PHP & MySQL

Student Book

Hà nội : 2012

Mục lục

Xây dựng ứng dụng Web bằng PHP&MySQL

1. Ngôn ngữ lập trình

1.1 Giới thiệu và cài đặt PHP

1.1.1 Giới thiệu chung

1.1.1.1 Trang web tĩnh và trang web động

Trang Web tĩnh là các trang Web không cho phép bạn có thể tương tác với người dùng (chẳng hạn như là trao đổi hay thu thập các thông tin từ phía người dùng). Nó là các trang web có đuôi *.htm thông thường. Ngược lại, các trang Web động cho phép bạn nhận thông tin từ người dùng, xử lý thông tin đó, và có thể đáp trả lại các yêu cầu của họ.

1.1.1.2 Lập trình Script

Các trang web nguyên thuỷ sử dụng ngôn ngữ định dạng chuẩn là **HTML** (HyperText Markup Language). HTML chuẩn chỉ bao gồm các cặp thẻ đánh dấu để định khuôn dạng của tài liệu. Tuỳ theo tên thẻ là gì mà trình duyệt sẽ tự động hiểu và làm các công việc do thẻ đó quy định. Chẳng hạn như cặp thẻ <B>….</B> quy định đoạn văn bản trong đó sử dụng chữ đậm. Vì vậy, trên thực tế người ta không coi nó là một ngôn ngữ (vì nó chẳng liên quan gì đến những thứ mà ta hay gặp trong lập trình như biến, câu lệnh rẽ nhánh, lặp…). Cũng chính vì nguyên nhân này, nó phải tự mở rộng bằng cách cho phép "nhúng" vào bản thân nó một số đoạn mã lệnh chương trình đặc biệt, người ta thường gọi chúng là các đoạn mã Script hay các đoạn mã nhúng . Ngôn ngữ sử dụng trong các đoạn mã lệnh đó gọi là các ngôn ngữ Script. Các ngôn ngữ script thường đơn giản và không có nhiều sức mạnh như các ngôn ngữ "kinh điển" cùng tên, hay nói cách khác, chúng là một phần rất nhỏ của một ngôn ngữ nào đó được tích hợp vào trình duyệt để thực hiện một số thao tác nhất định.

Lập trình Script ở máy khách

Như tên gọi của nó, lập trình script ở máy khách là viết các đoạn script chạy trên máy khách. Các đoạn mã này được máy chủ gửi kèm trong tài liệu, đưa về máy khách và được thực hiện ở đây.

Trong tài liệu gửi về trình duyệt, các đoạn mã này thường được tìm thấy trong cặp thẻ **<Script language ="xxxxxx">…</Script>**.
Có nhiều ngôn ngữ script phía máy khách. Nổi tiếng hơn cả là javascript. Kế đến là vbscript và PerlScript.

Vì tài liệu này chủ yếu tập trung vào PHP - một ngôn ngữ script chạy trên máy chủ, nên chi tiết những ngôn ngữ này không được nhắc đến trong tài liệu.

Lập trình Script ở máy chủ

Trái ngược với lập trình Script ở máy khách (thực thi mã lệnh ở máy khách), lập trình script ở máy chủ cho phép thực thi các đoạn mã ngay ở trên máy chủ. Không như các đoạn mã script hoạt động ở máy khách, các tài liệu có chứa các đoạn mã script phía máy chủ thường được lưu ở các file tài liệu có đuôi mở rộng riêng biệt, và các đoạn mã thi hành trên máy chủ cũng phải được đặt trong một cặp thẻ đặc biệt tuỳ theo quy định của chương trình xử lý. Chú ý rằng đối với mỗi loại ngôn ngữ server script sẽ có một chương trình xử lý riêng. Chẳng hạn các đoạn mã ASP thường được đặt trong các file *.asp, và chúng được xử lý bằng file ASP.dll. Chi tiết về cách thức hoạt động của loại này, có thể tóm tắt như sau:

- Bước 1: Client gửi yêu cầu đến máy chủ
- Bước 2: Web server kiểm tra xem yêu cầu đó cần loại tài liệu nào. Nếu đó là loại tài liệu có chứa các đoạn mã server script, nó sẽ triệu gọi chương trình xử lý tương ứng với loại tài liệu đó
- Bước 3: Chương trình xử lý sẽ thực thi các đoạn mã server script trong tài liệu đó, và trả kết quả (thường là dưới khuôn dạng HTML) về cho web server.
- Bước 4: Web server trả kết quả tìm được cho Client và ngắt kết nối.

Lịch sử phát triển các ứng dụng trên Web server. ASP, JSP và PHP

Vài năm trước đây, con đường thực sự duy nhất để vận chuyển các dữ liệu động tới trang Web là kỹ thuật CGI (Common Gateway Interface). Các chương trình CGI cung cấp một sự liên hệ đơn giản để tạo các ứng dụng Web cho phép tiếp nhận các dữ liệu nhập vào, các yêu cầu truy vấn cơ sở dữ liệu từ phía người dùng và trả một vài kết quả về cho trình duyệt. Các chương trình CGI có thể được viết trên một vài ngôn ngữ, trong đó phổ biến nhất là Perl. Web server sử dụng CGI như là một cổng truy cập chặn giữa yêu cầu của người dùng và dữ liệu được yêu cầu. Nó sẽ được nạp vào bộ nhớ như một chương trình bình thường. Thông thường các web server sẽ chuyển các yêu cầu và triệu gọi chương trình CGI. Sau khi chương trình kết thúc, web server sẽ đọc dữ liệu trả về từ chương trình và gửi nó đến trình duyệt.
Nhược điểm lớn nhất của kỹ thuật CGI là nó hoạt động kém hiệu quả. Mỗi khi web server nhận một yêu cầu, một tuyến trình mới được tạo ra. Mỗi tuyến trình lại chứa trong nó các đoạn mã lệnh, dữ liệu… và không được chia sẻ lẫn nhau, do đó gây ra lãng phí bộ nhớ. Để khắc phục nhược điểm này, Microsoft và Netscape đã hợp tác và đưa ra một cải tiến đáng kể là chuyển chúng về dạng các file thư viện liên kết động (DLL), cho phép chia sẻ mã lệnh giữa các tuyến trình. Đây chính là các kỹ thuật ISAPI và NSAPI.

Các kỹ thuật dựa trên DLL không phải là đã hoàn thiện. Chúng vẫn còn một số vấn đề:

- Khi các thư viện nền tảng được gọi, nếu muốn thoát các ứng dụng này, ta phải tắt chương trình triệu gọi (Web server) và khởi động lại máy tính.

- Các thư viện cần được đặt trong các tuyến trình bảo vệ, tức là chúng cần phải được cảnh giác về cách sử dụng các biến chung hoặc các biến tĩnh.

- Nếu chương trình triệu gọi gây ra lỗi truy cập, nó có thể dẫn đến tình trạng server bị treo tắc tử.

- Và cuối cùng: khi đã được dịch ra các file DLL, công việc gỡ lỗi cũng như bảo trì mã lệnh trở nên vất vả hơn bao giờ hết.

Kỹ thuật Web mới nhất của Microsoft, kết hợp HTML, các đoạn Script, các thành phần xử lý phía server trong cùng một file, được gọi là ASP (Active Server Pages), với phiên bản mới nhất hiện nay là ASP.Net. ASP được triệu gọi bởi một thư viện liên kết động gắn với các Web server của Microsoft. Về bản chất, ta có thể coi ASP như là một ngôn ngữ thông dịch vậy. Một trang ASP có thể sử dụng HTML, JScript và vbscript. Qua các đoạn mã nhúng này, ASP có thể truy cập đến các thành phần phía server. Các thành phần này có thể được viết trên bất kỳ ngôn ngữ nào hỗ trợ các thành phần COM của Microsoft. Và đây chính là sức mạnh của ASP: Nó có thể làm được bất kỳ cái gì mà máy chủ có thể làm được với các thành phần COM. Sau khi được thi hành, ASP sẽ sản sinh ra một trang Web có khuôn dạng HTML và trả nó về cho Web server.

Một bất lợi lớn đối với ASP là nó chỉ có thể hoạt động trên các họ Web server của Microsoft (bao gồm PWS trên Win9x hay IIS trên WinNT/2000/XP). Các nhà phát triển đang hướng đến những môi trường khác như Unix/Linux (hiện đã có bản Chili! ASP chạy trên các môi trường này), nhưng kết quả thì còn phải đợi thêm một thời gian nữa

Trước khi đi vào tìm hiểu lịch sử của PHP, có lẽ chúng ta cũng phải nhắc đến một tên tuổi khác là Java Server Pages. hay JSP. Giống như ASP, trang JSP cho phép chứa HTML, các đoạn mã Java và các thành phần Java Bean và chúng sẽ thực hiện các công việc để sản sinh ra một trang Web để gửi về Client. Bất lợi chính của loại này là phải đi kèm với "máy ảo Java", vốn không được coi là nhanh về mặt tốc độ.

1.1.2 Giới thiệu ngôn ngữ PHP

PHP - viết tắt của **PHP Hypertext Preprocessor** - một định nghĩa đệ quy khó hiểu!
Vào khoảng năm 1994, Rasmus Lerdorf đưa một số đoạn Perl Script vào trang Web để theo dõi xem ai đang đọc tài liệu của ông ta. Dần dần, người ta bắt đầu thích các đoạn Script này và sau đó đã xuất bản một gói công cụ có tên là "Personal Home Pages" (nghĩa đầu tiên của PHP). Ông ta đã viết một cơ chế nhúng và kết hợp với một số công cụ khác để phân tích đầu vào từ các mẫu biểu HTML: FI, Form Interpreter hay Phiên dịch mẫu biểu, được tạo ra theo cách đó và được đặt tên là PHP/FI hay PHP2. Nó được hoàn thành vào khoảng giữa năm 1995.

Sau đó, người ta bắt đầu sử dụng các công cụ này để xây dựng những thứ rắc rối hơn, và đội ngũ phát triển đã thay đổi từ một người duy nhất thành một nhóm các nhà phát triển nòng cốt trong dự án, và nó đã được tổ chức hoá. Đó là sự bắt đầu của PHP3. Đội ngũ các nhà phát triển (Rasmus Lerdorf, Andi Gutmans, Zeev Suraski, Stig Bakken, Shane Caraveo và Jim Winstead) đã cải tiến và mở rộng bộ máy nhúng và bổ sung thêm một số hàm API đơn giản cho phép các lập trình viên khác tự do bổ sung nhiều tính năng vào ngôn ngữ bằng cách viết các module cho nó. Cấu trúc của ngôn ngữ đã được tinh chế, được kết cấu thân thiện hơn đối với những người đến từ các ngôn ngữ hướng đối tượng hay các ngôn ngữ hướng thủ tục. Nếu bạn đã biết một vài ngôn ngữ lập trình khác thì khi đến với PHP, bạn sẽ không cảm thấy khó khăn.

Phiên bản mới nhất cho đến thời điểm này là PHP 5.0.1. Các bạn có thể tham khảo chi tiết tại trang web PHP: Hypertext Preprocessor

Như vậy PHP là kịch bản trình chủ (Server Script) được chạy trên nền PHP Engine, cùng với ứng dụng Web Server để quản lý chúng. Web Server thường sử dụng là IIS, Apache Web Server, ..

1.2 Hướng dẫn cài đặt APACHE, PHP, MYSQL

Chúng ta có 2 sản phẩm rất nổi tiếng là **XAMPP** và **WAMP**. Tuy nhiên với mục đích cài đặt trên **localhost** để chạy thử thì **WAMP** chính là lựa chọn đầu tiên và dễ dàng.

WAMP: Một gói phần mềm Web Server tất cả trong một (All-in-One) gồm: **Apache, MySQL, PHP** chạy trên nền Windows.

Các đặc điểm nổi trội của WAMP

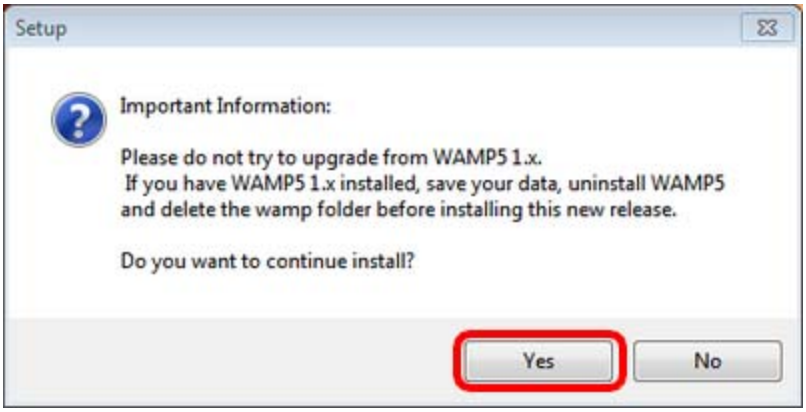
- Có thể cài đặt WAMP dễ dàng.
- WAMP được cập nhật đều đặn.
- Cho phép lựa chọn các phiên bản PHP, MySQL khác nhau.
- Rất tốt cho việc tạo máy chủ Web để chạy thử, thiết kế Website bằng PHP.
- Hỗ trợ phiên bản PHP5 mới nhất
- Tương thích Windows XP / Windows Vista / Windows 7 / Windows Server.
- Hoàn toàn miễn phí

Download bản cài đặt WAMP

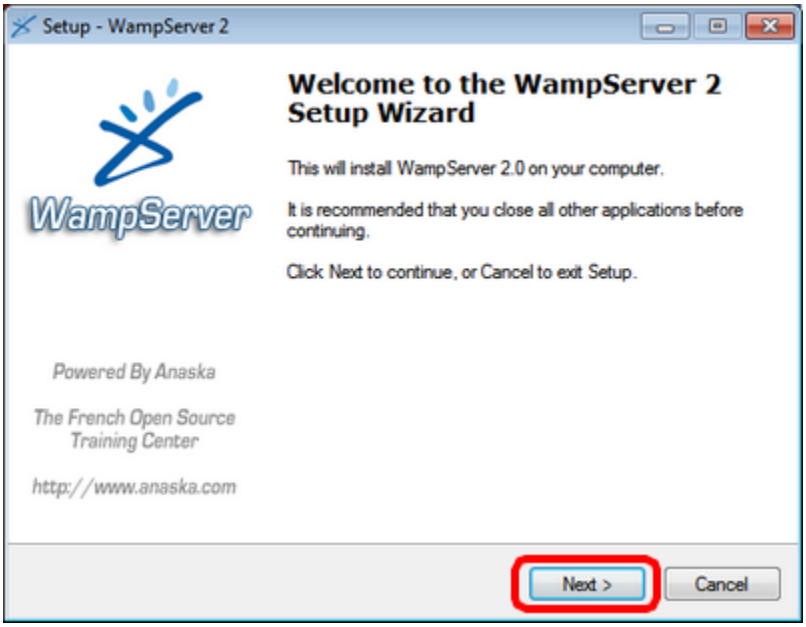
- Download WAMP tại địa chỉ http://www.wampserver.com/en/download.php

Các bước cài đặt WAMP

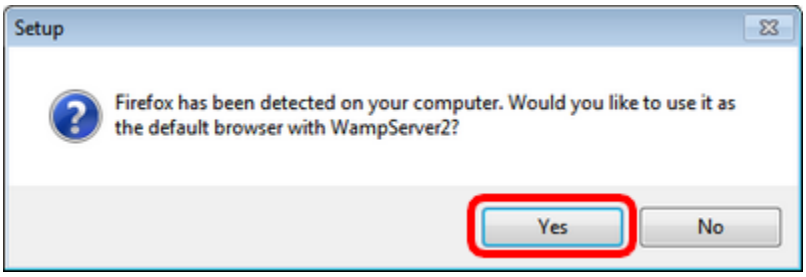
Sau khi download các bạn sẽ có file cài đặt tên là **WampServer2.0i.exe**
Các bạn click đúp vào file đó sẽ hiển thị như sau:



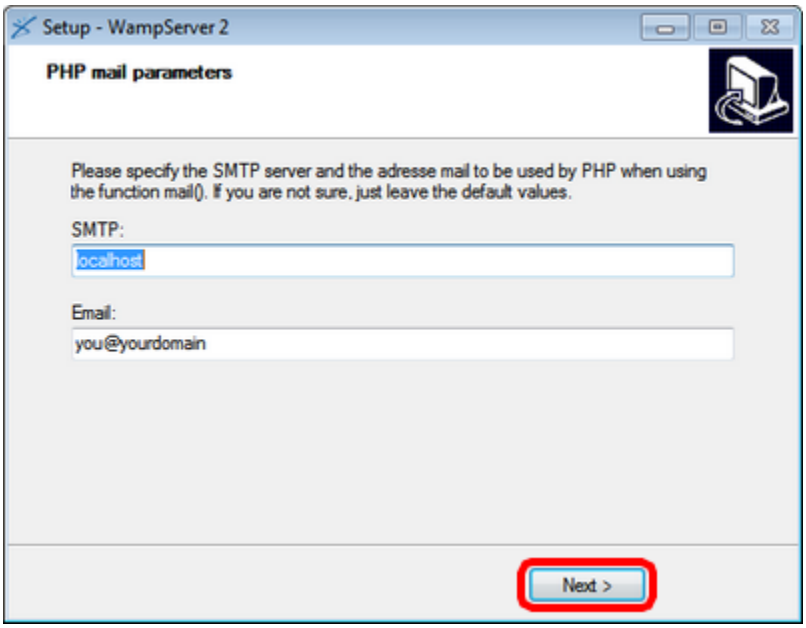
Nếu trước đó bạn đã cài đặt WAMP 5.1.x thì tốt nhất là gỡ nó đi sau đó mới cài WAMP Server 2.0



*Nhấn [**Next**] và thực hiện các bước tiếp theo*



Nếu trên máy của bạn có cài trình duyệt Firefox và
*bạn muốn chọn Firefox làm trình duyệt mặc định khi mở http://localhost thì nhấn [**Yes**]*

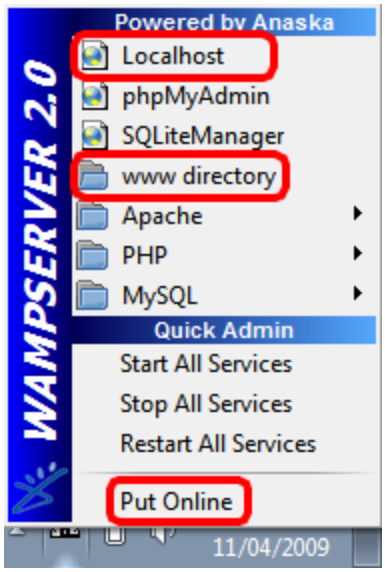


Điền các thông số để có thể gửi/nhận mail thông qua một SMTP server.
*Nếu không biết hãy để mặc định và nhấn [**Next**]*

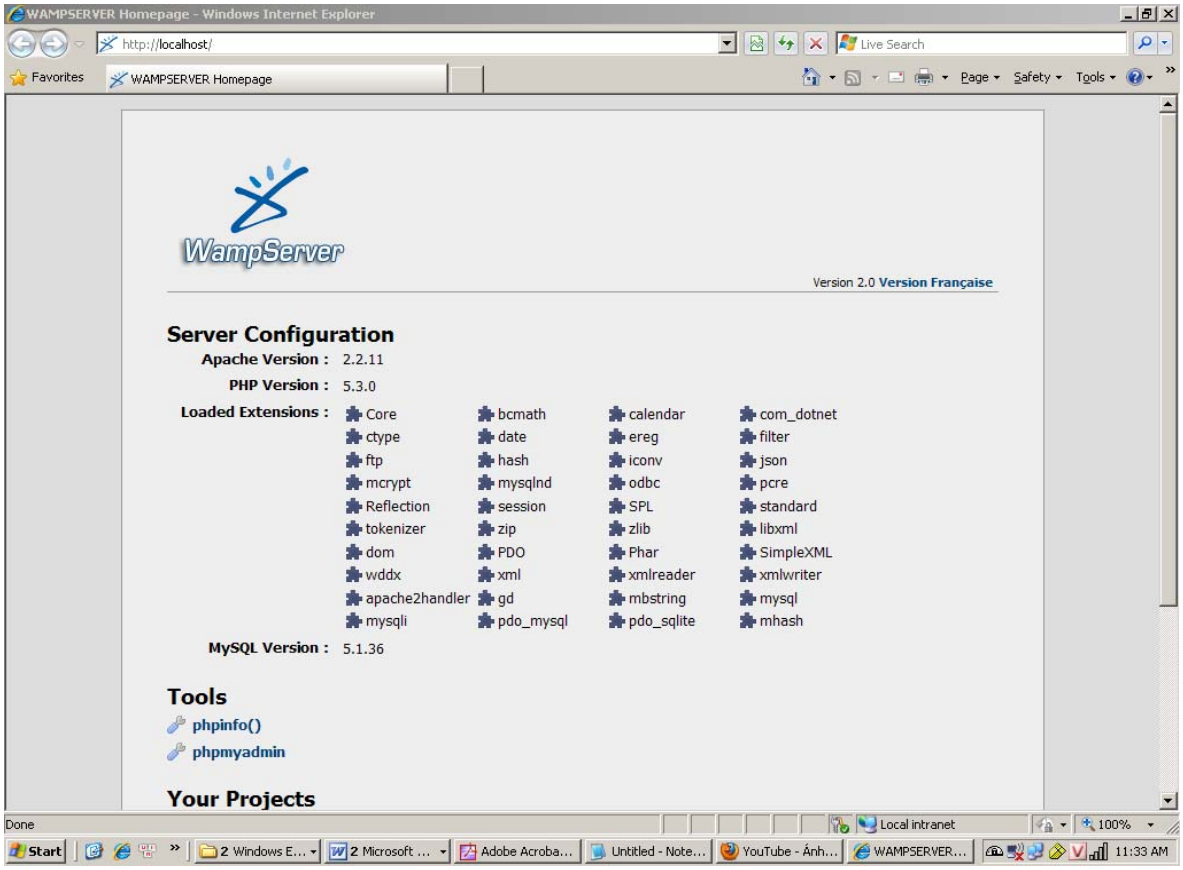
*Nhấn nút [**Finish**] để hoàn tất quá trình cài đặt*

Hoàn tất việc cài đặt và chạy thử

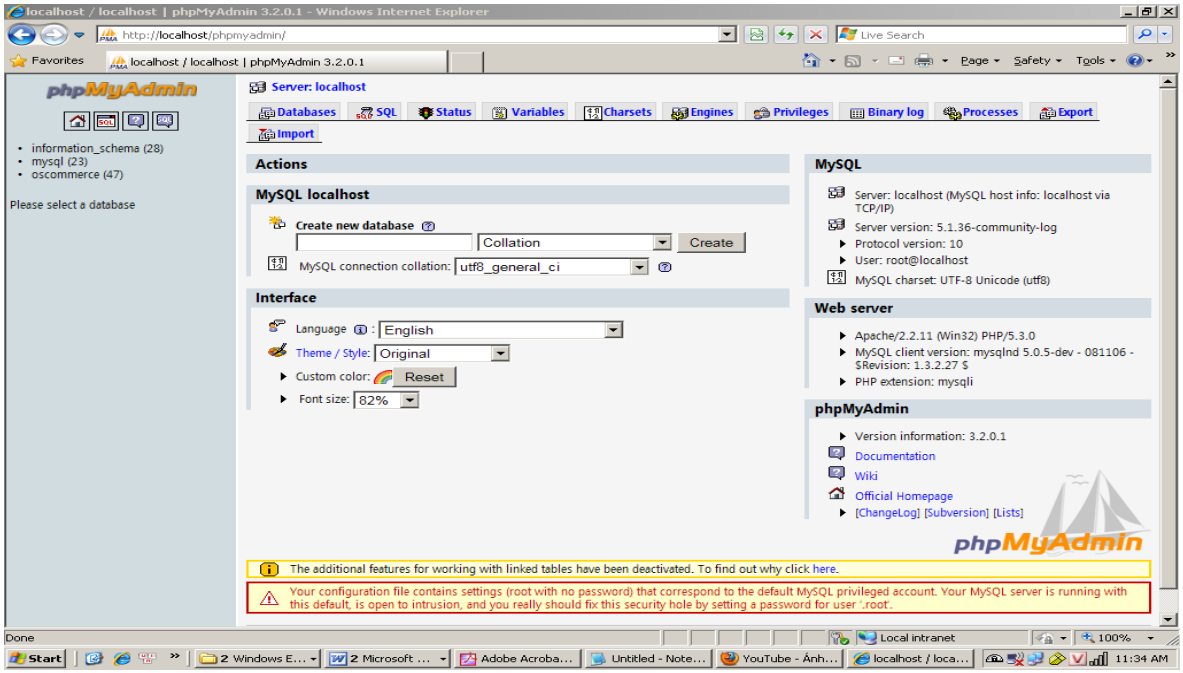
Sau khi cài đặt thành công, bạn sẽ thấy biểu tượng của WAMP ở góc màn hình như dưới đây :



Để chạy thử, nhấn vào mục "**localhost**"
Hoặc mở trình duyệt và gõ địa chỉ **http://localhost** (hoặc http://127.0.0.1)



Để quản lý cơ sở dữ liệu (database) nhấn vào mục "**phpMyAdmin**"
Hoặc mở trình duyệt và gõ vào địa chỉ **http://localhost/phpMyAdmin**
Hoặc **http://127.0.0.1/phpMyAdmin**



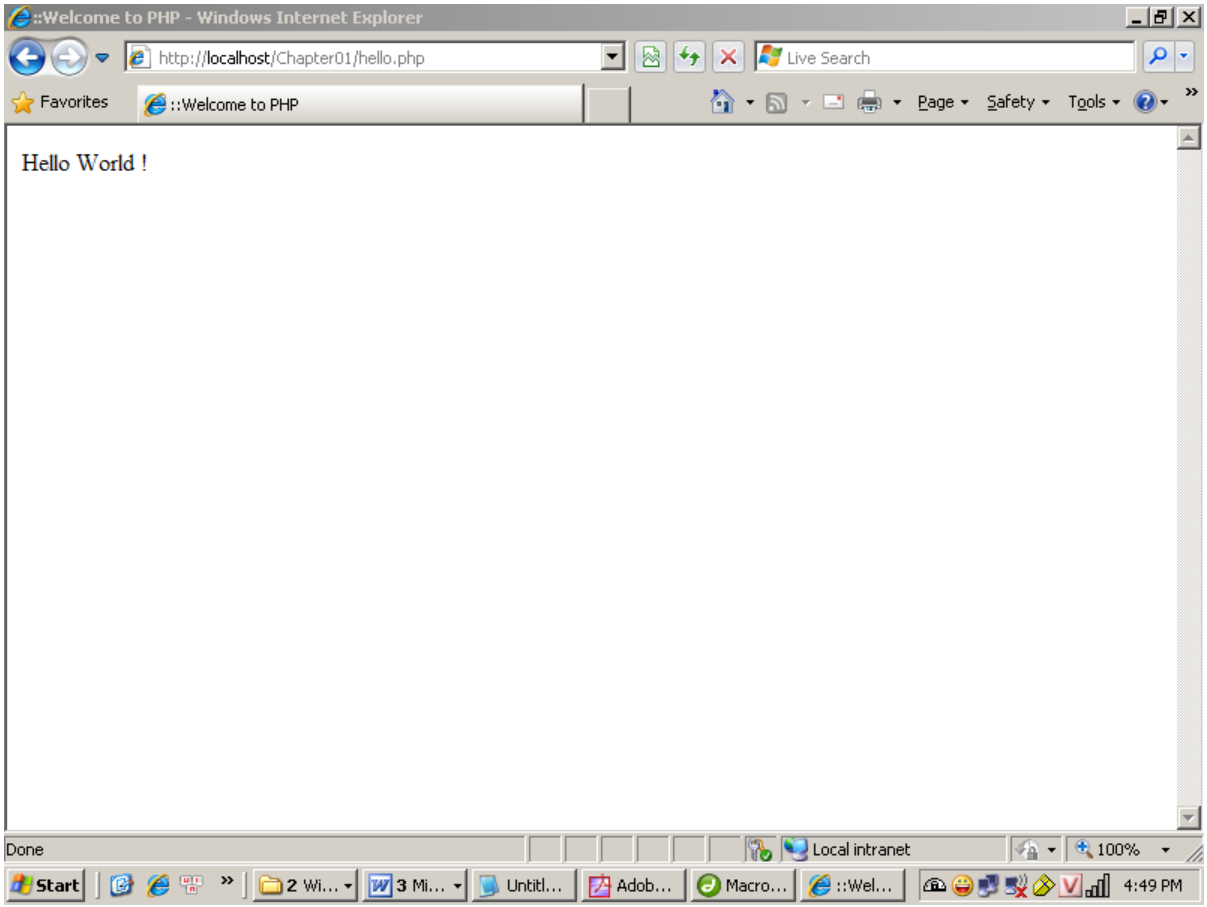
1.3 Kịch bản PHP (script)

Nội dung của PHP có thể khai báo lẫn lộn với HTML, chính vì vậy bạn sử dụng cặp thẻ **<?php ?>** để khai báo mã PHP. Chẳng hạn bạn khai báo trang hello.php với nội dung như ví dụ 1-1 sau:

Ví dụ : Trang hello.php

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
   Hello , <?php echo " World !" ?>
</BODY>
</HTML>
```

Kết quả trả về như hình 1-1 khi triệu gọi trang này trên trình duyệt.



Hình 1-1: Kết quả trang hello.php

Kết quả cho ra là "Hello, World !". Ta thấy chữ " Hello," nằm trong tag lệnh PHP còn chữ " World !" thuộc về HTML.

1.4 Cú pháp căn bản

Trang PHP là 1 trang HTML có nhúng mã PHP

Để minh hoạ cho điều này, ta hãy xem qua một số ví dụ sau:

Ví dụ : lưu file sau lên đĩa với tên vd1.php và chạy thử:

```
<html><head><title>Testing page</title></head>
<body><?php echo "Hello, world!"; ?></body>
</html>
```

Bạn sẽ nhận được 1 trang HTML mà khi view source bạn xẽ nhận được nội dung như sau:

```
<html><head><title>Testing page</title></head>
<body>Hello, World!</body>
</html>
```

Ví dụ : lưu file sau lên đĩa với tên vd2.php và chạy thử:

```
<?php echo "<html><head><title>Testing page</title></head>
<body>Hello, world!</body>
</html>"; ?>
```

Bạn cũng nhận được 1 trang HTML có source là:

```
<html><head><title>Testing page</title></head>
<body>Hello, World!</body>
</html>
```

Như vậy có thể nhận xét rằng 1 trang PHP cũng chính là 1 trang HTML có nhúng mã PHP ở bên trong và có phần mở rộng là .php. Phần mã PHP được đặt trong thẻ mở <?php và thẻ đóng ?>.

Khi trình duyệt truy cập vào 1 trang PHP, server sẽ đọc nội dung file PHP lên, lọc ra các đoạn mã PHP, thực thi các đoạn mã đó, lấy kết quả xuất ra của các đoạn mã PHP thay thế vào chỗ ban đầu của chúng trong file PHP, cuối cùng server trả về kết quả cuối cùng là 1 trang nội dung HTML về cho trình duyệt.

Ở ví dụ 1-2 bên trên, server thực thi đoạn mã <?php echo "Hello, world!"; ?>, đoạn mã này sẽ xuất ra dòng chữ Hello, world!, dòng chữ này sẽ được server thay thế ngược lại vào vị trí của đoạn mã PHP và trả về kết quả cuối cùng cho trình duyệt:

```
<html><head><title>Testing page</title></head>
<body>Hello, World!</body>
</html>
```

Như vậy thì ta hoàn toàn có thể tạo ra 1 file **vd3.php** với nội dung như sau:

```
<html><head><title>Testing page</title></head>
<body>Hello, World!</body>
</html>
```

Và kết quả vẫn như mong đợi.

Lệnh echo dùng để xuất 1 chuỗi văn bản về trình duyệt

Ở các ví dụ bên trên, ta đã dùng 1 lệnh của PHP là lệnh echo. Lệnh này dùng để xuất 1 chuỗi văn bản về cho trình duyệt.

Ví dụ câu lệnh echo "Hello, world!"; trình duyệt sẽ nhận được chỗi văn bản Hello, world!.
Câu lệnh echo 1+2; sẽ trả về cho trình duyệt chỗi văn bản 3.
Và câu lệnh echo 1+2, "Hello, world!"; sẽ trả về trình duyệt chỗi 3Hello, world!.

Phân cách các lệnh bằng dấu chấm phảy (;)

Tương tự như các ngôn ngữ lập trình C hoặc Pascal, 1 câu lệnh của PHP được kết thúc bằng dấu chấm phảy (;). Ví dụ:
echo 1+2;
echo "Hello, world!";

Chú thích trong chương trình

Các chú thích không phải là mã chương trình, nhưng nó giúp ta ghi chú về 1 đoạn chương trình nào đó. Khi lập trình, bạn nên để các ghi chú vào trong chương trình để sau này khi đọc lại code, bạn sẽ nhanh chóng nắm bắt được nội dung và ý nghĩa của đoạn chương trình đã biết.

PHP cho phép ta ghi 2 loại chú thích:

- Chú thích trên 1 dòng (chú thích loại này chỉ có thể ghi trên 1 dòng mà thôi)
- Chú thích nhiều dòng (chú thích loại này có thể ghi dài bao nhiêu cũng được).

Chú thích 1 dòng được bắt đầu bằng // hoặc #, và những gì được ghi từ đó về sau là chú thích.
Chú thích nhiều dòng được bắt đầu bằng /* và kết thúc bằng */, những gì ở giữa là chú thích.
Ví dụ:

```
<?php
//Đây là chú thích 1 dòng, đoạn chương trình sau sẽ in ra chuỗi 123
echo 123;
#Đây cũng là chú thích 1 dòng, đoạn chương trình sau sẽ in ra chuỗi abc
echo "abc";
/*
Đây là chú thích nhiều dòng
Đoạn chương trình sau sẽ in ra chuỗi abc123
*/
echo "abc123";
?>
```

Ngoài ra, bạn cũng có thể sử dụng dấu # để khai báo ghi chú cho từng dòng

ví dụ khai báo sau là ghi chú:

```
<?php
# Khai báo biến để paging
$sotrang=$pagenumber;
$record=$rownumber;
$totalRows = 0;
$paging="";
?>
```

Để lập trình bằng ngôn ngữ PHP cần chú ý những điểm sau:

- Cuối câu lệnh có dấu ;
- Biến trong PHP có tiền tố là $
- Mỗi phương thức đều bắt đầu { và đóng bằng dấu }
- Khi khai báo biến thì không có kiểu dữ liệu
- Nên có giá trị khởi đầu cho biến khai báo
- Phải có chi chú (comment) cho mỗi feature mới
- Sử dụng // hoặc # để giải thích cho mỗi câu ghi chú
- Sử dụng /* và */ cho mỗi đoạn ghi chú
- Khai báo biến có phân biệt chữ hoa hay thường

1.5 Biến, Hằng số và Kiểu dữ liệu trong PHP

1.5.1 Biến và giá trị logic

Một biến trong PHP được bắt đầu bằng ký tự $ và đi theo ngay sau đó là tên của biến.
Ví dụ:

```
$a: biến có tên là a
$abc123: biến có tên là abc123
```

Biến trong PHP phân biệt chữ hoa và chữ thường. Tức $Abc và $abc là 2 biến hoàn toàn khác nhau.
Tên biến chỉ được bao gồm các ký tự chữ cái (a..z hoặc A...Z), chữ số (0...9) và ký tự gạch dưới (_); nhưng tên biến không được bắt đầu bằng ký tự gạch dưới hoặc chữ số. Các tên biến sau là không hợp lệ!

```
$_abc Không hợp lệ! bắt đầu bằng ký tự gạch dưới
$1abc Không hợp lệ! bắt đầu bằng chữ số
$nguyễn Không hợp lệ! tên biến có ký tự đặt biệc (ễ)
```

Cũng giống với C/C++, PHP không có khái niệm TRUE và FALSE.
Các giá trị TRUE đư ợc hiểu là những giá trị bằng 1 và giá trị FALSE là những giá trị bằng 0 hoặc xâu rỗng.

Khi sử dụng biến chúng ta không cần khai báo kiểu .

Ví dụ :

```
   $a = 1; // $a là một biến kiểu integer.
   $a = 1.2; // bây giờ $a là một biến kiểu double.
   $a = "A" ; // bây giờ $a lại là một biến kiểu string.
```

Nếu thực hiện phép toán giữa biến có kiểu số và kiểu string,PHP sẽ coi chuỗi là một dãy số như sau :

```
$str = "222B Baker Street";
```

Ta thấy biến $str có giá trị kiểu string, và nếu cộng số 3 với giá trị này thì :

```
$x = 3 + $str ;
// $x = 225
```

khi đó biến $x nhận được giá trị 255 vì PHP đã cộng 3 với ba số đầu.
Nhưng nếu ta in giá trị của biến $str thì

```
echo ($str);
// print : "222B Baker Street"
```

Chú ý rằng các phép toán giữa số và chuỗi chỉ đúng khi ký tự đầu của chuỗi là số .

Ta cũng có thể làm thay đổi kiểu giá trị của một biến bằng cách ép kiểu

```
$a = 11.2;
$a = (int) $a;
// biến $a có kiểu double
// bây giờ $a có kiểu integer ,giá trị là 11
$a = (double) $a;    // bây giờ $a lại có kiểu double, giá trị là 11.0
$b = (string) $a ;    // biến $b có kiểu string ,giá trị là "11"
```

Cũng phải biết rằng PHP tự động chuyển đổi kiểu rất tốt. Nếu thật sự cần thiết chúng ta mới phải dùng cách trên.

1.5.1.1 Các hàm làm việc với biến

+gettype() : hàm này trả lại kiểu của một biến nào đó. Giá trị trả về có thể là :

"integer"
"double"
"string"
"array"
"object"
"class"

"unknown type"

ví dụ :

```
if (gettype($user_input) == "integer")
{
  $age = $user_input;
}
```

+settype() : hàm này ép kiểu cho một biến nào đó. Nếu thành công hàm trả về giá trị 1 (true) ,ngược lại là 0 (false).

ví du :

```
$a = 7.5;
settype($a, "integer");
if (settype($a, "array")){
echo ("Conversion succeeded. ");
}else{
echo ("Conversion error. ");
}
```

+isset() và unset() : Hàm isset() kiểm tra một biến đã được gán giá trị hay chưa, hàm unset() sẽ giải phóng bộ nhớ cho một biến nào đó .

ví dụ :

```
$id = "323bb";
if (isset($id))
 {
   echo ("Dữ liệu đã được gán");
}
else
{
  echo ("Dữ liệu chưa được gán");
}
unset($id);
if(!isset($id))
 {
   echo ("Dữ liệu đã được giải phóng");
}
```

+empty() : Cũng giống hàm isset(), hàm empty() sẽ trả về giá trị 1 (true) nếu một biến là rỗng và ngược lại 0 (false). Đối với biến có kiểu số giá trị bằng 0 được coi là rỗng, biến kiểu string được coi là rỗng nếu xâu là xâu rỗng.

ví dụ:

```
echo empty($new) ;
$new = 1;
// true
echo empty($new); // false
$new = "";
echo empty($new); // true
$new = 0;
echo empty($new); // true
$new = "So 323";
echo empty($new); // false
```

```
unset($new);
echo empty($new); // true
```

1.5.1.2 Tầm vực (scope) của biến

Tầm vực của biến là ngữ cảnh mà ở trong đó biến được định nghĩa.

Ví dụ:

```
<?php
$a = 1; //tầm vực của biến $a bắt đầu từ đây
include 'b.php'; trải dài tới bên trong file b.php
//tới cuối file vẫn còn hợp lệ
?>
```

Tuy nhiên khi gặp 1 hàm do người dùng định nghĩa, bên trong hàm, biến cục bộ sẽ được dùng thay vì biến toàn cục. Ví dụ:

```
<?php
$a = 1; //biến toàn cục
//hàm do tự tạo
function test()
{
   echo $a;
}
 //end test
?>
```

Ở ví dụ trên, câu lệnh echo $a sẽ không in ra giá trị nào hết vì câu lệnh này nằm bên trong hàm test nên $a ở đây được hiểu là biến cục bộ $a của hàm (mà hàm này ta chưa khai báo biến cục bộ nào cả).

Để truy cập tới các biến toàn cục ở bên trong 1 hàm do người dùng định nghĩa, ta có thể dùng 1 trong 2 cách sau:

+Cách 1:

```
<?php
$a = 1; //biến toàn cục
//hàm do tự tạo
function test()
{
        //từ khoá global báo cho php biết là bên trong hàm test
        //bây giờ ta sẽ dùng biến toàn cục $a
        global $a;
        echo $a; //in ra giá trị: 1
} //end test
?>
```

+Cách 2:

```
<?php
$a = 1; //biến toàn cục
//hàm do tự tạo
```

```
function test()
 {
        echo $GLOBALS['a']; //in ra giá trị: 1
} //end test
?>
```

1.5.2 Hằng số

1.5.2.1 Khai báo và sử dụng hằng

Hằng là giá trị không thay đổi kể từ sau khi khai báo, bạn có thể sử dụng phát biểu Define để khai báo hằng như sau:

```
define("COMPANY", "Phop's Bicycles");
define("YELLOW", "#FFFF00");
define("VERSION", 4);
define("NL", "<BR>\n");
```

Trong ví dụ trên chúng ta đã dùng hàm define() để khai báo hằng số NL. Hằng số này là một thẻ ngắt dòng trong HTML.

Chúng ta sẽ sử dụng các hằng số trong PHP như sau :

```
echo ("Employment at ". COMPANY. NL);
```

Cách viết trên cũng giống như các viết sau:

```
echo ("Employment at Phop's Bicycles<BR>\n");
```

Chú ý : hằng số phải ở ngoài hai dấu " và ". Trờng hợp sau là không có hiệu lực :

```
echo ("Employment at COMPANY NL");
```

Khi thực hiện nó sẽ cho kết quả là : "Employment at COMPANY NL".

1.5.2.2 Kiểm tra hằng

Hàm defined() : hàm này dùng để kiểm tra xem một hằng số nào đó đã được khai báo chưa.

Ví dụ :

```
if ( defined ("YELLOW"))
   {

     echo ("<BODY BGCOLOR=". YELLOW. ">\n");
   }
```

1.5.3 Kiểu dữ liệu

PHP hỗ trợ 8 kiểu dữ liệu chính:

4 kiểu dữ liệu vô hướng: **boolean**, **integer**, **float** (**double**), **string**.
2 kiểu dữ liệu tổ hợp: **array**, **object**.

2 kiểu dữ liệu đặt biệc: **resource**, **NULL**.

1.5.3.1 Kiểu Boolean

Kiểu boolean mang 1 trong 2 giá trị TRUE (đúng) hoặc FALSE (sai).

Ví dụ:

```
<?php
$a = TRUE;
$b = FALSE;

//phép toán == kiểm tra xem 2 biểu thức có giá trị bằng nhau hay không
$c = (1==2); //vì 1 khác 2 nên $c mang giá trị FALSE
$d = ("abc" == "def"); //$d mang giá trị TRUE
?>
```

"Ép" kiểu sang boolean: một số giá trị được chuyển đổi thành FALSE trong các biểu thức boolean nếu như giá trị đó là:

+Số nguyên 0,
+Số thực 0.0,
+Chuỗi rỗng "", hoặc chuỗi "0",
+Mảng rỗng (không chứa phần tử nào) Array(),
+Đối tượng không chứa phần tử nào (chỉ đúng với PHP4),
+Giá trị NULL

Các giá trị còn lại sẽ được chuyển đổi thành TRUE.

1.5.3.2 Kiểu Integer

Kiểu integer mang các giá trị số nguyên ..., -2, -1, 0, 1, 2, ...Trên hầu hết các hệ thống, kiểu số nguyên có kích thước 32 bit, mang giá trị từ -2147483647 cho đến 2147483648. Ví dụ:

```
<?php
$a = 1234;
$b = -123;
$c = 0123; //giá trị 123 ở hệ cơ số 8, tương đương với 83 ở hệ cơ số 10
$d = 0x1F; //giá trị 1F ở hệ cơ số 16, tương đương với 31 ở hệ cơ số 10
?>
```

1.5.3.3 Kiểu Float (Double)

Kiểu float (hoặc double) là kiểu số thực, có thể mang bất cứ giá trị số thực nào. Trên hầu hết các hệ thống, kiểu số thực có kích thước 64 bit. Ví dụ:

```
<?php
$a = 1.234;
$b = 1.2e3; //= 1.2*10^3 = 1200
$c = 7E-10; //= 7*(10^-10) = 0.0000000007
$d = -1.23;
?>
```

1.5.3.4 Kiểu String

Kiểu string lưu giữ 1 chuỗi ký tự, mỗi ký tự có kích thước 1 byte. Nội dung string được đặt giữa 2 dấu nháy, nháy đơn (') hoặc nháy kép (").

Ví dụ

```
<?php
$a = 'Đây là 1 chuỗi được đặt giữa dấu nháy đơn';
$b = "Đây là 1 chuỗi được đặt giữa dấu nháy kép";
$c = 'Đây là 1 chuỗi được đặt giữa dấu nháy đơn với "vài dấu nháy kép ở
giữa"';
$d = "Đây là 1 chuỗi được đặt giữa dấu nháy kép với 'vài dấu nháy đơn ở
giữa'";
?>
```

Nếu bạn muốn sử dụng dấu nháy đơn ở trong 1 chuỗi được bọc bởi dấu nháy đơn, hoặc sử dụng dấu nháy kép đặt giữa chuỗi được bọc bởi dấu nháy kép thì bạn để thêm ký tự \ (gọi là ký tự escape) ở phía trước.

Ví dụ:

```
<?php
$a = 'Dấu \'nháy đơn\' ở giữa chuỗi'; //$a mang giá trị: Dấu 'nháy đơn' ở
giữa chuỗi
$b = "Dấu \"nháy kép\" ở giữa chuỗi";  //$b mang giá trị: Dấu "nháy kép" ở
giữa chuỗi
$c = "Dùng ký tự \\ ở giữa câu \\ thì sao?";  //$c mang giá trị: Dùng ký tự
\ ở giữa câu \ thì sao?
?>
```

Khi sử dụng dấu nháy đôi để bọc chuỗi, ngoài \', \" và \\, PHP có thể nhận dạng thêm một số chuỗi ký tự escape đặct biệc nữa:

\n: ký tự xuống hàng LF (ký tự có mã 10 trong bảng mã ASCII)
\r: ký tự về đầu dòng CR (ký tự có mã 13 trong bảng mã ASCII)
\t: ký tự tab (ký tự có mã 9 trong bảng mã ASCII)
\$: ký tự $
\ooo: (với o là 1 chữ số từ 0 đến 7) biểu thị 1 ký tự có mã ASCII ooo trong hệ cơ số 8.
Ví dụ \101 sẽ là ký tự 'A' (101 trong hệ cơ số 8 tương đương 65 trong hệ cơ số 10, ký tự ASCII có mã 65 chính là ký tự 'A').
\xhh: (với h là 1 chữ số từ 0 đến 9 hoặc 1 chữ cái từ A tời F) biểu thị 1 ký tự có mã ASCII hh trong hệ cơ số 16.
Ví dụ \0x41 sẽ là ký tự 'A' (41 trong hệ cơ số 16 chính là 65 trong hệ cơ số 10).

Ngoài ra, nếu bạn để 1 biến vào giữa 1 chuỗi được bọc với dấu nháy kép, giá trị của biến sẽ được thay thế vào trong chuỗi. ví dụ:

```
<?php
$a = 1;
$b = 2;
$c = 3;
$d = "$a $b $c"; //$d sẽ mang giá trị là chuỗi "1 2 3"
?>
```

1.5.3.5 Kiểu Array

Array là một mảng gồm nhiều phần tử. Array được tạo qua lệnh Array. Ví dụ:

```
<?php
$a = Array(1,2,3);?>
```

Lúc này $a sẽ là 1 mảng gồm 3 phần tử số nguyên là 1, 2 và 3
Các phần tử trong mảng $a được tạo ở trên sẽ được đánh số thứ tự từ 0, 1 cho đến 2
Để truy cập tới từng phần tử của $a

```
echo $a[0]; //in ra giá trị 1
echo $a[2]; //in ra giá trị 3

$a[1] = 5; //giờ đây $a = Array(1,5,3)
```

Mảng còn có thể được tạo thành bởi các cặp (khoá, giá trị).

Ví dụ :

```
<?php
$a = Array(
        "khoá 1" => "giá trị 1",
        "khoá 2" => "giá trị 2",
        "khoá 3" => "giá trị 3"
);
echo $a["khoá 1"]; //in ra: giá trị 1

$b = Array(
        "a" => "Nguyễn",
        "b" => "Văn",
        "c" => "Tuấn"
);
echo $b["a"]; //in ra: Nguyễn

$b["a"] = "Nguyen";
$b["b"] = "Hung";
$b["c"] = "Cuong";
//giờ đây $b = Array("a" => "Nguyen", "b" => "Hung", "c" => "Cuong")
?>
```

1.5.3.6 Kiểu Object

Kiểu object (đối tượng) lưu giữ 1 bản thể (instance) của 1 lớp (class). Ta sẽ tìm kiểu kỹ thêm về kiểu object trong phần Lập trình hướng đối tượng với PHP.

1.5.3.7 Kiểu Resource

Kiểu resource (tài nguyên) được sử dụng bởi các hàm đặt biệc của PHP (ví dụ hàm mysql_connect sẽ trả về kiểu resource). Ta sẽ tìm hiểu kỹ hơn về kiểu resource trong các bài viết khác.

1.5.3.8 Kiểu NULL

Đây là 1 giá trị đặt biệc, báo cho PHP biết rằng 1 biến nào đó chưa/không mang giá trị nào cả. Ví dụ:

```
<?php
$a = 1; //$a mang giá trị 1
$a = NULL; //bây giờ $a không mang giá trị nào cả
 $a = 2; //giờ đây $a mang giá trị 2
//hàm unset sẽ làm cho 1 biến có giá trị là NULL
unset($a); //giùo $a lại là NULL
?>
```

1.6 Toán tử và biểu thức

1.6.1 Toán tử

Toán tử kết hợp các giá trị hoặc biểu thức lại với nhau và tạo ra một giá trị mới.
Các toán tử trong PHP được chia thành 3 nhóm:

+ Các toán tử áp dụng trên 1 giá trị, ví dụ như toán tử ++ hoặc --
+ Các toán tử kết hợp 2 hoặc nhiều giá trị, ví dụ như toán tử +, -, *, /
+ Toán tử ?: dùng để chọn 1 trong 2 giá trị tuỳ thuộc vào 1 điều kiện cho trước

1.6.1.1 Bảng các phép toán số học

Phép toán	Ý nghĩa	Ví dụ	Giải thích
+	Phép cộng	7 + 2	Thực hiện phép cộng giữa 7 và 2 : 9
-	Phép trừ	7 – 2	Thực hiện phép trừ giữa 7 và 2 : 5
*	Phép nhân	7 * 2	Thực hiện phép nhân giữa 7 và 2 : 14
/	Phép chia	7 / 2	Thực hiện phép chia giữa 7 và 2 : 3.5
%	Chia d	7 % 2	Thực hiện phép chia d giữa 7 và 2 : 1

Ta có thể viết các phép toán ngắn gọn như bảng sau :

Khi viết	Tương đương với
$h += $i	$h = $h + $i
$h -= $i	$h = $h - $i
$h *= $i	$h = $h * $i
$h /= $i	$h = $h / $i
$h %= $i	$h = $h % $i

1.6.1.2 Phép gán :

ví dụ :

```
$x = 1;
$y = $x + 1;
$length = $area / $width;
```

1.6.1.3 Bảng các phép toán quan hệ

Phép toán	Ý nghĩa	Ví dụ	Giải thích
==	So sánh bằng	$h ==$i	Kiểm tra $h có bằng $i nhau không
<	So sánh nhỏ hơn	$h < $i	Kiểm tra $h có nhỏ hơn $i không
>	So sánh lớn hơn	$h > $i	Kiểm tra $h có lớn hơn $i không
<=	Nhỏ hơn hoặc bằng	$h <=$i	Kiểm tra $h có nhỏ hơn hoặc bằng $i
>=	Lớn hơn hoặc bằng	$h =>$i	Kiểm tra $h có lớn hơn hoặc bằng $i
!=	So sánh khác	$h != $i	Kiểm tra $h có khác $i không
<>	So sánh khác	$h <>$i	Kiểm tra $h có khác $i không

Các phép so sánh thường dùng kiểm tra điều kiện trong các câu lệnh điều khiển mà ta sẽ học ở bài sau.

1.6.1.4 Bảng các phép toán logic

Phép toán logic cùng với toán hạng tạo thành biểu thức logic. Biểu thức logic có thể có giá trị là 1 (true) hoặc 0 (false).

Toán hạng a	Toán hạng b	a && b	a || b	!a	!b
1	1	1	1	0	0
1	0	0	1	0	1
0	1	0	1	1	0
0	0	0	0	1	1

1.6.1.5 Các phép toán với biến kiểu string

Ta sử dụng dấu "**.** " để ghép hai biến kiểu string với nhau.

ví dụ :

```
$first = "Phineas";
$last = "Phop";
$full = $first. " ". $last;
echo ($full);
// $full = "Phineas Phop" ;
```

Ta có thể ghép hai xâu như sau:

```
echo ($last. "'s Bicycles");
//print : Phop's Bicycles
```

Để có thể chèn một biến vào trong hàng có kiểu string thì tên biến
phải để trong dấu đóng mở ngoặc nhọn.

```
echo ("${last}'s Bicycles");
```

1.6.1.6 Các phép toán tăng giảm

Phép tăng : phép tăng (toán tử tăng) tăng giá trị của toán hạng lên một đơn vị.

$a ++ : $a được sử dụng rồi mới tăng
++ $a : $a tăng rồi mới được sử dụng

Phép giảm : tương tự như phép tăng, khác là giá trị bị giảm đi một đơn vị.

$a -- : $a được sử dụng rồi mới giảm
-- $a : $a giảm rồi mới được sử dụng

ví dụ :

```
$a = 10;
// $a bằng 10
$b = $a++ ;  // $a bằng 11 nhưng $b bằng 10
$a = 10;
// $a bằng 10
$b = -- $a ; // $a bằng 9 và $b bằng 9
```

1.6.1.7 Phép toán điều kiện. ?

Phép toán điều kiện cùng với toán hạng tạo nên biểu thức điều kiện. Ta ký hiệu e1 ,e2, e3 là ba toán hạng.

Biểu thức có dạng : e1 ? e2 : e3
Nếu e1 != 0 thì giá trị của biểu thức điều kiện là e2
Nếu e1 == 0 thì giá trị của biểu thức điều kiện là e3

ví dụ : tìm max

```
max = $a>$b ? a : b ;
```

1.6.1.8 Toán tử sizeof (đối tượng)

Phép toán sizeof cho biết kích thước (tính bằng byte) ô nhớ mà đối tượng chiếm trong bộ nhớ. Đối tượng ở đây có kiểu là integer, double, string.

ví dụ :

```
$a = 10;
echo sizeof($a);       //sẽ in ra màn hình là : 4
```

1.6.1.9 Thứ tự ưu tiên của toán tử

Các toán tử khác nhau có thể có độ ưu tiên khác nhau. Trong cùng 1 biểu thức có nhiều toán tử, toán tử nào có độ ưu tiên cao hơn sẽ được thực hiện trước (trừ khi bạn nhóm các biểu thức lại bằng dấu ngoặc (và)). Nếu trong biểu thức có 2 toán tử có cùng độ ưu tiên thì qui tắc liên kết của từng toán tử sẽ qui định thứ tự thực hiện của các toán tử đó.
Sau đây là bảng liệt kê các toán tử cùng thứ tự ưu tiên của chúng (toán tử có độ ưu tiên cao hơn được liệt kê bên trên, các toán tử có độ ưu tiên thấp hơn được lệt kê bên dưới).

Qui tắc liên kết	Toán tử	Ghi chú
	new	Tạo 1 đối tượng từ 1 class, toán tử này chỉ áp dụng trên 1 toán hạng nên không có qui tắc liên kết
Bên phải trước	[	Toán tử truy cập 1 phần tử trong mảng
	++ --	Tăng/Giảm 1 đơn vị, toán tử này chỉ áp dụng trên 1 toán hạng nên không có qui tắc liên kết
	! ~ - (int) (float) (string) (array) (object) @	Các toán tử này chỉ áp dụng trên 1 toán hạng nên không có qui tắc liên kết
Bên trái trước	* / %	
Bên trái trước	+ - .	
Bên trái trước	<< >>	
	== != === !===	Toán tử so sánh, chỉ áp dụng trên 2 toán hạng nên không có qui tắc liên kết
Bên trái trước	&	
Bên trái trước	^	
Bên trái trước	|	
Bên trái trước	&&	
Bên trái trước	||	
Bên trái trước	? :	
Bên phải trước	= += -= *= /= .= %= &= |= ^= <<= >>=	
Bên trái trước	and	
Bên trái trước	xor	
Bên trái trước	or	
Bên trái trước	,	

1.6.2 Biểu thức

Biểu thức là nền tảng quan trọng của PHP. Hầu như mọi thứ bạn ghi trong file php đều là biểu thức. Nói một cách đơn giản, bất cứ cái gì mang 1 giá trị nào đó đều có thể là 1 biểu thức.

Ta xét câu lệnh đơn giản sau:

```
$a = 5;
```

Ở đây 5 là một biểu thức, kết của của biểu thức này là giá trị 5, và kết quả này được gán cho biến $a.

```
$b = $a;
```

Ở đây $a lại là 1 biểu thức, giá trị của $a được gán cho biến $b.

Biểu thức trong PHP có thể phức tạp hơn thế.

ví dụ:

```
$a = 1;
$b = 2;
$c = 3;
$d = $a + $b + $c;
```

1.7 Phát biểu có điều kiện và vòng lặp

1.7.1 Phát biểu If_Else

Đây là lệnh rẽ nhánh có điều kiện.

Dạng 1

if (điều kiện) { câu lệnh; };

Câu lệnh ở đây tương đương với một khối lệnh. Một khối lệnh được đặt trong dấu ngoặc kép.

ý nghĩa :

+ Nếu biểu thức khác không ,thì câu lệnh được thực hiện.
+ Nếu biểu thức bằng không, thì câu lệnh không được thực hiện

Ví dụ:

```
<?php
$b=true;
$j=3;
if(($j>=3) &&($b!=true))
  echo "result is true";
if(($j<3) ||($b==true))
  echo "result is false";
?>
```

Dạng 2

if (biểu thức)
{
** câu lệnh 1;**
}
else
{
** câu lệnh 2;**
}

ý nghĩa :

+ Nếu biểu thức khác không ,thì câu lệnh 1 được thực hiện.

+ Nếu biểu thức bằng không, thì câu lệnh 2 được thực hiện

Chú ý :

* Câu lệnh 1 ở dạng 2 là lệnh if_else

Ví dụ:

```
<?php
$j=3;
if ($j>3)
  echo "result is true";
else
{
  $j++;
  echo "result is $j";
}
?>
```

Nếu lượng else bằng lượng If thì else thuộc về If gần nhất theo từng cặp từ trong ra ngoài.

Ví dụ :

```
$a = 10;
$b = 10;
$c = 3;
$d = 3;
$e = 12;
$f = 8;
if($a == $b)
if($c == $d)
     if($e == $f)
   $max = $e;
     else
   $max = $f;
else
$max = $d;
else
$max = $b

echo $max ;

//printf max = 8
```

Nếu lượng else ít hơn lượng If thì else thuộc về If ngần nhất theo từng cặp từ trong ra ngoài.

Ví dụ :

```
<?php
if ($a == $b)
if ($c == $d)
$max = 0
else
$max = $d;
?>
```

Tương đương với :

```
<?php
if ($a == $b)
{
  if ( $c == $d)
     $max = 0;
}
else
$max = $b;
?>
```

* Câu lệnh 2 của dạng 2 là elseif :

Bắt nguồn từ :

```
if ( biểu_thức1 )
 {
  câu_lệnh 1;
 }
else
 {
  if ( biểu_thức 2 )
    {
     câu_lệnh 2;
    }
  else
   {
    if ( biểu_thức 3 )
      {
       câu_lệnh 3;
      }
    ...
    else
      {
       if ( biểu_thức i )
       {
         câu_lệnh i;
       }
       ...
       else
       {
```

```
        câu_lệnh n;
      }
     }
   }
}
```

Có thể viết lại như sau:

```
if ( biểu_thức1 )
 {
  câu_lệnh 1;
 }
elseif (biểu_thức 2)
 {
  câu_lệnh 2;
 }
elseif (biểu_thức 3)
  {
    câu_lệnh 3;
   }
     ...
elseif (biểu_thức i)
    {
     câu_lệnh i;
    }
     ...
else
    {
     câu_lệnh n;
    }
```

Câu lệnh elseif tạo ra lệnh rẽ nhánh có điều kiện trong đó thực hiện 1 trong n cách khác nhau.

+ Nếu biểu_thức i khác không (i = 1,..n-1) thì thực hiện câu lệnh i .
+ Nếu biểu_thức i bằng không (i = 1, ..n-1) thì câu lệnh thứ n được thực hiện.

Ví dụ:

```
<?php
$j=3;
if ($j>3)
  echo "result is true";
elseif ($j==0)
{
  $j++;
  echo "result is ".$j;
}
else
{
  $j--;
  echo "result is ". $j;
```

```
}
?>
```

1.7.2 Phát biểu Switch

Phát biểu switch là phần của phát biểu elseif nhiều nhánh, khi có nhiều điều kiện chọn lựa thì bạn sử dụng switch, cú pháp của chúng như sau:

switch (biểu_thức n)
{
case n1:
câu lệnh 1;
break;
case n2:
câu lệnh 2;
break;
...
case nn:
câu lệnh nn;
default:
câu lệnh default;
}

Break: dùng để thoát ra khỏi **switch** khi thoả một **case** nào đó trong **switch**, **default**: khi không có bất kỳ giá trị nào thoản trong các **case** thì giá trị cuối cùng là **defaule** statement.

Ví dụ:

```
<?php
$j=3;
$j=date("w");
$str="";
switch($j)
     {
  case 0:
    $str="Today is  Sunday";
    break;
  case 1:
    $str="Today is Monday";
    break;
  case 2:
    $str="Today is Tueday";
    break;
  case 3:
    $str="Today is Wednesday";
    break;
  case 4:
    $str="Today is Thursday";
    break;
  case 5:
    $str="Today is Friday";
    break;
```

```
  case 6:
    $str="Today is Saturday";
    break;
  default:
    $str="Today is Sunday";
    break;
}
echo $str;
?>
```

1.7.3 Phát biểu While

while (biểu_thức)
{
câu lệnh ;
}

Lệnh while là một lệnh tạo chu trình có điều kiện. Điều kiện thực hiện được kiểm tra ở đầu chu trình.

+ Bước 1 : Tính biểu thức
Nếu biểu thức khác không, sang bước 2
Nếu biểu thức bằng không, kết thúc vòng while

+ Bước 2 : Thực hiện câu lệnh.
Quay lại bước 1.

Ví dụ:

```
<?php
$j=10;
while($j>0)
{
  echo $j."<br>";
  $j--;
}
?>
```

Chú ý :

+ Biểu thức có thể bao gồm nhiều biểu thức. Khi đó các biểu thức được viết cách nhau một dấu phẩy ,và được tính lần lượt từ trái qua phải. Biểu thức cuối cùng quyết định thực hiện câu lệnh.

+ Không được phép vắng mặt biểu thức

+ Để tạo chu trình vô tận thì

```
while(1)
{
   …
```

```
   if (biểu_thức) break;
   …
}
```

Ví dụ:

```
<?php
$i = 11;
while (--$i)
{
  if (my_function($i) == "error")
    {
      break;
    }
   ++ $number;
}
?>
```

1.7.4 Phát biểu do while

Phát biểu do while cho phép duyệt và kiểm tra điều kiện sau phát biểu thứ nhất, điều này có nghĩa là ít nhất một phát biểu được thực hiện. cú pháp có dạng như sau:

```
do
{
 câu lệnh;
}
while (biểu_thức);
```

Hoạt động :

+ Bước 1 : Thực hiện câu lệnh

+ Bước 2 : Tính biểu thức biểu_thức
Nếu biểu thức biểu_thức khác không thì quay lại bước 1
Nếu biểu thức biểu_thức bằng không thì kết thúc do ... while.

Ví dụ:

```
<?php
$j=10;
do
{
   echo $j."<br>";
   $j--;
}
while($j>0)
?>
```

1.7.5 Phát biểu For

Phát biểu for dùng cho vòng lặp có giới hạn cho trước, cú pháp có dạng như sau:

for (biểu_thức_1; biểu_thức_2; biểu_thức_3)
{
câu lệnh ;
}

Ví dụ:

```
<?php
for($j=1;$j<=10;$j++)
{
  echo $j."<br>";
}
?>
```

Lệnh for là lệnh tạo chu trình có điều kiện.

+ Bước 1 : tính biểu_thức_1
+ Bước 2 : tính biểu_thức_2 :

Nếu biểu_thức_2 khác 0 thì thực hiện câu lệnh và sang Bước 3.
Nếu biểu_thức_2 bằng 0 thì kết thúc vòng for

+ Bước 3 : tính biểu_thức_3 và quay lại bước 2.

* Biểu_thức_1, biểu_thức_2, biểu_thức_3 là các thành phần. Mỗi thành phần có thể gồm nhiều biểu thức. Khi đó mỗi biểu thức được viết cách nhau một dấu phẩy (“,”).

* Các biểu thức được tính lần lượt từ trái qua phải.

* Biểu thức trong biểu_thức_2 quyết định thực hiện thân của for.

Ví dụ :

```
<?php
for($i = 0; $j = 4,$i < $j; $i++,$j--)
 {
   echo (“i =”. $i. “ ,j = “. $j. “<br>”);
 }
?>
```

* Có thể vắng mặt bất kể thành phần nào. Nếu vắng mặt biểu_thức_2 thì câu lệnh luôn được thực hiện. Mặc dù vắng mặt vẫn phải có dấu chấm phẩy (“;”)
Ví dụ :

```
<?php
for (  ;  ; )
{
  if (my_function() == “stop”) break;
}
?>
```

* Nếu vắng biểu_thức_1 và biểu_thức_3 thì :

```
for ( ; biểu_thức 2 ; )
{
  cau_lenh ;
}
```

tương đương với :

```
while (biểu_thức_2)
{
  cau_lenh ;
}
```

1.7.6 Lệnh Break

Là lệnh rẽ nhánh không điều kiện và thường dùng để ra khỏi thân của switch, while, do ... while, for .
Lệnh break chỉ cho phép thoát khỏi thân các lệnh bên trong nhất chứa nó

1.7.7 Lệnh Continue

Là lệnh rẽ nhánh không điều kiện .Lệnh thường dùng để bắt đầu lại một chu trình mới trong các lệnh for, while, do ... while mà không cần thực hiện hết toàn bộ thân của của lệnh tạo chu trình.

1.7.8 Thoát khỏi cấu trúc điều khiển bằng phát biểu Exit

Để thoát khỏi vòng lặp hay phát biểu điều khiển nào đó, bạn có thể sử dụng phát biểu Exit. Khi chương trình gặp phát biểu Exit, lập tức thông dịch sẽ kết thúc quá trình diễn dịch cho khối chương trình mặc dù bên dưới còn có các phát biểu khác.

Ví dụ:

```
<?php
$qtty=5;
$price=5000;
While($qtty>0)
{
  echo $qtty*$$price;
  $qtty--;
  If($qtty==3)
    exit;
}
?>
```

Trong ví dụ trên nếu biến $qtty bằng 3 thì thoát khỏi vòng lặp while, ngược lại thì cứ tiếp tục thực hiện.

1.8 Xử lý giá trị form trong PHP

Khi bạn muốn submit dữ liệu người dùng nhập vào từ trang web phía Client lên Server bạn có hai cách để làm điều này ứng với hai phương thức Post và Get trong thẻ Form, bạn cho phép Server Scrip lấy các giá trị từ các thẻ nhập liệu. Chẳng hạn, các thẻ input, select, texterea, hiden trên trang Web mà người dùng đang sử dụng.

1.8.1 Cách sử dụng Form trong HTML

Một mẫu biểu (form) trong HTML bao gồm nhiều thành phần khác nhau. Các thành phần có thể là ô văn bản, ô kéo thả, ô danh sách, nút bấm, hay các ô check ...

Mẫu biểu được bắt đầu bằng thẻ <form> và kết thúc bởi thẻ </form>. Giữa 2 cặp thẻ này, các bạn có thể sử dụng các cặp thẻ HTML khác.

Thẻ form có một số thuộc tính sau:

- Thuộc tính method : Thuộc tính này có 2 giá trị POST hoặc GET, để xác định dữ liệu gửi lên theo kiểu POST hay GET.

Kiểu GET chính là kiểu mà khi nhập dữ liệu lên máy chủ, các dữ liệu này sẽ được hiển thị trên ô Address dưới dạng các căp tên=giá_tri. Nhược điểm của kiểu này là toàn bộ cái URL và xâu tên=giá_trị kia sẽ bị giới hạn dưới 255 ký tự (do đặc điểm của trình duyệt). Vì vậy để có thể gửi nhiều dữ liệu hơn, người ta đã sinh ra kiểu POST. Với kiểu này, dữ liệu sẽ không bị giới hạn chiều dài 255 ký tự của chuỗi địa chỉ do không bị gộp vào chuỗi địa chỉ. Kiểu POST cũng thường dùng để truyền các dữ liệu nhạy cảm mà người sử dụng không muốn hiển thị trên ô Address (password chẳng hạn).

Ví dụ:

```
<form method = "post"></form>
```

- Thuộc tính action: Thuộc tính này sẽ chỉ định form gửi dữ liệu đến trang nào. Trong trường hợp thuộc tính này không được khai báo, form sẽ gửi thẳng dữ liệu và yêu cầu về chính trang hiện hành (sau đó trình duyệt sẽ tải lại nội dung mới).

Ví dụ:

```
<form method = "post" action ="login.php"></form>
```

- Thuộc tính enctype: Enctype=”Mime_type” :chỉ ra loại dữ liệu sẽ gửi đi. Giá trị ngầm định là application/x-www-form-urlencode.
Tuy nhiên, 2 ví dụ trên chưa có ý nghĩa gì, vì chúng ta chưa trang bị các thành phần cơ bản của form như các thẻ nhập liệu, submit...

Các thành phần cơ bản của FORM

Một FORM ,thông thường có ba thành phần chính :

+Textarea
+Select

+Input

Để thực hiện quá trình sử dụng phương thức yêu cầu dữ liệu từ phía Client, bạn cần khai báo thẻ Form bao(bounce) hết các thẻ nhập liệu cần đưa lên phía Server.

Lưu ý rằng, trong một trang Web có nhiều thẻ Form khác nhau, nhưng các thẻ Form này không được lồng nhau mỗi thẻ Form sẽ được khai báo hành động(action) để chỉ đến một trang Web khác.

Nếu bạn không chỉ định URL hay UNC cho action, trang Web được chuyển đến chính là trang Web hiện tại, đối với trang HTML được chuyển đến trong action, bạn chỉ sử dụng phương thức GET.

Với phương thức Get, trình duyệt cho phép bạn truyền dữ liệu từ các thẻ nhập liệu lên chuỗi QueryString(chuỗi bao gồm nhiều cặp tham số cũng với giá trị đi kèm nếu có kể từ sau dấu ?. các cặp cách nhau bằng dấu &, tham số và giá trị cách nhau bởi dấu =). Trong trường hợp phương thức Post, dữ liệu truyền trực tiếp từ trang Web đến trang ServerScrip.

Dữ liệu của người dùng từ trình duyệt sẽ được gửi lên máy chủ dưới dạng từng cặp biến=giá_trị và có thể đi theo 3 con đường khác nhau. Tuỳ theo từng con đường cụ thể, trên máy chủ ta cũng có các cách khác nhau để lấy dữ liệu được gửi lên.. 3 con đường đó là: GET, POST và COOKIES. Vậy GET, POST và COKIES là gì?

Chúng ta phân tích thẻ form trong HTML sau:

```
<form name=”Tên Form” action=”Link xử lý” method=”Phương thức”>
<Các thẻ nhập liệu hay các thành phần cơ bản của Form>
</form>
```

Chúng ta thấy rằng 1 form phải bao gồm:

+ Tên form để dễ dàng tách biệt với các Form khác trong trang.
+ Action: hành động chuyển tiếp đến Trang xử lý.
+ Method: Là phương thức truyền bao gồm POST hoặc GET.

Ví dụ:

```
<form name=”form1” action=”mypage.php” action=”post”>
<input type=”text” name=”fullname”>
<input type=”submit” name=”submit” value=”OK”></form>
```

khi khách nhấp chuột vào nút SUBMIT (chấp nhận) thì các biến như $fullname và $submit sẽ được chuyển giao sang trang action là mypage.php. Sau đó, trong trang mypage.php bạn sẽ xử lý các biến này tuỳ thuộc vào mục đích chương trình.

Trong trang mypage.php bạn phải viết các lệnh để xử các thao tác của người truy cập. Giá trị mà chúng ta gởi là username vừa nhập liệu.

Vậy làm cách nào để chúng ta lấy được giá trị vừa nhập liệu nào ?.
PHP cho phép ta lấy giá trị dựa vào 2 phương thức POST và GET.

Đới với POST ta có : $_POST[‘Giá trị’]
Đối với GET ta có : $_GET[‘Giá trị’]

Vậy với đoạn code trên có thể lấy được biến xử lý là : $_POST[‘fullname’];
fullname là tên của field mà người sử dụng nhập liệu vào.

Bạn hãy xem cách xử lý trong trang mypage.php mẫu như sau:

```
<?php
  echo $_POST[“fullname”];
?>
```

Như vậy ứng với phương thức POST ta có thể truy xuất thông qua mảng $_POST và ứng với phương thức GET ta cũng có thể truy xuất qua mảng $_GET, đồng thời qua 2 phương thức POST hoặc GET ta cũng có thể truy xuất dữ liệu qua biến form như sau:

```
<?php
  echo $fullname;
?>
```

ở đây $fullname được gọi là Biến Forrm với fullname là tên của field mà người sử dụng nhập liệu vào.

1.8.2 Biến Form

Biến form trong PHP được biết đến như một loại biến, thay vì khai báo thì biến đó chính là tên của thẻ nhập liệu trong trang submit hay tham số trên querystring.

Trong trang bạn submit đến, nếu khai báo tên của thẻ nằm trong thẻ form có tên là xyz thì biến form được định nghĩa là $xyz.

Ví dụ, bạn khai báo báo thẻ form trong trang submit.php như sau:

```
<form name=”form1” action=”mypage.php” action=”post”>
<input type=”text” name=”fullname”>
<input type=”submit” name=”submit” value=”OK”></form>
```

Khi người sử dụng nhập giá trị vào phần fullname và nhấn nút submit có tên là OK thì trang mypage.php sẽ được triệu gọi, trong trang này bạn có thể lấy giá trị nhập từ trang submit.php bằng cách sử dụng biến form như ví dụ sau:

```
<?php
  echo $fullname;
?>
```

Trong đó, $fullname là tên của thẻ input trong trang submit.php, trong trường hợp này chúng ta sử dụng phương thức POST cho form.

Nếu bạn sử dụng phương thức GET trong thẻ form, bạn có thể lấy giá trị của các tham số trên chuỗi **QueryString** bằng biến form.

Ví dụ khai báo thẻ form có hai tuỳ chọn như ví dụ sau với phương thức GET trong thẻ form:

```
<form action=mypage.php method=get>
<table><tr><td>Province</td>
<td>
:<select name=province>
  <option value=HAN>Ha Noi</option>
  <option value=HCM>Ho Chi Minh</option>
  <option value=HUE>Hue</option>
</select>
</td></tr>
<tr><td>Industry</td>
<td>
:<select name=industry>
<option value=AUT>Automobile</option>
<option value=FOO>Foods</option>
<option value=ENG>Enginering</option>
<option value=GAR>Garment</option>
</select>
</td></tr>
<tr><td></td>
<td><input type=submit value=Submit></td></tr></table>
</form>
```

Nếu nhấn Submit thì hai giá trị chọn sẽ được truyền lên trên QueryString với hai tham số là tên của thẻ select. Trong đó, hai tham số và giá trị tương ứng là:

mypage.php?province=HAN&industry=FOO, bằng cách sử dụng biến form bạn có thể lấy được giá trị này như ví dụ sau:

```
<?php
  echo "Province = ".$province;
  echo "Industry = ".$industry;
?>
```

Đối với trường hợp bạn không sử dụng thẻ form như hai trường hợp trên, chúng ta cũng có thể lấy giá trị từ chuỗi QueryString bằng biến form. Chẳng hạn khi chạy trang mypage.php có cấu trúc như sau:

http://mydomain.com/mypage.php?firstname=manh&lastname=hoang
ở đây biến form là tên của tham số trên QueryString có giá trị bằng giá trị của tham số:

```
<?php
  echo "First Name = ".$firstname;
  echo "Last Name = ".$lastname;
?>
```

Chú ý rằng, khi sử dụng biến form bạn không nên khai báo biến cùng tên với các tham số hay tên của thẻ nhập liệu trong trang triệu gọi trước đó. Nếu không thì giá trị trả về là giá trị của biến thường thay vì biến form.

1.8.3 Phương thức GET

Ngoài cách sử dụng biến form trong trường hợp lấy giá trị từ tham số của **QueryString**,bạn có thể sử dụng **hàm $_GET**. Ví dụ, chúng ta khai báo trang PHP như sau:

```
<form action=ex4.php method=get>
  Province:
  <select name=province>
    <option value=HAN>Ha Noi</option>
    <option value=HCM>Ho Chi Minh</option>
    <option value=HUE>Hue</option>
  </select>
  <input type=submit value=Submit>
</form>

<?php
  if(isset($_GET["province"]))
  {
    $result=$_GET["province"];
    echo "Result: ".$result;
  }
?>
```

Lưu ý rằng, nếu bạn không sử dụng hàm **isset** để kiểm tra **$province** tồn tại hay không thì trang php sẽ phun lỗi trong trường hợp lần đầu tiên gọi đến trang mà không submit.

Tương tự như vậy trong trường hợp bạn không sử dụng thẻ form mà giá trị lấy tử chuỗi QueryString bằng cách sử dụng $_GET và nhiệm vụ chính của nó vẫn là lấy nội dung trang dữ liệu từ web server

```
Với url sau: shownews.php?id=50
Vậy với trang shownews ta dùng hàm $_GET[‘id’] sẽ được giá trị là 50.
```

1.8.4 Phương thức POST

Tương tự như $ _GET nhưng $ _POST cho phép bạn lấy giá trị lấy từ các thẻ nhập liệu của thẻ form trong traang submit trước đó.

Post là phần dữ liệu được gửi qua các form HTML có method ="POST

Để lấy các biến theo kiểu POST, PHP sẽ tự động sinh ra mảng có tên là $_POST[]. Mảng này có chỉ số chính là tên của các phần tử trong form (các thẻ input, select... có thuộc tính name) và giá trị là nội dung giá trị do người sử dụng nhập vào các phần tử có tên tương ứng. Chẳng hạn với FORM sau:

```
<form method="POST">
  User Name:
  <input type="text" name="T1" size="20">
  Password:
  <input type="password" name="T2" size="20">
  Sex:
  <Select name ="sex">
    <option value =1>Male </option>
    <option value =0>Female </option>
```

```
  </select>
  <input type="submit" value="Gui di" name="B1">
</form>
```

Khi người dùng nhập user name (giả sử là Minh), password (giả sử là 123456) và chọn sex là Male, khi đó, mảng $_POST sẽ có các phần tử sau:

$_POST["T1"] = Minh
$_POST["T2"] = 123456
$_POST["sex"] = 1

Sau khi lấy được các giá trị này rồi, các bạn có thể sử dụng giá trị các biến trên.

Trong bài này, chúng ta tìm hiểu cách sử dụng biến form và hai phương thức $_POST, $_GET. Ngoài ra, bạn cũng tìm hiểu cách kiểm tra biến tồn tại hay không bằng hàm isset().

Chú ý rằng, khi sử dụng biến form bạn tránh trường hợp khai báo biến cục bộ hay toàn cục trong tang PHP cùng tên với thẻ nhập liệu của form trước đó submit đến hay tham số trên querystring.

1.9 Khái niệm cơ bản về Cookie và Session trong PHP

Cookie và session là hai phương pháp sử dụng để quản lý các phiên làm việc giữa người sử dụng và hệ thống. Việc quản lý phiên làm việc này sẽ giúp bạn tạo ra sự chứng thực hiệu quả bởi việc xác nhận thông tin trước khi truy cập vào một phân vùng cố định. Ngoài ra, việc quản lý tốt phiên làm việc cũng giúp người truy cập cảm thấy dễ dàng sử dụng dịch vụ của trang web cho những lần truy cập sau. Bởi cơ chế quản lý phiên làm việc ghi nhận lại quá trình truy cập của người sử dụng khi họ thăm viếc trang web của bạn lần đầu.

1.9.1 Session

Một cách khác quản lý người sử dụng là session. Session được hiểu là khoảng thời gian người sử dụng giao tiếp với 1 ứng dụng. Một session được bắt đầu khi người sử dụng truy cập vào ứng dụng lần đầu tiên, và kết thúc khi người sử dụng thoát khỏi ứng dụng. Mỗi session sẽ có được cấp một định danh (ID) khác nhau và nội dung được lưu trong thư mục thiết lập trong file php.ini (tham số session.save_path).

a- Thiết lập session:

Để thiết lập 1 session ta sử dụng cú pháp**: session_start()**

Đoạn code này phải được nằm trên các kịch bản HTML. Hoặc những lệnh echo.

Để thiết lập 1 giá trị session, ngoài việc cho phép bắt đầu thực thi session. Chúng ta còn phải đăng ký 1 giá trị session. Để tiện cho việc gán giá trị cho session đó.

Ta có cú pháp sau: session_register("Name")

```
<?php
Session_start();
Session_register("username");
?>
```

b- Sử dụng giá trị của session:

Để sử dụng giá trị của session ta sử dụng mã lệnh sau:

Cú pháp:

```
$_SESSION["name"]
```

Với Name là tên mà chúng ta sử dụng hàm session_register("name") để khai báo.

Ví dụ :

Tạo trang session.php với nội dung sau:

```
<?php
session_start();
session_register("name");
$_SESSION["name"] = "manhhm";
?>
<html>
<head>
<title>Test page 1</title>
</head>

<body>
<b><a href=session2.php>Click here</a></b>
</body>
</html>
```

Tạo trang session2.php với nội dung sau:

```
<?php
session_start();
?>
<html>
<head>
<title>Result Page</title>
</head>
<body>
<?php
 echo "Ten cua ban la <b>".$_SESSION["name"]."</b>";
?>
</body>
</html>
```

c- Hủy bỏ session:

Để hủy bỏ giá trị của session ta có những cách sau:

session_destroy() // Cho phép hủy bỏ toàn bộ giá trị của session
session_unset()// Cho phép hủy bỏ session .

Ví dụ: Tạo trang session3.php với nội dung sau:

```
<html>
<head>
<title>Test page 1</title>
</head>
<body>
<?php
session_start();
session_destroy();
?>
<b><a href=session2.php>Click here</a></b>
</body>
</html>
```

1.9.2 Cookie

Cookie là 1 đoạn dữ liệu được ghi vào đĩa cứng hoặc bộ nhớ của máy người sử dụng. Nó được trình duyệt gởi ngược lên lại server mỗi khi browser tải 1 trang web từ server.
Những thông tin được lưu trữ trong cookie hoàn toàn phụ thuộc vào website trên server. Mỗi website có thể lưu trữ những thông tin khác nhau trong cookie, ví dụ thời điểm lần cuối ta ghé thăm website, đánh dấu ta đã login hay chưa, v.v...

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

Cookie được xem như session, tuy nhiên chúng lưu trữ thông tin trên trình khách. Để sử dụng Cookie, bạn sử dụng hàm setcookie để gán giá trị
a -Thiết lập cookie:

Để thiết lập cookie ta sử dụng cú pháp:

Setcookie("tên cookie","giá trị", thời gian sống)
Tên cookie là tên mà chúng ta đặt cho phiên làm việc.
Giá trị là thông số của tên cookie.

Ví dụ:

```
Setcookie("username","admin", time() +3600)
```

Như ví dụ trên ta thấy với tên là username và giá trị là admin, có thời gian sống là 1 giờ tính từ thời điểm thiết lập.

Chú ý: Kịch bản cookie phải đặt trên mọi giá trị trả về bao gồm thẻ HTML và lệnh echo.

Kịch bản cookie phải đặt trên mọi giá trị trả về bao gồm thẻ HTML và lệnh echo.

b - Sử dụng cookie:

Để sử dụng lại cookie vừa thiết lập, chúng ta sử dụng cú pháp:

Cú pháp: $_COOKIE["tên cookies"]
Tên cookie là tên mà chúng ta thiết lập phía trên.

Ví dụ : Tạo trang cookie.php với nội dung sau:

```
<?php
  setcookie("name","manhhm",time() + 3600);
?>
<html>
<head>
<title>Test page 1</title></head>
<body>
<b><a href=cookie2.php>Click here</a></b>
</body>
</html>
```

Tiếp tục tạo trang cookie2.php với nội dung sau:

```
<html>
<head><title>Result Page</title></head>
<body>
<?php
   echo "Ten cua ban la <b>".$_COOKIE['name']."</b>";
?>
</body>
</html>
```

c- Hủy Cookie:

Để hủy 1 cookie đã được tạo ta có thể dùng 1 trong 2 cách sau:

+ Cú pháp: setcookie("Tên cookie")
  Gọi hàm setcookie với chỉ duy nhất tên cookie mà thôi
+ Dùng thời gian hết hạn cookie là thời điểm trong quá khứ.

Ví dụ:

```
setcookie("name","manhhm",time()-3600);
```

Ví dụ: Tiếp tục tạo trang cookie3.php với nội dung sau:

```
<?php
  setcookie("name","Kenny Huy",time()-360);
?>
<html>
<head>
<title>Test page 1</title></head>
<body>
<b><a href=cookie2.php>Click here</a></b>
```

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

Sau bài học này chúng ta đã nắm được cách điều khiển phiên làm việc giữa cookie và session. Sử dụng chúng trong từng trường hợp cụ thể. Từ đó có thể áp dụng để viết những ứng dụng nhỏ như kiểm soát người đăng nhập, làm giỏ hàng online,…..

1.10 Dữ liệu dạng chuỗi

Như ta đã biết, trong mọi trang web, thông tin được lưu trữ dưới nhiều dạng: văn bản, hình ảnh, âm thanh, video… Trong đó, hình ảnh, âm thanh, video có thể có hoặc không nhưng có lẽ dữ liệu dưới dạng văn bản là không thể không có. Vì vậy, việc xử lí văn bản là một việc vô cùng quan trọng. Xử lí văn bản là gì? Đó chính là xử lí chuỗi. Hơn nữa, không phải chỉ có xử lí văn bản mới cần đến xử lí chuỗi. Các dữ liệu hình ảnh, âm thanh, video để hiển thị được trên trang web cũng cần rất nhiều đến xử lí chuỗi. Chúng ta đang học thiết kế web PHP, vì thế chúng ta sẽ đi tìm hiểu về xử lí chuỗi trong PHP.

1.10.1 Định nghĩa, khai báo

Cũng như trong mọi ngôn ngữ lập trình khác. Chuỗi trong PHP là một dãy các ký tự được đặt trong cặp dấu “”.

Biến chuỗi (biến dạng chuỗi, biến kiểu chuỗi) trong PHP là biến được khởi tạo giá trị (được gán giá trị lần đầu tiên) là một chuỗi.

Ví dụ:

```
$strSubject = “Thiết Kế Web”;
```

1.10.2 Các phép toán xử lí chuỗi

1.10.2.1 Định dạng chuỗi

1.10.2.1.1 Các hàm xử lí khoảng trắng.

Trước khi sử dụng các hàm xử lý khoảng trắng của chuỗi, bạn có thể sử dụng hàm *isempty(string str)* để kiểm tra chuỗi có rỗng hay không. Nếu rỗng trả về giá trị *true*, ngược lại trả về *false.*
Hàm *ltrim(string str)* : xóa bỏ khoảng trắng bên trái chuỗi *str.*
Hàm *chop(string str)* : xóa bỏ khoảng trắng bên phải chuỗi *str.*
Hàm *trim(string str)* : xóa bỏ tất cả các khoảng trắng tồn tại trong chuỗi *str.*

Ví dụ:

```
$name = “   Thich Tieu Tien    ”
$name = ltrim($name);  =>  $name = “Thich Tieu Tien    ”
$name = chop($name);  => $name = “    Thich Thieu Tien”
$name = trim($name); => $name = “Thich Tieu Tien”
```

1.10.2.1.2 Định dạng chuỗi bằng các thẻ HTML.

Khi trình bày trên 1 trang web, cần thay thế những kí tự xuống dòng trong cơ sở dữ liệu thành thẻ
, để trình bày xuống hàng trên trình duyệt Web. Ta sử dụng hàm *nl2br(string str)*.
Ngoài ra, ta cũng có thể đặt việc hiển thị các chuỗi trong các đoạn thẻ html để định dạng.

Ví dụ:

```
<p><? echo nl2br ($chuoiCanChuyen) ; ?></p>
```

1.10.2.1.3 Định dạng chuỗi khi in trong PHP với printf

Hàm *printf()* khác *echo* ở giá trị trả về là giá trị 0 hay 1.
Ngoài ra để in ra giá trị phù hợp với ý muốn cũng có thể sử dụng hàm in giống C là printf và sprintf với cú pháp như sau:

string sprintf (string format [, mixed args…])
int printf(string format [, mixed args…])

Ở 1 góc độ nào đó khi sử dụng echo để in chuỗi cũng giống như hàm printf:

```
$total = 12.5;
echo "total amount of order is $total.";
printf ( " total amount of order is % s.", $total );
```

Khai báo % dùng để chuyển đổi giá trị của $total thành chuỗi. Tuy nhiên, nếu có giá trị dấu chấm động sau số lẻ, có thể dùng hàm *printf()* với tham số để chuyển đổi sang nhiều kiểu dữ liệu muốn in ra.

1.10.2.1.4 Thay đổi kiểu chữ của chuỗi

Hàm	Diễn giải	Sử dụng	Kết quả
strupper	Chữ hoa	strupper("thu")	THU
strlower	Chữ thường	strlower("thu")	thu
ucfirst	Chữ hoa đầu tiên	ucfirst("bui thu")	Bui thu
ucword	Chữ hoa đầu tiên của mỗi từ	ucword("bui thu")	Bui Thu

1.10.2.1.5 Định dạng chuỗi để lưu trữ vào cơ sở dữ liệu

Tương tự như các hàm định dạng chuỗi để trình bày trên trình duyệt, trong trường hợp lưu giá trị chuỗi vào cơ sở dữ liệu phải sử dụng 1 hàm để loại hay thay thế 1 sô kí tự mà người dùng nhập.

Ví dụ: muốn lưu một dấu " hay cặp dấu "", hay \,\\..... cần thay thế bằng chuỗi khác trong cơ sở dữ liệu.

Dấu 'đổi thành \', dấu " thành \", dấu \ thành \\, dấu \\ thành \\\.
Để thêm dấu \ vào các kí tự trình bày trên chuỗi, dùng hàm *AddSlashes(string str)*

Ví dụ :

```
$feedback = AddSlashes($feedback);
```

Để loại dấu \ đã thêm trước đó ta dùng hàm *StripSlashes(string str)*

Ví dụ :

```
$feedback = StripSlashes($feedback);
```

1.10.2.2 Kết hợp và tách chuỗi

1.10.2.2.1 Hàm explode()

Dùng để tách chuỗi input thành nhiều chuỗi con bằng cách chỉ định chuỗi tách separator có cú pháp:

```
array = explode(string separartor, string input);
```

Ví dụ: muốn tách địa chỉ email thành 2 phần username và domail, sau đó lưu vào mảng có thể viết :

```
$email_array = explode("@", $email);
```

1.10.2.2.2 Hàm implode(), join()

Dùng để kết hợp giá trị của 2 phần tử mảng thành 1 chuỗi mới.

```
$new_email = implode($username, "@gmail.com");
```

1.10.2.2.3 Hàm strok()

Nhận các chuỗi con tại 1 thời điểm

Cú pháp:

```
string strtok ( string input, string separartor);
```

Khai báo hàm :

```
$token = strtok($feedback, " " );
echo $token. "<br>";
while ($token != "")
{
$token = strok("");
echo $token. "<br>";
}
```

1.10.2.2.4 Hàm substr()

Dùng để lấy chuỗi con với chiều dài *l* bắt đầu từ vị trí *i* từ chuỗi *str* với cú pháp:
string substr (string str, int l [, int l]);
Với *int l* là tham số tùy chọn.
Ví dụ sử dụng hàm *substr()*

```
$str = "your customer service is excellent";
echo $str. "<br>"; => your customer service is excellent
echo substr ($str,1)."<br>"; => our customer service is excellent
echo substr ($str,-9)."<br>"; => xcellent
echo substr ($str,0, 4)."<br>"; => your
echo substr ($str,4, -13)."<br>"; => customer service
```

1.10.2.2.5 So sánh chuỗi

Hàm strcmp(), strcasecmp(), strnatcmp()
Để so sánh 2 chuỗi ta dùng hàm *strcmp(),* hàm trả về giá trị 0 nếu 2 chuỗi bằng nhau, trả về giá trị lớn hơn 0 nếu *str1* lớn hơn *str2*, ngược lại trả về giá trị nhỏ hơn 0 nếu *str1* nhỏ hơn *str2*.

Có cú pháp : int strcmp(string str1,string str2) ;
Hai hàm *strcasecmp(), strnatcmp()* sử dụng tương tự nhưng có phân biệt chữ hoa và chữ thường.

1.10.2.2.6 Hàm strlen()

Dùng để kiểm tra chiều dài chuỗi.
Cú pháp: $len = strlen($input);
Ví dụ: strlen("test") =4;

1.10.2.3 Tìm kiếm và thay thế chuỗi con trong chuỗi khác

1.10.2.3.1 Tìm kiếm chuỗi trong chuỗi

- **Hàm strstr()**

  Ví dụ: $str = "your customer service is excellent";
  => strstr($str, "is") = "is excellent"

- **Hàm strchr()**

  Ví dụ: $str = "your customer service is excellent";
  => strchr ($str, "c") = "customer service is excellent"

- **Hàm strrchar()**

  Ví dụ: $str = "your customer service is excellent";
  => strrchar ($str, "e") = "ent"

- **Hàm stristr()**

  Ví dụ: $str = "your customer service is excellent";
  => stristr ($str, "er") = "er service is excellent"

1.10.2.3.2 Tìm kiếm vị trị chuỗi trong chuỗi

- **Hàm strpos**

  Ví dụ: $str = “your customer service is excellent”;
  => strpos ($str, “is”) = 22

- **Hàm strrpos**

  Ví dụ: $str = “your customer service is excellent”;
  => strrpos ($str, “e”) = 31

1.10.2.3.3 Thay thế chuỗi con trong chuỗi khác

- **Hàm str_replace()**

  Ví dụ: $str = “your customer service is excellent”;
  => str_replace(“er”, “ers”, $str) = your customers sersvice is excellent

- **Hàm substr_replace**

  Ví dụ: $str = “your customer service is excellent”;
  => strrpos ($str, “good.”, 27) = your customers sersvice is good.

1.10.2.4 Biểu thức chính quy

1.10.2.4.1 Định nghĩa

Biểu thức chính qui là một khuôn mẫu - một tiêu bản - để được sánh với một xâu. Việc sánh một biểu thức chính qui với một xâu thì hoặc thành công hoặc thất bại. Đôi khi, sự thành công hay thất bại này có thể là tất cả những gì bạn quan tâm tới. Vào lúc khác, bạn sẽ muốn lấy một khuôn mẫu đã sánh đúng và thay thế nó bằng một xâu khác, một phần trong đó có thể phụ thuộc đích xác vào cách thức và nơi chốn mà biểu thức chính qui được sánh đúng.

Một số mẫu

Mẫu	Diễn giải
[0123456789]	# sánh với mọi chữ số
[0-9]	# tương tự như trên
[0-9\-]	# sánh 0-9 hay dấu trừ
[a-z0-9]	# sánh bất kì chữ thường hay số nào
[a-zA-Z0-9_]	# sánh bất kì chữ, số hay dấu gạch dưới

Một số kí tự đặc biệt

Kí hiệu	Diễn giải
\	kí tự escape

^	phù hợp với đầu chuỗi
$	phù hợp với cuối chuỗi
.	phù hợp với các kí tự ngoại trừ \n
|	kí hiệu rẽ nhánh OR
*	lặp lại 0 hay nhiều lần
+	lặp lại 1 hay nhiều lần
{min,max}	lặp lại từ min đến max lần

Một số lớp kí tự chung

Mẫu	**Diễn giải**
\d	số ([0-9])
\D	không phải số (phủ định của \d) ([^0-9])
\w	từ ([a-zA-Z0-9_])
\W	không phải từ (phủ định của \w) ([^a-zA-Z0-9_])
\s	khoảng trắng ([\r\t\n\f])
\S	không phải là khoảng trắng (phủ định của \s) ([^ \r\t\n\f])

1.10.2.4.1.1 Cú pháp trong php

Cũng giống như việc xử lí chuỗi ở bất kì một ngôn ngữ nào khác. Trong PHP cũng hỗ trợ việc xử lí theo biểu thức chính quy.

Trong PHP, ta có thể sử dụng biểu thức regex thông qua các hàm regex. PHP cung cấp 3 nhóm hàm regex, tên của chúng dc bắt đầu bởi: ereg, mb_ereg và preg. 2 loại đầu sử dụng engine POSIX Extended, còn preg sử dụng engine PCRE (Perl-Compatible).

Cú pháp:

ereg($pattern, $string) trả về true nếu chuỗi $string chứa mẫu $pattern và trả về false nếu ngược lại.

preg_match($pattern, $string, $match); $match chứa kết quả của việc so khớp $string với mẫu $pattern.

Ví dụ:

```
$string = "One Two    Three"
ereg('/Two/', $ string); trả về true
preg_match('/^(\w+)\s+(\w+)\s+(\w+)$/', $ string, $match) thì
$march[1] = "One";
$march[2] = "Two";
$march[3] = "Three";
```

1.11 Làm việc với mảng

Như trong bài kiểu dữ liệu chúng ta đã làm quen với kiểu dữ liệu mảng, trong phần này chúng ta tiếp tục tìm hiểu các khai báo, truy cập và tương tác với tập tin từ mảng một chiều, hai chiều.

1.11.1.1 Mảng một chiều

Để khai báo mảng một chiều, bạn có thể sử dụng cú pháp như sau:

```
$arr=array();
$arrs=array(5);
```

Truy cập vào phần tử mảng, bạn có thể sử dụng chỉ mục của phần tử như sau:

```
$arr[0]=1;
$arrs[1]=12;
```

Lấy giá trị của phần tử mảng, bạn cũng thực hiện tương tự như trường hợp truy cập mảng phần tử.

```
echo $arr[0];
$x=$arrs[5];
```

Chẳng hạn, chúng ta khai báo mảng động và mảng có số phần tử cho trước, sau đó truy cập và lấy giá trị của chúng như ví dụ trong trang arrayone.php sau:

```
<?php
 $i=0;
 $myarr=array(1,2,3,4,5,6,7);
 $arr=array();
 $arrs=array(10);
 $arr[0]=10;$arr[1]=11;$arr[2]=12;$arr[3]=13;
 for($i=0;$i<sizeof($arr);$i++)
 {
   echo $arr[$i]." ";
 }
 echo "<br>";
 echo "Gia tri lon nhat ".max($arr)."<br>";
 echo "Gia tri nho nhat ".min($arr)."<br>" ;
 echo "Gia tri trung binh ".array_sum($arr) / sizeof($arr)."<br>" ;
 echo "<br>";
 for($i=0;$i<=10;$i++)
 {
   $arrs[$i]=10+$i;
 }
 for($i=0;$i<=10;$i++)
 {
   echo $arrs[$i]." ";
 }
 echo "<br>";
 echo "Gia tri lon nhat ".max($arrs)."<br>";
 echo "Gia tri nho nhat ".min($arrs)."<br>" ;
 echo "Gia tri trung binh ".array_sum($arrs) / sizeof($arrs)."<br>" ;
?>
```

1.11.1.2 Mảng hai chiều

Tương tự như mảng một chiều, trong trường hợp làm việc mảng hai chiều bạn khai báo tương tự như trang arraytwo.php.

```
<?php
$i=0;
$j=0;
$arr=array();
$arr[0][0]=10;
$arr[0][1]=11;
$arr[0][2]=12;
$arr[1][0]=13;
$arr[1][1]=14;
$arr[1][2]=15;
$arr[2][0]=16;
$arr[2][1]=17;
$arr[2][2]=18;
for($i=0;$i<sizeof($arr);$i++)
{
  for($j=0;$j<sizeof($arr);$j++)
  {
        echo $arr[$i][$j]." ";
  }
  echo "<br>";
}
echo "<br>";
$arrs=array(array(1,2,3,4,5,6,7), array(11,12,13,14,15,16,17));
for($i=0;$i<=7;$i++)
{
  for($j=0;$j<=7;$j++)
  {
    $arrs[$i][$j]=10+$i*$j;
  }
}
for($i=0;$i<=7;$i++)
{
  for($j=0;$j<=7;$j++)
  {
  echo $arrs[$i][$j]." ";
  }
  echo "<br>";
}
echo "<br>";
?>
```

1.11.1.3 Mảng kết hợp

Là các mảng được tạo index bằng các chuỗi , chúng được gọi là các mảng kết hợp rất thuận lợi khi dùng để ánh xạ một mảng sử dụng các từ hơn là sử dụng các số (integer), nó giúp ta giảm bớt thời gian và các mã yêu cầu để hiển thị một giá trị cụ thể.

Ví dụ mảng một chiều kết hợp:

```
$countries[“ca”] = “Canada”;
$countries[“cr”] = “Costa Rica”;
$countries[“de”] = “Germany”;
$countries[“uk”] = “United Kingdom”;
$countries[“us”] = “United States”;
```

```
echo (“$countries[“ca”]);  :// print Canada
Nếu dùng array thì sẽ là
$countries = (“ca” => “Canada”,
                      “cr” => “Costa Rica”,
                      “de” => “Germany”,
                       “uk” => “United Kingdom”,
                      “us” => “United States”);
```

Ví dụ mảng hai chiều kết hợp:

```
$countries = array (“Europs” => array (“de”, “uk”),

                                “North  America”  =>  array  (“ca”,  “cr”,
“us”));

echo ($countries[“Europs”][1]);           // print “uk”

echo ($countries[“North America”][2]); // print “us”
```

1.11.1.4 Truy xuất phần tử mảng

1.11.1.4.1 Dùng vòng lập for

Nếu biết trước số phần tử của mảng ta có thể dùng vòng lặp for để duyệt qua các phần tử mảng:

```
<?php
$giatri = array(1,2,3,4,5,6,7,8,9,10);
for ($i = 0; $i < 10 ; $i ++)
echo $giatri[$i]."<br>";
?>
```

Chạy đoạn mã trên PHP sẽ xuất ra từ 1 đến 10 .

Ví dụ trên sẽ đúng khi chỉ số của các phần tử tăng dần đều

1.11.1.4.2 Dùng Foreach hoặc While

Cú pháp :

```
foreach (array_expression as $value)
    statement
foreach (array_expression as $key => $value)
    statement
```

```
<?php
$giatri = array(1,2,3,4,5,6,7,8,9,10);
foreach ($giatri as $value)
echo $value."<br>";
?>
```

Với foreach này để nhập giá trị vào ta phải thêm dấu " &" trước biến $value , như thế này &$value

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
}
// $arr is now array(2, 4, 6, 8)
?>
```

Thêm 1 ví dụ nữa , lần này sẽ có sự xuất hiện của chỉ mục

```
<?php
$arr = array("mot"=>"one", "hai"=>"two","ba"=> "three");
foreach ($arr as $key => $value) {
    echo "Key: $key; Value: $value<br />\n";
}
?>
```

Một cách khác để làm như trên ta sẽ dùng vòng lặp while đi với list() và each(). Xét ví dụ sau , thay vì dùng foreach như trên thì ta dùng while như sau:

```
<?php
$arr = array("one", "two","three");
while (list($key, $value) = each($arr)) {
    echo "Key: $key; Value: $value<br />\n";
}
?>
```

List sẽ gán giá trị cho biến được khai báo bên trong hàm list với giá trị là giá trị tương ứng của mảng.

Còn hàm each() sẽ trả về giá trị chỉ mục và giá trị của phần tử mảng hiện tại , đồng thời chuyển vị trí của chỉ mục lên 1 đơn vị , sẽ trả về False nếu vị trí của chỉ mục là vị trí cuối cùng trong mảng.Cặp giá trị trả về này sẽ nằm trong 1 mảng 4 phần tử là 0,1 , key, value . Phần tử 0 và key chứa tên chỉ mục của mảng, phần tử 1 và value tất nhiên sẽ chứa giá trị .

Có thể chỉ sử dụng while và each thôi cũng được

```
<?php
$arr = array("one", "two","three");
while ($phantu = each($arr)) {
    echo "Key: $phantu['key']; Value: $phantu['value']<br />\n";
}
?>
```

Đối với mảng nhiều chiều thì cách làm cũng tương tự, chỉ phức tạp hơn là thêm 1 vòng lặp nữa.

```
<?php
$sanpham = array( array("ITL","INTEL","HARD"),
                        array("MIR", "MICROSOFT","SOFT"),
                        array("PHP", "PHPVN.ORG","TUTORIAL")
                         );
for ($row = 0; $row < 3; $row++)
 {
    for ($col = 0; $col <3; $col++)
```

```
        {
         echo "|".$sanpham[$row][$col];
         }
echo "<br>";
}
?>
```

1.11.1.5 Sắp xếp mảng

Do PHP lưu trữ các phần tử trong mảng theo thứ tự chúng được đưa vào mảng, chính vì vậy đôi lúc sẽ nảy sinh nhu cầu sắp xếp các phần từ trong mảng. Để sắp xếp ta có thể dùng các hàm có sẵn của PHP, tiêu biểu là hàm sort để sắp xếp tăng dần các phần tử của mảng, hàm rsort() sẽ sắp xếp các phần tử của mảng giảm dần .

```
$countries = array (“us”, “uk”, “ca”, “cr”, “de”);
sort ($countries);
while (list ($key, $val) = each ($countries))
{
        echo (“Element $key equals $val <BR>\n”);
}
```

Khi chạy chương trình sẽ là :

```
Element 0 equals ca
Element 1 equals cr
Element 2 equals de
Element 3 equals uk
Element 4 equals us
```

Với mảng chỉ số có kiểu string thì dùng hàm asort (), arsort () để sắp xếp mảng theo chiều tăng dần và giảm dần. Nếu bạn dùng các hàm sort() và rsort() thì các chỉ số có kiểu string sẽ chuyển thành các chỉ số có kiểu nguyên.

```
 $countries = array(“us” => “United States”,
                                “uk” => “United Kingdom”,
                                “ca” => “Canada”,
                                “cr” => “Costa Rica”,
                                “de” => “Germany”);
asort ($countries);
while (list($key, $val) = each($countries))
{
        echo “Chi so $key bang $val <BR>\n”;
}
```

Khi chạy chương trình sẽ là :

```
Chi so ca bang Canada
Chi so cr bang Costa Rica
Chi so de bang Germany
Chi so uk bang United Kingdom
Chi so us bang United States
```

Nhưng nếu thay dòng asort ($countries); bằng sort ($countries); kết quả sẽ là :

```
Chi so 0 bang Canada
Chi so 1 bang Costa Rica
Chi so 2 bang Germany
Chi so 3 bang United Kingdom
Chi so 4 bang United States
```

Để sắp xếp mảng tăng dần hay giảm dần theo chỉ số chúng ta có hàm ksort() – tăng dần và krsort() – giảm dần .

```
$countries = array("e" => "United States",
                                "d" => "United Kingdom",
                                "c" => "Canada",
                                "b" => "Costa Rica",
                                "a" => "Germany");
ksort ($countries);
while (list($key, $val) = each($countries))
{
        echo "Chi so $key bang $val <BR>\n";
}
```

Kết quả :

```
Chi so a bang Germany
Chi so b bang Costa Rica
Chi so c bang Canada
Chi so d bang United Kingdom
Chi so e bang United States
```

1.12 Kiểu DateTime

Để làm việc với kiểu dữ liệu Date và Time, bạn sử dụng hàm của PHP có sẵn. Chẳng hạn, muốn trình bày chuỗi ngày tháng, bạn dùng hàm date với các tham số như ví dụ sau:

```
<?php
echo date("j-S-F-Y");
echo "<br>";
echo date("M/Y");
echo "<br>";
echo "Days of ".date("M")." is ".date("t");
echo "<br>";
?>
```

Lưu ý rằng, tham số trong hàm date được trình bày trong bảng sau

---

CodeDiễn giải

---

a Buổi sáng/Chiều bằng hai ký tự thường *am/pm*.
A Buổi sáng/Chiều bằng hai ký tự hoa *AM/PM*.
B Định dạng thời gian *Swatch Internet*, bạn có thể tham khảo *http://swatch.com/internettime/internettime.php3*.
d *Day* (01-31) trong tháng với hai số, nếu ngày 1-9 sẽ có kèm số 0.
D *Day* (*Mon-Sun*) trong tuần với 3 ký tự.
F Tháng (*January-December*) trong năm với tên tháng đầy đủ dạng text.

g *Hour* (1-12) trong ngày 1 hoặc 2 số (không kèm 0 nếu giờ từ 1-9).
G *Hour* (0-23) trong ngày 1 hoặc 2 số (không kèm 0 nếu giờ từ 0-9).
h *Hour* (01-12) trong ngày 2 số (kèm 0 nếu giờ từ 01-09).
H *Hour* (00-23) trong ngày 2 số (kèm 00 nếu giờ từ 00-09).
i *Minutes* (01-59) đã trôi qua (kèm 00 nếu phút từ 00-59).
j *Day* (1-31) 1 hoặc 2 số (không kèm 0 nếu ngày từ 1-9).
l *Day* (*Monday-Sunday*) trong tuần dạng *text*.
L Năm nhuần trả về 1, ngược lại hàm trả về 0.
m *Month* (01-12) trong năm 2 số (kèm 00 nếu tháng từ 01-09).
M *Month* (*Jan-Dec*) trong năm 3 ký tự.
n *Month* (1-12) 1 hoặc 2 số (không kèm 0 nếu tháng từ 1-9).
s *Seconds* (01-59) đã trôi qua (kèm 00 nếu giây từ 00-59).
S Thêm hai ký tự *st*, *nd*, *rd* hay *th* theo sau ngày dạng hai ký tự số (ví dụ như 12th).
t Trả về tổng số ngày trong tháng (từ 28 -31).
T Ký tự *Timezone* của server với 3 ký tự, chẳng hạn như *EST*.
U Tổng số *Seconds* tứ 1 January 1970 tới hôm nay ứng với *UNIX Time Stamp*.
w Day (0-6) của tuần, 0 ứng với *Sunday* và 6 ứng với *Saturday*.
y Năm định dạng 2 con số (03).
Y Năm định dạng 4 con số (2003).
z Ngày trong năm một hoặc 2 con số (0-365).
X *Timezone* hiện tại tính bằng giây từ –43200 đến 43200.

1.13 Xây dựng hàm, tổ chức quản lí hàm

Trong hầu hết các ngôn ngữ lập trình, việc xây dựng và sử dụng lại hàm để quản lí mã chương trình và sử dụng lại mã chương trình là rất quan trọng.

Ngoài các hàm có sẵn trong thư viện của PHP,người dung có thể xây dựng hàm với nhiều mục đích khác nhau. Đó là một trong những lí do bạn cần xây dựng hàm.

Trong quá trình lập trình ứng dụng, bạn cũng có nhiều module hay những đoạn thủ tục được sử dụng lại.Vì vậy bạn có thể viết chúng thành những hàm khác nhau, nhằm dễ quản lí, dễ sử dụng và sử dụng lại cho các ứng dụng PHP khác nếu cần.
Trong phần này, chúng ta sẽ đi tìm hiểu về cách sử dụng hàm trong PHP, cách xây dựng và tổ chức hàm trong PHP để tăng cường khả năng sử dụng hàm.

1.13.1 Sử dụng hàm trong PHP

1.13.1.1 Đặc điểm hàm trong PHP

- Hàm trong PHP có thể có tham số hoặc không có tham số, có đối số hoặc không có đối số.
- Tên hàm trong PHP không phân biệt chữ hoa hay chữ thường
- Ngoài các hàm mà ngôn ngữ lập trình hỗ trợ,PHP còn cho phép người dung xây dựng hàm tự định nghĩa, những hàm này được gọi là hàm người dùng.

1.13.1.2 Cách gọi hàm thông thường

- Hàm không có giá trị trả về: function_name([tham số nếu có]);

- Hàm có giá trị trả về: $i=function_name([tham số nếu có]);

Chú ý:

- Gọi hàm chưa được khai báo có nghĩa là hàm đó không tồn tại trong PHP hay người dùng không định nghĩa thì lỗi sẽ phát sinh
- Tên hàm trong PHP không phân biệt chữ hoa hay chữ thường. Điều này có nghĩa là bạn gọi hàm có tên function_name hay FUNCTION_NAME đều cùng chung 1 kết quả, chẳng hạn như hàm name() hay Name() là một.

1.13.2 Khai báo require() và include()

1.13.2.1 Hướng dẫn sử dụng require() trong PHP

PHP cung cấp 2 cách để khai báo và sử dụng bất kỳ đoạn mã chương trình nào đã có bằng cách sử dụng 2 khai báo: **include()** và **require()**. Với 2 khai báo này bạn có thể chèn tập tin, kịch bản PHP, text, HTML hoặc các Class của PHP.

Sử dụng **require()**.

Trong ví dụ này chúng tôi sử dụng 2 file: **test.php** và **test2.php**.

Trang **test.php** có nội dung như sau.

```
<?php
 require("test2.php");
?>
```

Trang **test2.php** có nội dung như sau.

```
<?php
      echo    "Bạn    đang    truy    cập    Website:    http://vnskills.com";
?>
```

Trong ví dụ trên khi chạy file: **test.php** thì nội dung của trang **test2.php** sẽ được nhúng và trang **test.php**.

1.13.2.2 Tên mở rộng, thẻ php và hàm require()

PHP không quan tâm đến phần mở rộng của các file trong khai báo require(), điều này có nghĩa là bạn có thể đặt tên cho file muốn khai báo trong requite() tùy thích.
Khi bạn dùng require để nạp file, các phần nằm trong khai báo <?php … ?> của file trong khai báo require() sẽ được thông dịch còn các phần khác sẽ được xử lí như nội dung html thông thường.

1.13.2.3 Sử dụng include()

Sử dụng include() cũng giống như require(). Tuy nhiên có 1 sự khác biệt mà chúng ta cần chú ý tới khi quyết định sử dụng 1 trong 2 hàm trên. Khi sử dụng require() thì web server phải dịch lại tất cả nội dung bên trong nó. Ngược lại, include() thì không.

Ví dụ 1:

```
<?php
    if($variable=true){
        require("file01.php");
    }else{
       require("file02.php");
    }
?>
```

Ví dụ 2:

```
<?php
    if($variable=true){
        include("file01.php");
    }else{
        include("file02.php");
    }
?>
```

Trong ví dụ 1, trang web sẽ phải nạp cả 2 file còn trong ví dụ 2 trang web sẽ chỉ cần nạp 1 trong 2 trang.

1.13.2.4 Ứng dụng của khai báo require() và include()

- Xây dựng template: Vì các khai báo require() và include() có khả năng nhúng nội dung html từ một file vào một file khác nên rất thuận lợi để xây dựng template. Các thành phần của trang web sẽ được tách rời để dễ quản lí và sẽ được nhúng vào trong trang web bằng các khai báo trên. Các thành phần chắc chắn sẽ sử dụng có thể dùng khai báo require() còn các thành phần mà việc hiển thị còn phụ thuộc vào trạng thái nào đó thì có thể dùng khai báo include().

- Khai báo các thư viện do người dùng định nghĩa. Các đoạn mã php lưu trong các file tách rời sẽ có thể được sử dụng lại như một thư viện với khai báo require() và include().

1.13.3 Hàm do người dùng định nghĩa (xây dựng hàm)

Hàm do người sử dụng định nghĩa cho phép bạn xử lý những tác vụ thường lặp đi lặp lại trong ứng dụng.

1.13.3.1 Cấu trúc đơn giản của hàm

Một hàm được định nghĩa với từ khóa bắt đầu là function, tiếp theo là tên hàm và cặp dấu "()", bên trong cặp dấu "()" có thể có một hay nhiều tham số truyền vào, các tham số cách nhau bằng dấu ",", các tham số có thể được gán sẵn dữ liệu.

Bên trong hàm được bao hàm bởi các cặp dấu "{}", cũng giống như khai báo trong PHP kết thúc tất cả các câu lệnh đều có dấu ";".

Hàm không quy định trước kiểu giá trị trả về.

Giá trị trả về của một hàm được đặt sau lệnh return.

Các tham số của hàm cũng không cần khai báo kiểu. Các tham số có thể là mảng, chuỗi, số…

Để khai bao hàm, bạn sử dụng từ khoá function với cú pháp tương tự như sau:

```
function functioname($parameter)
{
   return       value;
}
```

Ví dụ:

```
function my_fuction($x)
{
   return $x;
}
```

Trong trường hợp hàm không có giá trị trả về thì hàm được xem như thủ tục.

Ví dụ:

```
function my_function()
{
        Echo "here is my function";
}
```

Ngoài ra, bạn có thể khai báo tham số tuỳ chọn bằng cách gán giá trị mặc định cho tham số. Vídụ chúng ta khai báo:

```
function functioname($parameter1, $parameter2=10 )
{
    return      value;
}
```

Đối với trường hợp này thì tham số $parameter1 là tham số bắt buộc và tham số $parameter2 là tham số tuỳ chọn, khi gọi hàm nếu không cung cấp tham số cho $parameter2 thì tham số này có giá trị là 10.

Ví dụ:

```
function my_fuction($x,$y=2)
{
        return $x;
}
```

Chú ý :

- Bạn ko thể khai báo hàm do bạn định nghĩa có tên trùng với tên của hàm có sẵn trong PHP.Nếu bạn làm điều đó,khi gọi hàm sẽ phát sinh lỗi.
- Tên của hàm hợp lệ bao gồm ký tự số và dấu _,tên hàm không bắt đầu bằng ký tự số
- Ngoài ra,bạn cũng có thể khai báo hàm có pha trộn các thẻ HTML và thẻ PHP như khai báo sau.

```
<?php
function my_function($x)
{
?>
        Here is my function.
<?php
}
?>
```

1.13.3.2 Tầm vực của hàm

Cũng như mọi ngôn ngữ lập trình khác, PHP cũng có một số quy chế tầm vực như:

- Biến khai báo trong function sẽ chỉ có tầm vực từ khi được khai báo đến cuối function đó. Biến này gọi là biến cục bộ.

- Biến khai báo ngoài function có phạm vi từ khi khai báo cho đến hết script nhưng không có tầm vực bên trong các function của script đó. Biến này được gọi là biến toàn cục.

- Từ khóa global trước tên biến được định nghĩa và sử dụng trong function chỉ ra rằng biến đó là biến toàn cục trong script chứa function đó.

- Khai báo include() và require() không làm ảnh hưởng tới tầm vực.

- Bạn có thể sử dụng hàm isset($ten_bien) để kiểm tra biến đó đã tồn tại hay chưa.

1.13.3.3 Tham biến

Thông thường, tham số truyền vào cho hàm là tham trị - tức là trong khi xử lí có làm thay đổi giá trị của tham số đó thì khi kết thúc hàm sự thay đổi đó cũng không được lưu lại ở tham số truyền vào.

Muốn truyền tham số kiểu tham biến (tức là khi xử lí, nếu có thay đổi giá trị tham số đó thì giá trị đó sẽ được lưu lại ở tham số tuyền vào), ta sẽ thêm & vào trước tham số trong hàm. Ví dụ:

```
<?php
function tang($x)
{
        $x = $x +1;
}
$value = 1;
tang($value);
echo($value);
?>
```

Khi đó, kết quả hiển thị ra vẫn là 1, tức là thay đổi trong hàm con đã không lưu lại trên $value.

Còn với

```
<?php
function tang(&$x)
{
        $x = $x +1;
}

$value = 1;
tang($value);
echo($value);
?>
```

Khi đó kết quả hiển thị ra sẽ là 2, tức là thay đổi đã được lưu lại.

Ví dụ:

```
<?php
function getAmount($quantity, $price,&$average)
{
  $result=0;
  $result=$quantity*$price;
  $average=$result*6/12;
  return       $result;
}
$bq=0;
echo "result is : ".getAmount(10,20,$bq);
echo "<br>";
echo "result of Average is : ".$bq;
echo "<br>";
function getAmounts($quantity, $price,$average)
{
  $result=0;
  $result=$quantity*$price;
  $average=$result*6/12;
  return       $result;
}
$bq=0;
echo "result is : ".getAmounts(10,20,$bq);
echo "<br>";
echo "result of Average is : ".$bq;
?>
```

1.13.3.4 Tổ chức hàm tự xây dựng

Trong PHP, các hàm tự xây dựng có thể được tổ chức theo 2 cách:

Cách 1: Hàm PHP được đặt tại trang web cần tới hàm đó.

Ví dụ: Khi viết trang web đăng kí, cần 1 hàm kiểm tra xem email nhập vào có đúng không? Ta sẽ viết luôn hàm kiểm tra email ở đó.

Cách 2: Hàm PHP được viết rời ra một file riêng và ở những nơi cần sử dụng 1 hàm chúng ta sẽ tham chiếu đến file chưa hàm đó để có thể sử dụng hàm đó.

Hai cách tổ chức này cũng tương tự với cách tổ chức hàm, code trong các ngôn ngữ lập trình khác. Và chúng ta cũng dễ dàng nhận thấy được ưu nhược điểm của 2 cách tổ chức này:

a) Hàm PHP được đặt tại trang web cần tới hàm đó.

- Đây là cách viết ban đầu của những người mới học PHP, rất đơn giản để viết. Với các trang nhỏ, xử lí các công việc không lặp lại nhiều cũng có thể sử dụng các hàm này. Trường hợp phải viết hàm có sự pha trộn giữa HTML và PHP cũng là 1 trường hợp cần xử dụng tới cách tổ chức này.

- Một lưu ý khi tổ chức hàm theo cách này là: Mặc dù có thể viết hàm ở bất kì vị trí nào của trang nhưng chúng ta nên chọn cách viết hàm ở đầu trang trừ trường hợp bắt buộc phải viết ở giữa trang.

b) Hàm PHP được viết rời ra một file riêng và ở những nơi cần sử dụng 1 hàm chúng ta sẽ tham chiếu đến file chưa hàm đó để có thể sử dụng hàm đó.

- Đây là một cách viết tốt. Tốt ở đây cả về việc sử dụng lẫn trình bày bởi lẽ việc tách các hàm ra sẽ giúp code dễ tổ chức và quản lí hơn. Với việc các hàm được tách ra cũng sẽ dễ dàng hơn cho việc quản lí code.

- Cần phải có tham chiếu hợp lí để có thể sử dụng lại hàm trong các trang web. Có thể dùng khai báo require() hoặc include() để tham chiếu đến file chứa hàm và sử dụng bình thường như là đã khai báo hàm đó ở trong trang.

- Một khuyến cáo đối với việc tổ chức hàm theo cách này là: Các hàm khi viết ra các file khác nhau phải có sự sắp xếp hợp lí (theo chức năng, theo đối tượng xử lí…)

Lưu ý chung cho cả 2 cách viết là trong mọi trường hợp, các hàm xử lí và các hàm hiển thị nên được tách rời, các chức năng nên được tách rời.

2. Hệ quản trị cơ sở dữ liệu MySQL

2.1 Hệ CSDL quan hệ

Relational Database Management Systems (Hệ Quản trị Cơ Sở Dữ Liệu Quan hệ - RDBMSs) cung cấp phương thức tuyệt vời để lưu trữ và truy xuất lượng thông tin lớn và phức tạp. Nó đã ra đời khá lâu. Thực tế, nó có trước Web, Linux và WindowsNT,cho nên không có gì ngạc nhiên khi có quá nhiều hệ CSDL để chọn lựa. Tất cả các CSDL này đề dựa trên cơ sở SQL (Structure Query Language).

Một số hệ phổ biến như Oracle, Sysbase, Informix, Ms SQL Server, IBM's DB2.
Hệ nguồn mở thông dụng hiện nay là MySQL mà quyển sách này đề cập đến, ngoài

ra còn có hai hệ nguồn mở khác là PostgresSQL đã một thời thay thế MySQL và Interbase là bộ nguồn mở của Borland giới thiệu vào tháng 8/1999.

2.2 Giới thiệu Mysql

MySQL là cơ sở dữ liệu được sử dụng cho các ứng dụng Web có quy mô vừa và nhỏ. Tuy không phải là một cơ sở dữ liệu lớn nhưng chúng cũng có trình giao diện trên Windows hay Linux, cho phép người dùng có thể thao tác các hành động liên quan đến cơ sở dữ liệu.

Cũng giống như các cơ sở dữ liệu, khi làm việc với cơ sở dữ liệu MySQL, bạn đăng ký kết nối, tạo cơ sở dữ liệu, quản lý người dùng, phân quyền sử dụng, thiết kế đối tượng Table của cơ sở dữ liệu và xử lý dữ liệu.

Tuy nhiên, trong bất kỳ ứng dụng cơ sở dữ liệu nào cũng vậy, nếu bản thân chúng có hỗ trợ một trình giao diện đồ hoạ, bạn có thể sử dụng chúng tiện lợi hơn các sử dụng Command line. Bởi vì, cho dù bạn điều khiển MySQL dưới bất kỳ hình thức nào, mục đích cũng quản lý và thao tác cơ sở dữ liệu.

2.3 Kiểu dữ liệu trong MySQL

Trước khi thiết kế cơ sở dữ liệu trên MySQL, bạn cần phải tham khảo một số kiểu dữ liệu thường dùng, chúng bao gồm các nhóm như: numeric, date and time và string.

Đều cần lưu ý trong khi thiết kế cơ sở dữ liệu, bạn cần phải xem xét kiểu dữ liệu cho môt cột trong Table sao cho phù hợp với dữ liệu của thế giới thực.

Điều này có nghĩa là khi chọn dữ liệu cho cột trong Table, bạn phải xem xét đến loại dữ liệu cần lưu trữ thuộc nhóm kiểu dữ liệu nào, chiều dài cũng như các ràng buộc khác, nhằm khai báo cho phù hợp.

2.3.1 Loại dữ liệu numeric

Kiểu dữ liệu numeric bao gồm kiểu số nguyên và kiểu số chấm động, trong trường hợp dữ liệu kiểu dấu chấm động bạn cần phải chỉ rõ bao nhiêu số sau đấu phần lẻ .

Kiểu dữ liệu số nguyên

Loại	Range	Bytes	Diễn giải
Tinyint	-127->128 hay 0..255	1	Số nguyên rất nhỏ.
smallint	-32768 ->32767 hay 0..65535	2	Số nguyên nhỏ.
mediumint	-8388608-> 838860 hay 0..16777215	3	Số nguyên vừa.
int	-2^{31}->2^{31}-1	4	Số nguyên.

	hay 0.. .2^{32}-1		
bigint	-2^{63}->2^{63}-1	8	Số nguyên lớn.
	hay 0..2^{64}-1		

Kiểu dữ liệu số chấm động

Loại	Range	Bytes	Diễn giải
float	phụ thuộc Số thập phân.		Số thập dạng *Single* hay Phân *Double*
Float(M,D)	±1.175494351E-38 ±3.40282346638	4	Số thập phân dạng S*ingle.*
Double(M,D)	±1.7976931348623157308 ±2.2250738585072014E-308	8	Số thập phân dạng *Double.*
Float(M[,D])			Số chấm động lưu dưới dạng *char*.

2.3.2 Loại dữ liệu Date and Time

Kiểu dữ liệu Date and Time cho phép bạn nhập liệu dưới dạng chuỗi hay dạng số

Kiểu dữ liệu Date Time

Loại	Range	Diễn giải
Date	1000-01-01	*Date* trình bày dưới dạng yyyy-mm-dd.
Time	-838:59:59 838:59:59	*Time* trình bày dưới dạng hh:mm:ss.
DateTime	1000-01-01 00:00:00 9999-12-31 23:59:59	*Date* và *Time* trình bày dưới dạng yyyy-mm-dd hh:mm:ss.
TimeStamp[(M)]	1970-01-01 00:00:00	TimeStamp trình bày dưới dạng yyyy-mm-dd hh:mm:ss.
Year[(2|4)]	1970-2069 1901-2155	Year trình bày dưới dạng 2 số hay 4 số.

Đối với kiểu dữ liệu *TimeStamp*, bạn có thể định dạng nhiều.

Trình bày đại diện của TimeStamp

Loại	Hiển thị
TimeStamp	YYYYMMDDHHMMSS
TimeStamp(14)	YYYYMMDDHHMMSS
TimeStamp(12)	YYMMDDHHMMSS
TimeStamp(10)	YYMMDDHHMM
TimeStamp(8)	YYYYMMDD
TimeStamp(6)	YYMMDD
TimeStamp(4)	YYMM
TimeStamp(2)	YY

2.3.3 Loại dữ liệu String

Kiểu dữ liệu String chia làm ba loại, loại thứ nhất như char (chiều dài cố định) và varchar (chiều dài biến thiên). Char cho phép bạn nhập liệu dưới dạng chuỗi với chiếu dài lớn nhất bằng chiều dài bạn đã định nghĩa, nhưng khi truy cập dữ liệu trên Field có khai báo dạng này, bạn cần phải xử lý khoảng trắng. Điều này có nghĩa là nếu khai báo chiều dài là 10, nhưng bạn chỉ nhập chuỗi 4 ký tự, MySQL lưu trữ trong bộ nhớ chiều dài 10.

Ngược lại với kiểu dữ liệu Char là Varchar, chiều dài lớn hất người dùng có thể nhập vào bằng chiều dài bạn đã định nghĩa cho Field này, bộ nhớ chỉ lưu trữ chiều dài đúng với chiều dài của chuỗi bạn đã nhập.

Như vậy, có nghĩa là nếu bạn khai báo kiểu varchar 10 ký tự, nhưng bạn hcỉ nhập 5 ký tự, MySQL chỉ lưu trữ chiều dài 5 ký tự, ngoài ra, khi bạn truy cập đến Field có kiểu dữ liệu này, bạn không cần phải giải quyết khoảng trắng.

Loại thứ hai là Text hay Blob, Text cho phép lưu chuỗi rất lớn, Blob cho phép lưu đối tượng nhị phân. Loại thứ 3 là Enum và Set.

Kiểu dữ liệu String

Loại	Range	Diễn giải
char	-255	Chiều dài của chuỗi lớn nhất characters 255 ký tự.
varchar	1-255	Chiều dài của chuỗi lớn nhất characters 255 ký tự
tinyblob	2^8-1	Khai báo cho *Field* chứa kiểu đối tượng nhị phân cỡ 255 ký tự.
tinytext	2^8-1	Khai báo cho *Field* chứa kiểu chuỗi cỡ 255 ký tự.
blob	2^{16}-1	Khai báo cho *Field* chứa kiểu *blob* cỡ 65,535 ký tự.
text	2^{16}-1	Khai báo cho *Field* chứa kiểu chuỗi dạng văn bản cỡ 65,535 ký tự.

Mediumblob	2^{24}-1	Khai báo cho *Field* chứa kiểu blob vừa khoảng 16,777,215 ký tự.
Mediumtext	2^{24}-1	Khai báo cho *Field* chứa kiểu chuỗi dạng văn bản vừa khoảng 16,777,215 ký tự.
Longblob	2^{32}-1	Khai báo cho *Field* chứa kiểu *blob* lớn khoảng 4,294,967,295 ký tự.
Longtext	2^{32}-1	Khai báo cho *Field* chứa kiểu chuỗi dạng văn bản lớn khoảng 4,294,967,295 ký tự.

2.4 Phát biểu SQL

Hầu hết sản phẩm cơ sở dữ liệu quan hệ hiện nay đều dựa trên chuẩn của SQL và ANSI-SQL, chẳng hạn như SQL Server, Oracle, PostgreSQL và MySQL. Điều này có nghĩa là tất cả những cơ sở dữ liệu quan hệ đều phải có những tiêu chuẩn theo cú pháp SQLvà MySQL cũng không phải là ngoại lệ.

Ngôn ngữ SQL chia làm 4 loại sau:

- DDL (Data Definition Language): Ngôn ngữ định nghĩa dữ liệu, dùng để tạo cơ sở dữ liệu,định nghĩa các đối tượng cơ sở dữ liệu như Table, Query, Views hay các đối tượng khác.
- DML (Data Manipulation Language): Ngôn ngữ thao tác dữ liệu, dùng để thao tác dữ liệu,chẳng hạn như các phát biểu: Select, Inert, Delete, Update, ...
- DCL: (Data Control Language): Ngôn ngữ sử dụng truy cập đối tượng cơ sở dữ liệu, dùng để thay đổi cấu trúc, tạo người dùng, gán quyền chẳng hạn như: Alter, Grant, Revoke, ...
- TCL: (Transaction Control Language): Ngôn sử dụng để khai báo chuyển tác chẳng hạn như: Begin Tran, Rollback, Commit, ...

2.4.1 Phát biểu SQL định nghĩa và thay đổi dữ liệu

2.4.1.1 Tạo cơ sở dữ liệu - Create Database

Khi xây dựng cơ sở dữ liệu, bạn bắt đầu từ mô hình cơ sở dữ liệu *ERD*, hay từ một giai đoạn nào đó trong quy trình phân tích thiết kế hệ thống. Để tạo cơ sở dữ liệu trên *MySQL* hay *SQL Server* bạn sử dụng cú pháp sau:

```
CREATE DATABASE <Database name>
```

2.4.1.2 Xóa cở sở dữ liệu - Drop Database

Lệnh này xóa tất cả các bảng trong database

Cú pháp:

```
DROP DATABASE <Database name>
```

2.4.1.3 Tạo bảng - Creat Table

Cú pháp:

```
CREAT{TEMPORARY}TABLE{IF NOT ESISTS}
Tbl_name [(creat_definition,…)]
[table_options] [select_statement]
```

Lệnh sẽ tạo ra một bảng trong cơ sở dữ liệu hiện tại nếu cơ sở dữ liệu chưa tồn tại thì sẽ có lỗi phát sinh.

Trong đó:
Tbl_name là tên bảng cần tạo
table_options :chỉ ra kiểu của bảng cần tạo và những đặc tính của bảng cần tạo
table_options ={ISAM | MYISAM | HEAP | MERGE}
or AUTO_INCREMENT=#:Giá trị tiếp theo mà bạn muốn đặt cho bảng
or AUTO_ROW_LENGTH=#:Giá trị trung bình độ dài của hàng trong bảng
or COMMENT="string":lời bình luận tối đa 60 kí tự
or MAX_ROWS=# số hàng tối đa mà bạn định lưu trữ
creat_definition:được định nghĩa như sau:
creat_definition:
col_name type [NOT NULL | NULL] [DEFAULT
default_value] { AUTO_INCREMENT}
[PRIMARY KEY}{reference_definition]
or PRIMARY KEY (index_col_name,…)
or KEY [index_name] (index_col_name,…)
or INDEX [index_name] (index_col_name,…)
or UNIQUE {INDEX} [index_name] (index_col_name,…)
or FULLTEXT {INDEX} [index_name] (index_col_name,…)
or {CONSTRANINT symbol] FOREIGN KEY index_name
((index_col_name,…)
[reference_defintion]
or CHECK (expr)
type:Kiểu của cột được tạo trong bảng
REFERENCES tab_name [(index_col_name,…)]
[MATCHFULL | MATCHPARTIAL]
[ON DELETE reference_option]
[ON UPDATE reference_option]
reference_option:
RESTRICT | CASCADE | SET NULL | NO ACTION | SET
DEFAULT

NOT NULL: Nếu cột được chỉ định là NOT NULL thì khi nhập liệu ta bắt buộc phải nhập dữ liệu cho cột này .

NULL :Đối với cột kiểu timestamp sẽ có sự khác biệt về giá trị NULL so với những cột có những kiểu khác,bạn không thể đưa ra giá trị

NULL với cột có kiểu timestamp nếu đặt là NULL thì nó sẽ tự động đặt ngày giờ hiện tại,bởi vì cột có kiểu stamp sử dụng cách thức này nên thuộc tính NULL hoặc NOT NULL sẽ không được đặt như những cột thông thường và nó sẽ được lờ đI nếu bạn chỉ định đIều này .

DEFAULT<giá trị ngầm định >:cột này sẽ tự động được đặt là giá trị ngầm định nếu như ta bỏ qua không nhập giáv trị cho cột này .Giá trị ngầm định phải là hằng số theo nghĩa là ta không thể đặt giá trị mặc định cho cột là giá trị trả về của một hàm ,nếu cột được khai báo là NOT NULL thì giá trị ngầm định sẽ phụ thuộc kiểu của cột Nếu không đặt DEFAULT .

-AUTO_INCREMENT:chỉ có đối với cột số nguyên cột này sẽ tự động được tăng khi nhập dữ liệu .Nếu bạn xoá một hàng chứa giá trị lớn nhất của cột auto_increment thì giá trị này sẽ dùng lại cho lần nhập liệu sau (đIều này chỉ xảy ra đối với bảng loại ISAM mà không xảy đối với bảng MYISAM).

chú ý:Mỗi hàng chỉ có một cột đặt là auto_increment và phải được đặt là chỉ số

-UNIQUE KEY:đây là khoá của bảng nó chỉ có thể nhận một giá trị nhất định,sẽ có lỗi nếu như ta chèn vào một hàng của bảng giá trị khoá trùng với khoá của hàng đã tồn tại .

-PRIMARY KEY:Định nghĩa khoá chính của bảng,là khoá duy nhất với ràng buộc rằng tất cả các giá trị khoá đều phải đặt NOT NULLmỗi bảng chỉ có thể có một khoá chính.

-FOREIGN KEY :khoá ngoại dùng làm khoá chính của bảng khác.

-Mệnh đề SELECT :Nếu có thêm mệnh đề SELECT sau câu lệnh CREATE TABLE thì Mysql sẽ tạo các trường mới cho tất cả các thành phần nằm trong câu lệnh SELECT

Ví dụ:

```
/* Tạo bảng danh sách khách hàng thường xuyên */
CREATE TABLE tblcustomers (
  CustID int(3) unsigned NOT NULL auto_increment,
  Username varchar(20) NOT NULL DEFAULT '' ,
  Password varchar(10) NOT NULL DEFAULT '' ,
  CustName varchar(50) ,
  Address varchar(100) ,
  Tel varchar(20) ,
  FaxNo varchar(10) ,
  Email varchar(50) ,
  Contact varchar(50) ,
  CountryCode char(3) ,
  ProvinceCode char(3) ,
  PRIMARY KEY (CustID),
  INDEX CustID (CustID)
);
```

```
/* Tạo bảng hợp đồng mua hàng qua mạng  */
CREATE TABLE tblorders (
  OrderID int(3) NOT NULL auto_increment,
  OrderDate date ,
  CustID int(11) ,
```

```
  Description varchar(100) DEFAULT '0' ,
  TranID tinyint(3) DEFAULT '0' ,
  PaymentID tinyint(3) DEFAULT '0' ,
  Amount float DEFAULT '0' ,
  ShipCost float DEFAULT '0' ,
  TotalAmount float DEFAULT '0' ,
  PRIMARY KEY (OrderID),
  INDEX OrderID (OrderID)
);
```

```
/* Tạo bảng hợp đồng chi tiết mua hàng qua mạng  */
CREATE TABLE tblorderdetails (
  ItemID int(3) unsigned DEFAULT '0' ,
  OrderID int(3) unsigned DEFAULT '0' ,
  No tinyint(3) unsigned DEFAULT '0' ,
  Qtty int(3) unsigned DEFAULT '0' ,
  Price int(3) unsigned DEFAULT '0' ,
  Discount int(3) unsigned DEFAULT '0' ,
  Amount bigint(3) unsigned DEFAULT '0'
);
```

Một số quy định khi thiết kế Table

Tên cột - Column Name

Đặt tên cột cũng giống như đặt tên bảng, có rất nhiều quy tắc đặt tên (như đã trình bày ở trên phần *table*), nhưng khuyến khích bạn nên theo một số quy tắc cơ bản sau:

- Tên cột bắt đầu chữ hoa, còn lại bằng chữ thường.
- Tên ngắn gọn và đầy đủ ý nghĩa.
- Không nên đặt tên cột có khoảng trắng, sau này bạn sẽ gặp những phiền toái khi tham chiếu đến cột đó.
- Không đặt tên cột trùng với những từ khoá, từ dành riêng, và những ký tự đặc biệt như những phép toán hay toán tử khác.
- Chú ý, nên đặt tên cột cùng tên những cột có quan hệ với những bảng khác trong cùng cơ sở dữ liệu, giúp dễ hiểu và tránh bị nhầm lẫn.

Một số người thích thêm vào dấu gạch chân (_) để phân biệt ý nghĩa hay tên gọi của cột, điều này là tùy vào sở thích của bạn. Tuy nhiên chúng tôi không thích qui tắc này.

Nhưng đối với kinh nghiệm lập thiết kế xây dựng cơ sở dữ liệu thì bạn không nên dùng dấu gạch dưới _, và dĩ nhiên trong nhiều trường hợp khác bạn sẽ cảm thấy khó chịu khi thêm một dấu _ trong tên của đối tượng của cơ sở dữ liệu.

Mặc dù không có vấn đề gì cho cú pháp hay các phát biểu tham chiếu đến chúng, nhưng bạn sẽ thấy tại sao chúng ta không nên dùng dấu gạch chân (_) khi đặt tên đối tượng hay tên cơ sở dữ liệu trong *MySQL*.

- Nếu bạn đặt tên có dấu _ ,bạn phải tốn thời gian hay năng lượng cho hành động tạo ra dấu _
- Trong chừng mực hay giới hạn nào đó do hiệu ứng của *Font* chữ có thể phát sinh lỗi sẽ gây

ra nhầm lẫn cho người lập trình.

- Nói tóm lại là bạn sẽ mất thêm thời gian lưu tâm đến chúng.

Kiểu dữ liệu - Data type

Như đã trình bày các lại dữ liệu trong phần trên, khi xây dựng cơ sở dữ liệu, tất cả những trường trong bảng cần phải có kiểu dữ liệu cụ thể. Vấn đề quan trọng là chọn kiểu dữ liệu nào cho phù hợp với dữ liệu mà người dùng sẽ nhập vào.

Để thiết kế dữ liệu phù hợp với thực tế, ngoài tính ứng dụng hợp với ngữ cảnh bạn cũng cần quan tâm đến kiểu dữ liệu tương thích và chiều dài của từng cột. Chẳng hạn như:

[CustID] [varchar] (10)
/ hay */*
[CustID] int

Giá trị mặc định - Default

Thông thường khi tạo ra một cột trong bảng đôi khi chúng ta cần áp dụng giá trị mặc định, không chỉ cho trường hợp số liệu không nhập từ bên ngoài mà còn cho các cột tự động có giá trị tự sinh. Với những lý do như vậy, chúng ta cần có một số giá trị mặc định cho những cột cần thiết,

ví dụ :

- Nếu cột đó là số chúng ta có giá trị mặc định là 0
- Nếu cột đó là ngày tháng chúng ta có giá trị mặc định là ngày nào đó (như 0000-00-00 là *CurDate*())
- Nếu cột đó có giá trị là 0 hoặc 1, bạn có thể khai báo giá trị mặc định là 0 hoặc 1
- Nếu cột đó là chuỗi chúng ta có giá trị mặc định như là 'A'

Số tự động auto_increment

auto_increment là khái niệm cực kỳ quan trọng trong *MySQL* (tương đương với *Identity* trong*SQL Server, Autonumber* trong *MS Access*). Khi bạn muốn một cột có giá trị tăng tự động như *AutoNumber/Identity*, bạn nên định nghĩa cột đó như *auto_increment*,.

Khi sử dụng *auto_increment* làm số tăng tự động thì kiểu dữ liệu là số nguyên hoặc số nguyên lớn. Trong trường hợp, bạn khai báo số tự động trong *SQL Server*, bạn cần phải khai báo thêm các thông số như *seed*. *Seed* là giá trị khởi đầu khi *SQL Server* tự động tăng giá trị, *Increament* là bước tăng, nó cho biết mỗi lần tăng cần bao nhiêu giá trị.

Vì dụ khi tạo*auto_increment* cho cột *ItemID [Int] auto_increment*, nghĩa là bắt đầu số 1 và mỗi lần tăng 1 số. Kết quả bạn sẽ có là 1,2,3,4, ...*n.*

Trong phát biểu *SQL* của *MySQL*, để tạo bảng có gá trị tăng tự động bạn chỉ cần khai báo tên cột, kiểu dữ liệu *Int (Integer)* và *auto_increment* như sau:

IDNO Int auto_increment NOT NULL

NULL / NOT NULL

Đây là trạng thái của một cột trong bảng cho phép chấp nhận giá trị *NULL* hay không? Nếu bạn chỉ ra ràng buộc giá trị *NOT NULL* thì bắt buộc phải có giá trị trong cột này mỗi khi mẩu tin được nhập vào.

Đối với một số kiểu dữ liệu không cho phép *NULL* bạn nên thiết lập giá trị mặc định cho cột đó, ví dụ như kiểu dữ liệu bit không cho phép *NULL*.

Trong phát biểu SQL tạo bảng, bạn chỉ cần khai báo *NULL* hay *NOT NULL* sau kiểu dữ liệu của cột đó.

2.4.1.4 Sửa cấu trúc bảng

Cú pháp :

```
ALTER[IGNORE] TABLE tbl_name
alter_spec[,alter_spec...] alter_specification :
ADD[COLUMN] create_definition[FIRST |AFTER colum_name]
Or ADD[COLUMN] (create_definition, create_definition,..)
Or ADD INDEX[index_name] (index_col_name,..)
Or ADD PRIMARY KEY(index_col_name,..)
Or ADD UNIQUE[index_name](index_col_name,..)
Or ADD FULLTEXT[index_name](index_col_name,..)
OR ADD[CONSTRAIN symbol] FOREIGN KEY index_name
(index_col_name)
[referent_definition]
or ALTER[COLUM] col_name{SET DEFAULT literal |DROP
DEFAULT}
OR CHANGE[COLUM] old_col_name create_definition
Or MODIFY[COLUMN] create_definition
OR DROP[COLUMN] column_name
OR DROP PRIMARY KEY
OR DROP INDEX index_name
Or RENAME[TO] new_tbl_name
Or table_options
```

Lệnh ALTER TABLE cho phép bạn sửa cấu trúc của một bảng đã có,ví dụ bạn có thể thêm hoặc xoá cột,tạo hoặc huỷ chỉ số,thay đổi kiểu của cột đã có ,hoặc đổi tên cột hoặc đổi tên bảng ,bạn có thể thay đổi lời chú thích cho bảng hoặc kiểu của bảng .

Nếu sử dụng lệnh ALTER TABLE để thay đổi một cột đã được chỉ định rõ nhưng DESCRIPE tbl_name chỉ ra rằng cột không thể thay đổi ,có nghĩa là Mysql bỏ qua sự thay đổi này do một lý do nào đó .Ví dụ nếu bạn cố thay đổi một cột có kiểu varchar sang cột kiểu char thì Mysql sẽ không thực hiện việc thay đổi nếu trong bảng có các cột có độ dài thay đổi .

Lệnh ALTER TABLE thực hiện việc tạo một bản sao tạm thời của bảng nguồn việc sửa đổi thực hiện trên bảng sao này ,bảng nguồn sẽ được xoá khi bảng mới được đổi tên vì hoạt động theo cách này nên việc cập nhật sẽ được thực hiện một cách tự động một lần nữa tới bảng mới .

Để sử dụng lệnh ALTER TABLE bạn cần phải có quyền : Select,insert,update,create,drop đối với bảng.

-Một số ví dụ dùng lệnh ALTER TABLE :

Để sửa đổi tên một cột :

```
ALTER TABLE hocsinh CHANGE hokhautt,hokhautt varchar(40);
```

Nếu bạn muốn thay đổi kiểu của cột mà không muốn đổi tên thì bạn vẫn phải viết tên của cột 2 lần .

```
ALTER TABLE hocsinh CHANGE hokhautt,hokhautt CHAR(40);
```

Ta cũng có thể dùng MODIFY để thay đổi kiểu của cột

```
ALTER TABLE hocsinh modify ngaysinh char(10);
```

Nếu sử dụng lệnh CHANGE hoặc modify với một cột đã sắp xếp mà file chỉ số đã tồn tại thì bạn không thể sắp xếp nhiều hơn số kí tự đã được chỉ số hoá .

Khi dùng change hoặc modify Mysql sẽ cố gắng chuyển dữ liệu sang kiểu mới một cách tốt nhất .

-drop index :huỷ file chỉ số nếu cột được sắp xếp lôgic (chỉ số) mà ta xoá nó thì nó cũng được xoá ở thành phần chỉ số này cũng sẽ được xoá Drop primary key xoá khoá chính

-ORDER BY:cho phép tạo một bảng mới với hàng được chỉ ra ở mệnh đề order by ,chú ý rằng bảng sẽ không còn như cũ trong mệnh đề order sau khi chèn và xoá

Đổi tên bảng :

```
ALTER TABLE old_name rename new_name;
```

để chuyển một cột từ kiểu integer sang tinyint not null(với cùng tên) và đổi cột có tên là hoten char(10) -> char(20) và đổi từ hoten->hovaten

```
ALTER TABLE sinhvien modify masv tinyint not null ,change hoten hovaten
char(20);
```

Bổ sung thêm trường ngaysinh

```
ALTER TABLE sinhvien add ngaysinh timestamp ;
```

Để chuyển cột thành khoá chính

```
ALTER TABLE sinhvien add primary key(masv);
```

2.4.1.5 Xóa bảng

```
/* Phát biểu DROP TABLE chỉ rõ bảng nào cần xoá,
nếu xoá nhiều bảng thì bạn cần dùng dấu phẩy (,)  */
DROP  TABLE tblCustomers, tblSuppliers
```

2.4.2 Phát biểu SQL quản lý người dùng

2.4.2.1 Tạo người dùng

Phát biểu:

```
GRANT
Select, Insert, Update,
Delete, Index, Alter,
Create, Drop, References
ON *.* TO 'myis'@'%'
IDENTIFIED BY '12345678'
```

Trong phát biểu trên, vừa tạo ra User có tên myis, với hostname là cơ sở dữ liệu hiện hành, password là 1234 và được các đặt quyền Select, Insert, Update, Delete, Index, Alter, Create, Drop trên cơ sở dữ liệu hiện hành.

Trong trường hợp bạn tạo ra một Username không cung cấp các đặt quyền trên cơ sở dữ liệu, bạn có thể thực hiện như phát biểu tạo username: test, password: 1234 sau:

```
GRANT
usage
ON *.* TO 'test'@'%'
IDENTIFIED BY '1234'
```

2.4.2.2 Cấp quyền cho người dùng

Các đặt quyền Select, Insert, Update, Delete, Index, Alter, Create, Drop trên cơ sở dữ liệu

Các đặt quyền trên cơ sở dữ liệu

Loại	áp dụng	Diễn giải
select	tables, columns	Cho phép *user* truy vấn mẩu tin từ *Table.*
insert	tables, columns	Cho phép *user* thêm mới mẩu tin vào *Table.*
update	tables, columns	Cho phép *user* thay đổi giá trị của mẩu tin tồn tại trong *Table.*
delete	tables	Cho phép user mẩu tin tồn tại trong *Table.*
index	tables	Cho phép *user* thêm mới hay xoá chỉ mục của *Table.*
alter	tables	Cho phép *user* thay đổi cấu trúc của đối tượng *Table* hay Database tồn tại, như thêm cột vào trong *Table* tồn tại, thay đổi kiểu dữ liệu của cột dữ liệu, ..

create	databases	Cho phép *user* tạo mới đối tables tượng *Table* hay Database.
Drop	databases	Cho phép user xoá đối tables tượng *Table* hay *Database*.

Xuất phát từ các quyền có ảnh hưởng đến cấu trúc cơ sở dữ liệu, các đối tượng của cơ sở dữ liệu và dữ liệu, bạn có thể xem xét kỹ càng trước khi cấp quyền cho user àm việc trên cơ sở dữ liệu.

Ngoài các quyền trên, trong MySQL còn có một số quyền không gán mặc định như trong bảng sau đây:

Các đặt quyền quản trị trên cơ sở dữ liệu

Loại	Diễn giải
reload	Cho phép người quản trị nạp lại các *Table*, quyền, *host, logs* và *Table*.
shutdown	Cho phép người quản trị chấm dứt hoạt động *MySQL Server*.
process	Cho phép người quản trị xem quá trình thực hiện của trình chủ và có thể chấm dứt một số quá trình đang thực thi.
file	Cho phép dữ liệu ghi vào *Table* từ tập tin.

Lưu ý: Những **username** bình thường không nên cấp quyền như trong bảng trên cho họ, trong trường hợp bạn muốn cấp tất cả các quyền trong bảng 2 bảng trên cho **username** khi tạo ra họ, bạn sử dụng từ khoá All thay vì All Privileges trong phát biểu tạo user như sau:

```
GRANT
ALL
ON *.* TO 'etest'@'%'
IDENTIFIED BY '12345678'
```

Tương tự như vậy, trong trường hợp bạn không cung cấp bất kỳ đặt quyền nào trên cơ sở dữ liệu hiện hành, bạn có thể khai báo phát biểu cấp quyền như sau:

```
GRANT
usage
ON *.* TO 'etest'@'%'
IDENTIFIED BY '12345678'
```

2.4.2.3 Xoá quyền của người dùng

Để xoá các quyền của user từ cơ sở dữ liệu hiện hành, bạn có thể sử dụng phát biểu SQL có tên Revoke, phát biểu Revoke ngược lại với phát biểu Grant.

Nếu bạn xoá một số quyền của user, bạn có thể sử dụng khai báo như phát biểu sau :

```
Revoke privileges [(columns)]
```

```
itemON
From username
```

Trong trường hợp xoá tất cả các quyền của user, bạn có thể sử dụng phát biểu như sau:

```
RevokeAll
itemON
From username
```

Nếu user đó được cấp quyền với tuỳ chọn Grant Option, để xoá các quyền đó của user, bạn có thể khai báo như sau:

```
Revoke Grant Option
itemON
From username
```

Để tham khảo chi tiết quá trình cấp và xoá quyền của một user, bạn có thể tham khảo một số phát biểu như sau:

Gán quyền Administrator cho user có tên fred trên mọi cơ sở dữ liệu trong MySQL, password của anh ta là abc123, bạn có thể khai báo như sau:

```
Grant all
On *
To fred indetifyed by 'abc123'
With Grant Option;
```

Nếu bạn không muốn user có tên fred trong hệ thống, bạn có thể xoá bằng cách khai báo phát biểu sau:

```
Revokeall
On *
From fred;
```

Tạo một user có tên etest với password là 12345678, được làm việc trên cơ sở dữ liệu Test, không cấp quyền cho user này, bạn có thể khai báo như sau:

```
Grant usage
On Test.*
To etest identified by '12345678';
```

Tương tự như vậy, trong trường hợp bạn muốn cấp một số quyền cho user có tên etest trên cơ sở dữ liệu Test, bạn có thể khai báo như sau:

```
Grant select, insert, delete, update, index, drop
On Test.*
etest;To
```

Nếu bạn muốn xoá bớt một số quyền của user có tên etest trên cơ sở dữ liệu Test, bạn có thể khai báo như sau:

```
Revoke update, delete, drop
On Test.*
```

```
From etest;
```

Nhưng trong trường hợp xoá tất cả các quyền của user có tên etest trên cơ sở dữ liệu Test, bạn có thể khai báo:

```
RevokeAll
On Test.*
From etest;
```

2.4.3 Phát biểu SQL thao tác dữ liệu

Phát biểu SQL bao gồm các loại như sau:

- SELECT (Truy vấn mẩu tin).
- INSERT (Thêm mẩu tin).
- UPDATE (Cập nhật dữ liệu).
- DELETE (Xoá mẩu tin).

2.4.3.1 Phát biểu SQL dạng SELECT

2.4.3.1.1 Phát biểu SELECT

Phát biểu Select dùng để truy vấn dữ liệu từ một hay nhiều bảng khác nhau, kết quả trả về là một tập mẩu tin thoã các điều kiện cho trước nếu có.
Cú pháp của phát biểu SQL dạng SELECT:

```
SELECT <danh sách các cột>
[FROM <danh sách bảng>]
[WHERE <các điều kiện ràng buộc>]
[GROUP BY <tên cột / biểu thức trong SELECT> ]
[HAVING <điều kiện bắt buộc của GROUP BY>]
[ORDER BY <danh sách cột>]
[LIMIT FromNumber | ToNumber]
```

Danh sách các cột: Khai báo các tên cột, biểu thức kết hợp giữa các cột của Table bạn cần truy lục. Trong trường hợp có hai cột cùng tên của hai Table trong phát biểu, bạn cần phải chỉ định tên Table đi trước.

Ví dụ : Phát biểu SELECT

```
Select ItemID,ItemName
From tblItems
Where Cost>100;
Select tblOrders.OrderID,OrderDate,ItemID,Qtty
From tblOrders,tblOrderDetails
Where tblOrders.OrderID = _ tblOrderDetail.OrderID;
```

2.4.3.1.2 Phát biểu SELECT với mệnh đề FROM

Phát biểu SQL dạng SELECT là một trong những phát biểu yêu cầu MySQL truy lục dữ liệu trên cơ sở dữ liệu chỉ định. SELECT dùng để đọc thông tin từ cơ sở dữ liệu theo những trường quy định, hay những biểu thức cho trường đó.

Mệnh đề FROM chỉ ra tên một bảng hay những bảng có quan hệ cần truy vấn thông tin. Thường chúng ta sử dụng công cụ MySQL-Front | Query để thực thi phát biểu SQL.

Sau khi thực thi phát biểu SQL, kết quả trả về số mẩu tin và tổng số mẩu tin được lấy ra từ bảng.

Dấu * cho phép lọc mẩu tin với tất cả các trường trong bảng, nếu muốn chỉ rõ những trường nào cần lọc bạn cần nêu tên cụ thể những trường đó.

Để tiện tham khảo trong giáo trình này chúng tôi sử dụng một phần cơ sở dữ liệu có sẵn của MySQL, đồng thời bổ sung thêm cơ sở dữ liệu dành cho ứng dụng bán hàng qua mạng.

Cơ sở dữ liệu bán hàng qua mạng có tên là Test, và bao gồm nhiều bảng. Bằng phát biểu SELECT chúng ta có thể biết số bảng hay đối tượng khác đang có trong cơ sở dữ liệu Test

Ví dụ : Thực thi phát biểu SQL SELECT hệ thống

```
show tables
from Test
/* Hiển thị tất cả tên bảng của cơ sở dữ liệu hiện hành */
```

Kết quả trả về danh sách bảng như sau:

TABLES_IN_TEST
---
tblCountries
tblProvinces
tblAuthors
tblPayment
tblItemsion
tblCustomers
tblSoftware

Ghi chú:

Bạn có thể sử dụng phát biểu SQL trên để hiển thị những đối tượng trong cơ sở dữ liệu, bằng cách thay thế các tham số và điều kiện.

Cú pháp đơn giản

```
Select *
From tablename
/* Lọc tất cả số liệu của tất cả các cột (field) của tablename*/
```

```
Select  field1,field2
From tablename
```

```
/* Lọc tất cả số liệu của 2 field: field1, field2 của tablename*/

Select *
From tablename
Limit 0,10
/* Lọc top 10 mẫu tin  đầu tiên của tất cả các field của tablename*/

Select field1, field2
From tablename
Limit 0,10
/*  Lọc  top  10  mẫu  tin   đầu  tiên  của  2   fields  field1,  field2  của
tablename*/
```

Ví dụ : phát biểu phát biểu SQL dạng Select

```
Select *
From tblCountries
/* Liệt kê tất cả các quốc gia trong bảng tblCountries hoặc bạn có thể liệt
kê tên như phát biểu sau */

Select CountryName
From tblCountries
```

Kết quả trả về như sau:

CountryCode	CountryName
VNA	Vietnam
SNG	Singapore
USS	United Stated
UKD	United Kingdom
GER	Germany
CAM	Cambodia
THA	Thai Land
MAL	Malaysia INC Indonesia CHN China

2.4.3.1.3 Phát biểu SELECT với mệnh đề WHERE

Khi bạn dùng mệnh đề WHERE để tạo nên tiêu chuẩn cần lọc mẩu tin theo tiêu chuẩn được định nghĩa, thông thường WHERE dùng cột (trường) để so sánh với giá trị, cột khác, hay biểu thức chứa cột (trường) bất kỳ có trong bảng. Phát biểu SQL dạng Select với mệnh đề Where cú pháp có dạng như sau:

```
Select *
from tablename
where conditions
Select field1, field2, field3
from tablename
where conditions
```

Với conditions trong cả hai phát biểu trên được định nghĩa điều kiện truy vấn như khai báo sau:

```
Select *
From tablename
where field1>10
select *
from tblCountries
where CountryCode in('VNA','CHN')
```

Các phép toán so sánh trong conditions bao gồm:

- ♦ > : lớn hơn where Amount > 100000;
- ♦ < : nhỏ hơn where Amount < 100000;
- ♦ >= : lớn hơn hoặc bằng where Amount >= 100000;
- ♦ >= : nhỏ hơn hoặc bằng where Amount <= 100000;
- ♦ = : bằng where CustID='12';
- ♦ != : Khác where CustID!='12';
- ♦ <> : Khác where CustID<>'12';

Các phép toán logic có thể sử dụng trong conditions

- ♦ and : Phép toán "and"

```
SELECT *
FROM tblOrders
Where Amount!>100000
And CustID='12';
```

- ♦ Or : Phép toán "or"

```
SELECT *
FROM tblOrderDetails
Where Amount!>100000
Or CustID='12';
```

- ♦ Not : Phép toán phủ định (not)

```
SELECT *
FROM tblOrders
where OrderDate is not null;
```

- ♦ Not in : Phép toán phủ định (not in)

```
SELECT *
FROM tblOrders
where OrderID not in ('12','15');
```

- ♦ Between: Kết quả thuộc trong miền giá trị

```
SELECT *
FROM tblOrders
Where Amount  between 10
And 500;
```

♦ Like : Phép toán so sánh gần giống, sử dụng dấu % để thể hiện thay thế bằng ký tự đại diện

```
SELECT  *
FROM tblCustomers
where CustName like '%A';
```

♦ Not Like : Phép toán phủ định so sánh gần giống, sử dụng dấu % để thể hiện thay thế bằng ký tự đại diện

```
SELECT  *
FROM tblCustomers
where CustName not like '%A';
```

♦ IN : Phép toán so sánh trong một tập hợp

```
SELECT  *
FROM tblOrders
Where OrderID in ('100','200','300');
```

Ví dụ : Ví dụ về SQL dạng SELECT và Where

```
 /*  > :  lớn hơn */

Select *
From tblOrders
Where Amount > 100000;
```

```
/*  < : nhỏ hơn */

Select*
From tblOrders
Where Amount < 100000;
```

```
/*  >=: lớn hơn hoặc bằng */

Select*
From tblOrders
Where Amount >= 100000;
```

```
/*  >=: nhỏ hơn hoặc bằng */

Select  *
From tblOrders
Where Amount <= 100000;
```

```
/*  = : bằng */

Select*
From tblOrders
Where CustID='12';
```

```
/*  != :Khác  */

Select*
```

```
From tblOrders
Where CustID !='12';
```

```
/*  <>: Khác  */

Select*
From tblOrders
Where CustID <>'12';
```

```
/*  !> : Không lớn hơn  */

Select*
From tblOrders
Where Amount !> 100000;
```

```
/*  !< : Không nhỏ hơn  */

Select *
From tblOrders
Where Amount !< 100000;
```

Các phép toán logic

```
/* and  : Phép toán và  */

Select*
From tblOrders Where Amount !>100000
And CustID='12';
```

```
/*  Or  : Phép toán hoặc  */

Select*
From tblOrders
Where Amount !>100000
Or CustID='12';
```

```
/*  Not : Phép toán phủ định  */

Select*
From tblOrders
Where OrderDate is NOT NULL;
```

```
/*  Between: giá trị nằm trong miền  */

Select*
From tblOrders
Where Amount
Between 10 and 500;
```

```
/*  Like : Phép toán so sánh gần giống, sử dụng dấu %
để thể hiện thay thế bất kỳ ký tự  */

Select*
From tblOrders
Where Descriion like'%A'
Or CustID ='152';
```

```
/* Not Like  : Phép toán phủ định so sánh gần giống,
sử dụng dấu % để thể hiện thay thế bất kỳ ký tự  */

Select*
From tblOrders
Where Descriion not like '%A'
Or CustID ='152';
```

```
/*  IN: Phép toán so sánh trong một tập hợp  */

Select *
From tblOrders
Where OrderID in ('134','244','433');
```

```
/* Not IN      : Phép toán phủ định so sánh trong một tập hợp  */

Select *
From tblOrders
Where OrderID not in ('134','244','433');
```

2.4.3.1.4 Phát biểu SELECT với mệnh đề ORDER BY

Thông thường, trong khi truy vấn mẩu tin từ bảng dữ liệu, kết quả hiển thị cần sắp xếp theo chiều tăng hay giảm dựa trên ký tự ALPHABET. Nhưng bạn cũng có thể sắp xếp theo một tiêu chuẩn bất kỳ, chẳng hạn như biểu thức.

Khi sắp xếp dữ liệu trình bày trong kết quả, cần phải chọn trường hay biểu thức theo trật tự tăng dần hoặc giảm dần.

Cú pháp cho mệnh đề ORDER BY cùng với trạng thái tăng hay giảm, ứng với ASC sắp xếp tăng dần, DESC giảm dần.

Cú pháp có dạng như sau:

```
Order by columnname DESC
Order by columnname1 + columnname2 DESC
Order by columnname ASC
Order by columnname1 ASC, columnname2 DESC
```

Ví dụ : SELECT với mệnh đề Order by DESC

```
/*-- Giảm dần theo thời gian */
Select OrderID , OrderDate, CustID, Amount
From tblOrders
Where Amount >1000
Order by OrderDate DESC
```

Kết quả trả về như sau:

OrderID	OrderDate	CustID	Amount
17	2001-09-20	12	178.243
18	2001-09-20	12	2.78534

16	2001-09-19	12	398.798
15	2001-09-18	12	5.758.876
14	2001-09-17	12	5.539.647
12	2001-09-16	12	1.330
13	2001-09-16	12	1.585.563
31	2001-09-16	13	459.525
11	2001-09-15	11	1.401.803
28	2001-09-15	13	1.45200

Ví dụ : SQL dạng SELECT với mệnh đề Order by và ASC

```
/*-- Tăng dần theo thời gian */
Select OrderID , OrderDate, CustID,  Amount
From tblOrders
Where Amount >1000
  Order by OrderDate ASC
```

Kết quả trả về như sau

OrderID	OrderDate	CustID	Amount
01	2001-09-05	10	2.903.576
02	2001-09-05	10	48.168.567
03	2001-09-05	10	5.107.032
04	2001-09-08	10	2.355.537
05	2001-09-08	16	1.817.487
06	2001-09-10	16	26.000
19	2001-09-10	12	575.667
29	2001-09-10	13	466.500
07	2001-09-11	16	186.782
23	2001-09-11	12	459.162

Nếu muốn sắp xếp theo nhiều cột (trường), chỉ cần sử dụng dấu phẩy (,) để phân cách các cột.

Ví dụ : SELECT với mệnh đề Order by với 2 cột dữ liệu

```
Select OrderID , OrderDate, CustID, Amount
From tblOrders
Where Amount >1000
Order by OrderID,CustID DESC
```

Kết quả trả về như sau:

OrderID	OrderDate	CustID	Amount
31	2001-09-16	13	459.525
30	2001-09-15	13	153.120
29	2001-09-10	13	466.500
28	2001-09-15	13	145.200
27	2001-09-14	13	603.033

26	2001-09-13	13	230.000
25	2001-09-11	13	244.904
24	2001-09-12	13	1.367.228
23	2001-09-11	12	459.162
19	2001-09-10	12	575.667

Nếu muốn sắp xếp theo nhiều trường kết hợp, chỉ cần dùng thứ tự từng cột cách nhau bằng dấu +

Ví dụ : SELECT với mệnh đề Order by hợp 2 cột

```
/*-- Giảm dần theo số OrderID và CustID */
Select OrderID , OrderDate, CustID, Amount
From tblOrders
Where Amount >1000
Order by OrderID + CustID DESC
```

Kết quả trả về như sau:

OrderID	OrderDate	CustID	Amount
31	2001-09-16	13	459.525
30	2001-09-15	13	153.120
29	2001-09-10	13	466.500
28	2001-09-15	13	145.200
27	2001-09-14	13	603.033
26	2001-09-13	13	230.000
25	2001-09-11	13	244.904
24	2001-09-12	13	1.367.228
23	2001-09-11	12	459.162
19	2001-09-10	12	575.667

Nếu trong phát biểu SQL dạng SELECT có nhiều bảng kết hợp lại với nhau, bạn có thể dùng thêm tên bảng ứng với cột của bảng đó. Phần này sẽ được diễn giải cụ thể hơn trong phần kế tiếp (JOIN -Phép hợp).

2.4.3.1.5 Phát biểu SELECT với mệnh đề GROUP BY

Khi truy vấn mẩu tin trên một hay nhiều bảng dữ liệu, thông thường có những nghiệp vụ thuộc trường nào đó có cùng giá trị, ví dụ khi hiển thị hợp đồng phát sinh trong tháng, kết quả sẽ có nhiều hợp đồng của khách hàng lặp đi lặp lại.

Ví dụ : SQL dạng SELECT với mệnh đề Order by

```
Select CustID, Amount
from tblOrders
```

Với phát biểu trên kết quả trả về như sau:

CustID	Amount
10	2.903.576
10	48.168.567
10	5.107.032
10	2.3555347
16	181.074.847
16	26.000
16	1.867.682
16	3.600.000
16	195.713.899
16	961.804.228
16	140.180.347
12	138
12	158.555.638
12	5.539.647
12	575.887.767
12	39.879.489
12	17.824.938
12	278.503.048
12	5.756.667
12	459.162
13	136.727.628
13	244.904
13	230.000
13	603.033
13	1.452.000
13	4.665.100
13	1.531.200
13	459.525

Trong báo cáo chúng ta lại cần phải biết mỗi khách hàng có bao nhiêu lần trả tiền, tổng số tiền của mỗi khách hàng đã trả là bao nhiêu?

Để làm điều này, chúng ta sử dụng mệnh đề GROUP BY trong phát biểu SQL dạng SELECT cùng với một số hàm trong MySQL và nhóm mẩu tin bằng mệnh đề Group By.

Ví dụ : SQL dạng SELECT với mệnh đề Group By

```
Select CustID, count (CustID),
Sum(Amount)
From tblOrders
Group by CustID
Order by CustID
```

Kết quả trả về như sau:

CustID	count	Amount
16	7	2.956.562.368

12	9	3.843.022.604
13	8	145.913.378
10	4	72.382.804

2.4.3.1.6 Phát biểu SELECT với AS

Khi cần thiết phải thay đổi tên trường nào đó trong câu truy vấn, bạn chỉ cần dùng phát biểu AS. AS cho phép ánh xạ tên cũ, hay giá trị chưa có tên thành tên mới (header).

Ví dụ, khi sử dụng GROUP BY ở trong phần trên, những cột tạo ra từ các phép toán count, sum, max, min, ... cho ra kết quả không có header, nghĩa là không có tên cột để tham chiếu trong khi gọi đến chúng. Chúng ta phải cần phát biểu AS cho những trường hợp này.

Ví dụ : SQL dạng SELECT với AS và các hàm

```
Select CustID,
Count (CustID) as No,
Sum(Amount) as TIENHD,
Max(Amount) as HDLONNHAT,
Min(Amount) as HDNHONHAT,
Avg(Amount) as TRUNGBINH
From tblOrders
Group by CustID
Order by CustID
```

Kết quả hiển thị như sau:

CustID	No	TIENHD	HDLONNHAT	HDNHONHAT	TRUNGBINH
16	7	2956562368	1.95713899	26000	422366052
12	9	3843022604	39879489	459162	427002511
13	8	145913378	1.36727628	230000	18239172.25
10	4	72382804	48168567	2903576	18095701

2.4.3.1.7 Phát biểu SELECT với Limit N , M

Phát biểu SQL dạng SELECT cho phép truy lục chỉ một số mẩu tin tính từ vị trí thứ n đến vị trí thứ m trong Table (theo một tiêu chuẩn hay sắp xếp nào đó). Để làm điều này, trong phát biểu SQL dạng SELECT bạn dùng chỉ định từ khoá LIMIT với số lượng mẩu tin cần lấy từ vị trí thứ n đến m.

Chẳng hạn, trong trường hợp bạn khai báo Select * from tblOrders limit 0,10. Kết quả sẽ trả về 10 mẩu tin đầu tiên trong bảng tblOrders.

Bạn cũng có thể sử dụng kết hợp LIMIT với các mệnh đề như WHERE, ORDER BY nhằm tạo ra kết quả như ý muốn.

Do yêu cầu khác nhau thông qua phát biểu SQL dạng SELECT có sử dụng LIMIT, nghĩa là kết quả trả về số lượng 10 mẩu tin đầu tiên với tất cả các cột trong bảng tblOrders

Ví dụ : Phát biểu SQL dạng SELECT với Limit N,M

```
Select *
From tblOrders
Limit 0,10
```

Kết quả trả về như sau:

OrderID	OrderDate	CustID	Amount
01	2001-09-05	10	2903576
02	2001-09-05	10	48168567
03	2001-09-05	10	5107032
04	2001-09-08	10	2.3555347
05	2001-09-08	16	1.81074847
06	2001-09-10	16	26000
07	2001-09-11	16	1867682
08	2001-09-12	16	3600000
09	2001-09-13	16	1.95713899
10	2001-09-14	16	9.61804228

Nếu muốn lọc ra 10 hợp đồng có số tiền nhiều nhất, bạn chỉ cần sử dụng sắp xếp theo cột TotalAmount hay Amount trong bảng tblOrders.

Ví dụ : Phát biểu SQL dạng SELECT với Limit N,M

```
Select OrderID,OrderDate,CustID,Amount
From tblOrders
Order by Amount Desc
Limit 0,10
```

Kết quả trả về như sau:

OrderID	OrderDate	CustID	Amount
06	2001-09-10	16	26000
26	2001-09-13	13	230000
25	2001-09-11	13	244904
23	2001-09-11	12	459162
31	2001-09-16	13	459525
27	2001-09-14	13	603033
28	2001-09-15	13	1452000
30	2001-09-15	13	1531200
07	2001-09-11	16	1867682
01	2001-09-05	10	2903576

Nếu muốn lọc ra 10 sản phẩm có số lượng bán nhiều nhất, bạn chỉ cần sử dụng sắp xếp theo

cột số lượng Qtty.

Ví dụ : Phát biểu SQL dạng Select với Limit N,M

```
Select ItemID,Qtty,Price,Amount
from tblOrderDetails
Where Amount>10
order by Qtty
Limit 0,10
```

Kết quả trả về như sau:

ItemID	Qtty	Price	Amount
1	900	12000	12960000
2	1000	12000	14400000
3	5000	12000	72000000
3	6000	12000	86400000
4	8000	12000	15200000
4	8000	12000	15200000
4	8000	10000	15200000
5	9000	12000	29600000
5	9000	12000	129600000
5	9000	12000	129600000

2.4.3.1.8 Phát biểu SELECT với DISTINCT

Nếu có một hay nhiều bảng kết nối với nhau, sẽ xảy ra trùng lặp nhiều mẩu tin. Nhưng trong trường hợp này bạn chỉ cần lấy ra một mẩu tin trong tập mẩu tin trùng lặp, bạn sử dụng phát biểu SQL dạng SELECT với chỉ định DISTINCT.

Ví dụ : Phát biểu SQL dạng SELECT

```
Select ItemID,Qtty,Price,Amount
from tblOrderDetails
order by Qtty
```

Kết quả trả về như sau:

ItemID	Qtty	Price	Amount
1	900	12000	12960000
2	1000	12000	14400000
3	5000	12000	72000000
3	6000	12000	86400000
4	8000	12000	115200000
4	8000	12000	115200000
4	8000	10000	115200000
5	9000	12000	129600000

5 9000 12000 129600000
5 9000 12000 129600000
...
...

Ví dụ : Phát biểu SQL dạng SELECT với DISTINCT

```
Select Distinct ItemID,Qtty,Price,Amount
From tblOrderDetails
Order by Qtty
```

Kết quả loại bỏ những mẩu tin trùng lắp như sau:

ItemID Qtty Price Amount
---
1 900 12000 12960000
2 1000 12000 14400000
3 6000 12000 86400000
4 8000 12000 115200000
5 9000 12000 129600000

2.4.3.1.9 Các hàm thông dụng trong MySQL

2.4.3.1.9.1 Các hàm trong phát biểu GROUB BY

- **Hàm AVG**: Hàm trả về giá trị bình quân của cột hay trường trong câu truy vấn, ví dụ như phát biểu sau:

```
Select AVG(Amount)
From tblOrders
```

- **Hàm MIN**: Hàm trả về giá trị nhỏ nhất của cột hay trường trong câu truy vấn, ví dụ như phát biểu sau:

```
Select Min(Amount)
From tblOrders
```

- **Hàm MAX**: Hàm trả về giá trị lớn nhất của cột hay trường trong câu truy vấn, ví dụ như các phát biểu sau:

```
Select Max(Amount)
From tblOrders
```

- **Hàm Count**: Hàm trả về số lượng mẩu tin trong câu truy vấn trên bảng, ví dụ như các phát biểu sau:

```
Select count(*)
From tblOrders
Select count(CustID)
From tblOrders
Select count(*)
```

```
From tblOrderDetails
```

- **Hàm Sum**: Hàm trả về tổng các giá trị của trường, cột trong câu truy vấn, ví dụ như các phát biểu sau:

```
Select sum(Amount)
From tblOrders
```

Chẳng hạn, bạn có thể tham khảo diễn giải toàn bộ các hàm dùng trong mệnh đề GROUP BY.

Ví dụ : SQL dạng SELECT với Group By và các hàm

```
Select CustID,
Count (CustID),Sum(Amount),
Max(Amount),
Min(Amount),
Avg(Amount)
From tblOrders
Group by CustID
Order by CustID
```

Kết quả trả về như sau:

CustID

16	7	2956562368	1.95713899	26000	422366052
12	9	3843022604	39879489	459162	427002511
13	8	145913378	1.36727628	230000	18239172.25
10	4	72382804	48168567	2903576	18095701

2.4.3.1.9.2 Các hàm xử lý chuỗi

- **Hàm ASCII**: Hàm trả về giá trị mã ASCII của ký tự bên trái của chuỗi, ví dụ như khai báo:

```
Select  ASCII('TOI')
```

Kết quả trả về như sau:

84

- **Hàm Char**: Hàm này chuyển đổi kiểu mã ASCII từ số nguyên sang dạng chuỗi:

```
Select  char(35)
```

Kết quả trả về như sau:

#

- **Hàm UPPER**: Hàm này chuyển đổi chuỗi sang kiểu chữ hoạ:

```
Select  UPPER('Manhhm')
```

Kết quả trả về như sau:

MANHHM

- **Hàm LOWER**: Hàm này chuyển đổi chuỗi sang kiểu chữ thường:

```
Select  LOWER('Manhhm')
```

Kết quả trả về như sau:

mannhm

- **Hàm Len**: Hàm này trả về chiều dài của chuỗi:

```
Select len('I Love You')
```

Kết quả trả về như sau:

10

- **Thủ tục LTRIM**: Thủ tục loại bỏ khoảng trắng bên trái của chuỗi:

```
Select ltrim(' Manhhm')
```

Kết quả trả về như sau:

'mannhhm
'

- **Thủ tục RTRIM**: Thủ tục loại bỏ khoảng trắng bên phải của chuỗi:

```
Select ltrim('Manhhm  ')
```

Kết quả trả về như sau:

'manhhm'

- **Hàm Left**: Hàm trả về chuỗi bên trái tính từ đầu cho đến vị trí thứ n:

```
Select  left('Manhhm',3)
```

Kết quả trả về như sau:

'Man'

- **Hàm Right**: Hàm trả về chuỗi bên phải tính từ cuối cho đến vị trí thứ n:

```
Select  Right('MAnhhm',4)
```

Kết quả trả về như sau:

'nhhm'

2.4.3.1.9.3 Các hàm về xử lý thời gian

- **Hàm CurDate()**: Hàm trả về ngày, tháng và năm hiện hành của hệ thống:

```
Select curdate() as 'Today is'
```

Kết quả trả về như sau

Today is

2001-11-21

- **Hàm CurTime()**: Hàm trả về giờ, phút và giây hiện hành của hệ thống:

```
Select  curtime()
as 'Time is'
```

Kết quả trả về như sau

Time is

09:12:05

- **Hàm Period_Diff**: Hàm trả về số ngày trong khoảng thời gian giữa 2 ngày:

```
Select
Period_diff (OrderDate,  getdate())
as 'So ngay giua ngay thu tien đen hom nay:'
from tblOrders
```

Kết quả trả về như sau:

So ngay giua ngay thu tien đen hom nay:

74
72

- **Hàm dayofmonth**: Hàm dayofmonth trả về ngày thứ mấy trong tháng:

```
Select dayofmonth(curdate())
as 'hom nay ngay
```

Kết quả trả về như sau:

21

2.4.3.1.9.4 Các hàm về toán học

- **Hàm sqrt**: Hàm trả về là căn bật hai của một biểu thức:

Select sqrt (4)

Kết quả trả về là :

2

- **Hàm Round**: Hàm trả về là số làm tròn của một biểu thức:

Select round (748.58,-1)

Kết quả trả về là :
7500

2.4.3.1.10 Phát biểu SQL dạng JOIN

Phát biểu *SQL* dạng *Select* để kết nối dữ liệu giữa các bảng có quan hệ với nhau

- *Khái niệm JOIN*
- *Phát biểu INNER JOIN*
- *Phát biểu LEFTJOIN*
- *Phát biểu RIGHT JOIN*

2.4.3.1.10.1 Khái niệm về quan hệ

Để phát triển ứng dụng *Web* bằng bất kỳ loại cơ sở dữ liệu nào, giai đoạn phân tích thiết kế hệ thống cực kỳ quan trọng. Nếu kết quả phân tích không tối ưu thì ứng dụng đó không thể đạt được giá trị kỹ thuật cũng như giá trị thương mại. Thiết kế cơ sở dữ liệu không tối ưu, chúng có thể dẫn đến việc chương trình chạy chậm và không bền vững.

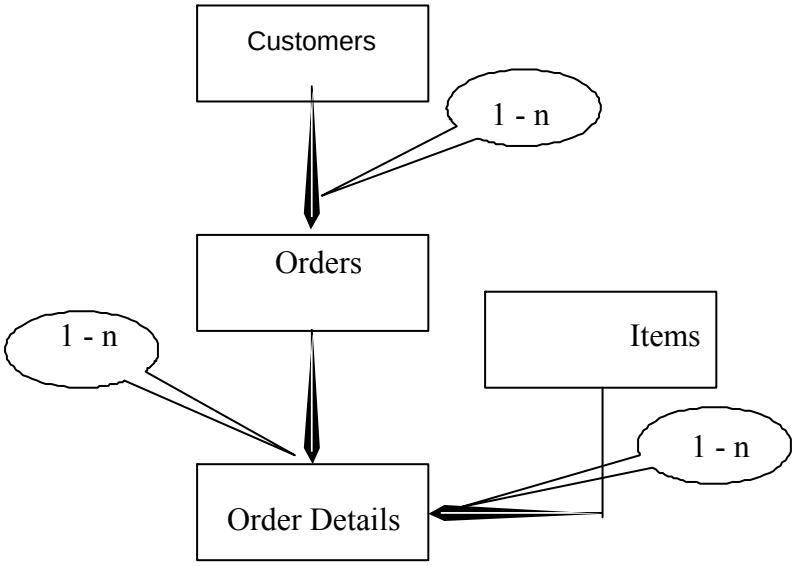
Một khi ứng dụng chạy chậm đi do cơ sở dữ liệu không tối ưu thì rất có thể bạn phải thiết kế và xây dựng lại từ đầu toàn bộ cấu trúc của chương trình và cơ sở dữ liệu.

Xuất phát từ lý do này, khi xây dựng một ứng dụng thông tin quản lý, chúng ta cần phải qua những bước phân tích thiết kế hệ thống kỹ lưỡng để có được mô hình quan hệ và *ERD* trước khi đến các mô hình chức năng chi tiết.

Tuy nhiên, trong lý thuyết một số kiến thức cơ bản bắt buộc bạn phải thực hiện theo mô hình hệ thống ứng với những quan hệ toàn vẹn, nhưng trong thực tế, do tính đặc thù của ứng dụng, thường bạn phải thiết kế lại mô hình theo nhu cầu cân đối giữa độ phức tạp và tính tối ưu.

Trong ứng dụng bán hàng qua mạng *Test* đã trình bày trong chương 3, khi quan tâm đến một hợp đồng trên mạng, ngoài những thông tin liên lạc về khách hàng, bạn cần phải lưu trữ dữ liệu khác như chiết hàng mua, phương thức trả tiền, phương thức giao hàng,... Vấn đề được thảo luận ở đây, mỗi hợp đồng có nhi6u mặt hàng chi tiết.

Trong trường hợp này, chúng ta có 6 thực thể liên quan như sau, thực thể danh mục *Customers* (thông tin liên lạc của khách hàng), *Orders* (hợp đồng mua hàng), *OrderDetails* (chi tiết hàng mua), *Items* (danh mục sản phẩm)



Sơ đồ : Mô hình quan hệ

Giả sử rằng khi nhập số liệu vào cơ sở dữ liệu, ứng với hợp đồng có mã 101, của khách hàng có tên Nguyễn Văn A, ... có hai sản phẩm chi tiết: 11 (Nước ngọt) và 32 (xà phòng Lux).

Trong trường hợp này bạn đang có một mẩu tin hợp đồng trong bảng tblCustomers, một mẩu tin hợp đồng trong bảng tblOrders và hai mẩu tin trong bảng tblOrderDetails.

Nếu muốn biết thông tin hợp đồng của khách hàng A, rõ ràng bạn cần dùng phát biểu SELECT với mệnh đề kết hợp từ 3 bảng trên. Kết quả trả về 2 mẩu tin là sự kết hợp thông tin từ hai bảng tblCustomers, tblOrders và tblOrderDetails.

Khi thực thi phát biểu SQL dạng SELECT ứng với cơ sở dữ liệu như trên bạn phải duyệt qua hai mẩu tin.

Tất nhiên, khi viết ứng dụng thì điều này chấp nhận được, và có thể coi là tối ưu. Giả sử rằng, ứng dụng này được phát triển trên WEB cần lưu tâm đến vấn đề tối ưu tốc độ truy vấn thì sao?

Người thiết kế cơ sở dữ liệu trong trường hợp này phải thay đổi lại cấu trúc để tăng tốc độ truy cập qua mạng khi xử lý trên cơ sở dữ liệu của người dùng.

2.4.3.1.10.2 Khái niệm về mệnh đề JOIN

Trong hầu hết phát biểu SELECT, phần lớn kết quả mà bạn mong muốn lấy về đều có liên quan đến một hoặc nhiều bảng khác nhau. Trong trường hợp như vậy, khi truy vấn dữ liệu bạn cần sử dụng mệnh đề JOIN để kết hợp dữ liệu trên hai hay nhiều bảng lại với nhau.

Khi sử dụng JOIN, bạn cần quan tâm đến trường (cột) nào trong bảng thứ nhất có quan hệ với trường (cột) nào trong bảng thứ hai. Nếu mô hình quan hệ của bạn không tối ưu hay không đúng, quản trình sử dụng JOIN sẽ cho kết quả trả về không như ý muốn.

Trở lại ứng dụng bán hàng qua mạng trong giáo trình này, khi xuất một hợp đồng bán hàng cho khách hàng, theo thiết kế trong cơ sở dữ liệu chúng ta có rất nhiều bảng liên quan đến nhau.

Chẳng hạn, nếu quan tâm bán hàng thì bán cho ai. Suy ra, liên quan đến thông tin khách hàng, bán sản phẩm gì cho họ thì liên quan đến mã sản phẩm, nếu khách hàng trả tiền thì liên quan đến phiếu thu, nếu khách hàng có công nợ thì liên quan đến nợ kỳ trước...

Trong phân này, chúng tôi tiếp tục thiết kế một số bảng dữ liệu cùng với kiểu dữ liệu tương ứng và quan hệ giữa các bảng được mô tả như sau:

tblCustomers (danh sách khách hàng)

```
[CustID] int auto_increment Primary key,
[CustName] [varchar] (50) NULL ,
[Address] [varchar] (100) NULL,
[Tel] [varchar] (20) NULL,
[FaxNo] [varchar] (20) NULL,
[Email] [varchar] (50) NULL,
[Contact] [varchar] (50) NULL
[Country] [varchar] (3) NULL,
[Province] [varchar] (3) NULL
```

tblOrders (Hợp đồng bán hàng)

```
[OrderID] [int] Not null
auto_increment Primary Key,
[OrderDate] [date] NULL ,
[CustID] int ,
[Description] [varchar] (200) NULL ,
[ShipCost] [float] NULL ,
[TranID] [tinyint] NULL ,
[PaymentID] [tinyint] NULL ,
[Amount] [float] NULL ,
[TotalAmount] [float] NULL
```

tblOrderDetails (Hợp đồng bán hàng chi tiết)

```
[SubID] [int] auto_increment  NOT NULL ,
[OrderID] int ,
[ItemID] int,
[No] int,
[Qtty] [int] NULL ,
[Price] int NULL ,
[Discount] [Float] NULL ,
[Amount] [Float]NULL
```

tblItems (Danh sách sản phẩm)

```
[ItemID] int auto_increment Primary key,
[ItemName] [varchar] (200) NULL ,
[Unit] [nvarchar] (20) NULL ,
[Cost] [Float] NULL ,
[Active] [tinyint] NOT NULL ,
[Category] int
```

2.4.3.1.10.3 Mệnh đề INNER JOIN

Phát biểu SQL dạng SELECT có sử dụng mệnh đề INNER JOIN thường dùng để kết hợp hai hay nhiều bảng dữ liệu lại với nhau, cú pháp của SELECT có sử dụng mệnh đề INNER JOIN:

```
SELECT [SELECT LIST]
FROM <FIRST_TABLENAME>
INNER JOIN <SECOND_TABLENAME>
ON <JOIN CONDITION>
WHERE <CRITERIANS>
ORDER BY <COLUMN LIST>
[ASC / DESC]
```

Nếu bạn cần lấy ra một số cột trong các bảng có kết nối lại với nhau bằng mệnh đề INNER JOIN thì cú pháp này viết lại như sau:

```
SELECT [FIELD1,FIELD2,  ...]
FROM <FIRST_TABLENAME>
INNER JOIN <SECOND_TABLENAME>
ON <JOIN CONDITION>
WHERE <CRITERIANS>
ORDER BY <COLUMN LIST>
[ASC / DESC]
```

Ví dụ : INNER JOIN với một số cột chỉ định
/* in ra danh sách khách hàng mua hàng trong tháng 10 */

```
Select CustName,OrderID,
OrderDate,Amount,
TotalAmount
from tblCustomers
inner join tblOrders
on tblCustomers.CustID = tblOrders.CustID
where month (OrderDate) = 10
order by  CustName
```

Kết quả trả về như sau:

CustName	OrderID	OrderDate ..	TotalAmount
CENTURY Hotel	13	2001-10-17	388800000
CENTURY Hotel	14	2001-10-18	518400000
CENTURY Hotel	16	2001-10-17	388800000

CENTURY Hotel	17	2001-10-18	14400000
CENTURY Hotel	18	2001-10-18	12960000
CENTURY Hotel	110	2001-10-18	216000000
Plaza Hotel	12	2001-10-17	403200000
Plaza Hotel	19	2001-10-17	86400000
Plaza Hotel	11	2001-10-17	576000000
Plaza Hotel	15	2001-10-17	288000000

Nếu bạn cần lấy ra tất cả các cột trong các bảng có kết nối lại với nhau bằng mệnh đề INNER JOIN, cú pháp trên có thể viết lại như sau:

```
SELECT first_tablename.*,
second_tablename.*
[,next table name]
FROM <first_tablename>
INNER JOIN <second_tablename>
ON <join conditions>
[INNER JOIN <next_tablename>
ON <join conditions>]
WHERE <conditions>
ORDER BY <column list>
[ASC / DESC]
```

Ví dụ : INNER JOIN với tất các trường liên quan
/* in ra danh sách khách hàng mua hàng trong tháng 10 */

```
Select CustID,CustName,OrderID,
OrderDate,TotalAmount
from tblCustomers
inner join tblOrders
On TblCustomers.CustID=tblOrders.CustID
where month (OrderDate) = 10
order by  CustName DESC
```

Kết quả trả về như sau:

CustID	CustName	OrderID	TotalAmount
13	Plaza Hotel	11	576000000
13	Plaza Hotel	15	288000000
12	Plaza Hotel	12	403200000
12	Plaza Hotel	19	86400000
16	CENTURY Hotel	13	388800000
16	CENTURY Hotel	14	518400000
16	CENTURY Hotel	16	388800000
16	CENTURY Hotel	17	14400000
16	CENTURY Hotel	18	12960000
16	CENTURY Hotel	110	216000000

Nếu trong những bảng cần kết nối có tên trường (cột) giống nhau thì khi thực thi phát biểu SQL dạng SELECT phải chỉ rõ cột thuộc bảng nào. Trong trường hợp cả hai cùng lấy dữ liệu ra thì bạn cần chuyển ánh xạ tên khác cho cột thông qua mệnh đề AS, ví dụ như:

```
SELECT first_tablename.CustID as CUSTID,
second_tablename.CustID as CUSTID
FROM <first_tablename>
INNER JOIN <second_tablename>
ON <join condition>
WHERE <criterians>
ORDER BY <column list>
[ASC / DESC]
```

Nếu trong những bảng cần kết nối đó có tên trường (cột) giống nhau và không được chỉ rõ như trường hợp trên khi khai báo trong cơ sở dữ liệu SQL Server, khi thực thi phát biểu SQL dạng SELECT bạn sẽ bị lỗi, chẳng hạn như:

```
SELECT first_tablename.*, second_tablename.*
FROM <first_tablename>
INNER JOIN <second_tablename>
ON <join condition>
WHERE <criterians>
ORDER BY <column list>
[ASC / DESC]
```

Server: Msg 209, Lecel 16, State Line 1
Ambiguous column name 'CustID'

Tuy nhiên, với phát biểu trên bạn có thể thực thi trong cơ sở dữ liệu MySQL. Ngoài ra, phát biểu SQL dạng SELECT sử dụng INNER JOIN bạn có thể ánh xạ (alias) tên của bảng thành tên ngắn gọn để dễ tham chiếu về sau.

Thực ra phát biểu ALIAS có ý nghĩa giống như AS với tên cột trong bảng thành tên cột khác trong phát biểu SELECT.

```
Select p.*,s.*
from tablename1
inner join tablename2
On tablename1.field1 = tablename2.field2
```

Ví dụ : INNER JOIN với ánh xạ tên bảng
/* in ra danh sách khách hàng mua hàng trong tháng 10 */

```
Select c.CustName,
s.OrderID,s.OrderDate,
s.TotalAmount
from tblCustomer c
inner join tblOrders s
On c.CustID=s.CustID
where month (s.OrderDate) = 10
order by  c.CustName DESC
```

Kết quả trả về như sau:

CustName	OrderID	OrderDate	TotalAmount
CENTURY Hotel	13	2001-10-17	388800000

CENTURY Hotel	14	2001-10-18	518400000
CENTURY Hotel	16	2001-10-17	388800000
CENTURY Hotel	17	2001-10-18	14400000
CENTURY Hotel	18	2001-10-18	12960000
CENTURY Hotel	11	2001-10-18	216000000
Plaza Hotel	12	2001-10-17	403200000
Plaza Hotel	19	2001-10-17	86400000
Plaza Hotel	11	2001-10-17	576000000
Plaza Hotel	15	2001-10-17	288000000

Tất nhiên, bạn cũng có thể viết phát biểu trên ứng với từng cột muốn lấy ra bằng cách khai báo tên cột.

2.4.3.1.10.4 Mệnh đề Left Join

Trường hợp bạn mong muốn kết quả lấy ra trong hai bảng kết hợp nhau theo điều kiện: Những mẩu tin bảng bên trái tồn tại ứng với những mẩu tin ở bảng bên phải không tồn tại bạn hãy dùng mệnh đề LEFT JOIN trong phát biểu SQL dạng SELECT, cú pháp có dạng:

```
select <Column list>
from lefttablename
LEFT  JOIN righttablename
on lefttabkename.field1=righttablename.field2
Where <conditions>
Order by <column name>
ASC/DESC
```

Chẳng hạn, bạn chọn ra tất cả các sản phẩm (với các cột) có hay không có doanh số bán trong tháng hiện tại. Một số sản phẩm không bán trong tháng sẽ có cột Amount có cột Amount giá trị NULL.

Ví dụ : SELECT dùng LEFT JOIN
/* in ra danh sách sản phẩm bán trong tháng 10 */

```
select ItemID,ItemName,Amount
from tblItems
left  join tblOrderDetails
on tblItems.ItemID=tblOrderDetails.ItemID
order by Amount
```

Kết quả trả về như sau:

ItemID	ItemName	Amount
12	ASW-60VP	NULL
13	ASW-60VT	NULL
14	ASW-660T 120V TW 29340	NULL
14	ASW-685V 120V TW 29440	NULL
15	ASW60VP 220V 34571	NULL
16	ASW-45Z1T1	2960000
17	ASW-45Y1T 127V	14400000

18	ASW-45Y1T 220V	72000000
19	ASW-45Y1T 220V	86400000
20	ASW-45Z1T	15200000

2.4.3.1.10.5 Mệnh đề Right Join

Ngược lại với phát biểu SQL dạng SELECT sử dụng mệnh đề LEFT JOIN là phát biểu SQL dạng SELECT sử dụng mệnh đề RIGHT JOIN sẽ xuất dữ liệu của bảng bên phải cho dù dữ liệu của bảng bên trái không tồn tại, cú pháp có dạng:

```
Select <Column list>
From lefttablename
RIGHT  JOIN righttablename
On lefttabkename.field1=righttablename.field2
Where <conditions>
Order by <column name>
ASC/DESC
```

Trong ví dụ sau, bạn có thể chọn ra tất cả các sản phẩm có hay không có doanh số bán trong tháng hiện tại. Các sản phẩm không tồn tại doanh số bán sẽ không hiện ra.

Ví dụ : SELECT dùng RIGHT JOIN
/* in ra danh sách sản phẩm bán trong tháng ngày 17 */
/* trong phát biểu SELECT này có sử dụng mệnh đề WHERE sử dụng phát biểu SELECT khác, kết quả của SELECT trong mệnh đề WHERE trả về một mảng OrderID */

```
Select ItemName,Qtty,
Price,Amount
From tblItems
Right  join tblOrderDetails
On tblItems.ItemID=tblOrderDetails.ItemID
Where OrderID in (12,14,23,15)
Order by ItemID
```

Kết quả trả về như sau:

ItemName	Qtty	Price	Amount
ASW-45Y1T 127V SDIA29350	11000	12000	58400000
ASW-45Y1T 127V SDIA29350	10000	12000	44000000
ASW-45Y1T 127V SDIA 29350	10000	12000	14400000
ASW-45Y1T 127V SDIA 29350	10000	12000	44000000
ASW-45Y1T 127V SDIA 29350	11000	12000	58400000
ASW-45Y1T 127V SDIA 29350	10000	12000	44000000
ASW-45Y1T 127V SDIA 29350	11000	12000	58400000
ASW-45Y1T 220V ARG 29391	6000	12000	86400000
ASW-45Z1T	9000	12000	29600000
ASW-45Z1T	9000	12000	29600000

2.4.3.1.10.6 Phép toán hợp (union)

Union không giống như những mệnh đề JOIN đã giới thiệu trên đây. Union là phép toán dùng để nối hai hay nhiều câu truy vấn dạng Select lại với nhau.

Đối với JOIN, bạn có thể kết nối dữ liệu được thực hiện theo chiều ngang. Đối với Union bạn kết nối dữ liệu được thực hiện theo chiều dọc.

Để chọn ra những khách hàng thường xuyên trong tblCustomers, kết quả trả về là danh sách các khách hàng thường xuyên.

Ví dụ : Khách hàng thường xuyên trong tblCustomers

```
Select CustID,CustName
from tblCustomers
```

Kết quả trả về như sau:

```
CustID        CustName
------------------------------
13            New World Hotel
12            Kinh Do Hotel
16            CENTURY Hotel
10            PLAZA Hotel
```

Để chọn ra những khách hàng vãng lai trong tblTempCustomers, kết quả trả về là danh sách các khách hàng vãng lai.

Ví dụ : Khách hàng vãng lai trong tblTempCustomers

```
Select CustID,CustName
from tblTempCustomers
```

Kết quả trả về như sau:

```
CustID        CustName
-----------------------------------------

23            Cong ty nuoc giai khat '12'COLA
24            Cong ty nuoc giai khat PEPSI
25            Cong ty nuoc giai khat REDBULK
26            Cong ty nuoc giai khat TRIBICO
```

Nếu dùng phép toán UNION để kết nối hai bảng trên, kết quả trả về là danh sách cả hai loại khách hàng trong cùng một recordset.

Ví dụ : SELECT sử dụng phép hợp UNION

```
Select CustID,CustName
From tblCustomers
```

```
UNION
Select CustID,CustName
From tblTempCustomers
```

Kết quả trả về như sau:

```
CustID      CustName
--------------------------------------------------
23          Cong ty nuoc giai khat '12'COLA
24          Cong ty nuoc giai khat PEPSI
25          Cong ty nuoc giai khat REDBULK
26          Cong ty nuoc giai khat TRIBICO
12          Kinh Do Hotel
10          PLAZA Hotel
16          CENTURY Hotel
13          New World Hotel
```

Ghi chú: Khi sử dụng phép toán Union trong phát biểu SQL dạng Select, bạn cần lưu ý các quy định sau:

- Tất cả những truy vấn trong UNION phải cùng số cột hay trường. Nếu truy vấn thứ nhất có hai cột thì truy vấn thứ hai được sử dụng UNION cũng phải có hai cột tương tự.
- Khi sử dụng UNION, những cột nào có tên cột hay bí danh (alias) mới thì kết quả trả về sẽ có tựa đề (header) của từng cột và tên là tên cột của truy vấn thứ nhất.
- Kiểu dữ liệu trong các cột của truy vấn 2 tương thích với kiểu dữ liệu các cột tương ứng trong truy vấn thứ nhất.
- Trong UNION bạn có thể kết hợp nhiều câu truy vấn lại với nhau.
- Kết quả hiện ra theo thứ tự của truy vấn từ dưới lên trên.

2.4.3.2 Phát biểu SQL dạng INSERT

Khi cần thêm mẩu tin vào bảng trong cơ sở dữ liệu MySQL, bạn có nhiều cách để thực hiện công việc này. Trong Visual Basic 6.0, VB.NET, C Sharp hay Java có những phương thức để thêm mẩu tin vào bảng trong cơ sở dữ liệu. Tuy nhiên, để sử dụng các phát biểu SQL mang tính chuyên nghiệp trong MySQL, bạn cần sử dụng phát biểu INSERT.

Bạn có thể sử dụng phát biểu Insert ngay trên ứng dụng kết nối với MySQL. Trong trường hợp bạn sử dụng cơ sở dữ liệu SQLServer hay Oracle, bạn có thể tạo ra một Stored Procedure với mục đích INSERT dữ liệu vào bảng chỉ định trước.

Khi thêm dữ liệu, cần chú ý kiểu dữ liệu giống hoặc tương ứng kiểu dữ liệu đã khai báo của cột đó, nếu không phù hợp thì lỗi sẽ phát sinh.

Ngoài ra bạn cần quan tâm đến quyền của User đang truy cập cơ sở dữ liệu. User phải được cấp quyền Insert dữ liệu vào từng bảng cụ thể (quyền này do nhà quản trị cơ sở dữ liệu phân quyền cho User đó).

Trong phát biểu INSERT INTO chúng tôi thực hiện trên bảng tblOrderDetails và bảng tblOrderDetailsHist, hai bảng này có cấu trúc như sau:

```
  /* Bảng tblOrderDetails*/
CREATE TABLE tblorderdetails (
ItemID int(3) unsigned DEFAULT '0' ,
OrderID int(3) unsigned DEFAULT '0' ,
No tinyint(3) unsigned DEFAULT '0' ,
Qtty int(3) unsigned DEFAULT '0' ,
Price int(3) unsigned DEFAULT '0' ,
Discount int(3) unsigned DEFAULT '0' ,
Amount bigint(3) unsigned DEFAULT '0'
);
```

```
  /* Bảng tblOrderDetailsHist, dùng để chứa các thông tin
hợp đồng chi tiết khi hợp đồng của khách hàng này kết thúc,
chương trình tự động xoá trong tblOrderDetails và lưu trữ lại
trong bảng tblOrderDetailsHist.*/

CREATE TABLE tblorderdetailshist (
ItemID int(3) unsigned DEFAULT '0' ,
OrderID int(3) unsigned DEFAULT '0' ,
No tinyint(3) unsigned DEFAULT '0' ,
Qtty int(3) unsigned DEFAULT '0' ,
Price int(3) unsigned DEFAULT '0' ,
Discount int(3) unsigned DEFAULT '0' ,
Amount bigint(3) unsigned DEFAULT '0'
);
```

Khi Insert dữ liệu vào bảng, có 3 trường hợp xảy ra: insert dữ liệu vào bảng từ các giá trị cụ thể, insert vào bảng lấy giá trị từ một hay nhiều bảng khác, và cuối cùng là kết hợp cả hai trường hợp trên.

2.4.3.2.1 Insert vào bảng lấy giá trị cụ thể

```
INSERT INTO <Tablename>[<columnname list>]
Values (data_value)
```

Ví dụ : INSERT dữ liệu vào bảng từ giá trị cụ thể

```
/*  Thêm mẩu tin với một số cột */
INSERT  INTO
TBLCUSTOMERS
(CustName,Username,Password,
Address,Tel,FaxNo,Email,Contact,
CountryCode,ProvinceCode)
Values ('Khach San CENTURY', 'century',
'1111','5 Le Loi','8676767','8767676',
'century@yahoo.com','Hoang Anh',
'VNA','HCM')
```

```
/*  Thêm mẩu tin với một số cột */
INSERT  INTO
TBLORDERS(OrderID,OrderDate,
```

```
CustID,Description,Amount)
Values ('11',curdate(),'1',
'Dat hang qua mang', 20000)
```

2.4.3.2.2 Insert vào bảng lấy giá trị từ bảng khác

```
INSERT INTO <Tablename1>[<columnname list>]
Select [columnname list]
From <Tablename2>
Where <Conditions>
```

Ví dụ : INSERT vào bảng từ giá trị của bảng khác

```
/*  Thêm mẫu tin với các cột cụ thể */
/* Chuyển tất cả những hợp đồng chi tiết từ bảng tblOrderDetails vào bảng
tblOrderDetailsHist */

INSERT  INTO
TBLORDERDETAILSHIST(
ItemID,
OrderID,
No,
Qtty,
Price,
Discount,
Amount)
SELECT
ItemID,
OrderID,
No,
Qtty,
Price,
Discount,
Amount
From tblOrderDetails
ORDER BY OrderID ASC
```

```
/* Có thể viết lại thêm mẫu tin với tất cả các cột như sau  Chuyển tất cả
những  hợp  đồng  chi  tiết  từ  bảng  tblOrderDetails  vào  bảng
tblOrderDetailsHist  với  điều  kiện  số  cột  tương  ứng  trong  bảng
tblOrderDetails bằng với số cột trong bảng tblOrderDetailsHist, bạn có
thể viết lại như sau */

INSERT  INTO TBLORDERDETAILSHIST
SELECT * from
tblOrderDetails
ORDER BY OrderID ASC
```

2.4.3.2.3 Insert vào bảng lấy giá trị cụ thể, bảng khác

```
INSERT INTO <Tablename1>[<columnname list>]
Select [columnname list], valueslist
From <Tablename2>
Where <conditions>
ORDER BY <column name> ASC/DESC
```

Ví dụ : INSERT vào bảng từ giá trị cụ thể, bảng khác

```
/*  Thêm mẫu tin với các cột cụ thể */
/* Chuyển tất cả những hợp đồng chi tiết từ bảng tblOrderDetails vào bảng
tblOrderDetailsHist.  Giả  sử  rằng,  ngoài  những  cột  giống  như
tblOrderDetails, bảng tblOrderDetailsHist còn có thêm cột Tranferdate.*/

INSERT  INTO
TBLORDERSHIST(
OrderID,
OrderDate,
ReceiveFolio,
CustID,
Descriion,
Amount,
Historydate)
SELECT
OrderID,
OrderDate,
ReceiveFolio,
CustID,
Descriion,
Amount,
getdate() as Historydate
From tblOrders
where Month(OrderDate)=12
Order by OrderDate,CustID
```

```
/* Có thể viết lại thêm mẫu tin với tất cả các cột như sau */
/* Chuyển tất cả những phiếu thu trong tháng 12 từ bảng tblOrders vào bảng
tblOrdersHist với điều kiện số cột tương ứng trong bảng tblOrders bằng với
số cột trong bảng tblOrdersHist, bạn có thể viết lại như sau */
INSERT  INTO
TBLORDERDETAILSHIST(
ItemID,
OrderID,
No,
Qtty,
Price,
Discount,
Amount,TranferDate)
SELECT
ItemID,
No,
Qtty,
Price,
Discount,
Amount,CurDate()
From tblOrderDetails
ORDER BY OrderID ASC
```

2.4.3.3 Phát biểu SQL dạng UPDATE

Phát biểu SQL dạng UPDATE dùng cập nhật lại dữ liệu đã tồn tại trong bảng. Khi UPDATE

dùng cập nhật dữ liệu cho một mẩu tin chỉ định nào đó thường UPDATE sử dụng chung với mệnh đề WHERE.

Nếu cần cập nhật tất cả các mẩu tin trong bảng bạn có thể bỏ mệnh đề WHERE. Phát biểu này có cấu trúc như sau:

```
/* nếu cập nhất giá trị cụ thể */
Update <table name>
Set <column>=<value>,[<column>=<value>]
[where <restrictive conditions>]
```

```
/* nếu cập nhất giá trị là kết quả trả về từ phát biểu
select trên một hay nhiều bảng khác */

Update <table name>
Set <column>=<select .. from tablename where ...>
[where <restrictive conditions>]
```

UPDATE có thể ảnh hưởng đến nhiều bảng, nhưng cập nhất giá trị chỉ có hiệu lực trên bảng đó, bạn có thể tham khảo phần này trong chương kế tiếp JOIN TABLE.

Cập nhật giá trị cụ thể vào một hay nhiều cột

Ví dụ : UPDATE trên các cột dữ liệu từ giá trị cụ thể

```
/* cập nhật cột với giá trị cụ thể */

Update tblCustomers
Set CustName='Cong ty TNHH Coca cola Vietnam'
Where CustID='12'
```

```
/* cập nhật một cột với giá trị cột khác trong bảng
tblOrderDetails*/

Update tblOrders
Set Amount= Amount*.01,
TotalAmount=Amount*0.1
Where Month(OrderDate)=12
```

```
/* cập nhật một cột với giá trị từ bảng khác*/
/* cập nhật cột Price với giá trị từ cột Cost của bảng tblItems, khai báo
sau chỉ đúng trong MySQL 4.1 trở về sau*/


Update tblOrderDetails
Set Price=
(select distinct Cost]
from  tblItems
where ItemID=tblOrderDetails.ItemID)
Where Price<1000
```

```
/* cập nhật một cột với giá trị cụ thể với điều kiện từ bảng khác, khai báo
sau chỉ đúng trong MySQL 4.1 trở về sau */

Update tblOrderDetails
```

```
Set Price= Price*10,
Amount= Qtty*(Price+1)
Where ItemID in
(select distinct ItemID
from tblOrderDetails
where Price>1000)
```

2.4.3.4 Phát biểu SQL dạng DELETE

Với phát biểu SQL dạng DELETE thì đơn giản hơn. Khi thực hiện lệnh xoá mẩu tin trong bảng chúng ta chỉ cần quan tâm đến tên bảng, và mệnh đề WHERE để xoá với những mẩu tin đã chọn lọc nếu có. Cú pháp của Delete:

```
Delete from <table name>
Where <condition>
```

Với mệnh đềWHERE giống như bất kỳ mệnh đề WHERE nào trong phát biểu SELECT hay UPDATE và INSERT của bất kỳ ứng dụng cơ sở dữ liệu nào có sử dụng SQL.

Conditions có thể là phép toán giữa các cột và giá trị, nhưng cũng có thể giá trị là kết quả trả về từ một phát biểu SELECT khác.

Ghi chú: Không có khái niệm xóa giá trị trong một cột, vì xóa giá trị một cột đồng nghĩa với cập nhật cột đó bằng giá trị rỗng.

Ví dụ : Xóa mẩu tin với phát biểu SQL dạng DELETE

```
/* Xoá mẩu tin từ bảng với điều kiện  */

Delete from tblCustomers
Where CustName is null
```

Trong trường hợp có ràng buộc về quan hệ của dữ liệu, thì xóa mẩu tin phải tuân thủ theo quy tắc: Xoá mẩu tin con trước rồi mới xoá mẩu tin cha.

Chẳng hạn, trong trường hợp ta có 2 bảng: hợp đồng bán hàng (tblOrders) và hợp đồng bán hàng chi tiết (tblOrderDetails).

Để xoá một hợp đồng bạn cần xóa mẩu tin trong bảng tblOrders trước rồi mới đến các mẩu tin trong bảng tblOrderDetails.
Ví dụ 8-20: Xoá mẩu tin với Delete

```
/* Xoá mẩu tin từ bảng con  */
Delete from tblOrderDetails
where OrderID=123
```

```
/* Xoá mẩu tin từ bảng cha  */
Delete from tblOrders
where OrderID=123
```

Bạn có thể thực hiện một phát biểu SQL dạng DELETE với điều kiện trong mệnh đề WHERE

lấy giá trị trả về từ phát biểu SELECT từ bảng khác, khai báo như vậy chỉ có hiệu lực trong cơ sở dữ liệu MySQL phiên bản 4.1 trở về sau hay trong cơ sở dữ liệu SQL Server và Oracle.

2.5 Kết hợp PHP và MYSQL trong ứng dụng website

Ở các bài trước, chúng ta đã cùng nghiên cứu về các cú pháp sql và Mysql cơ bản bao gồm việc tảo bảng, tạo kết nối, thêm, sửa, xóa các dòng dữ liệu trong cơ sở dữ liệu... Và tiếp theo bài này, chúng ta sẽ cùng tìm hiểu về cách sử dụng mysql kết hợp với PHP.

Các bước xây dựng chương trình có kết nối tới CSDL My SQL. Thông thường, trong một ứng dụng có giao tiếp với CSDL, ta phải làm theo bốn trình tự sau:

Bước 1: Thiết lập kết nối tới CSDL.
Bước 2: Lựa chọn CSDL.
Bước 3: Tiến hành các truy vấn SQL, xử lý các kết quả trả về nếu có
Bước 4: Đóng kết nối tới CSDL.

Nếu như trong lập trình thông thường trên Windows sử dụng các chương trình điều khiển trung gian (ADO, ODBC...) để thực hiện kết nối và truy vấn, thì trong PHP, khi lập trình tương tác với CSDL, chúng ta thường sử dụng thông qua các hàm do PHP cung cấp sẵn. Để làm việc với mysql và PHP chúng ta cần nắm các hàm cơ bản.

Chú ý: Trước khi sử dụng PHP để lập trình với MySQL, hãy sử dụng chương trình quản lý phpMyAdmin để tạo trước một CSDL, vài bảng cũng như người dùng... để tiện thực hành

2.5.1 Các hàm cơ bản làm việc với cơ sở dữ liệu MySQL.

2.5.1.1 Các hàm kết nối đến MySQL Server

PHP cung cấp hai hàm để kết nối với cơ sở dữ liệu MySQL :

mysql_connect và **mysql_pconnect** .

+ mysql_connect () : hàm này sẽ tạo ra một liên kết tới máy chủ MySQL .

Cú pháp :

```
int mysql_connect (string [hostname [:port] [:/path_to_socket]], string
[username], string [password]);
```

Trong đó :

- hostname : Tên máy chủ cơ sở dữ liệu, nơi trang web sẽ chứa cơ sở dữ liệu. Giá trị ngầm định là “localhost’’

- :port : Địa chỉ cổng, nơi bộ máy cơ sở dữ liệu lắng nghe yêu cầu. Giá trị ngầm định là “:3306” .

- :/path_to_socket : Cũng giống như :port nhưng chỉ cho hệ điều hành UNIX. Giá trị ngầm định là ":/tmp/mysql.sock" .

- username : Tên của người sử dụng được phép kết nối vào bộ máy cơ sở dữ liệu.

- password : Mật khẩu của người sử dụng để kết nối vào bộ máy cơ sở dữ liệu.

Hàm này trả về mã số nhận dạng nếu kết nối thành công, giá trị 0 (false) nếu việc kết nối có lỗi. Mã số nhận dạng này sẽ được sử dụng cho tất cả các yêu cầu tới bộ máy cơ sở dữ liệu sau này.

Ví dụ:

```
<?php
     $link = mysql_connect ("localhost", "root", "")
     or die ("Could not connect to MySQL Database");
?>
```

Sự kết nối này sẽ đóng lại khi gọi hàm **mysql_close()** hoặc kết thúc đoạn PHP script.

+ **mysql_pconnect()** : Hàm này tạo ra một liên kết bền vững với máy chủ MySQL.
Cú pháp :

```
int mysql_pconnect (string [hostname [:port] [:/path_to_socket]], string
[username], string [password]);
```

Tham số và giá trị trả về của hàm này cũng giống hàm mysql_connect(). Sự khác biết giữa hai hàm này là liên kết tới máy chủ MySQL không bị đóng lại kể cả khi kết thúc kịch bản (script) PHP hay gọi hàm mysql_close(). Mục đích của hàm này là luôn luôn duy trì liên kết tới máy chủ MySQL do luôn có sự yêu cầu tới máy chủ, tránh cho máy chủ phải tìm kiếm mã số nhận dạng mới từ đó giảm thời gian truy cập .

Chú ý : hàm này chỉ thực hiện được khi PHP được định cấu hình như là một module của Web server .

+ **mysql_close()** : Hàm này huỷ bỏ sự kết nối tới máy chủ MySQL .

Cú pháp :

```
int mysql_close(int [link_identifier]);
```

Tham số link_identifier là mã số nhận dạng tạo ra bởi hàm mysql_connect(). Hàm trả về là True nếu thành công, ngược lại là False .

Ví dụ mở và đóng kết nối CSDL MySQL

```
<?php
     $link = mysql_connect ("localhost", "root", "")
     or die ("Could not connect to MySQL Database");
     mysql_select_db("InterShop", $link);
```

```
    mysql_close($link);
?>
```

2.5.1.2 Các hàm thao tác tên CSDL

+ **mysql_create_db()** : Hàm tạo cơ sở dữ liệu

Cú pháp :

```
int  mysql_create_db(string name, int [link_identifier]) ;
```

Trong đó :

- string name : Tên của cơ sở dữ liệu cần tạo.

- int link_identifier : Mã số nhận dạng được cấp bởi hàm mysql_connect() . Chúng ta hoàn toàn có thể gửi câu lệnh SQL để tạo cơ sở dữ liệu thông qua hàm mysql_query() .

+ **mysql_drop_db()** : Hàm xoá cơ sở dữ liệu

Cú pháp :

```
int  mysql_drop_db(string name, int [link_identifier]);
```

Trong đó :

- string name : Tên của cơ sở dữ liệu cần xoá .

- int link_identifier : Mã số nhận dạng được cấp bởi hàm mysql_connect() .

Chúng ta hoàn toàn có thể gửi câu lệnh SQL để xoá cơ sở dữ liệu thông qua hàm mysql_query().

+ **mysql_select_db()** : Hàm cho cơ sở dữ liệu hoạt động .

Cú pháp :

```
int  mysql_select_db(string database_name, int [link_identifier]);
```

Trong đó:

- database_name : Tên của cơ sở dữ liệu mà sau này các hàm API khác của PHP sẽ thực hiện trên đó.

- int link_identifier : Mã nhận dạng được cấp bởi hàm mysql_connect().

Câu lệnh này sẽ gắn tên cơ sở dữ liệu với mã nhận dạng, sau này khi làm việc với link_identifier sẽ bao gồm cả cơ sở dữ liệu được chọn .

Ví dụ mở và đóng kết nối CSDL MySQL

```
<?php
     $link = mysql_connect ("localhost", "root", "")
     or die ("Could not connect to MySQL Database");
     mysql_select_db("InterShop", $link);
     mysql_close($link);
?>
```

2.5.1.3 Các hàm thao tác trên dữ liệu

+ **mysql_query()** : Hàm gửi câu lệnh SQL tới máy chủ MySQL .

Cú pháp :

```
int  mysql_query(string query, [int link_identifier]) ;
```

Trong đó :

- string query : Câu lệnh SQL cần gửi tới máy chủ MySQL .

- int link_identifier : Mã số nhận dạng, nó phải được thực hiện trong hàm mysql_select_db() trước đó .

+ **mysql_db_query()** : Hàm gửi câu lệnh SQL tới máy chủ MySQL .

Cú pháp :

```
int  mysql_db_query(string database, string query, int
[link_identifier]);
```

Trong đó :

- string database : Tên cơ sở dữ liệu câu lệnh SQL sẽ thực hiện trên đó.

- string query : Câu lệnh SQL cần thực hiện .
- link_identifier : Mã số nhận dạng được cấp bởi hàm mysql_connect()

Hàm này chỉ rõ câu lệnh được thực hiện trên cơ sở dữ liệu nào nên trước đó không cần thực hiện hàm mysql_select_db();

+mysql_insert_id() : Hàm lấy giá trị được sinh ra từ câu truy vấn INSERT trớc

Cú pháp :

```
int mysql_insert_id([link_identifier]) ;
```

trong đó:

- int link_identifier : Mã số nhận dạng được cấp bởi hàm mysql_connect() .

Hàm này trả về giá trị id được sinh ra trong cột AUTO_INCREMENT bởi câu truy vấn trước đó. Điều này chỉ có tác dụng trên link_identifier được chỉ ra trong hàm, nếu gọi hàm trên mà không chỉ định tham số link_identifier thì liên kết được mở cuối cùng sẽ được chỉ định.

Hàm mysql_insert_id() trả về giá trị 0 nếu câu truy vấn trước đó không sinh ra một giá trị AUTO_INCREMENT. Nếu ta muốn giữ lại giá trị cho lần sau, thì phải gọi hàm này ngay sau câu truy vấn sinh ra giá trị .

+ **mysql_fetch_row()** : Hàm trả về một mảng là giá trị của một bảng ghi hiện tại với chỉ số là số thứ tự của các trường (chỉ số bắt đầu từ 0). Sau đó hàm sẽ trỏ tới bảng ghi tiếp theo cho tới khi gặp bảng ghi cuối cùng hàm trả về giá trị false. Để truy xuất tới các giá trị của cột ta viết : tên_mảng[số thứ tự]

Cú pháp :

```
array  mysql_fetch_row( int result_identifier);
```

Trong đó :

-result_identifier là mã số trả về của hàm mysql_query() hoặc mysql_db_query() .

Ví dụ :

```
<?php
        $mysql = "select id, name from ds_thanhvien"; // cau lenh SQL
        $link = mysql_connect($host, $user, $password); //lay ma
        mysql_select_db($database_name, $link);
        $result = mysql_query($mysql ,$link);
        while ($row = mysql_fetch_row($result))
{
echo $row[0] ;
echo $row[1];
}
?>
```

+ **mysql_fetch_array()** : Hàm trả về một mảng là giá trị của một bảng ghi hiện tại, sau đó hàm sẽ trỏ tới bảng ghi tiếp theo cho tới khi gặp bảng ghi cuối cùng hàm trả về giá trị false.

Cú pháp :

```
array  mysql_fetch_array( int result_identifier [, int result_type] );
```

Trong đó :

-result_identifier là mã số trả về của hàm mysql_query() hoặc mysql_db_query() .

Để truy xuất đến các thành phần của cột :tên_biến_mảng["tên_trường"];

- result_type là một hằng số có thể nhận các giá trị sau:

-MYSQL_NUM : chỉ trả lại một mảng chứa các chỉ số là số (giống như hàm mysql_fetch_row())

-MYSQL_ASSOC: chỉ trả lại một mảng liên kết

-MYSQL_BOTH : trả lại mảng chứa đựng các chỉ số gồm cả các con số và chỉ số liên kết .

Hàm này là sự mở rộng của hàm mysql_fetch_row(). Nó cho phép truy cập trường dữ liệu của mảng kết quả không chỉ thông qua các chỉ số là các số mà chúng có thể là tên của các trường dữ liệu. Điều này làm cho việc lập trình đơn giản và chính xác hơn.

Ví dụ:

```
<?php
$mysql = "select id, name from ds_thanhvien";
$link = mysql_connect($host, $user, $password);
$result = mysql_db_query("php", $mysql);
while ($row = mysql_fetch_array($result))
 {
echo "user_id: ". $row["id"] ."<BR>\n";
echo "user_id: ". $row[0] ."<BR>\n";
echo "user_name: ". $row["name"] ."<BR>\n";
echo "user_name: ". $row[1] ."<BR>\n";
}
mysql_free_result ($result);
?>
```

+ **mysql_fetch_object()** : Hàm trả về một đối tượng là giá trị của một bảng ghi hiện thời. Sau đó hàm sẽ trỏ tới bảng ghi tiếp theo cho tới khi gặp bảng ghi cuối cùng hàm trả về giá trị false. Để truy xuất tới các giá trị của cột ta viết tên_object->tên_cột .

Cú pháp :

```
object  mysql_fetch_object(int  result_identifier);
```

Trong đó :

-result_identifier là mã số trả về của hàm mysql_query() hoặc mysql_db_query() .

Ví dụ :

```
<?php
  $mysql = "select id, name from ds_thanhvien";
  $link = mysql_connect($host, $user, $password);
  $result = mysql_db_query("php", $mysql);
  while ($row = mysql_fetch_object($result))
{
  echo $row->id ;
  echo $row->name;
}
?>
```

+mysql_fetch_assoc() :lấy về một dòng kết quả như là một mảng liên kết cú pháp: array mysql_fetch_assoc(int result_identifier)

Trong đó :

-result_identifier là mã số trả về của hàm mysql_query() hoặc mysql_db_query() .

Hàm trả về một mảng tương ứng với một bản ghi được lấy về và trả lại FALSE nếu không có bản ghi nào. Hàm này tương đương với hàm

array mysql_fetch_array() với tham số result_type là : MYSQL_ASSOC

ví dụ :

```
<?php
  $mysql = "select id, name from ds_thanhvien";
  $link = mysql_connect($host, $user, $password);
  $result = mysql_db_query("php", $mysql);
  while ($row = mysql_fetch_assoc($result))
{
echo $row["id"];
echo $row["name"];
}
?>
```

+ mysql_data_seek(): Di chuyển con trỏ bên trong "tập kết quả" (có được sau khi câu truy vấn SELECT được thực hiện)

Cú pháp:

```
 bool mysql_data_seek(int result_identifier, int row_number);
```

Trong đó :

-result_identifier là mã số trả về của hàm mysql_query(),mysql_db_query(), mysql_list_tables(), mysql_list_dbs() .

-row_number là chỉ số của bản ghi mà cần đặt con trỏ vào .

Hàm trả về true nếu thành công, false nếu lỗi .

Hàm này sẽ di chuyển con trỏ bên trong "tập kết quả" (được chỉ rõ bởi tham đối result_identifier) đến dòng có mã bằng tham đối row_number.

Các dòng trong tập kết quả được bắt đầu từ 0

Ví dụ:

```
<?php
    $link = mysql_pconnect ($host, $user, $password)
        or die ("Could not connect");
```

```
    $query = "SELECT last_name, first_name FROM friends";
    $result = mysql_db_query ("php",$query)
        or die ("Query failed");

    # fetch rows in reverse order

    for ($i = mysql_num_rows ($result) - 1; $i >=0; $i--) {
        if (! Mysql_data_seek ($result, $i)) {
            printf ("Cannot seek to row %d\n", $i);
            continue;
        }

        if(!($row = mysql_fetch_object ($result)))
            continue;

        printf ("%s %s<BR>\n", $row->last_name, $row->first_name);
    }

    mysql_free_result ($result);
?>
```

+ **mysql_num_rows()** :trả lại số dòng trong result_identifier (nơi chứa kết quả của câu lệnh SQL đã được thực hiện)

cú pháp:

```
mysql_num_rows(int result_identifier) ;
```

Trong đó :

-result_identifier là mã số trả về của hàm mysql_query(),mysql_db_query(), mysql_list_tables(), mysql_list_dbs() .

+**mysql_affected_rows()** : cú pháp : int mysql_affected_rows(int [link_identifier]) ;

Trong đó :

-int link_identifier là mã số nhận dạng, nó phải được thực hiện trong hàm mysql_select_db() trước đó .

Hàm trả về số dòng đã bị tác động bởi một câu truy vấn SQL :INSERT,UPDATE, DELETE trước đó theo tham số link_identifier. Nếu link_identifier không được chỉ định thì mã kết nối trước đó sẽ được chỉ định.

Chú ý :

- Nếu câu lệnh SQL trước đó là DELETE mà không có mệnh đề WHERE thì toàn bộ các bản ghi trong bảng đã bị xoá nhưng hàm mysql_affected_rows() sẽ trả về gián trị 0.

-Hàm này không có tác dụng đối với câu lệnh truy vấn SELECT. Để lấy được số dòng trả về (số dòng đã bị tác động) bởi câu lệnh SELECT ta dùng hàm mysql_num_rows().

+mysql_result() : lấy dữ liệu từ result_identifier

cú pháp :

```
mixed mysql_result(int result_identifier, int row, mixed [field]);
```

Trong đó :

-result_identifier là mã số trả về của hàm mysql_query(), mysql_db_query(), mysql_list_tables(), mysql_list_dbs() .

-row là bản ghi mà ta sẽ lấy dữ liệu

- field là trường trong dòng row mà ta sẽ lấy dữ liệu .

Các tham số result_identifier và row phải có, còn tham field là tùy chọn. Hàm sẽ trả lại các nội dung của dòng row và cột field từ tập kết quả được chỉ định bởi biến result_identifier. Nếu đối số field không được chỉ định rõ thì trường tiếp theo của bản ghi sẽ được trả về .

Ví dụ:

```
<?php
 $mysql = "select id, name from ds_thanhvien";
  $link = mysql_connect($host, $user, $password);
  $result = mysql_db_query("php", $mysql);
  echo  " mysql_result($result, 0, "id") <BR>\n";
  echo  " mysql_result($result, 0, "name") <BR>\n";
?>
```

+mysql_free_result() : Hàm giải phóng vùng bộ nhớ được liên kết với result_identifier .

cú pháp:

```
mysql_free_result(int result_identifier) ;
```

Trong đó :

-result_identifier là mã số trả về của hàm mysql_query(), mysql_db_query(), mysql_list_tables(), mysql_list_dbs() .

Hàm này chỉ được dùng nếu như bạn đánh giá thấy rằng kịch bản của bạn sử dụng quá nhiều bộ nhớ khi đang chạy. Gọi hàm này trên một trình xử lý kết quả sẽ giải phóng toàn bộ dữ liệu liên kết trong bộ nhớ .

Ngoài ra còn các hàm khác:

string **mysql_tablename (int result_identifier, int i)** : Hàm trả lại tên của bảng/csdl tại chỉ số i trong result_identifier.

string **mysql_field_name (int result_identifier, int field_index)** : Hàm trả lại tên của trường tại vị trí field_index trong mã result_identifier

int **mysql_list_dbs ([int link_identifier])** : Hàm trả lại một result_identifier là danh sách biến CSDL trên MySQL Server nếu thành công, lỗi trả về false .

int **mysql_list_tables (string database [, int link_identifier])** : Hàm trả về danh sách tất cả câc bảng trong một CSDL MySQL, thành công trả về một result identifier, giá trị false nếu có lỗi .

int **mysql_list_fields (string database_name, string table_name [, int link_identifier])** : Hàm trả về thông tin liên quan đến một bảng dữ liệu.

int **mysql_num_fields (int result_identifier)** : Trả về số trường trong tập kết quả .

int **mysql_num_rows (int result_identifier)** : Trả về số bản ghi trong tập kết quả, hàm này chỉ có giá đối với các câu lệnh SELECT ,để lấy lại số bản ghi được trả lại từ các lệnh :INSERT, UPDATE hoặc DELETE, dùng mysql_affected_rows().

string **mysql_field_type (int result_identifier, int field_index)** : Hàm trả về kiểu dữ liệu của trường tại vị trí field_index trong mã result_identifier .

int **mysql_field_len (int result_identifier, int field_offset)** : Hàm trả về độ dài của trường được chỉ định thông qua tham số field_offset .

array **mysql_fetch_lengths (int result_identifier)** : Hàm trả về một mảng tương ứng với các độ dài của mỗi trường trong bản ghi được lấy về bởi hàm mysql_fetch_row() hoặc false nếu có lỗi.

int **mysql_errno ([int link_identifier])** : Hàm trả về mã lỗi từ hàm thao tác CSDL MySQL trước ,trả về giá trị 0 nếu không có lỗi .

string **mysql_error ([int link_identifier])** : Hàm trả về xâu thông báo lỗi từ hàm thao tác CSDL MySQL trước, trả về xâu rỗng nếu không có lỗi .

object **mysql_fetch_field (int result_identifier [, int field_offset])** : Lấy thông tin về trờng từ tập kết quả rồi trả lại nh một đối tượng.

3. Luyện tập kỹ năng lập trình

Trước hết chúng ta phải thiết kế và xây dựng mô hình cơ sở dữ liệu

Trong PHPMyAdmin tạo Database tên project

```
create database project;
```

Trong Database project tạo Table user

```
create table user( id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT, username
VARCHAR(50) NOT NULL, password CHAR(50) NOT NULL, level CHAR(1) NOT NULL,
PRIMARY KEY(id));
```

Để kết nối cơ sở dữ liệu mySQL bạn sử dụng khai báp như sau:

```
<?php
     $link = mysql_connect ("localhost", "root", "")
     or die ("Could not connect to MySQL Database");
     mysql_select_db("project ", $link);
?>
```

Trong đó khai báo sau là kết nối cơ sở dữ liệu mySQL với tên server/ip cùng với username và password:

```
mysql_connect ("localhost", "root", "")
```

Và mysql_select_db("project", $link); để chọn tên cơ sở dữ liệu sau khi mở kết nối cơ sở dữ liệu, nếu biến $link có giá trị là false thì kết nối cơ sở dữ liệu không thành công.
Sau khi mở kết nối cơ sở dữ liệu mà không sử dụng thì bạn có thể đóng kết nối cơ sở dữ liệu với cú pháp như sau:

```
mysql_close($link);
```

Chẳng hạn, bạn khai báo trang connection.php để kết nối cơ sở dữ liệu và đóng kết nối ngay sau khi mở thành công.

```
<?php
     $link = mysql_connect ("localhost", "root", "")
     or die ("Could not connect to MySQL Database");
     mysql_select_db("project", $link);
     mysql_close($link);
?>
```

3.1 Viết module thêm mới thành viên bằng PHP và MYSQL

Thiết kế trang add_user.php với mã HTML như sau:

```
<form action=add_user.php method=POST>
  Level:
  <select name=level>
    <option value=1>Member</option>
    <option value=2>Admin </option>
  </select>
  <br />
  Username:
  <input type=text name=username size=25>
  <br />
  Password:
  <input type=password name=password size=25>
  <br />
  Re-Password:
  <input type=password name=re-password size=25>
```

```
  <br />
  <input type=submit name="adduser" value="Add New User">
</form>
```

Để thêm mẩu tin, bạn sử dụng hàm mysql_query(chuỗi Insert).

```
<?php
if(isset($_POST['adduser']))
{
  $u=$_POST['username'];
  $p=$_POST['password'];
  $l=$_POST['level'];

  $link = mysql_connect ("localhost", "root", "")
     or die ("Could not connect to MySQL Database");
  mysql_select_db("project", $link);

  $sql="select * from user where username='".$u."'";
  $query=mysql_query($sql);
  if(mysql_num_rows($query) != "" )
        {
         echo "Username nay da ton tai roi<br />";
        }
        else
        {
         $sql2="insert          into          user(username,password,level)
values('".$u."','".$p."','".$l."')";
         $query2=mysql_query($sql2);
               mysql_close($link);
         echo "Da them thanh vien moi thanh cong";
        }
}
?>
```

3.2 Viết module quản lý thành viên bằng PHP và MYSQL

Thiết kế trang có tên file là manage_user.php. Vì dữ liệu sẽ lặp lại toàn bộ user và ứng với từng user sẽ là 1 dòng dữ liệu được lặp lại. Chúng ta sẽ xây dựng 1 bảng gồm có STT là số thứ tự của từng user được đếm trên mỗi user khi lặp, username là tên truy cập của họ, level là cấp bậc của user (1 là member và 2 là admin), edit là cột chứa các link chỉnh sửa user, del là cột xóa các user.

```
<table align=center width=400 border=1>
<tr>
  <td>STT</td>
  <td>Username</td>
  <td>Level</td>
  <td>Edit</td>
  <td>Del</td>
</tr>
 <?php
        $link=mysql_connect("localhost","root","")  or  die("can't  connect
this database");
        mysql_select_db("project",$link);
        $sql="select * from user order by id DESC";
        $query=mysql_query($sql);
        if(mysql_num_rows($query) == "")
```

```
        {
                echo "<tr><td colspan=5 align=center>Chua co username
nao</td></tr>";
        }

        $stt=0;
    while($row=mysql_fetch_array($query))
        {
                $stt++;
                echo "<tr>";
                echo "<td>$stt</td>";
                echo "<td>$row[username]</td>";
                if($row[level] == "1")
                {
                        echo "<td>Member</td>";
                }
                else
                {
                        echo "<td>Admin</td>";
                }
                echo                                                    "<td><a
href=edit_user.php?userid=$row[id]>Edit</a></td>";
                echo                                                    "<td><a
href=del_user.php?userid=$row[id]>Del</a></td>";
                echo "</tr>";
        }
?>
</table>
```

Đoạn code ở trên tiến hành lựa chọn tất cả user có trong database. Đồng thời kiểm tra xem trong database có tồn tại user nào không. Nếu không sẽ xuất ra thông báo "chưa có username nào".

Để truy vấn dữ liệu bạn sử dụng hàm mysql_num_rows để biết được số mẩu tin trả về và hàm mysql_fetch_array để đọc từng mẩu tin và mảng sau đó trình bày giá trị từ mảng này.

3.3 Viết module sửa xóa thành viên bằng PHP và MYSQL

3.3.1 Sửa thành viên

Vì là trang chỉnh sửa thành viên, nên nội dung của chúng có phần sẽ giống với phần thêm thành viên, chỉ khác là các ô nhập liệu giờ đây đã có dữ liệu. Dữ liệu này chúng ta tiến hành lấy từ cơ sở dữ liệu thông qua biến truyền mà ở trang quản lý đã gửi edit_user.php?userid=$row[id]

Như vậy để lấy được giá trị từ liên kết này chúng ta sẽ sử dụng biến $_GET['userid']. Sau khi đã có được giá trị này, việc còn lại của bạn là lấy thông tin của id này từ cơ sở dữ liệu và đưa vào form để người dùng có thể chỉnh sửa.

```
<?php
$conn=mysql_connect("localhost","root","")  or  die("can't  connect  this
database");
mysql_select_db("project",$conn);
$id=$_GET['userid'];
if(isset($_POST['ok']))
{
```

```
        $u=$_POST['user'];
        $p=$_POST['pass'];
        $l = $_POST['level'];

 if($u && $p && $l )
 {
  $sql="update    user    set    username='".$u."',    password='".$p."',
level='".$l."' where id='".$id."'";
  mysql_query($sql);
  header("location : mana_user.php");
  exit();
 }
 else
 {
  if($u && $l)
  {
   $sql="update   user   set   username='".$u."',   level='".$l."'   where
id='".$id."'";
   mysql_query($sql);
   header("Location : mana_user.php");
   exit();
  }
 }
}
$sql="select * from user where id='".$id."'";
$query=mysql_query($sql);
$row=mysql_fetch_array($query);
?>

<form action="edit_user.php?userid=<?php echo $id?>" method=post>
  Level:
  <select name=level>
    <option  value=1  <?php  if($row['level']  ==  1)  echo  "selected";  ?>)
>Member</option>
    <option   value=2   <?php   if($row['level']   ==   2)   echo   "selected";
?>>Administrator</option>
  </select>
  <br />
  Username:
  <input type=text name=user size=20 value="<?php echo $row['username']?>"
/>
  <br />
  Password:
  <input type=password name=pass size=20 />
  <br />
  Re-password:
  <input type=password name=repass size=20 />
  <br />
  <input type=submit name=ok value="Edit User" />
</form>
```

3.3.2 Xóa thành viên

Đối với trang xóa dữ liệu, chúng ta cũng không cần phải xử lý quá nhiều. Bởi nhiệm vụ của chúng chỉ đơn giản là xóa đi những dòng trong bảng.

Như vậy cũng như trang edit chúng ta nhận giá trị từ nội dung liên kết ở trang quản lý đã gửi là del_user.php?userid=$row[id].

```
<?php
        $conn=mysql_connect("localhost","root","")  or  die("can't  connect
this database");
        mysql_select_db("project",$conn);
        $sql="delete from user where id='".$id."'";
        mysql_query($sql);
        header("Location : mana_user.php");
        exit();
?>
```

3.4 Viết module đăng nhập hệ thống bằng PHP và MYSQL

Thiết kế Form HTML để có màn hình đăng nhập khi người dùng truy cập.

Tiến hành kiểm tra xem username và password người sử dụng vừa nhập có trùng khớp với thông tin có trong cơ sở dữ liệu hay không ?. Nếu không thì chúng ta sẽ báo lỗi ngay. Ngược lại sẽ tiến hành lấy dữ liệu từ bảng và gán vào session.

```
<form action=login.php method=post>
  Username:
  <input type=text name=username size=25 />
  <br />
  Password:
  <input type=password name=password size=25 />
  <br />
  <input type=submit name=ok value="Dang Nhap" />
</form>

<?php
if(isset($_POST['ok']))
{
 if($_POST['username'] == NULL)
 {
  echo "Please enter your username<br />";
 }
 else
 {
  $u=$_POST['username'];
 }
 if($_POST['password'] == NULL)
 {
  echo "Please enter your password<br />";
 }
 else
 {
  $p=$_POST['password'];
 }
 if($u && $p)
 {
  $conn=mysql_connect("localhost","root","root") or die("can't connect this
database");
  mysql_select_db("project",$conn);
  $sql="select * from user where username='".$u."' and password='".$p."'";
  $query=mysql_query($sql);
  if(mysql_num_rows($query) == 0)
```

```
  {
   echo "Username or password is not correct, please try again";
  }
  else
  {
   $row=mysql_fetch_array($query);
   session_start();
   session_register("userid");
   session_register("level");
   $_SESSION['userid'] = $row[id];
   $_SESSION['level'] = $row[level];

  }
 }
}
?>
```

3.5 Xây dựng module Upload

Thiết kế form Upload như mã bên dưới:

```
<form enctype="multipart/form-data" action="" method="POST">
Choose a file to upload: <input name="userfile" type="file" /><br />
<input type="submit" value="Upload File" />
</form>
<?php
$save_path="upload/";
$file = $_FILES['userfile'];
move_uploaded_file($file['tmp_name'],$save_path.$file['name']);
?>
```

Ngay sau khi người dùng nhấn upload, hệ thống sẽ tạo ra tham số . Cụ thể là tên tạm (tmp_name), tên gốc (name), kích thước (size), định dạng mime (type) và lỗi (error) nếu có. Và khác với kiểu nhập liệu thông thường, khi chúng ta sử dụng <input type=file name=ten> thì lúc này sẽ phát sinh một biến môi trường mới là $_FILES[‘ten'

Ví dụ: $_FILES[‘ten'][‘name'] //Lấy ra tên gốc của file.

3.6 Tìm hiểu quy trình làm việc trên file trong PHP

Một trong những tác vụ đặc biệt của PHP đó là cho phép xử lý dữ liệu trực tiếp thông qua quá trình nhận và đọc nội dung trên 1 file dữ liệu. Điều này giúp cho PHP trở nên tinh tế và dễ tùy biến hơn khi xử lý 1 lượng dữ liệu có quy mô lớn. Việc thao tác mở, đọc, ghi, đóng file này cũng có ý nghĩa tuần tự như bạn đang làm việc trực tiếp trên 1 file dữ liệu thực thụ.

3.6.1 Đóng, mở 1 file trong PHP:

Để mở 1 file ta sử dụng cú pháp sau:

```
 fopen("Đường dẫn", thuộc tính).
```

Trong đó Đường dẫn chính là đường dẫn tới file cần mở.

Thuộc tính bao gồm các quyền hạn cho phép thao tác trên file đó như thế nào.

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

Ví dụ:

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

Tương tự như thế, để đóng 1 file ta có cú pháp như sau: fclose(file vừa mở)

Ví dụ:

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

Việc mở và đóng này không có ý nghĩa là chúng đã được đọc. Muốn đọc được nội dung của file chúng ta lại tiếp tục với thao tác lấy dữ liệu từ file nữa.

3.6.2 Đọc và ghi file trong PHP.

3.6.2.1 Đọc 1 file trong PHP

PHP cho ta nhiều sự lựa chọn trong việc đọc 1 file. Có nhiều hình thức hỗ trợ nhưng hiện nay 2 hình thức phổ biến nhất vẫn là đọc file theo từng dòng và đọc file theo từng ký tự.

- Đọc file theo từng dòng:

Cú pháp :

```
fgets(file vừa mở).
```

Ví dụ:

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

- Đọc file theo từng ký tự:

Cú pháp :

```
 fgetc(file vừa mở).
```

Ví dụ:

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

Quy trình đọc sẽ diễn ra theo từng yêu cầu của cú pháp sử dụng. Nhưng sẽ có sự ràng buộc bởi việc kiểm tra đã đến cuối file chưa ?.

Ở đây chúng ta dùng cú pháp sau:

```
feof(file vừa mở)
```

Ví dụ:

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

3.6.2.2 Ghi 1 file trong PHP

PHP cung cấp cho ta 1 cú pháp nhỏ để ghi dữ liệu vào 1file

Cú pháp cơ bản :

```
fwrite("file vừa mở", "Nội dung cần ghi vào file")
```

Ví dụ:

```
<?php
$fp=fopen("test.txt",a)or exit("khong tim thay file can mo");
$news="HOC VIEN CONG NGHE THONG TIN";
fwrite($fp,$news);
fclose($fp);
?>
```

Việc sử dụng file một cách thành thạo sẽ giúp bạn dễ dàng vận hành các ứng dụng mang quy mô vừa và nhỏ như: website nhiều ngôn ngữ, bộ đếm,…và cả những công nghệ web mới như XML một cách dễ dàng. Qua bài học này chúng ta cũng hiểu được nguyên lý hoạt động, trình tự xử lý 1 file dữ liệu khi chúng được triệu gọi trong tài liệu PHP.

3.7 Phân trang ứng dụng

Khi truy vấn dữ liệu mà nhận được một danh sách kết quả quá dài, người ta thường phải phân trang ứng dụng cho phù hợp .

Nguyên tắc của việc phân trang ứng dụng như sau:

- Bước 1: Tính toán số lượng bản ghi thỏa mãn điều kiện trả về (thường sử dụng hàm count trong câu lệnh SQL).
- Bước 2: Xác định số lượng bản ghi sẽ hiển thị trên một trang.
- Bước 3: Dựa trên các thông tin có được từ bước 1 và 2, xác định được số trang cần hiển thị.
- Bước 4: Tính toán số lượng bản ghi sẽ hiển thị tính từ trang nào đó do NSD lựa chọn (Sử dụng câu lệnh LIMIT).

Dưới đây là 2 function:

- Function GetPageLinks ($Sql, $PageSize): Trả về một chuỗi văn bản chứa số trang hiển thị, với dữ liệu vào bao gồm câu lệnh SQL ($Sql) xác định số lượng bản ghi thỏa mãn điều kiện tìm kiếm, và "kích thước" của một trang ($PageSize)

```
function GetPageLinks($Sql,$PageSize)
{
        $result=mysql_query ($Sql);
        if (!$result or mysql_num_rows ($result)==0)
        {
        }
        else
        {
                $line=mysql_fetch_array($result);
                $Pages=ceil($line[0]/$PageSize);
                if ($Pages>1)
                {
                        $PageLink="Trang ";
                        for ($i=0;$i<=$Pages-1;$i++)
                        {
                                $j=$i+1;
                                if ($j==$_GET["page"])
```

```
                                        {
                                                $PageLink.=" {$j} | ";
                                        }
                                else
                                        {
                                                $NewGet="";
                                                reset ($_GET);
                                                while   (list($key,   $val)   =
each($_GET))
                                                {
                                                   if ($key!='page')
                                                   {

        $NewGet.="&{$key}={$val}";
                                                   }

                                                }
                                                $NewGet.="&page={$j}";
                                                $NewGet=substr($NewGet,1);

                                                $PageLink.="                <a
href='?{$NewGet}'>{$j}</a> | ";
                                        }
                        }
                        $PageLink=substr($PageLink,0,-2);
                }
        }
        return $PageLink;
}
```

Hàm tiếp theo sẽ hiển thị danh sách các record của một trang nào đó, đồng thời demo cách sử dụng hàm GetPageLinks ở trên:

(VD dưới đây sử dụng một câu truy vấn lấy dữ liệu từ một bảng dulieu với một Category có id xác định)

```
function LoadList()
{
        if (isset($_GET["CatId"]) and is_numeric ($_GET["CatId"]))
                        $Dieukien="where CatId={$_GET["CatId"]}";

// Cau lenh truy van chinh khi chua phan trang:
                $Sql="Select * from dulieu {$Dieukien} ";
// Tinh toan so luong ban ghi tra ve:
                $PageSize=20;
                $CountSQL="Select count(*) from dulieu {$Dieukien}";
                $PageLinks= GetPageLinks ($CountSQL,$PageSize);

                if (isset ($_GET["page"]) and is_numeric ($_GET["page"]))
                {
                        $StartNum=$PageSize * ($_GET["page"]-1); // Xac dinh
vi tri ban ghi bat dau
                }
                else
                {
                        $StartNum=0;
                }
//Tiep tuc xay dung cau lenh truy van de lay du lieu theo trang
```

```
                $Sql.= " Limit {$StartNum}, {$PageSize}";
                $result=mysql_query ($Sql);
                if (!$result or mysql_num_rows ($result)==0)
                        {
                                $tmp="Híc, tui chẳng mò được chi cả!";
                        }
                else
                        {
                                $tmp="{$PageLinks}";
                                //========== Hiển thị dữ liệu, các bạn tự
viết cho phù hợp với yêu cầu
                                $tmp.="{$PageLinks}";
                        }

                return $tmp;
}
```

3.8 Viết module giỏ hàng - shopping cart

Xây dựng hệ thống giỏ hàng (shopping cart) một ứng dụng phổ biến rất thường gặp trên các website Thương mại điện tử cung cấp sản phẩm hiện nay.

3.8.1 Xây dựng trang hiển thị sản phẩm

Thiết kế bảng books trong Database như sau:

```
CREATE TABLE `books` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `title` varchar(255) NOT NULL,
  `author` varchar(100) NOT NULL,
  `price` int(30) NOT NULL,
  PRIMARY KEY  (`id`)
);
```

Ở trên là một bảng lưu thông tin của các quyển sách với tên, tác giả, giá tiền của các quyển sách

Khởi tạo một số bản ghi dữ liệu:

```
INSERT INTO `books` VALUES (1, 'PHP Can Ban', 'Kenny', 115);
INSERT INTO `books` VALUES (2, 'PHP Nang Cao', 'Kenny', 150);
INSERT INTO `books` VALUES (3, 'PHP Framework', 'Kenny', 300);
INSERT INTO `books` VALUES (4, 'Joomla Can Ban', 'Kenny', 100);
```

Sau khi đã hoàn tất việc chuẩn bị cơ sở dữ liệu và giao diện, tiếp theo ta sẽ xây dựng trang hiển thị các món hàng trên website, cho phép người sử dụng lựa chọn sách khi truy cập.

Để liệt kê danh sách các quyển sách đang có trong database(trang INDEX.PHP), ta cần kết nối CSDL với thao tác code trang INDEX.PHP như sau.

```
<?php
  session_start();
        if(isset($_SESSION['cart']))
        {
```

```
                foreach($_SESSION['cart'] as $k=>$v)
                {
                        if(isset($k))
                        {
                        $ok=2;
                        }
                }
        }

        if ($ok != 2)
         {
                echo '<p>Ban khong co mon hang nao trong gio hang</p>';
        } else {
                $items = $_SESSION['cart'];
                echo  '<p>Ban  dang  co  <a  href="cart.php">'.count($items).'
mon hang trong gio hang</a></p>';
        }
?>
</div>
<?php
$connect=mysql_connect("localhost","root","root")
or die("Can not connect database");
mysql_select_db("shop",$connect);
$sql="select * from books order by id desc";
$query=mysql_query($sql);
if(mysql_num_rows($query) > 0)
{
        while($row=mysql_fetch_array($query))
        {
                echo "<div class=pro>";
                echo "<h3>$row[title]</h3>";
                echo      "Tac      Gia:      $row[author]      -      Gia:
".number_format($row[price],3)." VND";
                echo  "<p  align=right><a  href=addcart.php?item=$row[id]>Mua
Sach Nay</a></p>";
                echo "</div>";
        }
}
?>
```

Đoạn code ở trên thực thi việc hiển thị sách nếu trong CSDL ít nhất 1 record. Và chúng sẽ liệt kê tiêu đề sách, tác giả, giá tiền. Ở đây sử dụng number_format() để lấy ra 3 số 000 cuối, ứng với đơn vị tiền tệ của Việt Nam là VNĐ.

Sau cùng tạo một liên kết cho phép thêm quyển sách đó vào giỏ hàng nếu người mua cảm thấy ưng ý. (addcart.php sẽ là trang thêm vào giỏ hàng với tham số là mã của quyển sách).

Nói đến ứng dụng shopping cart thì hiện nay có nhiều phương pháp code. Có thể sử dụng chuỗi để lưu giỏ hàng, cũng có thể lưu vào CSDL nháp giỏ hàng của người dùng và cũng có thể sử dụng mảng và session để lưu thông tin giỏ hàng. Trong khuôn khổ bài viết này, chúng ta sẽ sử dụng session và mảng để lưu thông tin giỏ hàng.

Tại trang addcart.php cần khởi tạo một session và lưu mã quyển sách vào một mảng. Cụ thể là: $_SESSION[‘cart'][$id]. Với $id là mã quyển sách mà người dụng đã chọn ở trang xem hàng hóa (index.php). Mục đích chính của trang addcart này là lưu trữ hoặc tính toán lại số lượng sản phẩm khi mà họ lựa chọn. (lưu ý là số lượng các món hàng).

Một vấn đề đặt ra trong trang này, là làm thế nào để nhận biết món hàng người đó đã chọn hay chưa. Chẳng hạn. Lần đầu tôi chọn mua quyển A, sau đó tôi quay lại chọn mua tiếp quyển A. vậy trong giỏ hàng phải ghi nhận số lượng quyển A này là 2. Chứ không thể chỉ lưu là 1 được.

Vậy, lúc này ta sẽ kiểm tra xem. Quyển sách mà ta vừa chọn có tồn tại trong giỏ hàng hay chưa. Nếu có, ta phải tiến hàng lấy số lượng đang có tăng lên 1 đơn vị. Còn nếu không, ta phải gán số lượng của chúng là 1.

Trang ADDCART.PHP như sau:

```
<?php
session_start();
session_register("cart");
$id=$_GET['item'];
if(isset($_SESSION['cart'][$id]))
{
        $qty = $_SESSION['cart'][$id] + 1;
}
else
{
        $qty=1;
}
$_SESSION['cart'][$id]=$qty;
header("location:cart.php");
exit();
?>
```

Nôm na, chúng ta có thể hiểu addcart chỉ đơn giản là xử lý số lượng hàng hóa và lưu chúng ở dạng mảng mà thôi.

Như vậy tại trang mua sách(trang INDEX.PHP), ta cũng cần cho khách hàng biết rằng trong giỏ hàng của họ hiện đang có bao nhiêu món hàng. Hoặc nếu chưa có món nào, ta cũng phải báo cho họ biết về việc đó.

Vậy khi nào thì giỏ hàng rỗng ?. Đó là khi session của giỏ hàng không tồn tại Id của quyển sách nào. Cụ thể, $_SESSION[‘cart'][id]. Khi id không tồn tại trong session này thì cũng là lúc giỏ hàng không tồn tại.

Vậy trước khi cho hiển thị giỏ hàng, ta cần kiểm tra xem có tồn tại id nào trong giỏ hàng hay không. Và vì id lưu ở dạng mảng đa chiều, nên ta cần dùng vòng lặp duyệt mảng foreach.

```
foreach($_SESSION[‘cart'] as $k=>$v)
```

Với $k có ý nghĩa tương đương $id quyển sách và $v tương đương là số lượng của quyển sách trong giỏ hàng. Vậy nếu tồn tại biến $k, thì tức có nghĩa là trong giỏ hàng có sách. Khi đó ta sử dụng một biến đã để báo hiệu rằng sách có tồn tại trong giỏ hàng hay không.

3.8.2 Xây dựng hệ thống quản lý giỏ hàng

Sau khi đã thêm một món hàng, tại giỏ hàng ta cũng cần phải xử lý để hiển thị các món hàng đã có trong giỏ. Chúng ta tạm lưu mã sách trong session. Vậy ở trang giỏ hàng này, ta sẽ khởi tạo session và sử dụng vòng lặp duyệt mảng foreach để lặp toàn bộ mã sách đang lưu trong session. Tiếp tục, ta lại chuyển chúng sang dạng chuỗi bằng hàm implode().

```
<?php
Session_start();
Foreach($_SESSION['cart'] as $key=>$value)
{
 $item[]=$key;
}
$str=implode(",",$item);
?>
```

Giả sử lúc này chuỗi của chúng ta sẽ có dạng 7,8,9. Công việc tiếp theo là kết nối CSDL để liệt kê các sản phẩm có mã như ở trên. Thay vì sử dụng select * from tên_bảng where id= ??. Thì để tối ưu hơn, tôi sẽ sử dụng phép in trong SQL. Lúc này câu truy vấn sẽ tương đương:

```
<?php
$sql="select * from books where id in ('$str') order by id desc";
$query=mysql_query($sql);
while($row=mysql_fetch_array($query))
?>
```

Tiếp tục, ta lặp toàn bộ thông tin sách bao gồm tên, tác giả, giá tiền và cả số lượng mà ta đã lưu trong session là $_SESSION['cart']['ID_Món_Hàng']. ID_Món_hàng chính là thông tin ta lặp ra từ CSDL ($row[id]).

Bên cạnh đó tại phần số lượng, ta sẽ đưa giá trị lưu ở session ra textbox (vì tại giỏ hàng, người dùng được phép điều chỉnh số lượng, nên lúc này ta cần tạo textbox cho họ điều chỉnh). Vì lặp toàn bộ các quyển sách nên tại tên tên của textbox số lượng ta cũng cần truyền id để nhận biết số lượng đó thuộc ID của quyển sách nào.

```
<?php
echo "<p align=right>So Luong: <input type=text name=qty[$row[id]] size=5
value={$_SESSION['cart'][$row[id]]}> - ";
?>
```

Chúng ta cũng cho người dùng được phép xóa 1 món hàng nào đó ra khỏi giỏ hàng của họ. Bằng cách truyền mã quyển sách của từng quyển vào liên kết delcart.php.

```
<?php
echo "<a href=delcart.php?productid=$row[id]>Xoa Sach Nay</a></p>";
?>
```

Tại đây, ta cũng cần tính luôn giá tiền của từng quyển sách tương ứng với số lượng mà họ đã chọn. Như vậy, số lượng là phần ta lưu ở session, còn giá tiền là phần ta lấy ra từ CSDL ứng với mảng $row ($row['price']).

```
<?php
echo "<p align=right> Gia tien cho mon hang: ".
number_format($_SESSION['cart'][$row[id]]*$row[price],3) ." VND</p>";
?>
```

Sau cùng, ta cần tính tổng tiền của toàn bộ sản phẩm có trong giỏ hàng. Bằng cách cộng dồn tổng giá tiền của từng món.

```
<?php
$total+=$_SESSION['cart'][$row[id]]*$row[price];
?>
```

Phần còn lại, là chúng ta hiển thị giá tiền với đúng định dạng VND của Việt Nam.

```
<?php
echo "<b>Tong tien cho cac mon hang: <font color=red>".
number_format($total,3)." VND</font></b>";
?>
```

Đồng thời, ta cũng tạo nút cho phép người dùng cập nhật. Và cho phép người dúng xóa toàn bộ giỏ hàng.

```
<?php
echo "<input type="submit" name=submit value="Cap Nhat Gio Hang">";
echo "<div class=pro align=center>";
echo "<b><a href=index.php>Mua Sach Tiep</a> - <a
href=delcart.php?productid=0>Xoa Bo Gio Hang</a></b>";
?>
```

Như vậy, code đầy đủ sẽ là:

```
<?php
echo "<form action=cart.php method=post>";
foreach($_SESSION['cart'] as $key=>$value)
{
 $item[]=$key;
}
$str=implode(",",$item);
$connect=mysql_connect("localhost","root","root") or die("Can not connect
database");
mysql_select_db("shop",$connect);
$sql="select * from books where id in ($str)";
$query=mysql_query($sql);
while($row=mysql_fetch_array($query))
{
 echo "<div class=pro>";
 echo "<h3>$row[title]</h3>";
 echo "Tac gia: $row[author] - Gia: ".number_format($row[price],3)." VND<br
/>";
 echo "<p align=right>So Luong: <input type=text name=qty[$row[id]] size=5
value={$_SESSION['cart'][$row[id]]}> - ";
 echo "<a href=delcart.php?productid=$row[id]>Xoa Sach Nay</a></p>";
 echo "<p align=right> Gia tien cho mon hang: ".
number_format($_SESSION['cart'][$row[id]]*$row[price],3) ." VND</p>";
 echo "</div>";
 $total+=$_SESSION['cart'][$row[id]]*$row[price];
}
echo "<div class=pro align=right>";
echo "<b>Tong tien cho cac mon hang: <font color=red>".
number_format($total,3)." VND</font></b>";
echo "</div>";
echo "<input type="submit" name=submit value="Cap Nhat Gio Hang">";
```

```
echo "<div class=pro align=center>";
echo      "<b><a      href=index.php>Mua      Sach      Tiep</a>      -      <a
href=delcart.php?productid=0>Xoa Bo Gio Hang</a></b>";
echo "</div>";
?>
```

Sau khi thiết lập thành công trang giỏ hàng cơ bản, lúc này ta đã có thể thêm sách một cách dễ dàng. Tuy nhiên, giả sử trong trường hợp không có sách thì sao ?. Chúng ta vẫn chưa xét đến trường hợp giỏ hàng rỗng thì sẽ như thế nào. Vậy khi nào thì giỏ hàng rỗng ?. Đó là khi session của giỏ hàng không tồn tại Id của quyển sách nào. Cụ thể, $_SESSION[‘cart'][id]. Khi id không tồn tại trong session này thì cũng là lúc giỏ hàng không tồn tại.

Vậy trước khi cho hiển thị giỏ hàng, ta cần kiểm tra xem có tồn tại id nào trong giỏ hàng hay không. Và vì id lưu ở dạng mảng đa chiều, nên ta cần dùng vòng lặp duyệt mảng foreach.

```
foreach($_SESSION[‘cart’] as $k=>$v)
```

Với $k có ý nghĩa tương đương $id quyển sách và $v tương đương là số lượng của quyển sách trong giỏ hàng. Vậy nếu tồn tại biến $k, thì tức có nghĩa là trong giỏ hàng có sách.

```
<?php
$ok=1;
if(isset($_SESSION['cart']))
{
 foreach($_SESSION['cart'] as $k => $v)
 {
  if(isset($k))
  {
   $ok=2;
  }
 }
}
if($ok == 2)
{
// code xử lý giỏ hàng ở trên.
}
?>
```

Đoạn code này, ta sử dụng biến $ok để làm biến kiểm tra, mặc định khi load dữ liệu biến $ok sẽ bằng 1. Và khi trong giỏ hàng tồn tại sách thì chúng ta sẽ thay đổi biến $ok thành 2. Và gọi giỏ hàng như code ở trên.

Ở phần trên, ta cũng có đề cập khi người dùng tiến hành chỉnh sửa số lượng từng món hàng đơn lẻ qua textbox và nhấn cập nhận thì hệ thống sẽ tiến hành chỉnh sửa lại thông tin giỏ hàng. Vậy chúng ta sẽ xử lý như thế nào cho trương hợp đó.

Nếu chú ý, các bạn sẽ thấy dòng code chứa textbox cho phép người dùng nhập số lượng có một tham số đặc biêt là name=qty[$row[id]]. Vậy tham số này được dùng để làm gì ?.

Tham số này, nói cho chúng ta biết số lượng đang hiển thị là thuộc mã sản phẩm nào. Vậy khi tiến hành cập nhật giỏ hàng ta sẽ kiểm tra, nếu $qty[$row[id]] mà có giá trị là 0. Tức là người đó muốn xóa bỏ giỏ hàng. Ngược lại, ta chỉ viêc cập nhật giỏ hàng $id tương ứng với số lượng nhập ở textbox.

```
<?php
if(isset($_POST['submit']))
{
 foreach($_POST['qty'] as $key=>$value)
 {
  if( ($value == 0) and (is_numeric($value)))
  {
   unset ($_SESSION['cart'][$key]);
  }
  elseif(($value > 0) and (is_numeric($value)))
  {
   $_SESSION['cart'][$key]=$value;
  }
 }
 header("location:cart.php");
}
?>
```

Việc xóa món hàng ở trên chỉ đơn giản là hủy bỏ session của id đó. Việc cập nhật số lượng chỉ đơn giản là gán đè số lượng người nhập (thẻ value trong textbox) vào số lượng đang lưu trong session.

Vậy code hoàn chỉnh của trang cart.php này sẽ như sau:

```
<?php
if(isset($_POST['submit']))
{
        foreach($_POST['qty'] as $key=>$value)
        {
                if( ($value == 0) and (is_numeric($value)))
                {
                        unset ($_SESSION['cart'][$key]);
                }
                elseif(($value > 0) and (is_numeric($value)))
                {
                        $_SESSION['cart'][$key]=$value;
                }
        }
        header("location:cart.php");
}
?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <title>Demo Shopping Cart - Created By My Kenny</title>
        <link rel="stylesheet" href="style.css" />
</head>
<body>
<h1>Demo Shopping Cart</h1>
<?php
$ok=1;
if(isset($_SESSION['cart']))
{
        foreach($_SESSION['cart'] as $k => $v)
        {
                if(isset($k))
                {
                        $ok=2;
```

```
                }
        }
}
if($ok == 2)
{


                        echo "<form action=cart.php method=post>";
                        foreach($_SESSION['cart'] as $key=>$value)
                        {
                                $item[]=$key;
                        }
                        $str=implode(",",$item);
                        $connect=mysql_connect("localhost","root","root")  or
die("Can not connect database");
                        mysql_select_db("shop",$connect);
                        session_start();
                        $sql="select * from books where id in ($str)";
                        $query=mysql_query($sql);
                        while($row=mysql_fetch_array($query))
                        {
                        echo "<div class=pro>";
                        echo "<h3>$row[title]</h3>";
                        echo     "Tac     gia:     $row[author]     -     Gia:
".number_format($row[price],3)." VND<br />";
                        echo  "<p  align=right>So  Luong:  <input  type=text
name=qty[$row[id]] size=5 value={$_SESSION['cart'][$row[id]]}> - ";
                        echo    "<a    href=delcart.php?productid=$row[id]>Xoa
Sach Nay</a></p>";
                        echo  "<p  align=right>  Gia  tien  cho  mon  hang:  ".
number_format($_SESSION['cart'][$row[id]]*$row[price],3) ." VND</p>";
                        echo "</div>";
                        $total+=$_SESSION['cart'][$row[id]]*$row[price];
                        }
                echo "<div class=pro align=right>";
                echo  "<b>Tong  tien  cho  cac  mon  hang:  <font  color=red>".
number_format($total,3)." VND</font></b>";
                echo "</div>";
                echo  "<input  type=\"submit\"  name=submit  value=\"Cap  Nhat
Gio Hang\">";
                echo "<div class=pro align=center>";
                echo   "<b><a   href=index.php>Mua   Sach   Tiep</a>   -   <a
href=delcart.php?productid=0>Xoa Bo Gio Hang</a></b>";
                echo "</div>";
        }

else
        {
                echo "<div class=pro>";
                echo '<p align=center>Ban khong co mon hang nao trong gio
hang<br /><a href=index.php>Buy Ebook</a></p>';
                echo "</div>";
        }
?>
</body>
</html>
```

Và cuối cùng, khi người dùng nhấn xóa toàn bộ giỏ hàng hay chỉ xóa một món hàng, ta sẽ gọi tới trang delcart.php. Vậy trang này sẽ xử lý như thế nào ?.

Tại đây, ta có thể nhận tham số là: $_GET[‘productid'];

Tham số này sẽ là $id mà chúng truyền qua liên kết. Nếu xóa toàn bộ giỏ hàng, tức ta sẽ truyền cho nó giá trị bằng 0. Lúc này, ta sẽ hủy toàn bộ $_SESSION[‘cart']. Ngược lại, nếu là một $id cụ thể, thì ta chỉ xóa món hàng đó mà thôi. $_SESSION[‘cart'][$id].

Code hoàn chỉnh của file delcart.php như sau:

```
session_start();
$cart=$_SESSION['cart'];
$id=$_GET['productid'];
if($id == 0)
{
        unset($_SESSION['cart']);
}
else
{
unset($_SESSION['cart'][$id]);
}
header("location:cart.php");
exit();
?>
```

Như vậy, chúng ta đã hoàn tất việc xây dựng hoàn chỉnh một hệ thống shopping cart đơn giản. Tuy rằng, đây không phải là một bài viết hoàn chỉnh trong việc xây dựng mô hình thương mại điện tử. Nhưng quá đó, phần nào giúp các bạn hiểu và dễ dàng phát triển hệ thống của mình có tổ chức hơn.