

NHẬP MÔN LẬP TRÌNH WEB

ThS. Trần Văn Hùng

Mail: hung.tranvan@stu.edu.vn

2020





NỘI DUNG CHÍNH



Chương 1: Giới thiệu



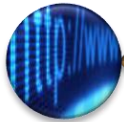
Chương 2. Ngôn ngữ lập trình php



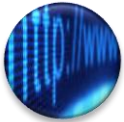
Chương 3. Mảng trong php



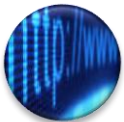
Chương 4. Làm việc với chuỗi



Chương 5. Lập trình hướng đối tượng trong Php



Chương 6. CSDL Mysql



Chương 7. Php Data Object



Nội dung

- Chương 1: Giới thiệu
 - Các mô hình lập trình ứng dụng
 - Ứng dụng web
 - Ứng dụng web php
 - Cài đặt - cấu hình và công cụ
 - Cấu hình:
 - Các bước cài đặt một ứng dụng web động php
- Chương 2. Ngôn ngữ lập trình php
 - Giới thiệu
 - Mô hình
 - Đặc điểm 1 trang php
 - Biến, hằng, kiểu dữ liệu, các phép toán cơ bản
 - Lệnh echo, print
 - Lệnh include, include_once, require, require_once
 - Xử lý lỗi trên Php

Nội dung

- Chương 3. Mảng trong php
 - Giới thiệu mảng trong php
 - Thao tác trên mảng
 - Mảng đa chiều
 - Một số hàm hay được sử dụng với Array
- Chương 4. Làm việc với chuỗi
 - Tạo chuỗi: 3 cách
 - Các ký tự đặc biệt
 - Phép toán nối chuỗi
 - Nội suy của biến trong chuỗi
 - Các phép toán trên chuỗi: so sánh, cắt chuỗi,..
 - Biểu thức chính quy (regular expression) trong Php

Nội dung

- Chương 5: Lập trình hướng đối tượng trong Php
 - Giới thiệu
 - Tạo một lớp và truy xuất các thành phần của lớp từ trong lớp
 - Tạo một đối tượng:
 - Hạn chế truy cập các thuộc tính và phương thức
 - Tính kế thừa của lớp trong php
 - Các static method trong OOP
 - Các ví dụ

Nội dung

- Chương 6: Thao tác với Mysql trong php
 - Các bước làm việc với CSDL Mysql:
 - Kết nối tới Mysql
 - Viết truy vấn
 - Thực thi truy vấn
 - Xử lý kết quả
 - Đóng kết nối

Nội dung

- Chương 7: Php Data Object
 - Giới thiệu
 - Cài đặt
 - Lớp PDO và PDOStatement
 - Các bước làm việc với CSDL bằng php

Tài liệu tham khảo

- [1]. **Steven Holzer**, Thiết kế web động với PHP 5, 2005, NXB Thống Kê (sách dịch)
- [2] **Wankyu Choi- Allan Kent - Chris Lea - Ganesh Prasad - Chris Ullman**, *Beginning PHP (1, 2)*, Wrox Press
- [3] <http://tranvanhung.fitstu.net/subjects/lap-trinh-web/>

Hình thức đánh giá

- Giữa kỳ: 40%: Thi trắc nghiệm + viết, được sử dụng tài liệu
- Cuối kỳ: 60%. Thi trắc nghiệm + viết, được sử dụng tài liệu



Chương 1:

NGÔN NGỮ LẬP TRÌNH 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.

Dữ liệu được chia sẻ
cho nhiều người dùng.

- Nhược 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 server.

- Ưu điểm

- □ Hỗ trợ nhiều người dùng
- □ Giảm bớt xử lý cho Client -> Không yêu cầu máy tính ở Client có cấu hình mạnh.
- □ Xử lý nhận và hiển thị dữ liệu tập trung tại Application Server <-> để quản lý, bảo trì và nâng cấp.
- □ Xử lý truy cập dữ liệu tập trung tại Database Server.

- Nhược điểm

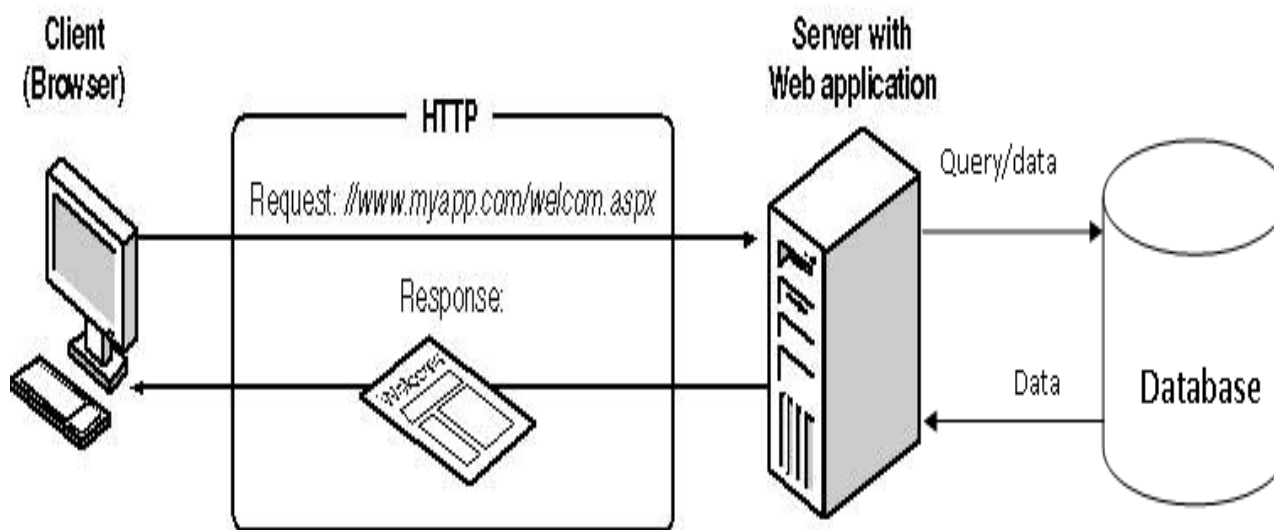
- □ Phải sử dụng thêm một Application Server -> Tăng chi phí



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

Các ngôn ngữ lập trình web

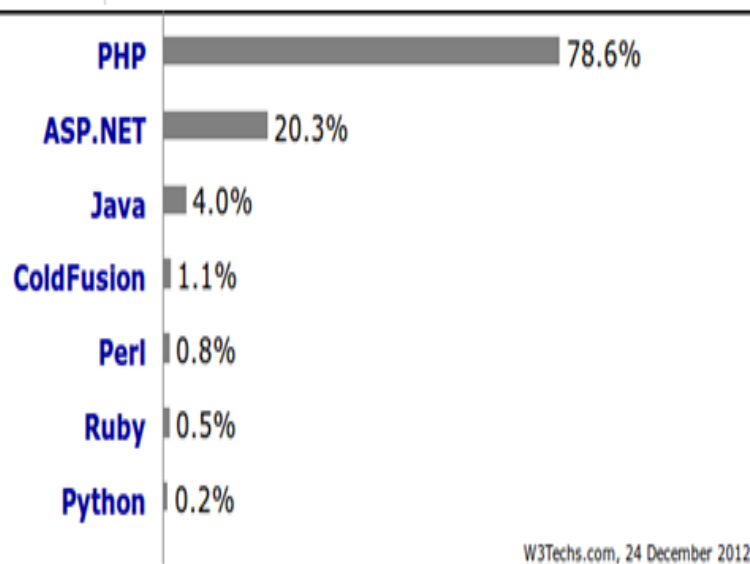
- Có 2 nhóm ngôn ngữ lập trình web: Ngôn ngữ lập trình phía client và ngôn ngữ lập trình phía server.
- Client side:
 - Code chạy trên trình duyệt web.
 - Ngôn ngữ tiêu biểu: javascript, vbscript, actionScript.
- Server side: là các ngôn ngữ mà mã được dịch và chạy trên máy chủ.
 - Asp.net:
 - là môi 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

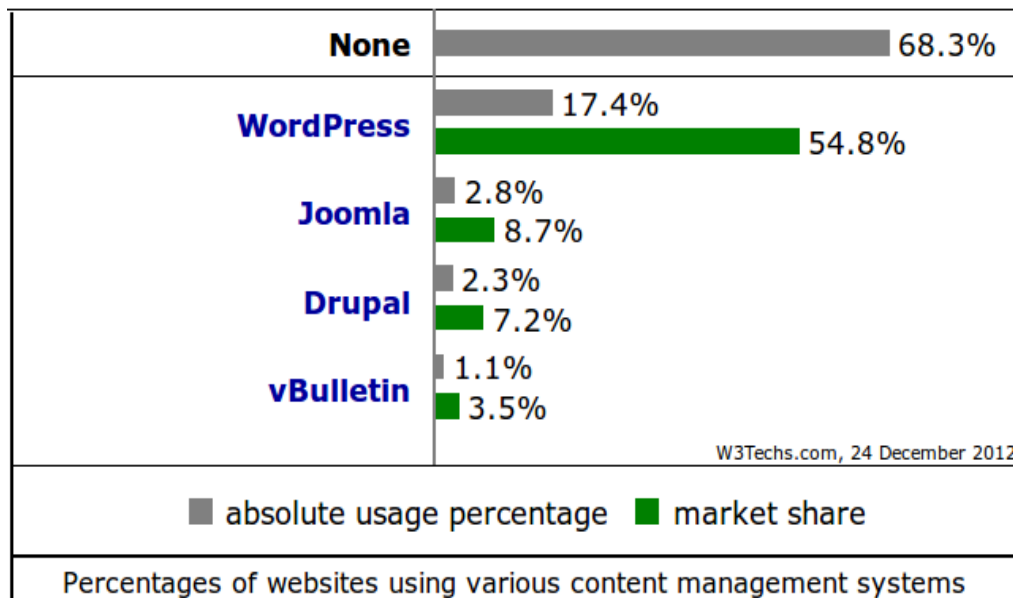
- Giới thiệu:
 - Mã nguồn mở miễn phí đang được phát triển nhanh trong những năm gần đây.
 - Php được chạy trên web sever IIS hoặc Apache.
 - Php có thể thao tác với mọi CSDL, nhưng thông thường đi với Mysql- một hệ quản trị CSDL miễn phí.
 - Code chạy nhanh, ổn định.
 - Php hỗ trợ rất nhiều hàm giúp 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/>



