echo "<td>" . $row["tenLop"] . "</td>";

Sử dụng vòng lặp while và hàm fetch\_assoc để thêm lần lượt và in ra các dòng dữ liệu vào cột.

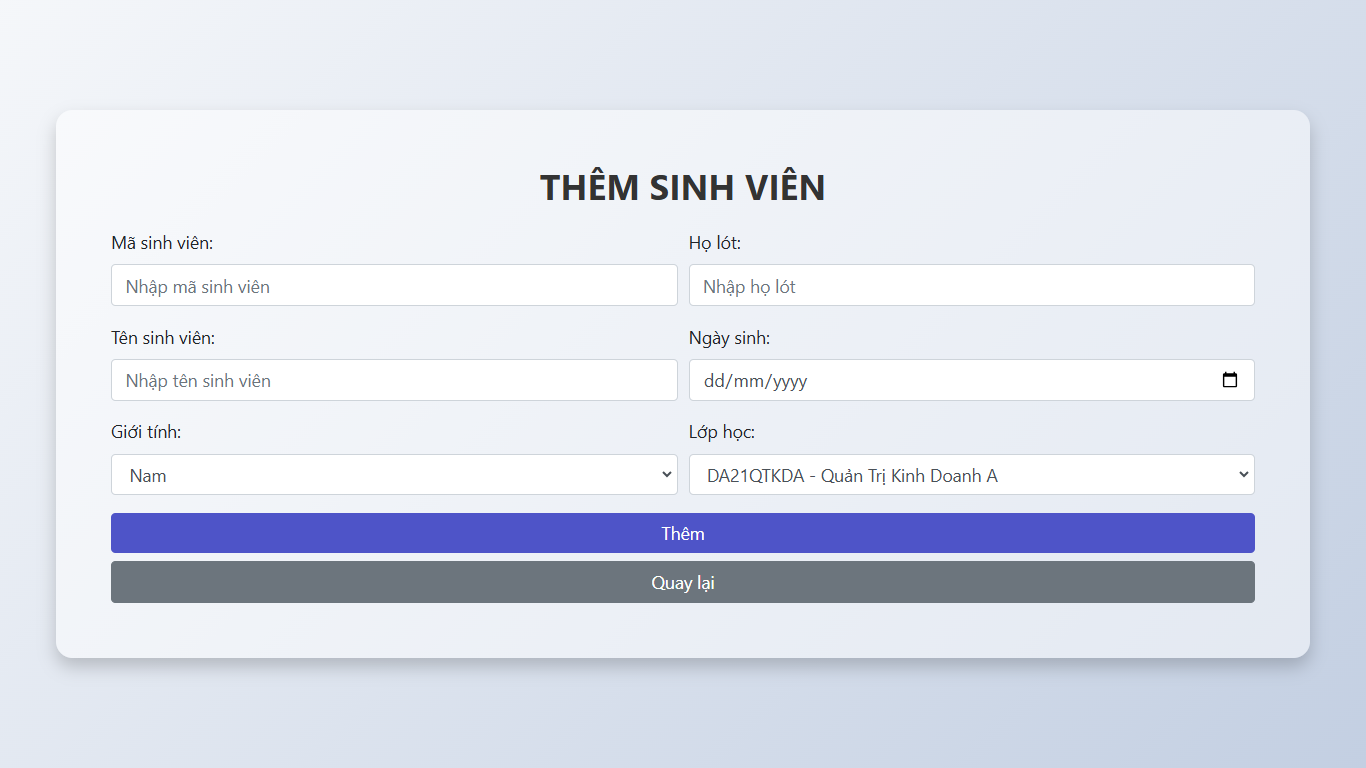
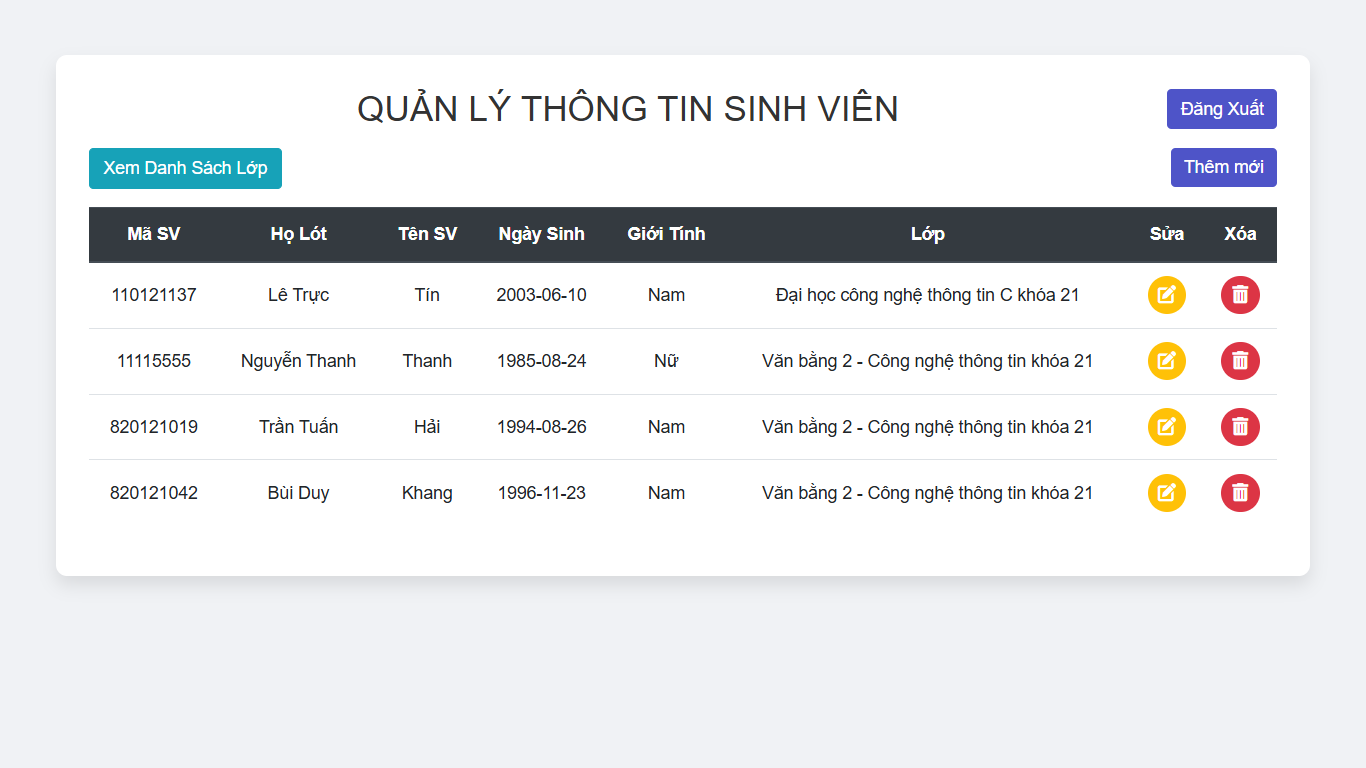
