PHẦN A: LÝ THUYẾT CƠ BẢN VỀ PHP

PHP Intro

PHP là gì?

- PHP là từ viết tắt của: "PHP: Hypertext Preprocessor"
- PHP được sử dụng rộng rãi, là mã nguồn mở
- PHP là các đoạn mã kịch bản được chạy trên Server
- Có thể download và sử dụng miễn phí

Tập tin PHP là gì?

- Tập tin PHP có thể chứa chữ, HTML, CSS, JavaScript và các đoạn mã PHP
- Các đoạn mã PHP được chạy trên máy chủ, và trả về kết quả trên trình duyệt dưới dạng HTML
- Phần mở rộng: {tên_file}**.php**

PHP có thể thực hiện những gì?

- PHP có thể tạo ra các trang web động
- PHP có thể tạo, mở, đọc, ghi, xóa và đóng các tập tin trên máy chủ
- PHP có thể thu thập dữ liệu từ form
- PHP có thể gửi và nhận cookies
- PHP có thể thêm, xóa, sửa dữ liệu trong cơ sở dữ liệu
- PHP có thể được sử dụng để điều kiển truy cập người dùng
- PHP có thể mã hóa dữ liệu

Với PHP, bạn có thể tạo ra những trang HTML với nội dung không giới hạn, bao gồm các hình ảnh, tập tin PDF, thậm chí cả Flash movies. Bạn cũng có thể tạo ra các đoạn mã như XHTML và XML.

Tại sao nên sử dụng PHP?

- PHP có thể chạy trên đa hệ điều hành (Windows, Linux, Unix, Mac OS X ...)
- PHP tương thích với hầu hết các servers được sử dụng hiện nay (Apache, IIS...)
- PHP hỗ trợ phần lớn các hệ quản trị cơ sở dữ liệu.
- PHP hoàn toàn miễn phí. Download tại website chính thức: **www.php.net**
- PHP dễ học và hoạt động hiệu quả bên phía Server.

PHP Install

Cần những gì để chạy PHP?

Để bắt đầu sử dụng PHP, bạn cần:

- Tìm một web host có hỗ trợ PHP và MySQL
- Cài đặt web server trên máy tính của bạn, sau đó cài đặt PHP & MySQL

Sử dụng Web Host hỗ trợ PHP?

Nếu server của bạn hỗ trợ PHP và đã kích hoạt, bạn không cần làm thêm gì.

Hãy tạo một vài file có phần mở rộng là **.php**, đưa chúng vào thư mục web và server sẽ tự động phân tích nội dung cho bạn.

Bạn không cần phải biên dịch hoặc cài đặt thêm bất kỳ công cụ nào.

Vì PHP miễn phí, nên hầu hết các web host đều hỗ trợ PHP.

Cài đặt PHP trên máy tính của bạn

Tuy nhiên, nếu server của bạn không hỗ trợ PHP, bạn cần:

- Cài đặt web server
- Cài đặt PHP
- Cài đặt hệ quản trị cơ sở dữ liệu, như MySQL

Bạn có thể xem thêm hướng dẫn cài đặt tại website PHP chính thức:

http://php.net/manual/en/install.php

PHP Syntax

Các đoạn mã PHP được thực thi trên máy chủ, và kết quả được trả về dưới dạng HTML cho trình duyệt đọc và hiển thị.

Cú pháp PHP cơ bản

Đoạn mã PHP có thể đặt ở bất kỳ vị trí nào trong văn bản.

Đoạn mã PHP bắt đầu với ký hiệu **<?php** và kết thúc bởi **?>**

```
<?php
// Code PHP ở đây
?>
```

Phần mở rộng mặc định cho file PHP là "**.php**"

Một file PHP thông thường chứa các thẻ HTML, và một vài đoạn mã PHP.

Dưới đây, chúng ta có một ví dụ về một file PHP đơn giản, với đoạn mã PHP được sử dụng nhờ hàm "echo" trong PHP để hiển thị chữ "Hello World!" trên trang web:

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

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

</body>
</html>
```

Kết quả:

My first PHP page

Hello World!

Chú ý: Câu lệnh PHP kết thúc bởi dấu chấm phẩy (;)

Chú thích trong PHP

Một comment trong PHP là một hoặc nhiều dòng không được đọc/thực thi trong chương trình. Nó chỉ có mục đích là để ai đó đọc và hiểu code.

Những comment có thể được dùng để:

- Cho ai đó thấy được bạn đang làm gì
- Tự ghi nhớ những gì đã làm – Hầu hết những lập trình viên có kinh nghiệm trở lại với công việc một hoặc hai năm sau và phải định hình những gì họ đã làm. Các comment có thể gợi nhớ những gì bạn nghĩ khi bạn viết code

PHP hỗ trợ nhiều cách comment đa dạng:

```
<!DOCTYPE html>
<html>
<body>

<?php
// Đây là comment một dòng

# Đây cũng là comment một dòng

/*
Đây là khối comment để bao quanh nhiều dòng
*/

// Có thể sử dụng comment để bỏ qua các phần của dòng code
$x = 5 /* + 15 */ + 5;
echo $x;
?>

</body>
</html>
```

Kết quả: 10

Phân biệt chữ cái trong PHP

Trong PHP, mọi từ khóa (if, else, while, echo...), lớp, hàm, và các hàm do người dùng định nghĩa đều KHÔNG phân biệt chữ hoa/chữ thường.

Trong ví dụ dưới đây, ba câu lệnh echo đều như nhau:

```
<!DOCTYPE html>
<html>
<body>

<?php
    ECHO "Hello World!<br>";
    echo "Hello World!<br>";
    EcHo "Hello World!<br>";
?>

</body>
</html>
```

Kết quả:

```
Hello World!
Hello World!
Hello World!
```

Tuy nhiên, tất cả tên biến đều phân biệt chữ hoa/chữ thường

Trong ví dụ dưới đây, chỉ có một câu sẽ được hiển thị giá trị của biến $color (đó là bởi vì $color, $COLOR, và $coLOR được hiểu là ba biến khác nhau)

```
<!DOCTYPE html>
<html>
<body>

<?php
    $color = "red";
    echo "My car is " . $color . "<br>";
    echo "My house is " . $COLOR . "<br>";
    echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

Kết quả:

```
My car is red
My house is
My boat is
```

PHP Variables

Biến là một nơi chứa thông tin được lưu trữ.

Tạo (khai báo) biến trong PHP

Trong PHP, một biến được bắt đầu với ký tự $, theo sau đó là tên biến:

```
<?php
    $txt = "Hello world!";
    $x = 5;
    $y = 10.5;
?>
```

Sau khi thực thi các câu lệnh trên, biến **$txt** mang giá trị **Hello world!**, biến **$x** mang giá trị **5**, và biến **$y** mang giá trị **10.5**.

Chú ý: Khi bạn khai báo biến với giá trị dạng xâu ký tự, hãy thêm ký hiệu " " bao quanh xâu ký tự đó.

Chú ý: Không giống như các ngôn ngữ lập trình khác, PHP không có lệnh khai báo một biến. Nó được tạo ra mỗi khi bạn gán giá trị đầu tiên cho nó.

Biến trong PHP

Một biến có thể có tên ngắn (như x và y) hoặc những tên dài mô tả (như age, carname, total_volume...)

Quy tắc về biến trong PHP:

- Một biến được bắt đầu bởi ký tự $, theo sau là tên biến
- Tên biến phải bắt đầu bằng **chữ cái** hoặc dấu gạch dưới.
- Tên biến không được bắt đầu bằng chữ số
- Tên biến chỉ có thể chứa các ký tự A-z, 0-9 và _
- Các tên biến phân biệt nhau về chữ hoa/chữ thường ($age và $AGE là hai biến khác nhau)

Biến output

Câu lệnh PHP thường đưa dữ liệu ra màn hình

Ví dụ dưới đây sẽ hiển thị cả văn bản output và biến

```
<?php
    $txt = "W3Schools.com";
    echo "I love $txt!";
?>
```

Kết quả:

```
I love W3Schools.com!
```

Ví dụ dưới đây sẽ xuất ra kết quả giống như ví dụ trên. Ở đây sử dụng cách nối biến với đoạn text.

```
<?php
    $txt = "W3Schools.com";
    echo "I love " . $txt . "!";
?>
```

Ví dụ dưới đây sẽ xuất ra kết quả tổng hai biến

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

Kết quả: 9

Chú ý: Bạn sẽ được học nhiều hơn về câu lệnh echo và cách đưa dữ liệu ra màn hình vào chương kế tiếp.

PHP là ngôn ngữ không chặt chẽ về kiểu dữ liệu

Trong các ví dụ trên, để ý chúng ta thấy không đề cập đến kiểu dữ liệu trong PHP là gì.

PHP tự động chuyển các biến về đúng kiểu, phụ thuộc vào giá trị của chúng.

Trong các ngôn ngữ khác như C, C++ hay Java, người lập trình phải định nghĩa tên và kiểu dữ liệu cho biến trước khi sử dụng chúng.

Phạm vi dùng biến trong PHP

Trong PHP, biến có thể được định nghĩa ở bất kỳ nơi nào trong đoạn mã.

Phạm vi sử dụng biến là một phần của đoạn mã nơi các biến có thể được tham chiếu/sử dụng.

PHP có 3 phạm vi đặt biến:

- Local
- Global
- Static

Phạm vi toàn cục (global) và cục bộ (local)

Một biến được định nghĩa ngoài một hàm là biến toàn cục chỉ khi nó được truy xuất ngoài hàm đó.

```
<?php
    $x = 5; // biến toàn cục

    function myTest() {
        // sử dụng x trong hàm sẽ tạo ra lỗi
        echo "<p>Biến x bên trong hàm là: $x</p>";
    }
    myTest();

    echo "<p>Biến x bên ngoài hàm là: $x</p>";
?>
```

Kết quả:

```
Biến x bên trong hàm là:

Biến x bên ngoài hàm là: 5
```

Một biến được định nghĩa bên trong hàm được gọi là biến cục bộ chỉ cho phép truy xuất ở hàm đó.

```php
<?php
    function myTest() {
        $x = 5; // biến cục bộ
        echo "<p>Biến x trong hàm là: $x</p>";
    }
    myTest();

    // sử dụng x bên ngoài hàm gây lỗi
    echo "<p>Biến x ngoài hàm là: $x</p>";
?>
```

Kết quả:

```
Biến x trong hàm là: 5

Biến x ngoài hàm là:
```

Từ khóa global trong PHP

Từ khóa **global** được sử dụng để truy xuất đến biến toàn cục (global) từ bên trong hàm. Để thực hiện nó, sử dụng từ khóa **global** trước tên biến (sử dụng bên trong hàm)

```php
<?php
    $x = 5;
    $y = 10;

    function myTest() {
        global $x, $y;
        $y = $x + $y;
    }

    myTest();
    echo $y;
?>
```

Kết quả: 15

PHP cũng lưu trữ tất cả các biến trong mảng gọi là $GLOBAL[chỉ số]. Trong đó, chỉ số mang tên biến. Mảng này cũng được truy cập từ bên trong hàm và có thể sử dụng để thay đổi trực tiếp biến toàn cục.

Ví dụ trên có thể được viết lại như sau:

```
<?php
    $x = 5;
    $y = 10;

    function myTest() {
        $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
    }

    myTest();
    echo $y;
?>
```

Từ khóa static trong PHP

Thông thường, khi một hàm được thực thi, tất cả các biến trong nó đều bị xóa. Tuy nhiên, đôi khi chúng ta muốn một số biến cục bộ không bị xóa. Chúng ta cần nó cho công việc sau này.

Để thực hiện, sử dụng từ khóa **static** khi lần đầu tiên khai báo biến

```
<?php
    function myTest() {
        static $x = 0;
        echo $x;
        $x++;
    }
    myTest();
    myTest();
    myTest();
?>
```

Kết quả:

```
0
1
2
```

Biến $x được giữ lại sau khi thực thi hàm myTest() và sau mỗi lần thực thi nó tăng lên.

Sau đó, mỗi khi hàm được gọi, biến đó sẽ vẫn còn có những thông tin mà nó chứa từ lần cuối cùng các hàm được gọi.

Chú ý: Các biến vẫn là cục bộ đối với hàm.

PHP echo and print

Trong PHP có hai cách cơ bản để xuất thông tin: echo và print

echo và print hầu như đều giống nhau. Chúng sử dụng xuất thông tin ra màn hình.

Có một sự khác biệt nhỏ: **echo** không trả về giá trị trong khi **print** trả về giá trị để có thể sử dụng trong các biểu thức. **echo** có thể gồm nhiều tham số (mặc dù ít khi sử dụng) trong khi **print** chỉ có thể gồm một tham số. Đồng thời, **echo** cũng nhẹ và nhanh hơn **print.**

Câu lệnh echo trong PHP

Câu lệnh echo có thể được dùng với dấu ngoặc hoặc không: echo hoặc echo().

- Hiển thị văn bản

Trong ví dụ dưới đây sẽ trình bày văn bản xuất ra với câu lệnh echo (lưu ý rằng văn bản có thể chứa các thẻ HTML.

```
<?php
    echo "<h2>PHP is Fun!</h2>";
    echo "Hello world!<br>";
    echo "I'm about to learn PHP!<br>";
    echo "This ", "string ", "was ", "made ", "with multiple
parameters.";
?>
```

Kết quả:

PHP is Fun!
Hello world!
I'm about to learn PHP!
This string was made with multiple parameters.

- Hiển thị các biến

Ví dụ dưới đây trình bày cách xuất ra văn bản và các biến với câu lệnh echo

```
<?php
    $txt1 = "Learn PHP";
    $txt2 = "W3Schools.com";
    $x = 5;
    $y = 4;
```

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

Kết quả:

Learn PHP
Study PHP at W3Schools.com
9

Câu lệnh print trong PHP

Câu lệnh print có thể được dùng với dấu ngoặc đơn hoặc không: print hoặc print()

- Hiển thị văn bản

Ví dụ dưới đây trình bày cách xuất ra văn bản với câu lệnh print (lưu ý rằng văn bản có thể chứa các thẻ HTML)

```
<?php
    print "<h2>PHP is Fun!</h2>";
    print "Hello world!<br>";
    print "I'm about to learn PHP!";
?>
```

- Hiển thị các biến

Ví dụ sau đây trình bày cách xuất ra văn bản cùng với các biến với câu lệnh print

```
<?php
    $txt1 = "Learn PHP";
    $txt2 = "W3Schools.com";
    $x = 5;
    $y = 4;

    print "<h2>$txt1</h2>";
    print "Study PHP at $txt2<br>";
    print $x + $y;
?>
```

PHP Data types

Biến có thể chứa dữ liệu thuộc nhiều dạng khác nhau, và các dạng dữ liệu khác nhau có thể làm việc khác nhau.

PHP hỗ trợ những kiểu biến sau đây:

- String
- Integer
- Float (còn gọi là double)
- Boolean
- Array
- Object
- NULL
- Resource

Chuỗi trong PHP

Một chuỗi là dãy các ký tự, như "Hello world!"

Một chuỗi có thể chứa trong dấu nháy, có thể sử dụng dấu nháy đơn hoặc nháy kép:

```
<?php
    $x = "Hello world!";
    $y = 'Hello world!';

    echo $x;
    echo "<br>";
    echo $y;
?>
```

Kết quả:

```
Hello world!
Hello world!
```

Số nguyên trong PHP

Một số nguyên là kiểu dữ liệu không phải là số thập phân giữa -2147,483,648 đến 2,147,483,647.

Quy tắc cho số nguyên

- Chứa ít nhất một chữ số
- Không chứa dấu phẩy thập phân
- Có thể là số âm hoặc dương
- Số nguyên có thể được chia làm ba định dạng: hệ 10, hệ 16 và hệ 8.

Trong ví dụ sau đây $x là một biến nguyên. Hàm **var_dump()** trong PHP trả về kiểu dữ liệu và giá trị:

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

Kết quả:

```
int(5985)
```

Số thực trong PHP

Một số thực (dấu phẩy động) là một số thập phân hoặc số có dạng hàm mũ.

Trong ví dụ sau $x là số thực. Hàm **var_dump()** trong PHP trả về kiểu dữ liệu và giá trị của $x:

```
<?php
    $x = 10.365;
    var_dump($x);
?>
```

Kết quả:

```
float(10.365)
```

Kiểu logic trong PHP

Biến kiểu boolean biểu thị hai trạng thái: ĐÚNG (true) hoặc SAI (false).

```
$x = true;
$y = false;
```

Boolean thường được sử dụng để kiểm chứng các điều kiện. Bạn sẽ được học nhiều hơn về biểu thức điều kiện trong chương kế tiếp.

Mảng trong PHP

Một mảng lưu nhiều giá trị trong cùng một biến.

Trong ví dụ sau, biến $cars là một mảng. Hàm var_dump() trong PHP trả về kiểu dữ liệu và giá trị:

```
<?php
    $cars = array("Volvo","BMW","Toyota");
    var_dump($cars);
?>
```

Kết quả:

```
array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6)
"Toyota" }
```

Bạn sẽ được học nhiều hơn về mảng trong chương kế tiếp.

Kiểu Object trong PHP

Object là kiểu dữ liệu lưu trữ dữ liệu và thông tin cách xử lý dữ liệu đó.

Trong PHP, một đối tượng phải được định nghĩa rõ ràng.

Trước tiên chúng ta phải định nghĩa class cho đối tượng. Để làm điều đó, chúng ta sử dụng từ khóa **class**. Một class (lớp) là một cấu trúc chứa đựng thuộc tính và phương thức:

```
<?php
    class Car {
        function Car() {
            $this->model = "VW";
        }
    }

    // Khởi tạo đối tượng
    $herbie = new Car();
```

```
    // Hiển thị thuộc tính của đối tượng
    echo $herbie->model;
?>
```

Kết quả: VW

Bạn sẽ được học nhiều hơn về kiểu đối tượng trong chương kế tiếp.

Giá trị NULL trong PHP

NULL là một kiểu dữ liệu đặc biệt chỉ chứa một giá trị: NULL.