Usage of server-side programming languages for websites



Percentages of websites using various content management systems

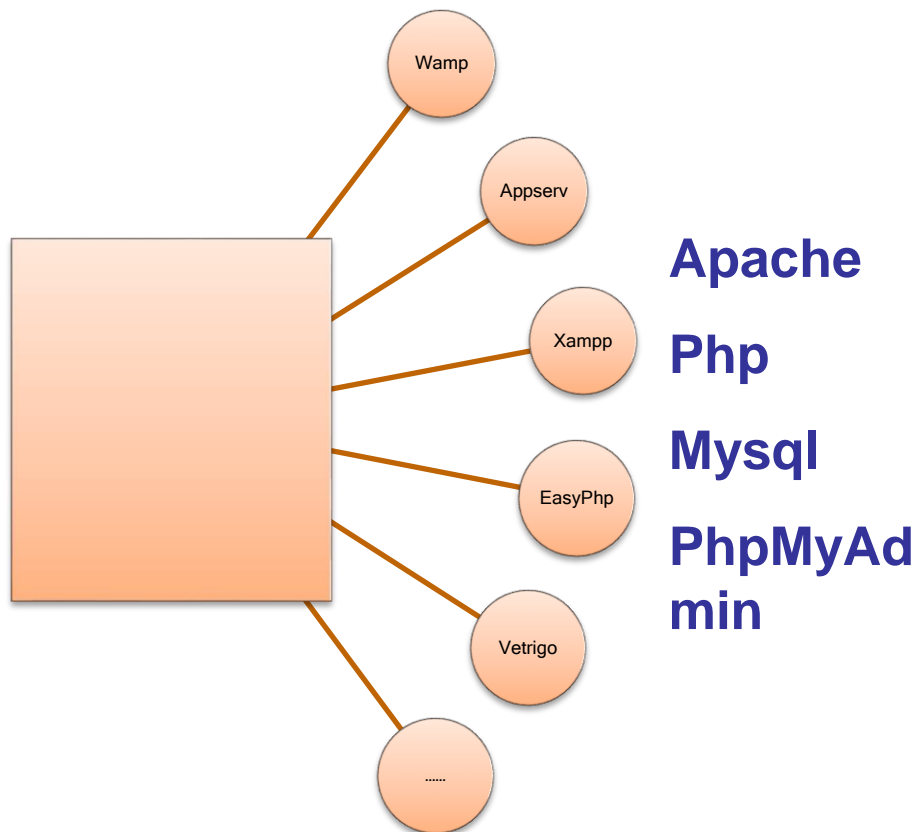
Cài đặt môi trường

- 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:
 - 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:
 - Cài đặt.
 - Kiểm tra.

Cài đặt thay thế

- Để dễ dàng cho người 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 thay thế



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:
- 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

 www.wampserver.com/en/#download-wrapper



WampServer

Apache, PHP, MySQL on Windows

START

DOWNLOAD

FORUM

DOWNLOAD WAMPSERVER (32 BITS & PHP 5.5) 2.5

Wampserver is available for free (under the GPL license). You can fill up this form that will enable us to send you the Alter Way Training news, publishing society, as well as all the informations linked to Wapserver evolutions. If you don't wish it, you can [download directly](#).

WARNING : Don't Use previous WampServer Extensions/Addons. There are no more compatible with the new wampserver version's (VC11)

WARNING : Vous devez avoir installé Visual Studio 2012 : VC 11 vcredist_x64/86.exe

Visual Studio 2012 VC 11 vcredist_x64/86.exe : <http://www.microsoft.com/en-us/download/details.aspx?id=30679>

WARNING : Do not try to install WampServer 2 over WAMP5.

If WAMP5 is installed on your computer, save your data, uninstall it and delete the WAMP5 directory before installing WampServer 2.

WARNING : All the components of the v2.2 WampServer stack have been compiled with VC9 version of Microsoft compiler.

Earlier versions of Wampserver have been made with VC6 version of Microsoft compiler.

So, You can't mix components of 2.2 stack with previous version of Wampserver Stack components.

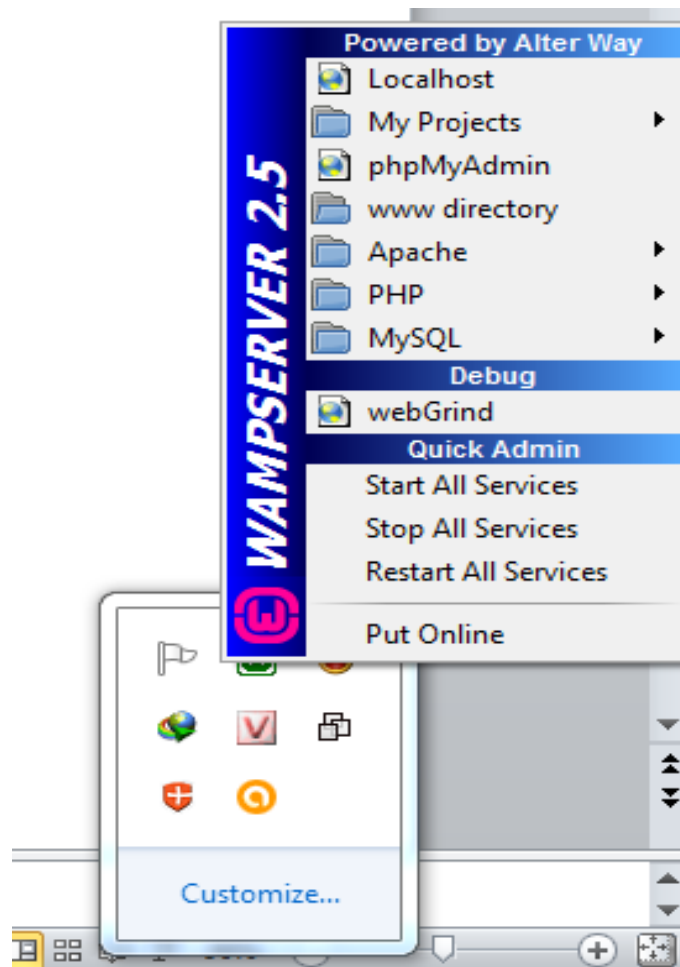
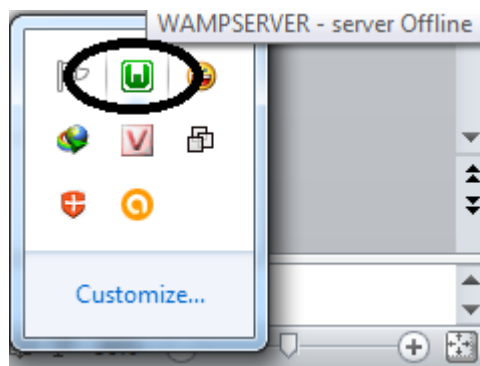
If you do it you will get an instable Wampserver.



Cài đặt wamp server

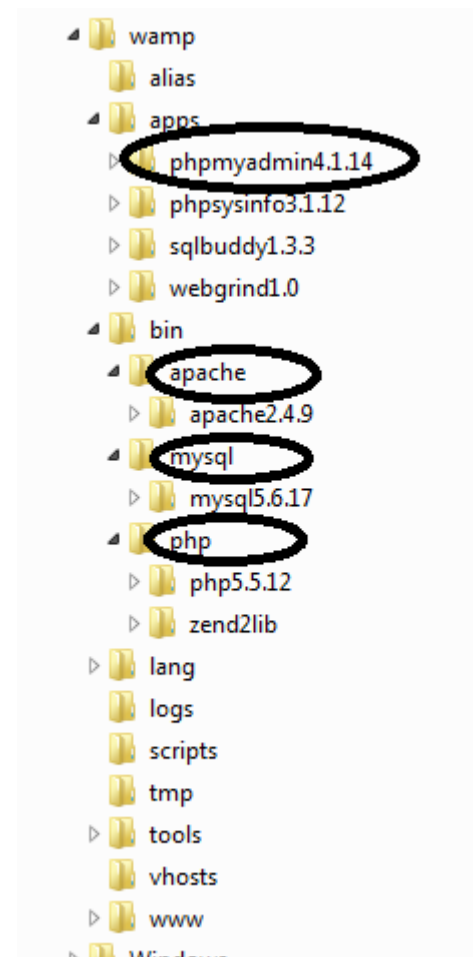
- Cài đặt: mặc định tại c:\wamp
 - Click vào file vừa tải về: wampserver2.5-Apache-2.4.9-Mysql-5.6.17-php5.5.12-32b.exe
 - Chọn Next,...
 - ...
 - finish
- Khởi động wamp.
- Kết quả:
 - Mở trình duyệt web và chạy thử: `http://localhost`
 - để kiểm
tra chi tiết các kết quả cài đặt

Cài đặt wamp server



Cấu hình wamp server

- Kết quả thư mục cài đặt của wamp
 - C:\wamp\bin\apache: 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 hay

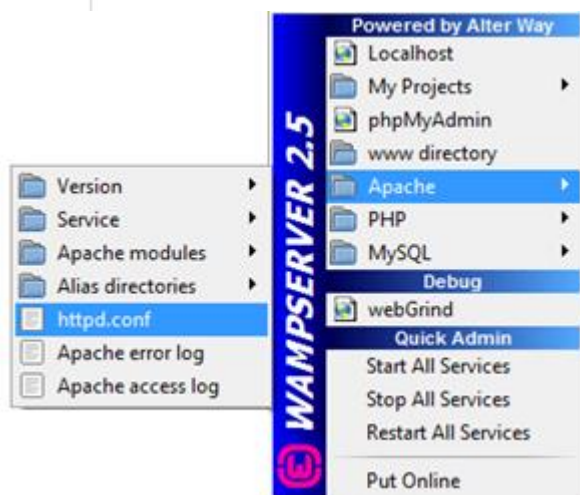


Cấu hình Wamp server

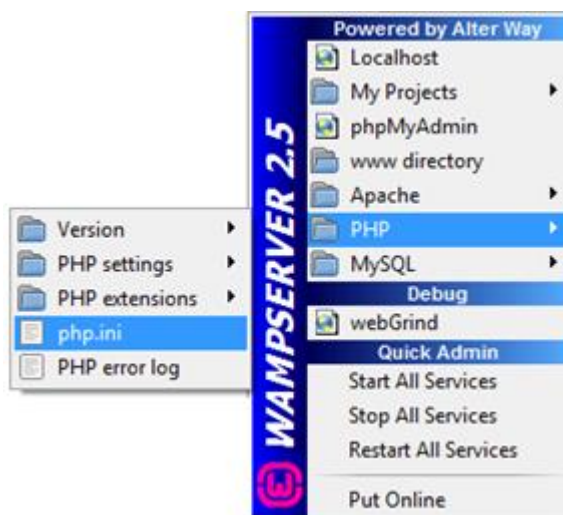
- Sau khi cài đặt xong, có thể chạy được các ứng dụng php mà không cần cấu hình. Tuy nhiên, để chạy được các chức năng khác, và để thuận lợi hơn khi cài và chạy ứng dụng web, ta nên cấu hình lại các thành phần này.
- Mỗi thành phần apache, php, mysql và phpmyadmin có các file cấu hình riêng và được lưu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

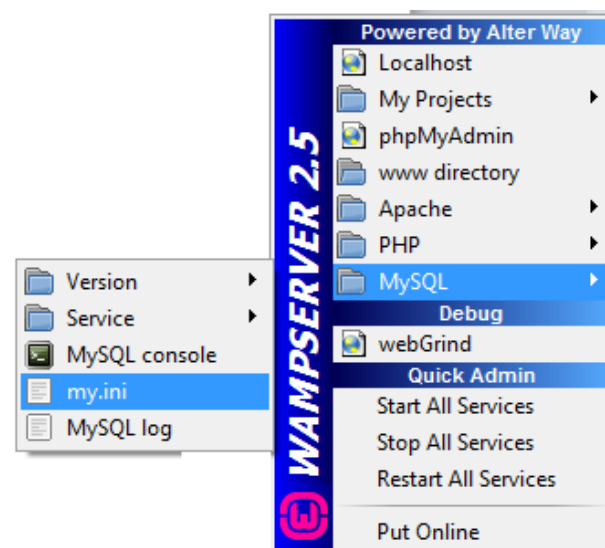
- Ta cũng có thể mở và sửa nhanh các file cấu hình từ wamp manager
- Sau khi sửa xong nên bấm vào: **restart All Services** trên wamp manager để cập nhật và load cấu hình mới.



