

TRƯỜNG ĐẠI HỌC QUY NHƠN  
KHOA CÔNG NGHỆ THÔNG TIN

---

# Lập trình ứng dụng Web

# Web Application Programming

Th.S Nguyễn Thị Kim Phượng

# LẬP TRÌNH ỨNG DỤNG WEB

---

- **Thời lượng**

30 tiết lý thuyết + 30 tiết thực hành

- **Đánh giá**

- Bài thực hành
- Bài tập lớn

- **Môi trường**

- PHP & MySQL
- WampServer
- Notepad++, Eclipse for PHP developers (Helios)

# NỘI DUNG

---

- 1. Những khái niệm cơ bản về web**
- 2. Javascript**
- 3. Ngôn ngữ lập trình php**
- 4. Quản lý cookies và session**
- 5. Dịch vụ web-base email – upload - phân trang dữ liệu và bắt lỗi trong ứng dụng**
- 6. Xây dựng ứng dụng web bằng php và mysql**

# Tài Liệu Tham Khảo

---

- “Lập trình ứng dụng Web”, TS. Hồ Văn Lâm, Nguyễn Ngọc Dũng, Nguyễn Thị Kim Phượng, 2019
- “Xây dựng ứng dụng web bằng PHP và MySQL”, Phạm Hữu Khang, NXB Phương Đông.
- “Sử dụng PHP và MySQL Thiết Kế Web Động”, Nguyễn Trường Sinh, NXB Thống Kê.
- “Beginning PHP6, Apache, MySQL Web Development”; Elizabeth Naramore, Jason Gerner, Yann Le Scouarnec, Jeremy Stolz, Michael K.Glass, Timothy Boronczyk.

# NHỮNG KHÁI NIỆM CƠ BẢN

---

1. Giới thiệu về mạng máy tính
2. Internet - Mạng máy tính toàn cầu
3. Các thành phần liên quan đến các ứng dụng Web
4. Giới thiệu về ngôn ngữ đánh dấu siêu văn bản HTML
5. Giới thiệu về Cascading Style Sheets (CSS)

# JAVASCRIPT

---

1. Giới thiệu về JavaScript
2. Lập trình với JavaScript
3. Làm việc với các sự kiện và trình duyệt
4. AJAX và tích hợp phía server
5. JQuery

# NGÔN NGỮ LẬP TRÌNH PHP

---

1. Kiểu dữ liệu mảng
2. Thẻ <form> HTML với PHP
3. Cách sử dụng lại mã lệnh
4. Giới thiệu về MySQL
5. Phát biểu SQL trong MySQL
6. MySQL và PHP
7. SQL Injection

# **QUẢN LÝ COOKIES và SESSION**

---

1. Quản lý Cookies
2. Quản lý Session

# MỘT SỐ DỊCH VỤ

---

**Dịch vụ web-base email – upload - phân trang dữ liệu và bắt lỗi trong ứng dụng**

1. Dịch vụ Web-Base Email
2. Làm việc với file và thư mục
3. Upload file trong PHP
4. Cách tổ chức phân trang cho dữ liệu
5. Phát hiện lỗi và sửa lỗi

# XÂY DỰNG ỨNG DỤNG

---

## XÂY DỰNG ỨNG DỤNG WEB BẰNG PHP và MySQL

1. Các thành phần cơ bản
2. Một số Module

# TỔNG QUAN

---

## Một số khái niệm

- Domain Name (tên miền)
  - Là tên máy chủ gắn với một địa chỉ IP
  - Máy chủ DNS sẽ thực hiện việc ánh xạ khi có yêu cầu truy cập
  - Ví dụ: *qnu.edu.vn* gắn với 203.162.31.116  
vn: Việt Nam (cấp 1) edu: tổ chức giáo dục (cấp 2) qnu: tên cơ quan (cấp 3)
  - Chú ý: tên *localhost* được gắn với IP 127.0.0.1

# MỘT SỐ KHÁI NIỆM

---

- Server (máy chủ)
  - Là máy tính có cấu hình cao, hoạt động ổn định, chuyên cung cấp tài nguyên, dịch vụ cho các máy tính khác.
  - Một máy chủ có thể dùng cho một hay nhiều mục đích. Tên máy chủ thường gắn với mục đích sử dụng, ví dụ:
    - **File server**
    - **Mail server**
    - **Web server**

# MỘT SỐ KHÁI NIỆM (tt)

---

- Client (máy khách)
  - Máy khai thác dịch vụ máy chủ
  - Với mỗi dịch vụ thường có các phần mềm riêng để khai thác
  - Một máy tính vừa có thể là client vừa là server
  - Một máy tính có thể khai thác dịch vụ của chính nó.

# MỘT SỐ KHÁI NIỆM (tt)

---

- Protocol (giao thức)

- Là tập hợp các quy định phải tuân theo để truyền tải thông tin trên mạng.
- Mỗi dịch vụ thường có một giao thức riêng, ví dụ:
  - HTTP: giao thức truyền siêu văn bản
  - FTP: giao thức truyền file
  - SMTP: giao thức gửi email
  - POP3: giao thức lấy bản sao email về client

# MỘT SỐ KHÁI NIỆM (tt)

---

- Port (cổng dịch vụ)

- Là một số nguyên nằm trong khoảng 0-65535
- Dùng để xác định dịch vụ của máy chủ.
- Hai dịch vụ khác nhau phải chiếm các cổng khác nhau, và mỗi dịch vụ có thể chiếm nhiều hơn 1 cổng.
- Một số cổng mặc định

HTTP: 80

SSH: 22

DNS: 53

SMTP: 25

FTP: 21

POP3: 110

# MỘT SỐ KHÁI NIỆM (tt)

---

- URL ()
  - Là chuỗi dùng để xác định vị trí và cách khai thác tài nguyên (file trên mạng).
  - Cấu trúc:  
*giao\_thức://địa\_chỉ\_server hoặc IP\_server/đường\_dẫn/tên\_file*
- Ví dụ: [http://itqnu.vn/tin\\_tuc/index.html](http://itqnu.vn/tin_tuc/index.html)
  - Một số thành phần của URL có thể được bỏ qua
    - Giao thức, cổng: được trình duyệt mặc định
    - Tên file: được server mặc định

# MỘT SỐ KHÁI NIỆM (tt)

---

- **Web browser**
  - Là phần mềm chạy ở client để khai thác dịch vụ web.
  - Ví dụ: Chrome, Firefox, Internet Explorer, Opera,...
- **Web page**
  - là một trang nội dung có thể được viết bằng nhiều ngôn ngữ khác nhau nhưng trả kết quả về client luôn được định dạng bởi ngôn ngữ đánh dấu siêu văn bản HTML.

# MỘT SỐ KHÁI NIỆM (tt)

---

- **Website**
  - là tập hợp các web page (trang web) có nội dung thống nhất phục vụ cho mục đích nào đó.
  - Là tập hợp các web page có cùng domain name.

# PHÂN LOẠI WEBSITE

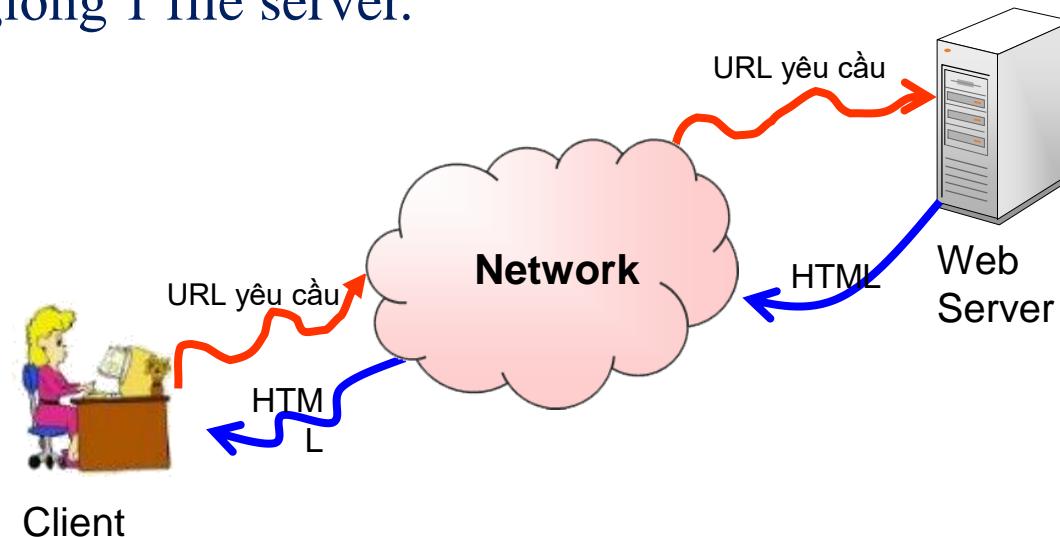
---

Dựa vào công nghệ phát triển, có 2 loại

- Web tĩnh (static web)
  - Viết bằng HTML (kết hợp cùng CSS, JS)
  - Dễ phát triển, chi phí xây dựng thấp, chạy nhanh
  - Tương tác yếu
- Web động (dynamic web)
  - Viết bằng nhiều ngôn ngữ (PHP, ASP, JSP,...)
  - Phát triển phức tạp
  - Tương tác mạnh

# WEB TĨNH

- Mọi người sử dụng nhận được kết quả giống nhau.
- Trang web được viết bằng HTML, chỉ thay đổi khi có sự thay đổi của người xây dựng
- Khả năng tương tác yếu
- Webserver hoạt động giống 1 file server.



# WEB ĐỘNG

---

- Mỗi người sử dụng có thể nhận được nội dung khác nhau phụ thuộc vào kết quả chạy chương trình.
- Trang web viết bằng HTML + Ngôn ngữ lập trình phía server. Có thể được thay đổi bởi người sử dụng
- Khả năng tương tác mạnh

# WEB ĐỘNG (tt)

---

## Một số công nghệ xây dựng web động

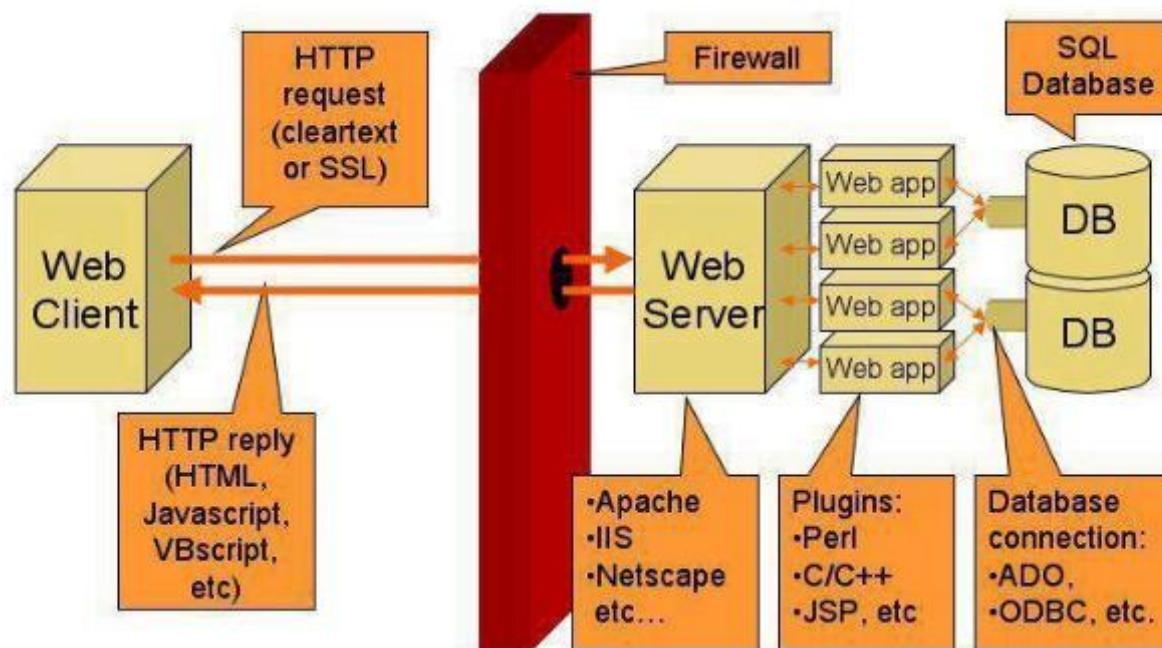
- Client side: Flash, JavaScript, Applet, VBScript chạy ở client

- Server side

- |                                      |                           |
|--------------------------------------|---------------------------|
| - Common Gateway Interface (CGI)     | - Java Server Pages (JSP) |
| - Access Server Pages (ASP, ASP.Net) | - Pert                    |
| - Personal Home Page Tools (PHP)     | - Mã nguồn mở             |

# WEB ĐỘNG (tt)

- Các thành phần trong mô hình web động



# QUY TRÌNH XÂY DỰNG WEBSITE

---

- Web tĩnh



- Web động



# QUY TRÌNH XÂY DỰNG WEBSITE (tt)

---

## Các bước xây dựng Website

1. Đặc tả
2. Phân tích
3. Thiết kế
4. Lập trình
5. Kiểm thử

# QUY TRÌNH XÂY DỰNG WEBSITE (tt)

---

- Đặc tả: Website để làm gì? Ai dùng? Trình độ người dùng? Bố cục? Nội dung? Hình ảnh? v.v...
- Phân tích: Mối liên hệ giữa các nội dung? Thứ tự các nội dung (kịch bản website)
- Thiết kế: Sơ đồ cấu trúc website, giao diện, CSDL, nội dung từng trang, liên kết các trang, v.v...

# QUY TRÌNH XÂY DỰNG WEBSITE (tt)

---

- Lập trình
  - Viết mã lệnh, xây dựng các module, các lớp dùng chung,...
- Kiểm thử
  - Kiểm tra các liên kết
  - Kiểm tra các lỗi bảo mật
  - Kiểm tra hiển thị trên các trình duyệt phổ biến
  - Kiểm tra tốc độ tải trang trên các loại mạng, các loại đường truyền khác nhau,v.v...

# Các phần mềm hỗ trợ

---

- Ngôn ngữ sử dụng: PHP 5.3
- Web server & DBMS: Apache & MySQL
  - WAMP (for Windows), LAMP (for Linux)
  - MAMP (for Mac), SAMP (for Solaris)
  - XAMPP (cross-platform)
- IDE:
  - Eclipse, NetBean,...
  - Dreamweaver, MS Expression,...
  - Notepad++,...

# Download

---

- Download:
  - WAMP: <http://www.wampserver.com>
  - Notepad++: <http://notepad-plus-plus.org>
- Các phần mềm khác
  - Co Rom+: <https://corom.vn>
  - Firefox: <http://www.mozilla.org>
  - XAMPP: <http://www.apachefriends.org/en/xampp.html>

# Cài đặt Webserver

---

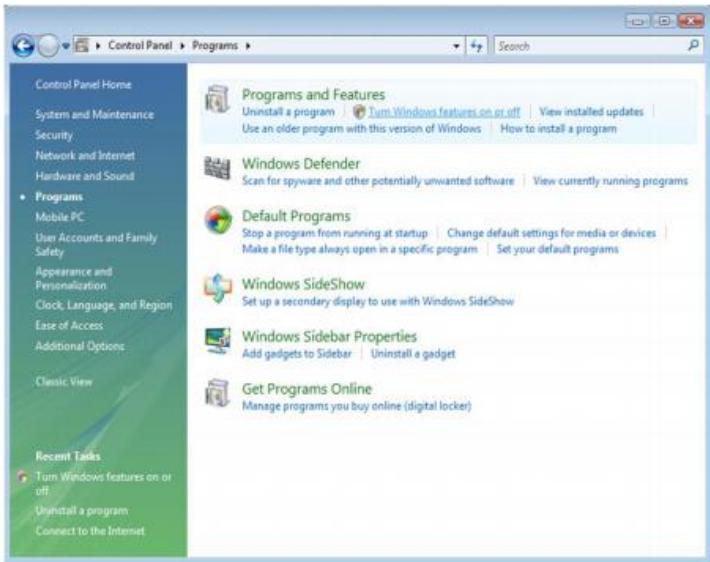
- Cài đặt IIS 7.0

- Click Start Menu -> Control Panel
- Trong Control Panel ta chọn Programs



# Cài đặt IIS

- Trong **Program and Feature** ta chọn **Turn Windows features on or off**



- Double click **Internet Information Services**

# Cài đặt PHP Engine

## Cài đặt PHP Engine (version 5.2.1)

PHP 5.2.1 website <http://www.php.net>

The screenshot shows the PHP download page for version 5.2.1. The top navigation bar includes links for downloads, documentation, faq, getting help, mailing lists, reporting bugs, php.net sites, links, conferences, and my php.net. A search bar allows searching for files in the function list.

**Binaries for other systems**

We do not distribute UNIX/Linux binaries. Most Linux distributions come with PHP these days, so if you do not want to compile your own, go to your distribution's download site. Binaries available on external servers:

- AS/400
- Mac OS X
- Novell NetWare
- OS/2
- RISC OS
- SGI IRIX 6.5.x
- Solaris (SPARC, INTEL)

**Development and archive versions**

Regular source and binary

**PHP 5.2.1**

**Complete Source Code**

- [PHP 5.2.1 \(tar.bz2\)](#) [6,995Kb] - 08 Feb 2007  
md5: 261218e3569a777dbd87c16a15f05c8d
- [PHP 5.2.1 \(tar.gz\)](#) [8,799Kb] - 08 Feb 2007  
md5: 604eaee2b834bb037d2c83e53e300d3f

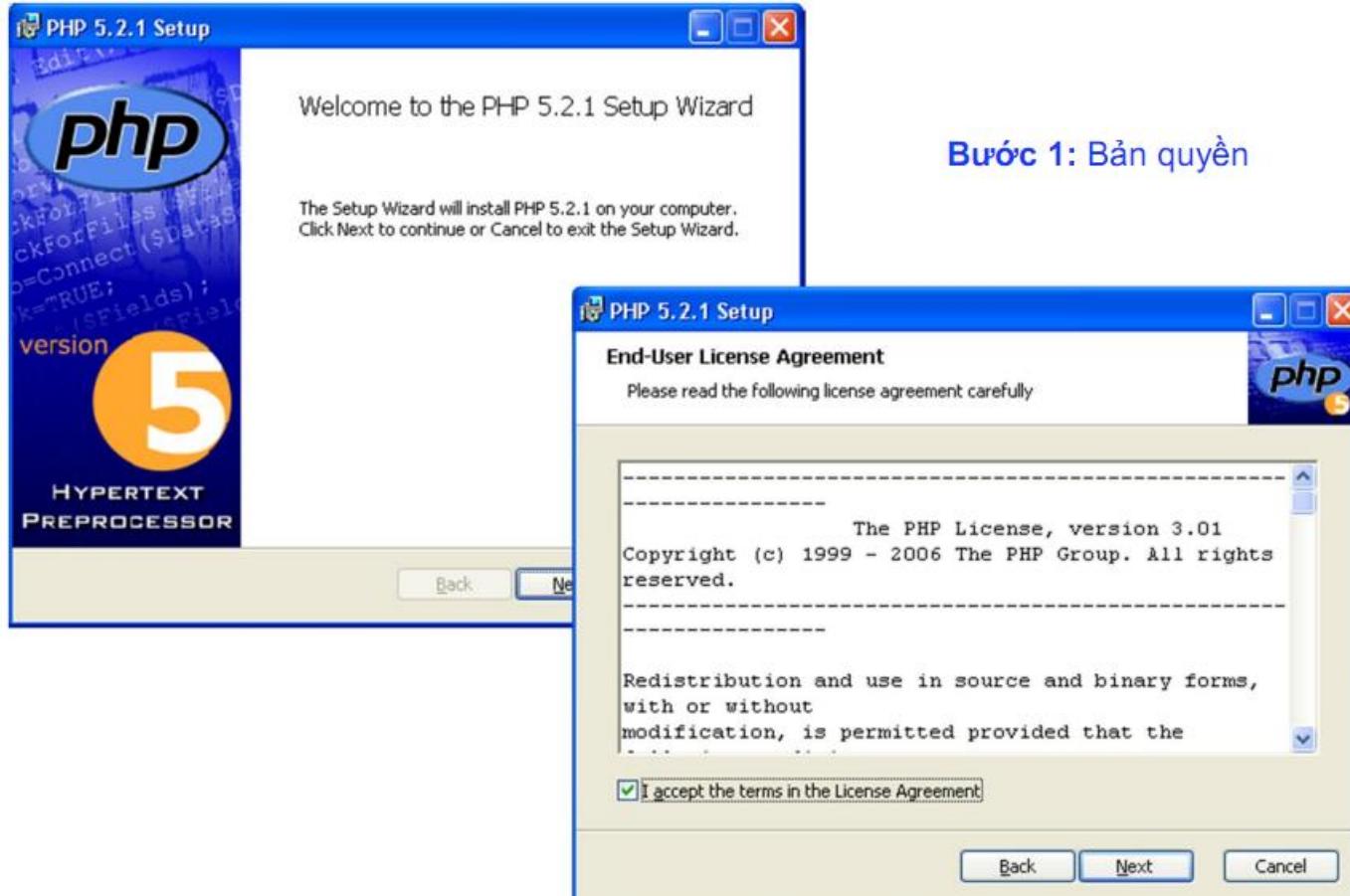
**Windows Binaries**

- [PHP 5.2.1 zip package](#) [9,545Kb] - 08 Feb 2007  
md5: 682dd66fb03c7dd24r522f474e1b04b6
- [PHP 5.2.1 installer](#) [19,568Kb] - 12 Feb 2007  
md5: f0a1445f4adfdc2e00a81b2eb788be5c