Một biến thuộc kiểu NULL là một biến mà không có giá trị nào được gán cho nó.

Tip: Nếu như một biến được khởi tạo mà không có giá trị, nó được tự động gán giá trị NULL.

Các biến có thể bị rỗng bởi việc thiết lập giá trị là NULL.

```
<?php
    $x = "Hello world!";
    $x = null;
    var_dump($x);
?>
```

Kết quả: NULL

Resource trong PHP

Là một dạng dữ liệu không tường minh về kiểu. Nó chứa các tham chiếu đến các chức năng và các nguồn dữ liệu bên ngoài PHP.

Một ví dụ phổ biến của việc sử dụng các kiểu dữ liệu tài nguyên là lời gọi cơ sở dữ liệu.

Chúng ta sẽ không đề cập về resource trong PHP ở đây, vì nó là một chủ đề nâng cao.

PHP Strings

Một chuỗi là một dãy các ký tự, như "Hello world!".

Hàm xử lý chuỗi trong PHP

Trong chương này chúng ta sẽ điểm qua một vài hàm thông dụng để vận dụng cho chuỗi.

Lấy độ dài một chuỗi

Hàm **strlen**() trong PHP trả về độ dài của một chuỗi.

Ví dụ dưới đây trả về độ dài của chuỗi "Hello world!":

```
<?php
    echo strlen("Hello world!"); // outputs 12
?>
```

Kết quả: 12

Đếm số từ trong một chuỗi

Hàm **str_word_count**() trong PHP đếm số lượng các từ trong một chuỗi:

```
<?php
    echo str_word_count("Hello world!"); // outputs 2
?>
```

Kết quả: 2

Đảo một chuỗi

Hàm **strrev**() trong PHP dùng để đảo ngược một chuỗi:

```
<?php
    echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

Kết quả:

```
!dlrow olleH
```

Tìm kiếm một văn bản nhất định trong chuỗi

Hàm **strpos**() trong PHP tìm kiếm một văn bản nhất định trong chuỗi.

Nếu kết quả là tìm thấy, hàm trả về vị trí ký tự đầu tiên khớp với văn bản cần tìm. Nếu không có kết quả nào khớp hàm trả về giá trị FALSE.

Ví dụ dưới đây tìm kiếm chữ "world" trong chuỗi "Hello world!":

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

Kết quả: 6

Chú ý: chỉ số của ký tự đầu tiên trong chuỗi là 0 (không phải 1)

Thay thế văn bản trong chuỗi

Hàm **str_replace**() trong PHP thay thế những ký tự trong chuỗi với những ký tự khác.

Ví dụ dưới đây thay thế từ "world" với từ "Dolly":

```
<?php
    echo str_replace("world", "Dolly", "Hello world!"); // outputs
Hello Dolly!
?>
```

Kết quả:

```
Hello Dolly!
```

Tham khảo đầy đủ về chuỗi trong PHP

Để tham khảo đầy đủ các hàm về chuỗi, xem thêm tại:

http://www.w3schools.com/php/php_ref_string.asp

Trang tham khảo chuỗi trong PHP chứa mô tả và ví dụ sử dụng cho tất cả các hàm.

PHP Constants

Các hằng số cũng như biến ngoại trừ một khi chúng đã được định nghĩa thì không thể thay đổi hay định nghĩa lại.

Các hằng số trong PHP

Một hằng số là một định danh (tên) cho một giá trị đơn nhất. Giá trị này không thể thay đổi trong suốt đoạn mã.

Tên hằng số hợp lệ bắt đầu bởi một chữ cái hoặc dấu gạch dưới (**không** có ký tự $ trước tên hằng)

Chú ý: Không như các biến, hằng số sẽ tự thiết lập là giá trị toàn cục trên toàn bộ đoạn mã.

Tạo một hằng số trong PHP

Để tạo một hằng số, sử dụng hàm **define**().

Cú pháp: **define**(*name, value, case-insensitive*)

Các tham số:

- Name: tên hằng
- Value: Giá trị
- Case-insensitive: được thiết lập để không phân biệt chữ hoa/chữ thường. Mặc định là **false**.

Ví dụ dưới đây tạo ra hằng số với tên **phân biệt chữ hoa/chữ thường:**

```
<?php
    define("GREETING", "Welcome to W3Schools.com!");
    echo GREETING;
?>
```

Ví dụ dưới đây tạo ra hằng số với tên **không phân biệt chữ hoa/chữ thường:**

```
<?php
    define("GREETING", "Welcome to W3Schools.com!", true);
    echo greeting;
?>
```

Kết quả của cả 2 ví dụ:

```
Welcome to W3Schools.com!
```

Các hằng số là toàn cục

Các hằng số tự động là phạm vi toàn cục và có thể dùng trên toàn bộ đoạn mã.

Ví dụ dưới đây sử dụng hằng số bên trong hàm, ngay cả khi nó được định nghĩa bên ngoài hàm:

```
<?php
    define("GREETING", "Welcome to W3Schools.com!");

    function myTest() {
        echo GREETING;
    }

    myTest();
?>
```

Kết quả:

```
Welcome to W3Schools.com!
```

PHP Operators

Các toán tử được sử dụng để biểu diễn các biểu thức trên biến và giá trị.

PHP chia các toán tử thành những nhóm sau:

- Toán tử số học
- Toán tử gán
- Toán tử so sánh
- Toán tử tăng/giảm
- Toán tử logic
- Toán tử trên chuỗi
- Toán tử trên mảng

Toán tử số học trong PHP

Các toán tử số học trong PHP được sử dụng với các giá trị là số để biểu diễn hầu hết các biểu thức toán học như cộng, trừ, nhân...

Toán tử	Tên	Ví dụ	Kết quả
+	Cộng	$x + $y	Tổng của $x và $y
-	Trừ	$x - $y	Hiệu của $x và $y
*	Nhân	$x * $y	Tích của $x và $y
/	Chia	$x / $y	Thương của $x và $y
%	Dư	$x % $y	Phần dư của phép chia $x cho $y
**	Mũ	$x ** $y	Kết quả $x mũ $y

Toán tử gán trong PHP

Toán tử gán trong PHP được sử dụng với những giá trị số để ghi giá trị vào biến.

Toán tử gán cơ bản trong PHP là “=”. Nó có nghĩa là toán hạng bên trái được gán giá trị của biểu thức bên phải.

Phép gán	Giống như...	Mô tả
x = y	x = y	Các toán hạng bên trái được thiết lập với giá trị của biểu thức bên phải
x += y	x = x + y	Phép cộng
x -= y	x = x – y	Phép trừ
x *= y	x = x * y	Phép nhân
x /= y	x = x / y	Phép chia
x %= y	x = x % y	Lấy dư

Toán tử so sánh trong PHP

Toán tử so sánh trong PHP được sử dụng để so sánh hai giá trị (số hoặc chuỗi):

Toán tử	Tên	Ví dụ	Kết quả
==	Bằng nhau	$x == $y	Trả về **đúng** nếu $x bằng với $y
===	Giống nhau	$x === $y	Trả về **đúng** nếu $x bằng với $y và chúng có cùng kiểu
!==	Không bằng nhau	$x != $y	Trả về **đúng** nếu $x không bằng $y
<>	Không bằng nhau	$x <> $y	Trả về **đúng** nếu $x không bằng $y
!==	Không giống nhau	$x !== $y	Trả về **đúng** nếu $x không bằng $y, và chúng không cùng kiểu.
>	Lớn hơn	$x > $y	Trả về **đúng** nếu $x lớn hơn $y
<	Nhỏ hơn	$x < $y	Trả về **đúng** nếu $x nhỏ hơn $y
>=	Lớn hơn hoặc bằng	$x >= $y	Trả về **đúng** nếu $x lớn hơn hoặc bằng với $y
<=	Nhỏ hơn hoặc bằng	$x <= $y	Trả về **đúng** nếu $x nhỏ hơn hoặc bằng với $y

Toán tử tăng/giảm trong PHP

Toán tử	Tên	Mô tả
++$x	Tăng trước	Tăng $x một đơn vị, sau đó trả về $x
$x++	Tăng sau	Trả về $x, sau đó tăng $x một đơn vị
--$x	Giảm trước	Giảm $x một đơn vị, sau đó trả về $x
$x++	Giảm sau	Trả về $x, sau đó giảm $x một đơn vị

Toán tử logic trong PHP

Toán tử logic trong PHP được dùng để ghép các mệnh đề điều kiện với nhau

Toán tử	Tên	Ví dụ	Kết quả
and	Và	$x and $y	**Đúng** nếu cả $x **và** $y đều đúng
or	Hoặc	$x or $y	**Đúng** nếu một trong $x **hoặc** $y đúng
xor	Khác dấu	$x xor $y	**Đúng** nếu một trong $x **hoặc** $y đúng, nhưng **không** phải cả hai.
&&	Và logic	$x && $y	Đúng nếu cả hai **đều** đúng
||	Hoặc logic	$x || $y	Đúng nếu một trong $x **hoặc** $y đúng
!	Đảo	!$x	Đúng nếu $x **không** đúng

Toán tử chuỗi trong PHP

PHP có hai toán tử được thiết kế đặc biệt dành cho các chuỗi.

Toán tử	Tên	Ví dụ	Mô tả
.	Nối	$txt1 . $txt2	Nối chuỗi $txt1 với chuỗi $txt2
.=	Gán thêm	$txt1 .= $txt2	Gắn thêm chuỗi $txt2 vào $txt1

Toán tử mảng trong PHP

Các toán tử mảng trong PHP được sử dụng để so sánh các mảng

Toán tử	Tên	Ví dụ	Kết quả
+	Hợp	$x + $y	Hợp mảng $x với mảng $y
==	Bằng nhau	$x == $y	Trả về **đúng** nếu như $x và $y có từng cặp chỉ số/giá trị giống nhau
===	Giống nhau	$x === $y	Trả về **đúng** nếu như $x và $y có từng cặp chỉ số/giá trị giống nhau, cùng thứ tự và cùng kiểu.
!=	Không như nhau	$x != $y	Trả về **đúng** nếu $x không bằng $y
<>	Không như nhau	$x <> $y	Trả về **đúng** nếu $x không bằng $y
!==	Khác nhau	$x !== $y	Trả về **đúng** nếu $x không giống $y

PHP if...else...elseif

Các câu lệnh điều kiện được sử dụng để biểu diễn những hành động khác nhau dựa trên các điều kiện khác nhau.

Câu lệnh điều kiện trong PHP

Thông thường khi viết code, bạn muốn biểu diễn các hành động khác nhau cho những trường hợp khác nhau. Bạn có thể sử dụng câu lệnh điều kiện trong code để thực hiện nó.

Trong PHP chúng ta có các câu lệnh điều kiện sau:

- Câu lệnh **if** – thực hiện một vài đoạn mã nếu một điều kiện là đúng
- Câu lệnh **if...else –** thực hiện một vài đoạn mã nếu điều kiện là đúng và các đoạn mã còn lại nếu điều kiện đó là sai.
- Câu lệnh **if...elseif...else** – thực hiện những đoạn mã khác nhau khi có nhiều hơn hai điều kiện.
- Câu lệnh **switch** – lựa chọn một trong nhiều đoạn mã để thực hiện.

Câu lệnh if trong PHP

Câu lệnh if thực hiện một vài đoạn mã nếu điều kiện là đúng.

Cú pháp:

```
if (điều kiện) {
   đoạn mã được thực hiện nếu điều kiện là đúng;
}
```

Ví dụ dưới đây sẽ xuất ra "Have a good day!" nếu thời gian hiện tại (giờ) nhỏ hơn 20.

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}
?>
```

Câu lệnh if...else trong PHP

Câu lệnh if...else thực hiện một vài đoạn mã nếu điều kiện là đúng và các đoạn mã còn lại nếu điều kiện là sai.

Cú pháp:

```
if (điều kiện) {
   đoạn mã được thực hiện nếu điều kiện là đúng;
} else {
   đoạn mã được thực hiện nếu điều kiện là sai;
}
```

Ví dụ dưới đây sẽ xuất ra "Have a good day!" Nếu thời gian hiện tại (giờ) nhỏ hơn 20, và "Have a good night!" với các trường hợp còn lại:

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

Câu lệnh if...elseif...else trong PHP

Câu lệnh if...elseif...else thực hiện các đoạn mã khác nhau khi có nhiều hơn hai trường hợp.

Cú pháp:

```
if (điều kiện) {
   đoạn mã được thực hiện nếu điều kiện này là đúng;
} elseif (điều kiện) {
   đoạn mã được thực hiện nếu điều kiện này là đúng;
} else {
   đoạn mã được thực hiện nếu tất cả các điều kiện trên là sai;
}
```

Ví dụ dưới đây sẽ xuất ra "Have a good morning!" nếu thời gian hiện tại nhỏ hơn 10, và "Have a good day!" nếu thời gian hiện tại nhỏ hơn 20. Các trường hợp còn lại xuất ra "Have a good night!":

```
<?php
$t = date("H");

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

Chú ý: thời gian trên được lấy từ server, chứ không phải trên client.

Câu lệnh switch trong PHP

Câu lệnh switch sẽ được trình bày trong chương kế tiếp.

PHP switch

Câu lệnh switch trong PHP được sử dụng để biểu diễn các hành động khác nhau dựa trên các điều kiện khác nhau.

Câu lệnh switch trong PHP

Sử dụng câu lệnh switch để **lựa chọn một trong các đoạn mã sẽ được thực hiện.**

Cú pháp:

```
switch (n) {
  case trường hợp 1:
    đoạn mã được thực hiện nếu n=trường hợp 1; break;
  case trường hợp 2:
    đoạn mã được thực hiện nếu n=trường hợp 2; break;
  case trường hợp 3:
    đoạn mã được thực hiện nếu n=trường hợp 3; break;
  ...
  default:
    đoạn mã được thực hiện nếu n khác với tất cả các trường hợp trên;
}
```

Đây là cách chúng hoạt động: Đầu tiên chúng ta có một biểu thức đơn n (thường là một biến), được đánh giá một lần. Giá trị của biểu thức sau đó được so sánh với các giá trị cho mỗi trường hợp trong cấu trúc. Nếu trùng với một trường hợp, các khối mã liên quan đến trường hợp được thực thi. Sử dụng **break** để ngăn chặn các mã thực thi vào trường hợp tiếp theo tự động. Từ khóa **default** được sử dụng nếu không có trường hợp nào trùng với biểu thức đơn được tìm thấy. Ví dụ:

```php
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
```

```
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

Kết quả:

```
Your favorite color is red!
```

PHP while loops

Vòng lặp while trong PHP thực hiện đoạn mã trong khi điều kiện được đưa ra là đúng.

Vòng lặp trong PHP

Thông thường khi bạn viết code, bạn muốn những đoạn mã giống nhau được thực thi và tiếp tục chạy lại trong một dòng. Thay vì phải thêm nhiều dòng giống nhau trong một đoạn mã, chúng ta sử dụng vòng lặp để biểu diễn công việc như thế.

Trong PHP, chúng ta có các câu lệnh lặp như sau:

- **while**: Lặp lại một khối mã miễn là các điều kiện quy định còn đúng
- **do...while**: Lặp lại một khối mã một lần, và sau đó lặp lại vòng lặp nếu như điều kiện quy định là đúng
- **for**: Lặp lại đoạn mã với số lần lặp xác định
- **foreach**: Lặp lại đoạn mã cho tất cả các phần tử trong một mảng.

Vòng lặp while trong PHP

Lặp lại một khối mã trong khi điều kiện quy định là đúng.

Cú pháp:

```
while (điều kiện là đúng) {
  đoạn mã được thực hiện;
}
```

Ví dụ dưới đây đầu tiên đặt một biến $x = 1. Sau đó, trong khi vòng lặp sẽ tiếp tục chạy miễn là $x nhỏ hơn hoặc bằng 5. $x sẽ tăng thêm 1 sau mỗi lần chạy vòng lặp. ($x ++):

```php
<?php
$x = 1;

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

Kết quả:

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

Vòng lặp do...while trong PHP

Vòng lặp do...while sẽ luôn thực hiện đoạn mã một lần, sau đó nó kiểm tra điều kiện, và lặp lại vòng lặp trong khi điều kiện quy định là đúng.

Cú pháp:

```
do {
   đoạn mã được thực hiện;
} while (điều kiện là đúng);
```

Ví dụ dưới đây đầu tiên đặt một biến $ x = 1. Sau đó, vòng lặp do while sẽ in ra một vài kết quả, tiếp theo tăng biến $x lên 1. Sau đó điều kiện được kiểm tra ($x có nhỏ hơn hoặc bằng 5?), và vòng lặp sẽ tiếp tục chạy nếu $x nhỏ hơn hoặc bằng 5:

```php
<?php
$x = 1;

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

Kết quả:

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

Chú ý rằng trong một vòng lặp do...while thì điều kiện sẽ được kiểm tra sau khi thực hiện các câu lệnh trong vòng lặp. Điều này có nghĩa là vòng lặp do...while sẽ thực hiện các câu lệnh của nó ít nhất một lần, thậm chí nếu điều kiện là sai lần đầu tiên.

