Giáo trình
Tự học lập trình PHP

Môn học: PHP

Bài 1

Những vấn đề chính sẽ được đề cập trong bài học:

- ✓ *Giới thiệu PHP*
- ✓ *Cấu hình IIS, Apache Web Server*
- ✓ *Cài đặt PHP.*
  - o *Cài đặt PHP.*
  - o *Cấu hình ứng dụng PHP*
- ✓ *Giới thiệu PHP.*
  - o *PHP Script.*
  - o *Ghi chú trong PHP*
  - o *In nội dung bằng PHP*

1. GIỚI THIỆU PHP

PHP viết tắt của chữ Personal Home Page ra đời năm 1994 do phát minh của Rasmus Lerdorf, và nó tiếp tục được phát triển bởi nhiều cá nhân và tập thể khác, do đó PHP được xem như một sản phẩm của mã nguồn mở.

PHP là kịch bản trình chủ (server script) chạy trên phía server (server side) như cách server script khác (asp, jsp, cold fusion).

PHP là kịch bản cho phép chúng ta xây dựng ứng dụng web trên mạng internet hay intranet tương tác với mọi cơ sở dữ liệu như mySQL, PostgreSQL, Oracle, SQL Server và Access.

Lưu ý rằng, từ phiên bản 4.0 trở về sau mới hỗ trợ session, ngoài ra PHP cũng như Perl là kịch bảng xử lý chuỗi rất mạnh chính vì vậy bạn có thể sử dụng PHP trong những có yêu cầu về xử lý chuỗi.

2. CÀI ĐẶT PHP

Cài đặt PHP trên nền Windows thì sử dụng php-4.0.6-Win32.zip, sau khi cài đặt ứng dụng này trên đĩa cứng sẽ xuất hiện thư mục PHP, trong thự mục này sẽ có tập tin php4ts.dll và php.exe cùng với thư mục sessiondata.

Ngoài ra, trong thư mục WINDOW hoặc WINNT sẽ xuất hiện tập tin php.ini, tập tin này cho phép bạn cấu hình cho ứng dụng PHP. Chẳng hạn, khi sử dụng session, PHP cần một nơi để lưu trữ chúng, trong tập tin này mặc định là session.save_path = C:\PHP\sessiondata, nếu bạn cài đặt PHP với thư mục PHP trên đĩa D thì bạn cần thay đổi đường dẫn trong khai báo này.

Tương tự như vậy, khi có lỗi trong trangPHP thì lỗi thường xuất hiện khi triệu gọi chúng, để che dấu các lỗi này thì bạn cần khai báo display_errors = Off thay vì chúng ở trạng thái display_errors = On.

Ngoài ra, trang PHP cũng có thể trình bày một số warning khi chúng phát hiện cú pháp không hợp lý, chính vì vậy để che dấu các warning này thì bạn cũng cần khai báo trạng thái Off thay vì On như assert.warning = Off.

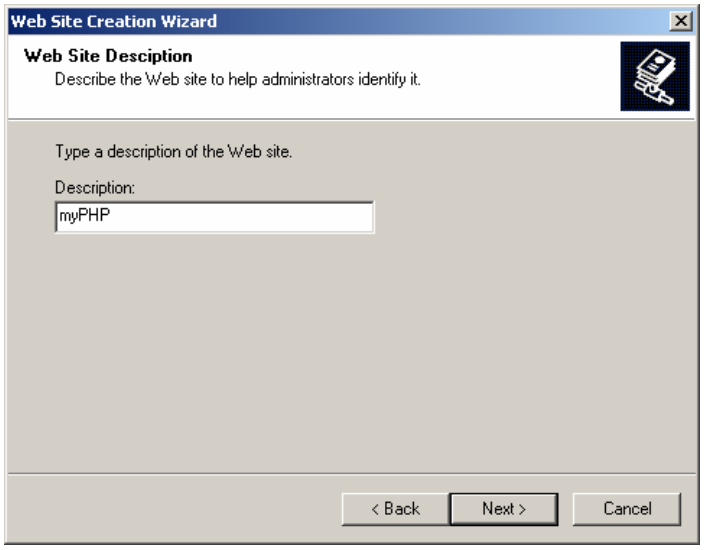
3. CẤU HÌNH ỨNG DUNG PHP

3.1. Cấu hình IIS

Sau khi cài đặt hệ điều hành Windows NT hay 2000 trở về sau, bằng cách khai báo mới một web site hay virtual site trong một site đang có theo các bước như sau:

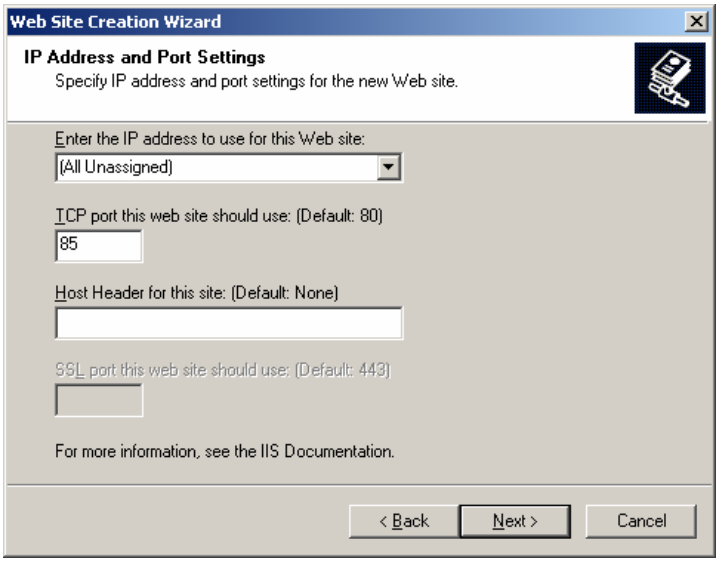
1. Tạo một thư mục có tên myPHP để lưu trữ các tập tin PHP
2. Khởi động IIS (tự động khởi động nếu Windows NT/2000)

3. Chọn Start | Programs | Administrative Tools | Internet Information Server
4. Nếu tạo virtual site thì chọn Default Web Ste | R-Click | New | Virtual Site
5. Trong trường hợp tạo mới Site thì Default Web Ste | R-Click | New | Site
6. Nếu chọn trường hợp 4 thì bạn cung tấp diễn giải của site như hình 1-1



Hình 1-1: Khai báo diễn giải

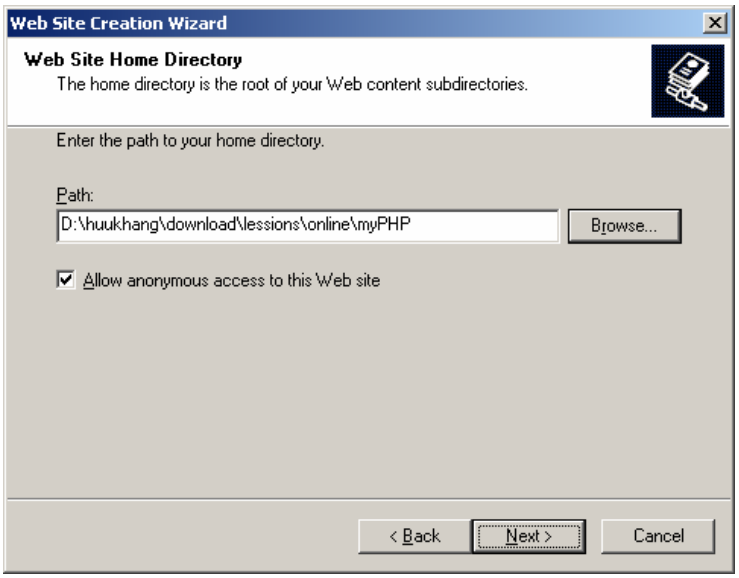
7. Chọn nút Next và khai báo IP và port, trong trường hợp bạn không sử dụng port 80 cho ứng site khác thì chọn giá trị mặc định. Tuy nhiên nếu có nhiều ứng dụng trước đó đã cấu hình trong IIS thì bạn có thể thay đổi port khác, ví dụ chọn port 85 như hình 1-2.



Hình 1-2: Khai báo IP và Port

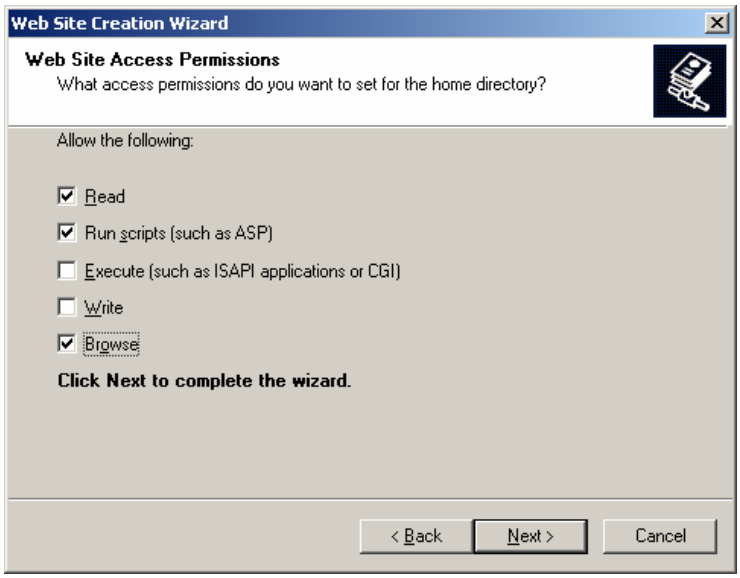
Lưu ý rằng, port 80 là port chuẩn điều này có nghĩa là khi triệu gọi trên trình duyệt bạn không cần gõ port, ví dụ http://localhost/. Đối với trường hợp port khác thì bạn phải gõ tương tự như http://localhost:85/

8. Chọn Next, bạn chọn thư mục của ứng dụng, đối với trường hợp này chúng ta chọn vào thư mục myPHP, chẳng hạn trong trường hợp này chúng ta chọn htư mục myPHP như hình 1-3.



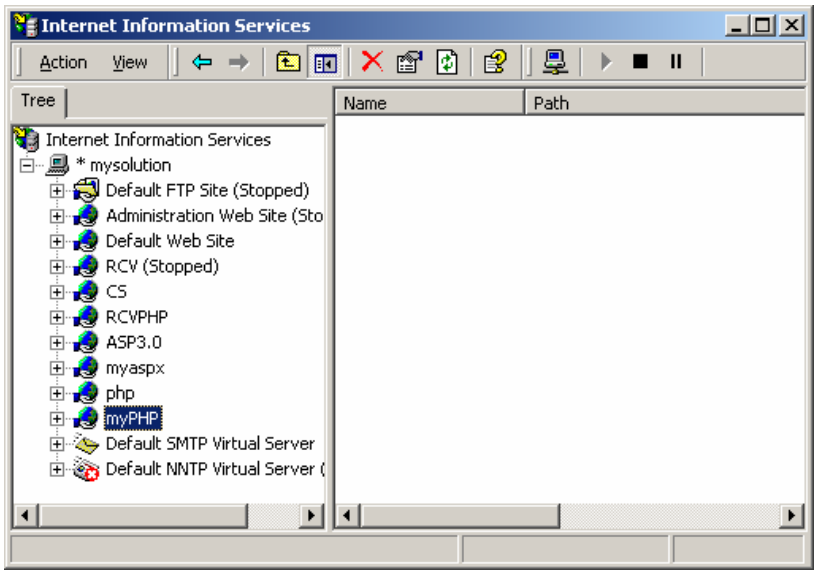
Hình 1-3: Chọn thư mục myPHP

9. Kế đến chọn quyền truy cập web site, trong trường hợp đang thiết kế thì bạn chọn vào Browse. Ngoài ra, nếu bạn cho phép người sử dụng internet có thể thực thi tập tin thực thi từ xa thì chọn vào tuỳ chọn execute.



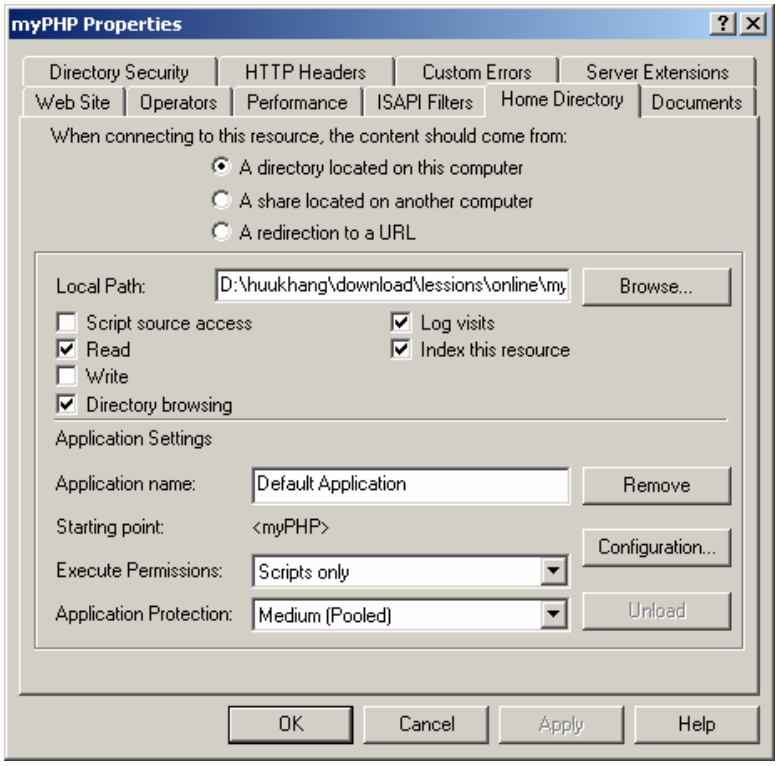
Hình 1-4: Quyền truy cập

10. Chọn Next và Finish, trong cửa sổ IIS xuất hiện ứng dụng có tên myPHP (khai báo trong phần diễn giải) như hình 1-5.



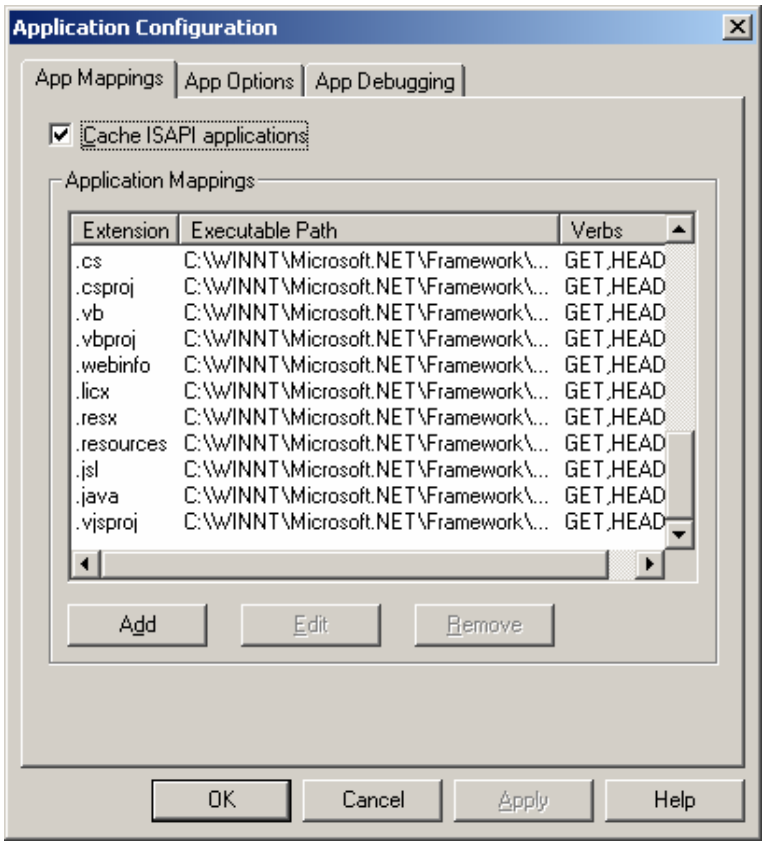
Hình 1-5: Tạo thành công ứng dụng PHP trong IIS

11.Sau khi tạo ứng dụng xong, bạn chọn tên ứng dụng myPHP | R-Click } Properties | cửa sổ xuất hiện như hình 1-5.



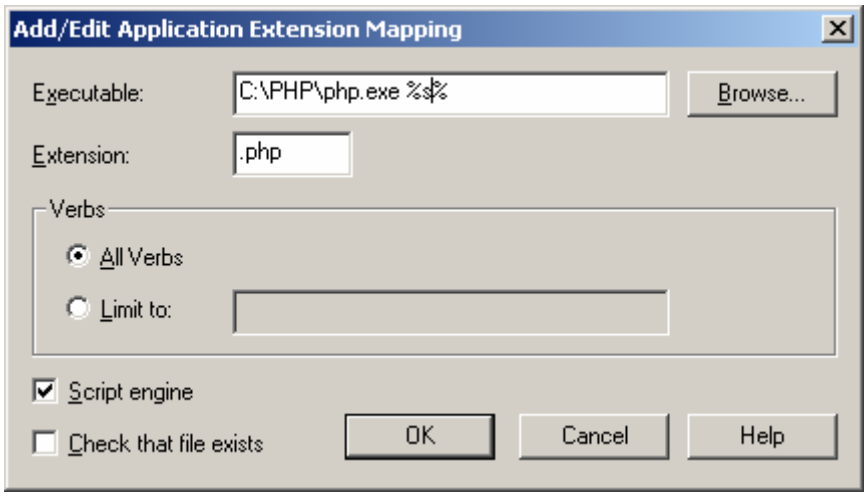
Hình 1-5: Cấu hình PHP trong IIS

12.Bằng cách chọn vào nút Configuration, cửa sổ sẽ xuất hiện như hình 1-6.



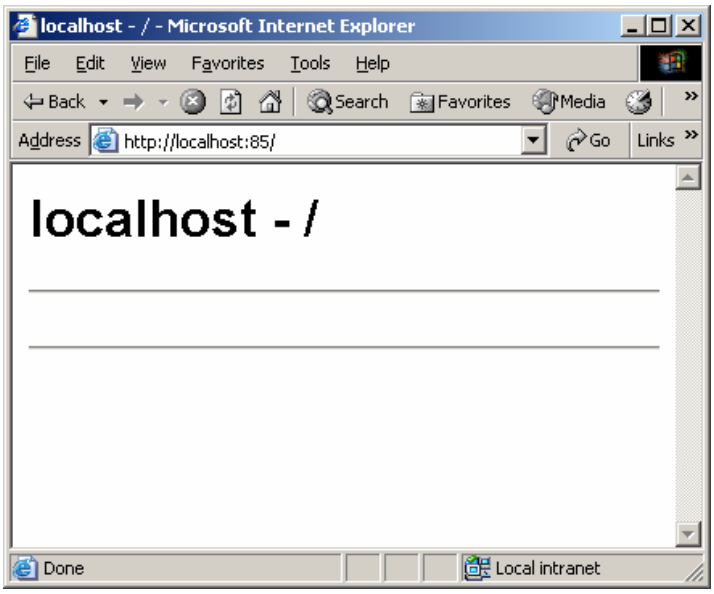
Hình 1-6: Thêm PHP Engine

13.Chọn nút Add, và khai báo như hình 1-7.



Hình 1-7: Khai báo PHP Engine

14.Để kiểm tra úng dụng, bạn mở cửa sổ IE và gõ trên thanh địa chỉ chuỗi như sau: http://localhost:85/ , kết quả xuất hiện như hình 1-8.



Hình 1-8: Ứng dụng PHP đã được khởi động

3.2. Cài đặt Apache Web Servr

Để cài đặt Apache Web Server, bạn theo các bước sau

1. Chep tap tin apache_1.3.22-win32-x86.exe xuong dia cung
2. Chay tap tin nay va cai dat len dia C:\Program Files\, sau khi ket thuc thanh cong phan cai dat Apache, bạn bắt đầu cấu hình ứng dụng PHP.
3. Chép ba dòng lệnh từ tập tin install.txt trong thư mục C:\PHP

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

4. Paste vào tập tin httpd.conf trong thư mục C:\Program Files\Apache Group\Apache\Conf\
5. Chon Start | Programs | Apache HTTP Server | Control Apache Server | Start
6. Viet trang test.php voi noi dung <?echo "hello";?>
7. Chep tap tin test.php vao thu muc C:\Program Files\Apache Group\Apache\htdocs\
8. Sau đó gõ trên trình duyệt http://localhost/test.php

4. GIỚI THIỆU PHP

4.1. Yêu cầu

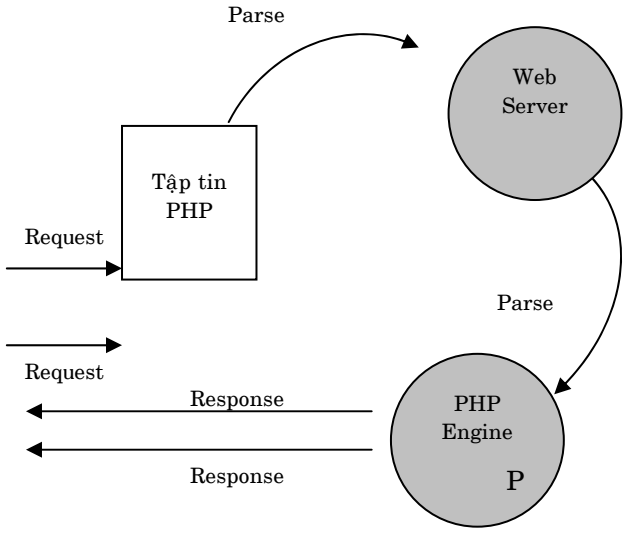
PHP dựa trên cú pháp của ngôn ngữ lập trình C, chính vì vậy khi làm việc với PHP bạn phải là người có kiến thức về ngôn ngữ C, C++, Visual C. Nếu bạn xây dựng ứng dụng PHP có kết nối cơ sở dữ liệu thì kiến thức về cơ sở dữ liệu MySQL, SQL Server hay Oracle là điều cần thiết.

4.2. Giới thiệu

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, ...

4.3. Thông dịch trang PHP

Khi người sử dụng gọi trang *PHP*, *Web Server* triệu gọi PHP Engine để thông dịch (tương tự như ASP 3.0 chỉ thông dịch chứ không phải biên dịch) dịch trang *PHP* và trả về kết quả cho người sử dụng như hình 1-9.



Hình 1-9: Quá trình thông dịch trang PHP

4.4. Kịch bản (script)

Nội dung của PHP có thể khai báo lẫn lộn với HTML, chính vì vậy bạn sử dụng cặp dấu giá <?=trị/biểu thức/biến?> để khai báo mã PHP. Chẳng hạn, chúng ta khai báo:

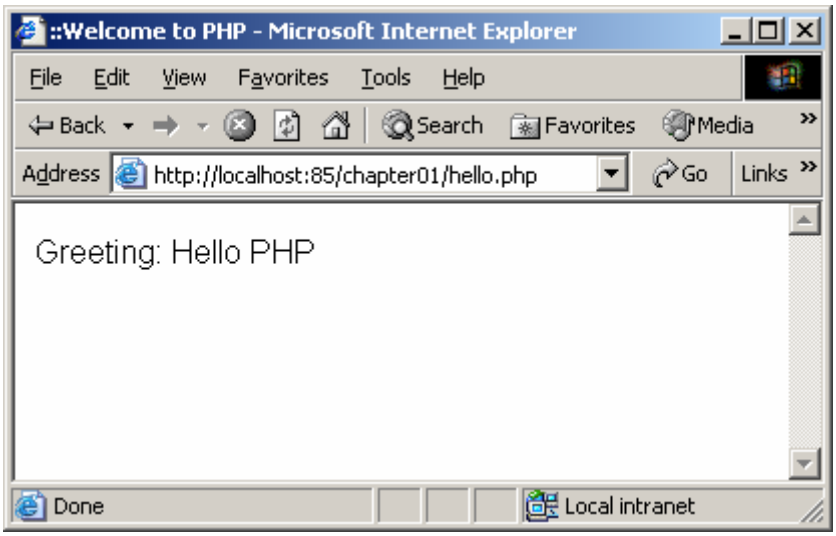
```
<br>
1-Giá trị biến Str: <?=$groupid?>
2-Giá trị biến i: <?=$i?>
3-Giá trị cũ thể: <?=10?>
```

Chẳng hạn bạn khai báo trang hello.php với nội dung như ví dụ 1-1 sau:

Ví dụ 1-1: Trang hello.php

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
  Greeting: <?="Hello PHP"?>
</BODY>
</HTML>
```

Kết quả trả về như hình 1-10 khi triệu gọi trang này trên trình duyệt.



Hình 1-10: Kết quả trang hello.php

Trong trường hợp có nhiều khai báo, bạn sử dụng Scriptlet, đều này có nghĩa là sử dụng cặp dấu trên như <?php Khai báo ?> với các khai báo PHP với cú pháp của C như sau:

```
<?php
     $sotrang=$pagenumber;
     $record=$rownumber;
     $totalRows = 0;
     $paging="";
?>
-Khai báo trên là Scriptlet
Giá trị của paging: <br>
<?= $paging ?>
-Khai báo này là Script
```

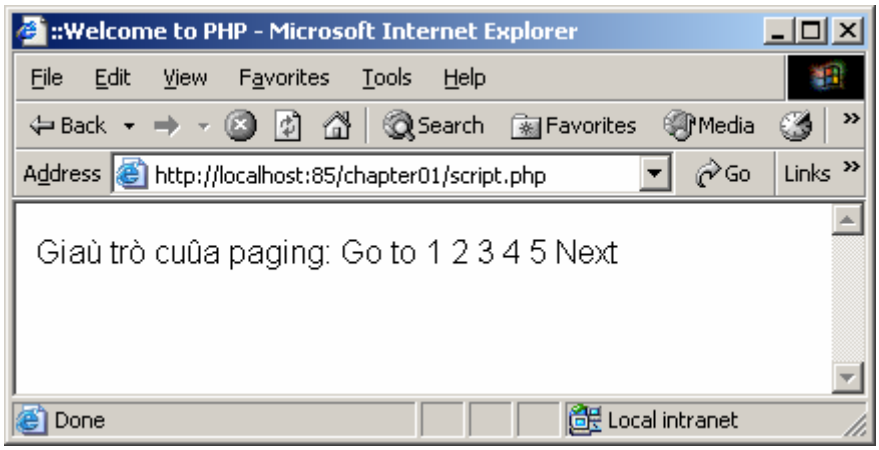
Lưu ý rằng, kết thúc mỗi câu lệnh phải dùng dấu ;

Ví dụ, bạn khai báo đoạn PHP trên trong tập tin script.php như ví dụ 1-2

Ví dụ 1-2: Trang script.php

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
     <?php
     $sotrang=$pagenumber;
     $record=$rownumber;
     $totalRows = 0;
     $paging="Go to 1 2 3 4 5 Next";
     ?>
     Giá trị của paging: <?= $paging ?>
</BODY>
</HTML>
```

Kết quả trả về như hình 1-11 khi triệu gọi trang này trên trình duyệt.



Hình 1-11: Kết quả trang hello.php

Lưu ý rằng, nếu bạn muốn sử dụng script hay scriptlet như ASP thì bạn khai báo trong tập tin php.ini như sau:

```
asp_tags = On
; Allow ASP-style <% %> tags. mặc định là Off
```

Khi đó trong trang PHP, thay vì bạn khai báo

```
<?php
   $sotrang=$pagenumber;
   $record=$rownumber;
   $totalRows = 0;
   $paging="";
?>
```

Thì bạn có thể khai báo như sau:

```
<%
   $sotrang=$pagenumber;
   $record=$rownumber;
   $totalRows = 0;
   $paging="";
%>
```

4.5. Ghi chú trong PHP

Ghi chú trong kịch bản PHP tương tự ngôn ngữ lập trình C, để ghi chú một dòng thì bạn sử dụng cặp dấu /. Chẳng hạn khai báo sau là ghi chú:

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

Trong trường hợp có nhiều dòng cần ghi chú bạn sử dụng cặp dấu /* và */, ví dụ khai báo ghi chú như sau:

```
/*
Khai báo biến để đọc dữ liệu
trong đó totalRows là biến trả
về tổng số mẩu tin
*/
$result = mysql_query($stSQL, $link);
$totalRows=mysql_num_rows($result);
```

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="";
?>
```

4.6. In kết quả trên trang PHP

Khác vớ các kịch bản như ASP, JSP, Perl, đối với PHP để in ra giá trị từ biến, biểu thức, hàm, giá trị cụ thể thỉ bạn có thể sử dụng script như trên:

```
Giá trị của paging: <%= $paging %>
```

Tuy nhiên, để sử dụng cú pháp của PHP khi in ra giá trị từ biến, biểu thức, hàm, giá trị cụ thể thì sử dụng khai báo echo như sau:

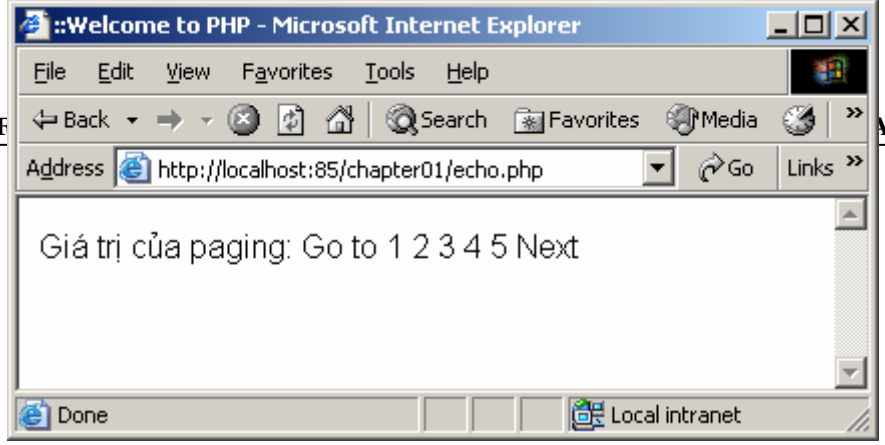
```
<?php
            $stSQLs=”select * from Customers”;
            echo $stSQLs;
?>
```

Chẳng hạn, khai báo echo như ví dụ 1-3.

Ví dụ 1-2: Trang echo.php

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
    <?php
    $sotrang=$pagenumber;
    $record=$rownumber;
    $totalRows = 0;
    $paging="Go to 1 2 3 4 5 Next";
    /*dùng phát biểu echo */
    echo “Giá trị của paging: “;
    echo $paging;
    ?>
</BODY>
</HTML>
```

Kết quả trả về như hình 1-12 khi triệu gọi trang này trên trình duyệt.



Hình 1-11: Kết quả trang hello.php

5. KẾT LUẬN

Trong bài này, chúng ta tập trung tìm hiểu cách cài đặt PHP và Apache Web Server, sau đó cấu hình ứng dụng PHP trong IIS hay sử dụng cấu hình mặc định của chúng.

Ngoài ra, bạn làm quen cách khai báo mã PHP trong trang .php cùng với script hay scriptlet.

Môn học: PHP

Bài 2

Bài học này chúng ta sẽ làm quen và tìm hiểu cú pháp và một số phương thức cơ bản của PHP:

- ✓ *Câu lệnh.*
- ✓ *Kiểu dữ liệu và biến*
- ✓ *Khai báo và sử dụng hằng.*
- ✓ *Dữ liệu mảng*
- ✓ *Chuyển đổi kiểu dữ liệu*

1. KHÁI NIỆM VỀ CÚ PHÁP PHP

Cú pháp PHP chính là cú pháp trong ngôn ngữ C, các bạn làm quen với ngôn ngữ C thì có lợi thế trong lập trình PHP.

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 dấu // 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

2. KHAI BÁO BIẾN

Khi thực hiện khai báo biến trong C, bạn cần phải biết tuân thủ quy định như: kiểu dữ liệu trước tên biến và có giá trị khởi đầu, tuy nhiên khi làm việc với PHP thì không cần khai báo kiểu dữ liệu nhưng sử dụng tiền tố $ trước biến.

Xuất phát từ những điều ở trên, khai báo biến trong PHP như sau:

- ❖ $variablename [=initial value];

```
$licount=0;
$lsSQL="Select * from tblusers where active=1";
$nameTypes  = array("first", "last", "company");
$checkerror=false;
```

- ❖ Chẳng hạn, khai báo như ví dụ 2-1 (variables.php)

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
```

```
<BODY>
<h4>Variable</h4>
<?php
   $sotrang=10;
   $record=5;
   $check = true;
   $strSQL="select * from tblCustomers";
   $myarr = array("first", "last", "company");
   $myarrs[2];
   $myarrs[0]="Number 0";
   $myarrs[1]="Number 1";
   $myarrs[2]="Number 2";
   echo $myarr[1];echo "<br>";
   echo $myarrs[2];
?>
</BODY>
</HTML>
```

3. KIỂU DỮ LIỆU

Bảng các kiểu dữ liệu thông thường

Boolean	True hay false
Integer	giá trị lớn nhất xấp xỉ 2 tỷ
Float	~1.8e308 gồm 14 số lẽ
String	Lưu chuỗi ký tự chiều dài vô hạn
Object	Kiểu đối tượng
Array	Mảng với nhiều kiểu dữ liệu

3.1. Thay đổi kiểu dữ liệu

Để thay đổi kiểu dữ liệu, bạn có thể sử dụng cách ép kiểu như trong các ngôn ngữ lập trình C hay Java. Chẳng hạn, khai báo ép kiểu như ví dụ 2-2 (box.php):

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Variable</h4>
<?php
     $i="S10A";
     echo $i+10;
     echo "<br>";
     $i="10A";
     $j=(float)$i;
     $j+=10;
     echo $i;
     echo "<br>";
     echo $j;
     echo "<br>";
     $q=12;$p=5;
     echo "Amount: ".(float)$q/$p;
?>
</BODY>
</HTML>
```

Lưu ý rằng, PHP tự động nhận biết giá trị chuỗi đằng sau số sẽ không được chuyển sang kiểu dữ liệu số như trường hợp trên.

Ngoài ra, bạn có thể sử dụng hàm settype để chuyển đổi dữ liệu này sag dữ liệu khác, ví dụ chúng ta khai báo như ví dụ 2-3 (settype.php).

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Change DataType of Variable</h4>
<?php
     $var="12-ABC";
     $check=true;
     echo $var;
     echo "<br>";
     echo $check;
     echo "<br>";
     settype($var,"integer");
     echo $var;
     echo "<br>";
     settype($check,"string");
     echo $check;
?>
</BODY>
</HTML>
```

3.2. Kiểm tra kiểu dữ liệu của biến

Để kiểm tra kiểu dữ liệu của biến, bạn sử dụng các hàm như sau:

is_int để kiểm tra biến có kiểu integer, nếu biến có kiểu integer thì hàm sẽ trả về giá trị là true (1). Tương tự, bạn có thể sử dụng các hàm kiểm tra tương ứng với kiểu dữ liệu là is_array, is_bool, is_callable, is_double, is_float, is_int, is_integer, is_long, is_null, is_numeric, is_object, is_real, is_string. Chẳng hạn, bạn khai báo các hàm này như ví dụ 2-4 (check.php).

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Check DataType of Variable</h4>
<?php
     $sotrang=10;
     $record=5;
     $check = true;
     $strSQL="select * from tblCustomers";
     $myarr = array("first", "last", "company");
     $myarrs[2];
     $myarrs[0]="Number 0";
     $myarrs[1]="Number 1";
     $myarrs[2]="Number 2";
     echo is_array($myarr);
     echo "<br>";
     echo is_bool($record);
?>
</BODY>
```

```
</HTML>
```

3.3. Thay đổi kiểu dữ liệu biến

Khi khai báo biến và khởi tạo giá trị cho biến với kiểu dữ liệu, sau đó bạn muốn sử dụng giá trị của biến đó thành tên biến và có giá trị chính là giá trị của biến trước đó thì sử dụng cặp dấu $$. Ví dụ, biến $var có giá trị là "total", sau đó muốn sử dụng biến là total thì khai báo như ví dụ 2-5 (change.php).

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Change DataType of Variable</h4>
<?php
     $var="total";
     echo $var;
     echo "<br>";
     $$var=10;
     echo $total;

?>
</BODY>
</HTML>
```

3.4. Kiểu Array

Kiểu mảng là một mảng số liệu do người dùng định nghĩa, chúng có cú pháp như sau:

```
$myarrs=array("first", "last", "company");

// mảng bao gồm các kiểu chuỗi
```

hay có thể khai báo như sau

```
$myarr[]=array(3);

$myarr[0]="Number 0";

$myarr[1]="Number 1";

$myarr[2]="Number 2";
```

Thứ tự index trong mảng bắt đầu từ vị trí 0. Chẳng hạn, bạn khai báo mảng một chiều theo hai cách trên như ví dụ 2-6 (array.php).

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Array on demenssion</h4>
<?php
     $myarr[]=array(3);
     $myarr[0]="Number 0";
     $myarr[1]="Number 1";
     $myarr[2]="Number 2";
     echo $myarr[0];
     echo $myarr[1];
```

```
      echo $myarr[2];
      echo "<br>";
      $myarrs=array("first", "last", "company");
      echo $myarrs[2];
?>
</BODY>
</HTML>
```

Nếu như bạn khai báo mảng hai chiều, thì cú pháp khai báo như sau:

```
$myarrs[][]=array(2,3);
```

Chẳng hạn khai báo như ví dụ 2-7 (arrays.php):

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Array two demenssions</h4>
<?php
   $myarrs[][]=array(2,3);
   $myarrs[0][0]="Number 00";
   $myarrs[1][0]="Number 10";
   $myarrs[0][1]="Number 01";
   $myarrs[1][1]="Number 11";
   $myarrs[0][2]="Number 02";
   $myarrs[1][2]="Number 13";
   echo $myarrs[0][2];
   echo "<br>";
?>
</BODY>
</HTML>
```

3.5. Kiểu đối tượng

Để khai báo đối tượng, bạn sử dụng khái niệm class như trong ngôn ngữ lập trình C hay java, ngoài ra phương thức trong PHP được biết đến như một hàm. Điều này có nghĩa là từ khoá là function.

Nếu hàm có tên trùng với tên của class thì hàm đó được gọi là constructor. Chẳng hạn, chúng ta khai báo class và khởi tạo chúng thì tự động constructor được gọi mỗi khi đối tượng khởi tạo, sau đó gọi hàm trong class đó như ví dụ 2-8 (object.php).

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Object</h4>
<?php
class clsA
{
  function clsA()
  {
    echo "I am the constructor of A.<br />\n";
  }
  function B()
```

```
  {
    echo "I am a regular function named B in class A.<br />\n";
    echo "I am not a constructor in A.<br />\n";
  }
}
// Gọi phương thức clsA() như constructor.
$b = new clsA();
echo "<br>";
// Gọi phương thức B().
$b->B();
?>
</BODY>
</HTML>
```

3.6. Tầm vực của biến

Tầm vực của biến phụ thuộc vào nơi khai báo biến, nếu biến khai báo bên ngoài hàm thì sẽ có tầm vực trong trang PHP, trong trường hợp biến khai báo trong hàm thì chỉ có hiệu lực trong hàm đó.

Ví dụ, chúng ta có biến $a khai báo bên ngoài hàm nhưng khi vào trong hàm thì biến $ được khai báo lại, biến này cótầm vựec bên trong hàm. Tương tự như vậy, khi biến $i khai báo trong hàm thì chỉ có tầm vực bên trong hàm cho dù chúng được khai báo lại bên ngoài như ví dụ 2-9 (scope.php).