**Note:** This file was updated February 12th to fix problems when upgrading from previous PHP versions.

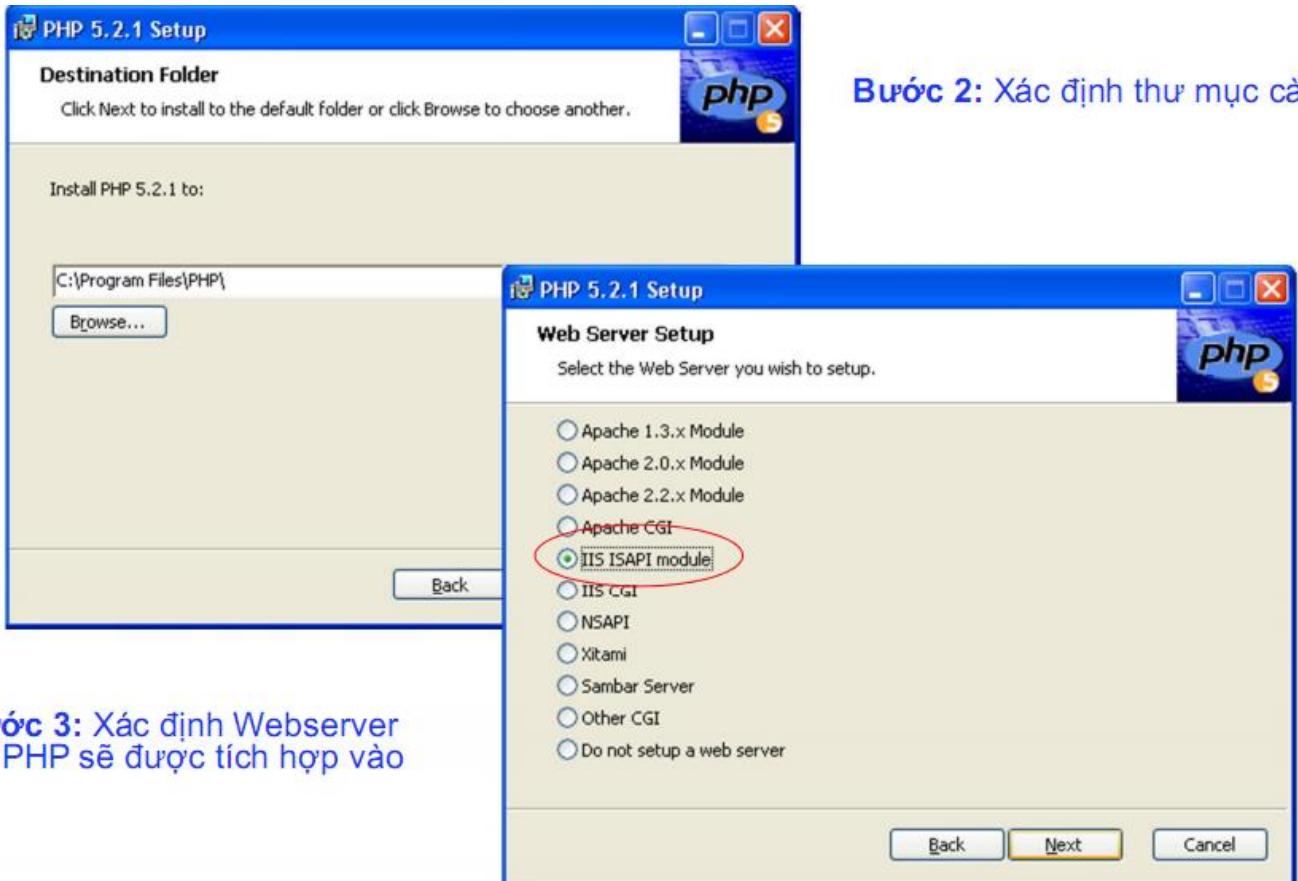
- [PECL 5.2.1 Win32 binaries](#) [2,354Kb] - 08 Feb 2007  
md5: dc8b394146faf7effa6f26df02e8e534

# Cài đặt PHP Engine (tt)



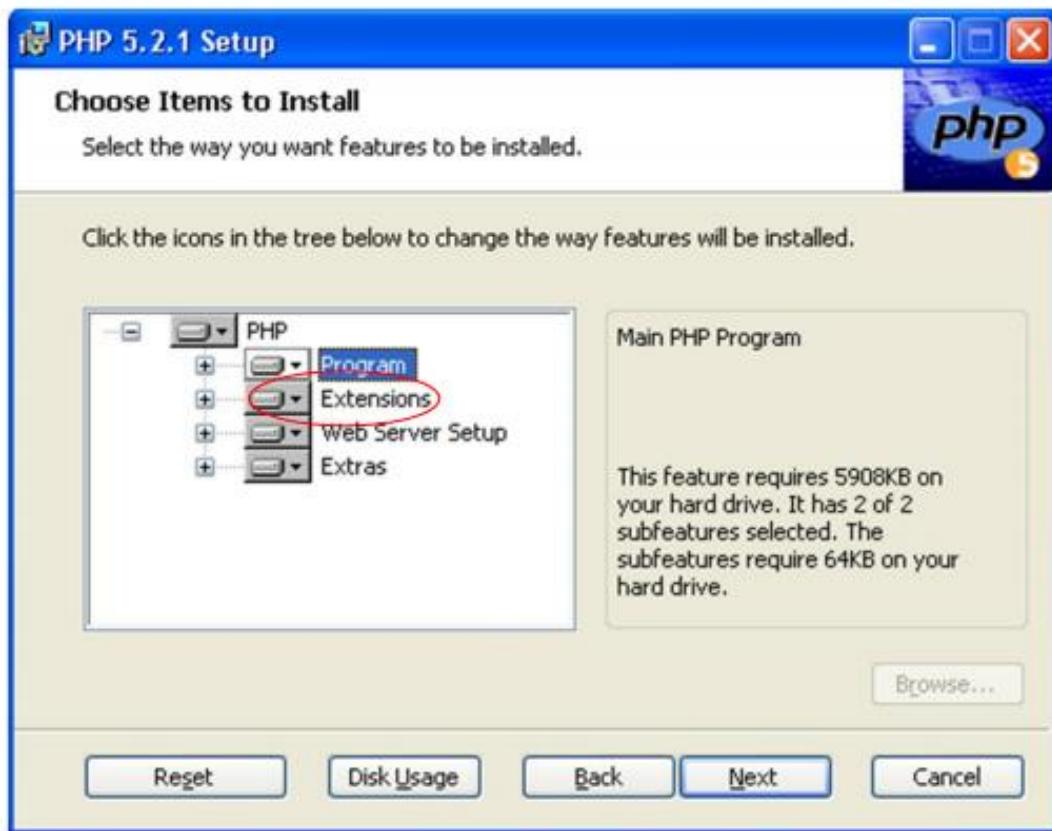
Bước 1: Bản quyền

# Cài đặt PHP Engine (tt)

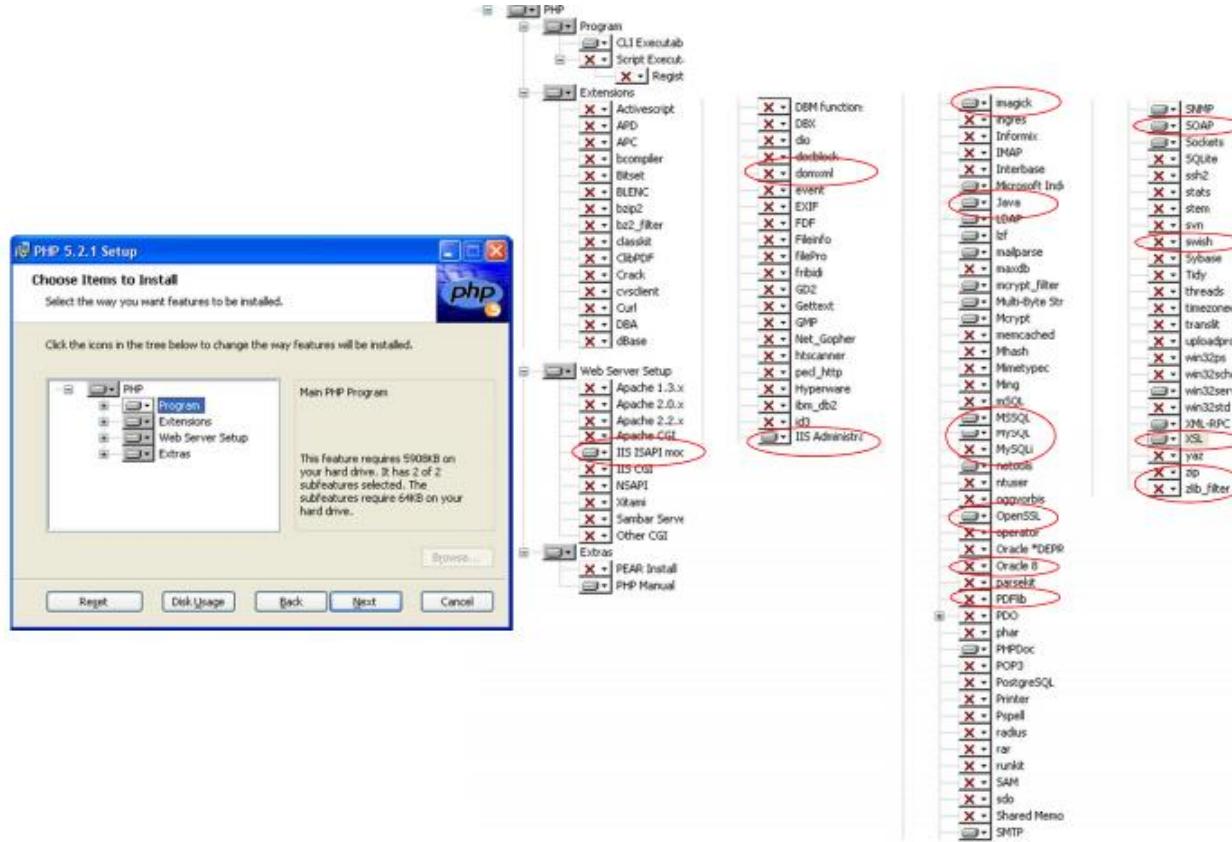


# Cài đặt PHP Engine (tt)

## Bước 4: Xác định các thành phần PHP sẽ cài đặt



# Cài đặt PHP Engine (tt)



# Cài đặt MySQL

---



Database Server - MySQL



# Cài đặt MySQL (tt)

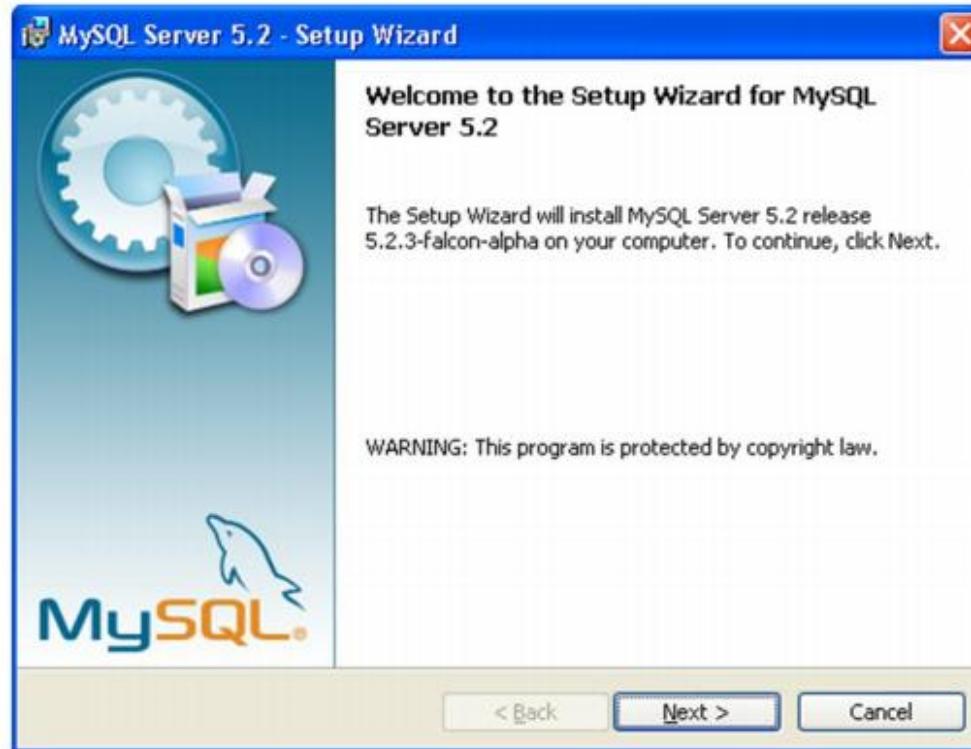
## Cài đặt Database Server – MySQL 5.2

MySQL Server 5.2 Alpha - <http://www.mysql.org/downloads/mysql/5.2.html>

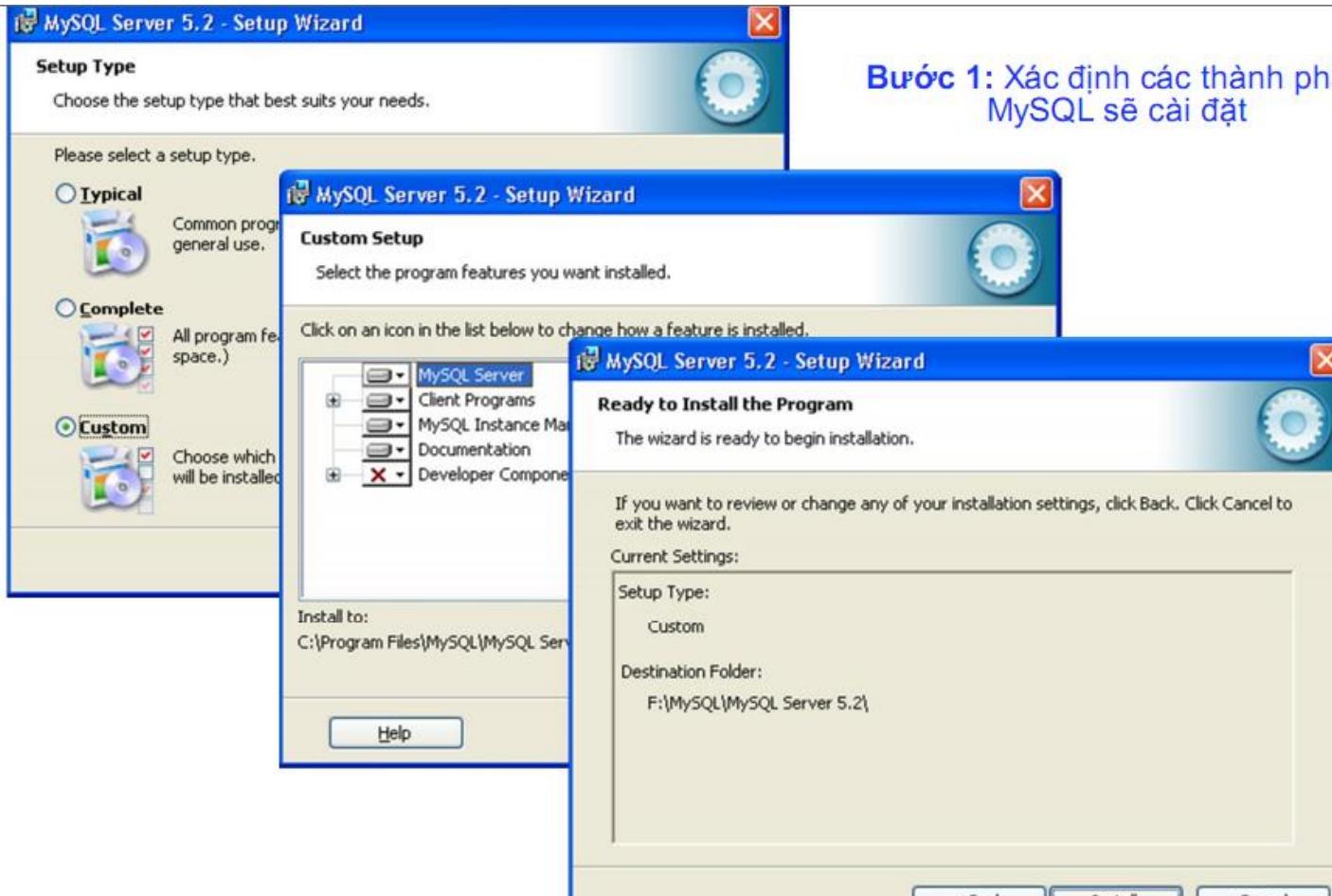
The screenshot shows the MySQL 5.2 Downloads page. On the left, there's a sidebar with links for MySQL Community Server (5.2, 5.1, 5.0, 4.1), MySQL Cluster, MaxDB, GUI Tools, Connectors, Archives, Mirrors, Documentation, Articles, Forums, Lists, Blogs, and Newsletter. The main content area is titled "MySQL 5.2 Downloads". It has three sections: "Windows downloads (platform notes)", "Linux (non RPM packages) downloads (platform notes)", and "Linux x86 generic RPM (dynamically linked) downloads". The "Windows downloads" section is highlighted with a red oval around the "Windows (x86) ZIP/Setup.EXE" link. This link points to a file named "Windows (x86).zip" which is 35.2M in size and was last modified on May 21, 2007. It includes MD5 checksums (9757b40feacd9443581d3263d21356d and 8dd3ee60b6a9295b3332fb75652b27bc) and download links for "Pick a mirror" and "Signature".

# Cài đặt MySQL (tt)

## Cài đặt Database Server – MySQL 5.2



# Cài đặt MySQL (tt)

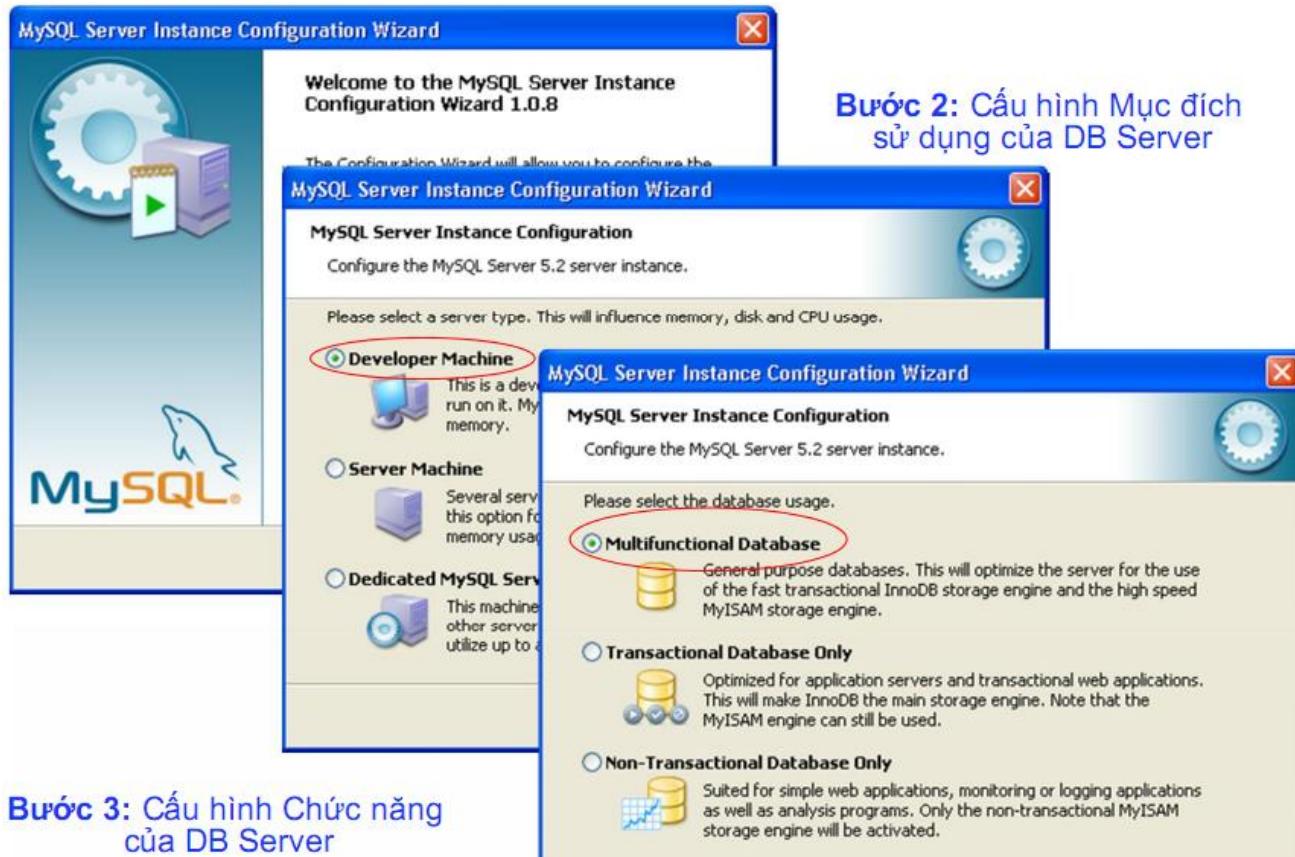


Bước 1: Xác định các thành phần MySQL sẽ cài đặt

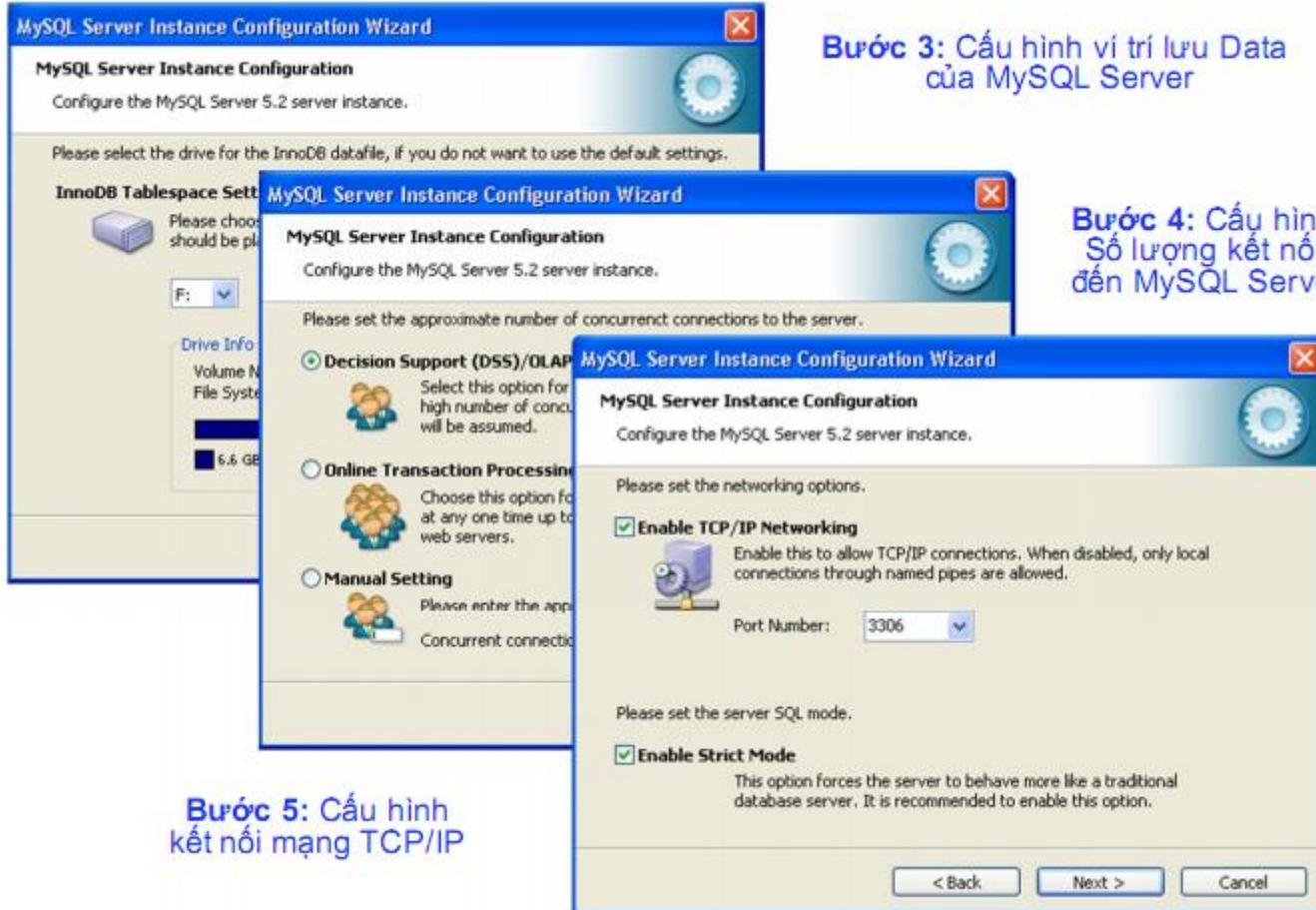
# Cài đặt MySQL (tt)