Ví dụ dưới đây đặt biến $x bằng 6, sau đó nó chạy vòng lặp, và sau đó điều kiện được kiểm tra:

```
<?php
$x = 6;

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

Kết quả:

```
The number is: 6
```

Vòng lặp for và foreach sẽ được trình bày trong chương kế tiếp.

PHP for loops

Vòng lặp for trong PHP thực hiện một đoạn mã với số lần lặp xác định.

Vòng lặp for trong PHP

Vòng lặp for được sử dụng khi bạn biết có bao nhiêu lần đoạn mã được chạy.

Cú pháp:

```
for (khởi tạo biến đếm; kiểm tra biến đếm; tăng/giảm biến đếm) {
  đoạn mã được thực hiện;
}
```

Các tham số:

- *Khởi tạo*: Khởi tạo giá trị đếm trong vòng lặp.
- *Kiểm tra:* Xem xét mỗi lần lặp. Nếu như giá trị là TRUE, vòng lặp tiếp tục thực hiện. Ngược lại, vòng lặp kết thúc.
- *Tăng/giảm:* Tăng hoặc giảm giá trị biến đếm.

Ví dụ dưới đây hiển thị các số từ 0 đến 10.

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

Kết quả:

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

Vòng lặp foreach trong PHP

Vòng lặp foreach chỉ hoạt động trên mảng, và được sử dụng để lặp qua mỗi cặp key/value trong một mảng.

Cú pháp:

```
foreach ($mảng as $giá trị) {
   đoạn mã được thực hiện;
}
```

Đối với mỗi lần lặp, giá trị của các phần tử mảng hiện tại được gán cho $giá trị và con trỏ mảng được di chuyển lên một, cho đến khi nó đạt đến các phần tử mảng cuối cùng.

Ví dụ sau đây chứng minh vòng lặp sẽ xuất ra giá trị nhất định của mảng ($colors):

```php
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

Kết quả:

```
red

green

blue

yellow
```

Chúng ta sẽ học nhiều hơn về mảng ở chương kế tiếp.

PHP Functions

Sức mạnh thực sự của PHP đến từ các hàm của nó. PHP có hơn 1000 hàm xây dựng sẵn.

Hàm người dùng định nghĩa trong PHP

Bên cạnh các hàm dựng sẵn trong PHP, chúng ta có thể tạo ra các hàm riêng.

Hàm là một khối các câu lệnh có thể được lặp lại trong chương trình.

Hàm sẽ không được thực thi ngay lập tức khi tải trang.

Hàm sẽ được thực thi thông qua lời gọi hàm.

Tạo ra hàm người dùng định nghĩa trong PHP

Một hàm người dùng định nghĩa bắt đầu với từ khóa "function":

Cú pháp:

```
function tên_function() {
  đoạn mã được thực hiện;
}
```

Chú ý: Tên hàm có thể bắt đầu bởi chữ cái hoặc dấu _ (không được là số)

Tip: Nên đặt tên function có liên quan đến những gì function thực hiện!

Tên hàm không phân biệt chữ hoa chữ thường: writeHello() và WriteHello() như nhau

Trong ví dụ dưới đây, chúng ta tạo ra hàm có tên "writeMsg()". Việc mở dấu ngoặc nhọn { cho thấy sự bắt đầu của đoạn mã và dấu } cho biết sự kết thúc của hàm. Hàm xuất ra "Hello world!". Để gọi hàm, chỉ cần viết tên của nó.

```php
<?php
function writeMsg() {
    echo "Hello world!";
}
```

```
writeMsg(); // gọi function
?>
```

Kết quả:

```
Hello world!
```

Đối số của hàm trong PHP

Thông tin có thể được truyền thông qua các đối số của hàm. Một đối số giống như một biến.

Đối số được quy định sau tên hàm, bên trong dấu ngoặc đơn (). Bạn có thể thêm nhiều đối số nếu bạn muốn, chỉ cần tách chúng bằng dấu phẩy.

Ví dụ sau có một hàm với một đối số ($fname). Khi familyName() được gọi, chúng ta truyền một tên (ví dụ Jani), và tên được sử dụng sử dụng bên trong hàm, mà đầu ra của First name thì khác nhau, nhưng Last name thì giống nhau:

```
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

Kết quả:

```
Jani Refsnes.
Hege Refsnes.
Stale Refsnes.
Kai Jim Refsnes.
Borge Refsnes.
```

Ví dụ sau đây có một hàm với 2 đối số ($fname và $year):

```php
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

Kết quả:

```
Hege Refsnes. Born in 1975
Stale Refsnes. Born in 1978
Kai Jim Refsnes. Born in 1983
```

Đối số mặc định trong PHP

Ví dụ dưới đây trình bày cách sử dụng đối số mặc định. Nếu chúng ta gọi hàm setHeight() mà không bao gồm bất cứ đối số nào thì đối số sẽ là đối số mặc định:

```php
<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // sẽ sử dụng giá trị mặc định là 50
setHeight(135);
setHeight(80);
?>
```

Kết quả:

```
The height is : 350

The height is : 50

The height is : 135

The height is : 80
```

Hàm trả về giá trị trong PHP

Để trả hàm trả về giá trị, sử dụng câu lệnh **return:**

```php
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

Kết quả:

```
5 + 10 = 15
7 + 13 = 20
2 + 4 = 6
```

PHP Arrays

Một mảng lưu nhiều giá trị trong đó:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] .
".";
?>
```

Kết quả:

```
I like Volvo, BMW and Toyota.
```

Mảng là gì?

Một mảng là một biến đặc biệt, nó có mang nhiều hơn một giá trị tại một thời điểm.

Nếu bạn có một danh sách các mục (ví dụ như danh sách tên các xe), việc lưu các xe trong các biến đơn lẻ có thể như sau:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

Tuy nhiên, nếu bạn muốn duyệt qua hết các xe và tìm một chiếc duy nhất? Và nếu bạn có không chỉ 3 chiếc xe, mà là 300 thì sẽ như thế nào?

Giải pháp là tạo một mảng!

Một mảng có thể lưu nhiều giá trị dưới tên duy nhất, và bạn có thể truy cập từng phần tử dựa vào chỉ số mảng.

Tạo mảng trong PHP

Trong PHP, hàm **array**() được sử dụng để tạo một mảng:

```
array();
```

Trong PHP, có 3 kiểu mảng:

- Mảng chỉ số: Mảng với chỉ mục dạng số

- Mảng liên kết: Mảng với chỉ mục được đặt tên
- Mảng nhiều chiều: Mảng có chứa một hoặc nhiều mảng

Mảng chỉ mục trong PHP

Có hai cách để tạo mảng chỉ mục:

Các chỉ mục có thể được gán tự động (chỉ mục luôn bắt đầu từ 0), như:

```
$cars = array("Volvo", "BMW", "Toyota");
```

Hoặc các chỉ mục có thể được gán thủ công:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

Ví dụ sau đây tạo một mảng chỉ mục được đặt tên là $cars, gán 3 phần tử cho nó, và sau đó in ra văn bản có chứa các giá trị của mảng:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] .
".";
?>
```

Kết quả:

```
I like Volvo, BMW and Toyota.
```

Lấy độ dài của mảng – Hàm count()

Hàm **count**() được sử dụng để trả về độ dài (số lượng phần tử) của mảng:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

Kết quả: 3

Lặp qua mảng chỉ mục

Để lặp qua và in ra tất cả các giá trị của mảng chỉ mục, bạn có thể sử dụng vong lặp **for**, như sau:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

Kết quả:

```
Volvo
BMW
Toyota
```

Mảng liên kết trong PHP

Các mảng liên kết là những mảng mà chỉ mục (key) được bạn đặt tên cho chúng.

Có hai cách để tạo ra mảng liên kết:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

hoặc:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

Các chỉ mục được đặt tên có thể được dùng trong đoạn mã:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Kết quả:

```
Peter is 35 years old.
```

Lặp qua mảng liên kết

Để lặp qua và in ra tất cả các giá trị của mảng liên kết, bạn có thể sử dụng vòng lặp **foreach**, như sau:

```php
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value . "<br>";
}
?>
```

Kết quả:

```
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

Mảng nhiều chiều

Mảng nhiều chiều sẽ được trình bày trong mục PHP nâng cao.

Tham khảo đầy đủ về mảng trong PHP

Để tham khảo đầy đủ tất cả các hàm về mảng trong PHP, xem tại:

http://www.w3schools.com/php/php_ref_array.asp

PHP Sorting Arrays

Các phần tử trong một mảng có thể được sắp xếp theo vần hoặc theo thứ tự, giảm dần hoặc tăng dần.

Các hàm sắp xếp cho mảng trong PHP

Trong chương này, chúng ta sẽ đi tìm hiểu qua vê các hàm sắp xếp mảng trong PHP dưới đây:

- sort(): Sắp xếp mảng theo thứ tự tăng dần
- rsort(): Sắp xếp mảng theo thứ tự giảm dần
- asort(): Sắp xếp mảng liên kết tăng dần, theo giá trị (value)
- ksort(): Sắp xếp mảng liên kết tăng dần, theo chỉ mục (key)
- arsort(): Sắp xếp mảng liên kết giảm dần, theo giá trị (value)
- krsort(): Sắp xếp mảng liên kết giảm dần, theo chỉ mục (key)

Sắp xếp mảng theo thứ tự tăng dần – sort()

Ví dụ sau đây sắp xếp các phần tử thuộc mảng $cars tăng dần theo thứ tự vần:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

Kết quả:

```
BMW
Toyota
Volvo
```

Ví dụ sau đây sắp xếp các phần tử thuộc mảng $numbers giảm dần theo thứ tự số:

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

Kết quả:

```
2
4
6
11
22
```

Sắp xếp mảng theo thứ tự giảm dần – rsort()

Ví dụ sau đây sắp xếp các phần tử thuộc mảng $cars giảm dần theo thứ tự vần:

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);
?>
```

Kết quả:

```
Volvo
Toyota
BMW
```

Ví dụ sau đây sắp xếp các phần tử thuộc mảng $numbers giảm dần theo thứ tự số:

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

Kết quả:

```
22
11
6
4
2
```

Sắp xếp mảng tăng dần, theo giá trị - asort()

Ví dụ sau đây sắp xếp một mảng liên kết tăng dần, theo thứ tự về giá trị:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);
?>
```

Kết quả:

```
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

Sắp xếp mảng tăng dần, theo thứ tự chỉ mục – ksort()

Ví dụ sau đây sắp xếp một mảng liên kết tăng dần, theo thứ tự về chỉ mục:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
?>
```

Kết quả:

```
Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35
```

Sắp xếp mảng giảm dần, theo thứ tự giá trị - arsort()

Ví dụ sau đây sắp xếp một mảng liên kết giảm dần, theo thứ tự giá trị:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);
?>
```

Kết quả:

```
Key=Joe, Value=43
Key=Ben, Value=37
Key=Peter, Value=35
```

Sắp xếp mảng giảm dần, theo thứ tự chỉ mục – krsort()

Ví dụ sau đây sắp xếp một mảng liên kết giảm dần, theo thứ tự chỉ mục:

```php
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
krsort($age);
?>
```

Kết quả:

```
Key=Peter, Value=35
Key=Joe, Value=43
Key=Ben, Value=37
```

Tham khảo đầy đủ về mảng trong PHP

Để tham khảo đầy đủ tất cả các hàm về mảng trong PHP, xem tại:

http://www.w3schools.com/php/php_ref_array.asp

PHP Superglobals

Các biến siêu toàn cục (superglobal) được giới thiệu từ PHP 4.1.0, và là các biến được định nghĩa sẵn, luôn có hiệu lực trong mọi phạm vi.

Biến toàn cục – siêu toàn cục trong PHP

Hầu hết các biến được định nghĩa sẵn trong PHP là "siêu toàn cục", nghĩa là chúng luôn có thể truy cập bất kể phạm vi nào – và bạn có thể truy xuất chúng từ mọi hàm, lớp hoặc file mà không cần làm gì đặc biệt cả.

Các biến siêu toàn cục trong PHP là:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

Chương này sẽ trình bày một vài biến siêu toàn cục, và phần còn lại sẽ được diễn giải trong các chương kế tiếp.

$GLOBALS trong PHP

$GLOBALS là một biến siêu toàn cục trong PHP được sử dụng để truy cập đến các biến toàn cục từ bất cứ nơi nào trong mã PHP (thường từ bên trong hàm hoặc phương thức)

PHP lưu toàn bộ các biến toàn cục trong mảng gọi là $GLOBALS[chỉ mục]. Chỉ mục là tên các biến.

Ví dụ dưới đây cho thấy cách sử dụng biến siêu cục bộ $GLOBALS

```
<?php
$x = 75;
$y = 25;
```

```
function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

Kết quả: 100

Trong ví dụ trên, khi z là biến đi kèm với mảng $GLOBALS, nó có thể truy xuất từ bên ngoài hàm.

$_SERVER trong PHP

$_SERVER là biến siêu toàn cục trong PHP, nó giữ thông tin về tiêu đề, đường dẫn và các vị trí các đoạn mã.

Ví dụ dưới đây trình bày cách sử dụng một số phần tử trong $_SERVER

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

Kết quả (không cố định):

```
/php/demo_global_server.php
www.w3schools.com
www.w3schools.com
http://www.w3schools.com/php/showphp.asp?filename=demo_global_server
Mozilla/5.0 (Windows NT 6.1; rv:47.0) Gecko/20100101 Firefox/47.0
/php/demo_global_server.php
```

Bảng sau đây liệt kê các thành phần quan trọng có thể đi với $_SERVER

Phần tử/Mã	Mô tả
$_SERVER['PHP_SELF']	Trả về tên tập tin mà server đang thực thi
$_SERVER['GATEWAY_INTERFACE']	Trả về phiên bản CGI server đang sử dụng
$_SERVER['SERVER_ADDR']	Trả về địa chỉ IP của host server
$_SERVER['SERVER_NAME']	Trả về tên của host server (như w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Trả về chuỗi định danh server (như Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Trả về tên và phiên bản của giao thức truyền tin (như HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Trả về phương thức yêu cầu được sử dụng để truy cập trang (như POST)
$_SERVER['REQUEST_TIME']	Trả về mốc thời gian bắt đầu yêu cầu (như 1377687496)
$_SERVER['QUERY_STRING']	Trả về chuỗi truy vấn nếu trang web được truy cập thông qua một chuỗi truy vấn
$_SERVER['HTTP_ACCEPT']	Trả về tiêu đề chấp nhận từ yêu cầu hiện tại
$_SERVER['HTTP_ACCEPT_CHARSET']	Trả về kiểu định dạng ký tự tiêu đề từ yêu cầu hiện tại (như utf-8, ISO-8859)
$_SERVER['HTTP_HOST']	Trả về tiêu đề host từ yêu cầu hiện tại
$_SERVER['HTTP_REFERER']	Trả về URL đầy đủ của trang hiện tại
$_SERVER['HTTPS']	Xác định có phải tập lệnh truy vấn thông qua giao thức HTTP an toàn

$_SERVER['REMOTE_ADDR']	Trả về địa chỉ IP từ nơi người dùng đang xem trang hiện tại
$_SERVER['REMOTE_HOST']	Trả về tên host từ nơi người dùng đang xem trang hiện tại
$_SERVER['REMOTE_PORT']	Trả về cổng đang được sử dụng trên máy người dùng để truyền thông với máy chủ web
$_SERVER['SCRIPT_FILENAME']	Trả về đường dẫn tuyệt đối của tập lệnh đang được thực thi
$_SERVER['SERVER_ADMIN']	Trả về giá trị cho các chỉ thị SERVER_ADMIN trong tập tin cấu hình máy chủ web
$_SERVER['SERVER_PORT']	Trả về cổng trên máy đặt server đang được sử dụng bởi web server để truyền thông (như cổng 80)
$_SERVER['SERVER_SIGNATURE']	Trả về phiên bản server và tên host ảo được thêm vào trang do máy chủ tạo ra
$_SERVER['PATH_TRANSLATED']	Trả về file hệ thống dựa trên đường dẫn tới tập lệnh hiện tại
$_SERVER['SCRIPT_NAME']	Trả về đường dẫn của tập lệnh hiện tại
$_SERVER['SCRIPT_URI']	Trả về URI của trang hiện tại

$_REQUEST trong PHP

$_REQUEST trong PHP được dùng để thu thập dữ liệu sau khi gửi trên form HTML.

