# PHẦN 1: GIỚI THIỆU CƠ BẢN VỀ LẬP TRÌNH WEB

## BÀI 1. GIỚI THIỆU CƠ BẢN VỀ LẬP TRÌNH WEB

### I. Các khái niệm cơ bản

#### 1.1. Internet

Là một hệ thống gồm nhiều máy tính nối mạng trên thế giới nhằm trao đổi thông tin với nhau thông qua giao thức kết nối TCP/IP. Hệ thống này bao gồm hàng ngàn mạng máy tính nhỏ hơn của các doanh nghiệp, của các viện nghiên cứu và các trường [đại học](http://vi.wikipedia.org/wiki/%C4%90%E1%BA%A1i_h%E1%BB%8Dc), [người dùng cá nhân](http://vi.wikipedia.org/w/index.php?title=Ng%C6%B0%E1%BB%9Di_d%C3%B9ng_c%C3%A1_nh%C3%A2n&action=edit&redlink=1) và các chính phủ trên toàn cầu.

Internet mang lại rất nhiều tiện ích hữu dụng cho người sử dụng, nổi bật nhất là thư điện tử (email), trò truyện trực tuyến (chat), mạng xã hội (facebook, switter ,..) và truy tìm dữ liệu (google).

Dịch vụ thường được dùng phổ biến trên Internet là Word Wide Web (WWW, thường gọi tắt là web). Tuy nhiên, giữa Internet và WWW có sự khác biệt cơ bản. Internet là một tập hợp các mạng máy tính kết nối với nhau bằng dây đồng, [cáp quang](http://vi.wikipedia.org/wiki/C%C3%A1p_quang), v.v.; còn WWW, hay Web, là một tập hợp các tài liệu liên kết với nhau bằng các [siêu liên kết](http://vi.wikipedia.org/wiki/Si%C3%AAu_li%C3%AAn_k%E1%BA%BFt) (hyperlink) và các địa chỉ [URL](http://vi.wikipedia.org/wiki/URL) và nó có thể được truy nhập bằng cách sử dụng Internet. Web là môi trường giao tiếp chính của người sử dụng trên internet.

#### 1.2. Web Server

Là máy tính lưu trữ website, dữ liệu có khả năng xử lý yêu cầu gởi tới nhanh chóng và trả kết quả về cho Web Client.

Tất cả các Web Server đều có một địa chỉ IP (IP Address) hoặc cũng có thể có một Domain Name. Server phải hoạt động liên tục để phục vụ cho việc cung cấp thông tin trực tuyến.

#### 1.3. Web Client

Là máy tính dùng để truy cập các trang web và có khả năng yêu cầu và nhận kết quả trả về từ Web Server. Tại client, người dùng cá nhân có thể sử dụng các trình duyệt web (Web browser) để gửi yêu cầu và hiển thị thông tin cần xem.

#### 1.4. Web Browser

Là phần mềm dùng để xem các tài liệu hoặc tìm kiếm các tài nguyên trên Word Wide Web.

Một số trình duyệt phổ biến hiện nay:

* Internet Explorer (IE) được tích hợp sẵn trong Microsoft Windows của Microsoft.
* Mozilla Firefox của tập đoàn Mozilla.
* Netscape Navigator của Netscape.
* Opera của Opera Software.
* Chrome của Google.

#### 1.5. HTTP (HyperText Transfer Protocol)

Là giao thức chuyển siêu văn bản trên web.

Giao thức này là tập hợp các quy định dùng để trao đổi các tài liệu như văn bản, hình ảnh, âm thanh, video, các tập tin đa truyền thông, giữa Web Server và các trình duyệt web.

Ví dụ, khi bạn gõ một địa chỉ Web URL vào trình duyệt Web, một lệnh HTTP sẽ được gửi tới Web server để ra lệnh và hướng dẫn nó tìm đúng trang Web được yêu cầu và kéo về mở trên trình duyệt Web.

#### 1.6. URL

Là từ viết tắt của Uniform Resouce Locator. Là đường dẫn dẫn tới một trang web cụ thể trên Internet.

Cú pháp đầy đủ: <Schema>://<host>[:port] [<path> [?<query String>]

Ví dụ:

<http://www.ctu.edu.vn>

<http://www.vnexpress.net/chuyenla?tab=kylucthegioinguoiviet>

<http://www.htmllib.com:80/example_files/file.html?lib=3>

#### 1.7. HTML (Hyper Text Markup Language)

Là ngôn ngữ dùng để xây dựng các trang web.

HTML bao gồm các tag giúp Web browser biết cách định dạng thông tin hiển thị

Ví dụ:

<html>

<head>

<title> Lập trình web </title>

</head>

<body>

<h1 align = “center”> Chào mừng các bạn học lớp lập trình web</h1>

</body>

</html>

#### 1.8. Web tĩnh và lập trình Client – side

Là trang web được trình bày dưới dạng văn bản. Nội dung trong trang được tạo ra lúc thiết kế và trong sử dụng và không thay đổi khi có người dung truy cập.

Được viết bằng ngôn ngữ HTML và không có kết nối với cơ sở dữ liệu (CSDL).

#### 1.9. Web động

Là trang web có nội dung được lấy từ cơ sở dữ liệu.

Nội dung trang sẽ thay đổi nếu cho phép người dùng tương tác, chỉnh sửa. Ví dụ: các trang web mua hàng trực tuyến, đặt vé…

Dùng ngôn ngữ Server-side script xử lý.

### II. Cấu trúc của một trang web

Cấu trúc cơ bản của trang HTML/XHTML có dạng như sau, thường gồm 3 phần:

* Phần DOCTYPE: khai báo kiểu văn bản. Nó cho trình duyệt biết bạn đang sử dụng phiên bản HTML nào.
* Phần tiêu đề: bắt đầu bằng thẻ <head> và kết thúc bởi thẻ </head>. Phần này chứa tiêu đề mà được hiển thị trên thanh điều hướng của trang web. Tiêu đề là phần nội dung nằm giữa cặp thẻ <title> và </title>.
* Phần thân: nằm sau phần tiêu đề. Nó bao gồm văn bản, hình ảnh và các liên kết mà bạn muốn hiển thị trên trang web của mình. Phần thân bắt đầu bằng thẻ <body> và kết thúc với thẻ </body>.

Ví dụ: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

<head>

<title>Tiêu đề trang web</title>

</head>

<body>

//Phần thân viết ở đây

</body>

</html

Mỗi trang web đều có cách thể hiện cấu trúc khác nhau, có trang 1 cột, có trang 2 và cũng có trang chứa nhiều cột.

Ví dụ cấu trúc một trang web:

Phần đầu: header, có thể chứa logo, câu slogan, các liên kết, các banner liên kết, các button, đoạn flash, hoặc các form ngắn như form tìm kiếm …

Phần thân của trang: page body, phần này chứa phần nội dung chính (content) và phần nội dung phụ (sidebar).

Phần nội dung chính: content, phần này chứa nội dung chính cần thể hiện cho người dùng xem.

Phần nội dung phụ: sidebar, phần này có thể chứa liên kết phụ của từng trang (local navigation), hoặc các banner chứa liên kết liên quan, hoặc có thể dùng để chứa các liên kết quảng cáo…

Phần cuối trang web: footer, phần này thường chứa phần liên hệ như: tên công ty, địa chỉ, số điện thoại, mail liên hệ,... và đặc biệt là copyright, hoặc có thể chứa các liên kết toàn trang, các banner liên kết,...



Hình 1. . Cấu trúc của một trang web (ctu.edu.vn)

#### 2.1. Các thẻ html cơ bản

Thẻ (tag) HTML là gì: Khi trình duyệt Web đọc một file HTML, nó sẽ tìm trong file đó các tag hay những đoạn mã đặc biệt để biết cách hiển thị file HTML đó.

Ví dụ: Khi trong file HTML có đoạn *<h3>* cấu trúc của file HTML *</h3>* thì sẽ hiển thị đoạn "cấu trúc của file HTML" lên trên màn hình với mức độ tiêu đề to ở mức thứ 3.

Các tag HTML báo cho trình duyệt biết cách khi nào thì in đậm một dòng văn bản, in nghiêng, biến dòng văn bản thành một siêu liên kết tới các trang Web khác, hiển thị ảnh...

Trong ví dụ trên ta thấy sau tag *<h3>* và đoạn văn bản là tag *</h3>*. Trong tag *</h3>* ta thấy có dấu /, dấu / nằm trong 1 tag báo hiệu cho trình duyệt biết là hiệu ứng của tag đó đã kết thúc.

##### 2.1.1. Thẻ phiên bản <!DOCTYPE>

Thẻ này dùng ghi thông tin về version HTML áp dụng trong tài liệu Web. Thường đây là dòng đầu tiên trong file HTML.

Ví dụ:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN">

Hoặc

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML Level 1//EN">

##### 2.1.2. Thẻ *<HTML>*

Thẻ chính yếu của file HTML. Đánh dấu điểm bắt đầu và chấm dứt nội dung của file.

Ví dụ:

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">

<HTML>

//Nội dung file gồm có 2 phần là <HEAD> và <BODY>

<HTML>

##### 2.1.3. Thẻ Head <HEAD>

Nội dung nằm giửa thẻ nầy là phần tiêu đề của trang Web. Chúng không hiển thị khi trình Browse đọc file.

<HEAD>

<TITLE>Introduction to HTML</TITLE>

</HEAD>

Trong nội dung HEAD có thể chứa các thành phần sau:

* <BASE>: Khai báo điạ chỉ cơ bản cho file HTML
* <ISINDEX>: Chỉ định từ khoá cho việc tìm kiếm
* <TITLE>: Đặt tên tiêu đề cho tài liệu.
* <META>: Các thông tin khác về tài liệu.

##### 2.1.4. Thẻ Body của trang web *<BODY>*

Đây là thẻ chứa nội dung chính của file HTML. Dạng thức tổng quát như sau:

<BODY>

//Tất cả nội dung và định dạng của các thành phần chính trong trang web.   
</BODY>

Các thuộc tính:

* **BACKGROUND:**Dùng chỉ định file hình ảnh làm nền:

<BODY BACKGROUND="imagename.gif">

//Rest of the document goes here

</BODY>

Hình nền phải có dạng thức GIF hay JPG. Nếu dùng Internet Explorer sẽ hổ trợ thêm dạng **BMP.**

* **BGCOLOR:**Xác lập màu cho nền:

<BODY BGCOLOR="#rrggbb">

//Màu nền của văn bản

</BODY>

Với "**#rrggbb**" là giá trị hexadecimal (thập lục) red-green-blue.

**Internet Explorer có thể xác lập 16 màu theo tên như sau:** Black, Silver, Gray, White, Maroon, Red, Purple, Fuchsia, Green, Lime, Olive, Yellow, Navy, Blue, Teal, Aqua.

* **TEXT:** Thuộc tính nầy chỉ định màu cho văn bàn thường trong file.

<BODY TEXT="#rrggbb">

//Màu của văn bản ở đây

</BODY>

Các xác lập màu giống như BGCOLOR.

* **LINK, VLINK, và ALINK:**Dùng chỉ định màu cho các dòng văn bản là đối tượng chủ cho mối nối liên kết. Trong đó **LINK: Liên** kết chưa xem, VLINK: Liên kết đã xem, ALINK: Liên kết đang xem. Mặc nhiên là: LINK=blue (#0000FF), VLINK=purple (#800080), và ALINK=red (#FF0000). Cách xác lập màu giống BGCOLOR và TEXT.
* **LEFTMARGIN:**Canh lề trái.

Ví dụ:

<BODY LEFTMARGIN="40”>

//Văn bản này được định dạng lề trái cách mép 40 pixels

</BODY>

* **TOPMARGIN:**Canh lề trên. Ví dụ:

<BODY TOPMARGIN="40”>

//Văn bản này được định dạng cách 40 pixels tính từ trên xuống

</BODY>

##### 2.1.5. Thẻ chú thích <!-- (chú thích) -->:

Để thêm những dòng chú thích trong file HTML, người ta dùng thẻ nầy. Nội dung văn bản nằm giữa **<!--** và **-->** sẽ được chương trình Browse bỏ qua. Cho phép có khoảnh trắng giửa -- và >, nhưng không được có khoảng trắng giửa <! và --.

Ví dụ:

<HEAD>

<Title> Tiêu đề</Title>

<!-- Đây là đoạn chú thích -->

</HEAD>

##### 2.1.6. Thẻ liên kết *<A>*

Đây là thẻ xác lập cho việc liên kết. Các thuộc tính của thẻ này như sau:

* **HREF:**Thuộc tính HREF chỉ định địa chỉ liên kết, dòng văn bản sau dấu = là địa chỉ đối tượng liên kết (được bao trong cặp " "). Dòng văn bản giữa dấu <A> và </A> (hoặc <a> và </a>) là đối tượng chính của mối nối liên kết. Đối tượng ở đây có thể là một hình ảnh hay một đoạn văn bản hay một vùng được chỉ định trước. Khi bạn kích mouse vào đối tượng chính trong trình Browse, bạn sẽ được chuyển đến đối tượng liên kết.

Ví dụ: <A HREF="http://ctu.edu.vn">Web site trường ĐHCT</A>

Trong ví dụ, nếu bạn kích vào dòng "Web site trường ĐHCT". Bạn sẽ chuyển đến địa chỉ <http://ctu.edu.vn>. Bạn có thể thêm "**#identifier**" để chỉ định chuyển đến 1 vị trí được quy định sẳn trong đối tượng liên kết.

Ví dụ**:** <A HREF="document.html#document">tailieu</A>

Trong ví dụ, chọn "tailieu" sẽ được chuyển đến tài liệu document.html, ngay tại vị trí tên document trong tài liệu này. Ví dụ vài thẻ liên kết sau:

<A HREF="http://..."> Liên kết với 1 WebSite.

<A HREF="ftp://..."> Với 1 Ftp Site.

<A HREF="gopher://..."> Với 1 Gopher server.

<A HREF="news:..."> Liên kết với 1 nhóm tin tức.

<A HREF="<."> Với 1 địa chỉ gởi Mail. Liên kết nầy sẽ kích hoạt chương trình Mail và tự động điền địa chỉ vào mục To dùm bạn. Bạn có thể khai báo luôn cả chủ đề thư (?subject).

##### 2.1.7. Thẻ địa chỉ *<ADDRESS>*

Dùng khi cần khai báo địa chỉ

Ví dụ:

<ADDRESS>

Địa chỉ khoa CNTT&TT: Khu2, đường 3/2, Ninh Kiều, Cần Thơ

</ADDRESS>

##### 2.1.8. Thẻ in đậm <B>

Thẻ in đậm (**bold**) dòng văn bản nó cần làm nổi bật.

Ví dụ:

<HTML>

Ghi chú đây là <B>phần in đậm </B> của câu.

</HTML>

Khi hiển thị sẽ có dạng sau: Ghi chú đây là **phần in đậm** của câu.

##### 2.1.9. Thẻ ngắt xuống dòng <BR/>

Thẻ này dùng để ngắt một đoạn văn và xuống hàng mới.

Ví dụ:

<HTML>

Văn bản này cần xuống dòng <br/> dòng mới bắt đầu

</HTML>

Sẽ hiển thị: Văn bản này cần xuống dòng

dòng mới bắt đầu

##### 2.1.10. Thẻ canh giữa <CENTER>

Tất cả text nằm trong thẻ nầy được canh giửa so với lề trái và phải.

<HTML>

<CENTER>dòng chữ này sẽ được canh giữa</CENTER>

</HTML>

##### 2.1.11. Thẻ phân chia khu vực, canh trái, phải, giữa theo lề <DIV>

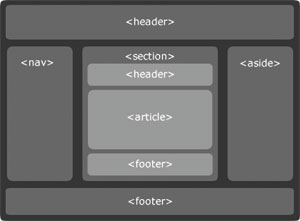
So hàng Text theo lề. Ví dụ:

<DIV ALIGN="left">dòng này sẽ được canh trái</DIV>

<DIV ALIGN="center">dòng này sẽ được canh giữa</DIV>

<DIV ALIGN="right">dòng này sẽ được canh phải</DIV>

Phân chia khu vực, chúng ta có thể dùng thẻ <div> để chia trang web ra nhiều phần khác nhau, mỗi phần có cùng định dạng cùng chức năng.

Ví dụ: Khu vực tiêu đề phía trên trang web (pannel), khu vực bên trái (tiêu đề, chỉ mục,…), khu vực bên phải (tin tức, thông báo,…), khu vực ở giữa (nội dung), khu vực phía dưới (địa chỉ liên hệ,…) như hình dưới đây:

Hình 1. . Dùng thẻ <div> để phân chia trang web

##### 2.1.12.Thẻ font <FONT>

Thành phần chính là <**FONT SIZE=value**>. Giá trị values từ **1-7**. Mặc nhiên FONT size là **3**. Giá trị thay đổi tương đối là '+' hay '-' so với giá trị chuẩn.

Ví dụ**:** <FONT SIZE=4> kích thước font chữ là 4</FONT>

<FONT SIZE=+2>Thay đổi font lên kích thước BASEFONT SIZE ... +2</FONT>

Thuộc tính: **COLOR = #rrggbb hay COLOR = color**: Xác lập màu giống như BGCOLOR. Ví dụ:<FONT COLOR="#FF0000">Dòng chữ này sẽ là màu đỏ.</FONT>

Hay <FONT COLOR="Red">Dòng này cũng là màu đỏ.</FONT>

**FACE=name [,name] [,name]:**Chỉ định font chữ khi hiển thị text.

Ví dụ:

<FONT FACE=".VNTIME, .VNARIAL">Dòng này font có thể là VNTIME hay VNARIAL, tùy thuộc vào font nào được cài rồi.</FONT>

##### 2.1.13. Thẻ kích thước tiêu đề <H1> đến <H6>

HTML có 6 mức chuẩn cho Tiêu đề (**heading**). Xác lập bằng các thẻ từ **<H1>** cho đến **<H6>**. To nhất là <H1>, bé nhất là <H6>. Ví dụ:

<H1>Tiêu đề to nhất</H1>

<H2>Tiêu đề to thứ 2, nhưng nhỏ hơn h1 </H2>

Bạn có thể sử dụng thuộc tính **ALIGN** để canh lề cho tiêu đề.   
Ví dụ:<H1 ALIGN=center>Tiêu đề h1 được canh trái </H1>

##### 2.1.14. Thẻ gạch ngang <HR>

Đường gạch ngang để phân cách các đoạn trong trang Web. Thuộc tính:

**<HR SIZE=number>:**Chỉ định kích thước.

**<HR WIDTH=number/percent>:**Chỉ định độ rộng.

**<HR ALIGN=left/right/center>:**So lề.

**<HR NOSHADE>:**Không bóng.

**<HR COLOR>=name/#rrggbb:**Chỉ định màu giống BGCOLOR.

Ví dụ:

<HR>

<HR SIZE="5">

<HR WIDTH="50%">

<HR WIDTH="50%" ALIGN="right">

<HR WIDTH="50%" ALIGN="center" NOSHADE>

##### 2.1.15. Thẻ in nghiêng <i>

Các văn bản đặt trong thẻ này sẽ được in nghiêng, thẻ bắt đầu là <i> kết thúc là </i>. Ví dụ: <i> dòng này sẽ được in nghiêng </i>

Sẽ hiển thị: *Dòng này sẽ được in nghiêng*

##### 2.1.16. Thẻ gạch chân <u>

Các văn bản trong thẻ này sẽ được gạch chân, bắt đầu là <u> kết thúc là </u>. Ví dụ: <u> dòng này sẽ được gạch chân</u>

Sẽ hiển thị: Dòng này sẽ được gạch chân

##### 2.1.17. Thẻ chèn hình ảnh <IMG>

Chèn một file hình vào tài liệu HTML. **Thuộc tính:**

**ALIGN="left/right/top/texttop/middle/absmiddle/baseline/bottom/absbottom":**So hàng hình ảnh với Text.

**ALT="Alternative Text":**Cho hiển thị 1 dòng text thay thế cho file hình trong trường hợp trình Browse đang ở trong chế độ không hiển thị hình ảnh. Dòng Text nầy cũng hiển thị theo dạng ToolTip khi dời Mouse đến hình.

Ví dụ:

<IMG SRC="triangle.gif" ALT="Cảnh báo”>

//SRC="URL of image": Chỉ định điạ chỉ file hình chèn vào trang Web.

**WIDTH=value/ HEIGHT=value:**Chỉ định khoảng cách dành sẳn cho hình trong khi trình Browse nạp toàn bộ hình.

**BORDER=value:**Chỉ định cho hiển thị đường viền bao quanh hình ảnh. Bạn có thể chọn "0" để hiển thị đường viền màu xanh khi có liên kết.

**VSPACE=value HSPACE=value:**Quy định khoảng trống giửa hình và Text. **VSPACE** cho trên và dưới hình, **HSPACE**cho trái và phải hình. Value tính theo pixel.

**LOWSRC:**Thuộc tính nầy cho phép hiển thị 2 hình lần lượt trong cùng 1 vị trí. Thường dùng để nạp một hình nhỏ trong khi chờ đợi nạp hình chính có dung lượng file lớn hơn. **Ví dụ:** <IMG SRC="hiquality.gif" LOWSRC="lowquality.gif">. Đầu tiên trình Browse sẽ hiển thị file hình "**lowquality.gif**". Sau khi nạp hoàn tất cả trang, trình Browse sẽ nạp file hình chính thức vào thay thế.

##### 2.1.18. Thẻ nhập textbox <INPUT>

Tạo một field để nhận tác động của người sử dụng. Thuộc tính:

**ALIGN:**So hàng cho field.

**CHECKED:**Kiểm tra người dùng đã đánh dấu cho checkbox hay radio button chưa.   
**MAXLENGTH:**Chỉ định độ dài ký tự có thể nhập vào text field, độ dài nầy có thể lớn hơn kích thước Text field. Mặc định là không giới hạn.

**NAME:**Tên của Field.

**SIZE:**Khai báo kích thước hay số lương ký tự cho field. Ví dụ: Khai báo 1 field với độ rộng là 24 ký tự: <INPUT TYPE=text SIZE="24"/>

**TYPE:**Chỉ định kiểu của Field.

**BUTTON:**Chèn một nút bấm vào tài liệu. Giá trị VALUE dùng chỉ định Text sẽ hiện trong nút này.

Ví dụ: <INPUT TYPE="BUTTON" VALUE="Hello" NAME="btnHello">

**CHECKBOX:**Chèn 1 chechbox vào tài liệu**.**

Ví dụ: <INPUT TYPE="CHECKBOX" NAME="chkbox">

**HIDDEN:**Với thuộc tính nầy, field sẽ không hiển thị ra nhưng nội dung của field vẫn có giá trị. Dùng trao đổi thông tin ngầm giửa client/server.

**IMAGE:**Chèn field chứa hình ảnh để người dùng bấm Mouse khi chọn.

Ví dụ: <INPUT TYPE="IMAGE" SRC="../images/iexplore.gif"

ALIGN="middle">  
**PASSWORD:**Giống như Text, nhưng ký tự nhập vào sẽ không hiển thị ra.

Ví dụ: <INPUT TYPE="PASSWORD">

**RADIO:**Chèn một field có dạng Nút Radio. Ví dụ: <INPUT TYPE="RADIO">

**RESET:**Chèn một nút bấm dùng phục hồi lại tình trạng cũ cho các field. Đặt tên của nút nầy qua thuộc tính Values. Ví dụ: <INPUT TYPE="RESET">

**SUBMIT:**Một dạng nút bấm giống RESET. Có tác dụng giống như xác nhận đồng ý. Ví dụ: Chèn một nút có tên "**SUBMIT**" và sẽ hiển thị thông báo "**alert. This is just an example, nothing will happen**" khi bạn bấm Mouse vào nút nầy.