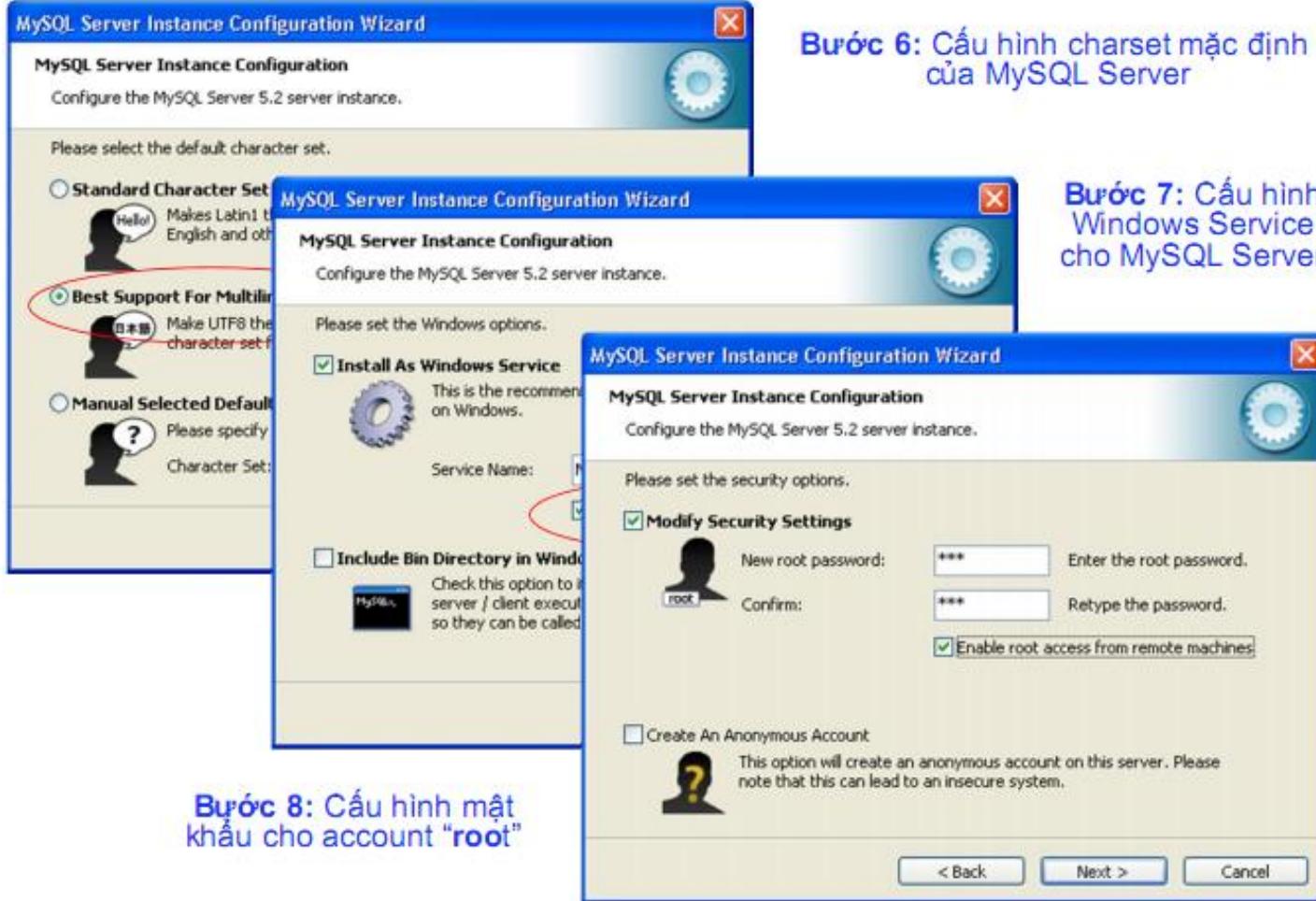
# Cài đặt MySQL (tt)



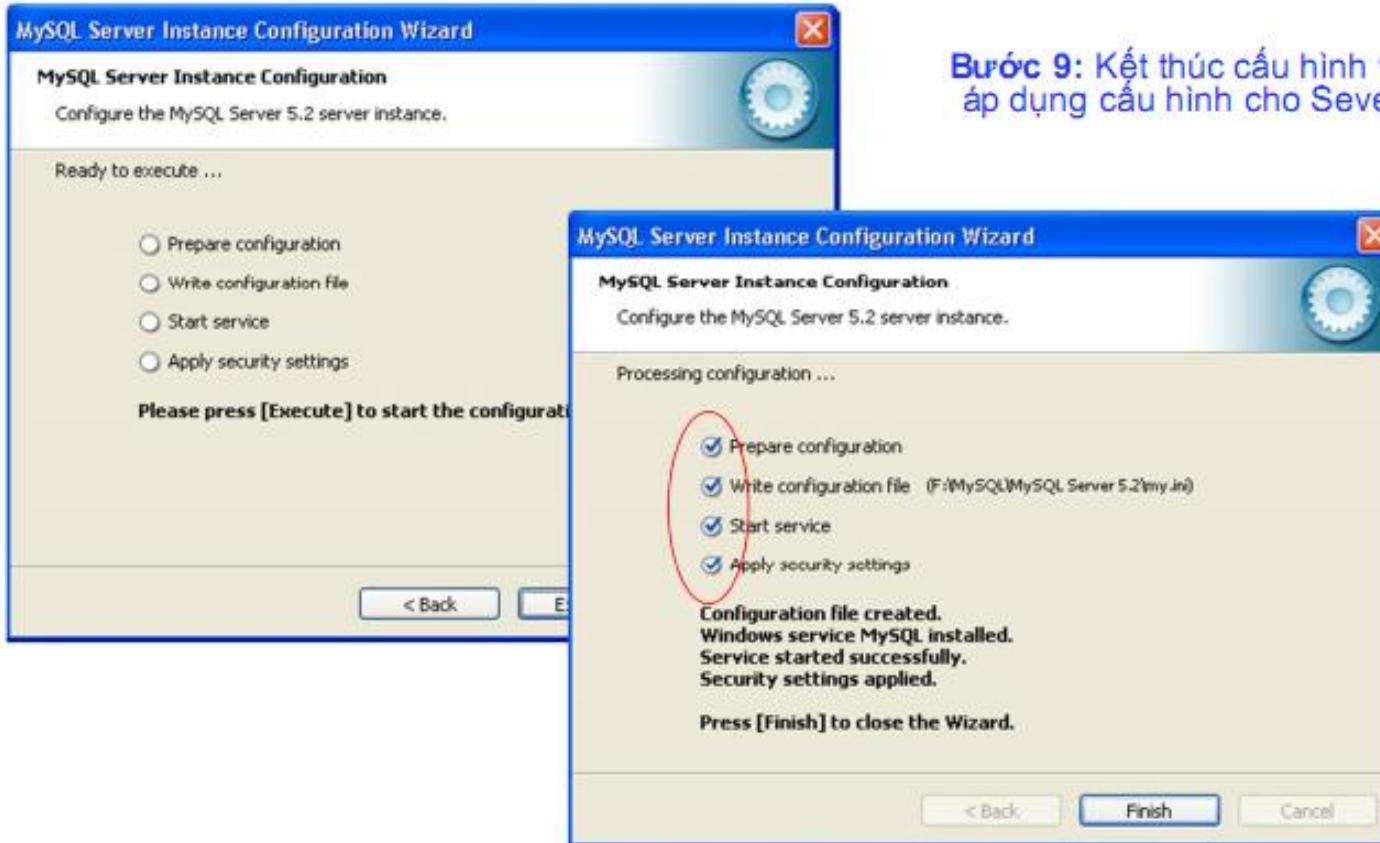
# Cài đặt MySQL (tt)



# Cài đặt MySQL (tt)



# Cài đặt MySQL (tt)



Bước 9: Kết thúc cấu hình và  
áp dụng cấu hình cho Sever

# Cài đặt WAMP

---

## Cài đặt WAMP

- Tự động cài đặt:
  1. PHP Engine
  2. MySQL
  3. Apache
  4. PHPMyAdmin (~MySQL GUI)



# Cấu hình đổi PORT của IIS và Apache

---

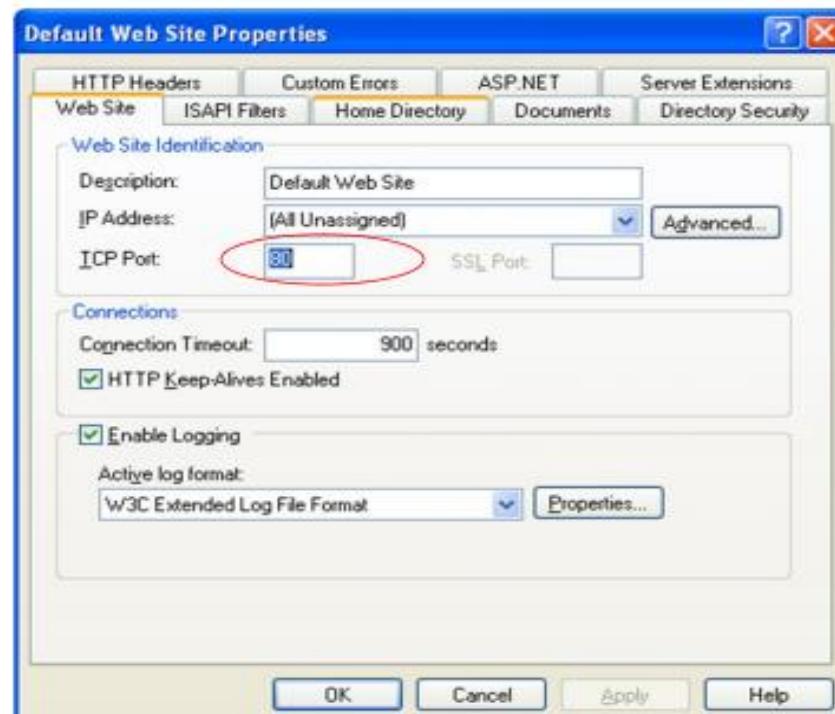
## Cấu hình đổi PORT của IIS và Apache

- URL:
  - `http://servername:port/pathfile#hash?search`
- ServerName : localhost
- Port mặc định của IIS và Apache là 80
  - ➔ Nếu cài cả IIS và Apache trên cùng 1 máy sẽ dẫn đến lỗi tranh chấp port
  - ➔ Cần cấu hình đổi PORT của IIS hoặc Apache
- Ví dụ: IIS (port 80) và Apache (port 81)
  - IIS: <http://localhost/localstart.asp>
  - Apache: <http://localhost:81/phpmyadmin/>

# Cấu hình đổi PORT của IIS và Apache (tt)

## Cấu hình đổi PORT của IIS

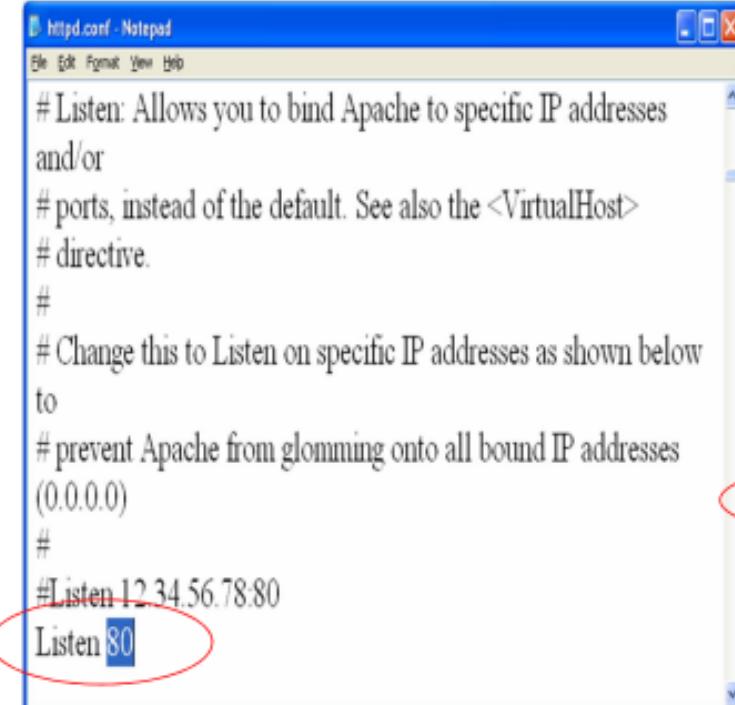
- IIS > Default Website Properties



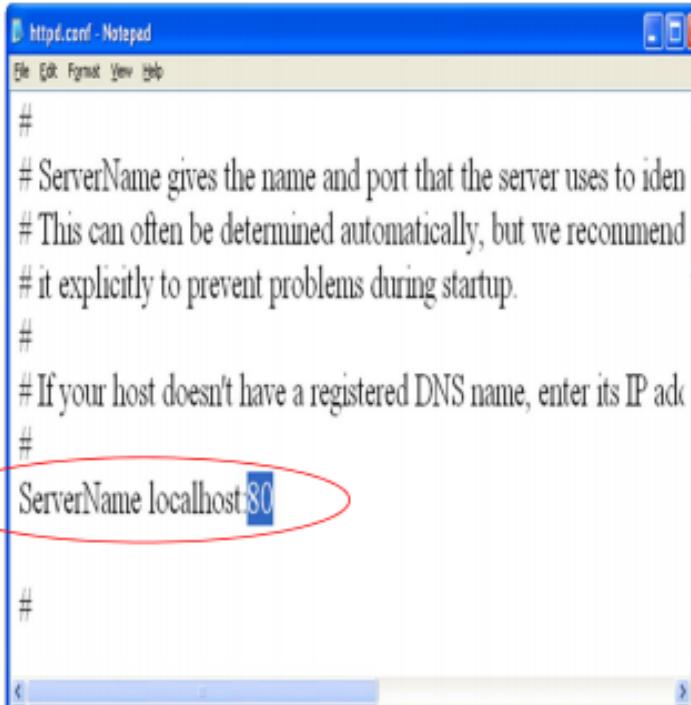
# Cấu hình đổi PORT của IIS và Apache (tt)

## Cấu hình đổi PORT của Apache

- WAMP Server > Config Files > httpd.conf



```
# Listen: Allows you to bind Apache to specific IP addresses  
and/or  
# ports, instead of the default. See also the <VirtualHost>  
# directive.  
  
#  
# Change this to Listen on specific IP addresses as shown below  
to  
# prevent Apache from glomming onto all bound IP addresses  
(0.0.0.0)  
  
#  
#Listen 12.34.56.78:80  
Listen 80
```



```
#  
# ServerName gives the name and port that the server uses to iden  
# This can often be determined automatically, but we recommend  
# it explicitly to prevent problems during startup.  
  
#  
# If your host doesn't have a registered DNS name, enter its IP ad  
#  
ServerName localhost:80  
  
#
```

# NGÔN NGỮ HTML

---

## HTML là gì?

- HTML viết tắt từ **Hypertext Markup Language**, là ngôn ngữ thông dụng nhất dùng viết các trang Web.
- **Hypertext** là cách mà các trang Web (các tài liệu HTML) được kết nối với nhau.
- **HTML** là ngôn ngữ đánh dấu bằng thẻ (**Markup Language**), HTML đánh dấu một tài liệu text bằng các **thẻ (tag)** để cho trình duyệt Web biết cấu trúc của nó mà hiển thị ra màn hình.

# NGÔN NGỮ HTML (tt)

---

Ví dụ 1: tệp HTML

```
<!DOCTYPE>
<html>
    <head>
        <title>Day la tieu de</title>
    </head>
    <body>
        <h1>Tieu de dau doan</h1>
        <p>Chao cac ban den lop LT UD WEB (Phan noi dung cua tai lieu)
        </p>
    </body>
</html>
```

# NGÔN NGỮ HTML (tt)

---

Cấu trúc tài liệu HTML

<!DOCTYPE>

<html>

    <head>

*Các thẻ liên quan tới đầu đề tài liệu*

    </head>

    <body>

*Các thẻ liên quan tới phần thân tài liệu*

    </body>

</html>

# NGÔN NGỮ HTML (tt)

---

- Ví dụ 2:

```
<!DOCTYPE>
<html>
    <head>
        <title>Vi du long phan tu</title>
    </head>
    <body>
        <h1>Vi du mot dau de co dang <i>in nghieng</i></h1> <p>Vi du
        phan van ban <u>bi gach chan</u></p>
    </body>
</html>
```

# NGÔN NGỮ HTML (tt)

---

- Ví dụ 3:

```
<html>
  <head> <title>Vi du the div trong HTML</title> </head>
  <body>
    <div id="menu" align="middle" >
      <a href="../home.html">Trang chủ</a> |
      <a href="../contact.html">Sản phẩm</a> |
      <a href="#">Giới thiệu</a>
    </div>
    <div id="content" align="left" bgcolor="white">
      <h5>Vi du the div</h5> <Chào các bạn đến với HTML <p>
    </div>
  </body>
</html>
```

# NGÔN NGỮ HTML (tt)

---

- Ví dụ 4:

```
<html>
    <head>
        <title>Vi du Text Input trong HTML</title>
    </head>
    <body>
        <form> Họ đệm: <input type="text" name="first_name" />
            <br> Tên: <input type="text" name="last_name" />
        </form>
    </body>
</html>
```

# Cascading Style Sheets -CSS

---

- **CSS** viết tắt từ Cascading Style Sheet, là một Design Language đơn giản được sử dụng để làm đơn giản hóa tiến trình trình bày các trang web.
- CSS dùng điều khiển màu văn bản (text color), font style, khoảng cách giữa các đoạn văn, kích cỡ các cột, hình nền hoặc màu nền, .... Bằng cách tạo một tệp .css ở ngoài (chứa định dạng) và include vào từng trang Web.
- .css chứa một CSS rule cho nhiều thẻ ->ít code hơn nên Webpage sẽ tải nhanh hơn.

# Cascading Style Sheets –CSS (tt)

---

- **CSS** được tạo thành từ các Style Rule. Style Rule bao gồm ba phần:
- **Selector:** là một thẻ HTML được áp dụng một style, nó có thể là bất kỳ một thẻ HTML nào.
- **Property:** là thuộc tính của **Selector**. Nói một cách đơn giản thì tất cả các thuộc tính trong HTML được chuyển đổi thành các **CSS property**.
- **Value:** là giá trị gán cho **Property**. ex:blue, 12px,..

# Cascading Style Sheets –CSS (tt)

---

- Cú pháp:

**Selector {Property : Value}**

Ví dụ: h1 {color:red; font-size:16px}

p {text-align:center; color:blue}

Hoặc muốn áp dụng Style cho tất cả các thẻ HTML trong tài liệu dùng Selector \*:

\* {color :#000000;}

# Cascading Style Sheets –CSS (tt)

---

- **Multiple Style Rule trong CSS, Group Selector**

Định nghĩa nhiều style rule cho cùng phần tử bằng cách viết gộp các cặp property và value, chngs cách nhau bởi dấu chấm phẩy (;

Ví dụ: h1, h2, h3 {  
color: red;  
font-weight: normal;  
text-transform:lowercase;  
}

# Cascading Style Sheets –CSS (tt)

---

- Chú thích trong CSS

Ví dụ:            p {

    color: red;

    /\*vd chu thich tren mot dong\*/

    text-transform:lowercase;

}

/\*vi du:

    chu thich

    tren nhieu dong

\*/

# Cascading Style Sheets –CSS (tt)

---

- Nhúng CSS trực tiếp vào HTML

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Tiêu đề trang web</title>
    <style type="text/css media="all">
      body{
        background-color:yellow;
      }
      h1 {
        color: blue;
      }
    </style>
  </head>
  <body>
    -- Phần thân viết ở đây...
  </body>
</html>
```

# Cascading Style Sheets –CSS (tt)

---

- Nhúng tệp CSS vào HTML

```
<html>
    <head>
        <meta charset="utf-8" />
        <title>Tiêu đề trang web</title>
        <link rel="stylesheet" type="text/css" href="filecss.css" />
    </head>

