



Trọn bộ kiến thức về

Lập trình PHP

Từ cơ bản đến nâng cao



Tổng hợp và biên soạn bởi **Stringee**

www.stringee.com



Lời mở đầu.....	3
Bài 1: PHP là gì? Các khái niệm cơ bản cho người mới bắt đầu.....	5
1. PHP là gì?.....	5
2. Lịch sử phát triển của PHP.....	6
3. Các khái niệm cơ bản trong PHP.....	7
4. Các thư viện và Framework phổ biến của PHP.....	19
Bài 2: Hướng dẫn sử dụng XAMPP: Tạo môi trường Lập trình PHP trên máy tính của bạn.....	20
1. Bước 1: Tải về và Cài đặt XAMPP.....	20
2. Bước 2: Khởi động và Dừng XAMPP.....	24
3. Bước 3: Tạo và chạy thử chương trình đầu tiên.....	25
Bài 3: Tìm hiểu về cú pháp cơ bản trong PHP (Phần 1).....	26
1. Mở và đóng đoạn code PHP.....	26
2. Comment code trong PHP.....	27
3. Xuất dữ liệu lên màn hình.....	28
4. Hàm var_dump() trong PHP.....	30
5. Nối chuỗi trong PHP.....	31
Bài 4: Tìm hiểu về cú pháp cơ bản trong PHP (Phần 2).....	32
2. Kiểu dữ liệu trong PHP.....	37
3. Toán tử trong PHP.....	44
Bài 5: Câu lệnh điều kiện và vòng lặp trong PHP.....	49
1. Câu lệnh điều kiện trong PHP.....	49
2. Vòng lặp trong PHP.....	54
Bài 6: Xử lý biểu mẫu trong PHP.....	56
1. Tạo biểu mẫu HTML.....	57
2. Xử lý biểu mẫu với PHP.....	58
3. Hiển thị kết quả sau khi được xử lý.....	59
Bài 7: Hàm trong PHP: Cách tạo và sử dụng hàm (Phần 1).....	59
1. Hàm trong PHP là gì?.....	59
2. Định nghĩa hàm (function) trong PHP.....	60
3. Hàm trả về kết quả.....	60
4. Sử dụng hàm trong PHP.....	60
5. Một số trường hợp khai báo hàm đặc biệt trong PHP.....	61
6. Một số trường hợp sử dụng hàm đặc biệt trong PHP.....	62
Bài 8: Hàm trong PHP: Cách tạo và sử dụng hàm (Phần 2).....	63
1. Hàm và phạm vi tác dụng của biến.....	63
\$globalVariable = "This is a global variable.";	64
function exampleFunction() {	64
global \$globalVariable;	64
echo \$globalVariable;	64
}	64
2. Tham biến và Tham trị.....	64
3. Tham số mặc định.....	65



4. Hàm với lượng tham số tùy ý.....	66
5. Chỉ dẫn kiểu (Type Hint).....	66
6. Một số ví dụ về hàm trong PHP.....	67
Bài 9: Kết nối cơ sở dữ liệu với PHP.....	70
1. Tại sao lại sử dụng MySQL trong PHP.....	70
2. Kết nối MySQL trong PHP.....	71
3. Chèn dữ liệu vào trong cơ sở dữ liệu.....	74
4. Cập nhật dữ liệu trong cơ sở dữ liệu.....	76
5. Truy vấn dữ liệu trong cơ sở dữ liệu.....	77
6. Xóa dữ liệu trong cơ sở dữ liệu.....	79
7. Một số lưu ý quan trọng khi truy vấn dữ liệu trong PHP.....	80
Bài 10: Bảo mật trong PHP.....	81
1. Lỗ hổng bảo mật trong PHP là gì?.....	81
2. SQL Injection (SQLi).....	82
3. Cross-Site Scripting (XSS).....	83
4. Cross-Site Request Forgery (CSRF).....	84
5. Một số lỗi thường gặp khác.....	85
Bài 11: Hướng dẫn tạo trang web động bằng PHP.....	85
1. PHP và web động.....	85
2. Các yếu tố cơ bản.....	86
3. Tương tác với cơ sở dữ liệu.....	89
4. Xử lý form và tương tác với người dùng.....	92
Bài 12: Xây dựng ứng dụng web hoàn chỉnh với PHP.....	94
1. Tạo cơ sở dữ liệu.....	94
2. Kết nối với cơ sở dữ liệu.....	95
3. Tạo trang đăng nhập (login.php).....	95
4. Trang Dashboard (dashboard.php).....	98
6. Những lưu ý khi viết khi xây dựng trang web bằng PHP.....	101
Tổng kết.....	102
Stringee Communication APIs - Giải pháp tích hợp tính năng giao tiếp vào App/web số 1 Việt Nam.....	103



Lời mở đầu

PHP đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới, đặc biệt trong lĩnh vực phát triển web. PHP không chỉ là công cụ lập trình linh hoạt mà còn là nền tảng cho nhiều dự án web lớn nhỏ trên toàn thế giới.

Hiểu biết sâu sắc về PHP giúp các nhà lập trình tự tin trong việc thiết kế, phát triển và duy trì các ứng dụng web, cũng như mở ra nhiều cơ hội nghề nghiệp trong lĩnh vực công nghệ thông tin.

Với phương châm mang đến giá trị hữu ích cho cộng đồng những nhà làm công nghệ, đội ngũ chuyên gia đến từ Stringee đã dành nhiều tâm sức nghiên cứu và biên soạn nên cuốn ebook hướng dẫn lập trình PHP từ cơ bản đến nâng cao. Dựa trên kinh nghiệm phát triển nền tảng lập trình giao tiếp cho hơn 70 triệu người dùng cuối, Stringee tin rằng những kiến thức được chia sẻ sẽ có tính ứng dụng cao cho bạn đọc.

Trong ebook này, bạn đọc sẽ được trang bị từ những kiến thức cơ bản nhất về PHP bao gồm cách cài đặt môi trường lập trình, các cú pháp, câu lệnh và hàm cơ bản. Từ đó, tiến tới những chủ đề nâng cao hơn, giúp bạn hiểu rõ hơn về cách sử dụng PHP để phát triển các ứng dụng web phức tạp.

Dù bạn là một người mới bắt đầu hoặc đã có kinh nghiệm trong lĩnh vực lập trình, ebook đều sẽ giúp bạn xây dựng nền tảng vững chắc và phát triển kỹ năng lập trình PHP của mình.

Trong quá trình biên tập không thể tránh khỏi sơ xuất, chúng tôi rất mong nhận được góp ý từ cộng đồng để Ebook được hoàn thiện hơn.

Ebook này được sản xuất bởi Đội ngũ biên tập thuộc Công ty Cổ phần Stringee. Vui lòng liên hệ với đơn vị thực hiện nếu muốn sao chép và chia sẻ thông tin trong cuốn sách này.



Stringee JSC.

Tầng 19, Leadvisors Tower, số 643 Phạm Văn Đồng, Bắc Từ Liêm, Hà Nội

1800 66 70

www.stringee.com

info@stringee.com

Từ bây giờ, hãy cùng bắt đầu hành trình khám phá ngôn ngữ lập trình PHP ngay thôi!

Ban biên tập Stringee,



Bài 1: PHP là gì? Các khái niệm cơ bản cho người mới bắt đầu

1. PHP là gì?

- PHP (viết tắt của "PHP: Hypertext Preprocessor") là một ngôn ngữ lập trình máy chủ được thiết kế đặc biệt cho phát triển ứng dụng web. PHP thường được sử dụng để tạo nội dung động trên các trang web, kết nối với cơ sở dữ liệu, xử lý biểu mẫu và thực hiện các tác vụ máy chủ.
- PHP chủ yếu được sử dụng trong việc phát triển ứng dụng web. Nó cho phép tạo các trang web động, tương tác với người dùng, và tạo nội dung dựa trên dữ liệu từ cơ sở dữ liệu hoặc các nguồn khác.
- PHP hỗ trợ [lập trình hướng đối tượng \(OOP\)](#), cho phép tạo và tái sử dụng mã một cách hiệu quả bằng cách sử dụng lớp, đối tượng và kế thừa.
- PHP có thể liên kết dễ dàng với các hệ quản trị cơ sở dữ liệu như MySQL, PostgreSQL, SQLite, và nhiều hệ quản trị cơ sở dữ liệu khác.
- PHP có khả năng tích hợp dễ dàng với HTML và các ngôn ngữ lập trình khác như JavaScript. Điều này giúp bạn tạo các ứng dụng web đa chức năng.
- Cộng đồng lập trình PHP rất lớn và đóng góp nhiều tài liệu, thư viện, và frameworks giúp bạn phát triển ứng dụng nhanh chóng.
- PHP là một ngôn ngữ mã nguồn mở, vì vậy bạn có thể tải về mã nguồn và sử dụng nó miễn phí. Điều này giúp tạo ra sự phát triển và đổi mới liên tục trong cộng đồng.



2. Lịch sử phát triển của PHP

- PHP được tạo ra bởi Rasmus Lerdorf vào năm 1994 với mục tiêu ban đầu là tạo ra một tập hợp các script để quản lý trang web cá nhân của ông ấy và được gọi là "Personal Home Page Tools" (PHP Tools).
- PHP/FI (1995-1997): PHP/FI (Personal Home Page/Forms Interpreter) là phiên bản đầu tiên của PHP được phát hành công khai vào năm 1995. Nó đã thêm tích hợp cơ sở dữ liệu và khả năng tương tác với người dùng qua biểu mẫu web.
- PHP 3 (1998): PHP 3 đã ra đời vào năm 1998 và đánh dấu sự phát triển mạnh mẽ của ngôn ngữ. Nó đã giới thiệu hỗ trợ lập trình hướng đối tượng, cải thiện khả năng kết nối cơ sở dữ liệu và nhiều tính năng mới khác.
- PHP 4 (2000): PHP 4 được phát hành vào năm 2000 với sự cải thiện về hiệu suất và bảo mật. Đây là phiên bản phổ biến và ổn định trong nhiều năm.
- PHP 5 (2004): PHP 5 được phát hành vào năm 2004 và đưa ngôn ngữ lên một tầm cao mới. Nó bổ sung nhiều tính năng mạnh mẽ bao gồm hỗ trợ cho lập trình hướng đối tượng, namespaces, và nhiều cải tiến về hiệu suất.



- **PHP 7 (2015)**: PHP 7, phát hành vào năm 2015, đã đem lại một sự cải tiến đáng kể về hiệu suất và tối ưu hóa bộ máy ảo Zend Engine. Nó giúp ứng dụng PHP chạy nhanh hơn và tiêu tốn ít tài nguyên hơn.

- **PHP 8 (2020)**: PHP 8 ra mắt vào năm 2020 và đưa ngôn ngữ vào một tầm cao mới với nhiều tính năng mới và cải tiến. Trong đó, JIT (Just-In-Time) Compiler là một trong những tính năng nổi bật, giúp tăng cường hiệu suất.

PHP đã trải qua một quá trình phát triển dài hơi và ngày càng trở nên mạnh mẽ, linh hoạt và hiệu quả. Nó được sử dụng rộng rãi cho phát triển ứng dụng web, từ các trang web cá nhân đến các hệ thống quản lý nội dung phức tạp và ứng dụng doanh nghiệp.

3. Các khái niệm cơ bản trong PHP

3.1 Biến trong PHP

Trong PHP, bạn có thể khai báo biến bằng cách sử dụng dấu "\$" theo sau là tên biến, được sử dụng để lưu trữ dữ liệu. Biến có thể chứa các giá trị như số nguyên, chuỗi văn bản, hoặc mảng:

Unset

```
$ten_bien = "Giá trị của biến"; // Chứa giá trị là một chuỗi văn bản  
$number = 1; // Giá trị là một số nguyên
```

3.2 Các kiểu dữ liệu trong PHP

PHP hỗ trợ nhiều kiểu dữ liệu khác nhau, bao gồm:

- **Kiểu số nguyên (Integer)**: Lưu trữ các số nguyên. Ví dụ: \$age = 25;
- **Kiểu số thực (Float hoặc Double)**: Lưu trữ các số thập phân. Ví dụ: \$price = 19.99;
- **Kiểu chuỗi (String)**: Lưu trữ các chuỗi ký tự. Ví dụ: \$name = "John";
- **Kiểu logic (Boolean)**: Lưu trữ giá trị true hoặc false. Ví dụ: \$is_active = true;



- **Kiểu mảng (Array)**: Lưu trữ nhiều giá trị trong một biến. Ví dụ:

Unset

```
$colors = array("red", "green", "blue");
```

- **Kiểu đối tượng (Object)**: Lưu trữ các đối tượng của các lớp đã định nghĩa. Ví dụ:

Unset

```
class Person {  
    public $name;  
    public $age;  
}
```

```
$person = new Person();  
$person->name = "Alice";  
$person->age = 30;
```

Kiểu không xác định (NULL): Một biến không có giá trị sẽ có kiểu NULL. Ví dụ: \$var = null;

3.3 Hàm trong PHP

Hàm là một khối mã được đặt tên và thực hiện một tác vụ cụ thể. PHP có nhiều hàm sẵn và bạn cũng có thể tự định nghĩa hàm riêng của mình. Dưới đây là cách định nghĩa và sử dụng hàm trong PHP:

Unset

```
function add($a, $b) {  
    return $a + $b;  
}  
  
$result = add(5, 3);  
echo $result; // Kết quả là 8
```



Một số loại hàm phổ biến trong PHP:

- **Hàm được xây dựng sẵn (Built-in Functions):** PHP cung cấp nhiều hàm được xây dựng sẵn để thực hiện các tác vụ phổ biến như xử lý chuỗi, mảng, số học, xử lý file, và nhiều hơn nữa. Ví dụ: `strlen()`, `strpos()`, `array_push()`, `file_get_contents()`...
- **Hàm người dùng tự định nghĩa (User-defined Functions):** Bạn có thể tự định nghĩa hàm trong PHP bằng cách sử dụng từ khóa `function`. Hàm người dùng tự định nghĩa cho phép bạn tái sử dụng mã và tách biệt các phần của mã thành các khối logic riêng biệt.

Unset

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

```
echo greet("John"); // Output: Hello, John!
```

- **Hàm vô danh (Anonymous Functions hoặc Closures):** Hàm vô danh là các hàm không có tên và thường được sử dụng khi cần chuyển hàm như một đối số của hàm khác.

Unset

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

```
echo $greet("John"); // Output: Hello, John!
```

- **Hàm inline (Arrow Functions):** Được giới thiệu từ **PHP 7.4**, hàm inline (arrow functions) là một dạng ngắn gọn của hàm vô danh cho các tác vụ đơn giản.

Unset

```
$greet = fn($name) => "Hello, $name!";
```



```
echo $greet("John"); // Output: Hello, John!
```

- **Hàm callback (Callback Functions):** Hàm callback là một hàm được truyền vào một hàm khác như một đối số và được gọi lại trong quá trình thực thi của hàm đó.

Ví dụ: Hàm `array_map()` thường được sử dụng với hàm callback để áp dụng một hàm lên mỗi phần tử của một mảng. Ta cùng xem cách mà hàm `array_map()` được sử dụng:

```
Unset
// Cú pháp: array_map(callable $callback, array $array1,
array ...$arrays): array
// $callback: Một hàm hoặc phương thức mà bạn muốn áp
dụng lên từng phần tử của mảng.
// $array1: Mảng đầu tiên hoặc mảng duy nhất mà bạn muốn
áp dụng hàm lên.
// $arrays: (Tùy chọn) Các mảng bổ sung nếu bạn muốn áp
dụng hàm lên nhiều mảng.

// Hàm định nghĩa
function square($x) {
    return $x * $x;
}

// Mảng ban đầu
$numbers = [1, 2, 3, 4, 5];

// Áp dụng hàm square lên mỗi phần tử của mảng numbers
$squaredNumbers = array_map("square", $numbers);

// In kết quả
print_r($squaredNumbers); // Output: Array ( [0] => 1 [1]
=> 4 [2] => 9 [3] => 16 [4] => 25 )
```



- **Hàm Generator (Generator Functions):** Generator functions cho phép bạn tạo ra các hàm có thể tạm dừng và tiếp tục từng bước khi cần thiết, giúp giảm bộ nhớ được sử dụng trong quá trình thực thi.

```
Unset
function gen() {
    yield 1;
    yield 2;
    yield 3;
}

foreach (gen() as $value) {
    echo $value; // Output: 123
}
```

Các loại hàm này cung cấp sự linh hoạt và mạnh mẽ để phát triển ứng dụng PHP. Việc chọn loại hàm phù hợp phụ thuộc vào nhu cầu và yêu cầu cụ thể của dự án hoặc tình huống lập trình.

3.4 Câu lệnh điều kiện trong PHP

Cũng giống với các ngôn ngữ lập trình khác, PHP có các câu lệnh điều kiện như:

- *if*: Kiểm tra một điều kiện và thực hiện một khối mã nếu điều kiện đúng.
- *if...else*: Kiểm tra một điều kiện và thực hiện một khối mã nếu điều kiện đúng và một khối mã khác nếu điều kiện sai.
- *switch*: Kiểm tra một biểu thức và thực hiện khối mã tương ứng với giá trị của biểu thức.

3.5 Vòng lặp trong PHP

Vòng lặp trong PHP giúp thực hiện các tác vụ lặp đi lặp lại:

- *for*: Thực hiện một khối mã nhiều lần với điều kiện kiểm tra.
- *while*: Thực hiện một khối mã miễn là điều kiện còn đúng.
- *do...while*: Thực hiện một khối mã ít nhất một lần và lặp lại nếu điều kiện đúng.



- **foreach**: Duyệt qua các phần tử trong mảng hoặc danh sách.