<INPUT TYPE="SUBMIT" OnClick="alert" This is just an example, nothing will happen'">

**TEXT:** Nhập một dòng text vào fields. Dùng thuộc tính **SIZE** và **MAXLENGTH**để quy định kích thước. Trong trường hợp cần nhập nhiều dòng, phải dùng thẻ**<TEXTAREA>**. Ví dụ: <INPUT TYPE="TEXT">

**TEXTAREA:**Nhập nhiều dòng Text vào field.Ví dụ:

<TEXTAREA NAME="descr" COLS="30" ROWS="3" OnBlur="count\_char(document.egForm.descr.value)">Nhập văn bản không quá 50 ký tự (max 50 chars) </TEXTAREA>

**VALUE:** Chỉ định Text sẽ hiển thị trên các nút bấm.

##### 2.1.19. Thẻ định dạng cho đoạn văn (pharagraph) <P>

Định dạng cho một paragraph. Ví dụ:

<H1>The Paragraph element</H1>

<P>The paragraph element is used to denote paragraph blocks</P>

<P>This would be the second paragraph.</P>

Thuộc tính:

**ALIGN=left/center/right.** Ví dụ:

<P ALIGN=LEFT> ... </P> //Tất cả text trong paragraph này sẽ được canh về bên trái lề trang. Đây cũng là mặc định.

<P ALIGN=CENTER> ... </P> //Canh giữa

<P ALIGN=RIGHT> ... </P> //Canh phải

##### 2.1.20. Thẻ gạch ngang chữ <STRIKE> hay <S>

Làm dấu gạch ngang giửa text.

Ví dụ: <STRIKE> Văn bản này bị gạch ngang</STRIKE>  
Sẽ hiển thị là:~~Văn bản này bị gạch ngang~~

##### 2.1.20. Thẻ làm nổi bật <STRONG>, <EM>

Làm đậm text.

Ví dụ: Đây là <STRONG>vấn đề cần đọc kỹ</STRONG> trước khi áp dụng.

Sẽ thành: Đây là **vấn đề cần đọc kỹ** trước khi áp dụng

##### 2.1.21. Thẻ tạo bảng <TABLE>

Tạo bảng trong trang Web. Thuộc tính:

**BORDER="value":**Điều khiển việc hiển thị và kích thước đường viền. Nếu giá trị =0 là không có viền.

**CELLSPACING="value":**Chỉ định khoảng cách giửa các ô. Giá trị mặc định là 2.  
**CELLPADDING="value":**Chỉ định khoảng trắng giửa đường viền và ô. Giá trị mặc định là 1

**WIDTH="value or percent":**Chỉ định độ rộng bảng tính theo pixels, hay % của cửa sổ hiển thị.

**HEIGHT="value or percent":**Chỉ định độ cao bảng.

**ALIGN="left/right":**Canh lề trái, phải.

**VALIGN="top/bottom":**Canh lề trên, dưới.

**BGCOLOR="#rrggbb|colour name":** Chỉ định màu nền cho bảng.

**BORDERCOLOR="#rrggbb|colour name":**Chỉ định màu cho đường viền của bảng.

**BORDERCOLORLIGHT="#rrggbb|colour name":**Chỉ định màu cho phần được chiếu sáng của đường viền.

**BORDERCOLORDARK="#rrggbb|colour name":**Chỉ định màu cho phần bị tối của đường viền.

**BACKGROUND="URL of image":**Chỉ định file hình ảnh dùng làm nền cho bảng.

##### 2.1.22. Thẻ tạo côt trong bảng <TD>

Định dạng cho dữ liệu trong bảng (Table data). Chỉ định nầy có giá trị cho ô dử liệu. Thuộc tính:

**ALIGN="left/center/right". VALIGN="top/middle/bottom/baseline":** Canh lề cho Text trong ô (so với ô).

**WIDTH="value\_or\_percent"/ HEIGHT="value\_or\_percent":** Chỉ định kích thước cho ô. Bạn chỉ cần xác lập cho 1 ô chuẩn theo hàng hay cột, các ô khác sẽ giống như vậy.

**COLSPAN="value":** Mở rộng ô theo cột. Giá trị mặc nhiên là 1.

**ROWSPAN="value":** Mở rộng ô theo hàng.

**BGCOLOR="#rrggbb|colour name"/ BORDERCOLOR="#rrggbb|colour name"/ BORDERCOLORLIGHT="#rrggbb|colour name"/ BORDERCOLORDARK="#rrggbb|colour name"/ BACKGROUND="URL of image":** Chỉ định màu cho nền ô, đường viền ô, hình nền ô. Cách xác lập giống như bảng.

##### 2.1.23. Thẻ tạo dòng trong bảng <TR>

Xác lập cho hàng (**table row**). Thuộc tính và cách xác lập giống như cột và bảng.

##### 2.1.24. Thẻ Tiêu đề <TITLE>

Dùng đặt tiêu đề cho tài liệu HTML. Tiêu đề nầy không hiển thị trong phần nội dung tranh Web, mà sẽ được hiển thị trong thanh tiêu đề của trình Browse, trong danh sách lưu trử hay tìm kiếm.

Thẻ nầy phải nằm trong phần <HEAD>

<HEAD>   
<TITLE>Dòng tiêu đề ở đây</TITLE>   
</HEAD>

##### 2.1.25. Thẻ danh sách <OL>

Sắp xếp văn bản theo kiểu danh sách. Thường đi chung với ***<LI>* (list item)**.

Thẻ <li> định nghĩa các mục trong một danh sách.

Thẻ <li> được dùng trong danh sách có thứ tự <ol> (order list) và không có thứ tự ***<ul>*** (unorder list).

<OL>

<LI>Café

<LI>Trà sữa

<LI>Sinh tố

</OL>

Thuộc tính:

**TYPE:** Chỉ định cách đặt ký hiệu đầu dòng:

* (TYPE=A) - Chữ hoa. Ví dụ: A, B, C ...
* (TYPE=a) - Chữ thường. Ví dụ: a, b, c ...
* (TYPE=I) - Số La Mã hoa. Ví dụ: I, II, III ...
* (TYPE=i) - Số La Mã thường. Ví dụ: i, ii, iii ...
* (TYPE=1) - Số Ả Rập (mặc định). Ví dụ: 1, 2, 3 ...

**START:**Khai báo số dùng làm mặc định và sẽ được chuyển đổi qua TYPE trước khi hiển thị. Ví dụ:START=5 sẽ được hiển thị là 'E', 'e', 'V', 'v', hay '5' theo TYPE.

Ví dụ:

<OL TYPE=a START=3>

<LI>Café

<LI>Trà sữa

<LI>Sinh tố

</OL>   
//Danh sách có ký hiệu đầu dòng là chử thường, bắt đầu từ 'c'.

* Và một số thẻ khác, các bạn có thể tham khảo ở địa chỉ: <http://www.w3schools.com/>

## BÀI 2. KIẾN THỨC CƠ BẢN VỀ LẬP TRÌNH PHP

### PHP là gì:

PHP là ngôn ngữ lập trình mã nguồn mở được ra đời năm 1994 và được cả cộng đồng phát triển. PHP là ngôn ngữ lập trình web sử dụng cho ứng dụng về web động (Web có liên kết cơ sở dữ liệu).

### Cách nhúng mã PHP vào trang web

Có nhiều cách để **nhúng PHP vào tài liệu html**, trong PHP có các cách như sau:

* Cách 1

<?php

// Viết code PHP ở đây

?>

* Cách 2

<?

// Viết code PHP ở đây

?>

* Cách 3

<script language="php">

// Viết code PHP ở đây

</script>

Trong 3 cách thì cách 1 đúng chuẩn nhất và thường được dùng nhất. Tiếp đến ta tạo file index.php (như bên dưới) và lưu trong thư mục test theo đường dẫn C://xampp\htdocs\test\index.php

<html>

<head>

<title>Hello world PHP code</title>

</head>

<body>

<?php

echo "<h2>Hello world</h2>";

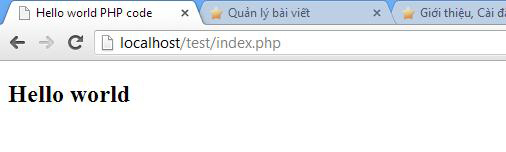
?>

</body>

</html>

Lưu lại và mở trình duyệt lên gõ link đến file index.php mà chúng ta vừa viết <http://localhost/test/index.php>

Kết quả:



Hình 1. . Sử dụng PHP viết website đầu tiên

Lưu ý: Có thể chèn nhiều thẻ <?php> vào tài liệu html.

Dưới đây là ví dụ **PHP** căn bản nhất khi chúng ta mới **học lập trình PHP**, qua đây các bạn hiểu được cách **nhúng mã PHP vào tài liệu html**, câu lệnh cơ bản của PHP.

<html>

<head>

<title>Hello world PHP code</title>

</head>

<body>

<div id="header">

<?php

echo "<h2>Hello world</h2>";

?>

</div>

<div id="content">

<?php

echo "Content here";

?>

</div>

<?php

// code here

?>

</body>

</html>

### Cú pháp PHP căn bản

Trong PHP kết thúc câu lệnh bằng dấu chấm phẩy " ; "

Xuất dữ liệu ra màn hình bằng câu lệnh " echo " hoặc " print"

|  |  |
| --- | --- |
|  | <?php     echo "Nguyen Minh Khiem"; // Nguyen Minh Khiem  print " Nguyen Minh Khiem "; // Nguyen Minh Khiem  ?> |

Nối chuỗi với chuỗi hoặc chuỗi với biến bằng dấu chấm " . " hoặc dấu phẩy " , "

|  |  |
| --- | --- |
|  | <?php      $firstname = " Nguyen Minh "      $lastname  = "Khiem";      $fullname  = $firstname.$lastname; // nối biến      $address   = "Ninh Kieu "." - "."Can Tho"; // nối chuỗi  ?> |
|  |  |

Ghi chú một hoặc nhiều dòng lệnh trong PHP

|  |  |
| --- | --- |
|  | <?php      // Ghi chú một dòng lệnh      /\* Ghi chú nhiều dòng lệnh \*/  ?> |

#### 3.1 Biến trong PHP

Trong PHP biến được ký hiệu bởi dấu "$" đi sau là các ký tự bao gồm các chữ cái thường (a - z) in hoa (A - Z) và các chữ số (0 - 9) và dấu "gạch dưới " ( \_ ).

Biến bắt đầu bằng dấu "$" sau đó là các ký tự chữ cái hoặc dấu gạch dưới  
Biến không được viết cách

Biến không được bắt đầu bằng chữ số

Ví dụ:

<?php

$name = " Nguyen Minh Khiem "; // kiểu chuỗi

$Name = 123; // kiểu số nguyên

$num12 = 1.2; // kiểu số thực

$\_bien = "giá trị";

$12name = 10; // biến không được bắt đầu bằng chữ số

a = "10"; // biến phải có dấu $ ở trước

?>

Chú ý: Trong PHP phân biệt chữ in hoa và in thường, biến $a và $A là hai biến hoàn toàn khác nhau.

#### 3.2. Hằng trong PHP

Trong PHP có hai loại hằng, hằng có sẵn trong PHP và hằng do chúng ta tự định nghĩa.

* Hằng có sẵn trong PHP, ví dụ:

|  |  |
| --- | --- |
|  | <?php     TRUE , FALSE ; // có giá trị tương ứng là 1 và 0     PHP\_VERSION  ; // cho biết phiên bản PHP trong server của bạn     PHP\_OS ;       // cho biết server của bạn là linux hay window  ?> |

Và rất nhiều các hằng khác như E\_ERROR, E\_WARNING,…

* Hằng do chúng ta định nghĩa, để định nghĩa hằng chúng ta dùng cấu trúc sau:

|  |  |
| --- | --- |
|  | <?php     define("NAME","NGUYENKHIEM"); // định nghĩa hằng có tên là NAME     echo NAME; // xuất ra màn hình " NGUYENKHIEM"  ?> |

#### 3.3 Toán tử trong PHP

##### 3.3.1. Toán tử gán:

Dễ thấy ngay trong khi ta gán giá trị cho một biến, ví dụ:

|  |  |
| --- | --- |
|  | <?php     $name = "MinhKhiem"; // Gán giá trị "MinhKhiem" cho bien $name     $age = 25 ; // gán giá trị 25 cho biến $age  ?> |

##### 3.3.2. Toán tử số học:

Là các phép toán cơ bản mà các bạn hay thấy như cộng, trừ, nhân, chia...

|  |  |  |  |
| --- | --- | --- | --- |
| Tên | Ký hiệu | Mô tả | Ví dụ |
| Phép cộng | + | Cộng hai số hạng | $a + $b |
| Phép trừ | - | Trừ hai số hạng | $a - $b |
| Phép nhân | \* | Nhân hai số hạng | $a \* b |
| Phép chia | / | Chia hai số hạng | $a / $b |
| Phép chia lấy dư | % | Chi lấy dư | 5 % 2 = 1 |

Bảng 1. . Toán tử số học trong PHP

##### 3.3.3. Toán tử so sánh:

Các phép toán logic như so sánh bằng, lớn hơn hoặc nhỏ hơn…

|  |  |  |  |
| --- | --- | --- | --- |
| Tên | Ký hiệu | Mô tả | Ví dụ |
| So sánh bằng | == | Hai số hạng bằng nhau | $a == $b |
| So sánh khác | != | Hai số hạng khác nhau | $a != $b |
| So sánh lớn hơn | > | So sánh lớn hơn | $a > b |
| So sánh nhỏ hơn | < | So sánh nhỏ hơn | $a < $b |
| Lớn hơn hoặc bằng | >= | Lớn hơn hoặc bằng | $a >= $b |
| Nhỏ hơn hoặc bằng | <= | Nhỏ hơn hoặc bằng | $a <= $b |

Bảng 1. . Toán tử so sánh trong PHP

Ví dụ: Toán tử số học

|  |  |
| --- | --- |
|  | <?php     // Toán tử số học     $a = 5;     $b = 10;     $c = $a + $b;     $d = $a \* $b;     echo "Tổng $a và $b : ".$c."<br />";     echo "Tích $a và $b : ".$d;    ?> |

Ví dụ: Toán tử so sánh

|  |  |
| --- | --- |
|  | <?php     // Toán tử so sánh     $a = 5;     $b = 10;     if($a > $b){        echo "$a lớn hơn $b";     }else{        echo "$a nhỏ hơn $b";     }  ?> |

##### 3.3.4. Toán tử logic

Toán tử logic là tổ hợp các giá trị boolean có kết quả trả về là TRUE hoặc FALSE

|  |  |  |  |
| --- | --- | --- | --- |
| Tên | Ký hiệu | Mô tả | Ví dụ |
| Phép và | && hoặc and | Cả 2 vế phải thoả mãn điều kiện | $a > $b && $a > $c |
| Phép hoặc | || hoặc or | Một trong 2 thoả mãn điều kiện là được | $a > $b || $a < $c |
| Phủ định | ! | Phủ định 1 điều kiện, giá trị nào đó | $a != b |

Bảng 1. . Toán tử logic trong PHP

Ví dụ:

<?php

// Toán tử logic

$diem = 7;

if($diem <= 5){

echo "Bạn học tệ quá!";

}elseif($diem > 5 && $diem <= 8){

echo "Bạn học rất khá!";

}else{

echo "Bạn học rất giỏi";

}

?>

#### 3.4 Biểu thức, câu lệnh điều kiện, vòng lặp trong PHP

##### 3.4.1.Biểu thức điều kiện (If –Else)

Cú pháp:

if (điều kiện) {// câu lệnh PHP}

if (điều kiện){// câu lệnh thoả mãn điều kiện}

else {// trái lại của điều kiện}

Ví dụ:

<?php

   $a = 10;

   $b = 8;

   if($a == $b){

      echo "a bằng b";

   }else{

      echo "a khác b";

   }

?>

##### 3.4.2 Vòng lặp trong PHP

**a. Vòng lặp for**: Vòng lặp for được lặp với số lần lặp biết trước, ví dụ:

<?php

   for($i=1;$i<=10;$i++){

      echo $i."<br />";

   }

?>

Ở đây vòng lặp sẽ chạy bắt đầu từ vị trí $i = 1, và chạy đến khi nào $i = 10 thì dừng lại.

**b. Vòng lặp while**: Vòng lặp while sẽ kiểm tra điều kiện nếu đúng thì mới thực hiện các câu lệnh. Cú pháp:

while (điều kiện) {// câu lệnh thực thi}

Ví dụ:

<?php

$a = 1;

while($a < 10){

echo "Gia tri cua a la : ".$a."<br />";

$a++;

}

?>

**c. Vòng lặp do while**: Khác với while, vòng lặp do while sẽ thực hiện ít nhất một lần xong mới bắt đầu kiể tra điều kiện.

<?php

$a = 1;

do{

echo "Gia tri cua a la : ".$a."<br />";

$a++;

}

while($a < 10);

?>

##### 3.4.3 Biểu thức switch case

Cú pháp:

switch(biến){    
      case  giá trị 1 : // hành động ; break;

case  giá trị 2 : // hành động ; break;

      case  giá trị n : // hành động ; break;

default   : // hành động ; break;

}

Switch case được sinh ra để hỗ trợ cho biểu thức If Else khi bài toán của chúng ta có quá nhiều điều kiện so sánh ví dụ ta có bài toán tìm thứ trong tuần:

<?php

$today = "";

switch($thu=4){

case "2" : $today = "Thu 2"; break;

case "3" : $today = "Thu 3"; break;

case "4" : $today = "Thu 4"; break;

case "5" : $today = "Thu 5"; break;

case "6" : $today = "Thu 6"; break;

case "7" : $today = "Thu 7"; break;

case "8" : $today = "Chu nhat"; break;

default : $today = "Khong tim thay!"; break;

}

echo "Hôm nay là : ".$today;

?>

#### 3.5 Mảng trong PHP

Mảng gồm có key và value, key là khoá để truy cập vào từng **phần tử của mảng**. Các bạn có thể hiểu nôm na mảng nó cũng giống như chiếc tủ quần áo nhà mình vậy, trong tủ có nhiều ngăn mỗi ngăn lại có một chìa khoá riêng, bạn muốn lấy quần áo ở ngăn nào thì cần phải có được chìa khoá ngăn đó.

Để khởi tạo mảng ta dùng từ khoá array. Cấu trúc: $mang = array(key => value);

Ví dụ:

<?php

    $mang   = array(); // khởi tạo mảng $mang rỗng

    $person = array(

                 "name" => "MinhKhiem",

                 "age" => "25",

                 "gender" => "male",

                 "email"=>"nmkhiem@gmail.com"

              );

            // $person[key] => $person[value]

            // name         =  MinhKhiem

            // age          =  25

            // .....................

    $score = array("toan" => 7,"ly" => 5,"hoa" => 3);

?>

Khi một mảng có nhiều cặp key và value thì mỗi cặp key => value cách nhau bởi dấu phảy ","

Phân loại mảng:  Mảng được chưa ra làm 3 loại chính:

+ Mảng tuần tự

+ Mảng không tuần tự

+ Mảng đa chiều

##### 3.5.1. Mảng tuần tự

Mảng tuần tự là mảng có key là chữ số được bắt đầu bằng 0 và sắp xếp tằng dần:

<?php

$person = array("Thanh Hải","Ngọc Quyền","Minh Khiem","Hưng");

echo $person[0] ; // lấy ra tên Thanh Hải

echo $person[1]; // lấy ra Ngọc Quyền

echo $person[3]; // lấy ra Hưng

?>

* Thêm phần tử vào mảng:

<?php

$person[] = "Tấn Tài"; // Được thêm vào vị trí cuối cùng của mảng

echo $person[4]; // lấy ra Tấn Tài

?>

##### 3.5.2. Mảng không tuần tự

Là mảng mà key của nó không sắp xếp theo thứ tự, nó là các ký tự chữ:

<?php

$person = array(

"name" => "MinhKhiem",

"age" => "25",

"gender" => "male",

"email"=>"nmkhiem@gmail.com"

);

echo $person['name']; // lấy ra MinhKhiem

echo $person['gender']; // lấy ra male

?>

* Thêm phần tử vào mảng:

<?php

   $person['single'] = "yes";

   echo $person['single'];

?>

##### 3.5.3. Mảng đa chiều

Là một mảng lớn có nhiều mảng con nằm trong nó, ví dụ như cái tủ quần áo nhà bạn, trong tủ có ngăn chứa quần áo, trong mỗi ngăn đó nó lại có nhiều ngăn con để chứa những đồ khác.

<?php

   $aosomi = array(

              "hang"    => "tomy",

              "ngaysx"  => "01/05/2015",

              "chitiet" => array(

                              "mau"  => "den",

                              "size" => 30,

                              "gia"  => 200000

                           ),

              "loai"    => "quan ao"

          );

?>

Như vậy để biết được cái áo màu gì, chúng ta cần phải truy cập đến $aosomi['chitiet']['mau'].

<?php

   echo "Hang san xuat : ".$aosomi['hang']."<br />";

   echo "Ngay san xuat : ".$aosomi['ngaysx']."<br />";

   echo "Mau : ".$aosomi['chitiet']['mau']."<br />";

   echo "Gia : ".$aosomi['chitiet']['gia'];

?>

##### 3.5.4. Lặp mảng

Có nhiều cách để lặp mảng nhưng thông dụng nhất là dùng hàm foreach.

Cú pháp :

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

               // câu lệnh PHP

          }

<?php

    $person = array(

                 "name" => "MinhKhiem",

                 "age" => "25",

                 "gender" => "male",

                 "email"=>"nmkhiem@gmail.com"

              );

    foreach($person as $key => $val){

        echo "Key : ".$key." - value : ".$val."<br />";

    }

?>

##### 3.5.5. Sử dụng Session và Cookie trong PHP

Session và Cookie nó được hiểu là một phiên làm việc của một sự kiện nào đó. Dễ thấy khi bạn vào facebook hay gmail, điều đầu tiên để sử dụng được các dịch vụ này là bạn phải đăng nhập, sau khi đăng nhập thì tài khoản của bạn được cấp gán cho một phiên làm việc nó giới hạn tài khoản của bạn sẽ sử dụng dịch vụ trong bao lâu.   
 Cụ thể ở đây là sau khi bạn đăng nhập hệ thống sẽ lưu lại thông tin cơ bản của bạn ví dụ như id của bạn, tên bạn, email của bạn... vào trong SESSION (Có thể là Session hoặc Cookie). Từ đó để hệ thống kiểm tra xem bạn đã đăng nhập hay chưa và giới hạn quyền của bạn, ví dụ bạn đăng nhập rồi thì mới có thể mua hàng chẳng hạn...  
 Thời gian sống của mỗi phiên làm việc này là khác nhau, do người lập trình website quy định, thường thì Session sẽ tồn tại trong 15 phút nếu bạn không thao tác gì với website trong 15 phút thì session sẽ bị huỷ.

**a. Về Cookie**

Cũng giống như session, cookie được khởi tạo để lưu trữ một phiên làm việc nào đó, nhưng với cookie thì nó được lưu trữ trên máy người dùng, thông tin lưu trên cookia dễ bị tìm thấy.