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]
ErrorDocument 404 /Doanweb/quanly/su-co.html
```



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 chứa source
 - Mở file và sửa lại các dòng config tối 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





Chương 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

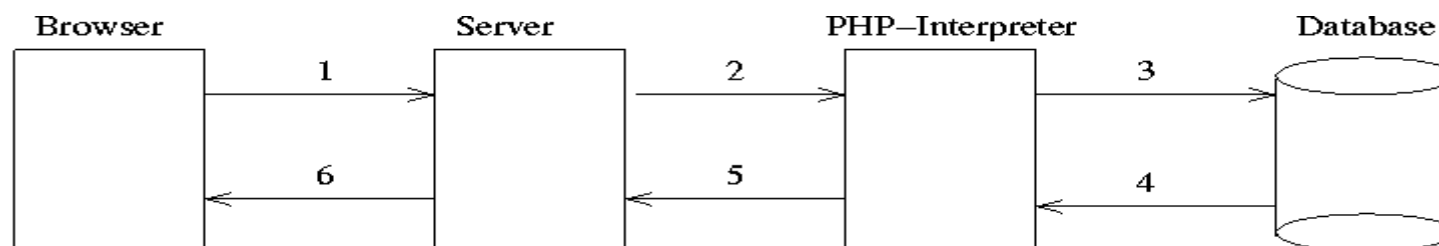
Giới thiệu

- 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 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 đoạn 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 quy 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.
- $\$x = \$a > \$b ? \$a : \$b$; \Leftrightarrow 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

Ký tự đặc biệt	
\n	Xuống dòng
\t	tab
\r	Về đầu dòng
\\	backslash
\'	Nháy đơn
\"	Nháy kép

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  
";
```

Kết quả hiển thị
Tổng \$s1 + \$s2
=\$s3
Tổng 5 + 10 =15
Tổng 5 + 10 = 15
arr['x1'] là 4

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 style='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 giá trị_1: S1; break;  
    case giá trị_2: S2; break;  
    case: giá trị_n: Sn; break;  
    [default: Sn+1]  
}
```

Kiểm tra biểu_thức, nếu biểu thức bằng giá trị giá trị_i ($i=1 \dots n$) sẽ thi hành lệnh (hay khối lệnh) S_i tương ứng. Ngược lại, sẽ thi hành lệnh S_{n+1} .

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

Ví dụ

```
switch ($i) {  
    case 0:  
        echo "i equals 0";  
        break;  
    case 1:  
        echo "i equals 1";  
        break;  
    case 2:  
        echo "i equals 2";  
        break;  
}
```

```
<?php  
if ($i == 0) {  
    echo "i equals 0";  
} elseif ($i == 1) {  
    echo "i equals 1";  
} elseif ($i == 2) {  
    echo "i equals 2";  
}
```

Phát biểu lặp

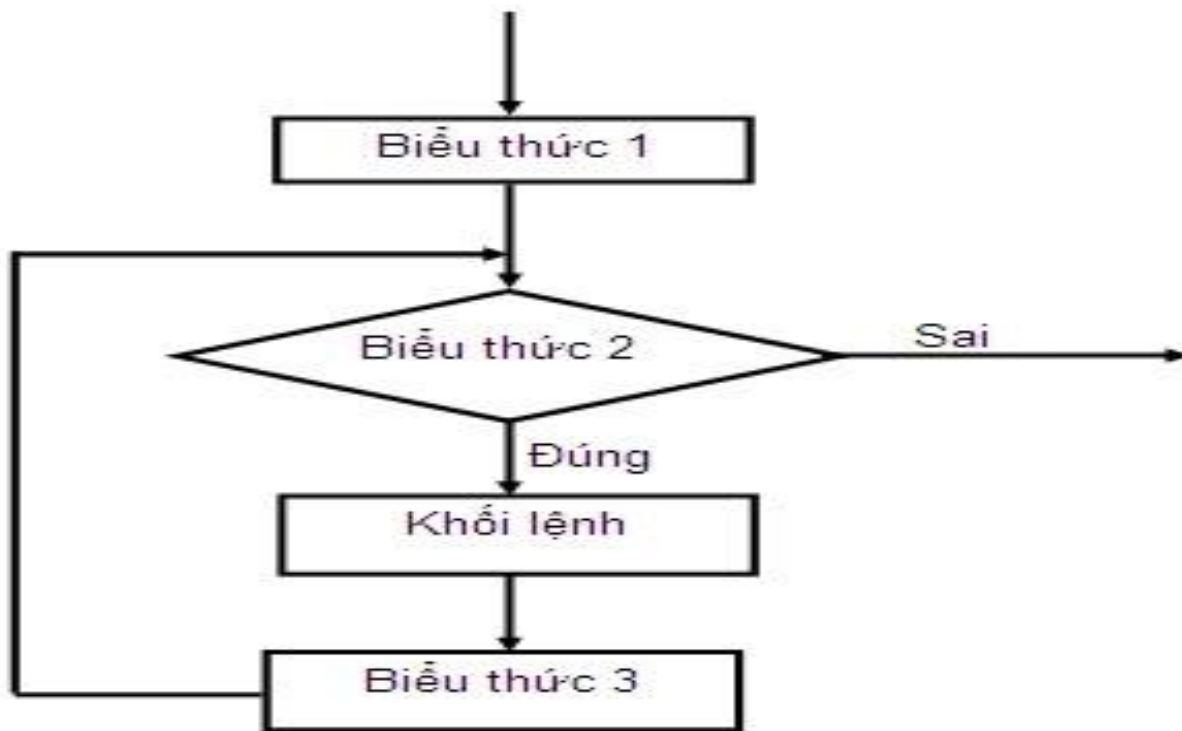
- Các loại phát biểu lặp
 - Vòng lặp for
 - Vòng lặp while
 - Vòng lặp do..while
 - Vòng lặp foreach



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 thức 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++)
```

```
{
```

```
    ?>
```

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

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

```
    <?php
```

```
}
```

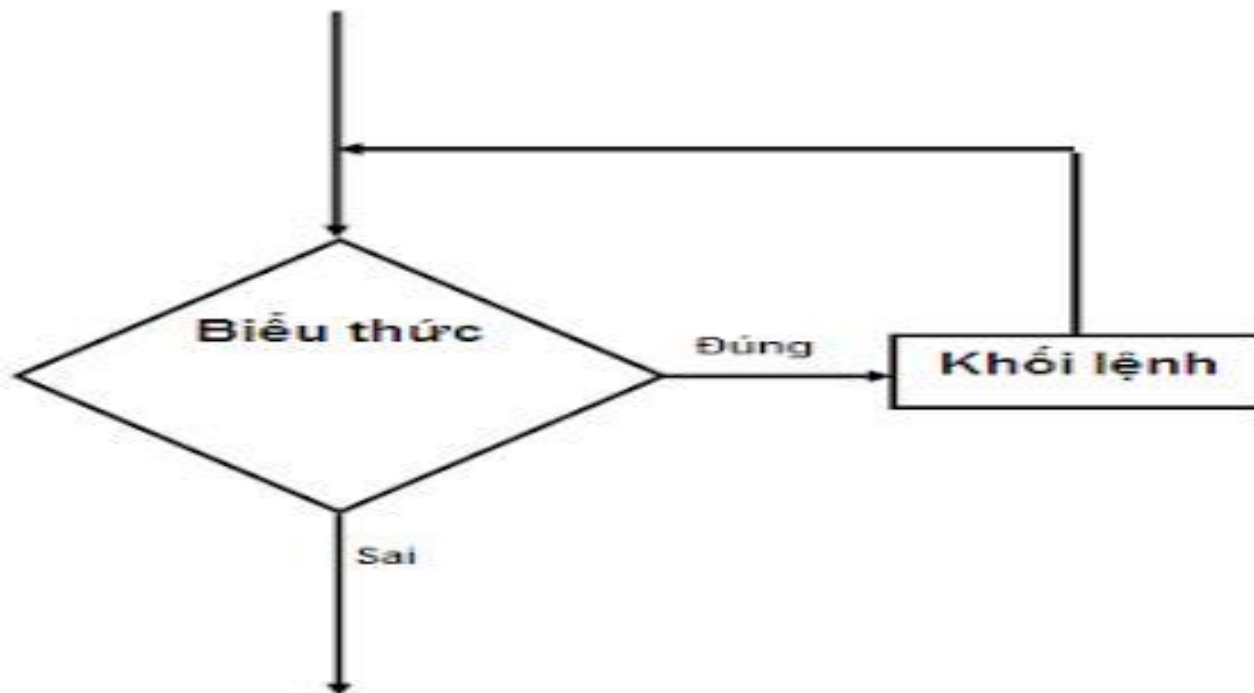
```
?>
```

```
</table>
```

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:

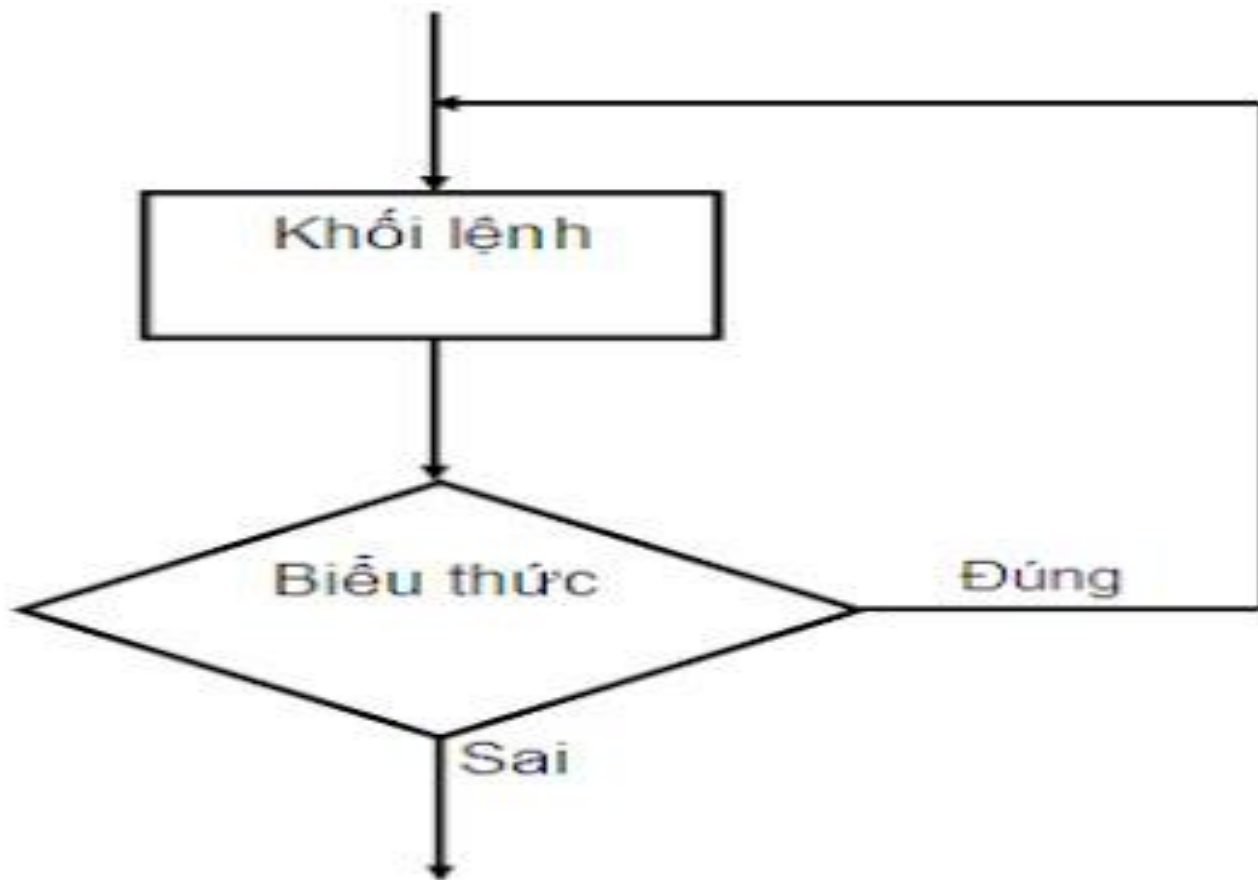
```
do{
```

```
    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 required 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);
```

```
?>
```

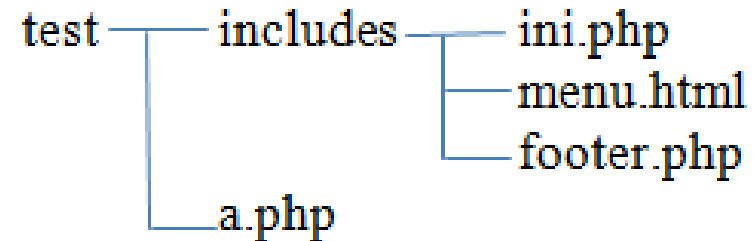
Nội dung: menu.html

```
<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

```
<?php
Require "includes/ini.php"; ?>
<div id=menu>
<?php
include "includes/menu.html"; ?>
</div>
<div id="content">
<?php
    $s ="Chào bạn :$user - " . AGE;
    echo $s;
</div>
<div id="footer">
```

Nội dung tương đương

```
<?php    $user= "admin";
define("AGE", 20); ?>
<div id=menu>
<a href='http://stu.edu.vn'>STU</a>
<a href='https://facebook.com/'>
    facebook</a>
</div>
<div id="content">
<?php
    $s ="Chào bạn :$user - " . AGE;
    echo $s;
?>
```



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 của 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: Số 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_set(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

Ký tự	Mô tả	Giá trị trả về
d	2 số ngày trong tháng	01 to 31
D	3 ký tự của thứ trong tuần	Mon through Sun
j	Ngày trong tháng	1 to 31
l	Thứ trong tuần	Sunday through Saturday
N	Số thứ tự của thứ trong tuần.	1 (for Monday) through 7 (for Sunday)
z	Ngày thứ bao nhiêu trong năm	0 through 365
W	Tuần thứ trong năm	Example: 42 (the 42nd week in the year)
F	Tên tháng đầy đủ bằng tiếng Anh	January through December
m	Tháng trong năm (dạng số)	01 through 12
M	A short textual representation of a month, three letters	Jan through Dec

Ký tự	Mô tả	Giá trị trả về
n	Tháng trong năm - dạng số	1 through 12
t	Ngày trong tháng	28 through 31
L	1: Nếu năm nhuận. 0: nếu không	1 if it is a leap year, 0 otherwise.
Y	Năm, 4 số	Examples: 1999 or 2003
y	Năm: 2 số	Examples: 99 or 03
a	Thời gian sáng hay chiều	am or pm
h	12-hour format of an hour with leading zeros	01 through 12
H	24-hour format of an hour with leading zeros	00 through 23
i	Minutes with leading zeros	00 to 59
s	Seconds, with leading zeros	00 through 59

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\l 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
- Phạm vi của biến



Cú pháp tạo function

- Lý do tạo hàm?
- Cách pháp khi tạo hàm.

function tênhàm([tham số 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. $\$thamsố = giá trị$.
- 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  $F(8) \leftrightarrow F(8, 1)$ 
```

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 triển khai ứng dụng web, với mỗi giai đoạn, ta cần biết cách cấu hình để debug lỗi cho phù hợp. Có nhiều cách để 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.
 - Hiện 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 httpd.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



Chương 3:

ARRAY - MẢNG

NỘI DUNG CHÍNH



Giới thiệu mạng php



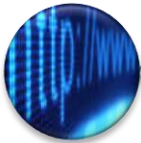
Tạo và truy xuất dữ liệu



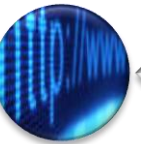
Mạng nhiều chiều



Một số hàm hay sử dụng với mạng



Mạng 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 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);
```

Array

(

[0] => 10

[a] => 20

array (size=3)
)

0 => int 10

"a" => int 20

"b" => int 15

?>



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 ";
}
?>
```

0 - 1
1 - 3
2 - 5

0 - 1
1 - 3
2 - 5

b1 - v1
b2 - 3

v1
3

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 = $arr["y"];              //$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";  
}  
?>
```

Kết quả:

fruits[0] = apple

fruits[1] = banana

fruits[2] = lemon

fruits[3] = orange

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
?>
```

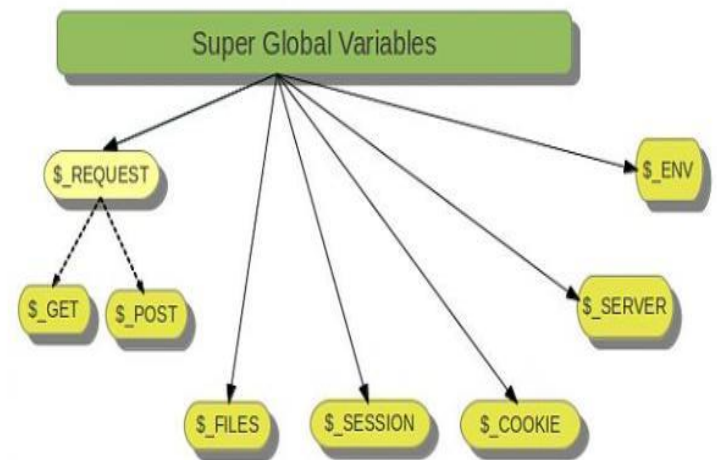
- Implode(): Ghép các phần tử của mảng thành chuỗi.
String implode (string \$glue , array \$pieces)

- \$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)

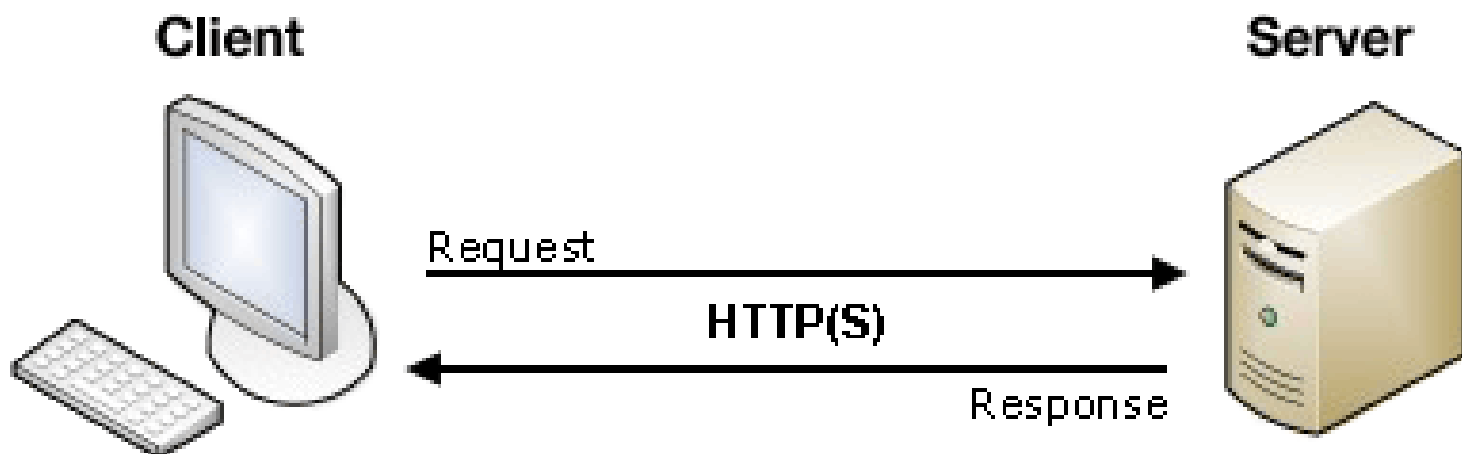
- `$_REQUEST`, `$_GET`, `$_POST`: chứa dữ liệu các biến gửi từ form hoặc URL client
- `$_FILES`: chứa thông tin các tập tin gửi từ client.
- `$_SERVERS`: `$_SERVER`: Mảng này chứa thông tin về ngữ cảnh mà đoạn mã đang chạy, như là tên server, tên file đang chạy... Dữ liệu mảng này được tạo bởi web server.
- `$_SESSION` & `$_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 dữ 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

```
<?php
```

```
echo $_GET['a'] + $_GET['b'];
```

```
?>
```

```
<?php
```

```
echo $_REQUEST['a'] + $_REQUEST['b'];
```

```
?>
```


GỬI NHẬN DỮ LIỆU QUA FORM GET

a:

b:

```
<form name="frm" method="get" action="cong.php">  
  a: <input type="text" name="a" /><br />  
  b: <input type="text" name="b" /><br />  
  <input type="submit" name="submit" value="Cộng" />  
</form>
```

www.abc.com/cong.php?a=1&b=2

<?php

echo \$_GET['a'] + \$_GET['b'];

?>

<?php

echo \$_REQUEST['a'] + \$_REQUEST['b'];

?>



GỬI NHẬN DỮ LIỆU QUA FORM POST

a: b:

```
<form name="frm" method="post" action="cong.php">
  a: <input type="text" name="a" /><br />
  b: <input type="text" name="b" /><br />
  <input type="submit" name="submit" value="Cộng" />
</form>
```

← → ↻ 🏠 📄 www.abc.com/cong.php

```
<?php
    echo $_POST['a'] + $_POST['b'];
?>
```

```
<?php
    echo $_REQUEST['a'] + $_REQUEST['b'];
?>
```

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

```
<input type="text" name="id"
value="user01" />
```

```
<input type="password" name="pass"
value="123456"/>
```

```
<input type="hidden" name="action"
value="login" />
```

Mã php nhận kết quả từ form

```
$user = $_POST["id"]; // $user
="user01"
```

```
$p = $_REQUEST["pass"];
// $p = "123456"
```

```
$action = $_POST["action"];
// action="login"
```

Nhận dữ liệu từ nhóm radio button

Mã HTML

```
<input type="radio" name="align"
value="left" /> Trái
<input type="radio" name="align"
value="center" /> Giữa
<input type="radio" name="align"
value="right" /> Phải
```

PHP

```
if (isset($_POST["align"]))
{
    echo "Chọn:" .
    $_POST["align"];
}
else
{
    echo "Chưa chọn";
}
```

☐ Trái ☐ Giữa ☐ Phải

Nhận dữ liệu từ nhóm checkbox button

Mã HTML

Chọn các màu bạn thích
`<input type="checkbox" name="color[]" value="Red" />Đỏ`
`<input type="checkbox" name="color[]" value="Blue" />Xanh`
`<input type="checkbox" name="color[]" value="Yellow" />Vàng`

PHP

```
if (isset($_POST["color"]))  
{  
    $arr = $_POST["color"];  
    foreach($arr as $item)  
        echo "Chọn:" . $item;  
}  
else  
{  
    echo "Bạn chưa chọn";  
}
```

Chọn các màu bạn thích

☐ Đỏ ☐ Xanh ☐ Vàng

Chọn dữ liệu từ dropdown list

Mã HTML

```
<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>
```

PHP

```
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

```
<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>
```

PHP

```
$_POST["vitri"]
```

Chọn vị trí ▼

Chọn vị trí

Trái

Giữa

Phải

Gửi tập tin

- Form phải có thuộc 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)

```
<?php
if (isset($_FILES["img"]))
{
    if ($_FILES["img"]["error"] !=0)
    {echo "Error upload!";
    exit;
    }

    $src = $_FILES["img"]["tmp_name"];
    $des = $_FILES["img"]["name"];
    if (move_uploaded_file($src, $des))
    {
        echo "Đã lưu file :";
    }
    else {
        echo "Lỗi lưu file!";
    }
}
?>
```

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: $root<br>";
```

```
$r = $_SERVER["HTTP_REFERER"]; echo " HTTP_REFERER:  
$r<br>";
```

```
$url = $_SERVER["REQUEST_URI"]; echo " URL: ".$url."<br>";
```

```
$sname = $_SERVER["SERVER_NAME"]; echo "Server  
Name:$sname<br>";
```

```
$ips = $_SERVER["SERVER_ADDR"]; echo " IP Server:  
".$ips."<br>";
```

```
$br = $_SERVER["HTTP_USER_AGENT"]; echo "Client:  
".$br."<br>";
```

```
?>
```



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 `$_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 \$_SESSION (tt)

- Mảng này được sử dụng để xây dựng các chức năng: quản lý đăng nhập, xây dựng giỏ 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 (tt)

- 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"].

a.php

```
<?php
if (!isset($_SESSION)) session_start();
if (isset($_GET["reset"]) &&
$_GET["reset"]==1)
    unset($_SESSION["n"]);
if (isset($_SESSION["n"]))
    $_SESSION["n"]++;
else $_SESSION["n"]=1;

echo "ket qua n:". $_SESSION["n"];
?><hr>
<a href="b.php">Trang b</a> <br>
```

b.php

```
<?php
if (!isset($_SESSION)) session_start();
if (isset($_GET["reset"]) &&
$_GET["reset"]==1)
    unset($_SESSION["n"]);
if (isset($_SESSION["n"]))
    $_SESSION["n"]++;
else $_SESSION["n"]=1;

echo "ket qua n:". $_SESSION["n"];
?><hr>
<a href="a.php">Trang a</a> <br>
```

```
<a href="a.php?reset=1">Reset</a>
```

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 gửi 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 gửi tới browser, do vậy 2 website khác nhau (cho dù cùng host trên 1 server) sẽ có 2 cookie khác nhau gửi 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

a.php

```
<?php
```

```
date_default_timezone_set("Asia/Ho_Chi_Minh");
```

```
//print_r($_COOKIE);
```

```
$s="";
```

```
if (isset($_COOKIE["name"])) {
```

```
    $name = $_COOKIE["name"];
```

```
    $lasttime = date("l jS \of F Y h:i:s A",  
$_COOKIE["time"]);
```

```
    $s = "Chao ban: $name , ban da truy cap lan cuoi  
vao: $lasttime ";
```

```
}
```

```
setcookie("name", "abc", time()+ 60*60);//60 phút
```

```
setcookie("time", time(), time()+60*60);
```

```
echo $s;
```

```
?>
```

```
<hr><a href="b.php">Trang b</a>
```

b.php

```
<?php
```

```
date_default_timezone_set("Asia/Ho_Chi_Minh");
```

```
//print_r($_COOKIE);
```

```
$s="";
```

```
if (isset($_COOKIE["name"])) {
```

```
    $name = $_COOKIE["name"];
```

```
    $lasttime = date("l jS \of F Y h:i:s A",  
$_COOKIE["time"]);
```

```
    $s = "Chao ban: $name , ban da truy cap lan  
cuoi vao: $lasttime ";
```

```
}
```

```
setcookie("name", "abc", time()+ 60*60);
```

```
setcookie("time", time(), time()+60*60);
```

```
echo $s;
```

```
?>
```

```
<hr><a href="a.php">Trang a</a>
```


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



Chương 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) . "<br>"; //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 addslashes, 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 giải 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 (tt)

- `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($str)`** — Thay thế ký tự xuống dòng (`\n`) bằng thẻ xuống dòng (`
`) trong chuỗi `$str`. Hàm này đừng 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 , $subject, &$amp;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 \$length 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 ($str)`: Viết hoa ở mỗi ký tự đầu tiên của từ.
 - `$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, hợp 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, $match))
{
    echo "Result: ";
    print_r($match);
}
else
{
    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 . chỉ đại diện cho một ký tự!
+	<p>Lặp lại ký tự hay cụm ký tự đứng trước nó (≥ 1)</p> <p>Ví dụ:</p> <p>\$pattern = "/a+/" : nghĩa là lặp lại ký tự a và phải có ít nhất là một ký tự a.</p> <ul style="list-style-type: none">- 123a: 1(đúng)- 123aa: 1(đúng)- 123 : 0 (sai). vì không có chữ a nào được lặp. <p>\$pattern = "/ax+/" : biểu diễn cho dòng: ax, axx, axxxx - nhưng không thể là "a" vì + đại diện cho ≥ 1 ký tự.</p>

Ý nghĩa các special symbols

Ký tự đặc biệt	Ý Nghĩa
*	<p>Cũng là lặp lại ký tự hay cụm ký tự đứng trước nó (≥ 0)</p> <p>Ví dụ: a^* : nghĩa là lặp lại chữ a, nếu không có chữ a cũng không sao.</p> <p>-123a: (1)</p> <p>-123aaa: (1)</p> <p>-123: (1). Không có chữ a cũng không sao.</p> <p>\$pattern="/ay*/" : biểu diễn cho dòng: a, ay, ayy, ayyyyyyyyyy,....</p>
?	<p>Tồn tại hay không tồn tại ký tự hay cụm ký tự đứng trước nó:</p> <p>\$pattern="/ab? /" :biểu diễn cho: a, ab (b có thể có hoặc không)</p>
\	<p>Dấu \ đi kèm với 1 meta symbol sẽ làm mất ý nghĩa của meta symbol đó - trả về symbol bình thường.</p> <p>\$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.</p>
\d	<p>Biểu diễn cho một số bất kỳ: 0-9:</p> <p>\$p = "/^\d/"; Tìm chuỗi bắt đầu bằng một ký tự số</p>

Ý nghĩa các special symbols

Ký tự đặc biệt	Ý Nghĩa
<code>\D</code>	Biểu diễn một ký tự không phải số (ngược với <code>\d</code>):
<code>\w</code>	Biểu diễn một ký tự bất kỳ: a-z, A-Z, 0-9
<code>\W</code>	Biểu diễn một ký tự đặc biệt (không phải a-z, A-Z, 0-9)
<code>\s+</code>	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}"/; Thích hợp với: (08)66780052, 66780052, (056) 3821013

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
```

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

```
$s="Monday Tuesday Wednesday";
```

```
preg_match($p, $s, $result);
```

```
print_r($result);
```

```
?>
```

Ví dụ

- Tách chuỗi, ta có thể sử dụng hàm `preg_match`.
Hàm 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 được 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.
- Nên ưu tiên các hàm xử lý chuỗi thông thường vì tốc độ xử lý nhanh hơn.



Chương 5:

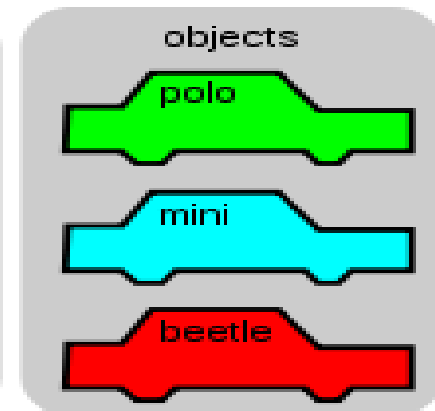
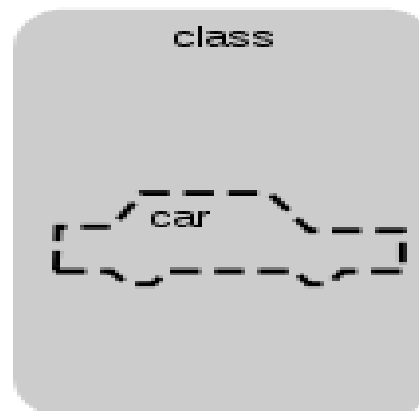
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PHP

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

Ví dụ

```
<?php
```

```
Class A{
```

```
    Public $var1;
```

```
    Private $var2, $var3=5;
```

```
    Protected $var4;
```

```
    Public function F()
```

```
    {
```

```
        echo "Funcion
```

```
A<br>";
```

```
        echo $this->var3;
```

```
    }
```

```
}
```

```
$x= new A();
```

```
$x->F();
```

Định nghĩa class A

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 debug đã sử dụng trong mảng là print_r hoặc var_dump;
- Ví dụ:
 - \$obj = new student("sv001");
//tạo thực thể đối tượng student. Hàm __construct(\$id) sẽ tự động được thực thi.
 - \$obj->show();//Gọi phương thức public show();

Ví dụ

```
Class student{  
    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 một 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;

}

Ví dụ

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


Ví dụ

```
<?php
class A {
    function funct1() {
        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



Chương 6:

MYSQL

Nội dung

- Tổng quan mysql
- Cài đặt
- Các kiểu dữ liệu, kiểu table trong mysql
- Các hàm hay sử dụng trong mysql
- Quản trị mysql bằng công cụ phpmyadmin
 - Tạo csdl
 - Tạo table
 - Tạo relationship giữa các table
 - Tìm kiếm, thêm, sửa, xóa dữ liệu
 - Sao lưu dữ liệu
 - Xóa CSDL và phục hồi CSDL
- Tạo user và chia sẻ CSDL trong mysql
- Thủ tục lưu trữ và hàm trong mysql.

Tổng quan mysql

- Mysql là:
 - Hệ quản trị cơ sở dữ liệu quan hệ
 - Dùng cho các ứng dụng vừa và nhỏ
 - Hỗ trợ chuẩn SQL
 - Phần mềm mã nguồn mở, miễn phí
 - Chạy trên nhiều platforms (Unix, Linux, Windows)
 - Đơn giản, tốc độ nhanh, ổn định
 - Phổ biến
 - Có tính bảo mật cao, chia sẻ truy cập cho nhiều đối tượng khác nhau.

Tổng quan mysql (tt)

- Các tính năng cung cấp:
 - SQL cơ bản (tạo bảng, chèn/xóa/cập nhật mẫu tin, truy vấn, ...)
 - Nhiều tính năng tiên tiến của SQL
 - Những câu truy vấn phức tạp
 - Ràng buộc khóa, ràng buộc dữ liệu, Trigger
 - View (bảng ảo)
 - Nhiều thư viện hàm
 - Stored procedure

Cài đặt

- Nếu sử dụng wamp server: Mysql được đặt trong c:\wamp\bin\mysql
- Cài đặt độc lập:
 - Chọn version mysql phù hợp, download miễn phí từ
 - Bấm file cài đặt và chạy theo hướng dẫn
- Tài khoản đăng nhập:
 - Mysql có một user có quyền cao nhất là root. Với user này, ta có thể tạo các user và chia sẻ quyền sử dụng database cho những user này.
 - Cần nhớ mật khẩu của user root này. Trong wamp: khi cài đặt, root có pass mặc định là rỗng.

Các kiểu table trong mysql 5

- MySQL hỗ trợ nhiều kiểu bảng dữ liệu hoặc các máy lưu trữ khác nhau để giúp chúng ta tối ưu hóa CSDL của mình
- Các loại bảng hỗ trợ trong mysql 5
 - MyISAM
 - InnoDB
 - Memory
 - Merge
 - ...

Các kiểu table trong mysql 5 (tt)

- MyISAM:
 - Kiểu lưu trữ mặc định
 - Tốc độ thực thi trên bảng cao.
 - Không hỗ trợ transaction và tạo các relationship
 - Thích hợp cho các dự án thực thi với tốc độ cao
- InnoDB
 - Hỗ trợ transaction và relationship
 - hỗ trợ khóa dòng (row level locking)
 - Cần nhiều không gian lưu trữ hơn so với MyISAM

Các kiểu table trong mysql 5

- Memory:
 - Tất cả dữ liệu được lưu trong RAM
 - Tốc độ truy xuất dữ liệu cực nhanh.
 - Dữ liệu mất khi tắt máy
 - Chiếm nhiều tài nguyên (RAM) để lưu trữ và thực thi
- CSV
 - Lưu dữ liệu trong file text sử dụng dấu ; để format dữ liệu.
 - Dễ dàng sử dụng để chuyển đổi table này qua lại giữa nhiều phần mềm khác nhau bằng cách import/export file theo format CSV

Kiểu dữ liệu trong mysql

- Kiểu dữ liệu số

| số nguyên | Bytes | Phạm vi giá trị |
|-----------|-------|--|
| TINYINT | 1 | -127 đến 128 hoặc 0 đến 255 |
| SMALLINT | 2 | -32768 đến 32767 hoặc 0 đến 65535 |
| MEDIUMINT | 3 | -8388608 đến 8388607 hoặc 0 đến 16777215 |
| INT | 4 | -2147483648 đến 2147483647 hoặc 0 đến 4294967295 |
| BIGINT | 8 | -9223372036854775808 đến 9223372036854775807 hoặc 0 đến 18446744073709551615 |

Kiểu dữ liệu trong mysql (tt)

- Kiểu dữ liệu số thực

| Kiểu dữ liệu số thực | | Bytes |
|----------------------|--|--------|
| FLOAT | | 4 |
| DECIMAL | | 5 – 17 |
| DOUBLE | | 8 |
| REAL | | 4 |

Kiểu dữ liệu trong mysql (tt)


| Kiểu dữ liệu | Số byte | Phạm vi giá trị | Mô tả |
|--------------|---------|---|---|
| DATE | 3 | '1000-01-01' – '9999-12-31' | Trả về ngày tháng năm theo định dạng yyyy-mm-dd(năm-tháng-ngày) |
| DATETIME | 8 | '1000-01-01 00:00:00' – '9999-12-31 23:59:59' | Trả về ngày tháng năm kèm với thời gian định dạng YYYY-MM-DD HH:MM:SS (năm-tháng-ngày giờ-phút-giây) |
| TIMESTAMP | 4 | bắt đầu từ 1970-01-01 00:00:00 | Trả về một timestamp(thời điểm một hành động được tạo ra như insert, update,...) theo định dạng YYYY-MM-DD HH:MM:SS |
| TIME | 3 | '-838:59:59' – '838:59:59' | Trả về một thời điểm nhất định, định dạng HH:MM:SS |
| YEAR | 1 | 1901 – 2155 | Trả về một năm nhất định, được viết theo định dạng 2 chữ số(95) hoặc 4 chữ số(1995) |



Kiểu dữ liệu trong mysql

- Kiểu dữ liệu String

| Kiểu dữ liệu | Phạm vi giá trị | Mô tả |
|-----------------|---------------------|--|
| CHAR | 1-255 ký tự | Chứa một chuỗi có độ dài tối đa là 255 ký tự |
| VARCHAR | 1-255 | Chứa một biến chuỗi có độ dài tối đa là 255 ký tự |
| TINYTEXT | 1-255 | Chứa một văn bản có độ dài tối đa là 255 ký tự |
| TEXT | 1-65,535 | Chứa một văn bản có độ dài tối đa là 65,535 ký tự |
| BLOB | 65,535 bytes | Chứa đối tượng nhị phân |
| MEDIUMTEXT | 16,777,215 | Chứa một đoạn văn bản có độ dài tối đa là 16,777,215 ký tự |
| MEDIUMBLOB | 16,777,215 bytes | Chứa đối tượng nhị phân |
| LONGTEXT | 4,294,967,295 | Chứa một đoạn văn bản có độ dài tối đa là 4,294,967,295 ký tự |
| LOBLOB | 4,294,967,295 bytes | Chứa đối tượng nhị phân |
| ENUM(x,y,z,...) | 65535 giá trị | Cho phép bạn tạo ra một danh sách các giá trị tùy chọn phù hợp |



Quản trị mysql

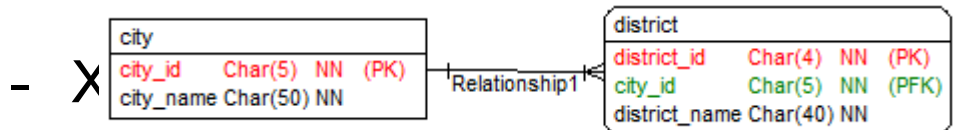
- Các thao tác thông dụng trong quản trị csdl:
 - Tạo, xóa database
 - Thêm, sửa, xóa, tạo các relationship cho các table.
 - Thêm sửa xóa dữ liệu trong table.
 - Backup và phục hồi database
- Các cách quản trị csdl mysql
 - Bằng dòng lệnh cmd
 - Bằng công cụ quản trị mysql miễn phí trên nền web: phpmyadmin
 - Các công cụ khác: Navicat, case studio,...
- Vị trí lưu database của mysql.
 - Sau khi được tạo, database mặc định được lưu trong thư mục data ví dụ: C:\wamp\bin\mysql\mysql5.6.17\data).
 - Mỗi database là một thư mục con trong data.
 - Có thể chỉnh sửa lại đường dẫn lưu database này trong file cấu hình mysql my.ini

Quản trị mysql bằng cmd

- Truy cập vào mysql bằng tài khoản root
- Tạo csdl
- Tạo user sử dụng database vừa tạo
- Tạo các table
- Tạo relationship giữa các table
- Tìm kiếm, thêm, sửa, xóa dữ liệu
- Sao lưu dữ liệu
- Xóa CSDL và phục hồi CSDL

Quản trị csdl bằng cmd

- Ví dụ:
 - tạo csdl region có các thực thể như hình



- X
- Thêm 2 dòng dữ liệu cho table city
- Hiển thị dữ liệu table city
- Tạo 1 user test01 với mật khẩu 123456 sử dụng database vừa tạo.
- Đăng nhập vào database vừa tạo với user test01
- Xóa database region

Quản trị mysql bằng phpmyadmin

- Phpmyadmin là công cụ quản trị mysql thông dụng, miễn phí chạy trên nền web php.
- Có thể tải về miễn phí hoặc có sẵn khi cài đặt wamp (xamp, appsev,...)
- Là công cụ quản lý csdl trực quan, dễ dàng và được hầu hết các hosting cài đặt cho người sử dụng host thao tác với database.

Quản trị mysql bằng phpmyadmin

phpMyAdmin

Current Server: phpMyAdmin demo - MySQL

(Recent tables) ...

filter databases by name

<< 3

- Usuarios
- Usuarios1
- VSet
- VsetiAdmin
- Xss
- uam
- uam2
- ube_db
- victoria base
- vseti
- world
 - New
 - City
 - Country
 - CountryLanguage

phpMyAdmin demo - MySQL » world » City

Browse Structure SQL Search Insert Export Import Operations More

| # | Name | Type | Collation | Attributes | Null | Default | Extra | Action |
|--------------------------|---------------|----------|-------------------|------------|------|---------|----------------|--|
| <input type="checkbox"/> | 1 ID | int(11) | | | No | None | AUTO INCREMENT | Change Drop More |
| <input type="checkbox"/> | 2 Name | char(35) | latin1_swedish_ci | | No | | | Change Drop More |
| <input type="checkbox"/> | 3 CountryCode | char(3) | latin1_swedish_ci | | No | | | Change Drop More |
| <input type="checkbox"/> | 4 District | char(20) | latin1_swedish_ci | | No | | | Change Drop More |
| <input type="checkbox"/> | 5 Population | int(11) | | | No | 0 | | Change Drop More |

☐ Check All With selected: [Browse](#) [Change](#) [Drop](#) [Primary](#) [Unique](#) [Index](#)

[Spatial](#) [Fulltext](#)

[Print view](#) [Relation view](#) [Propose table structure](#) [Track table](#) [Move columns](#)

[Add](#) column(s) ☒ At End of Table ☐ At Beginning of Table ☐ After [Go](#)

[Indexes](#)

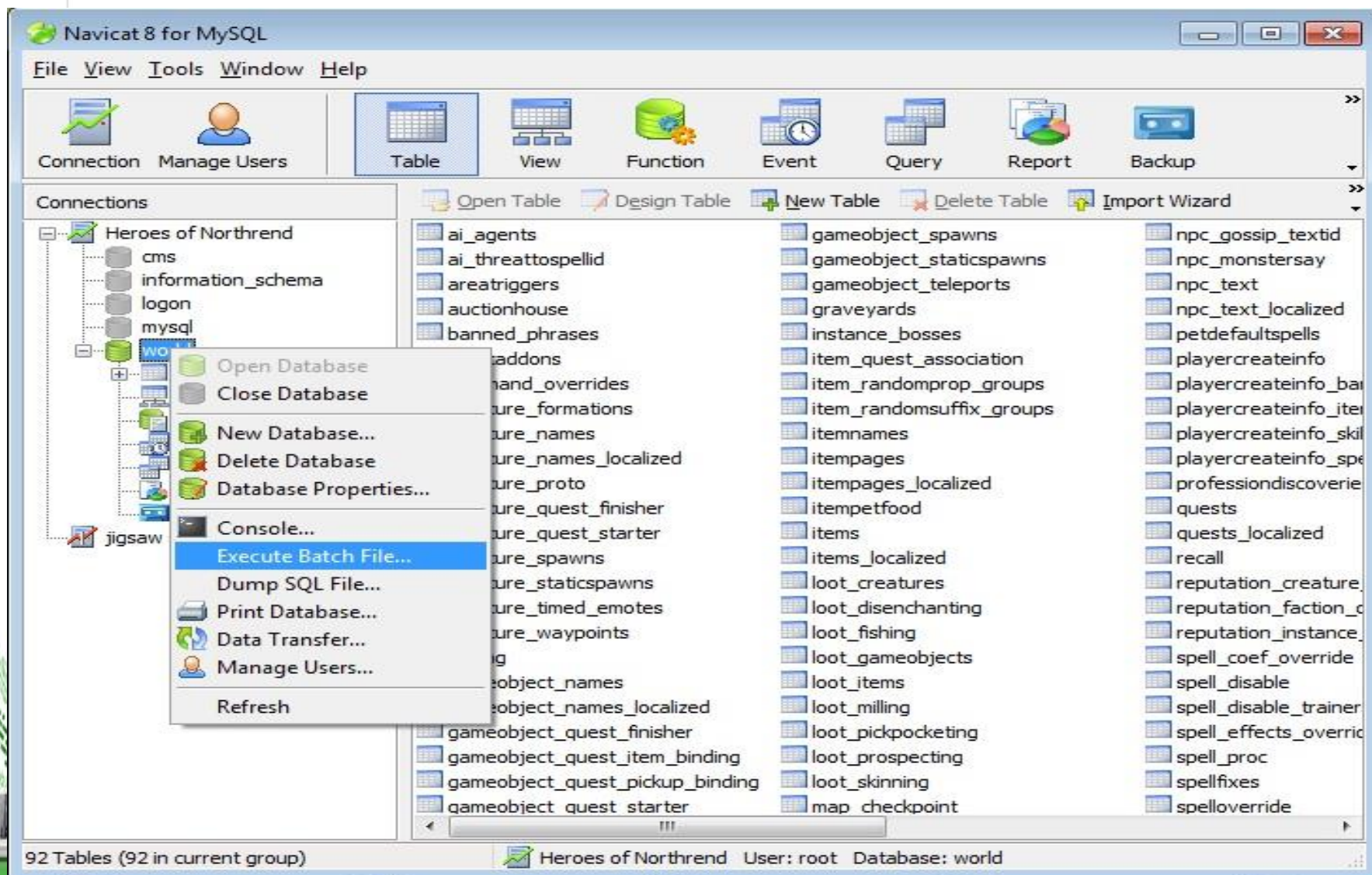
Information

| Space usage | | Row Statistics | |
|-------------|-----------|----------------|--------------------------|
| Data | 266.9 KiB | Format | static |
| Index | 42 KiB | Collation | latin1_swedish_ci |
| Total | 308.9 KiB | Rows | 4,074 |
| | | Row length | 67 |
| | | Row size | 78 B |
| | | Next autoindex | 4,080 |
| | | Creation | Apr 03, 2013 at 01:30 PM |
| | | Last update | Apr 03, 2013 at 01:30 PM |

Quản trị mysql bằng navicat

- Navicat là phần mềm có phí, cài đặt trên client.
- Là công cụ mạnh mẽ để quản lý nhiều hệ quản trị csdl khác nhau (mysql, sql server, oracle, sqlite, ...)
- Có nhiều chức năng mạnh mẽ: backup, đồng bộ hóa database, import/export giữa nhiều loại data khác nhau vào database (excel, csv, xml, ...).

Quản trị mysql bằng navicat



Stored procedure trong mysql

- Giới thiệu stored procedure
- Tạo các stored procedure cơ bản
 - Bằng dòng lệnh
 - Bằng tool
- Tham số truyền vào trong stored procedure
 - Tham số đầu vào
 - Tham số đầu ra
- Kết quả trả về
 - Của lệnh return
 - Của câu truy vấn select

Stored procedure trong mysql

- Giới thiệu

- Stored procedure là một nhóm các lệnh SQL được lưu trong sql server, có thể được gọi ra để thực thi một nhiệm vụ nào đó.
- Stored procedure cho phép truyền tham số, và có thể được gọi bởi nhiều Client qua mạng với các tham số khác nhau.

Stored procedure trong mysql

- Lợi ích của Stored Procedure
 - Giảm lưu lượng dữ liệu được truyền qua mạng và tăng hiệu năng (Performance).
 - Dễ bảo trì
 - Tăng khả năng bảo mật.
 - Tái sử dụng Code.
- Các loại stored procedure
 - Procedure
 - Function

Tạo Stored Procedure bằng lệnh

- Ta có thể tạo stored procedure trong:
 - Dòng lệnh
 - Phpmyadmin
 - Trong các công cụ quản trị mysql khác

Tạo Stored Procedure bằng lệnh

- Cú pháp tạo stored procedure:
 - Script trong dòng lệnh
 - Script trong Phpmyadmin
 - Trong các công cụ quản trị mysql khác

- Picking delimiter
- Create procedure `proce_name()`
- Begin
- end





Chương 7:

PHP DATA OBJECT



Nội dung

- Giới thiệu
- Cài đặt
- Lớp PDO và PDOStatement
- Các bước làm việc với CSDL bằng php

Giới thiệu PDO

- PDO (PHP Data Object): là API cung cấp cho chúng ta để kết nối cũng như thao tác với CSDL, chuyển dữ liệu thành các đối tượng để chúng ta dễ thao tác.
- PDO hỗ trợ thao tác trên nhiều CSDL khác nhau.
- Sử dụng PDO, chúng ta không phải quan tâm nhiều tới csdl nào chúng ta sử dụng để lưu trữ. Khi thay đổi CSDL, ta cũng không cần thay đổi code đã viết.

Cài đặt PDO

- Để sử dụng PDO, chúng ta load thư viện PDO với csdl tương ứng:
 - Mở file php.ini
 - Tìm dòng thư viện pdo thao tác với csdl tương ứng và gỡ bỏ chú thích. Ví dụ, thao tác với mysql, gỡ # trước dòng `extension=php_pdo_mysql.dll`.
 - Lưu lại
 - Khởi động lại webserver (apache)

Thao tác với CSDL

1. Kết nối tới CSDL
2. Chuẩn bị câu truy vấn
3. Thực thi truy vấn
4. Xử lý kết quả
5. Đóng kết nối



Kết nối tới CSDL

- Sử dụng hàm `__construct` của class PDO
 - ✓ `public PDO::__construct (string $dsn [, string $username [, string $password [, array $options]]])`
 - ✓ `$dsn`: (Data Source Name): chứa các thông tin yêu cầu cho việc kết nối tới database.
 - ✓ `$username` , `$password`: Username và mật khẩu để kết nối tới database này.
 - ✓ `$options`: Một số lựa chọn khi kết nối.
 - ✓ Kết nối thành công, trả về đối tượng PDO.
- Đặt câu lệnh trong cặp `try...catch`

Ví dụ

- DSN kết nối tới mysql:
 - `$dsnmysql ="
mysql:host=localhost;dbname=bookstore" ;`
- DSN kết nối tới ODBC (ví dụ Ms Access)
 - `$dsnodbc=" odbc:Driver={Microsoft Access
Driver (*.mdb)}; Dbq=c:\\ bookstore.mdb;" ;`
- DSN kết nối tới sqlite:
 - `$dsnsqlite=" sqlite:c:\\bookstore.sqlite.sqlite"
;`

Ví dụ

```
<?php
    $dsn =" mysql:host=localhost; dbname=bookstore" ;
    $user  = " root" ;   $pass  = " " ;
    try{
        $dbh = new PDO($dsn, $user, $pass);
        $dbh->query(" set names 'utf8' " );
    }
    catch(Exception $exc)
    {
        echo $exc->getMessage();   exit;
    }
?>
```

- Kết quả:
 - Thành công, trả về đối tượng PDO \$dbh
 - Thất bại: In báo lỗi trong Exception và dừng ứng dụng

Chuẩn bị và thực thi câu sql đơn giản

- Sau khi kết nối csdl thành công, sử dụng phương thức prepare trong PDO để đưa vào một chuỗi sql. Kết quả của phương thức này là một đối tượng PDOStatement nếu thành công hoặc false nếu không thành công.
- Sau khi đối tượng PDOStatement được tạo ra, ta có thể thiết lập và chạy các câu truy vấn bằng phương thức trong PDOStatement :
 - query(): khi câu sql không có biến truyền vào hoặc dữ liệu truyền vào được kiểm tra kỹ.
 - execute: khi câu sql có dữ liệu truyền vào (hoặc không).

Xử lý kết quả

- Sau khi thực thi sql (bằng `->query` hoặc `->execute`), thông tin kết quả được lưu trong các thuộc tính của đối tượng `PDOStatement`. Ta có thể nhận kết quả này để xử lý.
- `PDOStatement->rowCount()`: Trả về số dòng trả về (truy vấn `select`) hoặc số dòng bị tác động (các truy vấn `update`, `insert`, `delete`).
- `PDOStatement->fetchAll ([int $fetch_style])`: Trả về mảng các giá trị truy vấn trong câu truy vấn `select` tùy thuộc vào `$fetch_style`. `$fetch_style` là một giá trị `int` (giá trị của hằng số được định nghĩa trong PDO).

Xử lý kết quả (tt)

- Các giá trị hằng PDO:
 - PDO::FETCH_BOTH: (Mặc định). Kết quả dạng mảng chỉ số là số nguyên và tên cột
 - PDO::FETCH_ASSOC: Mảng kết quả chỉ số là cột trong bảng.
 - PDO::FETCH_COLUMN: Mảng các giá trị cột đầu tiên trong kết quả
 - PDO::FETCH_CLASS: mảng các class là các stdClass có các thuộc tính là các cột của chuỗi sql
 - PDO::FETCH_NUM: trả về mảng chứa chỉ số các cột là số bắt đầu từ 0.
- Dựa vào kết quả trả về là dạng mảng, duyệt qua mảng này bằng foreach để lấy giá trị từng hàng, cột ra xử lý.

Ví dụ

```
<?php
try{
$pdh = new PDO(" mysql:host=localhost; dbname=bookstore" , " root" , " " );
$pdh->query(" set names 'utf8'" );
}
catch(Exception $e){
    echo $e->getMessage(); exit;
}

$stmt = $pdh->query(" select * from loai" );
echo " Số dòng:" . $stmt->rowCount();
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
foreach($rows as $row)
{
    echo $row["maloi" ] . " -" . $row[" tenloai" ] . " <br>" ;
}
?>
```

Ví dụ

```
<?php
try{
$pdh = new PDO(" mysql:host=localhost; dbname=bookstore" , "
    root" , " " );
$pdh->query(" set names 'utf8'" );
}
catch(Exception $e){
    echo $e->getMessage(); exit;
}
$stmt = $pdh->prepare(" select * from loai " );
$stmt->execute();
echo " Số dòng:" . $stmt->rowCount();
print_r($stmt->fetchAll(PDO::FETCH_ASSOC));
?>
```

Các truy vấn có truyền dữ liệu

- Thiết lập sql PDO->prepare() cho các truy vấn có tham số
- Sử dụng phương thức PDOStatement->bindParam() hoặc PDOStatement->bindValue() để truyền dữ liệu vào các tham số này.

```
public bool PDOStatement::bindParam ( mixed $parameter , mixed  
    &$variable [, int $data_type = PDO::PARAM_STR [, int $length [,  
    mixed $driver_options ]]] ).
```

- \$parameter: Giá trị của thành phần trong truy vấn theo mẫu :tên trong truy vấn.
- \$value: Giá trị được truyền vào thành phần này.
- \$data_type: Kiểu giá trị cho thành phần này, sử dụng các hằng số đã định nghĩa trong PDO class.
- Thực thi truy vấn: bằng phương thức PDOStatement->execute();
- Ta cũng có thể tạo một mảng chứa các tham số và giá trị, sau đó truyền mảng vào phương thức execute()

Dữ liệu truyền vào sql

- Cách 1: Truyền trực tiếp vào truy vấn:

```
<?php
```

```
$t="%php%";
```

```
$sql="select * from sach where tensach like '$t' ";
```

```
$stm = $dbh->prepare($sql);
```

```
$stm->execute();
```

```
echo "Tìm được:" . $stm->rowCount() . " cuốn sách";
```

```
$rows = $stm->fetchAll(PDO::FETCH_ASSOC);
```

```
Print_r($rows);
```

```
?>
```


Dữ liệu truyền vào sql

- Cách 2_a: Truyền thông qua phương thức bindParam và các tham số không định danh: Các tham số này là các ký hiệu ? trong truy vấn.

```
<?php //Connect....
```

```
$ten = "%php%"; $maloai="TH";
```

```
$sql="select * from sach where tensach like ? and maloai =? ";
```

```
$stm = $dbh->prepare($sql);
```

```
$stm->bindParam(1, $ten); //Truyền $ten vào vị trí ? đầu tiên
```

```
$stm->bindParam(2, $maloai); //Truyền $maloai vào vị trí ? thứ 2
```

```
$stm->execute();
```

```
echo "Tìm được:" . $stm->rowCount() . " cuốn sách";
```

```
$rows = $stm->fetchAll(PDO::FETCH_ASSOC);
```

```
Print_r($rows);
```

```
?>
```

Dữ liệu truyền vào sql

- Cách 2_b: Truyền thông qua phương thức bindParam và các tham số định danh: Các tham số này là các chuỗi ký hiệu bắt đầu bằng : trong truy vấn.

```
<?php //Connect....
```

```
$ten = "%php%"; $maloai = "TH";
```

```
$sql = "select * from sach where tensach like :tensach and maloai = :maloai ";
```

```
$stm = $dbh->prepare($sql);
```

```
$stm->bindParam(":tensach", $ten);
```

```
//Truyền $ten vào vị trí :tensach trong $sql
```

```
$stm->bindParam(":maloai", $maloai);
```

```
//Truyền $maloai vào vị trí :maloai trong $sql
```

```
$stm->execute();
```

```
echo "Tìm được:" . $stm->rowCount() . " cuốn sách";
```

```
$rows = $stm->fetchAll(PDO::FETCH_ASSOC);
```

```
Print_r($rows);
```

```
?>
```

Dữ liệu truyền vào sql

- Cách 3: Truyền qua phương thức execute.
- Giống cách số 2, nhưng các tham số định danh hoặc không định danh được đặt trong một mảng và truyền vào qua phương thức execute.
- Số lượng các phần tử của mảng và các phần tử đặt chỗ trước phải khớp nhau (số lượng, tên).
- Mảng không định danh: Thứ tự các phần tử trong mảng phải tương ứng với thứ tự các phần tử đặt chỗ ? trong truy vấn.

Ví dụ 1

```
<?php //connect...
$ten = "%php%"; $maloai="TH";
$sql="select * from sach where tensach like :tensach and
maloai = :maloai ";
$stmt = $dbh->prepare($sql);
$arr = array(":tensach"=>$ten, ":maloai"=>$maloai);
$stmt->execute($arr);
echo "Tìm được:" . $stmt->rowCount() . " cuốn sách";
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
Print_r($rows);
?>
```

Ví dụ 2

```
<?php //connect...
```

```
$ten = "%php%"; $maloai="TH";
```

```
$sql="select * from sach where tensach like ? and maloai = ? ";
```

```
$stm = $dbh->prepare($sql); $arr = array($ten, $maloai);
```

```
$stm->execute($arr);
```

/* Giá trị các phần tử của mảng sẽ lần lượt được chèn vào các vị trí ? trong \$sql theo thứ tự. => Phần tử đầu tiên của mảng sẽ truyền vào vị trí ? đầu tiên trong \$sql.

```
*/
```

```
echo "Tìm được:" . $stm->rowCount() . " cuốn sách";
```

```
$rows = $stm->fetchAll(PDO::FETCH_ASSOC);
```

```
Print_r($rows);
```

```
?>
```

Các ví dụ

- Ví dụ cho các câu truy vấn select
- Ví dụ thêm dữ liệu
- Ví dụ update dữ liệu
- Ví dụ delete dữ liệu

Ví dụ - Select data

```
<?php try {  
$pdh = new PDO(" mysql:host=localhost; dbname=bookstore" , "  
    root" , " " );  
$pdh->query(" set names 'utf8'" );  
}  
catch(Exception $e){ echo $e->getMessage(); exit;}  
$stm = $pdh->prepare(" select * from sach where gia>:gia " );  
$gia =50000;          $stm->bindParam(" :gia" , $gia);  
$stm->execute();      echo " Số dòng:" . $stm->rowCount();  
foreach($stm->fetchAll(PDO::FETCH_ASSOC) as $row)  
{  
    echo " <br>" . $row[" masach" ] ." -" . $row[" tensach" ];  
}
```



Ví dụ - Select data

```
<?php try {  
$pdh = new PDO(" mysql:host=localhost; dbname=bookstore" , "  
    root" , " " );  
$pdh->query(" set names 'utf8'" );  
}  
catch(Exception $e){ echo $e->getMessage(); exit;}  
$stm = $pdh->prepare(" select * from sach where gia>:gia " );  
$gia =50000;          $arr = array(" :gia" =>$gia);  
$stm->execute($arr); echo " Số dòng:" . $stm->rowCount();  
foreach($stm->fetchAll(PDO::FETCH_ASSOC) as $row)  
{  
    echo " <br>" . $row[" masach" ] ." -" . $row[" tensach" ];  
}
```

Ví dụ- Insert data

```
<?php //connect
$sql=" insert into loai(maloai, tenloai) values(:maloai, :tenloai)"
;
$stmt=$dbh->prepare($sql);
$m=" T108" ; $t=" loại 108" ;
$stmt->bindParam(" :maloai" , $m);
$stmt->bindParam(" :tenloai" , $t);
$stmt->execute();
echo " Đã thêm:" . $stmt->rowCount();
$m=" T109" ; $t=" loại 109" ;
$stmt->execute();
echo " Đã thêm:" . $stmt->rowCount();
?>
```

Ví dụ- Insert data

- Ví dụ thêm vào 1 loại sách

```
<?php //connect...
$sql=" insert into loai(maloai, tenloai)
values(:maloai, :tenloai)" ;
$stmt = $dbh->prepare($sql);
$a = array(" :maloai" =>" L99" , " :tenloai"
=>" Loai 99" );
$stmt->execute($a);
echo " Đã thêm:" . $stmt->rowCount() ."
loại" ;
?>
```

Ví dụ- Xóa dữ liệu

```
<?php //connect
$sql="delete from loai where maloai
=:maloai";
$stmt = $dbh->prepare($sql);
$a = array(":"maloai"=>"L99");
$stmt->execute($a);
echo "Đã xóa:" . $stmt->rowCount() . "
loại";
?>
```

Truyền dữ liệu vào truy vấn qua các tham số

- Với các truy vấn like %: các truy vấn like '%string %', ta phải thiết lập value bên ngoài sql như sau: \$var = "%string%" ;
- Câu truy vấn limit: không truyền dữ liệu cho tham số của truy vấn cú pháp limit.

Kết luận

- Nắm vững các kết nối tới các database khác nhau
- Cách viết truy vấn và cách truyền tham số qua truy vấn
- Cách nhận và xử lý dữ liệu sau truy vấn