    <body>
        ...Phần thân viết ở đây...
    </body>
</html>
```

# JAVASCRIPT

---

1. Giới thiệu về JavaScript
2. Lập trình với JavaScript
3. Làm việc với các sự kiện và trình duyệt
4. AJAX và tích hợp phía server
5. JQuery

# Giới thiệu JAVASCRIPT

---

- JavaScript là ngôn ngữ lập trình phổ biến nhất trên thế giới trong suốt 20 năm qua. Nó cũng là một trong số 3 ngôn ngữ chính của lập trình Web:
  - HTML: Giúp bạn thêm nội dung cho trang web.
  - CSS: Định dạng thiết kế, bố cục, phong cách, canh lề của trang web.
  - JavaScript: Cải thiện cách hoạt động của trang web.
- JavaScript giúp cải thiện tính năng của Website và có hàng ngàn mẫu template JavaScript. Ngoài ra, nhờ vào sự cống hiến của cộng đồng, đặc biệt là Github, rất nhiều các ứng dụng mã nguồn mở có sẵn.

# Giới thiệu JAVASCRIPT (tt)

---

Một số lợi ích từ JavaScript

- Bạn không cần một compiler vì web browser có thể biên dịch nó bằng HTML;
- JS dễ học hơn các ngôn ngữ lập trình khác;
- Lỗi dễ phát hiện hơn và vì vậy dễ sửa hơn;
- JS có thể được gắn trên một số element của trang web hoặc event của trang web như là thông qua click chuột hoặc di chuột tới;

# Giới thiệu JAVASCRIPT (tt)

---

Một số lợi ích từ JavaScript

- JS hoạt động trên nhiều trình duyệt, nền tảng,...
- JavaScript có thể sử dụng để kiểm tra input và giảm thiểu việc kiểm tra thủ công khi truy xuất qua database;
- JS giúp Website tương tác tốt hơn với khách truy cập;
- JS nhanh hơn và nhẹ hơn các ngôn ngữ lập trình khác.

# Giới thiệu JAVASCRIPT (tt)

---

Tuy nhiên JavaScript có vài khuyết

- Dễ bị khai thác;
- Có thể được dùng để thực thi mã độc trên máy tính của người dùng;
- Nhiều khi không được hỗ trợ trên mọi trình duyệt;
- JS code snippets lớn;
- Có thể bị triển khai khác nhau tùy từng thiết bị dẫn đến việc không đồng nhất.

# Đặc điểm JAVASCRIPT

Ngôn ngữ	Đặc điểm
Java Script (JS)	JS sẽ giúp tăng tính tương tác trên website. Script này chạy trên các trình duyệt của người dùng thay vì trên server và thường sử dụng thư viện của bên thứ 3 nên có thể tăng thêm chức năng cho website mà không phải code từ đầu.
HTML	“Hypertext Markup Language”-HTML là một trong số các ngôn ngữ lập trình Web phổ biến nhất trên và xây dựng nên các khối chính của một trang Web. Ví dụ về HTML tags là <p> cho đoạn văn và <img> cho hình ảnh.
PHP	PHP là ngôn ngữ phía server, khác với JavaScript chạy trên máy client. Nó thường được sử dụng trong các hệ quản trị nội dung nền PHP như WordPress, nhưng cũng thường được dùng với lập trình back-end và có thể tạo ra kênh truyền thông tin hiệu quả nhất tới và từ database.
CSS	CSS -“Cascading Style Sheets”, nó giúp webmaster xác định styles và định nghĩa nhiều loại nội dung. Bạn có thể làm vậy thủ công với mọi yếu tố trong HTML, nhưng nếu vậy bạn sẽ cứ lặp đi lặp lại thành phần đó mà bạn dùng ở nhiều nơi khác nhau.

# JAVASCRIPT trong HTML

---

## Đưa JavaScript và HTML trực tiếp

- Sử dụng tag `<script></script>` để đặt tất cả mã JS, hoặc JS code có thể được thêm vào giữa tag `<head>` hoặc `<body>`
- Tùy vào nơi thêm code JavaScript trong HTML file, cách tải có thể khác nhau.
  - Tốt nhất là thêm nó vào trong `<head>` để nó tách hằng với nội dung chính của file HTML.
  - Và nếu đặt nó vào trong tag `<body>` có thể tăng tốc độ tải, vì nội dung website sẽ được tải nhanh hơn, và chỉ khi đó JavaScript sẽ được parsed.

# JAVASCRIPT trong HTML

---

Ví dụ: Tệp HTML hiển thị thời gian hiện tại có JS trực tiếp:

```
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Thoi gian bay gio la: </title>
        <script>
            var time = new Date();
            console.log(time.getHours() + ":" + time.getMinutes() + ":" + time.getSeconds());
        </script>
    </head>
    <body> Ví dụ đồng hồ </body>
</html>
```

# JAVASCRIPT trong HTML

---

```
<!DOCTYPE html>
<html lang="en-US">
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- Thẻ `<meta>` viewport thiết lập cho trang web hiển thị tương ứng với kích thước của từng thiết bị khác nhau.
- Thuộc tính `width=device-width` đặt chiều rộng của trang web theo chiều rộng màn hình của thiết bị.
- Thuộc tính `initial-scale=1.0` thiết lập mức độ phóng ban đầu khi trang được trình duyệt tải lần đầu tiên, người dùng sẽ không thể zoom khi thuộc tính này có giá trị bằng 1

# JAVASCRIPT trong HTML

---

```
<!DOCTYPE html>
<html lang="en-US">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Thoi gian bay gio la: </title>
</head>
<body>
    <script>
        let d = new Date();
        document.body.innerHTML =
            "<h1>Thoi gian bay gio la: " + d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds() " </h1>"
    </script>
</body>
</html>
```

# JAVASCRIPT trong HTML

---

## Chèn JavaScript trong HTML bằng một file độc lập

- JavaScript vào HTML trực tiếp chưa phải là cách hay nhất. Vì có thể một vài JS scripts cần được dùng ở nhiều trang khác nhau. Vậy tạo một file JavaScript riêng biệt để có thể thêm JavaScript vào HTML thông qua file đó, và được gọi trong HTML documents giống với cách gọi CSS documents. Lợi ích khác của thêm JS code vào file độc lập là:
  - HTML code và JavaScript code tách riêng nhằm tái sử dụng lại code
  - Việc đọc code dễ dàng, nên bảo trì đơn giản hơn
  - Files Cached JavaScript sẽ tăng tốc Website bằng cách giảm thiểu thời gian trang phải tải.

# JAVASCRIPT trong HTML

---

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Time right now:</title>
</head>
<body>
</body>
<script src="js/myscript.js"></script>
</html>
```

# JAVASCRIPT trong HTML

---

- NỘI DUNG CỦA FILE myscript.js LÀ:

```
let d = new Date();
```

```
document.body.innerHTML = "<h1>Thoi gian bay gio la: "
```

```
+ d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds() </h1>"
```

Lưu ý: file js/myscript.js được đặt trong cùng thư mục của file HTML.

# JAVASCRIPT trong HTML

---

```
<HTML>
    <HEAD>
        </HEAD>
    <BODY>
        <script language = "JavaScript">
            var Ten, Tuoi;

            Ten = prompt("Bạn hãy nhập vào tên ", "");
            Tuoi = prompt("Bạn hãy nhập vào Tuổi : ", 20);

            document.write("Chào bạn : <B> " + Ten + "</B>");
            document.write("<BR>"); // Xuống dòng
            document.write("Tuổi của bạn là : <U> " + Tuoi + "</U>");

        </script>
    </BODY>
</HTML>
```

# JAVASCRIPT trong HTML

---

```
<HTML>
    <HEAD>
        </HEAD>
<BODY>
    <input type=button name= welcome value="Welcome"
           onclick="alert ('Welcome to JavaScript');">
</BODY>
</HTML>
```

# JAVASCRIPT (tt)

---

## Kiểu dữ liệu JavaScript

Một trong những nét đặc thù chủ yếu nhất của một ngôn ngữ chương trình là tập hợp các kiểu dữ liệu nó hỗ trợ. Đây là kiểu các giá trị mà có thể được biểu diễn và được thao tác trong ngôn ngữ chương trình.

JavaScript cho bạn làm việc với 3 kiểu dữ liệu gốc sau:

- **Số**, ví dụ: 123, 120.50, ...
- **Chuỗi** văn bản, ví dụ: "This text string", ...
- **Boolean** ví dụ: true hoặc false.

# JAVASCRIPT (tt)

---

## Các toán tử

- Toán tử số học
- Toán tử so sánh
- Toán tử logic (hoặc quan hệ)
- Toán tử gán
- Toán tử điều kiện

## JAVASCRIPT (tt)

---

JavaScript hỗ trợ các form lệnh **if..else** sau:

- Lệnh if
- Lệnh if...else
- Lệnh if...else if...

# JAVASCRIPT (tt)

---

## Cú pháp:

```
if (expression){ Statement(s) to be executed if expression is true }

<html>
  <body>
    <script type="text/javascript">
      var age = 20;
      if( age > 18 ){ document.write("<b>Qualifies for driving</b>"); }
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

# JAVASCRIPT (tt)

Cú pháp:

---

```
if (expression){  
    Statement(s) }  
  
else{  
    Statement(s) }  
  
<html>  
<body>  
    <script type="text/javascript">  
        var age = 15;  
        if( age > 18 ){ document.write("<b>Qualifies for driving</b>"); }  
        else{ document.write("<b>Does not qualify for driving</b>"); }  
    </script>  
    <p>Set the variable to different value and then try...</p>  
</body>  
</html>
```

# Chương II: Giới thiệu PHP

---

- **PHP (PHP: Hypertext Preprocessor)**

- Là ngôn ngữ script trên server
- Là ngôn ngữ để viết các trang web động
  - Ảnh hưởng từ: C, C++, Java, Perl, Tcl
- Ra đời năm 1994
  - Rasmus Lerdorf (1968, GreenLand)
  - Tên gốc là Personal Home Page Tools
- Các phiên bản:
  - 1.0, 2.0, 3.0, 4.0, 4.1,... (nhiều người phát triển)
  - Dùng phổ biến hiện nay: PHP 5.4.4 (6/2012), PHP 6

# Ưu điểm của PHP

---

- Ngôn ngữ dễ học, dễ viết.
- Mã nguồn mở (open source code)
  - Miễn phí, download dễ dàng từ Internet.
  - Có thể làm việc trên mã nguồn, thêm, sửa, sử dụng và phân phối.
- Mã nguồn không cần sửa lại nhiều khi triển khai trên các hệ điều hành khác nhau (Windows, Linux, Unix,...)

# Ưu điểm của PHP

---

- Có thể chạy trên nhiều web server khác nhau (Apache, IIS,...).
- Có thể kết nối được với nhiều loại DBMS khác nhau:
  - SQL Server, Oracle, DB2, PostgreSQL, Adabas, dBase, Empress, FilePro, mSQL, Solid,...
  - Phổ biến nhất: MySQL

# Đặc điểm PHP

---

- Có khả năng hướng đối tượng
- Thông dịch

# Nhúng PHP vào HTML

---

- Có thể nhúng mã PHP vào mọi vị trí trong trang HTML.

- Đoạn mã PHP được đặt giữa: `<?php ... ?>`:

`<?php`

`//Đoạn lệnh PHP ở đây`

`?>`

- Có thể sử dụng dạng rút gọn như sau (không khuyến khích):

`<? //Đoạn lệnh PHP ở đây ?>`

`<% //Đoạn lệnh PHP ở đây %>`

# Nhúng PHP vào HTML

---

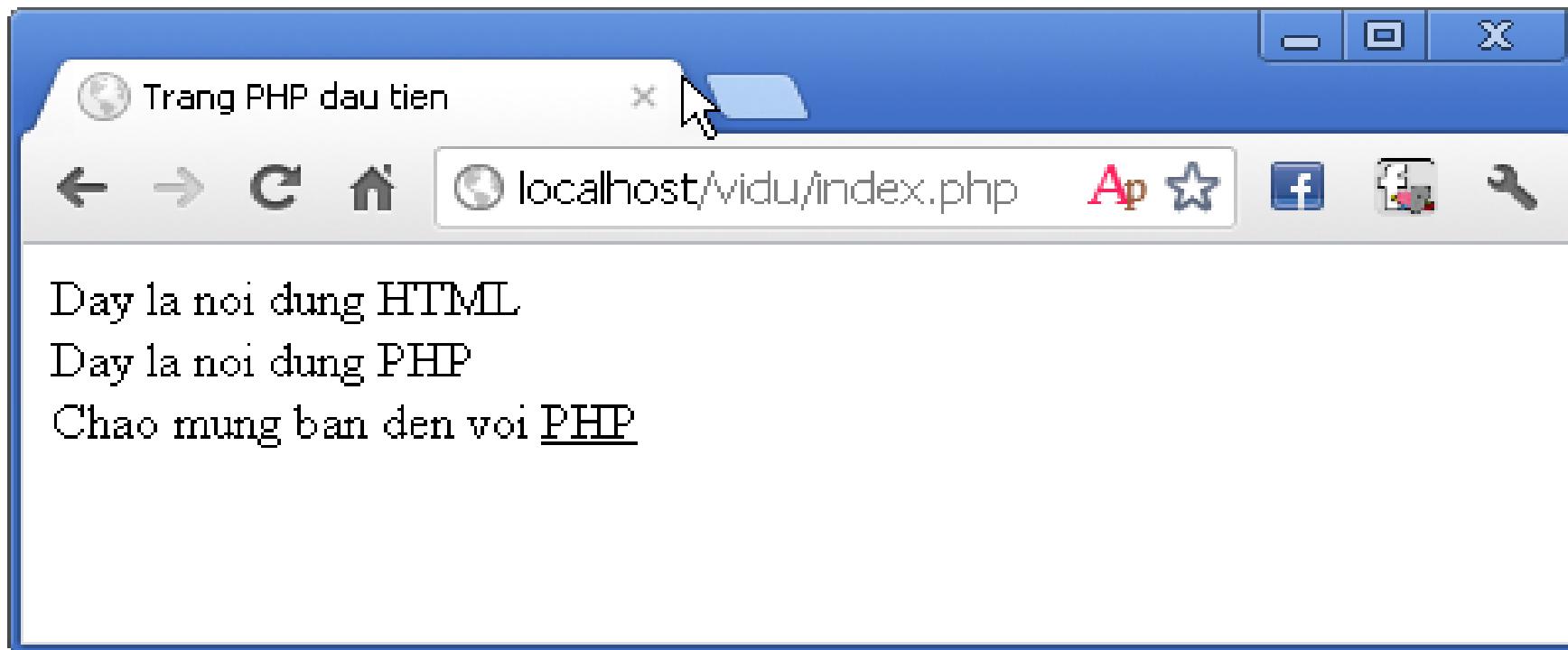
- Một cấu trúc lệnh thông thường của PHP có thể được tách làm nhiều phần, mỗi phần đặt giữa <?php ... ?>
- Kết quả do đoạn lệnh PHP đưa ra (thường là mã HTML) được đưa vào vị trí mà đoạn lệnh PHP đang chiếm chỗ.

# Trang PHP đầu tiên

---

```
<html>
  <head>
    <title>Trang PHP dau tien</title>
  </head>
  <body>
    Day la noi dung HTML
    <?php
      print "<br/>Day la noi dung PHP<br/>";
      echo "Chao mung ban den voi <u>PHP</u>";
    ?>
  </body>
</html>
```

# Trang PHP đầu tiên (tiếp)



# Quy tắc viết mã lệnh trong PHP

---

- Phân biệt chữ hoa/chữ thường
- Mỗi lệnh kết thúc bởi chấm phẩy (;)
  - Trừ lệnh cuối trước khóa ?> có thể bỏ qua ;
- Khối (nhiều) lệnh được đặt trong cặp { }
- Xem thêm tại:
  - <https://sites.google.com/site/pcdinh2/phpvietnamcodingstandards>
  - <http://framework.zend.com/manual/en/coding-standard.coding-style.html>

# Chú thích trong PHP

---

- //Dòng chú thích
- #Dòng chú thích
- /\*  
Đoạn chú thích trên nhiều dòng  
\*/

# Chương III: Lập Trình PHP

---

## Biến:

- Biến là vùng nhớ lưu giá trị có thể thay đổi được trong chương trình.
- Khai báo biến trong PHP:

`$tên_bien`

- Ví dụ: `$a;`    `$đem;`
- Biến trong PHP có tính định kiểu thấp
  - Không chỉ định kiểu dữ liệu khi khai báo biến

# Biến (tiếp)

---

- Một số quy định khi khai báo biến:
  - Bắt đầu bằng dấu đô la (\$), tiếp ngay sau \$ là tên biến (không có khoảng trắng ở giữa).
  - Tên biến bao gồm chữ cái, chữ số, dấu gạch nối (\_) và chỉ được bắt đầu là chữ cái hoặc dấu gạch dưới.

# Biến (tiếp)

---

- Chú ý:
  - Biến trong PHP phân biệt chữ hoa, chữ thường
  - Biến trong PHP không bắt buộc phải khai báo trước khi sử dụng
    - Được tự động khai báo vào lần gán giá trị đầu tiên.
    - Tuy nhiên nếu sử dụng biến chưa khởi tạo có thể gây lỗi. (có khai báo nhưng không khởi tạo)
  - Kiểu dữ liệu lưu trong biến có thể thay đổi trong quá trình lập trình.

# Ví dụ sử dụng biến

---

- Ví dụ 1

```
<?php
```

```
$a = 10; //biến a có giá trị kiểu số là 10
$a = "Hello"; //biến a có giá trị kiểu chuỗi là Hello
$_a1 = true; //biến _a1 có giá trị kiểu bool
_a12 = 20; //loi, vì tên biến không bắt đầu bằng $
$_12a = 20; //loi, vì sau $ bắt đầu bằng số
```

```
?>
```

- Ví dụ 2

```
<?php
```

```
$soLuong = 30;
$donGia = 20;
$thanhTien = $soLuong * $donGia;
echo "Tong tien: " . $thanhTien;
```

```
?>
```

# Biến động (biến biến)

---

- Cho phép sử dụng giá trị của biến làm tên biến khác.
- VD:

```
$a = "hello";
```

```
$$a = "world"; // $hello = "world"
```

# Tham trị và tham chiếu

---

- Tham trị: khi thực hiện phép gán biến cho biến thì mặc định giá trị được sao chép từ biến nguồn sang biến đích
  - Ví dụ: \$a = 10; \$b = \$a (giá trị của \$a được sao chép sang \$b)
- Dùng tham chiếu khi muốn đặt thêm một tên cho một biến có sẵn
  - Ví dụ: \$a = 10; \$b = &\$a (lúc này \$a và \$b là hai tên của cùng một biến)

# Thời gian biến tồn tại

---

- Biến được tạo ra khi được gán giá trị lần đầu và tồn tại trong suốt quá trình thực thi script.
  - Bao gồm cả các khai báo trong các file được chèn bởi include(), require().
- Mỗi lần script được thực thi là biến được tạo ra độc lập với các lần thực thi khác của cùng script đó.

# Phạm vi của biến

---

- Phạm vi của biến là phạm vi ở đó biến xác định.
- Trong PHP, biến có ba mức phạm vi:
  - *Local variables* – biến cục bộ
    - Khai báo, khởi tạo trong hàm
    - Có tác dụng từ khi khởi tạo đến hết hàm.

# Phạm vi của biến (tiếp)

---

- Ba mức phạm vi:
  - *Global variables* – biến toàn cục
    - Khai báo, khởi tạo ngoài hàm
    - Có tác dụng từ khi khởi tạo đến hết file
      - Thường sử dụng bên trong các script,
      - Mặc định là không thể sử dụng bên trong các hàm.
      - Nếu sử dụng trong hàm thì khi đó xem như biến được khai báo lại và trở thành biến cục bộ.

# Phạm vi của biến (tiếp)

---

- Ba mức phạm vi:
  - *SuperGlobal Variables* - biến siêu toàn cục:
    - Là các biến được PHP định nghĩa sẵn.
    - Không thể định nghĩa (lại) bởi người dùng.
    - Có thể sử dụng ở mọi nơi.
    - Một số biến siêu toàn cục:
      - \$GLOBALS,
      - \$\_GET, \$\_POST, \$\_REQUEST,
      - \$\_SESSION, \$\_COOKIE,
      - \$\_SERVER,...

# Phạm vi của biến (tiếp)

---

- Ví dụ

<?php

```
$a = 1; $b = 10;

function tang() {
    $a = $a + 1;
    $b = $b + 1;
    echo "(trong) a=$a b=$b";
}

tang();
echo "(ngoai) a=$a b=$b";

?>
```

# Sử dụng biến toàn cục trong hàm

---

- Để sử dụng biến toàn cục trong hàm:
  - Sử dụng từ khóa `global` để khai báo (lại) biến toàn cục bên trong hàm theo cú pháp:  
**global** \$biến1, \$biến2;
    - Ví dụ: `global $a, $b;`
    - Nếu trước đó biến chưa khai báo thì sẽ khai báo mới, và biến mới đó là biến toàn cục (mặc dù được khai báo bên trong hàm)

# Sử dụng biến toàn cục trong hàm (tiếp)

---

- Để sử dụng biến toàn cục trong hàm:
  - Sử dụng mảng siêu toàn cục **\$GLOBALS**  
**\$GLOBALS['tên\_var\_toàn\_cục']**
    - Khi một biến toàn cục được khai báo, một tham chiếu của biến đó sẽ được thêm vào trong biến mảng siêu toàn cục **\$GLOBALS** với key là tên của biến toàn cục.  
**\$GLOBALS['tên\_var\_toàn\_cục'] = &\$biến\_toàn\_cục**
    - Ví dụ: `$a = 10; //biến toàn cục`  
 $\Leftrightarrow \$GLOBALS['a'] = \&\$a;$

# Sử dụng biến toàn cục trong hàm (tiếp)

---

- Thảo luận: cho biết kết quả của đoạn mã sau

```
<?php
```

```
$a = 1; $b = 10;
```

```
function tang() {  
    global $a;  
    $a = $a + 1;  
    $b = $b + 1;  
    echo "(trong)a=$a b=$b";  
}
```

```
tang();  
echo "(ngoai)a=$a b=$b";
```

```
?>
```

# Sử dụng biến toàn cục trong hàm (tiếp)

---

- Thảo luận: cho biết kết quả của đoạn mã sau

```
<?php  
    $b = 10;  
  
    function tang() {  
        global $a;  
        $a = $a + 1;  
        $b = $b + 1;  
        echo "(trong)a=$a b=$b";  
    }  
    tang();  
    echo "(ngoai)a=$a b=$b";  
?>
```

# Sử dụng biến toàn cục trong hàm (tiếp)

---

- Thảo luận: cho biết kết quả của đoạn mã sau

```
<?php
```

```
$a = 1; $b = 10;
```

```
function tang() {
```

```
    $a = $a + 1;
```

```
    $GLOBALS['b'] = $GLOBALS['b'] + 1;
```

```
    echo "(trong)a=$a b=$b";
```

```
}
```

```
tang();
```

```
echo "(ngoai)a=$a b=$b";
```

```
?>
```

# Biến tĩnh

---

- Biến tĩnh:
  - Là biến cục bộ mà giá trị của nó được lưu trữ qua nhiều lần gọi hàm
  - Chỉ được khởi tạo một lần duy nhất tại thời điểm khai báo lần đầu tiên trong suốt quá trình thực thi script.

# Biến tĩnh (tiếp)

---

- Khai báo biến tĩnh theo cú pháp:

**static** \$biến\_tĩnh = giá\_trị;

- Chú ý:
  - giá\_trị khởi tạo của biến static chỉ có thể là một hằng (literal or constant) mà không được là một biểu thức (expression).