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Scope of Variable</h4>
<?php
$a = 100;
/* global scope */
function Test()
{
     $i=10;
     $a=10;
     echo "<br>a:=$a";
     echo "<br>i:=$i";
     /* reference to local scope variable */
}
Test();
echo "<br>a:=$a";
$i=1000;
echo "<br>i:=$i";
?>
</BODY>
</HTML>
```

Ngoài ra, để sử dụng biến toàn cục trong hàm, bạn sử dụng từ khoa global, khi đó biến toàn cục sẽ có hiệu lực bên trong hàm. Ví dụ khai báo biến $a bên ngoài hàm, sau đó bên trong hàm Test bạn sử dụng từ khoá global cho biến $a, khi đó biến $a sẽ được sử dụng và giá trị đó có hiệu lực sau khi ra khỏi hàm chứ không gống như trường hợp trong ví dụ scope.php như ví dụ 2-10 (global.php).

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
```

```
</HEAD>
<BODY>
<h4>Scope of Variable</h4>
<?php
$a = 100;
/* global scope */
function Test()
{
     global $a;
     $i=10;
     $a+=10;
     echo "<br>a:=$a";
     echo "<br>i:=$i";
     /* reference to local scope variable */
}
Test();
echo "<br>a:=$a";
$i=1000;
echo "<br>i:=$i";
?>
</BODY>
</HTML>
```

4. HẰNG TRONG PHP

4.1. 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("MAXSIZE", 100);
```

Để sử dụng hằng, bạn khai báo như ví dụ 2-11 (constant.php)

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Constant</h4>
<?php
   define("pi",3.14);
   function Test()
   {
            echo "<br>pi:=".pi;
            echo "<br>pi:=".constant("pi");
   }
   Test();
   echo "<br>pi:=".pi;
   echo "<br>pi:=".constant("pi");
?>
</BODY>
</HTML>
```

4.2. Kiểm tra hằng

Khi sử dụng hằng, mà hằng chưa tồn tại thì bạn sử dụng hàm defined như ví dụ 2-12 sau (defained.php):

```
<HTML>
```

```
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Constant</h4>
<?php
define("pi",3.14);
//define("hrs",8);
function Test()
{
     if(defined("pi"))
       echo "<br>pi:=".pi;
     else
       echo "<br>pi not defined";
     if(defined("hrs"))
       echo "<br>hrs:=".hrs;
     else
       echo "<br>hrs not defined";
}
Test();
?>
</BODY>
</HTML>
```

5. KẾT LUẬN

Trong bài này, chúng ta tìm hiểu cách khai báo hằng, biến và sử dụng hằng biến. Ngoài ra, bạn cũng tìm hiểu cách chuyển đổi kiểu dữ liệu, kiểm tra kiểu dữ liệu, tầm vựec của biến.

Bài 3

PHÉP TOÁN VÀ PHÁT BIỂU CÓ ĐIỀU KIỆN TRONG PHP

Chương này chúng ta sẽ làm quen và tìm hiểu toán tử, phát biểu có điều kiện và vòng lặp của PHP.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ *Toán tử.*
- ✓ *Phép gán trong PHP*
- ✓ *Phát biểu có điều khiển.*
- ✓ *Vòng lặp.*

1. KHÁI NIỆM VỀ CÁC TOÁN TỬ TRONG PHP

Khi bạn lập trình trên PHP là sử dụng cú pháp của ngôn ngữ C, C++. Tương tự như những ngôn ngữ lập trình khác, toán tử giúp cho bạn thực hiện những phép toán như số học hay trên chuỗi.

Bảng sau đây giúp cho bạn hình dung được những toán tử sử dụng trong PHP, PHP định nghĩa toá tử toán học, quan hệ, số học, bit và nột số phép toán gán.

Loại toán tử	Toán tử	Diễn giải	Ví dụ
Arithmetic	+	Addition	a + b
	-	Subtraction	a - b
	*	Multiplication	a * b
	/	Division	a / b
	%	Modulus	a % b
Relational	>	Greater than	a > b
	<	Less than	a < b
	>=	Greater than or equal	a >= b
	<=	Less than or equal	a <= b
	!=	Not equal	a != b
	==	Equal	a == b
Logical	!	Not	!a
	&&	AND	a && b
	||	OR	a || b

Assignment	= ++ -- += -= *= /= %= |= &= ^= .=	Increment and assign Decrement and assign Add and assign Subtract and assign Multiply and assign Divide and assign Take modulus and assign OR and assign AND and assign XOR and assign Concat and assign	a = b a++ a-- a += b a -= b a *= b a /= b a %= b a |= b a &= b a ^= b a .= b
Allocation	new	Create a new object of a class	new A()
Selection	? :	If...Then selection	a ? b : c

2. GIỚI THIỆU TOÁN TỬ

Khi nói đến toán tử, chúng ta luôn liên tưởng đến thứ tự xử lý, cũng như trong toán học, toán tử trong PHP cũng co độ ưu tiên add-subtract-multi-divide.

2.1. Toán tử AND

Khi thực hiện một việc tăng lên giá trị thì bạn sử dụng cú pháp như sau:

$ i=0;$j=0;

j=i++;// i tăng sau khi gán i vào j, chính vì vậy sau khi gán i vào j, j vẫn không thay đổi

j=++i;// i tăng trước khi gán i vào j, chính vì vậy sau khi gán i vào j, j thay đổi.

Ví dụ 3.1: Phép toán AND.

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>AND Operator</h4>
<?php
```

```
  $i=10;
  $j=5;
  $j+=$i++;
  echo "j=$j";
  echo "<br>";
  echo "i=$i";
  echo "<br>";
  $j+=++$i;
  echo "j=$j";echo "<br>";
?>
</BODY>
</HTML>
```

2.2. Toán tử Not: ~ And !

Toán tử ~ đảo nghịch tất cả các bit của tham số, còn toán tử ! đảo nghịch giá trị của giá trị trước đó. Chẳng hạn trong trường hợp này chúng ta sử dụng cho biểu thức hay biến có giá trị boolean.

Ví dụ 3.2: Phép toán ~ and !

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>~, ! Operator</h4>
<?php
  $i=10;
  $j=5;
  $j+=~$i;
  echo "j=$j";
  echo "<br>";
  $j+=~$i++;
  echo "i=$i";
  echo "<br>";
  $j+=++$i;
  echo "j=$j";
  echo "<br>";
?>
</BODY>
</HTML>
```

2.3. Toán tử nhân và chia: * and /

Bạn có thể tham khảo ví dụ sau

Ví dụ 3.3: Phép toán * và /, + và -

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
```

```
</HEAD>
<BODY>
<h4>Multi And Divide Operator</h4>
<?php
    $i=10;
    $j=5;
    echo $i/$j;
    echo "<br>";
    echo $i*$j;
?>
</BODY>
</HTML>
```

2.4. Toán tử modulus: %

Khi chia một số cho một số, bạn cần kết quả là số dư của phép chia đó thì dùng toán tử modulus

Ví dụ 3.4: Phép toán %

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Mod Operator</h4>
<?php
    $i=10;
    $j=7;
    echo $i%$j;
    echo "<br>";
?>
</BODY>
</HTML>
```

2.5. Toán tử quan hệ: >=,>,<,<=,==,!=

Khi cần so sánh kết quả giữa hai toán hạn với nhau, thông thường bạn nghĩ đến phép toán so sánh như là bằng, lớn hơn, nhỏ hơn, ví dụ sau diễn giải cho bạn các toán tử trên:

Ví dụ 3.5: Phép toán >,>=,<,<=,==,!=

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Comparation Operators</h4>
<?php
```

```
    $i=10;
    $j=9;
    echo $i<$j;
    echo "<br>";
    echo $i!=$j;
?>
</BODY>
</HTML>
```

2.6. Toán tử && và ||

&& là toá tử and trong só học, || là toán tử or trong số học. Hai toán tử này rất thường dùng trong khi lập trình trên PHP, ví dụ dưới đây diễn giải cho bạn đầy đủ hai toán tử này. Chú ý rằng khi sử dụng toán tử đều có kèm phát biểu có điều kiện.

Ví dụ 3.6: Phép toán && và ||

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Logic Operators</h4>
<?php
    $b=true;
    $j=3;
    if (($j>=3) && ($b!=true))
    {
      echo "result is true";
    }
    if(($j<3) || ($b==true))
      echo "result is false";
?>
</BODY>
</HTML>
```

2.7. Toán tử ?:

Toán tử này thay thế cho phát biểu có điều kiện if...else, khi bạn cần lấy kết quả theo điều kiện nào đó, nếu có thể không cần phát biểu if-else, thì hãy thay thế bằng toán tử ?:, cú pháp của chúng như sau:

str1=str2.equals(”khang”)?”Welcome to PHP”:”Good bye PHP”;

Ví dụ 3.7: Phép toán ?:

```
<HTML>
```

```
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Selection Operators</h4>
<?php
    $str1="Pham Huu Khang";
    $str2 ="Khang";
    $str1=(str1==str2)?"Welcome to PHP":"Good bye PHP";
    echo "result is ".$str1;
?>
</BODY>
</HTML>
```

3. PHÉP GÁN

Khi gán một giá trị hay biến vào một biến trong PHP, bạn phải dùng đến phép gán, nhưng trong PHP cũng giống như trong C thì có những phép gán được đơn giản hoá hay nói đúng hơn là chuẩn hoá để rút gọn lại trong khi viết.

3.1. Phép gán thông thường nhất như sau:

$j=i;

$str1 =” Hello!”;

$b=true;

3.2. Phép gán thêm một giá trị là 1

$k=0;

$k++;

3.3. Phép gán chuỗi

$strX="Hello";

$strX.=” world”;

$strX.=”ABCc”.$x;

3.4. Phép gán thêm một với chính nó giá trị

$k=0;$j=1;

$k+=$j;

tương tự như vậy chúng ta có $k*=2, nghĩa là $k=$k*2

4. PHÁT BIỂU CÓ ĐIỀU KIỆN

Các phat biểu có điều kiện như :

- IF (điều kiện) { câu lệnh; }
- IF (điều kiện) { câu lệnh; }ELSE { câu lệnh; }

- IF (điều kiện) { câu lệnh; }ELSEIF { câu lệnh; }
- switch (điều kiện)

```
{
   case Value1
        câu lệnh1;
        break;
   }
```

- While (điều kiện)
- Do - While (điều kiện)
- Break
- Continue

4.1. Phát biểu IF (điều kiện) { câu lệnh; }

Sử dụng phát biểu if để chọn lọc kết quả khi điều kiện đúng, ví dụ như sau:

Ví dụ 3.8: Phát biểu IF

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>IF Statement</h4>
<?php
    $b=true;
    $j=3;
    if(($j>=3) &&($b!=true))
      echo "result is true";
    if(($j<3) ||($b==true))
      echo "result is false";

?>
</BODY>
</HTML>
```

4.2. Phát biểu IF (điều kiện) { câu lệnh; }ELSE { câu lệnh; }

Sử dụng phát biểu if để chọn lọc kết quả khi điều kiện đúng, và xuất ra kết quả khi điều kiện sai, ví dụ như sau:

Ví dụ 3.9: Phát biểu IF - ELSE

```
<HTML>
<HEAD>
```

```
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>IF ELSE Statement</h4>
<?php
    $b=true;
    $j=3;
    if ($j>3)
      echo "result is true";
    else
    {
      $j++;
      echo "result is $j";
    }
?>
</BODY>
</HTML>
```

4.3. Phát biểu ELSEIF

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

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>ELSEIF Statement</h4>
<?php
    $b=true;
    $j=3;
    if ($j>3)
      echo "result is true";
    elseif ($j=0)
    {
      $j++;
      echo "result is $j";
    }
    else
    {
      $j--;
      echo "result is ". $j--;
    }
?>
</BODY>
</HTML>
```

4.4. Phát biểu Switch (điều kiện)

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(điều kiện)

{

case Value1

câu lệnh1;

break;

case Value2

câu lệnh2;

break;

…

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 3.10: Phát biểu Switch

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>SWITCH Statement</h4>
<?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;
?>
</BODY>
</HTML>
```

4.5. Phát biểu While(điều kiện)

Phát biểu while thực thi những câu lệnh trong while khi điều kiện có giá trị true.

Ví dụ 3.11: Phát biểu While

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>While Statement</h4>
<?php
    $j=10;
    while($j>0)
    {
      echo $j."<br>";
      $j--;
    }
?>
</BODY>
</HTML>
```

4.6. Phát biểu For

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

Ví dụ 3.12: Phát biểu For

```
<HTML>
<HEAD>
```

```
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>FOR Statement</h4>
<?php
    for($j=1;$j<=10;$j++)
    {
      echo $j."<br>";
    }
?>
</BODY>
</HTML>
```

4.7. 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.

Ví dụ 3.13: Phát biểu Do While

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Do While Statement</h4>
<?php
    $j=10;
    do
    {
      echo $j."<br>";
      $j--;
    }while($j>0)
?>
</BODY>
</HTML>
```

Phát biểu exit cho phép thóat ra khỏi phát biểu điều kiện khi thoả điều kiện nào đó.

Ví dụ 3.14: Phát biểu exit

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Exit Statement</h4>
<?php
    $j=10;
    do
```

```
        {
          if($j==3) exit;
          echo $j."<br>";
          $j--;
        }while($j>0)
    ?>
    </BODY>
    </HTML>
```

5. TÓM TẮT

Trong bài học này chúng tôi giới thiệu đến cho các bạn các phép gán, các toán tử, đồng thời giúp cho các bạn hiểu thêm vào các phát biểu có điều kiện như while, for, switch, ….

Môn học: PHP

Bài 4

*Bài học này chúng ta sẽ làm quen với biến form và hai phương thức **$HTTP_POST_VARS** và **$HTTP_GET_VARS** của PHP:*

- ✓ *Biến form.*
- ✓ *Phương thức $HTTP_GET_VARS*
- ✓ *Phương thức $HTTP_POST_VARS*

1. 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.

1.1. Biến form từ form được submit với phương thức POST

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.

Chẳng hạn, bạn khai báo báo thẻ form trong trang submit.php như ví dụ 4-1.

Ví dụ 4-1: Khai báo thẻ form

```
...
<form action=ex1-1.php method=post>
<tr>
  <td>Name</td><td>:<input type=text name=fullname></td>
</tr>
<tr><td>Gender</td>
  <td>:<input type=radio value=M name=gender> Male
  <input type=radio value=F name=gender> Female</td>
</tr>
<tr><td> </td>
  <td><input type=submit value=Submit></td>
</tr>
</form>
...
```

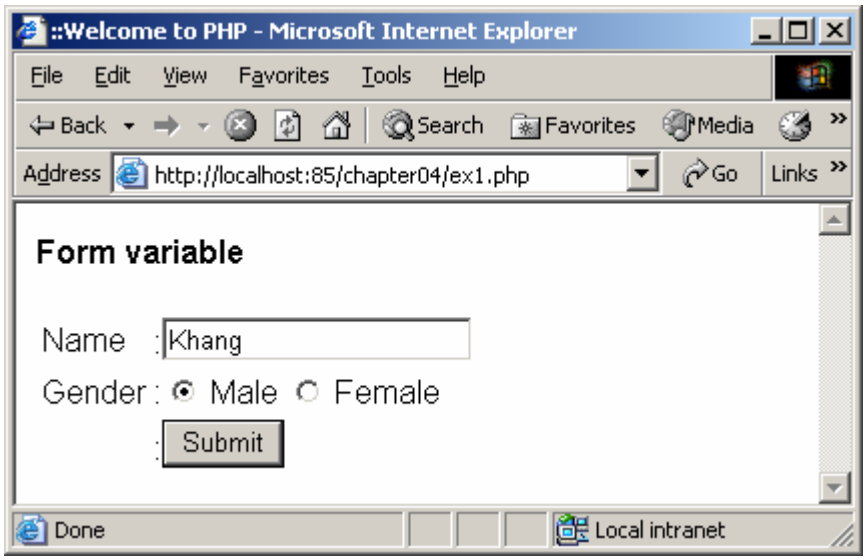
Khi người sử dụng nhập giá trị vào phần Name và chọn giới tính Male hay Female như hình 4-1, nếu nhấn nút submit thì trang ex1-1.php sẽ triệu gọi, trong trang này bạn có thể lấy giá trị nhập từ trang ex1.php bằng cách sử dụng biến form như ví dụ 4-1-1.

Ví dụ 4-2: Dùng biến form

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Form variable</h4>
<table>
<tr><td>Name</td>
<td>
  :<?=$fullname?>
```

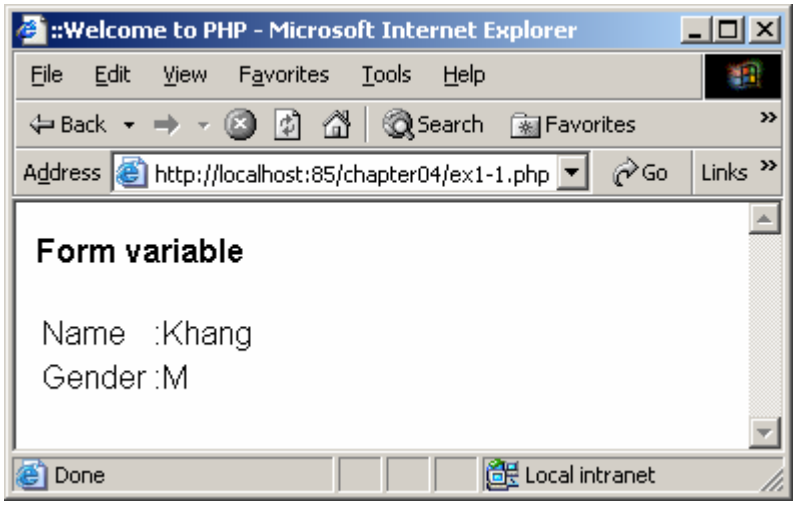
```
</td></tr>
<tr><td>Gender</td>
<td>
  :<?=$gender?>
</td></tr>
</table>
</BODY>
</HTML>
```

Trong đó, $fullname và $gender là tên của hai thẻ input trong trang ex1.php, trong trường hợp này chúng ta sử dụng phương thức POST cho form.



Hình 4-1: Nhập liệu

Kết quả trả về như hình 4-1-1.



Hình 4-1-1: Kết quả lấy từ trang submit bằng biến form

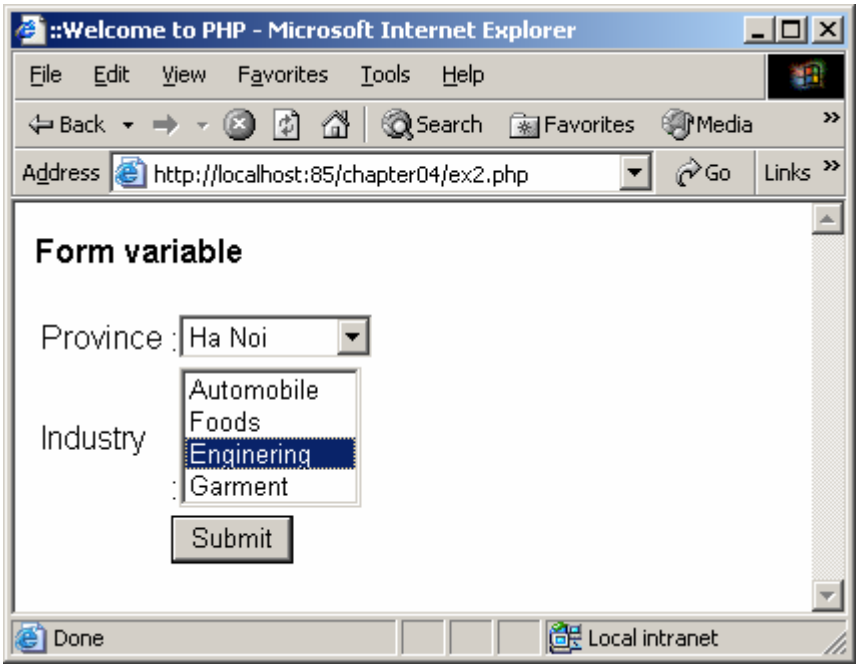
1.2. Biến form từ form được submit với phương thức GET

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 4-2 với phương thức GET trong thẻ form.

Ví dụ 4-2: Khai báo thẻ form

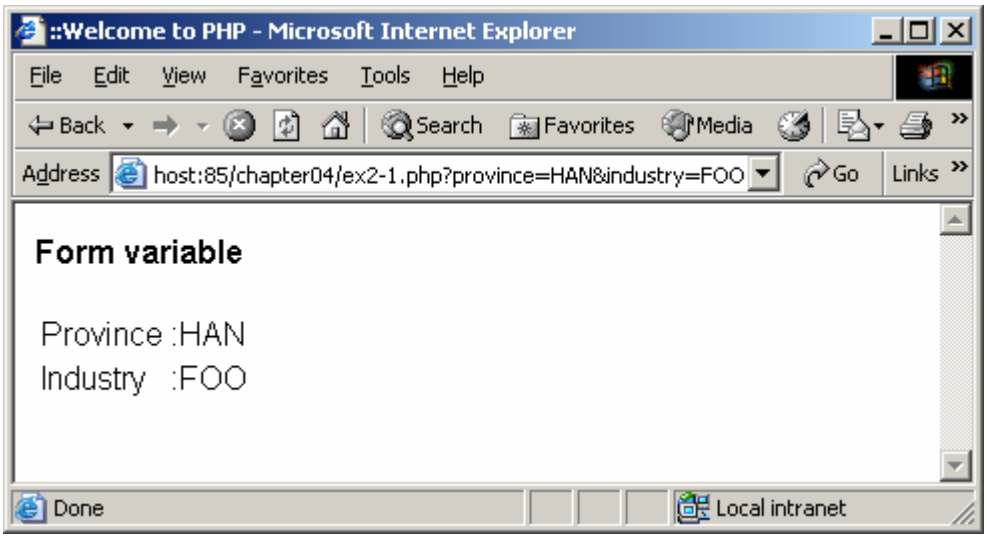
```
...
<form action=ex2-1.php method=get>
<tr><td>Province</td>
<td>
   :<select name=province>
   <option value=HAN>Ha Noi</option>
   <option value=HCM>Ho Chi Minh</option>
   <option value=HUE>Hue</option>
   </select>
</td></tr>
<tr><td>Industry</td>
<td>
   :<select name=industry multiple>
   <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>
</form>
...
```

Khi triệu gọi trang ex2.php trên trình duyệt, người sử dụng chọn giá trị trong hai tuỳ chọn Province và Industry như hình 4-2.



Hình 4-2: Phương thức GET

Nếu nhấn Submit thì hai giá trị chọn sẽ được truyền lên trên QueryString với hai tham số là tên của thẻ select. Ví dụ trong trường hợp này kết quả trả về như hình 4-2-1.



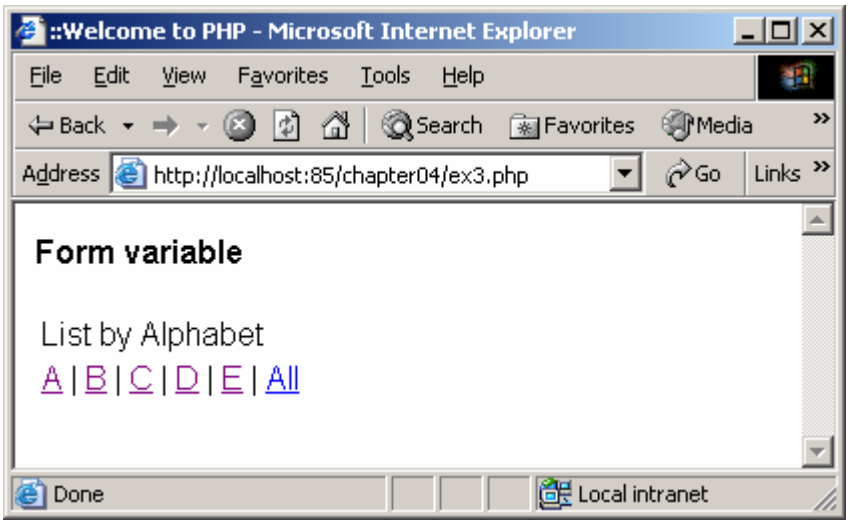
Hình 4-2-1: Biến form với phương thức GET

Trong đó, hai tham số và giá trị tương ứng là ex2-1.php?province=HAN&industry=FOO, bằng cách sử dụng biến form bạn có thể lấy được giá trị này như ví dụ 4-2-1.

Ví dụ 4-2-1: Khai báo thẻ form

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Form variable</h4>
<table>
<tr><td>Province</td>
<td>
  :<?=$province?>
</td></tr>
<tr><td>Industry</td>
<td>
  :<?=$industry?>
</td></tr>
</table>
</BODY>
</HTML>
```

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, bạn khai báo trang chop phép người sử dụng chọn ký tự để liệt kê danh sách khách hàng theo ký tự đó như hình 4-3.



Hình 4-3: Chọn ký tự

Bằng cách khai báo các thẻ <a> bạn định nghĩa 24 ký tự như hình trên với tham số al có giá trị tương ứng:

```
<tr><td>
<a href="ex3.php?al=A">A</a> |
<a href="ex3.php?al=B">B</a> |
<a href="ex3.php?al=C">C</a> |
<a href="ex3.php?al=D">D</a> |
<a href="ex3.php?al=E">E</a> |
<a href="ex3.php?al=">All</a>
</td></tr>
```

Khi người sử dụng chọn một ký tự thì sử dụng biến form là tên của tham số (al), bạn có thể lấy được giá trị của ký tự đang chọn:

```
<tr><td>Select:<?=$al?></td></tr>
```

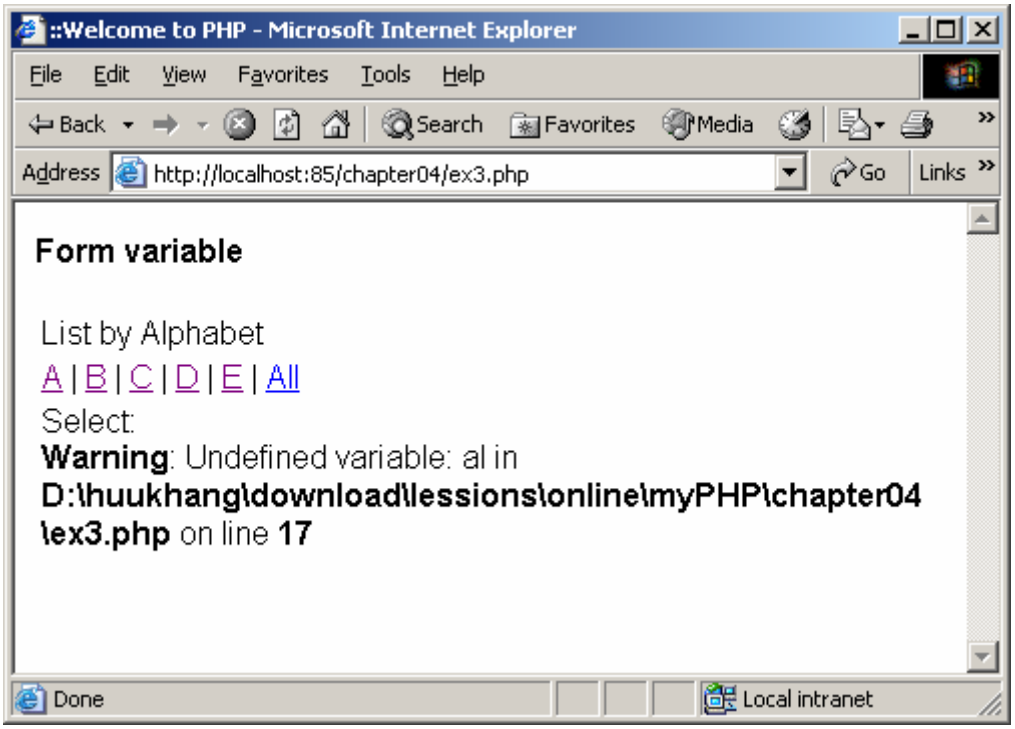
Tuy nhiên, lần đầu tiên triệu gọi trang này mà không có tham số trên QueryString, khai báo biến form sẽ phun ra lỗi như hình 4-3-1.

Để tránh trường hợp này, bạn sử dụng hàm isset để kiểm tra biến tồn tại hay không, nếu tồn tại thì bạn sử dụng biến form này. Ví dụ đối với trường hợp này chúng ta khai báo như ví dụ 4-3.

Ví dụ 4-3: Sử dụng biến form

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Form variable</h4>
<table>
<tr><td>List by Alphabet</td></tr>
<tr><td>
```

```
<a href="ex3.php?al=A">A</a> |
<a href="ex3.php?al=B">B</a> |
<a href="ex3.php?al=C">C</a> |
<a href="ex3.php?al=D">D</a> |
<a href="ex3.php?al=E">E</a> |
<a href="ex3.php?al=">All</a>
</td></tr>
<?php
if(isset($al))
  {
  ?>
     <tr><td>Select:<?=$al?></td></tr>
  <?php
  }
  ?>
</table>
</BODY>
</HTML>
```



Hình 4-3-1: Lỗi phát sinh

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.

2. PHƯƠNG THỨC $HTTP_GET_VARS

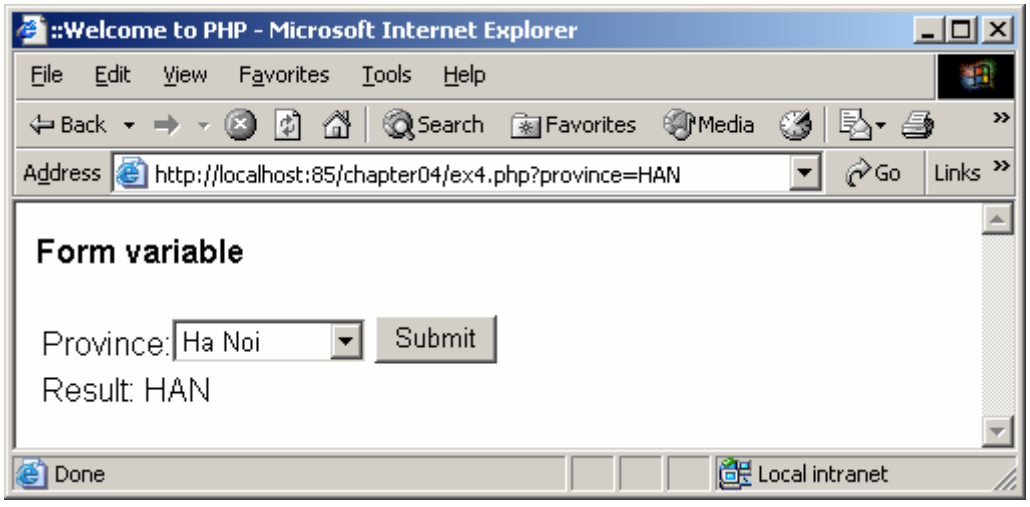
Ngoài cách sử dụng biến form trong trường hợp lấy giá trị từ tham số của QueryString, bạn có thể sử dụng hàm $HTTP_GET_VARS. Ví dụ, chúng ta khai báo trang PHP như ví dụ 4-4.

Ví dụ 4-4: Sử dụng $HTTP_GET_VARS

```
<HTML>
```

```
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Form variable</h4>
<table>
<form action=ex4.php method=get>
<tr><td>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></td></tr>
</form>
<tr><td>
<?php
     if(isset($HTTP_GET_VARS["province"]))
     {
      $result=$HTTP_GET_VARS["province"];
      echo "Result: ".$result;
     }
?>
</td></tr>
</table>
</BODY>
</HTML>
```

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 ex4.php mà không submit. Tuy nhiên, nếu bạn submit trang này thì kết quả trả về như hình 4-4.



Hình 4-4: Dùng $HTTP_GET_VARS

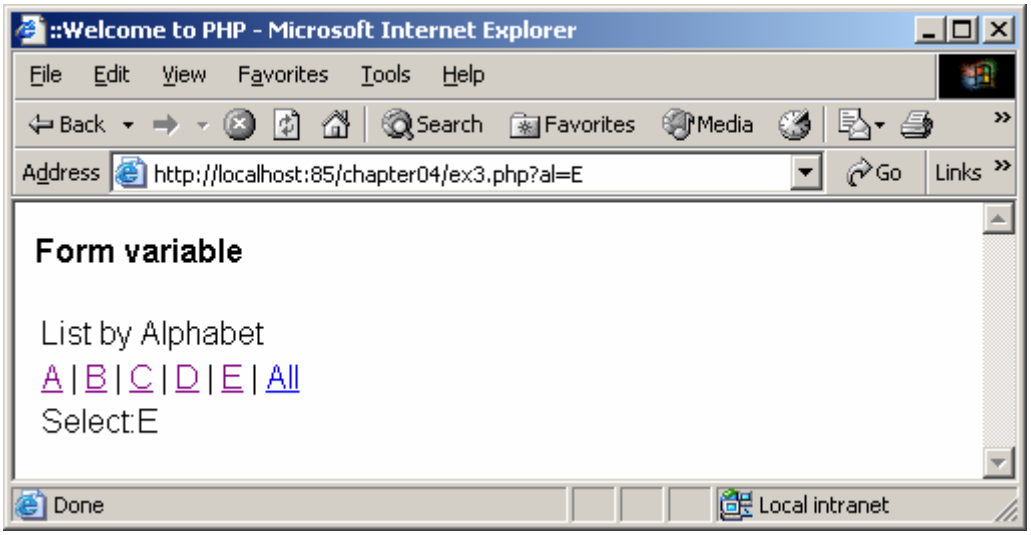
Tương tự như vậy trong trường hợp bạn không sử dụng thẻ form mà giá trị lấy từ chuỗi QueryString bằng cách sử dụng $HTTP_GET_VARS như ví dụ 4-5.

Ví dụ 4-5: Sử dụng $HTTP_GET_VARS

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
```

```
<BODY>
<h4>Form variable</h4>
<table>
<tr><td>List by Alphabet</td></tr>
<tr><td>
<a href="ex3.php?al=A">A</a> |
<a href="ex3.php?al=B">B</a> |
<a href="ex3.php?al=C">C</a> |
<a href="ex3.php?al=D">D</a> |
<a href="ex3.php?al=E">E</a> |
<a href="ex3.php?al=">All</a>
</td></tr>
<?php
if(isset($HTTP_GET_VARS["al"]))
     {
     ?>
     <tr><td>Select:<?=$HTTP_GET_VARS["al"]?></td></tr>
     <?php
     }
?>
</table>
</BODY>
</HTML>
```

Kết quả trả về như hình 4-5.



Hình 4-5: Sử dụng $HTTP_GET_VARS

3. PHƯƠNG THỨC $HTTP_POST_VARS

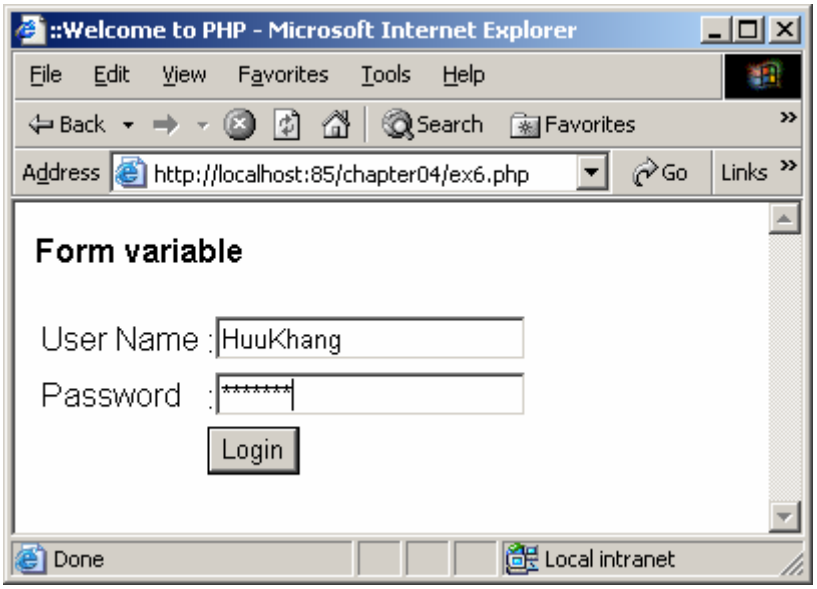
Tương tự như $HTTP_GET_VARS nhưng $HTTP_POST_VARS cho phép bạn lấy giá trị lấy từ các thẻ nhập liệu của thẻ form trong traang submit trước đó. Ví dụ, bạn khai báo trang nhập liệu như ví dụ 4-6.

Ví dụ 4-5: Khai báo form với phương thức POST

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
```

```
<BODY>
<h4>Form variable</h4>
<form action=ex7.php method=post>
<table>
<tr><td>User Name</td>
<td>
     :<input type=text name=username>
</td></tr>
<tr><td>Password</td>
<td>
     :<input type=password name=password>
</td></tr>
<tr><td> </td>
<td><input type=submit value=Login></td></tr>
</table>
</form>
</BODY>
</HTML>
```

Khi người sử dụng nhập username và password như hình 4-6 và nhấn nút Login.



Hình 4-6: Đăng nhập

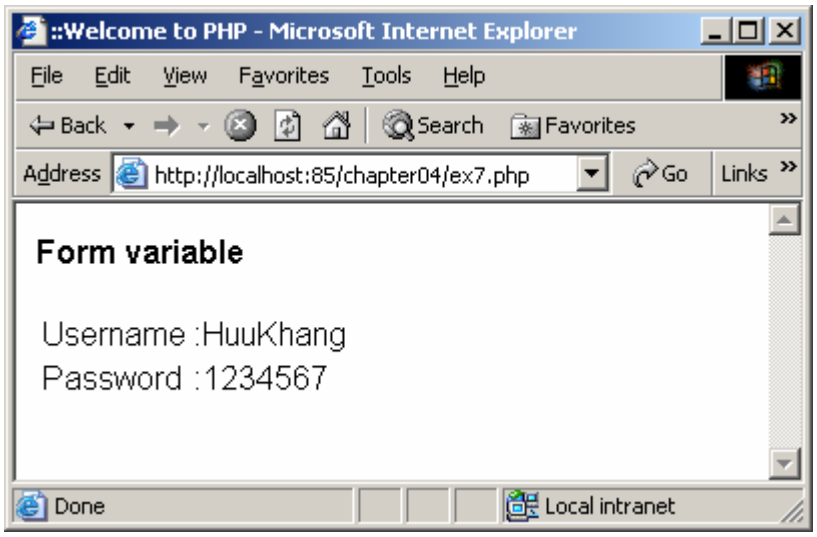
Bằng cách sử dụng $HTTP_POST_VARS để lấy giá trị username và password như ví dụ 4-7.

Ví dụ 4-5: Sử dụng $HTTP_POST_VARS

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Form variable</h4>
<?php
if(isset($HTTP_POST_VARS["username"]))
{
     ?>
```

```
        <table>
        <tr><td>Username</td>
        <td>:<?=$HTTP_POST_VARS["username"]?></td></tr>
        <tr><td>Password</td><td>
        :<?=$HTTP_POST_VARS["password"]?></td></tr>
        </table>
    <?php
    }
    ?>
    </BODY>
    </HTML>
```

Kết quả trình bày như hình 4-7.



Hình 4-7: Dùng $HTTP_POST_VARS

4. KẾT LUẬN

Trong bài này, chúng ta tìm hiểu cách sử dụng biến form và hai phương thức $HTTP_POST_VARS, $HTTP_GET_VARS. 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.

Môn học: PHP

Bài 5

Bài học này chúng ta sẽ làm quen với đối tượng Session và một số đối tượng khác:

- ✓ *Đối tượng Session.*
- ✓ *Đối tượng khác*

1. ĐỐI TƯỢNG SESSION

Trong PHP4.0 đối tượng Session được xem như một đối tượng cho phép bạn truyền giá trị từ trang PHP này sang PHP khác. Để sử dụng Session, bạn khai báo thư mục được lưu trữ dữ liệu do đối tượng nay ghi ra.

Session được sinh ra và được biến mất khi người sử dụng huỷ chúng, thời gian sống của chúng đã hết hoặc người sử dụng đóng trình duyệt.

Chẳng hạn, trong trường hợp này chúng ta sử dụng thư mục C:\PHP\sessiondata được khai báo trong tập tin php.ini.

```
session.save_path = C:\PHP\sessiondata
```

Ngoài ra, khi muốn sử dụng Sessoin thì bạn phải khởi tạo chúng. Để khởi tạo Session bạn có thể khởi tạo trong trang PHP mỗi khi truy cập hay gán giá trị cho Session.

```
session_start();
```

Tuy nhiên, bạn có thể cấu hình trong trang php.ini (1 là start).

```
session.auto_start = 0
```

1.1. Nhận dạng Session

Mỗi phiên làm việc được tạo ra từ Web Server thì sẽ có một nhận dạng duy nhất có giá trị là chuỗi do trình chủ Web tạo ra. Điều này có nghĩa là mỗi khi người sử dụng triệu gọi trang Web của Web Site lần đầu tiên thì phiên làm việc sẽ được tạo ra, khi đó một nhận dạng được cấp cho phiên làm việc đó.

Để lấn giá trị nhận dạng của Session do trình chủ Web cấp phát bạn sử dụng cú pháp:

```
$x= session_id();
```

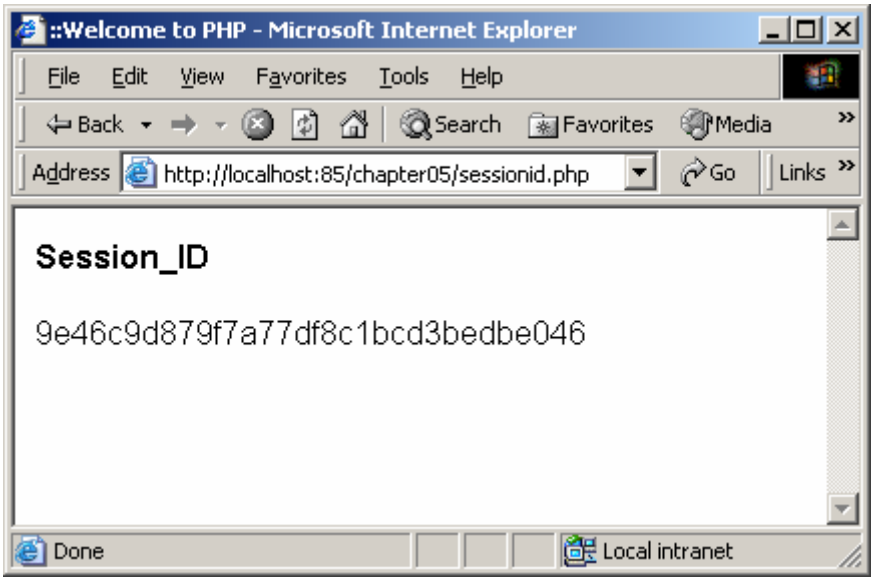
Chẳng hạn, bạn khai báo báo để lấy giá trị session_id trong trang sessionid.php như ví dụ 5-1.