3.6 Làm việc với file và thư mục trong PHP

Làm việc với file và thư mục luôn là một phần không thể thiếu, nó là một chức năng chúng ta sẽ cần dùng rất nhiều. Có thể là lưu lại nhật ký của hệ thống, cũng có thể là đọc ra một số thông tin từ file để sử dụng trong các hoạt động khác. Chúng ta hãy cùng nhau xem một vài ví dụ sau:

- **Mở một tệp tin:**

Unset

```
fopen(string      $filename,      string      $mode,      bool
$use_include_path, resource $context);
```

Trong đó:

- **\$filename**: Đường dẫn hoặc tên của tệp tin cần mở.
- **\$mode**: Chế độ mở tệp tin. Đây là một chuỗi chỉ định cách mà tệp tin sẽ được mở và xử lý. Dưới đây là danh sách các chế độ khi mở file:

r	Mở tệp tin để đọc. Tệp tin phải tồn tại, nếu không hàm fopen sẽ trả về false .
r+	Mở tệp tin để đọc và ghi. Tệp tin phải tồn tại, nếu không hàm fopen sẽ trả về false .
w	Mở tệp tin để ghi. Nếu tệp tin không tồn tại, PHP sẽ cố gắng tạo ra tệp tin mới. Nếu tệp tin đã tồn tại, tất cả dữ liệu trong tập tin sẽ bị xóa đi và bắt đầu với tập tin rỗng.
w+	Mở tệp tin để đọc và ghi. Tương tự như "w", nếu tệp tin không tồn tại, PHP sẽ cố gắng tạo ra tệp tin mới và nếu đã tồn tại, tất cả dữ liệu trong tập tin sẽ bị xóa đi và bắt đầu với tập tin rỗng.
a	Mở tệp tin để ghi. Tệp tin sẽ được mở ở chế độ thêm vào cuối tệp tin. Nếu tệp tin không tồn tại, PHP sẽ cố gắng tạo ra tệp tin mới.
a+	Mở tệp tin để đọc và ghi. Tương tự như "a", nhưng cũng cho phép đọc nội dung hiện có của tệp tin.
x	Tạo tệp tin để ghi. Nếu tệp tin đã tồn tại, fopen sẽ trả về false .



x+

Tạo tệp tin để đọc và ghi. Tương tự như "`x`", nhưng cũng cho phép đọc nội dung hiện có của tệp tin.

- `$use_include_path` (tùy chọn): Một biến boolean chỉ định liệu bạn muốn sử dụng đường dẫn include hay không. Mặc định là FALSE. Nếu bạn đặt tham số này là TRUE, PHP sẽ tìm kiếm tệp tin trong các đường dẫn include được định nghĩa trong cấu hình của môi trường.
- `$context` (tùy chọn): Một ngữ cảnh stream được tạo ra bởi hàm `stream_context_create()`. Tham số này cho phép bạn chỉ định các tùy chọn bổ sung khi mở tệp tin, chẳng hạn như quyền truy cập hoặc các tùy chọn mã hóa.

- Ví dụ

Unset

```
$file = fopen("file.txt", "r") or die("Không thể mở tệp tin!");
```

- **Đọc nội dung tệp tin:**

Unset

```
while(!feof($file)) {  
    echo fgets($file) . "<br>";  
}
```

- **Ghi vào tệp tin:**

Unset

```
$file = fopen("file.txt", "w") or die("Không thể mở tệp tin!");  
$txt = "Nội dung mới để ghi vào tệp tin.";  
fwrite($file, $txt);  
fclose($file);
```

- **Đóng tệp tin:**



Unset

```
fclose($file);
```

- Tạo một thư mục:

Unset

```
mkdir("name_folder");
```

- Di chuyển một tập tin/thư mục:

Unset

```
rename("name_folder", "new_path/name_folder");
```

- Xóa một thư mục:

Unset

```
rmdir("name_folder");
```

- Phân quyền cho tập tin:

Unset

```
chmod("file.txt", 0644);
```

Cú pháp `chmod()` trong PHP sử dụng một số để đại diện cho các quyền truy cập trên tệp và thư mục. Số này được biểu diễn dưới dạng bát phân (octal). Dưới đây là ý nghĩa của các số trong cú pháp `chmod()`:

- **0**: Không có quyền
- **1**: Thực thi (Execute)
- **2**: Ghi (Write)
- **4**: Đọc (Read)

Các quyền này được kết hợp lại với nhau. Số đại diện cho các quyền cụ thể là tổng của các giá trị tương ứng.

Ví dụ:

- **0755**: Owner có quyền đọc, ghi, thực thi ($4 + 2 + 1 = 7$), và Group và Others có quyền thực thi (1).



- **0644:** Owner có quyền đọc và ghi ($4 + 2 = 6$), và Group và Others có quyền chỉ đọc (4).

Để hiểu rõ hơn, đây là một số quyền thường gặp:

- **0700:** Owner có quyền đọc, ghi, thực thi; Group và Others không có quyền.
- **0755:** Owner có quyền đọc, ghi, thực thi; Group và Others có quyền thực thi.
- **0644:** Owner có quyền đọc và ghi; Group và Others chỉ có quyền đọc.
- **0600:** Owner có quyền đọc và ghi; Group và Others không có quyền.

- **Phân quyền cho thư mục:**

```
Unset
```

```
chmod( "tenduthumuc" , 0755 );
```

Chú ý: Khi làm việc với file bạn cần chú ý đến việc cấp quyền đúng và đầy đủ cho file. Thường thì đây là lỗi rất hay gặp phải, và cũng là một trong những lỗi thường bị bỏ qua trong lúc phát triển phần mềm.

3.7 Cookies và Session trong PHP

Cookie và Session là hai khái niệm quan trọng trong lập trình web, đặc biệt là trong PHP, để quản lý trạng thái của người dùng trên các trang web. Dưới đây là một tóm tắt về Cookie và Session trong PHP:

- **Cookie:**
 - Cookie là một dòng thông tin được lưu trữ trên máy tính của người dùng thông qua trình duyệt web.
 - Cookie thường được sử dụng để lưu trữ thông tin về người dùng như tên người dùng, giỏ hàng, cài đặt ngôn ngữ, vv.
 - Các cookie có thể được thiết lập, truy xuất và chỉnh sửa thông qua PHP bằng cách sử dụng hàm `setcookie()` hoặc `setrawcookie()`.
- **Session:**
 - Session là một cách để lưu trữ thông tin về phiên làm việc của người dùng trên máy chủ.
 - Khi một phiên được bắt đầu, PHP tạo ra một ID phiên duy nhất cho người dùng, và thông tin được lưu trữ trên máy chủ.
 - Các phiên thường được sử dụng để lưu trữ thông tin quan trọng như thông tin đăng nhập, giỏ hàng, vv.



- Trong PHP, các phiên được quản lý thông qua các biến toàn cục như `$_SESSION`.

So sánh:

- Sự khác biệt chính giữa Cookie và Session** là ở cách lưu trữ dữ liệu. Cookie được lưu trữ trên máy tính của người dùng, trong khi Session được lưu trữ trên máy chủ.
- Độ an toàn:** Do Cookie được lưu trữ trên máy tính của người dùng, nên nó có thể bị sửa đổi hoặc giả mạo. Trong khi đó, Session được lưu trữ trên máy chủ, giúp tăng cường bảo mật.
- Thời gian tồn tại:** Cookie có thể được đặt để tồn tại trong một khoảng thời gian nhất định hoặc đến khi người dùng đóng trình duyệt. Trong khi đó, Session tồn tại cho đến khi người dùng đăng xuất hoặc đóng trình duyệt.

Một số ví dụ về cách sử dụng Cookie và Session:

```
Unset
// Thiết lập một cookie
setcookie("user", "John Doe", time() + 3600, "/");

// Truy cập và hiển thị giá trị của cookie
echo $_COOKIE["user"];

// Thiết lập một session
session_start();
$_SESSION["username"] = "johndoe";

// Truy cập và hiển thị giá trị của session
echo $_SESSION["username"];

// Xóa một session
unset($_SESSION["username"]);
```

3.8 Xử lý ngoại lệ trong PHP

Trong PHP, một ngoại lệ (exception) là một cơ chế để xử lý các tình huống ngoại lệ, như các lỗi hoặc điều kiện không mong muốn xảy ra trong quá trình thực thi của một chương trình. Khi một ngoại lệ xảy ra, nó có thể được ném (thrown) từ



một phương thức hoặc khối mã và được bắt (caught) và xử lý trong một khối mã khác.

Dưới đây là một số điểm quan trọng về ngoại lệ trong PHP:

- **Ném Ngoại Lệ (Throwing Exceptions):** Để ném một ngoại lệ trong PHP, bạn sử dụng từ khóa `throw` kèm theo một đối tượng Exception hoặc các lớp con của Exception.

Unset

```
throw new Exception("Lỗi xảy ra!");
```

- **Bắt Ngoại Lệ (Catching Exceptions):** Để bắt ngoại lệ trong PHP, bạn sử dụng các khối `try` và `catch`. Mã trong khối `try` là nơi mà bạn muốn giám sát và bắt các ngoại lệ. Mã trong khối `catch` được thực thi khi một ngoại lệ được ném.

Unset

```
try {
    // Mã có thể gây ra ngoại lệ
    $result = 100 / 0;
} catch (Exception $e) {
    // Xử lý ngoại lệ
    echo "Caught exception: " . $e->getMessage();
}
```

- **Thông tin Ngoại Lệ (Exception Information):** Khi một ngoại lệ xảy ra, bạn có thể truy cập các thông tin về ngoại lệ đó, bao gồm thông điệp, mã lỗi và vị trí xảy ra lỗi, thông qua các phương thức như `getMessage()`, `getCode()`, `getFile()`, `getLine()`...
- **Xử Lý Ngoại Lệ (Exception Handling):** Xử lý ngoại lệ cho phép bạn kiểm soát và phản ứng với các tình huống ngoại lệ một cách linh hoạt, chẳng hạn như ghi log, hiển thị thông báo cho người dùng hoặc thực hiện các hành động phục hồi.

3.9 Mô hình MVC (Model-View-Controller) trong PHP

- Mô hình MVC:
 - o **Model:** Lớp Model chứa dữ liệu và logic xử lý dữ liệu.



- **View:** View là giao diện người dùng, hiển thị dữ liệu cho người dùng và thu thập dữ liệu từ người dùng.
- **Controller:** Controller là lớp điều khiển, điều hướng dữ liệu giữa Model và View, xử lý yêu cầu của người dùng và cập nhật Model dựa trên dữ liệu từ View.

Dưới đây chúng ta cùng nhau xem qua một ví dụ nhỏ cho mô hình MVC:

- Model: [models/ExampleModel.php](#)

```
Unset
class ExampleModel {
    public function getData() {
        return "Data from the model";
    }
}
```

- View: [views/example_view.php](#)

```
Unset
<html>
<head><title>Example View</title></head>
<body>
    <h1><?php echo $data; ?></h1>
</body>
</html>
```

- Controller: [controllers/ExampleController.php](#)

```
Unset
class ExampleController {
    public function show() {
        $model = new ExampleModel();
        $data = $model->getData();
        include 'views/example_view.php';
    }
}
```



Một số lưu ý trong mô hình MVC

- Mô hình MVC chia ứng dụng thành ba phần, mỗi phần đảm nhận một nhiệm vụ cụ thể.
- Controller là điểm khởi đầu của ứng dụng, nhận yêu cầu từ người dùng và quyết định xử lý yêu cầu đó.
- Model chứa dữ liệu và logic xử lý dữ liệu, trong khi View hiển thị dữ liệu cho người dùng.
- Việc sử dụng mô hình MVC giúp tăng tính bảo quản mã nguồn, dễ dàng bảo trì và mở rộng ứng dụng.

4. Các thư viện và Framework phổ biến của PHP

- *Laravel*: Laravel là một trong những framework PHP phổ biến nhất. Nó cung cấp cơ chế mạnh mẽ cho việc xây dựng ứng dụng web và ứng dụng di động. Laravel sử dụng cú pháp đẹp và dễ đọc, hỗ trợ lập trình hướng đối tượng và cung cấp nhiều tính năng như xử lý người dùng, quản lý cơ sở dữ liệu, và nhiều thứ khác. Laravel có một cộng đồng lập trình viên lớn và nhiều tài liệu học tập, package mở rộng, và khung làm việc (frameworks) bên ngoài giúp chúng ta có thể phát triển ứng dụng một cách nhanh chóng và hiệu quả.
- *Symfony*: Symfony là một framework PHP mạnh mẽ và linh hoạt, tập trung vào việc xây dựng ứng dụng web lớn và phức tạp. Nó cung cấp nền tảng cho việc phát triển các ứng dụng có khả năng mở rộng.
- *Zend Framework*: Zend Framework là một framework PHP do Zend Technologies phát triển. Nó tập trung vào lập trình hướng đối tượng và cung cấp nhiều thành phần và thư viện mạnh mẽ cho phát triển ứng dụng web.
- *CodeIgniter*: CodeIgniter là một framework PHP nhẹ và nhanh chóng. Nó thích hợp cho việc xây dựng ứng dụng web nhỏ và trung bình mà cần sự đơn giản và hiệu quả.
- *Yii*: Yii là một framework PHP nhanh và mạnh mẽ. Nó được xây dựng với mục tiêu tối ưu hóa hiệu suất và độ bảo mật.
- *Phalcon*: Phalcon là một framework PHP được viết bằng C và được tích hợp vào PHP dưới dạng một mô-đun mở rộng. Điều này làm cho nó rất nhanh và hiệu quả.



- *Slim*: Slim là một framework PHP nhẹ và đơn giản, tập trung vào việc xây dựng ứng dụng web RESTful.
- *CakePHP*: CakePHP là một framework PHP được xây dựng dựa trên mô hình lập trình mảng (MVC). Nó giúp bạn nhanh chóng xây dựng các ứng dụng web theo kiểu MVC.

Bài 2: Hướng dẫn sử dụng XAMPP: Tạo môi trường Lập trình PHP trên máy tính của bạn

XAMPP là một gói phần mềm mã nguồn mở phổ biến cho phép bạn tạo một môi trường lập trình web trên máy tính cá nhân. Nó bao gồm Apache (một máy chủ web), MySQL (hệ quản trị cơ sở dữ liệu), PHP (ngôn ngữ lập trình phía máy chủ) và thậm chí cả phpMyAdmin (một công cụ quản trị cơ sở dữ liệu MySQL) để giúp bạn phát triển và kiểm tra ứng dụng web dễ dàng. Trong bài này, chúng ta sẽ cùng nhau cài đặt và sử dụng XAMPP.

1. Bước 1: Tải về và Cài đặt XAMPP

Truy cập trang web chính thức của XAMPP tại <https://www.apachefriends.org/index.html>.

Tại trang chính, bạn sẽ thấy các phiên bản XAMPP cho cả Windows, macOS và Linux. Chọn phiên bản phù hợp với hệ điều hành của bạn và tải xuống gói cài đặt. Dưới đây là các bước cài đặt XAMPP cho hệ điều hành Windows:



XAMPP Apache + MariaDB + PHP + Perl

What is XAMPP?

XAMPP is the most popular PHP development environment

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.



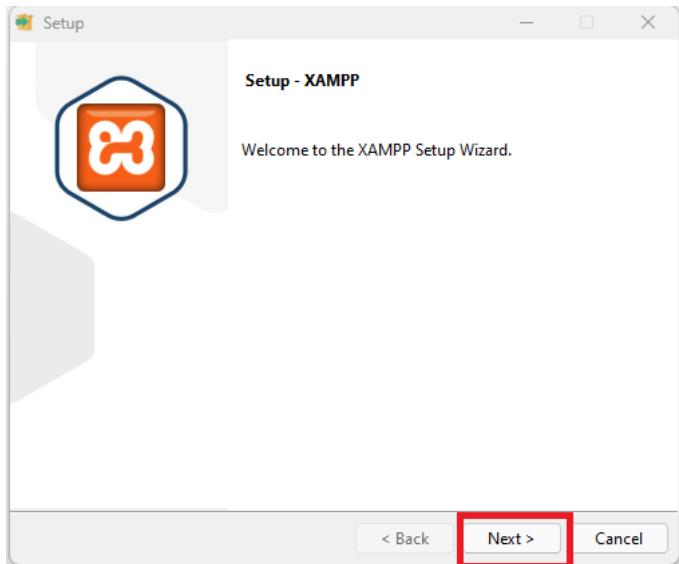
Download
Click here for other versions

XAMPP for Windows
8.2.4 (PHP 8.2.4)

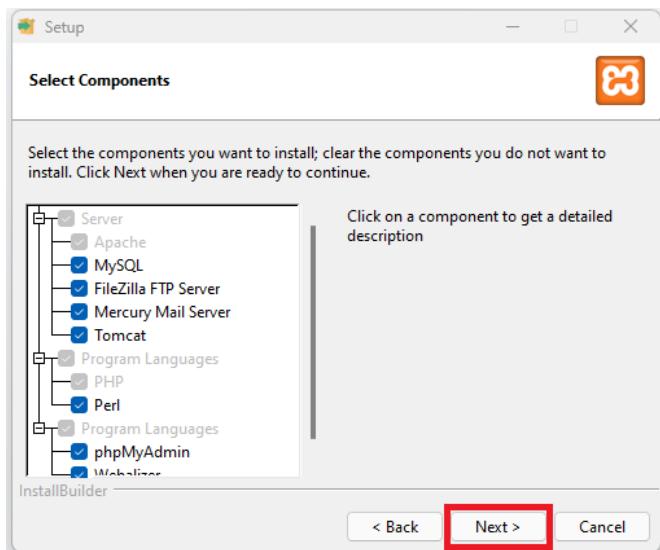
XAMPP for Linux
8.2.4 (PHP 8.2.4)

XAMPP for OS X
8.2.4 (PHP 8.2.4)

