

Chương 4 : PHP


Giảng Viên: ThS. Tạ Việt Phương

1

1

Nội dung

1. Giới thiệu PHP
2. Cơ chế hoạt động của WebServer
3. Cú pháp & Quy ước trong PHP
4. Lập trình hướng đối tượng trong PHP
5. Bài tập



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT-NGHIÊN

Phát triển ứng dụng Web

2

2

1. Giới thiệu PHP



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT-NGHIÊN

Phát triển ứng dụng Web

3

3

Giới thiệu về PHP - PHP là gì ?

- PHP là gì?

4

Giới thiệu về PHP - PHP là gì ?

- **PHP** viết tắt của **PHP Hypertext Preprocessor**
- Là ngôn ngữ server-side script, tương tự như ASP, JSP, ... thực thi ở phía WebServer
- Tập tin PHP có phần mở rộng là .php
- Cú pháp ngôn ngữ giống ngôn ngữ C & Perl



5

Giới thiệu về PHP - Lịch sử phát triển

- **PHP/FI** là phiên bản đầu tiên của PHP, được phát triển vào năm 1994 bởi Rasmus Lerdorf. Ban đầu, PHP/FI có tên là **Personal Home Page / Forms Interpreter**. PHP/FI cung cấp một số chức năng cơ bản như khả năng xử lý các biểu mẫu HTML và quản lý cơ sở dữ liệu, nhưng không hề có tính năng lập trình hướng đối tượng hoặc cú pháp phức tạp như PHP hiện đại.
- Sau PHP/FI, phiên bản chính thức đầu tiên của PHP được gọi là **PHP 3**. Phiên bản PHP 3 được phát hành vào năm 1997 sau khi Rasmus Lerdorf hợp tác với hai lập trình viên là Zeev Suraski và Andi Gutmans để **viết lại toàn bộ PHP/FI**. PHP 3 là phiên bản đầu tiên có tên gọi là "PHP" như ngày nay và được hỗ trợ bởi Zend Engine, đặt nền móng cho sự phát triển của PHP hiện đại.
- **PHP 4** (2000) Trở thành một thành phần độc lập cho các webserver. Parser đổi tên thành Zend Engine 1.0. Hỗ trợ sessions. Bổ sung các tính năng bảo mật cho PHP

6

Giới thiệu về PHP - Lịch sử phát triển

- **PHP 5** (2004) Zend Engine 2 được giới thiệu, tăng cường khả năng **OOP (lập trình hướng đối tượng)**. Hỗ trợ các tính năng như constructors, destructors, interfaces, và exception handling. Bổ sung các extension như SimpleXML, PDO, và SOAP.
- **PHP 7** (2015) Hiệu suất tăng gấp đôi nhờ Zend Engine 3. Scalar type hinting và return type declarations. Strict typing giúp kiểm soát kiểu dữ liệu chặt chẽ.
- **PHP 8** (2020) Just-In-Time (JIT) compiler: Cải thiện hiệu suất đáng kể. Match expressions: Giúp thay thế cú pháp switch với cú pháp ngắn gọn hơn. Attributes: Cho phép gắn thêm metadata vào các class, phương thức và thuộc tính. Hỗ trợ thêm Union types và Nullsafe operator, tăng cường tính linh hoạt và an toàn.
- Phiên bản mới nhất của PHP là **PHP 8.3.12** (www.php.net)

7

Giới thiệu về PHP – Ưu điểm 1

- PHP là Server Side Scripting được sử dụng để:
 - Xây dựng các ứng dụng web
 - Thường được sử dụng để xây dựng CMS như WordPress, Joomla, và Drupal...
 - Có sự linh hoạt và khả năng mở rộng khi xây dựng các ứng dụng web lớn, hỗ trợ RESTful APIs
 - Có tính bảo mật và các tính năng như lọc đầu vào (input filtering) và thoát đầu ra (output escaping), là hai công cụ hữu ích để bảo vệ trang web.

8

Giới thiệu về PHP – Ưu điểm 2

- **Đa môi trường – đa nền tảng (Multi-Platform):** tương thích đa nền tảng
 - Web Servers: Apache, Microsoft IIS, Caudium, Netscape Enterprise Server
 - Hệ điều hành: UNIX (HP-UX, OpenBSD, Solaris, Linux), Mac OSX, Windows
 - Hệ QTCSDL: Adabas D, dBase, Empress, FilePro (read-only), Hyperwave, IBM DB2, Informix, Ingres, InterBase, FrontBase, mSQL, Direct MS-SQL, MySQL, ODBC, Oracle (OCI7 and OCI8), Ovrimos, PostgreSQL, SQLite, Solid, Sybase, Velocis, Unix dbm

9

Giới thiệu về PHP – Ưu điểm 3

- Miễn phí

	PHP
Software	Free
Platform	Free: Linux, Apache, Nginx, MySQL, MariaDB...
Development Tools	Free: NetBeans, VS Code, Atom, Laravel, Symfony, CodeIgniter, Composer, phpMyAdmin ...

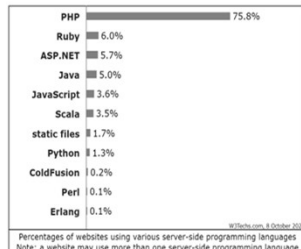
- Là nguồn mở và miễn phí, có sự hỗ trợ từ cộng đồng lớn và tài liệu phong phú

10

Giới thiệu về PHP – Ưu điểm 4

- Được sử dụng rộng rãi trong môi trường phát triển web: PHP được sử dụng bởi 75,8% trong số tất cả các trang web có ngôn ngữ lập trình phía máy chủ mà thống kê được biết.

(theo W3Techs.com, tháng 10 năm 2024
https://w3techs.com/technologies/overview/programming_language)



11

Một số website lớn sử dụng PHP



12

HackLang (Hack)

- Hack là ngôn ngữ lập trình được tạo bởi Facebook, nó có thể tương thích hoàn toàn với PHP, để nâng cấp PHP khi phải xử lý các hệ thống phức tạp và đòi hỏi cao về hiệu suất.
- Hack chạy trên HipHop Virtual Machine (HHVM) - một máy ảo được phát triển bởi Facebook nhằm thay thế cho Zend Engine,
- Cấu trúc cơ bản của một file Hack tương tự như PHP nhưng có 1 vài thay đổi nhỏ như Hack thì bắt đầu với <?hh, còn PHP thì là <?php.
- Một số tính năng Hack hỗ trợ: Generics (tổng quát hóa), kiểu Vectors, Maps, và Sets; Hack hỗ trợ Static Typing (kiểu tĩnh), giúp phát hiện lỗi trong quá trình biên dịch thay vì lúc chạy nhưng vẫn hỗ trợ Dynamic Typing (Kiểu Động), Lập Trình Bất Đồng Bộ ...

13

2. Cơ chế hoạt động của WebServer

14

Cơ chế hoạt động của WebServer

- **CGI (Common Gateway Interface)** là một giao diện tiêu chuẩn cho phép máy chủ web giao tiếp với các chương trình hoặc tập lệnh bên ngoài (ví dụ như PHP). Khi một yêu cầu HTTP được gửi đến máy chủ web, CGI sẽ tạo ra một quy trình mới cho mỗi yêu cầu để xử lý, sau đó trả về kết quả. Vấn đề của CGI: Mỗi khi có một yêu cầu mới, máy chủ phải khởi tạo lại toàn bộ quy trình, điều này làm giảm hiệu suất, đặc biệt là đối với các ứng dụng web có lưu lượng truy cập cao.
- **FastCGI** ra đời để giải quyết vấn đề hiệu suất của CGI. FastCGI giữ một số quy trình đã được khởi tạo sẵn trong bộ nhớ và tái sử dụng chúng, thay vì khởi tạo lại quy trình cho mỗi yêu cầu mới. Điều này giúp tăng tốc độ xử lý các yêu cầu HTTP.

15

Cơ chế hoạt động của WebServer

- **PHP-FPM (PHP FastCGI Process Manager)** là một triển khai của FastCGI dành cho PHP, quản lý việc xử lý các yêu cầu từ máy chủ web đến PHP thông qua FastCGI. PHP-FPM được sử dụng rộng rãi nhờ hiệu suất cao và khả năng xử lý nhiều yêu cầu đồng thời.
- Khi một yêu cầu HTTP được gửi đến Nginx hoặc Apache, máy chủ web sẽ chuyển yêu cầu đó đến PHP-FPM qua FastCGI. PHP-FPM đóng vai trò quản lý và điều phối các quy trình (process) PHP sẽ nhận yêu cầu, chọn một quy trình đang ở trạng thái nhàn rỗi (sẵn sàng) để xử lý yêu cầu đó. Quy trình PHP được chọn sẽ nhận mã PHP từ yêu cầu HTTP, đây có thể là một tập tin PHP hoặc mã PHP cần xử lý để tạo ra phản hồi động

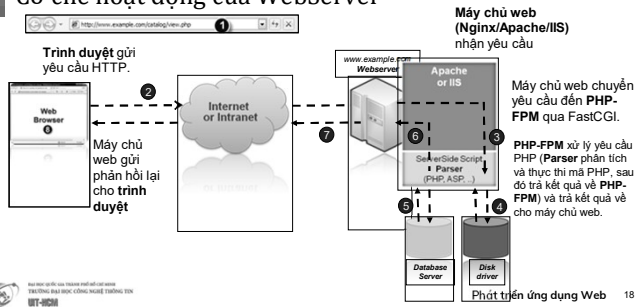
16

Cơ chế hoạt động của WebServer

- Trong quy trình PHP, có một PHP Engine, và Parser là thành phần chính trong PHP Engine chịu trách nhiệm xử lý mã PHP. Quy trình của Parser gồm các bước sau:
 - Bước 1: Lexical Analysis (Phân tích từ vựng)
 - Bước 2: Syntax Analysis (Phân tích cú pháp)
 - Bước 3: Compilation (Biên dịch mã thành Opcode)
 - Bước 4: Execution (Thực thi mã)
- Sau khi Parser trong quy trình PHP đã xử lý mã PHP và tạo ra kết quả, kết quả này sẽ được gửi trở lại PHP-FPM. PHP-FPM chuyển kết quả đến máy chủ web, từ đó máy chủ web gửi phản hồi cuối cùng về trình duyệt của người dùng.

17

Cơ chế hoạt động của WebServer



18

Chèn 1 đoạn lệnh PHP vào trang web

- Mã lệnh PHP được đặt trong các cặp thẻ sau :

Thẻ mở	Thẻ đóng	
<?	?>	short echo tag, viết tắt của <?php ... ?>
<?php	?>	Cách phổ biến
<script language="php">	</script>	

- Ví dụ:

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

```
<? echo "Hello, World!"; ?>
```

```
<?= "Hello, World!"; ?>
```

19

Cơ chế hoạt động của WebServer

```
1 <html>
2 <head>
3   <title>Test Server Script Parser</title>
4 </head>
5 <body>
6   <h1>Server Script Parser</h1>
7   Hello world HTML
8   <br />
9   <br />
10
11   <!-- ASP Code -->
12   <% response.write("Hello ASP Parser !!!") %>
13   <br />
14   <br />
15   <!-- PHP Code -->
16   <?php echo "Hello PHP Parser !!!" ?>
17   <br />
18   <br />
19
20 </body>
21 </html>
22
```

20

Stack phát triển web

- Là một tập hợp các công nghệ và công cụ được sử dụng cùng nhau để xây dựng và vận hành các ứng dụng web. Một stack phát triển web bao gồm các thành phần cần thiết để phát triển cả front-end (giao diện người dùng) và back-end (máy chủ và cơ sở dữ liệu) của một ứng dụng web, giúp cho ứng dụng có thể hoạt động đầy đủ từ phía người dùng đến máy chủ.
 - Front-end (Client-side):** HTML, CSS, JS và các thư viện & framework phổ biến: React, Angular, Vue.js...
 - Back-end (Server-side):** Ngôn ngữ lập trình, Web Server, Cơ sở dữ liệu: và các framework back-end phổ biến: Laravel (PHP), Express.js (Node.js)...
 - Các thành phần bổ sung khác:** API (Application Programming Interface), Middleware, Hệ thống quản lý phiên bản...

21

Stack phát triển web

• Một số stack phát triển web phổ biến:

- MEAN: MongoDB (Cơ sở dữ liệu NoSQL), Express.js (Framework back-end chạy trên Node.js), Angular (Framework front-end), Node.js
- LAMP: Linux, Apache, MySQL, PHP/Python/Perl
- LEMP: Linux, Nginx, MySQL, PHP/Python/Perl
- **XAMPP**: Cross-platform, Apache, MySQL, PHP, Perl
- **WAMP**: Windows, Apache, MySQL, PHP
- **MAMP**: macOS, Apache, MySQL, PHP
- ...