Khởi tạo 1 cookie với cú pháp: setcookie(tên cookie, giá trị, thời gian sống); File set\_cookie.php:

<?php

setcookie("name","ConMuoi",time()+1800);

// Set cookie tên name, giá trị = ConMuoi, thời gian tồn tại 30 phút

header("location:view\_cookie.php");

?>

Tiếp tục tạo trang view\_cookie.php để lấy cookie mà chúng ta vừa khởi tạo ở file set\_cookie.php, chúng ta lấy cookie thông qua biến $\_COOKIE[ ' tên cookie ' ];

File view\_cookie.php:

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<?php

     echo "Ten la : ".$\_COOKIE['name']."<br />";

     // Ten la : ConMuoi

     echo "<a href='del\_cookie.php'>Huỷ cookie</a>";

?>

Để huỷ cookie chúng ta sử dụng cú pháp:

setcookie(tên cookie, giá trị, time() - thời gian sống);

Với time() là thời gian hiện tại trừ đi thời gian sống mà lúc chúng ta set cookie File del\_cookie.php:

 <?php

     setcookie("name","ConMuoi",time()-1800);

     echo "Đã xoá cookie!";

?>

**b. Về session**

Session được lưu trữ hoàn toàn trên server, do vậy tính bảo mật rất cao, các website hiện này thường dùng session để lưu thông tin của người dùng khi họ đăng nhập.  
Khởi tạo session: $SESSION[tên session] = " Giá trị ";

Trước khi dùng session bạn phải gọi nó trước bằng cú pháp: session\_start();

<?php

     session\_start();

     $\_SESSION['name'] = "ConMuoi";

     // Khởi tạo session name có giá trị là ConMuoi

?>

Sau khi bạn khởi tạo session thì ở bất kỳ file .php nào trên ứng dụng bạn cũng có thể sử dụng lại biến session mà bạn vừa khởi tạo khi nãy, chỉ cần bạn gọi session\_start();

<?php

     session\_start();

     echo "Ten la : ".$\_SESSION['name'];

?>

Huỷ 1 biến session nào đó: unset(tên session);

<?php

session\_start();

unset($\_SESSION['name']); // Huỷ session name

session\_destroy(); // Huỷ tất cả session

?>

##### 3.5.6. Form và xử lý form trong PHP

Cấu trúc của một form có dạng:

<form action = "" method = ""> </form>

Trong đó action sẽ là nơi nhận và xử lý dữ liệu khi người dùng nhập vào form và ấn submit để gửi. Method là phương thức truyền dữ liệu trong form, có 2 phương thức truyền dữ liệu là POST và GET. Cụ thể chúng ta sẽ xét ví dụ sau:

Ta có form với method đang là POST và action dữ liệu sẽ được gửi đến file login.php:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/ TR/ xhtml1/DTD/xhtml1-transitional.dtd">

<html xmls=”http://www.w3.org/1999/xhtml”>

<head>

<meta http- equiv=”Content-Type” content =”text/html”; charset =utf-8 />

<title> Xu ly form trong php></tilte>

</head>

<body>

<form action =”login.php” method =”post”>

Tên đăng nhập: <input type = “text” name =”user” size =”20” /><br/>

Mật khẩu : <input type = “password” name =”pass” size =”20” /><br/>

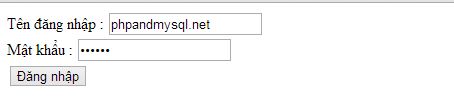
<input type =”submit” name =”login” value =”Dang nhap”

</form>

</body>

</html>

Giao diện của form khi view lên trình duyệt



- Khi người dùng nhấn nút "Đăng nhập" thì dữ liệu mà người dùng vừa nhập vào ( bao gồm username và password) sẽ được truyền theo phương thức POST sang file login.php, Ở file login.php này chúng ta sẽ lấy được dữ liệu thông qua biến môi trường $\_POST.

- Trong PHP chúng ta sẽ lấy được giá trị của các text field qua thuộc tính name, ở đây chúng ta đang có 2 text field là name="user\_name" và name="user\_pass" trong cặp thẻ input type="text" và input type="password"

File login.php:

<?php

$username = $\_POST['user\_name'];

$password = $\_POST['user\_pass'];

echo "Your name : ".$username."<br />";

echo "Your pass : ".$password;

?>

- Tương tự nếu người truyền dữ liệu theo phương thức GET thì chúng ta sẽ lấy được dữ liệu qua biến $\_GET

File login.php :

<?php

$username = $\_GET['user\_name'];

$password = $\_GET['user\_pass'];

echo "Your name : ".$username."<br />";

echo "Your pass : ".$password;

?>

- Sự khác nhau giữa POST và GET là gì?

Với phương thức POST thì dữ liệu được truyền đi ngầm sang file login.php, còn với phương thức GET dữ liệu được gửi đi và dữ liệu mà chúng ta vừa nhập vào sẽ được show lên ở thanh address bar của trình duyệt.

Việc truyền dữ liệu lên thanh địa chỉ như vậy khiến cho thông tin của chúng ta không được bảo mật, người dùng hoàn toàn có thể can thiệp đến dữ liệu của chúng ta vì vậy với những form có thông tin cần bảo mật như đăng nhập hay đăng ký thì chúng ta không nên sử dụng phương thức GET.

- Kiểm tra thông tin do người dùng nhập dữ liệu vào:

<?php

$username = "";

$password = "";

// Nếu người dùng chưa nhập dữ liệu, đưa ra thông báo

if($\_POST['user\_name'] == NULL){

echo "Please enter your name";

die();

}else{

$username = $\_POST['user\_name'];

}

// Nếu chưa nhập password

if($\_POST['user\_pass'] == NULL){

echo "Please enter your pass";

die();

}else{

$password = $\_POST['user\_pass'];

}

// Nhập đúng user và pass

if($username == "yourname" and $password == "123456"){

echo "Login successfuly";

}else{

echo "Login false";

}

?>

## BÀI 3. KIẾN THỨC CƠ BẢN VỀ JAVASCRIPT

### Giới thiệu về JavaScrip

Javascript là 1 ngôn ngữ hướng đối tượng dùng để xử lý các thành phần HTML trong 1 trang web.

Javascript chạy trên phía client (trên trình duyệt – IE, FF, Opera, Chrome...).

Javascript được tạo ra năm 1995 bởi Brendan Eich của Netscape (Mozilla hiện tại) dưới tên Mocha, sau đó đổi thành Livescript rồi Javascript.

Là ngôn ngữ hướng đối tượng. Ta có thể tạo, sử dụng các đối tượng.

Javascript chạy sử dụng trình thông dịch được tích hợp với trình duyệt.

Có 2 cách viết Javascript:

* Cách 1 : Nhúng đoạn javascript vào trong file HTML:

<script language=“javascript”>

Mã\_nguồn\_Javascript

</script>

* Cách 2 : Viết javascript thành 1 file riêng có đuôi .js và liên kết với file HTML.

Ta sử dụng thẻ <script> với thuộc tính src để liên kết 1 file javascript vào HTML:

<script language=“javascript”

src=“myscript.js”>

</script>

Cú pháp căn bản:

* Lệnh đơn: mỗi lệnh đơn kết thúc bằng;
* Khối lệnh: được bao bằng { }
* Chú thích: // và /\* .. \*/
* Cấu trúc điều khiển:
  + Rẽ nhánh: if, else, switch
  + Lặp: for, while, do... while, for ... in

Javascript không cần khai báo biến vẫn có thể sử dụng được.

Tên biến phân biệt hoa thường, phải bắt đầu bằng kí tự hoặc gạch dưới ( \_ ).

Biến nếu được khai báo thì không cần khai báo kiểu:

var a;

a = 10;

Một biến có thể chứa bất kỳ giá trị nào (nguyên, thực, chuỗi ...).

Tầm vực là tầm ảnh hưởng của biến:

Biến toàn cục: được khai báo ngoài các hàm. Biến có tác dụng từ vị trí khai báo cho đến cuối chương trình.

Biến cục bộ: được khai báo trong hàm. Biến chỉ có tác dụng trong hàm được khai báo.

Nếu trong hàm, biến cục bộ trùng tên với biến toàn cục thì biến cục bộ sẽ được sử dụng.

Kiểu dữ liệu: Biến trong javascript không cần khai báo kiểu dữ liệu

* Khai báo biến kiểu số: a = 1.4; b = 2
* Khai báo biến kiểu chuỗi: str = ‘Chuỗi dùng dấu nháy đơn’ ; str2 = “Dấu nháy kép”
* Khai báo biến boolean: var dung = true, sai = fase;
* Khai báo biến null: obj = null
* **Các phép toán cơ bản trong JavaScript**

|  |  |  |
| --- | --- | --- |
| **Phép toán** | **Mô tả** | **Ví dụ** |
| + | Phép cộng hai số | X+Y sẽ cho kết quả là 15 |
| - | Phép trừ hai số | X-Y sẽ cho kết quả là -5 |
| \* | Phép nhân hai số | X\*Y sẽ cho kết quả là 50 |
| / | Phép chia hai số | X/Y sẽ cho kết quả là 0.5 |
| % | Phép chia lấy phần dư giữa hai số | X%Y sẽ cho kết quả là 0 |
| ++ | Toán tử tăng. Tăng một giá trị cho biến | X++(hoặc ++X) sẽ cho kết quả là 6 |
| – | Toán tử giảm. Giảm một giá trị cho biến | X–(hoặc –X) sẽ cho kết quả là 4 |

Bảng 1. . Các phép toán cơ bản trong JavaScript

Lưu ý: Toán tử cộng (+) còn được dùng để nối chuỗi.

Ví dụ: ta có hai chuỗi là “” Xin chào,” và ” các bạn.”, khi thực hiện phép nối chuỗi: “Xin chào,” + ” các bạn.” ta sẽ được một chuỗi mới là tập hợp của hai chuỗi cũ: “Xin chào, các bạn.” .

* **Các phép toán so sánh**

Giả sử có biến X được gán giá trị là 5 và biến Y được gán giá trị là 10.

|  |  |  |
| --- | --- | --- |
| **Phép toán** | **Mô tả** | **Ví dụ** |
| == | Phép so sánh bằng | (X==Y) sẽ cho kết quả là false |
| != | Phép so sánh khác | (X!=Y) sẽ cho kết quả là true |
| > | Phép so sánh lớn hơn | (X>Y) sẽ cho kết quả là false |
| < | Phép so sánh nhỏ hơn | (X<Y) sẽ cho kết quả là true |
| >= | Phép so sánh lớn hơn hoặc bằng | (X>=Y) sẽ cho kết quả là false |
| <= | Phép so sánh nhỏ hơn hoặc bằng | (X<=Y) sẽ cho kết quả là true |

Bảng 1. . Các phép toán so sánh trong JavaScript

* **Các phép toán Logic**

Giả sử có biến X được gán giá trị là true và biến Y được gán giá trị là false.

|  |  |  |
| --- | --- | --- |
| **Phép toán** | **Mô tả** | **Ví dụ** |
| &&(AND) | Toán tử AND so sánh hai biểu thức. Chỉ trả về giá trị TRUE khi cả hai biểu thức đều mang giá trị là TRUE. | (X && Y) sẽ cho kết quả là false |
| ||(OR) | Toán tử OR so sánh hai biểu thức. Chỉ trả về giá trị FALSE khi cả hai biểu thức đều mang giá trị là FALSE. | (X || Y) sẽ cho kết quả là true |
| ! | Toán tử NOT phủ định biểu thức. Trả về giá trị là TRUE khi biểu thức mang giá trị FALSE và ngược lại. | !X sẽ cho kết quả là false |

Bảng 1. . Các phép toán Logic trong JavaScript

* **Các phép toán gán**

Giả sử có biến X được gán giá trị là 5 và biến Y được gán giá trị là 10.

|  |  |  |
| --- | --- | --- |
| **Phép toán** | **Mô tả** | **Ví dụ** |
| = | Gán giá trị cho một biến | X=5 |
| += | Tăng giá trị cho biến | X += 20 tương đương với X = X+20 |
| -= | Giảm giá trị cho biến | X -= 20 tương đương với X = X-20 |
| \*= | Tăng giá trị cho biến | X \*= 20 tương đương với X = X\*20 |
| /= | Giảm giá trị cho biến | X /= 20 tương đương với X = X/20 |
| %= | Gán giá trị phần chia lấy dư | X %= 20 tương đương với X = X%20 |

Bảng 1. . Các phép toán gán trong JavaScript

* **Cấu trúc cơ bản:**
* **if, else**

Ví dụ:

if (n % 2 == 0)

document.write(“Chẳn”);

else

document.write(“Lẻ”);

switch :

switch (n)

{

case 2 :

document.write(“Thứ 2”);

break;

case 3 :

document.write(“Thứ 3”);

break;

}

* **for**

Ví dụ:

var i=0;

for (i=0;i<=5;i++)

{

document.write("The number is " + i);

document.write("<br/>");

}

* **while**

Ví dụ:

var i=0;

while (i<=5)

{

document.write("The number is " + i);

document.write("<br/>");

i++;

}

* **For … in**

Ví dụ:

var x;

var mycars = new Array();

mycars[0] = "Saab";

mycars[1] = "Volvo";

mycars[2] = "BMW";

for (x in mycars)

{

document.write(mycars[x] + "<br/>");

}

* **Hàm**

Cú pháp:

function functionname (var1,var2,...,varX) {some code}

Hàm không được thực thi khi trang web được load

Hàm chỉ thực thi khi được gọi (ví dụ trong sự kiện hoặc trực tiếp)

Hàm có thể đặt ở trong <head> hoặc trong <body> nhưng nên đặt trong <head> . Nó sẽ luôn được nạp trước khi gọi**.**

* **Hàm trả về giá trị**

Từ khóa return dùng để trả về giá trị cho hàm

Cú pháp:

function Name(var1,var2,...,varX) {

some code

return value;

}

Ví dụ:

<html>

<head>

<script type =”text/javascript”>

function product(a,b)

{return a \*b;}

</script>

</head>

<body>

<script type =”text/javascript”>

Document.write(product(9,5))

</script>

</body>

<html>

* **Các hàm thông dụng**

**Hàm alert**

alert ( “Nội dung thông báo”): hiển thị hộp thoại thông báo có 1 nút OK.

<html>

<head>

<title>Fuction</title>

</head>

<body>

<script>

Alert(“hello world”);

</script>

</body>

<html>

**Hàm prompt**

prompt (“Thông báo”, Giá\_trị\_mặc\_định): hiển thị hộp thoại với câu thông báo và 2 nút OK, Cancel cho phép người dùng nhập vào 1 giá trị.

Nếu người dùng nhấn OK, hàm prompt() sẽ trả về chuỗi được nhập, ngược lại giá\_trị\_mặc\_định sẽ trả về**.**

<script>

a =prompt (“Ho và ten dem cua ban: ”);

b =prompt (“Ten cua ban: ”);

document.write(“Ho ten day du là: ”+a +’ ‘ +b)

</script>

**Hàm Confirm();**

confirm(“Thông báo”): hiển thị câu thông báo cũng với 2 nút OK, Cancel. confirm() trả về giá trị true nếu OK được nhấn và false nếu Cancel được nhấn.

<script>

a =prompt (“Nhap so a : ”);

b =prompt (“Nhap so b: ”);

c =confirm (a + “ lon hon ” +b +’khong?’);

if(c == true)

document.write(a+ “ < ”+ b)

</script>

**Hàm document.writeln()**

Hàm document.write(“Chuỗi”) và document.writeln(“Chuỗi”) dùng để ghi 1 chuỗi ra trang HTML.

**Hàm Eval()**

Hàm eval(“Chuỗi”) : hàm trả về kết quả thực thi câu lệnh javascript trong Chuỗi. Ví dụ:

<script>

var a = “123”

var b = “456”

var c = a+b;

document.write(c) // ket qua la :123456?

</script>

So sánh với kết quả

<script>

a = eval(prompt (“Nhap so a : ”));

b = eval(prompt (“Nhap so b: ”));

c = a + b ;

document.write(c)

</script>

**Hàm ParseInt()**

parseInt(“Chuỗi”): hàm đổi chuỗi ra số nguyên.

Nếu chuỗi gồm các kí số rồi đến kí tự (123abc) thì các kí tự sẽ bị bỏ qua (trả về 123).

Nếu chuỗi bắt đầu không phải kí số thì parseInt sẽ trả về NaN (Not a Number).

**Hàm ParseFloat()**

parseFloat(“Chuỗi”): hàm đổi chuỗi ra số thực.

Nếu Chuỗi gồm các kí số rồi đến kí tự (123abc) thì các kí tự sẽ bị bỏ qua (trả về 123).

Nếu Chuỗi bắt đầu không phải kí số thì parseFloat sẽ trả về NaN (Not a Number).

**Hàm IsNaN()**

isNaN(“Chuỗi”): hàm kiểm tra xem một chuỗi không phải là số không? Nếu là số trả về false, ngược lại true.

* **Xử lý sự kiện**

Sử dụng javascript ta có thể tạo các trang web động**.**

Event là các sự kiện phát sinh khi người dùng tương tác với các element trong trang web.

Mỗi element có 1 số các sự kiện riêng có thể kích hoạt javascript.

Một số các ví dụ:

* + 1 cú click chuột.
  + 1 trang web hay 1 hình ảnh đang được nạp.
  + Di chuyển chuột lên trên 1 element.
  + Chọn 1 ô text field trong HTMl.
  + Gởi dữ liệu trong form HTML.
  + Nhấn phím.
  + Cách dùng sự kiện.

Ta thêm vào element cần xử lý sự kiện thuộc tính như sau : Loại\_Sự\_Kiện = “ Mã nguồn Javascript hoặc các hàm” .

Loại sự kiện:

* onLoad, onUnload: khi người dùng nạp hay thoát khỏi trang.
* onFocus, onBlur , onChange: element được focus, rời focus, đang thay đổi giá trị.
* onSubmit: trước khi dữ liệu được gởi về server.
* onMouseOver hoặc onMouseOut: khi chuột phía trên hoặc ngoài element.
* onClick: element được click

****

# PHẦN 2: LẬP TRÌNH WEB VỚI FRAMEWORK

## BÀI 1. GIỚI THIỆU VÀ HƯỚNG DẪN CÀI ĐẶT

**Nội dung:**

1. Giới thiệu về các Framework hiện nay.
2. Giới thiệu CakePHP.
3. Một số khái niệm.
4. Cài đặt/cấu hình môi trường phát triển.
5. Các bước lập trình ứng dụng, triển khai.

### I. Giới thiệu về các Framework hiện nay

#### 1.1. Định nghĩa Framework

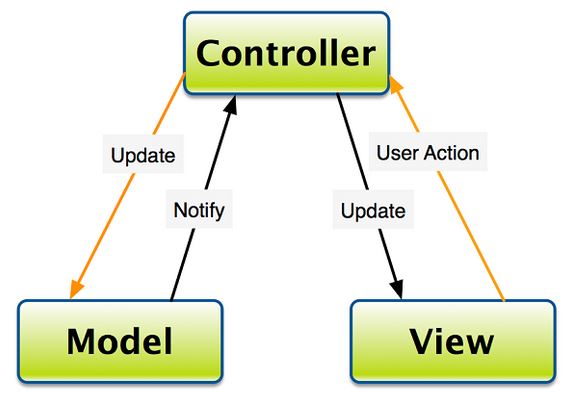
Có lẽ thuật ngữ **Framework** không xa lạ gì đối với những bạn đã từng học PHP, tuy nhiên cũng không ít người đã từng thắc mắc nó là gì. Vậy **Framework** là gì? **Framework** nói một cách đơn giản là **một thư viện mã lệnh được xây dựng sẵn** để chúng ta sử dụng.

* **PHP framework là gì?**

Như các bạn đã biết, **PHP** là một ngôn ngữ script rất phổ biến hiện nay bởi những lý do: linh hoạt, dễ sử dụng, dễ học,… Nhưng đôi khi việc viết mã **PHP**, hay bất cứ ngôn ngữ (lập trình) nào khác, có thể trở nên đơn điệu và lủng củng. Đó là lúc **PHP framework** có thể giúp bạn.

**PHP framework** làm cho sự phát triển của những ứng dụng web viết bằng ngôn ngữ PHP trở nên trôi chảy hơn, bằng cách cung cấp một cấu trúc cơ bản để xây dựng những ứng dụng đó. Hay nói cách khác, **PHP framework** giúp các bạn thúc đẩy nhanh quá trình phát triển ứng dụng, giúp bạn tiết kiệm được thời gian, tăng sự ổn định cho ứng dụng, và giảm thiểu số lần phải viết lại mã cho lập trình viên. Ngoài ra, **Framework** còn giúp những người mới bắt đầu có thể xây dựng các ứng dụng ổn định hơn nhờ việc tương tác chính xác giữa các Database, mã (PHP) và giao diện (HTML) một cách riêng biệt. Điều này cho phép bạn dành nhiều thời gian để tạo ra các ứng dụng web, hơn là phí thời gian để viết các đoạn mã lặp lại trong một project.

Ý tưởng chung đằng sau cách thức làm việc của một **PHP framework** được kể đến là **Model View Controller** (MVC). MVC là mô hình (kiến trúc) trong lập trình, cho phép tách biệt các mã nghiệp vụ (business logic) và giao diện (User Interface- UI) thành các phần riêng biệt, điều này đồng nghĩa với việc ta có thể chỉnh sửa chúng một cách riêng lẻ. Trong cụm từ **MVC** thì: Model (M) có thể hiểu là phần xử lý các thao tác về nghiệp vụ (business logic), View được hiểu là phần xử lý lớp giao diện (presentation layer), và Controller làm nhiệm vụ lọc các request được gọi từ user, có chức năng như 1 route: điều chỉnh, phân luồng các yêu cầu để gọi đúng Model & View thích hợp. Về cơ bản, **MVC** chia nhỏ quá trình xử lý của 1 ứng dụng, vì thế nên bạn có thể làm việc trên từng thành phần riêng lẻ, trong khi những thành phần khác sẽ không bị ảnh hưởng tới. Thực chất, điều này giúp đỡ bạn lập trình PHP nhanh hơn và ít phức tạp hơn.



Hình 2. . Mô hình MVC (Model – Controller - View)

#### 1.2. Các Framework hiện nay

Hầu hết các Framework có thế mạnh và điểm yếu khác nhau. Tùy vào mục đích và yêu cầu mà chúng ta lựa chọn Framework phù hợp nhất. Dưới đây là 5 framework được đánh giá là tốt và phổ biến nhất hiện nay (tính đến tháng 12/2014):

##### 1.2.1. Laravel

Đây là PHP framework phổ biến nhất hiện nay. Laravel xuất hiện vào tháng 04-2011 và cha đẻ của nó là Taylor Otwell. Laravel là một open source, là một framework dùng để xây dựng web application, được thiết kế dựa trên mô hình MVC (Model, Controller, View), toàn bộ source code được đặt trên github.

Theo kết qủa khảo sát của các Developer vào tháng 12 năm 2013, thì Laravel Framework đứng top 1 một trong những framework phổ biến nhất. Laravel có Document khá đầy đủ và rõ ràng, nên việc tiếp cận nó cho người mới học rất dễ dàng.

Website: <http://laravel.com/>

##### 1.2.2. Symfony

Được ra đời nhằm mục đích giúp đỡ nâng cao hơn cho những lập trình viên muốn tạo ra các ứng dụng website doanh nghiệp. Ra đời vào năm 2005 phiên bản mới nhất là Symfony2 cá tính năng cơ bản: một mã nguồn linh hoạt, dễ nâng cấp bảo trì, tối ưu hóa hiệu quả công việc (thích hợp Varnish, HTTP accelerator), kiểm soát bảo mật (chứng thực điện tử, eBanking), tự do sáng tạo, tích hợp ứng dụng và module,…

Một khuôn khổ PHP hoàn chỉnh được thiết kế để tối ưu hóa sự phát triển của các ứng dụng web. Symfony được viết hoàn toàn bằng ngôn ngữ lập trình web PHP. Nó đã được kiểm tra kỹ lưỡng trong các dự án thực tế, và thực sự là sử dụng cho các [thiết kế web bán hàng](http://vivicorp.com/thietkeweb-gioi-thieu-san-pham.html) thương mại điện tử có nhu cầu cao. Nó tương thích với hầu hết các công cụ cơ sở dữ liệu có sẵn, bao gồm cả [MySQL], PostgreSQL, Oracle, và Microsoft SQL Server. Nó chạy trên nền tảng Unix và Windows.

Website: <http://www.symfony-project.com/>

##### 1.2.3. CodeIgniter

Được biết đến như một framework dễ hiểu và dễ sử dụng, cho hiệu suất cao. Không giống như SymfonyPHP framework này phục vụ mục đích lý tưởng cho việc xây dưng các ứng dụng chia sẻ, lưu trữ. Nó cung cấp các giải pháp đơn giản, và có một thư viện video hướng dẫn phong phú và cung cấp sẵn hướng dẫn sử dụng cho người mới bất đầu. Phiên bản chính thức đầu tiên được phát hành 28/2/2006.