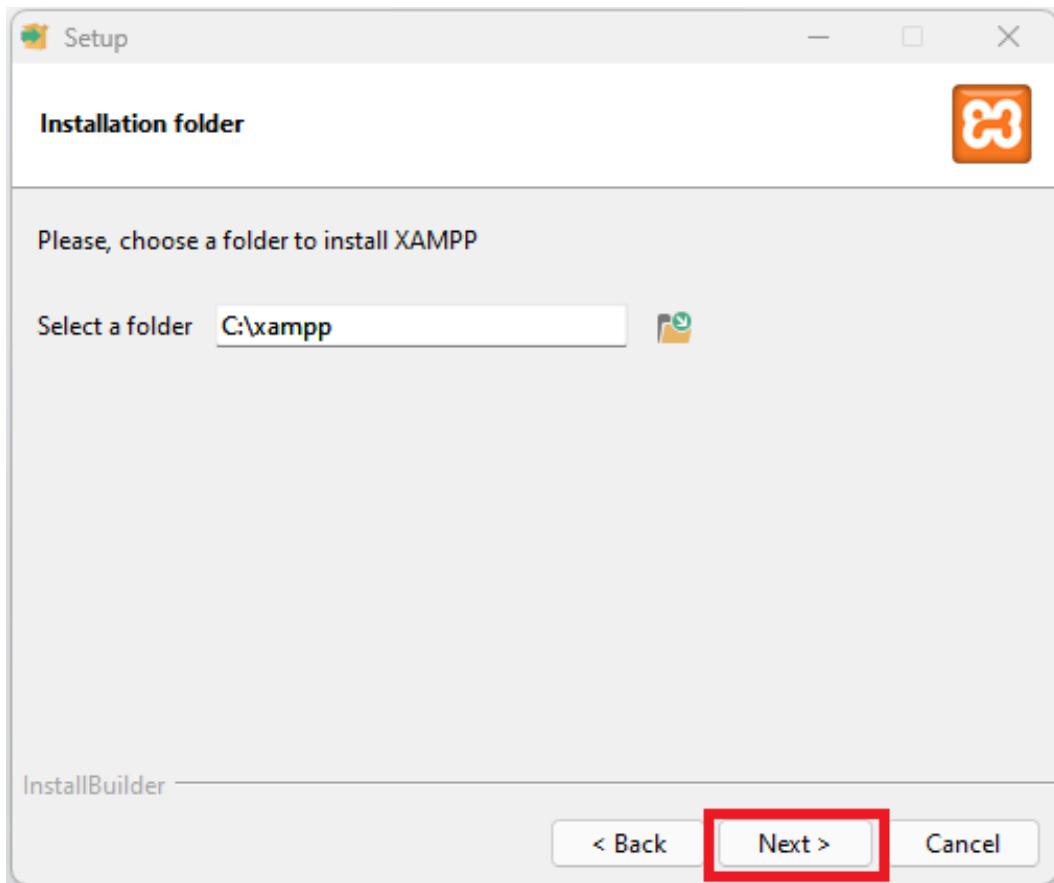
Chúng ta chọn XAMPP for Windows và tải về. Chạy file vừa tải về và thực hiện lần lượt các bước:



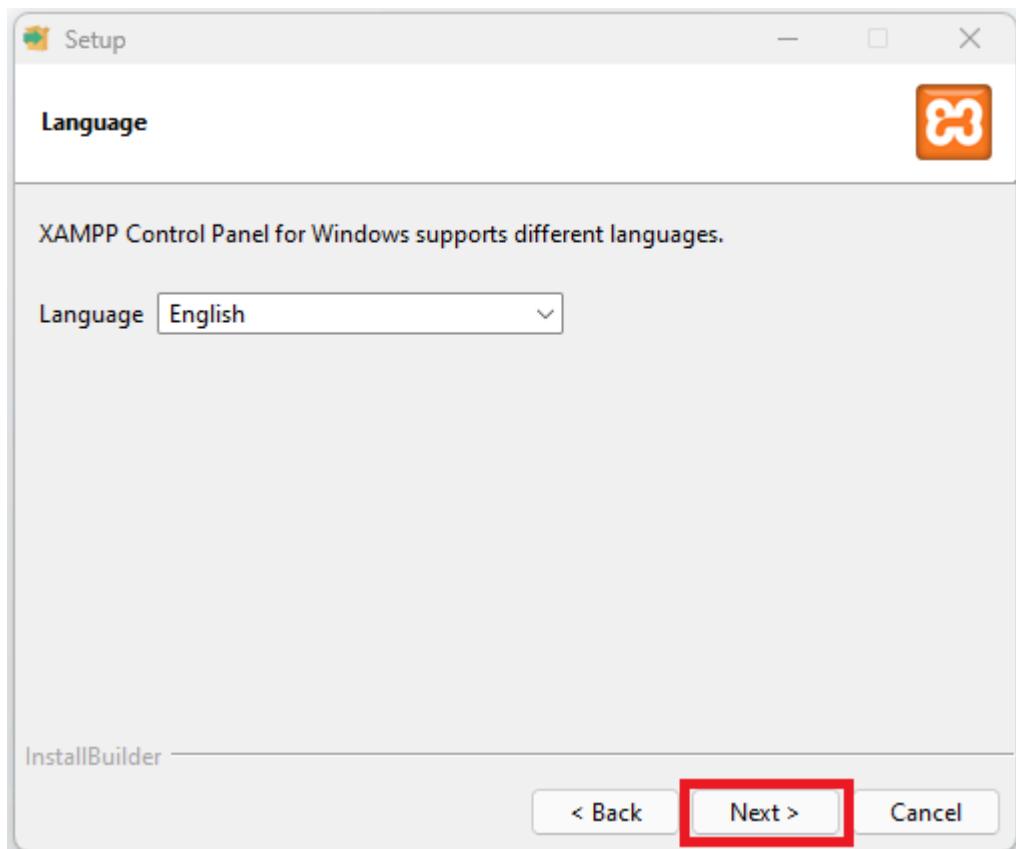
Bắt đầu cài đặt



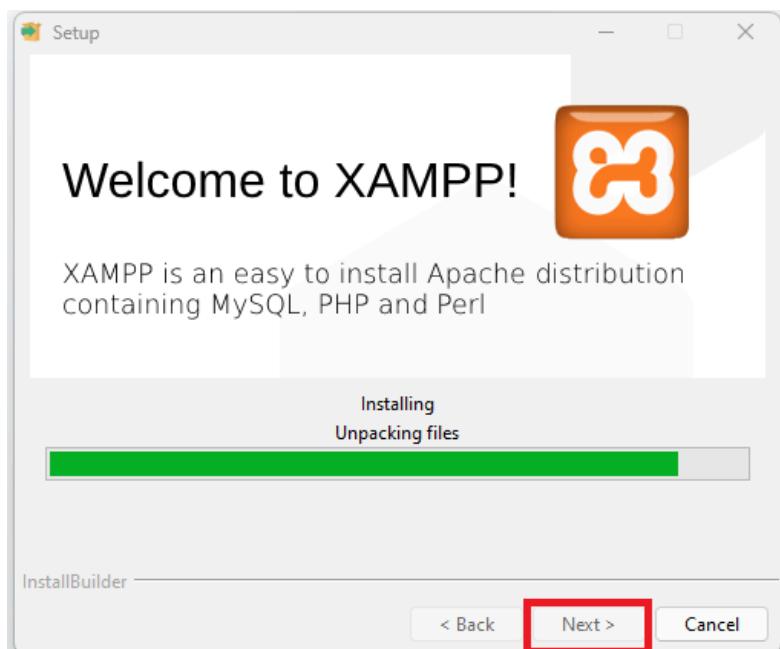
Bước này chúng ta chọn các thành phần để cài đặt, chúng ta để mặc định



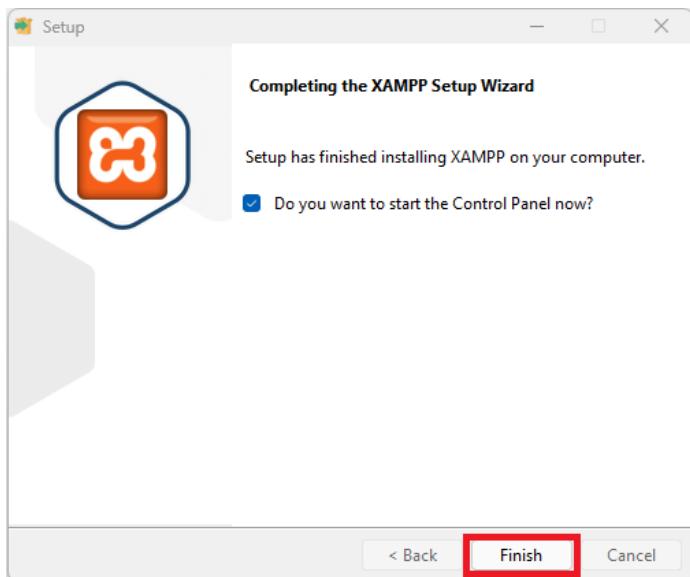
Bước này chúng ta chọn thư mục để cài đặt XAMPP



Chọn ngôn ngữ, chúng ta để tiếng Anh.

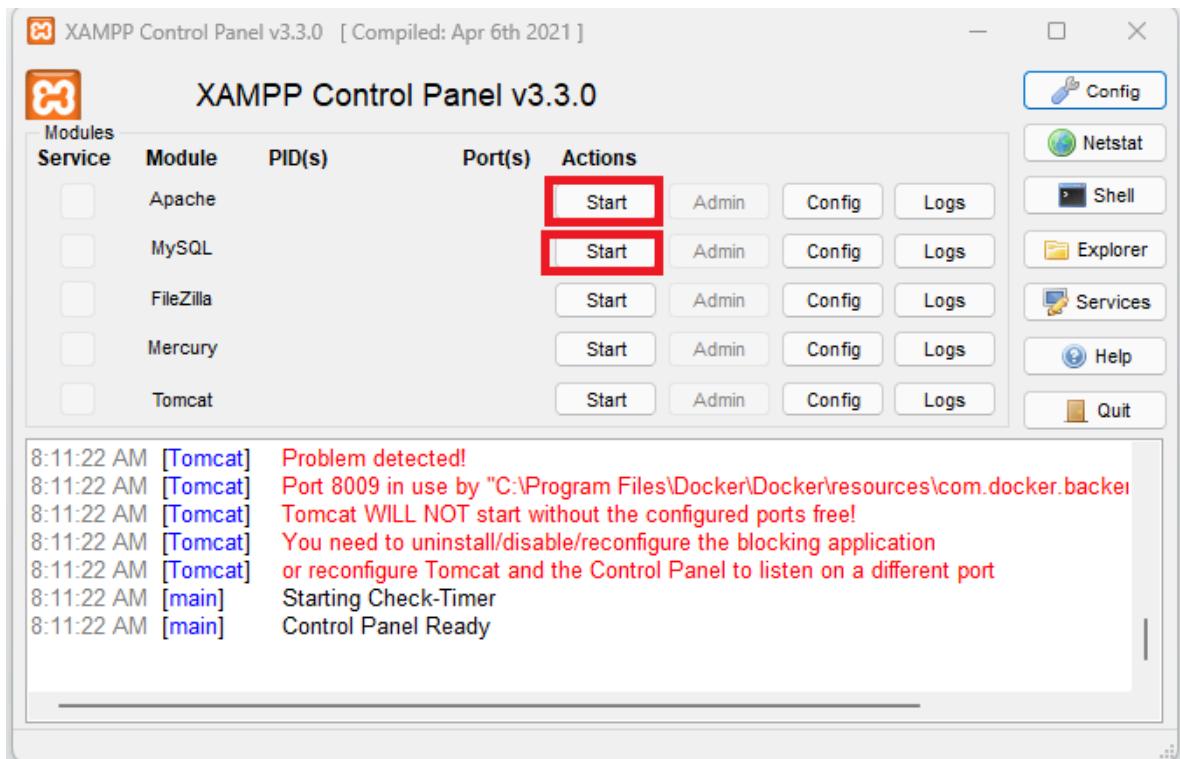


XAMPP đang được cài vào máy.



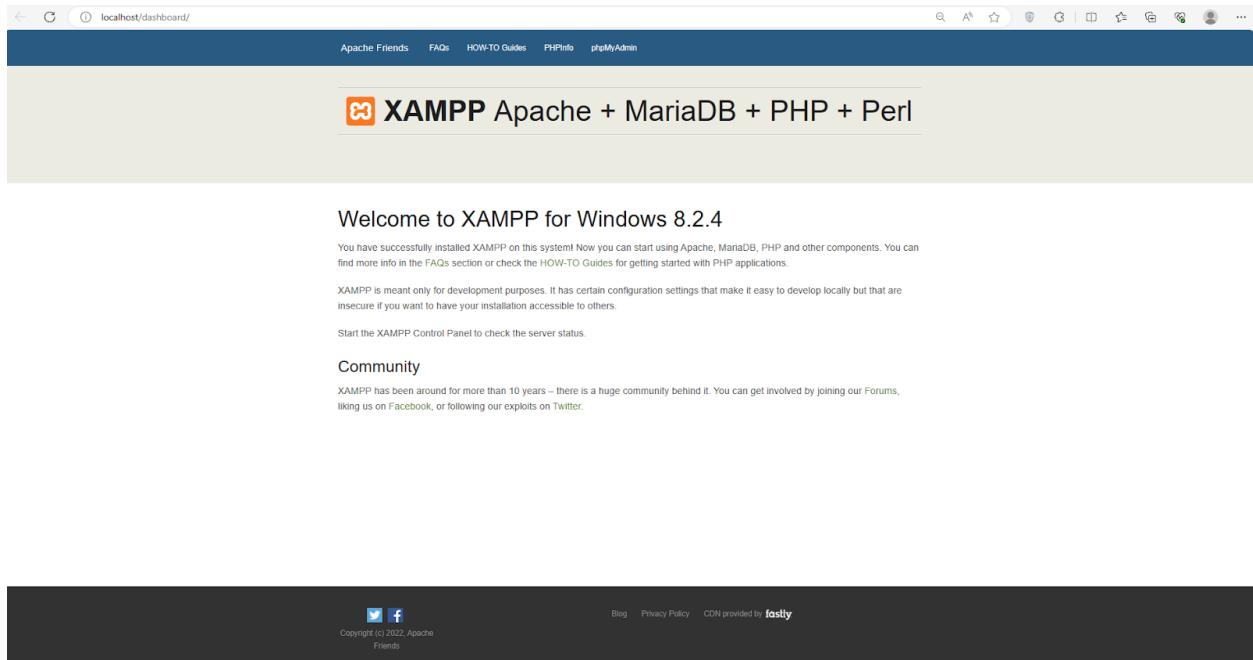
2. Bước 2: Khởi động và Dừng XAMPP

Sau khi cài đặt, bạn có thể tìm biểu tượng XAMPP Control Panel trên máy tính của bạn và khởi động nó. Trong giao diện này, bạn sẽ thấy các biểu tượng cho Apache và MySQL. Bạn có thể bấm vào nút "Start" ở cả hai để khởi động máy chủ web và cơ sở dữ liệu.





Để kiểm tra xem máy chủ web đã hoạt động, mở trình duyệt và nhập URL "<http://localhost>" hoặc "<http://127.0.0.1>". Bạn sẽ thấy trang chào mừng của XAMPP.



Để dừng máy chủ web hoặc cơ sở dữ liệu, bạn có thể quay lại giao diện XAMPP Control Panel và bấm nút "Stop" tương ứng.

3. Bước 3: Tạo và chạy thử chương trình đầu tiên

Trang chủ của máy chủ web XAMPP được lưu trữ trong thư mục gốc. Chúng ta vừa chọn "C:\xampp\htdocs" làm thư mục gốc, còn ở trên macOS và Linux, thư mục gốc thường là "/opt/lampp/htdocs/".

Để chạy một chương trình PHP đơn giản bằng XAMPP, tôi vừa tạo trong thư mục "C:\xampp\htdocs" folder có tên là Stringee và thêm 1 file mới với tên "index.php" trong thư mục gốc và thêm mã PHP vào đó. Ví dụ:

Unset

```
<?php  
echo "Chào mừng bạn đến với Stringee";
```



Chúng ta có được kết quả:



Chào mừng bạn đến với Stringee

Bài 3: Tìm hiểu về cú pháp cơ bản trong PHP (Phần 1)

Để bắt đầu với PHP, chúng ta cần nắm vững những cú pháp cơ bản trong PHP. Cùng nhau tìm hiểu về một số cú pháp quan trọng trong PHP.

1. Mở và đóng đoạn code PHP

Hiện tại, PHP hỗ trợ chúng ta 2 cách mở và đóng đoạn code PHP, sau đây chúng ta sẽ cùng nhau tìm hiểu về những cách đóng mở đoạn code PHP ngay sau đây.

- Cách 1: Đây là cách thông dụng, được sử dụng nhiều nhất. Đồng thời đây cũng là cú pháp được khuyên dùng để đảm bảo code PHP có khả năng chạy trên các môi trường khác nhau mà không có lỗi.

Unset

```
<?php  
// Bắt đầu đoạn mã PHP
```



```
echo "Đây là đoạn mã PHP sẽ được thực thi";  
?>
```

Chú ý: Với những file chỉ bao gồm mã php chúng ta có thể sử dụng cú pháp <?php để mở đoạn mã PHP mà không cần sử dụng ?> để đóng.

Unset

```
<?php  
// Bắt đầu đoạn mã PHP  
echo "Đây là đoạn mã PHP sẽ được thực thi";
```

- Cách 2: Cũng tương tự như cách trên, với cách này chúng ta hoàn toàn có thể thực thi được đoạn mã PHP.

Unset

```
<?  
// Bắt đầu đoạn mã PHP  
echo "Đây là đoạn mã PHP sẽ được thực thi";  
?>
```

Chú ý:

- Cú pháp trên không phải là cú pháp tiêu chuẩn trong PHP mà là cú pháp PHP viết tắt, cú pháp trên được gọi là "short tags" hoặc "short open tags".
- Việc sử dụng cú pháp rút gọn này không được khuyến nghị nhất là khi chúng ta cần đảm bảo sự di động cũng như tương thích mã nguồn trên nhiều máy chủ.
- Để sử dụng được cú pháp trên chúng ta cần cho phép sử dụng nó bằng cách chỉnh sửa cấu hình short_open_tag trong tập cấu hình PHP php.ini

2. Comment code trong PHP



Trong khi viết mã, chúng ta sẽ thường xuyên gặp những đoạn mã phức tạp và khó hiểu. Lúc này, việc comment được sử dụng để chúng ta có thể chèn những đoạn chú thích vào trong mã nguồn của mình, việc này góp phần giải thích hoặc đánh dấu một đoạn code nào đó. Với việc sử dụng comment một cách hợp lý, chúng ta có thể giúp cho việc bảo trì cũng như nâng cấp mã sau này trở nên dễ dàng hơn.