Ví dụ 5-1: Nhận dạng session

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

```
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Session_ID</h4>
<?php
  $sessionid=session_id();
  echo $sessionid;
?>
</BODY>
</HTML>
```

Mỗi người sử dụng truy cập đến Web Site sẽ có một nhận dạng khác như như hình 5-1.



Hình 5-1: Nhận dạng duy nhất

1.2. Khai báo Session

Khi muốn khai báo biến session, bạn phải sử dụng hàm session_register có cú pháp như sau:

```
session_register("sessioname");
```

Khi muốn khởi tạo session, bạn có thể gán giá trị cho session này như gán giá trị cho biến trong PHP, sau đó sử dụng hàm trên để đắng ký.

```
$sessioname=value;
session_register("sessioname");
```

Trong trường hợp có nhiều session, bạn có thể sử dụng hàm session_register để đăng ký cùng một lúc nhiều session như sau:

```
$sessioname1=value1;
$sessioname2=value2;
```

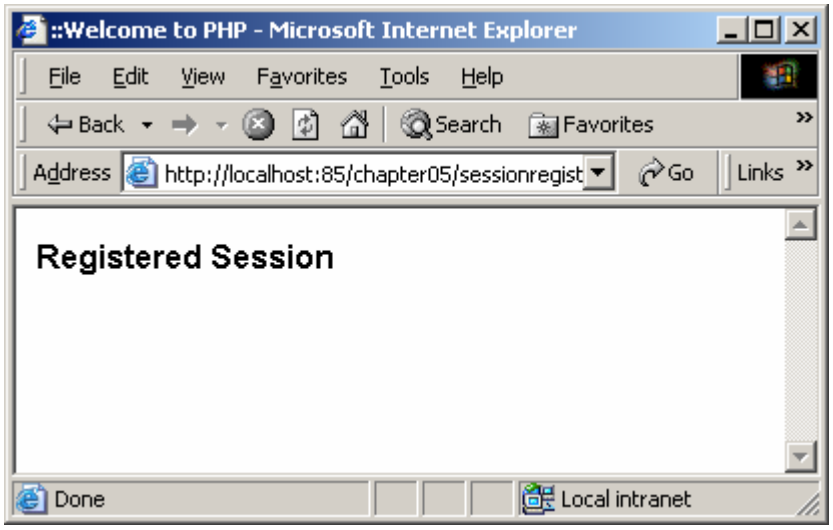
```
$sessioname3=value3;
session_register("sessioname1","sessioname2","sessioname3");
```

Chẳng hạn, trong trường hợp này chúng ta khai báo trang sessionregister.php và đăng ký 3 sessoin có tên userid, email và fullname như ví dụ 5-2 sau:

Ví dụ 5-2: Đăng ký session

```
<?php
  session_start();
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Registered Session</h4>
<?php
  $userid="123";
  $email="test@yahoo.com";
  $fullname="Nguyen Van Ba";
  session_register("userid");
  session_register("email","fullname");
?>
</BODY>
</HTML>
```

Kết quả trả về như hình 5-2.



Hình 5-2: Đăng ký Session

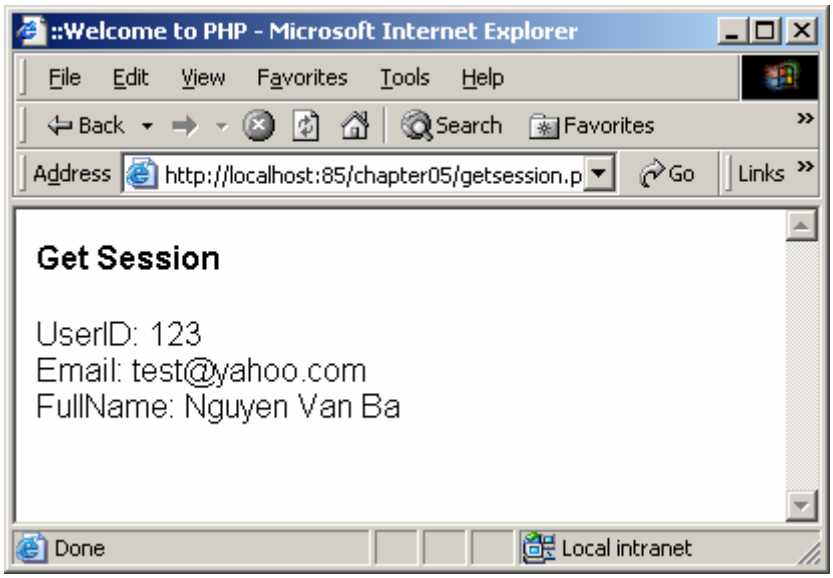
1.3. Lấy giá trị từ session

Sau khi khai báo khởi ạo một số session với giá trị tương ứng của session đó, bạn có thể truy cập các biến session này để lấy giá trị torng trang PHP khác. Chẳng hạn, chúng ta khai báo trang getsession.php để lấy các session của PHP vừa khai báo trong ví dụ trên như ví dụ 5-3.

Ví dụ 5-3: Lấy giá trị từ session

```
<?php
  session_start();
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Get Session</h4>
<?php
  echo "UserID: ". $userid."<br>";
  echo "Email: ".$email."<br>";
  echo "FullName: ".$fullname;
?>
</BODY>
</HTML>
```

Khi triệu gọi trang getsession.php trên trình duyệt bạn trình bày giá trị của session userid, email và fullname như hình 5-3.



Hình 5-3: Lấy giá trị của session

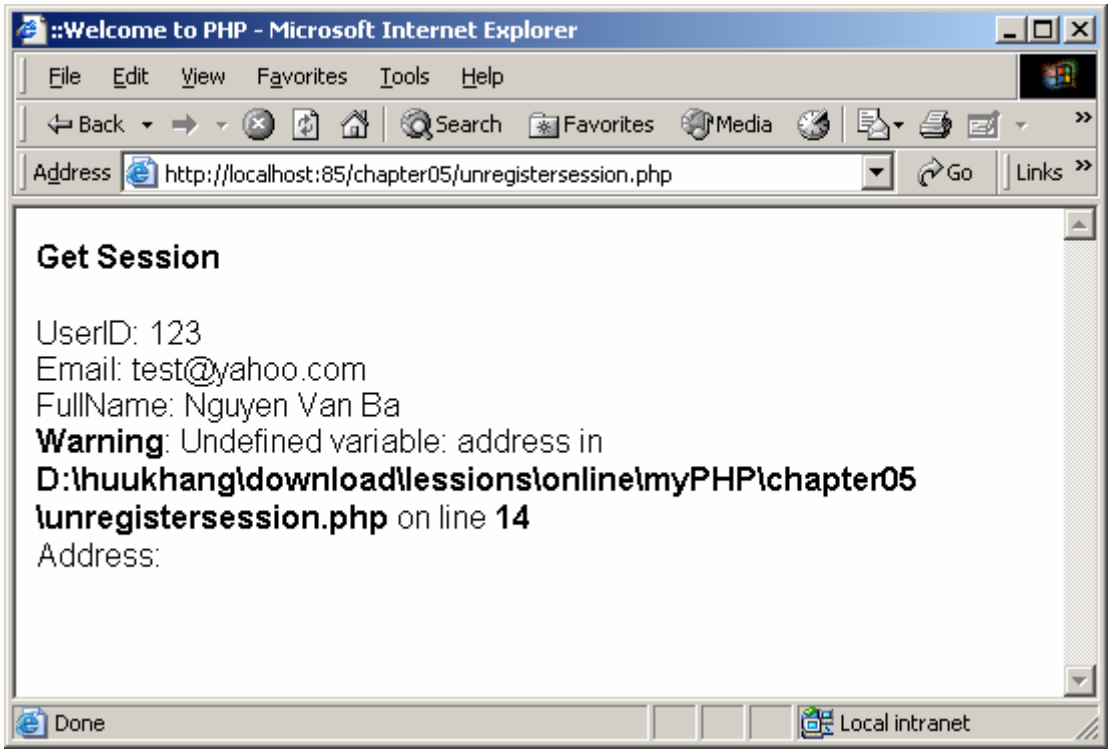
Tuy nhiên, trong trường hợp bạn truy cập một biến session chưa khởi tạo trước đó thì l64i sẽ phát sinh. Ví dụ trong trường hợp này chúng ta truy cập biến session có tên $address như ví dụ 5-4.

Ví dụ 5-4: Truy cập session chưa tồn tại

```
<?php
  session_start();
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
```

```
<h4>Get Session</h4>
<?php
   echo "UserID: ". $userid."<br>";
   echo "Email: ".$email."<br>";
   echo "FullName: ".$fullname;
   echo "Address: ".$address;
?>
</BODY>
</HTML>
```

Khi triệu gọi trang unregistersession.php trên trình duyệt thì lỗi phát sinh như hình 5-4.



Hình 5-4: Lỗi phát sinh

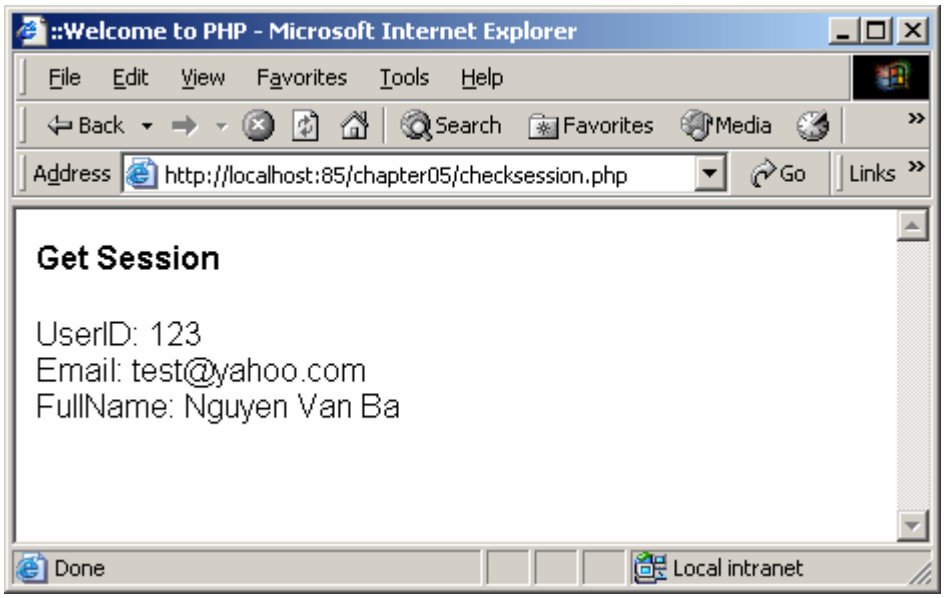
Để kiểm tra session đó có tồn tại hay chưa bạn sử dụng hàm session_is_register trong trang checksession.php. Đối với trường hợp này chúng ta cần kiểm tra 4 session trước khi truy cập đến chúng như ví dụ 5-5.

Ví dụ 5-5: Kiểm tra session

```
<?php
   session_start();
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Get Session</h4>
```

```
<?php
  if(session_is_registered("userid"))
   echo "UserID: ". $userid."<br>";
  if(session_is_registered("email"))
   echo "Email: ".$email."<br>";
  if(session_is_registered("fullname"))
   echo "FullName: ".$fullname;
  if(session_is_registered("address"))
   echo "Address: ".$address;
?>
</BODY>
</HTML>
```

Khi triệu gọi trang checksession.php thì kết quả sẽ trình bày như hình 5-5.



Hình 5-5: Không có lỗi phát sinh

Chú ý rằng, khi sử dụng đến session, bạn phai khởi động chúng bằng session_start() nếu không thì phải khai báo trong php.ini.

1.4. Huỷ session

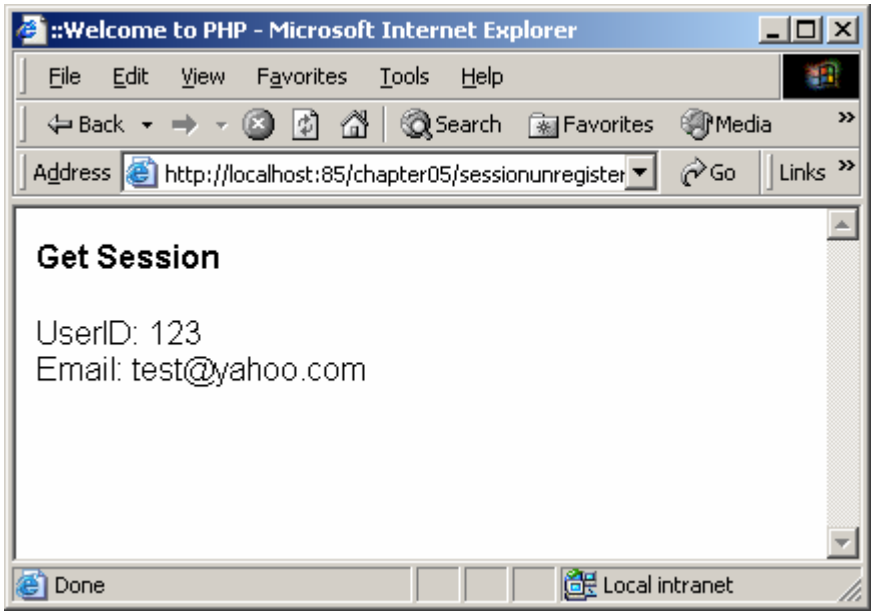
Khi không có nhu cầu sử dụng session nữa thì bạn sử dụng hàm session_unregister để loại session đó. Chẳng hạn, trong trường hợp này chúng ta muốn loại bỏ session có tên là fullname bạn khai báo trong trang sessionunregister.php như ví dụ 5-6.

Ví dụ 5-6: Loại bỏ một Session

```
<?php
  session_start();
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
```

```
<h4>UnRegister Session</h4>
<?php
   session_unregister("fullname");
   if(session_is_registered("userid"))
    echo "UserID: ". $userid."<br>";
   if(session_is_registered("email"))
    echo "Email: ".$email."<br>";
   if(session_is_registered("fullname"))
    echo "FullName: ".$fullname;
   if(session_is_registered("address"))
    echo "Address: ".$address;
?>
</BODY>
</HTML>
```

Khi triệu gọi trang sessionunregister.php trên trình duyệt thìkết quả trả về như hình 5-6.



Hình 5-6: Loại bỏ session

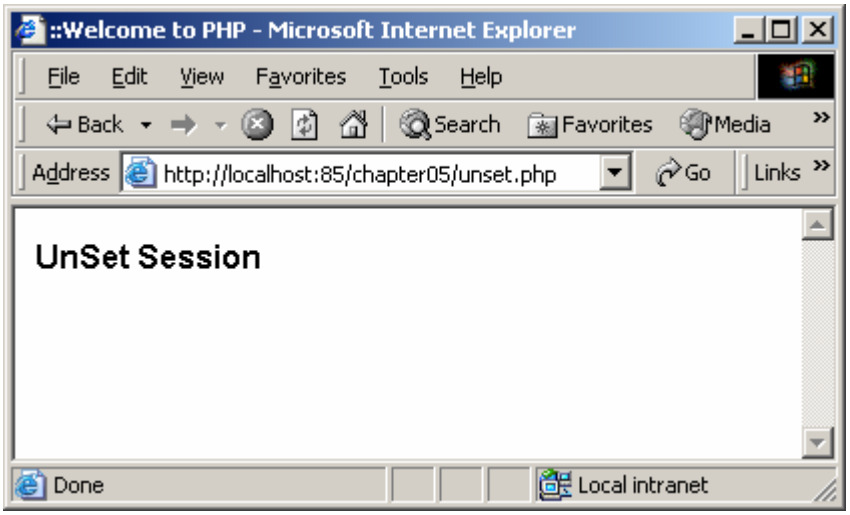
Trong trường hợp loại bỏ tất các session đang tồn tại thì sử dụng hàm session_unset(). Ví dụ dùng hàm này để lạoi bỏ session và dùng hàm sessin_destroy để huỷ tất cả session đó khai báo trong trang unset.php như ví dụ 5-7.

Ví dụ 5-7: Xoá tất cả session

```
<?php
    session_start();
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>UnSet Session</h4>
<?php
    session_unset();
    session_destroy();
    if(session_is_registered("userid"))
```

```
        echo "UserID: ". $userid."<br>";
       if(session_is_registered("email"))
        echo "Email: ".$email."<br>";
       if(session_is_registered("fullname"))
        echo "FullName: ".$fullname;
       if(session_is_registered("address"))
        echo "Address: ".$address;
?>
</BODY>
</HTML>
```

Kết quả trả về như hình 5-7.



Hình 5-7: Huỷ session

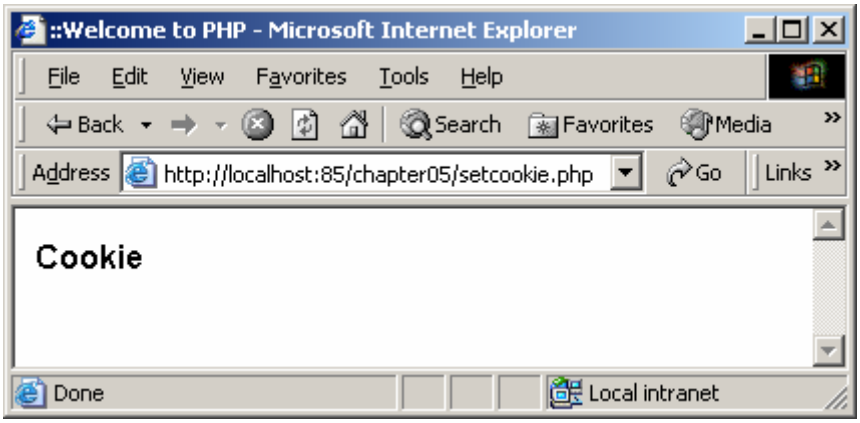
2. COOKIE

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 như ví dụ 5-8.

Ví dụ 5-8: Gán giá trị cho cookie

```
<?php
    setcookie("huukhang","Computer Learning Center");
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Cookie</h4>
</BODY>
</HTML>
```

Khi người sử dụng triệu gọi trang setcookie.php kết quả trả về như hình 5-8.



Hình 5-8: Đắng ký cookie

Ngài ra, bạn có thể gán giá trị cookie bằng session. Chẳng hạn, chúng ta sử dụng hàm session_set_cookie_params để gán cookie như ví dụ 5-9.

Ví dụ 5-9: Gán cookie bằng session

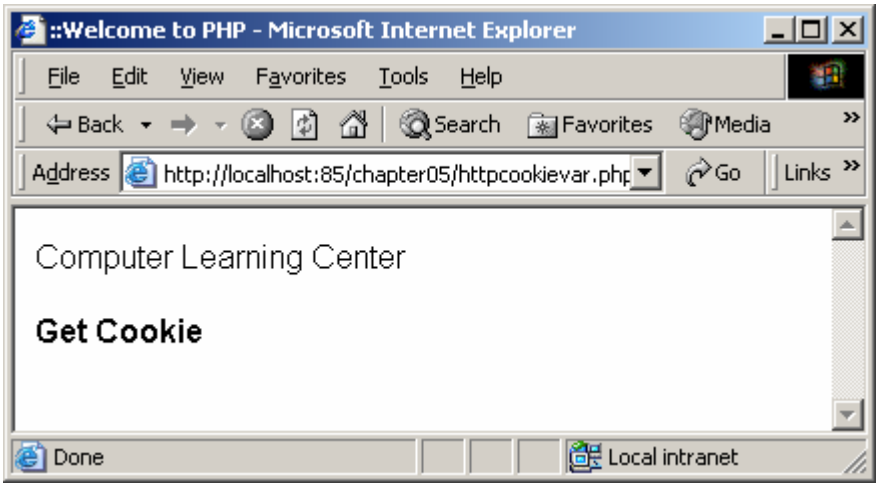
```
<?php
     session_start();
     $myvalue="Online Recruitment";
     session_set_cookie_params($myvalue);
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Session-Cookie</h4>
</BODY>
</HTML>
```

Bằng cách sử dụng $HTTP_COOKIE_VARS để lấy giá trị của cookie trước đó trong trang httpcookievar.php như ví dụ 5-10.

Ví dụ 5-10: Sử dụng $HTTP_COOKIE_VARS

```
<?php
     echo $HTTP_COOKIE_VARS["huukhang"];
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Get Cookie</h4>
</BODY>
</HTML>
```

Kết quả trình bày như hình 5-10.



Hình 5-10: Dùng $HTTP_COOKIE_VARS

Bằng cách sử dụng hàm session_get_cookie_params để lấy giá trị của cookie trước đó trong trang sessiongetcookie.php như ví dụ 5-11.

Ví dụ 5-11: Sử dụng session_get_cookie_params

```
<?php
     session_start();
     $myvalue= session_get_cookie_params();
     echo $myvalue[1];
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Get Cookie</h4>
</BODY>
</HTML>
```

3. KẾT LUẬN

Trong bài này, chúng ta tìm hiểu cách sử dụng biến session và cookie.

Môn học: PHP

Bài 6

Bài học này chúng ta sẽ làm quen cách khai báo hàm, chèn tập tin và tập tin dùng chung:

- ✓ *Cách khai báo hàm.*
- ✓ *Xây dựng tập tin định dạng nội dung*
- ✓ *Tập tin dùng chung*

1. KHAI BÁO HÀM TRONG PHP

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

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

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

Trong trường hợp hàm không có giá trị trả về thì hàm được xem như thủ tục. Ngoài ra, bạn có thể khai báo tham số tuỳ chọn bằng cách gán giá trị mặc định cho tham số. Ví dụ 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, bạn khai báo trang function.php có hàm getResult nhận hai số và phép toán sau đó tuỳ thuộc vào phép toán hàm trả về kết quả. Nếu người sử dụng không cung vấp phép toán thì mặc định là phép toán +.

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Function</h4>
<?php
     function getResult($number1, $number2,$operator="+")
     {
       $result=0;
       switch($operator)
       {
         case "+":
           $result=$number1+$number2;
           break;
         case "-":
```

```
            $result=$number1-$number2;
            break;
          case "*":
            $result=$number1*$number2;
            break;
          case "/":
            if($number2!=0)
                  $result=$number1/$number2;
            else
                  $result=0;
            break;
          case "%":
            if($number2!=0)
                  $result=$number1%$number2;
            else
                  $result=0;
            break;
        }
        return $result;
      }
      echo "result of default operator: ".getResult(10,20);
      echo "<br>";
      echo "result of * operator: ".getResult(10,20,"*");
?>
</BODY>
</HTML>
```

Nếu muốn định nghĩa function không có giá trị trả về, bạn có thể khai báo trong trang void.php như ví dụ sau:

```
...
      function calloperator()
      {
        echo "result of default operator: ".getResult(10,20);
        echo "<br>";
        echo "result of * operator: ".getResult(10,20,"*");
      }
            calloperator();
      ?>
</BODY>
</HTML>
```

Trong trường hợp truyền tham số như tham biến, bạn sử dụng ký hiệu & trước tham số, chẳng hạn chúng ta khai báo hàm có tham biến có tên average như trong trang reference.php như sau:

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Function</h4>
<?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;
?>
</BODY>
</HTML>
```

Trong trường hợp trên thì hàm getAmount có tham số $average là tham biến còn hàm getAmounts có tham số $average là tham trị, và kết quả trả về của biến $bq khi gọi hàm getAmount là 100 trong khi đó giá trị của biến này trong hàm getAmounts là 0.

2. XÂY DỰNG TẬP TIN ĐỊNH DẠNG NỘI DUNG

Khi trình bày nội dung trên trang *HTML* hay trang *PHP*, để thống nhất định dạng chuỗi trong thẻ *body* hay thẻ *div* chẳng hạn bạn cần khai báo thẻ *style* trong thẻ *<head>*.

```
<style>
   A {
     COLOR: #003063;
     TEXT-DECORATION: none
   }
   A:hover {
     COLOR: #003063;
     TEXT-DECORATION: underline
   }
   A:link {
     FONT-WEIGHT: bold;
     COLOR: red;
     TEXT-DECORATION: none
   }
   A:visited {
     FONT-WEIGHT: bold;
     COLOR: black;
     TEXT-DECORATION: none
   }
   .title {
     FONT-WEIGHT: normal;
     FONT-SIZE: 22px
   }
   .text{
     FONT: 11px Arial, Helvetica, sans-serif
   }
</style>
```

Trong đó, A tương ứng với liên kết (chuỗi trong thẻ <a>) có định dạng ứng với trường hợp liên kết, di chuyển con chuột, chọn liên kết.

```
   A {
     COLOR: #003063;
     TEXT-DECORATION: none
   }
   A:hover {
```

```
    COLOR: #003063;
    TEXT-DECORATION: underline
  }
  A:link {
    FONT-WEIGHT: bold;
    COLOR: red;
    TEXT-DECORATION: none
  }
  A:visited {
    FONT-WEIGHT: bold;
    COLOR: black;
    TEXT-DECORATION: none
  }
```

Chẳng hạn, chúng ta khai báo trang *PHP* với nội dung được áp dụng với kiểu định dạng khai báo trong thẻ *style* như vú dụ 6-1.

Ví dụ 6-1: Khai báo thẻ style

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>Style trong PHP</title>
<style>
A {
     COLOR: #003063;
     TEXT-DECORATION: none
}
A:hover {
     COLOR: #003063;
     TEXT-DECORATION: underline
}
A:link {
     FONT-WEIGHT: bold;
     COLOR: red;
     TEXT-DECORATION: none
}
A:visited {
     FONT-WEIGHT: bold;
     COLOR: black;
     TEXT-DECORATION: none
}
.title {
     FONT-WEIGHT: normal;
     FONT-SIZE: 22px;
     COLOR: #003063;
}
.text{
     FONT: 11px Arial, Helvetica, sans-serif
}
</style>
</head>
<body>
     <h4>Style Tag</h4>
     <TABLE cellSpacing=0 cellPadding=0
          width="100%" border=0>
     <TR>
        <TD vAlign=top class=title>
        *** Quản Trị SQL Server 2000 ***          </TD>
     </TR>
     <TR>
        <TD class=text>
        <div align=justify>
        Tìm hiểu cách cài đặt, cấu hình, quản trị,
        backup & restore, import & export, thiết
```

```
            kế, lập trình, tự động hoá tác vụ quản trị,
            bản sao dữ liệu, bảo mật và chống thâm nhập
            dữ liệu bằng.
            <b>SQL Injection</b>.</div>
            </TD>
        </TR>
        <TR><TD><hr size=1 color=red></TD</TR>
        <TR><TD>Welcome to
        <a href="www.huukhang.com" class=>
        www.huukhang.com</a></TD
        </TR>
      </TABLE>
</body>
</html>
```

Khi triệu gọi trang *style.PHP* trên trình duyệt, nội dung của trang *web* được định dạng theo thẻ *style* như hình 6-1.

Hình 6-1: Áp dụng thẻ style

Tương tự như vậy khi bạn muốn thống nhất nội dung trong những thẻ khác của một trang *web* thì khai báo một định dạng trong thẻ *style*. Tuy nhiên, khi đặt tên trùng với thẻ *HTML*, mọi thẻ đó trong trang sẽ cùng chung một định dạng. Chẳng hạn, bạn khai báo định dạng cho thẻ *td* như sau:

```
TD {
     FONT: 10px Arial, Helvetica, sans-serif
}
```

Mọi nội dung trình bày trong thẻ *td* sẽ có định dạng như trên. Nếu bạn muốn có định dạng khác thì khai báo thuộc tính *class* cho thẻ *td* đó, ví dụ sử dụng định dạng khác cho thẻ *td*:

```
<td class=text>ABC</td>
```

Thay vì chuỗi *ABC* sẽ có định dạng là *FONT: 10px Arial*, *Helvetica, sans-serif* thì chúng sẽ có định dạng của *FONT: 11px Arial, Helvetica, sans-serif*.

Chú ý rằng, trong mỗi trang *web* bạn phải khai báo thẻ *style* và định nghĩa thống nhất cho các thẻ. Khi có sự thay đổi bạn phải thay đổi trong mọi trang *web*. Để sử dụng chung cho mọi trang *web* trong ứng dụng, bạn cần xây dựng một tập tin *style*, tập tin được biết đến với tên gọi *custom style sheet* (*css*).

Bất kỳ trang *web* nào trong ứng dụng, muốn áp dụng kiểu định dạng trong tập tin *css* thì khai báo liên kết tập tin *css* bằng thẻ *link*.

Ví dụ, chúng ta khai báo tập tin *style.css* bao gồm các định dạng như ví dụ 6-2.

Ví dụ 6-2: Khai báo tập tin css

```
A {
     COLOR: #003063;
     TEXT-DECORATION: none
}
A:hover {
     COLOR: #003063;
     TEXT-DECORATION: underline
}
A:link {
     FONT-WEIGHT: bold;
     COLOR: red;
     TEXT-DECORATION: none
}
A:visited {
     FONT-WEIGHT: bold;
     COLOR: black;
     TEXT-DECORATION: none
}
.title {
     FONT-WEIGHT: bold;
     FONT-SIZE: 14px;
     COLOR: #003063;
}
.text{
     FONT: 11px Arial, Helvetica, sans-serif
}
```

Sau đó trong trang *PHP*, bạn khai báo liên kết tập tin này bằng thẻ *link*, nếu muốn áp dụng định dạng này trong mỗi thẻ *HTML* bạn sử dụng thuộc tính *class* như khai báo định dạng của thẻ *style* ngay trong trang đó như ví dụ 6-3.

Ví dụ 6-3: Khai báo sử dụng tập tin css

```
<html>
<head>
<title>
     Welcome to Link Style Sheet File
</title>
<LINK href="style.css" rel=stylesheet>
     <META http-equiv=Content-Type
     content="text/html; charset=utf-8">
</head>
<body>
     <h4>Style File</h4>
     <TABLE cellSpacing=0 cellPadding=0
          width="100%" border=0>
```

```
        <TR>
           <TD vAlign=top class=title>
           *** Quản Trị SQL Server 2000 ***          </TD>
        </TR>
        <TR>
           <TD class=text>
           <div align=justify>
           Tìm hiểu cách cài đặt, cấu hình, quản trị,
           backup & restore, import & export, thiết
           kế, lập trình, tự động hoá tác vụ quản trị,
           bản sao dữ liệu, bảo mật và chống thâm nhập
           dữ liệu bằng.
           <b>SQL Injection</b>.</div>
           </TD>
        </TR>
        <TR><TD><hr size=1 color=red></TD</TR>
        <TR><TD>Welcome to
        <a href="www.huukhang.com" class=>
        www.huukhang.com</a></TD
        </TR>
      </TABLE>
   </body>
   </html>
```

Triệu gọi trang *includestyle.php* trên trình duyệt như hình 6-3, màu và kích thước *font* cùng với kiểu chữa của nội dung không thay đổi so với *style.php*, bởi vì phần thẻ *style* được tách ra thành tập tin *style.css*, sau đó dùng thẻ *link* để liên kết tập tin *css* này vào trang *PHP* trở lại.

Hình 6-3: Liên kết tập tin css

Chú ý rằng, nếu khai báo thuộc tính class trong thẻ <*table*> thì những nội dung trong thẻ <*table*> sẽ có định dạng *theo* định dạng khai báo trong thuộc tính *class*. Tương tự, nếu khai báo thuộc tính *class* trong thẻ <*tr*> thì nội dung trong thẻ <tr> sẽ có định dạng giống như định dạng khai báo trong thông tin *class*.

3. THỐNG NHẤT KÍCH THƯỚC CỦA MOI TRANG PHP

Khi xây dựng ứng dụng *web* chuyên nghiệp, điều đầu tiên bạn nên quan tâm là sự thống nhất về kích thước của các phần trên trang *web*. Điều này có nghĩa là khi người sử dụng thay đổi trang *web* khi duyệt, phần *top, left, right, bottom* có kích thước như nhau.

Để làm điều này, bạn chia trang *web* ra thành 5 phần: *top*, *left*, *right, body* và *bottom*.

Phần top thường trình bày các thuộc tính như quản cáo (*baner*), *logo* (biểu tượng của công ty), *menu* (thực đơn của ứng dụng) và một số thông tin khác.

Phần *left* là thông tin về các *menu* phụ hay còn gọi là *menu* của *menu* chính, bên cạnh *menu* con này trang *web* thường có các liên kết về liên hệ, quảng cáo, *mailing list* (đăng ký *email*), gởi đến bạn bè (*send to friend*), ...

Đối với phần *right*, thường là phần giới thiệu về các thông đặc biệt và quảng cáo, chẳng hạn đối với ứng dụng bán sách, phần *right* thường là danh sách các nhóm sách bán chạy, sắp phát hành, ...

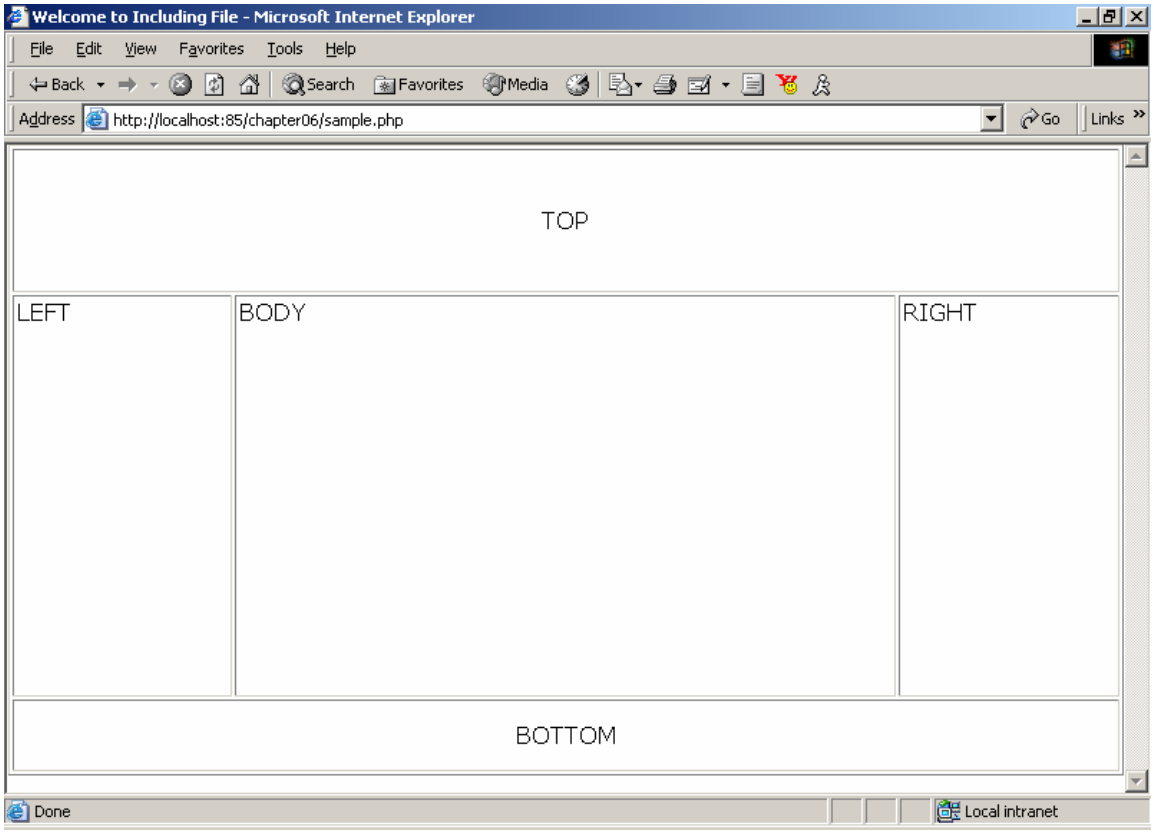
Phần *bottom* thường thông tin liên lạc của công ty, chủ nhân của *web site* và bản quyền. Ngoài ra, phần *bottom* đôi khi là danh sách các *menu* con khác.

Tóm lại, tuỳ thuộc vào ý tưởng thiết kế mỗi phần như trên bao gồm các thuộc tính mà nhà thiết kế cần trình bày sao cho phù hợp. Tuy nhiên, phần *body* là phần trình bày nội dung chính của mỗi trang *web*. Ngoài ra, tuỳ vào từng trường hợp cụ thể, trang *web* có thể không có phần *left* và *right*.

Như vậy, chúng ta sẽ chia trang *web* ra thành 5 phần, phần *body* chính là phần chính của trang *web* đó, còn 4 phần còn lại được chèn vào khi có nhu cầu.

Chẳng hạn, có những trang *web* do thông tin trình bày trong phần *body* nhiều, nên cần không gia lớn hơn, bạn có thể không cần sử dụng hai phần *left* và *right*.

Để làm điều này, trước tiên chúng ta thiết kế trang *sample.php* có 5 phần như hình 6-3.



Hình 6-3: Trang sample.php

Lưu ý:

- *Tạo một table gồm 3 hàng 3 cột và khai báo border=1 để dễ canh lề sau đó bạn có thể khai báo lại thuộc tính này bằng 0.*
- *Phần top và bottom là một hàng và merge 3 cột thành 1.*
- *Bên trong mỗi phần có thể có một hay nhiều thẻ table khác.*
- *Có thể không có phần left và right nhưng bắt buộc phần top và bottom phải có.*
- *Bạn có thể sử dụng chiều rộng của table theo kích thước tương đối (%) hay số chỉ định, đối với màn hình 600*800 thì chiều rộng thường sử dụng là 780, khi người sử dụng chọn độ phân giải của màn hình lớn hơn thì kích thước của table này không thay đổi, trong khi đó nội dung sẽ phủ đầy màn hình khi bạn khai báo kích thước theo 100%.*

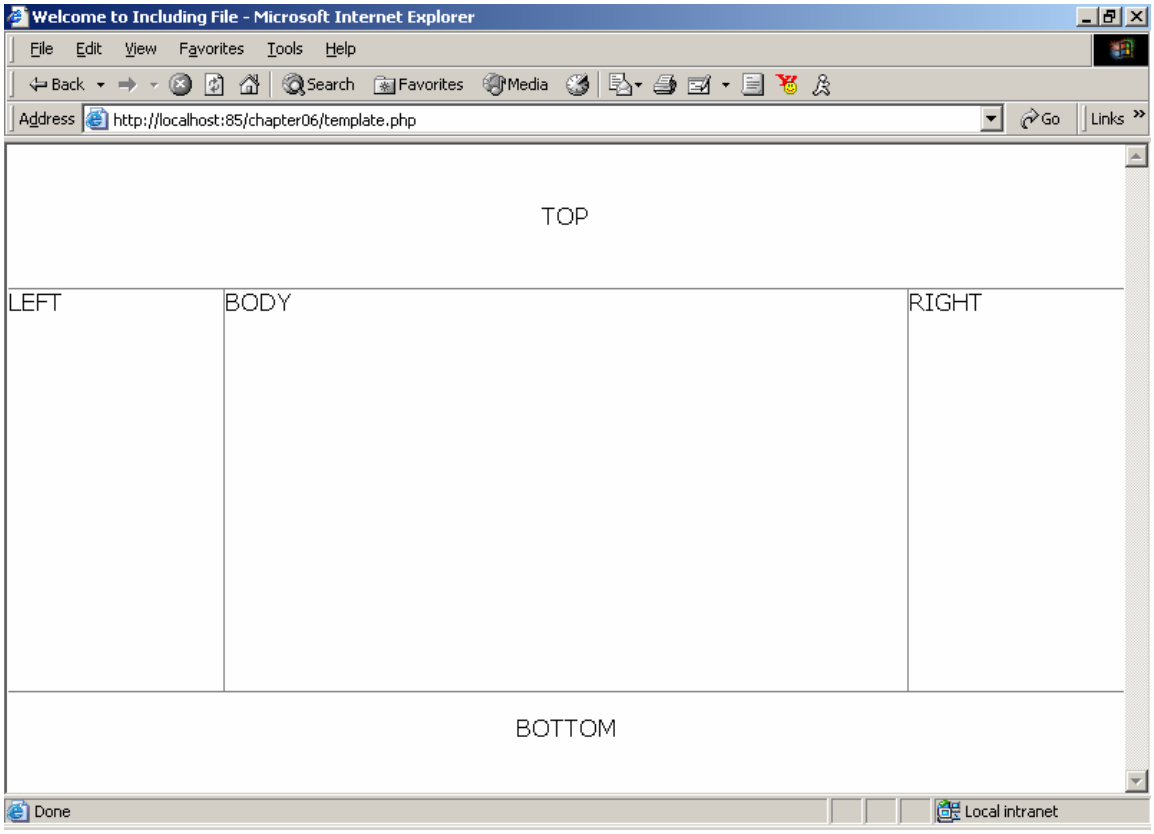
Để có giao diện như trang *sample.php* như trên, bạn có thể khai báo như ví dụ 6-3.

Ví dụ 6-3: Nội dung trang sample.PHP