Phát triển ứng dụng Web 22

22

Stack phát triển web

Đặc điểm	XAMPP	WAMP	LAMP
Nền tảng	Windows, macOS, Linux	Windows	Linux
Máy chủ Web	Apache	Apache	Apache/Nginx
Cơ sở dữ liệu	MySQL/MariaDB	MySQL/MariaDB	MySQL/MariaDB
Ngôn ngữ	PHP, Perl	PHP	PHP, Python, Perl
Dễ cài đặt	Cao	Cao	Trung bình
Bảo mật	Trung bình (phục vụ phát triển)	Trung bình	Cao
Mục đích sử dụng	Phát triển cục bộ	Phát triển cục bộ trên Windows	Triển khai thực tế và phát triển trên Linux
Quản lý	Giao diện đồ họa, dễ sử dụng	Giao diện đồ họa, dễ sử dụng	Chủ yếu dùng dòng lệnh

Phát triển ứng dụng Web 23

23

Stack phát triển web

- XAMPP: <https://www.apachefriends.org/>
- WAMP: <https://www.wampserver.com/en/>
- LAMP: <https://www.ampps.com/downloads/> hoặc các trang của các nhà phân phối Linux
- **Chú ý:** Khi cài đặt cần cấu hình (config) lại các port để có thể sử dụng, tránh xung đột với các ứng dụng khác



Phát triển ứng dụng Web 24

24

3. Cú pháp & Quy ước trong PHP

25

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Xuất dữ liệu
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm

26

Quy ước

- Tất cả các câu lệnh php đều cách nhau bởi dấu ";"
- Không phân biệt khoảng trắng, Tab, xuống dòng trong câu lệnh

```
<?php echo "Hello"; echo " World!"; ?>
<?php
    echo"Hello"      ;
    echo " World!";
?>
```

- Ghi chú : Theo cú pháp ghi chú của C++ & Perl
// Đây là ghi chú
Đây là ghi chú
/* Đây là ghi chú nhiều dòng*/

27

Quy ước

{ }	Khởi lệnh	[]	Sử dụng cho mảng
()	Sử dụng cho hàm	\n, \t	Xuống hàng, ký tự Tab
\'	Ký tự nháy đơn trong chuỗi	\"	Ký tự nháy kép trong chuỗi

- Chuỗi phân biệt trong dấu nháy đơn '' và dấu nháy kép ""

Nháy đơn và nháy kép:

```
$name = "Diddy";  
echo 'Hello, $name'; // Hiển thị: Hello, $name  
echo "Hello, $name"; // Hiển thị: Hello, Diddy
```

28

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Xuất dữ liệu
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm

29

Khai báo biến

- \$ten_bien = value;
- Không khai báo kiểu dữ liệu cho biến (theo cách khai báo thông thường)
- Biến tự động được khởi tạo ở lần đầu tiên gán giá trị cho biến
- Tên biến:
 - Có thể bao gồm các Ký tự (A..Z, a..z), Ký số (0..9), _ , \$
 - Không được bắt đầu bằng ký số (0..9)
 - Phân biệt chữ hoa - chữ thường

Ví dụ:

```
$size, $my_drink_size, $drinks, $drink4you;
```

30

Khai báo biến

• Ví dụ:

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

```
<?php
$name = "Jennie";
$age = 28;
echo "Tên: " . $name . ", Tuổi: " . $age;
//Kết quả: Tên: Jennie, Tuổi: 28
?>
```

• Hằng số - Constants

○ Ví dụ:

```
define("MY_CONST", 10);
echo MY_CONST;
```

• Ví dụ:

```
const MAX_USERS = 100;
echo MAX_USERS; // Kết quả: 100
```

31

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Xuất dữ liệu
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm

32

Xuất dữ liệu

- Dùng hàm echo
- Dùng hàm print
- Dùng var_dump: để xuất kiểu dữ liệu kèm theo dữ liệu của biến

33

Xuất dữ liệu – hàm echo

```
<?php
$txt1 = "Learn PHP";
$txt2 = "UIT";
$x = 5;
$y = 4;

echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;

?>
```

34

Xuất dữ liệu – hàm print

```
<?php
$txt1 = "Learn PHP";
$txt2 = "UIT";
$x = 5;
$y = 4;
print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x + $y;

?>
```

35

Xuất dữ liệu – hàm var_dump

```
<?php
$x = 5985;
var_dump($x);

?>
```

- Kết quả:

```
int(5985)
```

36

Printing HTML tags in PHP = bad style

- In các thẻ HTML với các câu lệnh print là kiểu xấu và dễ xảy ra lỗi
- Phải trích dẫn HTML và sử dụng escape special characters, ví dụ: \"
- Nếu không có bản in, làm thế nào để chúng ta chèn nội dung động vào trang?

```
<?php
print "<!DOCTYPE html>\n";
print "<html>\n";
print "  <head>\n";
print "    <title>Geneva's web page</title>\n";
...
for ($i = 1; $i <= 10; $i++) {
    print "<p class=\"count\"> I can count to $i! </p>\n";
}
?>
```

PHP

Phát triển ứng dụng Web

37

PHP expression blocks

- `<?= expr ?>` tương đương với `<?php print expr; ?>`

• `<?= expression ?>`

PHP

`<h2> The answer is <?= 6 * 7 ?> </h2>`

PHP

The answer is 42
output

Phát triển ứng dụng Web

38

PHP expression blocks

```
<!DOCTYPE html>
<html>
<head><title>CSE 154: Embedded PHP</title></head>
<body>
  <?php for ($i = 99; $i >= 1; $i--) { ?>
    <p> <?= $i ?> bottles of beer on the wall, <br />
    <?= $i ?> bottles of beer. <br />
    Take one down, pass it around, <br />
    <?= $i - 1 ?> bottles of beer on the wall. </p>
  <?php } ?>
</body>
</html>
```

PHP

Phát triển ứng dụng Web

39

PHP expression blocks

Lỗi

```
<body>
  <p>Watch how high I can count:
  <?php for ($i = 1; $i <= 10; $i++) { ?>
    <? $i ?>
  </p>
</body>
</html>
PHP
```

40

PHP expression blocks

```
<body>
  <?php for ($i = 1; $i <= 3; $i++) { ?>
    <h<?= $i ?>>This is a level <?= $i ?> heading.</h<?= $i ?>>
  <?php } ?>
</body>
PHP
```

This is a level 1 heading.
This is a level 2 heading.
This is a level 3 heading.

output

41

Phạm vi biến

- **Có các mức phạm vi:**
 - Global scope (toàn cầu, toàn cục): Biến được khai báo bên ngoài hàm, có thể truy cập ở bất kỳ đâu trong code.
 - Local scope (cục bộ): Biến được khai báo bên trong hàm, chỉ có thể truy cập bên trong hàm đó.
 - Function parameters: Tham số của hàm cũng có phạm vi local trong hàm.
 - Static scope: Biến static bên trong hàm giữ nguyên giá trị giữa các lần gọi hàm.

42

Phạm vi biến

- Global scope:

```
$globalVar = "Tôi là Lý Trung Bình"; // Biến toàn cục

function showGlobalVar() {
    global $globalVar; // Sử dụng từ khóa global để truy cập
    biến toàn cục
    echo $globalVar;
}

showGlobalVar(); // Output: Tôi là Lý Trung Bình
echo $globalVar; // Output: Tôi là Lý Trung Bình
```

43

Phạm vi biến

- Local Scope:

```
function showLocalVar() {
    $localVar = "Tôi là Hồng Hải Nhi"; // Biến cục bộ
    echo $localVar;
}

showLocalVar(); // Output: Tôi là Hồng Hải Nhi
// Gọi biến $localVar ở đây sẽ gây lỗi vì $localVar không
tồn tại ngoài hàm
// echo $localVar; // Lỗi: Undefined variable: localVar
```

44

Phạm vi biến

- Function Parameters:

```
function greet($name) { // $name là tham số của hàm, có phạm vi cục
bộ trong hàm
    echo "Hello, $name!";
}

greet("Lisa"); // Output: Hello, Lisa!
greet("Jisoo"); // Output: Hello, Jisoo!
greet("Tieu Vy"); // Output: Hello, Tieu Vy!

// Gọi $name ngoài hàm sẽ gây lỗi vì $name không tồn tại ngoài hàm
// echo $name; // Lỗi: Undefined variable: name
```

45

Phạm vi biến

- **Static Scope:**

```
function counter() {  
    static $count = 0; // Biến static sẽ giữ nguyên giá trị giữa  
    các lần gọi hàm  
    $count++;  
    echo "Counter: $count<br>";  
}  
  
counter(); // Output: Counter: 1  
counter(); // Output: Counter: 2  
counter(); // Output: Counter: 3
```

46

Biến siêu toàn cục

- Các biến được PHP tự động tạo ra, có thể truy cập ở bất kỳ đâu trong code, bên trong hoặc bên ngoài hàm.
 - `$GLOBALS`: Mảng chứa tất cả các biến global.
 - `$_SERVER`: Chứa thông tin về server và môi trường thực thi.
 - `$_GET`: Chứa dữ liệu được gửi từ form bằng phương thức GET.
 - `$_POST`: Chứa dữ liệu được gửi từ form bằng phương thức POST.
 - `$_REQUEST`: Chứa dữ liệu từ `$_GET`, `$_POST`, và `$_COOKIE`.
 - `$_SESSION`: Lưu trữ thông tin phiên làm việc (session) của người dùng.
 - `$_COOKIE`: Lưu trữ cookie trên máy client.
 - `$_FILES`: Chứa thông tin về file được upload.
 - `$_ENV`: Chứa các biến môi trường.

47

Biến siêu toàn cục

Ví dụ:

```
// Lấy giá trị từ form sử dụng $_POST  
if ($_SERVER["REQUEST_METHOD"] == "POST") {  
    $name = $_POST["name"];  
    $email = $_POST["email"];  
    echo "Tên: " . $name . ", Email: " . $email;  
}
```

```
// Lấy thông tin từ $_SERVER  
echo "Địa chỉ IP của client: " . $_SERVER["REMOTE_ADDR"];
```

48

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Xuất dữ liệu
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm

49

Kiểu dữ liệu

- boolean (bool) – kiểu luận lý
- integer (int) – kiểu số nguyên
- double (float, real) – kiểu số thực
- string – kiểu chuỗi lý tự
- array – kiểu mảng
- object – kiểu đối tượng

❖ 1 Biến trong PHP có thể lưu **bất kỳ kiểu dữ liệu** nào

50

Kiểu dữ liệu (tt)

- Chuyển kiểu dữ liệu
 - Cách 1 (automatic)
\$var = "100" + 15;
\$var = "100" + 15.0;
\$var = 39 . " Steps";
 - Cách 2: (datatype) \$var
 - Cách 3: settype(\$var, "datatype")

\$var	(int)\$var	(bool)\$var	(string)\$var
null	0	flase	""
true	1		"1"
false	0		""
"6 feet"	6	true	
"foo"	0	true	

51

Kiểu dữ liệu (tt)

- PHP ban đầu là một ngôn ngữ loosely typed (kiểu dữ liệu lỏng), nghĩa là **không cần phải khai báo kiểu dữ liệu khi khai báo biến**. Điều này làm cho PHP trở thành một ngôn ngữ rất dễ học và linh hoạt, vì lập trình viên không cần lo lắng về việc kiểm soát kiểu dữ liệu khi viết mã.
- Lý do: PHP được thiết kế để tạo ra các trang web động một cách nhanh chóng và dễ dàng. Tập trung vào việc xử lý nội dung HTML, cơ sở dữ liệu, và các tác vụ của trang web mà không quá quan tâm đến kiểu dữ liệu.
- Nhược điểm: Khó khăn trong việc bảo trì. Lỗi khó phát hiện. Giảm hiệu suất.