Unset

```
<?php
// Chúng ta có thể comment 1 dòng như ví dụ này.

# Hoặc chúng ta cũng có một cách khác để comment như
# trong ví dụ này.

/*
    Hoặc khi chúng ta cần comment một đoạn mã quá dài
    chúng ta có thể sử dụng cú pháp như trên.
*/
?>
```

>>> Xem thêm bài viết tương tự tại đây:

- [Lập trình hướng đối tượng \(OOP\) với PHP](#)
- [Hàm ẩn danh \(Anonymous Function\) trong PHP](#)

3. Xuất dữ liệu lên màn hình

Trong PHP, các hàm echo và print đều cho phép chúng ta xuất dữ liệu ra màn hình hoặc trả về nội dung dưới dạng chuỗi, tuy nhiên chúng sẽ có những khác biệt nhỏ về cách sử dụng. Sau đây chúng ta sẽ cùng nhau tìm hiểu về chúng.

- echo:
 - Hàm echo thông thường được sử dụng mà không cần đến dấu ngoặc tròn.
 - Hàm echo cho phép xuất nhiều giá trị cùng lúc và không trả về giá trị khi thực thi.

Ví dụ:



Unset

```
<?php  
echo 'Đây là dòng text được in ra bởi hàm echo' .  
PHP_EOL;  
  
echo 'Đây là dòng text', ' được in ra bởi hàm echo' .  
PHP_EOL;  
  
$text = 'Đây là dòng text được in ra bởi hàm echo';  
echo $text;
```

Kết quả:

Unset

```
Đây là dòng text được in ra bởi hàm echo  
Đây là dòng text được in ra bởi hàm echo  
Đây là dòng text được in ra bởi hàm echo
```

- Chú ý:

- Trong một số trường hợp, thường gặp nhất là trong file *.html chúng ta sẽ thấy echo được viết tắt bằng cú pháp <?= ... ?>.
- Vì đây cũng là cú pháp viết tắt cho nên chúng ta cũng cần điều chỉnh cấu hình short_open_tag, cũng như nó không được khuyến cáo sử dụng.

Unset

```
<p><?= 'Đây là đoạn text sẽ được in ra màn hình' ?></p>
```

- print:

- Hàm print ta sẽ thường thấy nó được sử dụng có dấu ngoặc tròn (tuy việc này là không cần thiết).



- Tuy nhiên, khác với echo, hàm print chỉ cho phép thực hiện với một giá trị tại một thời điểm và nó sẽ trả về giá trị sau khi thực thi, nên ta có thể gán giá trị trả về của print cho một biến.

Ví dụ:

Unset

```
<?php  
print("Đây là đoạn text sẽ được in ra màn hình!" .  
PHP_EOL);  
  
$message = "Đây là đoạn text sẽ được in ra màn hình với  
biến message";  
$result = print($message);
```

Kết quả:

Unset

```
Đây là đoạn text sẽ được in ra màn hình!  
Đây là đoạn text sẽ được in ra màn hình với biến message
```

4. Hàm var_dump() trong PHP

Trong PHP, var_dump() được sử dụng để hiển thị thông tin về kiểu dữ liệu, giá trị và kích thước của một biến hoặc biểu thức. Thông thường, hàm var_dump() sẽ được sử dụng nhiều trong quá trình debug và kiểm tra dữ liệu.

Ví dụ:

Unset

```
<?php  
$number = 2023;
```



```
$string = "Chuỗi được in ra bởi hàm var_dump()";
$array = [2, 0, 2, 3];
$assocArray = ['text' => "Hello World!", "number" =>
2023];

var_dump($number);
var_dump($string);
var_dump($array);
var_dump($assocArray);
```

Kết quả:

Unset

```
int(2023)
string(44) "Chuỗi được in ra bởi hàm var_dump()"
array(4) {
    [0]=>
    int(2)
    [1]=>
    int(0)
    [2]=>
    int(2)
    [3]=>
    int(3)
}
array(2) {
    ["text"]=>
    string(12) "Hello World!"
    ["number"]=>
    int(2023)
}
```

5. Nối chuỗi trong PHP



Trong PHP, chúng ta sử dụng dấu . để nối chuỗi với chuỗi, hoặc chuỗi với biến.

Ví dụ:

Unset

```
<?php  
echo "Chúng ta sẽ nối 2 đoạn này lại với nhau" . " như  
sau." . PHP_EOL;  
  
$text1 = "Đoạn text 1";  
echo 'Chúng ta có thể nối nó vào như sau: ' . $text1 .  
PHP_EOL;  
echo "Chúng ta có thể nối nó vào như sau: $text1";
```

Kết quả:

Unset

Chúng ta sẽ nối 2 đoạn này lại với nhau như sau.
Chúng ta có thể nối nó vào như sau: Đoạn text 1
Chúng ta có thể nối nó vào như sau: Đoạn text 1

Bài 4: Tìm hiểu về cú pháp cơ bản trong PHP (Phần 2)

Trong bài "[Tìm hiểu về cú pháp cơ bản trong PHP \(Phần 1\)](#)", chúng ta đã cùng nhau tìm hiểu về những cú pháp cơ bản trong PHP. Nhưng, để hiểu và có thể sử dụng PHP một cách tốt và thành thạo những cú cơ bản như thế vẫn là chưa đủ. Hôm nay, chúng ta cùng nhau đến với Phần 2 để tìm hiểu thêm về biến, hằng số, kiểu dữ liệu và toán tử trong PHP.



1. Biến và hằng trong PHP

1.1. Biến trong PHP

Biến là một khối dữ liệu, giá trị của biến có thể thay đổi trong quá trình thực thi chương trình. Biến trong PHP được bắt đầu bằng ký tự \$ và sau đó là tên biến. Các quy tắc đặt tên biến trong PHP: Phải bắt đầu bằng một ký tự hoặc một dấu gạch dưới _. Các ký tự tiếp theo có thể sử dụng chữ cái, số và dấu gạch dưới. Tên biến trong PHP có phân biệt hoa thường.

Ví dụ:

Unset

```
$stringVar = "Đây là một chuỗi trong PHP";
$int = 2023;
$float = 3.14;
```

- Trong quá trình thực thi chúng ta có thể thay đổi giá trị của biến khi cần thiết như sau:

Unset

```
$stringVar = "Đây là một chuỗi đã được thay đổi khi thực
thi chương trình";
$int = 2024;
$float = 4.25;
```

Chúng ta hoàn toàn có thể in giá trị các biến này ra màn hình với echo và print.

Ví dụ:

Unset

```
echo $stringVar;
```



```
// Kết quả: Đây là một chuỗi đã được thay đổi khi thực thi chương trình
```

Lưu ý: Trong PHP, có một số quy ước đặt tên biến mà các lập trình viên thường tuân theo để làm cho mã nguồn dễ đọc, dễ bảo trì và dễ hiểu hơn. Dưới đây là một số quy ước đặt tên biến phổ biến:

- **Sử dụng chữ cái và dấu gạch dưới (underscore):** Tên biến nên bắt đầu bằng một chữ cái và có thể bao gồm chữ cái, chữ số và dấu gạch dưới. Ví dụ: `$userName`, `$total_amount`, `$user_age`.
- **Không sử dụng dấu cách hoặc ký tự đặc biệt:** Tên biến không được chứa dấu cách hoặc ký tự đặc biệt như @, !, &, và #.
- **Sử dụng phong cách lowerCamelCase hoặc snake_case:** Có hai kiểu chính để đặt tên biến là lowerCamelCase (chữ cái đầu tiên viết thường và mỗi từ sau đó được viết hoa chữ cái đầu tiên) hoặc snake_case (tất cả các từ được viết thường và cách nhau bằng dấu gạch dưới).
 - + **lowerCamelCase:** `$userName`, `$totalAmount`, `$userAge`.
 - + **snake_case:** `$user_name`, `$total_amount`, `$user_age`.
- **Sử dụng tên biến có ý nghĩa:** Tên biến nên phản ánh mục đích hoặc nội dung của dữ liệu mà nó đại diện. Điều này giúp mã nguồn dễ hiểu hơn.
- **Tránh viết tắt:** Hãy tránh sử dụng viết tắt trong tên biến, trừ khi viết tắt đó là thông thường và dễ hiểu.
- **Sử dụng tiếng Anh:** Trong phát triển phần mềm, thường được khuyến khích sử dụng tiếng Anh để đặt tên biến, vì đây là ngôn ngữ phổ biến nhất trong cộng đồng lập trình.
- **Tuân thủ quy ước của dự án hoặc nhóm:** Nếu chúng ta đang làm việc trong một dự án hoặc nhóm, hãy tuân thủ quy ước đặt tên được đặt ra bởi dự án hoặc nhóm mà bạn đã tham gia.

Tuân thủ các quy ước này sẽ giúp mã nguồn của bạn dễ đọc, dễ bảo trì và dễ mở rộng trong tương lai.

1.2. Biến toàn cục và biến địa phương



Trong PHP, chúng ta có khái niệm biến toàn cục (Global Variables) và biến địa phương (Local Variables). Với những nhu cầu khác nhau chúng ta có thể chọn những cách khai báo khác nhau. Sau đây chúng ta sẽ cùng nhau tìm hiểu về hai khái niệm trên:

1.2.3. Biến toàn cục (Global Variables) là gì?

Một số đặc điểm của biến toàn cục:

- Biến toàn cục được khai báo bên ngoài của bất kỳ hàm hoặc khối mã cụ thể nào. Chúng có phạm vi sử dụng là toàn bộ chương trình.
- Biến toàn cục thường được sử dụng khi chúng ta cần chia sẻ dữ liệu giữa nhiều phần hoặc hàm của chương trình.
- Để khai báo một biến toàn cục, chúng ta cần sử dụng từ khóa global trước biến hoặc định nghĩa biến bên ngoài bất kỳ hàm nào

Ví dụ:

```
Unset
$globalVar = 10; // Đây là biến toàn cục

function checkUseGlobalVar() {
    echo "Giá trị: " . $globalVar;
}

checkUseGlobalVar();

/*
Warning: Undefined variable $bienToanCuc in
/tmp/Y0d9I8tozz.php on line 7
Giá trị:
*/
```

Trong ví dụ trên chúng ta có thể thấy rằng chúng ta không thể in ra màn hình giá trị của biến \$bienToanCuc vì chúng ta chưa khai báo global cho nó. Ta sẽ viết lại nó như sau:



Unset

```
$globalVar = 10; // Đây là biến toàn cục

function checkUseGlobalVar() {
    global $globalVar;
    echo "Giá trị: " . $globalVar;
}

checkUseGlobalVar();

// Kết quả : Giá trị: 10
```

>>> Xem thêm bài viết tương tự tại đây:

- [Tìm hiểu Enum trong PHP 8.1](#)
- [Tìm hiểu về hàm parse_ini_file trong PHP](#)

1.2.3. Biến địa phương (Local Variables) là gì?

Một số đặc điểm của biến địa phương:

- Biến địa phương được khai báo bên trong hàm hoặc khối mã cụ thể và nó chỉ được truy cập từ bên trong đó.
- Biến địa phương thường được sử dụng tạm thời và giá trị của nó không ảnh hưởng đến các biến toàn cục.

Ví dụ:

Unset

```
function checkUseLocalVar() {
    $localVar = 5; // Đây là biến địa phương
    echo $localVar;
}

checkUseLocalVar();
```



```
// Kết quả: 5
```

1.2. Hằng số trong PHP

Hằng số là một giá trị không đổi trong quá trình thực thi của một chương trình. Khi chúng ta đã định nghĩa hằng số, chúng ta không thể thay đổi giá trị của nó. Để định nghĩa hằng số, chúng ta sử dụng hàm `define()` hoặc sử dụng toán tử `const`.

Ví dụ:

Unset

```
define("TEN_HANG_SO", "Gia tri cua hang so");
const TEN_HANG_SO = "Gia tri cua hang so";
echo TEN_HANG_SO; // In ra "Gia tri cua hang so"
```

2. Kiểu dữ liệu trong PHP

2.1. Dữ liệu kiểu số trong PHP

- Kiểu số trong PHP bao gồm số nguyên (`Integer`) và số thực (`floating-point`).

Ví dụ:

Unset

```
$int = 2023;
$float = 3.14;
```

Chú ý:

- Dữ liệu kiểu `int` có ít nhất 1 ký tự.
- Giá trị dữ liệu kiểu `int` nằm trong khoảng: -2,147,483,648 và 2,147,483,647



Chúng ta cũng có thể chuyển đổi một dữ liệu sang kiểu int hoặc float như sau:

Unset

```
$int = "2023";
/int $int;
var_dump($int);

// Kết quả: int(2023)

$float = "3.14128";
/float $float;
var_dump($float);

// Kết quả: float(3.14128)
```

Ngoài ra chúng ta cũng có thể sử dụng `is_int()` hoặc `is_integer()` để kiểm tra một số có phải kiểu integer hay không. Hoặc `is_float()` để kiểm tra một số có phải là kiểu float hay không.

Ví dụ:

Unset

```
$int = 2023;
var_dump(is_int($int)); // Kết quả: bool(true)

$float = 3.14128;
var_dump(is_float($float)); // Kết quả: bool(true)
```

2.2. Dữ liệu kiểu chuỗi trong PHP

Dữ liệu kiểu chuỗi được định nghĩa bên trong cặp dấu nháy kép hoặc nháy đơn. Và được sử dụng để lưu trữ văn bản.

Ví dụ:



Unset

```
$string = "Đây là một chuỗi cho dữ liệu kiểu chuỗi trong PHP";
```

Tương tự, ta có thể sử dụng `is_string()` để kiểm tra một biến có phải kiểu `string` hay không.

Unset

```
$string = "Đây là một chuỗi cho dữ liệu kiểu chuỗi trong PHP";
var_dump(is_string($string)); // Kết quả: bool(true)
```

2.3. Dữ liệu kiểu mảng trong PHP

Kiểu mảng cho phép chúng ta lưu trữ nhiều giá trị trong một biến duy nhất như sau:

Ví dụ:

Unset

```
$arr = array(1, 2, 3, "bốn");

var_dump($arr);

/*
Kết quả:

array(4) {
    [0]=>
    int(1)
    [1]=>
```



```
int(2)
[2]=>
int(3)
[3]=>
string(5) "bốn"
}
*/
```

Tương tự, ta có thể sử dụng `is_array()` để kiểm tra một biến có phải kiểu array hay không.

Unset

```
$arr = array(1, 2, 3, "bốn");

var_dump(is_array($arr)); // Kết quả: bool(true)
```

- Một số thao tác phổ biến khi làm việc với mảng PHP:
 - + **Thêm phần tử vào mảng:** Sử dụng hàm `array_push()` hoặc `$array[] = $value` để thêm một phần tử vào cuối mảng.

Unset

```
$fruits = array("apple", "banana");
array_push($fruits, "orange");
// or
$fruits[] = "grape";
```

- + **Xóa phần tử khỏi mảng:** Sử dụng hàm `unset()` hoặc `array_splice()` để xóa một hoặc nhiều phần tử khỏi mảng.

Unset

```
$fruits = array("apple", "banana", "orange");
```



```
unset($fruits[1]); // Xóa phần tử có chỉ số 1 ("banana")
// or
array_splice($fruits, 1, 1); // Cũng xóa "banana" từ mảng
```

- + **Lấy độ dài của mảng:** Sử dụng hàm `count()` để đếm số lượng phần tử trong mảng.

Unset

```
$fruits = array("apple", "banana", "orange");
$length = count($fruits); // $length sẽ là 3
```

- + **Truy cập phần tử trong mảng:** Sử dụng chỉ số của phần tử để truy cập hoặc thay đổi giá trị của nó.

Unset

```
$fruits = array("apple", "banana", "orange");
echo $fruits[0]; // Output: apple
$fruits[1] = "grape"; // Thay đổi giá trị của phần tử có chỉ số 1
```

- + **Duyệt qua mảng:** Sử dụng vòng lặp `for`, `foreach`, hoặc `while` để duyệt qua từng phần tử trong mảng.

Unset

```
$fruits = array("apple", "banana", "orange");
foreach ($fruits as $fruit) {
    echo $fruit . ", ";
}
// Output: apple, banana, orange,
```



- + **Sắp xếp mảng:** Sử dụng hàm `sort()` hoặc `asort()` để sắp xếp mảng theo thứ tự tăng dần hoặc `rsort()` hoặc `arsort()` để sắp xếp mảng theo thứ tự giảm dần.

Unset

```
$numbers = array(4, 2, 8, 1);
sort($numbers); // Sắp xếp theo thứ tự tăng dần
// Hoặc
rsort($numbers); // Sắp xếp theo thứ tự giảm dần
```

- Các loại mảng phổ biến trong PHP:

- + **Mảng Số Học (Numeric Arrays):** Mảng số học là mảng mà các phần tử được truy cập thông qua chỉ số số học (số nguyên không âm).

Unset

```
$numericArray = array("apple", "banana", "orange");
```

- + **Mảng Liên Kết (Associative Arrays):** Mảng liên kết là mảng mà các phần tử được truy cập thông qua các khóa được đặt tên.

Unset

```
$assocArray = array("name" => "John", "age" => 30, "city"
=> "New York");
```

- + **Mảng Đa Chiều (Multidimensional Arrays):** Mảng đa chiều là mảng mà mỗi phần tử có thể là một mảng khác.

Unset

```
$multiArray = array(
    array("apple", "banana", "orange"),
    array("car", "bike", "bus"),
    array("red", "blue", "green")
);
```



- + **Mảng Liên Kết Multilevel (Multilevel Associative Arrays):** Đây là một dạng đặc biệt của mảng liên kết, trong đó các phần tử có thể là mảng liên kết hoặc mảng số học.