```
<html>
<head>
<title>
     Welcome to Including File
</title>
<LINK href="style.css" rel=stylesheet>
     <META http-equiv=Content-Type
     content="text/html; charset=utf-8">
</head>
<body bottomMargin=0 leftMargin=0
     topMargin=0 rightMargin=0>
```

```
        <TABLE cellSpacing=2 cellPadding=2
            width="778" border=1 align=center>
         <TR HEIGHT="100">
           <TD  Align=center colspan=3>
             TOP
           </TD>
         </TR>
         <TR HEIGHT="280">
           <TD vAlign=top width="20%">
                LEFT
           </TD>
           <TD vAlign=top width="60%">
                BODY
           </TD>
           <TD vAlign=top width="20%">
                RIGHT
           </TD>
         </TR>
         <TR HEIGHT="50">
         <TD colspan=3 align=center>
              BOTTOM
         </TD
         </TR>
        </TABLE>
    </body>
    </html>
```

Trong trường hợp bạn muốn có đường phân cách giữa mỗi phần bằng *image*, bạn có thể khai báo lại trang *sample.php* có 5 hàng và 5 cột như *template.php* như hình 6-4.



Hình 2-4: Phân cách có viềng

Để trình bày trang *tempale.PHP* như hình 6-4, bạn khai báo nội dung trang này như ví dụ 6-4.

Ví dụ 6-4: Khai báo template.php

```
<html>
<head>
<title>
     Welcome to Including File
</title>
<LINK href="style.css" rel=stylesheet>
     <META http-equiv=Content-Type
     content="text/html; charset=utf-8">
</head>
<body bottomMargin=0 leftMargin=0
     topMargin=0 rightMargin=0>
     <TABLE width="778" border=0 cellSpacing=0
      cellPadding=0 align=center>
     <TR HEIGHT="100">
       <TD Align=center colspan=5>
          TOP
       </TD>
     </TR>
     <!---Khai báo đường phân cách--->
     <TR HEIGHT="1">
       <TD colspan=5 bgcolor=gray></TD>
     </TR>
     <TR HEIGHT="280">
       <TD vAlign=top width="150">LEFT</TD>
       <!---Khai báo đường phân cách--->
```

```
        <TD bgcolor=gray width="1"></TD>
        <TD vAlign=top width="476">BODY</TD>
        <!---Khai báo đường phân cách--->
        <TD bgcolor=gray width="1"></TD>
        <TD vAlign=top width="150">RIGHT</TD>
      </TR>
      <!---Khai báo đường phân cách--->
      <TR HEIGHT="1">
        <TD colspan=5 bgcolor=gray></TD>
      </TR>
      <TR HEIGHT="50">
      <TD colspan=5 align=center>
        BOTTOM
      </TD
      </TR>
     </TABLE>
</body>
</html>
```

Sau đó tách trang *template.php* này thành 5 trang khác nhau được đặt tên tương ứng là *top.htm, left.htm, right.htm* và *bottom.htm*, trong đó phần *body* tương ứng với trang *templates.php*.

Để khai báo chèn tập tin trong trang *PHP*, bạn sử dụng cú pháp như sau:

```
<?php
include(”filename”);
?>
```

Hay

```
<?php
require(”filename”);
?>
```

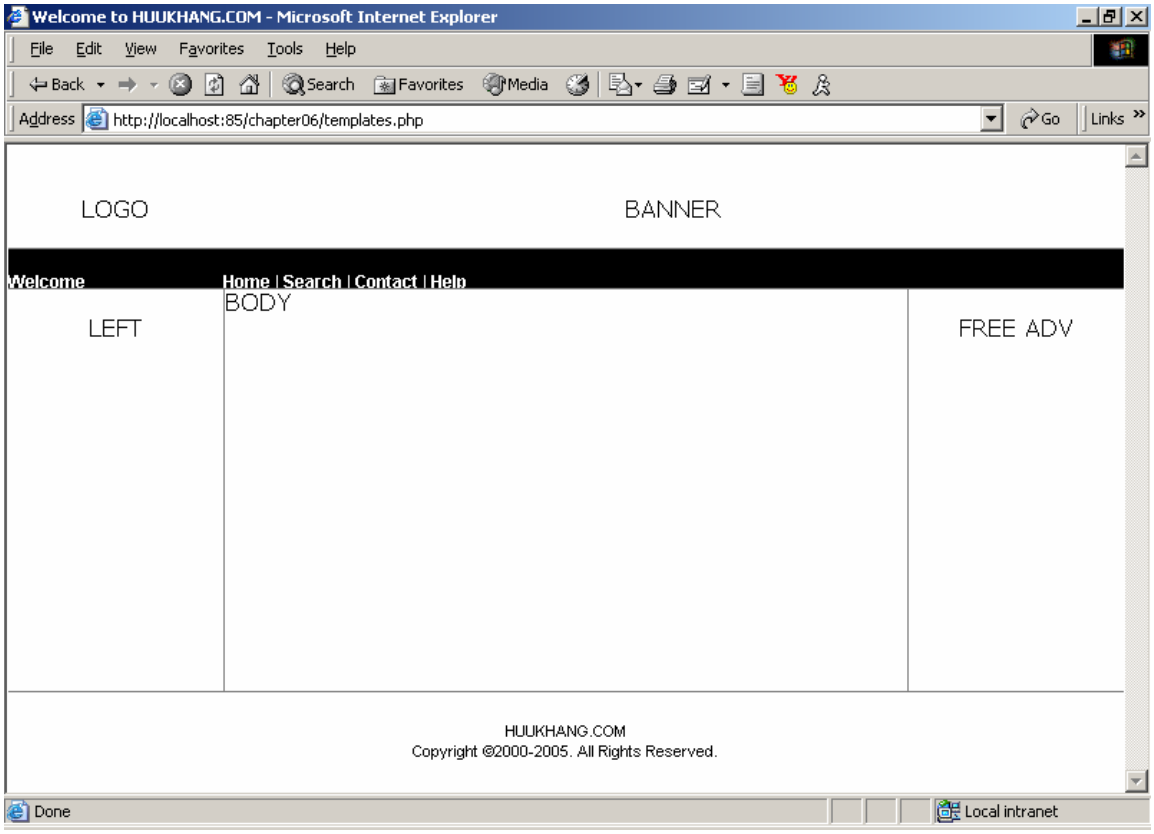
Trong đó trang *templates.PHP* khai báo chèn *top.htm, left.htm, right.htm* và *bottom.htm* như ví dụ 6-5.

Ví dụ 6-5: Khai báo chèn tập tin trong templates.php

```
<html>
<head>
<title>
     Welcome to HUUKHANG.COM
</title>
<LINK href="style.css" rel=stylesheet>
     <META http-equiv=Content-Type
     content="text/html; charset=utf-8">
</head>
<body bottomMargin=0 leftMargin=0
     topMargin=0 rightMargin=0>
     <TABLE width="778" border=0 cellSpacing=0
      cellPadding=0 align=center>
     <TR HEIGHT="100">
       <TD  Align=center colspan=5>
          <?php include("top.htm")?>
       </TD>
      </TR>
     <!---Khai báo đường phân cách--->
     <TR HEIGHT="1">
       <TD colspan=5 bgcolor=gray></TD>
     </TR>
```

```
        <TR HEIGHT="280">
           <TD vAlign=top width="150">
           <?php include("left.htm")?>
           </TD>
           <!---Khai báo đường phân cách--->
           <TD bgcolor=gray width="1"></TD>
           <TD vAlign=top width="476">BODY</TD>
           <!---Khai báo đường phân cách--->
           <TD bgcolor=gray width="1"></TD>
           <TD vAlign=top width="150">
           <?php include ("right.htm")?>
           </TD>
        </TR>
        <!---Khai báo đường phân cách--->
        <TR HEIGHT="1">
           <TD colspan=5 bgcolor=gray></TD>
        </TR>
        <TR HEIGHT="50">
        <TD colspan=5 align=center>
           <?php include("bottom.htm")?>
        </TD
        </TR>
       </TABLE>
    </body>
    </html>
```

Khi triệu gọi trang *templates.php*, nội dung của 4 tang *left.htm*, *right.htm, top.htm, bottom.htm* chèn vào trang *templates.php* như hình 6-5.



Hình 6-5: Trang templates.php sau khi chèn

Trong đó, nội dung của trang *top.htm* định nghĩa tương tự như ví dụ 6-5-1.

Ví dụ 6-5-1: Nội dung trang top.htm

```
<TABLE width="100%" border=0 cellSpacing=0
    cellPadding=0 HEIGHT="100%" align=center>
       <TR >
         <TD width="150" Align=center>
       LOGO
     </TD>
     <TD  Align=center>
       BANNER
     </TD>
    </TR>
    <TR HEIGHT="1">
         <TD colspan=2 bgcolor=gray></TD>
    </TR>
    <TR HEIGHT="20%" bgcolor=black class=menu>
         <TD width="150" >
       Welcome
     </TD>
     <TD>
       Home | Search | Contact | Help
     </TD>
    </TR>
</TABLE>
```

Nội dung của tập tin *left.htm* được định nghĩa tương tự như ví dụ 6-5-2.

Ví dụ 6-5-2: Nội dung trang left.htm

```
<TABLE width="100%" border=0 cellSpacing=0
     cellPadding=0 HEIGHT="100%" align=center>
        <TR >
          <TD width="150" Align=center>
          LEFT
     </TD>
     </TR>
</TABLE>
```

Nếu có sử dụng trang right.htm thì nội dung của tập tin này được định nghĩa tương tự như ví dụ 6-5-3.

Ví dụ 6-5-3: Nội dung trang right.htm

```
<TABLE width="100%" border=0 cellSpacing=0
     cellPadding=0 HEIGHT="100%" align=center>
        <TR >
          <TD width="150" Align=center>
          FREE ADV
     </TD>
     </TR>
</TABLE>
```

Tương tự như vậy, trang *bottom.htm* có nội dung như ví dụ 6-5-4.

Ví dụ 6-5-4: Nội dung trang bottom.htm

```
<TABLE width="100%" border=0 cellSpacing=0
     cellPadding=0 HEIGHT="100%" align=center>
        <TR class=text>
          <TD Align=center>
          HUUKHANG.COM<br>
        Copyright ©2000-2005.
        All Rights Reserved.
     </TD>
     </TR>
</TABLE>
```

Chú ý rằng, trong mỗi trang khai báo chèn không có các thẻ đóng và mở *html*, *body* bởi khi chèn thì nội dung của tập tin được chèn sẽ được chèn vào tập tin bị chèn và trong tập tin bị chèn đã có hai thẻ này.

Kịch bản trình chủ PHP hỗ trợ các tập tin được chèn với các tên mở rộng như *htm, PHP, inc, lib, html*. Do thực chất của việc khai báo chèn là chèn đoạn mã trong tập tin chèn vào tập tin bọ chèn, trong trường hợp này trang chèn *htm* hay *PHP* đều giống nhau đó là lý do tại sao các trang chèn ở trên đều có tên mở rộng là *htm*.

Tuy nhiên, khi bạn gọi trang chèn này một mình ví dụ *tom.htm*, nếu bên trong có mã *PHP* thì mã đó không được thông dịch. Nếu những trang chèn này có nhu cầu gọi một mình thì bạn có thể chuyển chúng thành trang *PHP* thay vì *htm* như đã trình bày.

Sau khi có được trang *templates.php*, bạn có thể sử dụng trang này là mẫu cho các trang khác bằng cách *save as* thành các trang *PHP* khác khi lập trình. Khi khai báo chèn tập tin, bạn có thể sử dụng đường dẫn tương đối hoặc tuyệt đối của tập tin chèn so với ập tin bị chèn.

4. TẬP TIN DÙNG CHUNG

Ngoài cách chèn ở trên, nếu bạn có những hàm sử dụng chung cho các trang PHP khác thì bạn khai báo thành một trang PHP khác sau đó dùng cú pháp chèn tập tin để chèn chúng vào khi có nhu cầu.

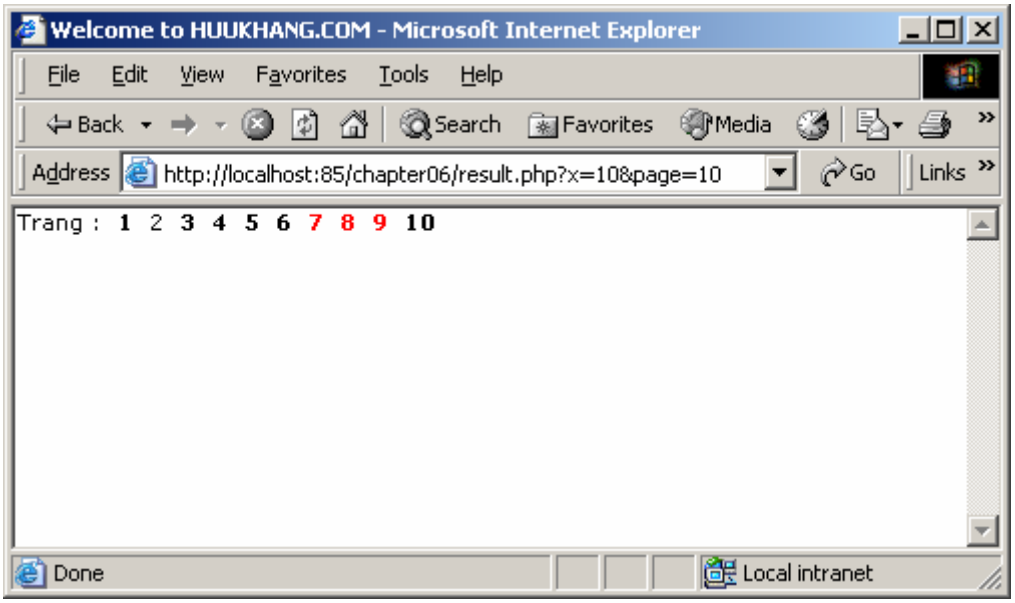
Ví dụ trong trường hợp này chúng ta muốn sử dụng chung hàm có tên getPaging nhận 5 tham số $totalRows (tổng số mẩu tin), $curPg (số trang hiện hành), $pg (số trang trình bày), $re (số mẩu tin trên 1 trang), $file (trang php cần gọi) trong tập tin paging.php.

```
<?php
function paging($totalRows,$curPg,$pg,$re,$file)
{
$paging="";
$mxR = $re;
$mxP = $pg;
if($totalRows%$mxR==0)
     $totalPages = (int)($totalRows/$mxR);
else
     $totalPages = (int)($totalRows/$mxR+1);
$curRow = ($curPg-1)*$mxR+1;
if($totalRows>$mxR)
{
$start=1;
$end=1;
$paging1 ="";
for($i=1;$i<=$totalPages;$i++)
{
if(($i>((int)(($curPg-1)/$mxP))* $mxP) && ($i<=((int)(($curPg-
1)/$mxP+1))* $mxP))
     {
       if($start==1) $start=$i;
       if($i==$curPg)
         $paging1 .= $i."  ";
       else
         {
         $paging1 .= "<a class=lslink href='$file";
         $paging1 .="&page=".$i."'>".$i;
         $paging1 .="</a>  ";
         }
       $end=$i;
     }
}
$paging.= "Trang :  " ;
if($curPg>$mxP)
     {
     $paging .="<a class=lslink href='$file";
     $paging .="&page=".($start-1);
     $paging .="'>Previous</a>  ";
     }
$paging.=$paging1;
if((((($curPg-1)/$mxP+1)*$mxP) < $totalPages)
     {
     $paging .= "<a class=lslink href='$file";
     $paging .="&page=".($end+1);
     $paging .="'>Next</a>  ";
     }
}
return $paging;
}
?>
```

Sau đó khai báo trang result.php, chèn tập tin paging.php và gọi hàm getPaging nhưu sau:

```
<html>
<head>
<title>
     Welcome to HUUKHANG.COM
</title>
<LINK href="style.css" rel=stylesheet>
     <META http-equiv=Content-Type
     content="text/html; charset=utf-8">
</head>
<body bottomMargin=0 leftMargin=0 topMargin=0 rightMargin=0>
<?php
     include("paging.php");
     echo paging(47,2,10,5,"result.php?x=10");
?>
</body>
</html>
```

Kết quả trả về như hình 6-6 sau



Hình 6-6: Hàm dùng chung

5. KẾT LUÂN

Trong bài này, chúng ta tìm hiểu cách khai báo hàm, trang php và khai báo chèn tập tin.

Môn học: PHP

Bài 7

Bài học này chúng ta sẽ làm quen cách xử lý chuỗi, mảng, kiểu DataTime trong PHP:

- ✓ *Xử lý chuỗi*
- ✓ *Làm việc với mảng dữ liệu*
- ✓ *Kiểu DateTime*

1. XỬ LÝ CHUỖI

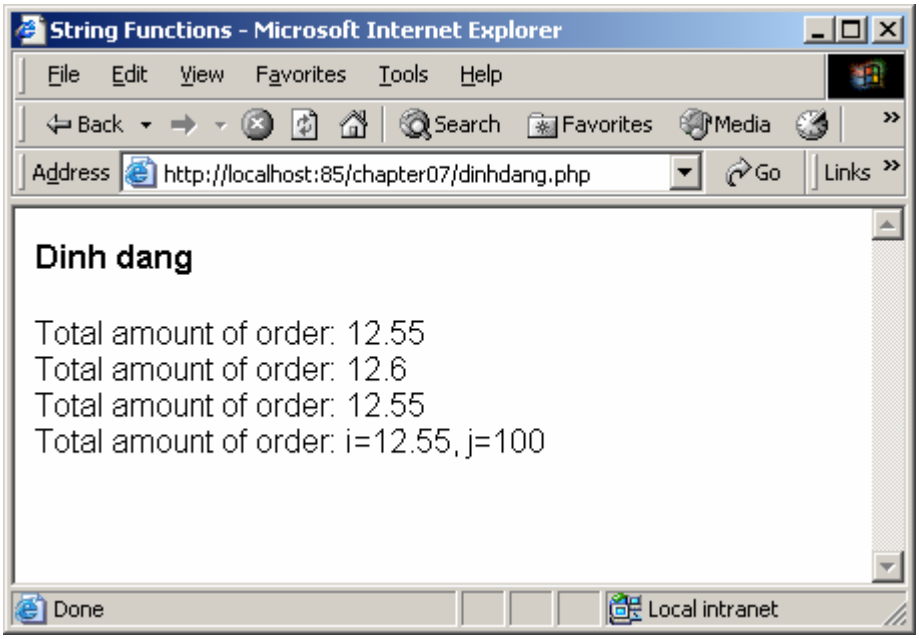
PHP là kịch bản được xem là tốt nhất cho xử lý chuỗi, bằng cách sử dụng các hàm xử lý chuỗi, bạn có thể thực hiện các ý định của mình khi tương tác cơ sở dữ liệu, tập tin hay dữ liệu khác.

1.1. Định dạng chuỗi

Khi xuất kết quả ra trình duyệt, bạn có thể sử dụng các định dạng chuỗi tương tự như ngôn ngữ lập trình C. Chẳng hạn, chúng ta in giá trị của biến $i trong trang dinhdang.php như ví dụ 7-1.

```
<html>
<head>
 <title>String Functions</title>
</head>
<body>
<h4>Dinh dang</h4>
<?php
$i=12.55;
$j=100;
echo "Total amount of order: $i<br>";
printf("Total amount of order: %.1f", $i);
echo "<br>";
printf("Total amount of order: %.2f", $i);
echo "<br>";
printf("Total amount of order: i=%.2f, j=%.0f", $i,$j);
?>
</body>
</html>
```

Kết quả xuất hiện như hình 7-1



Hình 7-1: Định dạng chuỗi in

Trong đó các định dạng được chia ra nhiêu loại tuỳ thuộc vào các ký tự bạn sử dụng.

```
% - Không yêu cầu tham số.
b - Trình bày dạng số integer và hiện thực dưới dạng binary.
c - Trình bày dạng số integer và hiện thực dưới dạng mã ASCII.
d - Trình bày dạng số integer và hiện thực dưới dạng decimal.
e - Trình bày dạng số logic và hiện thực dưới dạng 1.2e+2.
u - Trình bày dạng số integer và hiện thực dưới dạng decimal không dấu.
f - Trình bày dạng số float và hiện thực dưới dạng số chấm động.
o - Trình bày dạng số integer và hiện thực dưới dạng hệ số 10.
s - Trình bày dạng chuỗi.
x - Trình bày dạng số integer và hiện thực dưới dạng hệ số 16 với ký tự thường.
X - Trình bày dạng số integer và hiện thực dưới dạng hệ số 16 với ký tự hoa.
```

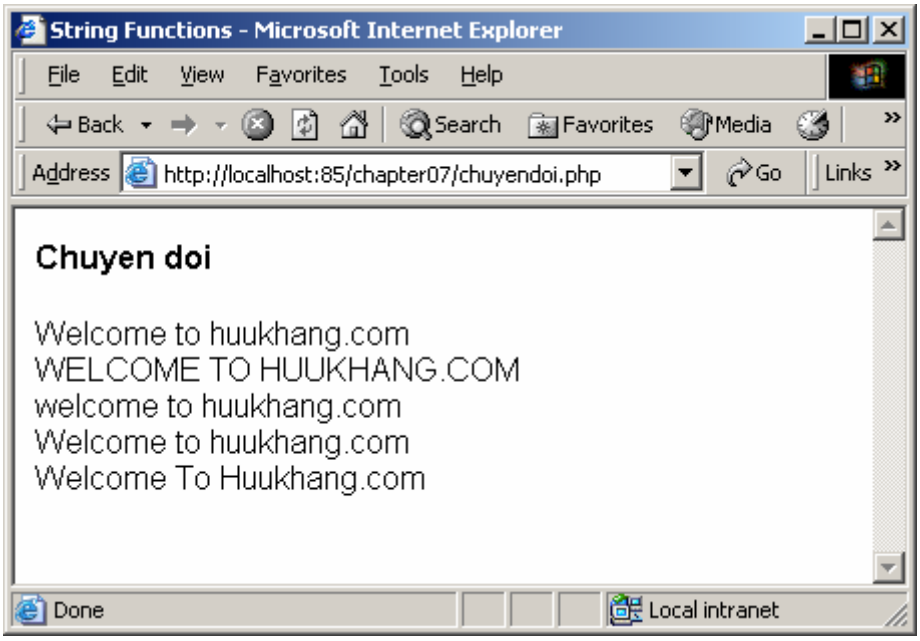
1.2. Hàm chuyển đổi chuỗi

Để chuyển đổi chuỗi ra ký tự hoa thường bạn sử dụng một trong 4 hàm như ví dụ 7-2 trong trang chuyendoi.php:

```
<html>
   <head>
    <title>String Functions</title>
   </head>
   <body>
   <h4>Chuyen doi</h4>
   <?php
      $str="Welcome to huukhang.com";
      echo $str;
      echo "<br>";
      echo strtoupper($str);
      echo "<br>";
      echo strtolower($str);
      echo "<br>";
      echo ucfirst($str);
      echo "<br>";
      echo ucwords($str);
      echo "<br>";
   ?>
```

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

Kết quả trình bày như hình 7-2.



Hình 7-2: Chuyển đổi chuỗi

1.3. Hàm tách hay kết hợp chuỗi

Để tách hay kết hợp chuỗi, bạn sử dụng một trong các hàm thường sử dụng như strtok, explode hay substr. Chẳng hạn, chúng ta sử dụng 4 hàm này trong ví dụ 7-4 trong trang tachchuoi.php.

```
<html>
<head>
 <title>String Functions</title>
</head>
<body>
<h4>Tach hop chuoi</h4>
<?php
   $string = "Xin chao ban da den voi huukhang.com";
   $str = $string;
   echo $string."<br>";
   $tok = strtok($string, " ");
   while ($tok)
   {
     echo "Word= $tok<br />";
     $tok = strtok(" \n\t");
   }
   echo $str."<br>";
   echo substr($str,24)."<br>";
   $a[]=array();
   $a=explode(" ",$str);
   while($i=each($a))
   {
```

```
        echo $i["value"]."<br>";
    }
?>
</body>
</html>
```

Kết quả trình bày như hình 7-4.

Hình 7-4: Hàm tách chuỗi

Trong trường hợp kết hợp giá trị của các phần tử của mảng thành chuỗi, bạn sử dụng hàm implode như ví dụ 7-5 trong trang kethop.php:

```
<html>
<head>
 <title>String Functions</title>
</head>
<body>
<h4>Ket hop chuoi</h4>
<?php
     $str = "Xin chao ban da den voi huukhang.com";
     $a[]=array();
     $a=explode(" ",$str);
     while($i=each($a))
     {
        echo $i["value"]."<br>";
     }
     $str=implode(" ",$a);
     echo $str;
```

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

Kết quả trình bày như hình 7-5.

Hình 7-5: Hàm kết hợp chuỗi

1.4. Tìm kiếm và thay thế chuỗi

Để thay thế chuỗi, bạn sử dụng hàm str_replace, chẳng hạn trong trường hợp hợp bạn lấy giá trị từ thẻ nhập liệu, sau đó tìm kiếm nếu phát hiện dấu ‘ thì thay thế thành hai dấu nháy như trang replace.php.

```
<html>
<head>
 <title>String Functions</title>
</head>
<body>
<h4>That the chuoi</h4>
<?php
   $str="";
   if (isset($txtfullname))
      $str = $txtfullname;
   if($str != "");
      $str=str_replace("o","a",$str);
   echo $str."<br>";
?>
<form action=replace.php method=post>
fullname: <input name=txtfullname value="<?=$str?>"><br>
<input type=submit value=Submit>
</form>
```

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

Khi triệu gọi trang replace.php trên trình duyệt, bạn sẽ có kết quả như sau:

Hình 7-6: Hàm thay thế chuỗi

Ngoài ra, bạn có thể sử dụng các hàm như strpos (trả về vị trí chuỗi con trong chuỗi mẹ), ...

2. LÀM VIỆC VỚI MẢNG DỮ LIỆU

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.

2.1. 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;
```

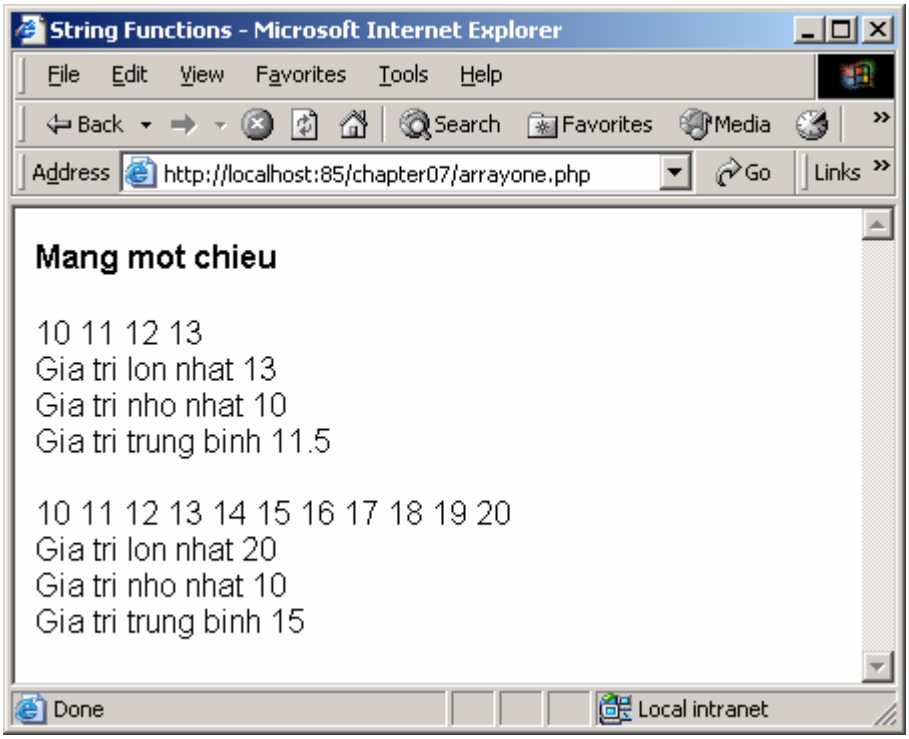
Lấy giá trị của phần tử mảng, bạn cũng thực hiện tương tự như trường hợp truy cập mảng phần tử.

```
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:

```
<html>
<head>
 <title>Array</title>
</head>
<body>
<h4>Mang mot chieu</h4>
<?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>" ;
?>
</body>
</html>
```

Kết quả trình bày như hình 7-7 khi triệu gọi trang arrayone.php.



Hình 7-7: Khai báo và sử dụng mảng một chiều

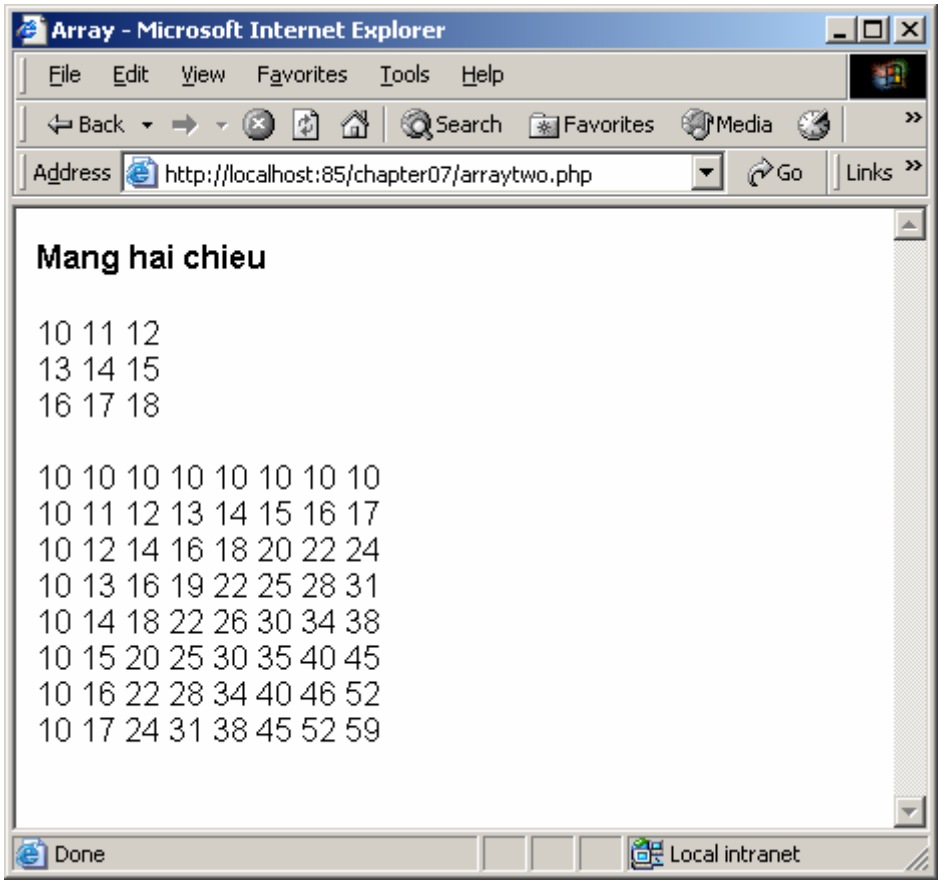
2.2. Mảng hai chiều

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

```
  <html>
<head>
 <title>Array</title>
</head>
<body>
<h4>Mang hai chieu</h4>
<?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>";

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

Khi triệu gọi trang này trên trình duyệt, kết quả trình bày như hình 7-8.



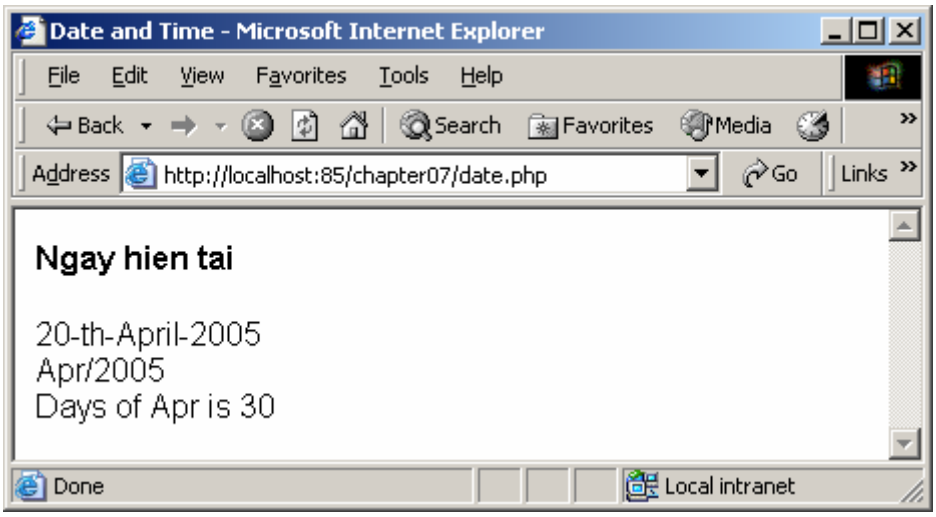
Hình 7-8: Mảng hai chiều

3. 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:

```
<html>
<head>
 <title>Date and Time</title>
</head>
<body>
<h4>Ngay hien tai</h4>
<?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>";
?>

</body>
</html>
```

Kết quả trả về như hình 7-9.



Hình 7-9: Sử dụng hàm Date

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

Code	Diễn 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 12^{th}).
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* từ 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.

---

4. <u>KẾT LUẬT</u>

Trong bài này, chúng ta tập trung tìm hiểu xử lý chuỗi, mảng và hàm ngày tháng. Trong bài tiếp, chúng ta tiếp tục tìm hiểu cơ sở dữ liệu mySQL.

Môn học: MySQL

Bài 8

Bài học này chúng ta sẽ làm quen cách thao tác trên cơ sở dữ liệu MySQL:

- ✓ *Giới thiệu cơ sở dữ liệu MySQL*
- ✓ *Cài đặt MySQL*
- ✓ *Cấu hình*
- ✓ *Kiểu dữ liệu*
- ✓ *Khai báo các phát biểu*

1. GIỚI THIỆU CƠ SỞ DỮ LIÊU 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, phần 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. CÀI ĐẶT MYSQL

Để cài đặt MySQL trên nền Windows bạn theo các bước sau:

- Trước tiên bạn chép tập tin mysql-4.0.0a-alpha-win.zip vào đĩa cứng hoặc chọn chúng từ đĩaq CD và giải nén tập tin
- Chạy tập tin Setup.exe, chọn đĩa C hay D
- Sau khi cài đặt thành công, bạn kiểm tra trong Windows Services xuất hiện dịch vụ mySQL hay không?. Để sử dụng được MySQL thì trạng thái của dịch vụ này phải ở chế độ Started.

Lưu ý rằng, trong trường hợp MySQL không thể chạy được, do dịch vụ của MySQL chưa Started như , để có thể chạy được MySQL thì bạn cần một số thay đổi trong tập tin my.ini trong thư mục WINNT

```
---------------------------------------------------------------------------------
#This File was made using the WinMySQLAdmin 1.3
#Tool
#9/11/2003 10:50:13 AM
#Uncomment or Add only the keys that you know how works.
#Read the MySQL Manual for instructions
[mysqld-nt]
basedir=C:/mysql
#bind-address=127.0.0.1
datadir=C:/mysql/data
#language=C:/mysql/share/your language directory
#slow query log#=
#tmpdir#=
#port=3306
#set-variable=key_buffer=16M
```

```
[WinMySQLadmin]
Server=C:/mysql/bin/mysqld-nt.exe
user=root
password=
QueryInterval=10
```

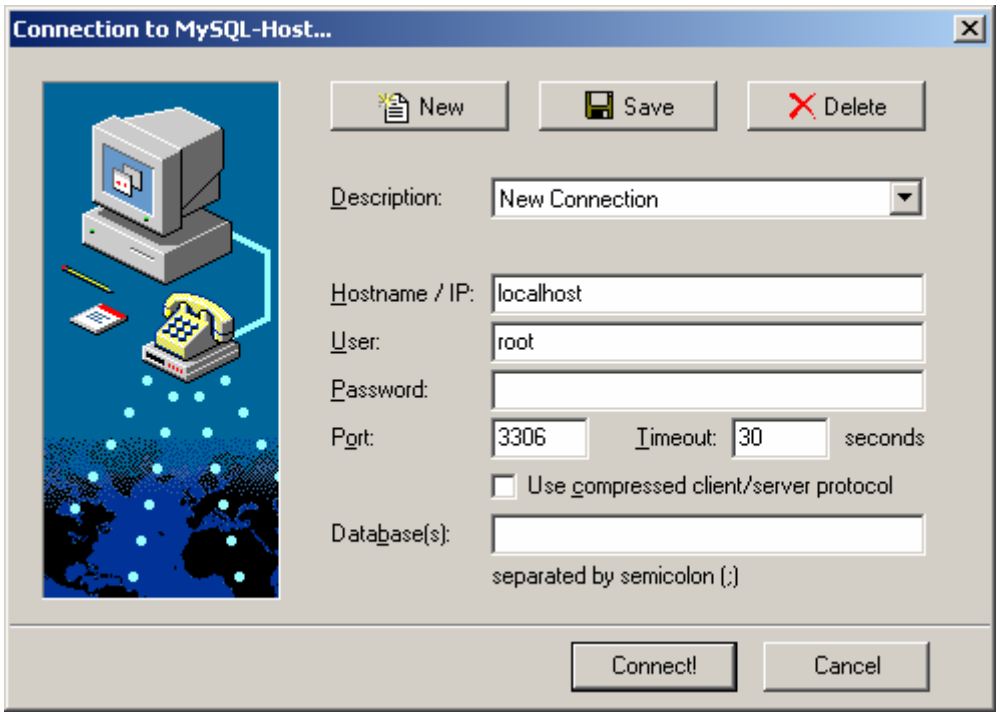
3. TAO CƠ SỞ DỮ LIỆU VÀ NGƯỜI DÙNG

Trong trường hợp bạn sử dụng giao diện đồ hoạ thì dùng ích quản trị cơ sở dữ liệu MySQL, bạn có thể chạy tập tin *mysqlfront.exe* trong thư mục *MySQL Control*, bằng cách chạy tập tin cửa sổ xuất hiện như hình 8-1. Nếu lần đầu tiên tạo kết nối cơ sở dữ liệu, bạn cần phải tạo một *Connection*, cung cấp tên *Server* hay *IP* của máy chứa *MySQL*.

Tuy nhiên, trong trường hợp máy chứa cơ sở dữ liệu *MySQL* là máy đang sử dụng, bạn có thể sử dụng *localhost*. Ngài ra, cũng giống như các cơ sở dữ liệu khác, *Username* mặc định của cơ sở dữ liệu *MySQl* là *root* và *Password* là rỗng.

Nếu bạn đã có cơ sở dữ liệu đang tồn tại, bạn có thể gõ tên cơ sở dữ liệu trong phần *Databases* (nếu muốn mở nhiều *database*, bạn có thể dùng dấu ; để phân cách).

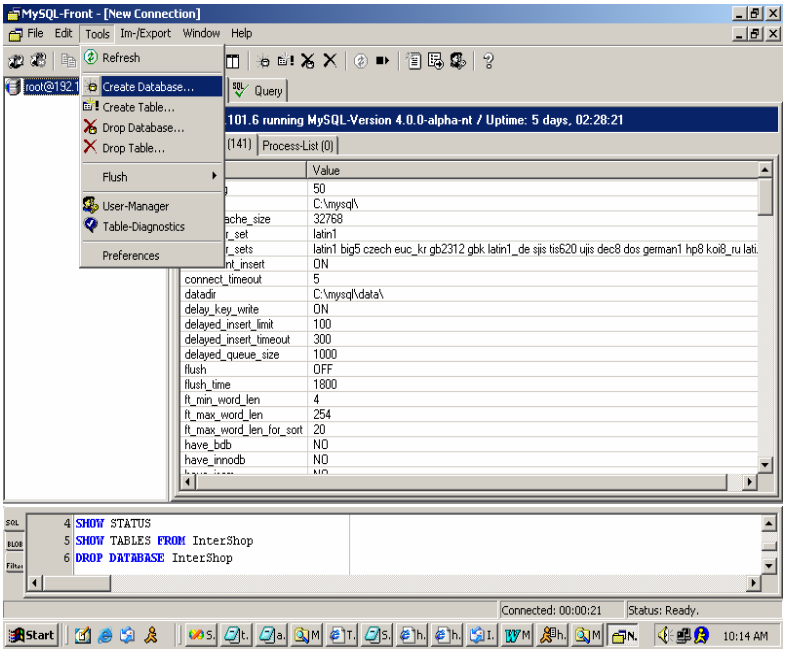
Trong trường hợp lần đầu tiên, bạn không cần cung cấp tên cơ sở dữ liệu, bạn có thể tạo chúng sau khi kết nối.