- Ví dụ

**static** \$a = 0;

**static** \$a = 1+2; //lỗi

**static** \$a = sqrt(121); //lỗi

# Biến tĩnh (tiếp)

---

- Thảo luận: cho biết kết quả của đoạn mã sau

```
<?php
    function bienTinh(){
        static $dem = 1;
        echo "Bien tinh: $dem <br/>";
        $dem++;
    }
    function bienThuong(){
        $dem = 10;
        echo "Bien thuong: $dem <br/>";
        $dem++;
    }
    bienTinh(); bienThuong();
    bienTinh(); bienThuong();
    bienTinh(); bienThuong();
?>
```

# Kiểu dữ liệu

---

- PHP hỗ trợ các loại kiểu dữ liệu sau:
  - Logic (boolean)
  - Số (integer, double)
  - Chuỗi (string)
  - Mảng & đối tượng (array, object)

# Kiểu logic

---

- Có 2 trạng thái: TRUE và FALSE
  - Hoặc true và false (~không phân biệt hoa/thường)
  - Các giá trị sau "được coi là" false (convert)
    - 0 (int), 0.0 (float), "" (chuỗi rỗng), "0", NULL,...
  - Các giá trị khác "được coi là" true

# Kiểu số

---

- Số nguyên từ  $-2^{31}$  đến  $2^{31}-1$ 
  - Hệ thập phân. VD: \$a = 16;
  - Hệ 16 (hexa). VD: \$a = 0x10;
  - Hệ 8 (bát phân). VD: \$a = 020;
- Số thực (thập phân): từ  $1.7E-308$  đến  $1.7E+308$ 
  - Biểu diễn: \$a = 0.017
  - Dạng khoa học: \$a = 17.0E-03

# Kiểu số (tiếp)

---

- Ví dụ

```
<?php
```

```
    $a1 = 15; $a2 = 015; $a3 = 0x15;  
    echo "vd1: $a1, $a2, $a3";  
    $a4 = 80; $a5 = 080; $a6 = 0x80;  
    echo "vd2: $a4, $a5, $a6";
```

```
?>
```

# Kiểu chuỗi

---

- Chuỗi trong PHP là một dãy các ký tự 1 byte
  - Chỉ hỗ trợ tập 256 ký tự, không hỗ trợ UNICODE thuần (native Unicode)
  - Kích thước tối đa là 2GB.
- Có 4 cách xác định một chuỗi trong PHP:
  - Dấu nháy đơn ('') và dấu nháy kép ("")
  - Cú pháp heredoc và nowdoc

# Kiểu chuỗi (tiếp)

---

- Chuỗi đặt trong nháy kép ("")
  - Nếu trong chuỗi có các biến thì giá trị của biến sẽ được đưa vào chuỗi tại vị trí của biến.
  - Nếu trong chuỗi có các ký tự thoát (escape) thì các ký tự thoát sẽ được xử lý.
    - Ký tự thoát là các ký tự đặc biệt được đặt sau dấu backslash (\)
- Chuỗi đặt trong nháy đơn ('')
  - Không xử lý biến và các ký tự thoát xuất hiện trong chuỗi ngoại trừ 2 ký tự thoát \' và \\

# Kiểu chuỗi (tiếp)

---

- Ví dụ:

```
$a = "Hello";  
$b = "$a world"; //\Leftrightarrow $b="Hello world"  
$c = '$a world';  
//\Leftrightarrow $c='$a world' (không thay đổi)
```

# Kiểu chuỗi (tiếp)

---

- Chú ý: để tránh lỗi thì nên đặt biến trong chuỗi vào giữa ngoặc nhọn { }

- Ví dụ:

```
$a = "He";
```

```
$b = "{$a}llo"; //sai, vì PHP hiểu là $allo
```

```
$c = "{$a}llo"; //đúng ($c = "Hello")
```

```
$d = "{$c} World" // $d = "Hello World"
```

```
$d = "{$c} World" // $d = "Hello World"
```

```
$d = "{$c} World" // lỗi
```

# Kiểu chuỗi (tiếp)

---

- Ký tự thoát (escape):

- Là các ký tự đặc biệt trong chuỗi, đặt ngay sau dấu backslash (\)
- Một số ký tự thoát hay dùng:

- \n \t \\ \\$ \" \'

| Sequence | Meaning   |
|----------|---|
| \n       | Linefeed (LF or 0x0A (10) in ASCII)                       |
| \r       | carriage return (CR or 0x0D (13) in ASCII)                |
| \t       | horizontal tab (HT or 0x09 (9) in ASCII)                  |
| \v       | vertical tab (VT or 0x0B (11) in ASCII) (since PHP 5.2.5) |
| \e       | escape (ESC or 0x1B (27) in ASCII) (since PHP 5.4.0)      |
| \f       | form feed (FF or 0x0C (12) in ASCII) (since PHP 5.2.5)    |
| \\\      | backslash   |
| \\$      | dollar sign   |
| \"       | double-quote  |

# Kiểu chuỗi (tiếp)

---

- Ví dụ:

- cần in chuỗi: Hãy nói "KONICA"

echo "Hãy nói "KONICA""; //Lỗi

echo "Hãy nói \"KONICA\""; //Đúng

echo 'Hãy nói "KONICA"'; //Đúng

- cần in chuỗi: Hãy nói 'KONICA'

echo "Hãy nói 'KONICA"'; //Đúng

echo 'Hãy nói \'KONICA\"'; //Đúng

# Kiểu chuỗi (tiếp)

---

- Kiểu tài liệu: (heredoc, nowdoc)
  - Cho phép viết 1 chuỗi trên nhiều dòng.
  - **heredoc**: tạo 1 chuỗi tương tự như chuỗi đặt trong cặp dấu nháy kép ("")
    - Xử lý các biến và các ký tự đặc biệt có trong chuỗi
  - **nowdoc**: tạo 1 chuỗi tương tự như chuỗi đặt trong cặp dấu nháy đơn ('')
    - Không xử lý các biến và các ký tự đặc biệt có trong chuỗi.
    - Từ PHP 5.3.0

# Kiểu chuỗi (tiếp)

---

- Cách viết kiểu tài liệu

`$biến = <<<Ký_hiệu`

    nội dung trên nhiều dòng

`Ký_hiệu;`

- ✓ Chú ý:

- ✓ Ký\_hiệu ở vị trí kết thúc phải được viết ở ký tự đầu tiên của dòng
- ✓ Trong nowdoc, Ký\_hiệu bắt đầu phải được đặt trong cặp dấu nháy đơn (' )

# Kiểu chuỗi (tiếp)

---

- Ví dụ heredoc:

\$a = <<<EOD

Đây là chuỗi nằm trên nhiều dòng sử dụng cú pháp  
kiểu tài liệu 'heredoc'  
EOD;

- Ví dụ nowdoc:

\$a = <<<'EOT'

Đây là chuỗi nằm trên nhiều dòng sử dụng cú pháp  
kiểu tài liệu 'nowdoc'  
EOT;

# Sự chuyển đổi kiểu dữ liệu

---

- Có hai hình thức ép kiểu chính
  - Ép kiểu ngầm định: xảy ra tự động khi thực hiện các toán tử đòi hỏi hai biểu thức cùng kiểu
  - Ép kiểu chỉ định: chỉ định một kiểu dữ liệu cụ thể đặt trong cặp () trước biểu thức cần ép kiểu.

# Ép kiểu

| Ký hiệu                   | Ý nghĩa kiểu                   |
|---------------------------|--------------------------------|
| (int), (integer)          | Số nguyên                      |
| (real), (double), (float) | Số thập phân                   |
| (string)                  | Chuỗi                          |
| (array)                   | Mảng                           |
| (object)                  | Đối tượng                      |
| (bool), (boolean)         | Logic                          |
| (unset)                   | NULL, tương tự như gọi unset() |

# Ép kiểu

---

- Ví dụ:

\$x = '123abc'; // \$x là chuỗi

\$y =(int) \$x; // \$y là số nguyên 123

\$x = (int) 'abc'; // \$x = ???

\$x = (int) '123abc45'; // \$x = ???

\$x = (int) 'abc123'; // \$x = ???

# Một số hàm kiểm tra kiểu

---

- bool **is\_tênkiểu** (\$tên\_bien hay biểu thức)
  - Kiểm tra dữ liệu của một biến, kết quả trả về true hoặc false
  - **is\_tênkiểu** có thể là:
    - **is\_integer, is\_numeric, is\_int, is\_double, is\_real, is\_float**
    - **is\_string, is\_bool, is\_array, is\_object**
- string **gettype** (\$tên\_bien hay biểu thức)
  - Trả về loại kiểu dữ liệu như: integer, double, long ...
- int **settype** (\$tên\_bien, 'kiểu\_dữ\_liệu')
  - Gán kiểu dữ liệu cho tên biến

# Một số hàm kiểm tra kiểu (tiếp)

---

- Ví dụ

```
<?php
    $a = 10;
    echo gettype($a);      // integer
    $a = 'hello';
    echo gettype($a);      // string
?
<?php
$n='abc';
echo is_integer($n);
?>
```

# Toán tử

---

- Kiểu số:
  - Kết hợp: +, -, \*, /, %
  - Tăng giảm: ++, --  
VD: \$a++; \$a--; ++\$a; --\$a;
- Chuỗi: Toán tử chấm (.)
- Logic: AND (&&), OR (||), XOR, !
- Gán: =, +=, -=, .=
- So sánh: ==, ===, !=, <, >, <=, >=

# Toán tử so sánh

---

| Ký hiệu | Ý nghĩa                     |
|---------|-----------------------------|
| $= =$   | Bằng giá trị                |
| $= = =$ | Bằng giá trị và cùng kiểu   |
| $! =$   | Khác giá trị                |
| $< >$   | Khác giá trị                |
| $! = =$ | Khác giá trị hoặc khác kiểu |
| $<$     | Nhỏ hơn                     |
| $>$     | Lớn hơn                     |
| $< =$   | Nhỏ hơn hoặc bằng           |
| $> =$   | Lớn hơn hoặc bằng           |

# Toán tử bitwise

---

| Ký hiệu | Ý nghĩa   |
|---------|-----------|
| &       | And       |
|         | Or        |
| ^       | Xor       |
| ~       | Not       |
| <<      | Dịch trái |
| >>      | Dịch phải |

## Toán tử error @

---

- Toán tử error @ ngăn không cho thông báo lỗi nếu có lỗi thực thi xảy ra.
- ✓ Ví dụ : \$a=10; \$b=0; \$c=@\$a/\$b

# Thứ tự ưu tiên phép toán

| TT | Toán tử  | Quy tắc liên kết | Ghi chú   |
|----|--|------------------|---|
| 1  | new  |                  | Tạo đối tượng từ 1 class                                      |
| 2  | [ ]  | Bên phải trước   | Truy cập 1 phần tử của mảng                                   |
| 3  | ++ --  |                  | Tăng/giảm 1 đơn vị  |
| 4  | ! ~ -<br>(int)(float)(string)(aray)(object)<br>@ |                  |   |
| 5  | * / %  | Bên trái trước   |   |
| 6  | + - .  | Bên trái trước   |   |
| 7  | << >>  | Bên trái trước   |   |
| 8  | == != === !==                                    |                  | Chỉ áp dụng trên 2 toán hạng<br>nên không có quy tắc liên kết |

# Thứ tự ưu tiên phép toán

| TT | Toán tử                              | Quy tắc liên kết |
|----|--------------------------------------|------------------|
| 9  | &                                    | Bên trái trước   |
| 10 | ^                                    | Bên trái trước   |
| 11 |                                      | Bên trái trước   |
| 12 | &&                                   |                  |
| 13 |                                      | Bên trái trước   |
| 14 | ? :                                  | Bên trái trước   |
| 15 | = += -= *= /= .= %= &=  = ^= <<= >>= | Bên trái trước   |
| 16 | and                                  | Bên trái trước   |
| 17 | xor                                  | Bên trái trước   |
| 18 | or                                   | Bên trái trước   |
| 19 | ,                                    | Bên trái trước   |

# Hằng

---

- Hằng
  - Là một vùng nhớ chứa dữ liệu
  - Hằng chỉ được phép gán giá trị duy nhất 1 lần.
  - Giá trị của hằng chỉ thuộc các kiểu cơ bản.
  - Hằng có thể được truy cập tại bất cứ vị trí nào trong mã lệnh của chương trình.

# Hằng (tiếp)

---

- Hằng trong PHP được định nghĩa bởi hàm define theo cú pháp:

define (string tên\_hằng, giá\_trị\_hằng)

- Ví dụ :

```
<?php  
    define('PI',3.14);  
    $r=4;  
    echo 'Diện tích hình tròn: ' . $r*$r*PI;  
?>
```

# Hằng (tiếp)

---

- Kiểm tra sự tồn tại của hằng bằng lệnh:  
`defined('tên_hằng')`
- Một số quy định của hằng:
  - Đặt tên hằng giống cách đặt tên biến
  - Hằng không có dấu "\$" ở trước tên.
  - Hằng thường được viết toàn bộ bằng chữ in để phân biệt với biến.
  - Bắt buộc phải định nghĩa hằng trước khi sử dụng.

# Các câu lệnh điều khiển

---

- Rẽ nhánh
  - If
  - Switch
- Lặp
  - For/Foreach
  - While/Do...while
  - Break/Continue

# Rẽ nhánh if

```
if (BTLG)
```

```
    nhóm_lệnh;
```

```
if (BTLG)
```

```
    nhóm_lệnh1;
```

```
else
```

```
    nhóm_lệnh2;
```

```
$biến = BTLG ? Giá_trị1 : Giá_trị2
```

```
if (BTLG1)
```

```
    nhóm_lệnh1;
```

```
elseif (BTLG2)
```

```
    nhóm_lệnh2;
```

```
elseif (BTLG3)
```

```
    nhóm_lệnh3;
```

```
else
```

```
    nhóm_lệnh_cuối;
```

# Rẽ nhánh **if** (tiếp)

---

- Ví dụ:

```
<?php  
    $a = 2;  
    $b = 3;  
    if ($a > $b) {  
        echo 'biên $a có giá trị lớn hơn $b';  
    }  
    else {  
        echo 'biên $b có giá trị lớn hơn $a';  
    }  
?>
```

# Rẽ nhánh **if** (tiếp)

---

- Ví dụ:

```
<?php
```

```
$a = 10;  
if ($a % 4 == 0) {  
    echo '$a chia het cho 4';  
} elseif ($a % 4 == 1) {  
    echo '$a chia 4 du 1';  
} elseif ($a % 4 == 2) {  
    echo '$a chia 4 du 2';  
} else {  
    echo '$a chia 4 du 3';  
}
```

```
?>
```

## Rẽ nhánh if (tiếp)

---

- Ví dụ:

```
<?php  
    $a = 10;  
    $thongBao = $a % 2 == 0 ? '$a la so chan': '$a la  
so le';  
    echo $thongBao;  
?>
```

# switch

```
switch ($biến) {  
    case (giá_trị1):  
        nhóm_lệnh1; break;  
    case (giá_trị2):  
        nhóm_lệnh2; break;  
    ...  
    case (giá_trị_n):  
        nhóm_lệnh_n; break;  
    default:  
        nhóm_lệnh_khác;  
}
```

- Lưu ý: giá trị của \$biến phải là đếm được, rời rạc

# switch (tiếp)

---

- Ví dụ:

```
<?php  
    $thu = 3;  
    switch ($thu) {  
        case 1: echo 'Hom nay la chu nhat'; break;  
        case 2: echo 'Hom nay la thu 2'; break;  
        case 3: echo 'Hom nay la thu 3'; break;  
        case 4: echo 'Hom nay la thu 4'; break;  
        default: echo 'Hom nay co the la thu 5,6,7'; break;  
    }  
?>
```

# Lặp không xác định

```
while (BTLG) {  
    các_lệnh;  
}
```

các số từ 1 đến 10

```
do {  
    các_lệnh;  
} while (BTLG);
```

Ví dụ: in ra màn hình các số từ 1 đến 10

```
<?php $i++;  
    $i = 1;  
} while ($i <= 10);{  
?> echo "$i \n";  
    $i++;  
}  
?>
```

# Lặp "xác định"

```
for (khởi_tạo; kiểm_tra; hành_động) {  
    các_lệnh;  
}  
  
foreach ($biến_mảng as $giá_trị) {  
    các_lệnh;  
}  
  
foreach ($biến_mảng as $khoá=>$giá_trị) {  
    các_lệnh;  
}
```

# Lặp "xác định" (tiếp)

---

- Ví dụ: in ra màn hình các số từ 1 đến 10

```
<?php  
    for ($i = 1; $i <= 10; $i++) {  
        echo $i, "\n";  
    }  
?>
```

# Lặp "xác định" (tiếp)

---

- Ví dụ: in ra màn hình các số từ 1 đến 10

```
<?php  
    $dem = 1;  
    for ($count = ""; $count != "done"; ) {  
        if ($dem > 10) {  
            $count = "done";  
        } else {  
            echo $dem, "\n";  
            $dem++;  
        }  
    }  
?>
```

# Lặp "xác định" (tiếp)

---

- Ví dụ: in ra màn hình các số từ 1 đến 10

```
<?php  
    for ($count = 1, $total = 300;  
        $count <= 10 && $total > 200;  
        $count++, $total--) {  
        echo $count, "\n";  
    }  
?>
```

## **break và continue**

---

- **break**: kết thúc khối lệnh for, while, do-while hoặc switch
- **continue**: bỏ qua không thực hiện các lệnh nằm sau continue trong vòng lặp, chuyển sang vòng lặp tiếp.

## **break** và **continue** (tiếp)

---

- Thảo luận: cho biết kết quả của đoạn mã sau

```
<?php  
    echo 'bat dau <br/>';  
    for ($i = 1; $i < 10; $i++) {  
        echo "$i <br/>";  
        break;  
    }  
    echo 'ket thuc';  
?>
```

# **break** và **continue** (tiếp)

---

- Thảo luận: cho biết kết quả của đoạn mã sau

```
<?php  
    echo 'bat dau <br/>';  
    for ($i = 1; $i < 10; $i++) {  
        continue;  
        echo "$i <br/>";  
    }  
    echo 'ket thuc';  
?>
```

# Bài tập cấu trúc điều khiển

---

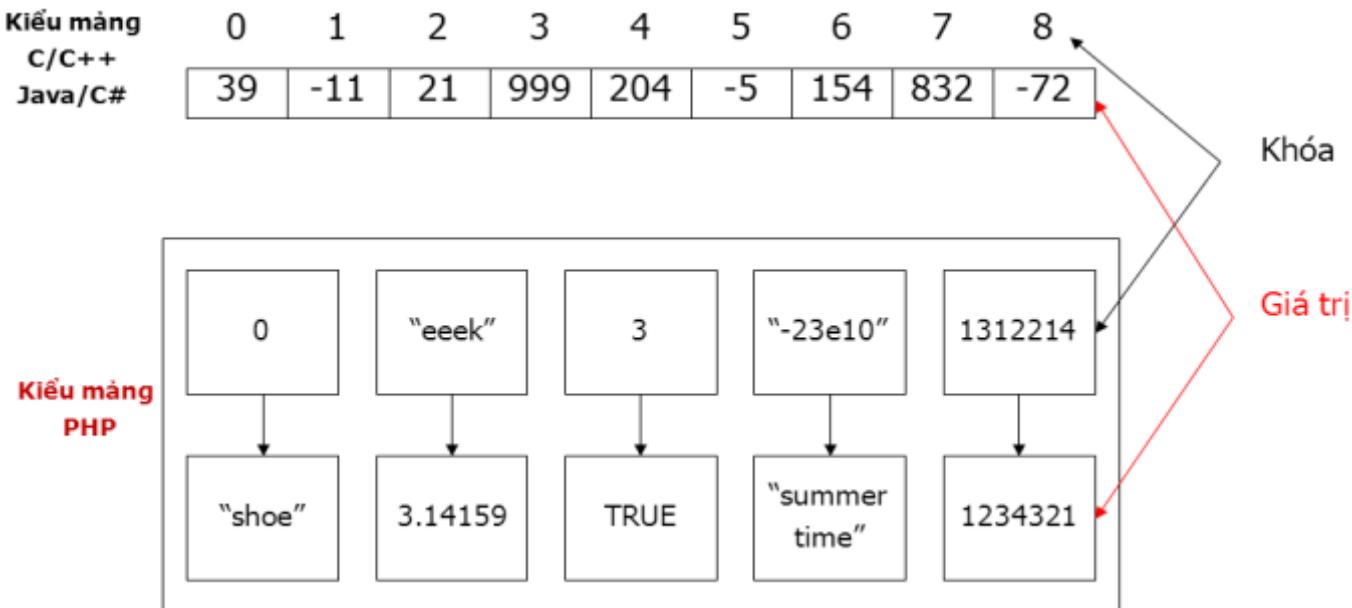
- Bài tập 1: Xuất ra trình duyệt bảng cửu chương từ 2 đến 9.
- Bài tập 2: Viết chương trình in ra các số nguyên tố  $\leq 20$
- Bài tập 3: viết chương trình in ra mà hình các số chẵn có 2 chữ số và nhỏ hơn 100

# Mảng

---

- Mảng trong PHP là một danh sách các dữ liệu có dạng `$key=>$value`, trong đó giá trị \$key của tất cả các phần tử trong danh sách là khác nhau.
  - \$key dùng để tham chiếu đến một phần tử của mảng.
  - \$key có thể có kiểu số nguyên hoặc kiểu chuỗi.
    - \$key trong PHP có ý nghĩa rộng hơn index (chỉ mục) trong các ngôn ngữ khác (Pascal, C, C++, C#, Java,...).
  - \$value có thể là bất kỳ kiểu dữ liệu gì (any type)

# Mảng (tiếp)



# Khai báo mảng

---

- Khai báo và gán giá trị

**\$tên\_bien = array([khóa=>]giá\_trị, [khóa=>]giá\_trị,...)**

- Trường hợp không định nghĩa các khóa (\$key) thì khóa của mảng sẽ được gán mặc định theo kiểu số nguyên liên tiếp tăng dần từ **0**.
- Trường hợp một khóa trong mảng có kiểu chuỗi hoặc số không liên tục thì mảng được gọi là *mảng kết hợp*.

# Khai báo mảng (tiếp)

---

- Ví dụ:

```
$a1 = array ('ipod', 'iphone', 'ipad');  
$a2 = array (product=>'laptop', brand=>'Dell',  
CPU=>'corei7', ram=>'8Gb', 5=>'USA');
```

**0 = 'ipod'**  
↓      ↓  
**\$key \$value**

**brand = 'Dell'**  
↓      ↓  
**\$key    \$value**

# Khai báo mảng (tiếp)

---

- Khai báo mảng (không khởi tạo giá trị)

\$tên\_biến = array(kích\_thước\_mảng);

- Tạo mảng động

\$tên\_biến = array();

- Mảng động là mảng chưa xác định kích thước của mảng khi khai báo.

- Kích thước có thể thay đổi linh hoạt khi sử dụng.

- Tạo mảng từ mảng có sẵn

\$tên\_mảng\_mới = \$tên\_mảng\_cũ;

- Tham trị hay tham chiếu? (BT: làm vd)

# Khai báo mảng (tiếp)

---

- Ví dụ:

```
/*Tạo và in mảng 10 phần tử có giá trị ngẫu nhiên từ 0-100 ($key mặc định)*/
$n = 10;
$a1 = array($n);
for ($i = 0; $i < $n; $i++)
    $a1[$i] = rand(0,100);
for ($i = 0; $i < $n; $i++)
    echo "phan tu {$i} la: {$a1[$i]}";
```

# Thao tác mảng

---

- Truy cập 1 phần tử của mảng

tên\_biến\_mảng [\$key]

- Chú ý về cách sử dụng \$key ở dạng chuỗi

- Các cách viết \$key=>\$value trong khai báo mảng \$a3 sau đều được chấp nhận.

```
$a3 = array('san pham'=>'laptop',  
            'brand'=>Dell, CPU=>"core i7",  
            "xuất xứ"=>"USA", 5=>1000);
```

- Tuy nhiên khuyến cáo là nên đặt \$key dạng chuỗi giữa cặp " hoặc '".