Unset

```
$multilevelArray = array(  
    "person1" => array("name" => "John", "age" => 30),  
    "person2" => array("name" => "Alice", "age" => 25)  
) ;
```

- + **Mảng Thưa (Sparse Arrays):** Mảng thưa là mảng mà các chỉ số không cần phải là một chuỗi số liên tục.

Unset

```
$sparseArray[1] = "apple";  
$sparseArray[3] = "banana";  
$sparseArray[10] = "orange";
```

>>> Xem thêm các bài viết tương tự tại đây:

- [Bài 1: Tìm hiểu về ngôn ngữ lập trình PHP](#)
- [Bài 2: Hướng dẫn sử dụng XAMPP: Tao môi trường Lập trình PHP trên máy tính của bạn](#)
- [Bài 3: Tìm hiểu về cú pháp cơ bản trong PHP \(Phần 1\)](#)
- [Bài 5: Câu lệnh điều kiện trong PHP](#)

2.4. Dữ liệu kiểu object trong PHP

Kiểu đối tượng cho phép chúng ta định nghĩa và sử dụng các lớp và đối tượng trong PHP

Ví dụ:

Unset

```
class ObjVar {  
    public $message;
```



```
    public $year;  
}  
  
$sv = new ObjVar();  
$sv->message = "Đây là một tin nhắn!";  
$sv->year = 2023;
```

2.5. Dữ liệu kiểu boolean trong PHP

Kiểu boolean có hai giá trị: true (đúng) hoặc false (sai).

Ta có thể kiểm tra dữ liệu có phải kiểu boolean không như sau:

Unset

```
$arr = array(1, 2, 3, "bốn");  
  
var_dump(is_bool($arr));
```

Tìm hiểu thêm: Boolean trong Javascript

2.6. Dữ liệu kiểu null trong PHP

Kiểu dữ liệu null là biểu thị rằng biến không có giá trị hoặc giá trị của biến bị thiếu.

Ví dụ:

Unset

```
$var2 = null;
```

3. Toán tử trong PHP



3.1. Toán tử số học

Toán tử	Mô tả
<code>+</code>	Cộng
<code>-</code>	Trừ
<code>*</code>	Nhân
<code>/</code>	Chia
<code>%</code>	Chia lấy phần dư

3.2. Toán tử gán

Toán tử	Mô tả
<code>=</code>	Gán giá trị
<code>+=</code>	Cộng và gán
<code>-=</code>	Trừ và gán
<code>*=</code>	Nhân và gán
<code>/=</code>	Chia và gán
<code>%=</code>	Chia lấy phần dư và gán

3.3. Toán tử so sánh

Toán tử	Mô tả
<code>==</code>	So sánh giá trị
<code>==></code>	So sánh giá trị và kiểu dữ liệu
<code>!=</code> hoặc <code><></code>	Không bằng
<code>! ==</code>	Không bằng về giá trị hoặc kiểu dữ liệu
<code><</code>	Nhỏ hơn
<code>></code>	Lớn hơn
<code><=</code>	Nhỏ hơn hoặc bằng



>=

Lớn hơn hoặc bằng

Trong PHP, `==` và `===` được sử dụng để so sánh giá trị của hai biến hoặc biểu thức, trong khi `!=` và `!==` được sử dụng để kiểm tra xem chúng có bằng nhau hay không. Dưới đây là sự khác biệt giữa chúng:

- `==` (Equal Operator): Để kiểm tra xem hai giá trị có bằng nhau không, không quan tâm đến kiểu dữ liệu.

```
Unset
$x = 10;
$y = "10";
if ($x == $y) {
    echo "Equal";
} else {
    echo "Not Equal";
}
// Kết quả: Equal
```

Ở đây, mặc dù `$x` là một số nguyên và `$y` là một chuỗi, nhưng vì giá trị của chúng bằng nhau, điều kiện `if` trả về true.

- `===` (Identical Operator): Để kiểm tra cả giá trị và kiểu dữ liệu của hai biến.

```
Unset
$x = 10;
$y = "10";
if ($x === $y) {
    echo "Identical";
} else {
    echo "Not Identical";
}
// Kết quả: Not Identical
```

Ở đây, điều kiện `if` trả về false vì dù giá trị của `$x` và `$y` giống nhau, nhưng kiểu dữ liệu của chúng khác nhau.



- **`!=`** (Not Equal Operator): Để kiểm tra xem hai giá trị có khác nhau không, không quan tâm đến kiểu dữ liệu.

```
Unset
$x = 10;
$y = "20";
if ($x != $y) {
    echo "Not Equal";
} else {
    echo "Equal";
}
// Kết quả: Not Equal
```

Ở đây, điều kiện `if` trả về `true` vì `$x` và `$y` có giá trị khác nhau.

- **`!==`** (Not Identical Operator): Để kiểm tra cả giá trị và kiểu dữ liệu của hai biến.

```
Unset
$x = 10;
$y = "10";
if ($x !== $y) {
    echo "Not Identical";
} else {
    echo "Identical";
}
// Kết quả: Not Identical
```

Ở đây, điều kiện `if` trả về `true` vì dù giá trị của `$x` và `$y` giống nhau, nhưng kiểu dữ liệu của chúng khác nhau.

3.4. Toán tử logic

Toán tử	Mô tả
<code>&&</code> hoặc <code>and</code>	AND logic
<code> </code> hoặc <code>or</code>	OR logic



<code>! hoặc not</code>	NOT logic
<code>XOR</code>	XOR logic

3.5. Toán tử chuỗi

Toán tử	Mô tả
<code>.</code>	Ghép chuỗi
<code>.=</code>	Ghép chuỗi và gán

3.6. Toán tử khác

Toán tử	Mô tả
<code>? :</code>	Toán tử ba ngôi (Conditional) - rút gọn của if-else
<code>.</code>	Toán tử chuỗi
<code>[]</code>	Toán tử truy cập mảng
<code>-></code>	Toán tử truy cập đối tượng
<code>++</code>	Tăng giá trị lên 1
<code>--</code>	Giảm giá trị xuống 1

3.7. Toán tử bitwise (bit)

Toán tử	Mô tả
<code>&</code>	AND bitwise
<code> </code>	OR bitwise
<code>^</code>	XOR bitwise
<code>~</code>	NOT bitwise
<code><<</code>	Dịch trái
<code>>></code>	Dịch phải



3.8. Toán tử xác định kiểu (Type Casting)

Toán tử	Mô tả
(int)	Ép kiểu thành số nguyên
(float)	Ép kiểu thành số thực
(string)	Ép kiểu thành chuỗi
(array)	Ép kiểu thành mảng
(object)	Ép kiểu thành đối tượng
(bool)	Ép kiểu thành boolean

Bài 5: Câu lệnh điều kiện và vòng lặp trong PHP

Trong [bài 4](#) chúng ta đã cùng nhau tìm hiểu về các cú pháp cơ bản của PHP. Thông qua đó, chúng ta đã có thể nắm vững được những gì cơ bản nhất của PHP. Để tiếp tục series về PHP này, chúng ta sẽ cùng nhau đi sâu hơn vac chi tiết hơn về: Câu lệnh điều kiện và vòng lặp trong PHP.

1. Câu lệnh điều kiện trong PHP

1.1. Câu lệnh if trong PHP

- Cú pháp:

Unset

```
if (condition) {
    // Khối mã sẽ được thực thi khi thỏa mãn điều kiện
}
```

- Sử dụng: Dùng để kiểm tra một điều kiện cụ thể có thỏa mãn hay không.

VD: Kiểm tra độ tuổi người dùng có đủ điều kiện để truy cập trang web hay không.



Unset

```
if ($age > 18) {  
    // Content this page  
}
```

1.2. Câu lệnh else trong PHP

- Cú pháp:

Unset

```
if (condition) {  
    // Khối mã được thực hiện nếu điều kiện là đúng  
} else {  
    // Khối mã được thực hiện nếu điều kiện là sai  
}
```

- Sử dụng: thường được sử dụng với câu lệnh if để xác định một hành động thay thế khi điều kiện if không đúng.

Ví dụ: Hiển thị ra thông báo khi người dùng không thỏa mãn điều kiện về tuổi

Unset

```
if ($age > 18) {  
    echo 'Tuổi của bạn thỏa mãn';  
} else {  
    echo 'Tuổi của bạn không thỏa mãn';  
}
```

>>> Xem thêm các bài viết tương tự tại đây:



- [Bài 2: Hướng dẫn sử dụng XAMPP: Tạo môi trường Lập trình PHP trên máy tính của bạn](#)
- [Bài 3: Tìm hiểu về cú pháp cơ bản trong PHP \(Phần 1\)](#)
- [Bài 4: Tìm hiểu về cú pháp cơ bản trong PHP \(Phần 2\)](#)

1.3. Câu lệnh else if theo PHP

- Cú pháp: Về mặt cú pháp thông thường với các ngôn ngữ điều kiện sẽ chấp nhận hai cách viết elseif hoặc else if trong bài viết này chúng ta sẽ sử dụng cách viết else if là cách được sử dụng thông dụng và thường gặp nhất để có thể dễ dàng quan sát về sau này.

Unset

```
if (condition_1) {  
    // Khối mã được thực hiện nếu điều kiện 1 là đúng  
} else if (condition_2) {  
    // Khối mã được thực hiện nếu điều kiện 2 là đúng  
} else {  
    // Khối mã được thực hiện nếu không có điều kiện nào  
    // đúng  
}
```

- Sử dụng: Kiểm tra các điều kiện liên tục để xem điều kiện nào sẽ đúng và thực hiện khối lệnh.

Ví dụ: Kiểm tra xếp loại sinh viên

Unset

```
if ($point > 8) {  
    echo 'Giỏi';  
} else if (5 < $point <= 8) {  
    echo 'Khá';  
} else {  
    echo 'Trung bình';  
}
```



1.4. Câu lệnh switch trong PHP

- Cú pháp:

Unset

```
switch (condition) {  
    case value_1:  
        // Khối mã được thực hiện nếu biểu thức trùng với  
        value_1  
        break;  
    case value_2:  
        // Khối mã được thực hiện nếu biểu thức trùng với  
        value_2  
        break;  
    // ...  
    default:  
        // Khối mã được thực hiện nếu không có trường hợp  
        nào trùng khớp  
}
```

- Sử dụng: Câu lệnh switch tương tự như else if, để kiểm tra một giá trị nào đó có thỏa mãn hay không và thực hiện hành động cụ thể.

Ví dụ:

Unset

```
$color = "blue";  
  
switch ($color) {  
    case "red":  
        echo "Màu đỏ";  
        break;  
    case "blue":  
        echo "Màu xanh";  
        break;
```



```
case "green":  
    echo "Màu xanh lá cây";  
    break;  
default:  
    echo "Màu không xác định";  
}
```

Trong `switch ... case` chúng ta có thể sử dụng `break` để kết thúc việc thực thi của một khối lệnh. Ta cùng xem xét ví dụ sau đây:

```
Unset  
$day = "Monday";  
  
switch ($day) {  
    case "Monday":  
        echo "Today is Monday.";  
        // Không có break ở đây  
    case "Tuesday":  
        echo "Today is Tuesday.";  
        // Không có break ở đây  
    case "Wednesday":  
        echo "Today is Wednesday.";  
        // Không có break ở đây  
    default:  
        echo "Today is not Monday, Tuesday, or  
Wednesday.";  
}
```

Trong trường hợp này, kết quả sẽ là: "Today is Monday.Today is Tuesday.Today is Wednesday." Và chương trình sẽ tiếp tục thực thi lệnh sau cấu trúc `switch` vì không có `break` nào được sử dụng để kết thúc các `case`.

Lưu ý: So sánh điều kiện `switch ... case` trong PHP được sử dụng là == không quan tâm đến kiểu dữ liệu. Bạn có thể thử ví dụ dưới đây để có thể hiểu rõ hơn về cách `switch ... case` so sánh điều kiện:



```
Unset
$day = "1";

switch ($day) {
    case 1:
        echo "Today is Monday.";
        break;
    case "1":
        echo "Today is Tuesday.";
        break;
    default:
        echo "Today is not Monday or Tuesday.";
}
```

2. Vòng lặp trong PHP

2.1. Vòng lặp for trong PHP

- Cú pháp:

```
Unset
for (init counter; test counter; increment counter) {
    // Khối mã được lặp lại cho đến khi điều kiện là sai
}
```

- Sử dụng: Thường được sử dụng khi chúng ta có thể xác định được số lần lặp.

Ví dụ:

```
Unset
for ($i = 0; $i < 5; $i++) {
    echo 'Số lần lặp lại' . $i;
}
```



Trong vòng lặp for chúng ta có thể sử dụng **break** để kết thúc vòng lặp ngay lập tức, chúng ta cùng xem ví dụ để có thể dừng vòng lặp ngay khi giá trị **\$i** bằng 5:

```
Unset
for ($i = 1; $i <= 10; $i++) {
    echo $i . " ";
    if ($i == 5) {
        break;
    }
}
```

2.2. Vòng lặp while trong PHP

- Cú pháp:

```
Unset
while (condition) {
    // Khối mã được lặp lại cho đến khi điều kiện là sai
}
```

- Sử dụng: Thường thì với vòng lặp while người dùng sẽ không biết trước được số lần lặp mà vòng lặp while sẽ thực hiện, với vòng lặp while thì chúng ta chỉ quan tâm khi điều kiện lặp vẫn đúng thì vòng lặp vẫn sẽ được thực hiện.

Ví dụ:

```
Unset
$i=0;
while ($i <= 5) {
```



```
    echo 'Số lần lặp lại' . $i . PHP_EOL;
    $i++;
}
```

2.3. Vòng lặp foreach trong PHP

- Cú pháp:

Unset

```
foreach ($arr as $element) {
    // Khối mã được thực hiện cho mỗi phần tử trong mảng
}
```

- Sử dụng: Để lặp qua một mảng danh sách các phần tử có sẵn.

Ví dụ:

Unset

```
$arr = [5, 10, 20, 30, 15];

foreach ($arr as $item) {
    echo 'Phần tử: ' . $item;
}
```

Bài 6: Xử lý biểu mẫu trong PHP

Trong [Bài 5: Câu lệnh điều kiện và vòng lặp trong PHP](#) chúng ta đã cùng nhau tìm hiểu về câu lệnh điều kiện và vòng lặp trong PHP bao gồm cú pháp và cách sử dụng chúng. Trong bài này chúng ta sẽ tiếp tục với công việc xử lý form (biểu mẫu) trong PHP.



1. Tạo biểu mẫu HTML

Để bắt đầu chúng ta sẽ cùng nhau nhắc lại một số mục chính về form trong HTML. Các thuộc tính chính:

- method: Có 2 phương thức chính là GET và POST
- action: Là đường dẫn sẽ xử lý khi người dùng thực hiện submit form HTML.

Ví dụ:

- Phương thức post:

Unset

```
<!DOCTYPE html>
<html>
<head>
    <title>Stringee demo</title>
</head>
<body>
    <form method="post" action="process.php">
        Name: <input type="text" name="name"><br>
        Email: <input type="text" name="email"><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

- Phương thức get:

Unset

```
<!DOCTYPE html>
<html>
<head>
    <title>Stringee demo</title>
</head>
```



```
<body>
    <form method="get" action="process.php">
        Name: <input type="text" name="name"><br>
        Email: <input type="text" name="email"><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Đây là hai loại biểu mẫu mà chúng ta sẽ hay gặp nhất khi thao tác với form trong PHP. Sau đây chúng ta sẽ tìm cách để xử lý đầu vào từ hai form này.

2. Xử lý biểu mẫu với PHP

Bây giờ chúng ta sẽ tạo một file có tên là `action.php` nằm cùng với thư mục của biểu mẫu phía trên.

