

TRƯỜNG ĐẠI HỌC ĐỒNG THÁP
KHOA CÔNG NGHỆ VÀ KỸ THUẬT



ThS. TRẦN KIM HƯƠNG (Chủ biên)

BÀI GIẢNG
LẬP TRÌNH WEB PHP

LƯU HÀNH NỘI BỘ

Đồng Tháp, tháng 05 năm 2025

LỜI NÓI ĐẦU

Bài giảng Lập trình Web với PHP được biên soạn nhằm cung cấp cho sinh viên chuyên ngành Công nghệ Thông tin những kiến thức nền tảng về lập trình web PHP, giúp sinh viên có thể tự học, nghiên cứu và ứng dụng vào thực tế.

Bài giảng này trang bị cho sinh viên các kỹ năng xây dựng trang web động, kết hợp giữa HTML, CSS, JavaScript, PHP và MySQL để tạo nên các ứng dụng web hoàn chỉnh. Bên cạnh đó, sinh viên còn được học về cách tương tác với cơ sở dữ liệu, xử lý biểu mẫu, quản lý phiên làm việc (Session, Cookie), truy vấn dữ liệu bằng PHP Data Object (PDO) và các kỹ thuật lập trình web khác.

Nội dung bài giảng gồm 5 chương:

- Chương 1: Kiến thức cơ bản về Internet và Web – trình bày tổng quan về Internet, mô hình Client-Server, giao thức HTTP/HTTPS, và giới thiệu các ngôn ngữ lập trình web.

- Chương 2: Ngôn ngữ HTML và CSS – cung cấp kiến thức về các thành phần cơ bản của một trang web, các thẻ HTML quan trọng, bảng (Table), biểu mẫu (Form) và HTML5. Hướng dẫn cách sử dụng CSS để định dạng trang web, cách áp dụng các bộ chọn (Selector) và các thuộc tính CSS thông dụng.

- Chương 3: Ngôn ngữ JavaScript – trình bày các khái niệm về biến, kiểu dữ liệu, hàm, cấu trúc lệnh, xử lý DOM, sự kiện và kết hợp JavaScript với PHP.

- Chương 4: Lập trình Web với PHP – giới thiệu PHP, cú pháp cơ bản, cách nhúng PHP vào HTML, xử lý biến, toán tử, cấu trúc điều khiển, hàm và các kỹ thuật lập trình PHP quan trọng.

- Chương 5: Lập trình PHP với Cơ sở dữ liệu quan hệ MySQL – trình bày các khái niệm về hệ quản trị cơ sở dữ liệu quan hệ, cách sử dụng MySQL, và các câu lệnh thao tác dữ liệu như SELECT, INSERT, UPDATE, DELETE. Hướng dẫn cách kết nối PHP với MySQL, thực thi truy vấn, xử lý lỗi, làm việc với biểu mẫu, Cookie, Session, và xây dựng ứng dụng web hoàn chỉnh.

Bài giảng này được biên soạn theo hướng học tập chủ động, giúp sinh viên có thể tự nghiên cứu và thực hành. Mỗi chương đều có ví dụ minh họa, bài tập thực hành và câu hỏi ôn tập để sinh viên củng cố kiến thức.

Trong quá trình biên soạn, mặc dù đã cố gắng hoàn thiện, nhưng bài giảng không thể tránh khỏi những thiếu sót. Rất mong nhận được sự góp ý từ quý Thầy/Cô, đồng nghiệp và sinh viên để tài liệu ngày càng hoàn thiện hơn.

Trân trọng cảm ơn!
Tác giả

Trần Kim Hương

MỤC LỤC

---o0o---

Chương 1	11
Kiến thức cơ bản về Internet và Web	11
Mục tiêu	11
1.1. Tổng quan về Internet	11
1.1.1. Các khái niệm cơ bản.....	11
1.1.2. Mô hình Client – Server	12
1.1.3. Giao thức HTTP và HTTPS.....	13
1.1.4. Cấu trúc URL và DNS	13
1.2. Website	13
1.2.1. World Wide Web (WWW)	13
1.2.2. Lịch sử phát triển Web	14
1.2.3. Phân loại Website.....	15
1.2.3.1. Web tĩnh.....	15
1.2.3.2. Web động.....	16
1.3. Giới thiệu ngôn ngữ lập trình Web.....	16
1.3.1. Tổng quan	16
1.3.2. Ngôn ngữ thiết kế	16
1.3.3. Ngôn ngữ lập trình.....	16
Câu hỏi	17
Chương 2	19
Ngôn ngữ HTML và CSS	19
Mục tiêu	19
2.1. Ngôn ngữ HTML	19
2.1.1. Giới thiệu HTML.....	19
2.1.2. Các thành phần trong trang HTML	19
2.1.3. Các thẻ (tag) cơ bản	21
2.1.3.1. Thẻ xử lý văn bản.....	21
2.1.3.2. Thẻ xử lý hình ảnh.....	22
2.1.3.3. Thẻ liên kết (<a>).....	22
2.1.3.4. Thẻ phân chia bố cục (<div>)	23
2.1.3.5. Thẻ Table	23

2.1.3.6. Thẻ Form	24
2.1.4. Các ký tự đặc biệt thường dùng.....	26
2.1.5. HTML5	27
2.2. Ngôn ngữ CSS.....	29
2.2.1. Giới thiệu CSS	29
2.2.2. Cú pháp và cách dùng CSS.....	30
2.2.2.1. Cú pháp CSS	30
2.2.2.2. Cách sử dụng CSS.....	30
2.2.3. CSS selector	31
2.2.3.1. Selector cơ bản	31
2.2.3.2. Selector nâng cao.....	31
2.2.4. Thuộc tính CSS thông dụng.....	31
2.2.4.1. Thuộc tính định dạng văn bản	31
2.2.4.2. Thuộc tính bố cục và hiển thị	32
2.2.4.3. Thuộc tính màu sắc và nền	32
2.2.4.4. Thuộc tính định dạng danh sách.....	32
2.2.4.5. Thuộc tính xử lý hiển thị linh hoạt (Flexbox)	33
Câu hỏi, bài tập	33
Câu hỏi	33
Chương 3	40
Ngôn ngữ JavaScript	40
Mục tiêu.....	40
3.1. Giới thiệu JavaScript.....	40
3.1.1 JavaScript là gì?	40
3.1.2. Vai trò và ứng dụng của JavaScript	41
3.1.3. Sử dụng JavaScript trong HTML và PHP	42
3.1.3.1. Sử dụng JavaScript trong HTML	42
3.1.3.2. Sử dụng JavaScript trong PHP	43
3.1.4. Viết chương trình JavaScript đầu tiên.....	45
3.2. Biến, kiểu dữ liệu và phép toán trong JavaScript	46
3.2.1. Biến.....	47
3.2.2. Kiểu dữ liệu	47
3.2.3. Phép toán.....	48
3.3. Hàm trong JavaScript.....	50

3.4. Cấu trúc lệnh trong JavaScript	52
3.4.1. Cấu trúc điều kiện	52
a) Cấu trúc IF...ELSE	52
b) Cấu trúc IF...ELSE IF... ELSE.....	52
c) Cấu trúc SWITCH.....	53
3.4.2. Cấu trúc lặp.....	54
a) Vòng lặp FOR	54
b) Vòng lặp WHILE	54
b) Vòng lặp DO...WHILE	54
3.4.3. Câu lệnh điều khiển	55
a) Câu lệnh BREAK.....	55
b) Câu lệnh CONTINUE.....	55
3.5. DOM (Document Object Model).....	55
3.5.1. Khái niệm DOM	55
3.5.2. Thao tác với DOM	55
3.5.3. Xử lý sự kiện trong DOM.....	56
3.6. Đối tượng trong JavaScript	56
3.6.1. Khái niệm đối tượng.....	56
3.6.2. Tạo đối tượng.....	57
3.6.3. Truy cập và thay đổi đối tượng.....	57
3.7. Kết hợp JavaScript với PHP.....	57
3.8. Xác thực dữ liệu phía Client	58
3.8.1. Kiểm tra trường dữ liệu	58
3.8.2. Kiểm tra định dạng email	59
Câu hỏi, bài tập	59
Chương 4	62
Lập trình Web với PHP.....	62
Mục tiêu	62
4.1. Giới thiệu về PHP	62
4.1.1. PHP là gì?	62
4.1.2. Lịch sử phát triển PHP.....	62
4.1.3. Cách xử lý trang web động với PHP	63
4.1.4. Các phần mềm ứng dụng Web.....	64
4.2. Cài đặt môi trường và công cụ	65

4.2.1. Cài đặt Visual Studio Code	65
4.2.2. Cài đặt XAMPP Server	65
4.3. Kỹ năng PHP cơ bản	67
4.3.1. Cú pháp PHP	67
4.3.2. Nhúng mã PHP vào mã HTML	69
4.3.3. Tạo trang web PHP đầu tiên	70
4.4. Biến, phép toán, kiểu dữ liệu	71
4.4.1. Biến trong PHP	71
4.4.2. Hằng số trong PHP	73
4.4.3. Kiểu dữ liệu trong PHP	73
4.4.4. Toán tử và Biểu thức trong PHP	74
4.5. Cấu trúc điều khiển	76
4.5.1. Cấu trúc điều kiện trong PHP	76
5.5.1.1 Cấu trúc IF	76
4.5.1.2. Cấu trúc IF...ELSE	77
4.5.1.3. Cấu trúc IF...ELSEIF...ELSE	78
4.5.1.4. Cấu trúc SWITCH...CASE	78
4.5.2. Cấu trúc vòng lặp trong PHP	79
4.5.2.1. Vòng lặp FOR	79
4.5.2.2. Vòng lặp WHILE	80
4.5.2.3. Vòng lặp DO...WHILE	81
4.5.2.4. Vòng lặp FOREACH	81
4.5.3. Lệnh BREAK – CONTINUE – DIE() – EXIT()	83
5.5.3.1. Lệnh BREAK	83
4.5.3.2. Lệnh CONTINUE	83
4.5.3.3. Lệnh DIE()	84
4.5.3.4. Lệnh EXIT()	84
4.5.4. Lệnh INCLUDE – INCLUDE_ONCE – REQUIRE – REQUIRE_ONCE	84
4.5.4.1. Lệnh INCLUDE	84
4.5.4.2. Lệnh INCLUDE_ONCE	85
4.5.4.3. Lệnh REQUIRE	85
4.5.4.2. Lệnh REQUIRE_ONCE	85
4.5.5. Toán tử 3 ngôi và các cú pháp thay thế	86
4.5.5.1. Toán tử 3 ngôi	86

4.5.5.2. Cú pháp thay thế IF	86
4.5.5.3. Cú pháp thay thế FOR	87
4.5.5.4. Cú pháp thay thế WHILE	87
4.5.5.5. Cú pháp thay thế FOREACH	87
4.6. Hàm trong PHP	88
4.6.1. Khai báo và gọi hàm	88
4.6.2. Tham số và giá trị trả về	89
4.6.4. Hàm dựng sẵn (build-in) trong PHP	90
5.6.4.1. Hàm xử lý chuỗi	90
4.6.4.2. Hàm xử lý mảng	93
4.6.4.3. Hàm xử lý Số (Number)	101
4.6.4.4. Hàm xử lý ngày giờ (DateTime)	103
4.6.4.5. Hàm ISSET và EMPTY	109
4.6.5. Hàm ẩn danh (Anonymous Function) và Closure	111
4.6.5.1. Hàm ẩn danh	111
4.6.5.2. Closure	112
Câu hỏi, bài tập	112
Chương 5	118
Lập trình Web PHP với Cơ sở dữ liệu MySQL	118
Mục tiêu	118
5.1. Giới thiệu về hệ quản trị cơ sở dữ liệu quan hệ	118
5.2. Các câu lệnh thao tác dữ liệu trong SQL	119
5.2.1. SELECT	120
5.2.2. INSERT, UPDATE, DELETE	121
5.2.2.1. INSERT	121
5.2.2.2. UPDATE	121
5.2.2.3. DELETE	122
5.3. Giới thiệu MySQL	122
5.3.1. Khái Niệm MySQL	122
5.3.2. Lợi Thế Của MySQL	122
5.3.3. Chức Năng Của MySQL	122
5.3.4. Tổ Chức Lưu Trữ Của MySQL	123
5.3.5. Quản lý cơ sở dữ liệu với phpMyAdmin	123
5.3.5.1. Khởi động phpMyAdmin	123

5.3.5.2. Cấu hình phpMyAdmin và Quản lý bảo mật.....	124
5.3.5.3. Hướng dẫn nhập và chạy mã SQL cho việc tạo cơ sở dữ liệu	127
5.3.5.4. Hướng dẫn xem dữ liệu và cấu trúc bảng.....	132
5.3.5.6. Hướng dẫn tạo tài khoản người dùng với quyền hạn chế	133
5.4. Kết nối PHP với cơ sở dữ liệu MySQL	136
5.4.1. Cách kết nối PHP với MySQL sử dụng PDO	136
5.4.2. Thực thi câu lệnh SELECT	137
5.4.3. Thực thi câu lệnh INSERT, UPDATE, DELETE.....	138
5.4.3.1. Thực thi câu lệnh INSERT.....	138
5.4.3.2. Thực thi câu lệnh UPDATE	139
5.4.3.3. Thực thi câu lệnh DELETE	139
5.5. Xử lý ngoại lệ với Try/ Catch	140
5.6. Truy vấn dữ liệu bằng PDO	142
5.7. Làm việc với FORM	145
5.7.1. Nhận dữ liệu từ Form.....	145
5.7.2. Hiển thị dữ liệu trên trang web	146
5.8. Phương thức GET/ POST trong PHP	149
5.8.1. Phương thức GET trong PHP	149
5.8.2. Phương thức POST trong PHP	149
5.9. Làm việc với Cookie và Session.....	151
5.9.1. Cookie	151
5.9.2. Session	152
Câu hỏi, bài tập	154
Bài tập tự thực hành	161
TÀI LIỆU THAM KHẢO	162

DANH MỤC HÌNH

Hình 4. 1 Mở file php trong VS Code	66
Hình 4. 2 Giao diện trang web php đầu tiên.....	71
Hình 5. 1 Khởi động server web Apache	124
Hình 5. 2 Khởi động phpMyAdmin	124
Hình 5. 3 Chỉnh sửa quyền người dùng root	126
Hình 5. 4 Cập nhật mật khẩu cho tài khoản	126
Hình 5. 5 Tạo cơ sở dữ liệu sử dụng giao diện phpMyAdmin.....	128
Hình 5. 6 Tạo cơ sở dữ liệu sử dụng mã SQL	129
Hình 5. 7 Tạo bảng sử dụng giao diện phpMyAdmin.....	130
Hình 5. 8 Tạo bảng sử dụng mã SQL	130
Hình 5. 9 Nhập dữ liệu cho bảng sử dụng giao diện phpMyAdmin	131
Hình 5. 10 Nhập dữ liệu cho bảng sử dụng mã SQL	132
Hình 5. 11 Xem cấu trúc bảng	132
Hình 5. 12 Xem dữ liệu bảng	133
Hình 5. 13 Tạo tài khoản người dùng.....	134
Hình 5. 14 Nhập thông tin người dùng và thiết lập quyền hạn chế.....	134
Hình 5. 15 Thiết lập quyền trên Cơ sở dữ liệu	135
Hình 5. 16 Chọn Cơ sở dữ liệu để thiết lập quyền	135
Hình 5. 17 Chọn các quyền cho Cơ sở dữ liệu.....	136

DANH MỤC BẢNG

Bảng 2. 1 Thành phần trang HTML	20
Bảng 2. 2 Ký tự đặc biệt trong HTML	26
Bảng 3. 1 Thuộc tính định dạng văn bản.....	31
Bảng 3. 2 Thuộc tính bố cục và hiển thị.....	32
Bảng 3. 3 Thuộc tính màu sắc và nền.....	32
Bảng 3. 4 Thuộc tính định dạng danh sách	32
Bảng 3. 5 Thuộc tính Flexbox	33
Bảng 4. 1 Các kiểu dữ liệu cơ bản trong Javascript	47
Bảng 4. 2 Phép toán số học	48
Bảng 4. 3 Phép toán so sánh.....	49
Bảng 4. 4 Phép toán logic.....	49
Bảng 4. 5 Phép toán gán	49
Bảng 5. 1 So sánh với trang web tĩnh.....	64
Bảng 5. 2 Kiểu dữ liệu cơ bản trong PHP	73
Bảng 5. 3 Toán tử số học	75
Bảng 5. 4 Toán tử so sánh.....	75
Bảng 5. 5 Toán tử gán.....	75
Bảng 5. 6 Toán tử luận lý logic	76
Bảng 5. 7 Hàm xử lý chuỗi.....	90
Bảng 5. 8 Các kiểu định dạng cho hàm Date	107

Chương 1

Kiến thức cơ bản về Internet và Web

Mục tiêu

Sau khi học xong chương này, người học sẽ có thể:

- Trình bày được các khái niệm cơ bản về Internet, mô hình Client-Server và các giao thức HTTP/HTTPS.
- Giải thích cách hoạt động của cấu trúc URL, hệ thống tên miền (DNS) và vai trò của chúng trong Internet.
- Phân biệt Web tĩnh và Web động, mô tả sự khác nhau giữa hai loại web này.

1.1. Tổng quan về Internet

1.1.1. Các khái niệm cơ bản

Internet (tạm dịch là mạng thông tin toàn cầu) là một hệ thống mạng máy tính toàn cầu kết nối hàng tỷ thiết bị và máy tính trên khắp thế giới. Nó cho phép giao tiếp, truyền thông giữa các máy tính và thiết bị khác nhau từ mọi nơi trên Trái Đất.

Mạng Internet được hình thành từ việc kết nối các mạng máy tính lớn nhỏ lại với nhau thông qua giao thức TCP/IP (Transmission Control Protocol/Internet Protocol). Điều này tạo ra một hệ thống mạng phức tạp và linh hoạt, cho phép trao đổi thông tin, tài liệu, dữ liệu, hình ảnh, âm thanh, video và nhiều hình thức khác của thông tin qua các giao thức và phương tiện truyền thông.

Lịch sử hình thành và phát triển của mạng Internet bắt đầu từ những năm 1960 và diễn ra qua nhiều giai đoạn quan trọng:

- Những năm 1960: Internet khởi đầu từ dự án ARPANET, được phát triển bởi Bộ Quốc phòng Hoa Kỳ nhằm kết nối các trường đại học và trung tâm nghiên cứu để chia sẻ thông tin. ARPANET là mạng máy tính đầu tiên, với bốn nút mạng được kết nối vào năm 1969.
- Những năm 1970: ARPANET mở rộng và phát triển giao thức TCP/IP do Vinton Cerf và Bob Kahn tạo ra vào năm 1974, tạo nền tảng cho việc kết nối các mạng khác nhau thành một hệ thống toàn cầu. Email cũng được phát minh trong giai đoạn này.
- Những năm 1980: ARPANET chính thức chuyển đổi sang hệ thống Internet, với việc sử dụng giao thức TCP/IP trên toàn bộ mạng. Tên gọi “Internet” lần đầu tiên được sử dụng vào thời điểm này, và mạng bắt đầu phục vụ mục đích thương mại.
- Những năm 1990: Internet trở nên phổ biến với sự ra đời của World Wide Web (WWW) do Tim Berners-Lee phát triển vào năm 1991, mang lại giao diện dễ sử dụng cho người dùng. Internet bắt đầu được ứng dụng rộng rãi trong đời sống hàng ngày.
- Những năm 2000: Sự phát triển của công nghệ không dây và thiết bị di động đã thúc đẩy việc truy cập Internet mọi lúc mọi nơi. Số lượng người dùng Internet tăng nhanh chóng, Internet trở thành công cụ quan trọng trong giao tiếp, mua sắm và giải trí.
- Từ những năm 2010 đến nay: Internet tiếp tục phát triển với sự xuất hiện của Internet of Things (IoT), kết nối hàng tỷ thiết bị thông minh. Các công nghệ mới như trí tuệ nhân tạo và blockchain cũng đang làm thay đổi cách chúng ta tương tác với Internet.

Nguyên lý hoạt động của Internet dựa trên các khái niệm và công nghệ cơ bản:

- Mạng lưới toàn cầu: Internet là một mạng lưới toàn cầu phát triển từ ARPANET, kết nối hàng triệu máy tính và thiết bị điện tử. Các thiết bị này được gọi là “nút” (nodes), chúng đóng vai trò là điểm kết nối giữa các mạng con, tạo thành một mạng lớn hơn.

- Giao thức TCP/IP (Transmission Control Protocol/Internet Protocol): là nền tảng cho việc truyền thông và trao đổi dữ liệu giữa các thiết bị trên Internet. Giao thức này đảm bảo tính tin cậy và chính xác trong việc truyền tải dữ liệu, cho phép các thiết bị giao tiếp hiệu quả.

- Địa chỉ IP: Mỗi thiết bị trên Internet được gán một địa chỉ IP (Internet Protocol address) duy nhất, giúp xác định vị trí và định danh của thiết bị trong mạng. Địa chỉ IP đóng vai trò quan trọng trong việc xác định nguồn và đích khi truyền thông tin.

- DNS (Domain Name System): là hệ thống chuyển đổi giữa địa chỉ IP và tên miền. Nó cho phép người dùng truy cập Internet thông qua các tên miền dễ nhớ như “google.com”. thay vì phải nhập địa chỉ IP phức tạp.

- Trình duyệt web: Là phần mềm cho phép người dùng truy cập và duyệt web. Khi người dùng nhập một URL vào trình duyệt, yêu cầu sẽ được gửi đến máy chủ web tương ứng, và phản hồi dạng HTML sẽ được nhận để hiển thị nội dung trang web.

- Truyền thông qua gói tin: Dữ liệu trên Internet được chia nhỏ thành các gói tin trước khi được gửi đi. Mỗi gói tin chứa thông tin về nguồn, đích, dữ liệu và các thông tin điều khiển. Các gói tin này được gửi qua mạng và sẽ được tập hợp lại ở điểm đích.

- Routing (Định tuyến): Các gói tin trên Internet được định tuyến qua nhiều đường dẫn khác nhau để đến đích một cách hiệu quả. Các thiết bị định tuyến (router) đảm nhiệm vai trò xác định đường đi tối ưu cho các gói tin, đảm bảo rằng chúng đến đúng nơi cần đến.

1.1.2. Mô hình Client – Server

Mô hình Client-Server là một cấu trúc mạng máy tính phân chia nhiệm vụ giữa hai thành phần chính: máy khách (client) và máy chủ (server). Trong mô hình này, máy khách là thiết bị hoặc ứng dụng yêu cầu dịch vụ từ máy chủ, trong khi máy chủ là nơi lưu trữ tài nguyên và cung cấp dịch vụ cho máy khách.

Nguyên tắc hoạt động:

- Yêu cầu từ máy khách: Khi người dùng muốn truy cập một dịch vụ, thiết bị của họ (máy khách) sẽ gửi một yêu cầu đến máy chủ thông qua một giao thức mạng, chẳng hạn như HTTP.

- Xử lý yêu cầu: Máy chủ nhận yêu cầu từ máy khách và xử lý nó. Quá trình này có thể bao gồm truy cập cơ sở dữ liệu, thực hiện các phép toán, hoặc gửi yêu cầu đến một máy chủ khác.

- Phản hồi từ máy chủ: Sau khi hoàn tất xử lý, máy chủ sẽ gửi kết quả trở lại cho máy khách. Kết quả này có thể là một trang web, một tệp tin, hoặc thông tin khác mà người dùng cần.

Đặc điểm của mô hình Client-Server:

- Tập trung hóa tài nguyên: Tài nguyên được lưu trữ tập trung trên máy chủ, giúp quản lý và bảo trì dễ dàng hơn.

- Khả năng mở rộng: Mô hình này cho phép thêm nhiều máy khách mà không làm giảm hiệu suất của hệ thống.

- Bảo mật: Dữ liệu có thể được bảo vệ tốt hơn khi được lưu trữ trên máy chủ, với các biện pháp bảo mật như mã hóa và xác thực.

Mô hình Client-Server được sử dụng rộng rãi trong nhiều ứng dụng như email, web, ngân hàng trực tuyến và thương mại điện tử.

1.1.3. Giao thức HTTP và HTTPS

Giao thức HTTP (Hypertext Transfer Protocol) và HTTPS (HTTP Secure) là hai giao thức chính được sử dụng để truyền tải thông tin giữa trình duyệt web và máy chủ web.

HTTP: là giao thức không an toàn, cho phép truyền tải dữ liệu giữa client và server mà không mã hóa thông tin. Khi người dùng nhập URL vào trình duyệt, trình duyệt sẽ gửi yêu cầu HTTP đến máy chủ để lấy dữ liệu trang web. Máy chủ sẽ xử lý yêu cầu và gửi lại phản hồi chứa nội dung trang web dưới dạng HTML.

HTTPS: là phiên bản an toàn của HTTP, sử dụng mã hóa SSL/TLS để bảo vệ thông tin trong quá trình truyền tải. Sử dụng HTTPS giúp bảo vệ dữ liệu nhạy cảm như thông tin cá nhân và tài khoản ngân hàng khỏi việc bị đánh cắp hoặc nghe lén trong quá trình truyền tải.

1.1.4. Cấu trúc URL và DNS

URL (Uniform Resource Locator) là địa chỉ duy nhất của một tài nguyên trên Internet. Cấu trúc cơ bản của URL bao gồm:

- Giao thức: Phần đầu tiên chỉ định giao thức sử dụng (ví dụ: <http://> hoặc <https://>).
- Tên miền: Phần tiếp theo chỉ định tên miền của máy chủ (ví dụ: www.example.com).
- Đường dẫn: Phần cuối cùng chỉ định vị trí cụ thể của tài nguyên trên máy chủ (ví dụ: [/index.html](http://www.example.com/index.html)).

Ví dụ về URL: <https://www.example.com/index.html>

DNS (Domain Name System) là hệ thống chuyển đổi tên miền thành địa chỉ IP, cho phép người dùng truy cập Internet bằng cách sử dụng các tên miền dễ nhớ thay vì phải nhớ địa chỉ IP phức tạp.

Chức năng của DNS:

- Phân giải tên miền: Khi người dùng nhập tên miền vào trình duyệt, DNS sẽ tìm kiếm địa chỉ IP tương ứng với tên miền đó.
- Quản lý tên miền: DNS cũng giúp quản lý các tên miền khác nhau trong một tổ chức hoặc doanh nghiệp.

1.2. Website

1.2.1. World Wide Web (WWW)

World Wide Web (WWW) là một hệ thống thông tin toàn cầu cho phép người dùng truy cập và chia sẻ tài nguyên trên Internet thông qua các trang web. WWW được phát minh vào năm 1989 bởi nhà khoa học máy tính Sir Tim Berners-Lee khi ông làm việc tại CERN (Tổ chức Nghiên cứu Hạt nhân Châu Âu). Mục tiêu ban đầu của WWW là tạo ra một

phương pháp chia sẻ thông tin và kết quả nghiên cứu giữa các nhà khoa học và tổ chức nghiên cứu trên toàn thế giới.

WWW bao gồm ba công nghệ chính:

- HTML (HyperText Markup Language): Ngôn ngữ đánh dấu siêu văn bản được sử dụng để cấu trúc nội dung trên các trang web.
- HTTP (Hypertext Transfer Protocol): Giao thức cho phép truyền tải thông tin giữa trình duyệt và máy chủ web.
- URL (Uniform Resource Locator): Địa chỉ duy nhất của một tài nguyên trên Internet, giúp người dùng dễ dàng truy cập các trang web.

1.2.2. Lịch sử phát triển Web

World Wide Web (WWW) đã trải qua nhiều giai đoạn phát triển quan trọng từ khi ra đời cho đến nay. Mỗi giai đoạn mang đến những cải tiến và thay đổi đáng kể trong cách thức mà người dùng tương tác với thông tin trên Internet. Dưới đây là cái nhìn tổng quan về lịch sử phát triển của WWW, được chia thành các giai đoạn chính: Web 1.0, Web 2.0, và Web 3.0.

a) Web 1.0: Thời kỳ tĩnh (1991-2004)

Web 1.0 là giai đoạn đầu tiên của World Wide Web, nơi mà nội dung chủ yếu là tĩnh và không có sự tương tác giữa người dùng với trang web.

Đặc điểm:

- Nội dung tĩnh: Các trang web được xây dựng chủ yếu bằng HTML, với nội dung không thay đổi trừ khi được cập nhật thủ công bởi lập trình viên.
- Giao diện đơn giản: Giao diện người dùng thường rất cơ bản, không có nhiều yếu tố đồ họa phức tạp.
- Thông tin một chiều: Người dùng chủ yếu chỉ có thể đọc thông tin mà không thể tương tác hay đóng góp nội dung.

Sự kiện nổi bật:

- Năm 1991, Tim Berners-Lee công bố trang web đầu tiên tại CERN.
- Sự ra đời của các trình duyệt web đầu tiên như Mosaic vào năm 1993, giúp người dùng dễ dàng truy cập và duyệt web.

b) Web 2.0: Thời kỳ tương tác (2004-2010)

Web 2.0 đánh dấu sự chuyển mình mạnh mẽ trong cách thức sử dụng Internet, với sự tập trung vào tính tương tác và khả năng chia sẻ nội dung giữa người dùng.

Đặc điểm:

- Nội dung động và tương tác: Người dùng không chỉ tiêu thụ thông tin mà còn có thể tạo ra và chia sẻ nội dung của riêng mình thông qua các nền tảng mạng xã hội, blog và diễn đàn.
- Ứng dụng web phong phú: Sự phát triển của các ứng dụng web như Facebook, Twitter, YouTube đã tạo ra một môi trường giao tiếp xã hội mạnh mẽ.

- Cộng đồng trực tuyến: Người dùng có thể kết nối, tương tác và hợp tác với nhau qua các nền tảng trực tuyến.

Sự kiện nổi bật:

- Năm 2004, Facebook ra đời, đánh dấu sự bùng nổ của mạng xã hội.
- Sự phát triển của YouTube (2005) đã thay đổi cách thức người dùng tiêu thụ nội dung video trực tuyến.

c) Web 3.0: Thời kỳ trí tuệ nhân tạo (2010-nay)

Web 3.0 là giai đoạn hiện tại của World Wide Web, nơi mà trí tuệ nhân tạo (AI) và công nghệ blockchain bắt đầu đóng vai trò quan trọng trong việc cải thiện trải nghiệm người dùng.

Đặc điểm:

- Dữ liệu có cấu trúc và thông minh: Web 3.0 tập trung vào việc cung cấp dữ liệu có cấu trúc hơn, cho phép máy móc hiểu và xử lý thông tin một cách hiệu quả hơn.
- Cá nhân hóa trải nghiệm người dùng: Các ứng dụng web ngày càng trở nên cá nhân hóa hơn nhờ vào AI và machine learning, giúp cung cấp nội dung phù hợp với nhu cầu của từng người dùng.
- Decentralization (phi tập trung): Sự phát triển của blockchain dẫn đến việc tạo ra các ứng dụng phi tập trung (DApps), cho phép người dùng kiểm soát dữ liệu cá nhân mà không cần phụ thuộc vào các công ty lớn.

Sự kiện nổi bật:

- Sự xuất hiện của các công nghệ như trí tuệ nhân tạo, machine learning và blockchain đã mở ra nhiều cơ hội mới cho phát triển ứng dụng trên web.
- Các nền tảng như Ethereum đã thúc đẩy sự phát triển của các ứng dụng phi tập trung.

Lịch sử phát triển của World Wide Web từ Web 1.0 đến Web 3.0 cho thấy sự tiến hóa mạnh mẽ trong cách thức mà con người tương tác với thông tin trên Internet. Từ những trang web tĩnh đơn giản đến những ứng dụng động và thông minh ngày nay, WWW đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày cũng như trong kinh doanh toàn cầu. Sự phát triển này không chỉ mở rộng khả năng truy cập thông tin mà còn tạo ra nhiều cơ hội mới cho giao tiếp và hợp tác trên toàn thế giới.

1.2.3. Phân loại Website

Website có thể được phân loại thành hai loại chính dựa trên cách thức hoạt động và nội dung của chúng:

1.2.3.1. Web tĩnh

Web tĩnh là loại website có nội dung cố định không thay đổi thường xuyên. Các trang web này thường được xây dựng bằng HTML đơn giản mà không có sự tương tác từ phía người dùng. Nội dung của web tĩnh được lưu trữ trên máy chủ và được gửi đến trình duyệt khi có yêu cầu.

Web tĩnh có các đặc điểm sau:

- Nội dung không thay đổi trừ khi được cập nhật thủ công.

- Thường sử dụng cho các trang thông tin như danh thiếp điện tử hoặc trang giới thiệu doanh nghiệp.

Ví dụ: Một trang giới thiệu sản phẩm hoặc dịch vụ cụ thể mà không cần cập nhật thường xuyên.

1.2.3.2. Web động

Web động là loại website có khả năng thay đổi nội dung dựa trên tương tác của người dùng hoặc dữ liệu từ cơ sở dữ liệu. Các trang web này thường được xây dựng bằng các ngôn ngữ lập trình như PHP, Python, Asp.Net hoặc JavaScript, cho phép tạo ra các trải nghiệm người dùng phong phú hơn.

Web động có các đặc điểm sau:

- Nội dung có thể thay đổi theo thời gian thực hoặc theo yêu cầu của người dùng.
- Thường sử dụng cho các ứng dụng trực tuyến như mạng xã hội, cửa hàng điện tử và diễn đàn.

Ví dụ: Một trang thương mại điện tử nơi người dùng có thể tìm kiếm sản phẩm, thêm vào giỏ hàng và thanh toán trực tuyến.

1.3. Giới thiệu ngôn ngữ lập trình Web

1.3.1. Tổng quan

Ngôn ngữ lập trình web là những công cụ thiết yếu trong việc phát triển các ứng dụng và trang web trên Internet. Chúng cho phép lập trình viên tạo ra nội dung động, tương tác và dễ dàng quản lý. Ngôn ngữ lập trình web có thể được chia thành hai loại chính: ngôn ngữ thiết kế và ngôn ngữ lập trình.

- Ngôn ngữ thiết kế: Chủ yếu được sử dụng để định dạng và trình bày nội dung trên web, bao gồm HTML và CSS.

- Ngôn ngữ lập trình: Được sử dụng để xử lý logic, dữ liệu và tương tác với cơ sở dữ liệu, bao gồm JavaScript, PHP, Python, Java, và nhiều ngôn ngữ khác.

1.3.2. Ngôn ngữ thiết kế

Ngôn ngữ thiết kế là những công cụ giúp lập trình viên tạo ra giao diện người dùng hấp dẫn và dễ sử dụng. Hai ngôn ngữ thiết kế chính trong phát triển web là:

- HTML (HyperText Markup Language): Là ngôn ngữ đánh dấu siêu văn bản, HTML được sử dụng để cấu trúc nội dung trên trang web. Nó cho phép lập trình viên xác định các phần tử như tiêu đề, đoạn văn, hình ảnh và liên kết. HTML là nền tảng cơ bản cho mọi trang web.

- CSS (Cascading Style Sheets): CSS được sử dụng để định dạng và trình bày giao diện của trang web. Nó cho phép lập trình viên điều chỉnh màu sắc, font chữ, kích thước và bố cục của các phần tử HTML. CSS giúp tạo ra trải nghiệm người dùng tốt hơn thông qua việc cải thiện tính thẩm mỹ của trang web.

1.3.3. Ngôn ngữ lập trình

Ngôn ngữ lập trình được sử dụng để xây dựng logic phía máy chủ và phía máy khách trong các ứng dụng web. Dưới đây là một số ngôn ngữ lập trình phổ biến:

- JavaScript: Là ngôn ngữ lập trình kịch bản phía máy khách, JavaScript cho phép tạo ra các tương tác động trên trang web mà không cần tải lại trang. Nó được hỗ trợ bởi hầu hết các trình duyệt và có thể tương tác với HTML và CSS để thay đổi nội dung một cách linh hoạt.

- PHP (Hypertext Preprocessor): Là một ngôn ngữ lập trình phía máy chủ, PHP thường được sử dụng để phát triển các ứng dụng web động. PHP cho phép tương tác với cơ sở dữ liệu và xử lý dữ liệu từ người dùng trước khi gửi phản hồi về cho máy khách.

- Python: Python là một ngôn ngữ lập trình đa năng với cú pháp rõ ràng và dễ đọc. Nó thường được sử dụng trong phát triển web thông qua các framework như Django và Flask, Fast API giúp xây dựng ứng dụng nhanh chóng và hiệu quả.

- Csharp: là một ngôn ngữ lập trình hướng đối tượng do Microsoft phát triển, thường được sử dụng trong phát triển web thông qua framework ASP.NET để xây dựng các ứng dụng web mạnh mẽ. C# cung cấp cú pháp rõ ràng và tính năng mạnh mẽ cho việc phát triển ứng dụng doanh nghiệp, đồng thời tích hợp tốt với cơ sở dữ liệu SQL Server.

Ngôn ngữ lập trình web đóng vai trò quan trọng trong việc xây dựng các ứng dụng và trang web hiện đại. Sự kết hợp giữa các ngôn ngữ thiết kế như HTML và CSS với các ngôn ngữ lập trình như JavaScript, PHP, Python, C# giúp tạo ra trải nghiệm người dùng phong phú và linh hoạt trên Internet.

Câu hỏi

Câu 1. Trình bày khái niệm Internet.

Gợi ý: Internet là gì? Hoạt động như thế nào? Vai trò trong cuộc sống và lập trình web.

Câu 2. Mô hình Client – Server là gì?

Gợi ý: Giải thích khái niệm máy khách (Client) và máy chủ (Server), cách chúng giao tiếp với nhau.

Câu 3. Giao thức HTTP và HTTPS có gì khác nhau?

Gợi ý: Nêu mục đích của HTTP và HTTPS, ưu điểm bảo mật của HTTPS.

Câu 4. Cấu trúc URL và vai trò của DNS trong hoạt động của website.

Gợi ý: Phân tích các phần trong URL và cách DNS giúp chuyển đổi tên miền thành địa chỉ IP.

Câu 5. World Wide Web (WWW) là gì?

Gợi ý: Phân biệt giữa Internet và World Wide Web.

Câu 6. Trình bày lịch sử phát triển của Web.

Gợi ý: Các giai đoạn phát triển từ Web 1.0, Web 2.0 đến Web 3.0.

Câu 7. So sánh Web tĩnh và Web động.

Gợi ý: Cách hoạt động của Web tĩnh (HTML, CSS) và Web động (PHP, JavaScript, CSDL).

Câu 8. Trình bày sự khác nhau giữa ngôn ngữ thiết kế và ngôn ngữ lập trình web.

Gợi ý: HTML, CSS thuộc nhóm nào? PHP, JavaScript thuộc nhóm nào?

Câu 9. Liệt kê các ngôn ngữ lập trình web phổ biến hiện nay và vai trò của chúng.

Gợi ý: JavaScript, PHP, Python,...

Chương 2

Ngôn ngữ HTML và CSS

Mục tiêu

Sau khi học xong chương này, người học sẽ có thể:

- Trình bày được cấu trúc, khái niệm và vai trò của ngôn ngữ HTML và CSS trong quy trình thiết kế và xây dựng giao diện web
- Vận dụng thành thạo các thẻ HTML cơ bản để xây dựng cấu trúc nội dung và tổ chức dữ liệu trên trang web.
- Áp dụng hiệu quả cú pháp CSS, các bộ chọn (selectors) và thuộc tính CSS thông dụng để định dạng nội dung, tạo bố cục (layout), thiết lập màu sắc và các hiệu ứng cho giao diện trang web
- Thiết kế và xây dựng được giao diện trang web tĩnh bằng sự kết hợp giữa HTML và CSS, qua đó tạo nền tảng cho việc phát triển các ứng dụng web động với PHP.

2.1. Ngôn ngữ HTML

2.1.1. Giới thiệu HTML

HTML (HyperText Markup Language) là ngôn ngữ đánh dấu siêu văn bản được sử dụng để tạo ra các trang web. HTML cung cấp cấu trúc cơ bản cho nội dung trên Internet, cho phép người dùng định dạng văn bản, hình ảnh, liên kết và nhiều thành phần khác trên trang web.

Các đặc điểm chính của HTML:

- Đánh dấu siêu văn bản: HTML cho phép tạo ra các liên kết giữa các trang web thông qua thẻ <a>, giúp người dùng dễ dàng điều hướng giữa các tài nguyên trên Internet.
- Cấu trúc và định dạng: HTML sử dụng các thẻ (tag) để xác định cấu trúc của nội dung, như tiêu đề, đoạn văn, danh sách, bảng và hình ảnh.
- Tính tương thích: HTML được hỗ trợ bởi tất cả các trình duyệt web hiện đại, đảm bảo rằng nội dung có thể được hiển thị đúng trên nhiều thiết bị khác nhau.

2.1.2. Các thành phần trong trang HTML

Một tài liệu HTML bao gồm nhiều thành phần cơ bản, mỗi thành phần được biểu diễn bằng thẻ (tag). Các thẻ này đóng vai trò cấu trúc, giúp trình duyệt hiểu cách hiển thị nội dung trên trang web.

Cấu trúc của thẻ HTML:

- Hầu hết các thẻ HTML đều tồn tại theo cặp gồm:
 - + Thẻ mở: đánh dấu điểm bắt đầu của một thành phần. Ví dụ: <body>
 - + Thẻ đóng: đánh dấu điểm kết thúc. Ví dụ: </body>
 - + Nội dung nằm giữa hai thẻ chính là nội dung được hiển thị trên trang web.