Hình 8-1: Kết nối cơ sở dữ liệu bằng MySQLFront Tool

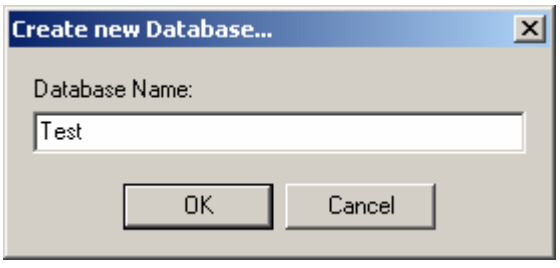
Sau kết nối cơ sở dữ liệu thành công, trình điều khiển cơ sở dữ liệu *MySQL* có giao diện như hình 8-2, cộng việc đầu tiên bạn phải thực hiện là tạo cơ sở dữ liệu.

Bắt đầu từ *menu* có tên *Tools* | *Create Database* hay chọn tên *root@localhost* | *R-Click* | *Create Database*, cửa sổ xuất hiện như hình 8-3.



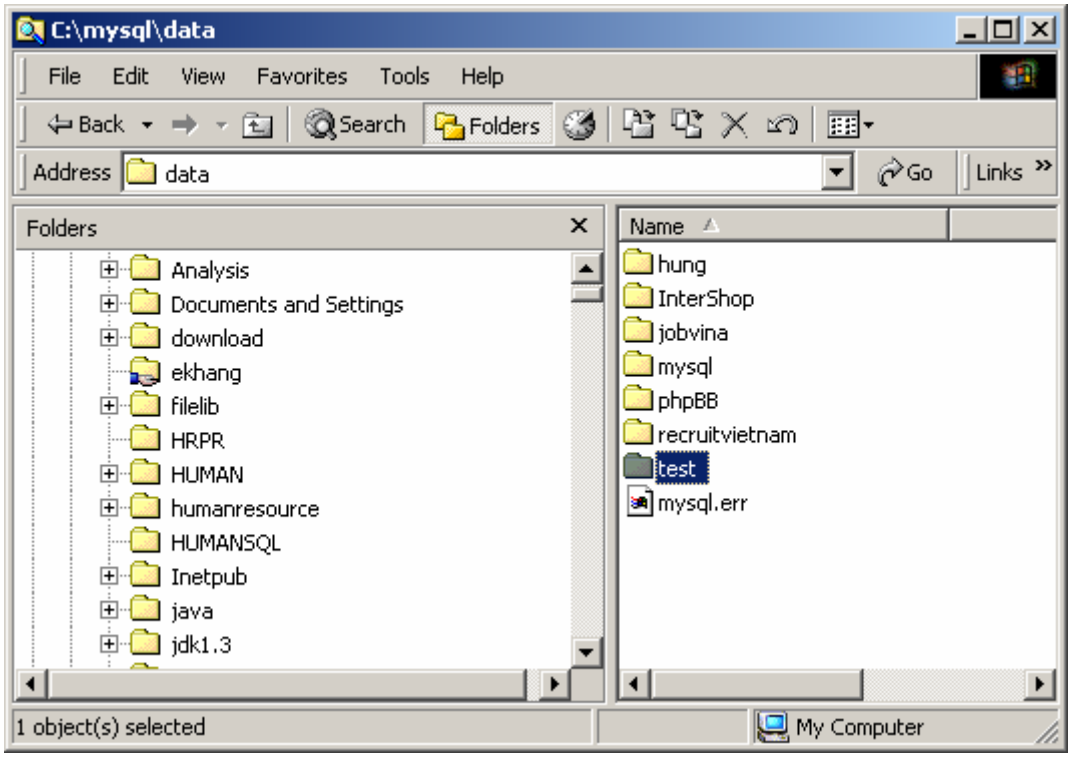
Hình 8-2: Giao diện điều khiển cơ sở dữ liệu MySQL

Cung cấp tên cơ sở dữ liệu, trong trường hợp này bạn có thể nhập *Test*, bấm nút *OK*, cơ sở dữ liệu xuất hiện trong cửa sổ điều khiển.



Hình 8-3: Tạo cơ sở dữ liệu có tên Test

Trong cả hai trường hợp tạo cơ sở dữ liệu bằng *MySQL* thành công như trên, bạn có thể tìm thấy tên cơ sở dữ liệu đó trong thư mục *mysql/data* như hình 8-4 sau:



Hình 8-4: Thư mục tin cơ sở dữ liệu Test

3.1. Quản lý người dùng

Làm thế nào để đăng nhập vào cơ sở dữ liệu *MySQL*, bạn có thể sử dụng hai cách như trình bày ở trên. Tuy nhiên, sau khi tạo ra các *username* khác, bạn có thể sử dụng chúng để đăng nhập.

Để đăng nhập vào *MySQL* bằng *Command line*, bạn chỉ cần gõ *>mysql – hostname –u username – p* từ dấu nhắc hay đăng nhập bằng cách sử dụng trình giao diện đồ hoạ. Từ khoá *-h* hỉ ra rằng tên (*computer name*), *IP*, hay *localhost* của máy có sử dụng cơ sở dữ liệu *MySQL*, *-u* chỉ ra rằng bạn sử dụng *username*, *username* là tên *username*, *-p* được chỉ định khi *username* này có *password*. Trong trường hợp *password* là rỗng, bạn có thể không cung cấp tham số *–p*.

Để tạo *User* trong cơ sở dữ liệu *MySQL*, bạn có thể sử dụng hai cách trên. Nếu bạn thực hiện việc tạo một *Username* bằng *Command line*, bạn có thể gõ từ dấu nhắc như phát biểu sau:

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

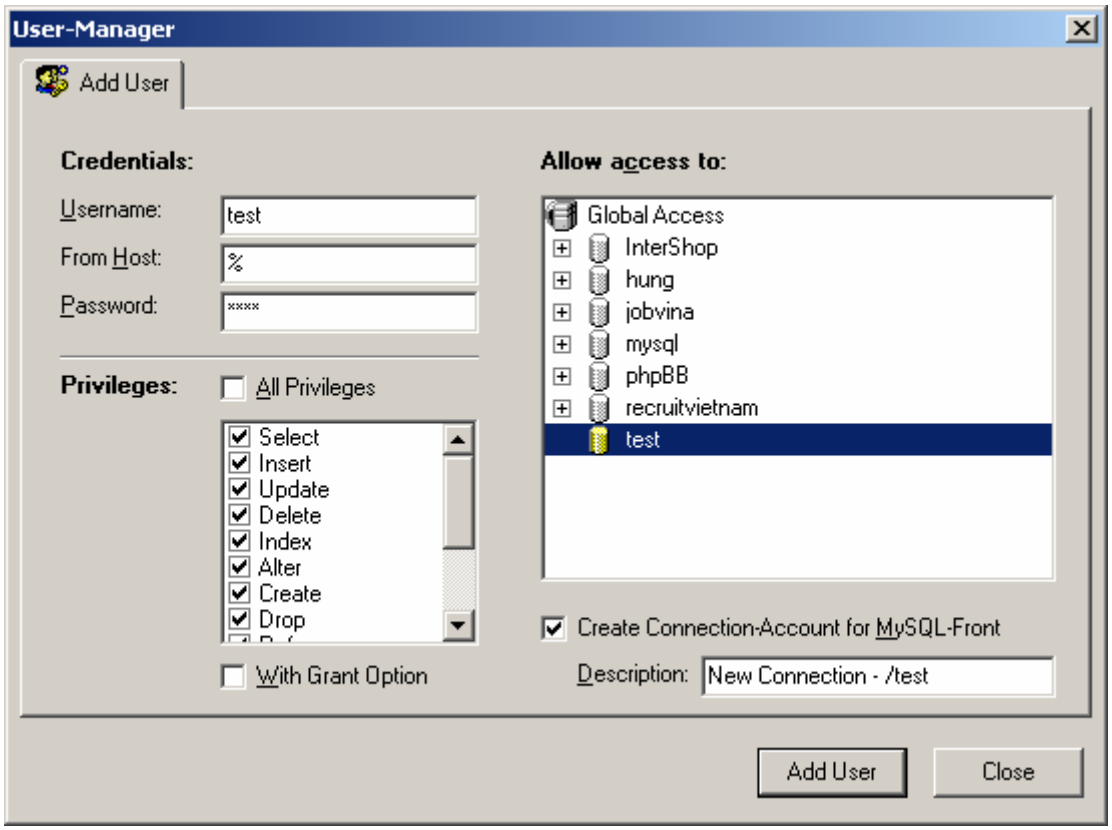
Trong phát biểu trên, vừa tạo ra *User* có tên *myis*, với *hostname* là cơ sở dữ liệu hiện hành, *password* là *1234* và được các đặt quyền *Select, Insert, Update, Delete, Index, Alter, Create*, *Drop* trên cơ sở dữ liệu hiện hành.

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'
```

Nếu bạn sử dụng giao diện đồ hoạ, bạn có thể tạo *username* và gán quyền như trên bằng cách sử dụng *menu* có tên *Tools | User-Manager*, cửa sổ xuất hiện như hình 8-5.



Hình 8-5: Tạo Username

3.2. Cấp quyền cho người dùng

Các đặt quyền *Select, Insert, Update, Delete, Index, Alter, Create*, *Drop* trên cơ sở dữ liệu, bạn có thể tham khảo chi tiết trong bảng 8-1.

Bảng 8-1: 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 tables	Cho phép *user* tạo mới đối tượng *Table* hay Database.
drop	databases tables	Cho phép user xoá đối 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 8-2, bạn có thể xem xét các đặt quyền quản trị để cấp cho người dùng.

Bảng 8-2: 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 8-2 cho họ, trong trường hợp bạn muốn cấp tất cả các quyền trong bảng 8-1 và Bảng 8-2 cho *username* khi tạo ra họ, bạn *Table* sử dụng từ khoá *All* thay vì *All Privileges* trong phát biểu tạo *user* như sau:

```
GRANT
            ALL
            ON *.* TO 'ekhang'@'%'
            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 uyền như sau:

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

3.3. Xoá quyền của user

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)]
ON item
From username
```

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

```
Revoke All
ON item
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
ON item
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 *mnb123*, bạn có thể khai báo như sau:

```
Grant all
On *
To fred indetifyed by 'mnb123'
With Grant Option;
```

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

```
Revoke all
On *
From fred;
```

Tạo một *user* có tên *ekhang* 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 ekhang identified by '12345678';
```

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

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

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

```
Revoke update, delete, drop
On Test.*
From ekhang;
```

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

```
Revoke All
On Test.*
From ekhang;
```

4. KIỂU DỮ LIÊU CỦA CƠ SỞ DỮ LIÊU 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.

4.1. Loại dữ liệu numeric

Kiểu dữ liệu *numeric* bao gồm kiểu số nguyên trình bày trong bảng 8-3 và kiểu số chấm động, trong trường hợp dữ liệu kiểu dấu chấm động bạn cần phải chỉ rõ bao nhiều số sau đấu phần lẻ như trong bảng 8-4.

Bảng 8-3: 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 hay 0..2^{32}-1	4	Số nguyên.
bigint	-2^{63}->2^{63}-1 hay 0..2^{64}-1	8	Số nguyên lớn.

Bảng 8-4: Kiểu dữ liệu số chấm động

Loại	Range	Bytes	Diễn giải
float	phụ thuộc Số thập Phân		Số thập phân dạng *Single* hay *Double*.
Float(M,D)	±1.175494351E-38 ±3.40282346638	4	Số thập phân dạng *Single*.
Double(M,D)	±1.7976931348623157308 ±2.2250738585072014E-308	8	Số thập phân dạng *Double*.

```
Float(M[,D])
                                Số chấm động lưu
                                dưới dạng char.
-------------------------------------------
```

4.2. Loại dữ liệu Datet and Time

Kiểu dữ liệu *Date and Time* cho phép bạn nhập liệu dưới dạng chuỗi hay dạng số như trong bảng 8-5.

Bảng 8-5: Kiểu dữ liệu số nguyên

```
-------------------------------------------
Loại      Range       Diễn giải
-------------------------------------------
Date      1000-01-01  Date trình bày dưới dạng
                      yyyy-mm-dd.

Time      -838:59:59  Time trình bày dưới
          838:59:59   dạng hh:mm:ss.

DateTime  1000-01-01  Date và Time trình bày dưới
          00:00:00    dạng yyyy-mm-dd hh:mm:ss.
          9999-12-31
          23:59:59

TimeStamp[(M)]
          1970-01-01  TimeStamp trình bày dưới
          00:00:00    dạng yyyy-mm-dd hh:mm:ss.
Year[(2|4)]
          1970-2069   Year trình bày dưới
          1901-2155   dạng 2 số hay 4 số.
-------------------------------------------
```

Đối với kiểu dữ liệu *TimeStamp*, bạn có thể định dạng nhiều cách như trình bày trong bảng 8-6.

Bảng 8-6: 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
-------------------------------------------
```

4.3. 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 hcuỗi 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*. Bạn có thể tham khảo cả ba loại trên trong bảng 8-7.

Bảng 8-7: Kiểu dữ liệu String

Loại	Range	Diễn giải
char	1-255 characters	Chiều dài của chuỗi lớn nhất 255 ký tự.
varchar	1-255 characters	Chiều dài của chuỗi lớn nhất 255 ký tự (*characters*).
tinyblob	2^8-1	Khai báo cho *Field* chứa kiểu đối tượng nhị phân cở 255 *characters*.
tinytext	2^8-1	Khai báo cho *Field* chứa kiểu chuỗi cở 255 *characters*.
blob	2^{16}-1	Khai báo cho *Field* chứa kiểu *blob* cở 65,535 *characters*..
text	2^{16}-1	Khai báo cho *Field* chứa kiểu chuỗi dạng văn bản cở 65,535 *characters*.
Mediumblob	2^{24}-1	Khai báo cho *Field* chứa kiểu blob vừa khoảng 16,777,215 *characters*.
Mediumtext	2^{24}-1	Khai báo cho *Field* chứa kiểu chuỗi dạng văn bản vừa khoảng 16,777,215 *characters*.
Longblob	2^{32}-1	Khai báo cho *Field* chứa kiểu *blob* lớn khoảng 4,294,967,295 *characters*.
Longtext	2^{32}-1	Khai báo cho *Field* chứa kiểu chuỗi dạng văn bản lớn khoảng 4,294,967,295 *characters*.

5. PHÁT BIỂU SQL

MySQL là một hệ thống quản lý cơ sở dữ liệu quan hệ (*RDBMS*) hay còn được gọi là *Relational Database Management System*. *RDBMS* là một trong những mô hình cơ sở dữ liệu quan hệ thông dụng hiện nay.

5.1. Nhóm phát biểu SQL

Như đã trình bày trong chương 3, 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 *SQL* và *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, ...*

5.2. 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).

5.2.1. Khái niệm cơ bản về 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. Chẳng hạn, như ví dụ 8-1.

Ví dụ 8-1: 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;
```

5.2.2. 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 8-2: 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 mẩu tin  đầu tiên của tất cả các field của tablename*/

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

tablename/*

Ví dụ 8-3: 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
```

5.2.3. Phát biểu SQL dạng 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 8-5: 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');
```

5.2.4. 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 8-6: 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 8-7: 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 8-7: 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 8-8: 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).

5.2.5. SQL dạng 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 như ví dụ 8-9.

Ví dụ 8-9: 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*, bạn tham khảo ví dụ 8-10 được trình bày chi tiết từ ví dụ 4-8 nhưng nhóm mẩu tin bằng mệnh đề *Group By*.

Ví dụ 8-10: 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
----------   ----    -------------------
16            7      2.956.562.368
12            9      3.843.022.604
13            8      145.913.378
10            4      72.382.804
```

5.3. Các hàm thông dụng trong MySQL

5.3.1. 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 8-11: 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
```

5.3.2. 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('Khang')
  ```

  Kết quả trả về như sau:

  ```
  KHANG
  ```

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

  ```
  Select LOWER('Khang')
  ```

  Kết quả trả về như sau:

  ```
  khang
  ```

- 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(' Khang')
  ```

  Kết quả trả về như sau:

  ```
  'khang'
  ```

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

```
Select ltrim('Khang ')
```

Kết quả trả về như sau:

```
'khang'
```

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

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

Kết quả trả về như sau:

```
'Kha'
```

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

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

Kết quả trả về như sau:

```
'Hang'
```

- Hàm *Instr*: Hàm trả về vị trí chuỗi bắt đầu của chuỗi con trong chuỗi xét:

```
Select INSTR ('Khang','Pham Huu Khang')
```

Kết quả trả về như sau:

```
11
```

11 là tương đương vị trí thứ 11 của chữ *Khang* trong chuỗi *"Pham Huu Khang"*

5.3.3. 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 den hom nay:'
from tblOrders
```

Kết quả trả về như sau

```
So ngay giua ngay thu tien den hom nay:
---------------------------------
74
72
```

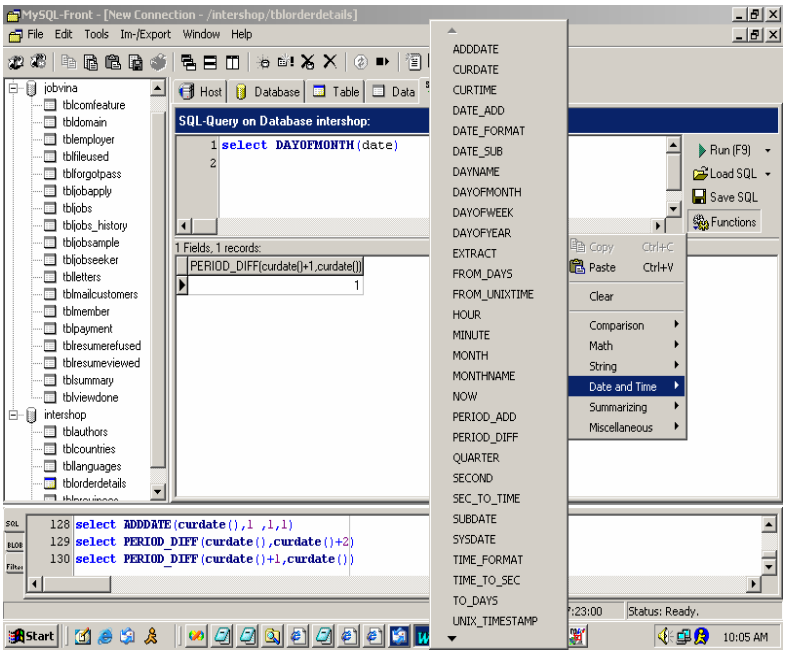
- Hàm *dayofmonth*: Hàm *dayofmonth* trả về ngày thứ mấy trong tháng:

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

Kết quả trả về như sau:

```
21
```

Ngoài các hàm trình bày như trên, bạn có thể tìm thấy nhiều hàm xử lý về thời gian trong phần *Funtions* xuất hiện bên phải màn hình của trình điều khiển như hình 8-6.



Hình 8-6: Sử dụng chức năng Funcitons

5.3.4. 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
```

Để tham khảo thêm một số hàm khác bạn có thể tham khảo trong phần *Functions* như hình 8-9.

5.4. Phát biểu SQL dạng 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 4-11: 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
```

5.5. Phát biểu SQL dạng 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 8-12: 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 8-13: 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 8-14: 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
```

5.6. Phát biểu SQL dạng 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 8-14: 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 8-15: 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
...
...
...
```

5.7. Nhập dữ liệu bằng 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 *SQL Server* 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.

5.7.1. Insert vào bảng lấy giá trị cụ thể:

```
INSERT INTO <Tablename>[<columnname list>]
Values (data_value)
```

Ví dụ 8-16: 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)
```

5.7.2. Insert vào bảng lấy giá trị từ bảng khác:

```
INSERT INTO <Tablename1>[<columnname list>]
Select [columnname list]
From <Tablename2>
Where <Conditions>
```

Ví dụ 8-17: INSERT vào bảng từ giá trị của bảng khác

```
/* Thêm mẩu 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 mẩu 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
```

5.7.3. 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 8-18: INSERT vào bảng từ giá trị cụ thể, bảng khác

/ Thêm mẩu 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 mẩu 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,
```

```
OrderID,
No,
Qtty,
Price,
Discount,
Amount,CurDate()
From tblOrderDetails
ORDER BY OrderID ASC
```

5.8. 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 minh hoạ trong ví dụ 8-18 sau:

Ví dụ 8-18: 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)
```

5.9. 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 đề *WHERE* 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 8-19: 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 8.1 trở về sau hay trong cơ sở dữ liệu *SQL Server* và *Oracle*.

Ví dụ 8-21: Xoá mẩu tin theo quy tắc có ràng buộc quan hệ

```
/* Xoá mẩu tin từ bảng với điều kiện lấy giá trị từ bảng khác */
Delete from tblOrderDetails
where ItemID in
(select ItemID
from tblItems
where ItemName like 'IT%')
```

6. PHÁT BIỂU SQL DẠNG JOIN

Ngoài các phát biểu *SQL* với 4 dạng trên, trong phần kế tiếp, chúng tôi trình bày một số phát biểu *SQL* dạng *Select* để kết nối dữ liệu giữa các bảng có quan hệ với nhau, những phát biểu sẽ trình bày trong chương 5 như:

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

6.1. 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.

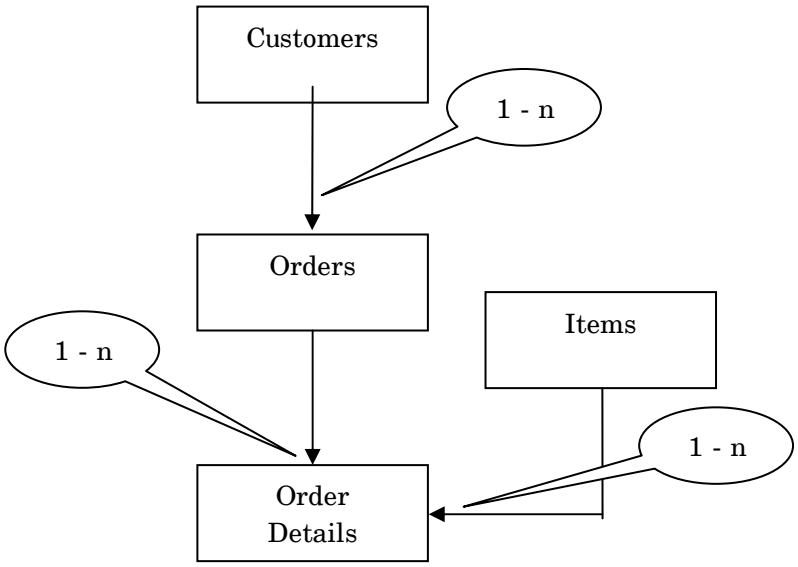
Một khi ứng dụng chạy chậm đi do cơ sở dữ liệu không tối ưu thì rất có thể bạn phải thiết kế và xây dựng lại từ đầu toàn bộ cấu trúc của chương trình và cơ sở dữ liệu.

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 8-1: 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.

6.2. 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
```

Bạn có thể tìm thấy các bảng dữ liệu còn lại trong dữ liệu *Test* trong đĩa đính kèm theo sách.

6.3. 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 8-23: 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 8-24: 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 8-25: 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.

6.4. 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, c*ú 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 8-26: 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
...
```

6.5. 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 8-27: 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
...
```

6.6. 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 8-28: 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 8-29: 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 8-30: 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.

6.7. SQL dạng thay đổi và định nghĩa cơ sở dữ liệu

6.7.1. Phát biểu SQL dạng CREATE

Phát biểu *SQL* dạng *CREATE* dùng để tạo cơ sở dữ liệu và những đối tượng của cơ sở dữ liệu trong *MySQL*, *SQL Server*, *Oracle*, ..., chúng cú pháp như sau:

```
CREATE Database <Database NAME>

CREATE <OBJECT TYPE>
<OBJECT NAME>
```

- *OBJECT TYPE*: Loại đối tượng của cơ sở dữ liệu ví dụ như *Procedure*, *Table*, *View*,...
- *OBJECT NAME:* Tên của đối tượng trong cơ sở dữ liệu *SQL* như *sp_IC*, *tblEmployer*, ...

6.7.2. 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>
```

Cú pháp đầy đủ của phát biểu tạo cơ sở dữ liệu như sau, nếu bạn sử dụng cơ sở dữ liệu *SQL Server*:

```
CREATE DATABASE <database_name>
[ ON [PRIMARY]  (
     [Name= <'Logical file name'>,]     FileName=<'File Name'>
     [, SIZE=<Size in Megabyte or KiloByte> ]
     [, MAXSIZE=<Size in Megabyte or KiloByte> ][, FILEGROWTH = <No of
     Kylobyte|Percentage>]
  )]
```

```
[ LOG ON
  (
    [Name= <'Logical file name'>,]     FileName=<'File Name'>
    [, SIZE=<Size in Megabyte or KiloByte> ]
    [, MAXSIZE=<Size in Megabyte or KiloByte> ][, FILEGROWTH = <No of
    Kylobyte|Percentage>]
  )]
[COLLATE <Collation Name>]
[For Load | For Attach]
```

6.7.3. Diễn giải CREATE Database trong SQL Server

- *ON*: Dùng để định nghĩa nơi chứa cơ sở dữ liệu và không gian chứa tập tin *log*.
- *NAME*: Dùng định nghĩa tên của cơ sở dữ liệu. Tên này dùng tham chiếu khi gọi đến cơ sở dữ liệu, tên được dùng cho quá trình *backup, export, Import, Shrink* cơ sở dữ liệu đó.
- *FILENAME*: Tên tập tin cơ sở dữ liệu lưu trong đĩa cứng, thông thường khi cài *SQL Server* lên ổ đĩa nào thì giá trị mặc định cho phép lưu tập tin đến thư mục đó. Tuy nhiên, nếu muốn bạn cũng có thể thay đổi vị trí các *file* này.

Khi tạo cơ sở dữ liệu, bạn đã định nghĩa vị trí đặt tập tin ở thư mục nào thì không thể di chuyển một cách thủ công (như dùng *Explorer* của *Windows*), vì làm điều đó thật nguy hiểm nhất là khi dữ liệu trong cơ sở dữ liệu đang có giá trị kinh tế.

- *SIZE*: Dung lượng của cơ sở dữ liệu khi khởi tạo chúng. Thông thường giá trị mặc định là *1 MB*.
- Dung lượng phải là số nguyên, có thể tăng thêm bằng cách sử dụng thủ tục *Shrink* trong *SQL Server*.
- *MAXSIZE*: Dung lượng lớn nhất, khi dung lượng cơ sở dữ liệu tăng lên đến mức *MaxSize* thì dừng lại.

Nếu khi dung lượng bằng *MaxSize*, các chuyển tác có thể bị huỷ bỏ hay trả về lỗi không thể thực hiện được, và có thể làm cho cơ sở dữ liệu của bạn bị treo.

Để tránh điều này xảy ra, thì người quản trị cơ sở dữ liệu phải thường xuyên theo giỏi quá trình tăng dung lượng cơ sở dữ liệu theo thời gian, để có biện pháp tránh mọi rủi ro có thể xảy ra.

- *FILEGROWTH*: Dung lượng khởi tạo cùng dung lượng tối đa cho phép tăng trong quá trình thêm dữ liệu vào cơ sở dữ liệu. Nhằm tự động hóa, chúng ta phải thiết lập quá trình tăng tự động theo chỉ số *KB* cho trước hay tỷ lệ phần trăm theo dung lượng đang có.
- *LOG ON*: *Log* on cho phép bạn quản lý những chuyển tác xảy ra trong quá trình sử dụng cơ sở dữ liệu của *SQL Server*.

Xây dựng cơ sở dữ liệu Test

Như đã trình bày ở trên, sau đây ví dụ tạo cơ sở dữ liệu *Test* có cú pháp như sau:

Ví dụ 8-31: Tạo cơ sở dữ liệu Test trong SQL Server

```
USE master

GO

CREATE DATABASE Test
  ON
  ( NAME = Test,
  FILENAME = 'c:\mssql7\data\Testdat.mdf',
```

```
  SIZE = 10,
  MAXSIZE = 50,
  FILEGROWTH = 5 )

  LOG ON
  ( NAME = 'Testlog',
  FILENAME = 'c:\mssql7\data\Testlog.ldf',
  SIZE = 5MB,
  MAXSIZE = 25MB,
  FILEGROWTH = 5MB )
  GO
```

Để đơn giản hoá các đối tượng *Table* trong cơ sở dữ liệu *Test*, chúng tôi chỉ trình bày một vài phát biểu *SQL* dạng *Create Table*, các *Table* khác bạn có thể tìm thấy trong cơ sở dữ liệu đính kèm.

Ví dụ 8-32: Tạo một số bảng trong Test

```
/* 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

6.7.4. 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.

6.7.5. 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
```

6.7.6. 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'

6.7.7. 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.

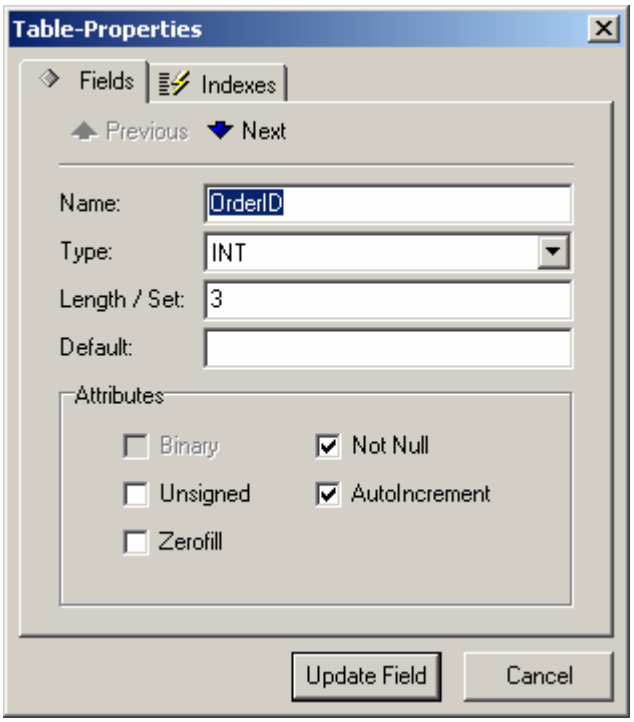
Trong trường hợp, bạn khai báo số tự động trong *SQL Server*, bạn cần phải khai báo thêm các thông số như *seed*. *Seed* là giá trị khởi đầu khi *SQL Server* tự động tăng giá trị, *Increament* là bước tăng, nó cho biết mỗi lần tăng cần bao nhiêu giá trị.

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
```

Trong giao diện đồ họa bạn chỉ cần *check* vào tuỳ chọn *AutoIncreament* như hình 8-10.



Hình 8-10: Chọn auto_increment

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. Trong giao diện đồ họa chỉ cần đánh dấu chọn vào tuỳ chọn *Not NULL* như hình 8-10.

6.8. Thay cấu trúc đối tượng bằng ALTER

Khi chúng ta cần thiết phải sửa đổi một phần cấu trúc của các đối tượng như *table (view*, hay *SP* trong *SQL Server*) vì mục đích nào đó, thì Bạn sử dụng phát biểu *ALTER* để thay đổi cấu trúc của đối tượng hiện có:

```
ALTER <Object type>
<Object Name>
```

Khi một bảng tồn tại trong cơ sở dữ liệu, do nhu cầu cần thiết phải thay đổi cấu trúc bảng, bạn sử dụng phát biểu *ALTER TABLE* cùng các tham số của chúng như cú pháp sau:

```
ALTER TABLE table alteration [,alteration]
```

Chẳng hạn, bạn có thể sử dụng phát biểu *ALTER TABLE* để thêm một cột tên Activate với kiểu dữ liệu TinyInt có giá trị mặc định là 1.

Ví dụ 8-33: Thêm một cột tên Activate vào bảng tblOrders

```
ALTER TABLE tblorders
ADD Activate TINYINT DEFAULT "1"
```

Khi thay đổi thiết lập giá trị mặc định cho cột bạn nên quan tâm đến giá trị mặc định đó có phù hợp cho những mẩu tin đang tồn tại hay không.

Muốn thay đổi giá trị mặc định của cột cho những mẩu tin đang tồn tại, bạn sử dụng đến mệnh đề phụ như trong ví dụ sau:

Ví dụ 8-34: Thiết lập giá trị mặc định trong bảng tblOrders

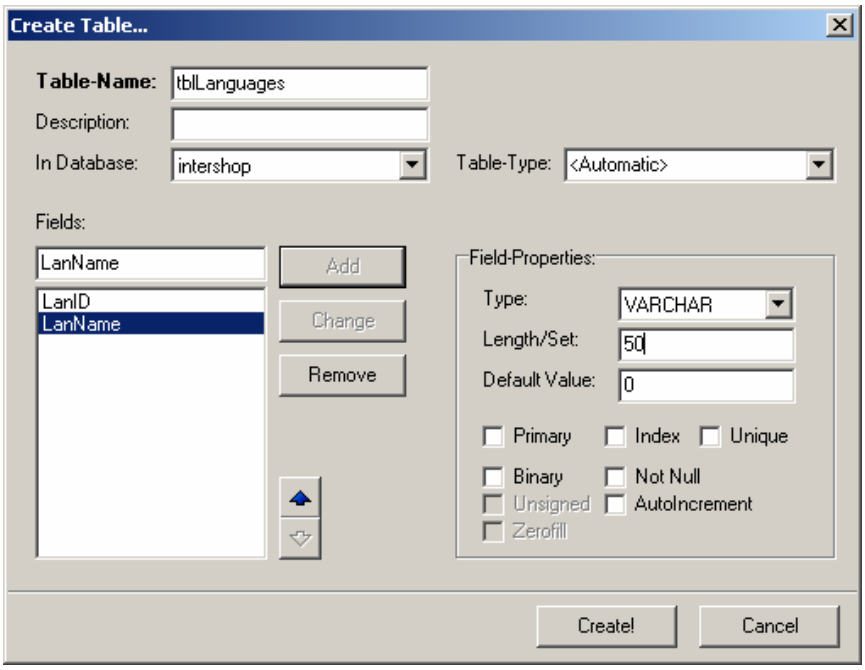
```
ALTER TABLE tblorders
CHANGE OrderDate
OrderDate DATETIME
DEFAULT "0000-00-00"
```

Thay đổi kiểu dữ liệu từ *Date* dang *DateTime*, bạn có thể khai báo như ví dụ 4-35 sau:

Ví dụ 8-35: Thay đổi kiểu dữ liệu

```
ALTER TABLE tblorders
CHANGE OrderDate
OrderDate DATE
DEFAULT "0000-00-00 00:00:00"
```

Mặc khác, bạn cũng có thể tạo hay thay đổi bảng trong màn hình *MySQL-Front*. Chỉ cần chọn ngăn *Database* | *R-Click* | *Create New Table*, cửa sổ xuất hiện như hình 8-11.



Hình 8-11: Giao diện tạo bảng bằng MySQL-Front

6.9. Phát biểu SQL dạng DROP

Drop là phát biểu thực hiện phép xoá. *DROP* dùng để xoá đối tượng của cơ sở dữ liệu như bảng, cơ sở dữ liệu, ...Cú pháp của phát biểu *DROP:*

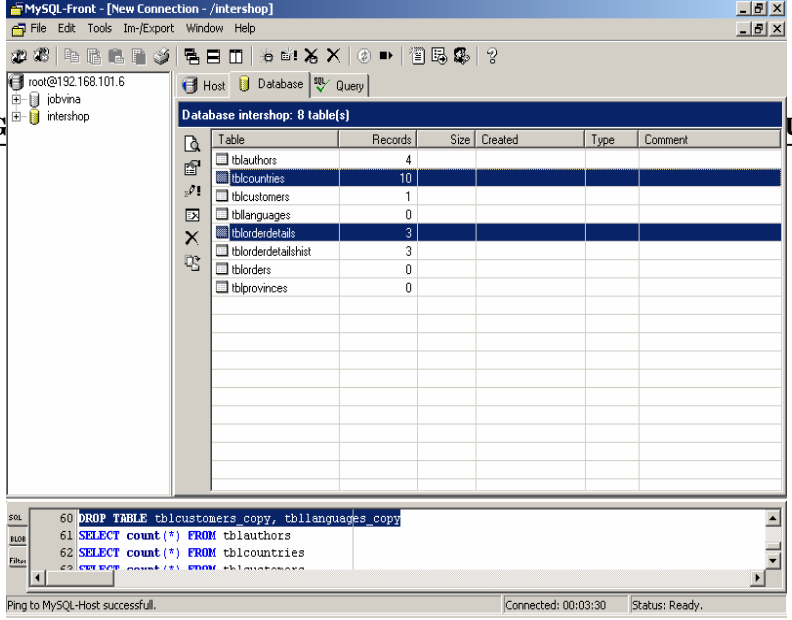
```
DROP <Object type> <Object name> [, .... n]
```

Bạn có thể xoá cơ sở dữ liệu, bằng cách khai báo như sau:

```
Drop Database Test
```

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

Ngoài ra, bạn cũng có thể dùng *MySQl-Front* để xoá bảng hay các đối tượng *Table* trong cơ sở dữ liệu chỉ định. Nếu chọn nhiều bảng cùng một lúc bạn sử dụng phím *Control* hay *Shift* như sau:



Hình 8-12: Chọn đối tượng để xoá bảng trong MySQL-Front

7. TAO KICH BẢN SQL- SQL SCRIPTS

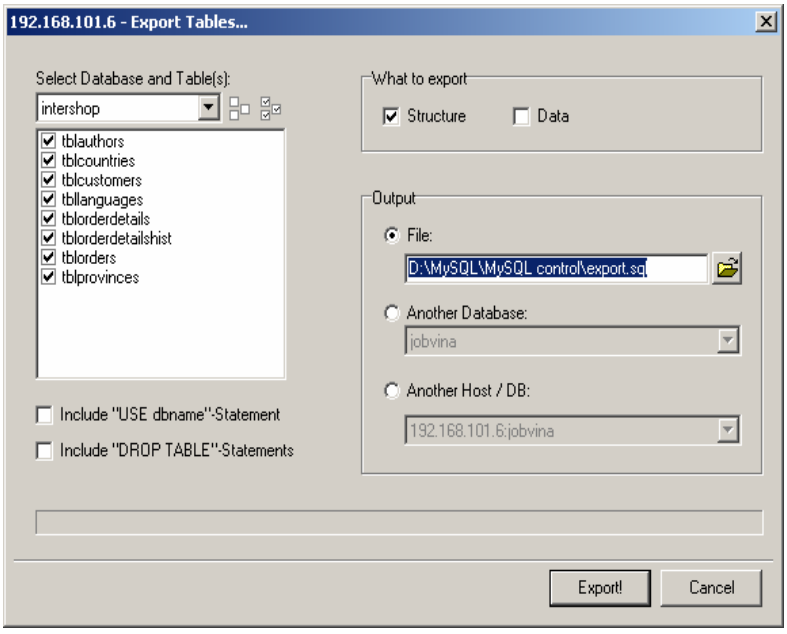
Thông thường khi xây dựng cơ sở dữ liệu để phát triển ứng dụng, đôi khi bạn cần chuyển cơ sở dữ liệu từ máy này sang máy khác, hay từ khu vực này hay đến khu vực khác.

Có rất nhiều cách để làm điều này, ở đây chúng tôi giới thiệu đến các bạn một công cụ tái tạo lại cơ sở dữ liệu mới từ kịch bản của cơ sở dữ liệu gốc.

Kịch bản *SQL* (*SQL Script*) là tổng hợp tất cả các phát biểu *SQL* dùng để tạo ra cơ sở dữ liệu trong quá trình xây dựng chúng, chúng lưu trữ dưới dạng văn bản có tên mở rộng *.sql* (*cautruc.sql*).

Công cụ này tạo kịch bản cho tất cả các đối tượng của cơ sở dữ liệu với những thuộc tính căn bản. Tuy nhiên, nếu bạn chọn vào tuỳ chọn *Data*, *SQL Script* bao gồm các phát biểu *SQL* dạng *Insert* cùng với dữ liệu trong bảng.