Unset

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

    // Thực hiện xử lý dữ liệu ở đây
    echo "Xin chào, $name! Email của bạn là: $email";
} elseif ($_SERVER[ "REQUEST_METHOD" ] == "GET") {
    $name = $_GET[ "name" ];
    $email = $_GET[ "email" ];

    // Thực hiện xử lý dữ liệu ở đây
    echo "Xin chào, $name! Email của bạn là: $email";
}
?>
```



Chú ý: Ở đây ta thấy rằng cách thực hiện với phương thức GET và POST không có quá nhiều sự khác biệt.

Chúng ta hãy cùng nhau nhìn vào sự khác biệt URL khi sử dụng phương thức POST và GET:

- GET:

<http://localhost/action.php?name=Stringee&email=info@stringee.com>

- POST: <http://localhost/action.php>

Chúng ta có thể nhận thấy rằng phương thức GET kém bảo mật hơn. Cho nên thông thường chúng ta sử dụng phương thức GET cho những biểu mẫu chứa thông tin không quá quan trọng.

3. Hiển thị kết quả sau khi được xử lý

Sau khi đã được xử lý, chúng ta cần hiển thị dữ liệu cho người dùng có thể quan sát được. Chúng ta đã sử dụng hàm echo để hiển thị tên và email mà người dùng đã nhập. Hoặc ta có thể xử lý dữ liệu bằng cách hiển thị nó với một file *.html.php và lưu nó vào cơ sở dữ liệu.

Bài 7: Hàm trong PHP: Cách tạo và sử dụng hàm (Phần 1)

Hàm là một khái niệm vô cùng quan trọng trong tất cả các ngôn ngữ lập trình, trong đó có PHP. Trong bài viết này, chúng ta sẽ khám phá cách định nghĩa, cách sử dụng và áp dụng của hàm vào chương trình viết bằng PHP.

1. Hàm trong PHP là gì?

Hàm trong PHP cho phép chúng ta chia mã của mình thành các đoạn mã nhỏ, có thể tái sử dụng, để thực hiện các tác vụ cụ thể. Hàm giúp cho mã code của chúng ta trở nên rõ ràng, dễ đọc và dễ bảo trì bằng cách chia nó thành các phần nhỏ có mục tiêu cụ thể.

Mỗi hàm có một tên duy nhất và có thể nhận đối số, tham số đầu vào, thực hiện các tính toán và tác vụ, sau đó trả về một giá trị hoặc không trả về giá trị tùy theo



yêu cầu. Sử dụng hàm, bạn có thể thực hiện các công việc phức tạp, tính toán, xử lý dữ liệu và nhiều tác vụ khác.

>>> Xem thêm bài viết về PHP:

- [Tìm hiểu về hàm preg_match_all trong PHP](#)
- [Xử lý hàm date \(ngày tháng\) trong PHP](#)
- [Giới hạn ký tự hiển thi và chức năng read more trong PHP](#)

2. Định nghĩa hàm (function) trong PHP

Hàm là một khái niệm quan trọng trong lập trình PHP. Nó cho phép bạn tạo một khối mã có thể tái sử dụng và thực hiện các tác vụ cụ thể. Để định nghĩa một hàm trong PHP, bạn sử dụng từ khóa function, sau đó là tên hàm và danh sách đối số (nếu có). Cấu trúc cơ bản như sau:

Unset

```
function ten_ham($doi_so1, $doi_so2, ...) {  
    // Mã code của hàm  
}
```

3. Hàm trả về kết quả

Hàm trong PHP có thể trả về kết quả bằng cách sử dụng từ khóa return. Bất kỳ khi nào hàm gặp lệnh return, nó sẽ kết thúc và trả về giá trị đó. Ví dụ:

Unset

```
function tinh_tong($so1, $so2) {  
    $tong = $so1 + $so2;  
    return $tong;  
}
```

4. Sử dụng hàm trong PHP



Để sử dụng một hàm trong PHP, bạn gọi tên hàm bằng cách sử dụng dấu ngoặc tròn và truyền giá trị cho các đối số (nếu cần). Kết quả trả về từ hàm có thể được gán vào biến hoặc sử dụng trực tiếp. Ví dụ:

Unset

```
$ket_qua = tinh_tong(5, 3); // $ket_qua sẽ chứa giá trị 8
```

5. Một số trường hợp khai báo hàm đặc biệt trong PHP

5.1. Khai báo hàm trong lệnh

Trong PHP, bạn có thể khai báo hàm bên trong một lệnh. Điều này cho phép bạn tạo hàm tạm thời và sử dụng chúng một cách nhanh chóng.

5.2. Khai báo hàm trong hàm

Unset

```
$ham_tam_thoi = function($doi_so) {
    return "Chào, $doi_so!";
};

echo $ham_tam_thoi("John"); // In ra "Chào, John!"
```

Trong PHP, bạn cũng có thể khai báo hàm bên trong một hàm khác. Điều này cho phép bạn sử dụng hàm bên trong một phạm vi hẹp hơn.

Unset

```
function ham_ngoai() {
    function ham_trong() {
        return "Hàm bên trong!";
    }
}
```



```
        return ham_trong();
    }

echo ham_ngoai(); // In ra "Hàm bên trong!"
```

6. Một số trường hợp sử dụng hàm đặc biệt trong PHP

6.1. Hàm tự gọi chính nó – đệ quy trong PHP

Một hàm có thể gọi chính nó bằng cách sử dụng kỹ thuật đệ quy. Điều này thường được sử dụng để giải quyết các vấn đề có cấu trúc lặp lại.

```
Unset

function tinh_giai_thua($n) {
    if ($n <= 1) {
        return 1;
    } else {
        return $n * tinh_giai_thua($n - 1);
    }
}

echo tinh_giai_thua(5); // In ra "120"
```

6.2. Gọi hàm động

Bạn có thể gọi một hàm bằng tên được lưu trong một biến. Điều này cho phép bạn thay đổi hàm mục tiêu tùy theo tình huống.



Unset

```
$ham_muon_goi = "tinh_tong";  
$ket_qua = $ham_muon_goi(5, 3); // $ket_qua sẽ chứa giá  
trị 8
```

Bài 8: Hàm trong PHP: Cách tạo và sử dụng hàm (Phần 2)

Trong phần trước, chúng ta đã tìm hiểu các khái niệm cơ bản về hàm trong PHP. Bài viết này sẽ tìm hiểu với những khái niệm và kỹ thuật nâng cao về hàm, cũng như cách ứng dụng chúng vào thực tế.

1. Hàm và phạm vi tác dụng của biến

1.1. Phạm vi tác dụng của biến

Phạm vi tác dụng của biến là phạm vi mà biến có thể được truy cập và sử dụng. Có hai loại phạm vi chính trong PHP: phạm vi cục bộ (local scope) và phạm vi toàn cục (global scope).

1.2. Biến cục bộ và biến toàn cục biến cục bộ

Là biến được khai báo bên trong một hàm và chỉ có thể được sử dụng trong phạm vi của hàm đó.

Unset

```
function exampleFunction() {  
    $localVariable = "This is a local variable.";  
    echo $localVariable;  
}
```

Biến toàn cục: Là biến được khai báo ở phạm vi toàn cục và có thể được sử dụng ở mọi nơi trong chương trình.



Unset

```
$globalVariable = "This is a global variable.";

function exampleFunction() {
    global $globalVariable;
    echo $globalVariable;
}
```

1.3. Sử dụng biến toàn cục trong hàm

Từ khóa global để sử dụng biến toàn cục trong hàm, bạn cần sử dụng từ khóa global.

Unset

```
$globalVariable = "This is a global variable.";

function exampleFunction() {

    global $globalVariable;

    echo $globalVariable;

}
```

2. Tham biến và Tham trị

Trong PHP, tham số khi truyền vào hàm có thể là tham trị hoặc tham biến.

Tham trị: Trong trường hợp này, giá trị của biến được sao chép vào tham số của hàm.



Unset

```
function addTen($num) {  
    $num += 10;  
    echo $num;  
}  
  
$number = 5;  
addTen($number); // In ra "15"  
echo $number; // Vẫn là "5"
```

Tham biến: Trong trường hợp này, hàm nhận vào biến và thay đổi trực tiếp giá trị của biến.

Unset

```
function addTen(&$num) {  
    $num += 10;  
    echo $num;  
}  
  
$number = 5;  
addTen($number); // In ra "15"  
echo $number; // Giờ là "15"
```

3. Tham số mặc định

Trong PHP, bạn có thể đặt giá trị mặc định cho tham số của hàm.

Unset

```
function greet($name = "Guest") {  
    echo "Hello, $name!";  
}
```



```
greet(); // In ra "Hello, Guest!"  
greet("John"); // In ra "Hello, John!"
```

4. Hàm với lượng tham số tùy ý

Bạn có thể định nghĩa hàm với lượng tham số không cố định bằng cách sử dụng dấu ba chấm (...). Các tham số này sẽ được đóng gói vào một mảng.

Unset

```
function sum(...$numbers) {  
    $result = 0;  
    foreach ($numbers as $num) {  
        $result += $num;  
    }  
    echo $result;  
}  
  
sum(1, 2, 3, 4); // In ra "10"
```

5. Chỉ dẫn kiểu (Type Hint)

Chỉ dẫn kiểu giúp bạn xác định kiểu dữ liệu mà một tham số hoặc giá trị trả về của hàm phải có.

Unset

```
function add(int $a, int $b): int {  
    return $a + $b;  
}  
  
$result = add(5, 3); // Đúng  
$result = add("5", "3"); // Sai, vì kiểu dữ liệu không  
đúng
```



6. Một số ví dụ về hàm trong PHP

6.1. Hàm vô danh (Anonymous Functions)

Hàm ẩn danh (Anonymous Function) trong PHP (stringee.com) Hàm vô danh, hay còn gọi là closure, là một hàm không có tên. Bạn có thể gán hàm này cho một biến và sử dụng nó như một đối số hoặc giá trị trả về. Đây là một ví dụ:

Unset

```
$add = function($a, $b) {
    return $a + $b;
};

$result = $add(3, 5); // $result sẽ chứa giá trị 8
```

6.2. Hàm biến (Variable Functions)

Trong PHP, bạn có thể sử dụng một biến để chứa tên của hàm và gọi nó thông qua biến đó. Điều này thường được sử dụng khi bạn muốn tùy chọn hàm cần gọi dựa trên điều kiện nào đó. Ví dụ:

Unset

```
function greetEnglish($name) {
    echo "Hello, $name!";
}

function greetFrench($name) {
    echo "Bonjour, $name!";
}

$language = "English";
$greet = "greet" . $language;

$greet("John"); // Tương đương với greetEnglish("John");
```



6.3. Hàm đệ quy

Hàm đệ quy là hàm có khả năng gọi chính nó. Điều này thường được sử dụng để giải quyết các vấn đề được mô tả dưới dạng đệ quy. Dưới đây là một ví dụ tính giai thừa:

```
Unset

function factorial($n) {
    if ($n <= 1) {
        return 1;
    } else {
        return $n * factorial($n - 1);
    }
}

echo factorial(5); // In ra "120"
```

6.4. Hàm trả về đa giá trị

Trong PHP, một hàm có thể trả về nhiều giá trị thông qua các cú pháp đặc biệt như list() hoặc sử dụng mảng kết hợp. Điều này cho phép bạn trả về nhiều kết quả từ một hàm duy nhất.

```
Unset

function getCoordinates() {
    return ['latitude' => 40.7128, 'longitude' =>
-74.0060];
}
$coordinates = getCoordinates();
echo "Latitude: " . $coordinates['latitude'] . ",";
echo "Longitude: " . $coordinates['longitude'];
```

6.5. Sử dụng Closure



Với use Closure (hàm vô danh) có thể sử dụng biến bên ngoài phạm vi của nó thông qua từ khóa use. Điều này cho phép bạn truy cập và thay đổi các biến bên ngoài trong hàm.

Unset

```
$message = "Hello";  
  
$greeting = function() use ($message) {  
    echo $message;  
};  
  
$message = "Bonjour";  
$greeting(); // In ra "Hello", vì giá trị của $message đã  
được capture khi khai báo closure
```

6.6. Hàm __invoke trong PHP

Nếu một đối tượng được khai báo để triển khai __invoke(), thì đối tượng đó có thể được gọi như một hàm.

Unset

```
class Example {  
    public function __invoke($x) {  
        return $x * $x;  
    }  
}  
  
$example = new Example();  
echo $example(5); // In ra "25"
```

6.7. Hàm Generator

Generator là một cách linh hoạt để tạo ra dữ liệu mà không cần phải lưu trữ toàn bộ trong bộ nhớ. Cú pháp yield được sử dụng để tạo ra các giá trị từ một hàm generator.



Unset

```
function generateNumbers($start, $end) {
    for ($i = $start; $i <= $end; $i++) {
        yield $i;
    }
}

foreach (generateNumbers(1, 5) as $number) {
    echo $number . ", "; // In ra "1, 2, 3, 4, 5, "
}
```

Bài 9: Kết nối cơ sở dữ liệu với PHP

Trong các ứng dụng hiện nay, cơ sở dữ liệu (CSDL) luôn đóng một vai trò rất quan trọng trong việc lưu trữ và quản lý dữ liệu. Việc cấu hình và tương tác với các CSDL một cách hợp lý sẽ mang lại một trải nghiệm tốt hơn cho khách hàng. Trong bài viết này, chúng ta sẽ cùng nhau tìm hiểu về CSDL trong PHP, cách tương tác với nó trong PHP, một ngôn ngữ lập trình phổ biến trong phát triển web.

1. Tại sao lại sử dụng MySQL trong PHP

Hiện nay, rất nhiều lập trình viên vẫn đang tin dùng CSDL MySQL. Một phần vì sự thuận tiện, đồng thời cũng là sự thân thiện dễ tiếp cận mà nó mang lại cho các lập trình viên. Sau đây, chúng ta sẽ cùng nhau tìm hiểu một số lý do mà chúng ta nên sử dụng MySQL trong PHP:

- Miễn phí và mã nguồn mở: Vì là một hệ quản trị cơ sở dữ liệu mã nguồn mở, chúng ta hoàn toàn có thể sử dụng MySQL mà không cần phải trả bất kỳ chi phí nào. Điều này là một trong những lý do quan trọng để nó trở thành lựa chọn hàng đầu đối với các dự án có ngân sách hạn chế.
- Phổ biến và ổn định: Là một trong những CSDL ra đời và được tin dùng từ lâu. Việc tìm hiểu tài liệu về MySQL rất dễ dàng, đi kèm với đó là một cộng đồng sử dụng lớn và hỗ trợ kỹ thuật đáng tin cậy. Chúng khiến cho MySQL ngày càng được tin tưởng và khó thay thế.



- Tương thích với nhiều ngôn ngữ lập trình: Với việc có khả năng tương thích với nhiều ngôn ngữ lập trình ngoài PHP, nó mang lại sự linh hoạt khi cần phát triển các hệ thống hay ứng dụng đa nền tảng.
- Hiệu suất cao: Đi kèm với sự thuận tiện, MySQL cũng được thiết kế để đạt được hiệu suất cao và có khả năng xử lý hàng nghìn truy vấn mỗi giây. Điều này giúp các lập trình viên có thể xử lý các dữ liệu lớn và phức tạp một cách dễ dàng hơn.
- Hỗ trợ ACID: MySQL hỗ trợ các tính chất ACID (Atomicity, Consistency, Isolation, Durability), giúp đảm bảo tính nhất quán và an toàn của dữ liệu trong quá trình thực hiện các giao dịch.
- Cộng đồng lớn và hỗ trợ: Với sự tin tưởng lớn của rất nhiều lập trình viên, đồng thời nó mang lại cho MySQL các cộng đồng với các diễn đàn, tài liệu và nguồn thông tin trực tuyến để các lập trình viên có thể giải quyết vấn đề và nắm vững việc sử dụng MySQL một cách dễ dàng hơn.

Phía trên là một số những lý do mà chúng ta nên tìm hiểu, sử dụng MySQL như một CSDL chính trong ứng dụng của chúng ta. Ngoài những gì cơ bản mà CSDL mang lại thì MySQL mang lại một sự tin tưởng và an toàn nhờ cộng đồng hỗ trợ lớn mà nó mang lại. Tiếp theo, chúng ta sẽ cùng nhau tìm hiểu cách kết nối với MySQL trong PHP.

2. Kết nối MySQL trong PHP

Trong PHP có rất nhiều cách để chúng ta có thể kết nối đến CSDL MySQL với PHP, chúng ta có một số cách phổ biến, an toàn và thường được sử dụng là dùng extension `mysqli` (MySQL Improved) hoặc `PDO` (PHP Data Objects). Sau đây ta sẽ tìm hiểu cách sử dụng chúng để có thể kết nối với CSDL MySQL trong PHP.

2.1. Sử dụng MySQLi (MySQL Improved) trong PHP

Unset

```
<?php  
$server = "localhost"; // IP hoặc domain kết nối để CSDL  
$username = "username"; // Người dùng sử dụng để kết nối  
đến CSDL với PHP
```



```
$password = "password"; // Mật khẩu sử dụng để kết nối  
đến CSDL với PHP  
$database = "database"; // Tên database  
  
// Kết nối đến MySQL với PHP sử dụng MySQLi  
$conn = new mysqli($server, $username, $password,  
$database);  
  
// Kiểm tra kết nối  
if ($conn->connect_error) {  
    die("Kết nối thất bại: " . $conn->connect_error);  
}  
  
echo "Kết nối thành công";  
  
// Thực hiện các truy vấn và công việc khác ở đây  
  
// Đóng kết nối  
$conn->close();
```

2.2. Sử dụng PDO (PHP Data Objects) trong PHP

Unset

```
<?php  
$server = "localhost"; // IP hoặc domain kết nối đến CSDL  
$username = "username"; // Người dùng sử dụng để kết nối  
đến CSDL với PHP  
$password = "password"; // Mật khẩu sử dụng để kết nối  
đến CSDL với PHP  
$database = "database"; // Tên database  
  
try {  
    // Kết nối đến MySQL với PHP sử dụng PDO
```



```

$conn = new
PDO("mysql:host=$server;dbname=$database", $username,
$password);

// Thiết lập chế độ lỗi và ngoại lệ
$conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
echo "Kết nối thành công";

// Thực hiện các truy vấn và công việc khác ở đây

// Đóng kết nối
$conn = null;
} catch (PDOException $e) {
    echo "Kết nối thất bại: " . $e->getMessage();
}

```

2.3. Sự khác biệt chính giữa MySQLi và PDO

	MySQLi	PDO
Loại CSDL hỗ trợ	Chủ yếu được thiết kế để hoạt động với CSDL MySQL	Hoạt động với 12 hệ thống CSDL khác nhau (MySQL, PostgreSQL, SQLite, MSSQL, ...)
Cú pháp	Sử dụng cú pháp hướng đối tượng (Object-Oriented) và cú pháp dựa trên thủ tục (Procedural)	Sử dụng cú pháp hướng đối tượng
Tính năng	- Hỗ trợ prepared statements và transactions	-Hỗ trợ prepared statements và transactions



	<ul style="list-style-type: none"> - Cung cấp nhiều tính năng mạnh mẽ như multiple statements trong một lần thực thi 	<ul style="list-style-type: none"> - Đồng nhất với một API cho nhiều loại cơ sở dữ liệu
Bảo mật	Cả hai đều hỗ trợ prepared statements, giúp ngăn chặn tấn công như SQL injection. Điều này làm cho cả hai extension đều có khả năng bảo mật tốt nếu được sử dụng đúng cách	
Hiệu Suất	Thường thì MySQL được đánh giá là nhanh hơn so với PDO vì nó thiết kế để tương thích nhiều hơn với MySQL	Không tận dụng được hết tính năng của từng hệ quản trị cơ sở dữ liệu
Khả Năng Mở Rộng	Hạn chế mở rộng với các loại CSDL khác ngoài MySQL	Linh hoạt chuyển đổi giữa các loại CSDL có thể sử dụng

Bảng trên là một số những sự khác biệt giữa MySQLi và PDO thông thường việc chọn cách nào để kết nối đến CSDL với PHP là hoàn toàn tùy thuộc vào định hướng phát triển của mỗi dự án. Trong hướng dẫn này chúng ta sẽ làm việc với MySQLi cách kết nối phổ biến hơn và luôn được ưu tiên tiếp cận cho những người mới bắt đầu trong việc kết nối CSDL MySQL trong PHP.

3. Chèn dữ liệu vào trong cơ sở dữ liệu

Trong PHP chúng ta có thể chèn dữ liệu vào bảng theo 2 cách sau đây:

- Chèn dữ liệu vào bảng



Unset