# Thao tác mảng (tiếp)

---

- Thêm một phần tử vào mảng

`tên_biến_mảng [$key] = giá_trị`

- Nếu mảng chưa tồn tại thì mảng sẽ được tạo và 1 giá trị sẽ được thêm vào.
- Nếu \$key không được chỉ định thì phần tử mới luôn được thêm vào cuối mảng
  - "\$key mới" có giá trị bằng \$key có giá trị số nguyên không âm lớn nhất hiện có trong mảng cộng thêm 1.
  - Nếu chưa có \$key nào như vậy thì "\$key mới" sẽ bằng 0.
- Nếu \$key được chỉ định
  - Nếu \$key chưa tồn tại thì 1 phần tử mới sẽ thêm vào cuối mảng
  - Nếu \$key đã tồn tại thì sẽ không có phần tử nào được thêm (lúc này giá trị cũ sẽ bị đè)

# Thao tác mảng (tiếp)

---

- Ví dụ:

```
$a=array(5=>'xin', 'chào', 'bạn');  
/*tương đương với 3 câu lệnh sau  
 $a[5]='xin';  
 $a[]='chào';      // $a[6]='chào'  
 $a[]='bạn';      // $a[7]='bạn' */
```

# Thao tác mảng (tiếp)

---

- Thảo luận:

```
$a[5] = 'LTWEB';
$a['khoa'] = 'CNTT';
$a[] = 'DHQN';           // ???
$a[5] = 'ITQNU';         // ???
```

```
$b=array(-3=>'xin', 'chào', 'bạn');
/*tương đương với 3 câu lệnh sau
$b[-3]='xin';
$b[]='chào';           // ???
$b[]='bạn';           // ??? */
```

# Thao tác mảng (tiếp)

---

- Thảo luận:

```
$c=array('san pham' => 'cong nghe',
          -3=>'dell', 'acer',
          false=>'apple', 'intel',
          true=>'hp', 'AMD',
          2.2=>'vaio', 'lg',
          '3.3'=>'lenovo',
          '5'=>'asus', 'fujisu',
          '09'=>'acer', 'toshiba');
```

// khi đó mảng \$c sẽ như thế nào???

# Thao tác mảng (tiếp)

---

- **Duyệt mảng**
  - Dùng các cấu trúc lặp:
    - **while** (điều\_kiện) { ... }
    - **do** { ... } **while** (điều\_kiện);
    - **for** (khởi\_tạo; kiểm\_tra; hành\_động) { ... }
    - **foreach** (\$biến\_mảng **as** [\$key=>] \$value)
      - Rất hữu dụng khi duyệt các mảng kết hợp. (vì với mảng kết hợp, cách truyền thông "không" áp dụng được)
  - Dùng con trỏ mảng
    - Kết hợp sử dụng các hàm: **each**(\$biến\_mảng), **list**(\$key, \$value)
    - Dùng các hàm điều khiển con trỏ

# Thao tác mảng (tiếp)

---

- Ví dụ **foreach**: in mảng a2 đã khai báo ở trên

```
foreach ($a2 as $khoa=>$giaTri) {  
    echo "$khoa -là- $giaTri <br/>";  
}
```

kết quả:

product –là– laptop

brand –là– Dell

CPU –là– corei7

ram –là– 8Gb

5 –là– USA

```
$a2 = array(product=>"laptop", brand=>"Dell",  
CPU=>"corei7", ram=>"8Gb", 5=>"USA");
```

# Thao tác mảng (tiếp)

---

- Ví dụ dùng **foreach** nhập/thay đổi giá trị các phần tử của mảng:

```
$arr = array(1,2,3,4);  
foreach($arr as &$gtri) {  
    $gtri = $gtri * 2;  
}  
// $arr = (2,4,6,8)
```

# Thao tác mảng (tiếp)

---

- Dùng hàm **each(\$biến\_mảng)**
  - Hàm each trả về phần tử kế tiếp của mảng.
  - Cặp \$key=>\$value của phần tử trả về được lưu trong mảng 4 phần tử có khóa là 0,1, key,value.
    - 0, key: chứa \$key
    - 1, value: chứa \$value
  - Ví dụ: in mảng a2 đã khai báo ở trên

```
while ($item = each($a2)) {  
    echo "{$item['key']} là {$item['value']}";  
}
```

# Thao tác mảng (tiếp)

---

- Ví dụ **each**: tạo và in mảng 10 phần tử có giá trị ngẫu nhiên từ 0-100.

```
<?php  
    $arr = array(10);  
    for($i = 1; $i <= 10; $i++)  
        $arr[] = rand(0,100);  
    while ($item = each($arr))  
        echo "{$item[1]} <br>"; // $item[0] ???  
?>
```

# Thao tác mảng (tiếp)

---

- Dùng hàm **list()**
  - Hàm `list($k,$v)` tách cặp giá trị *key=>value* của phần tử mảng ra hai biến `$k` và `$v`.
  - Ví dụ:  
in mảng `a2` đã khai báo ở trên  
`while (list($k,$v) = each($a2)) {  
 echo "$k là $v <br/>";  
}`

# Thao tác mảng (tiếp)

---

- Ví dụ **list**: tạo và in mảng 10 phần tử có giá trị ngẫu nhiên từ 0-100.

```
<?php  
    $arr = array(10);  
    for($i = 1; $i <= 10; $i++)  
        $arr[] = rand(0,100);  
    while (list($k,$v) = each($arr))  
        echo "$v <br>";  
?>
```

# Thao tác mảng (tiếp)

---

- Ví dụ **list**: tạo và in TKB.

```
<?php  
$arr = array('thứ hai'=>'Đi học',  
             'thứ ba'=>'Thực hành',  
             'thứ tư'=>'Đi chơi',  
             'thứ bảy'=>'Đìu hiu một mình',  
             'chủ nhật'=>'Còn tôi với nồng nàn');  
while (list($index,$value) = each($arr))  
    echo "$index - $value <br>";  
?>
```

# Thao tác mảng (tiếp)

---

- Duyệt mảng bằng con trỏ mảng:
  - Con trỏ mảng trỏ vào phần tử đầu tiên khi mảng được tạo ra.
  - Một số hàm di chuyển con trỏ mảng:
    - `current($mảng)`: trả về phần tử hiện tại
    - `prev($mảng)`: trả về phần tử liền trước
    - `next($mảng)`: trả về phần tử liền sau
    - `end($mảng)`: trả về phần tử cuối cùng của mảng
    - `reset($mảng)`: đưa con trỏ về đầu mảng

# Thao tác mảng (tiếp)

---

- Chú ý:
  - Sau mỗi lần duyệt mảng bằng hàm each() thì phải dùng hàm reset() để đưa con trỏ mảng trở lại đầu mảng trước khi duyệt tiếp.
  - Ví dụ:

```
<?php
    $arr = array(10);
    for($i = 1; $i <= 10; $i++)
        $arr[$i] = rand(0,100);
    while (list($k,$v) = each($arr))
        echo "$v <br>";
?>
```

# Các hàm xử lý mảng

---

- `is_array()`: kiểm tra một biến có phải là kiểu mảng.
  - trả về 1 (true) và 0 (false)
- `count()`, `sizeof()`: trả về số phần tử của mảng

# Các hàm xử lý mảng (tiếp)

---

- **unset()**: xóa phần tử
  - **unset(\$tên\_mảng[\$key])**: xóa phần tử có khóa **\$key** khỏi mảng.
  - **unset(\$tên\_mảng)**: xóa toàn bộ mảng.
  - Chú ý: hàm unset() không đánh lại "chỉ mục"  
Ví dụ:  

```
$a = array(1=>'one', 2=>'two', 3=>'three') ;
unset( $a[2] ) ;
/* kết quả là: $a = array(1=>'one', 3=>'three') ;
chú không là:
$a = array(1 =>'one', 2 =>'three') ;
*/
```

# Các hàm xử lý mảng (tiếp)

---

- Tạo mảng khóa số nguyên trung gian

`$mảng_khóa = array_keys ($mảng)`

- Truy xuất thông qua mảng khóa:

`$mảng [ $mảng_khóa [$i] ]`

# Các hàm xử lý mảng (tiếp)

---

- Các hàm sắp xếp mảng theo \$value của phần tử :
  - `sort($mảng)` : tăng dần
  - `rsort($mảng)` : giảm dần
  - `usort($mảng, 'hàm_so_sánh')`: tự định nghĩa thứ tự
  - `uasort($mảng, 'hàm_so_sánh')`: tự định nghĩa thứ tự

# Các hàm xử lý mảng (tiếp)

---

- Các hàm sắp xếp mảng theo \$key của phần tử :
  - **ksort(\$mảng)**: tăng dần
  - **krsort(\$mảng)**: giảm dần
  - **uksort(\$mảng, 'hàm\_so\_sánh')**: tự định nghĩa thứ tự

# Các hàm xử lý mảng (tiếp)

---

- Ví dụ: ksort()

- Cho mảng các phần tử SV=>bạn\_của\_SV\_đó
- In danh sách SV-BSV theo thứ tự Alphabet

```
<?php  
    $arr = array('Lê'=>'Phan Đã', 'Thị'=>'Đỗ Văn Vé', 'Bồ'=>'Hồ  
Như Ao', 'Câu'=>'Tạ Thị Tấn');  
    ksort($arr);  
    echo 'Student - His friend<br>';  
    while (list($person, $friend) = each($arr))  
        echo "$person - $friend <br>";  
?>
```

# Các hàm xử lý mảng (tiếp)

---

- Ví dụ: usort()

```
<?php
    function cmp($a, $b) {
        if ($a == $b) {
            return 0;
        }
        return ($a < $b) ? -1 : 1;
    }

    $a = array(3, 2, 5, 6, 1);
usort($a, 'cmp');

    foreach ($a as $key => $value) {
        echo "$key: $value\n";
    }
?>
```

# Các hàm xử lý mảng (tiếp)

---

- Nối ghép hai mảng
  - array\_merge(\$mảng1, \$mảng2)
  - array\_combine(\$mảng1, \$mảng2)
  - array\_intersect(\$mảng1, \$mảng2)
- Tách mảng
  - array\_slice(\$mảng, vị trí tách, số lượng tách)
- Tìm kiếm
  - array\_search(\$giá\_trị, \$mảng)
  - in\_array(\$giá trị tìm, \$mảng)

# Ví dụ về mảng động

---

- Cho mảng gồm 10 số tự nhiên, tạo và in ra mảng con chứa những số lẻ của mảng này.

```
<?php  
    $arr = array(2, 4, 9, 2, 3, 5, 8, 1, 4, 7);  
    $child = array(); //sử dụng mảng động  
    while (list($k, $v) = each($arr))  
        if ($v % 2 != 0)  
            $child[] = $v; //thêm ptử vào mảng con  
    echo 'So phan tu cua mang con: '.count($child);  
    for($i = 0; $i < count($child); $i++)  
        echo "<br/> $child[$i]"  
?>
```

# Một số hàm xử lý chuỗi

---

- **strlen(\$str)**: trả về độ dài của chuỗi *\$str*.
- **substr(\$str, \$pos, \$len)**: trả về chuỗi con tách từ chuỗi *\$str* từ vị trí *\$pos* và lấy *\$len* ký tự.
- **strpos(\$str, \$sub)**: trả về vị trí xuất hiện lần đầu tiên của chuỗi *\$sub* trong chuỗi *\$str*.
- **str\_replace(\$rep, \$with, \$str)**: thay thế chuỗi *\$rep* bằng chuỗi *\$with* trong chuỗi *\$str*.
- **strtoupper (\$chuỗi)**: chuyển chuỗi thành các ký tự in hoa
- **strtolower (\$chuỗi)**: chuyển \$chuỗi thành các ký tự thường

# Một số hàm xử lý chuỗi (tiếp)

- **explode(\$separator, \$str)**: trả về mảng các chuỗi con được tách từ chuỗi \$str dựa trên dấu phân cách \$separator.
- Ví dụ:

```
<?php  
$danhSach      =      'Hằng      Hạnh      Mai      Thủy      Lý';  
$mang          =      explode('      ',      $danhSach);  
echo 'Danh sách ngũ cô nương K33 CNTT: <br/>';  
foreach ($mang as $key=>$value)  
    echo ($key+1) . ' - ' . $value . '<br/>';  
?>
```

# Một số hàm xử lý chuỗi (tiếp)

---

- `implode($sep, $arr)`: nối các chuỗi con trong mảng `$arr` thành 1 chuỗi phân cách nhau bởi `$sep`.
- Ví dụ:

```
<?php
$array = array('lastname', 'email', 'phone');
$comma_separated = implode(',', $array);
echo $comma_separated; // lastname,email,phone
?>
```

# Một số hàm xử lý chuỗi (tiếp)

---

- **trim()**, **ltrim()**, **rtrim()**: cắt các ký tự trắng ở 2 đầu chuỗi.
- **nl2br()**: (định dạng HTML) chuyển các ký tự **\n** trong chuỗi thành **<br>**
- **addslashes()** : thêm các dấu backslashes "**\**" vào trước các ký tự đặc biệt (vd: ' " \ null ...) (định dạng CSDL) chuyển các ký tự đặc biệt (vd: -> \) trong chuỗi để có thể lưu trữ được trong CSDL.

# Một số hàm xử lý chuỗi (tiếp)

---

- Ví dụ: Chuẩn hóa chuỗi văn bản (không có ký tự trùng ở hai đầu, giữa 2 từ chỉ có một ký tự trùng)

```
<?php  
$str = '    Thầy    Hồ    Anh    Minh    ';  
  
function normalize($st) {  
    $st = trim($st);  
    $mot=' '; $hai=' ';  
    while(strpos($st,$hai) !== false)  
        $st = str_replace($hai,$mot,$st);  
    return $st;  
}  
  
echo 'Kết quả: '. normalize($str);  
?>
```

# HTML FORM

---

- INPUT Text:

```
<input type="text" name="tên_biến"  
       value="giá trị" />
```

- INPUT Checkbox:

```
<input type="checkbox" name="tên_biến"  
       value="giá trị" [checked] />
```

- INPUT Radio:

```
<input type="radio" name="tên_biến"  
       value="giá trị" [checked] />
```

- INPUT Submit:

```
<input type="submit" name="tên_biến"  
       value="giá trị" />
```

# HTML FORM (tiếp)

---

- List Box:

```
<select name="tên_biến">
    <option value="giá trị 1">
        chuỗi hiển thị 1
    </option>
    ...
    <option value="giá trị n">
        chuỗi hiển thị n
    </option>
</select>
```

# HTML FORM (tiếp)

---

- Multiline List Box:

```
<select multiple name="tên_biến_mảng[] ">
    <option value="giá trị 1">
        chuỗi hiển thị 1
    </option>
    ...
    <option value="giá trị n">
        chuỗi hiển thị n
    </option>
</select>
```

# Truyền dữ liệu của Form HTML

---

- Thẻ <form> trong HTML có 2 thuộc tính:
  - action="chuỗi": url của trang web nhận dữ liệu (thường là url của trang PHP xử lý dữ liệu)
    - Nếu trang xử lý chính là trang của thẻ Form thì khai báo:

```
action=<?php echo $_SERVER['PHP_SELF'] ?>"
```
  - method="chuỗi": phương thức gửi dữ liệu đến trang xử lý. Có 2 cách:
    - GET (mặc định)
    - POST

# Phương thức GET

---

- Khi ta Submit 1 Form sử dụng phương thức GET, dữ liệu được truyền qua URL theo từng cặp tham số=giá trị:
  - Tên các tham số là thuộc tính "name" của các đối tượng trên form
  - Giá trị tham số là giá trị được nhập vào đối tượng (hoặc giá trị của thuộc tính "value" của đối tượng)
  - Các cặp phân cách bởi dấu & và phân cách với url nguyên thủy của trang bởi dấu ?
  - Ví dụ:

`http://localhost/gptb2.php?a=1&b=2&c=-3`

# Phương thức POST

---

- Khác với phương thức GET, khi sử dụng phương thức POST, dữ liệu được truyền trong thân của yêu cầu HTTP, và cũng theo từng cặp tham số=giá trị :
  - Người dùng không thấy được các giá trị này

# Đọc dữ liệu gửi từ Form

---

- PHP cung cấp 2 mảng **`$_GET`** và **`$_POST`** để lấy các giá trị được gửi từ form.
  - Mảng **`$_GET`** dùng để đọc giá trị gửi đi bằng phương thức GET
  - Mảng **`$_POST`** dùng để đọc giá trị gửi đi bằng phương thức POST
  - \$key của các mảng là tên các tham số tương ứng.

# So sánh GET & POST

GET	POST
<ul style="list-style-type: none"><li>Dữ liệu được gắn thêm vào URL khi gọi script.</li><li>Các dữ liệu được đưa vào biến mảng siêu toàn cục <code>\$_GET</code> với khóa tương ứng với tên các thành phần input trong form</li><li>Nên dùng trong trường hợp dữ liệu chỉ dùng để truy vấn, không đòi hỏi bảo mật</li><li>Không hỗ trợ uploading file</li><li>Chỉ hỗ trợ bảng mã ASCII chuẩn</li></ul>	<ul style="list-style-type: none"><li>Dữ liệu được nhúng vào trong HTTP request khi gửi đến server</li><li>Các dữ liệu được đưa vào biến mảng siêu toàn cục <code>\$_POST</code> với khóa tương ứng với tên các thành phần input trong form</li><li>An toàn hơn so với khi dùng GET nên được dùng phổ biến hơn</li><li>Hỗ trợ uploading file</li><li>Hỗ trợ nhiều bảng mã</li></ul>

# Đọc dữ liệu gửi từ Form

---

- Ví dụ: đọc các biến a, b, c trong trang gptb2.php ở ví dụ trên:

- Gửi bằng GET

```
$a = $_GET['a'];  
$b = $_GET['b'];  
$c = $_GET['c'];
```

- Gửi bằng POST

```
$a = $_POST['a'];  
$b = $_POST['b'];  
$c = $_POST['c'];
```

# Đọc dữ liệu gửi từ Form (tiếp)

---

- Ví dụ:

- Trang "gui.php"

```
<form action="nhan.php" method="POST">
    <input type="text" name="hoten">
    <input type="submit" name="Gui">
</form>
```

- Trang "nhan.php"

```
<?php
    $hten = $_POST['hoten'];
    echo $hten;
?>
```

# Đọc dữ liệu gửi từ Form (tiếp)

---

- Ví dụ: nhận dữ liệu gửi từ các checkbox cùng tên
  - Trang "gui.php"

```
<form action="nhan.php" method="POST">  
    <input type="checkbox" name="box[]" value=1 />Một  
    <input type="checkbox" name="box[]" value=2 />Hai  
    <input type="checkbox" name="box[]" value=3 />Ba  
    <input type="submit" name="Sumit">  
</form>  
• Trang "nhan.php"  
<?php  
    $val = $_POST['box'];      // $val là mảng  
    foreach($val as $k=>$v)  
        echo "$v <br>";  
?>
```

# Biến \$REQUEST

---

- Lấy các giá trị của GET, POST, COOKIE.
- Chú ý:
  - Các phần tử trong mảng REQUEST hoàn toàn độc lập với các phần tử trong GET, POST,...
  - Ta có thể thay đổi các giá trị trong mảng REQUEST mà không ảnh hưởng đến các giá trị lưu trong GET, POST,...

## Ví dụ mẫu

---

- Xây dựng 1 trang HTML với nội dung gồm form cho phép nhập họ tên. Sau đó dùng 1 file PHP để xuất ra thông tin họ tên mà người sử dụng vừa nhập.

## Ví dụ mẫu (tiếp)

---

- Tạo file dangnhap.php để xuất ra dữ liệu với nội dung như sau:

```
<html>
    <head><title>Vi du mau</title></head>
    <body>
        <form action="xuly.php" method="POST">
            Nhập họ tên: <br/>
            <input type="text" name="hoten" /> <br/>
            <input type="submit" value="Gửi" />
        </form>
    </body>
</html>
```

## Ví dụ mẫu (tiếp)

---

- Tạo file xuly.php để xuất ra dữ liệu với nội dung như sau:

```
<html>
  <head><title>Vi du mau</title></head>
  <body>
    Chào mừng
    <?php
      echo $_POST['hoten'];
    ?>
  </body>
</html>
```

# Ví dụ tổng hợp

---

- Tạo file info.php để nhập dữ liệu như sau:

```
<html>
  <head><title>Ví dụ tổng hợp</title></head>
  <body>
    <form name="myform" method="post" action="xly.php">
      <p>Name <input type="text" name="txtname"/></p>
      <p>Job <input type="text" name="txtjob"/></p>
      <p>Giới tính
        <input type="radio" name="rdbsex" value="Nam"/>Nam
        <input type="radio" name="rdbsex" value="Nữ"/>Nữ
      </p>
    </form>
  </body>
</html>
```

## Ví dụ tổng hợp (tiếp)

---

```
<p>Quê quán</p>
<select name="selhomeland">
    <option value="Hà Nội"> Hà Nội </option>
    <option value="Huế"> Huế </option>
    <option value="Đà Nẵng"> Đà Nẵng </option>
    <option value="Quy Nhơn" selected="selected">
        Quy Nhơn </option>
    <option value="Hồ Chí Minh"> Hồ Chí Minh </option>
</select>
```

# Ví dụ tổng hợp (tiếp)

---

```
<p>Ngoại ngữ</p>
<input type="checkbox" name="ckb[]"
       value="Anh"/> Tiếng Anh
<input type="checkbox" name="ckb[]"
       value="Pháp"/> Tiếng Pháp
<input type="checkbox" name="ckb[]"
       value="Nhật"/> Tiếng Nhật
<p>Thông tin thêm</p>
<textarea name="txtmore" cols="45" rows="5">
</textarea>
<p><input type="submit" value="Gửi" /></p>
</form>
</html>
```

# Ví dụ tổng hợp (tiếp)

---

- Tạo file xuly.php như sau:

```
<?php
$name = $_POST['txtname'];
$job = $_POST['txtjob'];
$sex = $_POST['rdbsex'];
$homeland = $_POST['selhomeland'];
$language = '';
foreach ($_POST['ckb'] as $value) {
    $language .= $value.'<br />';
}
$more = $_POST['txtmore'];
// xử lý tiếp...
?>
```

# Bài tập

---

- Bài 1: tạo 1 trang web với hộp thoại nhập liệu username và password.
  - Nếu người sử dụng nhập thông tin username, password là admin, 123456 thì xuất ra thông báo "Welcome to admin" với kiểu chữ Tahoma, màu đỏ.
  - Ngược lại nếu nhập sai thì xuất thông báo "Username hoặc password không chính xác. Vui lòng đăng nhập lại".