Phiên bản mới nhất hiện nay là **CodeIgniter 3.x** các tính năng cơ bản hỗ trợ kết nối và tương tác đa nền tảng cơ sở dư liệu, tương tác với cơ sở dữ liệu thông qua active records, quản lý Session, định dạng và chuẩn hóa form và dữ liệu đầu vào, hỗ trợ Caching để tăng tốc độ thực thi và giảm tải cho máy chủ, hỗ trợ Template Engine hoặc sử dụng chính PHP tags để điều hướng trong views, hỗ trợ Hooks, các lớp ngoại và Plugins, mềm dẻo trong việc định tuyến URL.

Website: <http://www.codeigniter.com/>

##### 1.2.4. Zend Framework

Zend là một trong những framework phổ biến nhất hiện nay. Nó có tính năng mạnh mẽ, thường được sử dụng cho các công ty lớn. Được ra đời vào năm 2005 phiên bản Zend framework 1.0 (phát hành 1/7/2007) Phiên bản mới nhất hiện nay là Zend Framework 2 (phát hành 6/9/2012) là một mã nguồn mở phát trển các ứng dụng web và dịch vụ sử dụng trên PHP 5.3+. Có các thành phần thư viện hỗ trợ API của các nhà cung cấp hàng đầu như Google, Yahoo, Flick...

Zend Framework có một cộng đồng phát triển rộng lớn, và nó tập trung vào các ứng dụng web theo phong cách 2.0. Vì được phổ biến rộng rãi, và có một cộng đồng người dùng tích cực, Zend được gọi là “Công ty PHP”. Zend là một trong những framework phổ biến nhất hiện nay. Nó có các tính năng mạnh mẽ, thường được sử dụng cho các công ty lớn, và bạn cần phải có lượng kiến thức khá sâu rộng về PHP để có thể sử dụng được nó.

Website: <http://framework.zend.com/>

##### 1.2.5. CakePHP

CakePHP là một framework phổ biến cho phát triển ứng dụng web. Đơn giản của nó để tìm hiểu rằng cung cấp một cấu trúc cơ bản của tổ chức, giữ cho toàn bộ ứng dụng của nhất quán và hợp lý. Nó có thể xử lý mọi khía cạnh, từ yêu cầu ban đầu của người dùng tất cả các cách để cuối cùng vẽ được một thiết kế web. Và kể từ khi khuôn khổ theo các nguyên tắc của MVC, nó cho phép bạn dễ dàng tùy biến và mở rộng hầu hết các khía cạnh của ứng dụng của bạn.

Ra đời năm 2005 đến nay đã có 3 phiên bản, phiên bản mới nhất hiện nay Cake PHP 3.0.5 (phát hành 17/5/2015). Một số tính năng Cake PHP là mã nguồn mở, tương thích PHP4 và PHP5, đa ngôn ngữ, sinh code tự động, phân quyền (ACL), xây dựng nhiều thư viện hỗ trợ cho View (Ajax, HTML Form, Javascript,..) và Controller (Email, Security, Session, Cookies,..), dễ dàng viết thêm thư viện, liên kết với ứng dụng khác thông qua Vendors và đa giao diện.

Nó dựa trên cùng một nguyên tắc thiết kế với Ruby on Rails, là một framework mạnh về khía cạnh rapid development, giúp lập trình viên đẩy nhanh quá trình phát triển ứng dụng của họ.

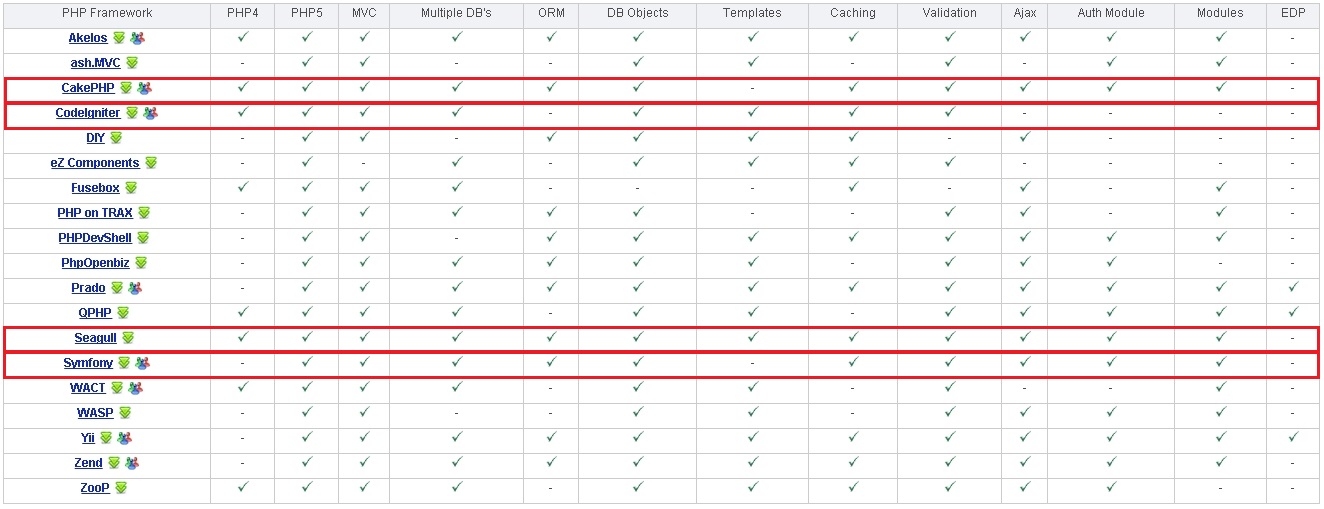
Website: <http://www.cakephp.org>

##### 1.2.6. Kết luận

Có rất nhiều lý do khác nhau để các lập trình viên sử dụng PHP framework, nhưng một trong những lý do chính vẫn là khả năng giúp các lập trình viên tăng tốc quá trình phát triển ứng dụng. Việc sử dụng lại các mã lệnh giống nhau trong nhiều project sẽ giúp các bạn tiết kiệm được thời gian và công sức 1 cách đáng kể. Một framework sẽ cung cấp sẵn các module nền tảng cần thiết để xây dựng 1 project, vì thế, các lập trình viên có thể tận dụng được thời gian để phát triển các ứng dụng thực tế, hơn là mất thời gian để xây dựng lại nền tảng trên mỗi project.

Sự ổn định là lý do lớn đối với các lập trình viên đang sử dụng Framework. Tính đơn giản là một điểm mạnh của *PHP*, đó là lý do tại sao lại có nhiều người thích sử dụng nó, nhưng đồng thời đó cũng là điểm yếu của nó. PHP thì khá dễ học và sử dụng, đặc biệt là đối với những người mới làm quen với lập trình, tuy nhiên, họ có thể thường xuyên viết mã một cách không khoa học và thậm chí không hề nhận thức được điều này, với PHP, trong nhiều trường hợp các ứng dụng vẫn sẽ làm việc được, nhưng vô tình họ có thể tạo ra các lỗ hổng bảo mật lớn trong mã lệnh của mình, và bị hacker khai thác.

Hầu hết các PHP framework đều có sẵn rất nhiều thành phần mở rộng (extensive), và cũng có rất nhiều framework khác nhau để các bạn lựa chọn. Bạn thậm chí còn có thể tự viết riêng cho mình một framework. Tuy nhiên, bạn nên cân nhắc kỹ trước khi quyết định sử dụng một framework nào cho mình hay không, nên tự đặt câu hỏi cho chính mình là: Nó có tiết kiệm được cho bạn thời gian và công sức hay không? Có giúp ứng dụng bạn hoạt động tốt hơn không? Có cải thiện được sự ổn định cho ứng dụng không? Hãy tìm ra câu trả lời cho chính mình để quyết định xem bạn có cần sử dụng framework hay chưa, và nên sử dụng framework nào.



Hình 2. . Bảng so sách sự hỗ trợ và tính năng của các phiên bản Framework

### II. Giới thiệu CakePHP

Ở mục I chúng ta đã có định nghĩa, cũng như khái quát sơ lược về CakePHP. Ở mục này chúng ta sẽ đi vào cụ thể và chi tiết hơn về CakePHP: CakePHP là gì? Các tính năng của CakePHP.

#### 2.1. CakePHP là gì?

Chúng ta đã có khái niệm về CakePHP nói riêng và Framework nói chung ở phần I. Ở đây chỉ muốn sẽ nhắc lại và nhấn mạnh vài điểm:

* CakePHP là một Framework mã nguồn mở, miễn phí dành cho việc phát triển các ứng dụng web bằng ngôn ngữ PHP, mục đích của CakePHP là cung cấp một framework cho người sử dụng PHP phát triển những ứng dụng web nhanh, mạnh mà không mất tính linh hoạt của nó.
* CakePHP là một nền tảng phát triển ứng dụng nhanh, cấu trúc của nó được tạo ra để lập trình viên tạo các ứng dụng web. Nó tạo ra một nền tảng có cấu trúc, cho phép bạn làm việc trên cấu trúc đó một cách nhanh chóng mà không mất đi sự uyển chuyển. Ngoài ra, CakePHP loại bỏ sự nhàm chán trong phát triển ứng dụng web: cung cấp các công cụ để viết thứ ta cần, thay vì phải làm đi làm lại một thứ. Khi bắt đầu tạo mới dự án (project), chỉ cần tạo một bản copy của CakePHP và tập trung vào việc chính của dự án.
* CakePHP có một đội ngũ phát triển và cộng đồng năng động, điều này mang lại giá trị to lớn cho các dự án. Ngoài việc giúp bạn khỏi phải làm đi làm lại một việc nào đó, sử dụng CakePHP đồng nghĩa với phần cốt lõi của ứng dụng của bạn đã được kiểm chứng và cải tiến không ngừng.

#### 2.2. Các tính năng của CakePHP

Nguồn mở, miễn phí, có cộng đồng sử dụng và hỗ trợ rộng lớn (trên website chính, trên kênh chat IRC, và diễn đàn những người yêu thích CakePHP), những người mới bắt đầu có thể tìm thấy rất nhiều tài liệu, project để tham khảo.

Tương thích PHP4 và PHP5 và trong tương lai là PHP6.

CakePHP được xây dựng theo mô hình MVC (Model – View – Controller). Với mô hình MVC, việc xử lý dữ liệu, giao tiếp CSDL, hiển thị dữ liệu được phân tách rõ ràng, giúp cho công tác lập trình cũng như bảo trì dễ dàng hơn.

Hỗ trợ tạo kết nối đến cơ sở dữ liệu một cách đơn giản và thuận lợi, chỉ với một chuỗi kết nối đặt trong file configs/database.php.

Sinh code tự động dựa trên cơ sở dữ liệu đã có: với việc sử dụng Cake Console được CakePHP hỗ trợ sẵn, chỉ mất vài phút để tạo ra các view, model, controller tương ứng với table đã có.

Kiểm tra và nhận biết tự động các ràng buộc, quan hệ trong cơ sở dữ liệu.

Phân quyền cho người dùng, nhóm người dùng (ACL) một cách uyển chuyển.

Cho phép tạo ra các URL rõ ràng, dễ hiểu.

Cung cấp khả năng bắt lỗi.

Cung cấp sẵn tiện ích xử lý dữ liệu (Data Sanitization).

Cung cấp khả năng lưu tạm (Caching).

Cung cấp khả năng lưu tạm (Caching).

Có khả năng triển khai trên hầu hết các máy chủ, hosting…hỗ trợ php và mysql hiện nay.

Tích hợp sẵn nhiều thư viện hỗ trợ cho việc lập trình và thiết kế giao diện trở nên đơn giản như: Ajax, HTML Form, Javascript…

Có sẵn các công cụ xử lý Email, Security, Session, Cookies, Request Handling.

Dễ dàng viết thêm thư viện hỗ trợ, liên kết với ứng dụng khác (thông qua vendors) ví dụ như liên kết với Zend Framework.

Hỗ trợ nhiều hệ Quản Trị CSDL (MySql, Oracle, PostgreSQL, DB2...).

Hỗ trợ xây dựng website đa ngôn ngữ.

Và một số tính năng khác… (việc sử dụng tính năng nào tùy thuộc vào từng dự án và sở thích của người lập trình).

### III. Một số khái niệm

### IV. Cài đặt/cấu hình môi trường phát triển

Download CakePHP tại địa chỉ: [*https://github.com/cakephp/cakephp/tags*](https://github.com/cakephp/cakephp/tags)

Download Xampp tại địa chỉ: [*https://www.apachefriends.org/download.html*](https://www.apachefriends.org/download.html)

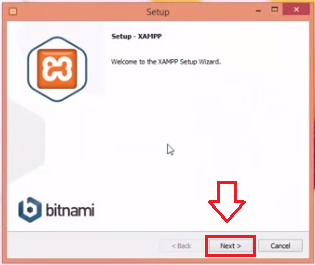
**Cài đặt:** Để sử dụng Cake PHP ta phải đáp ứng các nhu cầu sau: Apache Web Server - PHP Script Language - MySQL Database - phpMyAdmin Database Manager. Tuy nhiên tất cả các công cụ trên đã được tích hợp vào xampp.

#### 4.1. Cài đặt Xampp:

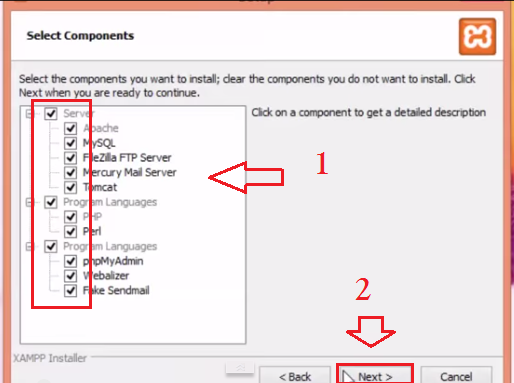
Sau khi download phiên bản Xampp phù hợp. Ta được file xampp.exe

Click phải lên file xampp.exe chọn **Run as administrator**.

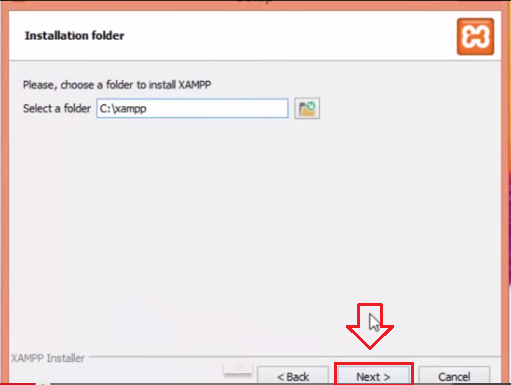
Chọn next để cài đặt.



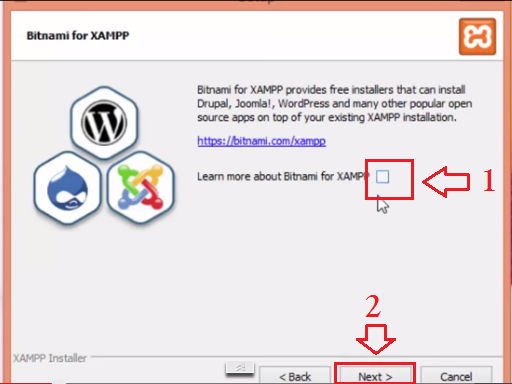
Kiểm tra xem đã check hết chưa nếu đã check hết rồi. Chọn Next để tiếp tục.



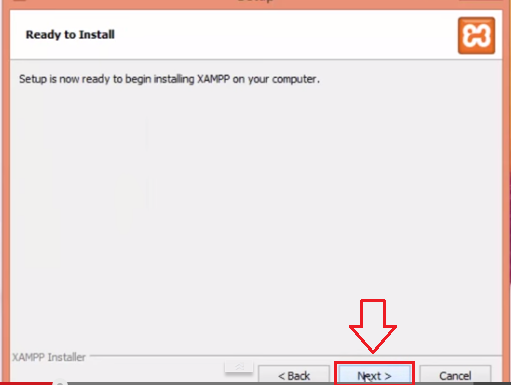
Chọn thư mục để cài đặt. Mặc định sẽ là ổ C. chọn Next để tiếp tục.



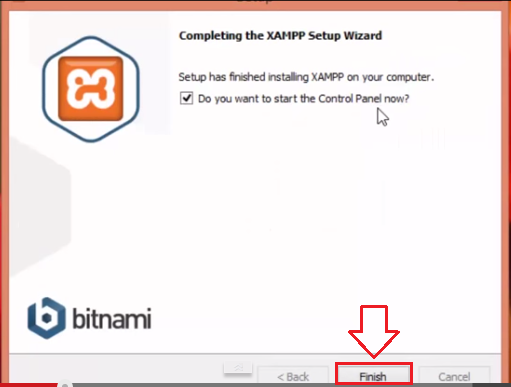
Bỏ chọn ở mục Learn more about Bitnami for Xampp. Chọn Next.



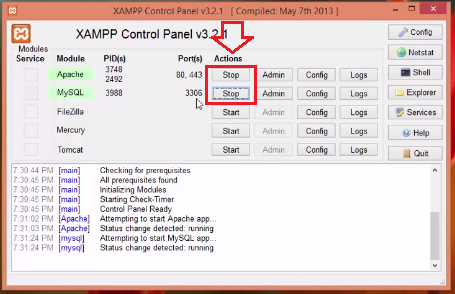
Chọn Next tiếp tục.



Đợi quá trình cài đặt và chọn Finish để kết thúc.



Sau khi cài đặt thành công. Chạy chương trình chọn Start để mở Apache và MySQL.

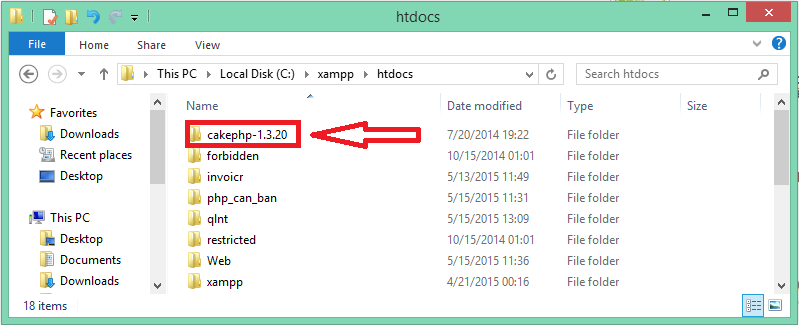


#### 4.2. Cài đặt Cake PHP

Giải nén, vào thư mục của web server.

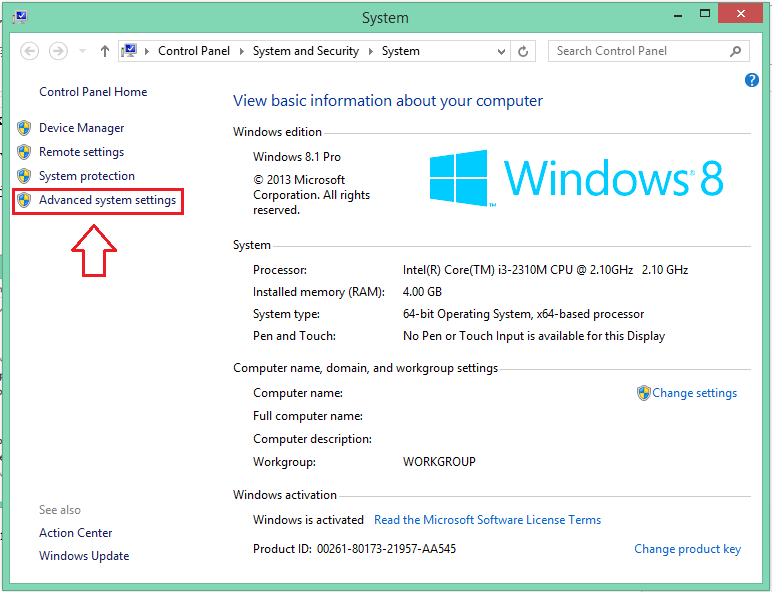
Trong ví dụ này thư mục webser gốc là: **C://xampp/htdocs**

Vậy ta có cấu trúc như sau: **C://xampp/htdocs/cakephp-1.3.20**

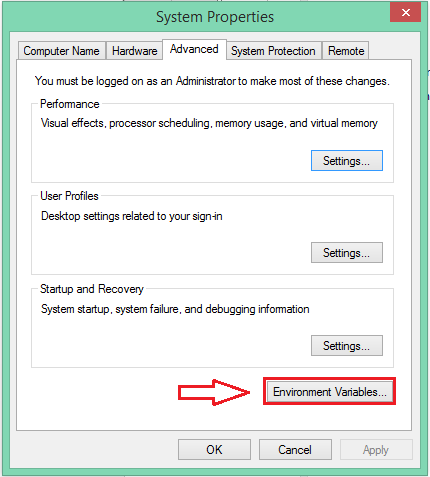


Sau khi giải nén xong. Click phải lên My computer (This PC) chọn Properties.

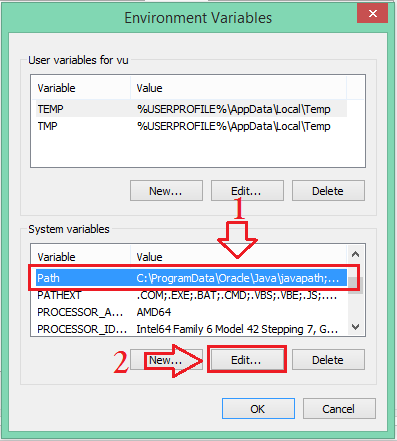
Chọn Advanced system settings.



Trong thẻ Advanced, chọn Environment Variables…

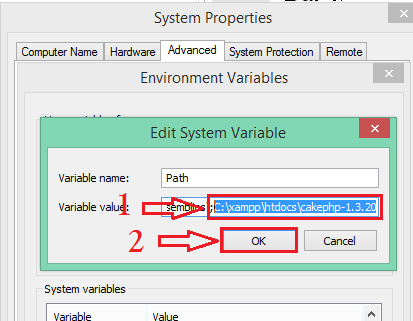


Chọn Path => chọn Edit…



Copy đường dẫn và past vào. Chọn OK.

* Cake php: *C:\xampp\htdocs\cakephp-1.3.20\cake\console\;*
* PHP: *C:\xampp\php\;*
* MySQL: *C:\xampp\mysql\bin\;* (Cái này để chạy các câu lệnh SQL trực tiếp trên Console, có thể bỏ qua phần cái đặt này)

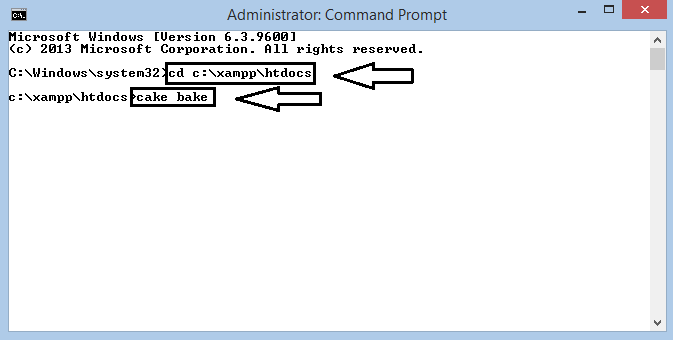


#### 4.3. Test cake PHP:

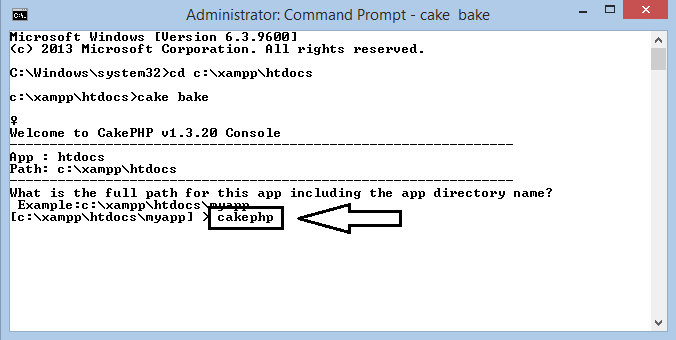
Mở Command prompt.