52

Kiểu dữ liệu (tt)

- PHP 7 là một bước ngoặt quan trọng trong việc phát triển ngôn ngữ PHP, với việc **giới thiệu khai báo kiểu dữ liệu**.
- Khai báo kiểu dữ liệu cho các tham số của hàm và phương thức (int, float, string, bool, array, callable) - Scalar Type Declarations
- Khai báo kiểu dữ liệu cho giá trị trả về của hàm - return type declarations
- Chế độ **strict typing** để đảm bảo rằng kiểu dữ liệu phải chính xác (tránh việc tự động chuyển đổi kiểu dữ liệu)
- Ví dụ Scalar Type Declarations
function addNumbers(int \$a, int \$b): int {
 return \$a + \$b;
}

53

Kiểu dữ liệu (tt)

- **Strict typing**: Cần thêm dòng lệnh declare(strict_types=1); ở đầu tập tin PHP
declare(strict_types=1); // Bật strict typing
function addNumbers(int \$a, int \$b): int {
 return \$a + \$b;
}
echo addNumbers(5, "10"); // Lỗi: Chuỗi không thể chuyển đổi thành số nguyên

54

Kiểu dữ liệu (tt)

Ví dụ **Scalar Type Declarations**:

```
<?php
declare(strict_types=1);

function addNumbers(int $a, int $b): int {
    return $a + $b;
}

var_dump(addNumbers(5, 10)); // int(15)
var_dump(addNumbers(5, "10")); // Lỗi!

?>
```

55

Kiểu dữ liệu (tt)

Ví dụ **Return Type Declarations**:

```
<?php
declare(strict_types=1);

function getFullName(string $firstName, string $lastName): string {
    return $firstName . " " . $lastName;
}

var_dump(getFullName("Liễu Như", "Yên")); // string(12) "Liễu Như Yên"
var_dump(getFullName("Beyonce", 123)); // Lỗi!

?>
```

56

Kiểu dữ liệu (tt)

- **Lý do:** Ban đầu, PHP được dùng cho các dự án nhỏ, nơi mà **tính linh hoạt** là ưu tiên. Tuy nhiên, khi PHP phát triển thành ngôn ngữ chủ đạo cho các dự án lớn (như các framework Laravel, Symfony), việc kiểm soát chặt chẽ kiểu dữ liệu trở nên quan trọng để tránh lỗi và tăng tính bảo trì.

57

Kiểu dữ liệu (tt)

• Nullable Types trong PHP 7.1

- Nullable Types giúp khai báo rằng một biến có thể là kiểu dữ liệu cụ thể (ví dụ: int, string, float,...) hoặc là null. Khi sử dụng Nullable Types, chỉ cần thêm dấu ? trước kiểu dữ liệu.

• Ví dụ:

```
function getUserId(?int $id): ?int {  
    return $id;  
}  
var_dump(getUserId(123)); // int(123)  
var_dump(getUserId(null)); // NULL
```

Tham số \$id có thể là kiểu int hoặc null. Hàm getUserId() có thể trả về một giá trị kiểu int hoặc null.



58

Kiểu dữ liệu (tt)

• Lợi ích:

- Xử lý các giá trị không xác định
- Tăng tính an toàn và bảo trì: Bằng cách khai báo rõ ràng rằng một tham số hoặc giá trị trả về có thể là null, giúp mã dễ bảo trì hơn và ngăn chặn các lỗi liên quan đến dữ liệu không xác định.
- Tăng tính rõ ràng: Khi khai báo Nullable Types, developer xác định rõ ràng rằng một biến có thể là null hoặc một kiểu dữ liệu cụ thể.



59

Kiểu dữ liệu (tt)

Ví dụ trước PHP 7.1: Trong trường hợp này, không có khai báo kiểu dữ liệu, vì vậy phải kiểm tra xem giá trị \$name có phải là null hay không. Điều này dễ dẫn đến lỗi, đặc biệt khi làm việc với mã phức tạp.

```
function getName($name) {  
    if ($name === null) {  
        return "No name provided";  
    }  
    return $name;  
}
```



60

Kiểu dữ liệu (tt)

Ví dụ **Nullable Types**: Khi truy vấn dữ liệu từ cơ sở dữ liệu, có thể một dòng không tồn tại hoặc một ô không có giá trị.

```
function getUserEmail(?string $email): ?string {  
    if ($email === null) {  
        return "No email provided";  
    }  
    return $email;  
}  
echo getUserEmail(null); // Output: No email provided
```

61

Kiểu dữ liệu (tt)

Ví dụ **Nullable Types**: Khi gọi API từ bên ngoài, có thể xảy ra trường hợp API không trả về giá trị mong đợi hoặc trả về null

```
function fetchDataFromApi(?array $data): ?array {  
    if ($data === null) {  
        return null; // Trả về null nếu không có dữ liệu từ API  
    }  
    return $data;  
}
```

62

Kiểu dữ liệu (tt)

- PHP 8 tiếp tục phát triển dựa trên những cải tiến của PHP 7, với sự xuất hiện của **Union Types**. Union Types cho phép một tham số hoặc giá trị trả về có thể thuộc nhiều kiểu dữ liệu khác nhau, điều này giúp mã nguồn linh hoạt hơn mà vẫn giữ được tính rõ ràng và an toàn.

- Lý do PHP 8 giới thiệu Union Types:

- Tăng tính linh hoạt
- Giảm sự phụ thuộc vào kiểu dữ liệu lỏng
- Tăng khả năng kiểm soát

```
function calculate(int|float $number): int|float {  
    return $number * 2;  
}
```

```
var_dump(calculate(5)); // int(10)  
var_dump(calculate(2.5)); // float(5)
```

63

Kiểu dữ liệu (tt)

- PHP 8 cũng có **Mixed Types**: Kiểu mixed trong PHP 8 cho phép biến có thể nhận **bất kỳ kiểu dữ liệu nào**, nhưng không cần liệt kê từng loại trong Union Types. Sử dụng khi không thể xác định trước kiểu dữ liệu, hoặc khi hàm cần xử lý nhiều kiểu dữ liệu khác nhau.

- Ví dụ:

```
function process(mixed $data) {  
    // $data có thể là int, string, array, hoặc object...  
}  
processData(123); // int(123)  
processData("Hello"); // string(5) "Hello"  
processData(array(1, 2, 3)); // array(3) { ... }
```



64

Các hàm liên quan

- Kiểm tra kiểu dữ liệu

Hàm	Ý nghĩa
gettype	Lấy kiểu dữ liệu
is_integer	Kiểm tra kiểu dữ liệu nguyên
is_double	Kiểm tra kiểu dữ liệu số thực
is_string	Kiểm tra kiểu dữ liệu chuỗi
is_array	Kiểm tra có phải kiểu dữ liệu mảng
is_object	Kiểm tra kiểu dữ liệu đối tượng
isset	Kiểm tra sự tồn tại của biến
unset	Hủy một biến
empty	Kiểm tra một biến có rỗng hay không



65

Các hàm liên quan – ví dụ 1

```
<?php  
$a=9.6;  
$b=(int)$a;  
echo "b=".$b;  
if(empty($b))  
    echo 'biến $b rỗng';  
else  
    echo 'giá trị biến $b  
='.$b;  
?>
```

❖ Kết quả:

b=9
giá trị biến \$b =9



66

Các hàm liên quan – ví dụ 2

```
<?php
$b="";
if(empty($b))
    echo 'biến $b rỗng';
else
    echo '<br>giá trị biến $b ='.$b;
if(isset($b))
    echo '<br>có biến $b';
else
    echo '<br>không có biến $b';
?>
```

❖ Kết quả: biến \$b rỗng
có biến \$b



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 67

67

Các hàm xử lý trên số

• Một số hàm xử lý số

- **abs** **pow** **decbin** **srand(seed)**
- **ceil** **sqrt** **bindec** **rand**
- **floor** **log** **dechex** **rand(min, max)**
- **round** **log10** **hexdec**
- **pi ...**



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 68

68

Các hàm xử lý trên số _ Ví dụ

```
<?php
echo(pi()); // returns 3.1415926535898
echo(min(0, 150, 30, 20, -8, -200)); // returns -200
echo(max(0, 150, 30, 20, -8, -200)); // returns 150
echo(abs(-6.7)); // returns 6.7
echo(sqrt(64)); // returns 8
echo(round(0.60)); // returns 1
echo(round(0.49)); // returns 0
echo(rand());
echo(rand(10, 100));
?>
```



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 69

69

Kiểu chuỗi - string

- Toán tử nối chuỗi : dấu chấm .

```
$s = "Hello" . " World"; // $s = "Hello World"
```

- Phân biệt dấu nháy đơn và nháy kép

```
$user = "Bill";  
echo 'Hi $user'; // Hi $user  
echo "Hi $user"; // Hi Bill  
echo 'Hi' . $user; // ???  
echo 'Hi' . 'user'; // ???
```

- Một số hàm xử lý chuỗi

◦ printf	trim	strtolower
◦ str_pad	str_replace	strtoupper
◦ strlen	substr	strcasecmp



Phát triển ứng dụng Web 70

70

Kiểu chuỗi - string

- **printf()**: Xuất dữ liệu theo định dạng cụ thể. Thường được dùng để định dạng chuỗi hoặc số theo cách tùy chỉnh.

```
printf("Số %.2f", 3.14159); // Output: Số 3.14
```

- **trim()**: Loại bỏ khoảng trắng (hoặc ký tự xác định) từ đầu và cuối chuỗi.

```
$str = " Liều Như Yên ";  
echo trim($str); // Output: "Liều Như Yên"
```

- **str_pad()**: Thêm ký tự vào đầu hoặc cuối chuỗi đến khi đạt độ dài mong muốn.

```
echo str_pad("PHP", 6, "-"); // Output: "PHP---"
```

- **strcasecmp()**: So sánh hai chuỗi không phân biệt chữ hoa chữ thường. Trả về 0 nếu chúng bằng nhau.

```
echo strcasecmp("Hello", "hello"); // Output: 0
```



Phát triển ứng dụng Web 71

71

Ví dụ định dạng xuất số thực

```
<?php  
$a = 10.123456;  
printf( "0.2f<br>", $a );  
$a = number_format($a, 2);  
echo $a;  
?>
```

➤ Kết quả

```
10.12  
10.12
```



Phát triển ứng dụng Web 72

72

Ví dụ nối chuỗi

```
<?php
    $x=2.5;
    $nghiem="Phuong trinh co  nghiệm:". $x;

    echo $nghiem;
?>
```

❖ Kết quả: **Phương trình có nghiệm:2.5**

73

Các hàm liên quan đến string – ví dụ

```
<?php
    echo strlen("Hello world!"); // outputs 12
    echo str_word_count("Hello world!"); // outputs 2
    echo strrev("Hello world!"); // outputs !dlrow olleH

    echo strpos("Hello world!", "world"); // outputs 6
    echo str_replace("world", "UIT", "Hello world!");
    // outputs Hello UIT!
?>
```

74

Mảng - array

- Khai báo
\$a=array();
- Khai báo và khởi tạo
\$words = array("Web", "Database", "Applications");
- Truy xuất phần tử của mảng
echo \$words[0];
- Đặt tên chỉ số của mảng
\$numbers = array(1=>"one", "two", "three", "four");
 - ❖ echo \$numbers[1];
 - \$array = array("first"=>1, "second"=>2, "third"=>3);
 - ❖ echo \$array["second"];

75

Ví dụ

```
<?php
$a = array();
$a[0]=1;
$a[1]=2;
$a[2]="Tâm";
$a[3]="Tú";
for ($i=0; $i<count($a); $i++)
{
    echo $a[$i]."<br>";
}
?>
```

```
1
2
Tâm
Tú
```

76

Duyệt mảng

```
<?php
$ten=array("Tú","Tùng","Tâm");
for ($i=0;$i<count($ten);$i++)
    echo $ten[$i]."<br>";
?>
```

```
<?php
$page = array("Tú"=>"35", "Tùng"=>"37", "tâm"=>"43");
foreach ($page as $key => $value)
{
    echo "Key=" . $key . ", Value=" . $value;
    echo "<br>";
}
?>
```

```
Key=Tú, Value=35
Key=Tùng, Value=37
Key=Tâm, Value=43
```

77

Ví dụ khởi tạo mảng và dùng chỉ số phần tử

```
<?php
$page = array();
$page["Tâm"]=32;
$page["Tú"]=22;
foreach ($page as $key=>$value)
{
    echo $key."\"t\".$value."<br>";
}
?>
```

```
Tâm 32
Tú 22
```

78

Mảng 1 chiều – single arrays

- Một số hàm xử lý trên mảng

- `count`
 - `min`
 - `max`
- Ví dụ:**

```
<?php
$ten=array("Tú","Tùng","Tâm");
sort($ten);
for($i=0;$i<count($ten);$i++)
echo $ten[$i]."<br>";
?>
```

Tâm
Tùng
Tú

79

Ví dụ về sắp xếp giảm

```
<?php
$num=array(1,2,3,4,5);
rsort($num);
for($i=0;$i<count($num);$i++)
echo $num[$i] . "<br>";
?>
```

➤ Kết
quả:

5
4
3
2
1

80

Một số hàm liên quan đến mảng

- `array_push(array, elements)` : Thêm elements vào cuối mảng
- `array_pop(array)` : Lấy phần tử cuối ra khỏi mảng
- `array_unshift(array, elements)` : Thêm elements vào đầu mảng
- `array_shift(array)` : Lấy phần tử đầu ra khỏi mảng
- `array_merge(array, array)` : kết 2 mảng lại và trả ra mảng mới
- `shuffle(array)` : Sort random mảng
- `sort(array, flag)` : `flag = {sort_regular, sort_numeric, sort_string, sort_locale_string}`

