Chương 1

Giới thiệu lập trình web php

Các mô hình lập trình ứng dụng client-

server

• Mô hình 2 lớp

– Ưu điểm

Dữ liệu tập trung:

đảm bảo dữ liệu được

nhất quán.GOLO

Dữ liệu được chia sẻ cho nhiều người dùng.

– Khuyết điểm

Các xử lý tra cứu và cập

nhật dữ liệu được thực hiện ở Database Server, việc nhận kết quả và hiển thị phải được thực hiện ở Client. Khó khăn trong vấn đề bảo trì và nâng cấp. □Khối lượng dữ liệu truyền trên mạng lớn chiếm dụng đường truyền, thêm gánh nặng cho Database Server.

Các mô hình lập trình ứng dụng client-

server

• Mô hình 3 lớp

– Thêm Server giữ nhiệm vụ tương tác giữa Client và Database server, giảm bớt

các xử lý trên Database server, tập trung các xử lý nhận và hiển thị dữ liệu tại Application serve.

– Ưu điểm

• □Hỗ trợ nhiều người dùng

• □Giảm bớt xửlý cho Client ->Không yêu cầu máy tính ở Client có cấu hình mạnh.

• □Xử lý nhậnvà hiển thị dữ liệu tập trung tại Application Server <-> dễ quản lý, bảo trì và nâng cấp.

• □Xử lý truy cập dữ liệutập trung tại Database Server.

– Khuyết điểm

• □Phải sửdụng thêm một Application Server ->Tăng chi phí

Mô hình ứng dụng web

• Ứng dụng web

– Dựa trên mô hình Client-Server 3 lớp. Đây là một hệ thống phức tạp, dựa trên nhiều yếu tố: phần cứng, phần mềm, giao thức, ngôn ngữ và thành phần giao diện.

– Loại ứng dụng Internet cho phép cho phép các máy (Client) sử dụng trình duyệt Web (Internet Explorer, firefox. ..) để truy cập và xem thông tin được cung cấp bởi trình chủ Web (Web Server).

– Server : Một máy tính trung tâm lưu trữ trong ổ cứng ở thư mục mạng các tập tin làm việc. Server còn là phần mềm mà được cài đặt trên máy tính đó và cho phép xử lý truy vấn của các máy tính khác ở xa văn phòng, cũng như phản hồi các truy vấn đó. Sự tương tác này được thực hiện theo những qui tắc nhất định, các giao thức.

– Máy chủ webserver là các máy chủ có cài các phần mềm web server : Phần mềm chuyên xử lý các thông tin về web của client. Có nhiều phần mềm webserver: IIS, Apache.

– Client là máy tính khởi xướng truy vấn cho Server. Đây là máy có cài các phần mềm yêu cầu xử lý từ server như trình duyệt Web.

– Những ứng dụng web động: là các ứng dụng web có nội dung chứa trong các cơ sở dữ liệu và được truy xuất bởi webserver. Tùy thuộc vào yêu cầu của client, server sẽ lấy dữ liệu phù hợp trả về cho client hiển thị lên trình duyệt web.

Ứng dụng web và giao thức http

Giao thức HTTP

• Giao thức HTTP (Hypertext Transfer Protocol). Trình duyệt web liên

lạc với webserver thông qua một giao thức riêng: giao thức HTTP.

• HTTP được xác định qua URLs (Uniform Resource Locators), với cấu trúc chuỗi có định dạng như sau: http: // <host> [: <port>] [

<path> [? <query>]]

• Ví dụ:

1. http://localhost:8080/myweb/index.php?a=1&b=2

Host: localhost. Port: 8080, path: myweb/index.php, Query:

a=1&b=2

2. http://www.thanhnien.com.vn/pages/default.aspx

Trang web default.aspx được lưu trữ trong thư mục /page tại Web Server với host là www.thanhnien.com.vn. Port: không có (port= 80: Mặc định).

Các ngôn ngữ lập trình web

• Có 2 nhóm ngôn ngữ lập trình web: Ngôn ngữ lập trình phía client và ngôn ngữ lập trình phía server.

• Client side:

– Code chạy trên trình duyệt web.

– Ngôn ngữ tiêu biểu: javascript, vbscript, actionScript.

• Server side: là các ngôn ngữ mà mã được dịch và chạy trên máy chủ.

– Asp.net:

• là môi trường lập trình của microsoft, sử dụng C#, VB.net, ….

• Webserver: IIS

– Java: Ngôn ngữ java,máy chủ web IIS, apache,…

– Php: Ngôn ngữ mã nguồn mở, sử dụng webserver IIS, Apache,..

Ứng dụng web với PHP

• Giới thiệu:

– Mã nguồn mở miễn phí đang được phát triển nhanh trong những năm gần đây.

– Php được chạy trên web sever IIS hoặc Apache.

– Php có thể thao tác với mọi CSDL, nhưng thông thường đi với Mysql- một hệ quản trị CSDL miễn phí.

– Code chạy nhanh, ổn định.

– Php hỗ trợ rất nhiều hàm giúp lập trình web ngày càng thuận lợi hơn.

– Có nhiều framework, CMS mã nguồn mở và ngày càng được sử dụng nhiều.

Ứng dụng web với PHP

• Số liệu: Tham khảo http://w3techs.com/

Cài đặt môi trường

• Hệ thống: Windows, Apache, php, mysql, phpMyAdmin

– Webserver: Apache:

• Download miễn phí tại: http://httpd.apache.org/download.cgi

• Cài đặt:

• Kiểm thử: http://localhost

– Php:

• Download miễn phí tại: http://php.net/downloads.php

• Cài đặt:

• Kiểm thử:

– Mysql: hệ quản trị CSDL

• Download miễn phí tại: http://dev.mysql.com/downloads/windows/

– PhpMyAdmin: Công cụ quản trị CSDL mysql

• Download miễn phí tại:

http://www.phpmyadmin.net/home\_page/downloads.php

• Cài đặt.

• Kiểm tra.

Cài đặt thay thế

- Để dễ dàng cho nười sử dụng khi cài đặt hệ thống web php, thay vì phải cài đặt và cấu hình từng thành phần Apache + Php+Mysql và PhpMyadmin, ta có thể sử dụng một số phần mềm thay thế miễn phí khác.

- Khi cài đặt một phần mềm này, tức là chúng ta đã cài đặt và cấu hình xong để chạy apache, php, mysql và phpmyadmin.

- Các phần mềm này tích hợp toàn bộ hệ thống web và tự cấu hình để chúng có thể chạy thông suốt.

- Ta chỉ nên cài một trong những phần mềm kể trên (không nên cài đặt cùng lúc 2 phần mềm)

Cài đặt wamp server

• Wamp (Windows Apache Mysql Php) server: là phần mềm miễn phí

tích hợp các thành phần của môi trường web php. Cài đặt thành công wamp, ta đã cài đặt và cấu hình thành công apache, php, mysql và phpmyadmin.

• Download miễn phí tại: http://www.wampserver.com/en/

• Chú ý: Phiên bản hiện tại của wamp server (12/2014) là 2.5 (php5.5,

mysql 5, apache 2.49):

– Phiên bản không thích hợp với windows XP (không dùng sp3), Windows Server 2003.

– Có 2 bản cho windows 32 và 64 bit.

– Máy cài đặt cần cài: Visual C++ Redistributable for Visual Studio 2012.

– Tải và cài đặt Visual C++ 2012 tại: http://www.microsoft.com/en-

us/download/details.aspx?id=30679

Cài đặt wamp server

Cài đặt wamp server

• Cài đặt: mặc định tại c:\wamp

– Click vào file vừa tải về: wampserver2.5-Apache-2.4.9-

Mysql-5.6.17-php5.5.12-32b.exe

– Chọn Next,…

– …

– finish

• Khởi động wamp.

• Kết quả:

– Mở trình duyệt web

và chay thử :http://localhost

– http://localhost/?phpinfo=1 để kiểm tra chi tiết các kết quả cài đặt

Cấu hình wamp server

• Kết quả thư mục cài đặt của wamp

– C:\wamp\bin\apche: Chứa máy chủ apache

– C:\wamp\bin\mysql: chứa hệ quản trị CSDL Mysql

– C:\wamp\bin\php: Chứa trình dịch php

– C:\wamp\apps\phpmyadmin4.1.14: Chứa

công cụ quản trị csdl mysql thông dụng trên web là phpMyadmin.

– C:\wamp\www: Là thư mục chứa source

code của các ứng dụng php, đường dẫn này được ánh xạ thông qua apache là http://localhost hay http://127.0.0.1

Cấu hình Wamp server

• Sau khi cài đặt xong, có thể chạy được các ứng dụng php mà không cần cấu hình. Tuy nhiên, để chạy được các chức năng khác, và để thuận lợi hơn khi cài và chạy ứng dụng web, ta nên cấu hình lại các thành phần này.

• Mỗi thành phần apache, php, mysql và phpmyadmin có các file cấu hình riêng và được lưu trong các thư mục tương ứng. Đây là các file text, có thể sử dụng notepad++, dreamweaver để mở và sửa dễ dàng.

• Vị trí các file cấu hình như sau

– Apache: C:\wamp\bin\apache\apache2.4.9\conf\httpd.conf

– Php: C:\wamp\bin\apache\apache2.4.9\bin\php.ini

– Mysql: C:\wamp\bin\mysql\mysql5.6.17\my.ini

– Phpmyadmin: C:\wamp\apps\phpmyadmin4.1.14\ config.inc.php

Cấu hình wamp sever

• Ta cũng có thể mở và sửa nhanh các file cấu hình từ wamp manager

• Sau khi sửa xong nên bấm vào: restart All Servises trên wamp manager để cập nhật và load cấu hình mới.

Cấu hình apache Cấu hình php Cấu hình mysql

Cấu hình Server Apache

• Cấu hình apache: config/httpd.conf

– Dòng lệnh, chú thích?

– Restart lại apache mỗi khi cần update sự thay đổi.

– Ví dụ:

Listen 80

LoadModule rewrite\_module modules/mod\_rewrite.so DocumentRoot

<IfModule dir\_module>

DirectoryIndex index.php index.php3 index.html index.htm

</IfModule>

AddType application/x-httpd-php .php

Cấu hình website

• Một máy chủ có thể có nhiều website. Mỗi website có thể được cấu hình riêng dựa vào file

.htaccess.

• Cấu hình cho website, thư mục: ta đặt file văn bản có tên .htaccess trong thư mục đó.

• File .htaccess: chỉ có tác dụng nếu thuộc tính AllowOverride trong httpd.conf của apache là all

• Mỗi thư mục có thể được config dựa vào file .htaccess

• Thư mục chứa file .htaccess sẽ có cấu hình chung kết hợp của httpd.conf và .htaccess

• Cấu trúc file .htaccess giống file httpf.conf

• Cần thiết lập chế độ bảo vệ cho .htaccess (có thể chmod cho file là 644).

• Ví dụ:

RewriteRule ^gioi-thieu.html index.php?p=gioithieu RewriteRule ^gio-hang.html index.php?p=giohang

RewriteRule ^tin-tuc/(.\*).html index.php?p=chitiettin&matin=$1

RewriteRule ^tin-tuc.html/(.\*) index.php?p=tintuc&page=$1 RewriteRule ^tin-tuc.html index.php?p=tintuc

RewriteRule ^san-pham/(.\*).html$

index.php?p=sanpham&loai=$1&page=1&%{QUERY\_STRING} [L]

RewriteRule ^chi-tiet-san-pham/(.\*)/(.\*).html$ index.php?p=chitietsp&loai=$1&masp=$2 [L]

Cấu hình php

• Mỗi máy chủ chạy php có một file văn bản để cấu hình cho trình dịch php.

File có tên: php.ini, (vị trí C:\wamp\bin\apache\apache2.4.9\bin)

• Nội dung:

– Mỗi dòng trong file cấu hình này có thể là chú thích hoặc câu lệnh.

– Dòng chú thích: bắt đầu bằng dấu chấm phẩy (;)

– Dòng lệnh: không có dấu ;

– Restart lại apache để load cấu hình mới

• Ví dụ

short\_open\_tag = On display\_errors = On post\_max\_size = 2M file\_uploads = On

upload\_tmp\_dir = "c:/wamp/tmp " upload\_max\_filesize = 2M

extension=php\_mysql.dll extension=php\_gd2.dll

Cấu hình database mysql

• Cấu hình Mysql

– File cấu hình: my.ini (hoặc my.cnf trong hệ linux) trong thư mục cài đặt mysql

– Dòng chú thích: bắt đầu bằng #

– Dòng lệnh:

– Restart lại mysql để load cấu hình mới

– Ví dụ:

datadir=c:/wamp/bin/mysql/mysql5.1.30/data

skip-innodb

• Cấu hình phpMyAdmin

– File config.inc.php trong thư mục chưa source

– Mở file và sửa lại các dòng config tới database phù hợp:

– Ví dụ:

$cfg['Servers'][$i]['host'] = 'localhost';

$cfg['Servers'][$i]['user'] = 'root';

$cfg['Servers'][$i]['password'] = '';

Cài đặt một ứng dụng web

• Tạo csdl

• Upload source code tới thư mục web.

• Config code kết nối tới csdl

Câu hỏi

Chapter 2

Lập trình php căn bản

Ngôn ngữ lập trình php

• Giới thiệu

• Đặc điểm ngôn ngữ Php

• Mô hình ứng dụng web php

• Đặc điểm một trang php

• Hằng số, Biến, kiểu dữ liệu,các phép toán cơ bản

• Các cấu trúc điều khiển

• Một số lệnh cơ bản trong php

• Xây dựng hàm trong php.

• Xử lý lỗi trên Php

Giới thiệu

• PHP (Hypertext Preprocessor):ngôn ngữ script trên server được thiết kế để dễ dàng xây dựng các trang Web động.

• Mã PHP có thể thực thi trên Webserver để tạo ra mã HTML và xuất ra trình duyệt web theo yêu cầu của người sử dụng.

• Ngôn ngữ PHP ra đời năm 1994 Rasmus Lerdorf . Phiên bản hiện tại là 7.2.x (2018)

Đặc điểm

• Có nhiều hàm, thủ tục sẵn chuyên dụng trong lập trình Web.

• Dễ học, dễ sử dụng với cú pháp kết hợp giữa C và Perl.

• Là ngôn ngữ Script mã nguồn mở.

• Là ngôn ngữ lập trình hướng đối tượng gần với với C++.

• PHP có bản chạy trên hầu hết các hệ điều hành: MS Windows, Linux, SunOS, … Quá trình cài đặt đơn giản, dễ dàng.