Ví dụ dưới đây đưa ra một form một trường đầu vào và một nút gửi. Khi mội người bấm vào "Gửi", các dữ liệu form sẽ được gửi đến các tập tin chỉ định trong thuộc tính **action** của thẻ <form>. Trong ví dụ này, chúng ta cho dữ liệu tự đặt hình thức xử lý. Nếu bạn muốn sử dụng một file PHP để xử lý dữ liệu form, thay

thế với tên file mà bạn chọn. Sau đó, chúng ta có thể sử dụng biến siêu toàn cục $_REQUEST để thu thập giá trị của các trường đầu vào.

```
<!DOCTYPE html>
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

Kết quả:

Nếu ta input giá trị vào trường text trên rồi ấn Gửi, trang sẽ hiển thị giá trị mà ta vừa nhập, nếu bỏ trống thì kết quả là: *Name is empty*

$_POST trong PHP

Biến siêu toàn cục $_POST được sử dụng rộng rãi để thu thập dữ liệu form sau khi gửi form HTML với method = "post". $_POST cũng được sử dụng rộng rãi để bỏ qua các biến.

Ví dụ dưới đây cho một form với một trường đầu vào và một nút gửi. Khi một người bấm "Gửi", các dữ liệu form được gửi đến tập tin chỉ định trong thuộc tính action của thẻ <form>. Trong ví dụ này, chúng ta chỉ đến tập tin riêng để xử lý các dữ liệu form. Nếu bạn muốn sử dụng một file PHP để xử lý dữ liệu form, thay thế với tên tập tin bạn đã chọn. Sau đó, chúng ta có thể sử dụng các biến siêu toàn cục $_POST để thu thập các giá trị form đầu vào:

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

Kết quả:

Nếu ta input giá trị vào trường text trên rồi ấn Gửi, trang sẽ hiển thị giá trị mà ta vừa nhập, nếu bỏ trống thì kết quả là: *Name is empty*

$_GET trong PHP

Biến siêu toàn cục $_GET cũng có thể được sử dụng để thu thập dữ liệu form sau khi gửi form HTML với method = "get".

$_GET cũng có thể thu thập dữ liệu được gửi **trong URL**

Giả sử chúng ta có một trang HTML có chứa một siêu liên kết với các thông số:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

Khi người dùng nhấp vào liên kết "Test $GET", các thông số **subject** và **web** được gửi đến "test_get.php", và sau đó bạn có thể truy cập các giá trị của chúng trong "test_get.php" với biến siêu toàn cục $_GET.

Ví dụ dưới đây cho thấy đoạn mã trong "test_get.php" sau khi nhấp vào link trên

```
<html>
<body>

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>

</body>
</html>
```

Kết quả:

Ở file html nguồn: Test $GET

Ở file test_get.php đích sau khi nhấp vào link trên: *Study PHP at W3schools.com*

Tip: Bạn sẽ được học nhiều hơn về $_POST và $_GET trong phần PHP Forms:

www.w3schools.com/php/php_forms.asp

PHẦN B: FORM TRONG PHP

PHP Form Handling

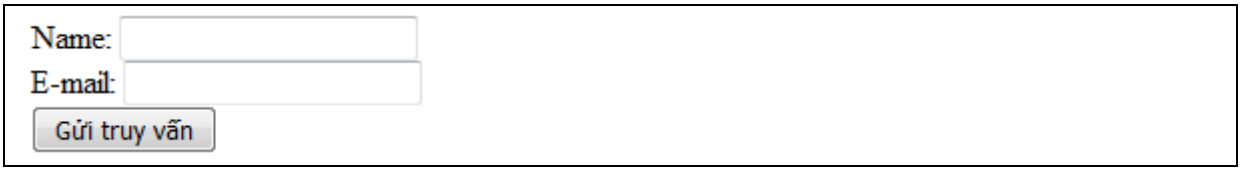
Các biến siêu toàn cục $_GET và $_POST được sử dụng để thu thập dữ liệu form.

Form HTML đơn giản

Ví dụ dưới đây hiển thị một form HTML đơn giản với hai trường nhập dữ liệu và nút gửi.

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Kết quả:



Khi người dùng điền vào form ở trên và nhấp chuột vào nút gửi, các dữ liệu form được gửi để xử lý trong file PHP tên "welcome.php". Dữ liệu form được gửi với phương thức HTTP POST.

Để hiển thị các dữ liệu đã gửi, bạn có thể echo tất cả các biến. Nội dung file "welcome.php" như sau:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

Kết quả sẽ như sau:

```
Welcome John
Your email address is john.doe@example.com
```

Kết quả giống như trên cũng có thể đạt được nếu sử dụng phương thức HTTP GET

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

và nội dung file “welcome_get.php” như sau:

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

Đoạn mã trên khá đơn giản. Tuy nhiên, điều quan trọng nhất lại thiếu. Bạn cần phải **xác nhận** dữ liệu form để bảo vệ tập lệnh khỏi các đoạn mã độc hại.

Luôn nghĩ về an ninh khi xử lý form PHP Trang này không chứa bất kỳ hình thức xác nhận, nó chỉ cho thấy cách để bạn có thể gửi và nhận biểu mẫu. Tuy nhiên, các trangtiếp theo sẽ trình bày cách để xử lý form PHP bảo mật hơn. Xác nhận dữ liệu form phù hợp là rất quan trọng để bảo vệ form của bạn trước các hacker và spammer!

GET với POST

Cả GET và POST đều tạo một mảng (ví dụ mảng (key => value, key2 => value2, key3 => value3, ...)). Mảng này chứa cặp khóa/giá trị, nơi các chỉ mục (khóa) có tên giống với tên control trong form và giá trị là giá trị người dùng nhập vào form.

Cả GET và POST đều được coi là $_GET và $_POST. Đây là những biến siêu toàn cục, có nghĩa là chúng luôn truy xuất được, bất kể phạm vi nào – và bạn có thể truy cập chúng từ bất kỳ hàm, class hoặc file nào mà không cần phải làm bất cứ điều gì đặc biệt.

$_GET là một mảng của các biến được truyền tới tập lệnh hiện tại thông qua các thông số thuộc URL.

$_POST là một mảng của các biến được truyền tới tập lệnh hiện tại thông qua phương thức HTTP POST.

Khi nào sử dụng GET?

Thông tin được gửi từ một form tới phương thức GET có thể được tất cả mọi người nhìn thấy (tất cả các tên và các giá trị biến được hiển thị trong URL). GET cũng có giới hạn về số lượng thông tin để gửi. Giới hạn là khoảng 2000 ký tự. Tuy nhiên, vì các biến được hiển thị trong URL nên việc đánh dấu trang là khả dụng. Điều này có thể hữu ích trong một số trường hợp.

GET có thể được sử dụng cho việc gửi dữ liệu **không nhạy cảm.**

Chú ý: GET **không** bao giờ nên sử dụng cho việc gửi mật khẩu hoặc các thông tin nhạy cảm khác!

Khi nào sử dụng POST?

Thông tin được gửi từ phương thức POST là vô hình với những người khác (tất cả tên/giá trị được nhúng vào trong body của các yêu cầu HTTP) và không có giới hạn về số lượng thông tin để gửi.

Hơn nữa POST hỗ trợ các chức năng tiên tiến như hỗ trợ đa đầu vào nhị phân khi tải file lên máy chủ.

Tuy nhiên, bởi vì các biến không được hiển thị trong URL, nên nó không dành cho việc đánh dấu trang.

Các developer ưu tiên dùng POST hơn để gửi dữ liệu form

Tiếp theo, hãy xem cách chúng ta có thể xử lý form PHP an toàn!

PHP Form validation

Chương này và chương kế trình bày các sử dụng PHP để xác nhận dữ liệu form

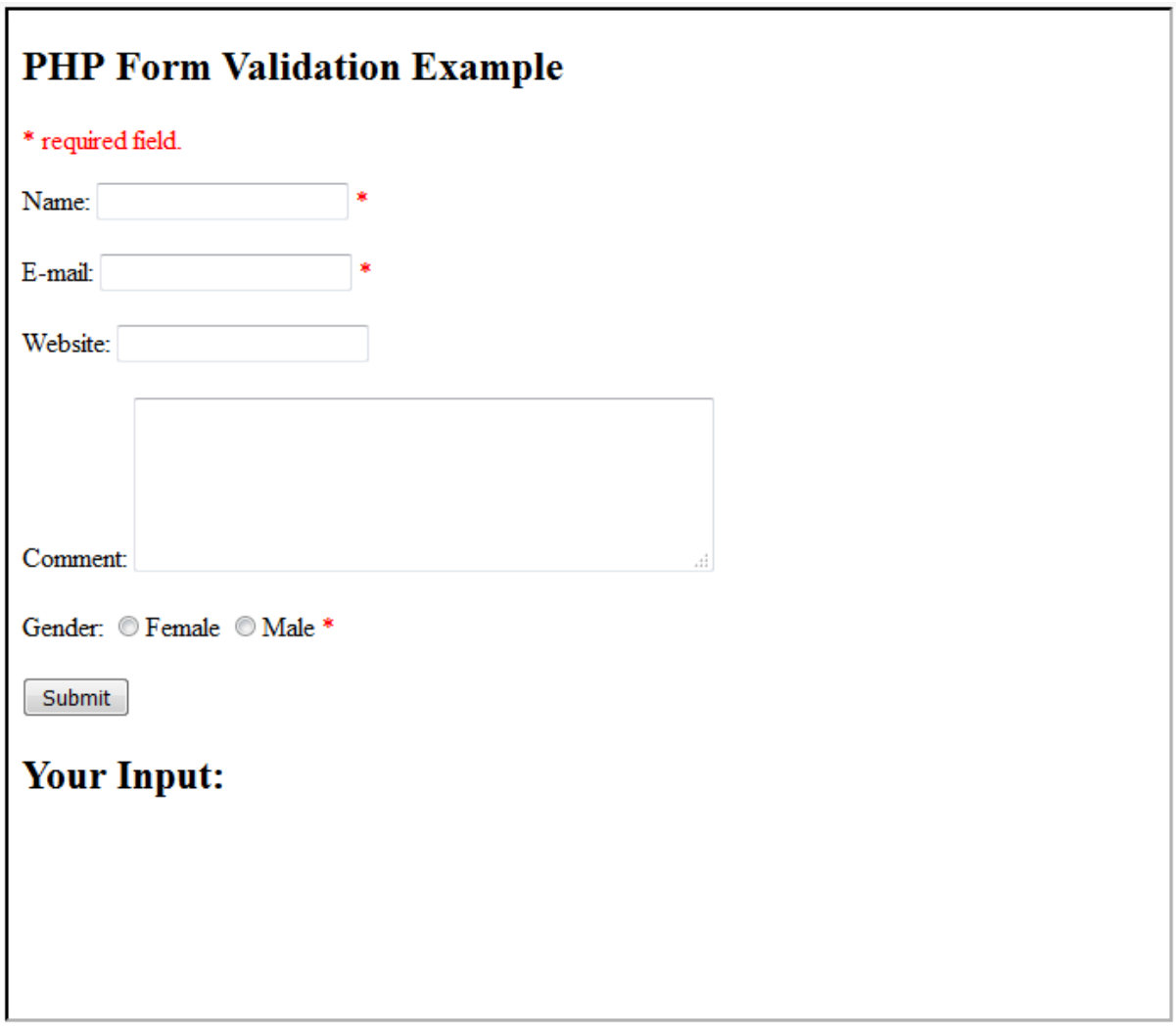
Xác nhận form trong PHP

Luôn nghĩ về bảo mật khi xử lý các form PHP!

Trang này sẽ trình bày cách để xử lý form PHP với an ninh tốt hơn. Xác nhận dữ liệu form phù hợp là rất quan trọng để bảo vệ form của bạn trước các hacker và spammer!

Form HTML chúng ta làm việc trong những chương này chứa các trường đa dạng: trường text tùy chọn và bắt buộc, radio button, và nút submit.

Dưới đây là hình ảnh về form:



Quy tắc xác nhận cho form trên được đưa ra như sau:

Trường	Quy tắc xác nhận
Name	Bắt buộc. + chỉ được chứa chữ cái và dấu cách
E-mail	Bắt buộc. + phải chứa một địa chỉ email hợp lệ (với @ và .)
Website	Tùy chọn. Nếu có, phải chứa URL hợp lệ
Comment	Tùy chọn. Là trường nhập vào nhiều dòng (textarea)
Gender	Bắt buộc. Phải chọn một trong hai

Đầu tiên chúng ta xem xét đoạn mã HTML của form:

Trường text

Tên, email, và website là các thành phần nhập vào dạng văn bản, và trường comment là textarea. Đoạn mã HTML như sau:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio buttons

Trường giới tính là radio button và đoạn mã HTML như sau:

```
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
```

Các phần tử form

Đoạn mã HTML của form như sau:

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

Khi form được submit, dữ liệu form sẽ gửi với phương thức "post"

Biến **$_SERVER["PHP_SELF"]** là gì?

Đây là một biến siêu toàn cục trả về tên của tập lệnh đang thực hiện.

Vì thế, $_SERVER["PHP_SELF"] gửi các dữ liệu form cho trang đang thực hiện, thay vì nhảy đến một trang khác. Bằng cách này, người dùng sẽ nhận được thông báo lỗi trên cùng một trang giống trang chứa form.

Hàm **htmlspecialchars**() có chức năng gì?

Nó chuyển đổi ký tự đặc biệt cho các đối tượng HTML. Điều này có ý nghĩa rằng nó sẽ thay thế ký tự HTML như **<and>** với **&lt**; và **&gt**;. Điều này ngăn chặn các tấn công khai thác mã bằng phương pháp tiêm mã HTML (**HTML injecting**) hoặc JavaScript (**Cross-site Scripting)** cho form.

Chú ý quan trọng trong bảo mật form PHP

Biến $_SERVER["PHP_SELF"] **có thể** bị tấn công bởi các hacker!

Nếu PHP_SELF được sử dụng trong trang của bạn sau đó người dùng có thể nhập dấu gạch chéo (/) và một số lệnh Cross Site Scripting (XSS) để thực thi.

Cross-site scripting (XSS) là một loại lỗ hổng bảo mật máy tính thường thấy trong các ứng dụng web. XSS cho phép kẻ tấn công tiêm đoạn mã phía client vào các trang web xem bởi những người dùng khác.

Giả sử chúng ta có một form sau trong một trang có tên "test_form.php"

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Bây giờ, nếu người dùng nhập URL bình thường trên thanh địa chỉ như:

http://www.example.com/test_form.php, đoạn code trên sẽ được chuyển thành:

```
<form method="post" action="test_form.php">
```

Không sao cả, vẫn ổn.

Tuy nhiên, để ý nếu người dùng nhập vào thanh địa chỉ URL như sau:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked
')%3C/script%3E
```

Trong trường hợp này, đoạn code trên sẽ trở thành:

```
<form method="post"
action="test_form.php/"><script>alert('hacked')</script>
```

Đoạn mã này gán thêm một thẻ script và một lệnh cảnh báo. Và khi tải trang, các mã JavaScript sẽ được thực hiện (người dùng sẽ thấy một hộp thoại cảnh báo). Đây chỉ là một ví dụ đơn giản và vô hại để thấy cách khai thác biến PHP_SELF.

Nên biết rằng bất kỳ mã JavaScript nào cũng có thể thêm vào trong thẻ <script>! Một hacker có thể chuyển hướng người dùng đến một tập tin trên máy chủ khác, và tập tin có thể chứa mã độc có thể làm thay đổi các biến toàn cục hoặc gửi form đến địa chỉ khác để lưu dữ liệu người dùng, chẳng hạn như vậy.

Cách để chống khai thác $_SERVER["PHP_SELF"]?

Việc khai thác $_SERVER["PHP_SELF"] có thể được chống bởi hàm **htmlspecialchars**()

Đoạn mã cho form như sau:

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

Hàm **htmlspecialchars**() chuyển đổi các ký tự đặc biệt thành phần tử HTML. Bây giờ nếu người dùng thử khai thác biến PHP_SELF, kết quả thu được sẽ như sau:

```
<form method="post"
action="test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/sc
ript&gt;">
```

Việc cố gắng khai thác thất bại, và không có sự gây hại nào được tạo ra.

Xác nhận dữ liệu form với PHP

Việc đầu tiên chúng ta sẽ làm là đưa tất cả các biến qua hàm **htmlspecialchars**() trong PHP.

Khi chúng ta sử dụng hàm **htmlspecialchars**(), và sau đó người dùng cố gắng submit đoạn mã trong trường text như sau:

```
<script>location.href('http://www.hacked.com')</script>
```

Nó sẽ không được thực hiện, bởi vì nó đã được lưu dưới dạng mã HTML escape, như:

```
&lt;script&gt;location.href('http://www.hacked.com')&lt;/script&gt;
```

Đoạn mã được an toàn để hiển thị trên trình duyệt hoặc trong e-mail.

Chúng ta cũng có thể dùng hai cách nữa khi người dùng submit form:

1. Loại bỏ tất cả các ký tự không cần thiết (khoảng trắng thừa, tab, dòng mới) do người dùng nhập vào (với hàm **trim**() trong PHP)
2. Xóa đi dấu gạch chéo (\) do người dùng nhập vào (với hàm **stripslashes**() trong PHP)

Bước tiếp theo là tạo ra function có thể thực hiện tất cả việc kiểm tra cho chúng ta (nó sẽ tiện lợi hơn việc đánh những dòng code giống nhau lặp đi lặp lại)

Chúng ta sẽ đặt tên hàm là **test_input**().

Bây giờ, chúng ta có thể kiểm tra mỗi biến $_POST với hàm test_input(), và đoạn mã có dạng như sau:

```php
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
```

```
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Chú ý rằng khi bắt đầu đoạn mã, chúng ta kiểm tra form đã được submit chưa bằng biến $_SERVER["REQUEST_METHOD"]. Nếu REQUEST_METHOD là POST thì form được submit – và nó cần xác nhận. Nếu như chưa được submit, bỏ qua việc xác nhận và hiển thị form trống.

Tuy nhiên, trong ví dụ trên, mọi trường đầu vào đều tùy chọn. Đoạn mã vẫn chạy tốt cả khi người dùng không nhập bất cứ dữ liệu nào.

Bước tiếp theo là tạo ràng buộc cho các trường đầu vào và hiển thị thông báo lỗi nếu cần.

PHP Form – Required

Chương này trình bày cách tạo ràng buộc cho các trường dữ liệu đầu vào tạo hiển thị thông báo lỗi nếu cần.

Các trường bắt buộc trong PHP

Từ bảng quy tắc xác nhận trong trang trước, chúng ta nhận thấy "Name", "E-mail" và "Gender" là trường bắt buộc. Những trường này không thể để trống và phải được điền vào trong form HTML.

Trường	Quy tắc xác nhận
Name	Bắt buộc. + chỉ được chứa chữ cái và dấu cách
E-mail	Bắt buộc. + phải chứa một địa chỉ email hợp lệ (với @ và .)
Website	Tùy chọn. Nếu có, phải chứa URL hợp lệ
Comment	Tùy chọn. Là trường nhập vào nhiều dòng (textarea)
Gender	Bắt buộc. Phải chọn một trong hai

Trong chương trước, mọi trường đầu vào đều tùy chọn.

Trong đoạn mã sau đây chúng ta thêm một số biến: $nameErr, $emailErr, $genderErr, và $websiteErr. Những biến lỗi này sẽ giữ thông báo lỗi cho trường bắt buộc. Chúng ta cũng thêm câu lệnh if else cho mỗi biến $_POST. Nó kiểm tra nếu như biến $_POST trống (với hàm empty() trong PHP). Nếu nó trống, một thông báo sẽ được lưu vào biến lỗi khác, và nếu nó không trống, nó sẽ gửi tới dữ liệu người dùng nhập vào thông qua hàm test_input():

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
```

```
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

Hiển thị thông báo lỗi trong PHP

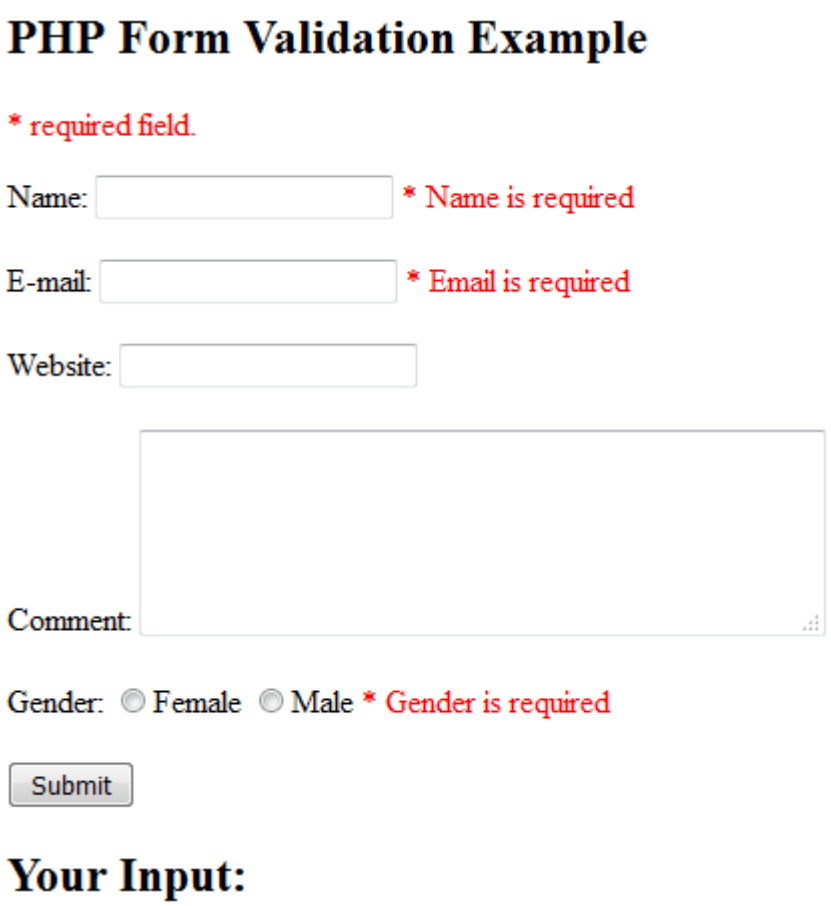
Sau đó, trong form HTML, chúng ta thêm một vài đoạn mã vào sau mỗi trường bawtst buộc, nó sẽ tạo ra các thông báo lỗi nếu cần (khi người dùng thử submit form mà không điền vào các trường bắt buộc):

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">

Name: <input type="text" name="name">
<span class="error">* <?php echo $nameErr;?></span>
<br><br>
E-mail:
<input type="text" name="email">
<span class="error">* <?php echo $emailErr;?></span>
```

```
<br><br>
Website:
<input type="text" name="website">
<span class="error"><?php echo $websiteErr;?></span>
<br><br>
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
<br><br>
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<span class="error">* <?php echo $genderErr;?></span>
<br><br>
<input type="submit" name="submit" value="Submit">

</form>
```

Kết quả:

Khi không điền vào trường bắt buộc mà ấn submit:



Toàn bộ code:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
```

```
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
```

```
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

Bước tiếp theo là xác nhận các dữ liệu đầu vào, đó là “Liệu các trường Name chứa chữ cái và khoảng trắng?” và “Liệu trường E-mail có chứa một cú pháp địa chỉ e-mail hợp lệ?”, và nếu điền thì “Liệu các trường trang web có chứa URL hợp lệ?”.

PHP Form URL/E-mail

Chương này trình bày cách xác nhận các tên, e-mail và URL.

Xác nhận tên trong PHP

Đoạn code dưới đây trình bày một cách kiểm tra đơn giản nếu trương tên chỉ chứa chữ cái và khoảng trắng. Nếu giá trị trong trường tên không hợp lệ, sẽ gán thông báo lỗi cho biến.

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

Hàm **preg_match**() tìm kiếm trong chuỗi một cấu hình, trả về **true** nếu cấu hình tồn tại, ngược lại trả về **false** .

Xác nhận E-mail trong PHP

Đoạn code dưới đây trình bày cách kiểm tra nếu địa chỉ URL hợp lệ về cú pháp (biểu thức chính quy này cũng cho phép dấu gạch ngang trong URL). Nếu địa chỉ URL có cú pháp không hợp lệ, sẽ lưu thông báo lỗi vào biến:

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";
}
```

Xác nhận tên, e-mail và URL trong PHP

Bây giờ, tập lệnh có thể như sau:

```
<?php
// khai báo các biến và đặt giá trị mặc định là trống
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";
```

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // kiểm tra nếu tên chỉ chứa chữ cái và khoảng trắng
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // kiểm tra nếu địa chỉ e-mail đúng định dạng
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // kiểm tra nếu cú pháp địa chỉ URL hợp lệ (biểu thức chính quy
này cũng cho phép dấu gạch nganng trong URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

Toàn bộ code bao gồm form HTML + xác nhận form:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
```

```
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
```

```
  <input type="radio" name="gender" value="male">Male
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

Chương tiếp theo trình bày cách để bảo vệ form khỏi bị làm trống tất cả các trường khi user submit form.

PHP Form complete

Chương này trình bày cách để **giữ lại** những giá trị trong các trường đầu vào khi user ấn nút submit.

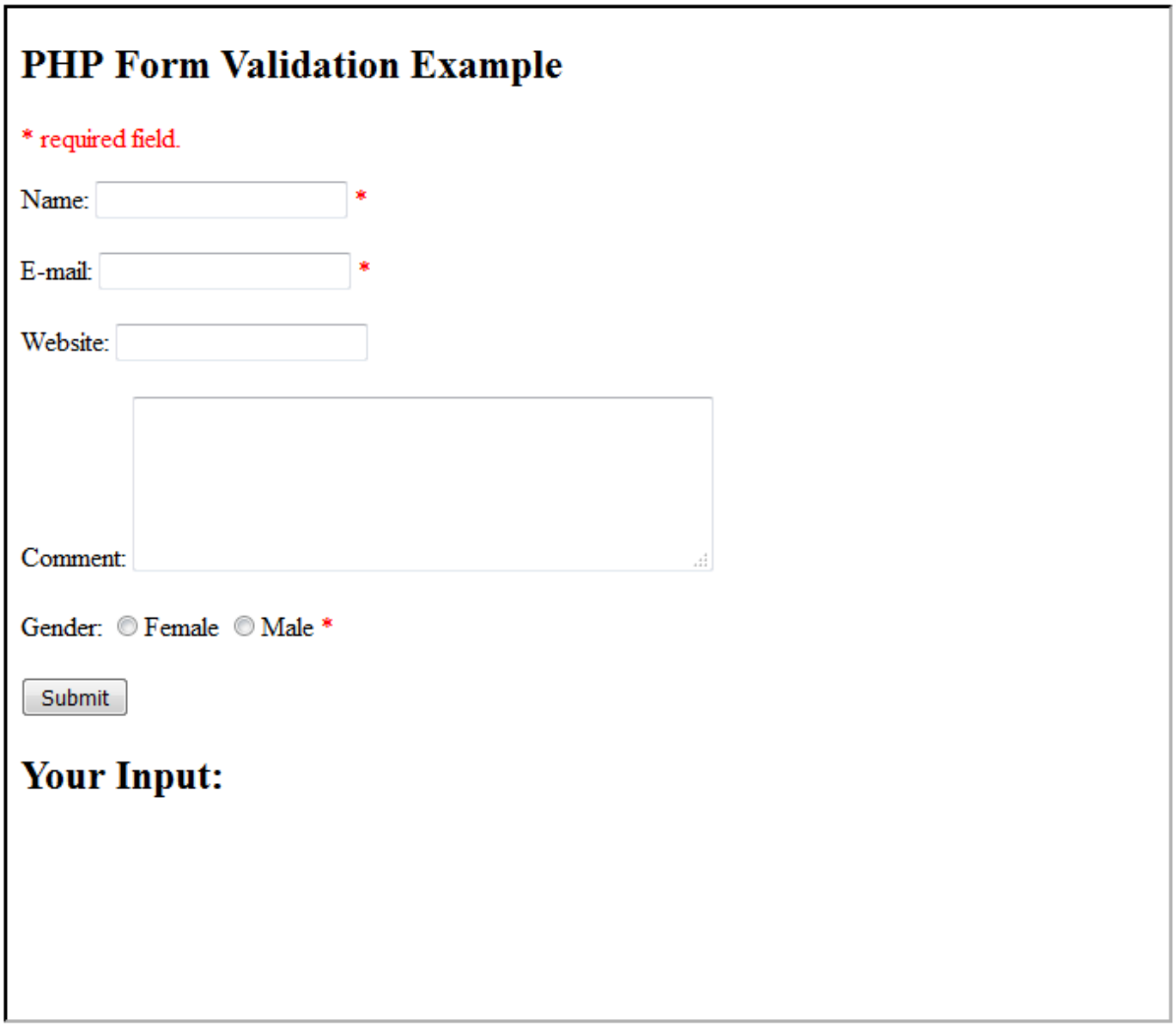
Giữ lại giá trị trong form PHP

Để hiển thị những giá trị trong các trường đầu vào sau khi người dùng ấn nút submit, chúng ta thêm một số đoạn mã PHP bên trong thuộc tính **value** của các trường: name, email và website. Trong trường comment dạng **textarea** (khác hơn so với các thẻ input trên), ta đặt đoạn mã PHP **giữa** cặp thẻ <textarea> </textarea>. Các tập lệnh nhỏ này sẽ xuất ra giá trị của biến $name, $email, $website và $comment.

Sau đó, chúng ta cũng cần phải hiển thị radio button đã được chọn. Đối với trường hợp này, chúng ta phải thao tác với thuộc tính **checked** (không phải là thuộc tính **value** cho các radio button):

```
Name: <input type="text" name="name" value="<?php echo $name;?>">

E-mail: <input type="text" name="email" value="<?php echo
$email;?>">

Website: <input type="text" name="website" value="<?php echo
$website;?>">

Comment: <textarea name="comment" rows="5" cols="40"><?php echo
$comment;?></textarea>

Gender:
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="female") echo "checked";?>
value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="male") echo "checked";?>
value="male">Male
```

Mẫu form hoàn chỉnh trong PHP



Code tương ứng:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression
also allows dashes in the URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
```

```
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name" value="<?php echo $name;?>">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email" value="<?php echo
$email;?>">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website" value="<?php echo
$website;?>">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"><?php echo
$comment;?></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="female") echo "checked";?> value="female">Female
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="male") echo "checked";?> value="male">Male
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
```

```
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

PHẦN C: CÁC CHỦ ĐỀ NÂNG CAO TRONG PHP

PHP Arrays Multi

Trước đó, trong giáo trình này, chúng ta đã đề cập đến mảng là một danh sách duy nhất của cặp khóa/giá trị.

Tuy nhiên, đôi khi bạn muốn lưu trữ các giá trị với nhiều hơn một khóa.

Điều này có thể thực hiện trong mảng đa chiều.

Mảng đa chiều trong PHP

Mảng đa chiều là mảng chứa một hoặc nhiều mảng khác.

PHP hiểu mảng đa chiều là mảng gồm hai, ba, bốn, năm hoặc nhiều cấp độ sâu. Tuy nhiên mảng lớn hơn ba cấp độ sâu rất khó quản lý đối với hầu hết mọi người.

Kích thước của một mảng xác định số chỉ mục bạn cần để chọn ra một phần tử.

- Đối với một mảng hai chiều bạn cần hai chỉ số để chọn một phần tử
- Đối với một mảng ba chiều bạn cần ba chỉ số để chọn một phần tử

Mảng hai chiều trong PHP

Mảng hai chiều là một mảng của các mảng (mảng ba chiều là một mảng của các mảng của các mảng).

Đầu tiên, hãy nhìn vào bảng sau:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

Chúng ta có thể lưu dữ liệu từ bảng trên vào một mảng hai chiều như sau:

```
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
```

Bây giờ mảng hai chiều $cars chứa 4 mảng con, và nó có hai chỉ mục: hàng và cột.

Để truy cập vào từng phần tử của mảng $cars, chúng ta phải trỏ đến hai chỉ mục (hàng và cột):

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold:
".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold:
".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold:
".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold:
".$cars[3][2].".<br>";
?>
```

Kết quả:

```
Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.
```

Chúng ta cũng có thể đặp vòng lặp for bên trong vòng lặp for để lấy ra từng phần tử của mảng $cars (vẫn phải trỏ đến hai chỉ mục)

```php
<?php
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

Kết quả:

Row number 0

- Volvo
- 22
- 18

Row number 1

- BMW
- 15
- 13

Row number 2

- Saab
- 5
- 2

Row number 3

- Land Rover
- 17
- 15

PHP Date and Time

Hàm date() trong PHP được sử dụng để định dạng ngày và/hoặc thời gian.

Hàm Date() trong PHP

Hàm date() trong PHP định dạng mốc thời gian để chúng ta có thể đọc được.

Cấu trúc:

date(*định dạng*, *mốc thời gian*)

Tham số	Mô tả
Định dạng	Bắt buộc. Xác định kiểu định dạng mốc thời gian
Mốc thời gian	Tùy chọn. Xác định mốc thời gian. Mặc định là ngày hiện tại và thời gian

Một dấu thời gian (timestamp) là một chuỗi các ký tự, biểu thị ngày tháng và / hoặc thời gian mà tại đó một sự kiện nào đó xảy ra.

Lấy ra ngày đơn lẻ

Tham số bắt buộc: *định dạng* của hàm date() xác định cách biểu diễn ngày (hoặc giờ)

Đây là một số ký tự được sử dụng rộng rãi cho ngày:

- d: Biểu diễn ngày trong tháng (01 đến 31)
- m: biểu diễn tháng (01 đến 12)
- Y: biểu diễn năm (dạng 4 số)
- l (chữ L in thường): Biểu diễn ngày trong tuần

Một số ký tự khác, như "/", ".", hoặc "-" cũng có thể được chèn vào giữa các ký tự trên để thêm định dạng cho ngày.

Ví dụ dưới đây định dạng ngày hiện tại theo những cách khác nhau:

```
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>
```

Kết quả (phụ thuộc vào ngày giờ hiện tại):

```
Today is 2016/07/15
Today is 2016.07.15
Today is 2016-07-15
Today is Friday
```

Mẹo PHP – Năm copyright tự động

Sử dụng hàm date() để tự động cập nhật năm copyright trên website của bạn:

```
&copy; 2010-<?php echo date("Y");?>
```

Kết quả: © 2010-2016

Lấy ra thời gian đơn lẻ

Đây là một vài ký tự được sử dụng rộng rãi cho thời gian:

- h - 12-hour format of an hour with leading zeros (01 to 12)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds with leading zeros (00 to 59)
- a - Lowercase Ante meridiem and Post meridiem (am or pm)

Ví dụ dưới đây xuất ra thời gian hiện tại theo dạng quy định:

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

Kết quả: The time is 05:44:55am (thời gian lấy ở server, không phải ở client)

Lấy ra time zone của bạn

Nếu thời gian bạn lấy từ code không đúng với thời gian hiện tại, có thể là vì server của bạn đặt ở vùng miền khác hoặc cài đặt với time zone không đúng.

Vì vậy, nếu bạn cần thời gian chính xác theo một địa điểm cụ thể, bạn có thể đặt múi giờ để sử dụng.

Ví dụ dưới đây đặt múi giờ là "Asia/Ho_Chi_Minh" (GMT+7), sau đó xuất ra thời gian hiện tại theo định dạng quy định:

```
<?php
date_default_timezone_set("Asia/Ho_Chi_Minh");
echo "The time is " . date("h:i:sa");
?>
```

Tạo ngày với hàm mktime() trong PHP

Tham số tùy chọn: *mốc thời gian* trong hàm date() xác định mốc thời gian. Nếu bạn không xác định mốc thời gian, ngày hiện tại và thời gian sẽ được dùng (như đã trình bày ở ví dụ trên).

Hàm mktime() trả về mốc thời gian Unix. Mốc thời gian Unix chứa số giây giữa Unix Epoch (1/1/1970 00:00:00 GMT) và thời gian xác định.

Cú pháp:

```
mktime(giờ, phút, giây, tháng, ngày, năm)
```

Ví dụ dưới đây tạo ra ngày và giờ từ các đối số thuộc hàm **mktime**():

```
<?php
$d=mktime(11, 14, 54, 8, 12, 2014);
echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

Kết quả:

```
Created date is 2014-08-12 11:14:54am
```

Tạo ngày từ chuỗi với hàm strtotime() trong PHP

Hàm **strtotime**() trong PHP được dùng để chuyển đổi chữ đọc được sang thời gian Unix.

Cú pháp:

```
strtotime(time,now);
```

Ví dụ dưới đây tạo ra ngày và giờ từ hàm strtotime():

```
<?php
$d=strtotime("10:30pm April 15 2014");
echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

Kết quả:

```
Created date is 2014-04-15 10:30:00pm
```

PHP khá thông minh trong việc chuyển đổi một chuỗi sang một ngày, vì vậy bạn có thể đặt tham số như sau:

```
<?php
$d=strtotime("tomorrow");
echo date("Y-m-d h:i:sa", $d) . "<br>";

$d=strtotime("next Saturday");
echo date("Y-m-d h:i:sa", $d) . "<br>";

$d=strtotime("+3 Months");
echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

Kết quả:

```
2016-07-16 12:00:00am
2016-07-16 12:00:00am
2016-10-15 06:15:24am
```

Tuy nhiên, hàm strtotime() không thực sự hoàn hảo, bạn phải nhớ chuỗi ký tự để nhập vào tham số.

Các ví dụ khác về ngày

Ví dụ dưới đây xuất ra 6 ngày thứ bảy kế tiếp:

```
<?php
$startdate = strtotime("Saturday");
$enddate = strtotime("+6 weeks", $startdate);

while ($startdate < $enddate) {
  echo date("M d", $startdate) . "<br>";
  $startdate = strtotime("+1 week", $startdate);
}
?>
```

Kết quả:

```
Jul 16
Jul 23
Jul 30
Aug 06
Aug 13
Aug 20
```

Ví dụ dưới đây xuất ra số ngày tới ngày 4 tháng 7

```
<?php
$d1=strtotime("July 04");
$d2=ceil(($d1-time())/60/60/24);
echo "There are " . $d2 ." days until 4th of July.";
?>
```

Nếu hôm nay là 15/7 thì kết quả sẽ là

```
There are -11 days until 4th of July.
```

Tham khảo đầy đủ về ngày trong PHP

Bạn có thể tham khảo tại:

http://www.w3schools.com/php/php_ref_date.asp

PHP include

Câu lệnh include (hoặc require) lấy tất cả văn bản/code/đánh dấu tồn tại trong file đã xác định và copy vào file chứa câu lệnh include.

Việc đính kèm các tập tin rất hữu ích khi bạn muốn bao gồm các file PHP, HTML hoặc văn bản trùng lặp trên nhiều trang của một website.

Câu lệnh include và require trong PHP

Có thể chèn các nội dung của một file PHP vào một file PHP (trước khi server thực hiện nó) với câu lệnh include hoặc require.

Include và require là giống nhau, ngoại trừ khi xảy ra lỗi (không tìm thấy file được include/require):

- require sẽ tạo ra một lỗi nghiêm trọng (E_COMPILE_ERROR) và dừng lại tập lệnh
- include sẽ chỉ đưa ra một cảnh báo (E_WARNING) và tập lệnh sẽ tiếp tục.

Vì vậy, nếu bạn muốn thực hiện để luôn hiển thị cho người sử dụng đầu ra, ngay cả khi tập tin mất tích thì sử dụng câu lệnh **include**. Nếu không, trong trường hợp của FrameWork, CMS, hoặc một mã ứng dụng PHP phức tạp, luôn luôn sử dụng các câu lệnh **require** để đính kèm một tập tin quan trọng cho các luồng thực hiện. Điều này sẽ giúp tránh ảnh hưởng an ninh và toàn vẹn cho ứng dụng của bạn, chỉ trong trường hợp một tập tin quan trọng vô tình bị mất tích.

Đính kèm các tập tin giúp tiết kiệm rất nhiều công việc. Điều này có nghĩa rằng bạn có thể tạo một header, footer, hoặc menu tiêu chuẩn cho tất cả các trang web của bạn. Sau đó, khi cần cập nhật, bạn có thể chỉ cần cập nhật các tập tin được đính kèm.

Cú pháp:

```
include 'tên file';

hoặc

require 'tên file';
```

Ngoài ra còn có **include_once** và **require_once** cho phép đính kèm chỉ một lần.

Ví dụ include trong PHP

Ví dụ 1:

Giả sử chúng ta có một footer chuẩn mang tên "footer.php" như sau:

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";
?>
```

Để include file footer này vào trang, sử dụng câu lệnh include:

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

Kết quả:

Ví dụ 2:

Giả sử chúng ta có một menu chuẩn mang tên "menu.php":

```
<?php
echo '<a href="/default.asp">Home</a> -
<a href="/html/default.asp">HTML Tutorial</a> -
```

```
<a href="/css/default.asp">CSS Tutorial</a> -
?>
```

Tất cả các trang trong website đều cần sử dụng file menu. Đây là cách để hoàn thành yêu cầu này (chúng ta sử dụng thẻ và menu có thể được định kiểu trong CSS):

```
<html>
<body>

<div class="menu">
<?php include 'menu.php';?>
</div>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>

</body>
</html>
```

Kết quả:

Home - HTML Tutorial - CSS Tutorial - JavaScript Tutorial - PHP Tutorial

Welcome to my home page!

Some text.

Some more text.

Ví dụ 3:

Giả sử chúng ta có một file tên là "vars.php", với một vài biến được khai báo:

```
<?php
$color='red';
$car='BMW';
?>
```

Sau đó, nếu chúng ta include file "vars.php", các biến có thể được sử dụng trong file đã gọi.

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php include 'vars.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

Include với Require

Các câu lệnh require cũng được sử dụng để bao gồm một tập tin vào các mã PHP. Tuy nhiên, có một sự khác biệt lớn giữa include và require; khi một tập tin được bao gồm trong câu lệnh **include** và PHP không thể tìm thấy nó, tập lệnh sẽ **tiếp tục** thực hiện:

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php include 'noFileExists.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

Kết quả:

Welcome to my home page!

I have a .

Nếu chúng ta thực hiện ví dụ với câu lệnh **require**, lệnh echo sẽ không thể thực hiện bởi vì đoạn mã thực thi bị kết thúc sau khi câu lệnh require thất bại vì không tìm thấy file:

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php require 'noFileExists.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

Kết quả:

Welcome to my home page!

Warning: require(noFileExists.php): failed to open stream: No such file or directory

Fatal error: require(): Failed opening required 'noFileExists.php'

- Sử dụng **require** khi tập tin nhận được yêu cầu của các ứng dụng.
- Sử dụng **include** khi tập tin có thể không cần thiết và ứng dụng sẽ tiếp tục khi tập tin là không tìm thấy.

PHP File Handing

Xử lý tập tin là một phần quan trọng của bất kỳ ứng dụng web nào. Bạn thường phải mở và xử lý tập tin cho các công việc khác nhau.

Thao tác tập tin trong PHP

PHP có các hàm đa dạng để tạo, xem, upload và sửa file.

Hãy cẩn thận khi thao tác với tập tin! Khi thao tác tập tin bạn phải rất cẩn thận. Bạn có thể gây nhiều thiệt hại nếu bạn làm điều gì đó sai. Lỗi thường gặp là: chỉnh sửa các tập tin sai, ghi vào một ổ đĩa cứng với các dữ liệu rác, và xóa các nội dung của một tập tin vô tình.

Hàm readfile() trong PHP

Hàm readfile() dùng để đọc một tập tin và ghi nó vào bộ đệm đầu ra.

Giả sử chúng ta có một tệp văn bản tên "webdictionary.txt", được lưu trên máy chủ như sau:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

Đoạn mã PHP để đọc và ghi file vào bộ đệm đầu ra như sau: (hàm readfile() trả về số lượng byte đọc thành công):

```
<!DOCTYPE html>
<html>
<body>

<?php
echo readfile("webdictionary.txt");
?>
```

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

Kết quả:

```
AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible Markup Language236
```

Hàm **readfile**() hữu ích nếu tất cả mọi người muốn mở tập tin và đọc nội dung của nó.

Chương tiếp theo sẽ hướng dẫn bạn nhiều hơn về việc xử lý file.

PHP File Open/Read

Trong chương này chúng ta sẽ tìm hiểu cách mở, đọc và đóng tập tin trên server.

Mở file với hàm fopen()

Phương pháp tốt nhất để mở tập tin là dùng hàm **fopen**(). Hàm này đưa ra nhiều tùy chọn hơn so với hàm **readfile**().

Chúng ta sẽ sử dụng tập tin văn bản "webdictionary.txt" trong suốt bài học:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

Tham số đầu tiên của hàm **fopen**() chưa tên của tệp tin được mở và tham số thứ hai xác định chế độ mở. Ví dụ sau đây cũng tạo ra thông báo nếu hàm **fopen**() không thể mở file đã chọn:

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open
file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

Kết quả:

```
AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML =
Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL =
Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible
Markup Language
```

Tip: Hàm **fread**() và **fclose**() sẽ được trình bày dưới đây.

Tập tin có thể được mở ở một trong các chế độ sau:

Chế độ	Mô tả
r	**Mở file ở chế độ chỉ đọc**. Con trỏ tệp tin bắt đầu ở điểm đầu của tệp tin
w	**Mở tập tin ở chế độ chỉ ghi**. Xóa nội dung của tập tin hoặc tạo ra một tập tin mới nếu nó không tồn tại. Con trỏ tập tin bắt đầu vào đầu của tập tin
a	**Mở tập tin ở chế độ chỉ ghi**. Các dữ liệu hiện có trong tập tin được bảo tồn. con trỏ tập tin bắt đầu vào cuối của tập tin. Tạo một tập tin mới nếu các tập tin không tồn tại
x	**Tạo một tập in mới chỉ để ghi.** Trả về **FALSE** và một lỗi nếu tập tin đã tồn tại
r+	**Mở một tập tin để đọc / ghi**. Con trỏ tập tin bắt đầu ở đầu của tập tin
w+	**Mở một tập tin để đọc / ghi**. Xóa nội dung của tập tin hoặc tạo ra một tập tin mới nếu nó không tồn tại. Con trỏ tập tin bắt đầu vào đầu của tập tin
a+	**Mở một tập tin để đọc / ghi**. Các dữ liệu hiện có trong tập tin được bảo toàn. Con trỏ tập tin bắt đầu vào cuối của tập tin. Tạo một tập tin mới nếu các tập tin không tồn tại
x+	**Tạo một tập tin mới để đọc / ghi**. Trả về FALSE và một lỗi nếu tập tin đã tồn tại

Hàm đọc tệp - fread() trong PHP

Hàm fread() đọc dữ liệu từ một file đang mở.

Tham số đầu tiên trong hàm chứa tên của file để đọc và tham số thứ hai xác định số byte tối đa có thể đọc.

Đoạn mã PHP sau đây đọc file "webdictionary.txt" cho đến cuối:

```
fread($myfile,filesize("webdictionary.txt"));
```

Hàm đóng tệp - fclose() trong PHP

Hàm fclose() được sử dụng để đóng một tệp đang mở.

Đây là một thao tác lập trình tốt để đóng tất cả các tập tin sau khi bạn đã hoàn thành với chúng. Bạn không muốn một tập tin mở chạy vòng vo trên máy chủ của bạn và chiếm tài nguyên!

Hàm fclose() yêu cầu tên tập tin (hoặc biến lưu tên file) chúng ta muốn đóng lại:

```
<?php
$myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
fclose($myfile);
?>
```

Hàm đọc từng dòng – fgets() trong PHP

Hàm fgets() được sử dụng để đọc từng dòng từ một tệp tin.

Ví dụ dưới dây xuất ra dòng đầu tiên của tập tin “webdictionary.txt”:

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open
file!");
echo fgets($myfile);
fclose($myfile);
?>
```

Kết quả:

```
AJAX = Asynchronous JavaScript and XML
```

Chú ý: Sau khi gọi hàm fgets(), con trỏ sẽ được chuyển đến dòng tiếp theo.

Hàm kiểm tra kết thúc file – feof() trong PHP

Hàm feof() kiểm tra nếu đã đạt tới điểm cuối của file.

Hàm feof() hữu ích để duyệt qua dữ liệu nếu không biết độ dài.

Ví dụ dưới đây đọc file "webdictionary.txt" từng dòng, cho đến khi kết thúc:

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open
file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```

Kết quả:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

Hàm đọc ký tự đơn – fgetc() trong PHP

Hàm fgetc() được sử dụng để đọc một ký tự từ file.

Ví dụ dưới đây đọc file "webdictionary.txt" từng ký tự một, cho đến hết file:

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open
file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
?>
```

Kết quả:

```
AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML =
Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL =
Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible
Markup Language
```

Chú ý: Sau khi gọi hàm fgetc(), con trỏ di chuyển tới ký tự tiếp theo của văn bản.

PHP File Create/Write

Trong chương này chúng ta sẽ tìm hiểu cách tạo và ghi vào file trên server.

Hàm tạo file – fopen()

Hàm fopen() cũng được sử dụng để tạo một file. Có thể mâu thuẫn, nhưng trong PHP, một tập tin được tạo ra bằng cách sử dụng hàm tương tự như khi mở tập tin.

Nếu bạn sử dụng hàm fopen() với một tập tin không tồn tại, nó sẽ được tạo ra. Với điều kiện tập tin đó được mở để ghi (w) hoặc thêm nội dung (a).

Ví dụ dưới đây tạo ra một file tên "testfile.txt". File sẽ được tạo cùng với đường dẫn mà đoạn mã PHP nằm trong đó:

```
$myfile = fopen("testfile.txt", "w")
```

Cho phép truy cập file trong PHP

Nếu bạn gặp lỗi khi thử chạy đoạn mã trên, hãy kiểm tra rằng bạn đã cấp quyền truy cập cho file PHP để ghi thông tin lên đĩa cứng.

Hàm ghi file – fwrite() trong PHP

Hàm fwrite() được sử dụng để ghi vào một file.

Tham số đầu tiên trong hàm chứa tên của file cần ghi vào và tham số thứ hai là chuỗi ký tự sẽ được ghi.

Ví dụ dưới đây ghi hai tên vào file "newfile.txt"

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

Chú ý rằng chúng ta ghi vào file "newfile.txt" hai lần. Mỗi lần chúng ta ghi vào file chúng ta gửi chuỗi $txt đầu tiên chứa "John Doe" và tiếp theo chứa "Jane Doe". Sau khi hoàn thành ghi, chúng ta sử dụng hàm fclose() để đóng tập tin.

Nếu chúng ta mở tập tin "newfile.txt" sẽ thấy như sau:

```
John Doe
Jane Doe
```

Ghi đè trong PHP

Bây giờ file "newfile.txt" chứa một số dữ liệu chúng ta có thể hiển thị. Điều gì xảy ra khi chúng ta mở cùng tập tin đó với chế độ ghi? Mọi dữ liệu trước đó sẽ bị xóa và chúng ta lại bắt đầu với một tập tin rỗng.

Trong ví dụ dưới đây chúng ta mở file "newfile.txt" đã tồn tại, và ghi một vài dữ liệu vào nó:

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "Mickey Mouse\n";
fwrite($myfile, $txt);
$txt = "Minnie Mouse\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

Nếu chúng ta mở file "newfile.txt", cả chữ Jone và Jane đều mất, chỉ có các dữ liệu vừa nhập tồn tại

```
Mickey Mouse
Minnie Mouse
```

Tham khảo đầy đủ về file trong PHP

Để tham khảo đầy đủ về hệ thống các hàm file trong PHP, truy cập:

http://www.w3schools.com/php/php_ref_filesystem.asp

PHP File Upload

Với PHP, thật dễ dàng để upload tập tin lên server.

Tuy nhiên, cũng dễ dẫn đến nguy hại, vì vậy phải luôn cẩn thận khi cho phép upload file.

Cấu hình file "php.ini"

Đầu tiên, đảm bảo rằng PHP đã được cấu hình cho phép upload tập tin.

Trong file "php.ini", tìm chỉ thị **file_uploads** và đặt nó lên On:

```
file_uploads = On
```

Tạo ra form HTML

Tiếp theo, tạo ra một form HTML cho phép người dùng chọn file ảnh họ muốn upload:

```
<!DOCTYPE html>
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-
data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```

Một số quy tắc cho form trên:

- Đảm bảo rằng form sử dụng phương thức "post"
- Form cũng cần các thuộc tính sau đây: enctype="multipart/form-data". Nó xác định kiểu nội dung nào sẽ được sử dụng khi submit form.

Nếu không có những yêu cầu trên, các tập tin tải lên sẽ không hoạt động.

Một vài điểm cần chú ý:

- Thuộc tính type = “file” của thẻ <input> cho biết trường đầu vào như một control chọn file, với nút “Browse” bên cạnh control đầu vào.

Form trên gửi dữ liệu vào file có tên “upload.php”, tiếp theo chúng ta sẽ tạo nó.

Tạo đoạn mã PHP để upload file

File “upload.php” chứa đoạn mã cho việc upload file:

```php
<?php
$target_dir = "uploads/";
$target_file = $target_dir .
basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
?>
```

Giải thích các lệnh PHP:

- $target_dir = “uploads/” – xác định **thư mục** nơi tập tin sẽ được đặt trong đó
- $target_file xác định cụ thể **đường dẫn** của tập tin được tải lên
- $uploadOK=1 sẽ được dùng sau đó
- $imageFileType giữ phần mở rộng của file
- Tiếp theo, kiểm ra nếu tập tin ảnh là thật hoặc giả

Chú ý: Bạn cần phải tạo ra một thư mục tên “uploads” trong nơi chứa file “upload.php”. Các tập tin được tải lên sẽ lưu trong đó

Kiểm tra nếu file đã tồn tại trước đó

Bây giờ chúng ta có thể thêm một số hạn chế.

Đầu tiên, chúng ta sẽ kiểm tra nếu như file đã tồn tại trong thư mục "uploads". Nếu đúng, một thông báo lỗi sẽ hiển thị và $uploadOK đặt về 0:

```
// Kiểm tra tồn tại file
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
```

Giới hạn dung lượng file

Trường đầu vào trong form HTML trên có tên "fileToUpload".

Bây giờ, chúng ta muốn kiểm tra dung lượng của file. Nếu file lớn hơn 500kb, một thông báo lỗi sẽ hiển thị, và $uploadOK đặt về 0:

```
// Kiểm tra dung lượng file
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
```

Giới hạn kiểu file

Đoạn code dưới đây chỉ cho phép người dùng tải lên file JPG, JPEG, PNG và GIF. Mọi kiểu file khác đều đưa ra thông báo lỗi trước khi đặt $uploadOK về 0:

```
// Cho phép file đúng định dạng
if($imageFileType != "jpg" && $imageFileType != "png" &&
$imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
```

Hoàn thành đoạn mã PHP upload file

Bây giờ file "upload.php" hoàn chỉnh như sau:

```php
<?php
$target_dir = "uploads/";
$target_file = $target_dir .
basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Kiểm tra tập tin ảnh là ảnh thật hay giả
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
// Kiểm tra nếu file đã tồn tại
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
// Kiểm tra dung lượng file
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
// Cho phép định dạng file nhất định
if($imageFileType != "jpg" && $imageFileType != "png" &&
$imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
// Kiểm tra nếu biến $uploadOk đặt về 0 bởi lỗi
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
// Nếu không có lỗi, thử upload tập tin
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"],
$target_file)) {
        echo "The file ". basename(
$_FILES["fileToUpload"]["name"]). " has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```

Tham khảo đầy đủ về hệ thống file trong PHP

Để tham khảo đầy đủ về hệ thống các hàm file trong PHP, truy cập:

http://www.w3schools.com/php/php_ref_filesystem.asp

PHP Cookies

Cookie thường được dùng để định danh người dùng.

Cookie là gì?

Cookie thường được sử dụng để xác định người dùng. Cookie là một file nhỏ mà các máy chủ nhúng trên máy tính của người dùng. Mỗi lần máy tính yêu cầu mộ trang trên trình duyệt, nó cũng sẽ gửi cookie. Với PHP, bạn có thể tạo và lấy giá trị cookie.

Tạo cookies với PHP

Một cookie được tạo với hàm **setcookie**()

Cú pháp:

setcookie(*name, value, expire, path, domain, secure, httponly*);

Chỉ tham số *name* là bắt buộc, các tham số còn lại là tùy chọn.

Tạo/Lấy Cookie trong PHP

Ví dụ sau đây tạo ra một cookie với tên "user" cùng giá trị "John Doe". Cookie này sẽ hết hạn sau 30 ngày (86400*30). Dấu "/" nghĩa là cookie tồn tại trong toàn bộ trang web (nếu không, chọn thư mục mà bạn thích).

Khi bạn muốn lấy giá trị của cookie "user" (sử dụng biến toàn cục $_COOKIE). Chúng ta cũng sử dụng hàm **isset**() để kiểm tra cookie có giá trị hay không:

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
// 86400 giây = 1 ngày
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
```

```
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Kết quả:

```
Cookie named 'user' is set!

Value is: John Doe
```

Chú ý: Hàm **setcookie**() phải được đặt trước thẻ <html>

Chú ý: Các giá trị của cookie được tự động urlencoded khi gửi cookie và tự động giải mã khi nhận được (nếu không muốn mã hóa URL, sử dụng **setrawcookie**() thay thế).

Thay đổi giá trị Cookie

Để thay đổi cookie, phải đặt lại cookie bằng hàm **setcookie**():

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
```

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

Kết quả:

```
Cookie 'user' is set!

Value is: Alex Porter
```

Xóa cookie

Để xóa cookie, sử dụng hàm **setcookie**() với hạn là thời điểm trước đó:

```
<?php
// Đặt thời gian hạn là một giờ (3600 giây) trước đó
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Kiểm tra nếu Cookies đã được bật

Ví dụ sau đây tạo ra một tập lệnh nhỏ kiểm ra mỗi khi cookies được bật. Đầu tiên, thử tạo test cookie với hàm setcookie(), sau đó đếm biến thuộc mảng $_COOKIE:

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
```

```
?>

</body>
</html>
```

Tham khảo đầy đủ về PHP HTTP

Để tham khảo thêm về các hàm HTTP, truy cập:

http://www.w3schools.com/php/php_ref_http.asp

PHP Sessions

Session là một cách lưu thông tin (trong các biến) để được sử dụng trên nhiều trang.

Không như cookie, thông tin **không** được lưu trên máy tính của người dùng.

PHP Session là gì?

Khi bạn làm việc với một ứng dụng, bạn mở nó, thực hiện một số thay đổi, và sau đó bạn đóng nó. Điều này giống như một phiên làm việc. Máy tính sẽ biết bạn là ai. Nó biết khi bạn khởi động ứng dụng và khi bạn kết thúc. Nhưng trên mạng internet có một vấn đề: các máy chủ web không biết bạn là ai hay bạn làm gì, bởi vì địa chỉ HTTP không duy trì trạng thái.

Các biến session giải quyết vấn đề này bằng cách lưu trữ thông tin người dùng sẽ được sử dụng trên nhiều trang (ví dụ: username, favorite color,...). Mặc định, các biến session kéo dài cho đến khi người dùng đóng trình duyệt.

Vì thế, các biến session giữ thông tin về người dùng, và có sẵn cho tất cả các trang trong một ứng dụng.

Mẹo: Nếu cần lưu trữ vĩnh viễn, bạn có thể lưu trong cơ sở dữ liệu.

Khởi động PHP Session

Một session được khởi động bằng hàm **session_start**()

Các biến session được đặt giá trị thông qua biến toàn cục PHP: $_SESSION

Bây giờ, hãy tạo một trang mới tên "demo_session1.php". Trong trang này, chúng ta khởi động một PHP session mới và đặt một số biến session:

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

```php
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Kết quả:

```
Session variables are set.
```

Chú ý: Hàm **session_start**() phải được đặt đầu tiên trong các đoạn mã, **trước** các thẻ HTML.

Lấy các biến session trong PHP

Tiếp theo, chúng ta tạo một trang khác tên "demo_session2.php". Từ trang này, chúng ta sẽ truy cập thông tin session chúng ta đã đặt ở trang trước ("demo_session1.php")

Lưu ý rằng các biến session không được truyền riêng lẻ qua từng trang mới, chúng được lấy từ session chúng ta đã bắt đầu mỗi trang (session_start())

Cũng chú ý rằng tất cả giá trị các biến session đều được lưu trong biến toàn cục $_SESSION:

```php
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
```

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

Kết quả:

```
Favorite color is green.
Favorite animal is cat.
```

Một cách khác để hiển thị tất cả giá trị các biến session cho người dùng là chạy đoạn mã sau:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
?>

</body>
</html>
```

Kết quả:

```
Array ( [favcolor] => green [favanimal] => cat )
```

Session làm việc như thế nào? Cách nó nhận biết tôi?

Hầu hết các session đặt user-key trên máy tính người dùng như: 765487cf34ert8dede5a562e4f3a7e12. Sau đó, khi session được mở trong trang khác, nó quét user-key trên máy tính. Nếu trùng hợp, nó truy cập session, nếu không, nó bắt đầu một session mới.

Thay đổi biến session trong PHP

Để thay đổi một biến session, phải ghi đè nó:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Kết quả:

```
Array ( [favcolor] => yellow [favanimal] => cat )
```

Hủy một session trong PHP

Để gỡ bỏ tất cả các biến session và hủy session, sử dụng hàm **session_unset**() và **session_destroy**():

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

PHP Filters

Xác nhận dữ liệu – Xác nhận nếu dữ liệu ở dạng hợp lệ

Làm sạch dữ liệu – Loại bỏ ký tự không phù hợp khỏi dữ liệu

Mở rộng bộ lọc trong PHP

Các bộ lọc trong PHP được sử dụng để xác nhận và làm sạch dữ liệu vào bên ngoài.

Bộ lọc trong PHP có nhiều hàm cần thiết để kiểm tra người dùng nhập vào, được thiết kế để xác nhận dữ liệu đơn giản và nhanh hơn.

Hàm **filter_list**() có thể sử dụng để liệt kê các phần mở rộng bộ lọc trong PHP:

```
<table>
  <tr>
    <td>Filter Name</td>
    <td>Filter ID</td>
  </tr>
  <?php
  foreach (filter_list() as $id =>$filter) {
      echo '<tr><td>' . $filter . '</td><td>' . filter_id($filter) .
'</td></tr>';
  }
  ?>
</table>
```

Kết quả:

Filter Name	Filter ID
int	257
boolean	258
float	259
validate_regexp	272
validate_url	273
validate_email	274
validate_ip	275
string	513
stripped	513
encoded	514
special_chars	515

full_special_chars	522
unsafe_raw	516
email	517
url	518
number_int	519
number_float	520
magic_quotes	521
callback	1024

Tại sao phải sử dụng bộ lọc?

Nhiều ứng dụng web nhận vào các dữ liệu bên ngoài. Dữ liệu có thể là:

- Người dùng nhập vào form
- Cookies
- Dữ liệu dịch vụ web
- Các biến server
- Các kết quả truy vấn cơ sở dữ liệu

Bạn hãy luôn hợp thức hóa dữ liệu bên ngoài

Dữ liệu gửi đi không hợp lệ có thể dẫn đến các vấn đề bảo mật và làm hỏng trang web của bạn!

Bằng việc sử dụng lọc trong PHP bạn có thể chắc rằng ứng dụng của bạn lấy được đúng dữ liệu đầu vào

Hàm filter_var() trong PHP

Hàm filter_var() vừa xác nhận và làm sạch dữ liệu.

Hàm filter_var() lọc một biến duy nhất với bộ lọc quy định. Nó có hai mẫu về dữ liệu:

- Biến bạn muốn kiểm tra
- Kiểu kiểm tra bạn muốn sử dụng

Làm sạch chuỗi

Ví dụ dưới đây sử dụng hàm filter_var() để bỏ qua mọi thẻ HTML từ chuỗi:

```php
<?php
$str = "<h1>Hello World!</h1>";
$newstr = filter_var($str, FILTER_SANITIZE_STRING);
echo $newstr;
?>
```

Kết quả:

```
Hello World!
```

Xác nhận số nguyên

Ví dụ sau đây sử dụng hàm filter_var() để kiểm tra nếu biến $int là một số nguyên. Nếu $int là số nguyên, đoạn mã sẽ xuất ra: “Integer is valid”. Ngược lại sẽ xuất ra: “Integer is not valid”:

```php
<?php
$int = 100;

if (!filter_var($int, FILTER_VALIDATE_INT) === false) {
    echo("Integer is valid");
} else {
    echo("Integer is not valid");
}
?>
```

Kết quả:

```
Integer is valid
```

Tip: Vấn đề về hàm filter_var() với giá trị 0

Trong ví dụ trên, nếu $int được đặt về 0, hàm sẽ trả về “Integer is not valid”. Để giải quyết vấn đề này, sử dụng đoạn code bên dưới:

```php
<?php
$int = 0;

if (filter_var($int, FILTER_VALIDATE_INT) === 0 || !filter_var($int,
FILTER_VALIDATE_INT) === false) {
    echo("Integer is valid");
} else {
    echo("Integer is not valid");
}
?>
```

Kết quả:

```
Integer is valid
```

Xác nhận địa chỉ IP

Ví dụ sau đây sử dụng hàm filter_var() để kiểm tra nếu biến $ip là một địa chỉ IP hợp lệ:

```php
<?php
$ip = "127.0.0.1";

if (!filter_var($ip, FILTER_VALIDATE_IP) === false) {
    echo("$ip is a valid IP address");
} else {
    echo("$ip is not a valid IP address");
}
?>
```

Kết quả:

```
127.0.0.1 is a valid IP address
```

Làm sạch và xác nhận địa chỉ email

Ví dụ sau đây sử dụng hàm filter_var() trước hết để loại bỏ tất cả ký tự không hợp lệ từ biến $email, sau đó kiểm tra nếu nó là địa chỉ email hợp lệ:

```php
<?php
$email = "john.doe@example.com";

// Loại bỏ tất cả ký tự không hợp lệ từ email
$email = filter_var($email, FILTER_SANITIZE_EMAIL);
```

```
// Xác nhận e-mail
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
    echo("$email is a valid email address");
} else {
    echo("$email is not a valid email address");
}
?>
```

Kết quả:

```
john.doe@example.com is a valid email address
```

Làm sạch và xác nhận URL

Ví dụ sau đây sử dụng hàm filter_var() trước hết để loại bỏ các ký tự không hợp lệ từ URL, sau đó kiểm tra nếu $url là một URL hợp lệ:

```
<?php
$url = "http://www.w3schools.com";

// Loại bỏ các ký tự không hợp lệ từ URL
$url = filter_var($url, FILTER_SANITIZE_URL);

// Xác nhận URL
if (!filter_var($url, FILTER_VALIDATE_URL) === false) {
    echo("$url is a valid URL");
} else {
    echo("$url is not a valid URL");
}
?>
```

Kết quả:

```
http://www.w3schools.com is a valid URL
```

Tham khảo đầy đủ về các bộ lọc trong PHP

Để tham khảo đầy đủ về tất cả các hàm lọc trong PHP, truy cập:

http://www.w3schools.com/php/php_ref_filter.asp

PHP Filter Advanced

Xác nhận một số nguyên với miền giá trị

Ví dụ sau đây sử dụng hàm filter_var() để kiểm tra nếu một biến vừa kiểu INT và giá trị từ 1 đến 200:

```
<?php
$int = 122;
$min = 1;
$max = 200;

if (filter_var($int, FILTER_VALIDATE_INT, array("options" =>
array("min_range"=>$min, "max_range"=>$max))) === false) {
    echo("Variable value is not within the legal range");
} else {
    echo("Variable value is within the legal range");
}
?>
```

Kết quả:

```
Variable value is within the legal range
```

Kiểm tra địa chỉ IP v6

Ví dụ sau đây sử dụng hàm filter_var() để kiểm tra nếu biến $ip là một địa chỉ Ipv6 hợp lệ:

```
<?php
$ip = "2001:0db8:85a3:08d3:1319:8a2e:0370:7334";

if (!filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6) ===
false) {
    echo("$ip is a valid IPv6 address");
} else {
    echo("$ip is not a valid IPv6 address");
}
?>
```

Kết quả:

```
2001:0db8:85a3:08d3:1319:8a2e:0370:7334 is a valid IPv6 address
```

Xác nhận URL chứa chuỗi truy vấn

Ví dụ sau đây sử dụng hàm filter_var() để kiểm tra nếu biến $url là một URL với chuỗi truy vấn:

```
<?php
$url = "http://www.w3schools.com";

if (!filter_var($url, FILTER_VALIDATE_URL,
FILTER_FLAG_QUERY_REQUIRED) === false) {
    echo("$url is a valid URL");
} else {
    echo("$url is not a valid URL");
}
?>
```

Kết quả:

```
http://www.w3schools.com is not a valid URL
```

Loại bỏ các ký tự với mã ASCII > 127

Ví dụ sau đây sử dụng hàm filter_var() để làm sạch chuỗi. Nó sẽ vừa loại bỏ các thẻ HTML, và mọi ký tự với mã ASCII > 127 từ chuỗi:

```
<?php
$str = "<h1>Hello WorldÆØÅ!</h1>";

$newstr = filter_var($str, FILTER_SANITIZE_STRING,
FILTER_FLAG_STRIP_HIGH);
echo $newstr;
?>
```

Kết quả:

```
Hello World!
```

Tham khảo đầy đủ về lọc trong PHP

Để tham khảo đầy đủ về tất cả các hàm lọc trong PHP, truy cập:

http://www.w3schools.com/php/php_ref_filter.asp

PHP Error Handling

Việc xử lý lỗi mặc định trong PHP là rất đơn giản. Một thông báo lỗi với tên tập tin, số dòng và một thông điệp mô tả lỗi được gửi đến trình duyệt.

Xử lý lỗi trong PHP

Khi tạo ra các đoạn mã và ứng dụng web, xử lý lỗi là một phần quan trọng. Nếu code của bạn thiếu đoạn mã kiểm tra lỗi, chương trình trông sẽ thiếu chuyên nghiệp và bạn có thể gây ra các rủi ro về an ninh.

Bài hướng dẫn này chứa một số phương pháp kiểm tra lỗi phổ biến trong PHP.

Chúng ta sẽ tìm hiểu các phương pháp xử lý lỗi khác nhau:

- Câu lệnh "die()" đơn giản
- Tùy biến lỗi và trigger lỗi
- Báo cáo lỗi.

Xử lý lỗi cơ bản: Sử dụng hàm die()

Ví dụ đầu tiên trình bày đoạn code đơn giản để mở một file văn bản:

```
<?php
$file=fopen("welcome.txt","r");
?>
```

Nếu file đó không tồn tại, bạn có thể sẽ gặp lỗi như sau:

Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:
No such file or directory in **C:\webfolder\test.php** on line **2**

Để phòng tránh người dùng gặp lỗi trên, chúng ta kiểm tra mỗi khi file được mở trước khi truy cập nó:

```
<?php
if(!file_exists("welcome.txt")) {
  die("File not found");
} else {
  $file=fopen("welcome.txt","r");
```

```
}
?>
```

Bây giờ nếu file không tồn tại thì bạn sẽ gặp lỗi như sau:

```
File not found
```

Đoạn mã trên là hiệu quả hơn so với đoạn mã trước đó, bởi vì nó sử dụng một cơ chế xử lý lỗi đơn giản để ngăn chặn các tập lệnh sau khi lỗi.

Tuy nhiên, việc dừng đơn giản đoạn mã không phải lúc nào cũng là cách làm đúng. Hãy dựa vào các hàm trong PHP để thay thế cho việc xử lý lỗi.

Tạo ra xử lý lỗi tùy chỉnh

Tạo một tùy chỉnh xử lý lỗi là khá đơn giản. Đơn giản là tạo ra một hàm đặc biệt có thể được gọi khi một lỗi xảy ra trong PHP.

Hàm này phải có khả năng xử lý tối thiểu với hai tham số (mức độ lỗi và thông báo lỗi) nhưng có thể chấp nhận đến 5 tham số (tùy chọn: file, số dòng, trường hợp lỗi)

Cú pháp:

```
error_function(error_level,error_message,
error_file,error_line,error_context)
```

Tham số	Mô tả
error_level	Bắt buộc. Xác định mức độ thông báo lỗi cho lỗi người dùng định nghĩa. Phải có giá trị là số. Xem bảng bên dưới về các mức độ lỗi có thể gặp
errror_message	Bắt buộc. Xác định thông báo lỗi cho lỗi người dùng định nghĩa
error_file	Tùy chọn. Chỉ định tên tập tin sinh ra lỗi
error_line	Tùy chọn. Chỉ định dòng nào sinh ra lỗi
error_context	Tùy chọn. Chỉ định một mảng chứa tất cả các biến và các giá trị của chúng, sử dụng khi xảy ra lỗi.

Mức độ thông báo lỗi

Những mức độ thông báo lỗi dưới đây thuộc nhiều kiểu lỗi khác nhau do người dùng định nghĩa, có thể được sử dụng để:

Giá trị	Hằng số	Mô tả
2	E_WARNING	Lỗi không nghiêm trọng khi chạy. Đoạn mã thực thi không dừng lại.
8	E_NOTICE	Thông báo khi chạy. Đoạn mã đã tìm thấy một số thứ có thể gây ra lỗi, nhưng cũng có thể xảy ra khi chạy một tập lệnh bình thường
256	E_USER_ERROR	Thông báo lỗi người dùng. Việc này giống như một E_ERROR thiết lập bởi lập trình viên sử dụng hàm trigger_error()
512	E_USER_WARNING	Tạo ra cảnh báo cho người dùng. Nó giống như E_WARNING thiết lập bởi lập trình viên thông qua hàm trigger_error()
1024	E_USER_NOTICE	Thông báo do người dùng tạo ra. Nó giống như một E_NOTICE thiết lập bởi lập trình viên thông qua hàm trigger_error()
4096	E_RECOVERABLE_ERROR	Lỗi nghiêm trọng có thể bắt. Nó giống như một E_ERROR nhưng có thể bắt bởi xử lý do người dùng định nghĩa
8191	E_ALL	Tất cả lỗi và cảnh báo.

Bây giờ hãy tạo ra hàm để xử lý các lỗi:

```
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Ending Script";
  die();
}
```

Đoạn mã trên là một hàm xử lý lỗi đơn giản. Khi nó được đặt, nó lấy mức độ lỗi và thông báo lỗi. Sau đó xuất ra các thông tin đã lấy rồi ngắt tập lệnh.

Bây giờ chúng ta đã tạo ra một hàm xử lý lỗi, chúng ta cần xác định khi nào nó được kích hoạt.

Đặt xử lý lỗi

Việc xử lý lỗi mặc định cho PHP dựa trên các bộ xử lý lỗi có sẵn. Chúng ta sẽ làm cho các hàm trên các lỗi xử lý mặc định trong suốt thời gian của tập lệnh.

Nó có thể thay đổi các lỗi xử lý để áp dụng cho chỉ có một số lỗi, cách mà các đoạn mã có thể xử lý các lỗi khác nhau theo những cách khác nhau. Tuy nhiên, trong ví dụ này chúng ta sẽ sử dụng bộ xử lý lỗi khách hàng của chúng ta cho tất cả các lỗi:

```
set_error_handler("customError");
```

Từ khi chúng ta muốn các hàm tùy chỉnh của chúng ta xử lý tất cả các lỗi, set_error_handler() chỉ cần một tham số, tham số thứ hai có thể được thêm vào để xác định mức độ lỗi.

Ví dụ:

```
<?php
//error handler function
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr";
}

//set error handler
set_error_handler("customError");

//trigger error
echo($test);
?>
```

Đoạn code trên có thể xuất ra như sau:

Error: [8] Undefined variable: test

Kích hoạt lỗi

Trong đoạn mã trên có thể khi người dùng nhập dữ liệu, nó sẽ hữu ích để kích hoạt các lỗi khi đầu vào không hợp lệ. Trong PHP, điều này được thực hiện bởi hàm trigger_error().

Ví dụ

Trong ví dụ này một lỗi sẽ sinh ra nếu biến "test" lớn hơn "1":

```
<?php
$test=2;
if ($test>=1) {
  trigger_error("Value must be 1 or below");
}
?>
```

Đoạn mã trên có thể xuất ra như sau:

Notice: Value must be 1 or below
in **C:\webfolder\test.php** on line **6**

Lỗi có thể được kích hoạt bất cứ nơi nào bạn muốn trong một đoạn mã, và bằng cách thêm một tham số thứ hai, bạn có thể xác định mức độ lỗi được kích hoạt.

Loại lỗi có thể xảy ra:

E_USER_ERROR – Lỗi nghiêm trọng do người dùng tạo ra khi chạy. Lỗi mà không thể được phục hồi. Đoạn mã thực thi phải dừng lại
E_USER_WARNING – Cảnh báo khi người dùng chạy tạo ra lỗi. Đoạn mã thực thi không dừng lại.
E_USER_NOTICE – Mặc định. Thông báo do người dùng tạo ra khi chạy. Tập lệnh tìm thấy một thứ có thể là một lỗi, nhưng cũng có thể xảy ra khi chạy một tập lệnh bình thường.

Ví dụ:

Trong ví dụ này một E_USER_WARNING sinh ra nếu biến "test" lớn hơn "1". Nếu E_USER_WARNING sinh ra chúng ta sẽ sử dụng bộ xử lý lỗi tùy chỉnh và kết thúc đoạn mã:

```
<?php
//error handler function
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Ending Script";
  die();
}

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>=1) {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>
```

Đoạn code trên xuất ra như sau:

Error: [512] Value must be 1 or below
Ending Script

Bây giờ chúng ta đã được học cách tạo lỗi và kích hoạt chúng, hãy tìm hiểu về việc ghi chép lỗi.

Ghi chép lỗi

Mặc định, PHP gửi một ghi chép lỗi đến hệ thống lưu trữ lỗi trên server hoặc vào file, phục thuộc vào cách cấu hình chỉ mục error_log trong tập tin php.ini. Bằng việc sử dụng hàm error_log() bạn có thể gửi các ghi chép lỗi vào một file nhất định hoặc một điểm từ xa.

Gửi thông báo lỗi cho chính mình bằng e-mail có thể là một cách tốt để nhận được các thông báo lỗi cụ thể.

Gửi thông báo lỗi bằng e-mail

Trong ví dụ dưới đây chúng ta sẽ gửi e-mail với một lỗi và kết thúc đoạn mã, nếu như có lỗi sinh ra:

```
<?php
//Hàm xử lý lỗi
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Webmaster has been notified";
  error_log("Error: [$errno] $errstr",1,
  "someone@example.com","From: webmaster@example.com");
}

//Đặt bộ xử lý lỗi
set_error_handler("customError",E_USER_WARNING);

//Kích hoạt lỗi
$test=2;
if ($test>=1) {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>
```

Đoạn code trên xuất ra như sau:

Error: [512] Value must be 1 or below
Webmaster has been notified

Và mail nhận được sẽ như sau:

Error: [512] Value must be 1 or below

Điều này không nên được sử dụng với tất cả các lỗi. Những lỗi thường xuyên cần được gửi đến máy chủ bằng cách sử dụng hệ thống mặc định của PHP.

PHP Exception

Các ngoại lệ thường được sử dụng để thay đổi dòng chảy bình thường của một đoạn code nếu có lỗi cụ thể xảy ra.

Ngoại lệ là gì?

PHP 5 đưa ra một cách hướng đối tượng để đối phó với các lỗi.

Xử lý ngoại lệ được sử dụng để thay đổi dòng chảy bình thường của mã thực thi nếu một lỗi nào đó xảy ra. Tình trạng này được gọi là một ngoại lệ.

Đây là những điều thường xảy ra khi một ngoại lệ được kích hoạt:

- Trạng thái của code hiện tại được lưu
- Việc thực thi mã sẽ chuyển sang một định nghĩa trước (tùy chỉnh) chức năng xử lý ngoại lệ
- Tùy thuộc vào tình hình, xử lý sau đó có thể tiếp tục thực thi từ trạng thái code lưu lại, chấm dứt việc thực hiện tập lệnh hoặc tiếp tục tập lệnh từ một vị trí khác nhau trong mã

Chúng ta sẽ trình bày các phương pháp xử lý lỗi khác nhau:

- Sử dụng cơ bản của ngoại lệ
- Tạo một trình xử lý ngoại lệ tùy chỉnh
- Nhiều trường hợp ngoại lệ
- Đưa vào một ngoại lệ
- Thiết lập một ngoại lệ xử lý cấp cao

Lưu ý: Các ngoại lệ chỉ nên sử dụng với điều kiện lỗi, và không nên được sử dụng để chuyển đến một nơi khác trong mã tại một điểm xác định.

Cách sử dụng ngoại lệ cơ bản

Khi một ngoại lệ được ném ra, các mã sau nó sẽ không được thực thi, và PHP sẽ cố gắng để tìm ra khối "bắt lỗi" phù hợp.

Nếu một ngoại lệ không bị bắt, một lỗi nghiêm trọng sẽ được ban hành với một thông báo "Uncaught Exception".

Cho phép thử để ném một ngoại lệ mà không bắt nó:

```
<?php
//create function with an exception
function checkNum($number) {
  if($number>1) {
    throw new Exception("Value must be 1 or below");
  }
  return true;
}

//trigger exception
checkNum(2);
?>
```

Đoạn mã trên sẽ nhận được một lỗi như sau:

Fatal error: Uncaught exception 'Exception'
with message 'Value must be 1 or below' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown in **C:\webfolder\test.php** on line **6**

Try, throw và catch

Để tránh lỗi từ ví dụ trên, chúng ta cần phải tạo ra các mã thích hợp để xử lý các trường hợp ngoại lệ.

Dạng ngoại lệ thích hợp bao gồm:

1. Try - Một hàm sử dụng một ngoại lệ phải ở trong một khối “try”. Nếu ngoại lệ không kích hoạt, mã sẽ tiếp tục như bình thường. Tuy nhiên nếu các ngoại lệ kích hoạt, một ngoại lệ được "throw"
2. Throw - Đây là cách bạn kích hoạt một ngoại lệ. Mỗi "throw" phải có ít nhất một "catch"
3. Catch - Một khối “catch” lấy ra một ngoại lệ và tạo ra một đối tượng có chứa các thông tin ngoại lệ

Hãy thử kích hoạt một ngoại lệ với đoạn code phù hợp:

```
<?php
//create function with an exception
function checkNum($number) {
  if($number>1) {
    throw new Exception("Value must be 1 or below");
  }
  return true;
```

```
}

//trigger exception in a "try" block
try {
  checkNum(2);
  //If the exception is thrown, this text will not be shown
  echo 'If you see this, the number is 1 or below';
}

//catch exception
catch(Exception $e) {
  echo 'Message: ' .$e->getMessage();
}
?>
```

Đoạn mã trên sẽ nhận được một lỗi như thế này:

```
Message: Value must be 1 or below
```

Giải thích ví dụ:

Đoạn mã trên ném một ngoại lệ và bắt nó:

1. Hàm checkNum() được tạo ra. Nó kiểm tra nếu một số lớn hơn 1. Nếu có, một ngoại lệ được ném
2. Hàm checkNum() được gọi trong một khối "try"
3. Ngoại lệ trong hàm checkNum() được ném
4. Khối "catch" lấy ngoại lệ và tạo ra một đối tượng ($e) có chứa các thông tin ngoại lệ
5. Các thông báo lỗi từ các ngoại lệ được lặp lại bằng cách gọi $ e-> getMessage() từ các đối tượng ngoại lệ

Tuy nhiên, một cách để lấy xung quanh qui tắc "mỗi throw phải có một catch" là thiết lập một trình xử lý ngoại lệ mức đầu để xử lý các lỗi mà lọt qua.

Tạo lớp ngoại lệ tùy chỉnh

Để tạo ra một trình xử lý ngoại lệ tùy chỉnh, bạn phải tạo ra một lớp đặc biệt với chức năng có thể được gọi khi một ngoại lệ xảy ra trong PHP. Lớp đó phải là mở rộng của lớp ngoại lệ.

Các lớp ngoại lệ tùy chỉnh được thừa kế từ lớp ngoại lệ của PHP và bạn có thể thêm các chức năng tùy chỉnh cho nó.
Hãy tạo ra một lớp ngoại lệ:

```
<?php
class customException extends Exception {
  public function errorMessage() {
    //error message
    $errorMsg = 'Error on line '.$this->getLine().' in '.$this-
>getFile()
    .': <b>'.$this->getMessage().'</b> is not a valid E-Mail
address';
    return $errorMsg;
  }
}

$email = "someone@example...com";

try {
  //check if
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
    //throw exception if email is not valid
    throw new customException($email);
  }
}

catch (customException $e) {
  //display custom message
  echo $e->errorMessage();
}
?>
```

Lớp mới là một bản sao của lớp ngoại lệ cũ với một sự bổ sung của hàm errorMessage(). Vì nó là một bản sao của lớp cũ, và nó được thừa hưởng các thuộc tính và các phương thức từ các lớp cũ, chúng ta có thể sử dụng các phương thức lớp ngoại lệ như getline () và getfile () và getMessage ().

Giải thích ví dụ:

Đoạn mã trên ném một ngoại lệ và bắt nó với một lớp ngoại lệ tùy chỉnh:

1. Lớp customException() được tạo ra như phần mở rộng của lớp ngoại lệ cũ. Bằng cách này, nó thừa kế tất cả các phương thức và thuộc tính từ lớp ngoại lệ cũ
2. Hàm errorMessage() được tạo ra. Hàm này trả về một thông báo lỗi nếu địa chỉ e-mail không hợp lệ
3. Biến $email là một chuỗi với địa chỉ email không hợp lệ
4. Khối “try” được thực thi và một ngoại lệ được ném từ địa chỉ e-mail không hợp lệ
5. Khối “catch” bắt ngoại lệ và hiển thị các thông báo lỗi.

Nhiều trường hợp ngoại lệ

Một đoạn mã có để sử dụng nhiều trường hợp ngoại lệ để kiểm tra nhiều điều kiện.

Có thể sử dụng một vài khối if..else, một switch, hoặc làm tổ nhiều trường hợp ngoại lệ. Những trường hợp ngoại lệ có thể sử dụng các lớp ngoại lệ khác nhau và trả lại thông báo lỗi khác nhau:

```
<?php
class customException extends Exception {
  public function errorMessage() {
    //error message
    $errorMsg = 'Error on line '.$this->getLine().' in '.$this-
>getFile()
    .': <b>'.$this->getMessage().'</b> is not a valid E-Mail
address';
    return $errorMsg;
  }
}

$email = "someone@example.com";

try {
  //check if
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
    //throw exception if email is not valid
    throw new customException($email);
  }
  //check for "example" in mail address
  if(strpos($email, "example") !== FALSE) {
    throw new Exception("$email is an example e-mail");
  }
}

catch (customException $e) {
  echo $e->errorMessage();
}

catch(Exception $e) {
  echo $e->getMessage();
}
?>
```

Giải thích ví dụ:

Đoạn mã trên kiểm tra hai điều kiện và ném một ngoại lệ nếu các điều kiện không được đáp ứng:

1. Lớp customException () được tạo ra như là một phần mở rộng của lớp ngoại lệ cũ. Bằng cách này, nó được thừa hưởng tất cả các phương thức và thuộc tính từ các lớp ngoại lệ cũ
2. Hàm errorMessage() được tạo ra. Hàm này trả về một thông báo lỗi nếu một địa chỉ e-mail không hợp lệ
3. Biến $email được thiết lập là một chuỗi chứa một địa chỉ e-mail hợp lệ, nhưng có chứa chuỗi "example"
4. Khối “try” được thực thi và một ngoại lệ không được ném vào điều kiện đầu tiên
5. Điều kiện thứ hai gây ra một ngoại lệ từ khi e-mail có chứa chuỗi "example"
6. Khối "catch" bắt ngoại lệ và hiển thị chính xác các thông báo lỗi.

Nếu ném ngoại lệ là lớp customException và không có customException bắt, chỉ các ngoại lệ cơ sở được bắt, ngoại lệ sẽ được xử lý ở đó.

Ném lại các ngoại lệ

Đôi khi, khi một ngoại lệ được ném ra, bạn mong muốn xử lý nó khác với cách tiêu chuẩn. Có thể ném một ngoại lệ một lần thứ hai trong khối “catch”

Đoạn mã nên giấu lỗi hệ thống từ người dùng. Lỗi hệ thống có thể quan trọng cho các coder, nhưng không quan trọng với người sử dụng. Để mọi việc dễ dàng hơn với người sử dụng, bạn có thể lại ném ngoại lệ với một thông điệp thân thiện với người sử dụng:

```
<?php
class customException extends Exception {
  public function errorMessage() {
    //error message
    $errorMsg = $this->getMessage().' is not a valid E-Mail
address.';
    return $errorMsg;
  }
}

$email = "someone@example.com";

try {
  try {
    //check for "example" in mail address
    if(strpos($email, "example") !== FALSE) {
      //throw exception if email is not valid
      throw new Exception($email);
    }
  }
  catch(Exception $e) {
    //re-throw exception
```

```
    throw new customException($email);
  }
}

catch (customException $e) {
  //display custom message
  echo $e->errorMessage();
}
?>
```

Giải thích ví dụ:

Đoạn mã trên kiểm tra nếu địa chỉ e-mail chứa chuỗi “example” trong đó, nếu có, ngoại trừ được ném lại:

1. Lớp customException() được tạo ra như phần mở rộng của lớp ngoại lệ cũ. Bằng cách này, nó được thừa hưởng tất cả các phương pháp và thuộc tính từ các lớp ngoại lệ cũ
2. Hàm errorMessage() được tạo ra. Hàm này trả về một thông báo lỗi nếu một địa chỉ e-mail không hợp lệ
3. Biến $email được thiết lập là một chuỗi địa chỉ e-mail hợp lệ, nhưng có chứa chuỗi "example"
4. Khối "try" chứa một khối "try" khác để làm cho nó trương hợp để lại ném ngoại lệ
5. Các ngoại lệ được kích hoạt kể từ khi e-mail có chứa chuỗi "example”
6. Các khối “catch” bắt ngoại lệ và ném lại một "customException"
7. Các "customException" bị bắt và hiển thị một thông báo lỗi

Nếu ngoại lệ không được bắt trong khối "try" hiện tại của nó, nó sẽ tìm kiếm một khối catch "cấp cao hơn".

Đặt một xử lý ngoại lệ mức độ top

Hàm **set_exception_handler**() thiết lập một hàm người dùng định nghĩa để xử lý tất cả các trường hợp ngoại lệ tự do.

```
<?php
function myException($exception) {
  echo "<b>Exception:</b> " . $exception->getMessage();
}

set_exception_handler('myException');
```

```
throw new Exception('Uncaught Exception occurred');
?>
```

Đoạn mã trên có thể xuất ra như sau:

```
Exception: Uncaught Exception occurred
```

Trong đoạn mã trên không có khối “catch”. Thay vào đó, việc xử lý ngoại lệ mức đầu kích hoạt. Chức năng này nên được sử dụng để bắt ngoại lệ tự do.

Quy định đối với các trường hợp ngoại lệ

- Code có thể được bọc trong khối try, để giúp bắt các ngoại lệ tiềm ẩn
- Mỗi khối try hoặc "throw" phải có ít nhất một khối catch tương ứng
- Nhiều khối catch có thể được dùng để bắt các ngoại lệ khác nhau
- Các ngoại lệ có thể được ném (hoặc tái ném) trong một khối catch thuộc một khối try.

Một nguyên tắc đơn giản: Nếu bạn ném một cái gì, bạn phải bắt cái nó.