```
<?php
// Dữ liệu cần thêm
$company = "Stringee";
$email = "info@stringee.com";
$year = 2023;

// Truy vấn INSERT
$sql = "INSERT INTO information (company, email, year)
VALUES ('$company', '$email', $year)";
$result = $conn->query($sql);

if ($result === TRUE) {
    echo "Inserted";
} else {
    echo "Error: " . $conn->error;
}
```

- Chèn dữ liệu vào bảng sử dụng Prepared statement

Unset

```
<?php
// Dữ liệu cần thêm
$company = "Stringee";
$email = "info@stringee.com";
$year = 2023;

// Truy vấn INSERT
$sql = "INSERT INTO information (company, email, year)
VALUES (?, ?, ?)";
$stmt = $conn->prepare($sql);
$stmt->bind_param("ssi", $company, $email, $year);

if ($stmt->execute()) {
```



```
        echo "Inserted";
} else {
    echo "Error: " . $stmt->error;
}

// Close statement
$stmt->close();
```

4. Cập nhật dữ liệu trong cơ sở dữ liệu

- Cập nhật dữ liệu

Unset

```
<?php
// Dữ liệu cần cập nhật
$id = 1;
$new_year = 2024;

// Truy vấn UPDATE
$sql = "UPDATE information SET year = $new_year WHERE id
= $id";
$result = $conn->query($sql);

if ($result === TRUE) {
    echo "Updated";
} else {
    echo "Error: " . $conn->error;
}
```

- Cập nhật dữ liệu sử dụng Prepared statement

Unset



```
<?php
// Dữ liệu cần cập nhật
$id = 1;
$new_year = 2024;

// Truy vấn UPDATE với Prepared Statement
$sql = "UPDATE information SET year = ? WHERE id = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("ii", $new_year, $id);

if ($stmt->execute()) {
    echo "Updated";
} else {
    echo "Error: " . $stmt->error;
}

// Đóng statement
$stmt->close();
```

5. Truy vấn dữ liệu trong cơ sở dữ liệu

- Truy vấn dữ liệu

Unset

```
<?php
// Truy vấn SELECT
$sql = "SELECT id, company, email, year FROM
information";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Xuất dữ liệu từ mỗi hàng
    while ($row = $result->fetch_assoc()) {
```



```
        echo "ID: " . $row["id"] . " - Company: " .
$row["company"] . " - Email: " . $row["email"] . " - Năm: "
. $row["year"] . "<br>";
    }
} else {
    echo "No data";
}
```

- Truy vấn dữ liệu sử dụng Prepared Statement

Unset

```
<?php
// Dữ liệu cần tìm kiếm
$year = 2020;

// Truy vấn SELECT với WHERE clause và Prepared Statement
$sql = "SELECT id, company, email, year FROM information
WHERE year > ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("i", $year);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    // Xuất dữ liệu từ mỗi hàng
    while ($row = $result->fetch_assoc()) {
        echo "ID: " . $row["id"] . " - Company: " .
$row["company"] . " - Email: " . $row["email"] . " - Year:
" . $row["year"] . "<br>";
    }
} else {
    echo "No data";
}
```



```
// Đóng statement  
$stmt->close();
```

6. Xóa dữ liệu trong cơ sở dữ liệu

- Xóa dữ liệu

Unset

```
<?php  
// ID của dữ liệu cần xóa  
$id = 1;  
  
// Truy vấn DELETE  
$sql = "DELETE FROM information WHERE id = $id";  
$result = $conn->query($sql);  
  
if ($result === TRUE) {  
    echo "Deleted";  
} else {  
    echo "Error: " . $conn->error;  
}
```

- Xóa dữ liệu sử dụng Prepared Statement

Unset

```
<?php  
// ID của dữ liệu cần xóa  
$id = 1;  
  
// Truy vấn DELETE với Prepared Statement  
$sql = "DELETE FROM information WHERE id = ?";  
$stmt = $conn->prepare($sql);
```



```
$stmt->bind_param("i", $id);

if ($stmt->execute()) {
    echo "Deleted";
} else {
    echo "Error: " . $stmt->error;
}

// Đóng statement
$stmt->close();
```

7. Một số lưu ý quan trọng khi truy vấn dữ liệu trong PHP

- Sử dụng Prepared Statements: Luôn luôn sử dụng Prepared Statements là một việc rất quan trọng để ngăn chặn tấn công SQL injection. Điều này giúp bảo vệ ứng dụng khỏi những sự tấn công từ những người dùng có ác ý thông qua phương pháp SQL injection.
- Kiểm tra lỗi: Cần đặt hàm kiểm tra lỗi ở những vị trí thích hợp để đảm bảo không có lỗi xảy ra trong quá trình thực thi chương trình. Hàm `mysqli_error()` là một trong những hàm thông dụng trong việc kiểm tra những lỗi này.
- Đóng kết nối đúng cách: Thông thường các lập trình viên thường xuyên bỏ quên, hay xem nhẹ việc đóng kết nối đến CSDL khi đã sử dụng xong. Chúng ta nên sử dụng hàm `$conn->close()` để thực hiện việc đóng kết nối để đảm bảo giải phóng tài nguyên đã sử dụng và tránh rò rỉ bộ nhớ.
- Xử lý kết quả: Luôn kiểm tra kết quả được trả ra từ câu truy vấn. Sử dụng các hàm như `mysqli_num_rows()` để kiểm tra số hàng được trả về hoặc `mysqli_fetch_assoc()` để lấy dữ liệu từ mỗi hàng.
- Xử lý dữ liệu đầu vào: Việc người dùng nhập liệu vào những dữ liệu độc hại cho hệ thống là một việc sẽ thường xuyên gặp phải trong quá trình làm việc với hệ thống. Việc sử dụng các hàm như `mysqli_real_escape_string()` có thể giúp chúng ta giảm thiểu được tình trạng này xảy ra cho ứng dụng của mình.
- Bảo mật mật khẩu: Chúng ta không nên lưu trực tiếp mật khẩu CSDL trong mã nguồn chương trình. Thông thường những dữ liệu nhạy cảm này sẽ



được lưu trong các file cấu hình riêng, bằng cách này sẽ giúp giảm thiểu việc lộ thông tin kết nối trực tiếp đến CSDL.

- Tối ưu hóa truy vấn: Tuy CSDL đã được tối ưu hóa rất nhiều qua các phiên bản. Nhưng, việc sử dụng các cột một các có tính toán, đồng thời là đặt các index cho các cột thường xuyên sử dụng, và xử lý kích thước trang kết quả sẽ giúp cho việc truy vấn trở nên tối ưu hơn.

Bài 10: Bảo mật trong PHP

Ở bài [Kết nối cơ sở dữ liệu với PHP](#), chúng ta đã cùng nhau tìm hiểu cách kết nối và truy vấn cơ sở dữ liệu trong PHP. Với bài này, chúng ta sẽ cùng nhau tìm hiểu về các biện pháp bảo mật trong PHP. Bất kỳ ứng dụng nào cũng đều có thể có những lỗ hổng bảo mật, nếu như các lập trình viên không có các biện pháp tốt để bảo vệ ứng dụng PHP khỏi các lỗ hổng bảo mật thì có thể gây ra những hậu quả không thể lường trước được.

1. Lỗ hổng bảo mật trong PHP là gì?

Trong các ngôn ngữ lập trình nói chung và trong PHP nói riêng, việc xuất hiện các lỗ hổng bảo mật là điều không thể tránh khỏi. Nó có thể đến từ rất nhiều nguyên nhân khác nhau và thường thì chúng sẽ phụ thuộc vào cách mà ứng dụng được phát triển. Sau đây, chúng ta sẽ cùng nhau tìm hiểu một số lỗ hổng bảo mật phổ biến thường xuất hiện trong PHP:

Vấn đề	Mô tả
SQL Injection (SQLi)	Đây là lỗ hổng gặp phải khi dữ liệu đầu vào không được kiểm soát một cách chặt chẽ, những giá trị đầu vào đó có thể khiến cho việc thực thi mã SQL trở nên không an toàn.
Cross-Site Scripting (XSS)	XSS cũng là một lỗi xảy ra khi các lập trình viên không kiểm soát tốt dữ liệu đầu vào của người dùng, khi hiển thị những dữ liệu đầu vào mà không kiểm



	soát kỹ càng người dùng có thể thực thi các mã JavaScript độc hại gây phát sinh lỗi cho hệ thống.
Cross-Site Request Forgery (CSRF)	Đây là kỹ thuật tấn công khi người dùng không thao tác nhưng kẻ tấn công lại có thể giả mạo các yêu cầu (request) được gửi từ máy của người dùng.
File Inclusion Vulnerabilities	Việc sử dụng các hàm như <code>include</code> hoặc <code>require</code> mà dữ liệu đầu vào không có sự kiểm soát nhất định thì cũng có thể gây ra những lỗ hổng cho phép tấn công bằng cách chèn mã độc vào những tập tin này.
Insecure Session Management	Người tấn công có thể chiếm đoạt phiên đăng nhập không an toàn của người dùng thông qua đó giả mạo danh tính người dùng.
Insecure Direct Object References (IDOR)	Cần kiểm tra quyền truy cập của người dùng một cách chính xác, việc này rất quan trọng trong việc kiểm soát các đối tượng người dùng có thể truy cập phù hợp với quyền của mình.
Remote Code Execution (RCE)	Việc có khả năng thực thi mã từ xa thường xảy ra khi ứng dụng chấp nhận các đầu vào không an toàn và thực thi nó mà không kiểm tra chúng một cách thích hợp.

Phía trên là một số lỗ hổng bảo mật thường gặp, chúng ta sẽ cùng nhau tìm hiểu chi tiết hơn về các lỗ hổng bảo mật trên cũng như cách phòng ngừa chúng sau đây.

2. SQL Injection (SQLi)

Lỗ hổng này thường xuất hiện khi ứng dụng PHP không kiểm soát được dữ liệu đầu vào từ phía người dùng. Điều này có thể dẫn đến việc người dùng sử dụng lối



để triển khai tấn công ứng dụng thông qua việc thực thi trái phép các mã SQL không an toàn trên cơ sở dữ liệu.

Ví dụ: Biến \$name trong câu query sau không được kiểm tra cũng như xử lý trước khi đưa vào câu SQL.

Unset

```
$sql = "SELECT * FROM posts WHERE name = '' . $name .  
''";
```

Ở đây, chúng ta có rất nhiều cách để có thể xử lý đâu vào trước khi thực thi câu SQL như sử dụng Prepared Statements hoặc Parameterized Queries thay vì chỉ nối chuỗi như trên, hoặc chúng ta cũng có thể sử dụng các hàm như `mysqli_real_escape_string()` để có thể kiểm soát được dữ liệu từ người dùng nhập vào.

Unset

```
// Sử dụng Prepared Statements  
$stmt = $conn->prepare("SELECT * FROM posts WHERE name =  
?");  
$stmt->bind_param("s", $name);  
$stmt->execute();
```

>>> Xem thêm bài viết tương tự tại đây:

- [Câu lệnh điều kiện và vòng lặp trong PHP](#)
- [Xử lý biểu mẫu trong PHP](#)

3. Cross-Site Scripting (XSS)

Lỗ hổng XSS cũng là một lỗ hổng xảy ra khi mà dữ liệu đầu vào của người dùng không được kiểm soát đúng cách. Gây nên nguy cơ khi mà người dùng hoàn toàn có thể chèn các mã JavaScript độc hại và thực thi nó trên ứng dụng web trong trình duyệt của người sử dụng.

Ví dụ: Dữ liệu được hiển thị lên trang web mà không được xử lý trước



Unset

```
echo "Welcome, " . $user_input;
```

Ở đây cách thông dụng nhất là sử dụng hàm htmlspecialchars() để có thể xử lý các ký tự đặc biệt thành các HTML entities trước khi hiển thị dữ liệu nhập từ người dùng.

Unset

```
echo "Welcome, " . htmlspecialchars($user_input);
```

4. Cross-Site Request Forgery (CSRF)

Đây là lỗ hổng mà những hacker có thể lợi dụng để tạo ra những request giả mạo trên máy khách của người dùng mà không cần sự tương tác của họ.

Ví dụ: Dưới đây là một biểu mẫu có thể bị tận dụng bởi lỗ hổng CSRF

Unset

```
<form action='http://stringeex.com/change_password' method='post'>
    <input type='hidden' name='password' value='malicious_password'>
    <input type='submit' value='Submit'>
</form>
```

Chúng ta có thể ngăn chặn phương pháp tấn công này bằng cách tạo ra một chuỗi token CSRF để xác thực yêu cầu mỗi khi người dùng gửi request lên server để đảm bảo những thao tác đó là do chính người dùng thao tác, chứ không phải là một cuộc tấn công lợi dụng lỗ hổng CSRF.

Unset



```
// Tạo ra token CSRF
$token = bin2hex(random_bytes(32));

// Lưu trữ lại token được tạo ra, ở đây chúng ta sẽ lưu
// vào $SESSION
$_SESSION['csrf_token'] = $token;

// Kiểm tra token CSRF trước khi xử lý yêu cầu
if ($_POST['csrf_token'] === $_SESSION['csrf_token']) {
    // Xử lý yêu cầu
}
```

5. Một số lỗi thường gặp khác

- Sử dụng các phiên bản quá cũ: Với sự phát triển của nhiều cách tấn công phức tạp hiện nay, việc liên tục cập nhật phần mềm cũng có tác dụng không hề nhỏ trong việc cập nhật các phiên bản vá lỗi. Nó sẽ giúp cho các lỗi từ thư viện được khắc phục nhờ đó giúp hệ thống hoạt động tốt và ổn định hơn.
- Giới hạn quyền truy cập một số tệp tin quan trọng: Trong từng ứng dụng sẽ có những tệp tin không được phép truy cập từ những người dùng thông thường, chúng ta cần tìm hiểu xem hệ thống đang sử dụng có phương pháp nào để giữ an toàn cho những tệp tin này.

Bài 11: Hướng dẫn tạo trang web động bằng PHP

Trong các trang web động, PHP có thể xử lý dữ liệu động, tương tác với cơ sở dữ liệu, xử lý biểu mẫu và dữ liệu người dùng... Trong bài viết này chúng ta sẽ cùng nhau tìm hiểu về web động trong PHP và dựng một trang web động bằng PHP.

1. PHP và web động

1.1. Sự phổ biến và tính linh hoạt của PHP trong phát triển web động

Sự phổ biến của PHP:



PHP là một trong những ngôn ngữ phát triển web phổ biến nhất trên toàn cầu, được sử dụng rộng rãi bởi hàng triệu website và ứng dụng web. PHP có một cộng đồng phát triển lớn, với hàng ngàn nhà phát triển, diễn đàn, và tài liệu hỗ trợ, cung cấp sự giúp đỡ và giải pháp cho các vấn đề phát triển.

Tính linh hoạt của PHP trong phát triển web động:

- Xử lý dữ liệu động: PHP làm việc tốt trong việc tạo ra nội dung động trên web dựa trên dữ liệu đầu vào hoặc dữ liệu từ cơ sở dữ liệu.
- Tương tác với cơ sở dữ liệu: PHP kết nối và tương tác với nhiều hệ quản trị cơ sở dữ liệu (ví dụ: MySQL, PostgreSQL), cho phép truy xuất và quản lý dữ liệu một cách linh hoạt.
- Xử lý biểu mẫu và dữ liệu người dùng: PHP giúp xử lý dữ liệu từ các biểu mẫu người dùng, thực hiện kiểm tra và xác thực dữ liệu trước khi lưu vào cơ sở dữ liệu.

1.2. Các tính năng chính của PHP hỗ trợ xây dựng ứng dụng web động

- Xử lý dữ liệu động: PHP có các hàm và công cụ để tạo và điều chỉnh nội dung động dựa trên yêu cầu cụ thể của người dùng hoặc dữ liệu từ cơ sở dữ liệu.
- Tương tác với cơ sở dữ liệu: PHP kết nối và thực hiện các thao tác đến cơ sở dữ liệu, cho phép lấy dữ liệu và hiển thị thông tin từ cơ sở dữ liệu lên trang web một cách linh hoạt.
- Xử lý biểu mẫu và dữ liệu người dùng: PHP dễ dàng xử lý dữ liệu từ các biểu mẫu người dùng, thực hiện kiểm tra, xác thực dữ liệu và lưu trữ thông tin một cách an toàn.
- Hỗ trợ framework và thư viện: Có nhiều framework và thư viện PHP như Laravel, Symfony, CodeIgniter cung cấp cấu trúc, công cụ hỗ trợ cho quá trình phát triển nhanh chóng và hiệu quả.
- PHP không chỉ phổ biến mà còn linh hoạt trong việc xây dựng các ứng dụng web động, bằng cách cung cấp nhiều tính năng mạnh mẽ và hỗ trợ từ các công cụ, framework và cộng đồng phát triển.

2. Các yếu tố cơ bản

2.1. Cú pháp PHP cơ bản

Biến trong PHP:



Unset

```
$tenBien = "Giá trị của biến"; // Khai báo biến  
$soNguyen = 10;  
$soThuc = 3.14;  
$chuoi = "Hello, World!" ;
```

Câu lệnh điều kiện:

Unset

```
if ($condition) {  
    // Code nếu điều kiện đúng  
} elseif ($another_condition) {  
    // Code nếu điều kiện khác đúng  
} else {  
    // Code nếu không có điều kiện nào đúng  
}
```

Vòng lặp trong PHP:

- Vòng lặp while:

Unset

```
$i = 0;  
while ($i < 5) {  
    // Code lặp  
    $i++;  
}
```

- Vòng lặp for:



Unset

```
for ($i = 0; $i < 5; $i++) {  
    // Code lặp  
}
```

Hàm trong PHP:

Unset

```
function tenHam($thamSo1, $thamSo2) {  
    // Code trong hàm  
    return $ketQua;  
}
```

Gọi hàm:

Unset

```
$ketQua = tenHam($giaTri1, $giaTri2);
```

2.2. Liên kết PHP với HTML/CSS/JavaScript

Nhúng mã PHP vào trang web, nhúng mã PHP vào HTML:

Unset

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>Trang web PHP</title>  
</head>  
<body>  
    <h1>Chào mừng đến với <?php echo "PHP" ; ?></h1>
```



```
<p>Biến PHP: <?php $bien = "Hello"; echo $bien;  
?></p>  
</body>  
</html>
```

Nhúng mã PHP vào CSS:

PHP không thể nhúng trực tiếp vào CSS. Tuy nhiên, bạn có thể tạo mã CSS dựa trên các giá trị PHP và gán nó vào các phần tử HTML.

Nhúng mã PHP vào JavaScript:

Unset

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>Trang web PHP và JavaScript</title>  
</head>  
<body>  
    <button onclick="<?php echo "alert('Hello, World!')";  
?>">Click me</button>  
</body>  
</html>
```

Nhúng mã PHP vào các ngôn ngữ khác như HTML, CSS, và JavaScript giúp tương tác và điều chỉnh nội dung của trang web dựa trên dữ liệu PHP được xử lý từ máy chủ.

3. Tương tác với cơ sở dữ liệu

3.1. Kết nối và truy vấn cơ sở dữ liệu sử dụng PHP

Kết nối đến cơ sở dữ liệu MySQL:



Unset

```
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "ten_database";

// Tạo kết nối
$conn = new mysqli($servername, $username, $password,
$dbname);

// Kiểm tra kết nối
if ($conn->connect_error) {
    die("Kết nối thất bại: " . $conn->connect_error);
} else {
    echo "Kết nối thành công";
}
```

Truy vấn cơ sở dữ liệu MySQL:

Unset

```
$sql = "SELECT * FROM ten_bang";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "ID: " . $row["id"]. " - Tên: " .
$row["ten"]. "<br>";
    }
} else {
    echo "Không có dữ liệu";
}
$conn->close();
```



3.2. Hiển thị dữ liệu động từ CSDL lên trang web

Hiển thị dữ liệu từ cơ sở dữ liệu trên trang web:

```
Unset

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "ten_database";

// Tạo kết nối
$conn = new mysqli($servername, $username, $password,
$dbname);

// Kiểm tra kết nối
if ($conn->connect_error) {
    die("Kết nối thất bại: " . $conn->connect_error);
}

// Truy vấn dữ liệu
$sql = "SELECT * FROM ten_bang";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "ID: " . $row["id"]. " - Tên: " .
$row["ten"]. "<br>";
    }
} else {
    echo "Không có dữ liệu";
}
$conn->close();
?>
```



Đoạn mã trên sẽ kết nối đến cơ sở dữ liệu MySQL, truy vấn dữ liệu từ bảng ten_bang, và hiển thị thông tin lấy được lên trang web. Bạn có thể tùy chỉnh mã để hiển thị dữ liệu theo cách phù hợp với giao diện của trang web.

4. Xử lý form và tương tác với người dùng

4.1. Xử lý form

Nhận dữ liệu từ người dùng và xác thực thông tin từ biểu mẫu HTML bằng PHP:

Unset

```
<form action="xuly.php" method="post">
    Tên: <input type="text" name="ten"><br>
    Email: <input type="text" name="email"><br>
    <input type="submit" value="Gửi">
</form>
```

Xử lý dữ liệu biểu mẫu và xác thực thông tin trong tệp xuly.php:

Unset

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

    // Kiểm tra và xử lý dữ liệu
    if (empty($ten) || empty($email)) {
        echo "Vui lòng điền đầy đủ thông tin";
    } else {
        // Xử lý dữ liệu, ví dụ: lưu vào cơ sở dữ liệu,
        gửi email, vv.
        echo "Dữ liệu đã được gửi: Tên - " . $ten . ", 
Email - " . $email;
    }
}
```



```
}
```

```
?>
```

4.2. Tương tác với người dùng thông qua các yêu cầu và phản hồi (GET, POST)

Yêu cầu và phản hồi GET:

```
Unset
```

```
<!-- Link gửi yêu cầu GET -->
<a href="xuly.php?name=John&age=30">Gửi yêu cầu</a>

<!-- Xử lý yêu cầu GET -->
<?php
if(isset($_GET['name']) && isset($_GET['age'])) {
    $name = $_GET['name'];
    $age = $_GET['age'];
    echo "Xin chào, $name. Bạn $age tuổi.";
}
?>
```

Yêu cầu và phản hồi POST:

```
Unset
```

```
<form action="xuly.php" method="post">
    Tên: <input type="text" name="ten"><br>
    Email: <input type="text" name="email"><br>
    <input type="submit" value="Gửi">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
```



```
$ten = $_POST["ten"];
$email = $_POST["email"];

// Xử lý dữ liệu từ biểu mẫu POST ở đây
echo "Dữ liệu đã được gửi: Tên - " . $ten . ", Email
- " . $email;
}
?>
```

Mã PHP trên giúp nhận dữ liệu từ người dùng thông qua biểu mẫu HTML và xử lý thông tin đó. Ngoài ra, bạn cũng có thể tương tác với người dùng thông qua yêu cầu GET và POST để nhận dữ liệu từ URL hoặc biểu mẫu gửi dữ liệu.

Bài 12: Xây dựng ứng dụng web hoàn chỉnh với PHP

Trong bài trước, chúng ta đã cùng nhau tìm hiểu các kiến thức cần có để tạo ra một trang web. Hôm nay chúng ta sẽ cùng nhau xây dựng một trang web hoàn chỉnh. Dưới đây là các ví dụ đơn giản về cách tạo một trang web đăng nhập bằng PHP và MySQL.

1. Tạo cơ sở dữ liệu

Tạo một bảng trong cơ sở dữ liệu MySQL để lưu trữ thông tin người dùng. Ví dụ:

Unset

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);
```



2. Kết nối với cơ sở dữ liệu

Tạo kết nối từ PHP đến cơ sở dữ liệu MySQL. Tạo file config.php để lưu thông tin kết nối:

Unset

```
<?php
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$dbname = "your_database_name";

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

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

3. Tạo trang đăng nhập (login.php)

Tạo một form HTML để người dùng nhập thông tin đăng nhập và xác thực với cơ sở dữ liệu:

Unset

```
<!DOCTYPE html>
<html>
<head>
    <title>Trang Đăng Nhập</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
```



```
        margin: 0;
        padding: 0;
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
    }

    .container {
        background-color: #fff;
        border-radius: 5px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        padding: 20px;
        width: 300px;
    }

    h2 {
        text-align: center;
    }

    form {
        display: flex;
        flex-direction: column;
    }

    label {
        margin-bottom: 5px;
    }

    input[type="text"],
    input[type="password"] {
        padding: 8px;
        margin-bottom: 10px;
        border-radius: 3px;
        border: 1px solid #ccc;
```



```
        flex: 1; /* Để input mở rộng theo chiều ngang
    */
}

.input-group {
    display: flex;
    flex-direction: row;
    justify-content: space-between;
    margin-bottom: 10px;
}

button {
    padding: 10px;
    background-color: #4caf50;
    color: white;
    border: none;
    border-radius: 3px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #45a049;
}

</style>
</head>
<body>
    <div class="container">
        <h2>Đăng Nhập</h2>
        <form method="post" action="">
            <div class="input-group">
                <label for="username">Tên đăng
nhập:</label>
                <input type="text" id="username"
name="username" required>
            </div>
        </form>
    </div>

```



```
<div class="input-group">
    <label for="password">Mật khẩu:</label>
        <input type="password" id="password"
name="password" required>
    </div>
    <button type="submit">Đăng Nhập</button>
</form>
<?php if (isset($error)) { echo $error; } ?>
</div>
</body>
</html>
```

Chúng ta có được trang Login:

The screenshot shows a login page with the following elements:

- A large title "Đăng Nhập" centered at the top.
- An input field labeled "Tên đăng nhập:" followed by a text input box.
- An input field labeled "Mật khẩu:" followed by a text input box.
- A green button labeled "Đăng Nhập" at the bottom.

4. Trang Dashboard (dashboard.php)



Nếu đăng nhập thành công, người dùng sẽ được chuyển hướng đến trang dashboard:

```
Unset

<?php
session_start();

if (!isset($_SESSION['username'])) {
    header("Location: login.php");
    exit;
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>Dashboard</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }

        .container {
            width: 80%;
            margin: 0 auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            margin-top: 50px;
        }

        h2 {
            text-align: center;
        }
    </style>

```



```
        margin-bottom: 20px;
    }

    p {
        text-align: center;
        font-size: 18px;
        color: #333;
    }

    a {
        display: block;
        width: 100px;
        margin: 20px auto;
        padding: 10px;
        text-align: center;
        color: #fff;
        background-color: #4caf50;
        text-decoration: none;
        border-radius: 5px;
        transition: background-color 0.3s ease;
    }

    a:hover {
        background-color: #45a049;
    }
</style>
</head>
<body>
    <div class="container">
        <h2>Xin chào, <?php echo $_SESSION['username']; ?></h2>
        <p>Đây là trang dashboard của bạn.</p>
        <a href="logout.php">Đăng Xuất</a>
    </div>
</body>
</html>
```



Chúng ta có được màn hình sau khi login thành công:

Xin chào, dungtq

Đây là trang dashboard của bạn.

Đăng Xuất

5. Tính năng đăng xuất (logout.php)

Tạo một trang logout để người dùng có thể đăng xuất:

Unset

```
<?php
session_start();
session_destroy();
header("Location: login.php");
exit;
?>
```

6. Những lưu ý khi viết khi xây dựng trang web bằng PHP

- Bảo mật mật khẩu: Không lưu mật khẩu trong cơ sở dữ liệu dưới dạng văn bản thô, hãy sử dụng hàm băm (hashing) như password_hash() trong PHP để mã hóa mật khẩu.
- Bảo mật thêm: Sử dụng các biện pháp bảo mật như sử dụng Prepared Statements để ngăn chặn tấn công SQL Injection.
- Đây chỉ là một ví dụ đơn giản, để xây dựng một hệ thống đăng nhập an toàn và đầy đủ, cần xem xét và triển khai nhiều biện pháp bảo mật và tính năng hơn.



Tổng kết

Cuốn ebook "Hướng Dẫn Lập Trình PHP từ Cơ Bản đến Nâng Cao" đã đồng hành cùng bạn trong hành trình khám phá và học tập về ngôn ngữ lập trình PHP. Từ những khái niệm cơ bản nhất đến những kỹ thuật nâng cao, chúng ta đã cùng nhau tìm hiểu và áp dụng kiến thức vào thực tiễn. Trong cuốn ebook này, chúng ta đã:

- Tìm hiểu về cú pháp và cấu trúc cơ bản của PHP: Bắt đầu từ những khái niệm cơ bản như biến, điều kiện, vòng lặp và hàm, chúng ta đã xây dựng nền tảng vững chắc cho việc lập trình PHP.
- Khám phá các chủ đề nâng cao: Tiếp tục với các chủ đề nâng cao như xử lý mảng, chuỗi, session và cookies, chúng ta đã mở rộng kiến thức và áp dụng vào các ứng dụng thực tế.
- Tạo ra các ứng dụng PHP động và tương tác: Từ việc xây dựng các trang web đơn giản đến các ứng dụng động với cơ sở dữ liệu MySQL, chúng ta đã áp dụng kiến thức để tạo ra các ứng dụng web phức tạp và tương tác.
- Học hỏi và phát triển kỹ năng lập trình PHP: Cuốn ebook này không chỉ là một hướng dẫn lập trình PHP, mà còn là một công cụ để bạn phát triển kỹ năng lập trình của mình thông qua việc áp dụng kiến thức vào thực tiễn.

Hy vọng bạn đã có được một cơ sở vững chắc về lập trình PHP và tự tin để tiếp tục phát triển kỹ năng của mình trong lĩnh vực này. Hãy trau dồi thực hành và không ngừng học hỏi để trở thành một lập trình viên PHP giỏi nhé.

Nhưng hành trình học tập không bao giờ kết thúc, Stringee chắc chắn sẽ còn mang đến cho bạn những tài liệu chất lượng về thế giới công nghệ và các ngôn ngữ, công cụ lập trình nổi tiếng.

Đừng quên theo dõi các bài viết mới nhất về công nghệ thông tin trên blog chia sẻ kiến thức của Stringee tại địa chỉ: <https://stringee.com/vi/blog>.

Cảm ơn bạn đã tham gia hành trình học lập trình PHP cùng Stringee!



Stringee Communication APIs - Giải pháp tích hợp tính năng giao tiếp vào App/web số 1 Việt Nam

Ngày nay, nhu cầu liên lạc giao tiếp đều trở nên rất phổ biến và cần thiết trong hầu hết các doanh nghiệp. Để xây dựng được các tính năng giao tiếp vào ứng dụng thì doanh nghiệp cần đầu tư nguồn lực, chi phí lớn cho đội ngũ lập trình viên trong vòng 1 - 3 năm để phát triển.

Giờ đây, với các API và SDK hiện đại, các developers có thể thêm các tính năng này trong thời gian ngắn, chỉ từ 15 phút. Tiết kiệm đến 60% nguồn lực, chi phí cho doanh nghiệp.

Stringee là công ty chuyên cung cấp giải pháp API giao tiếp (Communication APIs) và phần mềm tổng đài chăm sóc khách hàng đa kênh cho doanh nghiệp. Là đơn vị tiên phong về lĩnh vực API giao tiếp và Contact Center tại Việt Nam, Stringee có cơ hội triển khai phần mềm cho nhiều khách hàng lớn thuộc top **VNR500** như: Viettel, MobiFone, VietinBank, TPBank, VNDIRECT, Shinhan Finance, PTI, Prudential... và hơn 1000 Doanh nghiệp khác.

Với các giải pháp API đầy đủ, đa dạng bao phủ mọi nghiệp vụ như Voice, Video, SMS, Chat, Contact Center, Stringee chính là chìa khoá giúp doanh nghiệp số hoá quy trình giao tiếp và CSKH một cách triệt để, đồng thời hỗ trợ tối đa hoạt động kinh doanh.

Hiện Stringee đang cung cấp các APIs bao gồm:

Voice API (API thoại)

Voice API hay [Voice Call API](#) thường được lập trình viên thêm vào ứng dụng/web để giúp khách hàng liên lạc với doanh nghiệp dễ dàng hơn. Tính năng này bao gồm chức năng cơ bản của gọi thoại là kết nối bằng số điện thoại truyền thống, tuy nhiên các tính năng nổi bật có thể được thêm vào như: ghi âm cuộc gọi, định



tuyển cuộc gọi, cuộc gọi hội nghị, hay chức năng chuyển văn bản thành giọng nói (text to speech) vào ứng dụng một cách nhanh chóng.

Với Voice API, khách hàng chỉ cần 1 click ngay trên ứng dụng của doanh nghiệp là có thể gọi điện đến phòng chăm sóc khách hàng, không cần mất thời gian thao tác ấn số gọi như thông thường.

Video Call API

Video Call API cũng là một API thuộc [Call API](#). Tương tự như Voice API, lập trình viên có thể sử dụng API này để nhúng tính năng gọi video vào ứng dụng, web của mình một cách nhanh chóng. Khách hàng chỉ cần click-to-call trên ứng dụng là có thể gọi video với bộ phận bán hàng/CSKH của doanh nghiệp.

Với đặc thù một số lĩnh vực, và đặc biệt cần gặp mặt Face-to-Face, thì video call chính là giải pháp tối ưu nhất giúp doanh nghiệp cá nhân hóa trải nghiệm khách hàng, đặc biệt an toàn trong giai đoạn dịch Covide-19 cần giãn cách xã hội, hạn chế gặp mặt trực tiếp.

SMS API

Theo thống kê, hơn 65% dân số thế giới gửi tin nhắn văn bản, và hơn hai nghìn tỷ tin nhắn được gửi chỉ trong năm 2018. Vì vậy, mọi doanh nghiệp đều muốn tận dụng tính năng nhắn tin SMS để tiếp cận khách hàng. Ví dụ như: tin nhắn thông báo chương trình khuyến mãi, chăm sóc khách hàng,...

Hầu hết các công ty CPaaS đều cung cấp tính năng SMS API này. Giải pháp [Stringee SMS API](#) giúp doanh nghiệp có thể gửi tin nhắn SMS hàng loạt đến hàng nghìn khách hàng cùng lúc với tên thương hiệu của mình, đáp ứng nhiều giới hạn về kích thước tệp. Đồng thời cho phép theo dõi trạng thái tin nhắn như: đã gửi thành công hay chưa, báo cáo chiến dịch,...

Chat API

Sử dụng [Chat API](#), bạn có thể thêm tính năng chat vào ứng dụng của mình cho khách hàng chat miễn phí với nhau, chat với nhân viên chăm sóc khách hàng hoặc chat với AI trả lời tự động ngay trên chính website/ứng dụng của bạn.



Chat API cho phép người dùng gửi hình ảnh, gif, link dẫn, file đính kèm, Rich text,... Đồng thời, nội dung hội thoại được lưu trữ gắn với tài khoản người dùng, giúp họ dễ dàng xem lại, tìm kiếm thông tin. Doanh nghiệp có thể xem báo cáo - thống kê chi tiết và lịch sử chat

Video Conference API

Video Conference API giúp lập trình viên thêm tính năng hội nghị truyền hình, gọi hình đa điểm, gọi nhóm ứng dụng, phần mềm của mình một cách nhanh chóng nhất. Thông thường để xây dựng từ đầu tính năng video conference, doanh nghiệp sẽ phải mất từ 1 - 3 năm để phát triển, nghiên cứu các công nghệ khó (VoIP, xử lý Video,...) cũng như phải duy trì 1 hệ thống máy chủ VoIP phức tạp, công kenneh.

Sử dụng Video Conference API, việc xây dựng trở nên nhanh chóng và dễ dàng. Ngoài ra API này còn có các tính năng hỗ trợ như: ghi âm, ghi hình, chia sẻ màn hình (share screen),... rất hữu ích khi sử dụng.

Quý bạn đọc quan tâm xin mời đăng ký [NHÂN TƯ VẤN TẠI ĐÂY](#):