Gõ: *cd C:\xampp\htdocs => enter*

Gõ: *cake bake => enter*



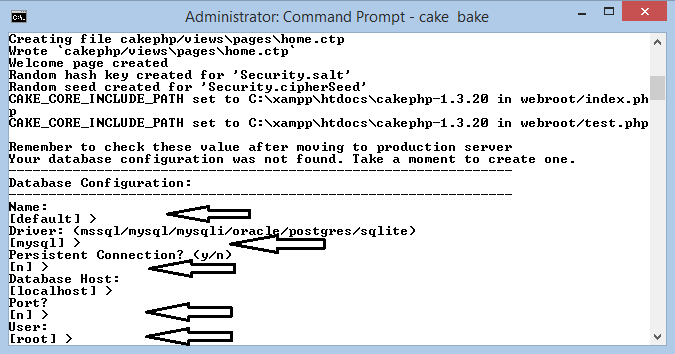
Gõ: tên thư mục hoặc tên dự án, website muốn tạo, trong ví dụ này ta gõ *cakephp*



Gõ: *enter* để chọn mặc định ở các bước này.



Phần User: nếu ta có tài khoảng của phpmyadmin thì gõ vào. Nếu không thì gõ e*nter* để tiếp tục (tài khoảng mặc định là Root).



Nhập Password: nếu không thì gõ *enter* để tiếp tục.

Gõ: *y => enter* (để sử dụng password rỗng). Lưu ý: Cần sử dụng chính xác user và password để có thể thực hiện được tất cả các thao tác trên cơ sở dữ liệu (nếu không phải chủ sở hữu hoặc người có đầy đủ quyền trên cơ sở dữ liệu sẽ không thể thực hiện câu truy vấn, hoặc tạo mới,…)

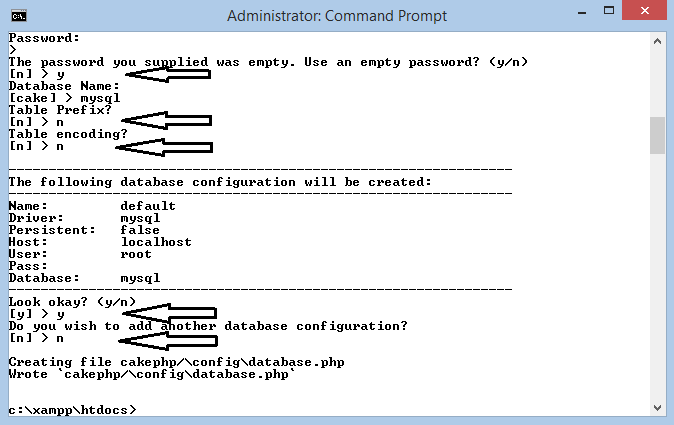
Database: gõ tên database trên phpmyadmin, trong ví dụ này là *mysql.* Lưu ý: các quyền của user phía trên với database để có thể thực hiện các thao tác.

Gõ: *n => enter*

Gõ: *n => enter*

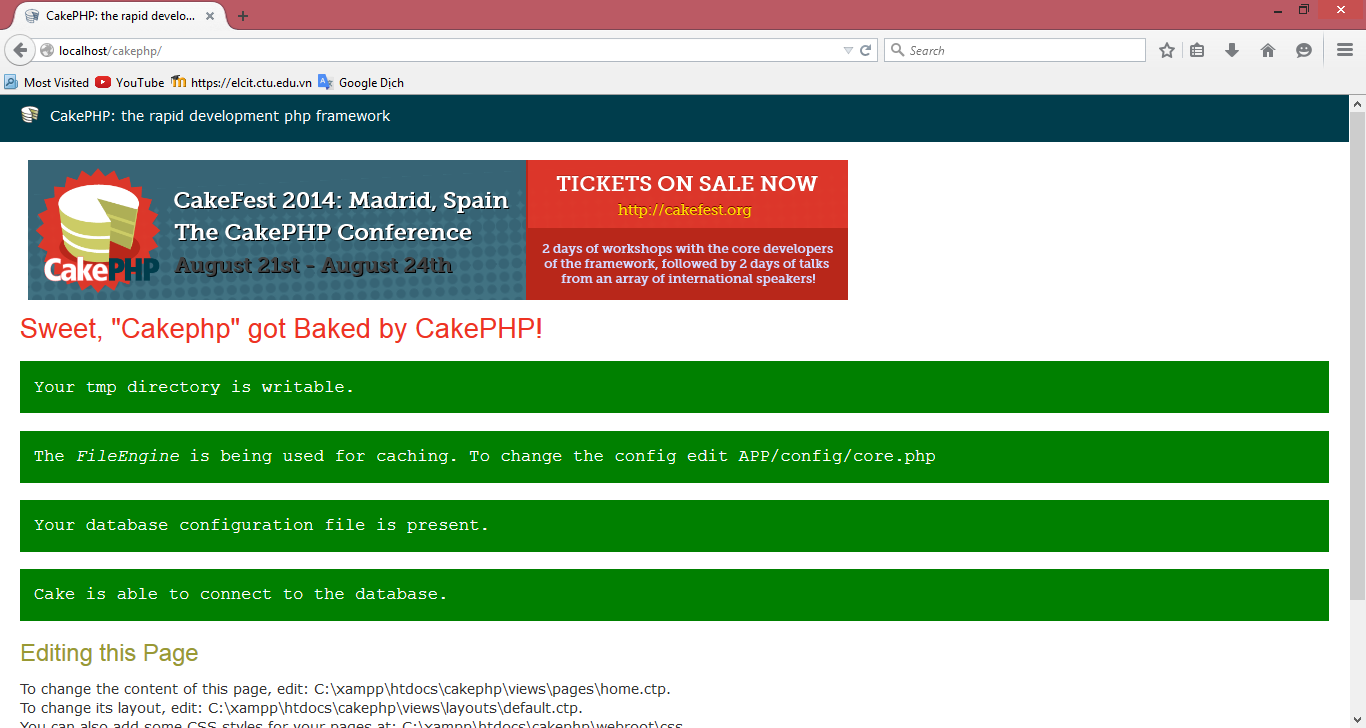
Gõ: *y =>enter*

Gõ: *n => enter* và kết thúc quá trình cài đặt Cake PHP.



Đến đây thì ta đã hoàn thành quá trình cài đặt Xampp và Cake PHP.

Mở trình duyệt web: http://localhost/cakephp/ để kiểm tra.



### V. Các bước lập trình ứng dụng, triển khai

## BÀI 2. CAKEPHP – CẤU TRÚC VÀ CÁC THÀNH PHẦN CỦA FRAMEWORK CAKEPHP

**Nội dung:**

1. Giới thiệu các cấu trúc MVC trong CakePHP.
2. Quy tắc thiết kế cơ sở dữ liệu, cách đặt tên trong CakePHP.
3. Tự động sinh code.

### I. Giới thiệu các cấu trúc MVC trong CakePHP

#### 1.1. Giới thiệu cấu trúc MVC trong CakePHP

Cũng như đa số các PHP framework khác CakePHP được xây dựng trên mô hình MVC (Model - Controller - View).

Cấu trúc folder của CakePHP:

root/app/

/Config/

/Console/

/Controller/

/Lib/

/Locate/

/Model/

/Plugin/

/Test/

/tmp/

/Vendor/

/View/

/webroot/

  /lib/

/vendors/

/plugins/

/.htaccess

/index.php

/README

Nhìn vào cấu trúc folder ở trên bạn sẽ thấy tất cả ba thành phần quan trọng này chính là ba folder Controller, Model và View trong thư mục app. Vậy ba thành phần này có mối liên hệ như thế nào? Để trả lời câu hỏi này thì chúng ta phải đi tìm hiểu từng phần một ở dưới đây:

##### 1.1.1. Model

Model là một trong ba phần quan trọng của Web Application MVC và nhiệm vụ của nó là thao tác trực tiếp với database. Dưới đây là các nhiệm vụ của nó:

* Xử lý, kiểm tra dữ liệu
* Lưu , xóa dữ liệu
* Tạo các mối quan hệ giữa Model này và Model khác ( hasMany, belongTo, hasOne)

##### 1.1.2. View

View chính là nơi tiếp nhận dữ liệu từ Controller và nó sẽ hiển thị thành một layout Web bình thường. Ví dụ khi bạn vào một website bất kì thì những gì bạn thấy đó chính là thể hiện của View. Tuy nhiên ta có thể hiển thị nó ở nhiều định dạng khác nhau thông qua Header trong PHP:

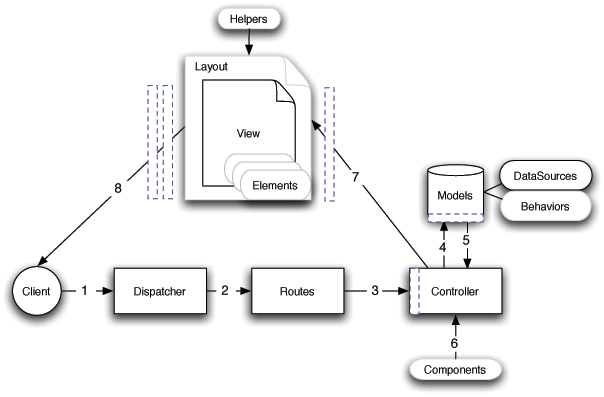
* HTML
* XML
* Media( video, music, image)
* Stream( download file )

##### 1.1.3. Controller

Controller là nơi nhận và xử lý các request từ người dùng, sau đó trả về dữ liệu cho người dùng qua View. Thông thường việc xử lý các requests đều phải thông qua Model nên Controller như "cầu nối" giữa Model và View. Ngoài ra Controller còn có 1 số nhiệm vụ sau

* Kiểm tra authentication hoặc authorize
* Xác định định dạng để xuất ra View
* Truy xuất và xử lý data từ Model
* Kiểm tra dữ liệu nhận về từ client thông qua request

**Sau đây là mô hình request và response trong CakePHP**:



Hình 2. . mô hình request và response trong CakePHP

1. Trước tiên, Dispatcher nhận request của người dùng  sau đó dựa vào Routes để xác định được Controller nào cần gọi để xử lý request đó (bước 1 và bước 2)
2. Tại Controller, request của người dùng được xử lý. Những xử lý này có thể được thông qua Components như Authenticate, Session, Cookie ... Sau đó Controller sẽ response dữ liệu ra View. Những dữ liệu được response ra View có thể được truy xuất từ Model (bước 4, 5, 6, 7)
3. Sau đó Views sẽ truyền dữ liệu nhận được từ Controller đến người dùng thông qua browser (xem bằng HTML hoặc cho download về)
4. Ở đây ta thấy ở Model có Behaviours, DataSources, View thì có Layout, Helpers, Elements,  đây là những thành phần chỗ trợ cho việc thao tác database (Model) hoặc xuất dữ liệu ra View
   * **Model** :
     + *Behaviours* : hỗ trợ những callback functions, có tác dụng trong việc truy xuất, xử lý và lưu trữ dữ liệu ( ví dụ : afterFind, afterSave, beforeSave ...
     + *Datasources*: hỗ trợ kết nối nhiều database trong 1 applications.
   * **View** :
     + *Layout*: dùng để định nghĩa layout chung cho loại đối tượng sử dụng.
     + *Elements*: dùng để viết những  code HTML dùng chung

*Helpers*: Tích hợp nhiều phương thức, giúp việc xử lý HTML đơn giản hơn.  
Ví dụ: FormHelper, HtmlHelper, PaginationHelper

#### 1.2. Ví dụ

Các bạn xem thêm phần ví dụ cụ thể dưới đây để hiểu rõ hơn về cấu trúc MVC:

* Mô hình bình thường:

<?php   
$connect = mysql\_connect('myserver', 'mylogin', 'mypassword');   
mysql\_select\_db('myDB');   
if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {   
    $news\_id = $\_POST['news\_id'];   
    mysql\_query("INSERT INTO commentaires SET news\_id='$news\_id',   
           auteur='".mysql\_escape\_string($\_POST['auteur'])."',   
                        texte='".mysql\_escape\_string($\_POST['texte'])."',   
                        date=NOW()"   
               );   
    header("location: ".$\_SERVER['PHP\_SELF']."?news\_id=$news\_id");   
    exit;   
} else {   
    $news\_id = $\_GET['news\_id'];   
}   
?>   
<html>   
    <head>   
        <title>Les news</title>   
    </head>   
    <body>   
    <h1>Les news</h1>   
    <div id="news">   
        <?php   
        $news\_req = mysql\_query("SELECT \* FROM news WHERE id='$news\_id'");   
        $news = mysql\_fetch\_array($news\_req);   
        ?>   
    <h2><?php echo $news['titre'] ?> postée le <?php echo $news['date'] ?></h2>   
    <p><?php echo $news['texte\_nouvelle'] ?> </p>   
        <?php   
        $comment\_req = mysql\_query("SELECT \* FROM commentaires WHERE news\_id='$news\_id'");   
        $nbre\_comment = mysql\_num\_rows($comment\_req);   
        ?>   
    <h3><?php echo $nbre\_comment ?> commentaires relatifs à cette nouvelle</h3>   
        <?php while ($comment = mysql\_fetch\_array($comment\_req)) {?>   
            <h3><?php echo $comment['auteur'] ?> a écrit le <?php echo $comment['date'] ?></h3>   
            <p><?php echo $comment['texte'] ?></p>   
        <?php } ?>   
    <form method="POST" action="<?php echo $\_SERVER['PHP\_SELF'] ?>" name="ajoutcomment">   
        <input type="hidden" name="news\_id" value="<?php echo $news\_id?>">   
        <input type="text" name="auteur" value="Votre nom"><br />   
        <textarea name="texte" rows="5" cols="10">Saisissez votre commentaire</textarea><br />   
        <input type="submit" name="submit" value="Envoyer">   
    </form>   
    </div>   
    </body>   
</html>

* Mô hình MVC (code trên sẽ được chia làm 3 file khác nhau)
  + mymodel.php (xử lý thông tin trong database)

<?php   
function dbconnect()   
{   
static $connect = null;   
if ($connect === null) {   
    $connect = mysql\_connect('myserver', 'mylogin', 'mypassword');   
    mysql\_select\_db('myDB');   
}   
return $connect;   
}   
  
function get\_news($id)   
{   
$news\_req = mysql\_query("SELECT \* FROM news WHERE id='$news\_id'",dbconnect());   
return mysql\_fetch\_array($news\_req);   
}   
  
function get\_comment($news\_id)   
{   
$comment\_req = mysql\_query("SELECT \* FROM commentaires WHERE news\_id='$news\_id'",dbconnect());   
$result = array();   
while ($comment = mysql\_fetch\_array($comment\_req)) {   
    $result[] = $comment;   
}   
return $result;   
}   
  
function insert\_comment($comment)   
{   
mysql\_query("INSERT INTO commentaires SET news\_id='{$comment['news\_id']}',   
                                          auteur='".mysql\_real\_escape\_string($comment['auteur'])."',   
                                          texte='".mysql\_real\_escape\_string($comment['texte'])."',   
                                          date=NOW()"   
            ,dbconnect() );   
}

* + myview.php (chứa phần html hiển thị)

<html>

<head>

<title>Les news</title>

</head>

<body>

<h1>Les news</h1>

<div id="news">

<h2><?php echo $news['titre'] ?> postée le <?php echo $news['date'] ?></h2>

<p><?php echo $news['texte\_nouvelle'] ?> </p>

<h3><?php echo $nbre\_comment ?> commentaires relatifs à cette nouvelle</h3>

<?php foreach ($comments AS $comment) {?>

<h3><?php echo $comment['auteur'] ?> a écrit le <?php echo $comment['date'] ?></h3>

<p><?php echo $comment['texte'] ?></p>

<?php } ?>

<form method="POST" action="<?php echo $\_SERVER['PHP\_SELF'] ?>" name="ajoutcomment">

<input type="hidden" name="news\_id" value="<?php echo $news\_id?>">

<input type="text" name="auteur" value="Votre nom"><br />

<textarea name="texte" rows="5" cols="10">Saisissez votre commentaire</textarea><br />

<input type="submit" name="submit" value="Envoyer">

</form>

</div>

</body>

</html>

* + mycontroller.php (để xử lý vấn đề)

<?php   
require ('mymodel.php');   
if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {   
    insert\_comment($\_POST);   
    header("HTTP/1.1 301 Moved Permanently");    
    header("location: {$\_SERVER['PHP\_SELF']}?news\_id={$\_POST['news\_id']}");   
    exit;   
} else {   
    $news = get\_news($\_GET['news\_id']);   
    $comments = get\_comments($\_GET['news\_id']);   
    require ('myview.php');   
}   
?>

### II. Quy tắc thiết kế cơ sở dữ liệu, cách đặt tên trong CakePHP

Khi sử dụng CakePHP, bạn phải tuân thủ theo một số qui ước mà framework này yêu cầu: cách đặt tên file, tên lớp, các qui ước khi dùng view, model, controller.

#### 2.1. Cách đặt tên lớp (Class) và tên file

Tên file sử dụng chữ thường, dùng dấu “\_” để phân cách các từ.

Tên lớp dùng CamelCased (viết hoa chữ cái đầu tiên của từ).

Ví dụ: ta có tên lớp là **MyClass** thì tương ứng tên file phải là **my\_class.php**

Dưới đây là các ví dụ về cách đặt tên file cho mỗi dạng lớp khác nhau mà bạn thường sử dụng trong ứng dụng CakePHP:

* + Một controller có tên lớp là **NewsCategoriesController** thì tên file tương ứng là **news\_categories.php.**
  + Một component có tên lớp là **SmtpMailComponent** thì tên file là **smtp\_mail.php.**
  + Một model có tên lớp là **Product** thì tên file là **product.php.**
  + Một behavior có tên lớp là **UploadBehavior** thì tên file là **upload.php.**
  + Một view có tên lớp là **MySimpleView** thì tên file là **my\_simple.php.**
  + Một helper có tên lớp là **MyImageHelper** thì tên file là **my\_image.php.**

#### 2.2. Các quy ước về Model và cơ sở dữ liệu

Tên các lớp model được đặt ở dạng số ít và việc viết hoa thường tuân theo CamelCased. Ví dụ: **Person**, **BigPerson** và **ReallyBigPerson**.

Tên bảng tương ứng với các model của CakePHP ở dạng số nhiều và sử dụng gạch dưới (\_). Tên bảng tương ứng với các model ở trên là **people**, **big\_people**, và r**eally\_big\_people**. Tuy nhiên, bạn vẫn có thể đặt tên bảng khác đi và cho CakePHP biết điều này thông qua biến **$useTable** trong model. Bạn có thể dùng thư viện **Inflector** có sẵn trong CakePHP để kiểm tra dạng số ít/số nhiều của các từ. Các khóa ngoại trong các quan hệ hasMany, belongsTo, hasOne có tên mặc định là số ít của tên bảng kèm theo **\_id** ở cuối cùng. Ví dụ với quan hệ NewsCategory có nhiều News, bảng news sẽ có khóa ngoại đến bảng news\_categories là **news\_category\_id**.

Các bảng liên kết được sử dụng trong quan hệ hasAnhBelongsToMany (viết tắt là HABTM – quan hệ nhiều nhiều) được đặt tên bằng cách kết nối các tên model theo thứ tự abc, vì vậy tên đúng sẽ là **apples\_zebras** chứ không phải là **zebras\_apples**.

Tất các các bảng của model (không bao gồm bảng kết nối), bắt buộc có một khóa chính. CakePHP không hỗ trợ khóa chính gồm nhiều cột. Nếu bạn muốn thao tác trực tiếp kết nối (join) các bảng, hãy truy vấn trực tiếp hoặc thêm một khóa chính thay cho khóa nhiều cột. Ví dụ:

CREATE TABLE posts\_tags (  
id INT(10) NOT NULL AUTO\_INCREMENT,  
post\_id INT(10) NOT NULL,  
tag\_id INT(10) NOT NULL,  
PRIMARY KEY(id));

Lẽ ra trong table post\_tags, bình thường thì 2 cột **post\_id** và **tag\_id** ta có thể cho nó làm khóa chính nhưng CakePHP không chấp nhận điều này, do đó ta phải thêm một khóa khác có tên là **id.** Khóa chính mặc định trong CakePHP là cột id, kiểu int, tự tăng. Tuy nhiên, bạn cũng có thể định nghĩa lại khóa chính cho bảng thông qua biến **$primaryKey** trong model.

Thay cho việc sử dụng kiểu **int**, auto-increment làm khóa chính, bạn có thể sự dụng char(36) hoặc binary(36), khi đó CakePHP sẽ tự động tạo ra UUIDs để làm khóa khi ta thêm mới một record. UUID là một chuỗi gồm 32 byte, tương ứng với 36 ký tự, ví dụ: 550e8400-e29b-41d4-a716-446655440000.

#### 2.3. Quy ước về Controller

Tên lớp của controller đặt theo dạng số nhiều, tuân theo CamelCased và cuối tên phải có chữ “Controller”. Ví dụ: **ProductsController**, **NewsController.**

Phương thức mặc định trong controller là index(). Nếu bạn gọi một controller mà không có một action cụ thể nào, index() sẽ được gọi. Ví dụ: bạn gõ http://abc.com/products/ cũng giống như http://abc.com/products/index.

Khi một phương thức trong controller được đặt tên với dấu “**\_**” ở phía trước, bạn sẽ không thể truy xuất nó từ bên ngoài mà chỉ có thể truy xuất trong controller đó (cái này gọi là private). Ví dụ: bạn viết một method có tên là **\_get\_list\_news()** trong NewsController thì bạn sẽ chỉ có thể truy xuất **$this->\_get\_list\_news()** trong NewsController mà không thể gọi **http://abc.com/news/\_get\_list\_news**, sẽ báo lỗi ngay!

#### 2.4. Quy ước về View

Tên file của view được đặt tương ứng với phương thức trong controller và để trong thư mục theo tên file controller (không kèm theo chữ “controller”). Ví dụ: trong controller NewsController bạn có phương thức view\_all() thì tên file view là view\_all.ctp, đặt trong thư mục app/views/news/

Mặc định, nếu trong controller NewsController có method view(), khi gọi action view trên trình duyệt, controller sẽ tìm file view.ctp trong app/views/news/view.ctp, tuy nhiên, bạn có thể gọi một file view khác không phải là view.ctp bằng lệnh $this->render(‘ten\_view’);

#### 2.5. Ví dụ

Tên bảng trong CSDL: products.

Tên lớp của model tương ứng: Product, file nằm tại app/models/product.php.

Tên lớp của controller tương ứng: ProductsController, file nằm tại app/controllers/products\_controller.php.

View nằm tại app/views/products/index.ctp.

Khi gõ **http://abc.com/products/** trên trình duyệt, CakePHP sẽ thực thi hàm **index()** trong **ProductsController** (vì hàm index() là hàm mặc định), khi đó model **Product** sẽ được dùng để lấy dữ liệu trong bảng **products**, đổ ra file **index.ctp** và hiển thị lên trình duyệt.

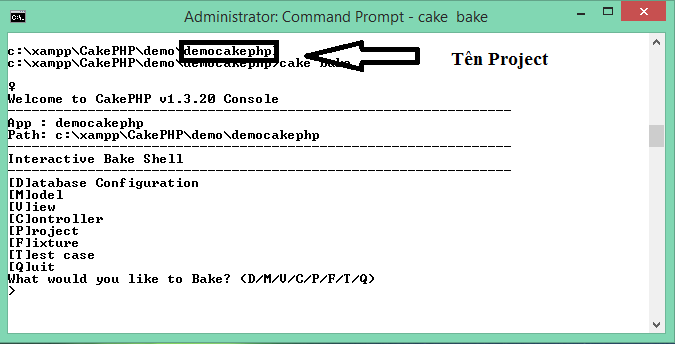
### III. Tự động sinh code

Sử dụng cake console để sinh code tự động trong cake PHP. Cake consonle là một giao diện dòng lệnh giúp ta tạo các project bằng cake php một cách nhanh chóng, mà ta không cần phải coppy 1 thư viện cho các project khác nhau. Ngoài ra ta cũng có thể tạo các model, controller, view từ cake console tuy nhiên ta phải tạo CSDL trước.

Ta sẽ thiết lập môi trường cho cakephp và cake console. Lưu ý sinh code tự động bàng cake console ta phải thực hiện theo từng bước tạo Model , Controller, View.

**Bước 1:** Tạo project để chứa các model, controller, view

* Mở CMD và trỏ tới thư mục project mới tạo.

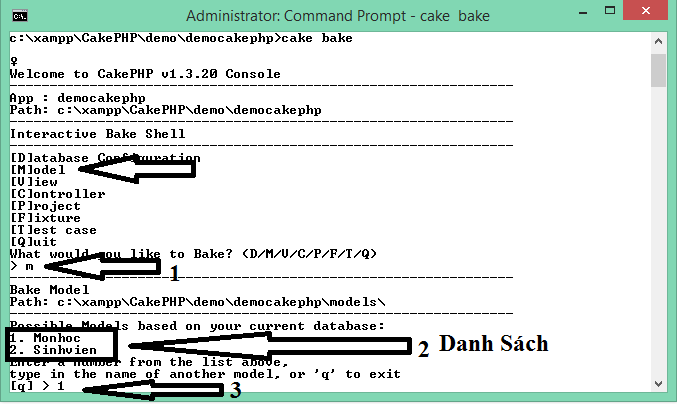


* Gõ cake bake => hiển thị danh sách chúng ta có thể bake được (database, model, view,…).

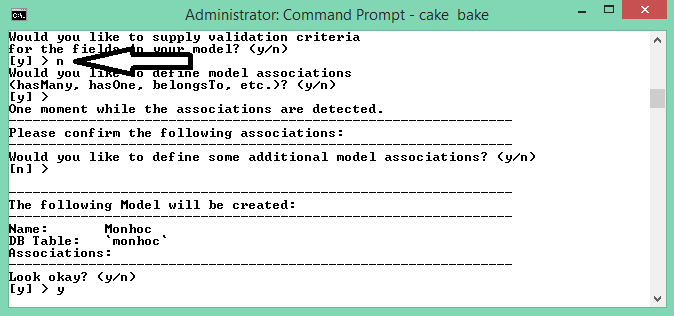


**Bước 2:** Tạo model

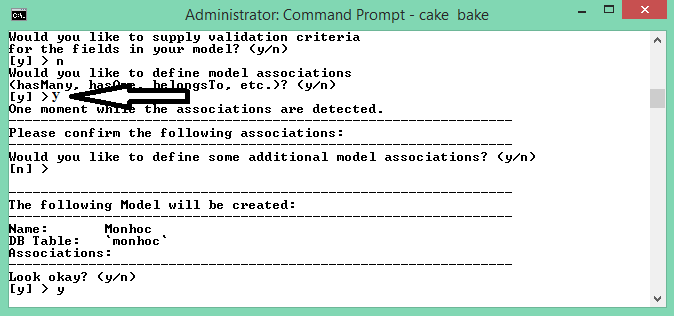
* Gõ m => enter… hiển thị danh sách các model mà ta có thể tạo trong database. Muốn tạo model nào thi gõ số thứ tự của model đó.



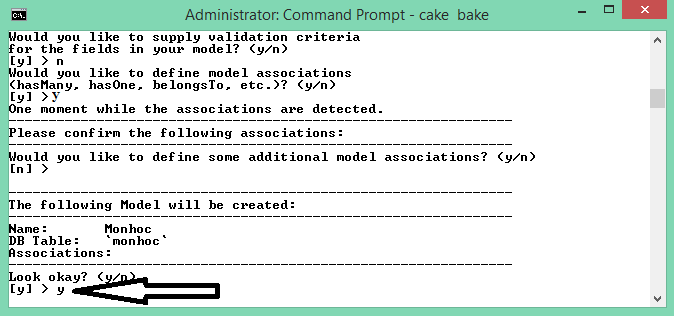
* Chọn ràng buộc dữ liệu chọn n => enter để bỏ qua.



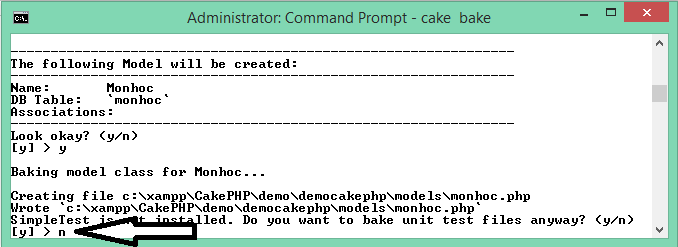
* Chọn mối quan hệ các bảng y => enter.



* Chọn y => enter.

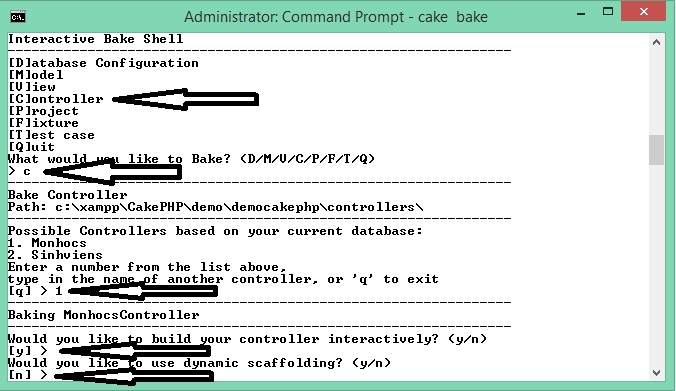


* Tạo file để test chọn n => enter.

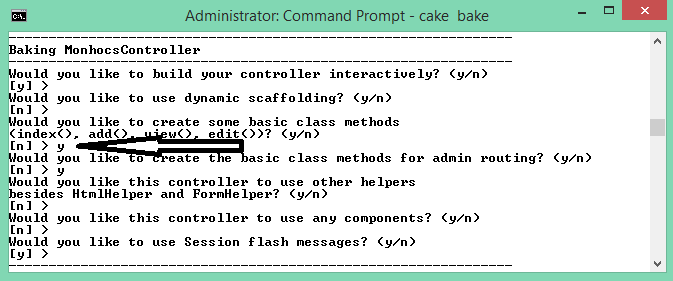


**Bước 3**: Tạo controler

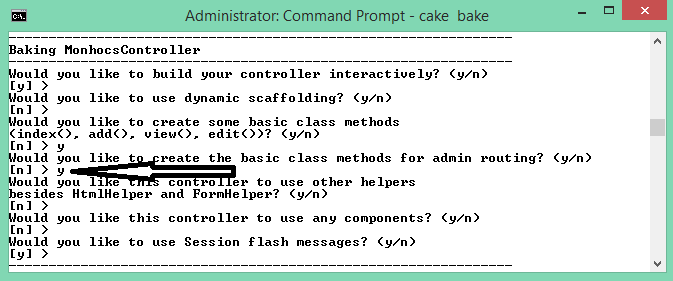
* Gõ c => enter. Và gõ Enter các phần còn lại.



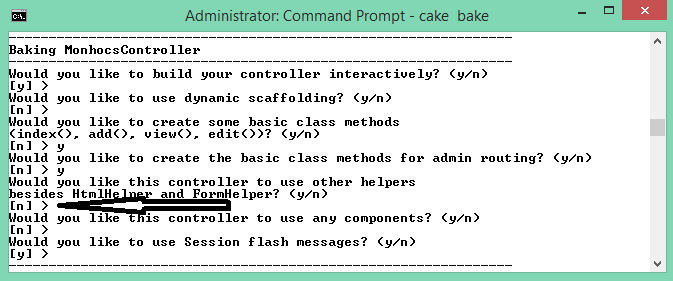
* Đến phần tạo các phương thức cơ bản (index (), add(),view(),…) chọn y => enter.



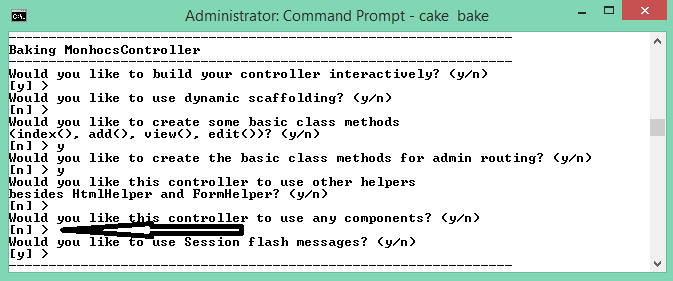
* Tạo các phương thức cho quyền Admin y => enter.



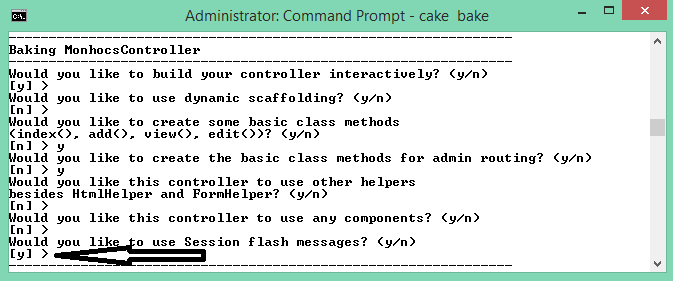
* Hỏi có dùng các heper nào khác không n => enter.



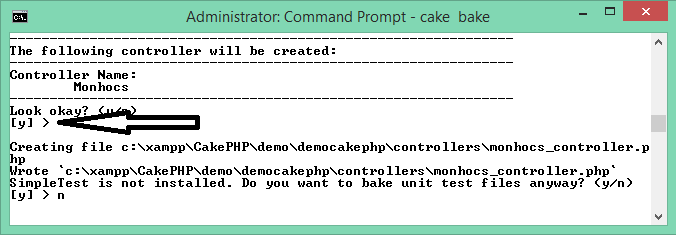
* Hỏi có dùng các component nào khác không chọn n => enter.



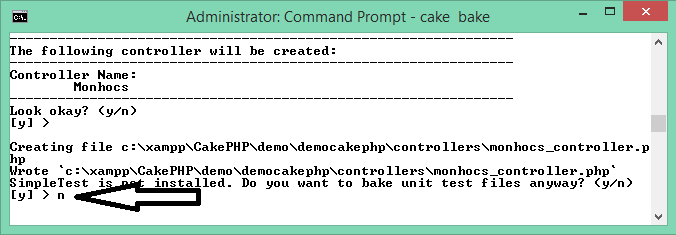
* Chọn các session chọn y=>enter.



* Chọn y => enter.

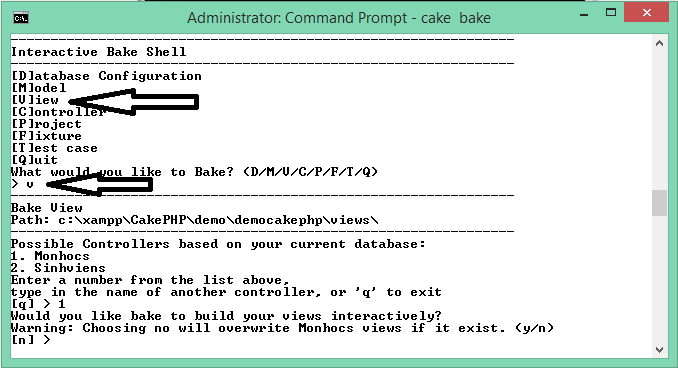


* Tạo file test chọn n => enter.

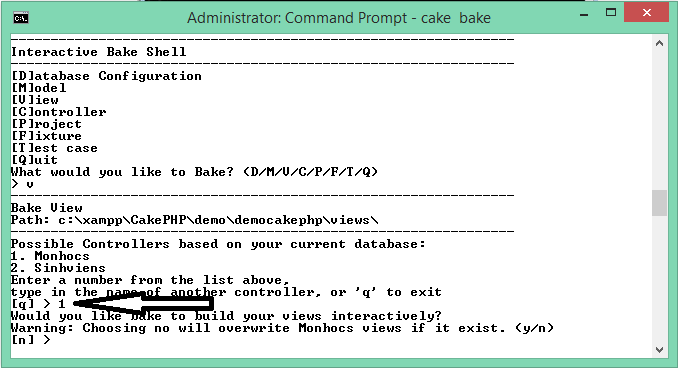


**Bước 4**: Tạo view

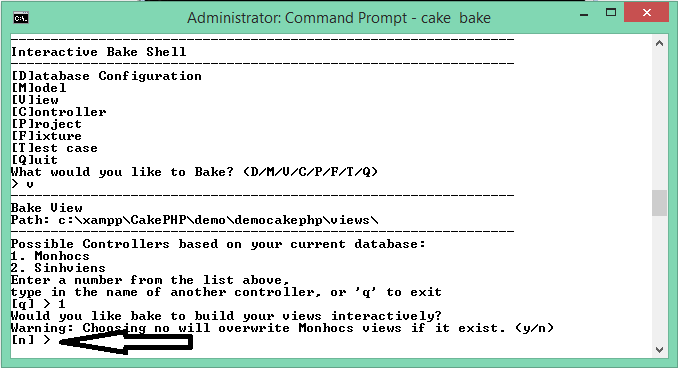
* Gõ v => enter.



* Tương tự như trên muốn tạo view nào thì ta gõ số thứ tự của view đó.



* Tùy chọn tiếp theo sẽ ghi đè lên các view đã có sẵn gõ để mặc định gõ Enter.



## BÀI 3. THIẾT KẾ MODEL

**Nội dung:**

1. Định nghĩa các khóa, quan hệ giữa các bảng.
2. Kiểm tra dữ liệu (Validation).
3. Viết behaviors cho Model.

### I. Định nghĩa các khóa, quan hệ giữa các bảng

#### 1.1. Định nghĩa các khóa

**Khóa (key):** Một hoặc một tập thuộc tính dùng để nhận diện một bản ghi/bộ. Các loại khóa:

* **Khóa chính:** Là khóa được sử dụng để nhận diện một bản ghi/bộ.
* **Khóa ngoại:** Là một thuộc tính/trường trong một bảng mà là khóa chính trong một bảng khác.
* **Siêu khóa:** Là tập hợp các thuộc tính dùng để xác định tính duy nhất trong quan hệ bất kỳ.

#### 1.2. Quan hệ giữa các bảng

Quan hệ hay còn gọi là liên kết: là sự kết hợp giữa hai hay nhiều bảng phản ánh sự ràng buộc trong quản lý. Đặc biệt: Một bảng có thể liên kết với chính nó ta thường gọi là tự liên kết. Giữa hai bảng có thể có nhiều hơn một liên kết.

Có ba kiểu quan hệ: một - một, một - nhiều, nhiều - nhiều.

* **Quan hệ một – một (1-1):**

Mỗi đối tượng dữ liệu của bảng A quan hệ với một đối tượng dữ liệu của bảng B và ngược lại.

Kí hiệu:

Ví dụ: Hai bảng NhanVien và Luong có quan hệ 1-1 vì mỗi Nhân viên có một mức lương cơ bản và một bảng lương là của một Nhân Viên.

|  |  |
| --- | --- |
| **NhanVien** | |
| **MaNV** | **Hoten** |
| 1 | Nguyễn Văn A |
| 2 | Nguyễn Thị B |
| 3 | Đặng Thanh Ngọc |

|  |  |  |
| --- | --- | --- |
| **Luong** | | |
| **MNV** | **NgayHD** | **LuongCB** |
| 1 | 2/2/2012 | 2,500,000 |
| 2 | 3/32013 | 2,600,000 |
| 3 | 4/4/2014 | 2,700,000 |

NhanVien

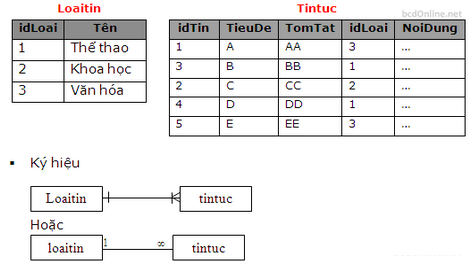
Luong

* **Liên kết một – nhiều (1-N):**

Mỗi đối tượng dữ liệu của bảng A quan hệ với nhiều đối tượng dữ liệu của bảng B. Ngược lại mỗi đối tượng dữ liệu của bảng B quan hệ với chỉ một đối tượng dữ liệu của bảng A.

Kí hiệu:

Ví dụ: Hai bảng loại tin và tin có quan hệ một - nhiều vì mỗi loại tin có một hoặc nhiều tin và mỗi tin là ở trong một loại nào đó.

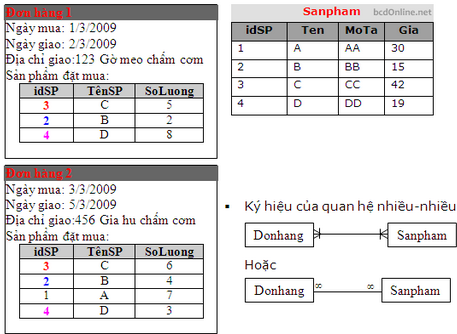


* **Liên kết nhiều – nhiều (N-N):**

Mỗi đối tượng dữ liệu của bảng A quan hệ với nhiều đối tượng dữ liệu của bảng B. Ngược lại mỗi đối tượng dữ liệu của bảng B quan hệ với nhiều đối tượng dữ liệu của bảng A.

Kí kiệu:

Ví dụ: Hai bảng donhang và sanpham có quan hệ nhiều - nhiều vì mỗi đơn hàng có mua nhiều sản phẩm và một sản phẩm được mua trong nhiều đơn hàng.



* **Chú ý:**

- Mô hình dữ liệu quan hệ không chỉ là công cụ phân tích thiết kế mà còn như một phương pháp kiểm tra chặt chẽ các yêu cầu nghiệp vụ của người sử dụng.

- Nếu hai bảng có quan hệ một - một thường có ít lý do để coi chúng như hai bảng tách biệt nên người ta thường gộp hai bảng làm một.

- Nếu hai bảng có quan hệ nhiều - nhiều thì không có sự khác biệt về bản chất giữa các chiều vì vậy ít khi được sử dụng.

* **Tóm lại:** Trong ba kiểu liên kết trên, liên kết một nhiều là quan trọng hơn cả và hầu như các mối quan hệ trong mô hình liên kết đều là một nhiều.

### II. Kiểm tra dữ liệu (Validation)

Cú pháp cho một Validation Rules

*var*  $validate = *array*( 'fieldName' => 'ruleName');

fieldName: là tên các trường cần kiểm tra.

ruleName: là tên luật được định nghĩa trước ví dụ: “alphaNumeric”, “email”,”username”.

#### 2.1. alphaNumeric

Kiêm tra dữ liệu nhập vào các trường là chữ cái và số.

*var* $validate = *array*(

'login' => *array*( //”login”: tên trường

'rule' => 'alphaNumeric',

'message' => 'Tên người dùng phải chứa chữ cái và số !' //dòng thông báo lỗi !

)

);

#### 2.2. Between

Kiểm tra giới hạn cho dữ liệu nhập vào. Giá trị min và max cần được khai báo.

*var* $validate = *array*(

'password' => *array*(

'rule' => *array*('between', 5, 15), //giới hạn giá trị độ dài của password. min=5, max=15

'message'=>'mật khẩu có chiều dài từ 5 đến 15 ký tự !'

)

);

#### 2.3. Blank

Khai báo ký tự khoảng trắng. Bao gồm: “space”, “tab”, “carriage return” và “newline”. Nếu là khoảng trắng thì một là tạo mới “create” hoặc cập nhật lại “update”.

*var* $validate = *array*(

'id' => *array*(

'rule' => 'blank',

'on' => 'create' // để tạo mới hoặc cập nhật lại dữ liệu

)

);

#### 2.4. Boolean

Kiểm tra giá trị true hoặc false. Hay số 0 hoặc 1, chuỗi là “0” hoặc “1”.

*var* $validate = *array*(

'myCheckbox' => *array*(

'rule' => *array*('boolean'),

'message' => 'Giá trị không chính xác cho myCheckbox'

)

);

#### 2.5. Cc (credit card)

Kiểm tra dữ liệu thẻ card có hợp lệ hay không. Dựa vào ba thông số: ‘type’, ‘deep’ and ‘regex’.

Type từ khóa dùng cho các thẻ card. Thiết lập ở các cấp độ khác nhau “fast”,”all” và “any”

* Amex
* Bankcard
* Diners
* Disc
* Electron
* Enroute
* Jcb
* Maestro
* Mc
* Solo
* Switch
* Visa

Nếu “type” ở cấp độ “fast” xác nhận dữ liệu và đánh số với số lượng lớn các thẻ card. Nếu “type” ở cấp độ “all” kiểm tra tất cả các thẻ card. Cũng có thể thiết lập “type” là một mảng.

Các “deep” nên đặt là giá trị true hoặc false. Nếu là true thì thực hiện kiểm tra các thuật toán (Luhn). Mặc định là false.

Các “regex” cung cấp các ký hiệu riêng để nhận dạng chủ thẻ card (ví dụ:phương pháp sinh trắng học).

*var* $validate = *array*(

'ccnumber' => *array*(

// kiểm tra thẻ “visa”, “maestro”.nếu giá trị nhập sai hoặc rỗng thì thông báo lỗi

'rule' => *array*('cc', *array*('visa', 'maestro'), *false*, *null*),

'message' => 'số thẻ bạn cung cấp không hợp lệ!'

)

);

#### 2.6. Comparison

Dùng để so sánh giá trị là số. Hỗ trợ các phép so sánh “>”,”<”,”>=”,”<=”,”==”,”!=”.

*var* $validate = *array*(

'wage' => *array*(

'rule' => *array*('comparison', '>=', 4500000), // so sánh mức lương

'message' => 'Lương cơ bản phải lớn 4,500,000 !'

)

);

#### 2.7. Date

Kiểm tra dữ liệu ngày. Mặc định là “ymd”

* “dmy” ví dụ: 27-12-2006 hoặc 27-12-06 (phân cách có thể là khoảng trắng”space”,dấu gạch ngang”-”, dấu gạch chéo”/”).
* “mdy” ví dụ: 12-27-2006 hoặc 12-27-06 (phân cách có thể là khoảng trắng”space”,dấu gạch ngang”-”, dấu gạch chéo”/”)
* “ymd” ví dụ: 2006-12-27 hoặc 06-12-27 (phân cách có thể là khoảng trắng”space”,dấu gạch ngang”-”, dấu gạch chéo”/”)
* “dMy” ví dụ: 27 December 2006 hoặc 27 Dec 2006
* “Mdy” ví dụ: December 27,2006 hoặc Dec 27,2006
* “My” ví dụ: December 2006 hoặc Dec 2006
* “my” ví dụ: 12/2006 hoặc 12/06.

*var* $validate = *array*(

'born' => *array*(

'rule' => *array*('date','ymd'),

'message' => 'Nhập ngày theo định dạng YY-MM-DD ',

'allowEmpty' => *true*

)

);

// kiểm tra năm sinh

*var* $validate = *array*(

'dateOfBirth' => *array*(

'rule' => *array*('comparison', '<=', 1997),

'message' => 'Năm sinh phải nhỏ hơn năm 1997.'

)

);

#### 2.8. Decimal

Kiểm tra dữ liệu là số thập phân.

*var* $validate = *array*(

'price' => *array*(

'rule' => *array*('decimal', 2) // có 2 chữ số thập phân \*.00

)

);

#### 2.9. Email

Kiểm tra dữ liệu nhập vào là một địa chỉ email hợp lệ.

*var* $validate = *array*('email' => *array*('rule' => 'email'));

*var* $validate = *array*(

'email' => *array*(

'rule' => *array*('email', *true*),

'message' => 'Nhập địa chỉ email hợp lệ !'

)

);

#### 2.10. equalTo

Đảm bảo kiểu giá trị nhập và là nhất định.

*var* $validate = *array*(

'food' => *array*(

'rule' => *array*('equalTo', 'cake'),

'message' => 'giá tri là kiểu String'

)

);

#### 2.11. Extension

Kiểm tra tính hợp lệ cho phần mở rộng như: \*.png, \*.jpg. Thường dùng trong upload file.

*var* $validate = *array*(

'image' => *array*(

'rule' => *array*('extension', *array*('gif', 'jpeg', 'png', 'jpg')), // kiểm tra định dang file

'message' => 'Cung cấp một ảnh hợp lệ!'

)

);

#### 2.12. File

Đảm bảo rằng giá trị là một tên tập tin hợp lệ.

#### 2.13. Ip

Kiểm tra tính hợp lệ của một địa chỉ Ip.

*var* $validate = *array*(

'clientip' => *array*(

'rule' => *array*('ip', 'IPv4'), *//or 'IPv6' or 'both' (default)*

'message' => 'Nhập một địa chỉ IP hợp lệ !'

)

);

#### 2.14. isUnique

Kiểm tra dữ liệu là duy nhất. Dữ liệu này không được sử dụng ở bất cứ hàng nào khác.

*var* $validate = *array*(

'login' => *array*(

'rule' => 'isUnique', // kiểm tra tên người dung đã có trong CSDL hay chưa

'message' => 'Tên người dùng đã tồn tại!'

)

);

#### 2.15. minLength

Kiểm tra dữ liệu nhập vào phải có chiều dài tối thiểu.

*var* $validate = *array*(

'login' => *array*( // kiểm tra ở trương login

'rule' => *array*('minLength', 8), // giá trị tối thiểu là 8 ký tự

'message' => 'Tên người dùng phải ít nhất 8 ký tự! '

)

);

#### 2.16. maxLength

Kiểm tra dữ liệu nhập vào phải có chiều dài tối đa.

*var* $validate = *array*(

'login' => *array*(

'rule' => *array*('maxLength', 15), // chiều dài tối đa 15 ký tự

'message' => 'Tên người dùng không vượt quá 15 ký tự !'

)

);

#### 2.17. Money

Kiểm tra tính hợp lệ giá trị có kiểu dữ liệu là money.

*var* $validate = *array*(

'salary' => *array*(

'rule' => *array*('money', 'left'), // ‘left’ xác định vị trí “$”

'message' => 'Nhập giá trị tiền hợp lệ !'

)

);

#### 2.18. Multiple

Xác định tiêu chuẩn cho giá trị đầu vào. Hỗ trợ “in”,”max” và “min”

*var* $validate = *array*(

'multiple' => *array*(

'rule' => *array*('multiple', *array*('in' => *array*('do', 'ray', 'me', 'fa', 'so', 'la', 'ti'), 'min' => 1, 'max' => 3)), //chọn ít nhất là 1, nhiều nhất là 3 trong mảng (do,ray,…,ti)

'message' => 'Please select one, two or three options'

)

);

#### 2.19. inList

Kiểm tra các giá trị có cùng kiểu sẽ nằm trong một tập hợp. Cần một mảng để lưu các giá trị nhất định (mảng mẫu). Kiểm trả các giá trị nhập vào có nằm trong mảng mẫu

*var* $validate = *array*(

'function' => *array*( //tên hàm

'allowedChoice' => *array*(

'rule' => *array*('inList', *array*('Foo', 'Bar')),

'message' => 'Enter either Foo or Bar.'

)

)

);

#### 2.20. Numeric

Kiểm tra các dữ liệu nhập vào là một số .

*var* $validate = *array*(

'pens' => *array*(

'rule' => 'numeric',

'message' => 'Nhập số lượng bút !'

)

);

#### 2.21. notEmpty

Kiểm tra giá trị rỗng ở các trường nhập liệu.

*var* $validate = *array*(

'title' => *array*( //tên tiêu đề

'rule' => 'notEmpty',

'message' => 'Trường này không được rỗng! '

)

);

#### 2.22. Phone

Kiểm tra tính hợp lệ của số điện thoại được nhập vào.

*var* $validate = *array*(

'phone' => *array*(

// giá trị là một số điện thoại,có thể để trống,”vn”= việt nam

'rule' => *array*('phone', *null*, 'vn')

)

);

#### 2.23. Postal

Kiểm tra tính hợp lệ của mã bưu chính.

*var* $validate = *array*(

'zipcode' => *array*(

'rule' => *array*('postal', *null*, 'vn')

)

);

#### 2.24. Range

Đảm bảo giá trị trong một phạm vi là nhất định. Nếu không cung cấp phạm vi, các quy tắc sẽ kiểm tra để đảm bảo giá tri là một hữu hạn trong phạm vi cho phép.

*var* $validate = *array*(

'number' => *array*(

'rule'=>*array*('range', -1, 11), // giá trị từ 0 đến 10

'message' => 'Nhập giá trị từ 0 đến 10 !'

)

);

#### 2.25. Ssn (Social Security number)

Kiểm tra số an sinh xã hội (số CMND).

*var* $validate = *array*(

'ssn' => *array*(

'rule' => *array*('ssn', *null*, 'vn')

)

);

#### 2.26. Url

Kiểm tra tính hợp lệ đường dẫn. Hỗ trợ các giao thức http, ftp, file, news,…

*var* $validate = *array*(

'website' => *array*(

'rule' => 'url'

)

);

Hay

*var* $validate = *array*(

'website' => *array*(

'rule' => *array*('url', *true*)

)

);

#### 2.27. mimeType (Multipurpose Internet Mail Extensions)

Kiểm tra các dạng tập tin trên internet (MIME).

*public* $validate = *array*(

'image' => *array*( //trường hình ảnh

'rule' => *array*('mimeType', *array*('image/gif')),

'message' => 'Tập tin MIME không hợp lệ!'

),

'logo' => *array*( // trường logo

'rule' => *array*('mimeType', '#image/.+#'),

'message' => 'Tập tin MIME không hợp lệ!'

),

);

#### 2.28. naturalNumber

Kiểm tra dữ liệu là số tự nhiên.

*public* $validate = *array*(

'windows' => *array*(

'rule' => 'naturalNumber', // giá trị là một số tự nhiên kiểu int

'message' => 'Nhập số lượng cửa !'

),

'chair' => *array*(

'rule' => *array*('naturalNumber', *true*),

'message' => 'Nhập số lượng ghế !'

),

);

#### 2.29. uploadError

Kiểm tra lỗi một tập tin khi được tải lên.

*public* $validate = *array*(

'image' => *array*(

'rule' => 'uploadError',

'message' => 'Không thể tải tập tin !'

),

);

### III. Viết behaviors cho Model

Trong CakePHP, khái niệm Behavior mang ý nghĩa là một cách tổ chức những hàm chức năng được định nghĩa trong Model. Nó cho phép chúng ta tách và tái sử dụng logic mà tạo nên loại behavior, thêm nữa nó không bắt buộc sự kế thừa. Một ví dụ là bạn xử lý những dữ liệu được lưu dạng cây thì bạn nên dùng TreeBehavior, bởi nó đã bao gồm các hàm thông dụng nhất dùng để thao tác với tree. Do đó bạn không cần phải tốn công sức viết ra các hàm để xử lý nữa. CakePHP cung cấp cho chúng ta bốn loại behavior là ACL, Containable, Translate, Tree, nhưng trong bài này tôi sẽ chỉ đề cập đến cách sử dụng chung và cách tạo ra behavior cũng như phương thức trong nó. Phần chi tiết của từng loại tôi sẽ đề cập đến ở các phần sau.

#### 3.1. ****Sử dụng Behaviors****

Thông qua biến $actAs thì các behavior sẽ được attach vào Model của bạn. Bạn có thể khai báo đơn giản như *public $actsAs array('Tree')*; câu lệnh này sẽ báo cho CakePHP là sẽ sử dụng tới *TreeBehavior*. Một số behaviors có thể bắt buộc hoặc cho phép có những settings khi attach vào Model. Ví dụ:

    public $actsAs = array('Tree' => array(

            'left'  => 'left\_node',

            'right' => 'right\_node'

        ),

        ‘Translate’

    );

Như ví dụ trên thì đã có setting cho *TreeBehavior* biết tên của trường ‘*left*’ và ‘*right’* lần lượt là **‘left\_node’** và **'right\_node'**. Trong khai báo biến *$actAs* thì bạn cũng hoàn toàn có thể khai báo thêm behavior như đoạn lệnh trên. Một khi behavior đã được attach vào Model thì nó sẽ có hiệu lực trong suốt lifetime của Model, nhưng nếu bạn lúc nào đó bỏ nó đi thì hoàn toàn có thể, ví dụ : *$this->Category->Behaviors->unload('Translate');* Nhưng trong thực tế chúng ta thường sẽ vô hiệu hóa việc behavior handle những callbacks của Model:

    $this->Category->Behaviors->disable('Translate');

Và khi bạn cần đến thì hãy enable lên:

    $this->Category->Behaviors->enable('Translate');

Trong một số trường hợp bạn sẽ cần attach mới một behavior nào đó bằng hàm *load()*:

    $this->Category->Behaviors->load(‘SomeBehavior’);

Một khả năng nữa của hàm *load()* là override setting của một behavior nào đó mà đã được attach vào Model, ta sẽ thử ghi đè lên tên trường ‘*left’* đã khai báo ở trên:

    $this->Category->Behaviors->load('Tree', array('left' => 'new\_left\_node'));

Cuối cùng, CakePHP cho phép xem danh sách những behavior nào đang được attach vào Model rồi bằng hàm *loaded()*:

    $behaviors = $this->Category->Behaviors->loaded();

#### 3.2. Tạo ra Behavior

Khi behavior được attach vào Model thì callbacks sẽ được gọi một cách tự động. Những callbacks tương tự như trong Models*: beforeFind*, *afterFind*, *beforeValidate*, *afterValidate*, *beforeSave*, *afterSave*, *beforeDelete*, *afterDelete* và *onError*. Behavior nên được đặt trong thư mục *app/Model/Behavior*. Và tên thì sẽ đặt dạng CamelCase với hậu tố là Behavior, ví dụ như *NameBehavior.php.*

Khi mà behaviors đã được share trong tất cả instances của Model mà đang dùng đến chúng, thì nên lưu trữ settings trên *alias/model* mà đang sử dụng behavior. Và khi tạo behaviors thì hàm *setup()* sẽ được gọi.

    public function setup(Model $Model, $settings = array()) {

        if (!isset($this->settings[$Model->alias])) {

            $this->settings[$Model->alias] = array(

                'option1\_key' => 'option1\_default\_value',

                'option2\_key' => 'option2\_default\_value',

                'option3\_key' => 'option3\_default\_value',

            );

        }

        $this->settings[$Model->alias] = array\_merge(

            $this->settings[$Model->alias], (array)$settings);

    }

#### 3.3. Tạo phương thức của behavior

Phương thức của behavior được available tới bất kì một model nào làm việc với behavior. Ví dụ chúng ta có:

    class Duck extends AppModel {

        public $actsAs = array('Flying');

    }

Và bạn sẽ có thể gọi FlyingBehavior hệt như chúng là phương thức trong Model của bạn: *$this->Duck->fly('hanoi','tphcm');* Như bạn thấy phương thức này có 2 tham số truyền vào nhưng thực tế là có 3, dạng như đoạn bên dưới:

    public function fly(Model $Model, $from, $to) {

        // code thực hiện điều gì đó ở đây.

    }

Cuối cùng trong phần này chúng ta sẽ đi qua một lượt những callbacks trong behavior. Chúng sẽ được trigger trước khi mà những callbacks của Model được gọi:

* *setup****(****Model $Model*, *array $settings = array()*): được gọi khi một behavior được attach vào model.
* *cleanup****(****Model $Model*): dùng khi detach behavior khỏi model.
* *beforeFind****(****Model $Model*, *array $query*): nếu mà trả về false thì sẽ không thực hiện hàm find().
* *afterFind****(****Model $Model*, *mixed $results*, *boolean $primary = false*): có thể dùng để bổ sung kết quả find().
* *beforeValidate****(****Model $Model*, *array $options = array()*): có ích khi muốn modify validate của model hay handle những logic nào đó trước khi validate.
* *afterValidate*(*Model $Model*): có thể dùng để thực hiên cleanup hay chuẩn bị gì đó khi cần.
* *beforeSave*(*Model $Model*, *array $options = array()*): có thể trả về true để tiếp tục việc lưu dữ liệu hay trả về false để bỏ việc lưu trữ.
* *afterSave*(*Model $Model*, *boolean $created*, *array $options = array()*): có thể đùng để thực hiện cleanup sau khi lưu trữ thành công.
* *beforeDelete*(*Model $Model*, *boolean $cascade = true*): có thể trả về true để tiếp tục việc lưu dữ liệu hay trả về false để bỏ việc lưu trữ.
* *afterDelete*(*Model $Model*): có thể đùng để thực hiện cleanup sau khi xóa thành công.

## BÀI 4. THIẾT KẾ CONTROLLER

**Nội dung:**

1. Các thành phần trong Controller và cách sử dụng.
2. Các khai báo, các biến hàm để chuyển tiếp cho view.
3. Viết components để sử dụng Controller.

### I. Các thành phần trong Controller và cách sử dụng

#### 1.1. Giới thiệu

**Controller** là các lớp điều khiển luồng ứng dụng, tiếp nhận yêu cầu người dùng thông qua HTTP header, sau đó chuyển tiếp nó đến các lớp phụ trách trực tiếp xử lý yêu cầu. Tùy theo cách thiết kế lớp mà chúng ta thường thấy Controller gồm:

* **Front Controller**. Là một controller xử lý tất cả các yêu cầu người dùng cho website. Front Controller có nhiệm vụ hợp nhất tất cả các xử lý yêu cầu vào một kênh yêu cầu thông qua một đối tượng.
* **Dispatcher**: Lớp điều phối hướng các điều khiển đi mức cao hơn.
* **Action**: lớp xử lý các sự kiện chính, nơi dẫn đến application flow chủ yếu (Main Event Handler)**.**
* **Request**: xử lý một phần dữ liệu đầu vào ở mức GET, POST và PUT.
* **Session**: xử lý một phần dữ liệu đầu vào ở mức SESSION   
  Tùy theo dữ liệu đầu vào, Controller sẽ thực hiện các phép lọc (với dịch vụ lấy từ Model), các tính toán lựa chọn (Action Mapping) dựa trên kiến trúc và cấu hình nhằm xác định thành phần lớp chính sẽ thực hiện yêu cầu của người dùng. Hiểu một cách đơn giản, Controller là thành phần trung gian giữa View và Model. Nó nhận dữ liệu nhập vào qua View, sau đó gọi Model tương ứng rồi lấy kết quả trả về từ Model này. Tiếp theo, một View thích hợp sẽ được lựa chọn. Controller sẽ chuyển tiếp dữ liệu vào view để nó xử lý.

Một số hoạt động thường thấy của Controller:

* Tạo form, gửi tin nhắn đến form để yêu cầu kiểm tra dữ liệu.
* Tạo các dịch vụ liên quan đến nghiệp vụ ứng dụng, yêu cầu các lớp dịch vụ tương tác với nguồn dữ liệu để trả về hay thay đổi trạng thái dữ liệu: thực hiện các thao tác chuyển đổi dữ liệu, kiểm tra quyền truy cập trên một hoạt động cụ thể, tương tác với database, tương tác với các web services.
* Tạo đối tượng view, gán các nguồn dữ liệu lấy được từ đối tượng dịch vụ vào cho view.

#### 1.2. App controller

**AppController** là một lớp cha của tất cả các lớp controller trong ứng dụng. Bản thân **AppController** kế thừa lớp Controller trong thư viện CakePHP. AppController được định nghĩa trong ***src/Controller/AppController.php****:*

namespace App\Controller;

use Cake\Controller\Controller;

class AppController extends Controller{

}

Vì lớp **AppController** là lớp cha của các lớp controller trong ứng dụng, do đó mà các thuộc tính và các phương thức trong lớp **AppController** sẽ có thể được chia sẽ và sử dụng bởi các lớp con.

#### 1.3. Pages Controllers

CakePHP hỗ trợ chúng ta Pages Controller (cake/libs/controller/pages\_controller.php). Trang chủ (home page) mà bạn thấy sau khi cài đặt là được tạo ra bằng cách dùng controller này. Page controller thường được dùng để tạo ra các trang tính.

Ví dụ: bạn tạo trang About (bạn tạo 1 file view /app/views/pages/about\_us.ctp, bạn truy cập nó bằng cách sử dụng đường dẫn http://example.com/pages/about\_us  
để sử dụng Pages controllers, chúng ta chỉ cần copy pages\_controllers.php từ core của CakePHP đến ứng dụng của mình.

#### 1.4. Các thuộc tính của controller

##### 1.4.1. $name

Được dùng để thiết lập tên của controller

Thông thường thì đây là hình thức số nhiều của tên model mà controller sử dụng. Ví dụ:  
# $name controller attribute usage example  
  
class RecipesController extends AppController {  
var $name = 'Recipes';  
}

##### 1.4.2. $components, $helpers and $uses

Ví dụ:  
class RecipesController extends AppController {  
var $name = 'Recipes';  
var $uses = array('Recipe', 'User');  
var $helpers = array('Ajax');  
var $components = array('Email');  
}

##### 1.4.3. $layout

Sử dụng $layout để chọn layout cho chương trình /app/views/layouts ($layout sẽ lưu tên của layout trong /app/views/layouts)

Nếu giá trị này ko được thiết lập thì layout mặc định sẽ được sử dụng cho ứng dụng của bạn: default.ctp. Ví dụ:

// Using $layout to define an alternate layout  
class RecipesController extends AppController {  
function quickSave() {  
$this->layout = 'ajax';  
}  
}

##### 1.4.4. $params

Được sử dụng để cung cấp quyền truy cập vào các thông tin về request hiện tại:

* form: $this->params['form'];
* admin:$this->params['admin'];
* bare: $this->params['bare'];
* form: $this->params['isAjax'];

### II. Các khai báo, các biến hàm để chuyển tiếp cho View

Cũng giống như các Model. Tât cả các Controller trong Cake PHP cũng được đặt trong thư mục riêng gọi là Controller trong thư mục app.

<?php

// class <tên hàm> extends <tên thư mục chứa Controller>

*class* RecipesController *extends* AppController {

// chứa các biến và các hàm

}

?>

#### 2.1 Khai báo biến

Tên biến phải được bắt đầu bằng dấu “$” <tên biến> = “<định nghĩa cho biến>”.

Cú pháp:

*var* $name = 'Recipes'; // var <tên biến> = ‘<định nghĩa cho biến>’

Biến sẽ được khai báo bên ngoài các hàm. Nằm trong một lớp với các hàm

<?php

*class* RecipesController *extends* AppController { // khai báo lớp

*var* $name = 'Recipes'; // phần khai báo các biến

*var* $uses = *array*('Recipe', 'User');

*var* $helpers = *array*('Ajax');

*var* $components = *array*('Email');

*function* search() { //phần khai báo các hàm

// lệnh thực thi

}

}

?>

#### 2.2 Khai báo hàm

Cú pháp:

<?php

// Using $layout to define an alternate layout

*class* RecipesController *extends* AppController {

*function* quickSave() { // function <tên hàm>(){…}

$this->layout = 'ajax'; //các lệnh thực thi

}

}

?>

#### 2.3 Các hàm thông dụng (có sẵn trong Controller)

##### 2.3.1 Hàm *set()*

Hàm *set()* được sử dụng để chuyển dữ liệu từ controller action tới view. *Đầu tiên là tên của biến mà dữ liệu sẽ có trong view, và thứ 2 là dữ liệu thực cần được chuyển qua.*

set(string $var, mixed $value) // string <tên biến>,minxed <dữ liệu cần chuyển>

<?php

//First you pass data from the controller:

$this->set('color', 'pink');

//Then, in the view, you can utilize the data:

?>

You have selected <?php *echo* $color; ?> icing for the cake.

##### 2.3.2 Hàm *render()*

Các *render()* sẽ được gọi một cách tự động ở cuối mỗi controller action. Thực hiện tất cả các view logic (bằng cách sử dụng dữ liệu từ trong các *set()* ).

render(string $action, string $layout, string $file)

Nếu hàm *search()* được gọi thì các tập tin trong view (/app/views/recipes/search.ctp) sẽ được gọi.

*class* RecipesController *extends* AppController {

...

*function* search() {

// Render the view in /views/recipes/search.ctp

$this->render();

}

...

}

Ngoài ta cũng có thể dùng để xác định một tập tin view trong controller bằng cách sử dụng $action. Nếu $action bắt đầu với dấu “ / ” có nghĩa là xem hoặc là tập tin liên quan /app/views. Nó hữu ích trong việc sử dung ajax.

// Render the element in /views/elements/ajaxreturn.ctp

$this->render('/elements/ajaxreturn');

Ta cũng có thể gọi hàm render() trực tiếp (app/views/posts/custom\_file.ctp) thay cho (app/views/posts/my\_action.ctp).

*class* PostsController *extends* AppController {

*function* my\_action() {

$this->render('custom\_file');

}

}

##### 2.3.3 Hàm *redirect()*

Dùng chuyển hướng đến một địa chỉ khác (Url).

redirect(mixed $url, integer $status, boolean $exit)

*function* placeOrder() {

//Logic for finalizing order goes here

*if*($success) {

$this->redirect(*array*('controller' => 'orders', 'action' => 'thanks'));

} *else* {

$this->redirect(*array*('controller' => 'orders', 'action' => 'confirm'));

}

}

Có thể sử dụng một địa chỉ tương đối hoặc tuyệt đối

$this->redirect('/orders/thanks'); //địa chỉ tương đối

$this->redirect('http://www.example.com'); //địa chỉ tuyệt đối

Cũng có thể truyền dữ liệu đến các hành động:

$this->redirect(*array*('action' => 'edit', $id));

Nếu muốn chuyển hướng đến trang referer:

$this->redirect($this->referer());

##### 2.3.4 Hàm *flash()*

Giống như *redirect().* Hàm *flash()* cũng dùng để chuyển hướng đến một trang khác (hay Url khác). Điểm khác nhau giữa chúng là hàm flash() sẽ hiển thị một thông báo trước khi chuyển đến trang mới (hay Url khác).

##### 2.3.5 Hàm callbacks()

Có thể chèn nó vào trước hay sau khi thực hiện một controller action.

beforeFilter().

Dùng để kiểm tra một phiên làm việc hoặc quyền truy cập của người dùng.

beforeRender()

Được dùng khi ta gọi các *render()* bằng cách thủ công.

afterFilter()

được gọi sau khi thực hiện các controller action và rendering.

\_beforeScaffold($method)

$method: tên gọi của một phương pháp như: index, edit.

\_afterScaffoldSave($method)

$method: tên gọi của một phương pháp edit hoặc update.

\_afterScaffoldSaveError($method)

$method: tên gọi của một phương pháp edit hoặc update.

\_scaffoldError ($ method)

$method: tên gọi của một phương pháp như: index, edit.

##### 2.3.6 Hàm *referer()*

Trả về một địa chỉ Url mặc định khi mà trang hiện tại không thể tải được địa chỉ Url. $default “địa chỉ Url mặc định”.

string referer(mixed $default = null, boolean $local = false)

<?php

*class* UserController *extends* AppController {

*function* delete($id) {

// delete code goes here, and then...

*if* ($this->referer() != '/') {

$this->redirect($this->referer());

} *else* {

$this->redirect(*array*('action' => 'index'));

}

}

}

?>

##### 2.3.7 Hàm disableCache()

Dùng để thông báo cho người dùng trình duyệt biết, khi các yêu cầu không được tài về. Với các tiêu đề như:

Expires: Mon, 26 Jul 1997 05:00:00 GMT

Last-Modified: [current datetime] GMT

Cache-Control: no-store, no-cache, must-revalidate

Cache-Control: post-check=0, pre-check=0

Pragma: no-cache

##### 2.3.8 Hàm postcondition()

Dùng để xây dựng các phím tắt nhằm giúp việc tìm kiếm được nhanh chóng.

*function* index() {

$conditions = $this->postConditions($this->data);

$orders = $this->Order->find("all",compact('conditions'));

$this->set('orders', $orders);

}

Ta có thể sử dụng để truy xuất đến CSDL.