81

Mảng 2 chiều

- Khai báo mảng 2 chiều

- Khai báo và tạo mảng 2 chiều có 2 dòng, mỗi dòng chứa 3 cột

```
$a=array();
```

```
for($i=0;$i<2;$i++)
```

```
    $a[$i]=array();
```

82

Ví dụ mảng 2 chiều – cách 1

```
<?php
    $lop=array();
    $a1=array("1125001","Tuấn",7.5);
    $a2= array("1125002","Tùng",8.0);
    $a3 = array("1125003","Tâm",7.0);
    $a4 = array("1125003","Tú",6.5);
    $lop[0] = $a1;
    $lop[1] = $a2;
    $lop[2] = $a3;
    $lop[3] = $a4;
?>
```

83

Ví dụ mảng 2 chiều – cách 1

```
<?php echo "<table border='1' bordercolor='#0000FF'
cellspacing='0'>";
echo "<tr><th>Mã số</th><th>Tên</th><th>Điểm trung bình</th>";
for($i=0;$i<count($lop);$i++)
{
    echo "<tr>";
    for($j=0;$j<count($lop[$i]);$j++){
        echo "<td>".$lop[$i][$j]."</td>";
    }
    echo "</tr>";
}
echo "</table>";
?>
```

Mã số	Tên	Điểm trung bình
1125001	Tuấn	7.5
1125002	Tùng	8
1125003	Tâm	7
1125003	Tú	6.5

84

Mảng 2 chiều – cách 2

```
<?php
$lop=array(
    array("1125001","Tuần",7.5),
    array("1125002","Tùng",8.0),
    array("1125003","Tâm",7.0),
    array("1125003","Tú",6.5) );
echo "<table border='1' bordercolor='#CC00FF' cellspacing='0'>";
echo "<tr><th>Mã số</th><th>Tên</th><th>Điểm trung bình</th>";
for($i=0;$i<count($lop);$i++){
    echo "<tr>";
    for($j=0;$j<count($lop[$i]);$j++){
        echo "<td>".$lop[$i][$j]."</td>";
    }
    echo "</tr>";
}
echo "</table>";
?>
```

Mã số	Tên	Điểm trung bình
1125001	Tuần	7.5
1125002	Tùng	8
1125003	Tâm	7
1125003	Tú	6.5

85

Ví dụ mảng 2 chiều

```
<?php
$lop= array(array("1125001","Tuần",7.5),
    array("1125002","Tùng",8.0),
    array("1125003","Tâm",7.0),
    array("1125003","Tú",6.5) );
echo "<table border='1' bordercolor='#CC00FF' cellspacing='0'>";
echo "<tr><th>Mã số</th><th>Tên</th><th>Điểm trung bình</th>";
for($i=0;$i<count($lop);$i++){
    echo "<tr>";
    foreach($lop[$i] as $key=>$value) {
        echo "<td>".$value."</td>";
    }
    echo "</tr>";
}
echo "</table>";
```

Mã số	Tên	Điểm trung bình
1125001	Tuần	7.5
1125002	Tùng	8
1125003	Tâm	7
1125003	Tú	6.5

86

Đặt và lấy múi giờ

- Đặt múi giờ
 - `date_default_timezone_set("Asia/Ho_Chi_Minh");`
- Lấy múi giờ
 - `$timezone = date_default_timezone_get();`

87

Các hàm thời gian

- Lấy ngày hệ thống: date("định dạng")

Định dạng	Mô tả	Giá trị
d	Ngày trong tháng, lấy 2 ký tự	01-31
D	Ngày trong tuần, lấy 3 ký tự	Mon-Sun
J	Ngày trong tháng, lấy hai ký tự, không lấy số 0 đứng trước	1-31
S	Hàng tử của ngày trong tháng	0-365
z	Ngày trong năm	0-365
w	Ngày trong tuần theo số	0: Sunday - 6: Saturday
W	Tuần trong năm	1 - 52
F	Tháng trong năm dạng chữ viết	January
m	Tháng trong năm, lấy 2 ký tự	01 - 12
N	Tháng trong năm, lấy 3 ký tự	Jan - Dec
n	Tháng trong năm không lấy số 0	1 - 12
t	Số ngày tối đa trong tháng	28 - 31
L	Năm nhuận có giá trị 1, thường có giá trị 0	0, 1
Y	Năm viết đầy đủ	2014
y	Lấy 2 ký tự cuối của năm	14

Phát triển ứng dụng Web

88

Định dạng giờ, phút, giây

Định dạng	Mô tả	Giá trị
a	Buổi trong ngày chữ thường	am-pm
A	Buổi trong ngày chữ hoa	AM-PM
g	Định dạng 12h không có số 0 đứng đầu	1-12
G	Định dạng 24h không có số 0 đứng đầu	0-24
h	Định dạng 12h có số 0 đứng đầu	01-12
H	Định dạng 24h có số 0 đứng đầu	00-23
i	Định dạng phút	00-59
s	Định dạng giây	00-59

Phát triển ứng dụng Web

89

Ví dụ

```
<?php
echo date("g:m:s a d-m-y");
?>
```

Kết quả:

2:10:28 am 27-10-14

Phát triển ứng dụng Web

90

Lấy ngày hiện tại

- Dùng hàm : array getdate();
- Hàm trả về một mảng gồm 10 phần tử có chỉ số là chuỗi chứa các giá trị của ngày hiện tại. Các chỉ số của mảng theo định dạng sau:

91

Các chỉ số của mảng

Chỉ số	Ý nghĩa
[seconds]	Chỉ số lấy giá trị giây
[minutes]	Chỉ số lấy giá trị phút
[hours]	Chỉ số lấy giá trị giờ
[mday]	Ngày trong tháng
[wday]	Ngày trong tuần 0 -> chủ nhật...
[weekday]	Thứ trong tuần
[mon]	Tháng 1-12
[month]	Tháng dạng chuỗi
[year]	Năm
[yday]	Ngày trong năm

92

Ví dụ

```
<?php
$m=getdate();
$th=array("Monday"=>"Thứ 2", "Tuesday"=>"Thứ 3",
"Wednesday"=>"Thứ 4", "Thursday"=>"Thứ 5",
"Friday"=>"Thứ 6", "Saturday"=>"Thứ 7", "Sunday"=>"Chủ nhật");
$thutrongtuan=$m['weekday'];
echo "hôm nay là: ".$th[$thutrongtuan]. " ngày
". $m['mday']. " Tháng ". $m['mon']. " Năm ". $m['year'];
?>
```

Kết quả:
hôm nay là: Chủ nhật ngày 26 Tháng 10 Năm 2014

93

Tổng số giây từ 1/1/1970

- Dùng hàm : `long time()`
- Ví dụ:

```
<?php
    echo time();
?>
```

Kết quả: 1414296069

94

Chuyển chuỗi thành thời gian



- `long strtotime(chuỗi);`

```
<?php
    echo "Tổng số giây từ 1-1-1970 đến ";
    //Lấy ngày sau khi đổi chuỗi ra ngày
    $t=getdate(strtotime("October 14 2014"));
    echo "Ngày ". $t["mday"]. "Tháng ". $t["mon"]. " Năm
    ". $t["year"]. " là ";
    //in tổng số giây từ 1-1-1970 đến ngày cần chuyển
    echo strtotime("October 14 2014");
?>
```

Kết quả:
Tổng số giây từ 1-1-1970 đến Ngày 14 Tháng 10 Năm 2014
là 1413244800

95

Lấy ngày từ control "date"

01/06/2015  

2015-01-06
Ngày 6 Tháng 1 Năm 2015

96

Code lấy ngày từ control

```
<body>
<form method="GET">
<input type="date" name="date"></input>
<input type="Submit" value="GetDate" name="getdate">
</form>
<?php
if (isset($_GET['getdate'])) && ($_GET['getdate'])=='GetDate')
{
    $date=$_GET['date'];
    echo $date."<br>";
    $m=getdate(strtotime($date));
    echo "Ngày ".$m['mday']." Tháng ".$m['mon']." Năm ".$m['year'];
}
}??
</body>
```

Phát triển ứng dụng Web

97

97

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm

Phát triển ứng dụng Web

98

98

Toán tử

Loại	Toán tử	Ghi chú
	new .	
	. [] ()	
Toán học	+, -, *, **, /, %, ++, --	
So sánh	< > <= >= != == === !==	
Logic	&& ! ? :	
Xử lý bit	! ~ << >> >>> AND OR XOR	
Gán	= += -= *= /= %= >>= <<= &= = ^= ,=	
Ép kiểu	(kiểu dữ liệu)	(int) (double) (string)...

Phát triển ứng dụng Web

99

99

Ví dụ

```
<?php
    $x = 2;
    $y = 3;
    echo $x ** $y; // 2*2*2=8
?>
```

```
<?php
    $x = 100;
    $y = "100";
    if($x === $y)
        echo "true";
    else
        echo "false"; // out put false
?>
```

100

Ví dụ

```
<?php
    $x = 100;
    $y = "100";
    if($x !== $y)
        echo "true"; // vì 2 kiểu dữ liệu khác nhau
    else
        echo "false";
?>
```

```
<?php
echo $user = $_GET["user"] ?? "no user"; //no user
$size=40;
echo $s = $size ?? "41"; //40
?>
```

101

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm

102

Cấu trúc điều khiển

- Điều kiện if, if...else
- Điều khiển switch
- Vòng lặp for
- Vòng lặp while
- Vòng lặp do.. while
- Vòng lặp foreach

103

Điều kiện if

```
if (condition)
{
    statement[s] if true
}
else //(condition)
{
    statement[s] if false
}
```

Ví dụ:

```
$x = 5;
if ($x < 4)
    echo "$x is less than 4";
else
    print '$x isn't less than
4';
```

\$x isn't less than 4

104

Điều khiển switch

```
switch (expression)
{
    case label 1:
        statementlist
        break;
    case label 2:
        statementlist
        break;
    ...
    default:
        statementlist
}
```

You picked threeYou picked four

```
Ví dụ:
$menu = 3;
switch ($menu) {
    case 1:
        echo "You picked one";
        break;
    case 2:
        echo "You picked two";
        break;
    case 3:
        echo "You picked three";
        //break;
    case 4:
        echo "You picked four";
        break;
    default:
        echo "You picked another option";
}
```

105

Điều khiển switch

- Có một số hạn chế:
 - Fall-through: Nếu không sử dụng break, PHP sẽ tiếp tục thực thi các trường hợp tiếp theo, điều này đôi khi dẫn đến lỗi logic không mong muốn. Không trả về giá trị trực tiếp:
 - Switch chỉ là một khối logic điều kiện, không phải một biểu thức, nên không thể trực tiếp gán kết quả của switch vào biến.
 - So sánh chỉ theo giá trị: Switch chỉ hỗ trợ so sánh giá trị không theo kiểu dữ liệu nghiêm ngặt (không kiểm tra kiểu dữ liệu)

106

Điều khiển match

- Có từ PHP 8, là một sự thay thế mạnh mẽ và ngắn gọn hơn cho cấu trúc switch

```
Ví dụ:  
$status = 200;  
$message = match ($status) {  
    200 => 'OK',  
    404 => 'Not Found',  
    500 => 'Internal Server Error',  
};
```

- Match khắc phục nhiều hạn chế của switch: Không có fall-through: Không cần phải sử dụng break vì match chỉ kiểm tra một trường hợp và trả về kết quả. Match là một biểu thức: Match trả về giá trị trực tiếp, có thể gán giá trị của match vào biến. So sánh chặt chẽ kiểu dữ liệu ...

107

Điều khiển match

- Ví dụ:

```
$day = 'Monday';  
  
$result = match ($day) {  
    'Monday' => 'Start of the work week',  
    'Friday' => 'End of the work week',  
    default => 'Middle of the week',  
};  
  
echo $result;
```

108

Điều khiển match

- So sánh cú pháp switch và match

```
$message = match ($status) {  
    200 => 'OK',  
    404 => 'Not Found',  
    500 => 'Server Error',  
    default => 'Unknown',  
};
```

```
switch ($status) {  
    case 200:  
        $message = 'OK';  
        break;  
    case 404:  
        $message = 'Not Found';  
        break;  
    case 500:  
        $message = 'Server Error';  
        break;  
    default:  
        $message = 'Unknown';  
}
```

109

Điều khiển match

- Match thực hiện so sánh kiểu dữ liệu nghiêm ngặt (strict comparison) bằng cách sử dụng toán tử === thay vì chỉ so sánh giá trị như switch. Điều này ngăn chặn các lỗi tiềm ẩn khi kiểu dữ liệu không khớp

```
$value = '1';  
  
$result = match ($value) {  
    1 => 'Number one',  
    '1' => 'String one',  
};  
  
echo $result; // Output: String one (vì match so sánh chặt chẽ và kiểm tra  
kiểu dữ liệu)
```

110

Điều khiển match

- Match cho phép gộp nhiều điều kiện vào cùng một dòng, giúp mã nguồn gọn hơn.

```
$role = 'admin';  
  
$access = match ($role) {  
    'admin', 'superadmin' => 'Full Access',  
    'editor' => 'Edit Access',  
    'viewer' => 'Read-Only Access',  
    default => 'No Access',  
};  
  
echo $access; // Output: Full Access
```

111

Vòng lặp for

```
for([initial expression];[condition];[update expression])
{
    statement[s] inside loop
}
```

• Ví dụ:

```
echo "<select>";
for ($i = 1; $i <= 12; $i++) {
    echo "<option>$i</option>";
}
echo "</select>";
```



112

Vòng lặp while, do...while

```
while (expression)
{
    statements
}
```

```
do
{
    statements
}while (expression);
```

113

Vòng lặp while, do...while

- **while:** Kiểm tra điều kiện trước khi thực hiện đoạn mã. Nếu điều kiện sai ngay từ đầu, vòng lặp sẽ không chạy lần nào.
- **do...while:** Thực hiện đoạn mã ít nhất một lần trước khi kiểm tra điều kiện. Vòng lặp sẽ luôn chạy ít nhất một lần, ngay cả khi điều kiện sai từ đầu.

114

Vòng lặp while, do...while

Ví dụ:

```
$i = 1; $j = 9;

do {
    $temp = $i * $j;
    echo "$j * $i = $temp<br>";
    $i++;
} while ($i <= 10);
```

?

Ví dụ:

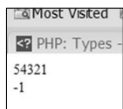
```
$i = 1; $j = 9;

while ($i <= 10)
{
    $temp = $i * $j;
    echo "$j * $i = $temp<br>";
    $i++;
}
```

```
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
```

115

Vòng lặp while, do...while



```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>PHP - Câu Trắc Dạng Nhies</title>
</head>
<body>
<?php
    $i = 5;
    while ($i > 0)
    {
        echo ($i--);
    }

    print('<br>');

    do
    {
        echo (--$i);
    }
    while ($i > 0);
?>
</body>
</html>
```

116

116

Vòng lặp foreach

```
foreach (array as variable)
{
    statements
}
```

Ví dụ:

```
<?php
    $sp = array('Mã sản phẩm' => 'SP001',
                'Tên sản phẩm' => 'NOKIA 720',
                'Giá bán' => 5000000);
    echo "<table border='1' bordercolor='#CC00FF' cellspacing='0'>";
    foreach ($sp as $key => $value) {
        echo "<tr><td>$key</td><td>$value</td></tr>";
    }
    echo "</table>";
```

Mã sản phẩm	SP001
Tên sản phẩm	NOKIA 720
Giá bán	5000000

117

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm

118

Hàm - function

- Khai báo với từ khoá function
- Không cần kiểu trả về
- Nên khai báo ở đầu file PHP hoặc file riêng.

119

Hàm - function

```
function functionName  
([parameter1]...[,parameterN])  
{  
    statement[s] ;  
}  
function functionName  
([parameter1]...[,parameterN])  
{  
    statement[s] ;  
    return .... ;  
}
```

120

Hàm - ví dụ

```
<?php
function Tong($a,$b)
{
    $s=$a+$b;
    return $s;
}
$t=Tong(5,6);
echo "Tổng =".$t;
?>
```

Tổng =11

121

Hàm - Phạm vi biến

```
<?php
function doublevalue($var=10)
{
    global $temp;
    $temp = $var * 2;
}

$temp = 5;
doublevalue();
echo "\$temp is: $temp";
?>
```

\$temp is: 5

\$temp is: 20

122

Hàm - Tham trị và Tham biến

```
<?php
function doublevalue(&$var)
{
    $var = $var * 2;
}

$a = 5;
doublevalue($a);
echo "\$a is: $a";
?>
```

\$variable is: 5

\$a is: 10

- Khi gọi doublevalue(\$a): giá trị của \$a được sao chép và truyền vào hàm doublevalue(), tạo ra một bản sao của \$a trong tham số \$var
- Bên trong hàm, \$var được nhân đôi, nhưng do chỉ là một bản sao, sự thay đổi này không ảnh hưởng đến giá trị của \$a bên ngoài hàm
- Bằng cách thêm dấu & trước tên tham số trong khai báo hàm, \$a sẽ bị thay đổi trực tiếp bên trong hàm và khi in ra, giá trị của \$a sẽ là 10.

123

Hàm – include & require

- Những file PHP (include và require): Cả include và require đều được sử dụng để nhúng code PHP từ một file khác vào file hiện tại.
- Sự khác biệt:
 - include: Nếu file không tồn tại, sẽ chỉ hiện warning và code tiếp tục thực thi.
 - require: Nếu file không tồn tại, sẽ dừng thực thi code và hiện fatal error.
- include_once và require_once:
 - Chỉ nhúng file một lần duy nhất, tránh lặp code.

124

Hàm – include & require

```
❖ khaibao.php
<?php
function InLuong($ten, $luong)
{
    echo "<table bordercolor='#CC00FF'
        border='1' cellspacing='0'>"; echo
        "<tr><th>Tên</th><th>Luong</th>";
    echo "<tr><td>". $ten. "</td>";
    echo "<td>". $luong. "</td>";
    echo "</tr>";
    echo "</table>";
}
?>
```

```
❖ goiInclude.php
<?php
$ten="Nguyễn Xuân Tú";
$luong=7000000;
include "khaibao.php";
InLuong($ten,$luong);

?>
```

125

Hàm – include & require

- **Giống nhau:**
 - Chèn file vào file hiện tại, nếu file được chèn có lỗi thì hiện thông báo lỗi
- **Khác nhau:**
 - Khi file được chèn bằng lệnh **require()** có lỗi thì trình biên dịch sẽ dừng lại, không dịch nữa và sẽ xuất hiện thông báo lỗi.
 - Khi file được chèn bằng lệnh **include()** có lỗi thì trình biên dịch vẫn tiếp tục dịch cho đến hết, đồng thời cũng xuất hiện **warning** để cảnh báo file đó bị lỗi.

126

Hàm – include & require

- **Ví dụ:** Giả sử 2 đoạn chương trình sau cùng sử dụng tập tin **require_include.php** không tồn tại như sau:

```
<html>
<head><title>Test include</title></head>
<body>
<?php
include("require_include.php"); echo "Hello you";
?>
</body>
</html>
```

```
<html>
<head><title>Test include</title></head>
<body>
<?php
require("require_include.php"); echo "Hello you";
?>
</body>
</html>
```



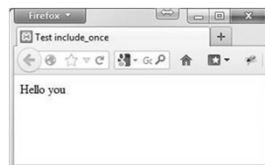
127

Hàm – include & require

- Hàm **include_once** và **require_once**
- Là 2 hàm biến đổi của hàm include và require nhằm mục đích nếu tập tin đã được chèn trước đó thì không chèn nữa.
- **Ví dụ:** giả sử có tập tin **require_include_once.php** như sau:

```
<?php
// Tập tin require_include_once.php
echo "Hello you <br>";
?>
```

```
<!-- Tập tin test_require_include_once.php -->
<?php
include_once("require_include_once.php");
include_once("require_include_once.php");
?>
```



128

Hàm – include & require

- **__DIR__** là hằng số magic, chứa đường dẫn tuyệt đối đến thư mục của file hiện tại.
- Sử dụng **__DIR__** để include/require file một cách an toàn, tránh lỗi khi đường dẫn tương đối thay đổi
- Ví dụ:
`require_once __DIR__ . '/my_functions.php';`

129

Hàm – Arrow Functions

- Arrow functions (hàm mũi tên) là cú pháp rút gọn cho hàm ẩn danh trong PHP, loại bỏ việc cần dùng từ khóa function và return
- Có từ PHP 7.4
- Cú pháp: `fn (parameters) => expression;`
- Ví dụ:

```
$multiply = fn($a, $b) => $a * $b;  
$sum = fn($a, $b) => $a + $b;
```
- So sánh với hàm ẩn danh (Anonymous function):

```
$multiply = function($a, $b) {  
    return $a * $b;  
};
```

130

Hàm – Arrow Functions

- Trong arrow functions, this không bị ràng buộc với object hiện tại như trong hàm thông thường.
- this trong arrow functions tham chiếu đến ngữ cảnh (context) lúc hàm được tạo ra.

```
class Person {  
    public $name = "Lisa";  
  
    public function greet() {  
        $greetArrow = fn() => echo "Hello,  
my name is " . $this->name;  
        $greetArrow();  
    }  
}
```

```
$person = new Person();  
$person->greet(); // Output: Hello,  
my name is Lisa  
  
$greetFunction = $person->greet;  
// Gán method greet vào biến  
$greetFunction();  
// Output: Hello, my name is Lisa
```

131

4. Lập trình hướng đối tượng

132

Khai báo lớp

```
<?php
class TênLớp
{
    Phạm vi truy xuất Thuộc tính;
    Phạm vi truy xuất Phương thức của lớp;
}
?>
```

133

Khai báo lớp

- public: Có thể truy cập từ bất kỳ đâu.
- protected: Chỉ có thể truy cập bên trong class và class con (kế thừa).
- private: Chỉ có thể truy cập bên trong class.
- Ví dụ:

```
class MyClass {
    public $publicProperty;
    protected $protectedProperty;
    private $privateProperty;
}
```

134

Khai báo lớp

- Từ PHP 7.4, có thể gán kiểu dữ liệu cho các **thuộc tính** của lớp. Gọi là Typed Properties, giúp lập trình hướng đối tượng chặt chẽ hơn

```
<?php
class User {
    public string $name;
    public int $age;
}

$user = new User();
$user->name = 'Thành An'; // Hợp lệ
$user->age = 23;           // Hợp lệ
?>
```

135

Khai báo lớp

- Ví dụ Typed Properties: Typed properties đảm bảo rằng giá trị gán cho thuộc tính phải khớp với kiểu dữ liệu đã định nghĩa, từ đó tăng tính an toàn và dễ bảo trì.

```
class Product {  
    public string $name;  
    public float $price;  
}  
  
$product = new Product();  
$product->name = "Laptop";  
$product->price = 1500.99;
```

136

Tạo ra đối tượng thuộc lớp

```
<?php  
  
    $đoituong = new TênLớp;  
  
    //Truy xuất đến các thành phần của đối tượng  
    $đoituong ->Tên Phương Thức;  
  
?>
```

137

Lớp - Ví dụ

❖ Sinhvien.php

```
<?php  
  
class SV{  
    private $mssv;  
    private $ten;  
    private $dtb;  
    public function Set($ms,$t,$d){  
        $this->mssv=$ms;  
        $this->ten=$t;  
        $this->dtb=$d;  
    }  
  
?>
```

138

Lớp - Ví dụ

```
public function In(){
    echo $this->mssv;
    echo "<br>";
    echo $this->ten;
    echo "<br>";
    echo $this->dtb;
    echo "<br>";
}
};
?>
```

139

Lớp - Ví dụ (tt)

❖ xulysinhvien.php

```
<?php
include "sinhvien.php";
$sv1= new SV();
$sv1->Set("001","Tiêu Bắc Thần",9.5);
$sv1->In();
$sv1->mssv="009";      ?
echo $sv1->mssv;
?>
```

140

Phạm vi truy xuất mặc định

- Nếu không chỉ định phạm vi truy xuất của các thành phần trong lớp khi khai báo thì mặc định phạm vi truy xuất là public
- Ví dụ

```
class PhanSo{
    $tuso;
    $mauso;
}
```

- \$tuso, \$mauso có phạm vi truy xuất là public

141

Phương thức khởi tạo

- Dùng để khởi tạo các giá trị ban đầu cho các thuộc tính của đối tượng
- Trước PHP 7, Có 2 cách dùng
 - ❖ Cách 1:

```
public function __construct($name,$age,$color);
```

Hoặc:

```
public function __construct();
```
 - ❖ Cách 2: đã lỗi thời từ PHP 7, không nên sử dụng

```
public function TênLớp($name,$age,$color);
```

Hoặc:

```
public function TênLớp();
```

142

Phương thức hủy

- Thường hủy các giá trị của biến được khởi tạo trong session
- Cách dùng:

```
public function __destruct()
{
    // ...
    echo "Destruct được gọi";
}
```

143

Ví dụ đối tượng

```
<?php
class ConMeo{

    private $name;
    private $age;
    private $color;
    public function ConMeo()
    {
        $this->name = 'Taylor';
        $this->age = 1;
        $this->color = 'Vàng';
    }
}
```

144

Ví dụ đối tượng

```
/*
public function __construct($name,$age,$color)
{
    $this->name = $name;
    $this->age = $age;
    $this->color = $color;
}
*/
/*public function ConMeo($name,$age,$color)
{
    $this->name = $name;
    $this->age = $age;
    $this->color = $color;
}*/
```

145

Ví dụ đối tượng

```
public function __construct()
{
    $this->name = 'Katy';
    $this->age = 2;
    $this->color = 'Trang';
}
public function Show()
{
    echo $this->name."<br>";
    echo $this->age."<br>";
    echo $this->color;
}
```

146

Ví dụ đối tượng

```
public function Gan()
{
    $this->name="keke";
}
public function Gan1($t)
{
    $this->name=$t;
}
public function __destruct()
{
    echo "Destruct duoc goi";
}
```

147

Ví dụ đối tượng

```
// $m=new ConMeo("Mimi",1,"do");  
$m=new ConMeo();  
$m->Show();  
?>
```

148

Thừa kế trong PHP

- Là hình thức định nghĩa lớp mới kế thừa từ một đã có sẵn.

Lớp cơ bản, cơ sở



Lớp dẫn xuất

149

Thừa kế trong PHP _ Khai báo

```
class Coso  
{  
  
}  
class Dãnxuất extends Coso  
{  
  
}
```

150

Thừa kế

- Lớp dẫn xuất thừa kế lại những phương thức và thuộc tính từ lớp cơ sở.
- Thừa kế trong PHP là thừa kế public

151

Ví dụ - Lớp cơ sở

```
<?php
class Hinh{
    private $mau;
    public function GanMau($m){
        $this->mau=$m;
    }
    public function LayMau(){
        return $this->mau;
    }
    public function XuatMau(){
        echo "Màu:".$this->mau;
    }
}
?>
```

152

Ví dụ - lớp dẫn xuất

```
<?php
include "hinh.php";
class HinhTron extends Hinh{
    private $R;
    public function DienTich()
    {
        return $this->R*pi();
    }
    public function GanR($m,$r)
    {
        $this->GanMau($m);
        $this->R=$r;
    }
}
```

153

Ví dụ _ Lớp dẫn xuất

```
public function Xuat(){  
  
    $this->XuatMau();  
    $dt=$this->DienTich();  
  
    $dt=number_format($dt,2);  
    echo "<br>Diện tích:".$dt;  
  
}  
?>
```

154

Ví dụ - Tạo đối tượng và gọi hàm

```
<?php  
    include "hinhtron.php";  
    $ht=new HinhTron();  
    $ht->GanR("Vàng",2);  
    $ht->Xuat();  
  
?>
```

Màu:Vàng
Diện tích:6.28

155

Ví dụ - truy xuất thành phần qua tên lớp

```
<?php  
    include "hinhtron.php";  
    HinhTron::GanR("Vàng",2);  
    HinhTron::Xuat();  
  
?>
```

Màu:Vàng
Diện tích:6.28

156

Hàm chồng (overloading)

- Trước PHP 8 thì không cho phép khai báo hàm chồng trong 1 lớp.

```
public function Tong($n)
{
    $this->Tu=$this->Tu*$n->Mau+$this->Mau*$n->Tu;
    $this->Mau=$this->Mau*$n->Tu;
}
public function Tong()
{
    $this->Tu++;
    $this->Mau++;
}
```

Chương trình báo lỗi

157

Hàm chồng (overloading)

- PHP hỗ trợ **giải lập** hàm chồng từ phiên bản 8.0, sử dụng mixed type hoặc Union Types
- Hàm chồng (Overloading) là khả năng định nghĩa nhiều hàm trong cùng một class có cùng tên, nhưng khác nhau về số lượng hoặc kiểu dữ liệu của tham số

```
class MyClass {
    public function myMethod(mixed $arg): mixed {
        // Xử lý tham số $arg
        return $arg;
    }
}
```

158

Hàm chồng (overloading)

```
class Calculator {
    public function add(mixed $a, mixed $b) {
        if (is_int($a) && is_int($b)) {
            return $a + $b; // Cộng hai số nguyên
        } elseif (is_float($a) && is_float($b)) {
            return $a + $b; // Cộng hai số thực
        } else {
            return "Không thể cộng các kiểu dữ liệu khác.";
        }
    }
}

$calc = new Calculator();
echo $calc->add(5, 10); // Output: 15
echo "<br>";
echo $calc->add(2.5, 3.7); // Output: 6.2
echo "<br>";
echo $calc->add("Hello", "World"); // Output: Không thể cộng các kiểu dữ liệu khác.
```

159

Hàm chồng (overloading)

```
class MyString {
    public function process(string|array $data) {
        if (is_string($data)) {
            return strtoupper($data); // Chuyển đổi chuỗi thành chữ hoa
        } elseif (is_array($data)) {
            return implode(' ', $data); // Nối các phần tử mảng thành chuỗi
        }
    }
}

$str = new MyString();
echo $str->process("hello"); // Output: HELLO
echo "<br>";
echo $str->process(["apple", "banana", "orange"]); // Output: apple, banana,
orange
```

Union Types: Hàm process() nhận tham số \$data có thể là string hoặc array

160

Traits

- Traits là một tính năng trong PHP giúp **chia sẻ phương thức giữa nhiều class khác nhau**. Traits cung cấp một giải pháp cho vấn đề đa thừa kế, vì PHP không hỗ trợ đa thừa kế trực tiếp (một class không thể kế thừa từ nhiều class khác nhau).

```
trait Logger {
    public function log($message) {
        echo $message;
    }
}

class FileHandler {
    use Logger;
}

class UserManager {
    use Logger;
}

$userManager = new UserManager();
$userManager->log("User created."); // Output: User created.
```

161

Tính override

- Trong lớp dẫn xuất định nghĩa lại một hay nhiều phương thức đã có trong lớp cơ sở
- Tính đa hình trong lập trình hướng đối tượng trên PHP thể hiện qua tính override.

162

Ví dụ - tính đa hình

```
<?php
class Hinh{
    public function Ve(){
        echo "Ve hình";
    }
}
class HinhVuong extends Hinh{
    public function Ve(){
        echo "Ve hình vuông";
    }
}
```



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 163

163

Ví dụ - Tính đa hình

```
class HinhTron extends Hinh{
    public function Ve(){
        echo "Ve hình tron";
    }
}
$a=new Hinh();
$a->Ve();
$b=new HinhTron();
$b->Ve();?>
```



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 164

164

clone và parent

- clone: tạo ra sao chép một đối tượng cho một đối tượng (tạo ra một bản sao độc lập của object, thay đổi object clone không ảnh hưởng đến object gốc.)

```
$m=new HinhVuong();
$m->Show();
$n=clone $m;
$n->Ve();
```



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 165

165

clone và parent

- parent: Dùng để gọi phương thức của lớp cha trong trường hợp định nghĩa lại hàm
- Ví dụ:

```
<?php
class A
{
    private $name;
    public function __construct($name)
    {
        $this->name=$name;
    }
    public function view()
    {
        echo "name:". $this->name;
    }
}
```

166

Ví dụ (tt)

```
class B extends A
{
    public function __construct($name)
    {
        parent::__construct($name);
    }

    public function view()
    {
        parent::view();
    }
}
```

167

Ví dụ (tt)

```
$c=new B("Nguyễn Văn Tú");
$c->view();
?>
```

168

Bài tập

- Cài đặt lớp hình chữ nhật:
 - Khởi tạo
 - Diện tích
 - Chu vi
- Cài đặt lớp hình hộp chữ nhật:
 - Khởi tạo
 - Diện tích
 - Thể tích(ghi chú: phương thức tính diện tích của 2 lớp đặt tên giống nhau)
- Tạo đối tượng thuộc lớp hình hộp chữ nhật, gọi phương thức thể tích, diện tích

169

Bài tập

- Triển khai lớp hình tròn:
 - Với các phương thức: Khởi tạo có đối số, diện tích hình tròn, chu vi hình tròn
- Cài đặt lớp hình trụ tròn: Khởi tạo có đối số, Diện tích hình trụ tròn, thể tích hình trụ tròn.
- Tạo đối tượng và gọi phương thức diện tích hình trụ tròn, thể tích hình trụ tròn

170

Lớp trừu tượng (abstract)

- Lớp abstract dùng để định nghĩa các phương thức mà các phương thức này sẽ được định nghĩa lại ở lớp dẫn xuất.
- Các phương thức của lớp abstract phải được khai báo abstract và có phạm vi truy xuất public hoặc protected.
- Có thể khai báo thuộc tính cho lớp abstract
- Không thể tạo ra một đối tượng thuộc lớp abstract

171

Ví dụ - lớp abstract

```
<?php
abstract class Hinh{
    private $mau;
    public function GanMau($m) {
        $this->mau=$m;
    }
    public function LayMau(){
        return $this->mau;
    }
}
```

172

Ví dụ - lớp abstract

```
    public function Xuat(){
        echo "Màu:". $this->mau;
    }
    abstract public function DienTich();
    abstract public function ChuVi();
}
```

173

Ví dụ - lớp abstract

```
class HinhTron extends Hinh{
    private $R;
    public function Xuat(){
        parent::Xuat();
        echo "<br>Diện tích:".number_format($this->DienTich(),2);
        echo "<br>Chu vi:".number_format($this->ChuVi(),2);
    }
}
```

174

Ví dụ - lớp abstract

```
public function DienTich(){
    return pi()*$this->R*$this->R;
}
public function ChuVi(){
    return 2*pi()*$this->R;
}
public function GanR($r){
    $this->R=$r;
}
}
```

175

Ví dụ - lớp abstract

```
class HinhVuong extends Hinh
{
    private $Canh;
    public function Xuat(){
        echo "<br>";
        parent::Xuat();
        echo "<br>Diện tích:".number_format($this-
>DienTich(),2);
        echo "<br>Chu vi:".number_format($this-
>ChuVi(),2);
    }
}
```

176

Ví dụ - lớp abstract

```
public function DienTich(){
    return $this->Canh*$this->Canh;
}
public function ChuVi(){
    return $this->Canh*4;
}
public function GanCanh($c){
    $this->Canh=$c;
}
}
```

177

Ví dụ - lớp abstract

```
$ht=new HìnhTron();
$ht->GanR(5);
$ht->GanMau("Xanh");
echo "Thông tin hình tròn<br>";
$ht->Xuat();
$hv=new HìnhVuong();
$hv->GanMau("Đỏ");
$hv->GanCanh(4);
echo "<br>Thông tin hình vuông";
$hv->Xuat(); ?>
```

178

Interface

- Interface không phải là 1 lớp. Nó được mô tả như là 1 bản thiết kế cho các class có chung cách thức hoạt động
- Không thể định nghĩa các thuộc tính, khởi tạo đối tượng mà chỉ khai báo các phương thức
- Chỉ khai báo mà không có phần thân hàm. Interface giúp tạo ra sự nhất quán và đảm bảo rằng các class tuân theo cùng một bộ phương thức.

179

Interface

- Không có khái niệm phạm vi của phương thức, tất cả đều là public.
- Lớp con kế thừa từ interface sẽ phải override tất cả các phương thức trong đó.
- Một lớp có thể kế thừa từ nhiều interface khác nhau bằng từ khóa implements

180

Interface – ví dụ

```
interface Move {  
    function run();  
}  
class Dog implements Move {  
    public function run ()  
    { echo "Con chó chạy bằng 4 chân";  
    }  
}  
class Car implements Move {  
    public function run ()  
    {  
        echo "Xe hơi chạy bằng 4 bánh";  
    }  
}
```

181

So sánh Abstract - interface

- **Không** thể khởi tạo đối tượng Abstract Class
- **Bất kỳ** lớp nào có chứa ít nhất 1 phương thức trừu tượng thì chắc chắn nó phải là Abstract Class. 1 Abstract Class có thể chứa các phương thức trừu tượng hoặc không trừu tượng.
- Phương thức abstract của Abstract Class không có thân hàm
- Các phương thức abstract phải được định nghĩa lại ở lớp dẫn xuất
- Không hỗ trợ đa thừa kế

182

So sánh Abstract - interface

- Interface được định nghĩa để cung cấp các tên hàm chung để có thể triển khai.
- Interface được xem như là bộ khung của lớp dẫn xuất
- Interface cũng không thể khởi tạo
- Các phương thức trong Interface mặc định là các phương thức trừu tượng

183

So sánh Abstract - interface

- Các phương thức trong Interface phải có phạm vi truy xuất public và không có thân hàm
- Interface có thể được **extends** với nhau
- 1 lớp thể **implements** nhiều Interface

184

SOLID

- SOLID là một tập hợp các nguyên tắc được đề xuất bởi Robert C. Martin, giúp lập trình viên xây dựng các hệ thống phần mềm dễ bảo trì, mở rộng, và có cấu trúc rõ ràng.
- SOLID bao gồm năm nguyên tắc chính:
 - **S - Single Responsibility Principle (SRP)**: Mỗi lớp chỉ nên có một trách nhiệm duy nhất.
 - **O - Open/Closed Principle (OCP)**: Một lớp nên mở để mở rộng, nhưng đóng để sửa đổi.
 - **L - Liskov Substitution Principle (LSP)**: Các lớp con có thể thay thế lớp cha mà không làm thay đổi tính đúng đắn của chương trình.
 - **I - Interface Segregation Principle (ISP)**: Một lớp không nên buộc phải triển khai các giao diện mà nó không sử dụng.
 - **D - Dependency Inversion Principle (DIP)**: Các module cấp cao không nên phụ thuộc vào các module cấp thấp, mà cả hai nên phụ thuộc vào các trừu tượng.

185

SOLID

- Chúng ta sẽ tập trung vào hai nguyên tắc cơ bản và quan trọng nhất trong số đó: SRP và OCP. Hai nguyên tắc này rất quan trọng trong việc đảm bảo mã nguồn dễ bảo trì, dễ mở rộng và tránh gây ra các lỗi không mong muốn khi phát triển phần mềm.

186

SOLID

- Nguyên tắc SRP (Single Responsibility Principle)
- Nguyên tắc Trách nhiệm Đơn lẻ nói rằng mỗi lớp chỉ nên có một lý do để thay đổi, hay nói cách khác, một lớp chỉ nên có một nhiệm vụ duy nhất.
- Điều này giúp cho mã nguồn dễ dàng bảo trì, bởi vì khi một lớp chỉ chịu trách nhiệm về một nhiệm vụ cụ thể, việc thay đổi logic của nhiệm vụ đó sẽ không ảnh hưởng đến các phần khác của hệ thống.

187

SOLID

```
class UserManager {                                //Ví dụ không tuân theo SRP
    public function createUser($username, $email) {
        // Lưu thông tin người dùng vào cơ sở dữ liệu
        echo "Creating user with username: $username";

        // Gửi email thông báo
        echo "Sending email to: $email";
    }
}
```

- Lớp UserManager chịu trách nhiệm cho cả việc tạo người dùng và việc gửi email thông báo. Điều này vi phạm nguyên tắc SRP vì lớp này có hơn một trách nhiệm: quản lý người dùng và gửi email. Nếu sau này hệ thống gửi email thay đổi (ví dụ sử dụng dịch vụ email bên ngoài), sẽ phải sửa đổi lớp UserManager, điều này có thể gây ra lỗi hoặc làm hỏng hệ thống.

188

SOLID

Ví dụ tuân theo SRP:

```
class UserManager {
    public function createUser($username, $email) {
        // Lưu thông tin người dùng vào cơ sở dữ liệu
        echo "Creating user with username: $username";
    }
}

class EmailService {
    public function sendEmail($email, $message) {
        // Gửi email thông báo
        echo "Sending email to: $email with message: $message";
    }
}
```

189

SOLID

- **Lợi ích của SRP**
- Dễ bảo trì: Khi có lỗi, dễ dàng xác định và sửa chữa lỗi trong từng nhiệm vụ riêng biệt mà không ảnh hưởng đến các phần khác.
- Dễ mở rộng: Khi cần thêm tính năng, có thể dễ dàng thêm vào mà không phải sửa đổi nhiều mã đã có.
- Tăng tính tái sử dụng: Các lớp được thiết kế có thể tái sử dụng ở nhiều nơi khác nhau mà không cần chỉnh sửa.

190

SOLID

- **Nguyên tắc OCP (Open/Closed Principle)**
- Nguyên tắc Mở/Đóng nói rằng một lớp nên mở để mở rộng, nhưng đóng để sửa đổi. Điều này có nghĩa là khi cần thêm tính năng mới, không nên chỉnh sửa mã gốc mà thay vào đó là mở rộng nó (thông qua kế thừa, hoặc sử dụng interface).
- Điều này giúp hệ thống ổn định hơn và tránh các lỗi không mong muốn xảy ra khi mã hiện tại bị sửa đổi.

191

SOLID

- **Ví dụ không tuân theo OCP**

```
class PaymentProcessor {  
    public function processPayment($type) {  
        if ($type === 'paypal') {  
            echo "Processing payment with PayPal.";  
        } elseif ($type === 'stripe') {  
            echo "Processing payment with Stripe.";  
        }  
    }  
}
```

Giả sử chúng ta có một hệ thống thanh toán trong PHP, nơi có thể thanh toán bằng PayPal và Stripe, mỗi khi cần thêm một phương thức thanh toán mới, chúng ta phải sửa đổi lớp PaymentProcessor. Điều này vi phạm nguyên tắc OCP vì chúng ta phải thay đổi mã gốc mỗi lần thêm tính năng mới.

192

SOLID


- Ví dụ tuân theo OCP

```
interface PaymentMethod {
    public function process();
}

class PayPalPayment implements PaymentMethod {
    public function process() {
        echo "Processing payment with PayPal.";
    }
}

class StripePayment implements PaymentMethod {
    public function process() {
        echo "Processing payment with Stripe.";
    }
}
```

```
class PaymentProcessor {
    public function
    processPayment(PaymentMethod
    $paymentMethod) {
        $paymentMethod->process();
    }
}
```



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT- HCM


Phát triển ứng dụng Web193

193

SOLID

- Lợi ích của OCP

- Giảm thiểu rủi ro lỗi: Không cần phải sửa đổi mã hiện có, điều này giúp giảm thiểu các lỗi không mong muốn.
- Dễ dàng mở rộng: Khi có tính năng mới, bạn chỉ cần thêm lớp mới thay vì phải chỉnh sửa mã cũ.
- Tăng tính ổn định: Mã cũ hoạt động tốt sẽ không bị ảnh hưởng khi thêm tính năng mới.



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN


UT- HCM

Phát triển ứng dụng Web194

194

Một số nguyên tắc lập trình khác áp dụng trong phát triển PHP

- Ngoài SOLID, còn nhiều nguyên tắc lập trình khác có thể được áp dụng để giúp quá trình phát triển PHP hiệu quả và tối ưu hơn:
- KISS (Keep It Simple, Stupid): Nguyên tắc này nhấn mạnh rằng mã nguồn và thiết kế nên đơn giản, dễ hiểu và dễ bảo trì. Trong phát triển PHP, không nên tạo ra các lớp hoặc phương thức phức tạp hơn mức cần thiết.
- DRY (Don't Repeat Yourself): Tránh việc lặp lại mã nguồn. Trong PHP, nếu có một đoạn mã xuất hiện nhiều lần, nên tách thành hàm hoặc lớp riêng để tái sử dụng. Điều này giúp mã dễ bảo trì và giảm rủi ro khi phải sửa đổi.
- YAGNI (You Aren't Gonna Need It): Tránh viết những đoạn mã không cần thiết cho chức năng hiện tại. Chỉ phát triển những tính năng mà bạn thực sự cần cho dự án, tránh viết thêm tính năng mà bạn "có thể cần trong tương lai".
- ...



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT- HCM

Phát triển ứng dụng Web195

195

65

A large, empty white rectangular area, likely a placeholder for content or a diagram.

[illegible]

Bài tập

- **Bài 1:** Cài đặt lớp mảng một chiều các số nguyên với các phương thức
 - ✧ Khởi tạo số phần tử và giá trị cho từng phần tử của mảng
 - ✧ Phương thức tính tổng các phần tử trong mảng
 - ✧ Phương thức xuất mảng ra trang web
 - ✧ Phương thức tìm phần tử lớn nhất
 - ✧ Phương thức tìm phần tử nhỏ nhất
 - ✧ Phương thức tìm một phần tử có hay không trong mảng
 - ✧ Phương thức xóa một phần tử trong mảng
 - ✧ Phương thức sắp xếp mảng tăng giảm
 - ✧ Phương thức kiểm tra mảng có đối xứng hay không
 - ✧ Phương thức đảo một mảng
 - ✧ Phương thức đếm số phần tử có giá trị bằng x trong mảng

Bài tập (tt)

- **Bài 2:** Cài đặt lớp sinh viên, biết rằng thông tin của các sinh viên gồm: Mã số sinh viên, tên sinh viên, điểm trung bình. Và các phương thức sau:
 - Phương thức khởi tạo
 - Gán mã sinh viên, tên sinh viên, điểm trung bình
 - Lấy mã sinh viên, tên sinh viên, điểm trung bình

Bài tập (tt)

- **Bài 3:** Cài đặt lớp danh sách sinh viên để quản lý danh sách các sinh viên trong lớp, với các chức năng sau:
 - Xuất lớp học ra trang web
 - Thêm một sinh viên vào lớp
 - Xóa một sinh viên theo mã số sinh viên
 - Tìm một sinh viên theo tên
 - Cho biết điểm trung bình cao nhất trong lớp học là bao nhiêu
 - Sắp xếp danh sách sinh viên tăng theo điểm trung bình

199

Bài tập (tt)

- **Bài 4:** Cài đặt lớp phân số với các phương thức
 - Khởi tạo một phân số
 - Xuất phân số ra trang web
 - Cộng, trừ, nhân, chia hai phân số
 - Đơn giản một phân số
 - So sánh hai phân số

200

Bài tập (tt)

- **Bài 5:** Cài đặt lớp ngày tháng năm với các phương thức cần thiết để in ra thứ của một ngày bất kỳ.

201

Bài tập (tt)

Bài 6: Một chiếc xe máy chạy 100km tốn 2 lít xăng, cứ chở thêm 10kg hàng xe tốn thêm 0.1lít xăng.

Một chiếc xe tải chạy 100km tốn 20lít xăng, cứ chở thêm 1000kg hàng xe tốn thêm 1lít xăng.

Dùng kế thừa xây dựng lớp XeMay và XeTai cho phép:

- Chất một lượng hàng lên xe.
- Bỏ bớt một lượng hàng xuống xe.
- Đổ một lượng xăng vào xe.
- Cho xe chạy một đoạn đường.
- Kiểm tra xem xe đã hết xăng chưa.

Cho biết lượng xăng còn trong xe.



Phát triển ứng dụng Web 202

202

6. Bổ sung Một số vấn đề bảo mật



Phát triển ứng dụng Web 203

203

AN TOÀN Thông tin

TIN TỨC CHÍNH SÁCH CHIẾN LƯỢC TẤN CÔNG MẠNG CHỐNG THỰC DIỆN TỬ MẬT MÃ DÂN SỰ GIẢI PHÁP

HOT nâng lực không gian mạng

An toàn thông tin

Năm 2023: số vụ tấn công mạng nhắm vào các hệ thống tại Việt Nam tăng 9,5%

17:00 | 16/12/2023 | AN TOÀN THÔNG TIN

Theo báo cáo tổng kết tình hình An ninh mạng Việt Nam năm 2023 của công ty Công nghệ An ninh mạng quốc gia Việt Nam (NCS) công bố mới đây, số vụ tấn công mạng vào các tổ chức tăng 9,5% so với năm 2022, trung bình 1.160 vụ mỗi tháng.



Phát triển ứng dụng Web 204

204

- Các chuyên gia NCS chỉ ra top 3 điểm yếu bị tấn công nhiều nhất tại Việt Nam năm 2023. Trong đó, tỷ lệ cao nhất là điểm yếu con người, chiếm 32,6% tổng số việc. Hacker sử dụng email giả mạo (phishing) có file đính kèm để dẫn dắt người dùng file văn bản hoặc nội dung có đường link đăng nhập giả mạo để chiếm tài khoản, kiểm soát máy tính người dùng từ xa.
- Điểm yếu có tỷ lệ cao thứ hai là lỗ hổng của các nền tảng, dịch vụ phần mềm cài đặt trên máy chủ chiếm 27,4%. Các phần mềm bị khai thác là phần mềm Mail Server, nền tảng quản lý nội dung, nền tảng chia sẻ dữ liệu...
- Điểm yếu thứ ba là các lỗ hổng của website do tổ chức tự phát triển chiếm 25,3% số vụ việc. Các lỗ hổng thường bị khai thác là SQL Injection, mật khẩu quản trị yếu hoặc sử dụng thư viện tồn tại lỗ hổng.

205

Các dạng tấn công phổ biến ứng dụng web

1. **SQL Injection (SQLi):** Cho phép kẻ tấn công chèn mã SQL độc hại vào các câu truy vấn cơ sở dữ liệu của ứng dụng. Tấn công này có thể dẫn đến việc rò rỉ, thay đổi hoặc xóa dữ liệu. Phòng ngừa: Sử dụng Prepared Statements, Stored Procedures, và kiểm tra đầu vào kỹ lưỡng.

Ví dụ: Một form đăng nhập không kiểm tra dữ liệu đầu vào:

```
SELECT * FROM users WHERE username = 'admin' AND password = 'password';
```

Nếu kẻ tấn công nhập admin' -- làm tên đăng nhập, câu lệnh SQL trở thành:

```
SELECT * FROM users WHERE username = 'admin' --' AND password = '';
```

Phần -- sẽ làm cho phần sau của câu lệnh bị bỏ qua, giúp kẻ tấn công đăng nhập mà không cần mật khẩu.