# Bài tập (tiếp)

- Bài 2:

KẾT QUẢ HỌC TẬP	
Điểm HK1:	7
Điểm HK2:	8.5
Điểm trung bình:	8
Kết quả:	Được lên lớp
Xếp loại học lực:	Giỏi
<a href="#">Xem kết quả</a>	

KẾT QUẢ THI ĐẠI HỌC	
Toán:	8.5
Lý:	6.5
Hoá:	8
Điểm chuẩn:	20
Tổng điểm:	23
Kết quả thi:	Đầu
<a href="#">Xem kết quả</a>	

# Bài tập (tiếp)

---

- Bài 3:

**DIỆN TÍCH HÌNH CHỮ NHẬT**

Chiều dài:	<input type="text" value="15"/>
Chiều rộng:	<input type="text" value="20"/>
Diện tích:	<input type="text" value="300"/>

**Tính**

# Bài tập (tiếp)

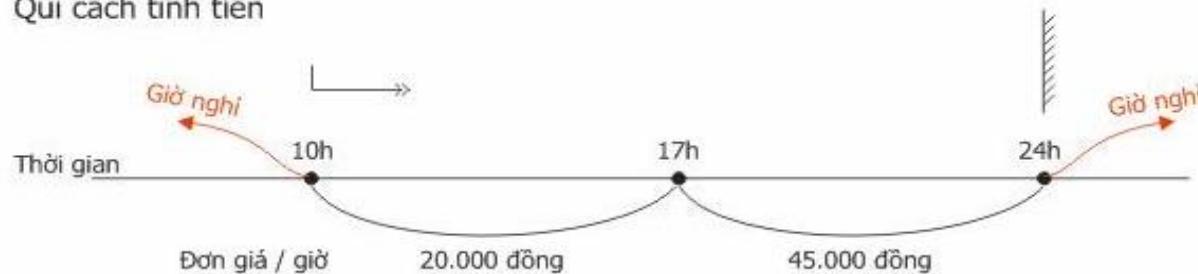
- Bài 4:

**TÍNH TIỀN KARAOKE**

Giờ bắt đầu:	17	(h)
Giờ kết thúc:	20	(h)
Tiền thanh toán:	135000	(VNĐ)

**Tính tiền**

Qui cách tính tiền



# Bài tập (tiếp)

- Bài 5:



**ĐỔI NGOẠI TỆ - VÀNG**

ĐỔI NGOẠI TỆ		MUA VÀNG	
Số tiền:	10	Số vàng:	10 (chi)
Loại ngoại tệ:	Đô la Úc	Loại vàng:	<input type="radio"/> SJC <input checked="" type="radio"/> AAA <input type="radio"/> PNJ
Tỷ giá VND:	14.057	Đơn giá:	1.305.000
Thành tiền VND:	140.570	Thành tiền VND	13.050.000

Quy đổi ngoại tệ / vàng

# Sử dụng lại mã lệnh

---

- Để sử dụng lại mã lệnh, ta có thể sử dụng 4 hàm sau:
  - include ('địa\_chỉ\_file');
  - require ('địa\_chỉ\_file');
  - include\_once ('địa\_chỉ\_file');
  - require\_once ('địa\_chỉ\_file');

# Sử dụng lại mã lệnh (tiếp)

---

- Phân biệt include() và require()
  - include('tên\_tệp'): nếu tên\_tệp không tồn tại thì PHP Thông báo lỗi và vẫn tiếp tục thực thi script còn lại.
  - require('tên\_tệp'): nếu tên\_tệp không tồn tại thì PHP thông báo lỗi (*fatal*) và dừng thực thi script.
- include\_once() và require\_once() tương tự như include() và require() nhưng chỉ thực hiện 1 lần duy nhất.

# Sử dụng lại mã lệnh (tiếp)

---

- Ví dụ: chèn code của 1 file khác vào trang php hiện tại
  - File khaiBao.php

```
<?php  
    $color = 'green'; $fruit = 'apple';
```

```
?>
```

- File test.php

```
<?php  
    echo "A $color $fruit"; //A  
    include 'khaiBao.php';  
    echo "A $color $fruit"; //A green apple  
?>
```

# Sử dụng lại mã lệnh (tiếp)

---

- Ví dụ: trang Index.php sẽ được chia thành 3 khối:
  - header.php: gồm các khai báo <html>... <body>
  - menu.php : gồm các liên kết
  - footer.php: gồm các thông tin về website và </body></html>

# Sử dụng lại mã lệnh (tiếp)

---

- Trang index.php

```
<?php  
    include ('header.php') ;  
    include ('menu.php') ;  
    include ('footer.php') ;  
?>
```

# Sử dụng lại mã lệnh (tiếp)

---

- Trang header.php

```
<html>
<head>
    <style>
        body{margin:10 10 10 60; color:navy;
              font: normal 18pt Arial;}
    </style>
</head>
<body>
    <H2> Ví dụ về include()</H2>
    <?php
        echo '<div>Hôm nay: '.date('d-m-y').
              '</div><br/>';
    ?>
```

# Sử dụng lại mã lệnh (tiếp)

---

- Trang menu.php

```
<?php  
    $strMenu=<a href=' #'>Home</a><br/>;  
    $strMenu.=<a href=' #'>PHP</a><br/>;  
    $strMenu.=<a href=' #'>JAVA</a><br/>;  
    $strMenu.=<a href=' #'>C#</a><br/>;  
    echo $strMenu;  
?>
```

# Sử dụng lại mã lệnh (tiếp)

---

- Trang footer.php

```
<?php  
$copyright='<hr/><div class=ft>  
    Copyright by NTB'.date('Y').'</div>';  
echo $copyright;  
?>  
</body>  
</html>
```

# Bài tập

--> Đây là TabMenu

--> Đây là Header

### Thông tin cá nhân

Họ và tên (\*)

Email (\*)

Điện thoại (\*)

Giới tính  Nam  Nữ

Công việc hiện tại (\*)

Quê quán   Dã ngoại  Ăn uống  Phim ảnh  Khiêu vũ  
 Đao bô  Nghỉ mát  Mua sắm  Câu cá

Mô tả thêm về bản thân

**Đăng ký**

### Đăng nhập

Tài khoản

Mật khẩu

**Đăng nhập**

Các trường đánh dấu (\*) là bắt buộc phải điền và điền chính xác

--> Đây là Footer

# Hàm tự tạo

---

- Hàm do người dùng định nghĩa
  - Cho phép xử lý các tác vụ thường lặp lại trong ứng dụng.
  - Phân tách các đoạn chương trình phức tạp thành các phần nhỏ hơn, giúp đoạn chương trình rõ ràng, trong sáng.

# Hàm tự tạo

---

- Khai báo hàm

```
function tên_hàm ([ts1] [,ts2],... [,tsN])
```

```
{
```

```
//các lệnh trong thân hàm
```

```
}
```

- Gọi hàm:

```
    tên_hàm(gt1, gt2,... , gtn);
```

# Hàm tự tạo (tiếp)

---

- Quy tắc viết tên \_ hàm:
  - Chỉ gồm các ký tự chữ cái, chữ số, gạch dưới.
  - Bắt đầu bằng chữ cái hoặc dấu gạch dưới
  - Không được trùng với hàm đã định nghĩa
- Nếu hàm không có tham số, khi khai báo hàm, gọi hàm vẫn phải giữ lại ()

# Hàm tự tạo (tiếp)

---

- Để trả ra giá trị cho hàm ta sử dụng lệnh :

`return biểu_thúc;`

- Nếu không có lệnh `return`, thì mặc định hàm sẽ trả về giá trị `NULL`
- Muốn trả về hơn 1 giá trị thì phải dùng mảng.
- Để thoát khỏi hàm ta dùng lệnh
  - `exit;`
  - `return;`

# Hàm tự tạo (tiếp)

---

- Ví dụ hàm không có giá trị trả về:

```
<?php  
function hienThi($text) {  
    echo $text;  
}  
hienThi('Hello World!');  
hienThi('Bonjour le monde!');  
?>
```

# Hàm tự tạo (tiếp)

---

- Ví dụ hàm có giá trị trả về:

```
<?php  
function tinh tong ($a, $b) {  
    $total=$a+$b;  
    return $total;  
}  
echo tinh tong (2, 3);  
?>
```

# Truyền tham chiếu

---

- Mặc định, các tham số truyền theo tham trị. Để truyền tham số như tham chiếu, ta thêm ký tự **&** trước tham số trong khai báo hàm cũng như lúc gọi hàm.

- Khai báo

```
function tên_hàm(&$tham_so, ...){  
    // các lệnh  
}
```

- Gọi hàm

```
function tên_hàm(&$tham_so, ...);
```

# Hàm có tham số mặc định

---

- Hàm có tham số mặc định:

```
function tên_hàm(ts1=gt1, ts2=gt2, ...)
```

- Chú ý:
  - Thông thường, loại tham số này đặt ở cuối danh sách.
  - Khi gọi hàm, nếu bỏ trống tại các vị trí tham số có giá trị mặc định thì giá trị mặc định sẽ được dùng cho tham số đó.

# Hàm có tham số mặc định (tiếp)

---

- Ví dụ hàm có tham số có giá trị mặc định:

```
<?php  
function tinh tong ($a=1, $b=2) {  
    $total=$a+$b;  
    return $total;  
}  
echo tinh tong(2,3); //kq: 5  
echo tinh tong(); //kq: 3  
?>
```

# Hàm có tham số mặc định (tiếp)

---

- Ví dụ hàm có tham số có giá trị mặc định:

```
<?php  
function cafe($type = 'G7') {  
    echo "1 coc cafe $type.";  
}  
echo cafe('sữa đá'); //1 coc cafe sữa đá.  
echo cafe(); //1 coc cafe G7.  
echo cafe(null); //1 coc cafe .  
?>
```

# Hàm có tham số không xác định

---

- Khi khai báo hàm, ta có thể không khai báo bất kỳ tham số nào  
function tên\_hàm() { . . . }
- Tuy nhiên khi gọi hàm, ta lại truyền các tham số cho nó. Khi đó, ta nói hàm có số lượng tham số không xác định.

# Hàm có tham số không xác định (tiếp)

---

- Để thao tác với các tham số của hàm, PHP cung cấp một số hàm sau:
  - func\_num\_args(): trả về số lượng tham số khi hàm được gọi
  - func\_get\_arg(i): trả về giá trị các tham số thứ i được truyền (tính từ 0)
  - func\_get\_args(): danh sách tất cả các tham số

# Hàm có tham số không xác định (tiếp)

---

- Ví dụ:

```
<?php
    function foo() {
        $sluong = func_num_args();
        echo "So tham so: $sluong <br/>";
    }
    foo(1, 2, 3);      //So tham so: 3
?>
```

# Hàm biến

---

- Khi một biến kiểu chuỗi được khai báo và gán giá trị trùng khớp với tên một hàm được định nghĩa thì tên biến đó có thể được dùng như một cách gọi hàm khác với cách gọi hàm bình thường bằng tên hàm.
- Một số hàm không thể dùng như hàm biến
  - echo, print, print\_r
  - var\_dump
  - isset, unset, is\_type, is\_null

# Hàm biến (tiếp)

---

- Ví dụ:

```
<?php
    function foo() {
        echo 'trong ham foo() <br/>';
    }
    function bar($arg = '') {
        echo "Trong ham bar(). Tham so la '$arg'.<br/>";
    }

    $func = 'foo'; $func();                      // Goi ham foo()
    $func = 'bar'; $func('test');                // Goi ham bar()
?>
```

# Bài tập

---

**Bài 1:** Viết mã lệnh để thực hiện cho nút lệnh đăng nhập trong bài 1 chương 1 với username cho trước là “hvlam” và password là ‘123456’.

**Bài 2:** Viết code tính tiền cho một form giỏ hàng với số lượng và đơn giá nhập vào từ bàn phím.

**Bài 3:** Viết form đổi các đơn vị tiền tệ phổ biến hiện nay, người dùng thực hiện thao tác nhập số tiền và chọn các đơn vị tiền tệ cần chuyển đổi ngay lập tức nhận được kết quả.

**Bài 4:** Sau khi đăng nhập thành công ở bài 1, ứng dụng sẽ chuyển sang trang index.php và viết mã lệnh cho nút upload bài thi trong trang này và các chức năng có trên trang index.php.

**Bài 5:** Viết hàm đếm ngược thời gian cho một ứng dụng Web được tạo bằng ngôn ngữ PHP.

# Chương IV: Giới thiệu MySQL

---

## Cú pháp truy vấn

- Truy xuất dữ liệu
  - Cú pháp: `SELECT tên_cột FROM tên_bảng`  
[`WHERE` điều\_kiện]  
[`ORDER BY` tên\_cột `ASC|DESC`]  
[`LIMIT` i,n]
  - `ORDER BY ASC|DESC`: kết quả trả về được sắp xếp tăng/giảm.
  - `LIMIT` vị\_trí\_bắt\_đầu, số\_bản\_ghi\_muốn\_truy\_xuất

# Cú pháp truy vấn

---

- Truy xuất dữ liệu (tiếp)
  - Ví dụ:

SELECT username, email FROM user WHERE user\_id = 1

SELECT \* FROM user WHERE user\_id = 1

SELECT username, email FROM user LIMIT 0,5

User_id	Username	Password	Email
1	admin	adminpass	admin@mail.com
2	user1	123456	user1@mail.com

# Cú pháp truy vấn

---

- Thêm dữ liệu
  - Cú pháp

```
INSERT INTO tên_bảng [(danh_sách_cột)] VALUES (danh_sách_các_giá_trị_tương ứng)
```
  - *Chú ý: Nếu không có danh sách cột thì số lượng danh sách các giá trị phải khớp với số cột và thứ tự của cột trong bảng.*

# Cú pháp truy vấn

---

- Thêm dữ liệu (tiếp)

- Ví dụ:

```
INSERT INTO user (username, password) VALUES ('binh', '1234')
```

```
INSERT INTO user VALUES ('3', 'binh', '1234', 'binh@mail.com')
```

# Cú pháp truy vấn

---

- Cập nhật dữ liệu
  - Cú pháp  
**UPDATE tên\_bảng SET tên\_cột=giá\_trị\_mới [WHERE điều\_kiện]**
  - Ví dụ:  
    **UPDATE user SET email="admin@mail.com" WHERE user\_id = 1**
  - Chú ý: Nếu không có phần điều\_kiện thì câu lệnh sẽ cập nhật giá trị mới cho toàn bộ các bản ghi có trong bảng

# Cú pháp truy vấn

---

- Xóa dữ liệu
  - Cú pháp  
**DELETE FROM tên\_bảng [WHERE điều\_kiện]**
  - Ví dụ:  
**DELETE FROM user WHERE user\_id = 1**
  - Chú ý: Nếu không có phần điều\_kiện thì câu lệnh sẽ xóa toàn bộ các bản ghi có trong bảng

# Các lệnh thông dụng trong MySQL

---

Kiểu	Mô tả
CREATE	tạo CSDL hoặc bảng
ALTER	thay đổi bảng có sẵn
SELECT	chọn dữ liệu từ bảng
DELETE	xóa dữ liệu khỏi bảng
DESCRIBE	xem thông tin mô tả về cấu trúc bảng
INSERT INTO	ghi giá trị vào bảng
UPDATE	cập nhật dữ liệu đã có trong bảng
DROP	xóa bảng hay toàn bộ CSDL

# Một số kiểu dữ liệu trong MySQL

---

Kiểu dữ liệu	Mô Tả
Char	Định dạng text có chiều dài từ 0->255
Varchar	Định dạng text có chiều dài từ 0->255
Text	Định dạng text có chiều dài 0->65535
Longtext	Định dạng text có chiều dài 0->4294967215
INT	Định dạng số có chiều dài từ 0->4294967215
Float	Định dạng số thập phân có chiều dài nhỏ
Double	Định dạng số thập phân có chiều dài lớn
Date	Định dạng thời gian theo định dạng: YYYY-MM-DD
DateTime	Định dạng thời gian theo định dạng: YYYY-MM-DD HH:MM:SS

# Các kiểu dữ liệu cơ bản trong MySQL

---

Kiểu	Mô tả
char( <i>length</i> )	tối đa 255 ký tự, chiều dài cố định = <i>length</i>
varchar( <i>length</i> )	tối đa 255 ký tự, chiều dài động <= <i>length</i>
text	tối đa 65536 ký tự
int( <i>length</i> )	-2.147.483.648 đến +2.147.483.647
decimal( <i>length,dec</i> )	tối đa <i>length</i> chữ số trong đó <i>dec</i> chữ số thập phân

# Các kiểu dữ liệu cơ bản trong MySQL

---

Kiểu	Mô tả
enum("option1", "option2",...)	tập hợp tự định, nghĩa tối đa 65.535 giá trị
date	yyyy-mm-dd
time	hh:mm:ss
datetime	yyyy-mm-dd hh:mm:ss

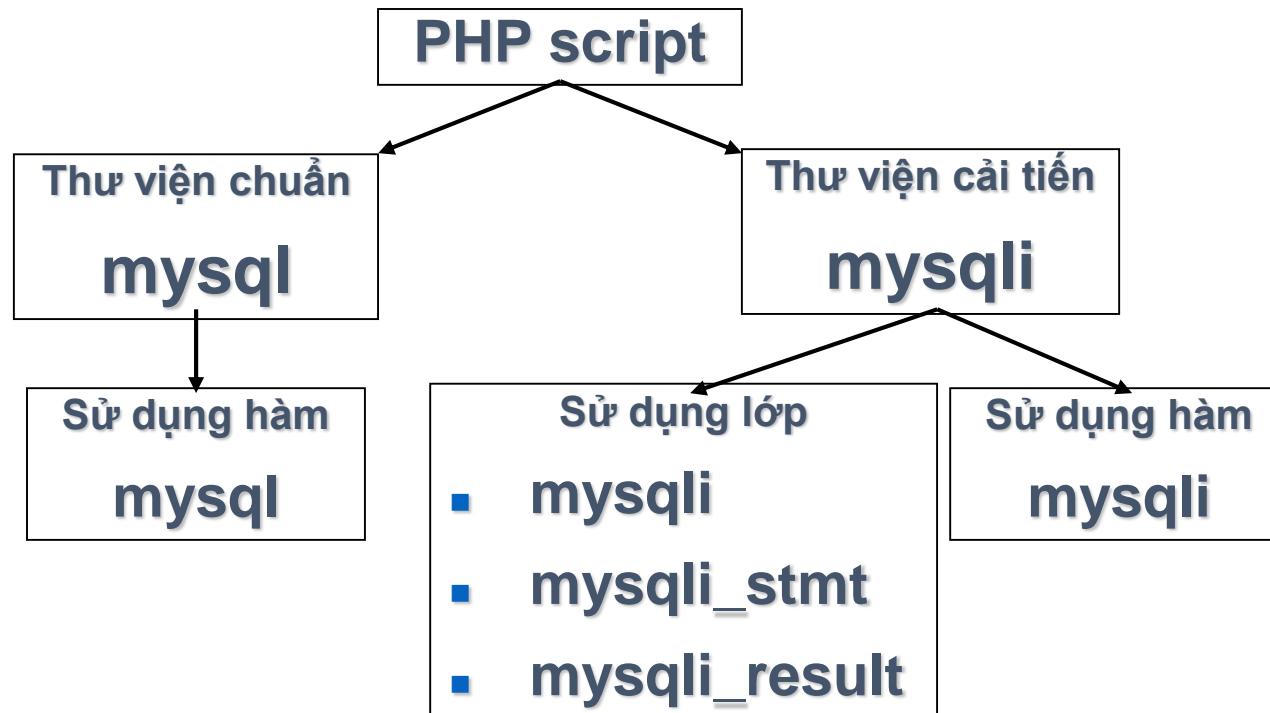
# Một số thuật ngữ

---

- NULL : Giá trị cho phép rỗng.
- AUTO\_INCREMENT : Cho phép giá trị tăng dần (tự động).
- UNSIGNED : Phải là số nguyên dương
- PRIMARY KEY : Cho phép nó là khóa chính trong bảng.

# Chương V: MySQL và PHP

## Kết nối MySQL - PHP



# Thư viện mysql cài tiến trong PHP5

---

- Thiết lập trong php.ini
  - extension=php\_mysqli.dll
- Ưu điểm
  - Hỗ trợ lập trình hướng đối tượng
  - Hỗ trợ nhân bản và phân tán CSDL
  - Nén và mã hóa dữ liệu trên kết nối
  - Tối ưu hiệu năng và mã

# Bước 1 - Tạo kết nối

---

- Dùng hàm mysql:

```
$biến_kết_nối =  
mysql_connect("máy_chủ", "tên_đăng_nhập", "mật_khẩu");
```

- Dùng hàm mysqli:

```
$biến_kết_nối =  
mysqli_connect("máy_chủ", "tên", "mật_khẩu", "CSDL_chọn");
```

- Dùng đối tượng mysqli (OOP):

```
$biến_kết_nối = new mysqli("máy_chủ", "tên",  
"mật_khẩu", "CSDL_chọn");
```

# Bước 1 - Tạo kết nối (tiếp)

---

- Để tránh lỗi thì ta kết hợp thêm hàm die:

```
$biến_kết_nối = mysql_connect("máy_chủ",  
                                "tên_đăng Nhập", "mật_khẩu")  
or die("Không kết nối được");
```

- Chú ý:
  - Hàm die ("chuỗi thông báo") : Đưa ra thông báo và kết thúc.
  - Với cách viết trên thì lệnh die () chỉ thực hiện khi lệnh trước nó không thành công.

## Bước 2 - Lựa chọn CSDL

---

- Dùng hàm mysql:

```
mysql_select_db("Tên CSDL", $biến_kết_nối)  
or die ("Chưa có CSDL");
```

- Dùng hàm mysqli:

```
mysqli_select_db($biến_kết_nối, "Tên CSDL")  
or die ("Chưa có CSDL");
```

- Dùng đối tượng mysqli (OOP):

```
$biến_kết_nối->select_db("Tên CSDL")  
or die ("Chưa có CSDL");
```

## Bước 3 – Chọn bảng mã (nếu cần)

---

- Dùng hàm mysql:

```
mysql_query ("SET NAMES 'character set'"  
            [, $biến_kết_nối]);
```

- Dùng hàm mysqli:

```
mysqli_query ([ $biến_kết_nối, ]  
              "SET NAMES 'character set'");
```

- Dùng đối tượng mysqli (OOP):

```
$biến_kết_nối->query (  
    "SET NAMES 'character set'");
```

## Bước 3 – Chọn bảng mã (nếu cần)

---

- Ví dụ:

```
mysql_query("SET NAMES 'utf8' ", $conn);  
mysqli_query($conn, "SET NAMES 'utf8' ");  
$conn->query("SET NAMES 'utf8' ");
```

## Bước 4 – Xây dựng truy vấn

---

- Xây dựng truy vấn:

```
$sql = "Câu lệnh SQL";
```

- Câu truy vấn có thể là SELECT, UPDATE, DELETE, CREATE, ALTER,...