• Trợ giúp mạnh cho CSDL qua các hàm, thủ tục sẵn có do vậy dễ sử dụng và có tốc độ cao. Hiện tại PHP hỗ trợ các CSDL của Oracle, MS SQL, PostgreSQL, Interbase...Với các CSDL không hỗ trợ, PHP có thể kết nối qua ODBC.

• Chi phí thấp, chạy rất tốt trên các hệ quản trị CSDL miễn phí

không đòi hỏi bản quyền như: MySQL, PostgreSQL, …

• Có tốc độ thực thi cao và gây tải ít cho máy chủ.

Mô hình xử lý trang php

 1: Trình duyệt gửi yêu cầu tới trang PHP.

 2: Web server gửi các yêu cầu đó tới trình thông dịch PHP.

 3-4: Trình thông dịch PHP thực thi các đọan mã PHP. Quá trình này có thể liên quan đến nhiều tài nguyên như filesystem, database...

 5: Kết quả của quá trình thông dịch là các mã HTML được trả về

cho Server..

 6: Server gửi mã kết quả HTML về lại trình duyệt. Trình duyệt sẽ dịch kết quả html này và hiện lên màn hình client

Đặc điểm một trang php

• Tên file: Mặc định có phần mở rộng bằng php (có thể thay bằng tên khác dựa vào cấu hình của apache). Khi gặp file .php, apache sẽ gửi trực tiếp nội dung file này sang trình dịch php để xử lý và chờ nhận kết quả trả về.

• Một trang php có thể chứa các mã html và php đan xen nhau. Khi muốn viết php, ta đặt code php trong các cặp thẻ <?php và ?>, <? và

?>

Thực thi một file php

• Cách trình thông dịch php dịch một file php:

– Tạo file c:\wamp\www\test\a.php có nội dung:

<html><head>

<?php $a = 3; $b=2;?>

</head>

<body> Tong a va b =<?php echo $a + $b;?>

</body>

</html>

— Mở trình duyệt web, nhập vào: http://localhost/test/a.php

• Kết quả trả về cho client?

• Kết quả hiển thị trên máy client?

Đặc điểm trang php

• Trong một trang php, các mã php xen kẽ với html

• Các hằng, biến tạo ra trong trang, sẽ bị hủy khi trang kết thúc.

• Các biểu thức, phép toán cơ bản, cấu trúc điều khiển, các qui tắc đặt tên (biến, hàm, hằng số,… ), chú thích của php giống với ngôn ngữ lập trình C++.

• Tên các hằng số, biến số của php: phân biệt hoa,

thường.

• Tên các hàm số: không phân biệt hoa thường.

Hằng và Biến số

• Hằng số: Các vùng lưu trữ không thay đổi dữ liệu

– Tạo hằng số: sử dụng hàm define("ten\_hang\_so", gia\_tri)

– Kiểm tra hằng số tồn tại chưa: defined('hang\_so')

– Sử dụng. define("S", "Chu Vi:");

$s = M\_PI \* 5\*2;

//M\_PI: là hằng số PI trong toán học đã được định nghĩa bởi php

Echo S . $s; //Chu Vi: 31.416

Hằng và Biến số (tt)

• Biến số. Vùng lưu trữ dữ liệu có thể thay đổi. Biến số luôn bắt đầu bằng ký tự $.

– Biến trong php không bắt buộc phải khai báo kiểu dữ liệu.

– Biến có thể chứa các kiểu dữ liệu khác nhau trong các thời điểm

khác nhau.

– Tạo biến: $var = giá trị; vd: $s = 10;

– Sau khi tạo biến, ta có thể sử dụng biến trong các biểu thức. Nếu chưa tạo biến mà đã sử dụng, ứng dụng sẽ báo lỗi. Thường cần kiểm tra biến đã tồn tại hay chưa trước khi sử dụng.

– Kiểm tra biến đã tồn tại hay chưa: isset($var): trả về true/false nếu biến đã/chưa tồn tại

– Hủy một biến khỏi vùng nhớ: unset($var)

– Hàm xem thông tin của biến: print\_r($var) hoặc var\_dump($var)

Các phép toán

• Kiểu number:

 +, -, \*, /, %,

 ++, --,

 +=, -=, \*=, /=, %=

Ví dụ:

<?php

$a=10; $b=8;

$x1 = $a/$b;//1.25

$x2 = $a % $b;//2

$x3 =5; $x3 +=$b;//13

$a--;//9

?>

Các phép toán

• Kiểu boolean: kết quả các phép toán là true/false:

 >, <, >=, <=, ==, !=, ===, !==

 &&, ||, !

• Ví dụ:

<?php

$a =5; $b=7; $c="5";

$x1 = $a>$b; $x2= $a==$c; $x3 = $a === $c;

if ($x1==true) echo "x1: true ";

else echo "x1: false "; //x1: false if ($x2==true) echo "x2: true ";

else echo "x2: false "; //x2: true if ($x3==true) echo "x3: true ";

else echo "x3: false "; //x3: false

?>

Phép toán ba ngôi ?

• Cú pháp: Biểu thức logic? Biểu thức 1: Biểu thức 2;

• Ta thường sử dụng phép toán này thay cho phát biểu if, else. Phép toán nhận biểu thức logic. Nếu biểu thức này true, giá trị của biểu thức 1 được trả về, ngược lại, giá trị biểu thức 2 được trả về.

• $x= $a>$b?$a:$b;  if ($a>$b) $x=$a; else $x=$b;

<?

$a = 4; $b=5;

$c = ($a>$b)? $a:$b;

echo $c; //5

$d = isset($x)?$x:0;

/\*Tương đương:

if (isset($x)) $d= $x;

else $d = 0;

\*/

?>

Phép Toán Error

• Php cung cấp phép toán điều khiển một biểu thức khi lỗi xảy ra là phép toán @. Khi đặt ký hiệu này trước biểu thức, bất cứ lỗi nào phát sinh bởi biểu thức sẽ bị lờ đi mà không hiển thị các lỗi xuất hiện của hệ thống.

<?php

$a = 0;

$b = 4;

$c = $b / $a;

echo "Gia tri cua c la : $c"; echo "<br>Loi chia cho 0.";

?>

<?php

$a = 0;

$b = 4;

$c = @($b / $a);

echo "Gia tri cua c la : $c"; echo "<br>Loi chia cho 0.";

?>

Các phép toán (tt)

• Kiểu String

– Phép toán nối chuỗi: (.) dấu chấm.

– Tạo chuỗi:

• Cách 1: Sử dụng cặp dấu nháy đơn

$s='Monday‘;

• Cách 2:Sử dụng cặp dấu nháy kép

$s= " Monday ";

• Cách 3: Sử dụng cấu trúc heredoc: Toán tử <<< và chuỗi định nghĩa kết

thúc chuỗi. Dòng xác định kết thúc chuỗi nằm độc lập trên 1 hàng

$s=<<<EOS

Nội dung Chuỗi

EOS;

Trong đó: EOS: là một chuỗi do người sử dụng đặt ra

• Cách 4: Sử dụng cấu trúc nowdoc: (php >= 5.3). Giống cú pháp heredoc nhưng định nghĩa kết thúc chuỗi được đặt trong cặp dấu nháy đơn

Các ký tự đặc biệt trong chuỗi

– Ký tự đặc biệt và cách xử lý: là các ký tự thoát (' trong cách 1 , " : trong cách 2, \,…)

– Xử lý: ta đặt ký tự backslash (\) trước ký tự đặc

Phân tích biến trong chuỗi

• Là trường hợp, trong chuỗi tạo ra có chứa các biến, giá trị các biến này sẽ thay thế biến ngay tại vị trí xuất hiện của biến trong chuỗi.

• Trường hợp này không xảy ra khi tạo chuỗi

bao bởi các dấu nháy đơn.

• Với các biến phức tạp: Mảng, đối tượng,..ta có

thể sử dụng cặp dấu {} bao quanh biến để được xử lý như biến đơn.

Ví dụ

• Ví dụ:

$s1= 5; $s2= 10;

$s3=$s1+ $s2;

$s4='Tổng $s1 + $s2 =$s3';

$s5 ="Tổng $s1 + $s2 =$s3";

$s6=<<<qwerty Tổng $s1 + $s2 = $s3 qwerty;

$arr = array("x1"=>4,"x2"=>6);

$s7= "arr['x1'] là {$arr['x1']} ";

echo " $s4 <br> $s5 <br>$s6 <br> $s7 ";

Cấu trúc điều khiển

• Cấu trúc tuần tự

– Lệnh

– Khối lệnh {}

• Cấu trúc lựa chọn

– If

– If…else

– Switch()

• Cấu trúc lặp

– For

– While

– Do..while

Cấu trúc tuần tự - cấu trúc khối

• Phát Biểu Tuần Tự

• Các câu lệnh trong php được thực thi tuần tự từ trên xuống dưới. Một tập hợp các câu lệnh đặt trong cặp dấu {} gọi là một khối lệnh. Khối lệnh cũng được xem như một câu lệnh đơn.

• Trong một khối lệnh, ta có thể đóng (?>), mở (<?php) thẻ php

<?php

$a = 3; $b=4;

if ($a>$b){ echo "<div stye='background:red'> a lớn hơn </div>";

}

else

?>

{

?>

<div style="background:green"> b lớn hơn </div>

<?php

}

Phát biểu lựa chọn

Cấu trúc if ..else

if (Biểu\_thức\_ĐK) S; Lệnh S sẽ được thi hành nếu Biểu\_thức\_ĐK có giá trị true if ($a > $b) echo "a is bigger than b";

• Biểu thức điều kiện if...else:

if (Biểu\_thức\_ĐK) S1; else S2;

Nếu Biểu\_thức\_ĐK đúng, S1 được thi hành, ngược lại (Biểu\_thức\_ĐK sai), S2 được thi hành. S1 và S2 là các câu lệnh đơn hay khối lệnh.

• Biểu thức điều kiện 3: Elseif: Kết hợp giữa if và else. if (Biểu\_thức\_ĐK\_1) S1;

elseif (Biểu\_thức\_ĐK\_2) S2 Else S3;

Nếu biểu Biểu\_thức\_ĐK\_1 đúng, S1 được thi hành, ngược lại nếu Biểu\_thức\_ĐK\_2 đúng thì S2 được thi hành, ngược lại S3 được thi hành. Đây chính là cách viết khác của biểu thức if..else lồng nhau:

if (Biểu\_thức\_ĐK\_1) S1; Else {

if (Biểu\_thức\_ĐK\_2) S2 else S3;

}

Phát biểu lựa chọn switch

• Giống cấu trúc if-else, nhưng sử dụng khi có nhiều lựa chọn

switch (Biểu\_thức)

{

case giátrị\_1: S1; break; case giátrị\_2: S2; break; case: giá trị\_n: Sn; break; [defaul: Sn+1]

}

Kiểm tra biểu\_thức, nếu biểu thức bằng giá trị giátrị\_i (i=1 .. n) sẽ thi hành lệnh (hay khối lệnh) Si tương ứng. Ngược lại, sẽ thi hành lệnh Sn+1.

Sau mỗi câu lệnh Si, ta sử dụng phát biểu break; để thoát khỏi switch. Nếu không có break sau lệnh Si, chương trình sẽ thi hành lệnh Si+1 khi đã thi hành Si.

Ví dụ

Phát biểu lặp

• Các loại phát biểu lặp

– Vòng lặp for

– Vòng lặp while

– Vòng lặp do..while

– Vòng lặp foreach

Vòng lặp for

• Thường sử dụng trong các phát biểu lặp biết trước số lần lặp. Cú pháp: for (expr1; expr2; expr3) S; Trong đó expr1; expr2; expr3 là các biểu thức.

S: là câu lệnh đơn hoặc kép.

• Thực thi vòng lặp:

– expr1 được thực thi trước tiên và duy nhất một lần.

Thường đây là biểu thức để gán giá trị cho biến lặp.

– Bắt đầu ở mỗi vòng lặp, expr2 được thi hành. Nếu giá trị trả về của biểu thúc này là false, vòng lặp dừng. Nếu giá trị trả về là true, lệnh S được thi hành. Cuối cùng thì expr3 được thi hành. Thông thường đây là biểu thức thay đổi biến điều khiển trong vòng lặp. Các biểu thức ở trên có thể rỗng, khi đó ta có thể sử dụng toán tử break; để thoát khỏi vòng lặp.

Vòng lặp for

Ví dụ

• Ví dụ: In ra bảng cửu chương $n

<?php

$n = 6;

?>

<table><tr><td colspan=3>Bảng cửu chương <?php echo $n;?></td></tr>

<?php

for($i=1; $i<=10; $i++)

{

}

?>

</table>

?>

<tr><td><?php echo $n;?></td><td><?php echo $i;?></td>

<td><?php echo $n\*$i;?></td></tr>

<?php

Vòng lặp while

• Cú pháp:

while (biểu\_thức\_ĐK)

S;

• Thi hành lệnh (hay khối lệnh) S trong khi biểu\_thức\_ĐK còn nhận giá trị true. Ta thường sử dụng vòng lặp này trong các biểu thức mà chưa xác định số lần lặp lại của S.

• Ví dụ 1. Tính tổng các số nguyên từ 1 đến $n.

$tong = 0; $n=10; $i=0; while($i<$n) $tong +=$i; Echo "Tong = $tong ";

• Ví dụ 2: Viết lại script in ra bảng cửu chương $n bằng cấu trúc while

Vòng lặp while

Vòng lặp do…while

• Cú pháp: do{

S;

}while(biểu\_thức\_ĐK);

• Giống như vòng lặp while, tuy nhiên vòng lặp này sẽ kiểm tra biểu\_thức\_ĐK sau khi đã thực thi xong S. Như vậy, với vòng lặp này, S luôn luôn được thi hành.

Vòng lặp do…while

Vòng lặp foreach

• Cấu trúc này sử dụng để duyệt qua một mảng hay một đối tượng.

• Cú pháp: có 2 dạng

1. foreach (array\_expression as $value)

S;

2. foreach (array\_expression as $key => $value) S;

array\_expression: Mảng cần duyệt.

Foreach, as: từ khóa.

• Ở dạng 1: Mỗi lần lặp để duyệt qua một phần tử của mảng

array\_expression, giá trị của phần tử mảng này được gán cho biến

$value.

• Ở dạng 2. Mỗi lần lặp để duyệt qua một phần tử của mảng array\_expression, 2 giá trị của phần tử mảng này: chỉ số và giá trị của mảng được gán cho 2 biến là $key và $value.

Vòng lặp foreach

• Ví dụ

<?php $arr = array("a"=>5, "b"=>3); ?>

<table><tr><td>Key </td><td>Value </td></tr>

<?php

foreach($arr as $k=>$v)

{ ?>

<tr><td><?php echo $k;?>

</td><td><?php echo $v;?></td></tr>

<?php

} ?>

</table>

Một số lệnh cơ bản trong php

• Require, include, require\_once, include\_once

• Isset, unset, define, defined

• Echo, print, Print\_r, var\_dump

• exit, sleep

• Các hàm về cấu hình: Ini\_set, ini\_get, set\_path

• Các hàm toán học: floor, ceil, round,…

• Các hàm về thời gian: date, time, mktime, …

include, include\_once, require,

require\_once

• Các câu lệnh sẽ chèn các file được xác định vào ngay tại vị trí câu lệnh. Nó được sử dụng để cho người sử dụng có thể chia các trang web lớn thành nhiều trang web nhỏ hơn để dễ quản lý và có thể tái sử dụng code. Có 4 câu lệnh làm các chức năng này

• Các file được chèn vào sẽ được tìm trong các vị trí theo các thứ tự:

– Trong giá trị được xác định trong include\_path (file php.ini).

– Hàm set\_include\_path().

– Thư mục hiện hành.

• Include(path): Chèn path vào vị trí câu lệnh. Nếu không tìm thấy path, xuất hiện lỗi cảnh báo nhưng trang vẫn được tiếp tục xử lý

• Include\_once (path): giống include nhưng nó chỉ chèn path khi trước đó path chưa được chèn vào trang.

• Require: giống include nhưng sau khi gặp lỗi, chương trình sẽ dừng

lại

• Require\_once: giống requied nhưng có kiểm tra xem và path chỉ được chèn vào khi trước đó path chưa được chèn.

Ví dụ

• Ví dụ: tổ chức các tập tin như hình

Nội dung ini.php

<?php

$user= "admin"; define("AGE", 20);

?>

Nội dung: menu.html

<a href='http://stu.edu.vn'>STU</a> &nbsp;

<a href='https://www.facebook.com/'>facebook</a>

Nội dung footer.php

<img src='footer.jpg'>

Ví dụ (tt)

• Nội dung a.php Nội dung tương đương

Isset, unset, define, defined

• Isset, unset:

– Isset: kiểm tra sự tồn tại của một biến: trả về true/false.

Hàm này thường sử dụng khi

nhận dữ liệu từ client.

– Unset: hủy một biến đã có

• Define, defined:

– Define: định nghĩa một hằng số

– Defined: kiểm tra xem một hằng số đã được định nghĩa hay chưa.

$a = 10;

Echo $a;

If (isset($a)) echo $a;

Else echo "Chưa có biến a"; Unset($a);

If (isset($a)) echo $a;

Else echo "Chưa có biến a";

define("A", 10);

If (defined(A)) echo $a; If (defined(B)) echo B;

Else echo "Chưa có hằng số B";

Echo, print, Print\_r, var\_dump

• Đây là 4 hàm sử dụng xuất dữ liệu về client

• echo (string), print(string): Sử dụng để xuất một chuỗi về client.

$a =10; $b=2; echo "Tổng $a và $b là ". ($a+$b); Print("Tổng $a và $b là ". ($a+$b));

• Print\_r(object), var\_dump(object): sử dụng để xuất dữ liệu là một mảng hay một đối tượng. Những hàm này sử dụng để debug chương trình, in ra cấu trúc và dữ liệu cuả mảng, đối tượng

$a = array(2, "abc"); Print\_r($a); var\_dump($a);

Các hàm toán học

• Abs: Trả về trị tuyệt đối của 1 số

<?php echo abs(-4.2); // 4.2 (double/float) echo abs(5); // 5 (integer)

echo abs(-5); // 5 (integer)

?>

• Floor: Trả về phần số nguyên kế nhỏ hơn hoặc bằng số thực đưa vào.

<?php echo floor(4.3); // 4

echo floor(9.999); // 9

echo floor(-3.14); // -4

?>

• Ceil: giống như floor nhưng trả về số nguyên lớn hơn hoặc bằng (làm tròn lên).

<?php echo ceil(4.3); // 5

echo ceil(9.999); // 10 echo ceil(-3.14); // -3

?>

Các hàm toán học (tt)

• Round: float round ( float $val [, int $precision = 0]). Làm tròn một số

thực.

<?php

echo round(3.4); // 3 echo round(3.5); // 4 echo round(3.6); // 4 echo round(3.6, 0); // 4

echo round(1.95583, 2); // 1.96

echo round(1241757, -3); // 1242000

echo round(5.045, 2); // 5.05

echo round(5.055, 2); // 5.06

?>

• Pow: number pow ( number $base , number $exp ). Hàm số mũ, trả về

$base$exp.

var\_dump(pow(2, 8)); // int(256) echo pow(-1, 20); // 1

echo pow(0, 0); // 1

Các hàm toán học (tt)

• number\_format: Sử dụng để xuất chuỗi số

theo định dạng được yêu cầu về client.

• Cú pháp: number\_format ($number ,

$decimals = 0 , $dec\_point = "." ,

$thousands\_sep = ",")

– $number: Sớ cần format

– $decimals: Số ký tự thập phân được xuất ra

– $dec\_point: Ký hiệu phân cách phần nguyên và

thập phân.

– $thousands\_sep: Ký hiệu phân cách các nhóm số

hàng ngàn.

– Hàm trả về chuỗi số được format.

Ví dụ

<?php

$number = 1234.56;

$english\_format\_number = number\_format($number);

// english notation (default) 1,235

$nombre\_format\_francais = number\_format($number, 2, ',', ' ');

// French notation 1 234,56

$number = 1234.5678;

$english\_format\_number = number\_format($number, 2, '.', '');

// 1234.57

?>

Hàm xử lý về thời gian

• Để xử lý các hàm về date/time, ta cần thiết lập timezone cho vùng cần xử lý thời gian trước khi gọi các hàm date/time bằng dòng lệnh date\_default\_timezone\_se(timezone);

• Ví dụ:

date\_default\_timezone\_set('UTC'); date\_default\_timezone\_set('Asia/Ho\_Chi\_Minh');

• Lệnh này chỉ ảnh hưởng đến các lệnh sau nó, vì vậy nó thường phải để ở đầu file.

Hàm xử lý về thời gian

• Hàm Date: string date ( string $format [, int $timestamp = time() ] ).

– Trả về chuỗi thời gian theo format.

– Tham số thứ nhất: Các ký hiệu format (xem trong bảng sau)

– Tham số thứ 2: thời gian cần format. Mặc định là thời gian hiện tại

• Ví dụ:

<?php date\_default\_timezone\_set('Asia/Ho\_Chi\_Minh');

$d = date("d/m/Y h:i:s");

echo "Bây giờ là: ".$d;

?>

Hàm xử lý thời gian

Hàm xử lý thời gian (tt)

• Time: int time ( void ): trả về 1 số nguyên là số giây tính từ 1/1/1970 00:00:00 tới thời điểm hiện tại. Ví dụ: $t = time();

• mkTime: int mktime ([ int $hour = date("H") [, int $minute = date("i") [, int $second = date("s") [, int $month = date("n") [, int $day = date("j") [, int $year = date("Y") ]]]]]] ); Trả về giá trị timestamp cho các giá trị ngày được đưa vào

Ví dụ:

echo "July 1, 2000 is on a " . date("l", mktime(0, 0, 0, 7, 1, 2000));

• Strtotime(): int strtotime ( string $time [, int $now = time() ] ): Trả về giá trị

timestamp nếu thành công, ngược lại trả về false.

Ví dụ: <?php

echo strtotime("now"), "\n"; echo strtotime("10 September 2000"), "\n";

$str = 'Not Good';

if (($timestamp = strtotime($str)) === false) {

echo "The string ($str) is not time string";

} else {

echo "$str == " . date('l dS \o\f F Y h:i:s A', $timestamp);

}

?>

Xây dựng function trong php

• Cú pháp tạo hàm

• Truyền tham số vào trong hàm

• Giá trị mặc định của tham số

• Tầm vực của biến

Cú pháp tạo function

• Lý do tạo hàm?

• Cú pháp khi tạo hàm.

function tênhàm([thamsố 1, tham số 2, ...])

{

//Nội dung phần định nghĩa của hàm

}

• Chú giải:

– function: từ khóa

– Tênhàm: tên do người sử dụng đặt ra phù hợp với qui tắc tạo định danh trong

php

– Tham số: Các tham số truyền vào hàm.

– {}: khối bao nội dung hàm

• Gọi hàm: Hàm được xây dựng có thể đặt bất cứ vị trí nào trong trang và

chỉ được thực thi khi ta gọi nó trong chương trình.

• Thông thường, ta định nghĩa các hàm dùng chung và đặt trong một file riêng. Khi cần sử dụng, ta load file này vào trang hiện tại bằng nhóm lệnh include.

Ví dụ

• Ví dụ tạo hàm: Function F($a, $b)

{ return $a\*$b;

}

• Sử dụng hàm

$s = F(3, 6);

//gọi hàm F và truyền vào 2 tham số $a=3, $b=6 echo $s;

Truyền tham số vào trong hàm

• Truyền tham trị: Các giá trị tham số truyền vào sẽ không thay

đổi khi hàm thực thi xong.

• Truyền tham chiếu: Các tham số truyền vào sẽ giữ nguyên giá trị thay đổi khi ra khỏi hàm.

• Để xác định tham số nào cần truyền tham chiếu, ta đặt dấu & trước tham số hình thức.

• Ví dụ:

function F($a, &$b) //$b truyền tham chiếu

{

$a = $b\*2; $b = $b\*2; return $a+$b;

}

$x =3; $y=4; $s =F($x, $y);

Echo "s= $s – x= $x – y= $y ";//s=16 – x=3 – y=8

Giá trị mặc định của tham số

• Các hàm trong php có thể nhận các tham số mặc định.

• Các tham số mặc định khi định nghĩa hàm, chúng được gán

giá trị. $thamsố = giátrị.

• Các hàm có tham số mặc định nên chuyển các tham số mặc định sang phải.

• Khi gọi hàm, nếu không đưa vào đầy đủ các tham số, các giá trị mặc

định sẽ được sử dụng.

• Ví dụ:

Function F($x, $y=1)

{return $x/$y;};//tham số $y có giá trị mặc định là 1

$s = F(8, 4); //$s = 2;

$s2 = F(8); //$s2 = 8 vì F(8) <-> F(8, 1)

Tầm vực của biến

• Biến định nghĩa trong hàm là biến cục bộ, sẽ bị hủy khi hàm kết thúc

• Ta muốn sử dụng biến trong hàm có giá trị toàn cục, hoặc sử dụng các biến ngoài hàm, ta đặt từ khóa global trước tên biến.

• Ví dụ:

$a =1; $b=2; function F()

{ $a=6; global $b; $b=7; global $c; $c=8;} F();

echo "a= $a , b= $b, c=$c"; //a=1 ,b=7, c=8

Xử lý lỗi trên php

• Trong quá trình phát triển và triễn khai ứng dụng web, với mỗi giai đoạn, ta cần biết cách cấu hình để debug lỗi cho phù hợp. Có nhiều cách đề làm việc này tùy thuộc vào giới hạn quyền và yêu cầu cấu hình của ta trên máy server. Các yêu cầu khi xử lý lỗi:

– Hiện/tắt lỗi, cảnh báo,… trong file php.

– Hiển thị thông tin báo lỗi riêng của người lập trình

– Chuyển trang khi xử lý các lỗi với máy chủ.

• Có nhiều cách để xử lý các lỗi theo các yêu cầu này.

– Cấu hình trong file php.ini

– Cấu hình trong file htdoc.conf, file .htaccess

– Xử lý trong code

Xử lý lỗi trong php

• Cấu hình trong file php.ini:

– Để cấu hình, cần có quyền thay đổi nội dung file này.

– Hiện hay tắt lỗi: dòng lệnh display\_errors = On/Off

• Cấu hình file htdoc.conf và .htaccess

– Các lỗi này thường liên quan đến việc tiếp nhận và

xử lý file trên phần mềm web server như:

• Người sử dụng nhập vào url không hợp lệ,…

• Máy chủ gặp sự cố

• …

Xử lý lỗi trong file php

• Sử dụng code php để thay đổi các thiết lập khi không được phép can thiệp vào file php.ini

• Sử dụng hàm error\_reporting (level) hoặc

ini\_set('error\_reporting', level).

• Sử dụng toán tử @ kết hợp hàm die.

• Viết các hàm xử lý lỗi riêng

error\_reporting và ini\_set

• error\_reporting và ini\_set: thiết lập cách hiển thị lỗi, cảnh báo trên trang web khi ta không có quyền can thiệp file php.ini.

• error\_reporting (level)

• ini\_set('error\_reporting', level).

– Các level là các hằng số đã được định nghĩa sẵn. Mỗi hằng số này có 1 giá trị là một số nguyên.

– Ta có thể kết hợp nhiều giá trị này lại.

Ví dụ: ini\_set('display\_errors', 0);

//giống error\_reporting(0);

error\_reporting và ini\_set(tt)

Value Constant Description

2 E\_WARNING Non-fatal run-time errors. Execution of the script is not halted

8 E\_NOTICE Run-time notices. The script found something that might be an error, but could also happen when running a script normally

256 E\_USER\_ERROR Fatal user-generated error. This is like an E\_ERROR set by the programmer using the PHP function trigger\_error()

512 E\_USER\_WARNING Non-fatal user-generated warning. This is like an E\_WARNING

set by the programmer using the PHP function trigger\_error()

1024 E\_USER\_NOTICE User-generated notice. This is like an E\_NOTICE set by the programmer using the PHP function trigger\_error()

4096 E\_RECOVERABLE\_ERROR Catchable fatal error. This is like an E\_ERROR but can be caught by a user defined handle (see also set\_error\_handler())

8191 E\_ALL All errors and warnings (E\_STRICT became a part of E\_ALL in PHP 5.4)

Toán tử @ và hàm die

• Sử dụng hàm die và toán tử @

– Toán tử @: đặt trước biểu thức sẽ ngăn hiển thị lỗi.

– Hàm die(string): sẽ xuất chuỗi string và dừng chương trình. Hàm này có thể sử dụng kết hợp với toán tử or.

– Ví dụ:

$file= @fopen("welcome.txt","r") or die(" File not found ");

//Hoặc

if(!file\_exists("welcome.txt")) { die("File not found");

} else {

$file=fopen("welcome.txt","r");

$conn = @mysql\_connect("localhost", "root", "") or die("lỗi

connect");

Tạo hàm xử lý lỗi riêng

• Ta có thể viết một hàm php để xử lý lỗi riêng theo ý mình.

– Tạo 1 hàm xử lý lỗi

– Sử dụng hàm set\_error\_handler(); để thiết lập hàm vừa định nghĩa. Hàm này sẽ được khởi chạy khi trang bị lỗi.

• Ví dụ:

function customError($errno, $errstr) { echo "<b>Error:</b> [$errno] $errstr<br>"; echo "Ending Script";

die();

}

set\_error\_handler("customError");

//mã php tiếp tục tại đây

Tổng kết

• Mô hình xử lý một trang php?

• Biến, hằng, biểu thức và các cấu trúc điều khiển.

• Xây dựng các hàm trong php

• Một số hàm cơ bản hay sử dụng trong php.

• Xử lý lỗi trong php

Array - Mảng

• Giới thiệu mảng php

• Tạo và truy xuất dữ liệu

• Mảng nhiều chiều

• Một số hàm hay sử dụng với mảng

• Mảng có sẵn trong php

• Gửi nhận dữ liệu client- server

Giới thiệu mảng php

• Mảng là một thành phần rất quan trọng và hay được sử dụng và rất dễ sử dụng trong php.

• Mảng là biến dữ liệu lưu trữ tập hợp dữ liệu gồm nhiều phần tử. Mỗi phần tử chứa các giá trị thuộc bất kỳ kiểu dữ dữ liệu nào, một mảng khác, đối tượng...

• Trong một mảng, mỗi phần tử có thể chứa các dữ liệu có kiểu khác nhau. Mỗi phần tử trong mảng có một chỉ mục (index, key) và giá trị (value) của phần tử đó trong mảng.

$arr[key] = value;

• Các chỉ mục key có thể là các số nguyên không âm hoặc chuỗi.

• Sử dụng các hàm print\_r và var\_dump để debug xem cấu

trúc, nội dung của mảng.

Tạo mảng

• Tạo mảng, sử dụng hàm array

– Cách 1: mảng rỗng: $arr =array();

– Cách 2: mảng không xác định key: key là các số nguyên bắt đầu từ 0

$arr = array(2, "a", 10); //$arr[0]-> 2, $arr[1]-> "a" và $arr[2] -> 10 Print\_r($arr);

Array (

[0] => 2

[1] => a

[2] => 10

)

– Cách 3: mảng với key xác định:

$arr =array("a"=>10, "b"=>6);

//$arr["a"] ->10, $arr["b"] ->6

– Để truy xuất phần tử có chỉ số là key, ta sử dụng cặp dấu ngoặc vuông[]: Ví dụ

$x = $arr["a"];

– Thêm phần tử cho mảng: Không cần phải khai báo cấp phát bộ nhớ.

Ví dụ

<?php

$arr = array(); //mảng rỗng

$arr[]= 10; //$arr[0] = 10;

$arr["a"] = 20; print\_r($arr);

$x ="b"; $arr[$x] = 15; var\_dump($arr);

?>

Kiểm tra và xóa phần tử mảng

• Hàm isset: Kiểm tra phần tử có thuộc

tính key có trong mảng hay không

ví dụ: if (isset($arr["c"])) echo "có phần tử c";

• Hàm unset: Hủy một phần tử: ví dụ

unset($arr["a"]);

• Hàm is\_array ($var ): Kiểm tra $var có phải là một mảng hay không. (trả về true nếu $var là mảng, ngược lại trả về false)

Duyệt mảng

• Sử dụng vòng lặp for: Sử dụng cho mảng chứa các phần tử có chỉ số là số nguyên

và theo thứ tự và bắt đầu từ chỉ số xác định.

for($i=0; $i<Count($arr); $i++)

{

//thao tác với $arr[$i];

}

• Vòng lặp foreach: Dùng để duyệt qua tất cả các loại mảng khác nhau. Mỗi lần lặp,

vòng lặp tự động di chuyển qua một phần tử của mảng.

Cú pháp: Có 2 dạng: Chỉ duyệt lấy value hoặc lấy cả các cặp key/value

Dạng 1: Foreach($arr as $value)

{

//xử lý các giá trị $value

}

Hoặc dạng 2: foreach($arr as $key=> $value)

{

//xử lý các giá trị $key - $value.

}

Trong đó $arr là biến mảng cần được duyệt qua

$key, $value: Là các biến để lấy giá trị key và value của phần tử mảng đang duyệt qua.

Ví dụ

<?php

$a = array(1, 3, 5);//$a[0]=1, $a[1]=3, $a[2]=5

$b = array("b1"=>"v1", "b2"=>3); for($i=0; $i<3; $i++){

echo "<br> $i - {$a[$i]} ";

}

echo "<hr>";

foreach($a as $k=>$v){

echo "<br>$k - $v ";

}

echo "<hr>";

foreach($b as $key=>$value){

echo "<br> $key - $value ";

}

echo "<hr>"; foreach($b as $v){

echo "<br> $v ";

}

?>

Mảng đa chiều

• Mảng đa chiều: Là mảng mà giá trị của phần tử mảng lại là một mảng khác

• Ví dụ

$arr1 = array("a"=>10, "b"=>20");

$arr2 = array("a"=>15, "b"=>$arr1);

//$arr2 là mảng 2 chiều

$arr3=array("x"=> array(5, 7), "y"=> array(4, 6) );

• Truy xuất:

$x = arr3["x"][1]; //$x = ?

$y = $arr3["y"]; //$y=?

$z = $y[0]; //$z=?

Một số hàm hay sử dụng với mảng

• Count: Đếm số phần tử trong mảng

$a = array(1, 4, 5);

Echo Count($a);//3

• In\_array(): bool in\_array ($v , $arr)kiểm tra một phần tử $v có thuộc mảng

$arr hay không. Trả về true/false nếu có hay không.

<?php

$ext ="jpg"; $arrExt = array("jpg", "png");

if (in\_array($ext, $arrExt)) echo "File hình hợp lệ";

else echo "Không hỗ trợ file hình này";

?>

• array\_rand($a, [$n=1]): Lấy ra ngẫu nhiên $n phần tử trong mảng $a; Trả về mảng các key.

<?php

$input = array("Neo", "Morpheus", "Trinity", "Cypher", "Tank");

$rand\_keys = array\_rand($input, 2); echo $input[$rand\_keys[0]] . "\n"; echo $input[$rand\_keys[1]] . "\n";

Sắp xếp mảng

• Sort: cú pháp bool sort ( array &$array [, int $sort\_flags = SORT\_REGULAR ] ): Hàm này sắp xếp một mảng. Các phần tử sẽ được sắp xếp từ thấp đến cao.

– $array: Mảng cần sắp xếp.

– $sort\_flags: xác định cách so sánh khi sắp xếp và có các giá trị:

• SORT\_REGULAR – So sánh bình thường (không đổi kiểu dữ liệu)

• SORT\_NUMERIC – So sánh theo dạng số

• SORT\_STRING – So sánh theo dạng chuỗi.

– Hàm trả về true/false nếu thành công hay lỗi.

Ví dụ:

<?php

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

foreach ($fruits as $key => $val) {

echo "fruits[" . $key . "] = " . $val . "\n";

}

?>

Chuyển đổi Array -String

• Explode(): Tách chuỗi thành mảng. array explode (string $delimiter , string $string )

– $delimiter: chuỗi phân cách.

– $string: chuỗi cần tách

– Kết quả trả về một mảng sau khi tách chuỗi.

– Ví dụ:

<?php

$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";

$pieces = explode(" ", $pizza); echo $pieces[0]; // piece1 echo $pieces[1]; // piece2

?>

• Implode(): Ghép các phần tử của mảng thành chuỗi. String implode ( string $glue , array $pieces )

– $glue: Chuỗi kết nối.

– $pieces: Mảng cần ghép các phần tử

– Kết quả: trả về chuỗi đã gắn kết các phần tử $glue và giữa các phần tử mảng.

– Ví dụ:

<?php

$array = array("lastname", "email", "phone");

$comma\_separated = implode("-", $array);

echo $comma\_separated; // lastname-email-phone

?>

Biến mảng siêu toàn cục

• Là các biến mảng có sẵn trong php

• Có thể truy xuất từ bất cứ vị trí nào của trang php mà không cần khai báo.

• Nếu cấu hình php.ini là register\_globals = On, các chỉ số của biến mảng này sẽ trở thành các biến toàn cục. Ví dụ: $\_POST["foo"] sẽ có thể truy xuất là $foo. Ta nên hạn chế sử dụng dạng này

Biến mảng siêu toàn cục (tt)

• $\_REQUEST, $\_GET, $\_POST: chứa dữ liệu các biến gửi từ

form hoặc URL client

• $\_FILES: chứa thông tin các tập tin gửi từ client.

• $\_SERVERS: $\_SERVER: Mảng này chứa thông tin về ngữ cảnh mà đoạn mã đang chạy, như là tên server, tên file đang chạy... Dữ liệu mảng này được tạo bởi web server.

• $\_SESSION & $\_COOKIE: 2 mảng này chứa các thông tin về

phiên làm việc và cookie của client.

• $\_ENV: Chứa các biến môi trường mà engine PHP đang chạy

• Để xem nội dung các mảng này, nên sử dụng hàm print\_r

hoặc var\_dump. Ví dụ: print\_r($\_SERVER).

Gửi nhận dữ liệu client- server

• Gửi nhận dữ liệu qua URL

• Gửi nhận dự liệu qua form với phương thức gửi GET

• Gửi nhận dữ liệu qua Form với phương thức gửi là POST

• Gửi và nhận và xử lý tập tin

GỬI NHẬN DỮ LIỆU QUA URL

Dữ liệu máy chủ nhận được, có thể truy xuất qua mảng $\_GET hoặc $\_REQUEST

GỬI NHẬN DỮ LIỆU QUA URL(tt)

Xét URL sau: http://abc.com/cong.php?a=1&b=2 Trang được gọi xử lý?

Trang xử lý là trang cong.php của website abc.com

GỬI NHẬN DỮ LIỆU QUA FORM GET

GỬI NHẬN DỮ LIỆU QUA FORM POST

Nhận dữ liệu từ form

• Nhận dữ liệu từ các thành phần: textbox, hidden, password, textarea.

• Nhận dữ liệu từ radio button

• Nhận dữ liệu từ checkbox

• Dropdown và listbox

• Submit button

• Nhận dữ liệu từ upload file

• Button

Nhận dữ liệu từ textbox, password, hidden field

Mã html Mã php nhận kết quả từ form

<input type="text" name="id" value="user01" /> $user = $\_POST["id"];//$user

="user01"

<input type="password" name="pass"

value="123456"/> $p = $\_REQUEST["pass"];

//$p = "123456"

<input type="hidden" name="action" value="login" /> $action = $\_POST["action"];

//action="login"

Nhận dữ liệu từ nhóm radio button

Mã HTML PHP

<input type="radio" name="align" value="left"

/> Trái

<input type="radio" name="align" value="center" /> Giữa

<input type="radio" name="align" value="right" /> Phải if (isset($\_POST["align"]))

{

echo "Chọn:" . \_POST["align"];

}

else

{

echo "Chưa chọn";

}

Nhận dữ liệu từ nhóm checkbox button

Mã HTML PHP

Chọn các màu bạn thích<br />

<input type="checkbox" name="color[]"

value="Red" />Đỏ

<input type="checkbox" name="color[]" value="Blue" />Xanh

<input type="checkbox" name="color[]" value="Yellow" />Vàng if (isset($\_POST["color"]))

{

$arr = $\_POST["color"]; foreach($arr as $item)

echo "Chọn:" . $item;

}

else

{

echo "Bạn chưa chọn";

}

Chọn dữ liệu từ dropdown list

Mã HTML PHP

<select name="vitri[]" multiple="multiple">

<option value="0">Chọn vị trí </option>

<option value="1">Trái</option>

<option value="2">Giữa</option>

<option value="3">Phải</option>

</select>

if (isset($\_POST["vitri"]))

{

$arr = $\_POST["vitri"]; foreach($arr as $item) echo "Chọn:" . $item;

}

else

{

echo "Bạn chưa chọn";

}

Chọn dữ liệu từ dropdown list

Mã HTML PHP

<select name="vitri">

<option value="0">Chọn vị trí </option>

<option value="1">Trái</option>

<option value="2">Giữa</option>

<option value="3">Phải</option>

</select>

$\_POST["vitri"]

Gửi tập tin

• Form phải có thuôc tính: method="post", enctype="multipart/form-data"

<form method="post" enctype="multipart/form-data">

• Sử dụng đối tượng file

<input type="file" name= "img" />

Nhận tập tin

• Thông tin tập tin được lưu trữ là mảng trong mảng $\_FILES

• Nếu không chọn file, mảng không có dữ liệu

• Nội dung mảng phần tử: $\_FILES["img"] Array (

[name] => Desert.jpg

[type] => image/jpeg

[tmp\_name] => C:\wamp\tmp\php4092.tmp [error] => 0

[size] => 845941

)

Nhận tập tin (tt)

• Thành phần mảng $\_FILES

– Name: Tên tập tin gửi từ client

– Type: loại tập tin

– Tmp\_name: Tập tin, trước khi ta xử lý, sẽ được lưu trữ tạm ở đường dẫn này trên máy chủ.

– Error: là một số nguyên qui định lỗi xảy ra.

• 0: không có lỗi

• 1: Lỗi, dung lượng file quá lớn so với config trong php.ini

• 2: file quá lớn so với qui định trong form

• 3: Quá trình tải bị lỗi, file mới chỉ tải một phần.

• 4: Không có file

– Size: dung lượng file vừa upload lên tính bằng bytes.

• Để move file từ tmp\_name tới vị trí xác định, ta sử dụng

hàm: move\_uploaded\_file

Mảng $\_SERVER

• Chứa các biến được cung cấp bởi máy chủ. Ví dụ: <?php print\_r($\_SERVER); ?>

• Các thông tin máy chủ cung cấp:?

– Thông tin máy khách đang truy cập: hệ điều hành, trình duyệt, phiên bản,…

– Tên máy chủ, hệ điều hành, phần mềm web server trên máy chủ,..

– Địa chỉ document\_root, đường dẫn script hiện tại, …

– Thông tin về queryString, REQUEST\_URI

– …

Mảng $\_SERVER (tt)

• Ví dụ: in ra một số thông tin hay sử dụng từ mảng server.

<?php

$host = $\_SERVER["HTTP\_HOST"]; echo " HTTP\_HOST: ".$host."<br>";

$self = $\_SERVER["PHP\_SELF"]; echo " PHP\_SELF: ".$self."<br>";

$query = $\_SERVER["QUERY\_STRING"]; echo "QUERY\_STRING: $query

<br>";

$root = $\_SERVER["DOCUMENT\_ROOT"]; echo " DOCUMENT\_ROOT:

Mảng $\_SESSION

• Session là khoảng thời gian người sử dụng giao tiếp với 1 ứng dụng. Session bắt đầu khi người sử dụng truy cập vào ứng dụng lần đầu tiên, và kết thúc khi người sử dụng thoát khỏi ứng dụng.

• Mỗi session sẽ có một định danh (session ID), 1 session khác nhau sẽ có 2 ID khác nhau và nội dung được lưu trong thư mục thiết lập trong file php.ini (tham số session.save\_path).

• Trong ứng dụng web, website sẽ quyết định khi nào session bắt đầu và kết thúc.

• Trong 1 session, website có thể lưu trữ một số thông tin như đánh dấu bạn

đã login hay chưa, những bài viết nào bạn đã đọc qua, v.v...

• Để lưu trữ thông tin bằng session, ta sử dụng mảng S\_SESSION: là mảng lưu trữ các thông tin toàn cục với tất cả các trang web trong của một phiên làm việc của một người sử dụng.

• Mảng này được sử dụng để xây dựng các chức năng: quản lý đăng nhập, xây dựng

gio hàng cho các website bán hàng, đếm số người online trên website,…

• Sử dụng mảng $\_SESSION như những mảng thông thường.

• Trước khi sử dụng, phải đảm bảo mảng $\_SESSION đã có sẵn sử dụng hay chưa. Nếu chưa ta có thể dùng hàm session\_start() để khởi động. Đoạn script này nên đặt ngay dòng đầu tiên của trang dùng $\_SESSION.

Mảng $\_SESSION

• Ví dụ:

tạo 2 file a.php và b.php và chạy thử kết quả. 2 file này sử dụng chung một biến $\_SESSION["n"].

Mảng $\_COOKIE

• Cookie là 1 đoạn dữ liệu được ghi vào đĩa cứng hoặc bộ nhớ của máy người sử dụng. Nó được trình duyệt gởi ngược lên lại server mỗi khi browser tải 1 trang web từ server.

• Những thông tin được lưu trữ trong cookie hoàn toàn phụ thuộc vào website trên server. Mỗi website có thể lưu trữ những thông tin khác nhau trong cookie, ví dụ thời điểm lần cuối ta ghé thăm website, đánh dấu ta đã login hay chưa, v.v...

• Cookie được tạo ra bởi website và gởi tới browser, do vậy 2 website khác nhau (cho dù cùng host trên 1 server) sẽ có 2 cookie khác nhau gởi tới browser. Ngoài ra, mỗi browser quản lý và lưu trữ cookie theo cách riêng của mình, cho nên 2 browser cùng truy cập vào 1 website sẽ nhận được 2 cookie khác nhau.

• Giống session, cookie cũng sử dụng để quản lý các phiên làm việc

giữa người sử dụng và hệ thống.

Mảng COOKIE (tt)

• Tạo cookie: Setcookie("tên\_cookie","giá\_trị", thời\_gian\_sống=0);

– Tên cookie: tên biến cookie cần lưu trữ.

– Giá\_trị: giá trị của cookie.

– thời\_gian\_sống: thời gian cookie còn tồn tại trên trình duyệt tính bằng số giây.

• Truy xuất cookie: Thông qua tên\_cookie và mảng $\_COOKIE.

– $v = $\_COOKIE["tên\_cookie"];

• Hủy cookie: sử dụng hàm setcookie

– Setcookie("tên\_cookie");//Hủy cookie

– Setcookie("tên\_cookie","giá\_trị",thời\_gian\_sống=số\_trong\_quá\_khứ);

– Ví dụ: Setcookie("name","abc", time()-100);

So sánh session-cookie

• Cookie và Session đều có chung mục đích là lưu giữ data để truyền từ 1 trang web sang 1 trang web khác (trên cùng website). Nhưng phương thức lưu trữ và quản lý data của Cookie và Session có phần khác nhau.

• Cookie sẽ được lưu trữ tại browser, do browser quản lý và browser sẽ tự động truyền cookie ngược lên server mỗi khi truy cập vào 1 trang web trên server.

• Dữ liệu lưu trữ trong Session sẽ được ứng dụng quản lý, trong ngữ cảnh web, ứng dụng ở đây sẽ là website và webserver. Browser chỉ truyền ID của session lên server mỗi khi truy cập vào website trên server.

Ví dụ

Sử dụng SESSION hay COOKIE

• Thường session hay được sử dụng hơn vì bảo mật hoặc thiết lập trên trình duyệt của người sử dụng.

• Trong một số trường hợp browser đã được thiết lập để không chấp nhận cookie, lúc đó session vẫn sử dụng được bằng cách truyền session ID giữa các trang web qua URL.

• Nếu các dữ liệu được lưu trên server, ta chỉ cần truyền session\_id giữa client-server là có thể quản lý được các phiên làm việc giữa client-server.

• Bảo mật: càng ít thông tin được truyền tải qua lại giữa browser và client càng tốt, và càng ít thông tin được lưu trữ tại client càng tốt.

Tổng kết

• Mảng: Cách tạo, truy xuất, duyệt, thêm, xóa

phần tử

• Các hàm trên mảng: sort, Count, is\_array, in\_array,…

• Các hàm có sẵn trên php

– Gửi nhận dữ liệu client-server: $\_GET, $\_POST,

$\_REQUEST, $\_FILES

– $\_SERVER

– Quản lý phiên làm việc: $\_SESSION, $\_COOKIE

Chapter 4

Làm việc với chuỗi

Nội dung

• Giới thiệu

• Những hàm xử lý về chuỗi hay sử dụng

• Biểu thức qui tắc

Giới thiệu

• Chuỗi là một thành phần rất quan trọng và

được sử dụng ở mọi nơi trong các trang web.

• Có rất nhiều hàm được xây dựng để người sử dụng thuận lợi thao tác với chuỗi.

• Luôn cần kiểm tra các giá trị dạng chuỗi hợp lệ trước khi lưu trữ trong csdl: email, mật khẩu,

…

Những hàm xử lý chuỗi

• strlen — Trả về chiều dài của chuỗi

– Int strlen($string);

– Ví dụ: $s='abc'; echo strlen($s);

• rtrim — cắt các khoảng trắng (hoặc các ký tự khác) từ cuối chuỗi

– string rtrim ( string $str [, string $character\_mask ] )

– $str: chuỗi cần cắt

– $character\_mask: chuỗi ký tự cần bỏ.

– Hàm trả về chuỗi đã cắt bỏ các ký tự yêu cầu.

• ltrim — Giống rtrim, nhưng loại bỏ các ký tự bên trái

chuỗi.

• trim — Loại bỏ các ký tự yêu cầu đầu và cuối chuỗi.

Ví dụ

<?php

$a=" function rtrim ";

$b=" function rtrim\*\*\*";

$s1 = rtrim($a);

$s2 = rtrim($b, "\*");

$s3 = trim($a);

echo $a ."(". strlen($a).")<br>"; //function rtrim (17)

echo $s1 ."(". strlen($s1).") <br>"; // function rtrim(15) echo $s2 ."(". strlen($s2).") <br>"; // function rtrim(15) echo $s3 ."(". strlen($s3).")"; //function rtrim(14)

Những hàm xử lý chuỗi

• addslashes — Thêm ký tự backslash (\) vào trước ký tự đặc biệt (nháy đơn, nháy kép, ký tự \) của chuỗi.

– Hàm này hay được sử dụng khi viết truy vấn thêm vào trong csdl.

– Cú pháp: addslashes(string): string: chuỗi ban đầu. Kết quả trả về chuỗi sau khi đã thêm ký tự (\).

– Chú ý: Các dữ liệu gửi từ client: $\_GET, $\_POST,

$\_COOKIE: nếu thuộc tính get\_magic\_quotes\_gpc trong file php.ini được thiết lập là 1 (mặc định), các giá trị trong các mảng này đã được xử lý qua hàm addslashes. Do vậy, cần kiểm tra bằng hàm get\_magic\_quotes\_gpc() trước khi xử lý chuỗi để tránh thêm các ký hiệu (\) một lần nữa (double escaping).

Những hàm xử lý chuỗi (tt)

• Stripslashes($str) có tác dụng ngược lại với addslashed, hàm này sẽ loại bỏ các ký tự \ trong chuỗi ký tự, thường được sử dụng để xử lý chuỗi trước khi hiển thị thông tin lên trình duyệt.

<?php

$str = "Is your name O\'reilly?";

// Outputs: Is your name O'reilly? echo stripslashes($str);

?>

Ví dụ

<?php

$str = "Who's Peter Griffin?";

echo $str . " This is not safe in a database query.<br>"; echo addslashes($str) . " This is safe in a database query."; if (get\_magic\_quotes\_gpc()) {

$lastname = stripslashes($\_POST['lastname']);

}

else {

$lastname = $\_POST['lastname'];

}

?>

Nhóm hàm về String và Array

• explode — Phân tích một chuỗi và chuyển sang mảng

• Explode(): Tách chuỗi thành mảng.

array explode (string $delimiter , string $string )

– $delimiter: chuỗi phân cách.

– $string: chuỗi cần tách

– Kết quả trả về một mảng sau khi tách chuỗi.

– Ví dụ:

<?php

$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";

$pieces = explode(" ", $pizza); echo $pieces[0]; // piece1 echo $pieces[1]; // piece2

?>

Nhóm hàm về String and Array (tt)

• Implode(): Ghép các phần tử của mảng thành chuỗi. String implode ( string $glue , array $pieces )

– $glue: Chuỗi kết nối.

– $pieces: Mảng cần ghép các phần tử

– Kết quả: trả về chuỗi đã gắn kết các phần tử $glue và giữa các

phần tử mảng.

– Ví dụ:

<?php

$array = array('lastname', 'email', 'phone');

$comma\_separated = implode("-", $array);

echo $comma\_separated; // lastname-email-phone

?>

Những hàm mã hóa chuỗi

• md5 — mã hóa chuỗi theo giải thuật MD5. Đây là giả thuật mã hóa một chiều, thường sử dụng cho việc mã hóa mật khẩu trong database. Một chuỗi có chiều dài

<=32 khi sử dụng hàm MD5 mã hóa, sẽ trả về kết quả chuỗi có chiều dài 32 ký tự dạng số hexa.

– Md5($string)

– $string: Chuỗi cần mã hóa.

– Kết quả trả về: chuỗi đã mã hóa.

• Ví dụ:

$s = md5("123456");

echo $s; //e10adc3949ba59abbe56e057f20f883e

Những hàm mã hóa chuỗi(tt)

• sha1 — Giống như MD5, nhưng kết quả trả về là chuỗi dạng số hexa có độ dài là 40.

• Sha1 cũng thường được sử dụng để mã hóa mật khẩu.

• Ta cũng có thể kết hợp 2 hàm này để ra một kết quả mới.

• Ví dụ:

<?php $str = "Hello"; echo "The string: ".$str."<br>";

$s1 = sha1($str);

$s2 = md5(sha1($str)); echo "$s1 <br> $s2 ($n2)";

?>

Hàm format

• Hàm number\_format được sử dụng để hiển thị các dữ liệu dạng số theo nhiều format khác nhau: tiền tệ, số theo định dạng các quốc gia.

• number\_format — Format một số cùng nhóm các

số ở vị trí hàng ngàn.

– number\_format ($number , $decimals = 0 ,

$dec\_point = "." , $thousands\_sep = "," )

– $number: số được format

– $decimals: số ký số thập phân được in

– $dec\_point: Ký tự phân cách phần thập phân và nguyên

– Ký tự phân cách cho các nhóm số hàng ngàn.

Ví dụ

<?php

$number = 1234.56;

$number1 = number\_format($number);

// english notation (default) echo $number1; // 1,235

$number2 = number\_format($number, 2, ',', ' ');

// French notation

echo $number2; // 1 234,56

$number = 1234.5678;

$number3 = number\_format($number, 2, '.', '');

echo $number3; //1234.57

?>

Các hàm liên quan đến HTML

• htmlentities (string)— chuyển tất cả các ký tự liên quan đến html (>, <) sang dạng thực thể. Hàm này thường sử dụng cho việc xử lý dữ liệu liên quan đến html khi nhập vào database

<?php $str = "A 'quote' is <b>bold</b>";

// Outputs: A 'quote' is &lt;b&gt;bold&lt;/b&gt; echo htmlentities($str);?>

• html\_entity\_decode — Ngược lại với htmlentities, hàm sẽ chuyển đổi tất cả các thực thể HTML sang những kí tự có thể dùng được của chúng.

Các hàm liên quan đến HTML

• strip\_tags($string, $allow\_tags): Loại bỏ các thẻ HTML hoặc PHP ra khỏi chuỗi, Thường được sử dụng để xử lý dữ liệu do người dùng nhập trước khi lưu trữ database, hiển thị văn bản dạng trích dẫn.

• $allow\_tags: Các thẻ cho phép giữ lại Ví dụ: Loại bỏ các thẻ HTML ra khỏi chuỗi $str, cho phép giữ lại các tag trong $allow\_tags.

• NL2br($str) — Thay thế ký tự xuống dòng (\n) bằng thẻ xuống dòng (<br>) trong chuỗi

$str. Hàm này đường dùng để hiển thị nội dung của file văn bản trên trình duyệt web.

Ví dụ

<?php

$text = '<p>Test paragraph.</p><!-- Comment --

> <a href="#fragment">Other text</a>'; echo strip\_tags($text);

// Allow <p> and <a>

echo strip\_tags($text, '<p><a>');

$str1="12

3

4";

$str2 = nl2br($str1); //Thay 2 ký hiệu xuống dòng bằng 2 tag br

echo $str1; echo "<hr>"; echo $str2;

?>

Những hàm tìm kiếm, thay thế

• strpos — Tìm vị trí đầu tiên của chuỗi con trong một chuỗi.

– strpos ( $string , $substring )

– $string: chuỗi chính

– Substring: chuỗi cần tìm

– Hàm trả về vị trí đầu tiên của chuỗi $substring xuất hiện

trong $string. Nếu không tìm thấy, kết quả trả về là false.

• strstr — tìm và trả về chuỗi con trong chuỗi

– string strstr ( $string , $substring )

– $string: chuỗi chính

– $substring: chuỗi cần tìm

– Hàm trả về chuỗi con bắt đầu từ vị trí tìm được $substring trong $string cho tới cuối chuỗi $string

Những hàm tìm kiếm, thay thế (tt)

• str\_replace — Thay thế tất cả các lần xuất hiện của các

chuỗi tìm kiếm với chuỗi cần thay thế.

– str\_replace ($search , $replace , $subject, &$count )

– $search: chuỗi cần tìm để thay thế

– $ replace : Chuỗi thay thế

– $subject: Chuỗi chính được tìm kiếm và thay thế.

– $count: Số lần được thay thế.

– Hàm trả về một chuỗi sau khi thay thế

Ví dụ:

<?php

echo str\_replace("world","Peter","Hello world!");

?>

Những hàm tìm kiếm, thay thế (tt)

• substr — Hàm lấy một phần của chuỗi.

– substr($string, $start, $length)

– Trả về chuỗi con trong $string tính từ vị trí $start và lấy

$lenght ký tự.

– Chú ý: trong chuỗi, ký tự đầu tiên bắt đầu tính từ vị trí 0.

• str\_shuffle — Đảo ngẫu nhiên các ký tự trong chuỗi.

– str\_shuffle($string)

– Ví dụ:

<?php

echo str\_shuffle("Hello World");

?>

Những hàm format chuỗi

• strtolower — Chuyển chuỗi sang lowercase

<?php

echo strtolower("Hello WORLD."); //hello world

?>

• strtoupper — Chuyển chuỗi sang uppercase (chữ hoa)

Những hàm format chuỗi (tt)

• ucwords ($str ): Viết hoa ở mỗi ký tự đầuu

tiên của từ.

– $str: Chuỗi cần xử lý

– Trả về chuỗi sau khi đã xử lý

• ucfirst ($str ): Viết hoa từ đầu tiên của câu trong chuỗi

– $str: chuỗi cần xử lý

– Trả về kết quả sau khi xử lý

Ví dụ

<?php

$foo = 'hello world!';

$foo = ucfirst($foo); // Hello world!

$bar = 'HELLO WORLD!';

$bar = ucfirst($bar); // HELLO WORLD!

$bar = ucfirst(strtolower($bar)); // Hello world!

$foo = 'hello world!';

$foo = ucwords($foo); // Hello World!

$bar = 'HELLO WORLD!';

$bar = ucwords($bar); // HELLO WORLD!

$bar = ucwords(strtolower($bar)); // Hello World!

?>

Biểu thức chính quy - regular expression

• Giới thiệu

• preg\_match.

• Preg\_replace

• preg\_match\_all

• Patterns

Giới thiệu

• Biểu thức chính quy (regular expression- Regex): là một chuỗi miêu tả một bộ các chuỗi khác, theo những quy tắc cú pháp nhất định.

• Biểu thức chính quy thường được dùng trong các trình biên tập văn bản và các tiện ích tìm kiếm và xử lý văn bản dựa trên các mẫu được quy định.

• Sử dụng biểu thức chính qui đơn giản hơn nhiều trong lập trình và quá trình xử lý văn bản, và có những vấn đề sẽ không thể giải quyết được nếu không sử dụng regxp.

• Một số trường hợp sử dụng Regex

– Kiểm tra định dạng dữ liệu của người dùng: Kiểm tra chuỗi có phải là số điện thoại hợp lệ không? Đây có phải là tên của domain hay không?...

– Bóc tách dữ liệu trong một chuỗi: Lọc tất cả email trong trang web.

– Kết hợp với CURL để lấy dữ liệu từ trang khác: Tự động lấy dữ liệu cho các website tổng hợp tin tức,…

Giới thiệu (tt)

• Để sử dụng regular expression cần phải có một kiến thức từ cơ bản đến nâng cao về những biểu thức và cách thức hoạt động của nó trong các ngôn ngữ lập trình.

• Regular expression là một công cụ mạnh mẽ trong việc thao tác và trích xuất văn bản trên máy tính. Do đó nắm vững các biểu thức chính quy sẽ giúp tiết kiệm nhiều thời gian và công sức.

• Trong php, ta hay sử dụng các hàm: preg\_match, preg\_match\_all và preg\_replace để kiểm tra, tách dữ liệu và thay thế chuỗi.

Hàm preg\_match

• Hàm preg\_match: sử dụng để kiểm tra và tách dữ liệu của một chuỗi.

– preg\_match($pattern, $string[, $match]);

– $pattern: Mẫu tìm kiếm, là một chuỗi được bắt đầu và kết

thúc bằng ký tự: /

– $string: Chuỗi input được sử dụng cho tìm kiếm.

– $match: Nếu $match được cung cấp, thì đây là mảng nhận kết quả trả về. Nếu có, ta có thể thấy các kết quả trong

mảng $match[0], $match[1],…

– Hàm tìm kiếm trong $string theo mẫu $pattern.

• Nếu kết quả tìm thấy, hơp lệ, hàm trả về true và các kết quả

đặt trong $match

• Nếu không tìm thấy: trả về false.

preg\_replace và preg\_match\_all

• Hàm: preg\_replace: sử dụng để tìm kiếm và thay thế dữ liệu.

– preg\_replace ($pattern , $replacement , $subject [, $limit = -1 [, &$count ]] )

– $pattern: Mẫu cần tìm kiếm (mảng hay chuỗi).

– $replacement : giá trị thay thế (mảng hay chuỗi)

– $subject: mảng hay chuỗi để tìm kiếm và thay thế.

– $limit: số lần tìm kiếm thay thế. Default = -1: no limit

– $count: số lần được thay thế.

• Hàm preg\_match\_all: giống hàm preg\_match, nhưng trả về tất cả các kết quả trong $match.

Ví dụ

• Kiểm tra tên và email có hợp lệ không?

<?php $name = "Nguyen ";

if (preg\_match("/^[a-zA-Z ]\*$/",$name)) echo $name ." is valid name";

else echo "$name: Only letters and white space allowed ";

$email = "abc 123@lolhaha"; // Invalid email address

$regex = "/^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-

.]+$/";

if ( preg\_match( $regex, $email ) )

echo $email . " is a valid email. We can accept it."; else echo $email . " is an invalid email. Please try again.";

?>

Ví dụ

• Tách thông tin host và domain

<?php

$url = "http://www.php.net/index.html";

$pattern = '@^(?:http://)?([^/]+)@i';

preg\_match($pattern , $url, $matches);

$host = $matches[1];

echo "Host is $host <br>"; //www.php.net

// get last two segments of host name preg\_match('/[^.]+\.[^.]+$/', $host, $matches); echo "domain name is: {$matches[0]}\n";

//php.net

?>

Ví dụ

• Thay đổi định dạng chuỗi Date.

<?php

$pat = array ('/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/', '/^\s\*{(\w+)}\s\*=/');

$replace = array ('\3/\4/\1\2', '$\1 =');

$date = 'Date = 1999-5-27';

$s= preg\_replace($pat, $replace, $date); echo $s; //Date = 5/27/1999

?>

Pattern

• pattern: là một chuỗi biểu thức có quy tắc (Regular expression - Regex) để mô tả những chuỗi khác.

• Để sử dụng được 3 hàm rất mạnh đã giới thiệu, cần nắm vững được cách xây dựng các pattern của regex.

• Regex trong php được cung cấp bởi thư viện PCRE (Perl Compatible Regular Expressions).

• Một pattern là chuỗi bắt đầu và kết thúc bằng ký tự /.

• Trong regex có 2 loại ký tự:

– Ký tự thông thường: tập các chữ cái …

– Ký tự đặc biệt: Những ký tự có ý nghĩa khác với giá trị của

nó. ^, :, +, …

Ký tự thông thường (Literal

Characters)

• Regex cơ bản nhất chính là biểu thức bao gồm 1 số ký tự

thông thường.

– VD: /a/. Nó sẽ so khớp với thực thể đầu tiên của ký tự đó trong chuỗi. VD nếu có chuỗi: LazyDog is a boy, nó sẽ so khớp với ký tự a sau ký tự L. Regex này cũng có thể so khớp với ký tự a thứ 2 nếu ta yêu cầu tiếp tục tìm kiếm sau khi đã so khớp dc 1 lần.

– regex dog sẽ so khớp với dog trong chuỗi LazyDog is not a dog. Regex này bao gồm 1 tập 3 ký tự thông thường. Engine sẽ hiểu biểu thức này là: tìm d, theo sau bởi o, theo sau bởi g.

• Chú ý rằng PCRE mặc định phân biệt chữ hoa và chữ thường. Dog ko so khớp với dog.

• Để so sánh không phân biệt chữ hoa và thường, ta sử dụng

ký tự i sau chuỗi pattern.

Ví dụ: $pattern="/dog/i" sẽ không phân biệt hoa-thường khi so sánh (match) mẫu.

Ví dụ

<?php

$s ="abcdef";

$pattern ="/def/";

if (preg\_match($pattern, $s, $match))

{

}

else

{

}

?>

echo "Result:"; print\_r($match);

echo "Not found!";

Ký tự đặc biệt (Special symbol)

• Vì ta cần làm nhiều công việc phức tạp hơn là tìm kiếm 1 đoạn văn bản, cho nên phải trưng dụng 1 số ký tự để làm những nhiệm vụ đặc biệt.

• Các ký tự đặc biệt này được gọi là các metacharacter. [, \, ^, $, ., |, ?, \*, +, (, ),…

• Nếu muốn sử dụng các ký tự đặc biệt này giống như những ký tự thông thường, ta thêm ký tự backslash (\) vào trước ký tự đó.

Ký tự đặc biệt

• Ví dụ kiểm tra số nguyên dương. Các ký tự đặc biệt ^, [, ], +,

$ được sử dụng.

<?php

$pattern = '/^[0-9]+$/';

$subject = '603';

if (preg\_match($pattern, $subject, $matches)){ echo "$subject is a prime number ";

}

else

{

}

echo "Not a prime number";

Ý nghĩa các special symbols

Ký tự đặc biệt Ý Nghĩa

^ Bắt đầu chuỗi: $pattern ="/^php/"; Sử dụng để tìm các chuỗi bắt đầu bằng php

$ Kết thúc chuỗi: $pattern ="/php$/"; tìm chuỗi kết thúc bằng php

. Đại diện cho một ký tự bất kỳ: pattern="/a./ ": có thể mang giá trị: ab, a1, au, af . Nhưng không thể là abc, ab1, vì dấu . chi đại diện cho một ký tự!

+ Lặp lại ký tự hay cụm ký tự đứng trước nó (>=1) Ví dụ:

$pattern="/a+/" : nghĩa là lặp lại ký tự a và phải có ít nhất là một

ký tự a.

- 123a: 1(đúng)

- 123aa: 1(đúng)

- 123 : 0 (sai). vì không có chữ a nào được lặp.

$patterm="/ax+/" : biểu diễn cho dòng: ax, axx, axxxx - nhưng không

thể là "a" vì + đại diện cho >=1 ký tự.

Ý nghĩa các special symbols

Ký tự đặc biệt Ý Nghĩa

\* Cũng là lặp lại ký tự hay cụm ký tự đứng trước nó (>=0)

Ví dụ: a\* : nghĩa là lặp lại chữ a, nếu không có chữ a cũng không sao.

-123a: (1)

-123aaa: (1)

-123: (1). Không có chữ a cũng không sao.

$patterm="/ay\*/" : biểu diễn cho dòng: a, ay, ayy, ayyyyyyyyy,....

? Tồn tại hay không tồn tại ký tự hay cụm ký tự đứng trước nó:

$pattern="/ab? /" :biểu diễn cho: a, ab (b có thể có hoặc không)

\ Dấu \ đi kèm với 1 meta symbol sẽ làm mất ý nghĩa của meta symbol

đó - trả về symbol bình thường.

$pattern="/a\?b/": Tìm chuỗi chứa chuỗi con "a?b" vì ? Là ký tự đặc biệt, nên đặt \? Để nó trở thành ký tự thường.

\d Biểu diễn cho một số bất kỳ: 0-9:

$p ="/^\d/"; Tìm chuỗi bắt đầu bằng một ký tự số

Ý nghĩa các special symbols

Ký tự đặc biệt Ý Nghĩa

\D Biểu diễn một ký tự không phải số (ngược với \d):

\w Biểu diễn một ký tự bất kỳ: a-z, A-Z, 0-9

\W Biểu diễn một ký tự đặc biệt (không phải a-z, A-Z, 0-9)

\s+ Có ít nhất một khoảng trắng.

Ý nghĩa các special symbols

Ký tự đặc

biệt Ý Nghĩa

() Gom các ký tự thành một nhóm:

$pattern ="/\w\w(123)/"; Tìm các chuỗi có chứa chuỗi 123 và trước chuỗi 123 là 2 ký tự. Ví dụ chuỗi: "ab0q123d4"

{} Lặp ký tự hay cụm ký tự trước đó với 1 số lần cho trước.

{n}: n lần

{n, }. Lặp lại >= n lần

{n, m} Lặp lại từ n đến m lần

Ví dụ: $pattern="/\w\w(\d\d){2,4}/"; Tìm các chuỗi bắt đầu bằng 2 ký

tự (số hay a-z), sau đó là 2 ký tự số lặp lại từ 2 đến 4 lần.

Hợp lệ: "ab0q1253d4"

| Toán tử or, để lựa chọn hoặc cái này hoặc cái kia.

[] Chỉ đoạn ký tự cho phép

[a-z]: các ký tự thường từ a đến z [A-Z]: Các ký tự hoa từ A đến Z

[0-9]: các ký tự số

Ví dụ: $p ="/^(\(0[1-9]{1,2}\))?[0-9]{7,8}/";

Các ví dụ

• Kiểm tra tên: Chỉ bao gồm các ký tự a-z và

khoảng trắng?

• Kiểm tra email?

• Kiểm tra url?

• Kiểm tra số điện thoại?

Ví dụ kiểm tra tên, số điện thoại, email

<?php

function post($index, $val='') { if (isset($\_POST[$index])) return $\_POST[$index]; return $val; } function nameField($val){

if (!preg\_match("/^[a-zA-Z ]\*$/",$val)) echo "Only letters and white space allowed";

}

function numberField($val){ if(!preg\_match("/^[0-9]\*$/", $val)) echo "Invalid Number";

}

function emailField($val) {$pattern = "/^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$/"; if (!preg\_match($pattern, $val)) echo "Invalid email format";

/\* Hoac su dung doan code sau: if (!filter\_var($val, FILTER\_VALIDATE\_EMAIL)) echo "Invalid email format"; \*/

} ?>

<form method="post">

Name:<input type="text" name="name" value="<?php echo post('name');?>"/>

<span><?php echo nameField(post('name'));?></span><br>

Phone:<input type="text" name="phone" value="<?php echo post('phone');?>">

<span><?php echo numberField(post('phone'));?></span><br>

email<input type="text" name="email" value="<?php echo post('email');?>">

<span><?php echo emailField(post('email'));?></span><br>

<input type="submit" name="submit" value="Submit"/>

</form>

Ví dụ tách chuỗi

• Tách chuỗi $s thành các từ và đưa vào mảng

$result

<?php

$p="/^(\w+)\s+(\w+)\s+(\w+)$/";

$s="Monday Tuesday Wednesday"; preg\_match($p, $s, $result); print\_r($result);

?>

Ví dụ

• Tách chuỗi, ta có thể sử dụng hàm preg\_match. Hà sẽ trả về kết quả đầu tiên

• Hàm preg\_match\_all để có thể lấy hết kết quả trả về.

• Ví dụ:

<?php

$st = '<table><tr><td>cell 1-1</td><td>cell 1-2</td></tr>

<tr> <td>cell 2-1</td><td>cell 2-2</td></tr>

</table>';

$pattern ="/<td>([^>]\*)<\/td>/U"; preg\_match($pattern, $st, $arr1); preg\_match\_all(($pattern, $st, $arr2); print\_r($arr1); print\_r( $arr2);

Hàm preg\_replace

• Sử dụng để thay thế dữ liệu trong chuỗi theo yêu

cầu.

• preg\_replace ($pattern , $replacement , $subject [, $limit = -1 [, &$count ]] )

– $pattern: Mẫu dò tìm

– $replacement: chuỗi thay thế

– $subject: Chuỗi cần thay thế

– $limit: Số lần giới hạn được thay thế. -1: thay thế hết.

– $count: Số lần được thay thế

Ví dụ

• Ví dụ: Loại bỏ tag h1 trong chuỗi html

<?php

$p1="/(<h1>)|(<\/h1>)/";

$p2 = "/(<\/?h1>)/";

$s="<h1>Welcome to Php</h1>";

$replace ='';

$result1 = preg\_replace($p1, $replace, $s); echo ($result1);

$result2 = preg\_replace($p2, $replace, $s); echo($result2);

?>

Kết luận

• Chuỗi là thành phần đuợc sử dụng nhiều

trong các trang web.

• Php cung cấp lượng lớn các hàm quan trọng

liên quan tới chuỗi.

• Để xử lý chuỗi, ta sử dụng các hàm chuỗi thông thường hoặc sử dụng các biểu thức chính quy.

• Nêu ưu tiên các hàm xử lý chuỗi thông thường

vì tốc độ xử lý nhanh hơn.

Chapter 5

Lập trình hướng đối tượng trong Php

Nội dung

• Giới thiệu

• Tạo class

• Tạo đối tượng và truy xuất các thành phần

trong đối tượng

• Các hàm magic

• Tính kế thừa…

Giới thiệu OOP

• Lập trình hướng đối tượng (object-oriented programming - OOP) phương pháp viết mã nhóm các action tương tự nhau vào các class.

• Làm chương trình: tin cậy, mềm dẻo, dễ bảo trì.

• Thành phần chính là các class (các khuôn mẫu) và các

đối tượng (thể hiện của các class).

• Có các tính chất: đóng gói, kế thừa, trừu tượng …

Class

• Giới thiệu

• Định nghĩa class trong php

• Tạo đối tượng và truy xuất các thành phần của đối tượng.

• Autoload class trong php

• Các hàm magic trong class

• Tính kế thừa

Giới thiệu class

• Class là một sự trừu tượng hóa của đối tượng.

• Nó được xem như một kiểu dữ liệu. Nhưng khác với kiểu dữ liệu thông thường, nó bao gồm các biến (thuộc tính) và hàm (phương thức).

• Sử dụng hướng đối tượng khi xây dựng ứng dụng web là cần thiết để đáp ứng những yêu cầu cao trong xây dựng website hiện nay.

Định nghĩa một class PHP

• Cú pháp tạo class php

Class classname{

<modifier> $var1; var $var;

<modifier> function functionName()

{ //code thân phương thức

}

}

Trong đó:

Class: từ khóa.

Var, functionName: là tên các biến (thuộc tính), hàm (phương thức) được đóng gói trong lớp

Định nghĩa class trong php (tt)

• <modifier>: có thể là public, protected, private: thể hiện các phạm vi truy xuất của các thành phần. Mặc định, nếu không sử dụng là public

• public: Cho phép truy xuất thuộc tính từ bên ngoài

lớp và được kế thừa cho các class con.

• Private: Không cho phép truy xuất từ bên ngoài và không cho phép kế thừa sang class con

• Protected: Không cho phép truy xuất từ đối tượng ngoài nhưng cho kế thừa sang class con

<?php Class A{

Public $var1;

Ví dụ

Private $var2, $var3=5; Protected $var4; Public function F()

{

Định nghĩa class A

}

}

$x= new A();

$x->F();

echo "Funcion A<br>"; echo $this->var3;

Tạo thể hiện của class A Truy xuất tới phương thức

public F trong A

Toán tử $this, hàm tạo, hàm hủy

• Toán tử $this-> sẽ truy xuất tới một thành phần (thuộc tính, phương thức) trong chính class này.

• Hàm tạo: là hàm được tự động thực thi khi tạo thể hiện của đối tượng. Hàm thường sử dụng để tạo các giá trị cho các thuộc tính, kết nối CSDL. Trong php5, hàm tạo có cú pháp như sau:

Public function

{

construct(danh\_sách\_tham\_số)

//code hàm tạo tại đây

}

• Hàm hủy: Hàm được tự động chạy khi đối tượng bị hủy. Hàm có cú pháp như sau:

function destruct() {

//code hàm hủy

}

Tạo đối tượng và truy xuất dữ liệu

• Tạo đối tượng: sử dụng toán tử new classname.

• Nếu hàm tạo có các tham số, ta truyền các tham số vào cho

đối tượng.

• Sử dụng toán tử -> để truy xuất các thành phần được phép trong đối tượng.

• In xem nội dung một đối tượng: Ta có thể sử dụng các hàm deburg đã sử dụng trong mảng là print\_r hoặc var\_dump;

• Ví dụ:

– $obj = new student("sv001");

//tạo thực thể đối tượng student. Hàm

động được thực thi.

construct($id) sẽ tự

– $obj->show();//Gọi phương thức public show();

Class student{

Ví dụ

var $stuID; //giống public $name; Private $stuAge;

Protected $stuName; Public function show()

{

echo $this->stuID . "–" . $this->stuName;

}

function construct($id)

{ $this->stuID = $id;

$this->stuName="";

$this->stuAge = 18;

}

}

autoLoad class

• Một ứng dụng web thường cần xây dựng nhiều lớp đối tượng khác

nhau. Để dễ quản lý, ta thường lưu tất cả các class trong một (vài)

thư mục và đặt mỗi định nghĩa class trong một file riêng biệt và đặt

tên file trùng với tên của lớp.

• Khi cần sử dụng class nào, ta sử dụng câu lệnh include để load file

chứa lớp tương ứng vào trang chính.

• Có thể load file class tương ứng môt cách tự động mà không phải bắt buộc sử dụng câu lệnh include mỗi khi sử dụng class nào đó, ta sử dụng các hàm đặc biệt trong php. Chọn 1 trong 2 cách

– Sử dụng hàm function autoLoad($class)

{ //định nghĩa path để load class vào file php

}

– Sử dụng hàm spl (Standard Php Library):

spl\_autoload\_register($autoload\_function); để register hàm

$autoload\_function. Hàm $autoload\_function sẽ được tự động chạy

khi tạo một đối tượng mới.

Ví dụ

• Ví dụ: giả sử các class được đặt trong thư mục classes và có tên file là: X.class.php, trong đó X là tên class. Trong file index.php ở ngoài classes, ta viết các hàm tự động để load các class theo 2 cách đã nêu.

• Sử dụng autoload()

• Nội dung index.php

<?php

function autoload($c)

{ include "classes/".$c. ".class.php"; }

$x1 = new Class1; // include file classes/Class1.class.php

$x2 = new Class2(2);// load file classes/Class1.class.php

}

Ví dụ (tt)

• Sử dụng hàm spl\_autoload\_register. Index.php

<?php

function my\_autoloader($class) { include 'classes/' . $class . '.class.php';

}

spl\_autoload\_register('my\_autoloader');

$x1 = new Class1; //include/classes/Class1.class.php

$x2 = new Class2(2); //include/classes/Class2.class.php

Các static methods trong OOP PHP

• Đây là những phương thức được tạo sẵn (reserves)

trong php và nó bắt đầu bằng dấu hai gạch dưới (còn gọi là các magic function).

• Các static methods quan trọng:

construct(): Hàm tạo

destruct(): Hàm hủy

set: Tạo ra một thuộc tính một cách tự động khi thuộc tính đó chưa được định nghĩa trong class.

get: Xử lý dữ liệu khi truy xuất một thuộc tính. Thường

được xử lý trong trường hợp các thuộc tính không xác định trước.

toString(): Cho phép thao tác đối tượng trong class giống như một chuỗi.

Ví dụ

• Ví dụ 1.

class A{

private $a=5;

function set($name,$value)

{ $this->$name = $value; }

function get($name)

{ if (isset($this->$name)) return $this->$name\*2; else return "\*\*\*";

}

}

$x1 =new A; print\_r($x1); //xem nội dung của $x1

$x1->b =6; print\_r($x1);

$x1->a = 6; print\_r($x1); echo $x1->a; echo $x1->c;

?>

Ví dụ (tt)

Ví dụ 2.

<?php

class student

{ var $id; var $name;

function construct($id, $name)

{ $this->id= $id;

$this->name = $name;

}

function

toString()

{ return $this->id.":". $this->name;

}

}

$obj = new student("0001","SV 1");

echo $obj; // Tương đương $obj-> toString()

Tính kế thừa

• Một class có thể được kế thừa từ một class nào đó đã được khai báo trước.

• Class được kế thừa gọi là class cha (parent), class kế thừa gọi là class con(hay lớp dẫn xuất).

• Khi mở rộng một lớp, lớp con được thừa hưởng tất cả các phương thức, thuộc tính của lớp cha ngoại trừ các thuộc tính hoặc phương thức được khai báo với từ khóa private.

• Cú pháp tính kế thừa: sử dụng từ khóa extends. class classcon extends classcha

{

//khối lệnh;

}

<?php

Ví dụ

class A{ Private $a1, $a2; Protected $a3;

Public function construct(){

$this->a1=1; $this->a2= 2; $this->a3= 3;

}

public function F()

{ echo $this->a1 .":". $this->a2 .":" . $this->a3; }

}

class B extends A{

function F2($b1){ $this->a3 = $b1; }

}

$x = new B(); $x->F();

$x->F2(7); $x->F();

Một số hàm, hằng số hay sử dụng với

class

• get\_class(): Trả về tên class của một biến.

– Get\_class($obj)

– $obj: Tên biến đối tượng cần lấy class.

– Kết quả trả về tên class của thể hiện $obj.

• CLASS : Hằng số chứa tên của class đang chạy.

• METHOD : chứa tên phương thức của

class

Ví dụ

<?php class foo {

function name() {

echo "My name is " , get\_class($this) , "\n";

}

}

// create an object

$bar = new foo();

// external call

echo "Its name is " , get\_class($bar) , "\n";

// internal call

$bar->name();

?>

<?php class A {

function funct1() {

Ví dụ

echo

}

}

CLASS . "-> ".

METHOD ."<br/>";

class B extends A {

function funct2(){

echo

}

}

CLASS . "-> ".

METHOD ."<br/>";

$obj = new B();

$obj->funct1(); // A-> A::funct1

$obj->funct2(); // B-> B::funct2 echo "class= ". get\_class($o); // class= B

Kết luận

• Cú pháp xây dựng một class

• Tạo và truy xuất một đối tượng.

• Hàm tạo, hủy

• Các static method

• Tính kế thừa

<!DOCTYPE html>

<html lang="vi">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Thêm Sinh Viên</title>

</head>

<body>

    <h1>Thêm Sinh Viên</h1>

    <form action="./vd11.php"mssv" method="post">MSSV:</label>

        <input type="text" id="mssv" name="mssv" required><br><br>

        <label for="hoten">Họ Tên:</label>

        <input type="text" id="hoten" name="hoten" required><br><br>

        <label for="hinh">Hình:</label>

        <input type="file" id="hinh" name="hinh" accept=".jpg, .png" required><br><br>

        <input type="submit" value="Thêm Sinh Viên">

    </form>

</body>

</html>

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    // Nhận dữ liệu từ form

    $mssv = trim($\_POST['mssv']);

    $hoten = trim($\_POST['hoten']);

    $hinh = $\_FILES['hinh'];

    // Kiểm tra MSSV

    if (strlen($mssv) !== 10 || substr($mssv, 0, 2) !== "DH") {

        die("MSSV phải có 10 ký tự và bắt đầu bằng 'DH'.");

    }

    // Kiểm tra Họ Tên

    if (str\_word\_count($hoten) < 2) {

        die("Họ tên phải có ít nhất hai từ.");

    }

    // Kiểm tra Hình

    $allowed\_extensions = ['jpg', 'jpeg', 'png'];

    $file\_extension = strtolower(pathinfo($hinh['name'], PATHINFO\_EXTENSION));

    $max\_size = 1 \* 1024 \* 1024; // 1MB

    if (!in\_array($file\_extension, $allowed\_extensions) || $hinh['size'] > $max\_size) {

        die("Hình ảnh chỉ được phép là .JPG hoặc .PNG và kích thước phải nhỏ hơn 1MB.");

    }

    // Di chuyển file hình vào thư mục img

    $target\_directory = "img/";

    $target\_file = $target\_directory . basename($mssv . '.' . $file\_extension);

    if (move\_uploaded\_file($hinh['tmp\_name'], $target\_file)) {

        echo "Sinh viên đã được thêm thành công: MSSV = $mssv, Họ Tên = $hoten, Hình = $target\_file";

    } else {

        die("Có lỗi xảy ra khi tải hình lên.");

    }

} else {

    die("Yêu cầu không hợp lệ.");

}

?>

<?php

    $a= [3, 'a'];

//        echo '<pre>';// exit;

    print\_r($a); //in ra không chi tiết

    $a[]=10;

    var\_dump($a); // in ra chi tiết

    unset($a[1]);

    echo'<hr> Mang a sau khi xoa 1 phan tu';

    var\_dump($a);

    $x=1;

    if (isset($a[$x])) { //kiem tra ton tai

        echo "Mang co phan tu $x";

    }else {

        echo "Mang ko co phan tu $x";

    }

    echo "<hr> Mang a co ".count($a)."phan tu" ;//đếm số lượng

?>

<?php

$a= [1,2,3];

echo "<hr> mang a co" .count($a) ."phan tu";

$b[0]= 4;

$b[1]= 5;

$b[2]= 6;

$b[3]= "a";

echo "<hr> mang b co" .count($b) ."phan tu";

foreach ($a as $k => $v) {//k và v là 2 từ khóa ==> duyet vs key và valuse

    echo "<br>" . "a[" .$k ."]= ". $v ."";

}

foreach ($b as $value) {//duyet vs value

    echo "<br> Value b = $value";

}

?>

<?php

    $a= [

        [1,2,3],

        [4,5,6],

        [7],

        8

    ];

    echo "<hr>"."mang a 2 chieu co " .count($a) ."phan tu";

    echo "<pre>";

    print\_r($a);

    $b1= [1,2,3]; $b2= [4,5,6];

    $b= [$b1, $b2];

    var\_dump($a);

    var\_dump($b);

    echo "<hr>"."mang b 2 chieu co " .count($b) ."phan tu";

    echo "<pre>";

    print\_r($b);

//   /  print\_r($a[1][1]);

    $c=['x'=> 1, 'y'=> 2,'z'=> 3, 'd'=> 4];

    // print\_r($c['y']);

    //cac cach khai bao va dung mang

$d=[];

$d=[1,3];

$d=['x'=>1, 'y'=>2];

var\_dump($d);

echo "{$c['x']} , {$c['y']} , {$c['z']}, {$c['d']}";

$e=array();

$e=array(1,2);

$e=array('x'=>1, 'y'=>2);

var\_dump($e);

?>

<table>

<?php

?>

Array ( [0] => 3 [1] => a )

E:\LapTrinhWeb\Lop\vd6.php:6:

**array** *(size=3)*

0 => int 3

1 => string 'a' *(length=1)*

2 => int 10

Mang a sau khi xoa 1 phan tu

E:\LapTrinhWeb\Lop\vd6.php:10:

**array** *(size=2)*

0 => int 3

2 => int 10

Mang ko co phan tu 1

Mang a co 2phan tu

mang a co3phan tu

mang b co4phan tu  
a[0]= 1  
a[1]= 2  
a[2]= 3  
Value b = 4  
Value b = 5  
Value b = 6  
Value b = a

mang a 2 chieu co 4phan tu

Array

(

[0] => Array

(

[0] => 1

[1] => 2

[2] => 3

)

[1] => Array

(

[0] => 4

[1] => 5

[2] => 6

)

[2] => Array

(

[0] => 7

)

[3] => 8

)

E:\LapTrinhWeb\Lop\vd6.php:58:

**array** *(size=4)*

0 =>

**array** *(size=3)*

0 => int 1

1 => int 2

2 => int 3

1 =>

**array** *(size=3)*

0 => int 4

1 => int 5

2 => int 6

2 =>

**array** *(size=1)*

0 => int 7

3 => int 8

E:\LapTrinhWeb\Lop\vd6.php:59:

**array** *(size=2)*

0 =>

**array** *(size=3)*

0 => int 1

1 => int 2

2 => int 3

1 =>

**array** *(size=3)*

0 => int 4

1 => int 5

2 => int 6

mang b 2 chieu co 2phan tu

Array

(

[0] => Array

(

[0] => 1

[1] => 2

[2] => 3

)

[1] => Array

(

[0] => 4

[1] => 5

[2] => 6

)

)

E:\LapTrinhWeb\Lop\vd6.php:74:

**array** *(size=2)*

'x' => int 1

'y' => int 2

1 , 2 , 3, 4

E:\LapTrinhWeb\Lop\vd6.php:81:

**array** *(size=2)*

'x' => int 1

'y' => int 2

|  |
| --- |
|  |

1

**DANH SÁCH CÁC HÀM XỬ LÝ CHUỖI TRONG PHP**

**1. strlen() - Đo độ dài của chuỗi**

$string = "Hello, World!";

echo strlen($string); // Output: 13

**2. strtoupper() - Chuyển chuỗi thành chữ hoa**

$string = "hello";

echo strtoupper($string); // Output: HELLO

**3. strtolower() - Chuyển chuỗi thành chữ thường**

$string = "HELLO";

echo strtolower($string); // Output: hello

**4. ucfirst() - Chuyển ký tự đầu tiên của chuỗi thành chữ hoa**

$string = "hello";

echo ucfirst($string); // Output: Hello

**5. ucwords() - Chuyển chữ cái đầu tiên của mỗi từ trong chuỗi thành chữ hoa**

$string = "hello world";

echo ucwords($string); // Output: Hello World

**6. trim() - Loại bỏ khoảng trắng ở đầu và cuối chuỗi**

$string = " Hello World! ";

echo trim($string); // Output: Hello World!

**7. ltrim() - Loại bỏ khoảng trắng ở đầu chuỗi**

$string = " Hello World!";

echo ltrim($string); // Output: Hello World!

**8. rtrim() - Loại bỏ khoảng trắng ở cuối chuỗi**

$string = "Hello World! ";

echo rtrim($string); // Output: Hello World!

**9. substr() - Cắt chuỗi theo vị trí**

$string = "Hello World!";

echo substr($string, 6, 5); // Output: World

**10. str\_replace() - Thay thế một chuỗi con trong chuỗi**

$string = "Hello World!";

echo str\_replace("World", " ", $string); // Output: Hello !

**11. strpos() - Tìm vị trí của một chuỗi con trong chuỗi**

$string = "Hello World!";

echo strpos($string, "World"); // Output: 6

**12. strrpos() - Tìm vị trí xuất hiện cuối cùng của một chuỗi con trong chuỗi**

$string = "Hello World, World!";

echo strrpos($string, "World"); // Output: 13

**13. explode() - Chia chuỗi thành mảng dựa trên dấu phân cách**

$string = "apple,banana,orange";

$array = explode(",", $string);

print\_r($array); // Output: Array ( [0] => apple [1] => banana [2] => orange )

**14. implode() - Kết hợp các phần tử trong mảng thành chuỗi**

$array = ["apple", "banana", "orange"];

echo implode(",", $array); // Output: apple,banana,orange

**15. str\_repeat() - Lặp lại chuỗi một số lần**

$string = "Hello ";

echo str\_repeat($string, 3); // Output: Hello Hello Hello

**16. str\_split() - Chia chuỗi thành mảng các phần tử theo ký tự**

$string = "Hello";

print\_r(str\_split($string)); // Output: Array ( [0] => H [1] => e [2] => l [3] => l [4] => o )

**17. md5() - Mã hóa chuỗi thành mã MD5**

$string = "Hello";

echo md5($string); // Output: 8b1a9953c4611296a827abf8c47804d7

**18. sha1() - Mã hóa chuỗi thành mã SHA-1**

$string = "Hello";

echo sha1($string); // Output: f7c3bc1d808e04732adf679965ccc34ca7ae3441

**19. stristr() - Tìm kiếm chuỗi con trong chuỗi (không phân biệt chữ hoa/thường)**

$string = "Hello World!";

echo stristr($string, "world"); // Output: World!

**20. addslashes() - Thêm dấu gạch chéo vào các ký tự đặc biệt**

$string = "It's a test!";

echo addslashes($string); // Output: It\'s a test!

**21. stripslashes() - Loại bỏ dấu gạch chéo từ chuỗi**

$string = "It\'s a test!";

echo stripslashes($string); // Output: It's a test!

**22. htmlspecialchars() - Chuyển các ký tự đặc biệt thành các mã HTML**

$string = "<div>Hello</div>";

echo htmlspecialchars($string); // Output: &lt;div&gt;Hello&lt;/div&gt;

**23. html\_entity\_decode() - Giải mã các ký tự HTML**

$string = "&lt;div&gt;Hello&lt;/div&gt;";

echo html\_entity\_decode($string); // Output: <div>Hello</div>

**24. strval() - Chuyển một giá trị thành chuỗi**

$number = 123;

echo strval($number); // Output: 123

**25. chr() - Trả về ký tự từ mã ASCII**

echo chr(65); // Output: A

**26. ord() - Trả về mã ASCII của ký tự đầu tiên**

echo ord("A"); // Output: 65

**27. str\_contains() ( 8.0+) - Kiểm tra xem chuỗi có chứa chuỗi con không**

$string = "Hello World!";

echo str\_contains($string, "World") ? "Found" : "Not Found"; // Output: Found

**DANH SÁCH CÁC HÀM XỬ LÝ MẢNG (array) TRONG PHP**

### 1. count() - Đếm số lượng phần tử trong mảng

$array = [1, 2, 3, 4];

echo count($array); // Output: 4

### 2. array\_push() - Thêm một hoặc nhiều phần tử vào cuối mảng

$array = [1, 2, 3];

array\_push($array, 4, 5);

print\_r($array); // Output: Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 )

### 3. array\_pop() - Xóa phần tử cuối cùng của mảng

$array = [1, 2, 3, 4];

array\_pop($array);

print\_r($array); // Output: Array ( [0] => 1 [1] => 2 [2] => 3 )

### 4. array\_shift() - Xóa phần tử đầu tiên của mảng

$array = [1, 2, 3, 4];

array\_shift($array);

print\_r($array); // Output: Array ( [0] => 2 [1] => 3 [2] => 4 )

### 5. array\_unshift() - Thêm một hoặc nhiều phần tử vào đầu mảng

$array = [2, 3, 4];

array\_unshift($array, 1);

print\_r($array); // Output: Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 )

### 6. array\_merge() - Hợp nhất hai hoặc nhiều mảng

$array1 = [1, 2];

$array2 = [3, 4];

$array3 = array\_merge($array1, $array2);

print\_r($array3); // Output: Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 )

### 7. array\_diff() - Tìm các phần tử có trong mảng này nhưng không có trong mảng khác

$array1 = [1, 2, 3];

$array2 = [2, 3, 4];

$result = array\_diff($array1, $array2);

print\_r($result); // Output: Array ( [0] => 1 )

### 8. array\_intersect() - Tìm các phần tử chung giữa hai mảng

$array1 = [1, 2, 3];

$array2 = [2, 3, 4];

$result = array\_intersect($array1, $array2);

print\_r($result); // Output: Array ( [1] => 2 [2] => 3 )

### 9. array\_slice() - Cắt một phần của mảng

$array = [1, 2, 3, 4, 5];

$slice = array\_slice($array, 2, 2);

print\_r($slice); // Output: Array ( [0] => 3 [1] => 4 )

### 10. array\_splice() - Thay thế một phần của mảng bằng các phần tử khác

$array = [1, 2, 3, 4, 5];

array\_splice($array, 2, 2, [6, 7]);

print\_r($array); // Output: Array ( [0] => 1 [1] => 2 [2] => 6 [3] => 7 [4] => 5 )

### 11. array\_map() - Áp dụng một hàm lên tất cả các phần tử trong mảng

$array = [1, 2, 3];

$result = array\_map(function($n) { return $n \* 2; }, $array);

print\_r($result); // Output: Array ( [0] => 2 [1] => 4 [2] => 6 )

### 12. array\_filter() - Lọc các phần tử trong mảng theo điều kiện

$array = [1, 2, 3, 4, 5];

$result = array\_filter($array, function($n) { return $n % 2 == 0; });

print\_r($result); // Output: Array ( [1] => 2 [3] => 4 )

### 13. array\_reduce() - Dùng một hàm để giảm mảng về một giá trị duy nhất

$array = [1, 2, 3, 4];

$result = array\_reduce($array, function($carry, $item) { return $carry + $item; });

echo $result; // Output: 10

### 14. in\_array() - Kiểm tra xem một giá trị có tồn tại trong mảng hay không

$array = [1, 2, 3];

echo in\_array(2, $array) ? 'Found' : 'Not Found'; // Output: Found

### 15. array\_search() - Tìm kiếm giá trị trong mảng và trả về key của phần tử

$array = [1, 2, 3];

echo array\_search(2, $array); // Output: 1

### 16. array\_keys() - Trả về tất cả các khóa (keys) trong mảng

$array = ["a" => 1, "b" => 2, "c" => 3];

print\_r(array\_keys($array)); // Output: Array ( [0] => a [1] => b [2] => c )

### 17. array\_values() - Trả về tất cả các giá trị (values) trong mảng

$array = ["a" => 1, "b" => 2, "c" => 3];

print\_r(array\_values($array)); // Output: Array ( [0] => 1 [1] => 2 [2] => 3 )

### 18. array\_flip() - Đổi chỗ key và value trong mảng

$array = ["a" => 1, "b" => 2];

print\_r(array\_flip($array)); // Output: Array ( [1] => a [2] => b )

### 19. array\_walk() - Áp dụng một hàm vào mỗi phần tử của mảng

$array = [1, 2, 3];

array\_walk($array, function(&$item, $key) { $item = $item \* 2; });

print\_r($array); // Output: Array ( [0] => 2 [1] => 4 [2] => 6 )

### 20. array\_unique() - Loại bỏ các phần tử trùng lặp trong mảng

$array = [1, 2, 2, 3, 4, 4];

print\_r(array\_unique($array)); // Output: Array ( [0] => 1 [1] => 2 [3] => 3 [4] => 4 )

### 21. array\_rand() - Lấy ngẫu nhiên một hoặc nhiều khóa từ mảng

$array = [1, 2, 3, 4, 5];

$randomKey = array\_rand($array);

echo $randomKey; // Output: (ví dụ) 2

### 22. array\_sum() - Tính tổng tất cả các phần tử trong mảng

$array = [1, 2, 3, 4];

echo array\_sum($array); // Output: 10

### 23. array\_product() - Tính tích tất cả các phần tử trong mảng

$array = [1, 2, 3, 4];

echo array\_product($array); // Output: 24

### 24. sort() - Sắp xếp mảng theo thứ tự tăng dần

$array = [4, 2, 5, 1, 3];

sort($array);

print\_r($array); // Output: Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 )

### 25. rsort() - Sắp xếp mảng theo thứ tự giảm dần

$array = [4, 2, 5, 1, 3];

rsort($array);

print\_r($array); // Output: Array ( [0] => 5 [1] => 4 [2] => 3 [3] => 2 [4] => 1 )

### 26. asort() - Sắp xếp mảng theo giá trị nhưng giữ nguyên khóa

$array = ["apple" => 3,"banana" => 1,"cherry" => 2];

asort($array);

print\_r($array);

**Kết quả**:

Array

(

[banana] => 1

[cherry] => 2

[apple] => 3

)