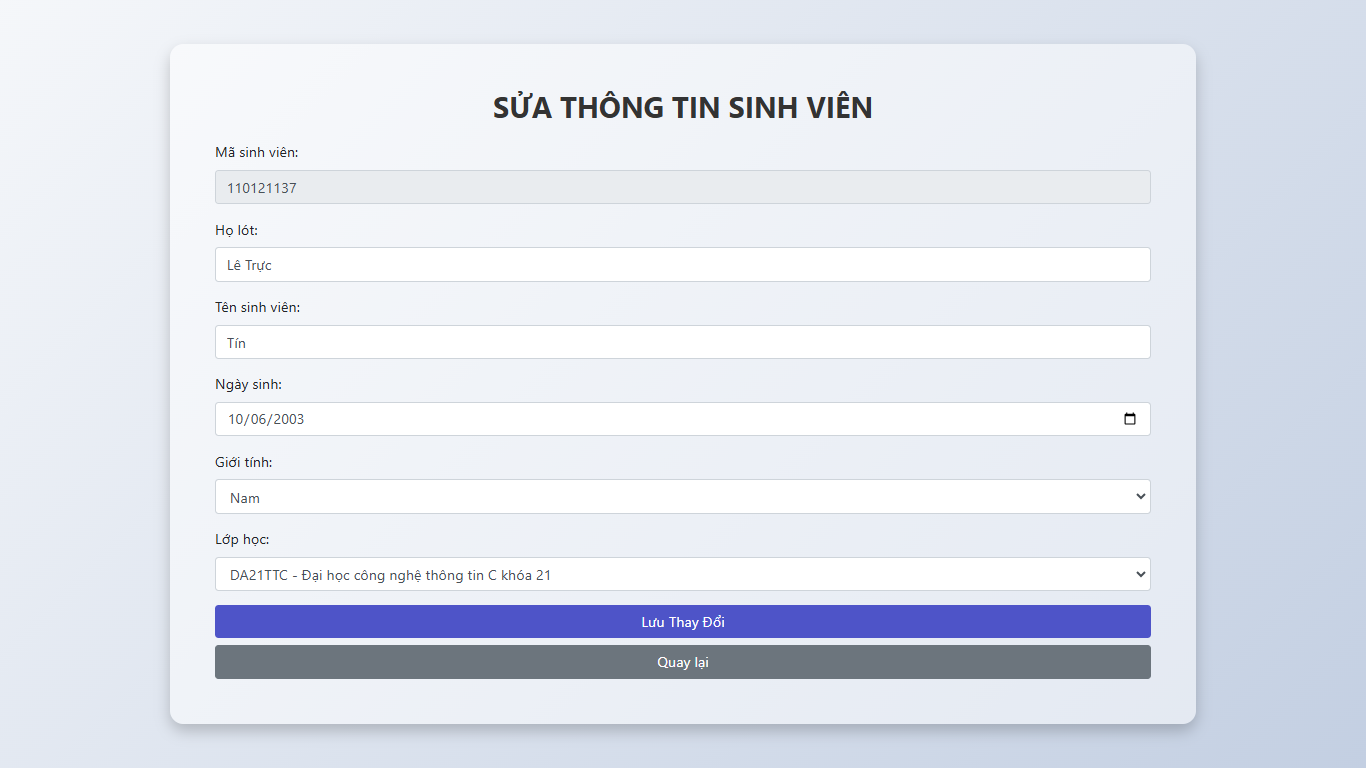
echo "<td><a href='suasinhvien.php?maSV=" . $row["maSV"] . "' title='Sửa'></a></td>";

echo "<td><a onclick=\"return confirm('Có thực sự muốn xóa không?')\" href='xoasinhvien.php?maSV=" . $row["maSV"] . "' title='Xóa'></a></td>";

echo "</tr>";

Nút liên kết đến trang Sửa hoặc Xóa, trong href được truyền cùng với dữ liệu maSV để phục vụ trong việc xử lý chức năng Xóa và Sửa.

* + 1. **Kết quả dạt được**

## 3.3 TUẦN 3

Nội dung thực hành trong tuần này là nghiên cứu về lập trình hướng đối tượng trong PHP, cụ thể hơn trong phần này sẽ sử dụng Giao diện chức năng (Interface) để xử lý các phương thức mà lớp (Class) đó phải triển khai.

**Interface là gì?**

Trong PHP cụ thể hơn là OOP, interface là một cấu trúc cho phép định nghĩa các phương thức mà lớp phải triển khai, nhưng không cung cấp phần cài đặt cho các phương thức đó. Interface hoạt động như một bản thiết kế cho các lớp thực thi, đảm bảo rằng các lớp có cùng interface sẽ có cùng các phương thức với tên và chữ ký như nhau.

Cú pháp:

<?php

**interface** InterfaceName { //định nghĩa một Interface bằng từ khóa ‘interface’

public function someMethod1();

public function someMethod2($name, $color);

public function someMethod3() : string;

}

?>

Sử dụng interface: Các lớp có thể "cài đặt" một interface bằng cách sử dụng từ khóa implements. Lớp cài đặt interface phải thực hiện tất cả các phương thức được định nghĩa trong interface.

class Class implements InterfaceName {

public function someMethod1($data) {

// Thực thi hàm

}

public function someMethod2($id, $data) {

// Thực thi hàm

}

public function someMethod3($id) {

// Thực thi hàm

}

}

Khi nhiều lớp **implements** cùng một interface, chúng buộc phải cung cấp các phương thức với cùng chữ ký (tên và tham số giống nhau). Điều này đảm bảo rằng bạn có thể tương tác với các đối tượng khác nhau bằng cùng một cách thức.

* + 1. **Cấu trúc thư mục**

/ql\_school

│

├── /models

│ ├── Manageable.php

│ ├── Student.php

│ ├── Classroom.php

│

├── /controllers

│ ├── Manager.php

│

├── /views

│ ├── sinhvien.php

│ ├── lophoc.php

│ ├── form\_sinhvien.php

│ ├── form\_lophoc.php

│

├── db.php

├── index.php

* + 1. **Mối liên kết giữa Manager, Manageable, và Student**

***Manageable.php***

<?php

interface Manageable {

    public function add($data);

    public function edit($id, $data);

    public function delete($id);

}

?>

Manageable là một interface định nghĩa một tập hợp các phương thức (add, edit, delete). Mục đích của interface này là để quy định rằng bất kỳ lớp nào cài đặt nó đều phải cung cấp cách thức thực thi cho các phương thức này.

***Student.php***

class Student implements *Manageable* {

    public function add($data) {

#hàm thực thi việc thêm

    }

    public function edit($id, $data) {

        #hàm thực thi việc sửa

    }

    public function delete($id) {

#hàm thực thi việc xóa

    }

}

?>

Các lớp (như Student trong ví dụ) muốn sử dụng các phương thức được định nghĩa trong Manageable phải cài đặt interface này bằng từ khóa **implements**. Khi một lớp cài đặt interface, nó phải định nghĩa (cung cấp thực thi) cho tất cả các phương thức được chỉ định trong interface.

***Manager.php***

<?php

class Manager {

    private $model;

    public function \_\_construct(Manageable $model) {

        $this->model = $model;

    }

    public function add($data) {

        $this->model->add($data);

    }

    public function edit($id, $data) {

        $this->model->edit($id, $data);

    }

    public function delete($id) {

        $this->model->delete($id);

    }

}

?>

Manager là một lớp điều phối các hoạt động quản lý cho các đối tượng mà nó nhận được. Nó nhận một đối tượng cài đặt interface Manageable trong **constructor** của nó. Điều này cho phép Manager làm việc với bất kỳ loại đối tượng nào mà cài đặt interface này, không chỉ riêng Student.

Nói cách khác thì Manager sẽ dựa vào $model sẽ nhận biết được đối tượng nào sẽ được khai báo khi sử dụng nó và gọi hàm tương ứng của đối tượng đó. Như trong ví dụ sau:  
$studentManager = new Manager(new Student());

Đoạn code này thiết lập một mối quan hệ giữa lớp quản lý (Manager) và lớp mô hình sinh viên (Student). Manager sẽ sử dụng đối tượng Student để thực hiện các thao tác quản lý sinh viên.

* + 1. **Chức năng thêm-sửa (Sinh viên)**

Trong phần này chức năng thêm và sữa sẽ được kết hợp chung trong 1 file. Tại sao lại làm như vậy?

***Giảm trùng lặp mã***

Vì phần lớn thêm và sửa dữ liệu có quy trình tương tự: Nhập thông tin vào form 🡪 Kiểm tra và xử lý dữ liệu đầu vào 🡪 Thực hiện thao tác với CSDL (thêm hoặc cập nhật).

***Đơn giản hóa luồng điều hướng***

Thay vì điều hướng người dùng qua nhiều file (một file cho thêm, một file cho sửa), bạn chỉ cần kiểm tra có đang chỉnh sửa hay không thông qua tham số (như $\_GET['id']) để quyết định hành động.

**Xử lý chức năng thêm-sửa**

**1. Khởi tạo Manager với model Student**

$studentManager = new Manager(new Student());

Ở đây, một đối tượng Manager được tạo ra và được truyền vào một đối tượng của Student. Manager đóng vai trò như bộ điều phối (controller) để gọi các phương thức add, edit, và delete từ lớp Student dựa trên hành động cần thực hiện.

**2. Kiểm tra xem có đang ở chế độ chỉnh sửa (editing) không**

$editing = isset($\_GET['id']);

$error = '';

**$editing** kiểm tra xem có id trong URL không.

Nếu có id trong query string (ví dụ: sinhvien.php?id=123), thì đây là chế độ sửa. Nếu không có, thì đây là thêm mới.

**3. Lấy dữ liệu sinh viên nếu đang chỉnh sửa**

if ($editing) {

$maSV = $\_GET['id'];

$stmt = $conn->prepare("SELECT \* FROM sinhvien WHERE maSV = ?");

$stmt->execute([$maSV]);

$sinhvien = $stmt->fetch(PDO::FETCH\_ASSOC);

}

Nếu ở **chế độ sửa**, mã sinh viên (maSV) sẽ được lấy từ URL qua id.

Sử dụng truy vấn chuẩn bị (prepare) để lấy thông tin sinh viên từ cơ sở dữ liệu.

Kết quả được lưu vào $sinhvien để hiển thị lại trong form.

**4. Xử lý khi người dùng bấm nút Lưu**

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$maSV = $\_POST['maSV'];

$hoLot = $\_POST['hoLot'];

$tenSV = $\_POST['tenSV'];

$ngaySinh = $\_POST['ngaySinh'];

$gioiTinh = $\_POST['gioiTinh'];

$maLop = $\_POST['maLop'];

Nếu phương thức của request là **POST**, có nghĩa là người dùng đã bấm nút Lưu trong form. Các dữ liệu được lấy từ form thông qua $\_POST và gán vào các biến tương ứng.

**5. Thêm hoặc cập nhật dữ liệu sinh viên**

if (empty($error)) {

$data = [

'maSV' => $maSV,

'hoLot' => $hoLot,

'tenSV' => $tenSV,

'ngaySinh' => $ngaySinh,

'gioiTinh' => $gioiTinh,

'maLop' => $maLop

];

if ($editing) {

$studentManager->edit($maSV, $data);

} else {

$studentManager->add($data);

}

header("Location: sinhvien.php");

exit;

}

Nếu không có lỗi khi kiểm tra dữ liệu đã tồn tại, dữ liệu từ form được gán vào một mảng **$data**.

Tùy thuộc vào trạng thái ($editing), **Manager** sẽ gọi phương thức **edit()** hoặc **add()** để sửa hoặc thêm mới sinh viên.

* + 1. **Chức năng xóa (Sinh viên)**

$studentManager = new *Manager*(new *Student*());

$maSV = $\_GET['id'];  // Lấy mã sinh viên từ URL

$studentManager->delete($maSV);  // Gọi hàm xóa

Khởi tạo đối tượng Manager với model Student.

Lấy mã sinh viên từ URL và gán vào $maSV

Gọi phương thức xóa sinh viên: Phương thức **delete($maSV)** từ lớp **Manager** được gọi, và lớp Manager sẽ tiếp tục gọi **delete()** từ đối tượng **Student** 🡪 Bên trong Student, mã SQL sẽ được thực thi để xóa bản ghi có **maSV** khớp với giá trị truyền vào.