Ví dụ:

```
<p>Đây là một đoạn văn.</p>
```

- Một số thẻ đặc biệt tự đóng, tức là không có nội dung bên trong và không cần thẻ kết thúc.

Ví dụ:

```
<br /> <!-- Xuống dòng -->
<hr /> <!-- Kẻ đường ngang -->

```

Ghi chú: HTML5 cho phép viết các thẻ tự đóng không cần `</>` (như `
`), nhưng vẫn hiển thị chính xác.

Bảng 2. 1 Thành phần trang HTML

Thành phần	Vai trò
<code><!DOCTYPE html></code>	Khai báo loại tài liệu để trình duyệt biết cách hiển thị trang
<code><html></code>	Là thẻ gốc của tài liệu HTML, chứa tất cả các nội dung của trang
<code><head></code>	Chứa thông tin về tài liệu như tiêu đề, liên kết đến CSS, Javascript và meta tags.
<code><body></code>	Chứa nội dung chính của trang web mà người dùng sẽ thấy

Ví dụ:

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <title>Ví dụ HTML</title>
</head>
<body>
  <h1>Đây là trang HTML của tôi</h1>
  <p>HTML giúp bạn xây dựng trang web.</p>
</body>
</html>
```

Đây là trang HTML của tôi

HTML giúp bạn xây dựng trang web.

2.1.3. Các thẻ (tag) cơ bản

2.1.3.1. Thẻ xử lý văn bản

Các thẻ HTML xử lý văn bản giúp định dạng nội dung hiển thị trên trang web.

a) Thẻ tiêu đề (<h1> - <h6>)

Dùng để hiển thị các tiêu đề, có 6 mức độ từ lớn nhất (<h1>) đến nhỏ nhất (<h6>).

Ví dụ:

<h1>Tiêu đề lớn nhất</h1>

<h2>Tiêu đề cấp 2</h2>

<h3>Tiêu đề cấp 3</h3>

Tiêu đề lớn nhất

Tiêu đề cấp 2

Tiêu đề cấp 3

b) Thẻ đoạn văn bản (<p>)

Dùng để tạo một đoạn văn bản có khoảng cách rõ ràng giữa các đoạn.

Ví dụ:

<p>Đây là một đoạn văn bản trong HTML.</p>

c) Thẻ in đậm, in nghiêng, gạch chân

 (Bold): Làm đậm chữ.

<i> (Italic): Làm nghiêng chữ.

<u> (Underline): Gạch chân chữ.

Ví dụ:

<p>Văn bản in đậm</p>

<p><i>Văn bản in nghiêng</i></p>

<p><u>Văn bản gạch chân</u></p>

Văn bản in đậm

Văn bản in nghiêng

Văn bản gạch chân

d) Thẻ danh sách (, ,)

Dùng để tạo danh sách:

- Danh sách không có thứ tự (): Hiển thị các mục có dấu đầu dòng, mặc định hiển thị các mục với dấu chấm tròn

- Danh sách có thứ tự (): Hiển thị danh sách có đánh số thứ tự, mặc định đánh số 1,2,3,...;

Ví dụ danh sách không thứ tự:

```
<ul>
  <li>Mục 1</li>
  <li>Mục 2</li>
  <li>Mục 3</li>
</ul>
```

- Mục 1
- Mục 2
- Mục 3

Ví dụ danh sách có thứ tự:

```
<ol>
  <li>Bước 1</li>
  <li>Bước 2</li>
  <li>Bước 3</li>
</ol>
```

1. Bước 1
2. Bước 2
3. Bước 3

- Có thể thay đổi kiểu đánh bằng thuộc tính *type*

ví dụ: `<ul type="circle">`, `<ul type="square">`, `<ol type="A">`, `<ol type="i">`. Tuy nhiên, *type* không còn được khuyến khích trong HTML5 → nên dùng CSS:

```
ul {
  list-style-type: square;
}
```

2.1.3.2. Thẻ xử lý hình ảnh

Hình ảnh trong HTML được hiển thị bằng thẻ ``.

Thuộc tính quan trọng:

- `src`: Đường dẫn đến file ảnh.
- `alt`: Văn bản thay thế khi ảnh không tải được.
- `width` và `height`: Định kích thước ảnh.

Ví dụ:

```


```



2.1.3.3. Thẻ liên kết (<a>)

Thẻ `<a>` (anchor) được dùng để tạo liên kết giữa các trang web hoặc điều hướng trong cùng một trang.

a) Liên kết đến trang web khác

Dùng thuộc tính ***href*** để chỉ định địa chỉ URL cần liên kết, ***target="_blank"*** để mở liên kết trong cửa sổ hoặc tab mới.

Ví dụ:

```
<a href="https://www.facebook.com" target="_blank">
```

Mở Facebook trong tab mới

```
</a>
```

b) Liên kết đến một phần cụ thể trong trang web

Dùng **id** để đánh dấu một phần của trang, sau đó sử dụng **#id** để tạo liên kết đến vị trí đó.

Ví dụ:

```
<h2 id="gioi-thieu">Giới thiệu</h2>
```

Giới thiệu

```
<a href="#gioi-thieu">Chuyển đến phần Giới thiệu</a>
```

[Chuyển đến phần Giới thiệu](#)

2.1.3.4. Thẻ phân chia bố cục (<div>)

Thẻ **<div>** được sử dụng để nhóm các phần tử HTML lại với nhau nhằm quản lý bố cục trang dễ dàng hơn.

Ví dụ:

```
<div style="background-color: pink;">
```

```
<h2>Giới thiệu</h2>
```

```
<p>Đây là nội dung của phần giới thiệu.</p>
```

```
</div>
```

Giới thiệu

Đây là nội dung của phần giới thiệu.

2.1.3.5. Thẻ Table

Bảng trong HTML được tạo ra bằng cách sử dụng các thẻ sau:

- **<table>**: Thẻ chính để tạo bảng.
- **<tr>**: Thẻ để tạo hàng trong bảng.
- **<th>**: Thẻ để tạo ô tiêu đề trong hàng.
- **<td>**: Thẻ để tạo ô dữ liệu trong hàng.

Các thuộc tính của bảng:

- **border**: Xác định độ dày của đường viền cho bảng.
- **cellpadding**: Xác định khoảng cách giữa nội dung ô và viền ô.

- cellpadding: Xác định khoảng cách giữa các ô trong bảng.

Ví dụ:

```
<table border="1" cellpadding="5" cellspacing="0">
  <tr>
    <th>Họ Tên</th>
    <th>Quê Quán</th>
  </tr>
  <tr>
    <td>Trần Kim Hương</td>
    <td>Đồng Tháp</td>
  </tr>
</table>
```

Họ Tên	Quê Quán
Trần Kim Hương	Đồng Tháp

Gộp hàng và cột: HTML cũng hỗ trợ việc gộp các ô lại với nhau bằng cách sử dụng thuộc tính rowspan và colspan.

- rowspan: Cho phép một ô chiếm nhiều hàng.

Ví dụ:

```
<table border="1">
  <tr>
    <td rowspan="2">Gộp 2 hàng</td>
    <td>Ô 1, Hàng 1</td>
  </tr>
  <tr>
    <td>Ô 1, Hàng 2</td>
  </tr>
</table>
```

Gộp 2 hàng	Ô 1, Hàng 1
	Ô 1, Hàng 2

- colspan: Cho phép một ô chiếm nhiều cột.

Ví dụ:

```
<table border="1">
  <tr>
    <td colspan="2">Gộp 2 cột</td>
  </tr>
  <tr>
    <td>Ô 1, Hàng 2</td>
    <td>Ô 2, Hàng 2</td>
  </tr>
</table>
```

Gộp 2 cột	
Ô 1, Hàng 2	Ô 2, Hàng 2

2.1.3.6. Thẻ Form

Form (biểu mẫu) trong HTML là một phần tử cho phép người dùng nhập dữ liệu và gửi đến máy chủ. Biểu mẫu thường được sử dụng để thu thập thông tin từ người dùng như tên, email, mật khẩu, v.v.

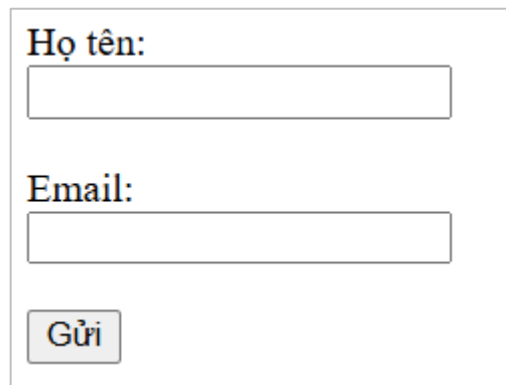
Một form trong HTML được khai báo bằng thẻ **<form>** và có thể chứa nhiều loại **input** khác nhau.

Ví dụ:

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

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

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



Các thành phần chính của form:

- Thẻ **<form>**:

- + action: URL mà dữ liệu sẽ được gửi đến khi người dùng nhấn nút gửi.
- + method: Phương thức gửi dữ liệu (thường là GET hoặc POST).

- Thẻ **<input>**: Dùng để tạo các trường nhập liệu khác nhau.

- + type: Xác định loại input (text, email, password, radio, checkbox, v.v.).

Ví dụ:

```
<input type="text" name="username" placeholder="Nhập tên đăng nhập">
<input type="password" name="password" placeholder="Nhập mật khẩu">

<input type="radio" name="gender" value="male"> Nam
<input type="radio" name="gender" value="female"> Nữ

<input type="checkbox" name="subscribe"> Đăng ký nhận bản tin
```

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

☐ Nam
☐ Nữ

☐ Đăng ký nhận bản tin

2.1.4. Các ký tự đặc biệt thường dùng

Trong HTML, một số ký tự có ý nghĩa đặc biệt hoặc không thể gõ trực tiếp bằng bàn phím, do đó cần sử dụng mã ký tự đặc biệt (HTML Entities).

Ví dụ về một số ký tự không thể gõ trực tiếp trong HTML:

- Dấu < và > không thể viết trực tiếp vì chúng trùng với thẻ HTML.
- Khoảng trắng nhiều lần không được hiển thị đúng trên trình duyệt.
- Một số ký tự như dấu bản quyền ©, ký hiệu toán học \pm , hoặc các biểu tượng đặc biệt khác không có trên bàn phím.

Các ký tự đặc biệt phổ biến:

Bảng 2. 2 Ký tự đặc biệt trong HTML

Ký tự	Mô tả	HTML Entity
	Khoảng trắng	
<	Dấu nhỏ hơn	<
>	Dấu lớn hơn	>
&	Dấu & (và)	&
"	Dấu nháy kép	"
'	Dấu nháy đơn	'
©	Dấu bản quyền	©
®	Dấu đăng ký thương hiệu	®
™	Dấu thương hiệu	™
€	Ký hiệu Euro	€
¥	Ký hiệu Yên Nhật	¥
±	Dấu cộng trừ	±

Ví dụ:

`<p><h1> Đây là tiêu đề </h1></p>`

`<p>Bản quyền © 2024 - Tất cả quyền được bảo lưu.</p>`

`<h1> Đây là tiêu đề </h1>`

`Bản quyền © 2024 - Tất cả quyền được bảo lưu.`

2.1.5. HTML5

HTML5 là phiên bản mới nhất của ngôn ngữ đánh dấu siêu văn bản (HTML), được phát triển để cải thiện khả năng tương tác và trải nghiệm người dùng trên web. HTML5 không chỉ mở rộng khả năng của HTML mà còn tích hợp nhiều tính năng mới, giúp lập trình viên xây dựng ứng dụng web phong phú hơn.

Các tính năng nổi bật của HTML5:

a) Hỗ trợ video và âm thanh:

HTML5 cung cấp các thẻ `<video>` và `<audio>` cho phép nhúng video và âm thanh trực tiếp vào trang web mà không cần sử dụng plugin bên ngoài.

Ví dụ:

`<video width="320" height="240" controls>`

`<source src="movie.mp4" type="video/mp4">`

Trình duyệt của bạn không hỗ trợ video.

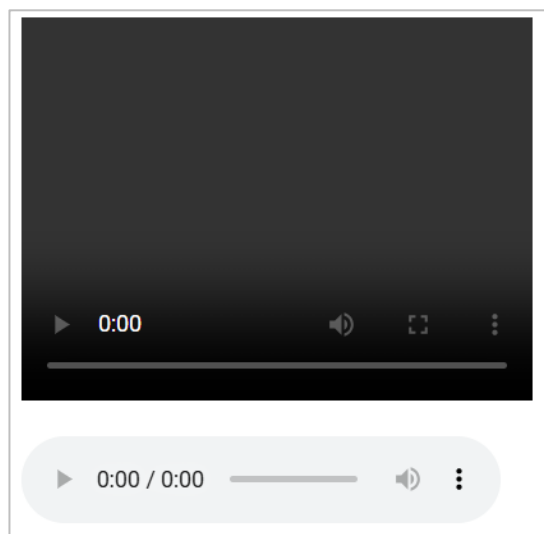
`</video>`

`<audio controls>`

`<source src="audio.mp3" type="audio/mpeg">`

Trình duyệt của bạn không hỗ trợ audio.

`</audio>`



b) Thẻ Semantic:

HTML5 giới thiệu các thẻ ngữ nghĩa giúp cải thiện khả năng đọc và tổ chức nội dung trang web

Thẻ	Chức năng
<header>	Định nghĩa phần đầu của trang web hoặc một phần nội dung.
<nav>	Xác định phần menu điều hướng.
<section>	Đại diện cho một phần nội dung độc lập.
<article>	Dùng để hiển thị một bài viết hoàn chỉnh.
<aside>	Dùng cho nội dung phụ như thanh sidebar.
<footer>	Định nghĩa phần chân trang.

Ví dụ:

```
<header>
  <h1>Tiêu đề trang web</h1>
</header>
<nav>
  <a href="#">Trang chủ</a> | <a href="#">Giới thiệu</a>
</nav>
<section>
  <article>
    <h2>Bài viết đầu tiên</h2>
    <p>Đây là nội dung bài viết.</p>
  </article>
</section>
<footer>
  <p>Bản quyền &copy; 2025</p>
</footer>
```

Tiêu đề trang web

[Trang chủ](#) | [Giới thiệu](#)

Bài viết đầu tiên

Đây là nội dung bài viết.

Bản quyền © 2025

2.2. Ngôn ngữ CSS

2.2.1. Giới thiệu CSS

CSS (Cascading Style Sheets) là ngôn ngữ dùng để định dạng và thiết kế giao diện cho các trang web. CSS cho phép lập trình viên tách biệt nội dung HTML với các quy tắc định dạng, giúp dễ dàng quản lý và cập nhật giao diện mà không ảnh hưởng đến cấu trúc nội dung. CSS đóng vai trò quan trọng trong việc tạo ra các trải nghiệm người dùng hấp dẫn và dễ sử dụng trên web.

Ví dụ Trang HTML có sử dụng CSS

```
<html>
<head>
  <title>Trang web có CSS</title>
  <style>
    body{
      background-color: #f0f0f0;
      font-family: Arial, sans-serif;
    }
    h1{
      color: blue;
    }
    p{
      font-size: 18px;
      color: gray;
    }
  </style>
</head>
<body>
  <h1>Chào mừng bạn đến với khóa học PHP</h1>
  <p>Đây là một đoạn văn bản.</p>
</body>
</html>
```

Chào mừng bạn đến với khóa học PHP

Đây là một đoạn văn bản.

2.2.2. Cú pháp và cách dùng CSS

2.2.2.1. Cú pháp CSS

Cú pháp CSS bao gồm ba phần chính:

- Selector (Bộ chọn): Phần tử HTML mà bạn muốn áp dụng kiểu.
- Property (Thuộc tính): Đặc điểm muốn thay đổi, ví dụ như màu sắc, kích thước chữ.
- Value (Giá trị): Giá trị của thuộc tính.

Cú pháp cơ bản được viết như sau:

```
selector {  
    property: value;  
}
```

Ví dụ:

```
h1 {  
    color: blue;  
    font-size: 24px;  
}
```

2.2.2.2. Cách sử dụng CSS

Có 3 cách để áp dụng CSS vào HTML:

- a) CSS nội tuyến (Inline CSS) – Viết trực tiếp trong thuộc tính style của thẻ HTML.

Ví dụ:

```
<h1 style="color: red; font-size: 24px;">Chào mừng bạn</h1>
```

- b) CSS nội bộ (Internal CSS) – Viết trong thẻ <style> bên trong <head>

Ví dụ:

```
<style>  
    h1 {  
        color: red;  
        font-size: 24px;  
    }  
</style>
```

- c) CSS ngoài (External CSS) – Viết trong một file .css riêng và liên kết bằng <link>.

Ví dụ:

```
<link rel="stylesheet" href="style.css">
```

2.2.3. CSS selector

CSS Selector là cách mà CSS chọn các phần tử HTML để áp dụng định dạng. Có nhiều loại bộ chọn khác nhau

2.2.3.1. Selector cơ bản

a) Selector theo thẻ (Tag selector)

```
h1 { color: blue; }  
p { font-size: 16px; }
```

b) Selector theo lớp (Class selector)

```
.highlight { color: red; font-weight: bold; }
```

c) Selector theo ID (ID selector)

```
#header { background-color: gray; }
```

2.2.3.2. Selector nâng cao

a) Child selector

```
div > p { color: blue; }
```

b) Group selector

```
h1, h2, h3 { color: green; }
```

c) Pseudo-class Selector

```
a:hover { color: orange; }
```

2.2.4. Thuộc tính CSS thông dụng

2.2.4.1. Thuộc tính định dạng văn bản

Bảng 3. 1 Thuộc tính định dạng văn bản

Thuộc tính	Mô tả	Ví dụ
color	Màu chữ	color: red;
font-size	Kích thước chữ	font-size: 16px;
font-weight	Độ đậm của chữ	font-weight: bold;
font-style	Kiểu chữ	font-style: italic;
text-align	Căn chỉnh văn bản	text-align: center;

text-decoration	Trang trí chữ (gạch chân, gạch ngang)	text-decoration: underline;
letter-spacing	Khoảng cách giữa các ký tự	letter-spacing: 2px;
line-height	Độ cao dòng chữ	line-height: 1.5;

2.2.4.2. Thuộc tính bố cục và hiển thị

Bảng 3. 2 Thuộc tính bố cục và hiển thị

Thuộc tính	Mô tả	Ví dụ
width	Chiều rộng phần tử	width: 100px;
height	Chiều cao phần tử	height: 50px;
margin	Khoảng cách ngoài	margin: 10px;
padding	Khoảng cách bên trong	padding: 15px;
border	Viền xung quanh phần tử	border: 2px solid black;
display	Kiểu hiển thị của phần tử	display: block;
visibility	Hiển thị hoặc ẩn phần tử	visibility: hidden;
overflow	Kiểm soát nội dung vượt quá kích thước phần tử	overflow: scroll;

2.2.4.3. Thuộc tính màu sắc và nền

Bảng 3. 3 Thuộc tính màu sắc và nền

Thuộc tính	Mô tả	Ví dụ
background-color	Màu nền	background-color: yellow;
background-image	Ảnh nền	background-image: url('image.jpg');
background-size	Kích thước ảnh nền	background-size: cover;
opacity	Độ trong suốt	opacity: 0.5;

2.2.4.4. Thuộc tính định dạng danh sách

Bảng 3. 4 Thuộc tính định dạng danh sách

Thuộc tính	Mô tả	Ví dụ
------------	-------	-------

list-style-type	Kiểu đánh dấu danh sách	list-style-type: square;
list-style-position	Vị trí dấu đầu dòng	list-style-position: inside;

2.2.4.5. Thuộc tính xử lý hiển thị linh hoạt (Flexbox)

Bảng 3. 5 Thuộc tính Flexbox

Thuộc tính	Mô tả	Ví dụ
display	Kích hoạt Flexbox	display: flex;
justify-content	Căn chỉnh nội dung theo trục chính	justify-content: center;
align-items	Căn chỉnh nội dung theo trục phụ	align-items: flex-start;
flex-direction	Hướng của các phần tử con	flex-direction: column;

Ví dụ:

```
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
}

.container {
  width: 80%;
  margin: auto;
  padding: 20px;
  border: 1px solid #ccc;
  background-color: white;
}

h1 {
  text-align: center;
  color: blue;
}

p {
  font-size: 16px;
  line-height: 1.5;
  color: gray;
}
```

Câu hỏi, bài tập

Câu hỏi

Câu 1. HTML và CSS là gì? Trình bày vai trò và mối quan hệ cơ bản giữa chúng trong quá trình thiết kế và phát triển giao diện web?

Gợi ý: HTML là ngôn ngữ đánh dấu dùng để tạo cấu trúc và nội dung cơ bản của trang web. CSS (Cascading Style Sheets) là ngôn ngữ định kiểu giúp định dạng và trình bày nội dung HTML, giúp trang web đẹp và chuyên nghiệp hơn. Chúng hoạt động bổ trợ lẫn nhau, HTML cung cấp khung sườn còn CSS “trang trí” cho khung sườn đó

Câu 2. Liệt kê các thẻ HTML cơ bản và chức năng của chúng?

Gợi ý: Bao gồm các thẻ tiêu đề, đoạn văn, hình ảnh, danh sách, bảng, liên kết, biểu mẫu,...

Câu 3. So sánh Web tĩnh và Web động?

Gợi ý: Web tĩnh chỉ dùng HTML, web động kết hợp ngôn ngữ lập trình phía server như PHP.

Câu 4. HTML5 có những cải tiến nào so với HTML4?

Gợi ý: Hỗ trợ thẻ ngữ nghĩa (<article>, <section>...), API nâng cao, quản lý đa phương tiện tốt hơn.

Câu 5. Trình bày cú pháp tạo bảng và form trong HTML?

Gợi ý: Sử dụng <table> để tạo bảng, <form> để tạo biểu mẫu thu thập dữ liệu.

Câu 6. Có mấy cách để nhúng CSS vào trang web HTML? Cho ví dụ minh họa.

Gợi ý: CSS có 3 cách nhúng: Inline, Internal, External.

Câu 7. Trình bày cú pháp cơ bản của CSS.

Gợi ý: Cấu trúc gồm: bộ chọn (selector), thuộc tính (property), giá trị (value).

Câu 8. CSS Selector là gì? Kể tên và mô tả một số bộ chọn phổ biến.

Gợi ý: CSS Selector dùng để chọn phần tử HTML cần áp dụng kiểu dáng. Ví dụ: bộ chọn id (#), class (.), bộ chọn phần tử (h1, p), bộ chọn con (div > p).

Câu 9. Phân biệt padding và margin trong CSS.

Gợi ý: padding là khoảng cách bên trong phần tử, còn margin là khoảng cách bên ngoài phần tử so với phần tử khác.

Bài tập

Bài 1. Tạo trang web giới thiệu bản thân

Yêu cầu:

a) Tạo một trang HTML đơn giản giới thiệu bản thân. Nội dung gồm:

- Họ và tên.
- Hình ảnh đại diện (dùng).
- Một đoạn mô tả ngắn về bản thân.
- Liên kết đến một trang web yêu thích (dùng <a>).

b) Định dạng trang giới thiệu bản thân với CSS để căn chỉnh bố cục và tạo hiệu ứng đơn giản như sau:

- Định dạng nội dung chính nằm giữa trang.
- Ảnh đại diện có hình tròn, kích thước 150px x 150px.
- Màu nền trang web: #f4f4f4.
- Hộp chứa nội dung có đổ bóng nhẹ (box-shadow), có viền bo góc (border-radius).
- Sử dụng Google Fonts để thay đổi kiểu chữ.



Bài 2. Tạo danh sách sở thích cá nhân

Yêu cầu:

Sử dụng danh sách có thứ tự () và danh sách không thứ tự () để liệt kê các sở thích cá nhân.

Ví dụ:

Sở thích thể thao: Bóng đá, Cầu lông.

Sở thích âm nhạc: Nghe nhạc Pop, Nhạc cổ điển.

Sở thích của tôi

Sở thích thể thao:

- Bóng đá
- Cầu lông

Sở thích âm nhạc:

1. Nghe nhạc Pop
2. Nghe nhạc cổ điển

Bài 3. Thiết kế danh sách sản phẩm có bảng giá

Yêu cầu:

- Sử dụng thẻ Table để tạo bảng chứa danh sách sản phẩm với:
 - Tiêu đề gồm các cột: Tên sản phẩm, Giá bán, Mô tả ngắn
 - Thêm ít nhất 3 sản phẩm vào bảng.
- Sử dụng CSS để tạo bảng có viền và màu nền xen kẽ như sau:
 - Kích thước bảng chiếm 60% chiều rộng màn hình.
 - Dòng tiêu đề (th) có màu nền #009688, chữ màu trắng, viết in hoa.
 - Dòng dữ liệu (td) có màu nền xen kẽ giữa các dòng (nth-child(even)).
 - Bảng có viền 1px solid #ddd.
 - Chữ trong bảng căn giữa (text-align: center).

Danh sách sản phẩm

TÊN SẢN PHẨM	GIÁ BÁN	MÔ TẢ
Điện thoại iPhone 14	22.000.000 VNĐ	Điện thoại thông minh với camera sắc nét
Laptop Dell XPS 13	30.000.000 VNĐ	Máy tính xách tay nhỏ gọn, hiệu năng mạnh mẽ
Loa Bluetooth JBL	1.500.000 VNĐ	Loa không dây với âm thanh sống động

Bài 4. Tạo form Đăng ký tài khoản người dùng

Yêu cầu:

a) Tạo form HTML (<form>) để người dùng nhập thông tin đăng ký.

Các trường dữ liệu gồm: Họ và tên (<input type="text">); Email (<input type="email">); Mật khẩu (<input type="password">); Giới tính (<input type="radio">); Sở thích (<input type="checkbox">); Nút gửi (<input type="submit">)

b) Sử dụng CSS để căn chỉnh và làm đẹp form:

- Form nằm giữa trang, có chiều rộng 50%.
- Các ô nhập liệu (input) có bo góc (border-radius: 5px), viền màu xám nhạt.
- Nút đăng ký (button) có màu nền xanh dương (#007bff), chữ trắng, hiệu ứng hover sáng hơn 10%.

Bài 5. Tạo trang web gồm nhiều trang HTML liên kết với nhau

Yêu cầu:

Tạo một website nhỏ với 3 trang HTML liên kết với nhau bằng thẻ <a>.

- Trang index.html là trang chính, có liên kết đến:
- Trang about.html (Giới thiệu bản thân).
- Trang contact.html (Thông tin liên hệ).

Chào mừng đến với website của tôi

[Giới thiệu bản thân](#)

[Thông tin liên hệ](#)

Trang Index.html

Thông tin cá nhân

Tôi là Trần Kim Hương, một lập trình viên web.

[Quay lại trang chủ](#)

Trang about.html

Liên hệ với tôi

Email: tkhuong@dthu.edu.vn

[Quay lại trang chủ](#)

Trang contact.html

Bài 6. Hãy tạo trang HTML hiển thị các ký tự đặc biệt như mẫu sau:

CÁC KÝ TỰ ĐẶC BIỆT

Bản quyền: ©

Thương hiệu: ™

Thương hiệu đã đăng ký: ®

Email: tkhuong@dthu.edu.vn

Bài tập 7: Xây dựng layout trang web đơn giản

Yêu cầu:

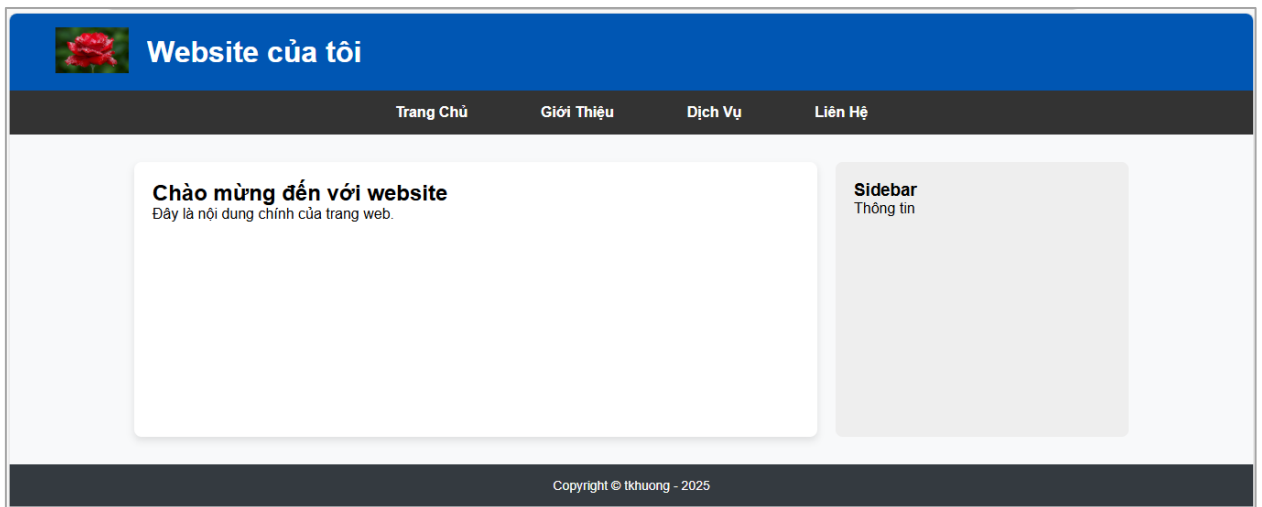
Tạo một trang HTML gồm các phần sau:

- Header: Hiển thị logo và tiêu đề trang web.
- Menu: Thanh điều hướng ngang có hiệu ứng hover chuyên nghiệp.
- Content: Nội dung chính của trang web, bố cục gọn gàng, chuyên nghiệp.

- Sidebar: Thanh bên chứa các mục nội dung phụ.
- Footer: Chân trang có thông tin bản quyền, căn giữa, chữ nhỏ hơn..

Yêu cầu CSS:

- Header: Logo bên trái, tiêu đề trang bên phải; Chiều cao: 80px, nền màu xanh đậm (#0056b3).
- Menu: Menu ngang chuyên nghiệp, có hiệu ứng hover mượt; Màu nền xám đậm (#333), chữ trắng.
- Nội dung chính & Sidebar: Content: Chiếm 70%, nền trắng, bo góc nhẹ, đổ bóng. Sidebar: Chiếm 30%, nền xám nhạt, có padding đẹp. Hiển thị theo dạng Flexbox.
- Footer: Căn giữa, màu nền #343a40, chữ trắng.



Chương 3

Ngôn ngữ JavaScript

Mục tiêu

Sau khi học xong chương này, người học sẽ có thể:

- Hiểu vai trò và ứng dụng của JavaScript trong lập trình web.
- Sử dụng JavaScript để thao tác DOM, xử lý sự kiện và xác thực dữ liệu.
- Kết hợp JavaScript với PHP để xây dựng ứng dụng web động.
- Xây dựng các ứng dụng web đơn giản, thực tế với JavaScript.

3.1. Giới thiệu JavaScript

3.1.1 JavaScript là gì?

JavaScript (JS) là một ngôn ngữ lập trình kịch bản phổ biến được sử dụng rộng rãi trong phát triển web, giúp các trang web trở nên động và tương tác hơn. Được phát triển bởi Brendan Eich tại Netscape vào năm 1995, JavaScript đã trở thành một phần không thể thiếu trong bộ ba công nghệ cốt lõi của phát triển web, bao gồm HTML, CSS và JavaScript.

JavaScript thường được nhúng trực tiếp vào các trang HTML hoặc liên kết qua các tệp có đuôi “.js”. Ngôn ngữ này chủ yếu hoạt động ở phía máy khách (client-side), nghĩa là mã sẽ được tải về và thực thi trên trình duyệt của người dùng, giúp giảm tải cho máy chủ. Với sự phát triển của tiêu chuẩn ECMAScript, JavaScript ngày càng được chuẩn hóa và mở rộng, trở thành nền tảng cho các ứng dụng web phức tạp, được sử dụng rộng rãi bởi các nhà phát triển trên toàn thế giới.

Một số đặc trưng của JavaScript như:

- Ngôn ngữ kịch bản (Scripting Language): Không cần biên dịch trước, mã JavaScript được thực thi trực tiếp bởi trình duyệt khi trang web được tải.
- Chạy phía client (Client-side): Thực thi trên máy của người dùng, giúp giảm tải cho server và cung cấp trải nghiệm người dùng nhanh chóng, mượt mà.
- Động và linh hoạt: Cho phép thay đổi nội dung HTML và CSS trong thời gian thực, làm cho các trang web sống động và tương tác.
- Bất đồng bộ (Asynchronous): Hỗ trợ lập trình bất đồng bộ qua callback, promises, và async/await, quản lý hiệu quả các tác vụ tốn thời gian như truy vấn API.
- Đa nền tảng: Chạy trên mọi trình duyệt và hệ điều hành, giúp ứng dụng web dễ dàng triển khai trên nhiều thiết bị.
- Cộng đồng hỗ trợ mạnh mẽ: Được hỗ trợ bởi một cộng đồng lớn với nhiều thư viện và framework như jQuery, React, Angular, và Vue.js, giúp tăng tốc và mở rộng khả năng phát triển web.

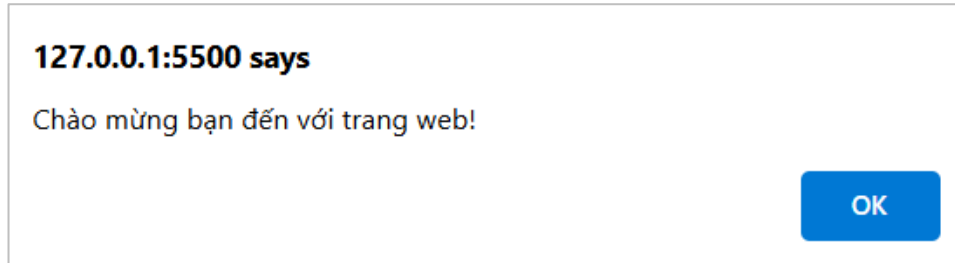
Ứng dụng thực tế:

- Tạo các trang web tương tác như xác thực form, hiển thị thông báo, hoặc thay đổi giao diện theo hành động của người dùng.
- Tích hợp với PHP để tạo ứng dụng web động.

Ví dụ: Một thông báo đơn giản hiển thị trên trình duyệt web

```
<script>
    alert("Chào mừng bạn đến với trang web!");
</script>
```

Kết quả:



3.1.2. Vai trò và ứng dụng của JavaScript

JavaScript đóng vai trò quan trọng trong việc làm cho các trang web trở nên linh hoạt và tương tác. Dưới đây là các ứng dụng chính:

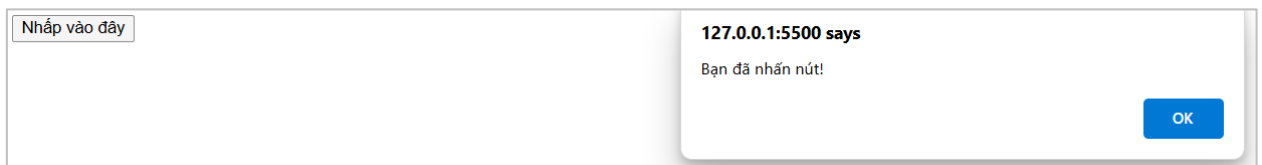
(1) *Tạo trải nghiệm người dùng tương tác:*

Cho phép phản hồi tức thì với các hành động của người dùng mà không cần tải lại trang.

Ví dụ: Khi người dùng nhấp vào nút, một thông báo xuất hiện.

```
<button onclick="alert('Bạn đã nhấn nút!')">Nhấp vào đây</button>
```

Kết quả:



(2) *Xử lý dữ liệu phía client:*

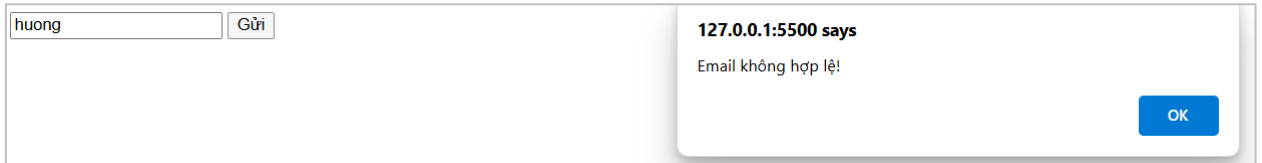
Kiểm tra và xác thực dữ liệu trước khi gửi lên server để giảm tải.

Ví dụ: Kiểm tra email có hợp lệ trước khi gửi form.

```
<form onsubmit="return validateEmail()">
    <input type="text" id="email" placeholder="Nhập email">
    <button type="submit">Gửi</button>
</form>
<script>
    function validateEmail() {
        let email = document.getElementById("email").value;
        if (!email.includes("@")) {
            alert("Email không hợp lệ!");
            return false;
        }
        return true;
    }
</script>
```

```
}  
</script>
```

Kết quả:



(3) Kết hợp với PHP:

PHP xử lý logic phía server, JavaScript đảm nhiệm phần hiển thị và tương tác.

Ví dụ: PHP gửi dữ liệu sản phẩm, JavaScript hiển thị danh sách động.

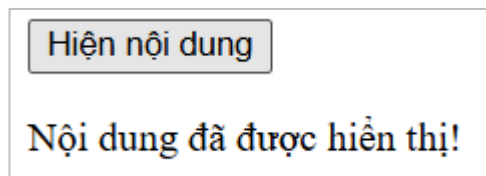
(4) Tạo hiệu ứng động:

Ẩn/hiện phần tử, tạo menu trượt, thay đổi màu sắc.

Ví dụ: Hiển thị một đoạn văn bản khi nhấp vào nút.

```
<button onclick="showText()">Hiện nội dung</button>  
<p id="content" style="display:none;">Nội dung đã được hiển thị!</p>  
<script>  
  function showText() {  
    document.getElementById("content").style.display = "block";  
  }  
</script>
```

Kết quả:



(5) Phát triển ứng dụng phức tạp:

Sử dụng các framework như React hoặc Vue.js để tạo ứng dụng web hiện đại.

3.1.3. Sử dụng JavaScript trong HTML và PHP

JavaScript có thể được nhúng trực tiếp vào trang HTML hoặc sử dụng trong các ứng dụng web viết bằng PHP. Điều này phù hợp với các ứng dụng web nơi PHP xử lý phía server, còn JavaScript xử lý phía client.

3.1.3.1. Sử dụng JavaScript trong HTML

. Trong một tập tin HTML, JavaScript có thể được nhúng trực tiếp vào phần `<head>`, `<body>`, hoặc thông qua một tập tin `.js` riêng biệt.

a) Nhúng JavaScript trực tiếp trong HTML

Chúng ta có thể viết mã JavaScript trực tiếp trong cặp thẻ `<script>` bên trong HTML. Cách này thường được sử dụng cho các đoạn mã JavaScript ngắn hoặc cụ thể cho một trang web.

Ví dụ:

```
<script>
  function greet() {
    alert("Chào mừng bạn!");
  }
</script>
<button onclick="greet()">Chào</button>
```

b) *Tham chiếu tệp JavaScript bên ngoài:*

Khi mã JavaScript trở nên phức tạp hoặc cần sử dụng lại trên nhiều trang, chúng ta cần tách mã JavaScript ra thành một tập tin .js riêng biệt và tham chiếu vào tệp HTML.

Ví dụ:

Tạo tập tin script.js

```
function showMessage() {
  alert("Chào mừng bạn đến với trang web!");
}
```

Tham chiếu vào tập tin HTML

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ví dụ Sử dụng JavaScript Ngoài</title>
  <script src="script.js"></script>
</head>
<body>
  <h1>Ví dụ JavaScript trong HTML</h1>
  <button onclick="showMessage()">Nhấp vào đây</button>
</body>
```

Kết quả:

Ví dụ JavaScript trong HTML

127.0.0.1:5500 says
Chào mừng bạn đến với trang web!

3.1.3.2. *Sử dụng JavaScript trong PHP*

Trong PHP, JavaScript thường được sử dụng để tạo các tương tác động trên giao diện người dùng, trong khi PHP đảm nhận vai trò xử lý logic phía máy chủ. JavaScript có thể được sử dụng trong tệp PHP theo cách tương tự như trong tệp HTML.

a) *Nhúng JavaScript trực tiếp trong PHP:*

Chúng ta có thể nhúng JavaScript trực tiếp trong mã PHP để tương tác với người dùng sau khi xử lý logic PHP.

Ví dụ:

```
<body>
```

```

<?php
    $message = "Chào mừng bạn đến với trang web!";
?>
<h1>Ví dụ JavaScript trong PHP</h1>
<button onclick="showMessage()">Nhấp vào đây</button>

<script>
    function showMessage() {
        var message = "<?php echo $message; ?>";
        alert(message);
    }
</script>
</body>

```

b) Sử dụng JavaScript để xử lý dữ liệu từ PHP:

JavaScript cũng có thể tương tác với dữ liệu PHP đã được xử lý và hiển thị trên trang web, cho phép xử lý các sự kiện động dựa trên kết quả từ PHP.

Ví dụ 1. Nhúng dữ liệu từ PHP vào JavaScript

```

<?php
    $userName = "Huong";
?>
<body>
    <h1>Xin chào, <?php echo $userName; ?>!</h1>
    <button onclick="displayGreeting()">Nhấp vào đây để nhận thông điệp</button>

    <script>
        function displayGreeting() {
            var userName = "<?php echo $userName; ?>";
            alert("Chào " + userName + ", chúc bạn một ngày tốt lành!");
        }
    </script>
</body>

```

Ví dụ 2. Xử lý động với PHP và JavaScript

```

<body>
    <h1>Danh sách sản phẩm</h1>
    <?php
        // Mảng sản phẩm từ PHP
        $products = ['Sản phẩm 1', 'Sản phẩm 2', 'Sản phẩm 3'];
    ?>

    <!-- Danh sách sẽ hiển thị ở đây -->
    <ul id="productList"></ul>

    <script>
        // Nhận dữ liệu sản phẩm từ PHP
        let products = <?php echo json_encode($products); ?>;
    </script>

```

```
// Thêm sản phẩm vào danh sách HTML
let list = document.getElementById("productList");
products.forEach(product => {
    let li = document.createElement("li");
    li.textContent = product;
    list.appendChild(li);
});
</script>
</body>
```

3.1.4. Viết chương trình JavaScript đầu tiên

Chúng ta có thể sử dụng phần mềm Visual Code hoặc các Text Editor Online để viết và chạy thử chương trình JavaScript được nhanh và trực quan nhất. Một số Text Editor Online gợi ý như: Codesandbox.io, Playcode.io, Jsfiddle.net.

Cấu trúc cơ bản của chương trình JavaScript bao gồm:

- HTML (HyperText Markup Language): Đây là ngôn ngữ định dạng cấu trúc cơ bản của trang web. JavaScript thường được nhúng vào các trang HTML để thực hiện các tác vụ tương tác.

- Thẻ <script>: Thẻ này được sử dụng để nhúng mã JavaScript vào trang HTML. Nó có thể được đặt trong phần <head> hoặc <body> của tài liệu HTML.

- Câu lệnh JavaScript: Đây là các dòng mã lệnh thực hiện các thao tác cụ thể, chẳng hạn như hiển thị thông báo hoặc ghi thông tin vào console.

Ví dụ. Chương trình Hiển thị thông tin người dùng

Chương trình cho phép sinh viên nhập tên của mình vào một ô nhập liệu và nhấn nút “Hiển thị”. JavaScript sẽ đọc tên và hiển thị một thông báo chào mừng, đồng thời ghi thông tin này vào console để minh họa cách JavaScript xử lý dữ liệu đầu vào.

```
<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Chương trình JavaScript Đầu tiên</title>
    <script>
        // Hàm xử lý khi người dùng nhấn nút
        function showGreeting() {
            // Lấy giá trị từ ô nhập liệu
            let userName = document.getElementById("nameInput").value;

            // Kiểm tra nếu người dùng không nhập tên
            if (userName === "") {
                alert("Vui lòng nhập tên của bạn!");
                return;
            }
        }
    </script>
</head>
<body>
    <div>
        <input type="text" id="nameInput" value="Tên của bạn" />
        <button type="button" value="Hiển thị" />
    </div>
</body>
```

```
// Hiển thị thông báo chào mừng
alert("Chào mừng, " + userName + " đến với JavaScript!");

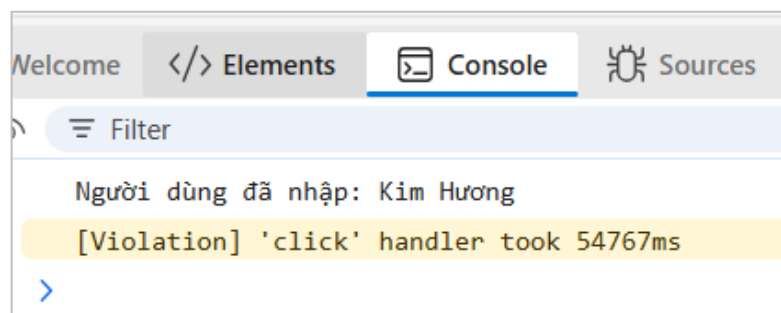
// Ghi thông báo vào console
console.log("Người dùng đã nhập: " + userName);
}
</script>
</head>
<body>
  <h1>Chương trình JavaScript Đầu tiên</h1>
  <p>Nhập tên của bạn vào ô bên dưới và nhấn nút để xem thông báo.</p>

  <!-- Ô nhập liệu -->
  <input type="text" id="nameInput" placeholder="Nhập tên của bạn">

  <!-- Nút nhấn -->
  <button onclick="showGreeting()">Hiển thị</button>

  <p><strong>Gợi ý:</strong> Mở console trình duyệt (F12 > Console) để xem thông tin chi tiết.</p>
</body>
</html>
```

Kết quả:



3.2. Biến, kiểu dữ liệu và phép toán trong JavaScript

Một chương trình máy tính, theo nghĩa cơ bản nhất, là một tập hợp các câu lệnh (statements) được thiết kế để chỉ đạo máy tính thực hiện các tác vụ cụ thể. Trong JavaScript, các câu lệnh này được thực thi bởi trình duyệt web thay vì trực tiếp bởi máy tính.

Khi nghiên cứu các ngôn ngữ lập trình khác, chúng ta sẽ nhận thấy rằng chúng đều dựa trên những nguyên tắc cốt lõi chung, bao gồm biến, hàm, kiểu dữ liệu, toán tử, mảng, và vòng lặp. Những nguyên tắc này tạo nên nền tảng của mọi ngôn ngữ lập trình, với sự khác biệt chính thường nằm ở cú pháp – cách mà các câu lệnh được viết và biểu đạt.

Chính vì vậy, khi đã thành thạo một ngôn ngữ lập trình, việc tiếp cận và học hỏi các ngôn ngữ khác sẽ trở nên dễ dàng hơn rất nhiều. Bây giờ, chúng ta sẽ đi vào tìm hiểu cú pháp cơ bản của Javascript.

3.2.1. Biến

Biến (variable) trong JavaScript là một đại diện cho một giá trị dữ liệu được lưu trữ trong bộ nhớ và có thể thay đổi trong quá trình thực thi chương trình. Biến là công cụ cơ bản để quản lý và thao tác dữ liệu.

Cú pháp khai báo biến:

JavaScript cung cấp ba từ khóa để khai báo biến: *var*, *let*, và *const*.

a) **var**: Từ khóa truyền thống để khai báo biến. Biến được khai báo bằng *var* có phạm vi toàn cục hoặc cục bộ dựa trên vị trí khai báo.

b) **let**: Được giới thiệu trong ES6, từ khóa *let* khai báo biến với phạm vi khối (block scope). Đây là từ khóa nên sử dụng cho biến có khả năng thay đổi giá trị.

c) **const**: Khai báo biến hằng số (constant), tức là biến mà giá trị không thể thay đổi sau khi đã được gán.

Ví dụ:

```
var x = 5;
let = 10;
const PI = 3.1416;
```

Quy tắc đặt tên biến:

- Tên biến phải bắt đầu bằng một chữ cái, dấu gạch dưới (`_`) hoặc ký tự `$`.
- Không được bắt đầu bằng số và không được chứa các ký tự đặc biệt khác ngoài (`_`) và `$`.
- JavaScript phân biệt chữ hoa và chữ thường, vì vậy *myVariable* và *myvariable* là hai biến khác nhau.

3.2.2. Kiểu dữ liệu

Kiểu dữ liệu (data type) là phân loại các loại giá trị mà biến có thể lưu trữ. JavaScript là ngôn ngữ lập trình có kiểu dữ liệu động, nghĩa là chúng ta không cần phải khai báo kiểu dữ liệu khi khai báo biến; kiểu dữ liệu của biến sẽ được xác định tự động dựa trên giá trị mà nó nắm giữ.

Bảng 4. 1 Các kiểu dữ liệu cơ bản trong Javascript

Kiểu dữ liệu	Mô tả	Ví dụ
number	Lưu trữ số, có thể là số nguyên hoặc số thực	var a = 10; let b = 3.14;
string	Là một chuỗi ký tự, đặt trong dấu nháy đơn ‘’ hoặc dấu nháy kép “”.	var name = “Tran Kim Huong”; let country = ‘Viet Nam’;
boolean	Chỉ có 2 giá trị: <i>true</i> hoặc <i>false</i>	var isActive = true;

undefined	Một biến chưa được gán giá trị sẽ có kiểu <i>undefined</i>	var x; console.log(x); // undefined
null	Đại diện cho một giá trị rỗng hoặc không tồn tại	var emptyValue = null;
array	Là danh sách có thứ tự của các giá trị, có thể lưu nhiều giá trị đồng thời, mỗi giá trị được gọi là một phần tử của mảng.	var numbers = [1, 2, 3, 4];
object	Tập hợp các cặp giá trị <i>key-value</i> . Một đối tượng có thể chứa nhiều thuộc tính và phương thức.	var student = { name: 'Huong', country: 'Viet Nam' }; student.name; student['name'];

Để xác định kiểu của một dữ liệu thì chúng ta sử dụng toán tử *typeof*.

Ví dụ:

```
<script>
var year = 1987;
var a = typeof year; //number
</script>
```

3.2.3. Phép toán

Phép toán (operators) là các ký hiệu được sử dụng để thực hiện các thao tác trên các giá trị dữ liệu. JavaScript hỗ trợ nhiều loại phép toán khác nhau.

Các phép toán chính trong JavaScript:

Bảng 4. 2 Phép toán số học

Phép toán	Ví dụ
Phép cộng (+)	let x = 10; let y = 5; console.log(x + y);
Phép trừ (-)	let kq = x-y;
Phép nhân (*)	let kq = x*y;
Phép chia (/)	let kq = x/y;
Phép chia lấy dư (%)	let kq = x%y;
Phép tăng (++)	let kq = ++x;

Phép giảm (--)	let kq = --y;
----------------	---------------

Bảng 4. 3 Phép toán so sánh

Phép toán	Ví dụ
So sánh bằng, không xét kiểu dữ liệu (==)	let x = 10; let y = '5'; console.log(x == y); // true
So sánh bằng, xét kiểu dữ liệu (===)	console.log(x === y); // false
So sánh khác, không xét kiểu dữ liệu (!=)	console.log(x != y); // true
So sánh khác, xét kiểu dữ liệu (!===)	console.log(x !== y); // true
Lớn hơn (>)	console.log(x > y); // true
Nhỏ hơn (<)	console.log(x < y); // false
Lớn hơn hoặc bằng (>=)	console.log(x >= y); // true
Nhỏ hơn hoặc bằng (<=)	console.log(x <= y); // false

Bảng 4. 4 Phép toán logic

Phép toán	Ví dụ
Toán tử AND (&&)	let x = 10; let y = 5; let isTrue = (x>5 && y<5); // false
Toán tử OR ()	let isTrue = (x>5 y<5); // true
Toán tử NOT (!)	let isTrue = !(x>5 && y<5); // true

Bảng 4. 5 Phép toán gán

Phép toán	Ví dụ
-----------	-------

Gán giá trị (=)	let x = 10; let y = 5; x = y; // x=5
Cộng và gán (+=)	let kq += y;
Trừ và gán (-=)	let kq -= y;
Nhân và gán (*=)	let kq *= y;
Chia và gán (/=)	let kq /= y;
Chia lấy dư và gán (%=)	let kq %= y;

3.3. Hàm trong JavaScript

Hàm (function) trong JavaScript là một khối mã được định nghĩa để thực hiện một tác vụ cụ thể. Hàm giúp chúng ta tổ chức mã lệnh một cách có cấu trúc, tái sử dụng mã lệnh, và làm cho chương trình dễ đọc hơn. Một hàm có thể nhận vào các tham số (parameters) và trả về một giá trị.

* **Hàm định nghĩa:**

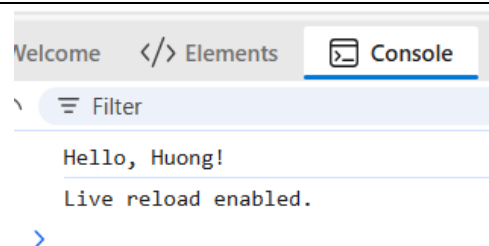
Hàm định nghĩa là cách khai báo hàm truyền thống trong JavaScript, được xác định bằng từ khóa *function*, theo sau là tên hàm, danh sách tham số và khối lệnh.

```
function functionName(parameters) {
    // Khối lệnh thực thi
    return value; // (tùy chọn) Giá trị trả về
}
```

Ví dụ:

```
function greet(name) {
    return "Hello, " + name + "!";
}

let message = greet("Huong");
console.log(message);
```



* **Biểu thức hàm:**

Ngoài cách định nghĩa hàm thông thường, JavaScript còn cho phép chúng ta định nghĩa hàm dưới dạng biểu thức. Hàm này có thể được gán cho một biến và có thể ẩn danh (không cần tên hàm).

Ví dụ:

```
const square = function(number) {
```

```
    return number * number;

};

console.log(square(5));
```

*** Hàm mũi tên**

Hàm mũi tên (Arrow Function) là cú pháp ngắn gọn hơn để viết hàm trong JavaScript, được giới thiệu từ ES6. Hàm mũi tên không có từ khóa function, thay vào đó, dấu mũi tên (=>) được sử dụng để định nghĩa hàm.

```
const square = (number) => {

    return number * number;

};
```

*** Hàm gọi lại (Callback)**

Hàm gọi lại là hàm được truyền như một tham số vào một hàm khác và được thực thi sau khi hàm chứa nó hoàn thành công việc. Callback là một khái niệm quan trọng trong lập trình bất đồng bộ (asynchronous programming) của JavaScript.

```
function process(callback) {

    // Gọi hàm callback sau khi hoàn thành công việc
    callback();

}
```

Ví dụ:

```
function processUserInput(callback) {

    let name = prompt("Please enter your name.");
    callback(name);

}

function greet(name) {

    console.log("Hello, " + name + "!");

}

processUserInput(greet);
```

Ví dụ minh họa:

```
// Hàm định nghĩa
function add(a, b) {

    return a + b;

}

// Biểu thức hàm
const multiply = function(a, b) {

    return a * b;

};

// Hàm mũi tên
```

```
const subtract = (a, b) => a - b;

// Hàm gọi lại
function calculator(a, b, operation) {
    return operation(a, b);
}

console.log(add(5, 3)); // Output: 8
console.log(multiply(5, 3)); // Output: 15
console.log(subtract(5, 3)); // Output: 2
console.log(calculator(5, 3, add)); // Output: 8
```

3.4. Cấu trúc lệnh trong JavaScript

3.4.1. Cấu trúc điều kiện

a) Cấu trúc IF...ELSE

Cấu trúc điều kiện *if...else* là một trong những cấu trúc phổ biến nhất, cho phép thực thi một đoạn mã khi một điều kiện cụ thể là đúng, và thực thi một đoạn mã khác khi điều kiện đó sai.

Cú pháp:

```
if (condition) {
    // Mã lệnh thực thi nếu điều kiện đúng
} else {
    // Mã lệnh thực thi nếu điều kiện sai
}
```

Ví dụ:

```
let age = 18;

if (age >= 18) {
    console.log("Bạn đủ tuổi để tham gia.");
} else {
    console.log("Bạn chưa đủ tuổi.");
}
```

b) Cấu trúc IF...ELSE IF... ELSE

Cấu trúc *if...else if...else* cho phép kiểm tra nhiều điều kiện khác nhau và thực thi các đoạn mã tương ứng với từng điều kiện.

Cú pháp:

```
if (condition1) {
    // Mã lệnh thực thi nếu điều kiện1 đúng
} else if (condition2) {
    // Mã lệnh thực thi nếu điều kiện2 đúng
```

```
} else {  
    // Mã lệnh thực thi nếu không có điều kiện nào đúng  
}
```

Ví dụ:

```
let score = 85;  
if (score >= 90) {  
    console.log("Xuất sắc");  
} else if (score >= 70) {  
    console.log("Khá");  
} else {  
    console.log("Trung bình");  
}
```

c) Cấu trúc **SWITCH**

Cấu trúc *switch* được sử dụng khi bạn muốn kiểm tra giá trị của một biến và thực hiện các lệnh tương ứng với từng giá trị cụ thể.

Cú pháp:

```
switch (expression) {  
    case value1:  
        // Mã lệnh khi expression === value1  
        break;  
    case value2:  
        // Mã lệnh khi expression === value2  
        break;  
    default:  
        // Mã lệnh khi không khớp với bất kỳ giá trị nào  
}
```

Ví dụ:

```
let day = 3;  
switch (day) {  
    case 1:  
        console.log("Thứ hai");  
        break;  
    case 2:  
        console.log("Thứ ba");  
        break;  
}
```

```
case 3:
    console.log("Thứ tư");
    break;
default:
    console.log("Không xác định");
}
```

3.4.2. Cấu trúc lặp

a) Vòng lặp *FOR*

Vòng lặp for được sử dụng khi biết rõ số lần lặp lại của vòng lặp.

Cú pháp:

```
for (initialization; condition; update) {
    // Mã lệnh thực thi trong mỗi lần lặp
}
```

Ví dụ:

```
for (let i = 0; i < 5; i++) {
    console.log("Lần lặp thứ " + i);
}
```

b) Vòng lặp *WHILE*

Vòng lặp while thực hiện lặp lại một đoạn mã cho đến khi điều kiện không còn đúng.

Cú pháp:

```
while (condition) {
    // Mã lệnh thực thi trong mỗi lần lặp
}
```

Ví dụ:

```
let i = 0;
while (i < 5) {
    console.log("Lần lặp thứ " + i);
    i++;
}
```

b) Vòng lặp *DO...WHILE*

Vòng lặp *do...while* thực hiện lệnh ít nhất một lần, sau đó kiểm tra điều kiện để xác định có tiếp tục lặp hay không.

Cú pháp:

```
do {
    // Mã lệnh thực thi
}
```

```
} while (condition);
```

Ví dụ:

```
let i = 0;  
do {  
    console.log("Lần lặp thứ " + i);  
    i++;  
} while (i < 5);
```

3.4.3. Câu lệnh điều khiển

a) Câu lệnh *BREAK*

Câu lệnh *break* được sử dụng để thoát ra khỏi vòng lặp hoặc cấu trúc switch trước khi hoàn thành tất cả các lần lặp hoặc các trường hợp khác.

Ví dụ:

```
for (let i = 0; i < 10; i++) {  
    if (i === 5) {  
        break;  
    }  
    console.log(i); // Chỉ in ra các số từ 0 đến 4  
}
```

b) Câu lệnh *CONTINUE*

Câu lệnh *continue* được sử dụng để bỏ qua lần lặp hiện tại và chuyển sang lần lặp tiếp theo trong vòng lặp.

Ví dụ:

```
for (let i = 0; i < 10; i++) {  
    if (i % 2 === 0) {  
        continue; // Bỏ qua các số chẵn  
    }  
    console.log(i); // Chỉ in ra các số lẻ  
}
```

3.5. DOM (Document Object Model)

3.5.1. Khái niệm DOM

DOM (Document Object Model) là một chuẩn được định nghĩa bởi W3C (World Wide Web Consortium) cho phép các ngôn ngữ lập trình như JavaScript tương tác và thay đổi nội dung, cấu trúc của tài liệu HTML hoặc XML. DOM là một mô hình cây phân cấp, trong đó mỗi phần tử trong trang HTML được biểu diễn như một nút (node) trong cây.

3.5.2. Thao tác với DOM

(1) Truy cập các phần tử trong DOM

- `document.getElementById(id)`: Truy cập phần tử bằng id.
- `document.querySelector(selector)`: Truy cập phần tử theo selector CSS.

Ví dụ:

```
<div id="content">Đây là nội dung cũ</div>

<script>

    let content = document.getElementById("content");
    content.innerHTML = "Nội dung đã thay đổi!";

</script>
```

(2) Thêm và xóa phần tử

- `appendChild()`: Thêm một phần tử con vào cuối phần tử cha.
- `removeChild()`: Xóa một phần tử con khỏi phần tử cha.

Ví dụ:

```
<ul id="productList"></ul>

<script>

    let list = document.getElementById("productList");
    let newItem = document.createElement("li");
    newItem.textContent = "Sản phẩm mới";
    list.appendChild(newItem);

</script>
```

3.5.3. Xử lý sự kiện trong DOM

(1) Thêm sự kiện

- `addEventListener()`: Để thêm sự kiện cho một phần tử.

Ví dụ:

```
<button id="myButton">Click me</button>

<script>

    document.getElementById("myButton").addEventListener("click", function() {
        alert("Button clicked!");
    });

</script>
```

3.6. Đối tượng trong JavaScript

3.6.1. Khái niệm đối tượng

- Đối tượng là một tập hợp các cặp key-value (tên thuộc tính và giá trị).
- Phương thức: Là hàm được khai báo bên trong đối tượng.


```
let person = {
  name: "Huong",
  country: VN,
  greet: function() {
    alert("Chào " + this.name);
  }
};

person.greet(); // Output: Chào Huong
```

3.6.2. Tạo đối tượng

Cách tạo đối tượng:

(1) *Object Literal*

```
let car = { brand: "Toyota", model: "Camry", year: 2020 };
```

(2) *Object Constructor*

```
function Car(brand, model, year) {
  this.brand = brand;
  this.model = model;
  this.year = year;
}

let myCar = new Car("Toyota", "Camry", 2020);
```

3.6.3. Truy cập và thay đổi đối tượng

(1) *Truy cập*

```
let person = { name: "Huong", country: VN };
console.log(person.name); // Huong
```

(2) *Thay đổi*

```
person.country = "US";
console.log(person.country); // US
```

3.7. Kết hợp JavaScript với PHP

* Truyền dữ liệu từ PHP sang JavaScript

Trong một ứng dụng web, đôi khi cần chuyển dữ liệu từ PHP (chạy phía server) sang JavaScript (chạy phía trình duyệt) để xử lý phía client.

Một cách phổ biến để làm điều này là sử dụng hàm `json_encode()` của PHP để chuyển dữ liệu PHP thành định dạng JSON, sau đó nhúng trực tiếp vào mã JavaScript.

Ví dụ:

Giả sử ta có một mảng sản phẩm trong PHP và muốn sử dụng mảng này trong JavaScript:

```
<?php
    // Mảng trong PHP
    $products = ["Laptop", "Smartphone", "Tablet"];
?>
```

Trong phần HTML, ta truyền mảng PHP này vào một biến JavaScript như sau:

```
<script>
    // Sử dụng json_encode để chuyển mảng PHP thành mảng JavaScript
    let products = <?php echo json_encode($products); ?>;

    // Kiểm tra kết quả trong console trình duyệt
    console.log(products);

    // Kết quả: ["Laptop", "Smartphone", "Tablet"]
</script>
```

3.8. Xác thực dữ liệu phía Client

3.8.1. Kiểm tra trường dữ liệu

Ví dụ: Kiểm tra trường dữ liệu không được trống
JavaScript

```
<script>
    function validateForm() {
        let name = document.getElementById("name").value;

        if (name.trim() === "") {
            alert("Vui lòng nhập tên trước khi gửi.");
            return false; // Ngăn form gửi đi
        }

        // Nếu hợp lệ, cho phép gửi form
        return true;
    }
</script>
```

HTML

```
<form onsubmit="return validateForm()">
    <label for="name">Tên:</label>
    <input type="text" id="name" placeholder="Nhập tên của bạn">
    <button type="submit">Gửi</button>
</form>
```

Kết quả:

Tên:

Gửi

127.0.0.1:5500 says

Vui lòng nhập tên trước khi gửi.

OK

3.8.2. Kiểm tra định dạng email

Ví dụ: Kiểm tra định dạng email hợp lệ

JavaScript

```
<script>
    function validateEmail() {
        let email = document.getElementById("email").value;
        let regex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;
        if (!regex.test(email)) {
            alert("Email không hợp lệ");
            return false;
        }
        return true;
    }
</script>
```

HTML

```
<form onsubmit="return validateEmail()">
    <input type="email" id="email" placeholder="Nhập email">
    <button type="submit">Gửi</button>
</form>
```

Kết quả:

Gửi

!

Please include an '@' in the email address. 'huong' is missing an '@'.

Câu hỏi, bài tập

Câu hỏi

Câu 1. JavaScript là gì? Trình bày vai trò của JavaScript trong lập trình web.

Gợi ý: JavaScript là ngôn ngữ lập trình phía client, dùng để làm cho trang web trở nên động và tương tác.

Câu 2. Phân biệt JavaScript với PHP về mặt xử lý dữ liệu phía client và server.

Gợi ý: JavaScript xử lý dữ liệu trên trình duyệt người dùng, PHP xử lý dữ liệu trên máy chủ.

Câu 3. Kể tên một số ứng dụng thực tế của JavaScript trong phát triển web.

Gợi ý: Tạo hiệu ứng động, kiểm tra dữ liệu form, tích hợp API, cập nhật nội dung mà không tải lại trang.

Câu 4. Viết cú pháp để khai báo biến trong JavaScript. Phân biệt var, let, và const.

Gợi ý: Sử dụng var cho biến cục bộ (không nên dùng trong ES6+), let cho biến có thể thay đổi giá trị, const cho biến không đổi.

Câu 5. Liệt kê các kiểu dữ liệu cơ bản trong JavaScript và cho ví dụ.

Gợi ý: Number, String, Boolean, Object, Array, null, undefined.

Câu 6. Toán tử typeof dùng để làm gì?

Gợi ý: Kiểm tra kiểu dữ liệu của một biến hoặc giá trị. Ví dụ: typeof "Hello" trả về string.

Câu 7. Làm thế nào để khai báo một hàm trong JavaScript?

Gợi ý: Dùng từ khóa function hoặc cú pháp hàm mũi tên (arrow function).

Câu 8. Viết một hàm kiểm tra xem một số có phải là số chẵn hay không.

Gợi ý: Sử dụng toán tử chia dư % để kiểm tra.

Câu 9. Làm thế nào để một hàm nhận tham số mặc định?

Gợi ý: Định nghĩa tham số trong hàm với giá trị mặc định. Ví dụ: function greet(name = "your name") { ... }.

Câu 10. Trình bày cú pháp của cấu trúc điều kiện if...else.

Câu 11. Viết một vòng lặp for để in ra các số từ 1 đến 10.

Gợi ý:

```
for (let i = 1; i <= 10; i++) {  
    console.log(i);  
}
```

Câu 12. DOM là gì? Làm thế nào để truy cập phần tử HTML trong JavaScript?

Gợi ý: DOM là mô hình đối tượng tài liệu. Truy cập phần tử dùng getElementById, querySelector.

Câu 13. Viết đoạn mã để thay đổi nội dung của một thẻ <p> có id là text.

Gợi ý: document.getElementById("text").innerHTML = "Nội dung mới";

Câu 14. Làm thế nào để thêm một phần tử mới vào danh sách ?

Gợi ý: Dùng createElement và appendChild.

Câu 15. Trình bày cách truyền dữ liệu từ PHP sang JavaScript.

Gợi ý: Dùng json_encode để xuất dữ liệu PHP sang JavaScript.

Câu 16. Làm thế nào để gửi dữ liệu từ JavaScript đến PHP mà không cần tải lại trang?

Gợi ý: Sử dụng fetch API hoặc AJAX.

Câu 17. Vì sao cần xác thực dữ liệu phía client trước khi gửi lên server?

Gợi ý: Để giảm tải cho server, cung cấp trải nghiệm người dùng tốt hơn.

Câu 18. Viết đoạn mã kiểm tra một trường nhập liệu không được để trống.

Gợi ý: Kiểm tra giá trị của input xem có rỗng hay không.

Bài tập

Bài 1. Tạo một trang web với một nút bấm. Khi nhấn nút, một thông báo “Xin chào JavaScript” sẽ hiển thị trên màn hình.

Gợi ý: Sử dụng thẻ <button> và hàm alert() trong JavaScript.

Bài 2. Tạo một form HTML cho phép người dùng nhập tên và email. Khi nhấn nút “Gửi”, hiển thị dữ liệu người dùng nhập trên trang web.

Gợi ý: Dùng document.getElementById() để lấy giá trị từ các trường nhập liệu và hiển thị bằng cách thay đổi nội dung của một thẻ HTML.

Bài 3. Tạo một danh sách sản phẩm (dưới dạng mảng JavaScript). Hiển thị danh sách sản phẩm này trên trang web.

Gợi ý: Dùng vòng lặp for để duyệt qua mảng, thêm từng sản phẩm vào danh sách bằng DOM.

Bài 4. Tạo một ứng dụng nhỏ với form nhập liệu (tên, email). Xác thực dữ liệu trước khi gửi: Tên không được để trống; Email phải đúng định dạng; Hiển thị thông báo lỗi nếu dữ liệu không hợp lệ.

Gợi ý: Sử dụng biểu thức chính quy (regex) để kiểm tra định dạng email.

Bài 5. Tạo một trang web với form đăng nhập. Khi nhấn “Đăng nhập”, gửi thông tin tài khoản đến PHP bằng AJAX. PHP kiểm tra tài khoản và trả về kết quả: Thành công: “Đăng nhập thành công!”; Thất bại: “Sai thông tin đăng nhập!”

Gợi ý: Dùng fetch API để gửi dữ liệu; Dùng PHP để xử lý POST request và trả về phản hồi.

Bài 6. Tạo một ứng dụng nhỏ hiển thị danh sách sản phẩm từ PHP trên trang web bằng JavaScript.

Gợi ý: PHP xuất danh sách dưới dạng JSON, JavaScript nhận và hiển thị danh sách trên giao diện.

Chương 4

Lập trình Web với PHP

Mục tiêu

Sau khi học xong chương này, người học sẽ có thể:

- Cài đặt môi trường PHP và tạo trang web động đầu tiên.
- Sử dụng cú pháp PHP để xử lý biến, kiểu dữ liệu, toán tử và cấu trúc điều khiển.
- Viết và sử dụng hàm trong PHP để tối ưu mã nguồn.
- Ứng dụng PHP để xây dựng trang web động, xử lý biểu mẫu và hiển thị dữ liệu.

4.1. Giới thiệu về PHP

4.1.1. PHP là gì?

PHP là từ viết tắt của Personal Home Page (nay đã chuyển thành Hypertext Preprocessor) là ngôn ngữ lập trình kịch bản (script) được dùng để phát triển các ứng dụng web chạy trên máy chủ (server). PHP là ngôn ngữ lập trình có mã nguồn mở, tương thích với nhiều nền tảng khác nhau như MacOS, Linux, Windows, ..., là ngôn ngữ mạnh mẽ có nhiều tiện ích mở rộng (extensions) và frameworks có sẵn, được đánh giá là ngôn ngữ dễ thiết lập, dễ học. Các tập tin PHP được lưu với phần mở rộng **“.php”**. Khi xây dựng website với ngôn ngữ PHP, chuỗi lệnh sẽ được xử lý trên server để từ đó sinh ra mã HTML trên client. Do đó, các ứng dụng website PHP sẽ hoạt động một cách dễ dàng và dễ kết nối với các website khác trên hệ thống mạng internet.

PHP là một trong những ngôn ngữ lập trình phổ biến vì nó có các ưu điểm như sau:

- *Tính đơn giản và linh động*: PHP sử dụng mã nguồn mở nên việc cài đặt và sử dụng nó rất dễ dàng.
- *Cộng đồng hỗ trợ lớn*: PHP là một ngôn ngữ phổ biến nên các diễn đàn, đội nhóm chuyên sâu của PHP thuộc hàng ngũ đầu của ngành. Bên cạnh đó, thị trường tuyển dụng cho công việc này cũng khá lớn.
- *Hệ quản trị Cơ sở dữ liệu đa dạng*: PHP cho phép kết nối với hầu hết các hệ quản trị cơ sở dữ liệu như MySQL, MS-SQL, SQLite, PostgreSQL, v.v. nên chúng ta có thể dễ dàng chọn hệ cơ sở dữ liệu tối ưu nhất cho ứng dụng của mình.
- *Thư viện phong phú*: Nhiều sách hướng dẫn và các tài liệu tham khảo có sẵn, cung cấp các kiến thức hữu ích cho các lập trình viên mới có thể làm quen dần.

Tuy nhiên, ngôn ngữ lập trình PHP cũng có một số hạn chế nhất định. Trong đó, hạn chế lớn nhất có thể nói đến đó là lỗi bảo mật, vì ngôn ngữ PHP có mã nguồn mở nên các lỗ hổng của mã nguồn sẽ bị công khai ngay sau khi chúng được phát hiện. Do đó, trong thời gian chờ sửa chữa, các lỗ hổng này có thể bị khai thác với mục đích xấu.

4.1.2. Lịch sử phát triển PHP

PHP ban đầu được tạo ra bởi Rasmus Lerdorf vào năm 1994, sau đó nhóm nghiên cứu PHP hoàn thiện mã nguồn và tạo ra phiên bản khởi đầu là PHP2 được phát hành năm 1995 với tên gọi *Personal Home Page*. Sau đó, năm 1998 thì PHP3 ra đời, với phiên bản này, PHP được đổi tên thành *PHP: Hypertext Processor*.

Năm 2000, PHP4 giới thiệu *Zend engine* – chương trình này cải thiện đáng kể hiệu suất của PHP và khiến PHP trở nên thông dụng trong cộng đồng phát triển web. Tiếp đến, năm 2004, PHP5 giới thiệu *Zend engine II*. Ngoài ra, PHP5 còn giới thiệu các tính năng như hỗ trợ toàn diện cho lập trình hướng đối tượng; mở rộng đối tượng dữ liệu PHP – PDO (PHP Data Object) cùng nhiều tính năng mới khác.

Năm 2015, PHP 7 được phát hành, đánh dấu sự khởi đầu của chuỗi phiên bản mới của ngôn ngữ lập trình PHP, mang đến nhiều cải tiến và tính năng so với phiên bản trước đó (PHP 5.6) như hiệu suất nhanh gấp đôi giúp ứng dụng chạy nhanh hơn và tiết kiệm tài nguyên hệ thống; sử dụng AST (Abstract System Tree) để biên dịch mã nguồn, giúp tối ưu quá trình thực thi; hỗ trợ 64-bit trên tất cả các nền tảng; cung cấp cấu trúc Exception tốt hơn, giúp quản lý lỗi dễ dàng hơn; hỗ trợ khai báo kiểu dữ liệu cho tham số và giá trị trả về của hàm; có thể tạo lớp ẩn danh (Anonymous Classes); Asserts giúp kiểm tra điều kiện trong mã nguồn và không tốn thêm chi phí thời gian thực thi.

PHP8 là phiên bản mới nhất của ngôn ngữ lập trình PHP (tại thời điểm viết tài liệu này), được phát hành vào tháng 11 năm 2020 mang đến nhiều cải tiến và tính năng mới nổi bật như: JIT (Just-In-Time) Compiler cho phép mã PHP được biên dịch thành mã máy tại thời điểm chạy, giúp cải tiến hiệu suất của ứng dụng; Typed Properties cho phép định nghĩa kiểu dữ liệu cho thuộc tính của lớp, giúp kiểm tra kiểu dữ liệu tốt hơn, tránh lỗi trong quá trình phát triển; Union Types có thể định nghĩa kiểu dữ liệu là một liên kết của nhiều kiểu khác nhau; Match Expression là phiên bản nâng cấp của Switch hỗ trợ kiểm tra giá trị trả về kết quả dựa trên điều kiện; hỗ trợ thêm metadata vào các phần tử như lớp, phương thức, thuộc tính.

4.1.3. Cách xử lý trang web động với PHP

Trang web động (Dynamic Web Page) là loại trang web có nội dung được tạo ra một cách linh hoạt dựa trên các yêu cầu từ phía người dùng hoặc dữ liệu từ cơ sở dữ liệu. Khác với trang web tĩnh (HTML), trang web động sử dụng ngôn ngữ server-side như PHP để xử lý và trả về kết quả phù hợp với từng tình huống.

PHP (Hypertext Preprocessor) là một ngôn ngữ lập trình phía máy chủ (server-side scripting), được nhúng vào HTML để tạo ra các trang web động. Khi người dùng truy cập vào một trang PHP, máy chủ sẽ thực thi các đoạn mã PHP, trả về kết quả dưới dạng HTML để trình duyệt hiển thị.

Quy trình xử lý trang web động với PHP

- Người dùng gửi yêu cầu (Request): Khi người dùng truy cập một trang web có đuôi .php (ví dụ: index.php), trình duyệt sẽ gửi yêu cầu đến máy chủ web (Apache/Nginx).
- Máy chủ web nhận yêu cầu: Máy chủ web (đã cài đặt PHP) nhận yêu cầu và xác định rằng đây là một tập tin PHP cần xử lý.
- PHP Engine xử lý mã PHP: Máy chủ chuyển tập tin PHP cho trình thông dịch PHP (PHP Engine). PHP Engine thực thi từng dòng mã PHP, kết nối với cơ sở dữ liệu (nếu cần) và xử lý logic.
- Trả kết quả về máy chủ web: Sau khi xử lý xong, PHP Engine trả về kết quả dưới dạng HTML thuần (không chứa mã PHP gốc).

- Máy chủ web gửi kết quả về trình duyệt: Máy chủ web gửi nội dung HTML đã được xử lý về trình duyệt của người dùng. Trình duyệt hiển thị nội dung HTML, CSS và JavaScript như một trang web thông thường.

Ưu điểm của trang web động với PHP

- Tương tác với CSDL: PHP có thể kết nối với MySQL, PostgreSQL, SQL Server... để truy vấn và hiển thị dữ liệu động.

- Xử lý dữ liệu form: PHP có thể nhận dữ liệu từ form HTML (\$_GET, \$_POST) và xử lý (đăng nhập, đăng ký, tìm kiếm...).

- Tích hợp dễ dàng: PHP có thể kết hợp với HTML, CSS, JavaScript và các thư viện/framework (Laravel, Symfony...).

- Hiệu suất tốt: PHP được tối ưu hóa cho web, hỗ trợ caching (OPcache) để tăng tốc độ xử lý.

Bảng 5. 1 So sánh với trang web tĩnh

Tiêu chí	Trang web tĩnh (HTML)	Trang web động (PHP)
Nội dung	Cố định, không thay đổi	Thay đổi theo yêu cầu người dùng
Xử lý dữ liệu	Không có	Hỗ trợ xử lý form, CSDL, session
Tương tác	Không	Có (đăng nhập, giỏ hàng, bình luận)
Tốc độ tải trang	Nhanh (do không cần xử lý server)	Phụ thuộc vào logic và CSDL

4.1.4. Các phần mềm ứng dụng Web

Một số framework PHP phổ biến mà các nhà phát triển web thường sử dụng:

- Laravel: Laravel là một trong những framework PHP phổ biến nhất hiện nay. Nó được thiết kế để giúp bạn xây dựng ứng dụng web nhanh chóng và dễ dàng.

- Symfony: Symfony là một framework mạnh mẽ, có khả năng mở rộng và module hóa. Nó được sử dụng rộng rãi trong các dự án lớn và phức tạp.

- Yii 2: Yii 2 là một framework nhanh, linh hoạt và dễ sử dụng. Nó hỗ trợ việc phát triển ứng dụng web một cách hiệu quả.

- CakePHP: là một framework đơn giản và dễ tiếp cận. Nó giúp bạn xây dựng ứng dụng nhanh chóng và tuân thủ các quy ước phát triển.

- CodeIgniter: là một framework nhẹ, phù hợp cho các dự án nhỏ và trung bình. Nó dễ dàng học và sử dụng.

- Zend Framework: hoặc Laminas Project là một framework mạnh mẽ, phù hợp cho các ứng dụng doanh nghiệp.

- Phalcon: Phalcon là một framework nhanh và nhẹ, được viết bằng C để tối ưu hóa hiệu suất.

- Slim Framework: Slim là một micro-framework nhẹ, thích hợp cho việc xây dựng các API và ứng dụng web nhỏ.

4.2. Cài đặt môi trường và công cụ

Trước khi bắt đầu lập trình web với PHP, chúng ta cần cài đặt môi trường phát triển phù hợp. Trong bài giảng này, chúng ta sẽ hướng dẫn cài đặt và cấu hình hai công cụ chính để phát triển ứng dụng web với PHP:

a) Visual Studio Code (VS Code) là phần mềm ứng dụng có thể chạy trên Windows, Linux và MacOS, gọn nhẹ, mạnh mẽ với nhiều Extension hỗ trợ.

b) XAMPP Server là hệ thống server web mã nguồn mở, miễn phí bao gồm Apache, MySQL và trình thông dịch cho PHP và Perl.

4.2.1. Cài đặt Visual Studio Code

Bước 1. Tải phần mềm Visual Studio Code:

- Truy cập vào trang web chính thức của Visual Studio Code:

<https://code.visualstudio.com/download>

- Tải xuống phiên bản phù hợp với hệ điều hành (Windows, macOS hoặc Linux).

- Tiến hành cài đặt VS Code theo các bước được hướng dẫn trong lúc cài (tương đối đơn giản).

Bước 2. Cài đặt các Extensions hữu ích cho PHP:

- Mở VS Code sau khi cài đặt xong.

- Di chuyển đến Extensions (hoặc nhấn tổ hợp phím Ctrl+Shift+X trên Windows/Linux hoặc Cmd+Shift+X trên macOS).

- Cài đặt các Extensions sau:

a) PHP Intelephense là một công cụ mạnh mẽ giúp nhắc code và hoàn thành mã nguồn một cách tự động. Để sử dụng chức năng này, chúng ta nhấn tổ hợp phím Ctrl + Space để sử dụng Editor hiển thị tất cả các Function PHP, đây là tính năng khá hữu ích khi lập trình PHP trên VS Code

b) PHP Debug là công cụ giúp debug ứng dụng PHP trong môi trường VS Code

c) PHP DocBlocker là công cụ hỗ trợ tạo DocBlocks cho mã PHP, giúp tăng khả năng đọc hiểu mã nguồn.

d) Bracket Pair Colorizer: Giúp nhận diện các cặp ngoặc một cách dễ dàng.

4.2.2. Cài đặt XAMPP Server

Bước 1. Tải phần mềm XAMPP:

- Truy cập vào trang chính thức của Xampp

<https://www.apachefriends.org/download.html>

- Tải xuống phiên bản phù hợp với hệ điều hành (Windows, Linux, macOS)

- Tiến hành cài đặt XAMPP theo các hướng dẫn trong quá trình cài.

Bước 2. Cấu hình XAMPP:

- Khởi động XAMPP Control Panel sau khi cài đặt xong.
- Start Apache và MySQL bằng cách nhấn vào nút “Start” phía bên trái của từng dịch vụ.

- Nếu cần, cấu hình lại cổng cho Apache (mặc định là cổng 80) và MySQL (mặc định là cổng 3306) tùy theo yêu cầu.

Bước 3. Kiểm tra XAMPP:

- Mở trình duyệt web và truy cập vào <http://localhost/> để đảm bảo rằng XAMPP đã hoạt động chính xác.

- Để kiểm tra PHP đã được cài đặt đúng, truy cập vào <http://localhost/phpmyadmin/> để mở giao diện quản lý MySQL.

* Chạy PHP script bằng command prompt

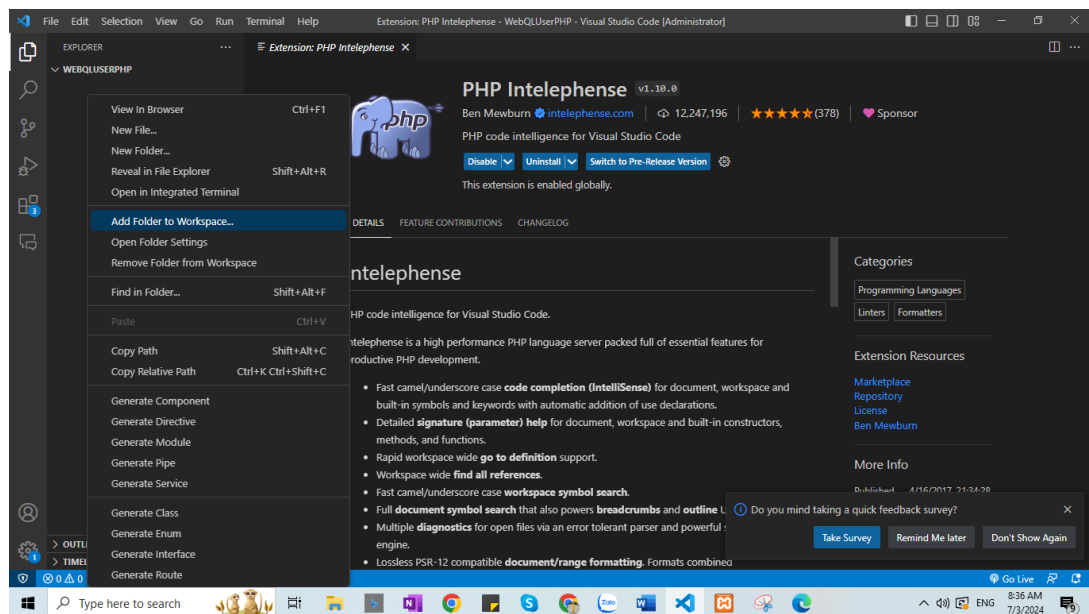
```
$ php test.php
```

* Hướng dẫn trên khai dự án PHP trên server cục bộ với VS Code và Xampp Server

- Sau khi cài đặt XAMPP server xong thì sẽ có thư mục xampp trong ổ đã đĩa cài đặt (thường là ổ C:/).

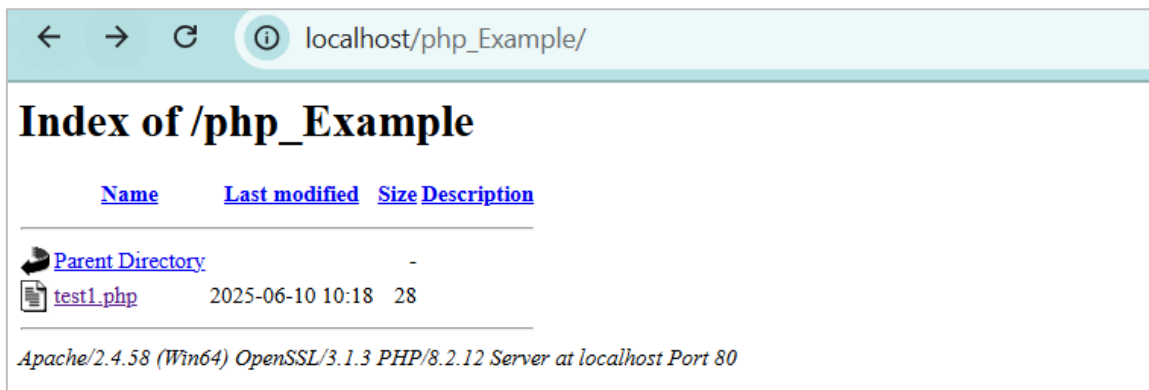
- Tạo một thư mục để làm việc với dự án website PHP trong thư mục htdocs của xampp theo đường dẫn sau: C:\xampp\htdocs.

- Mở VS Code lên, nhấn chuột phải vào vùng trống trong Explorer, chọn Add Folder to Workspace..., sau đó chọn đến thư mục của dự án như hình sau:



Hình 4. 1 Mở file php trong VS Code

- Tạo tập tin php (*.php) và viết code cho dự án.
- Để chạy dự án lên, chúng ta mở trình duyệt web và truy cập vào <http://localhost/<tên thư mục dự án>>, chọn tập tin php cần chạy nó.

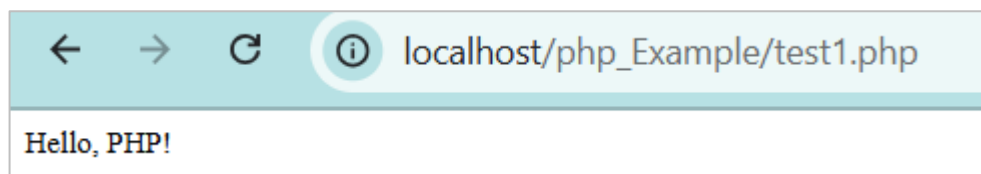


* Thực hành:

Hãy tạo một file PHP đơn giản và mở nó trong VS Code.

Viết mã PHP đơn giản để hiển thị chuỗi “Hello, PHP!”.

```
<?php
    echo "Hello, PHP!";
?>
```



Chú ý: Chạy file PHP này trên XAMPP để đảm bảo môi trường lập trình đã được cài đặt đúng.

4.3. Kỹ năng PHP cơ bản

4.3.1. Cú pháp PHP

Trước khi bắt đầu lập trình với bất kỳ ngôn ngữ nào, thì cú pháp cơ bản và cấu trúc chương trình là phần rất quan trọng. Trong phần này, chúng tôi sẽ giới thiệu các cú pháp PHP cơ bản như: thẻ php, comment, lệnh print, lệnh echo, và so sánh lệnh print với echo.

a) Thẻ PHP

Thẻ PHP được sử dụng phổ biến nhất là bắt đầu thẻ “<?php” và kết thúc bằng “?>”

```
<?php ..... ?>
```

Ngoài ra, chúng ta có thể sử dụng thẻ HTML script có dạng như sau:

```
<script language="PHP">.....</script>
```

b) Hiển thị văn bản lên trình duyệt với lệnh **print** và lệnh **echo**

Trong PHP chúng ta có thể sử dụng hai lệnh **print** và **echo** để in ra màn hình một chuỗi nào đó.

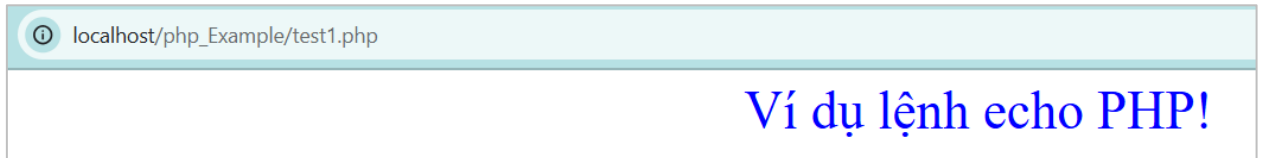
```
<?php
    echo "<p align='center'><font color='blue' size='25px'>Ví dụ lệnh echo PHP!</font></p>";
```

```
?>
```

hoặc:

```
<?php
    print "<p align='center'><font color='red' size='25px'>Ví dụ lệnh print PHP!</font></p>";
?>
```

Kết quả:



Điểm khác nhau giữa print và echo là:

(1) Lệnh **print** là một hàm số, khi được thực thi sẽ trả về kết quả là 1, nếu không thì trả kết quả 0. Do đó, chúng ta có thể gán kết quả của lệnh **print** này cho một biến, còn với lệnh **echo** thì không được.

```
<?php
    $test1 = print 'abcd';
    $test2 = echo 'cdef'; //sai
?>
```

(2) Lệnh **print** chỉ có thể sử dụng với một tham số, trong khi lệnh **echo** có thể được dùng với nhiều tham số.

```
<?php
    echo 'k','i','m'; //dùng với 3 tham số
    echo ('h'),('u'); //dùng được cả dấu ngoặc cho từng tham số

    print 'o'; //đúng
    print 'n','g'; //sai
?>
```

c) Comment trong PHP

Một comment là ghi chú cho người đọc code và nó sẽ bị lược bỏ trước khi hiển thị kết quả chương trình. Có hai kiểu comment trong PHP:

- Comment một dòng sử dụng ký hiệu “//” hoặc “#”.
- Comment nhiều dòng sử dụng ký hiệu “/* ...*/”.

```
<?php
#-----
/*
    Title: Bài giảng Lập trình Web PHP
    Version: 1.0
    Author: Trần Kim Hương
    Created: 05/07/2024
*/
```

```
*/  
#-----  
// print "Ví dụ minh họa comment";  
?>
```

d) Lệnh trong PHP được kết thúc bởi dấu chấm phẩy “;”.

d) PHP phân biệt kiểu chữ hoa thường, PHP bỏ qua mọi khoảng trắng thừa trong câu lệnh.

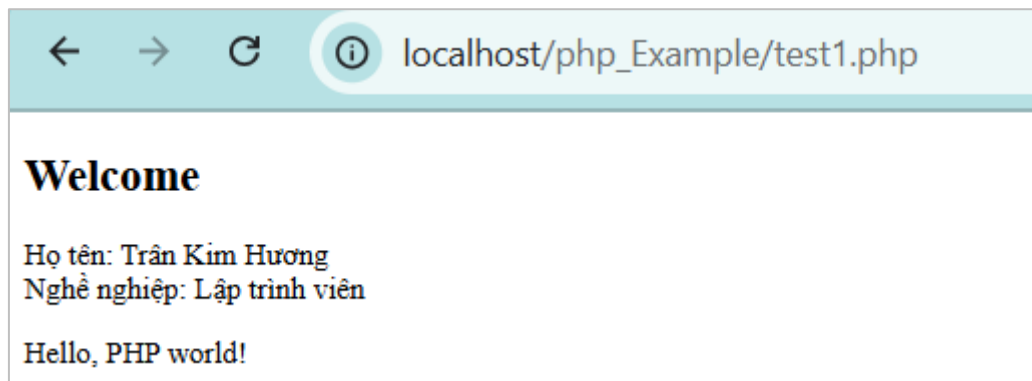
e) Khối lệnh trong PHP được bao quanh bởi cặp ngoặc “{...}”.

4.3.2. Nhúng mã PHP vào mã HTML

Khi viết mã PHP, chúng ta thường mở thẻ PHP trước thẻ HTML đầu tiên. Thẻ PHP này chứa các câu lệnh xử lý một số thao tác ban đầu và khởi tạo những biến được sử dụng sau này. Sau đó, các thẻ PHP ngắn hơn được sử dụng để hiển thị dữ liệu tại các phần khác nhau của tài liệu HTML, xem ví dụ sau:

```
<?php  
// lấy dữ liệu từ yêu cầu  
$name = "Trần Kim Hương";  
$job = "Lập trình viên";  
$message = "Hello, PHP world!";  
?>  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Nhúng PHP vào HTML</title>  
</head>  
<body>  
  <h2>Welcome</h2>  
  <div>Họ tên: <?php echo $name ?> </div>  
  <div>Nghề nghiệp: <?php echo $job ?> </div>  
  <?php  
    echo "<p>$message</p>";  
  ?>  
</body>  
</html>
```

Kết quả:



Ngoài ra, chúng ta có thể nhúng đoạn code của PHP vào cặp thẻ `<script>...</script>`, như sau:

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Nhúng PHP vào HTML</title>
  <script>
    <?php
      echo "alert('Hello from PHP!')";
    ?>
  </script>
</head>
```

4.3.3. Tạo trang web PHP đầu tiên

Đầu tiên, chúng ta sẽ tạo tập tin php với tên là `first_web.php` trong thư mục `DemoWebPHP` của `htdocs` xampp server, sau đó viết code vào như sau:

```
<?php
  $name = "Trần Kim Hương";
  $currentDate = date('d/m/Y');
?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Trang web PHP cơ bản</title>
</head>
<body>
  <div>
    <h1>Chào mừng đến với trang web PHP cơ bản!</h1>
    <p>Ngày hôm nay là <?php echo $currentDate; ?></p>
    <p>Xin chào,
      <?php echo $name; ?>! Hôm nay là ngày <?php echo $currentDate; ?>.
    </p>
```

```
</div>
</body>
</html>
```

Sau đó chạy chương trình lên với XAMPP sever (như đã hướng dẫn ở mục 4.2), sẽ có kết quả như sau:



Hình 4. 2 Giao diện trang web php đầu tiên

4.4. Biến, phép toán, kiểu dữ liệu

4.4.1. Biến trong PHP

Trong PHP, biến là một định danh dùng để lưu trữ vị trí của dữ liệu trong ô nhớ máy tính. Dữ liệu nằm trong ô nhớ, mỗi khi gọi đến biến thì máy tính sẽ trở đến ô nhớ đó. Biến là bộ nhớ tạm thời được sử dụng để lưu trữ dữ liệu tạm thời.

Cú pháp:

```
$ten_bien;
```

```
$ten_bien = giá trị;
```

Biến bắt đầu bằng dấu đô la “\$” và tiếp theo là tên biến theo quy tắc đặt tên biến:

- Tên biến phải bắt đầu bằng chữ cái hoặc dấu gạch dưới;
- Tên biến được phép chứa chữ cái, chữ số, gạch dưới;
- Trong PHP phân biệt chữ hoa và chữ thường trong tên biến;
- Khi đặt tên biến không cần khai báo kiểu dữ liệu;

- Trong PHP mặc định đặt tên biến và hàm theo quy chuẩn **Snake_case** là quy tắc đặt tên biến mà trong đó mọi từ trong tên biến hoặc hàm được phân cách bằng dấu gạch dưới “_” và tất cả đều viết thường.

Ví dụ:

```
<?php
$first_name = "Trần Kim";
$last_name = "Hương";
$date_of_birth = "2024-01-01";

function calculate_age($birth_date) {
```

```
// implementation
}
?>
```

* Xuất dữ liệu:

```
echo <tên biến, tên hàm, chuỗi, số>;
```

* Debug dữ liệu:

- var_dump(\$opp): trả về kiểu dữ liệu và giá trị;
- print_r(\$opp): dùng để in ra mảng và đối tượng;

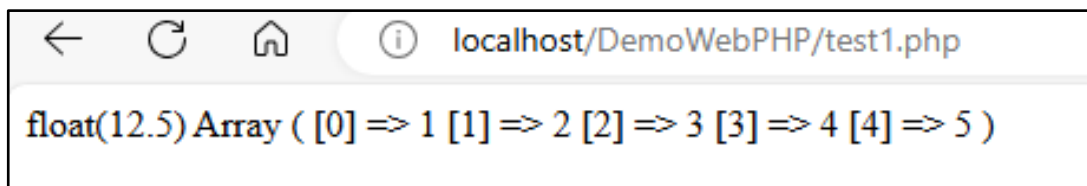
Ví dụ:

```
<?php
$bien_so_1 = 12.5;
$arr = [1,2,3,4,5];

var_dump($bien_so_1);

print_r($arr);
?>
```

Kết quả:



* Cách nối biến trong PHP

Cú pháp:

```
$bien_so_1 . $bien_so_2;
```

Ví dụ 1:

```
$bien_so_1 = 'Kim';
$bien_so_2 = 'Hương';
echo $bien_so_1 . ' ' . $bien_so_2;
```

Ví dụ 2:

```
$bien_so_1 = 'Kim';
$bien_so_2 = 'Trần' . $bien_so_1 . 'Hương';
$bien_so_2 = "Trần $bien_so_1 Hương";
```

Ví dụ 3:

```
$bien_1 = 'Lập trình';
$bien_2 = 'PHP';
echo '<ul class="format">
<li>' . $bien_1 . '</li>
```



```
<li>'.$bien_2.'</li>
</ul>;
```

4.4.2. Hằng số trong PHP

Hằng số trong PHP là tên hoặc mã định danh không thể thay đổi trong khi thực thi chương trình. Các hằng số có thể được định nghĩa theo 2 cách:

- Sử dụng hàm **define()**;
- Sử dụng từ khóa **const**;

Khai báo:

```
define('ten_hang', 'giá trị');
const ten_hang = giá trị;
```

Sử dụng:

```
Ten_hang;
```

Cách đặt tên hằng theo nguyên tắc chung như: tên hằng chứa các chữ cái, chữ số, dấu gạch dưới; bắt đầu bằng chữ cái hoặc gạch dưới; phân biệt chữ hoa, chữ thường; nên đặt tên theo quy chuẩn **Snake_case** nhưng viết hoa và có thể bắt đầu bằng dấu gạch dưới (ví dụ: **_WEB_HOST_ROOT**).

Ví dụ:

```
define('_WEB_HOST_ROOT', 'admin');
const _WEB_HOST_ROOT = 'admin';

echo _WEB_HOST_ROOT;
```

4.4.3. Kiểu dữ liệu trong PHP

PHP hỗ trợ nhiều kiểu dữ liệu khác nhau, trong đó có sáu kiểu dữ liệu cơ bản được trình bày tóm tắt trong bảng 5.1

Bảng 5. 2 Kiểu dữ liệu cơ bản trong PHP

Kiểu dữ liệu	Mô tả
Kiểu số nguyên (integer/long)	<ul style="list-style-type: none"> - Kiểu dữ liệu đại diện cho các số nguyên, không có phần thập phân. - Khai báo: \$ten_bien = so_nguyen; (ví dụ: \$x=10;) - Ép kiểu: (int)\$ten_bien; - Các hàm liên quan: is_int(\$ten_bien) hoặc is_integer(\$ten_bien), is_long(\$ten_bien).
Kiểu số thực (float/ double)	<ul style="list-style-type: none"> - Kiểu dữ liệu đại diện cho các số thực, có phần thập phân. - Khai báo: \$ten_bien = số_thực; (ví dụ: \$x=2.123) - Ép kiểu: (float)\$ten_bien; - Các hàm liên quan: is_float(\$ten_bien), is_double(\$ten_bien), is_real(\$ten_bien).
boolean	<ul style="list-style-type: none"> - Kiểu dữ liệu chỉ có 2 giá trị TRUE hoặc FALSE - Khai báo: \$ten_bien = true hoặc false (ví dụ: \$is_admin = true;)

	<ul style="list-style-type: none"> - Ép kiểu: (bool)\$ten_bien; hoặc (boolean) - Các ký tự: 0, trống, null để quy về false, ngược lại là true - Hàm liên quan: is_bool(\$ten_bien)
Kiểu chuỗi (string)	<ul style="list-style-type: none"> - Kiểu đại diện cho các dãy ký tự - Khai báo: \$ten_bien = 'chuỗi'; hoặc \$ten_bien = "chuỗi" - Ép kiểu: (string)\$ten_bien; - Gán chuỗi cho hằng: define('_AGENCY', 'Hương') - Nếu chuỗi hiển thị cần có dấu nháy đơn hoặc kép thì phải có ký tự escape (\) phía trước hoặc là sử dụng hai dấu ngược nhau. ví dụ: echo "Hương học \"lập trình\" PHP" hoặc echo 'Hương học "lập trình" PHP' - Các hàm liên quan: is_string(\$string); strlen((\$string); strpos((\$string, \$substring); str_replace((\$search, \$replace, \$string) - Ghi chú: chuỗi trong dấu nháy kép (") có phân tích biến và ký tự đặc biệt, chuỗi trong dấu nháy đơn (') không phân tích biến nên cải thiện hiệu suất xử lý hơn → sử dụng nháy kép cho chuỗi có chứa biến và sử dụng dấu nháy đơn cho chuỗi không chứa biến.
Kiểu mảng (array)	<ul style="list-style-type: none"> - Kiểu dữ liệu đại diện cho một tập hợp các giá trị, có thể là các kiểu dữ liệu khác nhau. - Khai báo: \$ten_bien = array(); hoặc [] - Ví dụ: \$fruits = array("Apple", "Banana", "Cherry"); hoặc \$numbers = [1, 2, 3, 4, 5]; hoặc \$arr = {'key1' => 'HTML', 'key2' => 'JS', 'key3' => 'PHP', 'key4' => 'CSS'}; - Ép kiểu: (array)\$ten_bien; (ví dụ: \$bien= '1,2,3 ' → \$bien = (array)\$bien) - Xem cấu trúc mảng: echo'<pre>'; print_r(\$array); echo'</pre>'; - Các hàm liên quan: is_array(\$ten_bien); count(\$fruits); array_merge(array1, array2); array_push(\$fruits, "mango", "watermelon");
Kiểu NULL	<ul style="list-style-type: none"> - Kiểu dữ liệu chỉ có một giá trị duy nhất là NULL, đại diện cho giá trị không có. - Khai báo: \$ten_bien = null; - Ép kiểu sang INT => 0; - Ép kiểu sang STRING => Rỗng; - Ép kiểu sang BOOLEAN => False - Các hàm liên quan: is_null(\$ten_bien)
Kiểu đối tượng (object)	<ul style="list-style-type: none"> - Kiểu dữ liệu đại diện cho một thực thể, chứa các thuộc tính và phương thức. - Các hàm liên quan: is_object(); get_class(); method_exists()

4.4.4. Toán tử và Biểu thức trong PHP

Biểu thức trong PHP là một tổ hợp của các toán hạng và toán tử. Toán hạng có thể là biến, giá trị, hoặc các kết quả của các biểu thức khác. Toán tử thực hiện các thao tác như cộng, trừ, nhân, chia, so sánh, và nhiều thao tác khác. Dưới đây là các loại toán tử cơ bản trong PHP cùng với ví dụ minh họa.

Bảng 5. 3 Toán tử số học

Toán tử	Mô tả	Ví dụ
+	Phép cộng hai số	$\$A + \$B : 25 + 30 = 55$
-	Phép trừ	$\$A - \$B : 25 - 30 = -5$
*	Phép nhân	$\$A * \$B : 25 * 30 = 750$
/	Phép chia	$\$A / \$B : 25 / 30 = 0.833333333333$
**	Lũy thừa	$\$x=2; \$y=3;$ $\$x**\$y: 2^3 = 8$
%	Phép chia lấy dư. Phần dư của phép chia hai số nguyên	$\$A / \$B : 25 \% 30 = 25$
++	Phép toán tăng thêm 1 vào biến	$\$A++$ kết quả $\$A = 26$
--	Phép toán giảm đi giá trị 1	$\$B--$ kết quả $\$B = 29$

Bảng 5. 4 Toán tử so sánh

Toán tử	Mô tả	Ví dụ
==	Kiểm tra nếu 2 toán hạng bằng nhau hay không. Nếu bằng thì điều kiện là true.	$(\$a == \$b)$ là false.
===	So sánh giá trị giữa các biến và hằng đúng theo giá trị và kiểu dữ liệu của nó.	$\$a = '123'; \$b = 123$ $\$a == \b là true; $\$a === \b là false (không cùng kiểu dữ liệu)
!=	Kiểm tra 2 toán hạng có giá trị khác nhau hay không. Nếu không bằng thì điều kiện là true.	$(\$a != \$b)$ là true.
>	Kiểm tra nếu toán hạng bên trái có giá trị lớn hơn toán hạng bên phải hay không. Nếu lớn hơn thì điều kiện là true.	$(\$a > \$b)$ là false.
<	Kiểm tra nếu toán hạng bên trái nhỏ hơn toán hạng bên phải hay không. Nếu nhỏ hơn thì là true.	$(\$a < \$b)$ là true.
>=	Kiểm tra nếu toán hạng bên trái có giá trị lớn hơn hoặc bằng giá trị của toán hạng bên phải hay không. Nếu đúng là true.	$(\$a >= \$b)$ là false.
<=	Kiểm tra nếu toán hạng bên trái có giá trị nhỏ hơn hoặc bằng toán hạng bên phải hay không. Nếu đúng là true.	$(\$a <= \$b)$ là true.

Bảng 5. 5 Toán tử gán

Toán tử	Mô tả	Ví dụ
=	Toán tử gán đơn giản. Gán giá trị toán hạng bên phải cho toán hạng trái	$\$c = \$a + \$b$ sẽ gán giá trị của $\$a + \b vào trong $\$c$
+=	Thêm giá trị toán hạng phải tới toán hạng trái và gán giá trị đó cho toán hạng trái	$\$c += \a là tương đương với $\$c = \$c + \$a$
-=	Trừ đi giá trị toán hạng phải từ toán hạng trái và gán giá trị này cho toán hạng trái	$\$c -= \a là tương đương với $\$c = \$c - \$a$

<code>*=</code>	Nhân giá trị toán hạng phải với toán hạng trái và gán giá trị này cho toán hạng trái	<code>\$c *= \$a</code> là tương đương với <code>\$c = \$c * \$a</code>
<code>/=</code>	Chia toán hạng trái cho toán hạng phải và gán giá trị này cho toán hạng trái	<code>\$c /= \$a</code> là tương đương với <code>\$c = \$c / \$a</code>
<code>%=</code>	Lấy phần dư của phép chia toán hạng trái cho toán hạng phải và gán cho toán hạng trái	<code>\$c %= \$a</code> là tương đương với <code>\$c = \$c % \$a</code>
<code>.=</code>	Ghép chuỗi	<code>\$bien = 'Kim ';</code> <code>\$bien .= 'Huong ';</code> <code>echo \$bien . '
';</code>

Bảng 5. 6 Toán tử luận lý logic

Toán tử	Mô tả	Ví dụ
and	Được gọi là toán tử Logic AND. Nếu cả hai toán hạng là true thì điều kiện trở thành true	<code>\$a = 10; \$b = 20;</code> <code>(\$a and \$b)</code> là true.
or	Được gọi là toán tử Logic OR. Nếu một trong hai toán hạng là đúng thì điều kiện trở thành true	<code>(\$a or \$b)</code> là true.
&&	Được gọi là toán tử Logic AND. Nếu cả hai toán hạng là true thì điều kiện trở thành true	<code>(\$a && \$b)</code> là true.
	Được gọi là toán tử Logic OR. Nếu một trong hai toán hạng là đúng thì điều kiện trở thành true	<code>(\$a \$b)</code> là true.
!	Được gọi là toán tử Logic NOT. Sử dụng để đảo ngược trạng thái logic của toán hạng. Nếu điều kiện là true thì toán tử Logic NOT sẽ cho kết quả là false	<code>!(\$a && \$b)</code> là false.

4.5. Cấu trúc điều khiển

Trong lập trình, các cấu trúc lệnh là những khối mã điều khiển luồng thực thi của chương trình. Chúng giúp chúng ta đưa ra các quyết định, lặp lại các hành động và tổ chức mã nguồn một cách có hệ thống và dễ hiểu hơn. PHP, giống như các ngôn ngữ lập trình khác, cung cấp nhiều loại cấu trúc lệnh khác nhau để giúp xây dựng các ứng dụng web phức tạp và linh hoạt.

4.5.1. Cấu trúc điều kiện trong PHP

Cấu trúc điều kiện cho phép chương trình thực hiện các hành động khác nhau dựa trên các điều kiện khác nhau. Điều này rất quan trọng để kiểm soát luồng chương trình và đưa ra các quyết định hợp lý dựa trên dữ liệu hoặc trạng thái hiện tại.

5.5.1.1 Cấu trúc IF

Cú pháp:

<pre>if (điều kiện) { // câu lệnh điều kiện đúng }</pre>
--

Ví dụ:

```
<?php
    $age = 20;

    if ($age >= 18) {
        echo "You are an adult.";
    }
?>
```

4.5.1.2. Cấu trúc IF...ELSE

Cú pháp:

```
if (điều kiện) {
    // câu lệnh điều kiện đúng
} else {
    // câu lệnh điều kiện sai
}
```

Ví dụ: Kiểm tra đăng nhập

Tạo file login.php

```
<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <title>Đăng nhập</title>
</head>
<body>
    <h2>Form Đăng nhập</h2>
    <form method="post" action="">
        <label for="username">Tên đăng nhập:</label>
        <input type="text" name="username" id="username" required><br><br>
        <label for="password">Mật khẩu:</label>
        <input type="password" name="password" id="password" required><br><br>
        <input type="submit" value="Đăng nhập">
    </form>

    <?php
    // Kiểm tra nếu form đã được gửi
    if ($_SERVER["REQUEST_METHOD"] === "POST") {
        // Nhận dữ liệu từ form
        $username = $_POST["username"];
        $password = $_POST["password"];
        // So sánh với thông tin mẫu
        if ($username === "admin" && $password === "12345") {
            echo "<p style='color:green;'>Đăng nhập thành công!</p>";
        } else {
            echo "<p style='color:red;'>Sai tên đăng nhập hoặc mật khẩu.</p>";
        }
    }
}
```

```

    }
  }
  ?>
</body>
</html>

```

Kết quả:



4.5.1.3. Cấu trúc IF...ELSEIF...ELSE

Cú pháp:

```

if (điều kiện1) {
    // câu lệnh điều kiện1 đúng
} elseif (điều kiện2) {
    // câu lệnh điều kiện2 đúng
} else {
    // câu lệnh nếu tất cả các điều kiện trên đều sai
}

```

Ví dụ:

```

<?php
    $score = 85;

    if ($score >= 90) {
        echo "Grade: A";
    } elseif ($score >= 80) {
        echo "Grade: B";
    } elseif ($score >= 70) {
        echo "Grade: C";
    } else {
        echo "Grade: D";
    }
?>

```

4.5.1.4. Cấu trúc SWITCH...CASE

Cấu trúc *switch...case* được sử dụng khi bạn có nhiều điều kiện cần kiểm tra, thay vì dùng nhiều câu lệnh *if...elseif...else*.

Cú pháp:

```

switch (biểu thức) {
    case giá_trị1:
        // câu lệnh nếu biểu thức == giá_trị1
        break;
    case giá_trị2:
        // câu lệnh nếu biểu thức == giá_trị2
        break;
    // ...
    default:
        // câu lệnh nếu không có case nào khớp
}

```

Ví dụ:

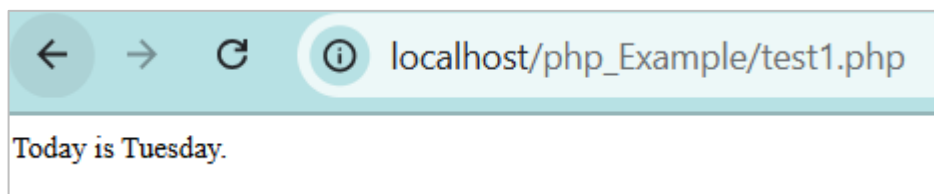
```

<?php
$day = "Tuesday";

switch ($day) {
    case "Monday":
        echo "Today is Monday.";
        break;
    case "Tuesday":
        echo "Today is Tuesday.";
        break;
    case "Wednesday":
        echo "Today is Wednesday.";
        break;
    default:
        echo "Invalid day.";
}
?>

```

Kết quả:



4.5.2. Cấu trúc vòng lặp trong PHP

Vòng lặp là các cấu trúc lệnh cho phép chúng ta thực hiện một đoạn mã lệnh nhiều lần, dựa trên một điều kiện hoặc một tập hợp các phần tử. Chúng rất hữu ích khi cần lặp lại các tác vụ giống nhau trên một tập hợp dữ liệu hoặc thực hiện các phép toán lặp lại.

4.5.2.1. Vòng lặp FOR

Cú pháp:

```

for (khởi_tạo; điều_kiện; bước_nhảy) {

```

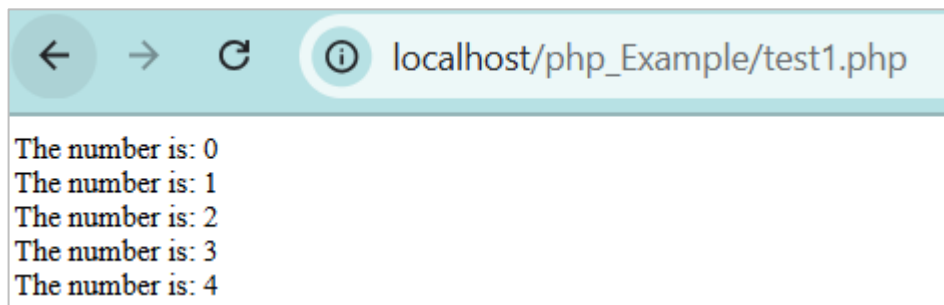
```
// Các câu lệnh thực thi trong mỗi lần lặp
```

```
}
```

Ví dụ 1:

```
<?php
for ($i = 0; $i < 5; $i++) {
    echo "The number is: $i <br>";
}
?>
```

Kết quả:

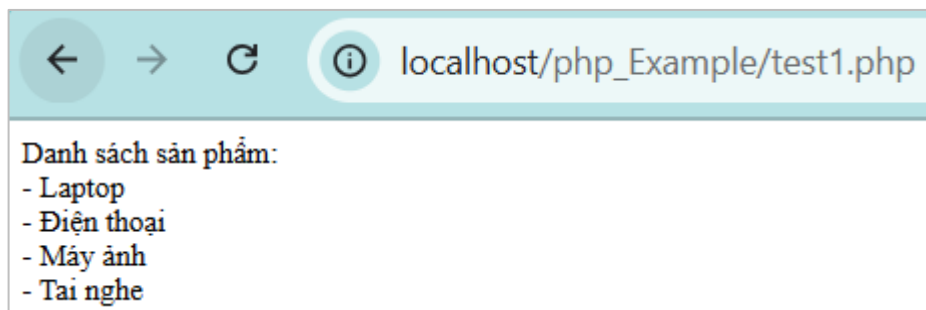


Ví dụ 2: Hiển thị danh sách sản phẩm

```
<?php
// Tạo mảng danh sách sản phẩm
$products = ["Laptop", "Điện thoại", "Máy ảnh", "Tai nghe"];

// Hiển thị danh sách sản phẩm
echo "Danh sách sản phẩm:<br>";
foreach ($products as $product) {
    echo "- $product<br>";
}
?>
```

Kết quả:



4.5.2.2. Vòng lặp WHILE

Cú pháp:

```
while (điều_kiện) {
```



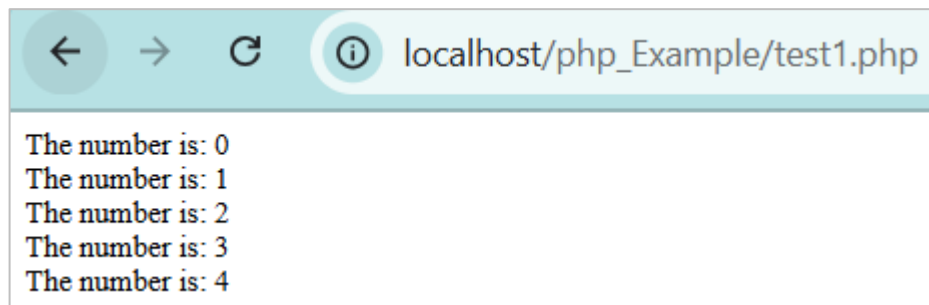
```
// Các câu lệnh thực thi trong mỗi lần lặp
```

```
}
```

Ví dụ:

```
<?php
$i = 0;
while ($i < 5) {
    echo "The number is: $i <br>";
    $i++;
}
?>
```

Kết quả:



4.5.2.3. Vòng lặp DO...WHILE

Cú pháp:

```
do {
    // Các câu lệnh sẽ được thực thi trong mỗi lần lặp
} while (điều_kiện);
```

Ví dụ:

```
<?php
$i = 0;
do {
    echo "The number is: $i <br>";
    $i++;
} while ($i < 5); ?>
```

4.5.2.4. Vòng lặp FOREACH

Cú pháp:

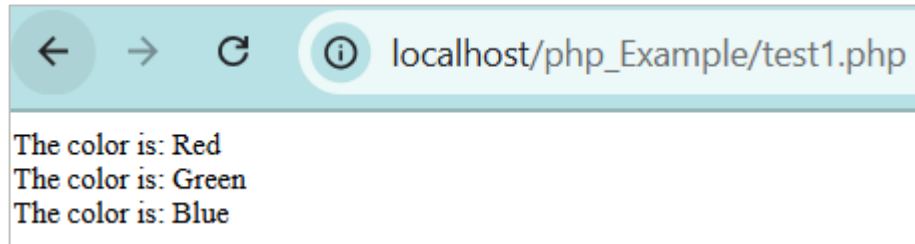
```
foreach ($array as $value) {
    // Các câu lệnh sẽ được thực thi trong mỗi lần lặp
}
```

Ví dụ 1:

```
<?php
    $colors = array("Red", "Green", "Blue");

    foreach ($colors as $color) {
        echo "The color is: $color <br>";
    }
?>
```

Kết quả:



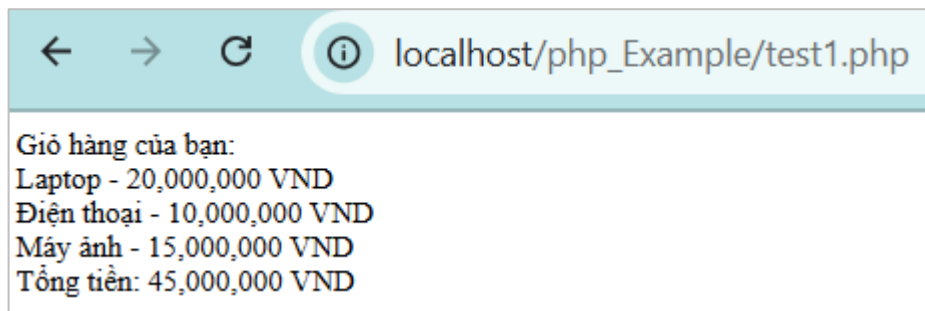
Ví dụ 2: Sử dụng mảng để quản lý giỏ hàng

```
<?php
    // Tạo giỏ hàng (mảng kết hợp)
    $cart = [
        ["name" => "Laptop", "price" => 20000000],
        ["name" => "Điện thoại", "price" => 10000000]
    ];

    // Thêm sản phẩm vào giỏ hàng
    $newProduct = ["name" => "Máy ảnh", "price" => 15000000];
    array_push($cart, $newProduct);

    // Hiển thị giỏ hàng
    $total = 0;
    echo "Giỏ hàng của bạn:<br>";
    foreach ($cart as $item) {
        echo $item['name'] . " - " . number_format($item['price']) . " VND<br>";
        $total += $item['price'];
    }
    echo "Tổng tiền: " . number_format($total) . " VND";
?>
```

Kết quả:



4.5.3. Lệnh BREAK – CONTINUE – DIE() – EXIT()

Trong PHP, các lệnh *break*, *continue*, *die()*, và *exit()* được sử dụng để điều khiển luồng thực thi của chương trình.

5.5.3.1. Lệnh BREAK

Lệnh ***break*** được sử dụng để thoát ra khỏi vòng lặp (for, foreach, while, do...while) hoặc thoát ra khỏi một câu lệnh switch.

Ví dụ:

```
<?php
for ($i = 0; $i < 10; $i++) {
    if ($i == 5) {
        break;
    }
    echo "The number is: $i <br>";
}
?>
```

* Kết quả:

The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4

4.5.3.2. Lệnh CONTINUE

Lệnh ***continue*** được sử dụng để bỏ qua các phần còn lại của vòng lặp hiện tại và chuyển đến lần lặp tiếp theo.

Ví dụ:

```
<?php
for ($i = 0; $i < 10; $i++) {
    if ($i % 2 == 0) {
        continue;
    }
    echo "The number is: $i <br>";
}
?>
```

* Kết quả:

The number is: 1

The number is: 3
The number is: 5
The number is: 7
The number is: 9

4.5.3.3. **Lệnh DIE()**

Lệnh **die()** dừng việc thực thi của script ngay lập tức và có thể hiển thị một thông báo lỗi tùy chọn. die là một alias của lệnh exit.

Ví dụ:

```
<?php
$file = fopen("somefile.txt", "r") or die("Unable to open file!");
// Code to read from the file...
?>
```

* Kết quả:
Unable to open file!

4.5.3.4. **Lệnh EXIT()**

Lệnh **exit()** cũng dừng việc thực thi của script ngay lập tức và có thể hiển thị một thông báo tùy chọn hoặc trả về một mã trạng thái.

Ví dụ:

```
<?php
if (!file_exists("somefile.txt")) {
    exit("File not found!");
}
// Code to read from the file...
?>
```

* Kết quả:
File not found!

4.5.4. **Lệnh INCLUDE – INCLUDE_ONCE – REQUIRE – REQUIRE_ONCE**

Trong PHP, việc tái sử dụng mã nguồn từ các tập tin khác là rất quan trọng để giữ cho mã nguồn gọn gàng và dễ bảo trì. PHP cung cấp bốn lệnh chính để sử dụng các tập tin php có sẵn vào tập tin php hiện tại là lệnh INCLUDE, INCLUDE_ONCE, REQUIRE, REQUIRE_ONCE.

4.5.4.1. **Lệnh INCLUDE**

Lệnh **include** được sử dụng để chèn nội dung của một tập tin php vào tập tin php hiện tại. Nếu tập tin không tồn tại, một cảnh báo sẽ được hiển thị và mã nguồn tiếp tục thực thi.

Cú pháp:

```
include 'path_to_file.php'
include ('path_to_file.php')
```

Ví dụ:

```
<?php
    include 'header.php';
    echo "This is the main content of the page.";
    include( 'footer.php');
?>
```

4.5.4.2. Lệnh **INCLUDE_ONCE**

Lệnh **include_once** hoạt động tương tự như include, nhưng đảm bảo rằng tập tin được chèn chỉ một lần. Nếu tập tin đã được chèn trước đó, nó sẽ không được chèn lại.

Cú pháp:

```
include_once 'path_to_file.php'
include_once ('path_to_file.php')
```

Ví dụ:

```
<?php
    include_once 'config.php';
    include_once 'config.php'; // Tập tin này sẽ không được chèn lần thứ hai
?>
```

4.5.4.3. Lệnh **REQUIRE**

Lệnh **require** tương tự như include, nhưng nếu tập tin không tồn tại, một lỗi nghiêm trọng sẽ được hiển thị và mã nguồn sẽ dừng thực thi. Sử dụng require khi tập tin được yêu cầu là thiết yếu cho ứng dụng.

Cú pháp:

```
require 'path_to_file.php'
require ('path_to_file.php')
```

Ví dụ:

```
<?php
    require 'config.php';
    echo "This is the main content of the page.";
?>
```

4.5.4.2. Lệnh **REQUIRE_ONCE**

Lệnh **require_once** kết hợp tính năng của require và include_once. Nó đảm bảo rằng tập tin được chèn chỉ một lần và dừng thực thi nếu tập tin không tồn tại.

Cú pháp:

```
require_once 'path_to_file.php'
```

```
require_once ('path_to_file.php')
```

Ví dụ:

```
<?php
    require_once 'config.php';
    require_once 'config.php'; // Tập tin này sẽ không được chèn lần thứ hai
?>
```

4.5.5. Toán tử 3 ngôi và các cú pháp thay thế

PHP cung cấp nhiều cách để viết các cấu trúc điều kiện và vòng lặp. Ngoài cú pháp thông thường, PHP còn hỗ trợ cú pháp thay thế cho các cấu trúc ***if, for, while, và foreach***, giúp mã nguồn trở nên gọn gàng hơn, đặc biệt hữu ích trong việc nhúng PHP vào HTML. Bên cạnh đó, toán tử 3 ngôi (ternary operator) là một công cụ mạnh mẽ giúp rút gọn các câu lệnh điều kiện đơn giản.

4.5.5.1. Toán tử 3 ngôi

Toán tử 3 ngôi là một dạng rút gọn của câu lệnh *if...else*. Nó giúp kiểm tra một điều kiện và trả về một giá trị nếu điều kiện đúng và một giá trị khác nếu điều kiện sai.

Cú pháp:

```
(condition) ? true_value : false_value;
```

Ví dụ:

```
<?php
    $age = 20;
    $status = ($age >= 18) ? "Adult" : "Minor";
    echo $status; // Kết quả: Adult
?>
```

4.5.5.2. Cú pháp thay thế IF

Cú pháp thay thế cho *if* sử dụng dấu hai chấm (:) và *endif* thay cho dấu ngoặc nhọn.

Cú pháp:

```
if (condition):
    // Các câu lệnh
endif;
```

Ví dụ:

```
<?php if ($age >= 18): ?>
    <p>You are an adult.</p>
<?php else: ?>
    <p>You are a minor.</p>
```

```
<?php endif; ?>
```

4.5.5.3. *Cú pháp thay thế FOR*

Cú pháp thay thế cho for sử dụng dấu hai chấm (:) và endfor thay cho dấu ngoặc nhọn.

Cú pháp:

```
for (initialization; condition; increment):  
    // Các câu lệnh  
endfor;
```

Ví dụ:

```
<?php for ($i = 0; $i < 5; $i++): ?>  
    <p>The number is: <?php echo $i; ?></p>  
<?php endfor; ?>
```

4.5.5.4. *Cú pháp thay thế WHILE*

Cú pháp thay thế cho while sử dụng dấu hai chấm (:) và endwhile thay cho dấu ngoặc nhọn.

Cú pháp:

```
while (condition):  
    // Các câu lệnh  
endwhile;
```

Ví dụ:

```
<?php  
    $i = 0;  
    while ($i < 5):  
        ?>  
        <p>The number is: <?php echo $i; ?></p>  
        <?php $i++; ?>  
    <?php endwhile; ?>
```

4.5.5.5. *Cú pháp thay thế FOREACH*

Cú pháp thay thế cho foreach sử dụng dấu hai chấm (:) và endforeach thay cho dấu ngoặc nhọn.

Cú pháp:

```
foreach ($array as $value):  
    // Các câu lệnh
```

```
endforeach;
```

Ví dụ:

```
<?php
    $colors = array("Red", "Green", "Blue");
    foreach ($colors as $color):
    ?>

        <p>The color is: <?php echo $color; ?></p>

    <?php endforeach; ?>
```

4.6. Hàm trong PHP

Hàm là một khối mã được đặt tên và có thể được gọi từ các phần khác nhau trong chương trình. Việc sử dụng hàm giúp mã nguồn trở nên dễ đọc, dễ bảo trì và tái sử dụng. Trong PHP, hàm có thể nhận các tham số đầu vào và trả về giá trị.

4.6.1. Khai báo và gọi hàm

a) Khai báo hàm

Hàm được khai báo bằng từ khóa `function`, theo sau là tên hàm, danh sách tham số (nếu có) trong dấu ngoặc đơn, và khối mã trong dấu ngoặc nhọn.

Cú pháp:

```
function functionName($param1, $param2, ...) {
    // Các câu lệnh
}
```

Ví dụ:

```
<?php

    function sayHello() {
        echo "Hello, World!";
    }

?>
```

b) Gọi hàm

Hàm được gọi bằng cách sử dụng tên hàm, theo sau là danh sách tham số (nếu có) trong dấu ngoặc đơn.

Ví dụ 1:

```
<?php

    sayHello(); // Kết quả: Hello, World!

?>
```

Ví dụ 2: Tính tổng đơn hàng


```
<?php
    function calculateTotal($prices) {
        return array_sum($prices);
    }

    $prices = [200000, 300000, 150000];
    $total = calculateTotal($prices);

    echo "Tổng đơn hàng: " . number_format($total) . " VND";
?>
```

4.6.2. Tham số và giá trị trả về

a) Tham số

Hàm có thể nhận các tham số đầu vào, cho phép truyền dữ liệu vào hàm.

Ví dụ:

```
<?php
    function greet($name) {
        echo "Hello, $name!";
    }

    greet("Huong"); // Kết quả: Hello, Huong!
?>
```

b) Giá trị trả về

Hàm có thể trả về một giá trị bằng cách sử dụng từ khóa return.

Ví dụ:

```
<?php
    function add($a, $b) {
        return $a + $b;
    }

    $result = add(3, 4); // Kết quả: 7
    echo $result;
?>
```

4.6.3. Phạm vi biến (Scope)

Phạm vi biến đề cập đến phạm vi mà biến có thể được truy cập. Biến được khai báo bên trong hàm là biến cục bộ và chỉ có thể được truy cập trong hàm đó.

a) Biến cục bộ

Ví dụ:

```
<?php
    function myTest() {
        $x = 5; // Biến cục bộ
        echo $x;
    }
```

```

}

myTest(); // Kết quả: 5
echo $x; // Lỗi: biến $x không tồn tại ngoài hàm
?>

```

b) Biến toàn cục

Biến được khai báo ngoài hàm là biến toàn cục và có thể được truy cập từ bất kỳ đâu trong mã nguồn.

Ví dụ:

```

<?php
    $x = 10; // Biến toàn cục

    function myTest() {
        global $x; // Sử dụng biến toàn cục trong hàm
        echo $x;
    }

    myTest(); // Kết quả: 10
?>

```

c) Từ khóa GLOBAL

Để sử dụng biến toàn cục bên trong hàm, cần khai báo biến đó với từ khóa global.

Ví dụ:

```

<?php
    $y = 20;

    function myTest() {
        global $y;
        echo $y;
    }

    myTest(); // Kết quả: 20
?>

```

4.6.4. Hàm dựng sẵn (build-in) trong PHP

PHP cung cấp nhiều hàm built-in mạnh mẽ cho các thao tác chuỗi, mảng, toán học, ngày giờ, và nhiều thao tác khác. Dưới đây là một số ví dụ về các hàm built-in phổ biến:

5.6.4.1. Hàm xử lý chuỗi

Bảng 5. 7 Hàm xử lý chuỗi

Hàm	Mô tả	Ví dụ
trim(\$string)	Hàm loại bỏ khoảng trắng thừa trong chuỗi	\$ho_ten = trim(' huong ');

<code>strlen(\$string)</code>	Hàm trả về chiều dài của chuỗi	<code>\$len_hoten = strlen('Kim Huong');</code>
<code>str_word_count(\$str)</code>	Hàm trả về số chữ trong chuỗi (không hỗ trợ tiếng việt)	<code>\$word_count = str_word_count('Kim Huong')</code>
<code>strcmp(\$str1, \$str2)</code>	Hàm so sánh hai chuỗi	<code>\$ket_qua = strcmp(\$str1, \$str2); // trả về 0, -1, 1 \$str1 = "hello"; \$str2 = "hello"; // 0: hai chuỗi bằng nhau \$str1 = "apple"; \$str2 = "banana"; // -1: str1 nhỏ hơn str2 \$str1 = "orange"; \$str2 = "apple"; // 1: str1 lớn hơn str2</code>
<code>strpos(\$str, \$substr)</code>	Hàm tìm chuỗi \$substr trong chuỗi \$str, nếu tìm thấy sẽ trả về số thứ tự, ngược lại sẽ trả về false.	<code>\$vi_tri = strpos('Huong', 'n'); // trả về vị trí đầu tiên hoặc false nếu không tìm thấy</code>
<code>str_replace(\$str1, \$str2, \$str)</code>	Hàm thay thế chuỗi str1 trong chuỗi str bằng chuỗi str2	<code>\$str1 = "world"; \$str2 = "PHP"; \$str = "Hello world!"; \$str_new = str_replace(\$str1, \$str2, \$str); echo \$str_new; // Hello PHP!</code>
<code>substr(\$str, \$start, \$len)</code>	Hàm lấy chuỗi con trong chuỗi cha str	<code>echo substr('Huong', 0, 2);</code>
<code>strstr(\$str, \$start_str)</code>	Hàm tách một chuỗi từ ký tự cho trước \$start_str cho đến hết chuỗi	<code>echo strstr('Huong biên soạn bài giảng Lập trình web PHP', 'bài'); // bài giảng Lập trình web PHP</code>
<code>implode(string \$glue, array \$pieces)</code>	Hàm này sẽ nối các phần tử của mảng và các phần tử nối với nhau bởi chuỗi \$glue	<code>\$str = implode(" ", array (\$str1, \$str2)); // kết hợp chuỗi str1 với str2 thành str và str1 cách str2 bằng 1 khoảng trắng " "</code>
<code>explode(string \$delimiter, string \$string[, int \$limit = PHP_INT_MAX])</code>	Hàm này sẽ chuyển một chuỗi \$string thành một mảng các phần tử với ký tự tách mảng	<code>\$arr = explode(' ', 'Lập Trình PHP') Hoặc: \$string = "apple,banana,orange"; \$arr = explode(",", \$string); print_r(\$array);</code>

	là \$delimiter, \$limit (tùy chọn) số phần tử tối đa trong mảng, mặc định là PHP_INT_MAX	
str_repeat(\$str, \$n)	Hàm lặp chuỗi \$str với \$n lần	echo str_repeat('Huong', 2);
md5(\$str)	Hàm này sẽ mã hóa MD5 chuỗi \$str (32 ký tự)	echo md5('huong');
sha1(\$str)	Hàm này sẽ mã hóa sha1 chuỗi \$str (40 ký tự)	echo sha1('huong');
strtoupper(\$str)	Hàm chuyển chuỗi \$str thành chữ in hoa	\$str = 'kim Huong'; echo strtoupper(\$str); // KIM HƯƠNG
strtolower(\$str)	Hàm chuyển chuỗi \$str thành chữ thường	\$str = 'kim Huong'; echo strtolower(\$str); // kim huong
ucwords(\$str)	Hàm chuyển chữ cái đầu tiên của các từ trong chuỗi sang chữ In hoa	\$str = 'kim Huong'; echo ucwords(\$str); // Kim Huong
htmlentities(\$str)	Hàm chuyển các thẻ html sang dạng thực thể, nghĩa là in ra màn hình sẽ hiển thị các thẻ html	echo htmlentities('Kim Huong'); // Kim Huong
html_entity_decode(\$str)	Hàm chuyển dạng thực thể HTML sang định dạng HTML ban đầu, nghĩa là trình duyệt sẽ biên dịch các thẻ HTML	\$str = htmlentities('Kim Huong'); echo html_entity_decode(\$str);
strip_tags(\$str, \$allow_tags)	Hàm này bỏ các thẻ html trong chuỗi \$str được khai báo ở \$allow_tags	\$str = '<p>This is a link.</p>'; \$clean_str = strip_tags(\$str); // This is a link.

	(nếu không khai báo sẽ loại bỏ tất cả thẻ)	<pre>\$allow_tags = '<a>'; \$clean_str = strip_tags(\$str, \$allow_tags); // This is a link. \$allow_tags = ['a', 'p'];</pre>
json_encode(\$array_or_object)	Hàm chuyển đổi một mảng hoặc một đối tượng trong PHP thành một chuỗi JSON	<pre>\$data = array("name" => "Hương", "job" => "Lập trình viên", "city" => "Cao Lãnh"); \$json_data = json_encode(\$data); echo \$json_data; //{"name":"Hương","job":"Lập trình viên","city":" Cao Lãnh "}</pre>
json_decode(\$json, \$is_array)	Hàm chuyển đổi một chuỗi JSON thành một mảng hoặc một đối tượng PHP. Nếu \$is_array=true trả về dạng mảng, false là Object (mặc định là false)	<pre>\$json_data = '{"name":"Hương","job":"Lập trình viên","city":" Cao Lãnh "}'; \$data = json_decode(\$json_data, true); print_r(\$data); // Array([name] => Hương [job] => Lập trình viên [city] => Cao Lãnh) \$data = json_decode(\$json_data); // stdClass Object ([name] => Hương [job] => Lập trình viên [city] => Cao Lãnh)</pre>

4.6.4.2. Hàm xử lý mảng

Mảng là một tập hợp các phần tử, mỗi phần tử có một chỉ số (index) riêng biệt. Chỉ số có thể là số nguyên (trong mảng chỉ số) hoặc chuỗi (trong mảng kết hợp).

Để khai báo mảng, sử dụng hàm array() hoặc [] (với PHP 5.4 trở lên).

a) Các loại mảng trong PHP

(1) Mảng chỉ số (Indexed Array)

Mảng chỉ số sử dụng các chỉ số số nguyên để truy cập các phần tử.

Ví dụ:

```
<?php
```

```

$fruits = array('Apple', 'Banana', 'Orange');
Hoặc:
$fruits = ['Apple', 'Banana', 'Orange'];
echo $fruits[0]; // Kết quả: Apple

// xem cấu trúc mảng
echo'<pre>';
print_r($arr);
echo'</pre>';

?>

```

(2) Mảng Kết Hợp (Associative Array)

Mảng kết hợp sử dụng các khóa (key) để truy cập các phần tử.

Ví dụ:

```

<?php
$ages = array("Lan" => 35, "Hue" => 37, "Cuc" => 43);
echo $ages["Hue"]; // Kết quả: 37

?>

```

(3) Mảng Đa Chiều (Multidimensional Array)

Mảng đa chiều là mảng chứa các mảng khác.

Ví dụ:

```

<?php
$products = array(
    array("Apple", 22, 18),
    array("Banana", 15, 13),
    array("Orange", 5, 2)
);
echo $products[0][0]; // Kết quả: Apple

?>

```

hoặc

```

<?php
$arr = [
    'address1' => [
        'name' => [
            'Ho' => 'Tran',
            'Ten' => 'Huong'
        ],
        'email' => 'tkhuong@dthu.edu.vn'
    ],
    'address2' => [

```

```

        'name' => 'Kim Huong',
        'email' => 'tkhuong@gmail.com'
    ],
    'address3' => 'PHP'
];

echo $arr['address1']['name']['Ho'];
?>

```

- * Thêm phần tử vào mảng
- Thêm phần tử có key

```

$ten_bien[key] = value;

Ví dụ: $ages['Hong'] = 20;

```

- Thêm phần tử không key

```

$ten_bien[] = value;

Ví dụ: $subj[] = 'PHP';

```

- Dùng thông qua hàm array_push(\$ten_bien, value1, value2);
ví dụ: array_push(\$subj, 'Python', 'C++', 'NodeJS');
- * Sửa phần tử

```

$ten_bien[$key] = value;

Ví dụ: $subj[0] = 'PHP edit';

```

- * Xóa phần tử

```

unset($ten_bien[$key]);

Ví dụ: unset($ages['Hong']);

```

- * Đọc mảng trong PHP
 - Dùng vòng lặp for
- Ví dụ:

```

<?php
    if(!empty($ages)){
        for($i = 0; $i < count($ages); $i++){
            echo $ages[$i] . '<br>';
        }
    }
?>

```

- Dùng vòng lặp foreach

Ví dụ:

```
<?php
    if(!empty($ages)){
        foreach($ages as $key = $value){
            echo $value . '<br>';
        }
    }
?>
```

Ví dụ: Đọc mảng đa chiều

```
<?php
    $products = array(
        array("Apple", 22, 18),
        array("Banana", 15, 13),
        array("Orange", 5, 2)
    );

    $length = count($products); // Lấy chiều dài mảng
    for($row = 0; $row < $length; $row++) {
        $length_sub = count($products[0]); // Lấy chiều dài mảng con
        for($col = 0; $col < $length_sub; $col++) {
            echo $products[$row][$col]. " ";
        }
        echo "<br>";
    }
?>
```

- Đọc trực tiếp từ key: \$ten_bien[key]

Ví dụ: \$res = \$ages['Hong']; echo \$res. '
';

b) Các hàm xử lý Mảng thông dụng

(1) Hàm count()

Hàm **count()** trả về số lượng phần tử trong một mảng.

Ví dụ:

```
<?php
    $fruits = array("Apple", "Banana", "Orange");
    echo count($fruits); // Kết quả: 3
?>
```

(2) Hàm array_merge()

Hàm array_merge() kết hợp các mảng lại với nhau.

Ví dụ:

```
<?php
    $array1 = array("color" => "red", 2, 4);
    $array2 = array("a", "b", "color" => "green", "shape" => "trapezoid", 4);
    $result = array_merge($array1, $array2);
    print_r($result);
?>
```

(3) Hàm *array_push()*

Hàm *array_push()* thêm một hoặc nhiều phần tử vào cuối mảng.

Ví dụ:

```
<?php
    $fruits = array("Apple", "Banana");
    array_push($fruits, "Orange", "Mango");
    print_r($fruits);
?>
```

(4) Hàm *array_pop()*

Hàm *array_pop()* loại bỏ phần tử cuối cùng của mảng.

Ví dụ:

```
<?php
    $fruits = array("Apple", "Banana", "Orange");
    array_pop($fruits);
    print_r($fruits); // Kết quả: Array ( [0] => Apple [1] => Banana )
?>
```

(5) Hàm *array_shift()*

Hàm *array_shift()* loại bỏ phần tử đầu tiên của mảng.

Ví dụ:

```
<?php
    $fruits = array("Apple", "Banana", "Orange");
    array_shift($fruits);
    print_r($fruits); // Kết quả: Array ( [0] => Banana [1] => Orange )
?>
```

(6) Hàm *array_unshift()*

Hàm *array_unshift()* thêm một hoặc nhiều phần tử vào đầu mảng.

Ví dụ:

```
<?php
    $fruits = array("Banana", "Orange");
    array_unshift($fruits, "Apple");
    print_r($fruits); // Kết quả: Array ( [0] => Apple [1] => Banana [2] => Orange )
?>
```

(7) Hàm *in_array()*

Hàm *in_array()* kiểm tra sự tồn tại của một phần tử trong mảng.

Ví dụ:

```
<?php
    $fruits = array("Apple", "Banana", "Orange");
    if (in_array("Banana", $fruits)) {
        echo "Banana is in the array.";
    } else {
        echo "Banana is not in the array.";
    }
?>
```

(8) Hàm *array_keys()* và *array_values()*

Hàm *array_keys()* lấy danh sách khóa, và *array_values()* lấy danh sách giá trị trong mảng.

Ví dụ:

```
<?php
    $ages = array("Peter" => 35, "Ben" => 37, "Joe" => 43);
    $keys = array_keys($ages);
    $values = array_values($ages);
    print_r($keys); // Kết quả: Array ( [0] => Peter [1] => Ben [2] => Joe )
    print_r($values); // Kết quả: Array ( [0] => 35 [1] => 37 [2] => 43 )
?>
```

(9) Hàm *sort()* và *rsort()*

Hàm *sort()* sắp xếp mảng theo thứ tự tăng dần, và *rsort()* sắp xếp mảng theo thứ tự giảm dần.

Ví dụ:

```
<?php
    $numbers = array(4, 6, 2, 22, 11);
    sort($numbers);
```

```
print_r($numbers); // Kết quả: Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 11 [4] => 22 )

rsort($numbers);
print_r($numbers); // Kết quả: Array ( [0] => 22 [1] => 11 [2] => 6 [3] => 4 [4] => 2 )

?>
```

(10) Hàm `array_reverse()`

Hàm `array_reverse()` trả về một mảng với thứ tự các phần tử bị đảo ngược.

Ví dụ:

```
<?php
$fruits = array("Apple", "Banana", "Cherry");
$reversedFruits = array_reverse($fruits);
print_r($reversedFruits); // Kết quả: Array ( [0] => Cherry [1] => Banana [2] => Apple )

?>
```

(11) Hàm `array_rand()`

Hàm `array_rand()` chọn ngẫu nhiên một hoặc nhiều khóa từ mảng.

Ví dụ:

```
<?php
$fruits = array("Apple", "Banana", "Cherry", "Date");
$randomKey = array_rand($fruits);
echo $fruits[$randomKey]; // Kết quả: Một trong các giá trị của mảng (Apple, Banana,
Cherry, hoặc Date)

$randomKeys = array_rand($fruits, 2);
print_r($randomKeys); // Kết quả: Mảng chứa hai khóa ngẫu nhiên

?>
```

(12) Hàm `array_search()`

Hàm `array_search()` tìm kiếm một giá trị trong mảng và trả về khóa đầu tiên tìm thấy.

Ví dụ:

```
<?php
$fruits = array("Apple", "Banana", "Cherry");
$key = array_search("Banana", $fruits);
echo $key; // Kết quả: 1

?>
```

(13) Hàm `array_slice()`

Hàm `array_slice()` trả về một phần của mảng.

Cú pháp:

```
array array_slice(array $array, int $offset [, int $length = null [, bool $preserve_keys = false]])
```

Trong đó: \$array: mảng đầu vào; \$offset: vị trí bắt đầu cắt; \$length: độ dài của phần mảng cần cắt (mặc định là toàn bộ phần còn lại); \$preserve_keys: nếu true thì giữ nguyên các khóa.

Ví dụ:

```
<?php
    $fruits = array("Apple", "Banana", "Cherry", "Date");
    $slicedFruits = array_slice($fruits, 1, 2);
    print_r($slicedFruits); // Kết quả: Array ( [0] => Banana [1] => Cherry )
?>
```

(14) Hàm array_unique()

Hàm **array_unique()** loại bỏ các phần tử trùng lặp trong mảng.

Ví dụ:

```
<?php
    $fruits = array("Apple", "Banana", "Apple", "Cherry");
    $uniqueFruits = array_unique($fruits);
    print_r($uniqueFruits); // Kết quả: Array ( [0] => Apple [1] => Banana [3] => Cherry )
?>
```

(15) Hàm array_key_exists()

Hàm **array_key_exists()** kiểm tra sự tồn tại của một khóa trong mảng.

Cú pháp:

```
bool array_key_exists(mixed $key, array $array)
```

Trong đó: \$key: khóa cần kiểm tra; \$array: mảng đầu vào

Ví dụ:

```
<?php
    $ages = array("Peter" => 35, "Ben" => 37, "Joe" => 43);
    if (array_key_exists("Ben", $ages)) {
        echo "Key 'Ben' exists in the array.";
    } else {
        echo "Key 'Ben' does not exist in the array.";
    }
?>
```

4.6.4.3. Hàm xử lý Số (Number)

(1) Hàm is_int(), is_float(), is_numeric()

Kiểm tra xem biến có phải là số nguyên/ số thực/ kiểu số (nguyên hoặc thực) không.

Ví dụ:

```
<?php
    $var1 = 123;
    $var2 = 12.34;
    $var3 = "456";

    echo is_int($var1); // Kết quả: 1 (true)
    echo is_float($var2); // Kết quả: 1 (true)
    echo is_numeric($var3); // Kết quả: 1 (true)

?>
```

(2) Hàm abs(), ceil(), floor(), round(), sqrt()

Hàm abs(\$number): Trả về giá trị tuyệt đối của một số; ceil(\$number): Làm tròn lên tới số nguyên gần nhất; floor(\$number): Làm tròn xuống tới số nguyên gần nhất; round(\$number [, \$precision]): Làm tròn số đến số chữ số thập phân xác định; sqrt(): Tính căn bậc hai của hai số dương, trả về một giá trị đặc biệt NAN cho các số âm.

Ví dụ:

```
<?php
    echo abs(-4.2); // Kết quả: 4.2
    echo ceil(4.2); // Kết quả: 5
    echo floor(4.8); // Kết quả: 4
    echo round(4.5); // Kết quả: 5
    echo round(4.567, 2); // Kết quả: 4.57
    echo sqrt(9); // outputs: 3
    echo sqrt(-16); // outputs: NAN

?>
```

(3) Hàm min(), max()

Hàm min()/ max() dùng để tìm giá trị nhỏ nhất/ lớn nhất trong danh sách các đối số

Ví dụ:

```
<?php
    echo(min(0, 150, 30, 20, -8, -200)); // Kết quả: -200
    echo(max(0, 150, 30, 20, -8, -200)); // Kết quả: 150
```

```
?>
```

(3) Hàm rand(), mt_rand()

Hàm rand([int \$min, int \$max]): Trả về một số nguyên ngẫu nhiên trong khoảng từ \$min đến \$max; mt_rand([int \$min, int \$max]): Trả về một số nguyên ngẫu nhiên sử dụng thuật toán Mersenne Twister.

Ví dụ 1:

```
<?php
    echo rand(1, 10);    // Kết quả: Một số ngẫu nhiên giữa 1 và 10
    echo mt_rand(1, 100); // Kết quả: Một số ngẫu nhiên giữa 1 và 100
?>
```

Ví dụ 2:

```
<?php
    $min = 1;
    $max = 100;
    $randomNumber = mt_rand($min, $max);
    echo "Số ngẫu nhiên giữa $min và $max: $randomNumber";
?>
```

(4) Hàm number_format()

Hàm number_format() định dạng một số với dấu phân cách hàng nghìn và dấu phân cách thập phân.

Cú pháp:

```
number_format($number, $decimals, $decimal_separator, $thousands_separator)
```

Ví dụ 1:

```
<?php
    $number = 1234567.89;
    echo number_format($number);    // Kết quả: 1,234,568
    echo number_format($number, 2); // Kết quả: 1,234,567.89
    echo number_format($number, 2, ',', '.'); // Kết quả: 1.234.567,89
?>
```

Ví dụ 2:

```
<?php
    $numbers = array(123, 12.34, "456", "abc");
    foreach ($numbers as $num) {
        if (is_numeric($num)) {
            echo number_format($num, 2) . "<br>";
        }
    }
?>
```

```

        } else {
            echo "'$num' is not a number.<br>";
        }
    }
}
?>

```

4.6.4.4. Hàm xử lý ngày giờ (DateTime)

a) Kiểu dữ liệu DateTime

- Khởi tạo đối tượng DateTime

```

<?php
// Khởi tạo đối tượng DateTime với thời gian hiện tại
$date = new DateTime();

// Khởi tạo đối tượng DateTime với một thời gian cụ thể
$specificDate = new DateTime('2023-12-31 12:34:56');
?>

```

- Định dạng Ngày Giờ: sử dụng phương thức **format()** của lớp DateTime

```

<?php
$date = new DateTime();
echo $date->format('Y-m-d H:i:s'); // Kết quả: 2024-07-08 15:34:56
?>

```

- Cộng thêm 1 ngày

```

<?php
// Tạo đối tượng DateTime với thời gian hiện tại
$date = new DateTime();

// Định dạng ngày giờ
echo $date->format('Y-m-d H:i:s'); // Kết quả: 2024-07-08 15:34:56

// Cộng thêm 1 ngày
$date->modify('+1 day');
echo $date->format('Y-m-d H:i:s'); // Kết quả: 2024-07-09 15:34:56
?>

```

- So sánh 2 ngày

```

<?php
$date1 = new DateTime('2024-01-01');

```

```

$date2 = new DateTime('2024-12-31');

if ($date1 < $date2) {
    echo "Ngày 1 nhỏ hơn Ngày 2";
} else {
    echo "Ngày 1 không nhỏ hơn Ngày 2";
}
?>

```

- Tính khoảng cách giữa 2 ngày

```

<?php
$date1 = new DateTime('2024-01-01');
$date2 = new DateTime('2024-12-31');

$interval = $date1->diff($date2);
echo $interval->format('%R%a ngày'); // Kết quả: +365 ngày
?>

```

b) Các hàm xử lý DateTime thông dụng

- Timestamp: số giây tính từ thời điểm 00:00:00 1/1/1970 đến thời điểm cần xác định.

- Timezone: `date_default_timezone_set()`, `date_default_timezone_get()`.

(1) Hàm `date_default_timezone_set()`

Hàm `date_default_timezone_set()` để đặt múi giờ mặc định cho tất cả các hàm liên quan đến ngày và giờ.

Cú pháp:

```
bool date_default_timezone_set ( string $timezone_identifier )
```

Trong đó, `$timezone_identifier`: Tên của múi giờ muốn thiết lập ('UTC'; 'Asia/Ho_Chi_Minh'; 'America/New_York'; 'Europe/London'; 'Australia/Sydney'). Hàm trả về True nếu thành công và False nếu thất bại. Xem thêm danh sách timezone mà PHP hỗ trợ tại <https://www.php.net/manual/en/timezones.php> hoặc viết đoạn mã lệnh sau:

```

// xem danh sách timezone mà PHP hỗ trợ
<?php
foreach (timezone_abbreviations_list() as $abbr => $timezone) {
    foreach ($timezone as $val) {
        if (isset($val['timezone_id'])) {
            var_dump($abbr, $val['timezone_id']);
        }
    }
}

```



```
}  
}  
}  
?>
```

Ví dụ 1: Thiết lập múi giờ và lấy thời gian hiện tại

```
<?php  
// Thiết lập múi giờ mặc định là Asia/Ho_Chi_Minh  
date_default_timezone_set('Asia/Ho_Chi_Minh');  
  
// Lấy thời gian hiện tại  
echo "Thời gian hiện tại ở Hồ Chí Minh: " . date('Y-m-d H:i:s') . "<br>";  
  
// Thiết lập múi giờ mặc định là America/New_York  
date_default_timezone_set('America/New_York');  
  
// Lấy thời gian hiện tại  
echo "Thời gian hiện tại ở New York: " . date('Y-m-d H:i:s') . "<br>";  
?>  
Kết quả:  
Thời gian hiện tại ở Hồ Chí Minh: 2024-07-08 22:34:56  
Thời gian hiện tại ở New York: 2024-07-08 11:34:56
```

Ví dụ 2: Tính toán thời gian theo múi giờ khác nhau

```
<?php  
// Thiết lập múi giờ mặc định là UTC  
date_default_timezone_set('UTC');  
  
// Lấy thời gian hiện tại ở UTC  
$current_time_utc = new DateTime();  
echo "Thời gian hiện tại ở UTC: " . $current_time_utc->format('Y-m-d H:i:s') . "<br>";  
  
// Thiết lập múi giờ mặc định là Asia/Ho_Chi_Minh  
date_default_timezone_set('Asia/Ho_Chi_Minh');  
  
// Lấy thời gian hiện tại ở Hồ Chí Minh  
$current_time_hcm = new DateTime();  
echo "Thời gian hiện tại ở Hồ Chí Minh: " . $current_time_hcm->format('Y-m-d H:i:s')  
. "<br>";  
  
// Tính toán khoảng cách thời gian giữa UTC và Hồ Chí Minh  
$interval = $current_time_utc->diff($current_time_hcm);  
echo "Khoảng cách thời gian giữa UTC và Hồ Chí Minh: " . $interval->format('%H giờ  
%I phút') . "<br>";
```

```
?>
```

Kết quả:

Thời gian hiện tại ở UTC: 2024-07-08 15:34:56

Thời gian hiện tại ở Hồ Chí Minh: 2024-07-08 22:34:56

Khoảng cách thời gian giữa UTC và Hồ Chí Minh: 07 giờ 00 phút

(2) Hàm `date_default_timezone_get()`

Hàm `date_default_timezone_get()` được sử dụng để lấy múi giờ mặc định hiện tại của hệ thống.

Cú pháp:

```
string date_default_timezone_get ( void )
```

Ví dụ 1: Lấy múi giờ mặc định hiện tại

```
<?php
```

```
// Lấy múi giờ mặc định hiện tại
```

```
$current_timezone = date_default_timezone_get();
```

```
echo "Múi giờ mặc định hiện tại là: " . $current_timezone;
```

```
?>
```

Kết quả:

Múi giờ mặc định hiện tại là: UTC

Ví dụ 2: So sánh múi giờ trước và sau khi thay đổi múi giờ

```
<?php
```

```
// Lấy múi giờ mặc định hiện tại
```

```
$original_timezone = date_default_timezone_get();
```

```
echo "Múi giờ mặc định ban đầu là: " . $original_timezone . "<br>";
```

```
// Thiết lập múi giờ mặc định mới
```

```
date_default_timezone_set('Asia/Ho_Chi_Minh');
```

```
// Lấy múi giờ mới
```

```
$new_timezone = date_default_timezone_get();
```

```
echo "Múi giờ mặc định mới là: " . $new_timezone;
```

```
?>
```

(3) Hàm `date()`

Hàm date() trả về ngày giờ hiện tại hoặc một thời gian cụ thể dưới dạng chuỗi đã được định dạng.

Cú pháp:

string date(string \$format [, int \$timestamp = time()])

Trong đó, \$format: chuỗi định dạng thời gian để hiển thị; \$timestamp: thời gian truyền vào, nếu để trống thì PHP tự động lấy timestamp thời gian hiện tại (chính là hàm time())

Ví dụ:

```
<?php
echo "Today is " . date("d/m/Y") . "<br>";
echo "Today is " . date("d.m.Y") . "<br>";
echo "Today is " . date("d-m-Y") . "<br>";
echo "Today is " . date("l");
"<br>";
echo date('d-m-Y H:i:s');
echo date('Y-m-d H:i:s'); // Kết quả: 2024-07-08 15:34:56 ?>
```

Trong PHP, hàm date() được sử dụng để định dạng và hiển thị ngày giờ theo các định dạng cụ thể. Dưới đây là một số format thường dùng:

Bảng 5. 8 Các kiểu định dạng cho hàm Date

Ký tự Format	Ý nghĩa	Ví dụ	Kết quả
h	Giờ trong ngày (12h AM/PM)	date('h')	02 (ví dụ: 2 giờ chiều)
H	Giờ trong ngày (24h)	date('H')	14 (ví dụ: 14 giờ)
i	Phút trong giờ	date('i')	30
s	Giây trong phút	date('s')	45
d	Ngày trong tháng (từ 01 đến 31, có số 0 nếu <10)	date('d')	07
j	Ngày trong tháng (từ 1 đến 31, không có số 0 nếu <10)	date('j')	7
D	Thứ trong tuần (English)	date('D')	Fri
l	Thứ trong tuần (English)	date('l')	Friday
m	Tháng trong năm (từ 01 đến 12, có số 0 nếu <10)	date('m')	03 (tháng 3)
M	Tháng trong năm (English)	date('M')	Mar (tháng 3)

y	2 chữ số cuối của năm	date('y')	23 (năm 2023)
Y	4 chữ số đầy đủ của năm	date('Y')	2023

Ví dụ: Hiển thị Ngày, Giờ và thông tin định dạng

```
<?php
// Đặt múi giờ về Việt Nam
date_default_timezone_set('Asia/Ho_Chi_Minh');

// Hiển thị giờ hiện tại theo định dạng 24 giờ
echo "Giờ hiện tại: " . date('H:i:s') . "<br>";

// Hiển thị ngày hiện tại với định dạng dd-mm-yyyy
echo "Ngày hiện tại: " . date('d-m-Y') . "<br>";

// Hiển thị thứ, ngày tháng năm đầy đủ
echo "Hôm nay là: " . date('l, d-m-Y') . "<br>";

// Hiển thị ngày định dạng đầy đủ với ngày/tháng bằng tiếng Anh
echo "Ngày định dạng đầy đủ: " . date('l, j F Y') . "<br>";
?>
```

* Kết quả:

Giờ hiện tại: 14:30:45
 Ngày hiện tại: 07-09-2024
 Hôm nay là: Saturday, 07-09-2024
 Ngày định dạng đầy đủ: Saturday, 7 September 2024

(4) Hàm time()

Hàm time() trong PHP trả về thời gian hiện tại dưới dạng Unix timestamp. Unix timestamp là số giây tính từ "epoch" (thời điểm bắt đầu), được xác định là 00:00:00 UTC ngày 1 tháng 1 năm 1970.

Cú pháp:

```
int time ( void )
```

Ví dụ:

```
<?php
// Lấy thời gian hiện tại dưới dạng Unix timestamp
$current_time = time();
echo "Thời gian hiện tại (Unix timestamp): " . $current_time;
?>
```

Kết quả:

Thời gian hiện tại (Unix timestamp): 1701942445

(5) Hàm strtotime()

Hàm `strtotime()` chuyển đổi một chuỗi ngày giờ thành Unix timestamp.

Cú pháp:

```
int strtotime(string $time [, int $now = time()])
```

Ví dụ:

```
<?php
    $timestamp = strtotime('2024-12-31 12:34:56');
    echo date('Y-m-d H:i:s', $timestamp); // Kết quả: 2024-12-31 12:34:56
    echo strtotime('now'). '<br>'; //
?>
```

4.6.4.5. Hàm **ISSET** và **EMPTY**

Trong lập trình PHP, việc kiểm tra sự tồn tại và giá trị của biến là một phần quan trọng để đảm bảo chương trình hoạt động đúng và tránh các lỗi không mong muốn. Các hàm `isset`, `empty`, và các hàm kiểm tra sự tồn tại của phần tử như `function_exists`, `in_array`, và `file_exists` giúp chúng ta thực hiện điều này một cách hiệu quả.

a) Hàm **ISSET()**

Hàm ***ISSET()*** kiểm tra biến có tồn tại hay không, không kiểm tra về kiểu dữ liệu của biến, không kiểm tra được trường hợp `null`.

Cú pháp:

```
bool isset ( mixed $var [, mixed $... ] )
```

Ví dụ:

```
<?php
    $var1 = "Hello, World!";
    $var2 = null;
    echo isset($var1); // Kết quả: 1 (true) //1
    echo isset($var2); // Kết quả: (false) // không hiện
    echo isset($var3); // Kết quả: (false) // không hiện
?>
```

b) Hàm **EMPTY()**

Hàm ***EMPTY()*** kiểm tra biến có tồn tại hay không và kiểu dữ liệu, kiểm tra một giá trị có “empty” không, một biến được xem là “empty” khi nó là `null`, `""`, `0`, `false`, mảng rỗng, đối tượng không có thuộc tính.

Cú pháp:

```
bool empty ( mixed $var )
```

Ví dụ:

```

<?php
    $var1 = 0;
    $var2 = "Hello, World!";
    $var3 = "";
    $var4 = null;
    $var5 = false;
    $var6 = array();

    $variables = array(
        'var1' => $var1,
        'var2' => $var2,
        'var3' => $var3,
        'var4' => $var4,
        'var5' => $var5,
        'var6' => $var6
    );

    foreach ($variables as $name => $value) {
        echo $name . ' is ' . (empty($value) ? 'empty' : 'not empty') . "<br>";
    }
?>

```

Kết quả:
 var1 is empty
 var2 is not empty
 var3 is empty
 var4 is empty
 var5 is empty
 var6 is empty

c) Các hàm EXISTS

* Hàm IN_ARRAY

Hàm *in_array* kiểm tra xem một giá trị có tồn tại trong một mảng hay không.

Cú pháp:

```
bool in_array ( mixed $needle , array $haystack [, bool $strict = FALSE ] )
```

Trong đó: \$needle: Giá trị cần tìm; \$haystack: Mảng cần tìm trong; \$strict (tùy chọn): Nếu được đặt là true, kiểm tra cả kiểu dữ liệu của giá trị.

Ví dụ:

```

<?php
    $array = array("apple", "banana", "cherry");

    echo in_array("banana", $array); // Kết quả: 1 (true)
    echo in_array("grape", $array); // Kết quả: (false)
?>

```

* Hàm FILE_EXISTS

Hàm ***file_exists*** kiểm tra xem một tệp hoặc thư mục có tồn tại hay không.

Cú pháp:

bool file_exists (string \$filename)

Ví dụ:

```
<?php
    $filename = 'test.txt';
    if (file_exists($filename)) {
        echo "The file $filename exists.";
    } else {
        echo "The file $filename does not exist.";
    }
?>
```

*** Hàm *FUNCTION_EXISTS***

Hàm ***function_exists*** kiểm tra xem một hàm đã được định nghĩa hay chưa. Điều này rất hữu ích khi cần kiểm tra sự tồn tại của một hàm trước khi gọi nó, đặc biệt khi làm việc với các thư viện hoặc mã nguồn phức tạp.

Cú pháp:

bool function_exists (string \$function_name)

Ví dụ:

```
<?php
    function myFunction() {
        echo "Hello, World!";
    }

    if (function_exists('myFunction')) {
        myFunction(); // Kết quả: Hello, World!
    } else {
        echo "Function does not exist.";
    }
?>
```

4.6.5. Hàm ẩn danh (Anonymous Function) và Closure

Hàm ẩn danh là hàm không có tên và thường được sử dụng như một đối số cho các hàm khác hoặc được gán cho một biến.

4.6.5.1. Hàm ẩn danh

Ví dụ:

```
<?php
```

```
$greet = function($name) {
    return "Hello, $name!";
};

echo $greet("Hương"); // Kết quả: Hello, Hương!
?>
```

4.6.5.2. Closure

Closure là hàm ẩn danh có thể truy cập các biến từ phạm vi bên ngoài thông qua từ khóa *use*.

Ví dụ:

```
<?php
$message = "Good morning";

$greet = function($name) use ($message) {
    return "$message, $name!";
};

echo $greet("Huong"); // Kết quả: Good morning, Huong!
?>
```

Câu hỏi, bài tập

Câu hỏi:

Câu 1. Liệt kê các kiểu dữ liệu cơ bản trong PHP và cho ví dụ minh họa.

Gợi ý: Kiểu dữ liệu cơ bản: int, float, string, array, boolean, null

Câu 2. Toán tử . trong PHP có chức năng gì?

Gợi ý: Toán tử . dùng để nối chuỗi

Câu 3. Viết một đoạn mã hiển thị danh sách sản phẩm từ mảng.

Gợi ý:

```
$products = ["Laptop", "Điện thoại"];
foreach ($products as $product) {
    echo $product . "<br>";
}
```

Câu 4. Hàm array_sum() trong PHP dùng để làm gì?

Gợi ý: Hàm array_sum() tính tổng các phần tử trong mảng

```
$prices = [100, 200];
echo array_sum($prices); // Kết quả: 300
```

Câu 5. Viết chương trình kiểm tra đăng nhập với điều kiện tên đăng nhập là “admin” và mật khẩu là “12345”.

Bài tập

Bài 1. Viết chương trình kiểm tra số chẵn lẻ trong mảng.

Yêu cầu:

Tạo một mảng số nguyên bất kỳ.

Kiểm tra từng phần tử là số chẵn hay lẻ và in kết quả.

Gợi ý:

```
$numbers = [1, 2, 3, 4, 5];  
foreach ($numbers as $number) {  
    if ($number % 2 == 0) {  
        echo "$number là số chẵn.<br>";  
    } else {  
        echo "$number là số lẻ.<br>";  
    }  
}
```

Bài 2. Viết hàm đệ quy tính giai thừa của một số.

Yêu cầu:

Hàm nhận một số nguyên dương và trả về giai thừa của số đó.

Gợi ý:

```
function factorial($n) {  
    if ($n == 0) return 1;  
    return $n * factorial($n - 1);  
}  
echo "Giai thừa của 5 là: " . factorial(5);
```

Bài 3. Viết một hàm kiểm tra số nguyên tố.

Yêu cầu: Trả về true nếu số đó là số nguyên tố, ngược lại trả về false.

Gợi ý:

```
function isPrime($number) {  
    if ($number < 2) return false;  
    for ($i = 2; $i <= sqrt($number); $i++) {  
        if ($number % $i == 0) return false;  
    }  
    return true;  
}  
echo isPrime(7) ? "Là số nguyên tố" : "Không phải số nguyên tố";
```

Bài 4. Viết hàm hiển thị danh sách sản phẩm với thông tin giá và thời gian.

Yêu cầu:

- Tạo danh sách sản phẩm kèm theo thông tin giá và ngày thêm vào.
- Hiển thị tất cả sản phẩm và tính tổng giá trị sản phẩm đã thêm trong ngày hôm nay.

- Cho phép người dùng lọc sản phẩm theo khoảng thời gian cụ thể.

Gợi ý:

```
<?php
// Danh sách sản phẩm kèm giá và ngày thêm vào
$products = [
    ["name" => "Laptop", "price" => 20000000, "added_date" => "2023-12-25"],
    ["name" => "Điện thoại", "price" => 10000000, "added_date" => "2023-12-24"],
    ["name" => "Máy ảnh", "price" => 15000000, "added_date" => "2023-12-25"],
    ["name" => "Tai nghe", "price" => 2000000, "added_date" => "2023-12-23"]
];

// Hàm hiển thị danh sách sản phẩm
function displayProducts($products, $startDate = null, $endDate = null) {
    $total = 0;
    echo "Danh sách sản phẩm:<br>";
    foreach ($products as $product) {
        // Nếu có lọc theo khoảng thời gian, kiểm tra điều kiện
        if ($startDate && $endDate) {
            if ($product['added_date'] < $startDate || $product['added_date'] > $endDate) {
                continue;
            }
        }
        // Hiển thị sản phẩm
        echo $product['name'] . " - " . number_format($product['price']) . " VND - Ngày thêm: " . $product['added_date'] . "<br>";
        $total += $product['price'];
    }
    echo "Tổng giá trị: " . number_format($total) . " VND<br>";
}

// Lọc sản phẩm theo ngày hôm nay
echo "<h3>Sản phẩm được thêm hôm nay:</h3>";
$today = date("Y-m-d");
displayProducts($products, $today, $today);

// Lọc sản phẩm theo khoảng thời gian do người dùng nhập
echo "<h3>Sản phẩm được thêm từ ngày 2023-12-23 đến 2023-12-25:</h3>";
displayProducts($products, "2023-12-23", "2023-12-25");
?>
```

Bài 5. Quản Lý Danh Sách Khách Hàng

Yêu cầu:

- Tạo một mảng lưu thông tin của khách hàng, bao gồm name, email, và phone.
- Viết một hàm để hiển thị danh sách khách hàng theo định dạng:
Tên: [name], Email: [email], Số điện thoại: [phone].
- Tìm kiếm khách hàng theo tên (có phân biệt chữ hoa chữ thường).

Gợi ý:

```
$customers = [  
  ["name" => "Alice", "email" => "alice@example.com", "phone" => "0123456789"],  
  ["name" => "Bob", "email" => "bob@example.com", "phone" => "0987654321"]  
];  
  
function searchCustomer($customers, $searchName) {  
  foreach ($customers as $customer) {  
    if ($customer['name'] === $searchName) {  
      echo "Tìm thấy: Tên: {$customer['name']}, Email: {$customer['email']}, Số điện  
thoại: {$customer['phone']}<br>";  
      return;  
    }  
  }  
  echo "Không tìm thấy khách hàng có tên $searchName.<br>";  
}  
  
searchCustomer($customers, "Alice");
```

Bài 6. Tính Tổng Giá Trị Đơn Hàng

Yêu cầu:

- Tạo một mảng lưu thông tin các sản phẩm trong giỏ hàng, mỗi sản phẩm bao gồm name, price, và quantity.
- Tính và hiển thị tổng giá trị đơn hàng.

Gợi ý:

```
$cart = [  
  ["name" => "Laptop", "price" => 20000000, "quantity" => 1],  
  ["name" => "Điện thoại", "price" => 10000000, "quantity" => 2]  
];  
  
function calculateTotal($cart) {  
  $total = 0;  
  foreach ($cart as $item) {  
    $total += $item['price'] * $item['quantity'];  
  }  
  echo "Tổng giá trị đơn hàng: " . number_format($total) . " VND<br>";  
}  
  
calculateTotal($cart);
```

Bài 7. Quản Lý Lịch Sự Kiện

Yêu cầu:

- Tạo một mảng lưu danh sách các sự kiện, mỗi sự kiện bao gồm name, date, và location.
- Hiện thị tất cả sự kiện sắp tới (ngày lớn hơn hoặc bằng ngày hiện tại).
- Cho phép thêm sự kiện mới vào danh sách.

Gợi ý:

```
$events = [
    ["name" => "Hội chợ công nghệ", "date" => "2023-12-28", "location" => "Hà Nội"],
    ["name" => "Triển lãm sách", "date" => "2023-12-20", "location" => "TP. Hồ Chí Minh"]
];
function displayUpcomingEvents($events) {
    $today = date("Y-m-d");
    echo "Sự kiện sắp tới:<br>";
    foreach ($events as $event) {
        if ($event['date'] >= $today) {
            echo "{$event['name']} - Ngày: {$event['date']} - Địa điểm: {$event['location']}<br>";
        }
    }
}
displayUpcomingEvents($events);
```

Bài 8. Xác Thực Đăng Nhập

Yêu cầu:

- Tạo một mảng lưu danh sách tài khoản gồm username và password.
- Viết chương trình cho phép người dùng nhập thông tin đăng nhập và kiểm tra thông tin có hợp lệ không.

Gợi ý:

```
$accounts = [
    ["username" => "admin", "password" => "12345"],
    ["username" => "user", "password" => "password"]
];
function authenticate($accounts, $username, $password) {
    foreach ($accounts as $account) {
        if ($account['username'] === $username && $account['password'] === $password) {
            echo "Đăng nhập thành công!<br>";
            return;
        }
    }
    echo "Thông tin đăng nhập không hợp lệ.<br>";
}
authenticate($accounts, "admin", "12345");
```

Bài 9. Hiện Thị Bảng Thống Kê

Yêu cầu:

- Tạo một mảng chứa thông tin về nhân viên, bao gồm name, position, salary.
- Hiện thị danh sách nhân viên theo bảng HTML.
- Tính tổng lương và hiện thị ở cuối bảng.

Gợi ý:

```
$employees = [
    ["name" => "Alice", "position" => "Manager", "salary" => 30000000],
    ["name" => "Bob", "position" => "Developer", "salary" => 20000000],
    ["name" => "Charlie", "position" => "Designer", "salary" => 15000000]
];

function displayEmployeeTable($employees) {
    $totalSalary = 0;
    echo "<table border='1'>";
    echo "<tr><th>Tên</th><th>Chức vụ</th><th>Lương</th></tr>";
    foreach ($employees as $employee) {
        echo "<tr>
            <td>{$employee['name']}</td>
            <td>{$employee['position']}</td>
            <td>". number_format($employee['salary']) . " VND</td>
        </tr>";
        $totalSalary += $employee['salary'];
    }
    echo "<tr>
        <td colspan='2'>Tổng lương</td>
        <td>". number_format($totalSalary) . " VND</td>
    </tr>";
    echo "</table>";
}

displayEmployeeTable($employees);
```

Chương 5

Lập trình Web PHP với Cơ sở dữ liệu MySQL

Mục tiêu

Sau khi học xong chương này, người học sẽ có thể:

- Trình bày được khái niệm cơ bản về hệ quản trị cơ sở dữ liệu quan hệ và vai trò của MySQL trong phát triển web động.
- Sử dụng thành thạo các câu lệnh SQL cơ bản như *SELECT*, *INSERT*, *UPDATE*, *DELETE* để thao tác dữ liệu.
- Triển khai PDO (PHP Data Objects) để truy vấn cơ sở dữ liệu một cách an toàn và hiệu quả.
- Xây dựng các chức năng web động có khả năng tương tác với người dùng thông qua FORM, quản lý trạng thái người dùng hiệu quả với Cookie và Session trong PHP.

5.1. Giới thiệu về hệ quản trị cơ sở dữ liệu quan hệ

Hệ quản trị cơ sở dữ liệu (Database Management System - DBMS) là một phần mềm được thiết kế để lưu trữ, quản lý, truy xuất và thực thi các truy vấn trên dữ liệu. DBMS đóng vai trò quan trọng như một giao diện giữa người dùng cuối và cơ sở dữ liệu, cho phép người dùng thực hiện các thao tác như tạo, đọc, cập nhật và xóa dữ liệu trong cơ sở dữ liệu.

DBMS không chỉ giúp người dùng thực hiện các thao tác trên dữ liệu mà còn đảm bảo tính bảo mật và toàn vẹn của dữ liệu. Các hệ thống này cung cấp các cơ chế kiểm soát truy cập, đảm bảo rằng chỉ những người dùng được ủy quyền mới có thể truy cập và chỉnh sửa dữ liệu. Hơn nữa, DBMS còn duy trì tính nhất quán và độ tin cậy của dữ liệu, giúp quản lý dữ liệu một cách thống nhất và hiệu quả.

Đặc Điểm Chính Của DBMS:

- *Tính Nhất Quán*: Đảm bảo dữ liệu luôn ở trạng thái hợp lệ thông qua các ràng buộc và quy tắc.
- *Tính Toàn Vẹn*: Bảo vệ dữ liệu khỏi các lỗi và sự cố, duy trì tính chính xác và đầy đủ của dữ liệu.
- *Tính Bảo Mật*: Cung cấp các cơ chế kiểm soát truy cập và bảo mật để bảo vệ dữ liệu khỏi truy cập trái phép.
- *Tính Độc Lập Dữ Liệu*: Tạo ra sự tách biệt giữa ứng dụng và dữ liệu, cho phép dễ dàng thay đổi cấu trúc dữ liệu mà không ảnh hưởng đến các ứng dụng đang sử dụng dữ liệu đó.

Hiện nay có rất nhiều hệ quản trị cơ sở dữ liệu phổ biến, mỗi loại có những đặc điểm và ứng dụng riêng. Một số hệ quản trị cơ sở dữ liệu thông dụng bao gồm:

- *MySQL*: Thường được lựa chọn trong quá trình phát triển website và xây dựng phần mềm nhờ vào tốc độ xử lý nhanh và tính bảo mật cao. MySQL phù hợp với các ứng dụng có yêu cầu truy cập cơ sở dữ liệu trên internet.

- *SQLite*: Là hệ thống CSDL nhỏ gọn, có thể cài đặt bên trong các ứng dụng nhỏ. SQLite được biết đến với ngôn ngữ lập trình C và sử dụng phổ biến trên hệ điều hành Android.

- *PostgreSQL*: Kết hợp với module PostGIS, cho phép lưu trữ các lớp dữ liệu không gian. PostgreSQL được sử dụng nhiều trong các ứng dụng về bản đồ.

- *Oracle*: Ngoài sản phẩm Oracle Database Server, Oracle còn cung cấp nhiều sản phẩm khác. Oracle là một trong những hệ quản trị cơ sở dữ liệu mạnh mẽ và chuyên nghiệp nhất.

- *MongoDB*: Được viết bởi ngôn ngữ C++. Đây là một mã nguồn mở và cũng là một tập tài liệu dùng cơ chế NoSQL để truy vấn.

- *Redis*: Hệ thống lưu trữ key-value với nhiều tính năng hữu ích, phát triển theo phong cách NoSQL. Redis hỗ trợ nhiều cấu trúc dữ liệu cơ bản và cho phép scripting bằng ngôn ngữ Lua.

DBMS là một phần không thể thiếu trong việc phát triển các ứng dụng web hiện đại, đặc biệt là trong lập trình web PHP. Nó giúp các nhà phát triển dễ dàng quản lý và thao tác dữ liệu, đồng thời đảm bảo rằng dữ liệu luôn được bảo vệ và duy trì ở trạng thái tốt nhất.

5.2. Các câu lệnh thao tác dữ liệu trong SQL

SQL (Structured Query Language) là ngôn ngữ chuẩn để truy vấn và thao tác dữ liệu trong cơ sở dữ liệu quan hệ. Dưới đây là một số câu lệnh SQL cơ bản mà chúng ta cần nắm vững.

Ví dụ: Cho cơ sở dữ liệu gồm có 2 bảng như sau:

Bảng Courses

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
course_id	INT	PRIMARY KEY, AUTO INCREMENT	Mã khóa học duy nhất
course_name	NVARCHAR(100)	NOT NULL	Tên khóa học
credit	INT		Số tín chỉ

Bảng Students

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
id	INT	PRIMARY KEY, AUTO INCREMENT	Mã sinh viên
name	NVARCHAR(100)	NOT NULL	Họ tên sinh viên
age	INT		Tuổi
gender	NVARCHAR(10)		Giới tính
email	VARCHAR(100)	UNIQUE	Email sinh viên
class	VARCHAR(20)		Mã lớp
course_id	INT	FOREIGN KEY courses(course_id)	Mã khóa học đang theo học

5.2.1. SELECT

Trong SQL, câu lệnh SELECT được sử dụng để truy vấn và lấy dữ liệu từ cơ sở dữ liệu. Dưới đây là những trường hợp phổ biến khi sử dụng câu lệnh SELECT trong lập trình web PHP:

a) Truy vấn toàn bộ dữ liệu từ một bảng

Cú pháp:

```
SELECT * FROM table_name;
```

Ví dụ:

```
SELECT * FROM students;
```

b) Truy vấn một số cột từ một bảng

Cú pháp:

```
SELECT column1, column2 FROM table_name;
```

Ví dụ:

```
SELECT name, age FROM students;
```

c) Lọc dữ liệu với mệnh đề WHERE

Cú pháp:

```
SELECT * FROM table_name WHERE condition;
```

Ví dụ:

```
SELECT * FROM students WHERE age > 18;
```

d) Sắp xếp dữ liệu với ORDER BY

Cú pháp:

```
SELECT * FROM table_name ORDER BY column_name ASC|DESC;
```

Ví dụ:

```
SELECT * FROM students ORDER BY name ASC;
```

e) Giới hạn số lượng kết quả với LIMIT

Cú pháp:

```
SELECT * FROM table_name LIMIT number;
```

Ví dụ:

```
SELECT * FROM students LIMIT 5;
```

f) Kết hợp dữ liệu từ nhiều bảng với JOIN

Sử dụng JOIN để kết hợp dữ liệu từ nhiều bảng dựa trên một cột chung: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN là các loại JOIN phổ biến.

Ví dụ:

```
SELECT students.name, courses.course_name  
FROM students  
INNER JOIN courses ON students.course_id = courses.course_id;
```

g) Gom nhóm dữ liệu với GROUP BY

Cú pháp:

```
SELECT column_name(s), aggregate_function(column_name) FROM table_name  
GROUP BY column_name(s);
```

Ví dụ:

```
SELECT class, COUNT(*) FROM students GROUP BY class;
```

h) Lọc dữ liệu sau khi gom nhóm với HAVING

Cú pháp:

```
SELECT column_name(s), aggregate_function(column_name) FROM table_name  
GROUP BY column_name(s) HAVING condition;
```

Ví dụ:

```
SELECT class, COUNT(*) FROM students GROUP BY class HAVING COUNT(*) > 5;
```

5.2.2. INSERT, UPDATE, DELETE

5.2.2.1. INSERT

Câu lệnh INSERT được sử dụng để thêm một hoặc nhiều bản ghi vào bảng.

Cú pháp:

```
INSERT INTO table_name (column1, column2, ...)  
VALUES (value1, value2, ...);
```

Ví dụ:

```
INSERT INTO students (name, age) VALUES ('Huong Tran', 30);
```

5.2.2.2. UPDATE

Câu lệnh UPDATE được sử dụng để cập nhật dữ liệu trong bảng.

Cú pháp:

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

Ví dụ:

```
UPDATE students SET age = 21 WHERE name = 'Huong Tran';
```

5.2.2.3. DELETE

Câu lệnh DELETE được sử dụng để xóa dữ liệu từ bảng.

Cú pháp:

```
DELETE FROM table_name  
WHERE condition;
```

Ví dụ:

```
DELETE FROM students WHERE name = 'Huong Tran';
```

5.3. Giới thiệu MySQL

5.3.1. Khái Niệm MySQL

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, được sử dụng rộng rãi trên thế giới. Nó được sử dụng để lưu trữ và quản lý cơ sở dữ liệu, bao gồm các bảng, cột, dữ liệu và các quan hệ giữa chúng. MySQL cung cấp một công cụ mạnh mẽ để thao tác dữ liệu bằng cách sử dụng ngôn ngữ truy vấn SQL (Structured Query Language), bao gồm các câu lệnh SELECT, INSERT, UPDATE, DELETE và nhiều câu lệnh khác để thao tác với dữ liệu. MySQL thường kết hợp với ngôn ngữ lập trình PHP để xây dựng các ứng dụng web. Với sự hỗ trợ từ MySQL, việc phát triển và triển khai các ứng dụng web trở nên hiệu quả và dễ dàng hơn.

MySQL là một phần của gói phần mềm XAMPP (chữ M trong XAMPP đại diện cho MySQL) hoặc chúng ta cũng có thể tải về miễn phí từ website của nó như sau:

<https://www.mysql.com/>

5.3.2. Lợi Thế Của MySQL

- *Tốc Độ Xử Lý Cao*: MySQL có khả năng xử lý nhanh chóng các truy vấn, đảm bảo hiệu suất cao cho các ứng dụng yêu cầu truy cập dữ liệu liên tục.
- *Dễ Sử Dụng*: MySQL có giao diện dễ sử dụng, hỗ trợ cả người mới bắt đầu và các chuyên gia.
- *Tương Thích Cao*: MySQL tương thích với nhiều hệ điều hành như Linux, Windows, macOS.
- *Mã Nguồn Mở*: MySQL là phần mềm mã nguồn mở, được phát triển và hỗ trợ bởi cộng đồng rộng lớn.
- *Miễn Phí*: MySQL là mã nguồn mở và miễn phí để tải xuống và sử dụng.
- *Hỗ Trợ SQL*: MySQL sử dụng ngôn ngữ truy vấn chuẩn SQL, giúp người dùng dễ dàng viết và hiểu các truy vấn dữ liệu.

5.3.3. Chức Năng Của MySQL

MySQL và PHP tạo thành một cặp đôi mạnh mẽ cho việc phát triển các ứng dụng web. MySQL được nhiều website nổi tiếng như Facebook, Google, Yahoo, Twitter, và YouTube sử dụng để lưu trữ thông tin, cho thấy mức độ phổ biến và tin cậy của nó. MySQL cung cấp nhiều chức năng quan trọng giúp quản lý dữ liệu hiệu quả:

- *Tạo Lập Cơ Sở Dữ Liệu*: MySQL cung cấp ngôn ngữ định nghĩa dữ liệu (DDL) để mô tả, khai báo kiểu dữ liệu và các cấu trúc dữ liệu.

- *Cập Nhật và Khai Thác Dữ Liệu*: MySQL cung cấp ngôn ngữ thao tác dữ liệu (DML) như *INSERT*, *UPDATE*, *DELETE*, và *SELECT* để thêm, sửa, xóa và truy xuất dữ liệu.

- *Điều Khiển Truy Cập*: MySQL đảm bảo an ninh dữ liệu, ngăn chặn truy cập trái phép, duy trì tính nhất quán của dữ liệu, và quản lý các giao dịch để đảm bảo các thao tác trên cơ sở dữ liệu diễn ra một cách toàn vẹn.

5.3.4. Tổ Chức Lưu Trữ Của MySQL

Dữ liệu trong MySQL được lưu trữ theo cấu trúc bảng. Mỗi bảng chứa các bản ghi (record) và mỗi bản ghi bao gồm các trường (field). Các thành phần chính trong tổ chức lưu trữ của MySQL bao gồm:

- *Bảng (Table)*: Tập hợp các dữ liệu có liên quan, gồm các cột (column) và hàng (row).

- *Cơ Sở Dữ Liệu (Database)*: Tập hợp các bảng.

- *Chỉ Mục (Index)*: Cấu trúc dữ liệu giúp tăng tốc độ truy xuất dữ liệu.

- *Khóa Chính (Primary Key)*: Xác định duy nhất mỗi hàng trong bảng.

- *Khóa Ngoại (Foreign Key)*: Liên kết với khóa chính của một bảng khác, đảm bảo tính toàn vẹn tham chiếu giữa các bảng.

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ, phổ biến và đáng tin cậy, đặc biệt phù hợp với lập trình web PHP. Việc hiểu rõ về cách MySQL tổ chức và quản lý dữ liệu sẽ giúp chúng ta tận dụng tối đa sức mạnh của hệ quản trị cơ sở dữ liệu này trong các dự án lập trình web.

5.3.5. Quản lý cơ sở dữ liệu với phpMyAdmin

Để quản lý cơ sở dữ liệu MySQL một cách hiệu quả trong môi trường phát triển web với PHP, chúng ta thường sử dụng công cụ phpMyAdmin. *phpMyAdmin* là một ứng dụng web mã nguồn mở được thiết kế để quản lý và thao tác với cơ sở dữ liệu MySQL thông qua giao diện người dùng đồ họa trên trình duyệt web. Nó cung cấp một giao diện đơn giản để thực hiện các tác vụ như tạo mới cơ sở dữ liệu, thêm, sửa, xóa bảng, thực thi các câu lệnh SQL, sao lưu và khôi phục dữ liệu, quản lý người dùng và phân quyền truy cập vào cơ sở dữ liệu. Trong bài giảng này, chúng ta sẽ sử dụng phpMyAdmin trong XAMPP như sau:

5.3.5.1. Khởi động phpMyAdmin

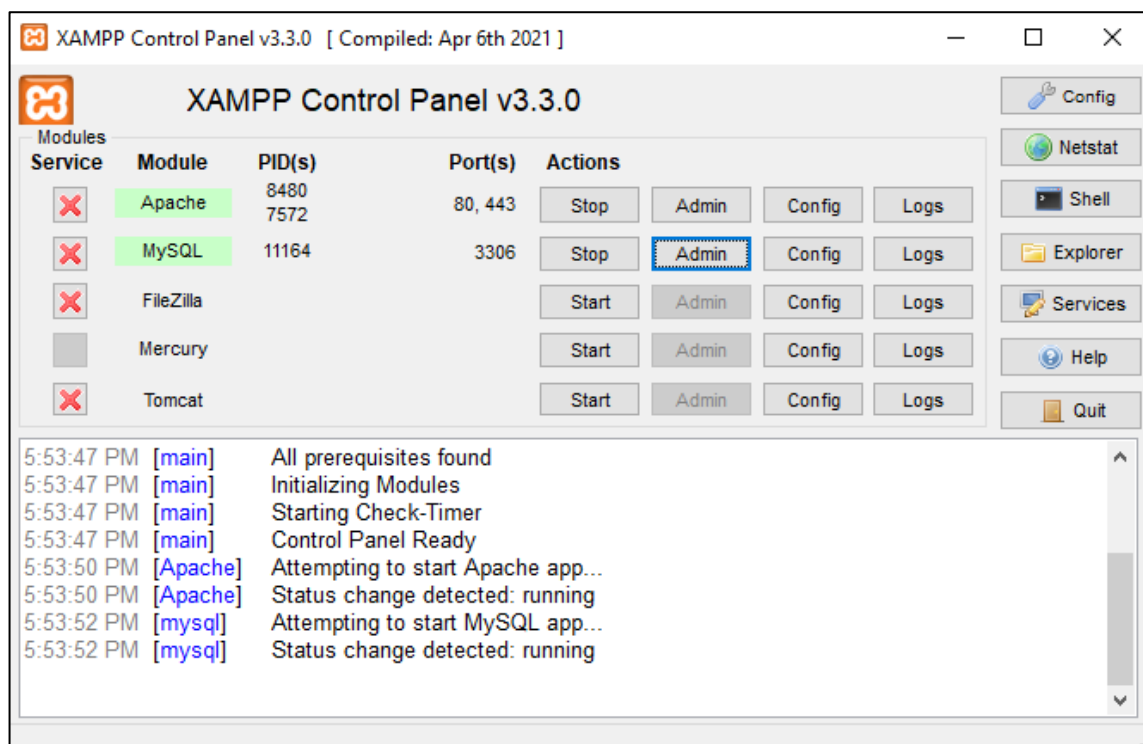
Để làm việc với phpMyAdmin, chúng ta có thể khởi động bảng điều khiển của XAMPP và chọn nút Admin (quản trị) cho MySQL. Nên nhớ rằng phpMyAdmin là một công cụ nền web nên nó cần server web để hoạt động chuẩn xác. Vì vậy, chúng ta cần khởi động server web Apache trước khi sử dụng phpMyAdmin. Tất nhiên, chúng ta cũng phải khởi động server MySQL như Hình 5.1 và Hình 5.2

Một cách để khởi động phpMyAdmin trên hệ thống cục bộ là sử dụng trình duyệt và duyệt URL sau:

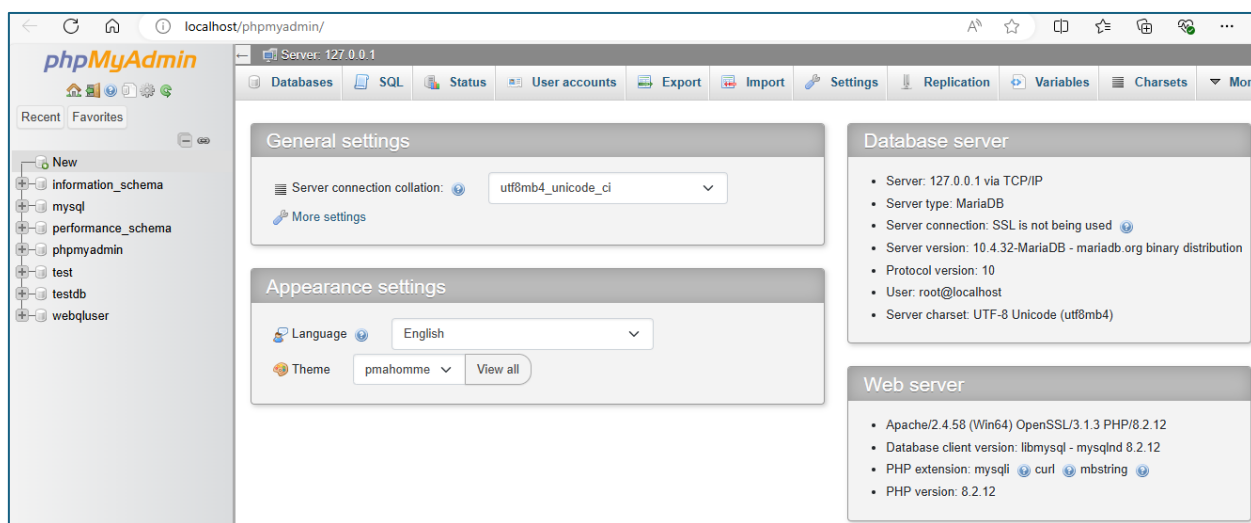
<http://localhost/phpMyAdmin>

Cách này cũng hoạt động nếu MySQL được cài đặt trên máy ở xa và phpMyAdmin cũng được cài đặt trên đó. Trong trường hợp này, chúng ta có thể chạy ứng dụng phpMyAdmin bằng cách nhập địa chỉ đúng của nó trên thanh địa chỉ trình duyệt, ví dụ như sau:

<https://www.example.com/phpMyAdmin>



Hình 5. 1 Khởi động server web Apache



Hình 5. 2 Khởi động phpMyAdmin

5.3.5.2. Cấu hình phpMyAdmin và Quản lý bảo mật

Trong phpMyAdmin, tài khoản 'root' là tài khoản quản trị cấp cao nhất của MySQL. Khi đăng nhập bằng tài khoản này, chúng ta sẽ có quyền truy cập vào tất cả các cơ sở dữ liệu trên máy chủ.

a) Cấu hình phpMyAdmin:

PhpMyAdmin được cấu hình thông qua tập tin "config.inc.php" nằm trong thư mục phpMyAdmin (C:\xampp\phpMyAdmin). Các thiết lập cơ bản bao gồm:

- Cấu hình máy chủ MySQL: giúp phpMyAdmin biết cách kết nối tới máy chủ MySQL.

```
$cfg['Servers'][$i]['host'] = 'localhost';  
$cfg['Servers'][$i]['port'] = '';  
$cfg['Servers'][$i]['user'] = 'root';  
$cfg['Servers'][$i]['password'] = '';  
$cfg['Servers'][$i]['auth_type'] = 'cookie';
```

Trong đó, host: Địa chỉ của máy chủ MySQL (thường là localhost); user: Tên người dùng MySQL; password: Mật khẩu của người dùng MySQL; auth_type: Loại xác thực (nên đặt là 'cookie' để yêu cầu đăng nhập).

- Cấu hình giao diện người dùng:

```
$cfg['DefaultLang'] = 'en';  
$cfg['DefaultDisplay'] = 'horizontal';  
$cfg['MaxRows'] = 50;
```

- Cấu hình bảo mật: không cho phép đăng nhập mà không có mật khẩu.

```
$cfg['Servers'][$i]['AllowNoPassword'] = false;
```

b) Quản lý bảo mật

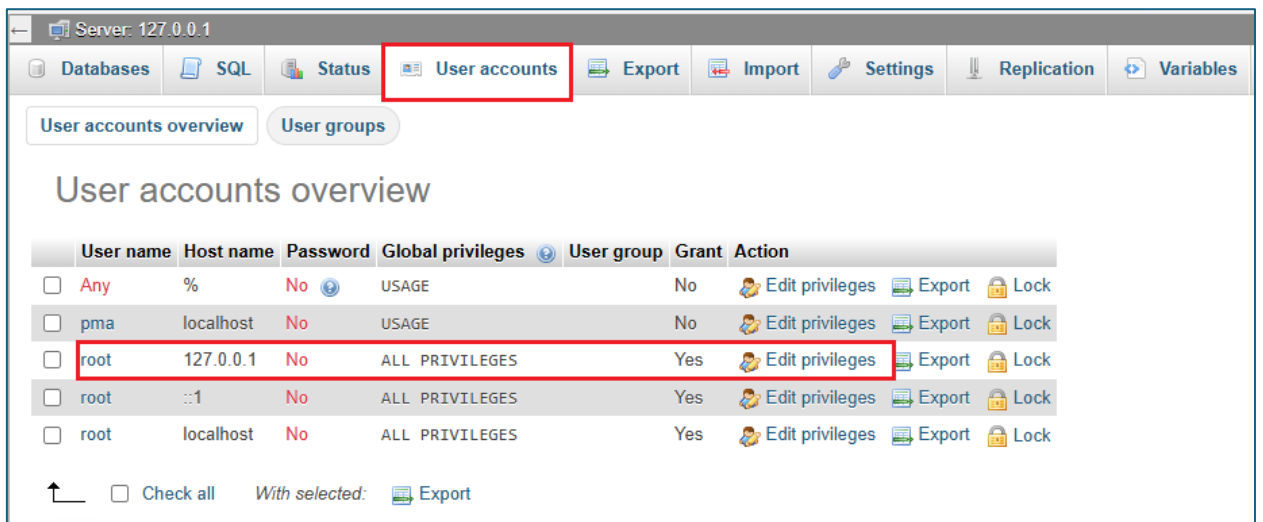
Mặc định, XAMPP không yêu cầu đăng nhập khi truy cập phpMyAdmin sau khi cài đặt mới. Điều này tiềm ẩn rủi ro bảo mật, đặc biệt nếu máy chủ XAMPP có thể truy cập từ mạng công cộng. Để thiết lập mật khẩu và tăng cường bảo mật, chúng ta thực hiện các bước sau:

(1) Thiết lập mật khẩu cho MySQL Root

* Cách 1. Thiết lập mật khẩu bằng phpMyAdmin

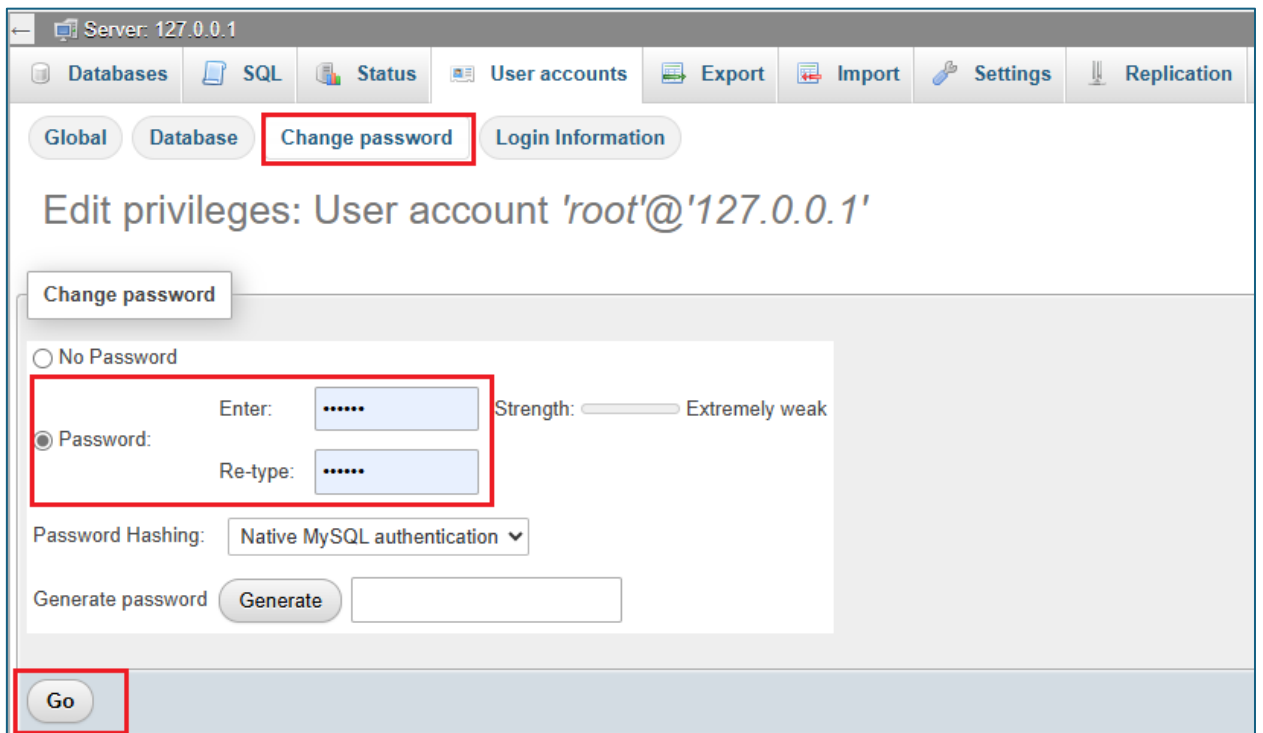
- Truy cập vào <http://localhost/phpmyadmin/>

- Chọn tab User account, trong danh sách người dùng, tìm tài khoản root và chọn Edit Privileges để chỉnh sửa quyền người dùng root, như Hình 5.3



Hình 5. 3 Chỉnh sửa quyền người dùng root

- Chọn tab *Change password*, nhập mật khẩu mới và xác nhận lại mật khẩu, nhấn nút *Go* để lưu thay đổi, như Hình 5.4



Hình 5. 4 Cập nhật mật khẩu cho tài khoản

* Cách 2. Thiết lập mật khẩu cho MySQL từ dòng lệnh (tùy chọn)

Nếu chúng ta không muốn sử dụng phpMyAdmin để thiết lập mật khẩu, thì có thể sử dụng dòng lệnh như sau:

- Mở XAMPP Control Panel, nhấn nút Shell để mở cửa sổ dòng lệnh.
- Chạy lệnh sau để thiết lập mật khẩu:

mysqladmin -u root password 'your_new_password'

(2) Cấu hình file *config.inc.php* để yêu cầu xác thực

- Mở file *config.inc.php* trong thư mục phpMyAdmin của XAMPP.
- Tìm dòng *auth_type* và đổi giá trị thành *cookie* để yêu cầu đăng nhập.

```
$cfg['Servers'][$i]['auth_type'] = 'cookie';
```

- Lưu file *config.inc.php*

Nếu sử dụng xác thực bằng cookie, phpMyAdmin sẽ lưu tên người dùng và mật khẩu vào cookie của trình duyệt. Điều này giúp chúng ta không cần phải nhập lại tài khoản và mật khẩu mỗi lần truy cập phpMyAdmin. Tuy nhiên, chúng ta phải đảm bảo rằng phpMyAdmin và trình duyệt đang sử dụng kết nối HTTPS để bảo mật thông tin đăng nhập được truyền tải qua cookie.

* Xóa Cookie khi không sử dụng như sau:

- Đăng xuất: khi hoàn thành công việc với phpMyAdmin, đảm bảo đăng xuất để xóa cookie khỏi trình duyệt.

- Hoặc xóa cookie thủ công: có thể xóa cookie thủ công từ trình duyệt để đảm bảo thông tin đăng nhập không còn lưu trữ.

Ngoài ra, để đảm bảo an toàn chúng ta cần phải phân quyền truy cập cho file *config.inc.php* chỉ có quyền đọc đối với người dùng và không thể truy cập từ trình duyệt web như sau:

- Click phải vào file *config.inc.php* và chọn Properties.
- Trong tab Security, chọn Edit.
- Chọn người dùng cần thiết (ví dụ: Users), đảm bảo chỉ cấp quyền Read và Read & execute (bỏ chọn quyền Write và Modify).
- Nhấn Apply và Ok để lưu thay đổi.
- Ngăn truy cập từ trình duyệt web thì vào file *httpd.conf* trong thư mục *apache/conf* và thêm vào đoạn mã sau:

```
<Files "config.inc.php">  
    Require all denied  
</Files>
```

5.3.5.3. Hướng dẫn nhập và chạy mã SQL cho việc tạo cơ sở dữ liệu

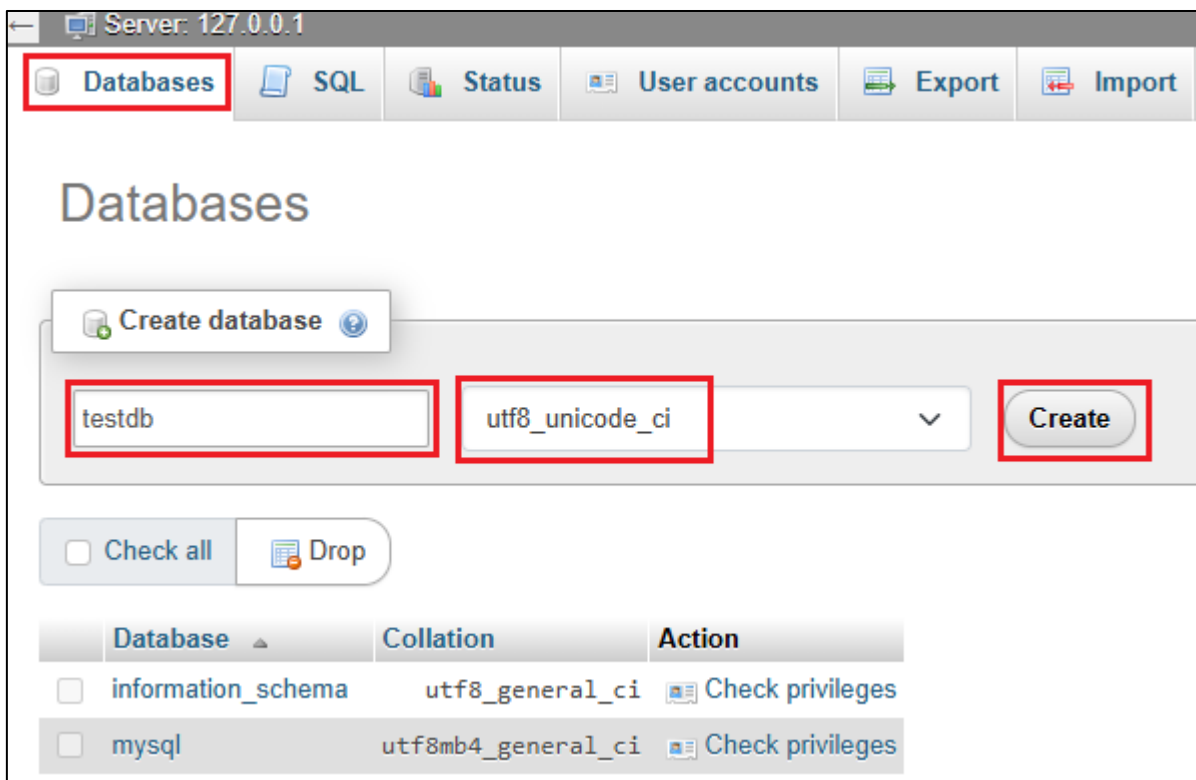
Sau khi khởi động phpMyAdmin và đăng nhập, chúng ta có thể sử dụng nó để chạy mã SQL (SQL script). Mã SQL gồm một hoặc nhiều câu lệnh SQL được lưu trong một file. Những lệnh này có thể là lệnh xử lý dữ liệu mà chúng ta đã được học hay là lệnh định nghĩa dữ liệu như tạo cơ sở dữ liệu, tạo bảng của cơ sở dữ liệu, thêm các dữ liệu ban đầu cho bảng.

a) Tạo Cơ sở dữ liệu mới

* Sử dụng giao diện phpMyAdmin như Hình 5.5

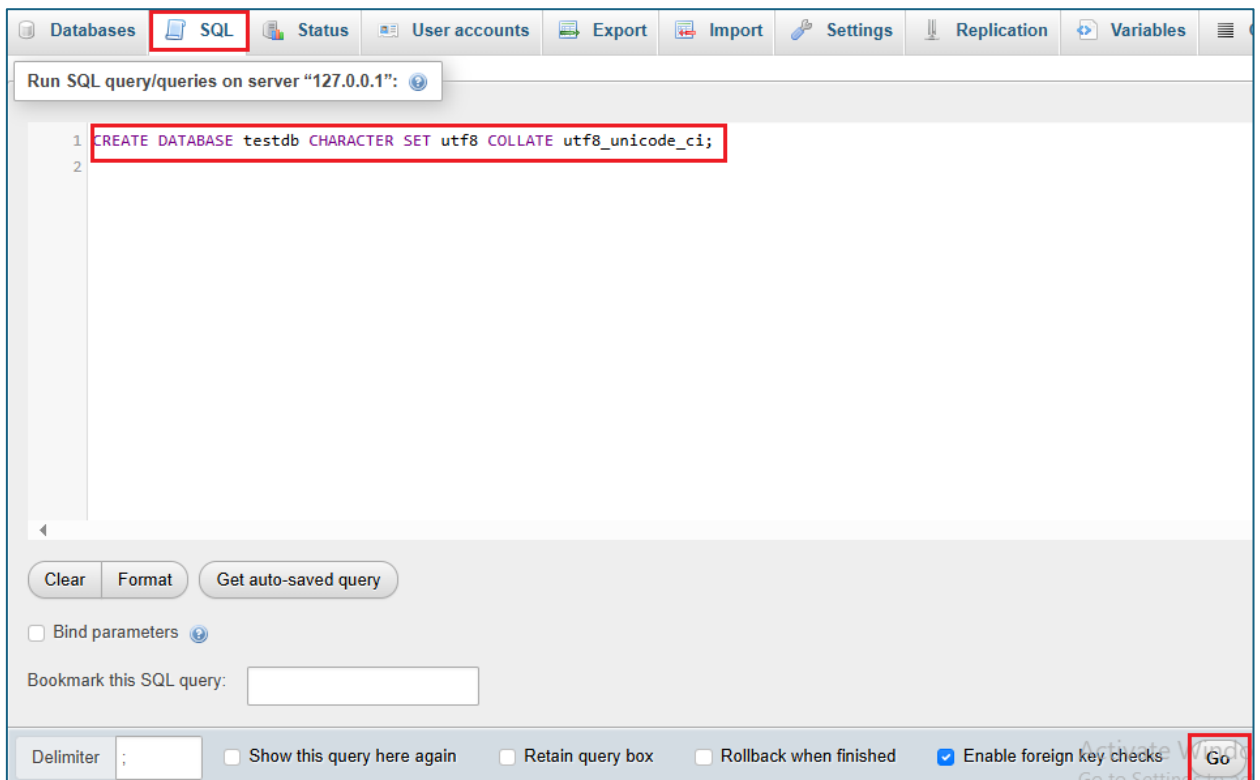
- Từ trang chủ của phpMyAdmin, chọn vào tab *Databases*.

- Trong vùng *Create database*, nhập tên cơ sở dữ liệu mới, chọn *utf8_unicode_ci*
- Nhấn nút *Create* để tạo.



Hình 5. 5 Tạo cơ sở dữ liệu sử dụng giao diện phpMyAdmin

- * Sử dụng mã SQL như Hình 5.6
- Từ trang chủ của phpMyAdmin, chọn vào tab *SQL*
- Nhập mã SQL tạo database
- Nhấn nút *Go* để chạy mã SQL

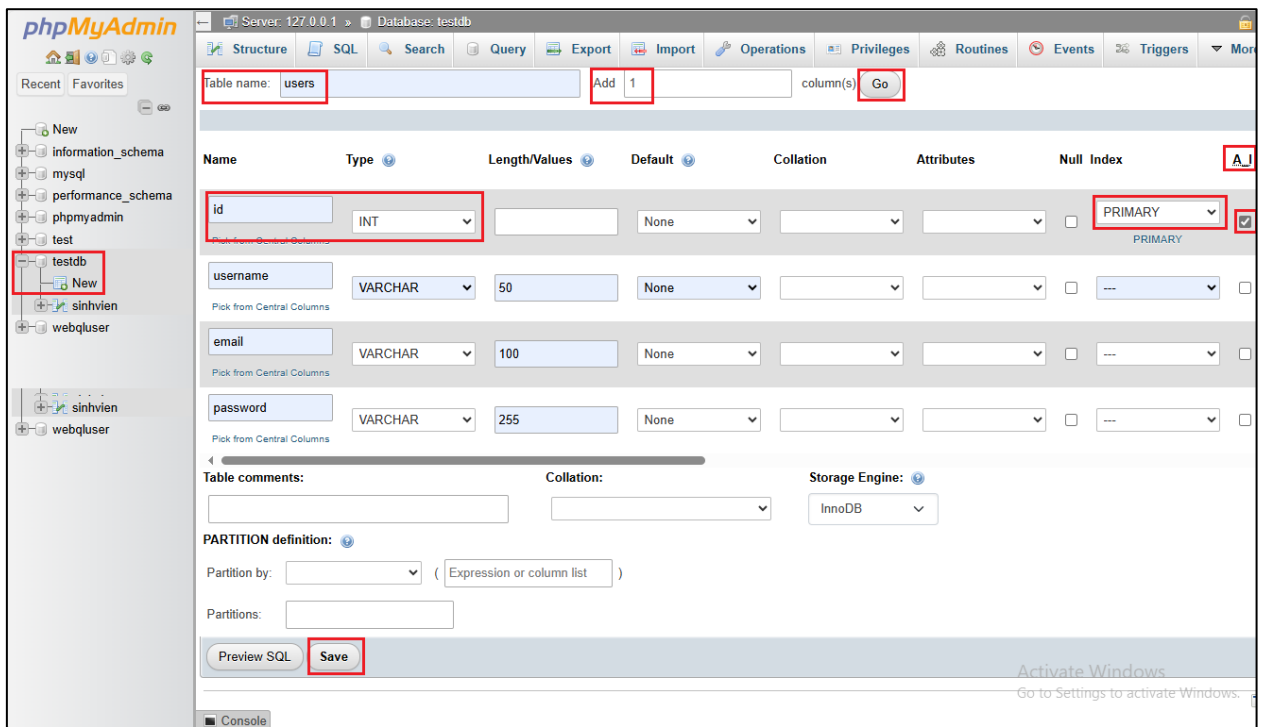


Hình 5. 6 Tạo cơ sở dữ liệu sử dụng mã SQL

b) Tạo Bảng trong Cơ sở dữ liệu

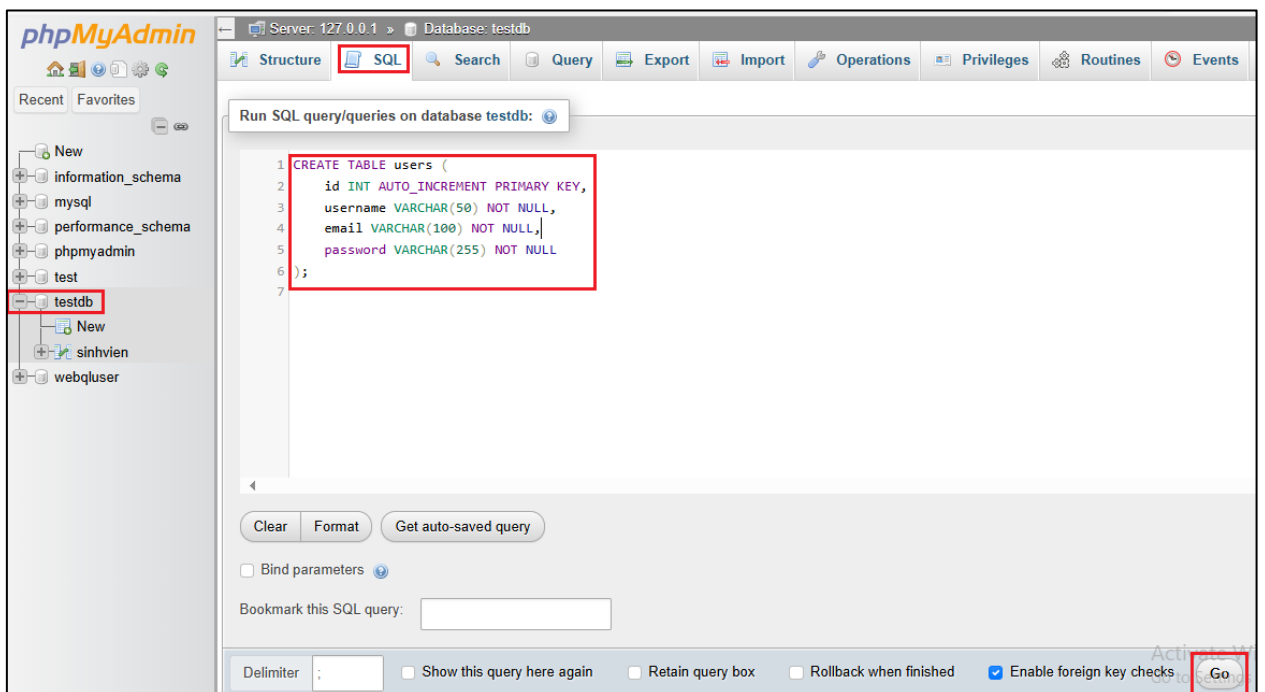
* Sử dụng giao diện phpMyAdmin như Hình 5.7

- Chọn cơ sở dữ liệu mới tạo và nhấn chọn *New* từ danh sách
- Nhập tên bảng mới và số cột
- Nhấn nút *Go* và nhập thông tin cột
- Nhấn *Save* để tạo bảng



Hình 5. 7 Tạo bảng sử dụng giao diện phpMyAdmin

- * Sử dụng mã SQL như Hình 5.8
- Từ trang chủ phpMyAdmin, chọn cơ sở dữ liệu mới tạo
- Nhấn vào *tab SQL* và *nhập mã SQL*
- Nhấn nút *Go* để chạy mã SQL



Hình 5. 8 Tạo bảng sử dụng mã SQL

c) Nhập dữ liệu cho bảng

- * Sử dụng giao diện phpMyAdmin như Hình 5.9
- Chọn bảng cần nhập dữ liệu (user) từ danh sách
- Nhấn chọn tab *Insert*
- Nhập thông tin vào các trường
- Nhấn nút *Go* để thêm dữ liệu vào bảng.

Server: 127.0.0.1 » Database: testdb » Table: users

Recent Favorites

New

- information_schema
- mysql
- performance_schema
- phpmyadmin
- test
- testdb
 - New
 - sinhvien
 - users**
 - webqluser

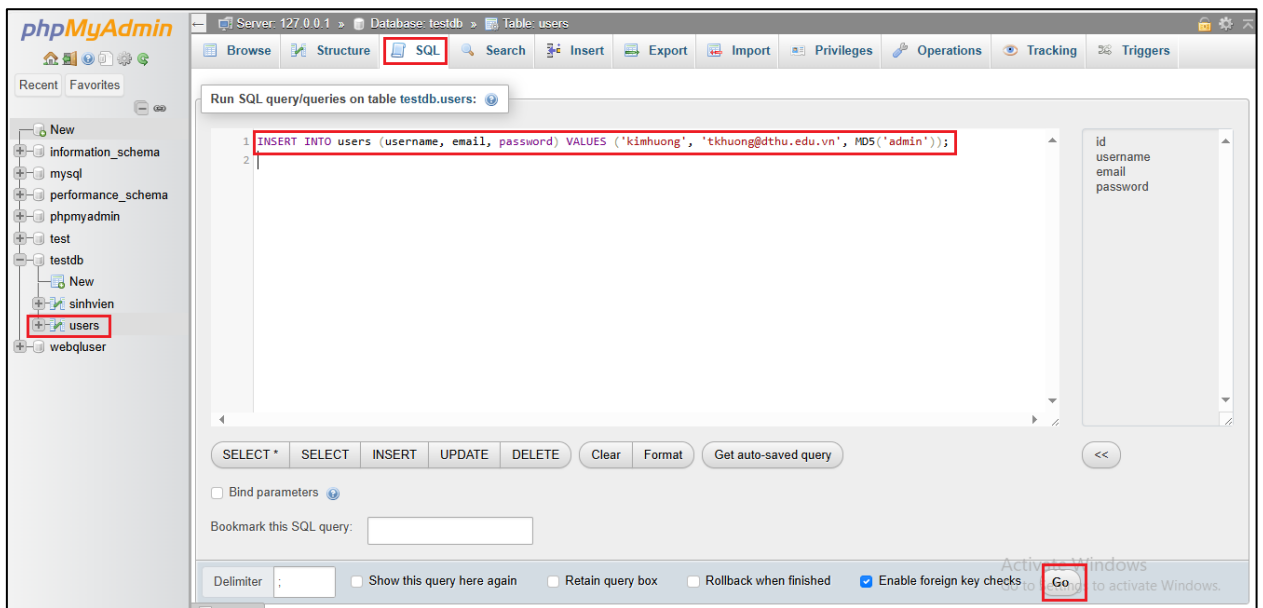
Column Type Function Null Value

Column	Type	Function	Null	Value
id	int(11)			
username	varchar(50)			kimhuong
				tkhuong@dtthu.edu.vn
email	varchar(100)			
				admin
password	varchar(255)	MD5		

Go

Hình 5. 9 Nhập dữ liệu cho bảng sử dụng giao diện phpMyAdmin

- * Sử dụng mã SQL như Hình 5.10
- Chọn bảng cần nhập dữ liệu (user) từ danh sách
- Nhấn chọn tab *SQL* và nhập mã SQL
- Nhấn nút *Go* để chạy mã SQL

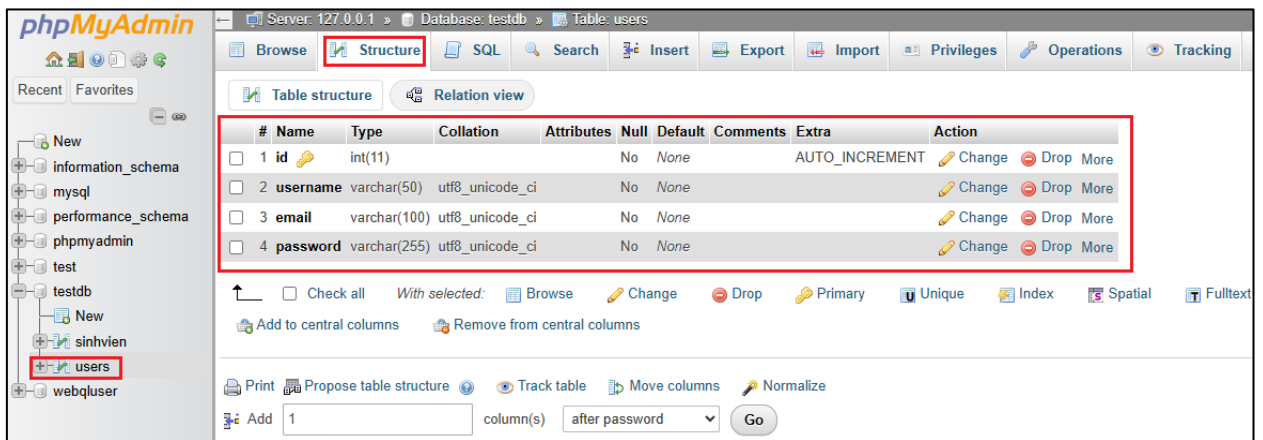


Hình 5. 10 Nhập dữ liệu cho bảng sử dụng mã SQL

5.3.5.4. Hướng dẫn xem dữ liệu và cấu trúc bảng

a) Xem cấu trúc bảng như Hình 5.11

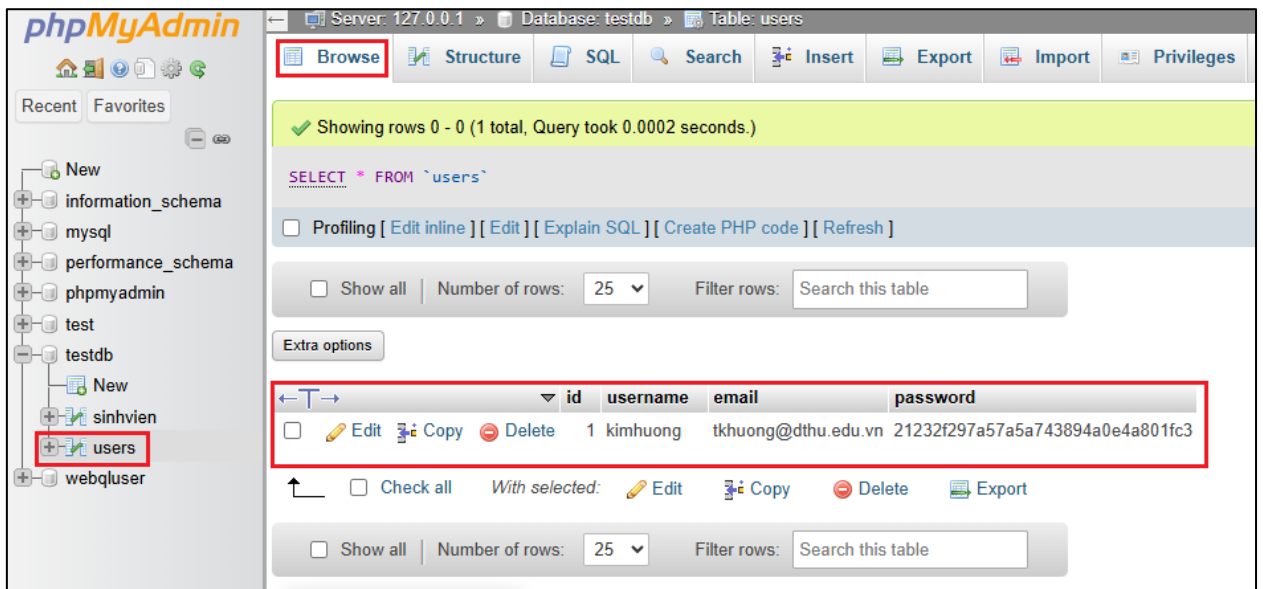
- Chọn cơ sở dữ liệu cần xem, nhấn vào tên bảng
- Chọn tab *Structure* để xem cấu trúc bảng



Hình 5. 11 Xem cấu trúc bảng

b) Xem dữ liệu trong bảng như Hình 5.12

- Chọn bảng muốn xem
- Chọn tab *Browse*



Hình 5. 12 Xem dữ liệu bảng

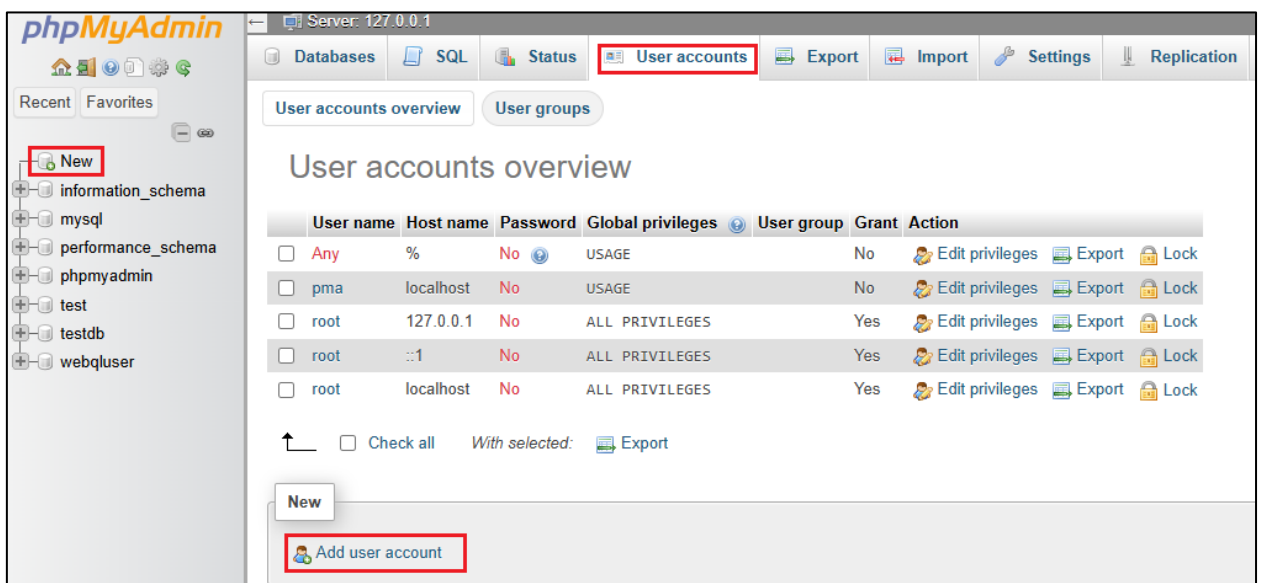
5.3.5.6. Hướng dẫn tạo tài khoản người dùng với quyền hạn chế

Khi làm việc với MySQL, chúng ta thường sử dụng tài khoản “root” - tài khoản quản trị cao cấp nhất. Tài khoản này có toàn quyền thực hiện mọi tác vụ trên bất kỳ cơ sở dữ liệu nào, bao gồm tạo và xóa toàn bộ cơ sở dữ liệu hoặc bảng. Tuy nhiên, để tránh các tình huống người dùng vô tình hoặc cố ý thay đổi dữ liệu quan trọng, chúng ta nên tạo thêm tài khoản người dùng với quyền hạn chế.

Để tạo người dùng mới với quyền hạn chế trong phpMyAdmin, thực hiện các bước sau:

(1) Tạo tài khoản người dùng mới như Hình 5.13

- Đăng nhập bằng tài khoản quản trị (root);
- Chọn tab User Accounts;
- Nhấn vào Add user account để bắt đầu tạo tài khoản mới.



Hình 5. 13 Tạo tài khoản người dùng

(2) Nhập thông tin người dùng như Hình 5.14

- User name: nhập tên người dùng;
- Host name: chọn localhost để giới hạn quyền truy cập từ máy cục bộ;
- Password: nhập mật khẩu cho tài khoản;
- Re-type: nhập lại mật khẩu để xác nhận.

(3) Thiết lập quyền hạn chế như Hình 5.14

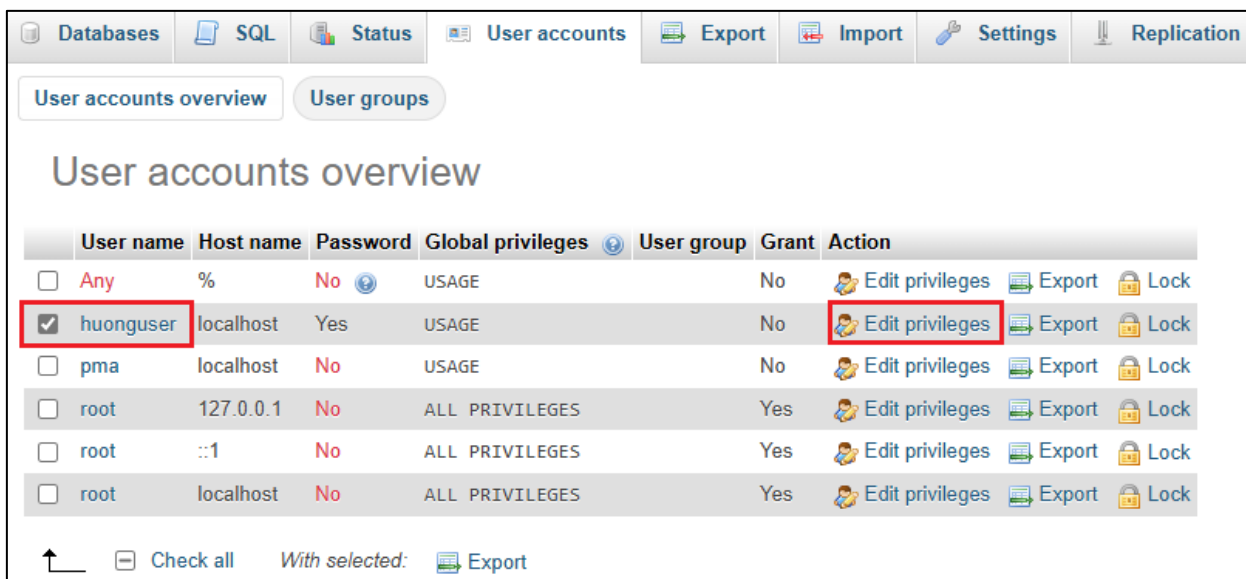
- Bỏ chọn mục *Grant all privileges on wildcard name (username_%)* để không cấp quyền cao nhất cho tài khoản.
- Trong phần *Global privilege*, bỏ chọn tất cả các quyền để không cấp quyền toàn cầu.
- Nhấn nút *Go* để tạo user mới

The screenshot shows the 'Add user account' interface. The top navigation bar includes 'Databases', 'SQL', 'Status', 'User accounts', 'Export', 'Import', and 'Settings'. The main title is 'Add user account'. Below it, the 'Login Information' section contains fields for 'User name' (set to 'huonguser'), 'Host name' (set to 'localhost'), 'Password' (masked with '.....'), and 'Re-type' (masked with '.....'). A 'Strength' indicator shows 'Extremely weak'. The 'Authentication plugin' is set to 'Native MySQL authentication'. There is a 'Generate password' button and an empty text field. The 'Database for user account' section has two checkboxes: 'Create database with same name and grant all privileges.' (unchecked) and 'Grant all privileges on wildcard name (username_%)' (unchecked). The 'Global privileges' section has a 'Check all' checkbox (unchecked). A 'Console' tab is visible at the bottom left.

Hình 5. 14 Nhập thông tin người dùng và thiết lập quyền hạn chế

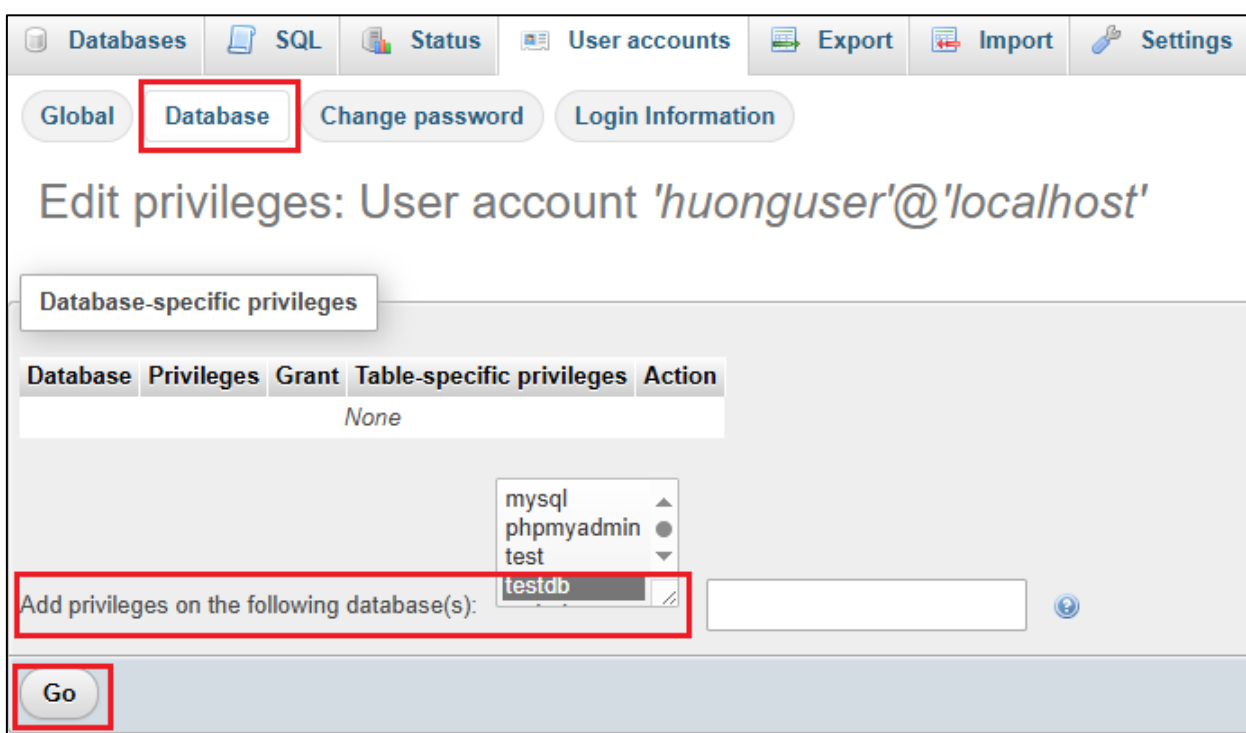
(4) Thiết lập quyền trên Cơ sở dữ liệu cụ thể như Hình 5.15

- Chọn *user* cần thiết lập quyền trên cơ sở dữ liệu (huonguser)
- Nhấn vào *Edit privileges* để thiết lập quyền



Hình 5. 15 Thiết lập quyền trên Cơ sở dữ liệu

- Nhấn tab Database, chọn cơ sở dữ liệu muốn thiết lập như Hình 5.16
- Nhấn nút Go để thực hiện



Hình 5. 16 Chọn Cơ sở dữ liệu để thiết lập quyền

- Trong phần *Database-specific privileges*, chọn quyền hạn chế. Ví dụ: huonguser: Chỉ chọn quyền SELECT, INSERT, UPDATE, DELETE cho tất cả các bảng trong cơ sở dữ liệu như Hình 5.17

- Nhấn nút Go để thiết lập quyền cho user trên cơ sở dữ liệu.

Databases SQL Status User accounts Export Import Settings Replication

Database Table Routine Login Information

Edit privileges: User account 'huonguser'@'localhost' - Database testdb

Database-specific privileges ☐ Check all

Note: MySQL privilege names are expressed in English.

☒ Data ☐ Structure ☐ Administration

☒ SELECT ☐ CREATE ☐ GRANT

☒ INSERT ☐ ALTER ☐ LOCK TABLES

☒ UPDATE ☐ INDEX ☐ REFERENCES

☒ DELETE ☐ DROP ☐

☐ CREATE TEMPORARY TABLES

☐ SHOW VIEW

☐ CREATE ROUTINE

☐ ALTER ROUTINE

☐ EXECUTE

☐ CREATE VIEW

☐ EVENT

☐ TRIGGER

Go

Hình 5. 17 Chọn các quyền cho Cơ sở dữ liệu

5.4. Kết nối PHP với cơ sở dữ liệu MySQL

Trong phần này, chúng ta sẽ sử dụng cách kết nối PHP với Cơ sở dữ liệu MySQL sử dụng PDO (PHP Data Objects). PDO định nghĩa một giao diện thống nhất cho việc truy cập các cơ sở dữ liệu khác nhau. PDO hỗ trợ nhiều loại cơ sở dữ liệu như MySQL, PostgreSQL, SQLite, và nhiều loại khác. PDO có sẵn trong PHP 5.1 và nằm trong gói mở rộng PECL của PHP 5.0. Tuy nhiên, PDO không làm việc với các phiên bản cũ hơn của PHP.

5.4.1. Cách kết nối PHP với MySQL sử dụng PDO

Cú pháp kết nối:

Để kết nối PHP với cơ sở dữ liệu MySQL sử dụng PDO, cần tạo một đối tượng PDO bằng cách sử dụng cú pháp sau:

```
$pdo = new PDO($dsn, $username, $password, $options);
```

Trong đó:

- \$dsn (Data Source Name): Là một chuỗi mô tả thông tin cần thiết để kết nối đến cơ sở dữ liệu. Đối với MySQL, chuỗi này thường có dạng:

`$dsn = "mysql:host=$host;dbname=$dbname;charset=$charset";`

(*\$host*: Địa chỉ máy chủ MySQL, 'localhost' nếu máy chủ MySQL chạy trên cùng máy với PHP; *\$dbname*: Tên cơ sở dữ liệu mà bạn muốn kết nối; *\$charset*: Bộ ký tự sử dụng trong kết nối, để đảm bảo hỗ trợ tốt cho tiếng Việt và các ngôn ngữ khác, nên sử dụng 'utf8mb4'; *\$options*: Tùy chọn bổ sung, thường bao gồm các tùy chọn xử lý lỗi và chế độ truy vấn.)

- `$username`: Tên người dùng để kết nối đến cơ sở dữ liệu.

- `$password`: Mật khẩu của người dùng.

- `$options`: Một mảng chứa các tùy chọn bổ sung cho kết nối PDO. Ví dụ: cấu hình chế độ lỗi, chế độ truy xuất dữ liệu, v.v.

Ví dụ:

```
<?php
const _HOST = 'localhost';
const _DB = 'testdb';
const _USER = 'root';
const _PASS = '';
const _CHARSET = 'utf8mb4';

$dsn = "mysql:host=_HOST; dbname=_DB; charset=_CHARSET ";
$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES => false,
];

try {
    $conn = new PDO($dsn, _USER, _PASS, $options);
    echo "Kết nối thành công!";
} catch (Exception $exception) {
    echo $exception->getMessage() . '<br>';
    die(); // Kết thúc chương trình nếu có lỗi
}
?>
```

Giải thích:

+ `PDO::ATTR_ERRMODE`: Đặt chế độ xử lý lỗi của PDO thành ném ngoại lệ (`PDO::ERRMODE_EXCEPTION`);

+ `PDO::ATTR_DEFAULT_FETCH_MODE`: Đặt chế độ lấy dữ liệu mặc định thành dạng mảng kết hợp (`PDO::FETCH_ASSOC`);

+ `PDO::ATTR_EMULATE_PREPARES`: Tắt giả lập chuẩn bị truy vấn của PDO.

5.4.2. Thực thi câu lệnh SELECT

Sau khi kết nối thành công với cơ sở dữ liệu, bước tiếp theo là thực thi các câu lệnh SQL để thao tác với dữ liệu. Câu lệnh SELECT được sử dụng để truy vấn dữ liệu từ cơ sở

dữ liệu. Trong PHP, chúng ta có thể sử dụng PDO để thực thi các câu lệnh *SELECT* một cách an toàn và hiệu quả.

Cú pháp:

```
$query = "SELECT * FROM table_name WHERE column_name = :value";  
$stmt = $conn->prepare($query);  
$stmt->execute(['value' => $variable]);  
$results = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Trong đó:

- + `prepare($query)`: Chuẩn bị câu lệnh SQL với tham số `:value`.
- + `execute(['value' => $variable])`: Thực thi câu lệnh SQL với giá trị thực tế cho tham số `:value`.
- + `fetchAll(PDO::FETCH_ASSOC)`: Lấy tất cả kết quả dưới dạng mảng kết hợp.

Ví dụ:

```
try {  
    $conn = new PDO($dsn, $username, $password, $options);  
    $query = "SELECT * FROM users WHERE username = :username";  
    $stmt = $conn->prepare($query);  
    $stmt->execute(['username' => 'admin']);  
    $results = $stmt->fetchAll(PDO::FETCH_ASSOC);  
  
    foreach ($results as $row) {  
        echo 'Username: ' . $row['username'] . ', Email: ' . $row['email'] . '<br>';  
    }  
} catch (PDOException $e) {  
    echo 'Lỗi: ' . $e->getMessage();  
}
```

5.4.3. Thực thi câu lệnh INSERT, UPDATE, DELETE

Các câu lệnh *INSERT*, *UPDATE* và *DELETE* được sử dụng để thao tác dữ liệu trong cơ sở dữ liệu. Chúng cho phép thêm mới, cập nhật và xóa dữ liệu. Trong PHP, chúng ta có thể sử dụng PDO để thực thi các câu lệnh này một cách bảo mật và hiệu quả.

5.4.3.1. Thực thi câu lệnh INSERT

Cú pháp:

```
$query = "INSERT INTO table_name (column1, column2) VALUES (:value1, :value2)";  
$stmt = $pdo->prepare($query);  
$stmt->execute(['value1' => $variable1, 'value2' => $variable2]);
```

Ví dụ:

```

try {
    $conn = new PDO($dsn, $username, $password, $options);
    $query = "INSERT INTO users (username, email, password) VALUES (:username, :email, :password)";
    $stmt = $conn->prepare($query);
    $stmt->execute([
        'username' => 'kimhuongr',
        'email' => 'tkhuong@dthu.edu.vn',
        'password' => password_hash('password', PASSWORD_BCRYPT)
    ]);
    echo "User added successfully!";
} catch (PDOException $e) {
    echo 'Lỗi: '. $e->getMessage();
}

```

5.4.3.2. Thực thi câu lệnh UPDATE

Cú pháp:

```

$query = "UPDATE table_name SET column1 = :value1, column2 = :value2 WHERE id = :id";
$stmt = $conn->prepare($query);
$stmt->execute(['value1' => $variable1, 'value2' => $variable2, 'id' => $id]);

```

Ví dụ:

```

try {
    $conn = new PDO($dsn, $username, $password, $options);
    $query = "UPDATE users SET email = :email, password = :password WHERE username = :username";
    $stmt = $conn->prepare($query);
    $stmt->execute([
        'email' => 'updateduser@example.com',
        'password' => password_hash('newpassword', PASSWORD_BCRYPT),
        'username' => 'newuser'
    ]);
    echo "User updated successfully!";
} catch (PDOException $e) {
    echo 'Lỗi: '. $e->getMessage();
}

```

5.4.3.3. Thực thi câu lệnh DELETE

Cú pháp:

```

$query = "DELETE FROM table_name WHERE column_name = :value";
$stmt = $conn->prepare($query);
$stmt->execute(['value' => $variable]);

```

Ví dụ:

```
try{
    $conn = new PDO($dsn, $username, $password, $options);
    $query = "DELETE FROM users WHERE username = :username";
    $stmt = $conn->prepare($query);
    $stmt->execute(['username' => 'newuser']);
    echo "User deleted successfully!";
} catch (PDOException $e) {
    echo 'Lỗi: ' . $e->getMessage();
}
```

5.5. Xử lý ngoại lệ với Try/ Catch

Khi làm việc với cơ sở dữ liệu, các lỗi và ngoại lệ có thể xảy ra, chẳng hạn như lỗi kết nối hoặc lỗi truy vấn. Việc sử dụng khối *try/catch* giúp chúng ta xử lý các ngoại lệ này một cách an toàn và hiệu quả.

Cú pháp:

```
try{
    //Mã có thể gây ra ngoại lệ
} catch (PDOException $e) {
    echo 'Lỗi: ' . $e->getMessage();
}
```

Trong đó:

+ try: Bắt đầu một khối mã mà có thể phát sinh ngoại lệ. Nếu có lỗi xảy ra trong khối này, một ngoại lệ sẽ được ném ra.

+ catch (PDOException \$e): Bắt ngoại lệ của loại PDOException (các lỗi liên quan đến PDO). Biến \$e chứa thông tin về ngoại lệ.

+ echo 'Lỗi: ' . \$e->getMessage();: Hiển thị thông báo lỗi từ ngoại lệ. getMessage() là một phương thức của lớp ngoại lệ, trả về thông báo lỗi.

Trong PHP có các lớp ngoại lệ như sau:

a) *Exception*:

Đây là lớp ngoại lệ cơ bản trong PHP, tất cả các ngoại lệ khác đều kế thừa từ lớp này. Gồm các phương thức sau:

- getMessage(): Lấy thông báo lỗi.
- getCode(): Lấy mã lỗi.
- getFile(): Lấy tên file nơi xảy ra lỗi.
- getLine(): Lấy số dòng nơi xảy ra lỗi.

- `getTrace()`: Lấy mảng truy vết lỗi.
- `getTraceAsString()`: Lấy truy vết lỗi dưới dạng chuỗi.

Ví dụ:

```
<?php
// Một hàm để gây ra lỗi bằng cách chia cho số không
function divide($numerator, $denominator) {
    if ($denominator == 0) {
        throw new Exception("Chia cho số không không hợp lệ.");
    }
    return $numerator / $denominator;
}

try {
    // Gọi hàm với phân số là 10 chia cho 0
    echo divide(10, 0);
} catch (Exception $e) {
    // Hiển thị thông tin lỗi chi tiết
    echo 'Lỗi: ' . $e->getMessage() . '<br>'; // Thông báo lỗi
    echo 'Mã lỗi: ' . $e->getCode() . '<br>'; // Mã lỗi (trong ví dụ này, mã lỗi mặc định là 0)
    echo 'Tập tin lỗi: ' . $e->getFile() . '<br>'; // Tên tập tin nơi xảy ra lỗi
    echo 'Dòng lỗi: ' . $e->getLine() . '<br>'; // Số dòng nơi xảy ra lỗi
    echo 'Truy vết lỗi: ' . print_r($e->getTrace(), true) . '<br>'; // Truy vết lỗi
}
?>
```

b) PDOException:

Đây là một lớp con của Exception, đại diện cho các lỗi xảy ra trong khi sử dụng PDO (PHP Data Objects), kế thừa tất cả các phương thức từ Exception, nhưng có thể chứa thêm thông tin đặc biệt về lỗi PDO. Thường được dùng để bắt và xử lý các lỗi liên quan đến kết nối cơ sở dữ liệu, truy vấn SQL không thành công.

Ví dụ:

```
<?php
// Thay đổi thông tin kết nối với cơ sở dữ liệu để gây lỗi
$dsn = 'mysql:host=localhost;dbname=nonexistentdb'; // Sử dụng CSDL không tồn tại
$username = 'root';
$password = '';

// Thử kết nối đến cơ sở dữ liệu
try {
    $pdo = new PDO($dsn, $username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // Thực hiện một câu lệnh SQL không hợp lệ
    $query = "SELECT * FROM users";
    $stmt = $pdo->query($query);
} catch (PDOException $e) {
```

```

// Hiển thị thông tin lỗi chi tiết
echo 'Lỗi: ' . $e->getMessage() . '<br>'; // Thông báo lỗi
echo 'Mã lỗi: ' . $e->getCode() . '<br>'; // Mã lỗi PDOException
echo 'Tập tin lỗi: ' . $e->getFile() . '<br>'; // Tên tập tin nơi xảy ra lỗi
echo 'Dòng lỗi: ' . $e->getLine() . '<br>'; // Số dòng nơi xảy ra lỗi
echo 'Truy vết lỗi: ' . print_r($e->getTrace(), true) . '<br>'; // Truy vết lỗi
}
?>

```

Ví dụ:

```

<?php
$dsn = 'mysql:host=localhost;dbname=testdb';
$username = 'root';
$password = '';

try{
    // Kết nối đến cơ sở dữ liệu
    $pdo = new PDO($dsn, $username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Thực hiện một câu lệnh SQL
    $query = "SELECT * FROM users WHERE id = :id";
    $stmt = $pdo->prepare($query);
    $stmt->execute(['id' => 1]);

    // Hiển thị kết quả
    $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
    foreach ($results as $row) {
        echo 'Username: ' . $row['username'] . ', Email: ' . $row['email'] . '<br>';
    }
} catch (PDOException $e) {
    // Xử lý lỗi PDOException
    echo 'Lỗi PDO: ' . $e->getMessage() . '<br>';
    echo 'Mã lỗi: ' . $e->getCode() . '<br>';
    echo 'Tập tin lỗi: ' . $e->getFile() . '<br>';
    echo 'Dòng lỗi: ' . $e->getLine() . '<br>';
    echo 'Truy vết lỗi: ' . print_r($e->getTrace(), true) . '<br>';
} catch (Exception $e) {
    // Xử lý các lỗi khác
    echo 'Lỗi: ' . $e->getMessage() . '<br>';
    echo 'Mã lỗi: ' . $e->getCode() . '<br>';
    echo 'Tập tin lỗi: ' . $e->getFile() . '<br>';
    echo 'Dòng lỗi: ' . $e->getLine() . '<br>';
    echo 'Truy vết lỗi: ' . print_r($e->getTrace(), true) . '<br>';
}
?>

```

5.6. Truy vấn dữ liệu bằng PDO

Trong các phần trước, chúng ta đã tìm hiểu cách kết nối với cơ sở dữ liệu MySQL sử dụng PDO và cách thực thi các câu lệnh SQL cơ bản như SELECT, INSERT, UPDATE, và DELETE. Tiếp theo, chúng ta sẽ tập trung vào việc truy vấn dữ liệu bằng PDO, một phần quan trọng của việc làm việc với cơ sở dữ liệu trong PHP. Mục tiêu là làm rõ cách thức sử dụng PDO để thực hiện các câu lệnh SELECT và cách xử lý kết quả trả về từ các câu lệnh này.

PDO cung cấp một cách tiếp cận linh hoạt và bảo mật khi thực hiện các câu lệnh SQL. Sử dụng PDO, chúng ta có thể thực hiện các truy vấn dữ liệu một cách hiệu quả và dễ dàng quản lý kết quả trả về. Phương thức *prepare()* kết hợp với *execute()* trong PDO giúp bảo vệ ứng dụng khỏi SQL Injection và cung cấp khả năng xử lý các dữ liệu động một cách an toàn.

Để truy vấn dữ liệu bằng PDO, chúng ta thường thực hiện các bước sau:

- Chuẩn bị câu lệnh SQL: Sử dụng phương thức *prepare()* để chuẩn bị câu lệnh SQL với các tham số thay thế.

- Thực thi câu lệnh SQL: Sử dụng phương thức *execute()* để thực thi câu lệnh đã chuẩn bị.

- Lấy kết quả: Sử dụng các phương thức của đối tượng *PDOStatement* để lấy kết quả từ truy vấn.

PDO cung cấp một số phương thức để lấy kết quả từ câu lệnh SELECT. Dưới đây là cách sử dụng các phương thức phổ biến:

- ***fetch()***: Trả về một hàng dữ liệu từ kết quả truy vấn, chúng ta có thể chỉ định kiểu dữ liệu trả về bằng cách sử dụng hằng số như `PDO::FETCH_ASSOC` (mảng liên kết) hoặc `PDO::FETCH_OBJ` (đối tượng).

```
$row = $stmt->fetch(PDO::FETCH_ASSOC);
```

- ***fetchAll()***: Trả về tất cả các hàng dữ liệu từ kết quả truy vấn và trả về dưới dạng một mảng. Chúng ta có thể chỉ định kiểu dữ liệu trả về bằng cách sử dụng hằng số như `PDO::FETCH_ASSOC` (mảng liên kết) hoặc `PDO::FETCH_NUM` (mảng số).

```
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

- ***fetchColumn()***: Trả về giá trị của cột chỉ định trong hàng hiện tại của kết quả. Thường được sử dụng khi chúng ta chỉ cần lấy một giá trị duy nhất, chẳng hạn như tổng số bản ghi.

```
$columnValue = $stmt->fetchColumn();
```

- ***fetchObject()***: Trả về một hàng dữ liệu dưới dạng đối tượng, chúng ta có thể chỉ định lớp để ánh xạ dữ liệu vào lớp đó, giúp dễ dàng truy cập dữ liệu bằng các thuộc tính.

```
$object = $stmt->fetchObject();
```

Ví dụ:

```
<?php
// Cấu hình kết nối
```

```

$dsn = 'mysql:host=localhost;dbname=testdb';
$username = 'root';
$password = '';
$options = [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION];

try{
    // Tạo đối tượng PDO
    $pdo = new PDO($dsn, $username, $password, $options);

    // Thực hiện câu lệnh SELECT
    $query = "SELECT id, username, email FROM users";
    $stmt = $pdo->query($query);

    // 1. Sử dụng fetch() để lấy từng hàng dữ liệu
    echo "<h3>Using fetch()</h3>";
    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
        echo 'ID: ' . $row['id'] . ', Username: ' . $row['username'] . ', Email: ' . $row['email'] . '<br>';
    }

    // Đặt lại con trỏ của kết quả để thực hiện lại câu lệnh SELECT
    $stmt->execute();

    // 2. Sử dụng fetchAll() để lấy tất cả các hàng dữ liệu
    echo "<h3>Using fetchAll()</h3>";
    $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
    foreach ($rows as $row) {
        echo 'ID: ' . $row['id'] . ', Username: ' . $row['username'] . ', Email: ' . $row['email'] . '<br>';
    }

    // Đặt lại con trỏ của kết quả để thực hiện lại câu lệnh SELECT
    $stmt->execute();

    // 3. Sử dụng fetchColumn() để lấy một cột duy nhất
    echo "<h3>Using fetchColumn()</h3>";
    $query = "SELECT COUNT(*) FROM users";
    $stmt = $pdo->query($query);
    $userCount = $stmt->fetchColumn();
    echo 'Total number of users: ' . $userCount . '<br>';

    // Đặt lại con trỏ của kết quả để thực hiện lại câu lệnh SELECT
    $stmt->execute();

    // 4. Sử dụng fetchObject() để lấy dữ liệu dưới dạng đối tượng
    echo "<h3>Using fetchObject()</h3>";
    class User {
        public $id;
        public $username;
        public $email;
    }

```



```

    }

    $query = "SELECT id, username, email FROM users";
    $stmt = $pdo->query($query);
    while ($user = $stmt->fetchObject('User')) {
        echo 'ID: ' . $user->id . ', Username: ' . $user->username . ', Email: ' . $user->email . '<br>';
    }

} catch (PDOException $e) {
    echo 'Lỗi: ' . $e->getMessage();
}
?>

```

5.7. Làm việc với FORM

Trong phần này, chúng ta sẽ tìm hiểu cách làm việc với Form trong PHP. Form là một phần không thể thiếu trong các ứng dụng web, cho phép người dùng nhập và gửi dữ liệu đến máy chủ để xử lý. Một form HTML có thể chứa nhiều loại điều khiển như text box, radio button, check box, drop-down list và text area, Khi một form chứa các điều khiển này được gửi tới server, dữ liệu sẽ được chuyển thành mảng chứa các cặp tên/giá trị rồi mới được đẩy lên máy chủ.

5.7.1. Nhận dữ liệu từ Form

Khi người dùng gửi dữ liệu từ form, dữ liệu này sẽ được gửi đến máy chủ thông qua các phương thức HTTP như GET hoặc POST. PHP cung cấp các biến siêu toàn cục `$_GET` và `$_POST` để truy cập dữ liệu từ form.

Chúng ta tạo một form cơ bản sử dụng các thẻ `<form>` và `<input>` để tạo các trường nhập liệu. Các thuộc tính quan trọng của thẻ form là *action* và *method*.

```

<form action="process.php" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username">
    <label for="email">Email:</label>
    <input type="email" id="email" name="email">
    <input type="submit" value="Submit">
</form>

```

Trong PHP, chúng ta sử dụng các biến siêu toàn cục `$_GET` và `$_POST` để lấy dữ liệu từ form.

```

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

```

Ví dụ:

(1) Tạo một file HTML như sau:

```

<!DOCTYPE html>

```

```

<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Form Example</title>
</head>
<body>
  <form action="process.php" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username">
    <label for="email">Email:</label>
    <input type="email" id="email" name="email">
    <input type="submit" value="Submit">
  </form>
</body>
</html>

```

(2) Tạo một file “process.php” để xử lý dữ liệu:

```

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $username = htmlspecialchars($_POST['username']);
  $email = htmlspecialchars($_POST['email']);
  echo "Username: $username<br>";
  echo "Email: $email<br>";
}
?>

```

Trong ví dụ này, chúng ta sử dụng *htmlspecialchars()* để ngăn ngừa tấn công XSS bằng cách mã hóa các ký tự đặc biệt.

5.7.2. Hiện thị dữ liệu trên trang web

Sau khi nhận dữ liệu từ biểu mẫu, bước tiếp theo là hiển thị dữ liệu này trên trang web. PHP cung cấp nhiều cách để hiển thị dữ liệu trên trang web như in trực tiếp bằng “*echo*” hoặc sử dụng biến để lưu trữ và hiển thị, sử dụng HTML, sử dụng vòng lặp để hiển thị danh sách dữ liệu.

Ví dụ 1. In trực tiếp bằng “*echo*”

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Form Example</title>
</head>
<body>
  <form method="post" action="display.php">
    <label for="username">Username:</label>

```

```

        <input type="text" id="username" name="username">
        <input type="submit" value="Submit">
    </form>
</body>
</html>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = htmlspecialchars($_POST['username']);
    echo "Username: " . $username;
}
?>

```

Ví dụ 2. Sử dụng biến để lưu trữ và hiển thị

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form Example</title>
</head>
<body>
    <form method="post" action="">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username">
        <input type="submit" value="Submit">
    </form>

    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $username = htmlspecialchars($_POST['username']);
        echo "<h2>Submitted Data</h2>";
        echo "Username: " . $username;
    }
    ?>
</body>
</html>

```

Ví dụ 3. Hiển thị dữ liệu bằng HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form Example</title>
</head>
<body>
    <form method="post" action="">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username">

```

```

<label for="email">Email:</label>
<input type="email" id="email" name="email">
<input type="submit" value="Submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = htmlspecialchars($_POST['username']);
    $email = htmlspecialchars($_POST['email']);

    echo "<h2>Submitted Data</h2>";
    echo "<table border='1'>";
    echo "<tr><th>Field</th><th>Value</th></tr>";
    echo "<tr><td>Username</td><td>$username</td></tr>";
    echo "<tr><td>Email</td><td>$email</td></tr>";
    echo "</table>";
}
?>
</body>
</html>

```

Ví dụ 4. Sử dụng vòng lặp để hiển thị danh sách dữ liệu

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form Example</title>
</head>
<body>
    <form method="post" action="">
        <label for="username1">Username 1:</label>
        <input type="text" id="username1" name="usernames[]">
        <label for="username2">Username 2:</label>
        <input type="text" id="username2" name="usernames[]">
        <label for="username3">Username 3:</label>
        <input type="text" id="username3" name="usernames[]">
        <input type="submit" value="Submit">
    </form>

    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $usernames = $_POST['usernames'];

        echo "<h2>Submitted Usernames</h2>";
        echo "<ul>";
        foreach ($usernames as $username) {
            $username = htmlspecialchars($username);
            echo "<li>". $username . "</li>";
        }
    }
    ?>

```

```

    }
    echo "</ul>";
}
?>
</body>
</html>

```

5.8. Phương thức GET/ POST trong PHP

Trong PHP, phương thức GET và POST là hai cách phổ biến để gửi dữ liệu từ client lên server. Việc hiểu rõ cách hoạt động và sử dụng hai phương thức này là rất quan trọng để xây dựng các ứng dụng web hiệu quả và bảo mật.

5.8.1. Phương thức GET trong PHP

Phương thức GET là một trong hai phương thức chính được sử dụng để gửi dữ liệu từ client (trình duyệt) tới server. Dữ liệu được gửi qua phương thức GET sẽ được đính kèm vào URL, hiển thị trên thanh địa chỉ của trình duyệt.

* Cách thức hoạt động của phương thức GET

- Client gửi dữ liệu: Khi người dùng gửi dữ liệu bằng phương thức GET, dữ liệu sẽ được gắn vào URL dưới dạng các tham số truy vấn sau dấu chấm hỏi (?). Ví dụ, khi một form được gửi đi bằng phương thức GET, URL sẽ có dạng:

`http://localhost/handle_get.php?key1=value1&key2=value2.`

- Server nhận dữ liệu: Server sẽ nhận URL này và phân tích các tham số truy vấn được đính kèm. Tất cả dữ liệu được gửi lên bằng phương thức GET sẽ được lưu trữ trong biến toàn cục `$_GET` mà PHP tự tạo ra. Biến `$_GET` là một mảng kết hợp lưu trữ dữ liệu dưới dạng các cặp `key => value`.

Ví dụ:

```

URL: http://localhost/DemoWebPHP/handle_get.php?module=home&page=2
Handle_get.php
<?php
    echo '<pre>';
    print_r($_GET); // in ra mảng
    echo '</pre>';
    if(isset($_GET['page'])){
        echo $_GET['page']; // in ra value của page
    }else{
        echo 'Giá trị không tồn tại.'
    }
?>

```

Các URL chứa tham số GET có thể được bookmark để truy cập lại sau này, tuy nhiên là dữ liệu được gửi qua URL có kích thước giới hạn (chỉ 1024 ký tự), không bảo mật do dữ liệu này hiển thị trên URL.

5.8.2. Phương thức POST trong PHP

Phương thức POST trong PHP là một cách gửi dữ liệu từ client đến server qua HTTP, khác với phương thức GET. Phương thức POST gửi dữ liệu qua phần thân của yêu cầu HTTP (HTTP body), không hiển thị dữ liệu trên thanh địa chỉ của trình duyệt, điều này giúp bảo mật thông tin nhạy cảm và hỗ trợ gửi dữ liệu lớn hơn.

* Cách thức hoạt động của phương thức POST

- Client gửi dữ liệu: Dữ liệu được gửi từ client đến server thông qua một form HTML. Các giá trị gửi đi được định nghĩa trong các thẻ `<input>` (như textbox, radio, checkbox,...) và được nhận diện qua thuộc tính *name* của các thẻ input. Khi form được gửi, dữ liệu sẽ được gửi qua phần thân của yêu cầu HTTP.

- Server nhận dữ liệu: Dữ liệu gửi từ client qua phương thức POST được lưu trữ trong biến toàn cục `$_POST` do PHP cung cấp. Biến này là một mảng kết hợp, trong đó mỗi phần tử của mảng tương ứng với một trường dữ liệu từ form, với tên trường làm key và giá trị trường làm value.

Ví dụ:

(1) HTML form (index.php)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>POST Method Example</title>
</head>
<body>
  <form action="handle_post.php" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

(2) handle_post.php

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // Lấy dữ liệu từ $_POST
  $username = $_POST['username'];
  $password = $_POST['password'];

  // Hiển thị dữ liệu
  echo '<h1>Received Data:</h1>';
  echo 'Username: ' . htmlspecialchars($username) . '<br>';
  echo 'Password: ' . htmlspecialchars($password) . '<br>';
} else {
```

```
        echo 'No data received.';
    }
?>
```

5.9. Làm việc với Cookie và Session

Cookie và Session là hai cơ chế lưu trữ dữ liệu trên trình duyệt và server để theo dõi và duy trì trạng thái giữa các lần truy cập trang web.

5.9.1. Cookie

Cookie cung cấp cho ứng dụng web một phương thức lưu trữ thông tin trên trình duyệt của người dùng và truy xuất khi trình duyệt gửi yêu cầu xem trang.

Cookie là cặp tên/giá trị được lưu trên trình duyệt. Trên server, ứng dụng web tạo cookie và gửi nó tới trình duyệt. Trên máy client, trình duyệt lưu cookie và gửi nó trở lại server mỗi khi truy cập trang từ server đó. Mặc định, cookie chỉ có hiệu lực cho đến khi người dùng đóng trình duyệt. Tuy nhiên, chúng ta có thể thiết lập để cookie tồn tại trong trình duyệt của người dùng với thời gian lên đến ba năm.

Để tạo và *thiết lập cookie* trong trình duyệt, chúng ta sử dụng hàm *setcookie*. Để lấy giá trị của cookie, dùng biến toàn cục tự động *\$_COOKIE*. Biến này là một mảng liên kết.

Cú pháp:

```
setcookie($name, $value, $expire, $path, $domain, $secure, $httponly);
```

Khi sử dụng hàm *setcookie*, tham số *\$name* là tham số bắt buộc duy nhất. Tuy nhiên, trong hầu hết các trường hợp, chúng ta cần cung cấp giá trị cho bốn tham số đầu tiên. Theo cách này, chúng ta có thể sử dụng tham số *\$name* và *\$value* để xác định cặp tên/giá trị cho cookie, *\$expire* để xác định ngày hết hạn và *\$path* để cookie có hiệu lực với tất cả các trang của ứng dụng web hiện tại. Nếu thiết lập tham số *\$expire* là 0, cookie sẽ chỉ tồn tại cho đến khi người dùng đóng trình duyệt. Đây được gọi là cookie theo phiên (per-session cookie). Tham số *\$path* gần như luôn phải được thiết lập là gốc của trang web (root). Theo cách này, mọi trang trong ứng dụng đều có thể truy cập cookie. *\$domain* là tên miền mà cookie có hiệu lực, mặc định là tên của server thiết lập cookie. *\$secure* là TRUE thì cookie sẽ chỉ có hiệu lực nếu được gửi qua HTTPS, mặc định là FALSE. *\$httponly* là TRUE thì cookie chỉ có hiệu lực qua giao thức HTTP chứ không qua các ngôn ngữ kịch bản phía client như Javascript, mặc định là FALSE.

Ví dụ:

```
<?php
// Tạo cookie
$name = 'username';
$value = 'kimhuong';
$expire = strtotime('+1 year');
$path = 1 /;
setcookie($name, $value, $expire, $path);

// Lấy giá trị từ cookie
if (isset($_COOKIE['username'])) {
    echo "Username: " . $_COOKIE['username'];
}
```

```

    } else {
        echo "Cookie 'username' chưa được thiết lập!";
    }

    // Xóa cookie khỏi trình duyệt web thì value rỗng và ngày hết hạn là một ngày trong quá khứ
    $expire = strtotime('-1 year1');
    setcookie('username', '', $expire, '/');
?>

```

Để bật/ tắt cookie thì chúng ta vào cài đặt của trình duyệt, chọn Quyền riêng tư và bảo mật và thiết lập.

5.9.2. Session

Theo dõi người dùng khi họ di chuyển trong trang web được gọi là theo dõi phiên (session tracking). HTTP là giao thức phi trạng thái, khi gửi yêu cầu, trình duyệt ngắt kết nối tới server. Để duy trì trạng thái, ứng dụng web thực hiện theo dõi phiên. Mặc định, PHP sử dụng cookie để lưu session ID trên mỗi trình duyệt. Khi đó, trình duyệt chuyển cookie tới server theo từng yêu cầu.

Mở đầu, trình duyệt trên máy client gửi yêu cầu xem trang PHP tới server web. Sau đó, PHP kiểm tra liệu yêu cầu đã bao gồm session ID (định danh phiên) chưa. Nếu chưa, PHP sẽ tạo một phiên (session) mới trên server và gán cho nó một session ID duy nhất. Lúc này, ứng dụng có thể lưu dữ liệu vào phiên. Sau đó, session ID sẽ được gửi trả lại trình duyệt như cookie trong phản hồi. Khi trình duyệt gửi các yêu cầu sau đó, cookie session ID được gộp trong yêu cầu. PHP cũng kiểm tra liệu yêu cầu đã bao gồm session ID PHP chưa. Nếu có, PHP sử dụng session ID để truy cập, điều chỉnh hoặc thêm dữ liệu khi cần.

Mặc dù cơ chế theo dõi phiên làm việc được tích hợp sẵn trong PHP, nhưng nó không tự động có hiệu lực. Để phiên có hiệu lực, chúng ta bắt đầu phiên làm việc mới hoặc chạy lại phiên làm việc cũ bằng cách gọi hàm *session start* ở đầu mỗi trang của ứng dụng cần truy cập dữ liệu về phiên.

Cú pháp khởi tạo Session với tham số Cookie mặc định:

```
session_start();
```

Hàm *session start* nhắc PHP kiểm tra sự tồn tại của session ID trong yêu cầu rồi tạo phiên và cookie theo phiên mới nếu session ID chưa được thiết lập. Vì hàm session start có thể tạo cookie nên nó phải được gọi trước khi có bất kỳ nội dung HTML nào được gửi tới trình duyệt. Mặc định, PHP sử dụng cookie theo phiên để lưu session ID trên trình duyệt của người dùng. Do đó, khi người dùng đóng trình duyệt, phiên sẽ kết thúc.

Khi khởi tạo phiên, chúng ta sử dụng biến toàn cục tự động `$_SESSION` để thiết lập và lấy dữ liệu của người dùng cho phiên. Biến này là một mảng liên kết lưu dữ liệu cho phiên. Nếu cần có thể sử dụng hàm *isset* để kiểm tra một phần tử có tồn tại trong mảng `$_SESSION` không. Để xóa tất cả các phần tử khỏi mảng `$_SESSION` bằng cách thiết lập mảng này thành một mảng rỗng.

Khi ứng dụng chạy, PHP lưu mảng `$_SESSION` trong bộ nhớ. Khi phiên làm việc kết thúc, PHP lưu nội dung của mảng `$_SESSION` trong một file nằm trên server web.

Cuối cùng, PHP xóa dữ liệu phiên khi phiên hết hạn. Mặc định, phiên hết hạn sau 24 phút không hoạt động.

Ví dụ: Thiết lập Session, biến và lấy biến từ Session

```
<?php
// Thiết lập Session
session_start();
// Thiết lập biến trong Session
$_SESSION['username'] = 'kimhuong';
$_SESSION['email'] = 'tkhuong@dthu.edu.vn';
$_SESSION['password'] = password_hash('admin', PASSWORD_BCRYPT);
// Lấy biến từ Session
$username = $_SESSION['username'];
$email = $_SESSION['email'];
$password = $_SESSION['password'];

echo 'Username: ' . $username . '<br>';
echo 'Email: ' . $email . '<br>';
echo 'Password Hash: ' . $password . '<br>';
?>
```

Ví dụ: Thiết lập và lấy mảng từ Session

```
<?php
session_start();
if (!isset($_SESSION['user_data'])) {
    $_SESSION['user_data'] = array();
}

// Thêm phần tử vào mảng
$_SESSION['user_data']['username'] = 'kimhuong';
$_SESSION['user_data']['email'] = 'tkhuong@dthu.edu.vn';
$_SESSION['user_data']['password'] = password_hash('admin', PASSWORD_BCRYPT);

// Lấy và sử dụng mảng
$user_data = $_SESSION['user_data'];
foreach ($user_data as $key => $value) {
    echo ucfirst($key) . ': ' . $value . '<br>';
}
?>
// ucfirst: chuyển ký tự đầu tiên của $key thành chữ hoa
```

Mặc định, Session sẽ kết thúc sau 24 phút nếu không có yêu cầu. Tuy nhiên, trong một số trường hợp, có thể chúng ta cần phải xóa Session, ví dụ, nếu người dùng thoát khỏi ứng dụng.

Cú pháp kết thúc phiên

```
$_SESSION = array(); // xóa session
```

```
session_destroy(); // hủy bỏ session
```

Ngoài ra, chúng ta cần xóa Cookie theo session khỏi trình duyệt (tương tự như ở mục Cookie).

***So sánh Cookie và Session**

Cookie	Session
Lưu trữ trên máy người dùng.	Lưu trữ trên server.
Có thể bị người dùng chỉnh sửa hoặc xóa.	An toàn hơn vì người dùng không thể chỉnh sửa.
Dữ liệu ít bảo mật hơn.	Sử dụng cho các dữ liệu cần bảo mật cao.

Câu hỏi, bài tập

Câu hỏi

Câu 1. Hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) là gì? Đặc điểm của RDBMS?

Gợi ý: Hệ quản trị CSDL quan hệ lưu trữ dữ liệu dưới dạng bảng (table), hỗ trợ truy vấn SQL, có tính toàn vẹn dữ liệu và mối quan hệ giữa các bảng

Câu 2. Trình bày cú pháp và chức năng của câu lệnh SELECT trong SQL

*Gợi ý: SELECT * FROM ten_bang; dùng để truy xuất dữ liệu. Có thể kết hợp với WHERE, ORDER BY, LIMIT*

Câu 3. So sánh sự khác nhau giữa INSERT, UPDATE và DELETE

Gợi ý: INSERT để thêm dòng mới; UPDATE để cập nhật dòng đã có; DELETE để xóa dòng.

Câu 4. MySQL là gì? Vì sao MySQL thường được sử dụng trong lập trình web với PHP?

Gợi ý: MySQL là hệ quản trị CSDL mã nguồn mở, dễ dùng, tích hợp tốt với PHP (qua PDO), hiệu năng cao

Câu 5. Hãy nêu các bước cơ bản để kết nối PHP với cơ sở dữ liệu MySQL?

Gợi ý: Sử dụng PDO → Chọn cơ sở dữ liệu → Thực hiện truy vấn → Xử lý kết quả → Đóng kết nối

Câu 6. Trình bày các cách kết nối PHP với MySQL

Gợi ý:

PHP hỗ trợ hai cách kết nối chính: MySQLi và PDO.

PDO hỗ trợ nhiều loại cơ sở dữ liệu khác nhau, còn MySQLi chỉ hỗ trợ MySQL.

Câu 7. Cách sử dụng try/catch trong PDO để xử lý lỗi kết nối là gì?

Gợi ý:

Sử dụng try/catch để bắt lỗi kết nối cơ sở dữ liệu hoặc các lỗi truy vấn. Ví dụ:

```
try {
```

```
$pdo = new PDO($dsn, $username, $password);  
} catch (PDOException $e) {  
    echo "Lỗi kết nối: " . $e->getMessage();  
}
```

Câu 8. Khi nào nên sử dụng phương thức prepare() trong PDO?

Gợi ý:

Khi cần bảo vệ dữ liệu khỏi SQL Injection.

Khi có tham số động trong câu lệnh SQL.

Câu 9:

Trình bày sự khác biệt giữa Cookie và Session trong PHP.

Gợi ý:

Cookie: Lưu trữ trên trình duyệt của người dùng, tồn tại qua nhiều phiên làm việc.

Session: Lưu trữ trên máy chủ, dữ liệu bị xóa khi kết thúc phiên làm việc hoặc sau một khoảng thời gian.

Câu 10:

Viết một câu lệnh SQL để:

Lấy danh sách tất cả sản phẩm có giá trên 1 triệu.

Cập nhật giá sản phẩm Laptop thành 22 triệu.

Gợi ý:

*SELECT * FROM products WHERE price > 1000000;*

UPDATE products SET price = 22000000 WHERE name = "Laptop";

Bài tập

Yêu cầu chung: Dùng phần mềm XAMPP và phpMyAdmin để thực hành. Tạo CSDL có tên "web_php"

Bài 1. Tạo cơ sở dữ liệu và bảng người dùng

Yêu cầu:

- Tạo CSDL *web_php*

- Tạo bảng **users** gồm các trường: id, fullname, email, age, created_at

Gợi ý:

```
CREATE DATABASE web_php;  
USE web_php;  
CREATE TABLE users (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    fullname VARCHAR(100),  
    email VARCHAR(100),
```

```
age INT,  
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

Bài 2. Thêm dữ liệu mẫu vào bảng *users*

Yêu cầu:

- Thêm ít nhất 3 người dùng mẫu vào bảng

Gợi ý:

```
INSERT INTO users (fullname, email, age)  
VALUES  
( 'Nguyen Van A', 'a@gmail.com', 20),  
( 'Tran Thi B', 'b@gmail.com', 22),  
( 'Le Van C', 'c@gmail.com', 19);
```

Bài 3. Viết truy vấn SELECT nâng cao

Yêu cầu:

- Hiện thị tất cả người dùng có tuổi > 20.
- Hiện thị danh sách người dùng theo thứ tự giảm dần của tuổi.

Gợi ý:

```
SELECT * FROM users WHERE age > 20;  
SELECT * FROM users ORDER BY age DESC;
```

Bài 4. Cập nhật và xóa dữ liệu

Yêu cầu:

- Cập nhật tuổi của “Nguyen Van A” thành 25.
- Xóa người dùng có “email = c@gmail.com”

Gợi ý:

```
UPDATE users SET age = 25 WHERE fullname = 'Nguyen Van A';  
DELETE FROM users WHERE email = 'c@gmail.com';
```

Bài 5. Xây Dựng Trang Đăng Nhập

Yêu cầu:

Xây dựng một form đăng nhập với các trường username và password.

Kiểm tra thông tin đăng nhập từ cơ sở dữ liệu.

Hiện thị thông báo “Đăng nhập thành công!” nếu thông tin đúng, ngược lại hiện thị “Sai thông tin đăng nhập!”.

Gợi ý:

```

*HTML
<form method="POST" action="login.php">
  <input type="text" name="username" placeholder="Username">
  <input type="password" name="password" placeholder="Password">
  <button type="submit">Đăng nhập</button>
</form>
*PHP (login.php)
session_start();
$dsn = "mysql:host=localhost;dbname=testdb;charset=utf8";
$username = "root";
$password = "";

try {
    $pdo = new PDO($dsn, $username, $password);
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username
AND password = :password");
        $stmt->execute(['username' => $_POST['username'], 'password' =>
$_POST['password']]);
        $user = $stmt->fetch();
        if ($user) {
            $_SESSION['username'] = $user['username'];
            echo "Đăng nhập thành công!";
        } else {
            echo "Sai thông tin đăng nhập!";
        }
    }
} catch (PDOException $e) {
    echo "Lỗi: ". $e->getMessage();
}

```

Bài 6. Xây dựng chức năng quản lý người dùng với PDO

Yêu cầu:

- Vận dụng kiến thức HTML form, xử lý PHP và kết nối MySQL bằng PDO.
- Tạo một hệ thống đơn giản để thêm và hiển thị người dùng.

Gợi ý:

- * Sử dụng cơ sở dữ liệu và bảng users (đã tạo ở Bài 1)
- * Tạo file cấu hình kết nối PDO (config.php)

```

<?php
$host = 'localhost';
$db = 'web_php';
$user = 'root';
$pass = '';

```

```

try {
    $pdo = new PDO("mysql:host=$host;dbname=$db;charset=utf8", $user, $pass);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Kết nối thất bại: " . $e->getMessage());
}
?>

```

** Tạo biểu mẫu thêm người dùng (add_user.php)*

```

<!DOCTYPE html>
<html>
<head>
    <title>Thêm người dùng</title>
</head>
<body>
    <h2>Thêm người dùng mới</h2>
    <form method="post" action="save_user.php">
        Họ tên: <input type="text" name="fullname" required><br><br>
        Email: <input type="email" name="email" required><br><br>
        Tuổi: <input type="number" name="age" required><br><br>
        <button type="submit">Thêm</button>
    </form>
    <br>
    <a href="list_users.php">Xem danh sách người dùng</a>
</body>
</html>

```

** Lưu dữ liệu vào cơ sở dữ liệu (save_user.php)*

```

<?php
require 'config.php';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $fullname = $_POST['fullname'];
    $email = $_POST['email'];
    $age = $_POST['age'];
}

```

```

$stmt = $pdo->prepare("INSERT INTO users (fullname, email, age) VALUES (?, ?, ?)");
$stmt->execute([$fullname, $email, $age]);

echo "Thêm thành công! <a href='list_users.php'>Xem danh sách</a>";
}
?>

```

* *Hiển thị danh sách người dùng (list_users.php)*

```

<?php
require 'config.php';

$stmt = $pdo->query("SELECT * FROM users ORDER BY id DESC");
$users = $stmt->fetchAll();
?>

<!DOCTYPE html>
<html>
<head>
    <title>Danh sách người dùng</title>
</head>
<body>
    <h2>Danh sách người dùng</h2>
    <table border="1" cellpadding="10">
        <tr>
            <th>STT</th>
            <th>Họ tên</th>
            <th>Email</th>
            <th>Tuổi</th>
            <th>Thời gian tạo</th>
        </tr>
        <?php foreach ($users as $index => $user): ?>
            <tr>
                <td><?= $index + 1 ?></td>
                <td><?= htmlspecialchars($user['fullname']) ?></td>
                <td><?= htmlspecialchars($user['email']) ?></td>
                <td><?= $user['age'] ?></td>

```

```

        <td><?= $user['created_at'] ?></td>
    </tr>
    <?php endforeach; ?>
</table>
<br>
<a href="add_user.php">Thêm người dùng mới</a>
</body>
</html>

```

*** Sinh viên tự thực hành các tính năng sau:**

- Thêm chức năng xóa người dùng.
- Thêm chức năng cập nhật thông tin người dùng.
- Sử dụng SESSION để hiển thị thông báo sau khi thêm/sửa/xóa.

Bài 7. Quản Lý Danh Sách Sản Phẩm

Yêu cầu:

Hiển thị danh sách sản phẩm từ cơ sở dữ liệu với các cột: ID, Tên sản phẩm, Giá.

Thêm sản phẩm mới vào cơ sở dữ liệu thông qua một form.

Xóa sản phẩm theo ID.

Gợi ý:

```

* PHP (hiển thị sản phẩm):
$stmt = $pdo->query("SELECT * FROM products");
echo "<table border='1'>";
echo "<tr><th>ID</th><th>Tên sản phẩm</th><th>Giá</th><th>Hành động</th></tr>";
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo "<tr>
        <td>{$row['id']}</td>
        <td>{$row['name']}</td>
        <td>{$row['price']}</td>
        <td><a href='delete.php?id={$row['id']}'>Xóa</a></td>
    </tr>";
}
echo "</table>";
* PHP (xóa sản phẩm):
if (isset($_GET['id'])) {
    $stmt = $pdo->prepare("DELETE FROM products WHERE id = :id");
    $stmt->execute(['id' => $_GET['id']]);
    echo "Xóa sản phẩm thành công!";
}

```

Bài 8. Lưu Trạng Thái Đăng Nhập bằng Session

Yêu cầu:

Khi người dùng đăng nhập thành công, lưu trạng thái vào Session.

Hiển thị thông báo "Xin chào, [username]" trên trang chủ nếu người dùng đã đăng nhập.

Gợi ý:

```
session_start();  
if (isset($_SESSION['username'])) {  
    echo "Xin chào, " . $_SESSION['username'];  
} else {  
    echo "Bạn chưa đăng nhập!";  
}
```

Bài tập tự thực hành

Bài Tập 1: Xây Dựng Trang Quản Lý Đăng Ký Thành Viên

Yêu cầu:

- Xây dựng một trang web cho phép người dùng đăng ký tài khoản.
- Lưu thông tin tài khoản bao gồm username, email, và password vào cơ sở dữ liệu MySQL.
- Kiểm tra các điều kiện đầu vào:
 - username: không được để trống và có độ dài ít nhất 5 ký tự.
 - email: phải đúng định dạng email.
 - password: không được để trống và có độ dài ít nhất 8 ký tự.
- Sau khi đăng ký thành công, hiển thị thông báo “Đăng ký thành công!”.

Bài Tập 2: Quản Lý Đơn Hàng

Yêu cầu:

- Xây dựng hệ thống quản lý đơn hàng bao gồm:
- Giao diện nhập thông tin đơn hàng với các trường customer_name, product_name, quantity, và order_date.
 - Lưu thông tin đơn hàng vào cơ sở dữ liệu MySQL.
 - Hiển thị danh sách đơn hàng đã lưu trong cơ sở dữ liệu.
 - Cho phép tìm kiếm đơn hàng theo customer_name.

TÀI LIỆU THAM KHẢO

Sách:

- [1] Jennifer Coleman Dowling (2022) Trường Đại học FPT (dịch), *Khám phá đa phương tiện: Multimedia demystified*, NXB Bách khoa Hà Nội
- [2] Jeremy Osborn, Nhóm AGI Creative (2019); Lê Hoàng Giang, Trần Tấn Minh Đạo (dịch), *HTML5 và CSS3: Thiết kế trang web thích ứng giàu tính năng: HTML5 digital classroom*, NXB Bách khoa Hà Nội
- [3] Phạm Thị Nhung (2008), *Giáo trình lập trình Web với HTML và Javascript*, NXB Đại học Quốc Gia TP HCM
- [4] Nguyễn Đình Thuân, Mai Xuân Hùng (2015), *Giáo trình Phát triển ứng dụng Web*, NXB Đại học Quốc Gia TP HCM
- [5] Trần Kim Hương (ch.b); Nguyễn Thị Thùy Linh, Lương Thái Ngọc (2021), *Bài giảng công nghệ Web*, Trường Đại học Đồng Tháp

Website:

<https://www.w3schools.com/>

<https://www.php.net/>