/\*

Contents of $this->data

array(

'Order' => array(

'num\_items' => '4',

'referrer' => 'Ye Olde'

)

)

\*/

//Let’s get orders that have at least 4 items and contain ‘Ye Olde’

$conditions=$this->postConditions(

$this->data,

*array*(

'num\_items' => '>=',

'referrer' => 'LIKE'

)

);

$orders = $this->Order->find("all",compact('conditions'));

##### 2.3.8 Hàm *paginate()*

Dùng để đánh số trang, định dạng kích thước trang,…

##### 2.3.9 Hàm requestAction()

Dùng để gọi một controller action ở bất kỳ vị trí nào và trả về kết quả cho từng yêu cầu: requestAction(string $url, array $options)

// controllers/comments\_controller.php

class CommentsController extends AppController {

function latest() {

$comments = $this->Comment->find('all', array('order' => 'Comment.created DESC', 'limit' => 10));

if (!empty($this->params['requested'])) {

return $comments;

} else {

$this->set(compact('comments'));

}

}

}

### III. Viết components để sử dụng Controller

Để chia sẽ các gói logic giữa các controller. Cakephp cung cấp các gói components, bạn có thể sử dụng cho các công việc khác nhau trong cùng một dự án. Ngoài ra bạn cũng có thể tạo ra các components cho từng công việc cụ thể khác nhau. Nên tạo các component riêng cho contronller với mã sạch để tái sử dụng cho các dự án tiếp theo.

#### 3.1. Cấu hình components

Một số thành phần yêu cầu cấu hình. Ví dụ về các thành phần đòi hỏi cấu hình là: *Authentication* và *Cookie*. Việc cấu hình cho các thành phần này và cho các thành phần khác nói chung thường được thực hiện thông qua *loadComponent()* trong controller là phương thức *initialize()* hoặc các mảng *$components*

*class* PostsController *extends* AppController

{

*public* *function* initialize()

{

*parent*::initialize();

$this->loadComponent('Auth', [

'authorize' => ['controller'],

'loginAction' => ['controller' => 'Users', 'action' => 'login']

]);

$this->loadComponent('Cookie', ['expiry' => '1 day']);

}

}

Ta có thể cấu hình các thành phần trong thời gian thực thi bằng cách sử dụng phương thức *config()*. Thường việc thay đổi được thực hiện trong phần controller bằng phương thức *beforeFilter()*.

*public* *function* beforeFilter()

{

$this->Auth->config('authorize', ['controller']);

$this->Auth->config('loginAction', ['controller' => 'Users', 'action' => 'login']);

$this->Cookie->config('name', 'CookieMonster');

}

#### 3.2. Tạo components

Bước đầu tiên tạo file components và lớp mới theo đường dẫn */app/controllers/components/tên tập tin (ví dụ:math.php)*. Cấu trúc cơ bản của components

<?php

*class* MathComponent *extends* Object { // tên lớp, phần mở rộng

*function* doComplexOperation($amount1, $amount2) { // tên hàm

*return* $amount1 + $amount2; // các dòng lệnh xử lý

}

}

?>

##### 3.2.1. Các components trong controllers

Khi một component kết thúc ta có thể tái sử dụng chúng cho các ứng dụng khác trong cotroller bằng cách đặt tên các component (trừ các “component” một phần) trong mảng *$component* của controller. Các controller sẽ tự động gửi đi các thuộc tính mới được đặt tên theo thành phần, qua đó ta có thể truy cập vào một thể hiện của nó.

/\* Make the new component available at $this->Math,

as well as the standard $this->Session \*/

var $components = array('Math', 'Session');

Các component khai báo trong *Appcontroller* sẽ được sử dùng chung với các component còn lại của controller.

Các thông số của component trong cotroller sẽ được biên dịch thông qua phương thức *initialize()*. Các thông số được xử lý bởi các component.

*var* $components = *array*(

'Math' => *array*(

'precision' => 2,

'randomGenerator' => 'srand'

),

'Session', 'Auth'

);

##### 3.2.2. Lớp truy cập MVC trong Components

Các component đặc trưng gọi là *callback* được sử dụng trong các lớp cha ở controller. Việc sử dụng đúng các *callback* sẽ giúp cho việc tạo mới và sử dụng các component được dễ dàng hơn.

initialize(&$controller, $settings=array())

Phương thức *initialize* được gọi trước khi gọi phương thức *beforeFilter* trong controller.

startup(&$controller)

Phương thức *start* được gọi sau phương thức *beforeFilter* của controller nhưng trước khi controller thực thi các yêu cầu xử lý hiện hành.

beforeRender(&$controller)

Phương thức *beforeRender* được gọi sau khi controller thực hiện các yêu cầu logic, nhưng trước các controller của *view* và *layout*.

shutdown(&$controller)

Phương thức *shutdown* được gọi trước khi gửi đến trình duyệt

beforeRedirect(&$controller, $url, $status=null, $exit=true)

Phương thức *beforeRedirect* được gọi khi phương thức *redirect* của controller được gọi, nhưng trước bất kỳ hành động nào khác. Nếu phương thức này trả về giá trị *false* thì controller sẽ không tiếp tục thực hiện các yêu cầu của redirect. Các biến *$url*, *$status* và *$exit* có cùng ý nghĩa như các biến trong controller. Bạn cũng có thể tra về một chuỗi địa chỉ “Url” hoặc trả về mảng kết hợp với phím “url”, tùy chọn “status” và “exit”.

<?php

*class* SkeletonComponent *extends* Object {

*//called before Controller::beforeFilter()*

*function* initialize(&$controller, $settings = *array*()) {

*// saving the controller reference for later use*

$this->controller =& $controller;

}

*//called after Controller::beforeFilter()*

*function* startup(&$controller) {

}

*//called after Controller::beforeRender()*

*function* beforeRender(&$controller) {

}

*//called after Controller::render()*

*function* shutdown(&$controller) {

}

*//called before Controller::redirect()*

*function* beforeRedirect(&$controller, $url, $status=*null*, $exit=*true*) {

}

*function* redirectSomewhere($value) {

*// utilizing a controller method*

$this->controller->redirect($value);

}

}

?>

Ta cũng có thể sử dụng các component khác bên trong các component tùy chỉnh. Ta sẽ tạo một biến *$components* trong lớp (điều này giống như ta đang ở trong một controller) như là một mảng các biến component mà ta muốn sử dụng.

<?php

*class* MyComponent *extends* Object {

*// This component uses other components*

*var* $components = *array*('Session', 'Math');

*function* doStuff() {

$result = $this->Math->doComplexOperation(1, 2);

$this->Session->write('stuff', $result);

}

}

?>

Để truy cập hoặc sử dụng trong một component mà chưa được khai báo.

<?php

*class* MathComponent *extends* Object {

*function* doComplexOperation($amount1, $amount2) {

*return* $amount1 + $amount2;

}

*function* doReallyComplexOperation ($amount1, $amount2) {

$userInstance = ClassRegistry::init('User');

$totalUsers = $userInstance->find('count');

*return* ($amount1 + $amount2) / $totalUsers;

}

}

?>

##### **3.2.3 Sử dụng các component khác trong component của bạn**

Có thể sử dụng các component khác bên trong các component mà ta định nghĩa trong controller. Sử dụng các biến *Var $components*.

<?php

*class* CustomComponent *extends* Object {

*var* $name = 'Custom'; *// the name of your component*

*var* $components = *array*('Existing'); *// the other component your component uses*

*function* initialize(&$controller) {

$this->Existing->foo();

}

*function* bar() {

*// ...*

}

}

?>

<?php

*class* ExistingComponent *extends* Object {

*var* $name = 'Existing';

*function* initialize(&$controller) {

$this->Parent->bar();

}

*function* foo() {

*// ...*

}

}

?>

## BÀI 5. THIẾT KẾ VIEW

**Nội dung:**

1. Thiết kế giao diện từ Controller chuyển qua.
2. Thiết kế giao diện trang chủ.
3. Thiết kế các helpers dùng cho View.

### I. Thiết kế giao diện từ Controller chuyển qua

#### 1.1. Gọi các hàm của controller trong view

Cú pháp:

$this->load->view("tên\_view");

Ví dụ: Tạo file hello.ctp tại thư mục controller với nội dung như sau:

<?php

class Hello extends appController{

public function index(){

echo "<h2> Hello Work </h2>";

$this ->load ->view ("hello\_view");

}

}

?>

Tạo file hello\_view.php trong thư mục views với nội dung như sau:

<?php

echo "Bạn đang ở trong hello view";

?>

Truyền tham số từ controller cho view.

$this -> load ->view ("hello\_view",$data);

$data: là biến chứa các giá trị mà ta muốn truyền.

Lưu ý: khi dữ liệu truyền về cho view thi ta nên đặt ở dưới dạng một phần tử mảng.

$data[‘title’]= “Hello user”;

Gọi tại view:

<?php

echo $title;

?>

#### 1.2 Có các view

Để gọi các view ta sử dụng cú pháp:

$this->method()

##### 1.2.1 set()

set(string $var, mixed $value)

Cũng giống như hàm set() ở phần controller. Cho phép bạn thêm biến vào viewVars.

$this->set('activeMenuButton', 'posts'); //biến “$activeMenuButton” sẽ chứa giá trị là “posts”

##### 1.2.2 getVar()

Các giá trị của viewVars với tên là $var.

##### 1.2.3 getVars()

Danh sách các biến có trong phạm vi hiện tại. sẽ trả về một mảng các tên biến.

##### 1.2.4 error()

error(int $code, string $name, string $message)

Hiển trị một trang báo lỗi cho người dùng. Trang báo lỗi nằm trong layout/error.ctp

$this->error(404, 'Not found', 'This page was not found, sorry');

##### 1.2.5 elment()

element(string $elementPath, array $data, bool $loadHelpers)

Giúp ta xây dựng các thành phần cố định trong một trang web. Ta có thể tái sử dụng các thành phần đó (ví dụ: phần quản cáo, help boxes, extra menus và login forms,…).

##### 1.2.6 *Uuid()*

uuid(string $object, mixed $url)

Tạo ra một DOM ID cho đối tượng và thường dựa trên đối tượng và url. Thường sử dụng trong AjaxHelper.

$uuid = $this->uuid('form', *array*('controller' => 'posts', 'action' => 'index'));

//$uuid contains 'form0425fe3bad'

##### 1.2.7 addScript()

addScript(string $name, string $content)

Giúp cho người dùng có thể thêm một Javascript hay CSS một cách trực tiếp.

#### 1.3. Themes

Sử dụng các chủ để nhằm thay đổi giao diện một trang web một cách nhanh chóng và dễ dàng.

*class* ExampleController *extends* AppController {

*var* $view = 'Theme'; // tạo một chủ đề

}

Gọi chủ đề

$this->theme = 'another\_example';

#### 1.4 media views

Cho phép bạn gửi các tập tin cho người dùng. Ví dụ bạn muốn có một thư mục tập tin bên ngoài webroot nhằm để ngăn chặn người dùng có thể kết nối trực tiếp vào. Ta sử dụng media views để xác thực trước khi chuyển các tập tin cho người dùng.

*class* ExampleController *extends* AppController {

*function* download () {

$this->view = 'Media';

$params = *array*(

'id' => 'example.zip',

'name' => 'example',

'download' => *true*,

'extension' => 'zip', *// must be lower case*

'path' => APP . 'files' . DS *// don't forget terminal 'DS'*

);

$this->set($params);

}

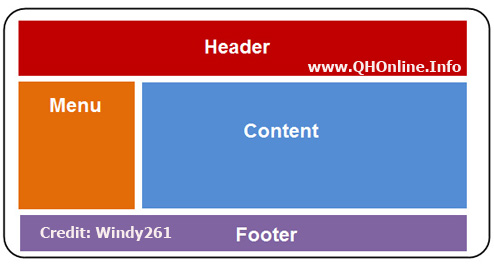
}

### II. Thiết kế giao diện trang chủ

Trong CakePHP Framework khi viết một ứng dụng thì mặc định CakePHP Framework đã xây dựng cho chúng ta giao diện có sẵn. Tuy nhiên, tùy theo nhu cầu, xu hướng, chúng ta muốn tùy biến hoặc tạo riêng cho mình một giao diện. Thì đó là lúc bạn cần phải **Sử dụng Layout trong CakePHP Framework** và cách áp dụng một Helper CakePHP Framework vào ứng dụng của mình.

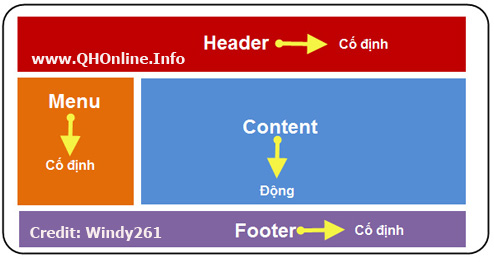
Trong một website không phải toàn bộ giao diện lúc nào cũng thay đổi mà có những thành phần cố định và thành phần động. Trong CakePHP chúng ta sẽ xử lý các thành phần cố định thông qua một class Helper. Và thành phần động sẽ được xử lý qua controller. Để hiểu rõ hơn phần này, chúng ta sẽ cùng đi xây dựng một layout trong CakePHP.

Khi phân tích 1 trang web, nhìn chung ta thấy gồm các phần chính như sau:



**www.giasu.com**

Để vận dụng được sự hổ trợ mạnh mẽ chắc năng load layout của **CakePHP FrameWork**, ta phân tích các *thành phần cố định và thành phần động*:



Như vậy ta để tránh việc xử lý các *thành phần cố định* ở controller ta chỉ cần viết một *class Hepler* để hiển thị nó. Còn *thành phần động* sẽ được xử lý thông qua Controllers.

Cần chuẩn bị các file như sau:

* **app/:** app\_controller.php
* **app/controllers/templates/:** templates\_controller.php (Controller chính để load layout).
* **app/views/templates/:**
  + index.ctp
  + view.ctp
* **app/views/helpers/:** common.php (Tạo các thành phần cố định : menu , header,footer).
* **template.ctp** (File chứa nội dung layout)
* **app/webroot/css/:** style.css (file CSS của layout)
* Tạo file **app\_controller.php** (app/):

<?php

class AppController extends Controller {

}

?>

* Tạo file **common.php** (app/views/helpers/):

<?php

class CommonHelper extends HtmlHelper {

// Hàm tạo menu

function create\_menu(){

$menu = "<ul>";

$menu .= "<li>".$this->link("CodeIgniter", array(

"controller"=>"templates",

"action"=>"view",

1))."</li>";

$menu .= "<li>".$this->link("CakePHP", array(

"controller"=>"templates",

"action"=>"view",

2))."</li>";

$menu .= "<li>".$this->link("Zend", array(

"controller"=>"templates",

"action"=>"view",

3))."</li>";

$menu .= "</ul>";

return $menu;

}

//Hàm tạo các thành phần cho header và footer

function general(){

$data = array(

"header" => "www.giasu.com",

"footer" => "Copyright 2011 &copy; | giasu",

);

return $data;

}

?>

* Tạo file templates\_controller.php (app/controllers/templates/):

<?php

class TemplatesController extends AppController {

var $layout = "template"; // load file chứa nội dung layout : views/layouts/template.ctp

var $helpers = array("Html","Common"); // Thành phần Helper Common được gọi để tạo menu,header,footer trong view

function index(){

$this->set('title\_for\_layout', 'Templates By giasu');

$this->set("content","giasu");

}

function view($id){

switch($id){

case 1 :{

$this->set('title\_for\_layout', 'CodeIgniter FrameWork');

$this->set("content","CodeIgniter FrameWork");

}

break;

case 2 :{

$this->set('title\_for\_layout', 'CakePHP FrameWork');

$this->set("content","CakePHP FrameWork");

}

break;

case 3 :{

$this->set('title\_for\_layout', 'Zend Framework');

$this->set("content","Zend Framework");

}

break;

default :

$this->set("content","Framwork");

break;

}

}

}

?>

* Tạo file layout template.ctp (app/views/layouts/template.ctp): File này chứa nội dung layout bao gồm các thành phần cố định và thành phần động như ban đầu mô tả. Nội dung file này gồm mã HTML và PHP…

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title><?php echo $title\_for\_layout;?></title>

<?php echo $this->Html->css("style"); // link oi file style.css (app/webroot/css/style.css)?>

<?php $general = $this->Common->general(); // Lấy các giá trị của thành phần tĩnh : header,footer ?>

</head>

<body>

<div id="top">

<center><h2><?php echo $general['header']; ?></h2></center>

</div>

<div id="main">

<div id="menu">

<?php echo $this->Common->create\_menu(); // goi ham tao menu tu common helper?>

</div>

<div id="content">

<h1><?php echo $content; // Thành phần động ?></h1>

</div>

</div>

<div id="footer">

<center><?php echo $general['footer'];?></center>

</div>

</body>

</html>

* Tạo file style.css (app/webroot/css/style.css):

body {

margin: auto;

width: 1000px;

font-family: Verdana, Geneva, sans-serif;

}

#top {

float: left;

width: 1000px;

height: 100px;

background-color: #F36;

color: #FFF;

}

#main {

float: left;

width: 1000px;

}

#menu {

float: left;

width: 200px;

background-color: #F96;

}

#menu ul {

margin: 0px;

}

#menu a {

color: #FFF;

font-size: 12px;

}

#content {

float: left;

width: 800px;

}

#content h1 {

font-size: 18px;

color: #0CF;

padding-left: 50px;

}

#footer {

float: left;

width: 1000px;

height: 50px;

background-color: #96C;

font-size: 12px;

font-weight: bold;

color: #FFF;

}

* Chạy thử ứng dụng:

*http://localhost/cakephp/templates*



**www.giasu.com**

**www.giasu.com**

Khi click vào link của Menu: [*http://localhost/cakephp/templates/view/1*](http://localhost/cakephp/templates/view/1)



### III. Thiết kế các helpers dùng cho View

#### 3.1. Helpers là gì?

Có thể hiểu nôm na rằng helper trong CakePHP là một tập hợp các thư viện hữu ích để dùng trong view. CakePHP đã xây dựng sẵn rất nhiều helper: form, html, ajax, number, session, rss, xml, time….

Phần này sẽ chỉ cho bạn cách để tạo ra các helpers của riêng bạn, và vạch ra những nhiệm vụ cơ bản của các helpers chính trong CakePHP có thể giúp bạn hoàn thành.

#### 3.2. Sử dụng Helpers

Bạn sử dụng helpers trong CakePHP bằng cách tạo một controller nhận biết nó. Mỗi controller có một biến helpers riêng mà ở đó danh sách các helpers được tạo sẵn trong view. Để kích hoạt một helper trong view của bạn, thêm tên của helpers cho mảng biến $helpers của controller (muốn dùng helper nào thì trong Controller ta phải khai báo thông qua biến $helpers).

<?php

class BakeriesController extends AppController {

var $helpers = array('Form', 'Html', 'Javascript', 'Time');

}

?>

Bạn cũng có thể thêm helpers vào bên trong một action, và nó sẽ chỉ phù hợp cho action đó mà không phù hợp với hành động khác trong cùng controller. Điều này sẽ tiết kiệm khả năng xử lý cho các action khác mà không sử dụng đến các helper cũng như giúp giữ cho các controller tổ chức tốt.

<?php

class BakeriesController extends AppController {

function bake {

$this->helpers[] = 'Time';

}

function mix {

// The Time helper is not loaded here and thus not available

}

}

?>

Nếu bạn cần phải kích hoạt một helper cho tất cả các controller chỉ cần thêm tên của những helpers đó vào mảng các biến $helpers trong: */app/app\_controller.php* (hoặc tạo ra nếu không có). Nhớ rằng là nó bao gồm những Html và Form helpers mặc định.

<?php

class AppController extends Controller {

var $helpers = array('Form', 'Html', 'Javascript', 'Time');

}

?>

Bạn cũng có thể tùy chọn cho helpers. Các tùy chọn này có thể được sử dụng để thiết lập các giá trị thuộc tính hay thay đổi behaviors của một helpers.

<?php

class AwesomeHelper extends AppHelper {

function \_\_construct($options = null) {

parent::\_\_construct($options);

debug($options);

}

}

?>

<?php

class AwesomeController extends AppController {

var $helpers = array('Awesome' => array('option1' => 'value1'));

}

?>

#### 3.3. Tạo Helpers

Nếu một helper chính không phù hợp với nhu cầu của bạn, hãy tạo một helper, helpers rất dễ tạo.

Chúng ta sẽ tạo ra một helper mà có thể được sử dụng để xuất ra một link CSS theo kiểu riêng mà ta cần ở nhiều nơi khác nhau trong ứng dụng của mình. Để phù hợp với logic của chúng ta vào cấu trúc của helper hiện tại trongCakePHP, chúng ta sẽ cần phải tạo ra một lớp mới trong /app/views/helpers. Tạm gọi helper đó là LinkHelper:

<?php

/\* /app/views/helpers/link.php \*/

class LinkHelper extends AppHelper {

function makeEdit($title, $url) {

// Logic to create specially formatted link goes here...

}

}

?>

#### 3.4. Các Helpers khác

Bạn có thể muốn sử dụng một số chức năng đã tồn tại trong các helpers khác. Để làm diều đó, bạn có thể chỉ định helpers bạn muốn sử dụng với một mảng các biến helpers, được định dạng giống như bạn muốn trong một controller.

<?php

/\* /app/views/helpers/link.php (using other helpers) \*/

class LinkHelper extends AppHelper {

var $helpers = array('Html');

function makeEdit($title, $url) {

// Use the HTML helper to output

// formatted data:

$link = $this->Html->link($title, $url, array('class' => 'edit'));

return "<div class=\"editOuter\">$link</div>";

}

}

?>

#### 3.5. Phương thức Callback

Các helpers đều có một callback được sử dụng bởi các lớp cha của controller.

beforeRender()

Phương thức beforeRender được gọi sau phương thức beforeRender của controller nhưng trước khi controlller render cho view và layout.

#### 3.6. Sử dụng Helpers của bạn

| **CakePHP Helper** | **Description** |
| --- | --- |
| [*AJAX*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/AJAX.html) | Sử dụng song song với các thư viện Prototype JavaScript để tạo ra các tính năng Ajax ở view. Chứa đựng các phương pháp tắt như kéo / thả, form ajax & các link, observers,… |
| [*Cache*](http://book.cakephp.org/1.3/en/The-Manual/Core-Utility-Libraries/Cache.html) | Được sử dụng bởi các thành phần chính để làm bộ nhớ đệm cho phần nội dung của View |
| [*Form*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/Form.html) | Tạo ra các from HTML và các mẫu element tự điền và xử lý các vấn đề xác nhận. |
| [*HTML*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/HTML.html) | Phương pháp tiện lợi cho việc tạo ra đánh dấu đúng định dạng như: hình ảnh, liên kết, bảng, thẻ tiêu đề,… |
| [*Js*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/Js.html) | Được sử dụng để tạo ra javascript tương thích với các thư viện Javascript khác nhau. Thay thế cho JavascriptHelper và AjaxHelper với một giải pháp linh hoạt hơn. |
| [*Javascript*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/Javascript.html) | Được sử dụng để thoát khỏi các giá trị sử dụng trong JavaScripts, viết ra dữ liệu cho các đối tượng JSON, và các khối mã lệnh định dạng. |
| [*Number*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/Number.html) | Định dạng số và tiền tệ. |
| [*Paginator*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/Paginator.html) | Phân trang Mô hình dữ liệu và phân loại. |
| [*RSS*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/RSS.html) | Phương pháp tiện lợi cho việc xuất RSS cung cấp dữ liệu XML |
| [*Session*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/Session.html) | Truy cập để đọc các giá trị của session trong views. |
| [*Text*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/Text.html) | Liên kết nhanh, làm nổi bật, rút gọn từ một cách thông minh |
| [*Time*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/Time.html) | Nhận diện tiệm cận (có là năm tiếp theo?), tốt trong định dạng chuỗi (Today, 10:30 am) và chuyển đổi múi giờ. |
| [*XML*](http://book.cakephp.org/1.3/en/The-Manual/Core-Helpers/XML.html) | Phương pháp tiện lợi cho việc tạo ra các tiêu đề XML và các element. |

## BÀI 6. CÁC THÀNH PHẦN KHÁC

**Nội dung:**

1. Thiết kế giao diện từ Controller chuyển qua.
2. Thiết kế giao diện trang chủ.
3. Thiết kế các helpers dùng cho View.