## Bước 5 - Thực hiện truy vấn

---

- Dùng hàm mysql:

```
$kqua = mysql_query($sql[, $biến_kết_nối])  
or die("Không thực hiện được SQL");
```

- Dùng hàm mysqli:

```
$kqua = mysqli_query([$biến_kết_nối,] $sql)  
or die("Không thực hiện được SQL");
```

- Dùng đối tượng mysqli (OOP):

```
$kqua = $biến_kết_nối->query($sql)  
or die("Không thực hiện được SQL");
```

# Bước 6: Xử lý kết quả trả về

---

- Một số hàm cần thiết:
  - `mysql_fetch_array($kqua [, kiểu_kqua])`:
    - Trả về 1 dòng dữ liệu trong \$kqua dưới dạng 1 mảng các giá trị.
    - Trả về `false` nếu không có dòng nào được lấy (\$kqua rỗng hoặc đã duyệt hết các dòng trong \$kqua).
    - Kiểu\_kqua nhận 3 giá trị:
      - `MYSQL_ASSOC`: các giá trị trong mảng trả về có thể truy cập qua \$key là tên của các trường trong "bảng".
      - `MYSQL_NUM`: các giá trị trong mảng trả về có thể truy cập qua \$key là số thứ tự của các trường trong bảng.
      - `MYSQL_BOTH`: có thể dùng cả 2 cách trên (mặc định).

## Bước 6: Xử lý kết quả trả về (tiếp)

---

- Một số hàm cần thiết (tiếp):
  - `mysql_fetch_row($kqua)`: có chức năng tương tự như khi sử dụng hàm `mysql_fetch_array($kqua, MYSQL_NUM)`
  - `mysql_fetch_assoc($kqua)`: có chức năng tương tự như khi sử dụng hàm `mysql_fetch_array($kqua, MYSQL_ASSOC)`

## Bước 6: Xử lý kết quả trả về (tiếp)

---

- Lựa chọn hàm sử dụng?
    - Hàm `mysql_fetch_array` có tốc độ chậm hơn không đáng kể so với 2 hàm `mysql_fetch_row`, `mysql_fetch_assoc` nhưng lại tiện lợi hơn.
- nên sử dụng hàm `mysql_fetch_array`.

## Bước 6: Xử lý kết quả trả về (tiếp)

---

- Một số hàm cần thiết (tiếp):
  - `mysql_num_rows ($kqua)`: trả về số lượng bản ghi trong \$kqua.
  - `mysql_affected_rows ()`: trả về số bản ghi bị tác động bởi lệnh `mysql_query` liền trước.
  - `mysql_error ()`: thông báo lỗi (nếu có)
  - `mysql_errno ()`: mã lỗi

# Bước 6: Xử lý kết quả trả về (tiếp)

---

- **Dùng hàm mysql:**

```
$row = mysql_fetch_array($kqua, kiểu_kquả);  
$row = mysql_fetch_row($kqua);  
$row = mysql_fetch_assoc($kqua);
```

- **Dùng hàm mysqli:**

```
$row = mysqli_fetch_array($kqua, kiểu_kquả);  
$row = mysqli_fetch_row($kqua);  
$row = mysqli_fetch_assoc($kqua);
```

- **Dùng đối tượng mysqli (OOP):**

```
$row = $kqua->fetch_array(kiểu_kquả);  
$row = $kqua->fetch_row();  
$row = $kqua->fetch_assoc();
```

## Bước 6: Xử lý kết quả trả về (tiếp)

---

- Ví dụ:

```
<?php  
$kqua = mysql_query("SELECT username, pass AS mkhau  
FROM user");  
while ($row = mysql_fetch_array($kqua, MYSQL_BOTH)) {  
    echo ("Name: {$row[0]} - Pass: {$row['mkhau']}");  
}  
?>
```

# Bước 7 – Dọn dẹp tài nguyên

---

- Dùng hàm mysql:

`mysql_free_result($kqua);`

- Dùng hàm mysqli:

`mysqli_free_result($kqua);`

- Dùng đối tượng mysqli (OOP):

`$kqua->close();`

# Bước 8 – Đóng kết nối

---

- Dùng hàm mysql:

**mysql\_close** (\$biến\_kết\_nối);

- Dùng hàm mysqli:

**mysqli\_close** (\$biến\_kết\_nối);

- Dùng đối tượng mysqli (OOP):

\$biến\_kết\_nối->**close()**;

# Ví dụ

---

- Giả sử ta có
  - CSDL test đặt trên server localhost.
  - User đăng nhập: root, mật khẩu: rỗng
  - Bảng user trong CSDL test có 2 trường
    - username & password
- Viết đoạn script PHP in ra dữ liệu trong bảng user

## Ví dụ (tiếp)

---

- Đầu tiên, ta kết nối đến CSDL ở server:

```
$conn = mysql_connect("localhost",
    "root", "") or die("Khong the ket noi!");
mysql_select_db("test", $conn);
```

- Viết câu truy vấn lấy dữ liệu:

```
$kqua = mysql_query
    ("SELECT * FROM user", $conn);
```

## Ví dụ (tiếp)

---

- Kiểm tra xem trong bảng dữ liệu đã tồn tại user nào chưa? Nếu có thì lặp lấy ra cho đến hết bảng dữ liệu, nếu không thì xuất ra thông báo.

```
if ($row = mysql_fetch_array($kqua)) {  
    do {  
        echo ("Tên: {$row['username']} -  
              Mật khẩu: {$row['password']} <br/>");  
    } while ($row = mysql_fetch_array($kqua));  
} else {  
    echo "không có bản ghi nào!";  
}
```

## Ví dụ (tiếp)

---

- Cuối cùng là giải phóng tài nguyên, đóng kết nối.

```
mysql_free_result($kqua);  
mysql_close($conn);
```

# Ví dụ (tiếp)

---

```
<?php
$conn = mysql_connect("localhost", "root", "")  
        or die("Khong the ket noi!");  
mysql_select_db("test", $conn);  
$kqua = mysql_query("SELECT * FROM user", $conn);  
if ($row = mysql_fetch_array($kqua)) {  
    do {  
        echo ("Tên: {$row['username']} -  
              Mật khẩu: {$row['password']} <br/>");  
    } while ($row = mysql_fetch_array($kqua));  
} else {  
    echo "không có bản ghi nào!";  
}  
mysql_free_result($kqua); mysql_close($conn);
?>
```

# Tiếng Việt trong PHP

---

- Phần không sử dụng CSDL MySQL, phải đảm bảo:
  - Lưu file mã lệnh PHP với mã UTF8
  - Khai báo trong phần head

```
<meta http-equiv="Content-Type"  
      content="text/html; charset=utf8"/>
```

# Tiếng Việt trong PHP (tiếp)

---

- Phần sử dụng CSDL MySQL, phải đảm bảo khai báo câu lệnh

```
mysql_query("SET NAMES 'utf8' ", $conn);
```

trước câu lệnh

```
mysql_query($sql) ;
```

với \$conn là biến kết nối, \$sql là một câu lệnh Select.

# Tiếng Việt trong PHP (tiếp)

---

- Ví dụ:

```
$conn = ... ;  
$sql = "SELECT ...";  
mysql_query("SET NAMES 'utf8' ", $conn);  
$kqua = mysql_query($sql);  
if (!$kqua) {  
    echo "Record not found!";  
} else {  
    while ($row = mysql_fetch_array($kqua)) {  
        echo $row['ho'] . " " . $row['ten'] . "<br>";  
    }  
}
```

# Bài tập

- 1. Thiết kế site hiển thị thông tin khách hàng (lưu trong CSDL) như sau:

**THÔNG TIN KHÁCH HÀNG**

Mã KH	Tên khách hàng	Giới tính	Địa chỉ	Số điện thoại
kh001	Khuất Thúy Phương		A21 Nguyễn Oanh quận Gò Vấp	9874125
kh002	Đỗ Lâm Thiên		357 Lê Hồng Phong Q.10	8351056
kh003	Phạm Thị Nhung		56 Đinh Tiên Hoàng quận 1	9745698
kh004	Nguyễn Khắc Thiện		12bis Đường 3-2 quận 10	8769128
kh005	Tô Trần Hồ Giang		75 Nguyễn Kiệm quận Gò Vấp	5792564
kh006	Nguyễn Kiến Thị		357 Lê Hồng Phong Q.10	9874125
kh008	Nguyễn Anh Tuấn		1/2bis Nơ Trang Long Q.BT TP.HCM	8753159

## Bài tập (tiếp)

- 2. Thiết kế site hiển thị thông tin các sản phẩm sữa (lưu trong CSDL) như sau:

THÔNG TIN CÁC SẢN PHẨM				
Similac Neo Sure 370 gr - 145.000 VNĐ 	Abbott Pedia Sure 400 gr - 146.000 VNĐ 	Abbott Grow School 400 gr - 87.000 VNĐ 	Abbott Grow 400 gr - 87.000 VNĐ 	Gain IQ 400 gr - 107.000 VNĐ 
Gain Advance 400 gr - 107.000 VNĐ 	Sữa đặc Trường Sinh 360 gr - 11.500 VNĐ 	Cô Gái Hà Lan 456 400 gr - 49.500 VNĐ 	Cô Gái Hà Lan 123 400 gr - 52.600 VNĐ 	Friso 400 gr - 52.000 VNĐ 

# SQL Injection

---

- Là một kỹ thuật cho phép hacker tấn công bằng cách thi hành những câu lệnh SQL bất hợp pháp.
- Thực hiện bằng cách lợi dụng các lỗ hổng trong việc kiểm tra dữ liệu nhập vào của các ứng dụng web, từ đó sử dụng một cách khéo léo các cú pháp của ngôn ngữ truy vấn SQL để thực hiện tấn công.
- Tùy thuộc vào môi trường và cách cấu hình hệ thống mà tác hại là nặng hay nhẹ.

# Ví dụ

---

- Câu lệnh kiểm tra username & password

```
$sql = mysql_query("SELECT * FROM user  
                    WHERE username = '$user' AND  
                          password = '$pass'");
```

- Nếu biến \$user được nhập là:

' OR 1 OR user='

thì khi đó câu lệnh SQL sẽ là:

```
SELECT * FROM user WHERE  
        username = '' OR 1 OR user='' AND  
        password = '$pass'
```

# Cách phòng tránh

---

- Kiểm soát chặt chẽ dữ liệu nhập vào
  - Giới hạn chiều dài của chuỗi nhập vào
  - Dùng hàm addslashes() để thêm dấu "\" vào trước các ký tự đặc biệt trong chuỗi nhập vào.
  - Xây dựng hàm tiện ích
    - Thay thế dấu 1 nháy đơn bằng 2 dấu nháy đơn
    - Loại bỏ một số ký tự và từ khóa "nguy hiểm"
    - ...

# Cách phòng tránh (tiếp)

---

- Ví dụ: hàm thay ' thành “

```
function replace($input) {  
    $output = str_replace("'", "“", $input);  
    return $output;  
}
```

# Cách phòng tránh (tiếp)

---

- Ví dụ: hàm loại bỏ các từ khóa của SQL

```
function killChar($input) {  
    $char = array("select", "insert", "xp_",  
                 "delete", "drop", "--", ";");  
    $output = str_replace($char, "", $input);  
    return $output;  
}
```

# Chương VI: Quản lý Session và Cookies

---

- Session là một đối tượng cho phép truyền giá trị từ trang PHP này sang trang PHP khác.
  - Session bắt đầu từ khi client truy cập vào website cho đến khi đóng trình duyệt hay kết thúc (abandon) phiên (hủy hoặc hết thời gian).
  - Session chỉ được hỗ trợ từ phiên bản PHP 4
  - Session có rất nhiều ứng dụng, ví dụ như lưu trữ thông tin về giỏ hàng trong E-commerce

# Session (tiếp)

---

- PHP cung cấp mảng siêu toàn cục **`$_SESSION[]`** để lưu trữ các session
  - Có thể truy cập mảng `$_SESSION[]` từ mọi trang PHP trong phiên.
  - Nơi lưu trữ PHP session được quy định tại thuộc tính **`session.save_path`** trong tập tin cấu hình **`php.ini`**

# Thao tác với Session

---

- Thiết lập session
  - `session_start(); $_SESSION['tên']=giá_trị;`
- Truy cập session
  - `session_start(); $biến = $_SESSION['tên'];`
- Xóa session đã thiết lập
  - `unset($_SESSION['tên']);`
- Hủy bỏ toàn bộ session:
  - `session_destroy();`

# Thao tác với Session (tiếp)

---

- Trang Save\_Session.php

```
<?php  
    session_start();  
    $_SESSION['user']='Webmaster';  
    header('Location:Read_Session.php');  
?>
```

- Hàm session\_start() phải đặt trước thẻ <html>.
- Hàm header() dùng để chuyển hướng trình duyệt đến URL sau từ khóa Location.

# Thao tác với Session (tiếp)

---

- Trang : Read\_Session.php

```
<?php session_start(); ?>
<html><body>
<?php
    if(isset($_SESSION['user']))
        $mess=$_SESSION['user'].' đã đăng nhập';
    else
        $mess='Bạn chưa đăng nhập hệ thống!';
    echo $mess;
?>
</body></html>
```

# Cookie

---

- Cookie là những mẩu thông tin nhỏ, có cấu trúc, dạng text lưu ở máy client.
  - Thường dùng để lưu thông tin về client
  - Khả năng tạo cookie phụ thuộc vào trình duyệt và sự cho phép của người sử dụng.
  - Cookie thường được lưu ở thư mục:  
**C:\Documents and Settings\Administrator\Cookies**

# Thiết lập cookie

---

- Cú pháp:  
**setcookie**(name, value, expire [, path, domain]);
  - name: tên cookie được tạo ra
  - value: giá trị được đặt cho cookie
  - expire: thời gian hết hạn của cookie
- Chú ý:
  - Lệnh setcookie phải được gọi trước khi gửi bất cứ nội dung gì về client (trước các thẻ HTML, trước các lệnh echo, print)
  - Để thiết lập giá trị expire, ta thường sử dụng hàm time() + khoảng thời gian (tính bằng giây)

# Thiết lập cookie

---

- Ví dụ

```
<?php  
    $expire = time() + 60 * 60 * 24 * 30;      //hạn 30 ngày  
    setcookie('ntb', 'Nguoi Dung', $expire);  
    echo 'Cookie has been created!';  
?>
```

## Đọc cookie

---

- Các cookie sau khi thiết lập sẽ được lưu trong mảng siêu toàn cục **`$_COOKIE`**.
  - Dùng hàm `isset()` để kiểm tra một cookie có tồn tại hay không.
  - **Chú ý:** Ta không thể đọc cookie vừa được thiết lập ngay trong trang cùng 1 trang vừa thiết lập gọi `setcookie`.

# Đọc cookie (tiếp)

---

- Ví dụ:

```
<?php
    if (isset($_COOKIE['ntb']))
        echo 'Welcome '. $_COOKIE['ntb'];
    else
        echo 'Welcome guest!<br />';
?>
```

# Bài tập

---

- Tổ chức các bài tập trong phần trước thành 1 site với yêu cầu:
  - Có menu ở mọi trang cho phép truy cập đến các trang chức năng riêng biệt.
  - Mọi trang chức năng yêu cầu phải đăng nhập mới được sử dụng, nếu người dùng chưa đăng nhập thì chuyển người dùng về trang đăng nhập. (tự động hoặc link)
  - Tạo trang logout.php cho phép người dùng thoát khỏi phiên đăng nhập.

# Chương VII: Email và Upload files

---

- Gởi mail trong PHP
- Cú pháp: **mail(to,subject,message,headers,parameters)**
- Ý nghĩa các tham số:
  - - to, subject, message : như ý nghĩa các text box khi soạn mail
  - - headers :tùy chọn, có thể sử dụng Bcc, Cc
  - - parameter: tùy chọn, các thông số về trình soạn, gởi mail
  - Trong phần message: sử dụng ký hiệu \n để xuống dòng.

# Chương VII: Email và Upload files

---

- Ví dụ: Tệp **Send\_mail.php** //viết thư gửi cho ranquangvinh@qnu.edu.vn

```
<?php  
    $to = "tranquangvinh@qnu.edu.vn";  
    $subject = "Test mail";  
    $message = "Hello! This is a simple email message.";  
    $from = "ndt473@yahoo.com";  
    $headers = "From: $from";  
    @mail($to,$subject,$message,$headers); // không cho warning!  
    echo "Mail Sent.";  
?>
```

- Lưu ý:
  - người gửi phải có một địa chỉ mail. Theo dõi các kỹ thuật chống spam của mỗi trình gửi, nhận mail của người nhận!
  - Phối hợp với form để soạn thảo một trình gửi mail
  - Có thể lập trình để gửi mail đến danh sách các địa chỉ đã lưu trong CSDL

# Chương VII: Email và Upload files

---

- Mở files:
- Để mở 1 file ta sử dụng cú pháp sau:

`fopen ('path', mode)`

- path là đường dẫn tới file cần mở.
- mode là các chế độ mở-thao tác với file.
- Chú ý: ký hiệu đường dẫn \\ hoặc /

# Mở file (tiếp)

- Danh sách các mode:

Mode	Điễn giải
r	Read only
r+	Read_Write
w	Write Only
w+	Write_Read. Nếu file này tồn tại, nội dung cũ sẽ bị xoá đi, còn nếu không tồn tại nó sẽ được tạo mới
a	Mở dưới dạng append dữ liệu, chỉ có write, nếu file tồn tại, sẽ ghi tiếp vào phần dưới của nội dung có sẵn, nếu file không tồn tại sẽ được tạo mới.
a+	Mở dưới dạng append dữ liệu (write và read), nếu file tồn tại, dữ liệu sẽ ghi tiếp vào phần bên dưới của nội dung cũ, nếu file không tồn tại sẽ được tạo mới
b	Mở dưới chế độ file binary

# Mở file (tiếp)

---

- Ví dụ:

```
<?php  
    $fp = fopen ('test.txt', r)  
        or exit('khong tim thay file');  
?>
```

- Lưu ý:

- Việc mở file ở trên không có nghĩa là file đã được đọc. Để đọc file ta cần thực hiện các thao tác đọc/duyệt file riêng.

# Đóng file

---

- Để đóng file, ta dùng câu lệnh:

**fclose** (\$biến\_file)

- Luôn nhớ thực hiện câu lệnh đóng file sau khi thao tác xong.
- Ví dụ:

```
<?php  
    $fp = fopen('test.txt', r)  
        or exit('khong tim thay file');  
    fclose($fp);  
?>
```

# Đọc file

---

- PHP hỗ trợ nhiều hình thức đọc file khác nhau. 2 hình thức đọc phổ biến nhất là:
  - Đọc theo từng dòng.
  - Đọc file theo từng ký tự.

# Đọc file (tiếp)

---

- Cú pháp đọc file theo từng dòng:

**fgets** (\$biến\_file\_vừa\_mở)

- Ví dụ:

```
<?php  
    $fp = fopen('test.txt',r)  
        or exit('khong tim thay file');  
    echo fgets($fp);  
    fclose($fp);  
?>
```

# Đọc file (tiếp)

---

- Cú pháp đọc file theo từng ký tự:

**fgetc** (\$biến\_file\_vừa\_mở)

- Ví dụ:

```
<?php  
    $fp = fopen('test.txt',r)  
        or exit('khong tim thay file');  
    echo fgetc($fp);  
    fclose($fp);  
?>
```

# Đọc file (tiếp)

---

- Dù đọc theo hình thức nào thì trong quá trình đọc, ta cần luôn kiểm tra xem đã đọc hết file chưa bằng cách dùng hàm:

**feof** (\$biến\_file\_vừa\_mở)

- Ví dụ:

```
<?php  
    $fp = fopen('test.txt', r)  
        or exit('khong tim thay file');  
    while(!feof($fp)) {  
        echo fgets($fp);  
    }  
    fclose($fp);  
?>
```

# Ghi file

---

- Để ghi dữ liệu vào file, ta dùng hàm sau:

**fwrite** (\$biến\_file, \$nội\_dung)

- Ví dụ:

```
<?php  
    $fp = fopen ('test.txt', r)  
        or exit('khong tim thay file');  
    $ndung = 'Dai hoc Quy Nhon';  
    fwrite($fp, $ndung);  
    fclose($fp);  
?>
```

## Ví dụ upload file

---

- Tạo file form.php như sau:

```
<form action="upload.php" method="post"
      enctype="multipart/form-data">
    File <input type="file" name="myfile" size="30"/>
    <input type="submit" name="submitBtn"/>
</form>
```

# Ví dụ upload file (tiếp)

---

- Tạo file Upload.php như sau:

```
<?php
    $destination_path = getcwd().DIRECTORY_SEPARATOR;
    $target_path = $destination_path.basename(
                    $_FILES['myfile']['name']);
    if (( $_FILES['myfile']['type'] == 'image/gif') ||
        ( $_FILES['myfile']['type'] == 'image/jpeg') &&
        ( $_FILES['myfile']['size'] < 5120000))
    {
        if(move_uploaded_file(
            $_FILES['myfile']['tmp_name'],
            $target_path))
        { //thông báo upload thành công }
    }
?>
```

# Phân trang với PhP

---

- Tổng số tin trên 1 trang
  - \$baitren\_mottrang = 10;
- chưa chọn trang để xem mặc định đang xem trang số 0 .

```
if( !$_GET['page'] )
{
    $page = 0 ;
}
```

# Phân trang với PHP

---

- Tính số trang cho dữ liệu

```
$sodu_lieu = mysql_num_rows(mysql_query("SELECT * FROM `data` ")) or die(mysql_error());
```

```
$sotrang = $sodu_lieu/$baitren_mottrang;
```

- Đưa dữ liệu các lên trang

```
$result =mysql_query("SELECT * FROM `data` ORDER BY `id` DESC  
LIMIT {$page}*{$baitren_mottrang},{$baitren_mottrang} ") or die(mysql_error());  
while ( $info = mysql_fetch_array($result ))  
{  
    echo <<<EOT  
    $info['tenbai_viet']  
    <br />  
    $info['noidung']  
    EOT;  
}
```

# Phân trang với PHP

---

- Tạo nút bấm chuyển qua các trang

```
for ( $page = 0; $page <= $sotrang; $page ++ )  
{  
    echo "<a href='index.php?page={$page}'>{$page}</a>";  
}
```

# Sao lưu dữ liệu phpMyAdmin

---

## Export

1. Truy cập vào phần quản lý Host và chọn công cụ quản lý cơ sở dữ liệu **phpMyAdmin**
2. Trong **phpMyAdmin** chọn **Database** muốn sao lưu.
3. Tiếp tục chọn **Export** để trích xuất các bảng dữ liệu trong Database
4. Chọn **Method Export** và nhấn **Go** để tiến hành sao lưu

# Sao lưu dữ liệu phpMyAdmin

---

## Import

1. Truy cập vào phần quản lý Host và chọn công cụ quản lý cơ sở dữ liệu **phpMyAdmin**
2. Trong **phpMyAdmin** chọn **Import** để đưa cơ sở dữ liệu vào Database
3. Chọn **File to Import**
4. Chọn Format và nhấn Go để tiến hành

**File to Import:**

File may be compressed (gzip, bzip2, zip) or uncompressed.  
A compressed file's name must end in .[format].[compression]. Example: .sql.zip

Browse your computer:  (Max: 128MiB)

Character set of the file: