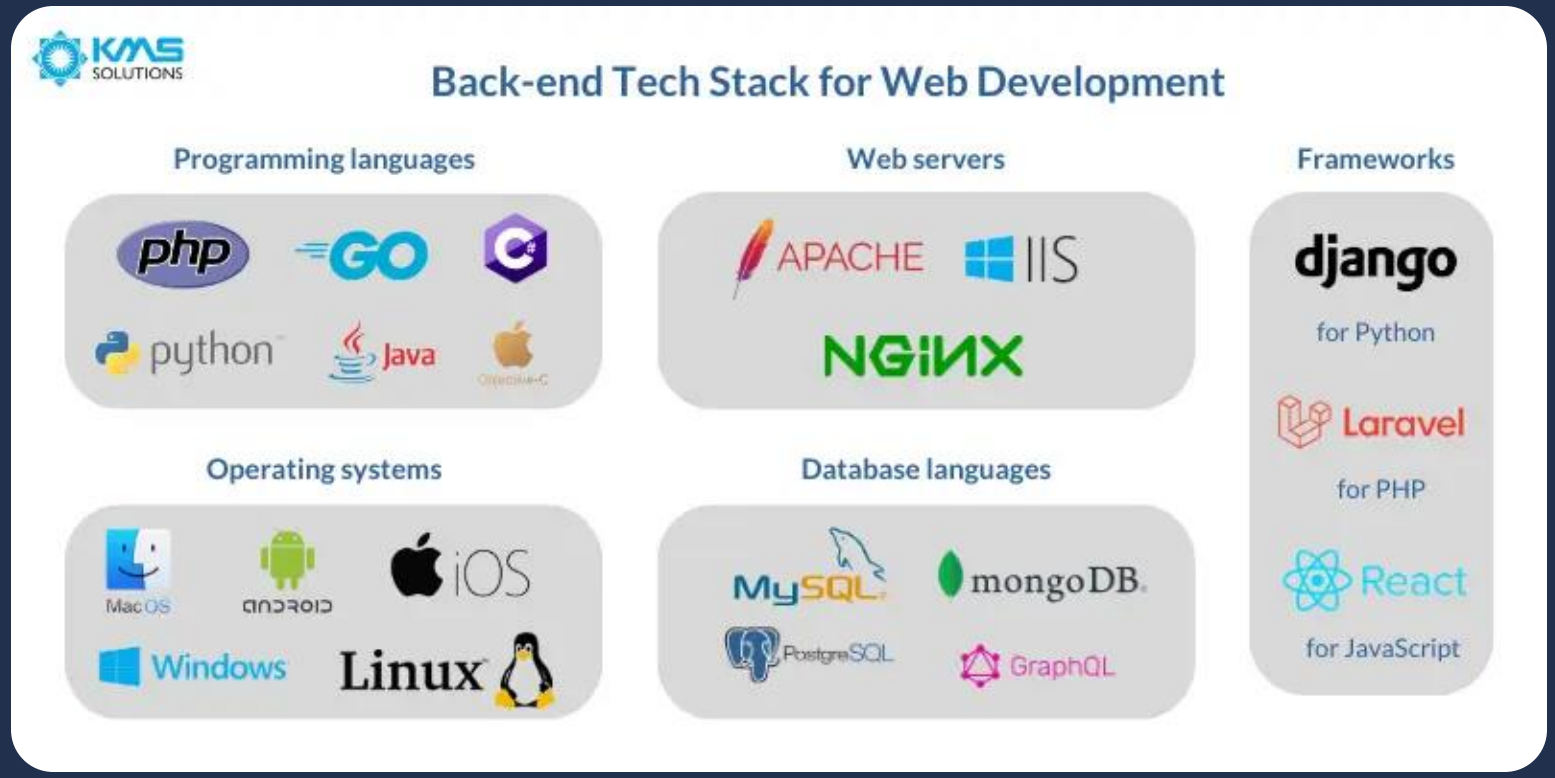
CSE485 – Công nghệ Web

dungkt@tlu.edu.vn



Bài 3. Các kỹ thuật căn bản cho trang Web động

NỘI DUNG

1. Giới thiệu
2. Các hàm dựng sẵn
3. Nhận dữ liệu từ trình duyệt
4. Tải lên ảnh và tệp tin
5. Ngày và thời gian
6. Cookie và Session
7. Xử lý lỗi

1. GIỚI THIỆU

Khi sử dụng các biến superglobal, đặc biệt là $_GET và $_POST, hãy cẩn thận với các vấn đề bảo mật như SQL Injection và Cross-Site Scripting (XSS). Luôn xác thực và làm sạch dữ liệu đầu vào.

- **Biến siêu toàn cục:**
  - Các biến superglobal là các biến tự động toàn cầu, có thể truy cập từ bất kỳ phần nào của script PHP, không cần phải khai báo là global. PHP cung cấp một số biến superglobal chứa thông tin quan trọng và có thể được sử dụng trong bất kỳ phạm vi nào của script.
  - $_GLOBALS: chứa tất cả các biến toàn cục của PHP
  - $_SERVER: Chứa thông tin về máy chủ và môi trường thực thi. Ví dụ: đường dẫn script, header HTTP, địa chỉ IP của máy chủ, v.v.
  - $_GET: Chứa dữ liệu được gửi thông qua phương thức HTTP GET, thường được sử dụng để lấy các tham số từ URL.
  - $_POST: Chứa dữ liệu được gửi thông qua phương thức HTTP POST, thường được sử dụng để thu thập dữ liệu từ một form.
  - $_FILES: Chứa thông tin về các tệp được tải lên thông qua phương thức HTTP POST.
  - $_COOKIE: Chứa tất cả các cookie được gửi bởi trình duyệt web.
  - $_SESSION: Cho phép lưu trữ thông tin phiên làm việc (session) cần thiết cho người dùng.
  - $_REQUEST: Chứa dữ liệu từ biến $_GET, $_POST, và $_COOKIE.
  - $_ENV: Chứa biến môi trường được gửi bởi máy chủ.

1. GIỚI THIỆU

- **Hàm var_dump:**
  - var_dump() là một hàm rất hữu ích được sử dụng để in thông tin về một biến. Hàm này in ra kiểu và giá trị của biến, cùng với thông tin chi tiết nếu biến đó là một mảng hoặc đối tượng. Đây là một công cụ quan trọng để gỡ lỗi, vì nó cho phép bạn xem chi tiết nội dung và cấu trúc của mọi loại biến.

```
$number = 123;
$string = "Hello, World!";
$bool = true;

var_dump($number);  // in ra: int(123)
var_dump($string);  // in ra: string(13) "Hello, World!"
var_dump($bool);    // in ra: bool(true)
```

```
$array = array(1, 2, 3);
class Person {
    public $name = "Alice";
}
$person = new Person();

var_dump($array);   // in ra thông tin chi tiết về mảng
var_dump($person);  // in ra thông tin chi tiết về đối tượng
Person
```

1. GIỚI THIỆU

- **Lỗi trong PHP:**
  - Có nhiều loại lỗi khác nhau có thể xảy ra trong quá trình viết và thực thi mã. Hiểu rõ các loại lỗi này giúp bạn gỡ lỗi và viết mã hiệu quả hơn. Dưới đây là một số loại lỗi chính trong PHP.

KIỂU LỖI	PHÂN TÍCH
1. Syntax Errors	**Lỗi cú pháp (syntax errors):** đây là những lỗi xảy ra do vi phạm ngữ pháp của PHP, như thiếu dấu phẩy hoặc dấu ngoặc. **Ví dụ:** echo "hello world (thiếu dấu ngoặc kép đóng). **Hậu quả:** lỗi cú pháp ngăn script thực thi.
2. Runtime Errors	**Lỗi thực thi (runtime errors):** các lỗi này xảy ra khi script PHP đang được chạy. **Ví dụ:** truy cập một biến chưa được khai báo. **Hậu quả:** lỗi thực thi thường dẫn đến việc script dừng hoạt động tại điểm lỗi xảy ra.
3. Fatal Errors	**Lỗi nghiêm trọng (fatal errors):** đây là những lỗi nghiêm trọng mà sau khi chúng xảy ra, script sẽ không thể tiếp tục thực thi. **Ví dụ:** gọi một hàm không tồn tại. **Hậu quả:** script dừng ngay lập tức.
4. Warning Errors	**Cảnh báo (warning errors):** là các lỗi không ngăn script thực thi nhưng báo hiệu rằng có điều gì đó không chính xác. **Ví dụ:** bao gồm (include) một file không tồn tại. **Hậu quả:** script vẫn tiếp tục chạy, nhưng chức năng nào đó có thể không hoạt động đúng.
5. Notice Errors	**Thông báo (notice errors):** những lỗi này thông báo về việc sử dụng không chính xác hoặc không an toàn các biến, hàm, v.V. **Ví dụ:** sử dụng biến không được khởi tạo. **Hậu quả:** script vẫn tiếp tục thực thi, nhưng có thể có kết quả không mong muốn hoặc không chính xác.
6. Parse Errors	**Lỗi parse:** tương tự như lỗi cú pháp, xảy ra khi PHP không thể phân tích cú pháp mã nguồn. **Ví dụ:** sử dụng từ khóa sai trong ngữ cảnh không phù hợp. **Hậu quả:** ngăn script thực thi.

1. GIỚI THIỆU

- **Cài đặt bộ thông dịch PHP:**
  - Cấu hình của trình thông dịch PHP (PHP interpreter) thường được quản lý thông qua tệp **php.ini**, là tệp cấu hình chính của PHP. Tệp này chứa các chỉ thị cấu hình có thể điều chỉnh để thay đổi hành vi của trình thông dịch PHP, từ việc quản lý lỗi, định cấu hình tải extension, đến quản lý tài nguyên như bộ nhớ và thời gian thực thi tối đa.
    - **Memory Limit (memory_limit)**: Đặt giới hạn bộ nhớ mà một script PHP có thể sử dụng. Ví dụ: memory_limit = 128M cho phép mỗi script sử dụng tối đa 128 MB bộ nhớ.
    - **Upload Max Size (upload_max_filesize)**: Giới hạn kích thước tối đa của file được tải lên thông qua PHP. Ví dụ: upload_max_filesize = 20M.
    - **Post Max Size (post_max_size)**: Đặt kích thước tối đa của dữ liệu POST. Nên đặt giá trị này cao hơn upload_max_filesize nếu bạn muốn tải lên các file lớn. Ví dụ: post_max_size = 30M.
    - **Max Execution Time (max_execution_time)**: Đặt thời gian thực thi tối đa cho mỗi script PHP. Ví dụ: max_execution_time = 30 giới hạn thời gian thực thi là 30 giây.
    - **Error Reporting (error_reporting)**: Định cấu hình cách PHP báo cáo lỗi. Ví dụ: error_reporting = E_ALL báo cáo tất cả các lỗi và cảnh báo.
    - **Display Errors (display_errors)**: Quyết định xem lỗi có được hiển thị ra màn hình hay không. Thường được tắt (Off) trong môi trường sản xuất.

1. GIỚI THIỆU

- **Cài đặt bộ thông dịch PHP:**
  - Cấu hình của trình thông dịch PHP (PHP interpreter) thường được quản lý thông qua tệp **php.ini**, là tệp cấu hình chính của PHP. Tệp này chứa các chỉ thị cấu hình có thể điều chỉnh để thay đổi hành vi của trình thông dịch PHP, từ việc quản lý lỗi, định cấu hình tải extension, đến quản lý tài nguyên như bộ nhớ và thời gian thực thi tối đa.
    - **Log Errors (log_errors)**: Kích hoạt ghi lỗi vào file log thay vì hiển thị trên màn hình.
    - **Time Zone (date.timezone)**: Đặt múi giờ mặc định cho các chức năng liên quan đến ngày giờ. Ví dụ: date.timezone = "America/New_York".
    - **Session Configuration**: Các chỉ thị liên quan đến quản lý session, như session.save_handler, session.save_path, v.v.
  - Sửa đổi và tải lại cấu hình
    - Khi thay đổi php.ini, cần khởi động lại máy chủ web (ví dụ: Apache, Nginx) để các thay đổi có hiệu lực.
    - Vị trí của php.ini phụ thuộc vào cách cài đặt và cấu hình PHP. Trên môi trường máy chủ chia sẻ, bạn có thể không có quyền truy cập để sửa đổi php.ini trực tiếp.

2. CÁC HÀM DỰNG SẴN

- **Built-in functions:**
  - PHP cung cấp một loạt các hàm dựng sẵn (built-in functions) rất mạnh mẽ và linh hoạt, giúp thực hiện nhiều tác vụ khác nhau mà không cần phải viết mã từ đầu. Dưới đây là một số loại hàm tích hợp chính trong PHP:
    - **Hàm xử lý chuỗi:**
      - strlen($string): Trả về độ dài của chuỗi.
      - str_replace($search, $replace, $subject): Thay thế tất cả các lần xuất hiện của một chuỗi tìm kiếm với một chuỗi thay thế.
      - strpos($haystack, $needle): Tìm vị trí đầu tiên của một chuỗi con trong một chuỗi.
      - strtolower($string), strtoupper($string): Chuyển đổi chuỗi sang chữ thường hoặc chữ hoa.
      - trim($string): Loại bỏ các khoảng trắng ở đầu và cuối chuỗi.
      - ..

2. CÁC HÀM DỰNG SẴN

- **Built-in functions:**
  - PHP cung cấp một loạt các hàm dựng sẵn (built-in functions) rất mạnh mẽ và linh hoạt, giúp thực hiện nhiều tác vụ khác nhau mà không cần phải viết mã từ đầu. Dưới đây là một số loại hàm tích hợp chính trong PHP:
    - **Hàm xử lý mảng:**
      - count($array): Đếm số phần tử trong mảng.
      - array_merge($array1, $array2): Kết hợp một hoặc nhiều mảng.
      - array_push($array, $value1, $value2, ...): Thêm một hoặc nhiều phần tử vào cuối mảng.
      - sort($array): Sắp xếp mảng..
      - ..
    - **Hàm xử lý tệp và thư mục**
      - fopen($filename, $mode): Mở một file hoặc URL.
      - fclose($handle): Đóng một tài nguyên file mở.
      - file_get_contents($filename): Đọc toàn bộ file vào một chuỗi.
      - file_put_contents($filename, $data): Viết một chuỗi vào file, tạo file nếu nó không tồn tại.

2. CÁC HÀM DỰNG SẴN

- **Built-in functions:**
  - PHP cung cấp một loạt các hàm dựng sẵn (built-in functions) rất mạnh mẽ và linh hoạt, giúp thực hiện nhiều tác vụ khác nhau mà không cần phải viết mã từ đầu. Dưới đây là một số loại hàm tích hợp chính trong PHP:
    - **Hàm xử lý JSON:**
      - json_encode($value): Mã hóa dữ liệu thành định dạng JSON.
      - json_decode($json, $assoc): Giải mã một chuỗi JSON.
      - ..
    - **Hàm xử lý ngày giờ**
      - date($format, $timestamp): Định dạng một thời gian/dấu thời gian local.
      - strtotime($time): Chuyển đổi một chuỗi thời gian tiếng Anh thành một dấu thời gian Unix.
    - **Hàm xử lý số**
      - rand($min, $max): Tạo một số nguyên ngẫu nhiên.
      - number_format($number): Định dạng số theo kiểu tiền tệ..

2. CÁC HÀM DỰNG SẴN

- **Built-in functions:**
  - PHP cung cấp một loạt các hàm dựng sẵn (built-in functions) rất mạnh mẽ và linh hoạt, giúp thực hiện nhiều tác vụ khác nhau mà không cần phải viết mã từ đầu. Dưới đây là một số loại hàm tích hợp chính trong PHP:
    - **Hàm xử lý hình ảnh:**
      - imagecreatefromjpeg($filename): Tạo một hình ảnh mới từ file JPEG.
      - imagepng($image, $filename): Xuất hình ảnh sang file hoặc trình duyệt.
      - ..
    - **Hàm xử lý biểu thức chính qui**
      - preg_match($pattern, $subject): Thực hiện so khớp biểu thức chính quy.
      - preg_replace($pattern, $replacement, $subject): Thay thế văn bản dựa trên biểu thức chính quy.
    - **Lưu ý**:
      - PHP cung cấp một bộ thư viện hàm rất lớn, vì vậy đây chỉ là một số ví dụ điển hình.
      - Để sử dụng một số hàm nhất định (ví dụ, các hàm liên quan đến hình ảnh), bạn cần cài đặt và kích hoạt các extension tương ứng trong php.ini.

2. CÁC HÀM DỰNG SẴN

- **Hằng số trong PHP:**
  - Hằng số (constants) là các giá trị không thể thay đổi sau khi chúng được định nghĩa. Hằng số thường được sử dụng để lưu trữ các giá trị không đổi trong suốt quá trình chạy của chương trình, như thông tin cấu hình, đường dẫn file, hoặc giá trị cố định. Cách định nghĩa hằng số:

```
define("SITE_URL", "https://www.example.com");
echo SITE_URL; // In ra: https://www.example.com
```

```
class Math {
    const PI = 3.14159;
}

echo Math::PI; // In ra: 3.14159
```

2. CÁC HÀM DỰNG SẴN

- **Chuyển hướng trang:**
  - Sử dụng hàm **header()** để thực hiện chuyển hướng (redirect) đến một URL khác. Chuyển hướng này được thực hiện bằng cách gửi một HTTP header Location đến trình duyệt.

```
header('Location: https://www.example.com');
exit(); // Luôn gọi exit sau header() để đảm bảo không có mã nào khác được thực thi
```

Trong ví dụ này, khi mã PHP này được thực thi, người dùng sẽ được tự động chuyển hướng đến https://www.example.com.

  - Lưu ý:
    - Kết Thúc Script: Luôn gọi exit() hoặc die() sau header() để đảm bảo rằng không có mã PHP nào khác được thực thi sau khi chuyển hướng.
    - Gửi Trước Khi Xuất Ra: header() phải được gọi trước khi script gửi bất kỳ đầu ra nào đến trình duyệt. Điều này có nghĩa là bạn không thể có bất kỳ HTML hay echo nào trước lệnh header().
    - Kiểm Tra Đã Gửi Headers hay Chưa: Bạn có thể sử dụng hàm headers_sent() để kiểm tra xem header đã được gửi đi chưa trước khi gọi header().
    - Cảnh Báo và Lỗi: Nếu cố gắng gửi header sau khi đã có đầu ra, PHP sẽ tạo ra một cảnh báo và lệnh chuyển hướng sẽ không hoạt động.

2. CÁC HÀM DỰNG SẴN

- **Các hàm xử lý tệp:**
  - PHP cung cấp một loạt các hàm để làm việc với file, giúp bạn đọc, ghi, tạo, và xử lý các file một cách dễ dàng.

```
#Đọc tệp
$content = file_get_contents("example.txt");
hoặc
$handle = fopen("example.txt", "r");
$content = fread($handle, filesize("example.txt"));
fclose($handle);
```

```
#Ghi tệp
file_put_contents("example.txt", "Hello, World!");
Hoặc
$handle = fopen("example.txt", "w");
fwrite($handle, "Hello, World!");
fclose($handle);
```

```
#Kiểm tra thông tin tệp
if (file_exists("example.txt")) {
    // File tồn tại
}

$size = filesize("example.txt");
```

2. CÁC HÀM DỰNG SẴN

- **Các hàm xử lý tệp:**
  - PHP cung cấp một loạt các hàm để làm việc với file, giúp bạn đọc, ghi, tạo, và xử lý các file một cách dễ dàng.

```
#Xử lý thư mục
mkdir($pathname): Tạo một thư mục mới.

rmdir($dirname): Xóa một thư mục rỗng.

scandir($directory): Liệt kê tất cả file và thư mục bên trong một thư mục.

$files = scandir("/path/to/directory");
```

```
#Xóa tệp
unlink("example.txt");
```

  - Lưu ý:
    - Khi làm việc với file, bạn cần chú ý đến quyền truy cập file và các vấn đề bảo mật.
    - Luôn đóng file sau khi sử dụng (fclose($handle)).
    - Kiểm tra và xử lý lỗi trong quá trình đọc/ghi file để đảm bảo tính ổn định của ứng dụng.

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **GET/POST:**
  - Việc lấy dữ liệu từ trình duyệt thường được thực hiện qua hai phương thức chính: GET và POST. Dữ liệu này có thể đến từ form HTML, URL, hoặc thậm chí là từ cookies.
  - **Dữ liệu phương thức GET**: Dữ liệu được gửi thông qua phương thức GET thường xuất hiện trong URL. Trong PHP, bạn có thể truy cập dữ liệu này qua mảng superglobal $_GET.
    - Ví dụ: Nếu URL là example.com/index.php?name=Alice&age=30, bạn có thể truy cập các giá trị này như sau:

```
$name = $_GET['name']; // Alice
$age = $_GET['age']; // 30
```

  - **Dữ liệu phương thức POST**: Dữ liệu gửi qua phương thức POST (thường là từ một form HTML) được truy cập thông qua mảng superglobal $_POST.
    - Ví dụ: Giả sử bạn có FORM

```
<form action="submit.php" method="post">
  Name: <input type="text" name="name">
  Email: <input type="text" name="email">
  <input type="submit">
</form>
```

```
$name = $_POST['name'];
$email = $_POST['email'];
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **GET/POST:**
  - Việc lấy dữ liệu từ trình duyệt thường được thực hiện qua hai phương thức chính: GET và POST. Dữ liệu này có thể đến từ form HTML, URL, hoặc thậm chí là từ cookies.
  - **Dữ liệu từ Cookie**: Cookies được lưu trữ trên trình duyệt và bạn có thể truy cập chúng thông qua mảng superglobal $_COOKIE.
    - Ví dụ:

```
$someCookieValue = $_COOKIE['someCookieName'];
```

  - **Dữ liệu từ Session**: Dữ liệu session có thể lưu trữ thông tin giữa các request và được truy cập qua mảng superglobal $_SESSION.
    - Ví dụ:

```
session_start();
$someSessionValue = $_SESSION['someSessionKey'];
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Xử lý dữ liệu thiếu hoặc không hợp lệ :**
  - Việc xử lý dữ liệu thiếu hoặc không hợp lệ từ người dùng là quan trọng để đảm bảo tính ổn định và an toàn của ứng dụng. Dưới đây là một số cách để xử lý dữ liệu thiếu khi làm việc với mảng superglobal như $_GET, $_POST, và $_COOKIE:.
  - **Kiểm tra sự tồn tại của dữ liệu**: Trước khi sử dụng dữ liệu từ $_GET, $_POST, hoặc $_COOKIE, bạn nên kiểm tra xem dữ liệu đó có tồn tại không.

```
if (isset($_POST['username'])) {
    $username = $_POST['username'];
} else {
    // Xử lý trường hợp username không tồn tại
}
```

  - **Kiểm tra dữ liệu rỗng**: Đôi khi một trường dữ liệu có thể tồn tại nhưng không có giá trị. Sử dụng empty() để kiểm tra cả trường hợp này.

```
if (empty($_POST['username'])) {
    // Xử lý trường hợp username rỗng
}
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Xử lý dữ liệu thiếu hoặc không hợp lệ :**
  - Việc xử lý dữ liệu thiếu hoặc không hợp lệ từ người dùng là quan trọng để đảm bảo tính ổn định và an toàn của ứng dụng. Dưới đây là một số cách để xử lý dữ liệu thiếu khi làm việc với mảng superglobal như $_GET, $_POST, và $_COOKIE:.
  - **Đặt giá trị mặc định**: Trong trường hợp một biến không tồn tại, bạn có thể muốn gán một giá trị mặc định cho nó.

```
$username = isset($_POST['username']) ? $_POST['username'] : 'default_username’;

Hoặc sử dụng toán tử Null Coalescing trong PHP 7 trở lên:

$username = $_POST['username'] ?? 'default_username';
```

  - **Làm sạch và xác thực dữ liệu**: Luôn làm sạch và xác thực dữ liệu đầu vào để ngăn chặn các lỗi và tấn công bảo mật như SQL Injection và Cross-Site Scripting (XSS)..

```
$username = htmlspecialchars($_POST['username']);
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Xử lý dữ liệu thiếu hoặc không hợp lệ:**
  - Việc xử lý dữ liệu thiếu hoặc không hợp lệ từ người dùng là quan trọng để đảm bảo tính ổn định và an toàn của ứng dụng. Dưới đây là một số cách để xử lý dữ liệu thiếu khi làm việc với mảng superglobal như $_GET, $_POST, và $_COOKIE:.
  - **Gửi thông báo lỗi hoặc phản hồi**: Thông báo cho người dùng nếu dữ liệu cần thiết không có hoặc không hợp lệ.

```
if (empty($_POST['username'])) {
    echo "Tên người dùng là bắt buộc.";
}
```

  - **Ghi nhật kí lỗi**: Trong môi trường sản xuất, việc ghi nhật ký lỗi khi xử lý dữ liệu thiếu có thể giúp trong việc gỡ lỗi và theo dõi vấn đề.

```
if (!isset($_POST['username'])) {
    error_log("Username missing in POST request");
}
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Xác thực dữ liệu và Sử dụng dữ liệu an toàn:**
  - Xác thực dữ liệu để đảm bảo rằng nó đáp ứng các yêu cầu cụ thể, như kiểu dữ liệu, định dạng, hoặc giới hạn giá trị

```
if (!filter_var($id, FILTER_VALIDATE_INT)) {
    // Xử lý trường hợp ID không phải là số nguyên
}
```

  - Sử dụng dữ liệu một cách an toàn: Sử dụng dữ liệu đầu vào một cách an toàn, đặc biệt là khi thực hiện các truy vấn cơ sở dữ liệu để tránh SQL Injection.

```
// Sử dụng Prepared Statements cho truy vấn cơ sở dữ liệu
$stmt = $pdo->prepare("SELECT * FROM table WHERE id = :id");
$stmt->execute(['id' => $id]);
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Ví dụ về tấn công XSS:**
  - Cross-Site Scripting (XSS) là một loại tấn công mà trong đó kẻ tấn công cố gắng chèn mã script vào các trang web mà người dùng cuối xem. Mục tiêu chính của XSS là thực thi mã script một cách trái phép trên trình duyệt của nạn nhân, thường với mục đích ăn cắp cookie hoặc thông tin phiên đăng nhập, thay đổi hành vi của trang web, hoặc thực hiện các hành động trái phép dưới danh nghĩa người dùng
  - Ví dụ: Giả sử bạn có một trang web với một trường nhập liệu, ví dụ, một trường bình luận. Nếu trang này không làm sạch đầu vào từ người dùng, kẻ tấn công có thể chèn một đoạn mã JavaScript độc hại:

```
// Trang "comment.php"
$comment = $_GET['comment'];
echo "Bình luận của bạn: " . $comment;
```

  - Kẻ tấn công tạo một URL với script độc hại như sau và gửi cho người dùng:

```
http://example.com/comment.php?comment=<script>alert('XSS');</script>
```

  - Biện pháp phòng chống:

```
$comment = htmlspecialchars($_GET['comment'], ENT_QUOTES, 'UTF-8');
echo "Bình luận của bạn: " . $comment;
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Thoát kí tự với htmlspecialchars():**
  - htmlspecialchars() rất hữu ích để ngăn chặn các cuộc tấn công Cross-Site Scripting (XSS) bằng cách chuyển đổi các ký tự đặc biệt thành các thực thể HTML. Khi một trình duyệt nhận dữ liệu đã được xử lý bằng htmlspecialchars(), nó sẽ hiển thị các ký tự đặc biệt thay vì thực thi chúng như mã HTML hoặc JavaScript.
  - Cách sử dụng:

```
htmlspecialchars(string $string, int $flags = ENT_COMPAT, string $encoding = 'UTF-8', bool $double_encode = true): string
```

    - $string: Chuỗi cần được xử lý.
    - $flags: Một loạt các tùy chọn để chỉ định cách thức chuyển đổi. Mặc định là ENT_COMPAT, chuyển đổi dấu ngoặc kép nhưng không phải dấu ngoặc đơn.
    - $encoding: Định dạng mã hóa của chuỗi. Mặc định là 'UTF-8'.
    - $double_encode: Khi đặt là true, nó sẽ chuyển đổi các thực thể HTML hiện có thành thực thể.

```
$userInput = "<script>alert('XSS');</script>";

$safeInput = htmlspecialchars($userInput, ENT_QUOTES, 'UTF-8');
echo $safeInput; // In ra: &lt;script&gt;alert('XSS');&lt;/script&gt;
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Thoát kí tự với hàm tùy chỉnh:**
  - Nếu bạn muốn tạo một hàm tùy chỉnh trong PHP để thoát các ký tự, mục tiêu chính của bạn sẽ là xử lý chuỗi để ngăn chặn các vấn đề liên quan đến bảo mật như tấn công Cross-Site Scripting (XSS) hoặc để đảm bảo chuỗi đầu vào không làm vỡ cấu trúc của mã hoặc truy vấn SQL. Dưới đây là một ví dụ về cách bạn có thể xây dựng một hàm tùy chỉnh để thoát các ký tự:

```
function escapeString($string) {
    // Chuyển đổi các ký tự đặc biệt thành thực thể HTML
    $escapedString = htmlspecialchars($string, ENT_QUOTES, 'UTF-8');

    // Thêm bất kỳ logic bổ sung nào ở đây nếu cần
    // Ví dụ: xử lý newline, tab, v.v.

    return $escapedString;
}
```

Trong hàm escapeString(), chúng ta sử dụng htmlspecialchars() để chuyển đổi các ký tự như <, >, " và ' thành thực thể HTML tương ứng của chúng. Điều này giúp ngăn chặn việc mã độc hại được thực thi trên trình duyệt.

```
$userInput = "<script>alert('XSS');</script>";
$safeInput = escapeString($userInput);
echo $safeInput; // In ra: &lt;script&gt;alert('XSS');&lt;/script&gt;
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Thu thập dữ liệu từ FORM:**
  - Thu thập dữ liệu từ các form HTML là một quá trình cơ bản và quan trọng. Có hai phương thức chính để gửi dữ liệu form: GET và POST. Dữ liệu form được truy cập thông qua các mảng superglobal $_GET và $_POST tương ứng. Dưới đây là cách bạn có thể thu thập dữ liệu form trong PHP:

Khi sử dụng phương thức GET, dữ liệu form được gửi thông qua URL và có thể truy cập thông qua mảng $_GET.

```
#FORM sử dụng GET
<form action="submit.php" method="get">
    Name: <input type="text" name="name">
    Email: <input type="text" name="email">
    <input type="submit" value="Submit">
</form>
```

```
#Xử lý trong PHP (submit.php)
if (isset($_GET['name']) && isset($_GET['email'])) {
    $name = htmlspecialchars($_GET['name']);
    $email = htmlspecialchars($_GET['email']);

    // Xử lý dữ liệu...
}
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Thu thập dữ liệu từ FORM:**
  - Thu thập dữ liệu từ các form HTML là một quá trình cơ bản và quan trọng. Có hai phương thức chính để gửi dữ liệu form: GET và POST. Dữ liệu form được truy cập thông qua các mảng superglobal $_GET và $_POST tương ứng. Dưới đây là cách bạn có thể thu thập dữ liệu form trong PHP:

Phương thức POST gửi dữ liệu thông qua HTTP request, không hiển thị trên URL. Dữ liệu được truy cập qua mảng $_POST.

```
#FORM sử dụng POST
<form action="submit.php" method="post">
    Name: <input type="text" name="name">
    Email: <input type="text" name="email">
    <input type="submit" value="Submit">
</form>
```

```
#Xử lý trong PHP (submit.php)
if (isset($_POST['name']) && isset($_POST['email'])) {
    $name = htmlspecialchars($_POST['name']);
    $email = htmlspecialchars($_POST['email']);

    // Xử lý dữ liệu...
}
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Thu thập dữ liệu từ FORM:**
  - **Kiểm tra FORM đã gửi dữ liệu đi với phương thức POST**: Để kiểm tra xem một form đã được gửi đi thông qua phương thức HTTP POST hay chưa, bạn có thể sử dụng hàm $_SERVER['REQUEST_METHOD']. Hàm này trả về phương thức request hiện tại, và bạn có thể so sánh giá trị của nó với chuỗi 'POST' để xác định xem form có được gửi bằng POST hay không.

Cách làm tương tự với phương thức GET (thay vào vị trí POST tương ứng)

```
#FORM sử dụng POST
<form action="submit.php" method="post">
    Name: <input type="text" name="name">
    <input type="submit" value="Submit">
</form>
```

```
#Xử lý trong PHP (submit.php)
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Form đã được gửi
    $name = $_POST['name'];

    // Làm sạch và xử lý dữ liệu $name ở đây
}
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Đảm bảo tính hợp lệ của dữ liệu trên FORM:**
  - **Xác thực phạm vi số**: Xác thực xem một số có nằm trong một phạm vi nhất định hay không bằng cách sử dụng hàm filter_var() với bộ lọc **FILTER_VALIDATE_INT** và một mảng tùy chọn để chỉ định giới hạn của phạm vi. Điều này rất hữu ích khi bạn cần đảm bảo rằng giá trị đầu vào là một số nguyên và nằm trong một phạm vi nhất định..

```
$number = $_GET['number']; // Giả sử đây là dữ liệu đầu vào

$options = array(
    "options" => array(
        "min_range" => 1,  // Giới hạn dưới của phạm vi
        "max_range" => 100 // Giới hạn trên của phạm vi
    )
);

if (filter_var($number, FILTER_VALIDATE_INT, $options) !== false) {
    echo "Số $number hợp lệ và nằm trong phạm vi từ 1 đến 100.";
} else {
    echo "Số $number không hợp lệ hoặc nằm ngoài phạm vi từ 1 đến 100.";
}
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Đảm bảo tính hợp lệ của dữ liệu trên FORM:**
  - **Xác thực độ dài văn bản**: Xác thực độ dài của văn bản là quan trọng để đảm bảo rằng dữ liệu đầu vào từ người dùng phù hợp với các yêu cầu và hạn chế của ứng dụng. Bạn có thể kiểm tra độ dài của chuỗi văn bản sử dụng hàm strlen() và sau đó so sánh với giới hạn độ dài mong muốn.

```
$text = $_POST['text']; // Giả sử đây là dữ liệu đầu vào

$minLength = 10;
$maxLength = 100;

$length = strlen($text);

if ($length < $minLength) {
    echo "Văn bản quá ngắn, phải có ít nhất $minLength ký tự.";
} elseif ($length > $maxLength) {
    echo "Văn bản quá dài, không được vượt quá $maxLength ký tự.";
} else {
    echo "Văn bản hợp lệ.";
}
```

Mã hóa ký tự: độ dài được đo bằng số ký tự. Nếu đang làm việc với các ký tự đa byte (như UTF-8), hãy sử dụng mb_strlen() thay vì strlen() để đảm bảo độ dài được tính toán chính xác.

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Đảm bảo tính hợp lệ của dữ liệu trên FORM:**
  - **Xác thực mật khẩu**: Xác thực mật khẩu đúng cách là rất quan trọng để đảm bảo an toàn cho thông tin người dùng. Xác thực mật khẩu không chỉ bao gồm việc kiểm tra độ dài và ký tự của mật khẩu mà còn bao gồm việc mã hóa mật khẩu một cách an toàn khi lưu trữ. Dưới đây là cách bạn có thể xác thực mật khẩu trong PHP.

```php
#Kiểm tra độ dài và kí tự mật khẩu
$password = $_POST['password'];

if (strlen($password) < 8) {
    echo "Mật khẩu phải có ít nhất 8 ký tự.";
} elseif (!preg_match("/[a-z]/", $password)) {
    echo "Mật khẩu phải có ít nhất một chữ cái thường.";
} elseif (!preg_match("/[A-Z]/", $password)) {
    echo "Mật khẩu phải có ít nhất một chữ cái hoa.";
} elseif (!preg_match("/[0-9]/", $password)) {
    echo "Mật khẩu phải có ít nhất một chữ số.";
} elseif (!preg_match("/[!@#$%^&*()\-_=+{};:,<.>]/", $password)) {
    echo "Mật khẩu phải chứa ít nhất một ký tự đặc biệt.";
} else {
    echo "Mật khẩu hợp lệ.";
}
```

```php
#Mã hóa mật khẩu
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);
// Lưu $hashedPassword vào cơ sở dữ liệu thay vì mật khẩu gốc

#Xác thực mật khẩu khi đăng nhập
if (password_verify($password, $hashedPassword)) {
    // Mật khẩu đúng, tiến hành đăng nhập
} else {
    // Mật khẩu sai
}
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Xác thực Tùy chọn:**
  - Xác thực và lọc các tùy chọn hoặc giá trị đầu vào sử dụng hàm **filter_var()** với một loạt các bộ lọc và tùy chọn. Hàm này cho phép bạn xác thực và làm sạch dữ liệu đầu vào như chuỗi, số, URL, email, và nhiều hơn nữa.

```
#Xác thực Email
$email = "user@example.com";
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Email không hợp lệ";
} else {
    echo "Email hợp lệ";
}
```

```
#Xác thực Số nguyên
$number = "123";
if (!filter_var($number, FILTER_VALIDATE_INT)) {
    echo "Không phải số nguyên";
} else {
    echo "Số nguyên hợp lệ";
}
```

```
#Xác thực URL
$url = "http://www.example.com";
if (!filter_var($url, FILTER_VALIDATE_URL)) {
    echo "URL không hợp lệ";
} else {
    echo "URL hợp lệ";
}
```

```
$options = array(
    "options" => array(
        "min_range" => 1,
        "max_range" => 100
    )
);
$number = "50";
if (!filter_var($number, FILTER_VALIDATE_INT, $options)) {
    echo "Số nằm ngoài phạm vi cho phép";
} else {
    echo "Số trong phạm vi cho phép";
}
```

Xác thực vs. Làm sạch: filter_var() có thể được sử dụng cả cho xác thực (kiểm tra dữ liệu có hợp lệ không) và làm sạch dữ liệu (chuyển đổi dữ liệu vào một dạng an toàn hơn).

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Xác thực Checkbox:**
  - Việc xác thực checkbox từ một form HTML có thể hơi khác so với xác thực các trường dữ liệu khác như văn bản hay số. Điều này là do checkbox không gửi dữ liệu nếu nó không được chọn. Dưới đây là cách bạn có thể xác thực checkbox trong PHP:

```
<form action="submit.php" method="post">
    <input type="checkbox" name="agree" value="yes"> Tôi đồng ý với các điều khoản
    <input type="submit" value="Submit">
</form>
```

```
if (isset($_POST['agree']) && $_POST['agree'] == 'yes') {
    echo "Bạn đã đồng ý với các điều khoản.";
} else {
    echo "Bạn cần đồng ý với các điều khoản để tiếp tục.";
}
```

LƯU Ý:

- *Kiểm tra tồn tại: sử dụng isset() để kiểm tra xem checkbox có được gửi trong request POST hay không. Điều này cho bạn biết liệu người dùng có chọn nó hay không.*
- *So sánh giá trị: nếu bạn muốn kiểm tra một giá trị cụ thể của checkbox (trong trường hợp này là 'yes'), hãy so sánh giá trị của $_POST['agree'] với giá trị mong muốn.*
- *Giá trị mặc định: nếu không gửi checkbox (không được chọn), nó sẽ không xuất hiện trong mảng $_POST.*

3. NHẬN

- **Xác thu**
  - Xác
    đảm

```
<fo

</f
```

```php
// Xác thực tên
if (empty($_POST['name'])) {
    echo "Tên là bắt buộc.";
} else {
    $name = $_POST['name'];
    // Kiểm tra thêm điều kiện cho tên, nếu cần
}

// Xác thực email
if (empty($_POST['email'])) {
    echo "Email là bắt buộc.";
} elseif (!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
    echo "Email không hợp lệ.";
} else {
    $email = $_POST['email'];
}

// Xác thực tuổi
if (empty($_POST['age'])) {
    echo "Tuổi là bắt buộc.";
} elseif (!filter_var($_POST['age'], FILTER_VALIDATE_INT, array("options" => array("min_range"=>1, "max_range"=>100)))) {
    echo "Tuổi không hợp lệ.";
} else {
    $age = $_POST['age'];
}

// Xác thực giới tính
if (empty($_POST['gender'])) {
    echo "Vui lòng chọn giới tính.";
} else {
    $gender = $_POST['gender'];
}
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Thu thập dữ liệu với filter_input():**
  - Hàm **filter_input()** là một cách hiệu quả và an toàn để thu thập dữ liệu từ các nguồn như GET, POST, và COOKIE. Hàm này không chỉ thu thập dữ liệu mà còn có khả năng lọc và xác thực dữ liệu đầu vào, giúp ngăn chặn các vấn đề bảo mật như Cross-Site Scripting (XSS) và SQL Injection.

```
filter_input(int $type, string $variable_name, int $filter = FILTER_DEFAULT, $options = null)
```

  - $type: Loại input, có thể là INPUT_GET, INPUT_POST, INPUT_COOKIE, v.v.
  - $variable_name: Tên của biến (ví dụ, tên trường trong form).
  - $filter: Loại bộ lọc bạn muốn áp dụng. Mặc định là FILTER_DEFAULT, không áp dụng bất kỳ bộ lọc nào.
  - $options: Một mảng các tùy chọn hoặc bitflags. Các tùy chọn này phụ thuộc vào bộ lọc được sử dụng.

```
$email = filter_input(INPUT_POST, 'email', FILTER_VALIDATE_EMAIL);
if (!$email) {
    echo "Email không hợp lệ";
}
```

```
$name = filter_input(INPUT_GET, 'name', FILTER_SANITIZE_SPECIAL_CHARS);
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

```
$filters = array(
    "name" => FILTER_SANITIZE_STRING,
    "email" => FILTER_VALIDATE_EMAIL,
    "age" => array(
        "filter" => FILTER_VALIDATE_INT,
        "options" => array("min_range" => 1, "max_range" => 100)
    )
);

$result = filter_input_array(INPUT_POST, $filters);

if (!$result) {
    echo "Các biến đầu vào không hợp lệ.";
} else {
    if ($result['name'] === false) {
        echo "Tên không hợp lệ.";
    }
    if ($result['email'] === false) {
        echo "Email không hợp lệ.";
    }
    if ($result['age'] === false) {
        echo "Tuổi không hợp lệ.";
    }
}
```

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Bộ lọc làm sạch:**
  - Bộ lọc làm sạch (sanitization filters) là các công cụ mạnh mẽ được sử dụng để làm sạch dữ liệu đầu vào, giúp loại bỏ hoặc chuyển đổi ký tự không mong muốn. Điều này rất quan trọng để ngăn chặn các vấn đề bảo mật như SQL Injection và Cross-Site Scripting (XSS). PHP cung cấp một loạt các bộ lọc làm sạch thông qua hàm filter_var() và filter_input().
  - Một số bộ lọc làm sạch gồm:
    - Làm sạch Chuỗi:
      - FILTER_SANITIZE_STRING: Loại bỏ hoặc mã hóa các ký tự đặc biệt trong một chuỗi.
      - FILTER_SANITIZE_STRIPPED: Tương tự FILTER_SANITIZE_STRING nhưng còn loại bỏ các thẻ HTML và PHP.
    - Làm sạch Email:
      - FILTER_SANITIZE_EMAIL: Loại bỏ tất cả các ký tự không hợp lệ khỏi một chuỗi địa chỉ email.
    - Làm sạch URL:
      - FILTER_SANITIZE_URL: Loại bỏ tất cả các ký tự không hợp lệ khỏi một chuỗi URL.
    - Làm sạch Số:
      - FILTER_SANITIZE_NUMBER_INT: Loại bỏ tất cả các ký tự trừ các số và dấu cộng/trừ.
      - FILTER_SANITIZE_NUMBER_FLOAT: Làm sạch số thực, loại bỏ tất cả các ký tự trừ số, dấu cộng/trừ, dấu chấm và dấu phẩy.

3. NHẬN DỮ LIỆU TỪ TRÌNH DUYỆT

- **Bộ lọc làm sạch:**
  - ```
    $dirtyString = "<script>alert('XSS');</script>Hello, World!";
    $cleanString = filter_var($dirtyString, FILTER_SANITIZE_STRING);
    echo $cleanString; // In ra: Hello, World!
    ```
  - ```
    $dirtyEmail = "user@@example.com";
    $cleanEmail = filter_var($dirtyEmail, FILTER_SANITIZE_EMAIL);
    echo $cleanEmail; // In ra: user@example.com
    ```
  - ```
    $dirtyNumber = "1,234abc";
    $cleanNumber = filter_var($dirtyNumber, FILTER_SANITIZE_NUMBER_INT);
    echo $cleanNumber; // In ra: 1234
    ```
    - FILTER_SANITIZE_EMAIL: Loại bỏ tất cả các ký tự không hợp lệ khỏi một chuỗi địa chỉ email.
  - Làm sạch URL:
    - FILTER_SANITIZE_URL: Loại bỏ tất cả các ký tự không hợp lệ khỏi một chuỗi URL.
  - Làm sạch Số:
    - FILTER_SANITIZE_NUMBER_INT: Loại bỏ tất cả các ký tự trừ các số và dấu cộng/trừ.
    - FILTER_SANITIZE_NUMBER_FLOAT: Làm sạch số thực, loại bỏ tất cả các ký tự trừ số, dấu cộng/trừ, dấu chấm và dấu phẩy.
    ```

4. Tải lên ảnh và tệp tin

- **Qui trình tải lên ảnh và tệp tin:**
  - **Bước 1**: Tạo FORM

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

  - **Bước 2**: Xử lý tệp tải lên

```
#Kiểm tra kích thước tệp
if ($_FILES["fileToUpload"]["size"] > 500000) { // Ví dụ, giới hạn là 500 KB
    echo "Xin lỗi, tệp tin của bạn quá lớn.";
    exit;
}
```

```
#Kiểm tra kiểu tệp
$imageFileType = strtolower(pathinfo($_FILES["fileToUpload"]["name"], PATHINFO_EXTENSION));

// Chỉ cho phép một số định dạng tệp tin nhất định
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Xin lỗi, chỉ các tệp JPG, JPEG, PNG & GIF mới được cho phép.";
    exit;
}
```

4. Tải lên ảnh và tệp tin

- **Qui trình tải lên ảnh và tệp tin:**
  - **Bước 1**: Tạo FORM

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

  - **Bước 2**: Xử lý tệp tải lên

```
#Di chuyển tệp tải lên (từ thư mục tạm tới thư mục đích trên Server)
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);

if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
    echo "Tệp ". htmlspecialchars( basename( $_FILES["fileToUpload"]["name"])). " đã được tải lên.";
} else {
    echo "Xin lỗi, đã có lỗi xảy ra trong quá trình tải tệp tin của bạn.";
}
```

4. Tải lên ảnh và tệp tin

- **Cấu hình tải lên ảnh và tệp tin:**
  - Để cấu hình chức năng tải lên tệp tin trong PHP, bạn cần phải chỉnh sửa một số thiết lập trong tệp cấu hình **php.ini**. Dưới đây là các thiết lập chính mà bạn cần quan tâm:

```
file_uploads = On #Cho phép hoặc từ chối chức năng tải lên tệp tin.

upload_max_filesize = 2M #Xác định kích thước tối đa của tệp tin mà PHP cho phép tải lên.

max_file_uploads = 20 #Xác định số lượng tối đa các tệp tin được phép tải lên trong một lần yêu cầu.

post_max_size = 8M #Xác định kích thước tối đa của dữ liệu POST. Giá trị này phải lớn hơn hoặc bằng upload_max_filesize.

upload_tmp_dir = "/path/to/tmp/dir" #Đường dẫn đến thư mục tạm thời mà PHP sẽ sử dụng để lưu trữ các tệp tin trong quá trình tải lên.
```

*Khởi động lại Server: Sau khi thay đổi **php.ini**, bạn cần khởi động lại máy chủ web (ví dụ: apache, nginx) để các thay đổi có hiệu lực.*
*Bảo mật: đảm bảo rằng thư mục tạm thời cho tải lên tệp tin (**upload_tmp_dir**) an toàn và không thể truy cập từ bên ngoài.*

4. Tải lên ảnh và tệp tin

- **Thư mục tạm thời:**
  - **Vai trò**:
    - Lưu trữ tệp tin tải lên: Khi một tệp tin được tải lên thông qua một form trên web, nó đầu tiên được lưu vào thư mục tạm thời trên máy chủ. Điều này xảy ra trước khi tệp tin được di chuyển đến địa điểm lưu trữ cuối cùng của nó.
    - Xử lý an toàn: Việc sử dụng thư mục tạm thời giúp đảm bảo rằng xử lý tệp tin được thực hiện một cách an toàn và hiệu quả. Nếu có vấn đề xảy ra trong quá trình tải lên hoặc xử lý, tệp tin có thể được loại bỏ mà không ảnh hưởng đến hệ thống lưu trữ chính.
    - Giảm tải cho hệ thống lưu trữ chính: Thư mục tạm thời giúp giảm tải cho hệ thống lưu trữ chính. Các dữ liệu tạm thời không cần thiết sau khi sử dụng có thể được xóa mà không để lại dấu vết, giúp tiết kiệm không gian lưu trữ và tài nguyên hệ thống.
    - Xử lý cache và dữ liệu phiên: Thư mục tạm thời cũng có thể được sử dụng để lưu trữ dữ liệu cache và thông tin phiên, giúp cải thiện hiệu suất và tốc độ phản hồi của ứng dụng.

4. Tải lên ảnh và tệp tin

- **Xử lý ảnh khi upload:**

```
function resizeImage($file, $w, $h, $crop=FALSE) {
    list($width, $height) = getimagesize($file);
    $r = $width / $height;
    if ($crop) {
        if ($width > $height) {
            $width = ceil($width-($width*abs($r-$w/$h)));
        } else {
            $height = ceil($height-($height*abs($r-$w/$h)));
        }
        $newwidth = $w;
        $newheight = $h;
    } else {
        if ($w/$h > $r) {
            $newwidth = $h*$r;
            $newheight = $h;
        } else {
            $newheight = $w/$r;
            $newwidth = $w;
        }
    }
    $src = imagecreatefromjpeg($file);
    $dst = imagecreatetruecolor($newwidth, $newheight);
    imagecopyresampled($dst, $src, 0, 0, 0, 0, $newwidth, $newheight, $width, $height);
    return $dst;
}

// Usage
$resizedImage = resizeImage($target_file, 200, 200); // Resize to 200x200
imagejpeg($resizedImage, 'path/to/save/resized_image.jpg'); // Save resized image
```

5. Làm việc với ngày tháng và thời gian

- Việc xử lý và làm việc với ngày tháng và thời gian là một phần quan trọng của nhiều ứng dụng web. PHP cung cấp nhiều chức năng để giúp bạn tạo, định dạng, và xử lý ngày tháng và thời gian.
  - Hàm **date()** được sử dụng để định dạng ngày tháng thành chuỗi
    - format là định dạng của chuỗi ngày tháng được trả về.
    - timestamp là mốc thời gian Unix (số giây kể từ ngày 01/01/1970 00:00:00 UTC). Nếu bỏ qua tham số này, ngày và giờ hiện tại sẽ được sử dụng.

```
date(format, timestamp)
```

```
// Định dạng ngày tháng thành chuỗi
$date = date("d/m/Y H:i:s");

// In kết quả
echo $date; // 22/11/2023 04:17:00
```

Ký tự	Giá trị
d	Ngày trong tháng (01 đến 31)
m	Tháng trong năm (01 đến 12)
Y	Năm 4 chữ số
H	Giờ trong ngày (00 đến 23)
i	Phút trong giờ (00 đến 59)
s	Giây trong phút (00 đến 59)

5. Làm việc với ngày tháng và thời gian

- Việc xử lý và làm việc với ngày tháng và thời gian là một phần quan trọng của nhiều ứng dụng web. PHP cung cấp nhiều chức năng để giúp bạn tạo, định dạng, và xử lý ngày tháng và thời gian.

Tên hàm	Cú pháp	Mô tả
date()	`date(format, timestamp)`	Định dạng ngày tháng thành chuỗi.
strtotime()	`strtotime(string)`	Chuyển đổi chuỗi ngày tháng thành mốc thời gian Unix.
mktime()	`mktime(hour, minute, second, month, day, year)`	Tạo mốc thời gian Unix.
date_default_timezone_set()	`date_default_timezone_set(timezone)`	Đặt múi giờ mặc định cho các hàm xử lý ngày tháng và thời gian.
gmdate()	`gmdate(format)`	Định dạng ngày tháng thành chuỗi theo múi giờ UTC.
strftime()	`strftime(format, timestamp)`	Định dạng ngày tháng thành chuỗi theo định dạng tùy chỉnh.
date_diff()	`date_diff(date1, date2)`	Tính toán sự khác biệt giữa hai ngày.
date_format()	`date_format(date, format)`	Định dạng lại ngày tháng.
date_parse()	`date_parse(date)`	Tách chuỗi ngày tháng thành các thành phần riêng lẻ.

5. Làm việc với ngày tháng và thời gian

- ```
  // Tạo đối tượng DateTime
  $date = new DateTime("2023-11-22 04:17:00");

  // Lấy mốc thời gian Unix
  $timestamp = $date->getTimestamp();

  // In kết quả
  echo $timestamp; // 1665071020

  // Định dạng ngày tháng thành chuỗi
  $formattedDate = $date->format("d/m/Y H:i:s");

  // In kết quả
  echo $formattedDate; // 22/11/2023 04:17:00

  // Thêm một ngày vào ngày tháng và thời gian
  $date->add(new DateInterval("P1D"));

  // In kết quả
  echo $date->format("d/m/Y H:i:s"); // 23/11/2023 04:17:00
  ```
  ```

5. Làm việc với ngày tháng và thời gian

```php
// Lấy ngày trong tháng
$day = $date->getDay();

// In kết quả
echo $day; // 22

// Lấy tháng trong năm
$month = $date->getMonth();

// In kết quả
echo $month; // 11

// Lấy năm
$year = $date->getYear();

// In kết quả
echo $year; // 2023
```

5. Làm việc với ngày tháng và thời gian

- ```
  // Tạo đối tượng DateTime
  $date = new DateTime("2023-11-22 04:17:00");

  // Đặt ngày thành ngày 23 tháng 11 năm 2023
  $date->setDate(2023, 11, 23);

  // In kết quả
  echo $date->format("d/m/Y"); // 23/11/2023

  // Đặt thời gian thành 10 giờ, 20 phút và 30 giây
  $date->setTime(10, 20, 30);

  // In kết quả
  echo $date->format("H:i:s"); // 10:20:30

  // Đặt ngày tháng và thời gian thành mốc thời gian Unix 1665158430
  $date->setTimestamp(1665158430);

  // In kết quả
  echo $date->format("d/m/Y H:i:s"); // 23/11/2023 10:20:30

  // Thay đổi ngày tháng và thời gian bằng cách thêm 2 ngày và 3 giờ
  $date->modify("+2 days +3 hours");

  // In kết quả
  echo $date->format("d/m/Y H:i:s"); // 25/11/2023 13:20:30
  ```
  ```

5. Làm việc với ngày tháng và thời gian

- **Đối tượng DateInterval**: Lớp DateInterval trong PHP đại diện cho một khoảng thời gian. Lớp này cung cấp một số phương thức để truy cập và thao tác khoảng thời gian, bao gồm:
  - y trả về số năm trong khoảng thời gian.
  - m trả về số tháng trong khoảng thời gian.
  - d trả về số ngày trong khoảng thời gian.
  - h trả về số giờ trong khoảng thời gian.
  - i trả về số phút trong khoảng thời gian.
  - s trả về số giây trong khoảng thời gian.
  - f trả về số phần nghìn giây trong khoảng thời gian.

```
// Tạo đối tượng DateInterval
$interval = new DateInterval("P1Y2M3D");

// Lấy số năm trong khoảng thời gian
$years = $interval->y;

// In kết quả
echo $years; // 1

// Lấy số tháng trong khoảng thời gian
$months = $interval->m;

// In kết quả
echo $months; // 2

// Lấy số ngày trong khoảng thời gian
$days = $interval->d;

// In kết quả
echo $days; // 3
```

5. Làm việc với ngày tháng và thời gian

- **Đối tượng DateInterval**: Lớp DateInterval cũng cung cấp một số phương thức để thao tác khoảng thời gian, bao gồm:
  - add(interval) thêm một khoảng thời gian vào khoảng thời gian hiện tại.
  - sub(interval) trừ một khoảng thời gian khỏi khoảng thời gian hiện tại.
  - format(format) định dạng khoảng thời gian thành chuỗi..

```
// Thêm một năm vào khoảng thời gian
$interval->add(new DateInterval("P1Y"));

// In kết quả
echo $interval->format("Y-m-d"); // 2024-12-31

// Trừ một tháng khỏi khoảng thời gian
$interval->sub(new DateInterval("P1M"));

// In kết quả
echo $interval->format("Y-m-d"); // 2024-11-30
```

5. Làm việc với ngày tháng và thời gian

- **Đối tượng DatePeriod**: ợp các ngày tháng và thời gian, lặp đi lặp lại theo ập một số phương thức để truy cập và thao tác tập
  - getStartDate() trả về
  - getEndDate() trả về
  - getInterval() trả về kh
  - getIterator() trả về m

```
// Tạo đối tượng DatePeriod
$period = new DatePeriod("2023-11-01", "P1M", "D");

// Lấy ngày tháng và thời gian bắt đầu của tập hợp
$startDate = $period->getStartDate();

// In kết quả
echo $startDate->format("d/m/Y"); // 01/11/2023

// Lấy ngày tháng và thời gian kết thúc của tập hợp
$endDate = $period->getEndDate();

// In kết quả
echo $endDate->format("d/m/Y"); // 30/11/2023

// Lấy khoảng thời gian của tập hợp
$interval = $period->getInterval();

// In kết quả
echo $interval->format("P%dY%mM%dD"); // P1M0D

// Lặp qua tập hợp
foreach ($period as $date) {
    echo $date->format("d/m/Y");
}
```

5. LÀM VIỆC VỚI NGÀY THÁNG VÀ THỜI GIAN

- **Đối tượng DateTimeZone**: Lớp DateTimeZone trong PHP đại diện cho một múi giờ cụ thể. Lớp này cung cấp một số phươn
  - getName() trả về tên
  - getOffset() trả về độ
  - getTransitions() trả v

```
// Tạo đối tượng DateTimeZone
$timezone = new DateTimeZone("Asia/Ho_Chi_Minh");

// Lấy tên của múi giờ
$name = $timezone->getName();

// In kết quả
echo $name; // Asia/Ho_Chi_Minh

// Lấy độ lệch của múi giờ so với UTC
$offset = $timezone->getOffset();

// In kết quả
echo $offset; // 7200

// Lấy các chuyển tiếp múi giờ
$transitions = $timezone->getTransitions();

// In kết quả
print_r($transitions);
```

6. COOKIES & SESSIONS

- **Khái niệm**
  - **Cookie** là những tệp văn bản nhỏ được lưu trữ trên máy tính của khách truy cập bởi trình duyệt của họ. Chúng thường được sử dụng để lưu trữ thông tin như trạng thái đăng nhập, sở thích hoặc thông tin giỏ hàng.
  - **Session (Phiên)** là một trạng thái lưu trữ cục bộ được tạo bởi máy chủ web. Nó cho phép lưu trữ thông tin trên máy chủ cho đến khi phiên kết thúc. Phiên thường được sử dụng để lưu trữ thông tin như thông tin đăng nhập, thông tin giỏ hàng hoặc thông tin về trang web mà khách truy cập đang xem.
- **Cookie và phiên khác nhau như thế nào?**
  - Cookie được lưu trữ trên máy tính của khách truy cập, trong khi phiên được lưu trữ trên máy chủ web.
  - Cookie có thể được sử dụng để lưu trữ bất kỳ loại dữ liệu nào, trong khi phiên thường được sử dụng để lưu trữ thông tin quan trọng.
  - Cookie có thể bị xóa bởi người dùng hoặc trình duyệt của họ, trong khi phiên kết thúc theo thời gian hoặc khi người dùng đóng trình duyệt của họ.

6. COOKIES & SESSIONS

- **Khi nào nên sử dụng cookie?**
  - Cookie nên được sử dụng khi bạn cần lưu trữ dữ liệu nhỏ và không quan trọng trên máy tính của khách truy cập. Ví dụ: bạn có thể sử dụng cookie để lưu trữ trạng thái đăng nhập, sở thích hoặc thông tin giỏ hàng.

```
// Tạo cookie lưu trữ trạng thái đăng nhập
$cookie = new Cookie("logged_in", true);

// Đặt cookie
setcookie($cookie);
```

```
// Tạo cookie lưu trữ ngôn ngữ
$cookie = new Cookie("language", "en");

// Đặt cookie
setcookie($cookie);
```

```
// Thêm sản phẩm vào giỏ hàng
$cart->addProduct("product_1", 1);

// Lưu thông tin giỏ hàng vào cookie
$cookie = serialize($cart);
setcookie("cart", $cookie);
```

6. COOKIES & SESSIONS

- **Khi nào nên sử dụng phiên?**
  - Phiên nên được sử dụng khi bạn cần lưu trữ dữ liệu quan trọng trên máy chủ web của mình. Ví dụ: bạn có thể sử dụng phiên để lưu trữ thông tin đăng nhập, thông tin giỏ hàng hoặc thông tin về trang web mà khách truy cập đang xem.

```
// Khởi tạo phiên
session_start();

// Lưu thông tin đăng nhập vào phiên
$_SESSION["logged_in"] = true;
```

```
// Khởi tạo phiên
session_start();

// Thêm sản phẩm vào giỏ hàng
$_SESSION["cart"][] = [
    "product_id" => "product_1",
    "quantity" => 1,
];
```

```
// Khởi tạo phiên
session_start();

// Lưu trang web hiện tại vào phiên
$_SESSION["current_page"] = "/product/1";
```

6. COOKIES & SESSIONS

- **Kết hợp session và cookie**

```
// Khởi tạo phiên
session_start();

// Tạo cookie lưu trữ trạng thái đăng nhập
$cookie = new Cookie("logged_in", false);

// Đặt cookie
setcookie($cookie);

// Nếu người dùng đã đăng nhập
if (isset($_SESSION["user_id"])) {
    // Chuyển hướng đến trang chủ
    header("Location: /");
    exit;
}
```

Trong ví dụ này, dụng phiên để lưu trữ thông tin đăng nhập của người dùng, cookie để lưu trữ trạng thái đăng nhập của người dùng.

Khi người dùng truy cập trang đăng nhập, chúng ta kiểm tra xem họ đã đăng nhập chưa. Nếu họ đã đăng nhập, chúng ta chuyển hướng họ đến trang chủ. Nếu họ chưa đăng nhập, chúng ta hiển thị biểu mẫu đăng nhập.

Khi người dùng gửi biểu mẫu đăng nhập, chúng ta kiểm tra thông tin đăng nhập của họ trong cơ sở dữ liệu. Nếu thông tin đăng nhập chính xác, chúng ta lưu thông tin đăng nhập vào phiên và đặt cookie lưu trữ trạng thái đăng nhập. Sau đó, chúng ta chuyển hướng người dùng đến trang chủ.

```
// Nếu người dùng đang cố gắng đăng nhập
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Lấy tên người dùng và mật khẩu từ biểu mẫu
    $username = $_POST["username"];
    $password = $_POST["password"];

    // Truy vấn cơ sở dữ liệu để xác minh thông tin đăng nhập
    $query = "SELECT * FROM users WHERE username = ? AND password = ?";
    $stmt = $db->prepare($query);
    $stmt->execute([$username, $password]);
    $user = $stmt->fetch();

    // Nếu thông tin đăng nhập chính xác
    if ($user) {
        // Lưu thông tin đăng nhập vào phiên
        $_SESSION["user_id"] = $user["id"];

        // Đặt cookie lưu trữ trạng thái đăng nhập
        $cookie->setValue(true);
        setcookie($cookie);

        // Chuyển hướng đến trang chủ
        header("Location: /");
        exit;
    }
}
// Hiển thị biểu mẫu đăng nhập
include("login.php");
```

7. XỬ LÝ LỖI & NGOẠI LỆ

- **Làm thế nào để xử lý lỗi trong PHP?**
  - Trong PHP, có một số cách để xử lý lỗi. Cách phổ biến nhất là sử dụng hàm **error_reporting()** để đặt mức báo cáo lỗi và sử dụng hàm **set_error_handler()** để định nghĩa hàm xử lý lỗi tùy chỉnh.
  - Hàm error_reporting() được sử dụng để đặt mức báo cáo lỗi. Mức báo cáo lỗi xác định loại lỗi nào sẽ được báo cáo. Mức báo cáo lỗi có thể được đặt thành một trong các giá trị sau:
    - E_ALL (mặc định): Báo cáo tất cả lỗi, bao gồm cả lỗi nghiêm trọng và lỗi không nghiêm trọng.
    - E_ERROR (mức nghiêm trọng nhất): Báo cáo chỉ các lỗi nghiêm trọng, chẳng hạn như lỗi cú pháp và lỗi logic.
    - E_WARNING (mức không nghiêm trọng nhất): Báo cáo các lỗi không nghiêm trọng, chẳng hạn như lỗi về kiểu dữ liệu và lỗi về định nghĩa hàm.
    - E_PARSE (mức nghiêm trọng): Báo cáo lỗi cú pháp.
    - E_NOTICE (mức không nghiêm trọng): Báo cáo các lỗi không nghiêm trọng, chẳng hạn như lỗi về kiểu dữ liệu và lỗi về định nghĩa hàm.
    - E_STRICT (mức không nghiêm trọng): Báo cáo các lỗi nghiêm ngặt, chẳng hạn như các lỗi về cách sử dụng ngôn ngữ.
    - E_DEPRECATED (mức không nghiêm trọng): Báo cáo các lỗi đã bị lỗi thời.

7. XỬ LÝ LỖI

- **Làm thế nào để xử lý lỗi trong PHP?**
  - Trong PHP, có một số cách để xử lý lỗi. Cách phổ biến nhất là sử dụng hàm **error_reporting()** để đặt mức báo cáo lỗi và sử dụng hàm **set_error_handler()** để định nghĩa hàm xử lý lỗi tùy chỉnh.
  - Hàm set_error_handler() được sử dụng để định nghĩa hàm xử lý lỗi của riêng bạn. Hàm xử lý lỗi sẽ được gọi khi xảy ra lỗi. Hàm xử lý lỗi có hai tham số:
    - handler: Hàm xử lý lỗi.
    - error_types: Mức báo cáo lỗi mà hàm xử lý lỗi sẽ được gọi.

```
function my_error_handler($errno, $errstr, $errfile,
$errline) {
    // Xử lý lỗi
}

set_error_handler("my_error_handler", E_ALL);
```

7. XỬ LÝ LỖI

- **Xử lý ngoại lệ trong PHP**
  - Trong PHP, ngoại lệ là một sự kiện bất ngờ xảy ra trong quá trình thực thi mã. Ngoại lệ có thể được gây ra bởi nhiều nguyên nhân, chẳng hạn như lỗi cú pháp, lỗi logic, lỗi truy cập dữ liệu, v.v..
  - **Cách bắt ngoại lệ**: Để bắt ngoại lệ, bạn sử dụng câu lệnh try...catch. Câu lệnh try được sử dụng để xác định đoạn mã có thể gây ra ngoại lệ. Câu lệnh catch được sử dụng để bắt và xử lý ngoại lệ.

```
try {
    echo $x;
} catch (Exception $e) {
    echo "Xảy ra ngoại lệ: " . $e->getMessage();
}
```

  - **Cách xử lý ngoại lệ**: Trong câu lệnh catch, bạn có thể sử dụng đối tượng Exception để truy cập thông tin về ngoại lệ. Đối tượng Exception có các thuộc tính sau:
    - message: Thông báo lỗi của ngoại lệ.
    - code: Mã lỗi của ngoại lệ.
    - file: Tên tệp chứa ngoại lệ.
    - line: Số dòng chứa ngoại lệ.

```
try {
    echo $x;
} catch (Exception $e) {
    echo "Xảy ra ngoại lệ: " . $e->getMessage();
    echo "Mã lỗi: " . $e->getCode();
    echo "Tên tệp: " . $e->getFile();
    echo "Số dòng: " . $e->getLine();
}
```

7. XỬ LÝ LỖI

- **Xử lý ngoại lệ trong PHP**
  - Trong PHP, ngoại lệ là một sự kiện bất ngờ xảy ra trong quá trình thực thi mã. Ngoại lệ có thể được gây ra bởi nhiều nguyên nhân, chẳng hạn như lỗi cú pháp, lỗi logic, lỗi truy cập dữ liệu, v.v..
  - **Cách ném ngoại lệ**: Bạn cũng có thể tạo và ném ngoại lệ của riêng mình. Để ném ngoại lệ, bạn sử dụng câu lệnh throw.

```
if ($x < 0) {
    throw new Exception("Giá trị của x phải lớn hơn hoặc
bằng 0");
}
```

  - **Lợi ích của việc xử lý ngoại lệ**:
    - Giúp bạn quản lý các ngoại lệ một cách có kiểm soát.
    - Giúp bạn ngăn chặn các ngoại lệ gây ra lỗi fatal.
    - Giúp bạn phát hiện và khắc phục lỗi một cách nhanh chóng và dễ dàng.

“Câu hỏi & Thảo luận”

THE END!