206

Các dạng tấn công phổ biến ứng dụng web

Ví dụ: Một form tìm kiếm sản phẩm trên trang web cho phép nhập tên sản phẩm. Nếu truy vấn SQL không được xử lý đúng, kẻ tấn công có thể nhập " OR '1'='1" vào trường tìm kiếm. Câu truy vấn SQL sẽ trở thành:

SELECT * FROM products WHERE name = " OR '1'='1';

Điều này sẽ trả về toàn bộ danh sách sản phẩm vì điều kiện $1=1$ luôn đúng.

207

Các dạng tấn công phổ biến ứng dụng web

2. **Cross-Site Scripting (XSS)**: Kẻ tấn công chèn mã JavaScript độc hại vào trang web, khiến mã này được thực thi trên trình duyệt của người dùng khi họ truy cập trang. XSS thường được chia thành 3 loại: **Stored XSS**, **Reflected XSS**, và **DOM-based XSS**. Phòng ngừa: Sử dụng HTML escaping, triển khai Content Security Policy (CSP), và lọc các đầu vào.

Ví dụ: Kẻ tấn công chèn mã JavaScript vào một trường bình luận

```
<script>alert('Bạn đã bị hack!');</script>
```

Khi người dùng khác truy cập trang có chứa bình luận này, hộp thoại sẽ xuất hiện với thông báo trên, chứng tỏ mã JavaScript đã được thực thi trên trình duyệt của người dùng



Các dạng tấn công phổ biến ứng dụng web

Ví dụ Stored XSS: Một trang blog cho phép người dùng viết bình luận. Kẻ tấn công nhập vào phần bình luận đoạn code JS sau:

```
<script>document.cookie="session="+document.cookie;</script>
```

Mỗi khi ai đó xem bình luận này, mã sẽ chạy và gửi cookie của họ cho kẻ tấn công, giúp kẻ tấn công chiếm quyền truy cập.

Ví dụ Reflected XSS: Một trang tìm kiếm hiển thị lại từ khóa mà người dùng nhập vào mà không kiểm tra đầu vào. Kẻ tấn công gửi liên kết:

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

Khi nạn nhân nhấp vào liên kết, đoạn mã JavaScript được thực thi và hiển thị thông báo, chứng tỏ mã đã được thực thi.



Các dạng tấn công phổ biến ứng dụng web

3. **Cross-Site Request Forgery (CSRF)**: Kẻ tấn công giả mạo yêu cầu HTTP từ người dùng đã xác thực, gây ra các hành động không mong muốn như chuyển tiền, thay đổi dữ liệu. Phòng ngừa: Sử dụng CSRF tokens cho các yêu cầu nhạy cảm, và SameSite cookies.

Ví dụ: Kẻ tấn công gửi một email chứa liên kết yêu cầu chuyển tiền từ tài khoản của người dùng:

```

```

Nếu người dùng đã đăng nhập vào tài khoản ngân hàng của họ, liên kết này có thể kích hoạt yêu cầu chuyển tiền đến tài khoản của kẻ tấn công.



Các dạng tấn công phổ biến ứng dụng web

Ví dụ: Kẻ tấn công gửi email chứa form ẩn và lừa người dùng nhấn vào một nút. Form này tự động gửi yêu cầu đổi mật khẩu:

```
<form action="http://example.com/change-password" method="POST">
  <input type="hidden" name="new_password" value="new_password123" />
  <input type="submit" value="Click me to win a prize!" />
</form>
```

Khi người dùng nhấn vào nút, yêu cầu đổi mật khẩu sẽ được gửi đi mà họ không hề hay biết.



211

Các dạng tấn công phổ biến ứng dụng web

- 4. Remote Code Execution (RCE)** Kẻ tấn công khai thác lỗ hổng để thực thi mã độc trên máy chủ của ứng dụng, cho phép truy cập trái phép và điều khiển hệ thống. Phòng ngừa: Cập nhật phần mềm thường xuyên, hạn chế quyền truy cập và sử dụng các biện pháp xác thực và mã hóa mạnh.
- 5. Insecure Deserialization:** Kẻ tấn công gửi dữ liệu được sửa đổi đến ứng dụng, làm cho ứng dụng giải mã và thực thi dữ liệu, dẫn đến việc thực thi mã độc. Phòng ngừa: Kiểm tra chặt chẽ dữ liệu đầu vào và hạn chế việc sử dụng deserialization khi không cần thiết.
- 6. Man-in-the-Middle (MitM):** Kẻ tấn công chặn và theo dõi dữ liệu giữa người dùng và ứng dụng, có thể dẫn đến việc đánh cắp dữ liệu nhạy cảm như mật khẩu và thông tin thanh toán. Phòng ngừa: Sử dụng HTTPS/TLS cho tất cả các kết nối, triển khai chứng chỉ SSL và hạn chế truy cập mạng.



212

Các dạng tấn công phổ biến ứng dụng web

- 7. Brute Force Attack:** Kẻ tấn công thử tất cả các tổ hợp mật khẩu để truy cập vào tài khoản người dùng hoặc các tài nguyên bảo mật khác. Phòng ngừa: Sử dụng giới hạn số lần đăng nhập thất bại, triển khai xác thực hai yếu tố (2FA), và yêu cầu mật khẩu mạnh.
- 8. File Inclusion Vulnerabilities:** Kẻ tấn công chen các file từ xa hoặc nội bộ vào ứng dụng để chiếm quyền kiểm soát hệ thống hoặc truy cập dữ liệu nhạy cảm. Phòng ngừa: Xác minh đường dẫn của file, không cho phép sử dụng file từ nguồn không tin cậy.
- 9. Directory Traversal:** Kẻ tấn công sử dụng đường dẫn để truy cập các file nhạy cảm hoặc ngoài phạm vi cho phép của hệ thống. Phòng ngừa: Kiểm tra và xác thực đường dẫn, giới hạn quyền truy cập của người dùng vào thư mục và file nhạy cảm.



213

Các dạng tấn công phổ biến ứng dụng web

10. **Security Misconfiguration:** Bao gồm các cấu hình sai lệch hoặc không an toàn, chẳng hạn như sử dụng các mật khẩu mặc định, cài đặt không đúng quy định hoặc hiển thị lỗi chi tiết trên giao diện người dùng. Phòng ngừa: Thường xuyên kiểm tra và cấu hình lại hệ thống, ẩn các thông tin nhạy cảm, và áp dụng phương pháp bảo mật theo chiều sâu (defense-in-depth)
11. **Phishing và Social Engineering:** Lừa người dùng chia sẻ thông tin nhạy cảm thông qua email, website giả mạo, hoặc tin nhắn. Phòng ngừa: Đào tạo người dùng về nhận diện tấn công phishing, triển khai công cụ phát hiện và ngăn chặn phishing.
12. **Session Hijacking:** Kê tấn công chiếm quyền kiểm soát session người dùng, giúp họ có quyền truy cập trái phép vào tài khoản người dùng. Phòng ngừa: Sử dụng các Session Tokens bảo mật, triển khai Session Timeout, và giới hạn quyền truy cập phiên.



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 214

214

Một số cách thức bảo mật web

Nguyên tắc:

- **HTML & JavaScript:** Sử dụng các thẻ HTML như maxlength, pattern, và input type="number" để hạn chế đầu vào từ người dùng. Sử dụng JavaScript để kiểm tra đầu vào ngay trên client-side trước khi gửi đến server.
- **PHP:** Lọc và xác thực tất cả đầu vào từ người dùng bằng các hàm filter_input(), htmlspecialchars(), và preg_match(). Luôn sử dụng Prepared Statements với PDO để ngăn chặn SQL Injection. Cài đặt CSP và X-Frame-Options thông qua header HTTP để chống XSS và Clickjacking.
- **MySQL:** Đảm bảo thiết lập quyền hạn tối thiểu cho tài khoản MySQL, không cấp quyền dư thừa như FILE, EXECUTE, hay SUPER. Sử dụng hàm mysqli_real_escape_string() hoặc các biện pháp tương đương để thoát các ký tự đặc biệt trong MySQL nếu không thể sử dụng PDO.



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 215

215

Một số cách thức bảo mật web

- **Sử dụng Prepared Statements:** Thay vì sử dụng truy vấn SQL với chuỗi nối (concatenation), hãy sử dụng Prepared Statements để bảo vệ cơ sở dữ liệu khỏi mã độc. Ví dụ, trong PHP với PDO:

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username");  
$stmt->execute(['username' => $userInput]);
```
- **Kiểm Tra và Lọc Đầu Vào:** Lọc đầu vào để đảm bảo rằng nó chứa các ký tự hợp lệ trước khi đưa vào truy vấn SQL. Ví dụ:

```
$email = filter_var($userInput, FILTER_SANITIZE_EMAIL);
```
- **Sử dụng Các Quyền Truy Cập Cơ Sở Dữ Liệu Thích Hợp:** Cấp quyền hạn chế cho người dùng cơ sở dữ liệu. Ví dụ, chỉ cấp quyền SELECT khi không cần thiết phải UPDATE hoặc DELETE.



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 216

216

Một số cách thức bảo mật web

- **Sử dụng Các Thuộc Tính Input Validation:** Trong HTML5, có thể sử dụng các thuộc tính như maxlength, pattern, và required để hạn chế đầu vào từ phía client:

```
<input type="text" name="username" required pattern="[A-Za-z0-9]{5,10}">
```

- **Sử dụng Các Thẻ Meta Bảo Mật:** Sử dụng thẻ <meta> để tăng cường bảo mật, chẳng hạn như ngăn chặn clickjacking:

```
<meta http-equiv="X-Frame-Options" content="DENY">
```

217

Một số cách thức bảo mật web

- **Thoát Đầu Ra (Output Escaping):** Luôn sử dụng hàm để thoát các ký tự đặc biệt trước khi hiển thị dữ liệu từ người dùng lên trang web. Trong PHP, có thể sử dụng htmlspecialchars() để thoát các ký tự như <, >, &, và ":

```
echo htmlspecialchars($userInput, ENT_QUOTES, 'UTF-8');
```

- **Kiểm tra và Lọc Đầu Vào:** Chỉ cho phép các ký tự hợp lệ được nhập vào và loại bỏ các ký tự không mong muốn. Ví dụ, nếu bạn chỉ chấp nhận số, có thể sử dụng filter_var() trong PHP:

```
$number = filter_var($userInput, FILTER_SANITIZE_NUMBER_INT);
```

218

Một số cách thức bảo mật web

- **Sử dụng HTTP Headers:** Sử dụng Content Security Policy (CSP) để ngăn chặn việc thực thi các mã không được phép từ các nguồn bên ngoài. CSP là một tiêu chuẩn bảo mật giúp ngăn chặn XSS bằng cách chỉ cho phép tải các tập lệnh từ nguồn đáng tin cậy.

```
// Thêm tiêu đề CSP từ phía máy chủ (Ví dụ với Node.js và Express)
```

```
app.use((req, res, next) => {  
  res.setHeader("Content-Security-Policy", "default-src 'self'");  
  next();  
});
```

219

Một số cách thức bảo mật web

- **Sử dụng các Thư viện JavaScript để Thoát Ký Tự:** Sử dụng thư viện giúp thoát các ký tự đặc biệt trong JavaScript để ngăn chặn việc mã độc được chèn vào HTML.

```
// Thoát các ký tự HTML đặc biệt trước khi đưa vào DOM
function escapeHTML(str) {
  const div = document.createElement('div');
  div.appendChild(document.createTextNode(str));
  return div.innerHTML;
}
// Sử dụng hàm
const safeText = escapeHTML(userInput);
document.getElementById("output").innerHTML = safeText;
```



Phát triển ứng dụng Web 220

220

Một số cách thức bảo mật web

- **Hạn chế Sử dụng innerHTML để Chèn Dữ Liệu vào DOM:** Tránh sử dụng innerHTML để gán dữ liệu vào DOM, vì nó có thể dễ dàng bị khai thác để chèn mã độc. Thay vào đó, sử dụng các phương thức như textContent hoặc innerText.

```
// Thay vì sử dụng innerHTML
// document.getElementById("output").innerHTML = userInput;
// Sử dụng textContent để an toàn hơn
document.getElementById("output").textContent = userInput;
```

- **Sử dụng document.createElement để Tạo Phần Tử:** Sử dụng document.createElement để tạo và chèn các phần tử DOM một cách an toàn.

```
const userInput = "Example";
const newElement = document.createElement("div");
newElement.textContent = userInput;
document.body.appendChild(newElement);
```



Phát triển ứng dụng Web 221

221

Q & A



Cảm ơn đã theo dõi

Hãy vọng cùng nhau đi đến thành công.



Phát triển ứng dụng Web 222

222