Trước tiên bạn có thể nhận thấy cửa sổ công cụ này trong MySQL-Front, bằng cách chọn tên cơ sở dữ liệu *Test*, sau đó chọn *Tools* / *Im-Export* / *Export Table*, cửa sổ xuất hiện như hình 8-13 sau:



Hình 8-13: Tạo kịch bản trong MySQl-Front

KẾT CHƯƠNG

Trong chương này, chúng tôi đã giới thiệu với bạn hầu hết các phát biểu *SQL* thuộc loại định nghĩa cơ sở dữ liệu, thao tác dữ liệu như *Select*, *Insert*, *Delete* và *Update*.

Phát biểu *SQL* dạng *Select* với các mệnh đề như *JOIN* cùng phép toán giữa hai hay nhiều bảng trong phát biểu *SQL* dạng *SELECT*.

Ngoài ra, chúng tôi cũng trình bày hai loại phát biểu *SQL* dạng định nghĩa và thay đổ cơ sở dữ liệu tạo như *CREATE* và *ALTER, DROP*.

Môn học: PHP

BÀI 9: PHP VÀ DATABASE

Để kết nối cơ sở dữ liệu mySQL trong PHP, chúng ta có nhiều cách ứng với nhiều phương thức kết nối cơ sở dữ liệu, trong phần này chúng ta tập trung tìm hiểu cách kết nối cơ sở dữ liệu mySQL từ PHP bằng chính gói của nó.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ Khai báo kết nối cơ sở dữ liệu
- ✓ Thêm mẩu tin
- ✓ Cập nhật mẩu tin.
- ✓ Xoá mẩu tin
- ✓ Truy vấn dữ liệu

1. KẾT NỐI CƠ SỞ DỮ LIỆU

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("TestDB", $link);
?>
```

Trong đó khai báo sau là kết nối cơ sở dữ liệu mySQL với tên server/ip cùng với username vá password:

```
mysql_connect ("localhost", "root", "")
```

Và mysql_select_db("TestDB", $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 tìh 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.

```
<HTML>
   <HEAD>
   <TITLE>::Welcome to PHP and mySQL</TITLE>
   </HEAD>
   <BODY>
   Mo va dong ket noi CSDL MySQL
   <?php
      $link = mysql_connect ("localhost", "root", "")
      or die ("Could not connect to MySQL Database");
      mysql_select_db("InterShop", $link);
      mysql_close($link);
```

```
?>
</BODY>
</HTML>
```

2. THÊM MẨU TIN

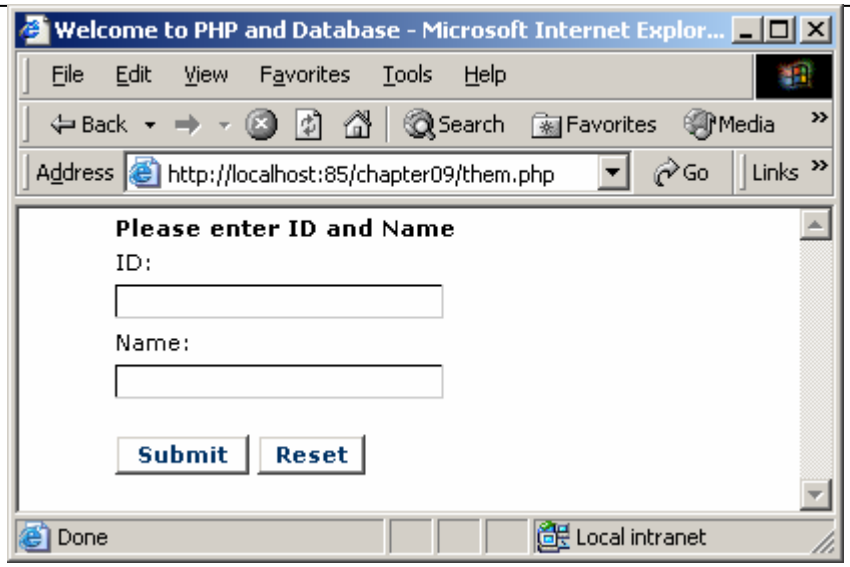
Để thêm mẩu tin, bạn sử dụng hàm mysql_query(chuỗi Insert). Chẳng hạn, chúng ta khai báo trang insert.php để thêm mẩu tin vào bảng tblships có hai cột dữ liệu là ShipID và ShipName như ví dụ trong trang insert.php.

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP and mySQL</TITLE>
</HEAD>
<BODY>
<h3>Them mau tin</h3>
<?php

     require("dbcon.php");
     $sql="insert into tblships values('A01','Testing')";
     $result = mysql_query($sql,$link);
     $affectrow=0;
     if($result)
      $affectrow=mysql_affected_rows();
     mysql_close($link);
?>
So mau tin them vao<?= $affectrow?>
</BODY>
</HTML>
```

Trong đó, bạn sử dụng hàm mysql_query với hai tham số là $sql và $link. Kết quả trả về là số mẩu tin thực thi. Ngoài ra, bạn có thể sử dụng đoạn kết nối cơ sở dữ liệu trong tập tin dbcon.php như ví dụ sau:

```
<?php
   $link = mysql_connect ("localhost", "root", "")
   or die ("Could not connect to MySQL Database");
   mysql_select_db("Test", $link);
?>
```

Trong trường hợp cho phép người sử dụng thêm mẩu tin thì bạn thiết kế form yêu cầu người sử dụng nhập hai giá trị sau đó submit đến trang kế tiếp để thực thi việc thêm gt sau đó submit đến trang kế tiếp để thực thi việc thêm giá trị vừa nhập vào cơ sở dữ liệu như hình 9-1.



Hình 9-1: Thêm mẩu tin

Để làm điều này, trước tiên bạn khai báo trang them.php, trong đó khai báo đoạn javascript để kiểm tra dữ liệu nhập như sau:

```
<SCRIPT language=JavaScript>
     function checkInput()
       {

        if (document.frmPHP.txtID.value=="")
        {
          alert("Invalid ID, Please enter ID");
          document.frmPHP.txtID.focus();
          return false;
        }

        if (document.frmPHP.txtName.value=="")
        {
          alert("Please enter Name");
          document.frmPHP.txtName.focus();
          return false;
        }
        return true;
       }
</script>
```

Kế đến khai báo thể form và hai thẻ input lại text yêu cầu người sử dụng nnập ID và Name như sau:

```
<form name="frmPHP" method="post"
     action="doinsert.php"
     onsubmit="return checkInput();">
  <tr>
   <td align="left" class="content-sm"><b>
    Please enter ID and Name
    </b></td>
  </tr>
  <tr>
   <td align="left" >ID:</td>
  </tr>
  <tr>
   <td align="left">
    <input type="text" name="txtID"
     size="25" maxlength="3" class="textbox">
   </td>
```

```
  </tr>
  <tr>
   <td align="left" >Name:</td>
  </tr>
  <tr>
   <td align="left" >
    <input type="text" name="txtName"
     size="25" maxlength="50" class="textbox">
   </td>
  </tr>
  <tr>
   <td align="left"  valign="top"> <br>
    <input type="submit"
     value="Submit" class="button">
    <input type="reset" value="Reset" class="button">
   </td>
  </tr>
 </form>
```

Lưu ý rằng, bạn khai báo số ký tự lớn nhất cho phép nhập bằng với kích thước đã khai báo trong cơ sở dữ liệu ứng với thuộc tính maxlength.

Khi người sử dụng nhập hai giá trị và nhấn nút submit, trang kế tiếp đực triệu gọi. Trang này lấy giá trị nhập bằng cách sử dụng biến form hay $HTTP_POST_VARS. Đối với trường hợp này chúng ta sử dụng biến form như trang doinsert.php.

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP and mySQL</TITLE>
</HEAD>
<BODY>
<h3>Them mau tin</h3>
<?php
     $affectrow=0;
     require("dbcon.php");
     $sql="insert into tblships(ShipID,ShipName) ";
     $sql .=" values('".$txtID."','".$txtName."')";
     $result = mysql_query($sql,$link);
     if($result)
      $affectrow=mysql_affected_rows();
     mysql_close($link);
?>
So mau tin them vao<?= $affectrow?>
</BODY>
</HTML>
```

3. CẬP NHẬT MẨU TIN

Đối với trường hợp cập nhật mẩu tin, bạn cũng sử dụng hàm mysql_query với phát biểu Update thay ví Insert như trên, ví dụ chúng ta khai báo trang update.php để cập nhật mẩu tin trong bảng tblShips với tên là UpdateTesting khi mã có giá trị là A01.

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP and mySQL</TITLE>
</HEAD>
<BODY>
<h3>Cap nhat mau tin</h3>
<?php

     require("dbcon.php");
     $sql="Update tblships set ShipName='UpdateTesting' ";
     $sql.=" where ShipID='A01'";
     $result = mysql_query($sql,$link);
     $affectrow=0;
     if($result)
```

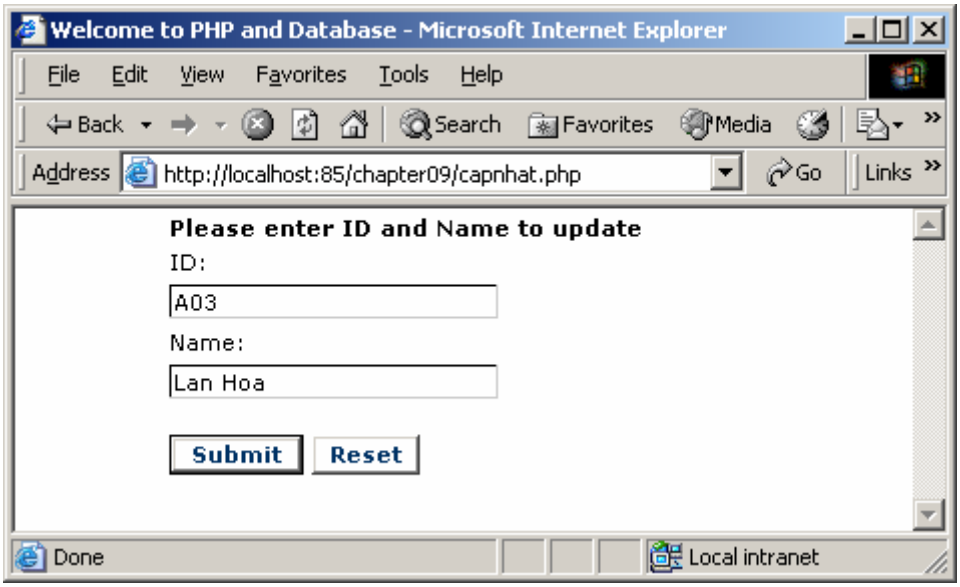
```
        $affectrow=mysql_affected_rows();
      mysql_close($link);
?>
So mau tin cap nhat <?= $affectrow?>
</BODY>
</HTML>
```

Lưu ý rằng, để biết số mẩu tin đã thực thi bởi phát biểu SQL bạn sử dụng hàm mysql_affected_rows.

```
if($result)
            $affectrow=mysql_affected_rows();
```

Tương tự như trên, bạn có thể thiết kế form cho phép người sử dụng cập nhật dữ liệu bằng cách thiết kế form yêu cầu người sử dụng nhập mã và tên cập nhận.

Trước tiên thiết kế from cho phép nhập dữ liệu để cập nhật như ví dụ trang capnhat.php, sau khi học phần truy vấn xong, thaqy vì nhập mã bạn cho phép người sử dụng chọn trong danh sách đã có như hình 9-2.



Hình 9-2: Cập nhật dữ liệu

Sau khi người sử dụng nhấn nút submit, trang doupdate.php sẽ triệu gọi, kết quả trả về 1 hay 0 mẩu tin.

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP and mySQL</TITLE>
</HEAD>
<BODY>
<h3>Cap nhat mau tin</h3>
<?php
      $affectrow=0;
      require("dbcon.php");
      $sql="update tblships set ShipName='";
      $sql .=$txtName."' where ShipID='".$txtID."'";
      $result = mysql_query($sql,$link);
      if($result)
        $affectrow=mysql_affected_rows();
      mysql_close($link);
?>
```

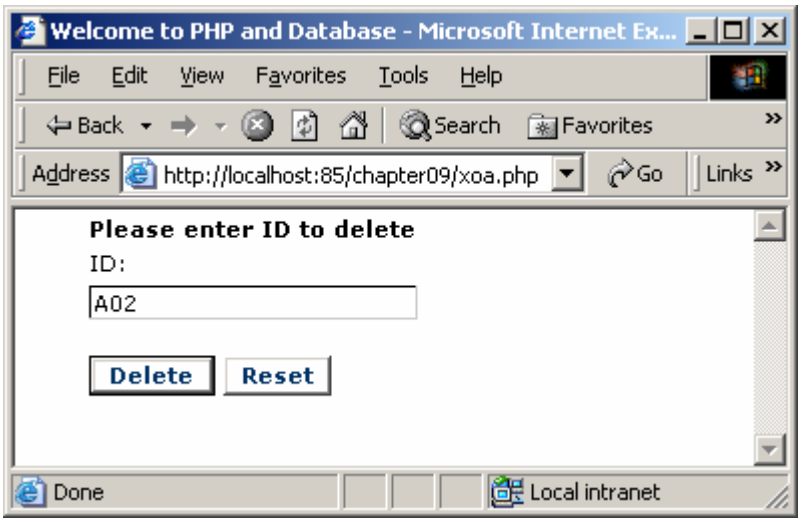
```
So mau tin cap nhat <?= $affectrow?>
</BODY>
</HTML>
```

4. XOÁ MẨU TIN

Tương tự như vậy khi xoá mẩu tin, bạn chỉ thay đổi phát biểu SQL dạng Delete như ví dụ trong tập tin delete.php.

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP and mySQL</TITLE>
</HEAD>
<BODY>
<h3>Xoa mau tin</h3>
<?php

     require("dbcon.php");
     $sql="Delete From tblships where ShipID='A01'";
     $result = mysql_query($sql,$link);
     $affectrow=0;
     if($result)
      $affectrow=mysql_affected_rows();
     mysql_close($link);
?>
So mau tin da xoa <?= $affectrow?>
</BODY>
</HTML>
```

Đối với trường hợp xoá thì đơn giản hơn, bạn cỉh cần biết được mã cần xoá, chính vì vậy trong trường hợp này chúng ta chỉ cần thiết kế trang cho phép nhập mã như hình 9-3.



Hình 9-3: Xoá 1 mẩu tin

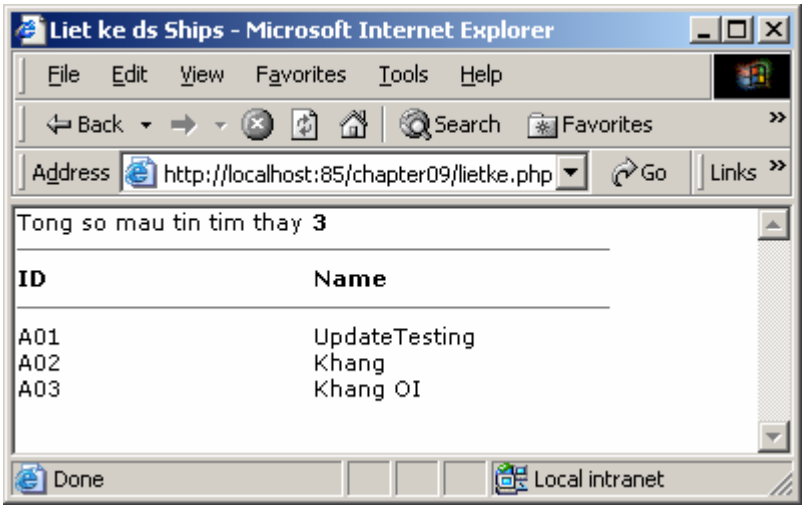
Sau khi nhập mã cần xoá, nếu người sử dụng nhấn nút Delete lập tức trang dodelete.php sẽ triệu gọi và xoá mẩu tin tương ứng.

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP and mySQL</TITLE>
</HEAD>
```

```
<BODY>
<h3>Xoa mau tin</h3>
<?php
      $affectrow=0;
      require("dbcon.php");
      $sql="delete from tblships ";
      $sql .=" where ShipID='".$txtID."'";
      $result = mysql_query($sql,$link);
      if($result)
       $affectrow=mysql_affected_rows();
      mysql_close($link);
?>
So mau tin xoa <?= $affectrow?>
</BODY>
</HTML>
```

5. TRUY VẤN DỮ LIỆU

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. Chẳng hạn, chúng ta tạo một tập tin lietke.php dùng để liệt kê danh sách mẩu tin trong bảng tblShips như hình 9-4.



Hình 9-4: Liệt kê mẩu tin

Để làm điều này, bạn khai báo đoạn chương trình đọc bảng dữ liệu tương tự như ví dụ sau:

```
<?php
      require("dbcon.php");
      $totalRows = 0;
      $stSQL ="select * from tblShips";
      $result = mysql_query($stSQL, $link);
      $totalRows=mysql_num_rows($result);
?>
```

Sau đó, dùng hàm mysql_fetch_array để đọc từng mẩu tin và in ra như sau:

```
<?php
if($totalRows>0)
      {
       $i=0;
       while ($row = mysql_fetch_array ($result))
       {
      $i+=1;
```

```
        ?>
      <tr valign="top">
       <td>
        <?=$row["ShipID"]?> </td>
       <td ><?=$row["ShipName"]?></td>
      </tr>
```

Trong trường hợp số mẩu tin trả về là 0 thì in ra câu thông báo không tìm thấy như sau:

```
<?php
     }
}else{
     ?>
        <tr valign="top">
         <td > </td>
         <td > <b><font face="Arial" color="#FF0000">
          Oop! Ship not found!</font></b></td>
        </tr>
        <?php
      }
?>
```

6. KẾT LUÂN

Trong bài này, chúng ta tập trung tìm hiểu cách kết nối cơ sở dữ liệu, thêm, xoá cập nhật và liệt kê mẩu tin. Trong bài kế tiếp chúng ta tìm hiểu nhiều các trình bày dữ liệu, xoá mẩu tin theo dạng mảng.

Môn học: PHP

BÀI 10: XOÁ, CẬP NHẬT DỮ LIỆU DẠNG MẢNG

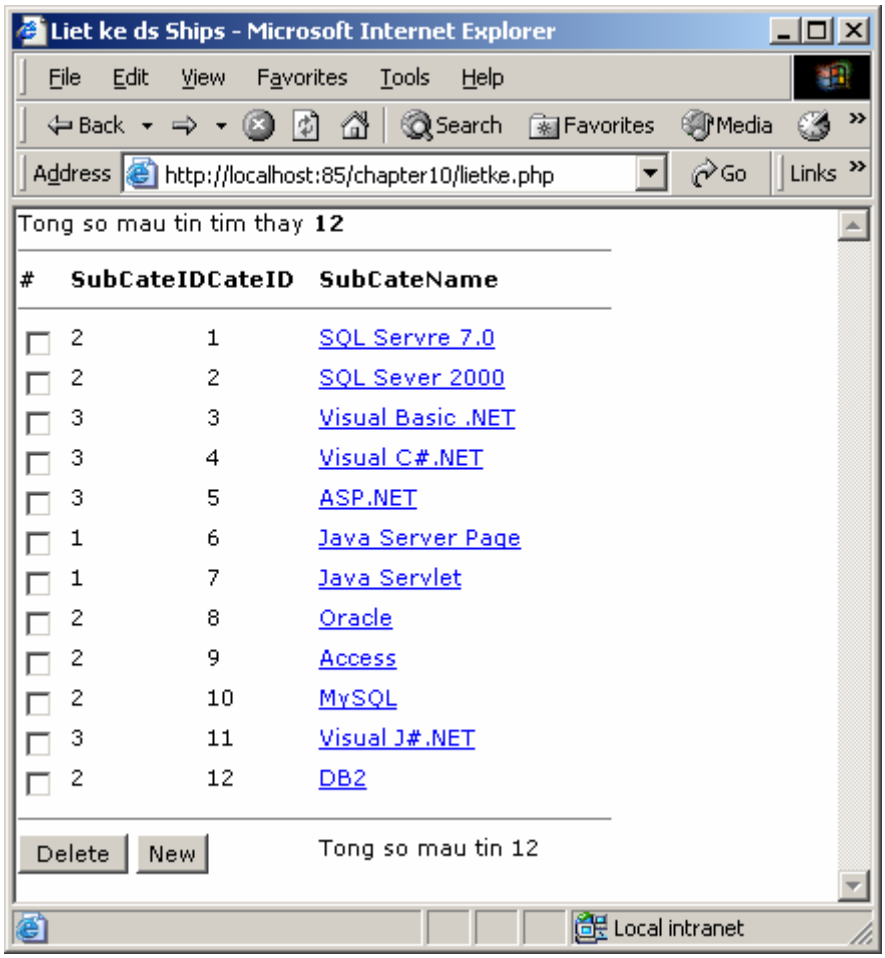
Trong bài trước chúng ta đã làm quen với cách xoá mẩu tin trong cơ sở dữ liệu mySQL. Đối với trường hợp xoá một lúc nhiều mẩu tin, chúng ta phải xây dựng trang PHP có sử dụng thẻ input dạng checkbox.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ Liệt kê dữ liệu dạng danh sách
- ✓ Xoá nhiều mẩu tin
- ✓ Cập nhật nhiều mẩu tin

1. LIỆT KÊ DỮ LIỆU

Để xoá nhiều mẩu tin cùng một lúc, trước tiên bạn khai báo trang PHP để liệt kê danh sách mẩu tin trong mảng dữ liệu chẳng hạn, mỗi mẩu tin xuất hiện một checkbox tương ứng. Checkbox này có giá trị là mã nhận dạng của mẩu tin đó. Trong trường hợp này chúng ta dùng cột khoá của mã chuyển hàng (SubCateID) trong bảng tblSubCategories định nghĩa trong trang lietke.php như hình 10-1.



Hình 10-1: Liệt kê danh sách lại sản phẩm

Để cho phép lấy được nhiều giá trị chọn của sản phẩm như hình tren, bạn khai báo các checkbox này cùng tên (giả sử tên là chkid) và giá trị trị là SubCateID của mỗi sản phẩm như ví dụ 10-1 trong trang lietke.php.

```
<?php
  if($totalRows>0)
    {
     $i=0;
      while ($row = mysql_fetch_array ($result))
      {
    $i+=1;
     ?>
  <tr valign="top">
  <td><input type=checkbox name=chkid
    value="<?=$row["SubCateID"]?>"> </td>
  <td><?=$row["CateID"]?> </td>
        <td><?=$row["SubCateID"]?> </td>
        <td ><a href="capnhat.php?id=<?=$row["SubCateID"]?>">
    <?=$row["SubCateName"]?></a></td>

       </tr>
       <?php
    }
  ?>
  <tr valign="top">
        <td colspan="4" align="middle">
            <hr noshade size="1">
     </td>
  </tr>
  <tr valign="top">
     <td colspan=3><input type=submit value="Delete">
    <input type=hidden name=from_ value="subcategories">
    <input type=hidden name=type value="0">
    <input type=hidden name=chon value="">
    <input type=button value="New"
    onclick="window.open('them.php',target='_main')"></td>
        <td >Tong so mau tin <?=$i?></td>
       </tr>
  <?php
  }else{
  ?>
       <tr valign="top">
        <td > </td><td > </td><td > </td>
        <td > <b><font face="Arial" color="#FF0000">
         Oop! Ship not found!</font></b></td>
       </tr>
       <?php
   }
    ?>
```

Trong đó, hai khai báo sau:

```
<input type=hidden name=from_ value="subcategories">
<input type=hidden name=type value="0">
<input type=hidden name=chon value="">
```

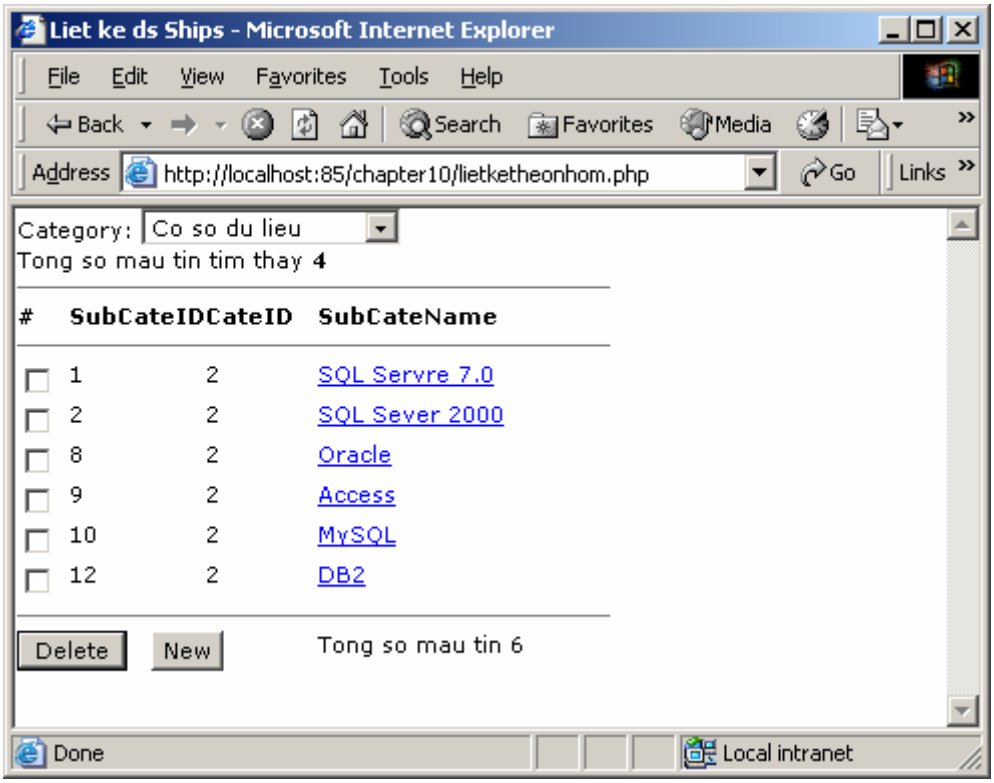
Cho biết bạn submit từ trang nào và loại xoá nhiều mẩu tin hay một mẩu tin đối với bảng tương ứng. Mục đích của vấn đề này là trang delete sử dụng chung cho nhiều bảng khác nhau và từ trang liệt kê (xoá nhiều) hoặc từ trang edit (1 mẩu tin cụ thể).

Ngoài ra, chúng ta khai báo <input type=hidden name=chon value=""> để nhận giá trị chọn trên cách checkbox bằng cách khai báo đoạn javascript như sau:

```
<script>
     function calculatechon()
```

```
        {
          var strchon="";
          var alen=document.frmList.elements.length;
          var buttons=1;

          alen=(alen>buttons)?document.frmList.chkid.length:0;
          if (alen>0)
          {
            for(var i=0;i<alen;i++)
            if(document.frmList.chkid[i].checked==true)
                        strchon+=document.frmList.chkid[i].value+",";
          }else
          {
            if(document.frmList.chkid.checked==true)
                        strchon=document.frmList.chkid.value;
          }
          document.frmList.chon.value=strchon;
          return isok();
        }

</script>
```

Tuy nhiên, do nhiều loại sản phẩm thuộc các nhóm sản phẩm khác nhau, chính vì vậy bạn khai báo danh sách nhóm sản phẩm trên thẻ select cho phép người sử dụng liệt kê sách theo nhóm sản phẩm như hình 10-2.



Hình 10-2: Liệt kê danh sách loại sách

Để liệt kê danh sách nhóm trong bảng tblCategories, bằng cách khai báo phương thức nhận chuỗi SQL dạng Select và giá trị mặc định trả về nhiều phần tử thẻ option trong tập tin database.php như ví dụ 10-2.

```
function optionselected($stSQL,$item,$links)
{
     $results = mysql_query($stSQL, $links);
        $totalRows=mysql_num_rows($results);
     $strOption="<option value=\"\" selected>";
     $strOption .="--Select--</option>";
     if($totalRows>0)
     {
      while ($row = mysql_fetch_array ($results))
       {
        $strOption .="<option value=\"" ;
        $strOption .=$row["ID"]."\"";
        if($row["ID"]==$item)
       $strOption .=" selected ";
        $strOption .= ">".$row["Name"];
        $strOption .="</option>";
       }
     }
     return $strOption;
}
```

Sau đó, gọi phương thức này trong trang lietketheonhom.php như ví dụ 10-3.

```
<?php
  require("dbcon.php");
  require("database.php");
  $id="";
  if (isset($cateid))
    $id=$cateid;
  $stSQL ="select CateID As ID, CateName as Name from tblCategories ";
  $result = mysql_query($stSQL, $link);
  $totalRows=mysql_num_rows($result);
  $strOption=optionselected($stSQL,$id,$link);
?>
<form name=frmMain method=post>
  <tr>
       <td align=left colspan=4>
    Category: <select name=cateid onchange="document.frmMain.submit();">

    <?=$strOption?>
    </select></td>
  <td align=right> </td>
  </tr>
</form>
```

Lần đầu tiên bạn có thể chọn mặc định một nhóm hoặc liệt kê tất cả, khi người sử dụng chọn nhóm sản phẩm nào đó thì trang lietketheonhom.php sẽ liệt kê danh sách loại sách của nhóm sách đó. Để làm điều này, bạn khai báo thẻ form với thẻ select như ví dụ 10-4.

```
<form name=frmMain method=post>
  <tr>
       <td align=left colspan=4>
    Category: <select name=cateid onchange="document.frmMain.submit();">

    <?=$strOption?>
    </select></td>
  <td align=right> </td>
  </tr>
</form>
```

Khi người sử dụng chọn các mẩu tin như hình 10-2 và nhấn nút Delete, dựa vào giá trị của nút có tên action (trong trường hợp này là Delete), bạn có thể khai báo biến để lấy giá trị chọn bằng cách khai báo như ví dụ 10-5.

```
$strid=$chon;
```

```
$strid=str_replace(",","',"',$strid);
```

Dựa vào thẻ hidden khai báo trong các trang trình bày danh sách (chẳng ạn lietketheonhom.php) mẩu tin như sau:

```
<input name="from" type=hidden value="subcategories">
```

Bạn có thể biết từ trang nào gọi đến trang dodelete.php để quay trở về khi thực hiện xong tác vụ xử lý.

Ngoài ra, dựa vào giá trị của nút action để thực hiện phát biểu SQL. Chẳng hạn, trong trường hợp này nếu người sử dụng nhấn hút Delete thì bạn khai báo như ví dụ 10-6 sau:

```
switch($strfrom)
   {
   case "subcategories":
      $stSQL ="delete from tblsubcategories where SubCateID in('".$strid."')";
      $strlocation="Location:lietketheonhom.php";
      break;
   case "categories":
      $stSQL ="delete from tblcategories where CateID in('".$strid."')";
      $strlocation="Location:nhom.php";
      break;
   }
```

Sau đó, bạn có thể thực thi phát biểu SQL vừa khai báo ở trên như ví dụ 10-7.

```
if($stSQL!="")
{
   $result = mysql_query($stSQL, $link);
}
```

Lưu ý rằng, bạn cũng nên khai báo try catch trong khi làm việc với cơ sở dữ liệu. Ngoài ra, bạn cũng phải xác nhận trước khi thực thi hành động xoá mẩu tin chọn bằng cách khai báo đoạn Javascript như sau:

```
<script>
function isok()
 {
  return confirm('Are you sure to delete?');
 }
</script>
```

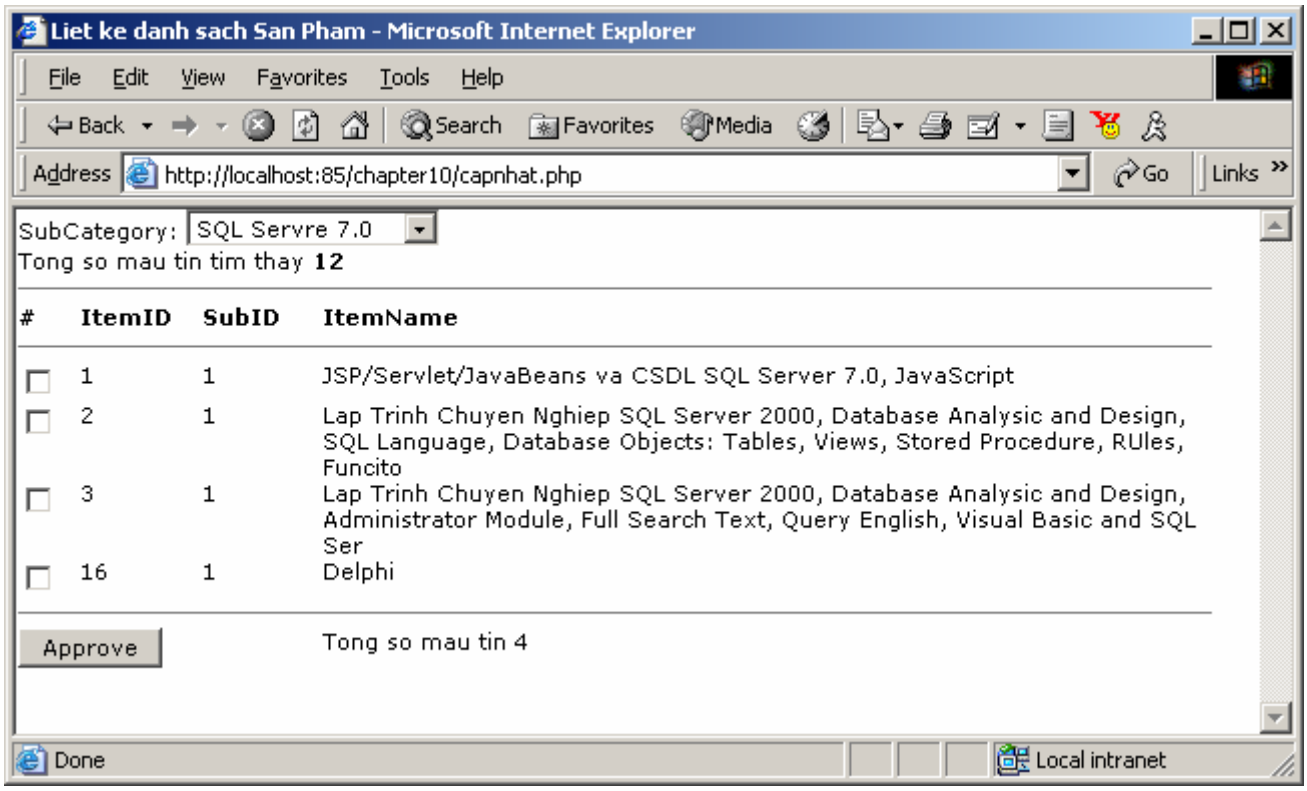
Sau đó gọi trong biến cố onsubmit của form như sau:

```
<form action=dosql.php method=post onsubmit="return calculatechon();">
```

2. CẬP NHẬT NHIỀU MẨU TIN

Tương tự như trường hợp Delete, khi bạn duyệt (approval) một số mẩu tin theo một cột dữ liệu nào đó, chẳng hạn, trong trường hợp này chúng ta cho phép sử dụng những sản phẩm đã qua sự đồng ý của nhà quản lý thì cột dữ liệu Activate của bảng tbltems có giá trị là 1.

Để làm điều này, trước tiên bạn liệt kê danh sách sản phẩm như hình 10-3.



Hình 10-3: Liệt kê danh sách sản phẩm duyệt hay chưa

Tương tự như trong trường hợp delete, bạn khai báo trang doUpdate như sau:

```
<HTML>
<HEAD>
<TITLE>::Welcome to PHP and mySQL</TITLE>
</HEAD>
<BODY>
<h3>Cap nhat mau tin</h3>
<?php
require("dbcon.php");
$strid=$chon;
$strid=str_replace(",","','",$strid);
$strfrom="";
if(isset($from_))
     {
        $strfrom=$HTTP_POST_VARS{"from_"};
     }
$strtype="";
if(isset($type))
     {
        $strtype=$HTTP_POST_VARS{"type"};
     }

$stSQL="";
if($strfrom<>"")
{
     switch($strfrom)
        {
        case "items":
           $stSQL ="update tblItems set Activate=1 where ItemID
in('".$strid."')";
```

```
            break;
          }
        if($stSQL!="")
        {
          $result = mysql_query($stSQL, $link);
          if($result)
             $affectrow=mysql_affected_rows();
          mysql_close($link);
        }
}
?>
So mau tin cap nhat <?= $affectrow?>
</BODY>
</HTML>
```

3. KẾT LUẬN

Trong bài này, chúng ta tìm hiểu chức năng xoá, cập nhật nhiều mẩu tin bằng cách sử dụng thẻ input loại checkbox cùng tên và khác giá trị, bài kế tiếp chúng ta tiếp tục tìm hiểu về chức năng đăng nhập trong PHP.

Tự học Matlab

Tự học lập trình web bằng ngôn ngữ PHP

MỤC LỤC

X

Hồ Diên Lợi

Chương 1: Quy trình thiết kế website

I. Các khái niệm cơ bản

1. HTML (Hypertext Markup Language) – Ngôn ngữ đánh dấu siêu văn bản

HTML là ngôn ngữ đánh dấu được sử dụng để tạo nên các trang Web, nó chứa các trang văn bản và những thẻ (tag) định dạng cho trình duyệt Web (web brower) biết làm thế nào để thể hiện các thông tin trên World Wide Web(WWW). HTML giờ đây trở thành một chuẩn Internet do tổ chức World Wide Web Consortium (W3C) duy trì. Phiên bản mới nhất của HTML là 4.01. Tuy nhiên, hiên hay HTML không còn được phát triển tiếp, nó được thay thế bằng XHTML.

2. Ngôn ngữ lập trình Web

Ngôn ngữ lập trình Web là ngôn ngữ lập trình (khác với ngôn ngữ HTML- ngôn ngữ đánh dấu siêu văn bản) được sử dụng để hỗ trợ và tăng cường các khả năng của ứng dụng Web, giúp cho việc điều khiển các phần tử của trang Web dễ dàng hơn.

Một số ngôn ngữ lập trình Web thường được dùng là: ASP, ASP.Net, PHP, JSP…

3. WebServer – trình chủ Web

WebServer là máy tính mà trên đó cài đặt các phần mềm phục vụ Web, và khi có phần mềm đó cũng được xem như một WebServer.

Tất cả các WebServer đều có thể biên dịch và chạy các file *.html và *.htm, tuy nhiên các WebServer lại phục vụ một số kiểu file riêng biệt, ví dụ như IIS của Microsoft dành riêng cho các file *.asp, *.aspx; Apache dành cho các file *.PHP; Sun Java System web server của SUN dành riêng cho các file *.jsp.

4. Database server – Trình chủ CSDL

Database server là máy tính mà trên đó có cài đặt một hệ quản trị CSDL (HQTCSDL) nào đó, ví dụ như SQL Server, MySQL, Oracle…

5. Web browser-Trình duyệt Web

Trình duyệt Web là một ứng dụng tương ứng với máy tính của người dùng, cho phép người dùng cập nhật và xem thông tin trên các trang Web. Các trình duyệt Web thông dụng hiện nay là: Internet Explorer, Netscape, FireFox, Opera, Safari…

6. URL (Uniform Resource Locator)- Tài nguyên trên Internet

URL là tài nguyên trên Internet. Sức mạnh của Web là khả năng tạo ra các liên kết siêu văn bản đến các thông tin có liên quan. Những thông tin này có thể là những trang web khác, hình ảnh, âm thanh…

Những liên kết này thường được biểu diễn bằng những chữ màu xanh có gạch dưới.

Các URL có thể truy xuất thông qua một trình duyệt (browser).

Ví dụ 1.1: Một URL có dạng http://www.hutc.edu.vn/index.html

Trong đó:

+ http: là giao thức

+ http://www.hutc.edu.vn/ là địa chỉ của máy chứa tài nguyên.

+ index.html là tên đường dẫn trên máy chứa tài nguyên.

Nhờ địa chỉ URL mà chúng ta có thể truy cập tới các trang web khác nhau.

7. HTTP (Hypertext Transfer Protocol)- Giao thức truyền siêu văn bản

HTTP là một trong các giao thức chuẩn về mạng Internet, được dùng để trao đổi thông tin giữa WebServer và người dùng (WebClient) thông qua mạng máy tính.

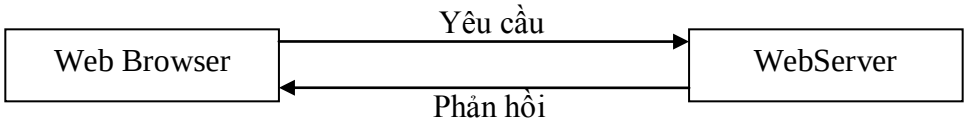
HTTP được sử dụng thông qua URL, với cấu trúc chuỗi có định dạng như sau:

http://<host>[:<port>][<path>[?query]]

8. Cơ chế Web

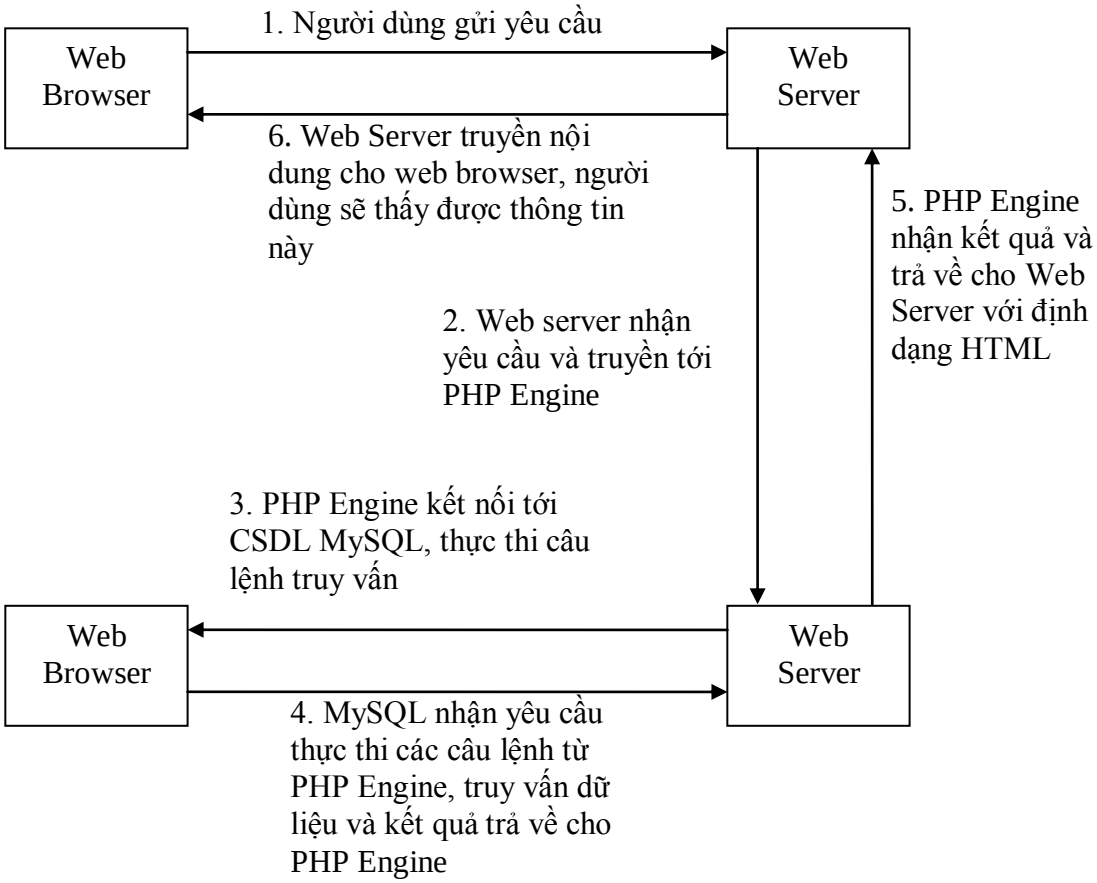
Cơ chế Web là cơ chế tương tác giữa người dùng – thông qua trình duyệt Web và WebServer.

+ Cơ chế tương tác từ người dùng với WebServer



Người dùng sẽ thông qua Web Browser để gửi yêu cầu tới WebServer và ngược lại Web Browser sẽ nhận phản hồi thông qua Web Browser đến người dùng.

+ Quy trình làm việc của PHP và MySQL thông qua Web Server



II. Quy trình thiết kế website

1. Xác định mục đích, yêu cầu của website

Để xây dụng một website có chất lượng và hiệu quả, trước tiên chúng ta cần xác định: Website dùng để làm gì? Độc giả là ai? Độc giả quan tâm đến cái gì? Cần xác định loại đọc giả là ai? Với mục tiêu "Đưa cho độc giả cái mà họ muốn xem, không phải là cái mà ta muốn họ xem".

Xây dụng website là một quá trình lâu dài, phải lên kế hoạch rõ ràng. Phải bám sát với mục đích và yêu cầu đã đăth ra.

2. Xác định độc giả

Sự thành công của một website phụ thuộc vào số lượng người truy cập(độc giả). Độc giả gồm nhiều đối tượng khác nhau như: Đọc giả chuyên nghiệp, độc giả không am hiểu nhiều về web hay mạng…

Tùy thuộc vào độc giả chính của website, chúng ta lựa chọn phong cách của website. Phong cách này thể hiện qua màu sắc, phông chữ, hình ảnh của trang web hay văn phong của website.

3. Thiết kế giao diện Website

a. Xác định kiểu chữ, màu sắc

Phụ thuộc vào các đặc điểm: người dùng, trình duyệt, độ phân giải, ngôn ngữ sử dụng

Gam màu phải thống nhất trong toàn bộ website

Phải làm cho đọc giả cảm nhận được kích thước của trang thông tin, biết họ đang ở đâu, có thể làm gì ?

*Lưu ý, hầu hết các trang web đều không vừa khớp với màn hình 14, 15 inch.

b. Xác định các kỹ thuật, công cụ thiết kế

Phụ thuộc nhiều yếu tố:

- Môi trường hosting
- Đội ngũ thiết kế
- Chi phí thiết kế
- Băng thông đường truyền

c. Cắt đoạn, tóm lược thông tin

Thông tin cần chia nhỏ, tóm lược lại vì:

- Người dùng có ít thời gian để đọc các tài liệu trên màn hình. Tuy nhiên cần lưu ý họ luôn có nhu cầu muốn tìm phần thông tin chủ định, không nên chia cắt quá nhỏ, tóm lược quá ngắn sẽ gây thất vọng
- Hình thức và cách tổ chức đồng nhất cho phép người dùng áp dụng kinh nghiệm tìm kiếm, khám phá thông tin và cho phép họ đoán được phần website mới, lạ sẽ được tổ chức như thế nào
- Thông tin ngắn gọn, súc tích thích hợp với màn hình máy tính (bị giới hạn tầm nhìn).
- Việc áp dụng phải linh động, nhất quán, với hệ thống logic và sự thuận tiện cho người dùng. Cách tốt nhất để phân chia và tổ chức thông tin là thực hiện theo bản chất của nội dung.

d. Xác định cấu trúc WebSite

Hệ thống phân cấp

Dùng để tổ chức các khối thông tin phức hợp, là hệ thống được dùng thông dụng nhất, gần với mô hình tổ chức thế giới thực nên dễ hình dung tổ chức website

Hệ thống các trang nối tiếp

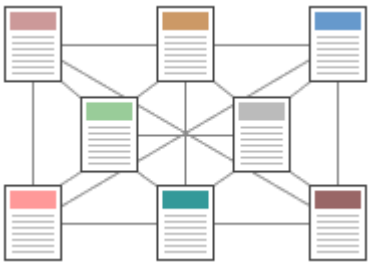
Được dùng để biểu diễn thông tin tuần tự, các bảng tường thuật nối tiếp theo thời gian, ví dụ như các thông tiin tra cứu tham khảo: tự điển, bácк khoa, tự điển thuật ngữ. Thích hợp cho hệ thống website nhỏ.

Ô Lưới

Từng đơn vị trong cấu trúc phải có cùng cấu trúc cho các chủ đề lớn và nhỏ, cấu trúc này khó hiểu khi xác định mối liên quan giữa các loại thông tin nhưng rất tốt đối với những đọc giả có kinh nghiệm, có sẵn kiến thức về hệ thống, chủ đề trong hệ thống

Mạng nhện

Mô hình này khai thác triệt để ưu điểm của hyperlink, tuy nhiên cấu trúc này phi thực tế nhất, khó hiểu, khó dự đoán cho người dùng, thích hợp với những site nhỏ, đọc giả chuyên nghiệp hoặc trình độ cao, tìm kiếm các kiến thức chuyên sâu



4. Các thành phần cơ bản của Website

a. Trang chủ (HomePage)

Tất cả các website đều được thiết lập xung quanh trang chủ (home page) giữ nhiệm vụ như một điểm xuất phát đến các trang web phúc tạp khác trong website. Trang chủ là địa chỉ web để hướng người dùng đến website của ta, là cái đầu tiên mà người dùng nhìn thấy khi truy cập đến website. Do đó trang chủ được thiết kế thích hợp là điều kiện cơ bản để website thành công.

b. Hệ thống Menu, Logo, định danh

Hệ thống menu phải rõ ràng, đầy đủ sẽ giúp đọc giả hình dung được cấu trúc, tổ chức website. Ngoài ra ta cần quan tâm đến vị trí, các thể hiện (có hay không có hiệu ứng), vị trí logo, định danh phải cố định nhất quán

c. Các trang thành viên

Ta xây dựng theo cấu trúc cơ bản của website, nhất quán, phù hợp với các thuộc tính đã được định dạng trước

III. Một số nguyên tắc khi phát triển website.

1 . Tổ chức website chặt chẽ và dễ sử dụng.

Website của bạn cần có cấu trúc càng rõ ràng dễ hiểu càng tốt. Điều quan trọng ở đây là phải làm sao để khách hàng thấy được ngay các thông tin mà họ hi vọng có thể thu được từ website của bạn. Nếu website của bạn có quá nhiều thông tin, bạn có thể làm cho trang chủ đơn giản bằng cách thiết kế bảng nội dung, bảng này cũng nên hết sức đơn giản và dễ sử dụng. Đồng thời sử dụng những từ và đoạn ngắn gọn dễ hiểu để thu hút người đọc.

2 . Sử dụng từ ngữ dễ hiểu.

Một ai đó sẽ không thể theo dõi được quảng cáo bán hàng của bạn cũng như mua những mặt hàng mà bạn đang cung cấp nếu như họ không thể hiểu được những gì bạn đang nói. Sử dụng những lời lẽ hoa mỹ để tán dương những sản phẩm bạn cung cấp thì rất dễ nhưng bạn sẽ không thể biết được có bao nhiêu người tới website và dự định của họ như thế nào?

Có thể bạn cung cấp những sản phẩm dịch vụ chất lượng tốt nhưng sẽ không ai mua nếu như họ không biết bạn đang chào bán những gì, hay không thể hiểu được lợi ích mà sản phẩm dịch vụ của bạn mang đến cho khách hàng. Hãy nhớ rằng khi một người đến thăm website của bạn, có thể anh ta chưa biết bạn là ai?. bạn đang chào bán sản phẩm gì?. Bạn phải giúp khách hàng hiểu rõ những vấn đề này trong thời gian ngắn nhất. Hãy dùng các câu ngắn gọn, cô đọng và giữ kiểu thiết kế cố định đối với tất cả các trang con.

3. Dễ dàng khám phá các đường link.

Bạn hãy tạo các đường link bằng chữ hay biểu tượng ở tất cả các trang con để mọi người có thể xem lại hoặc xem tiếp mà không phải sử dụng đến nút "Back" hay "Forward" của trình duyệt. Bạn cũng cần nhớ là phải có những chữ thay thế tất cả các đồ hoạ và đường liên kết trong trang của bạn. Đây là những từ sẽ xuất hiện thay thế đồ họa khi tuỳ chọn đồ hoạ trong trình duyệt bị tắt hoặc khi người ta nhấn nút "Stop" trước khi trang được tải về đầy đủ.

4. Thời gian tải về nhanh.

Bạn đừng nghĩ rằng tất cả mọi người đều sử dụng một đường truyền Internet có tốc độ cao. Liệu bạn có muốn mình phải đợi 10 phút để tải một trang về trước khi xem trang đó không?. Chắc chắn là không, vì thế bạn đừng hy vọng khách hàng sẽ đợi. Bạn cũng nên nhớ rằng 30 giây trước màn hình giống như 10 phút vậy.

Sử dụng đồ hoạ để trang trí là rất tốt nhưng đừng lạm dụng. Nếu bạn cần nhiều hình ảnh và đồ hoạ lớn thì nên có một biểu tượng nhỏ sẽ liên kết với hình ảnh đồng thời nhắc nhở người xem cần phải đợi. Sử dụng video và audio trong trang như một công cụ để bán hàng là ý tưởng khá hay, tuy nhiên bạn không nên sử dụng bởi hiện tại trừ các tỉnh thành lớn có đường truyền tốc độ cao ADSL hay cáp quang, vẫn còn đa số người vẫn đang sử dụng đường truyền Dial-Up với modem 28.8, 33.6 và 56.6.

Nhân tố thời gian là vô cùng quan trọng vì mọi người thường không kiên nhẫn khi vào mạng. Nếu trang của bạn phải mất thời gian quá lâu để tải về thì khách hàng có thể nhấn chuột và bỏ đi. Đừng để mất khách hàng chỉ vì trang web của bạn tải về quá chậm.

Hãy tăng tốc độ truyền của các trang web lên bằng cách:

Giảm kích cỡ đồ hoạ trong trang web của bạn. Nhiều file đồ hoạ không nhất thiết phải có kích cỡ như trên các trang web thông thường. Bạn chỉ cần 72 dpi cho độ phân giải của màn hình và đồ hoạ cũng chỉ cần 256 màu. Một đồ hoạ kích cỡ nhỏ 4" - 2" không nên lớn hơn 10K. Thu nhỏ kích cỡ đồ hoạ, độ sâu của màu. Hãy để chế độ phân giải đồ hoạ và hình ảnh nền ở mức 256 màu. Nếu bạn rất cần một đồ hoạ lớn thì bạn có thể cung cấp cho người xem một hình ảnh nhỏ với nút "phóng to" để xem tiếp một hình ảnh lớn hơn.

Quy định cụ thể kích cỡ file đồ họa trong mã HTML. Nếu bạn quy định cụ thể kích cỡ các file đồ hoạ trong mã HTML, trình duyệt web sẽ rút ngắn kích cỡ của trang nếu cần thiết và nơi hình ảnh sẽ xuất hiện, hiển thị văn bản và để một khoảng trống cho file đồ hoạ tải về.

Giảm số file trong trang web của bạn (cả file đồ hoạ và HTML kết hợp với nhau). Mọi người luôn xem nhẹ thủ thuật quản lý trang: giảm số file chứa trong website của bạn. Mọi người thường có tối đa bốn kết nối (socket) trong trình duyệt web của họ. Mỗi một socket sẽ cho phép chuyển một file về máy tính của bạn, vì thế nếu bạn có 4 socket thì bạn có thể tải cùng lúc 4 file về. Nếu bạn có 6 ảnh trong trang chủ và một file HTML thì tất cả là có 7 file

cần phải tải về. Trình duyệt sẽ tải 4 file về trước , sau khi tải xong một file socket sẽ tải tiếp file còn lại. Nói cách khác file thứ 5 sẽ chỉ được tải về khi file thứ nhất được tải xong. Và file thứ 6 sẽ chưa được tải về cho đến khi quá trình tải file thứ hai hoàn thành... quá trình tải về có thể kéo dài nếu có quá nhiều file đặc biệt khi những file này rất lớn. Theo như nguyên tắc, (giả sử đồ hoạ của bạn có kích cỡ khiêm tốn 5-12K) bạn hãy cố gắng có dưới 5 file mỗi trang.

5. Nội dung không có hình ảnh.

Nhiều người sử dụng ảnh "GIFS" và JavaScripts để tạo các logo và ký tự chạy ngang màn hình hay những gì tương tự. Điều này không chỉ làm tăng thời gian tải về mà còn làm người xem xao lãng nội dung bán hàng của bạn. Những người trên Internet là những con người của thông tin vì vậy bạn hãy chắc chắn rằng mình đang dành thời gian cho những thông tin có chất lượng chứ không phải là những hình ảnh vô bổ.

Nếu bạn có một nội dung vô giá trong trang web, hãy làm cho nó dễ đọc. Hãy chia thành những đoạn quan trọng, gạch chân hoặc bôi đậm những câu quan trọng trong từng đoạn và bạn đừng ngại trang trí với một số màu.

Tô màu văn bản thay thế file đồ hoạ nếu có thể. Nói cách khác thay vì sử dụng một file đồ họa để gây sự chú ý, bạn có thể sử dụng văn bản có màu sắc khác nhau.

Có thể bạn muốn cung cấpthông tin miễn phí dưới dạng bài báo hay bài phóng sự, và sau đó cố gắng bán hàng. Nếu bạn muốn cung cấp cho người sử dụng những thông tin bổ ích (với mục đích thu hút khách hàng), hãy thêm những nội dung có chất lượng chứ không phải là những hình ảnh bên ngoài. Trong trường hợp đó một chữ đáng giá hàng nghìn hình ảnh.

Thậm chí bạn muốn trang của mình sinh động hơn một chút (có những biểu tượng biến hình, các dòng chữ bôi đậm...) nhằm thu hút mọi người tiếp tục quan tâm tới sản phẩm và dịch vụ của bạn. Công việc của bạn chính là kiểm tra những kết quả mà khách hàng xem đem lại.

Tất cả sẽ phụ thuộc vào những sản phẩm và dịch vụ bạn đang bán cũng như đối tượng khách hàng bạn cần tiếp thị hay thị trường mục tiêu của bạn. Nếu bạn đang tiếp thị cho lớp trẻ thì sự sinh động của website sẽ làm tăng doanh số bán hàng. Nhưng nếu đối tượng khách hàng của bạn chủ yếu là những nhà kinh doanh có trình độ thì yếu tố sinh động đó có thể làm bạn giống như một hoạ sĩ nửa mùa. Đối với đối tượng khách hàng này bạn cần thu hút họ bằng những sự kiện, con số, sự trung thực và những lợi ích rõ ràng.

6. Dễ theo dõi "quá trình bán hàng".

Bạn phải tạo điều kiện để khách hàng hiểu rõ những lợi ích của sản phẩm và dịch vụ của bạn đem lại cũng như cung cấp cho khách hàng phương thức đặt hàng thuận tiện nhất. Liệu bạn đã xây dựng được uy tín đối với khách hàng trước khi bạn yêu cầu họ đặt hàng

chưa? Bạn đã tạo cho khách hàng sự yêu thích và hứng thú trước khi bạn mời họ đặt hàng chưa?. Bạn đã cung cấp cho khách hàng một số cách đặt hàng thuận tiện cả trên mạng và ngoài mạng chưa?. Và liệu bạn đã hướng dẫn khách xem tất cả từng bước một chưa?.

7. Tương thích với đa số trình duyệt web.

Nếu bạn sử dụng bảng biểu hãy xem xét cẩn thận việc nó sẽ hiển thị như thế nào ở các trình duyệt khác nhau (ví dụ Internet Explorer, Netscape) và ở tất cả các cấp độ phân giải (ví dụ 800 x 600, 1024 x 768, 1280 x 1024, 1400 x 1050).

8. Một số vấn đề quan trọng khác khi thiết kế website.

Đọc và kiểm tra cẩn thận tất cả các nội dung. Nếu bạn không quan tâm tới việc kiểm tra lỗi chính tả, người sử dụng sẽ nghi vấn làm sao họ có thể giao tiền của mình cho một công ty không thể tự sửa lỗi chính tả cho trang web của mình?. Hãy nhờ một người bạn hoặc đồng nghiệp đọc và sửa giúp bạn bởi họ có thể tìm thấy những lỗi mà bạn không bao giờ phát hiện ra được.

Trước khi đưa mọi việc vào hoạt động bạn cần có một đợt kiểm tra toàn bộ website (các đường liên kết, thời gian tải, form bán hàng...) và cố gắng kiểm tra bằng nhiều phương pháp.

Một điều hết sức quan trọng là bạn không nên nói ngay cho người xem biết bạn đang cố gắng bán hàng cho họ. Bất kể bạn đang có sản phẩm gì, cho dù sản phẩm của bạn có tốt như thế nào đi nữa thì hầu hết mọi người sẽ không ở lại trang của bạn nếu họ biết họ đang bị dụ dỗ mua hàng. Bạn cần để họ đọc, nhận ra được những lợi ích bạn sẽ đem lại cho họ và sau đó chỉ nên để họ biết rằng sản phẩm đó đang có bán. Nếu bạn thực hiện được điều này thì có nghĩa là khách hàng sẽ tự tìm thấy và mua sản phẩm bạn cung cấp.

Một yếu tố thành công khác trong marketing trực tuyến là bạn phải có khả năng chấp nhận được các giao dịch buôn bán trực tuyến. Khi bạn cung cấp sản phẩm thông tin thì bạn nên chào bán duới dạng điện tử thông qua email.

Nếu trang web của bạn lớn hơn 50K thì bạn hãy đặt một ghi chú nhỏ trên cùng của trang để thuyết phục khách hàng nên kiên nhẫn trong khi đồ họa được tải về (câu này sẽ hiện lên khi trang web của bạn tải về). Thậm chí ngay cả khi tại các trang web thử nghiệm mà bạn thấy không mất nhiều thời gian để tải về nhưng bạn cũng cần nhớ rằng khách hàng cũng có thể có đường kết nối chậm do đó làm tăng thời gian tải. Bạn không nên để khách hàng ra đi chỉ vì thời gian tải quá lâu mà bạn không thông báo về việc họ phải đợi.

Chương 2: Giới thiệu về ngôn ngữ HTML

I. Khái niệm cơ bản về html

1. HTML là gì?

HTML (viết tắt cho *HyperText Markup Language*, tức là "Ngôn ngữ Đánh dấu Siêu văn bản") do Tim Berner Lee phát minh và được W3C (World Wide Web Consortium) đưa thành chuẩn năm 1994. HTML là một ngôn ngữ đánh dấu được thiết kế ra để tạo nên các trang web được trình bày trên World Wide Web.

2. Thẻ HTML

HTML sử dụng các thẻ (tags) để định dạng dữ liệu. Các thẻ HTML không phân biệt chữ hoa, chữ thường. Các trình duyệt thường không báo lỗi Cú pháp HTML. Nếu viết sai thì kết quả hiển thị không đúng với dự định ban đầu.

HTML có 2 loại thẻ: thẻ đóng và thẻ mở

- Thẻ mở: **<tên_thẻ>**

Ví dụ 2.1: <html>, <body>, ,…

- Thẻ đóng tương ứng: **</tên_thẻ>**

Ví dụ 2.2: </html>, </body>, ,…

Có nhiều thẻ HTML, mỗi thẻ có 1 tên và mang ý nghĩa khác nhau.

* Chú ý: luôn có thẻ mở nhưng có thể không có thẻ đóng tương ứng. **Ví dụ:** <img> không có thẻ đóng

3. Cần gì để tạo một trang web

Có thể tạo trang HTML bằng bất cứ trình soạn thảo nào như Notepad, EditPlus,…Có nhiều trình soạn thảo HTML cho phép người sử dụng thực hiện một cách trực quan, kết quả sinh ra HTML tương ứng như:

- Microsoft FrontPage, notepad, notepad++
- Macromedia Dreamweaver

Trang HTML có phần mở rộng là .htm hoặc .html

II. Các thẻ định cấu trúc tài liệu

1. Thẻ html

Cặp thẻ này được sử dụng để xác nhận một tài liệu là tài liệu HTML, tức là nó có sử dụng các thẻ HTML để trình bày. Toàn bộ nội dung của tài liệu được đặt giữa cặp thẻ này.

Cú pháp:

```
<html>
    <!-- ….nội dung thẻ html -->
</html>
```

Trình duyệt sẽ xem các tài liệu không sử dụng thẻ **<html>** như những tệp tin văn bản bình thường.

2. Thẻ head

Thẻ **head** được dùng để xác định phần mở đầu cho tài liệu.

Cú pháp:

```
<head>
    <!-- Phần khai báo -->
</head>
```

3. Thẻ title

Cặp thẻ này chỉ có thể sử dụng trong phần mở đầu của tài liệu, tức là nó phải nằm trong thẻ phạm vi giới hạn bởi cặp thẻ <**head**>.

Cú pháp:

```
<title> Tiêu đề trang web </title>
```

4. Thẻ body

Thẻ này được sử dụng để xác định phần nội dung chính của tài liệu - phần thân (body) của tài liệu.

Trong phần thân có thể chứa các thông tin định dạng nhất định để đặt ảnh nền cho tài liệu, màu nền, màu văn bản siêu liên kết, đặt lề cho trang tài liệu... Những thông tin này được đặt ở phần tham số của thẻ.

Cú pháp:

```
 <body>
<!--… nội dung tài liệu html -->
</body>
```

Trên đây là Cú pháp cơ bản của thẻ **body**, tuy nhiên bắt đầu từ HTML 3.2 thì có nhiều thuộc tính được sử dụng trong thẻ **body**. Sau đây là các thuộc tính chính:

Background = Đặt một ảnh nào đó làm ảnh nền (background) cho văn bản. Giá trị của tham số này (phần sau dấu bằng) là URL của file ảnh. Nếu kích thước

	ảnh nhỏ hơn cửa sổ trình duyệt thì toàn bộ màn hình cửa sổ trình duyệt sẽ được lát kín bằng nhiều ảnh.
Bgcolor=	Đặt mầu nền cho trang khi hiển thị. Nếu cả hai tham số **background** và **bgcolor** cùng có giá trị thì trình duyệt sẽ hiển thị mầu nền trước, sau đó mới tải ảnh lên phía trên.
Text	Xác định màu chữ của văn bản, kể cả các đề mục.
Alink = Vlink = Link =	Xác định màu sắc cho các siêu liên kết trong văn bản. Tương ứng, alink (*active link*) là liên kết đang được kích hoạt - tức là khi đã được kích chuột lên; vlink (*visited link*) chỉ liên kết đã từng được kích hoạt;

Như vậy một tài liệu HTML cơ bản có cấu trúc như sau:

```
<html>
<head>
<title>tiêu đề của tài liệu</title>
</head>
<body các tham số nếu có>
... nội dung của tài liệu
</body>
</html>
```

III. Các thẻ định dạng khối

1. Thẻ định dạng khối văn bản <p>

Thẻ <**P**> được sử dụng để định dạng một đoạn văn bản.

Cú pháp:

```
<p> Nội dung đoạn văn bản </p>
```

2. Các thẻ định dạng đề mục h1/h2/h3/h4/h5/h6

HTML hỗ trợ 6 mức đề mục. Chú ý rằng đề mục chỉ là các chỉ dẫn định dạng về mặt logic, tức là mỗi trình duyệt sẽ thể hiện đề mục dưới một khuôn dạng thích hợp. Có thể ở trình duyệt này là font chữ 14 point nhưng sang trình duyệt khác là font chữ 20 point. Đề mục cấp 1 là cao nhất và giảm dần đến cấp 6. Thông thường văn bản ở đề mục cấp 5 hay cấp 6 thường có kích thước nhỏ hơn văn bản thông thường.

Dưới đây là các thẻ dùng để định dạng văn bản ở dạng đề mục:

…	Định dạng đề mục cấp 1
…	Định dạng đề mục cấp 2
…	Định dạng đề mục cấp 3
…	Định dạng đề mục cấp 4

<h5>…</h5> Định dạng đề mục cấp 5
<h6>…</h6> Định dạng đề mục cấp 6

3. Thẻ xuống dòng

Thẻ này không có thẻ kết thúc, nó có tác dụng chuyển sang dòng mới. Lưu ý, nội dung văn bản trong tài liệu HTML sẽ được trình duyệt Web thể hiện liên tục, các khoảng trắng liền nhau, các ký tự tab, ký tự xuống dòng đều được coi như một khoảng trắng. Để xuống dòng trong tài liệu, bạn phải sử dụng thẻ <**br**> hoặc </**br**>

Ví dụ: Sử dụng thẻ
 để định dạng xuống dòng văn bản

```
<body>
Tên đăng nhập: </br> Mật khẩu:
</body>
```

4. Thẻ pre và thẻ

Để giới hạn đoạn văn bản đã được định dạng sẵn bạn có thể sử dụng thẻ <**pre**>hoặc ** **. Văn bản ở giữa hai thẻ này sẽ được thể hiện giống hệt như khi chúng được đánh vào, ví dụ dấu xuống dòng trong đoạn văn bản giới hạn bởi thẻ <**pre**> hoặc ** ** sẽ có ý nghĩa chuyển sang dòng mới (trình duyệt sẽ không coi chúng như dấu cách)

Cú pháp:

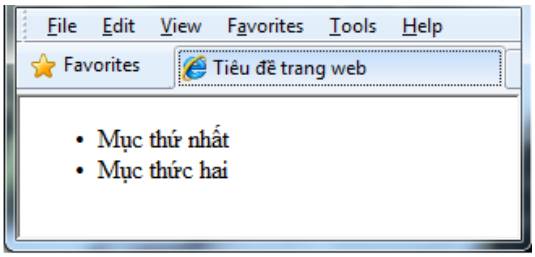
```
<pre> Văn bản được định dạng </pre>
<div> Văn bản được định dạng </div>
```

IV. Các thẻ định dạng danh sách

Kiểu 1: Danh sách không sắp xếp

Ví dụ:

```
<ul>
<li> Mục thứ nhất
<li> Mục thức hai
</ul>
```



Kiểu 2: Danh sách có sắp xếp <**ol**>, mỗi mục trong danh sách được sắp xếp thứ tự.

Cú pháp:

```
<ol type =1/a/A/i/I>
<li> Mục thứ nhất
```

```
<li> Mục thứ hai
<li> Mục thứ ba
</ol>
```

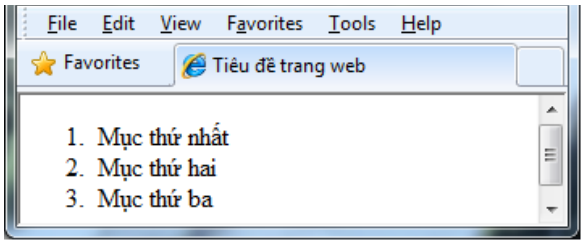
type =1 Các mục được sắp xếp theo thứ tự 1, 2, 3...
=a Các mục được sắp xếp theo thứ tự a, b, c...
=A Các mục được sắp xếp theo thứ tự A, B, C...
=i Các mục được sắp xếp theo thứ tự i, ii, iii...
=I Các mục được sắp xếp theo thứ tự I, II, III...

Ngoài ra còn thuộc tính **START**= xác định giá trị khởi đầu cho danh sách.

Thẻ < **LI** > có thuộc tính **TYPE=** xác định ký hiệu đầu dòng (bullet) đứng trước mỗi mục trong danh sách. Thuộc tính này có thể nhận các giá trị : ***disc*** (chấm tròn đậm); ***circle*** (vòng tròn); ***square*** (hình vuông).

Ví dụ:

```
<ol type =1>
<li> Mục thứ nhất
<li> Mục thứ hai
<li> Mục thứ ba
</ol>
```



Kiểu 3: Danh sách thực đơn <**menu**>

Kiểu 4: Danh sách phân cấp <**dir**>

V. Các thẻ định dạng ký tự

1. Các thẻ định dạng in ký tự

Sau đây là các thẻ được sử dụng để quy định các thuộc tính như in nghiêng, in đậm, gạch chân... cho các ký tự, văn bản khi được thể hiện trên trình duyệt.

… **In chữ đậm**
** …**
** …** *In chữ nghiêng*
** … **
<u>… </u> <u>In chữ gạch chân</u>
<S> ... </S> ~~In chữ bị gạch ngang.~~
<strike> ... </strike>
<big> ... </big> In chữ lớn hơn bình thường bằng cách tăng kích thước font hiện thời lên một.
<**small**> ... </**small**> In chữ nhỏ hơn bình thường bằng cách giảm kích thước font hiện thời đi một.

<**sup**> ... </**sup**>	Định dạng chỉ số trên (SuperScript)
<**sub**> ... </**sub**>	Định dạng chỉ số dưới (SubScript)
<basefont>	Định nghĩa kích thước font chữ được sử dụng cho đến hết văn bản. Thẻ này chỉ có một tham số size= xác định cỡ chữ. Thẻ <basefont> không có thẻ kết thúc.
<**font**> ... </**font**>	Chọn kiểu chữ hiển thị. Trong thẻ này có thể đặt hai tham số size= hoặc color= xác định cỡ chữ và màu sắc đoạn văn bản nằm giữa hai thẻ. Kích thước có thể là tuyệt đối (nhận giá trị từ 1 đến 7) hoặc tương đối

2. Căn lề văn bản trong trang Web

Trong trình bày trang Web của mình các bạn luôn phải chú ý đến việc căn lề các văn bản để trang Web có được một bố cục đẹp. Một số các thẻ định dạng như **p**, **hn**, **img**... đều có tham số **ALIGN** cho phép bạn căn lề các văn bản nằm trong phạm vi giới hạn bởi của các thẻ đó.

Các giá trị cho tham số **align** = "left | center | right";

Ngoài ra, chúng ta có thể sử dụng thẻ **center** để căn giữa trang một khối văn bản.

Cú pháp:

<**center**> Văn bản sẽ được căn giữa trang</**center**>

3. Các ký tự đặc biệt

Ký tự & được sử dụng để chỉ chuỗi ký tự đi sau được xem là một thực thể duy nhất. Ký tự ; được sử dụng để tách các ký tự trong một từ.

Ký tự	Mã ASCII	Tên chuỗi
<	&#060	&lt
>	&#062	&gt
&	&#038	&amp

4. Sử dụng màu sắc trong thiết kế các trang Web

Một màu được tổng hợp từ ba thành phần màu chính, đó là: Đỏ (Red), Xanh lá cây (Green), Xanh nước biển (Blue). Trong HTML một giá trị màu là một số nguyên dạng hexa (hệ đếm cơ số 16) có định dạng như sau:

#RRGGBB

trong đó:

RR - là giá trị màu Đỏ.

GG - là giá trị màu Xanh lá cây.

BB - là giá trị màu Xanh nước biển.

Màu sắc có thể được xác định qua thuộc tính bgcolor= hay color=. Sau dấu bằng có thể là giá trị RGB hay tên tiếng Anh của màu. Với tên tiếng Anh, ta chỉ có thể chỉ ra 16 màu trong khi với giá trị RGB ta có thể chỉ tới 256 màu.

Sau đây là một số giá trị màu cơ bản:

Màu sắc	Giá trị	Tên tiếng Anh
Đỏ	#FF0000	RED
Đỏ sẫm	#8B0000	DARKRED
Xanh lá cây	#00FF00	GREEN
Xanh nhạt	#90EE90	LIGHTGREEN
Xanh nước biển	#0000FF	BLUE
Vàng	#FFFF00	YELLOW
Vàng nhạt	#FFFFE0	LIGHTYELLOW
Trắng	#FFFFFF	WHITE
Đen	#000000	BLACK
Xám	#808080	GRAY
Nâu	#A52A2A	BROWN
Tím	#FF00FF	MAGENTA
Tím nhạt	#EE82EE	VIOLET
Hồng	#FFC0CB	PINK
Da cam	#FFA500	ORANGE
Màu đồng phục hải quân	#000080	NAVY
	#4169E1	ROYALBLUE
	#7FFFD4	AQUAMARINE

Cú pháp:

```
<body >
.... phần nội dung của tài liệu được đặt ở đây
</body>
```

Một số thuộc tính cơ bản của thẻ body:

Các tham số ý nghĩa

link	Chỉ định màu của văn bản siêu liên kết
alink	Chỉ định màu của văn bản siêu liên kết đang đang chọn
vlink	Chỉ định màu của văn bản siêu liên kết đã từng mở
background	Chỉ định địa chỉ của ảnh dùng làm nền
bgcolor	Chỉ định màu nền
text	Chỉ định màu của văn bản trong tài liệu
scroll	YES/NO - Xác định có hay không thanh cuộn
topmargin	Lề trên
rightmargin	Lề phải
leftmargin	Lề trái

5. Chọn kiểu chữ cho văn bản

Cú pháp:

```
<font
face      = font-name
color= color
size      = n >
...
</font>
```

6. Khái niệm văn bản siêu liên kết

Văn bản siêu liên kết hay còn gọi là siêu văn bản là một từ, một cụm từ hay một câu trên trang Web được dùng để liên kết tới một trang Web khác. Siêu văn bản là môi trường trong đó chứa các liên kết (link) của các thông tin. Do WWW cấu thành từ nhiều hệ thống khác nhau, cần phải có một quy tắc đặt tên thống nhất cho tất cả các văn bản trên Web. Quy tắc đặt tên đó là URL (Universal Resource Locator).

Các thành phần của URL được minh hoạ ở hình trên.

Dịch vụ: Là thành phần bắt buộc của URL. Nó xác định cách thức trình duyệt của máy khách liên lạc với máy phục vụ như thế nào để nhận dữ liệu. Có nhiều dịch vụ như **http, wais, ftp, gopher, telnet**.

Tên hệ thống : Là thành phần bắt buộc của URL. Có thể là tên miền đầy đủ của máy phục vụ hoặc chỉ là một phần tên đầy đủ – trường hợp này xảy ra khi văn bản được yêu cầu vẫn nằm trên miền của bạn. Tuy nhiên nên sử dụng đường dẫn đầy đủ.

Cổng : Không là thành phần bắt buộc của URL. Cổng là địa chỉ socket của mạng dành cho một giao thức cụ thể. Giao thức http ngầm định nối với cổng 8080.

Đường dẫn thư mục : Là thành phần bắt buộc của URL. Phải chỉ ra đường dẫn tới file yêu cầu khi kết nối với bất kỳ hệ thống nào. Có thể đường dẫn trong URL khác với đường dẫn

thực sự trong hệ thống máy phục vụ. Tuy nhiên có thể rút gọn đường dẫn bằng cách đặt biệt danh (alias). Các thư mục trong đường dẫn cách nhau bởi dấu gạch chéo (/).

Tên file : Không là thành phần bắt buộc của URL. Thông thường máy phục vụ được cấu hình sao cho nếu không chỉ ra tên file thì sẽ trả về file ngầm định trên thư mục được yêu cầu. File này thường có tên là index.html, index.htm, default.html hay default.htm. Nếu cũng không có các file này thì thường kết quả trả về là danh sách liệt kê các file hay thư mục con trong thư mục được yêu cầu

Các tham số : Không là thành phần bắt buộc của URL. Nếu URL là yêu cầu tìm kiếm trên một CSDL thì truy vấn sẽ gắn vào URL, đó chính là đoạn mã đằng sau dấu chấm hỏi (?).URL cũng có thể trả lại thông tin được thu thập từ form. Trong trường hợp dấu thăng (#) xuất hiện đoạn mã đăng sau là tên của một vị trí (location) trong file được chỉ ra.

Để tạo ra một siêu văn bản chúng ta sử dụng thẻ <**A**>.

Cú pháp:

```
<a …> Liên kết </a>
```

Một số thuộc tính của thẻ a

href	Địa chỉ của trang Web được liên kết, là một URL nào đó.
name	Đặt tên cho vị trí đặt thẻ.
tableindex	Thứ tự di chuyển khi ấn phím Tab
title	Văn bản hiển thị khi di chuột trên siêu liên kết.
target	Mở trang Web được liên trong một cửa sổ mới (*_blank*) hoặc trong cửa sổ hiện tại (*_self*), trong một frame (tên frame).

7. Địa chỉ tương đối

URL được trình bày ở trên là URL tuyệt đối. Ngoài ra còn có URL tương đối hay còn gọi là URL không đầy đủ. Địa chỉ tương đối sử dụng sự khác biệt tương đối giữa văn bản hiện thời và văn bản cần tham chiếu tới. Các thành phần trong URL được ngăn cách bằng ký tự ngăn cách (ký tự gạch chéo /). Để tạo ra URL tương đối, đầu tiên phải sử dụng ký tự ngăn cách. URL đầy đủ hiện tại sẽ được sử dụng để tạo nên URL đầy đủ mới. Nguyên tắc là các thành phần bên trái dấu ngăn cách của URL hiện tại được giữ nguyên, các thành phần bên phải được thay thế bằng thành phần URL tương đối. Chú ý rằng trình duyệt không gửi URL tương đối, nó bổ sung vào URL cơ sở đã xác định trước thành phần URL tương đối xác định sau thuộc tính href=. Ký tự đầu tiên sau dấu bằng sẽ xác định các thành phần nào của URL hiện tại sẽ tham gia để tạo nên URL mới.

Ví dụ, nếu URL đầy đủ là: http://it-department.vnuh.edu.vn/HTML/index.htm thì:

Dấu hai chấm (:) chỉ dịch vụ giữ nguyên nhưng thay đổi phần còn lại. Ví dụ **://www.fpt.com/** sẽ tải trang chủ của máy phục vụ **www.fpt.com** với cùng dịch vụ http.

Dấu gạch chéo (/) chỉ dịch vụ và máy phục vụ giữ nguyên nhưng toàn bộ đường dẫn thay đổi. Ví dụ **/Javascript/index.htm** sẽ tải file **index.htm** của thư mục **Javascript** trên máy phục vụ **www.it-department.vnuh.edu.vn**.

Không có dấu phân cách chỉ có tên file là thay đổi. Ví dụ **index1.htm** sẽ tải file **index1.htm** ở trong thư mục HTML của máy phục vụ **www.it-department.vnuh.edu.vn**.

Dấu thăng (#): chỉ dịch vụ, máy phuc vụ, đường dẫn và cả tên file giữ nguyên, chỉ thay đổi vị trí trong file.

Do đường dẫn được xem là đơn vị độc lập nên có thể sử dụng phương pháp đường dẫn tương đối như trong UNIX hay MS-DOS (tức là **.** chỉ thư mục hiện tại còn **..** chỉ thư mục cha của thư mục hiện tại).

URL cơ sở có thể được xác định bằng thẻ <BASE>.

8. Kết nối mailto

Nếu đặt thuộc tính href= của thẻ <a> giá trị mailto:address@domain thì khi kích hoạt kết nối sẽ kích hoạt chức năng thư điện tử của trình duyệt.

<address>

trang web này được
<**a** href=”mailto:webmaster@vnuh.edu.vn” >
webmaster
<**a**> bảo trì
<\address>

9. Vẽ một đường thẳng nằm ngang

Cú pháp:

```
<hr>
```

Một số thuộc tính của thẻ hr

align	Căn lề (căn trái, căn phải, căn giữa)
color	Đặt màu cho đường thẳng
noshade	Không có bóng
size	Độ dày của đường thẳng
width	Chiều dài (tính theo pixel hoặc % của bề rộng cửa sổ trình duyệt).

Thẻ này giống như thẻ **BR**, nó cũng không có thẻ kết thúc tương ứng.

VI. Các thẻ chèn âm thanh, hình ảnh

1. Giới thiệu

Liên kết với file đa phương tiện cũng tương tự như liên kết bình thường. Tuy vậy phải đặt tên đúng cho file đa phương tiện. Phần mở rộng của file phải cho biết kiểu của file.

Kiểu	Phần mở rộng	Mô tả
Image/GIF	.gif	Viết tắt của Graphics Interchange Format. Khuôn dạng này xuất hiện khi mọi người có nhu cầu trao đổi ảnh trên nhiều hệ thống khác nhau. Nó được sử dụng trên tất cả các hệ thống hỗ trợ giao diện đồ hoạ. Định dạng GIF là định dạng chuẩn cho mọi trình duyệt WEB. Nhược điểm của nó là chỉ thể hiện được 256 màu. Mở rộng của chuẩn này là GIF89, được thêm nhiều chức năng cho các ứng dụng đặc biệt như làm ảnh nền trong suốt - tức là ảnh có thể nổi bằng cách làm màu nền giống với màu nền của trình duyệt.
Image/JPEG	.jpeg	Viết tắt của Joint Photographic Expert Group. Là khuôn dạng ảnh khác nhưng có thêm khả năng nén.Ưu điểm nổi bật của khuôn dạng này là lưu trữ được hàng triệu màu và độ nén cao nên kích thước file ảnh nhỏ hơn và thời gian download nhanh hơn. Nó là cơ sở cho khuôn dạng MPEG. Tất cả các trình duyệt đều có khả năng xem ảnh JPEG.
Image/TIFF	.tiff	Viết tắt của Tagged Image File Format. Được Microsoft thiết kế để quét ảnh từ máy quét cũng như tạo các ấn phẩm.
Text/HTML	.HTML, .htm	
PostScript	.eps, .ps	Được tạo ra để hiển thị và in các văn bản có chất lượng cao.
Adobe Acrobat	.pdf	Viết tắt của Portable Document Format. Acrobat cũng sử dụng các siêu liên kết ngay trong văn bản cũng giống như **HTML**. Từ phiên bản 2.0, các sản phẩm của Acrobat cho phép liên kết giữa nhiều văn bản. Ưu điểm lớn nhất của nó là khả năng WYSISYG.
Video/MPEG	.mpeg	Viết tắt của Motion Picture Expert Group, là định

		dạng dành cho các loại phim (video). Đây là khuôn dạng thông dụng nhất dành cho phim trên WEB.
Video/AVI	.avi	Là khuôn dnạg phim do Microsoft đưa ra.
Video/QuickTime	.mov	Do Apple Computer đưa ra, chuẩn video này được cho là có nhiều ưu điểm hơn MPEG và AVI. Mặc dù đã được tích hợp vào nhiều trình duyệt nhưng vẫn chưa phổ biến bằng hai loại định dạng trên.
Sound/AU	.au	Là khuôn dạng dành cho âm nhạc điện tử hết sức thông dụng được nhiều trình duyệt trên các hệ thống khác nhau hỗ trợ. File Midi được tổng hợp số hoá trực tiếp từ máy tính.
Sound/MIDI	.mid	Định dạng audio theo dòng. Một bất tiện khi sử dụng các định dạng khác là file âm thanh thường có kích thước lớn - do vậy thời gian tải xuống lâu, Trái lại audio dòng bắt đầu chơi ngay khi tải được một phần file trong khi vẫn tải về các phần khác.Mặc dù file theo định dạng này không nhỏ hơn so với các định dạng khác song chính khả
Sound/RealAudio	.ram	năng dòng đã khiến định dạng này phù hợp với khả năng chơi ngay lập tức.
VRML	.vrml	Viết tắt của Virtual Reality Modeling Language. Các file theo định dạng này cũng giống như **HTML.** Tuy nhiên do trình duyệt có thể hiển thị được cửa sổ 3 chiều nên người xem có thể cảm nhận được cảm giác ba chiều.

2. Đưa âm thanh vào một tài liệu HTML

Cú pháp:

```
<bgsound  src = url    loop = n  >
```

Thẻ này không có thẻ kết thúc. Để chơi lặp lại vô hạn cần chỉ định **loop** = -1 hoặc **loop** = *infinite*. Thẻ **bgsound** phải được đặt trong phần mở đầu (tức là nằm trong cặp thẻ **head**).

3. Chèn một hình ảnh, một đoạn video vào tài liệu HTML

Để chèn một file ảnh (.jpg, .gif, .bmp) hoặc video (.mpg, .avi) vào tài liệu HTML, bạn có thể sử dụng thẻ **img.**

Cú pháp:

```
<img …/>
```

Một số thuộc tính của thẻ img

align = top/ middle/ bottom/ left/ right	Căn hàng văn bản bao quanh ảnh
alt = text	Chỉ định văn bản sẽ được hiển thị nếu chức năng show picture của browser bị tắt đi hay hiển thị thay thế cho ảnh trên những trình duyệt không có khả năng hiển thị đồ hoạ. Văn bản này còn được gọi là nhãn của ảnh. Đối với trình duyệt có khả năng hỗ trợ đồ hoạ, dòng văn bản này sẽ hiện lên khi di chuột qua ảnh hay được hiển thị trong vùng của ảnh nếu ảnh chưa được tải về hết. Chú ý phải đặt văn bản trong hai dấu nháy kép nếu triong văn bản chứa dấu cách hay các ký tự đặc biệt - trong trường hợp ngược lại có thể bỏ dấu nháy kép.
border = n	Đặt kích thước đường viền được vẽ quanh ảnh (tính theo pixel).
src = url	Địa chỉ của file ảnh cần chèn vào tài liệu.
width/height	Chỉ định kích thước của ảnh được hiển thị.
hspace/vspace	Chỉ định khoảng trống xung quanh hình ảnh (tính theo pixel) theo bốn phía trên, dưới, trái, phải.
title = title	Văn bản sẽ hiển thị khi con chuột trỏ trên ảnh
dynsrc = url	Địa chỉ của file video.
start = fileopen/mouseover	Chỉ định file video sẽ được chơi khi tài liệu được mở hay khi trỏ con chuột vào nó. Có thể kết hợp cả hai giá trị này nhưng phải phân cách chúng bởi dấu phẩy.
loop = n/infinite	Chỉ định số lần chơi. Nếu LOOP = INFINITE thì file video sẽ được chơi vô hạn lần.

VII. Các thẻ định dạng bảng biểu

Sau đây là các thẻ tạo bảng chính:

<**table**> ... </**table**>	Định nghĩa một bảng
<**tr**> **...** </**tr**>	Định nghĩa một hàng trong bảng
<**td**> **...** </**td**>	Định nghĩa một ô trong hàng
<**th**> **...** </**th**>	Định nghĩa ô chứa tiêu đề của cột
<caption> ... </caption>	Tiêu đề của bảng

Cú pháp:

```
<table >
... định nghĩa các dòng
<tr >
```

```
    ... định nghĩa các ô trong dòng
<td >
... nội dung của ô
</td>
...
</tr>
...
</table>
```

ý nghĩa các tham số:

align / valign	Căn lề cho bảng và nội dung trong mỗi ô.
border	Kích thước đường kẻ chia ô trong bảng, được đo theo pixel. Giá trị 0 có nghĩa là không xác định lề, giữa các ô trong bảng chỉ có một khoảng trắng nhỏ để phân biệt. Nếu chỉ để border thì ngầm định border=1. Với những bảng có cấu trúc phức tạp, nên đặt lề để người xem có thể phân biệt rõ các dòng và cột.
bordercolor	Màu đường kẻ
bordercolordark	Màu phía tối và phía sáng cho đường kẻ nổi.
bordercolorlight	
background	Địa chỉ tới tệp ảnh dùng làm nền cho bảng
bgcolor	Màu nền
cellspacing	Khoảng cách giữa các ô trong bảng
cellpadding	Khoảng cách giữa nội dung và đường kẻ trong mỗi ô của bảng.
colspan	Chỉ định ô sẽ kéo dài trong bao nhiêu cột
rowspan	Chỉ định ô sẽ kéo dài trong bao nhiêu hàng

VIII. FORM

Form HTML là một phần của tài liệu, nó chứa các phần tử đặc biệt gọi là các điều khiển. Các điều khiển được sử dụng để nhập thông tin từ người dùng và cung cấp một số tương tác.

Các form cho phép người sử dụng nhập dữ liệu trên trang web thông qua các điều khiển (control). Dữ liệu này có thể được xác nhận hợp lệ từ phía máy khách và được chuyển đến máy chủ để xử lý thêm.

Tất cả các điều khiển đều có tên được quy định qua thuộc tính **name**. Một số điều khiển không cần lấy dữ liệu thì thuộc tính name không quan trọng

Sau đây ta sẽ tìm hiểu về các loại điều khiển

X

Hồ Diên Lợi

1. Form

From dùng để chứa mọi đối tượng khác. Để tạo form ta dùng thẻ:

<form>…</form>

Một số thuộc tính của form:

- **name**="tên_form"
- **action**="địa chỉ nhận dữ liệu"
- **method**="phương thức gửi dữ liệu":
    - **GET** : thông tin xử lý theo phương thức get được hiển thị lên url
    - **POST:** thông tin xử lý theo phương thức post không hiển thị lên url

2. Hộp nhập văn bản 1 dòng (Oneline Textbox)

Oneline Textbox dùng để nhập các văn bản ngắn (trên 1 dòng) hoặc mật khẩu, sử dụng thẻ **<input>** để đưa vào form

Các thuộc tính:

- **name**="tên_đt"
- **type="text"**: nhập văn bản thường
- **type="password"**: nhập mật khẩu
- **value**="giá trị mặc định"

Ví dụ 14

```
<html>
<head>
    <title>Registered form</title>
</head>
<body>
<form name="application_form" action="test.php" method="post">
    <h1> Registered form</h1>
     <p>User name: <input type="text" name="username" value="" size="30px" />
    <p>Password: <input type="password" name="pass" value="123456" size="30px" />
    </form>
</body>
</html>
```

3. Radio Button

Radio button cho phép chọn một lựa chọn trong một nhóm lựa chọn được đưa ra. Các điều khiển radio trong một nhóm phải có cùng tên. Vào một thời điểm, người dùng chỉ có thể chọn một lựa chọn. Các nút radio nên đặt thuộc thuộc tính giá trị. Sử dụng thẻ **<input>** để đưa đối tượng radio vào form, mỗi ô cần 1 thẻ

Thuộc tính:

- **name**="tên_đt": Các đối tượng cùng tên thì thuộc cùng nhóm.
- **type="radio"**
- **value**="giá trị": đây là giá trị chương trình sẽ nhận được nếu ta chọn ô này.
- **checked**: nếu có thì nút này mặc định được chọn

4. Checkbox

Checkbox cho phép người dùng có thể chọn một hoặc nhiều lựa chọn trong một nhóm lựa chọn được đưa ra bằng cách đánh dấu tích. Sử dụng thẻ **<input>** để đưa đối tượng checkbox vào form, mỗi ô cần 1 thẻ

Thuộc tính:

- **name**="tên_đối_tượng"
- **type="checkbox"**
- **value**="giá trị": đây là giá trị chương trình sẽ nhận được nếu nó được chọn
- **checked**: nếu có thì nút này mặc định được chọn

5. Nút lệnh (Button)

Cho phép người sử dụng ra lệnh thực hiện một số công việc nào đó. Có 3 loại nút thường dùng

- ***submit***: khi người dùng nhấp vào nút submit, dữ liệu tự động được chuyển đến vị trí được xác định trong thuộc tính ACTION
- ***reset***: đưa mọi dữ liệu về trạng thái mặc định
- ***normal***: người lập trình tự xử lý

Đưa đối tượng button vào form ta dùng thẻ **<input>**

Thuộc tính:

- **name**="tên_đối_tượng"
- **type="submit"**: nút submit

- **type="reset"**: nút reset
- **type="button"**: nút thông thường (normal)
- **value**="tiêu đề nút"

6. Combo Box (Drop-down menu)

Combo box gồm một danh sách có nhiều phần tử, ta có thể chọn 1 phần tử trong danh sách xổ xuống bằng cách kích vào mũi tên bên phải hộp danh sách. Tại một thời điểm chỉ có 1 phần tử được chọn

Thẻ tạo hộp danh sách:

<select>Danh sách phần tử**</select>**

Thuộc tính:

- **name**="tên_đối_tượng"

Thẻ tạo 1 phần tử trong danh sách:

<option>Tiêu đề phần tử**</option>**

Thuộc tính:

- **value**="giá trị": giá trị chương trình nhận được nếu phần tử được chọn
- **selected**: nếu có thì phần tử này mặc định được chọn

7. Listbox

Tương tự như combobox, listbox là một danh sách gồm nhiều phần tử, tuy nhiên ta có thể nhìn thấy và lựa chọn các phần tử cùng một lúc

Thẻ tạo listbox:

<select>…</select>

Thuộc tính: tương tự như combobox nhưng có 2 thuộc tính khác:

- **size**="số dòng"
- **multiple**: cho phép lựa chọn nhiều phần tử cùng lúc

Thẻ tạo 1 phần tử trong danh sách:

<option>Tiêu đề phần tử**</option>**

Thuộc tính:

- **value**="giá trị": giá trị chương trình nhận được nếu phần tử được chọn
- **selected**: nếu có thì phần tử này mặc định được chọn

8. Hộp nhập văn bản nhiều dòng (TextArea)

Textarea cho phép người dùng nhập văn bản dài trên nhiều dòng.

Thẻ tạo textarea:

<textarea>Nội dung mặc định</textarea>

Các thuộc tính:

- **name**="tên_đối_tượng"
- **rows**="số dòng"
- **cols**="số cột"

Trong đó:

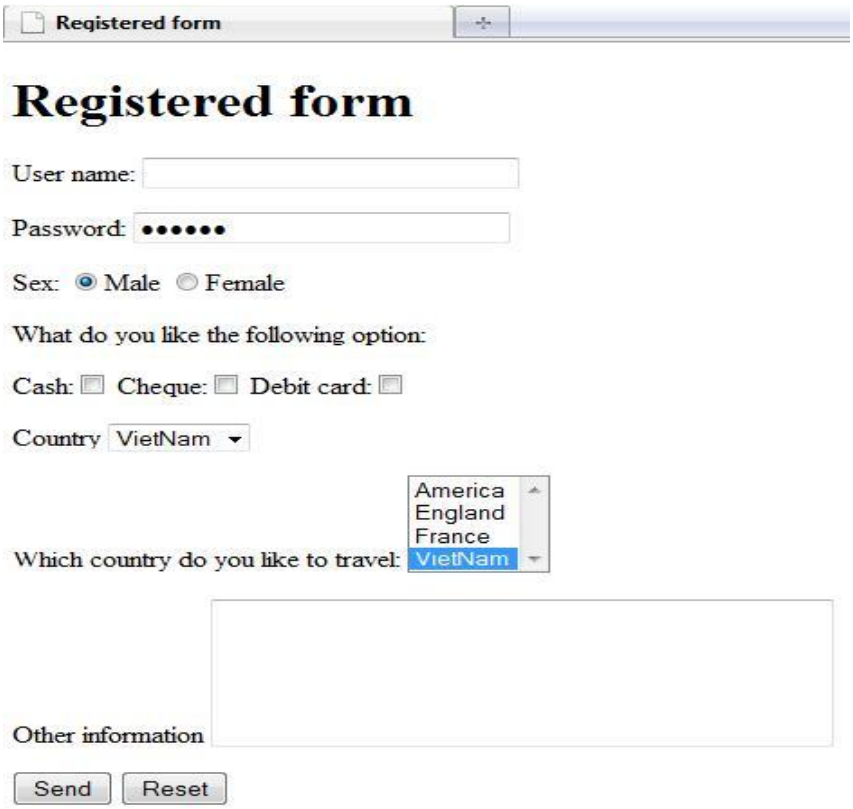
- rows: số dòng văn bản
- cols: số ký tự chuẩn trên dòng.

Ví dụ 15:

```
<html>
<head>
	<title>Registered form</title>
</head>
<body>
<form name="application_form" action="test.php" method="post">
 <h1> Registered form</h1>
<p>User name: <input type="text" name="username" value="" size="30px" />
<p>Password: <input type="password" name="pass" value="123456" size="30px" />
<p>Sex: <input type="radio" name="sex" checked="checked" value="M" />Male <input type="radio" name="sex" value="F" />Female</p>
<p>What do you like the following option:</p>
<p>Cash:<input type="checkbox" name="checkpay" value="cash" /> Cheque:<input type="checkbox" name="checkpay" value="cheque" /> Debit card:<input type="checkbox" name="checkpay" value="card" /></p>
<p>Country
	<select name="country">
	<option value="usa">America</option>
	<option value="eng">England</option>
	<option value="fra">France</option>
	<option value="vie" selected="selected">VietNam</option>
	</select>
</p>
<p>Which country do you like to travel:
```

```
        <select name="country" size="4" multiple="multiple">
        <option value="1">America</option>
              <option value="2">England</option>
          <option value="3">France</option>
          <option value="4" selected="selected">VietNam</option>
        </select>
</p>
        <p>Other information
        <textarea name="other_info" rows="5" cols="40"></textarea>
</p>
<p>
        <input type="submit" name="send" value="Send" />
          <input type="reset" name="reset" value="Reset" />
</p>
</form>
</body>
</html>
```

Kết quả hiển thị trên trình duyệt



IX. Một số thẻ đặc biệt

1. Thẻ <meta>

Thẻ <meta> được khái báo trong cặp thẻ <head>, thẻ <meta> thường được sử dụng để khái báo loại font sử dụng, tìm kiếm, xóa cache, chuyển trang…

a. Thẻ <meta> với font

Để sử dụng font Unicode đặc biệt Unicode tiếng việt trên trang web, chúng ta phải khai báo thẻ <meta> trong thẻ <head>.

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

b. Thẻ <meta> cho phép người dùng tìm kiếm.

Khi bạn đưa trang web của bạn lên internet, để người dùng có thể tìm thấy web site của bạn qua các công cụ trên như: Google, Yahoo …, khi đó chúng ta khai báo thẻ <meta> như sau

```
<meta http-equiv = “Content-Type” content = “text/html; charset = UTF-8”>
<META NAME = “author” CONTENT = “http://www.hutc.edu.vn/”>
```

Bạn có thể khái báo các thông tin khác của trang web để khi người dùng có thể tìm kiếm thông qua các thông tin này.

```
<META NAME = “keywords” CONTENT = “Công thương, kỹ thuật công nghiệp 2. ….. ”>
```

Các từ khóa này sẽ được đem so sánh với các từ khóa người dùng gõ và tìm kiếm trên Internet, nếu từ khóa người dùng tìm kiếm thuộc một trong số từ khóa của bạn đã khai báo trong thẻ này, web site của bạn sẽ được xuất hiện trong danh sách tìm kiếm được.

c. Thẻ <meta> dạng tự động chuyển đến URL

Để tự động chuyển đến địa chỉ URL hay UNC kế tiếp sau khi trang web nạp lên với thời gian nhất định, bạn có thể khái báo trong thẻ JavaScript.

Ví dụ:

```
 <html>
<head>
<title>Welcome to PHP and MySQL</title>
<META http-equiv=refresh content="8; URL=http://www.saigoninfotech.com">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#FFFFFF" text="#000000">
Trang này tự động chuyển đến trang <b>www.saigoninfotech.com</b> sau 8 giây
```

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

d. Thẻ <meta> dùng xóa cache

Thông thường sau khi nạp trang web nào đó lên trình duyệt web, nội dung của trang web đó có thể lưu vào trong bộ nhớ truy cập nhanh (cache).

Điều này có nghĩa là sau khi duyệt một vòng các trang web khác, bạn quay về gọi trang web đã truy cập trước đó, trình duyệt web nạp rất nhanh, do chúng đã lưu trang trong bộ nhớ cache.

Tuy nhiên, khi bạn là người phát triển ứng dụng web, có những trang web bạn phải xóa cache mỗi khi người dùng gọi nó. Nghĩa là, trang web này thường có thay đổi cấu trúc cho mỗi lần gọi, bạn cần khai báo thẻ <meta> như ví dụ sau:

Ví dụ:

```
<html>
<head>
<title>Welcome to PHP and MySQL</title>
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="cache-control" content="no-cache">
</head>
<body bgcolor="#ffffff"> Xoa Cache
</body>
</html>
```

2. Thẻ <marquee>

Thẻ <marquee> cho phép bạn khái báo một chuỗi chuyển động theo chiều hướng khác nhau trên trang web. Thẻ này thường được sử dụng để quảng cáo một đề tài nào đó.

Có 4 chiều di chuyển của chuỗi, bạn có thể khái báo tùy thuộc vào các tham số UP, DOWN, BACK, RIGHT của thuộc tính DIRECTION.

Ví dụ:

```
<html>
<head>
<title>Welcome to php and mysql</title>
```

```
</head>
<body>
<marquee direction=right scrolldelay=2 scrollamount=1 width=100% >
<font face="arial" size="1" color=#ff33ff>
<br>ashley- ana - chanthaly - kathleen - lena </marquee><br>
<marquee direction=up scrolldelay=2 scrollamount=1 width=100%
style="filter:wave(add=5, phase=1, freq=5, strength=20);"><font face="arial" size="1"
color=#ff33ff>
<br>ashley
<br>ana
<br>chanthaly
<br>kathleen
<br>lena </marquee>
</body>
</html>
```

3. Thẻ <style>

Thẻ <style> cho phép bạn định dạng tất cả nội dung trình bày trên trang web theo một kiểu nhất định. Điều này có nghĩa là mọi thẻ trên trang web có khai báo sử dụng một phần tử nào đó được khái báo thẻ <style>, chúng sẽ có kiểu định dạng như bạn định nghĩa.

Bạn khái báo thẻ <style> trong thẻ <head>.

Ví dụ: Định dạng style, chèn ảnh nền và đặt hình nền không lặp.

```
<style>
        background-image: url(“hinhnen.gif”);
        background-repeat: no-repeat;
</style>
```

4. Thẻ <link>

Khi khai báo các phần tử trong trang style.css, bạn có thể khái báo chúng trong một trang web bằng thẻ <link>. Để sử dụng bạn cũng khái báo như trường hợp sử dụng phần tử trong ví dụ sau:

Ví dụ: Chèn tyle.css vào trong tài liệu html

```
<link href="tyle.css" type="text/css" rel="stylesheet" />
```

5. Thẻ <script>

Trong trang web, bạn muốn kiểm soát tất cả các hành động của người dùng, bạn cần khái báo vá sử dụng một số phương thức và thuộc tính của Client Script hay các phương thức do bạn định nghĩa.

Để có thể khái báo kịch bản trên trang web, bạn sử dụng thẻ <script> với tên ngôn ngữ chỉ định JavaScript hay VBScript.

Cú pháp:

```
<script language = “javascript”>
// mã javascript
<script>
<script language = “vbscript”>
// mã vbscript
<script>
```

Ngoài ra trong trường hợp có nhiều phương thức do bạn định nghĩa được sử dụng chung trong nhiều trang web, bạn cũng có thể khai báo chúng trong một tập tin có tên mở rộng .js hay .vb. Sau đó bạn có thể chèn tập tin này và sử dụng như cách chèn trực tiếp.

Chương 3: Thiết kế CSS

I. Giới thiệu về CSS

CSS (Cascading Style Sheets) được hiểu một cách đơn giản đó là cách mà chúng ta thêm các kiểu hiển thị (font chữ, kích thước, màu sắc...) cho một tài liệu Web

Một số đặc điểm của Cascading Style Sheets:

- CSS quy định cách hiển thị của các thẻ HTML bằng cách quy định các thuộc tính của các thẻ đó (font chữ, màu sắc). Để cho thuận tiện bạn có thể đặt toàn bộ các thuộc tính của thẻ vào trong một file riêng có phần mở rộng là ".css"

- CSS nó phá vỡ giới hạn trong thiết kế Web, bởi chỉ cần một file CSS có thể cho phép bạn quản lí định dạng và layout trên nhiều trang khác nhau. Các nhà phát triển Web có thể định nghĩa sẵn thuộc tính của một số thẻ HTML nào đó và sau đó nó có thể dùng lại trên nhiều trang khác.

- Có thể khai báo CSS bằng nhiều cách khác nhau. Bạn có thể đặt đoạn CSS của bạn phía trong thẻ <head>...</head>, hoặc ghi nó ra file riêng với phần mở rộng ".css", ngoài ra bạn còn có thể đặt chúng trong từng thẻ HTML riêng biệt

- Tuy nhiên tùy từng cách đặt khác nhau mà độ ưu tiên của nó cũng khác nhau. Mức độ ưu tiên của CSS sẽ theo thứ tự sau.

Style đặt trong từng thẻ HTML riêng biệt

Style đặt trong phần <head>

Style đặt trong file mở rộng .css

Style mặc định của trình duyệt

Mức độ ưu tiên sẽ giảm dần từ trên xuống dưới.

CSS có tính kế thừa: giả sử rằng bạn có một thẻ đã được khai báo ở đầu file css với các thuộc tính như sau:

```
#vidu {
 width: 200px;
 height: 300px;
```

```
}
```

Ở một chỗ nào đó trong file css bạn lại khai báo một lần nữa thẻ với các thuộc tính.

```
#vidu {
width: 400px;
background-color: #CC0000;
}
```

Sau đoạn khai báo này thì thẻ sẽ có thuộc tính:

```
#vidu {
width: 400px; /* Đè lên khai báo cũ */
height: 300px;
background-color: #CC0000;
}
```

II. Cú pháp

1. Định dạng thuộc tính thẻ html

Các thuộc tính của thẻ html không được phong phú và đa dạng do vậy chúng ta có thể thay đổi lại thuộc tính mặc định của thẻ. Chúng ta có thể định dạng các thẻ ngay trên tài liệu html bằng thẻ <style> hoặc được định dạng trong một tập tin khác sau đó chèn vào tài liệu html.

Cú pháp:

```
name_tag
{
property_1: values;
property_2: values;
…
}
```

Ví dụ: Định dạng hình nền và các thuộc tính khác

```
body
{
background-image: url(“images\background_image.gif”);
background-repeat: no-repeat;
}
```

2. Định dạng một kiểu mới

a. Kiểu được chèn vào thẻ html bằng thuộc tính class của thẻ

Chúng ta có thể định dạng một kiểu riêng trong tài liệu html hoặc trong một tập tin riêng. Sau đó đưa định dạng đó vào trong thể html bằng thuộc tính class của thẻ.

Cú pháp:

```
.name_style
{
property_1: values;
property_2: values;
…
}
```

Ví dụ: Định dạng thuộc tính hình nền trang web

```
. mystyle
{
background-image: url(“images\background_image.gif”);
background-repeat: no-repeat;
}
```

Sau đó, ta áp dụng định dạng trên vào thẻ <body class= “mystyle”>

…

<body class= “mystyle”>

…

b. Kiểu được chèn vào thẻ html bằng thuộc tính id của thẻ

Chúng ta có thể định dạng một kiểu riêng trong tài liệu html hoặc trong một tập tin riêng. Sau đó đưa định dạng đó vào trong thể html bằng thuộc tính Id của thẻ.

Cú pháp:

```
# name_style
{
property_1: values;
property_2: values;
…
}
```

Ví dụ: Định dạng thuộc tính hình nền trang web

```
# mystyle
{
```

```
background-image: url(“images\background_image.gif”);
background-repeat: no-repeat;
}
```

Sau đó, ta áp dụng định dạng trên vào thẻ <body id= “mystyle”>

…

<body id= “mystyle”>

…

3. Định dạng ngay trong thẻ html

Ngoài các cách trên, chúng ta có thể định dạng kiểu trên thẻ html.

Cú pháp:

```
<name_tag  style= “property_1:values [; property_2: values; …] ”>
<!-- Nội dung thẻ -->
</name_tag>
```

Ví dụ:

```
<body style="background-image:url(anh_nen.JPG); background-repeat:no-repeat">
Nội dung tài liệu html
</body>
```

III. Sử dụng css trong tài liệu HTML

1. CSS được khai báo trong một tập tin riêng

Khi xây dựng website các trang web thường có những định dạng giống nhau tạo nên sự thống nhất của một website ví dụ như: Màu nền hay hình nền, các kiểu định dạng chữ, ký tự…

Nếu trang nào ta cũng sử dụng kiểu định dạng này, khi đó ta thấy mã lệnh trong một website được lặp đi lặp lại nhiều lần trên nhiều trang khác nhau.

Khi người dùng muốn thay đổi kiểu dáng hay định dạng của website người dùng chỉ cần thay đổi kiểu trong tập tin riêng này, khi đó toàn bộ trang website sẽ thay đổi theo.

Để chèn tập tin riêng chúng ta sử dụng thể <link> trong vùng thẻ <head>

```
<link href="name_style.css" rel="stylesheet" type="text/css" />
```

2. Định dạng ngay trên tài liệu html

Thường chúng ta thường định dạng css trong vùng thẻ head sử dụng thẻ <style>

Cú pháp:

```
<style>
<!-- Nội dung định dạng css -- >
</style>
```

IV. Một số thuộc tính thường dùng

1. Định kiểu nền

a. Màu nền

Để xác lập màu nền cho một thành phần của trang web ta sử dụng thuộc tính **background-color.** Các giá trị màu của background-color tương tự như color.

Ví dụ 4:

Mở file style.css ở trên thêm vào các thuộc tính màu nền như sau:

```
body { background-color:#000000;}
p{ background-color:#6495ed; color:#ffffff; }
h1 { background-color:#009fff; }
div {        background-color:#ffbf00; }
```

Kết quả hiển thị

b. Ảnh nền

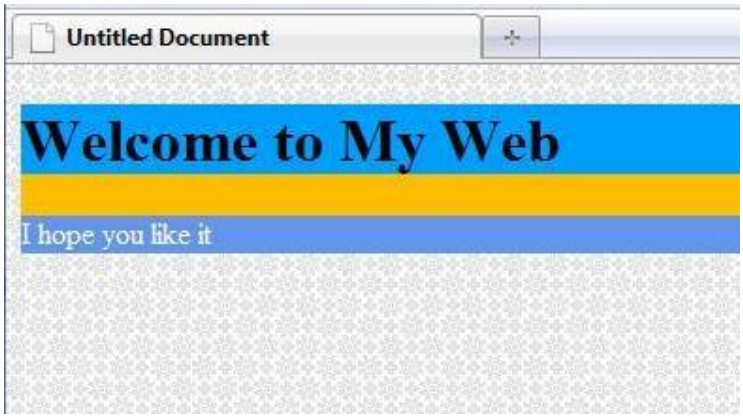
Để chèn ảnh nền vào một thành phần trên trang web chúng ta sử dụng thuộc tính background-image. Theo mặc định, ảnh nền sẽ được lặp lại để phủ kín toàn bộ trang web

Ví dụ 5

Mở file style.css ở trên và thay đổi nền cho trang web như sau

```
body {  background-image:url(page_bg.jpg); }
```

Lưu lại và xem kết quả



Lưu ý:

Để tiện cho sự quản lý các file trong website, ta nên tạo một thư mục images riêng để chứa hình ảnh. Khi đó, đường dẫn trong url như sau:

```
    background-image:url(images/page_bg.jpg);
```

Lặp lại ảnh nền:

Thuộc tính background-repeat cung cấp cho chúng ta các điều khiển giúp kiểm soát trình trạng lặp lại của ảnh nền. Thuộc tính này có 4 giá trị:

- repeat-x: Chỉ lặp lại ảnh theo phương ngang.
- repeat-y: Chỉ lặp lại ảnh theo phương dọc.
- repeat: Lặp lại ảnh theo cả 2 phương, đây là giá trị mặc định.
- no-repeat: Không lặp lại ảnh.

Ví dụ 6

Mở file style.css ở trên và sửa lại như sau

```
body {
background-image:url(page_bg.jpg);
background-repeat:no-repeat;
}
```

Lưu lại và xem kết quả.

Định vị ảnh nền

Theo mặc định, ảnh nền khi được chèn sẽ nằm ở góc trên, bên trái màn hình. Tuy nhiên với thuộc tính **background-position** ta có thể đặt ảnh nền ở bất cứ vị trí nào trong không gian của thành phần mà nó làm nền.

Ví dụ 7:

Mở file style.css và sửa lại như sau

```
body {
	background-image:url(page_bg.jpg);
	background-repeat:no-repeat;
	background-position:top right;
}
```

Mở trình duyệt và xem kết quả

2. Định kiểu chữ

a. Màu chữ

Sử dụng thuộc tính **color** để định dạng màu cho chữ trong CSS. Có nhiều cách để xác định giá trị của thuộc tính color

- Tên màu: red, blue, yellow,...
- Giá trị RGB: rgb(255,0,0)
- Giá trị HEX: #ff0000

Ví dụ 8:

```
body {color:blue}
h1 {color:#00ff00}
h2 {color:rgb(255,0,0)}
```

b. Canh lề:

Sử dụng thuộc tính **text-align** để canh chỉnh văn bản cho các thành phần trong trang web. Text-align có 4 giá trị :

- left (canh trái – mặc định)
- right (canh phải)
- center (canh giữa)
- justify (canh đều).

Ví dụ 9

```
h1 {  text-align:right  }
 p  {  text-align:justify  }
```

c. Trang trí chữ

Thuộc tính **text-decoration** dùng để thiết lập hay xóa các trang trí cho chữ

Text-decoration thường được dùng để xóa hiệu ứng gạch chân của link cho mục đích trang trí

Text-decoration thêm các hiệu ứng gạch chân (underline), gạch xiên (line-through), gạch đầu (overline), và một hiệu ứng đặc biệt là văn bản nhấp nháy (blink).

Ví dụ 10:

```
a {text-decoration:none}
h1 { text-decoration:underline }
h2 { text-decoration:overline}
```

d. Chuyển đổi chữ hoa/thường

Để chuyển đổi kiểu chữ hoa/thường ta dùng thuộc tính **text-transform**. Thuộc tính này có tất cả 4 giá trị:

- uppercase (in hoa)
- lowercase (in thường)
- capitalize (in hoa ký tự đầu tiên trong mỗi từ)
- none (không áp dụng hiệu ứng – mặc định).

Ví dụ 11

```
p { text-transform:uppercase }
h1 { text-transform:capitalize }
```

e. Thuộc tính letter-spacing:

Thuộc tính **letter-spacing** được dùng để định khoảng cách giữa các ký tự trong một đoạn văn bản.

Ví dụ 12

```
p { letter-spacing:3px }
h1{ letter-spacing:5px }
```

3. Định kiểu font

a. Tên font (font-family)

Thuộc tính font-family xác định các font sẽ được dùng để hiển thị trên trang web. Có hai loại tên font được dùng để chỉ định trong font-family:

- generic family
- font family

Generic family:

Generic family là tên của một họ gồm nhiều font. **Ví dụ:**

- **serif**
  - Times New Roman, Bodini, Garamond
- **sans-serif**
  - Trebuchet, Arial, Verdana, Futura, Gill Sans, Helvetica

- **cursive**
    - Poetica, Zapf-Chancery, Roundhand, Script
- **fantasy**
    - Critter, Cottonwood
- **monospace**
    - Courier, Courier New, Prestige, Everson Mono

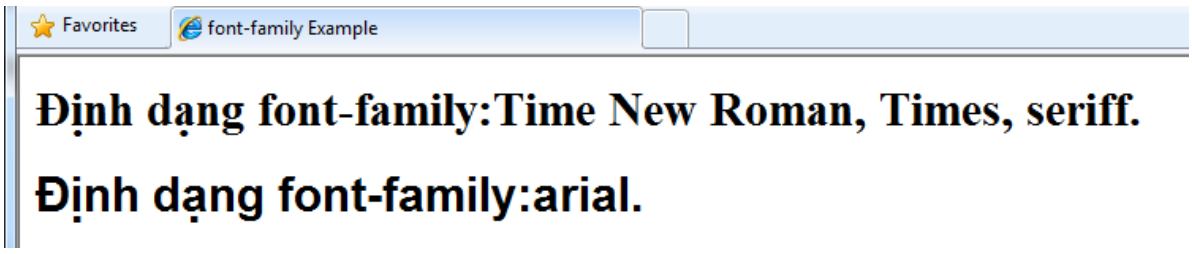
Font family:

Font family là tên cụ thể của một font. **Ví dụ:** Arial, Verdana, Time New Roman,…

Ví dụ 13

```
<html>
<head>
<title>font-family Example</title>
<style>
	body {font-size: 30px}
</style>
</head>
<body>
<p>
<strong style="font-family:'Times New Roman', Times, serif">Định dạng font-family:Time New Roman, Times, seriff..</strong>
</p>
<p>
<strong style="font-family: arial">Định dạng font-family:Time New Roman, Times, seriff..</strong>
</p>
</body>
</html>
```

Kết quả hiển thị



b. Kiểu font (font style)

Thuộc tính font style gồm 3 giá trị:

- Normal: in thườngg
- Italic: in nghiêng
- Oblique: tương tự như italic

Ví dụ 14

```
h1 { font-style: italic }
p { font-style: normal }
```

c. Cỡ font (font size)

Kích thước của một font được định bởi thuộc tính **font-size**. Font-size cung cấp 7 giá trị cho việc thiết lập size của font từ nhỏ nhất cho đến lớn nhất: xx-small, x-small, small, medium, large, x-large and xx-large. Chúng tương đương với giá trị của thẻ <font> với size="1" tới size="7".

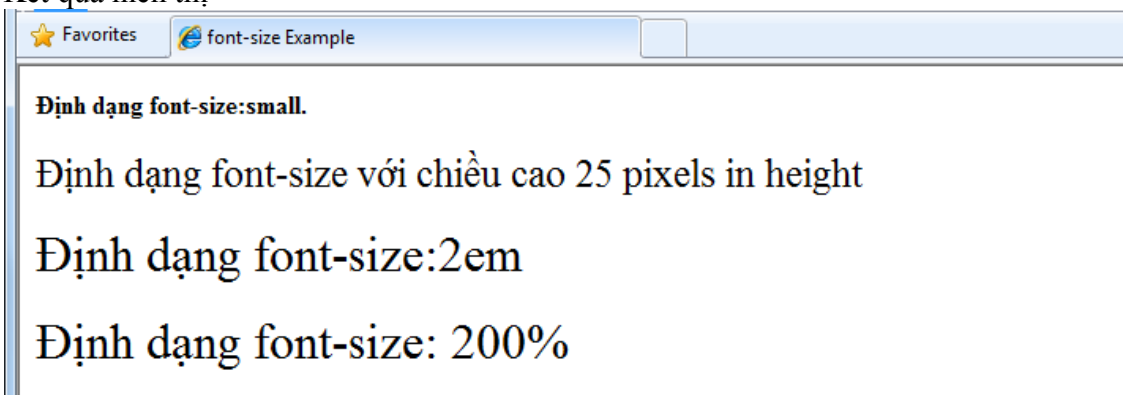
Ngoài ra các đơn vị dùng cho font thường là: pixel, em, %

Tùy theo mục đích sử dụng của website mà ta chọn những đơn vị phù hợp.

Ví dụ 15

```
<html>
<head>
<title>font-size Example</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<p><b style="font-size: small">Định dạng font-size:small.</b></p>
<p style="font-size: 25px">Định dạng font-size với chiều cao 25 pixels in height</p>
<p style="font-size:2em">Định dạng font-size:2em </p>
<p style="font-size: 200%">Định dạng font-size: 200%</p>
</body>
</html>
```

Kết quả hiển thị

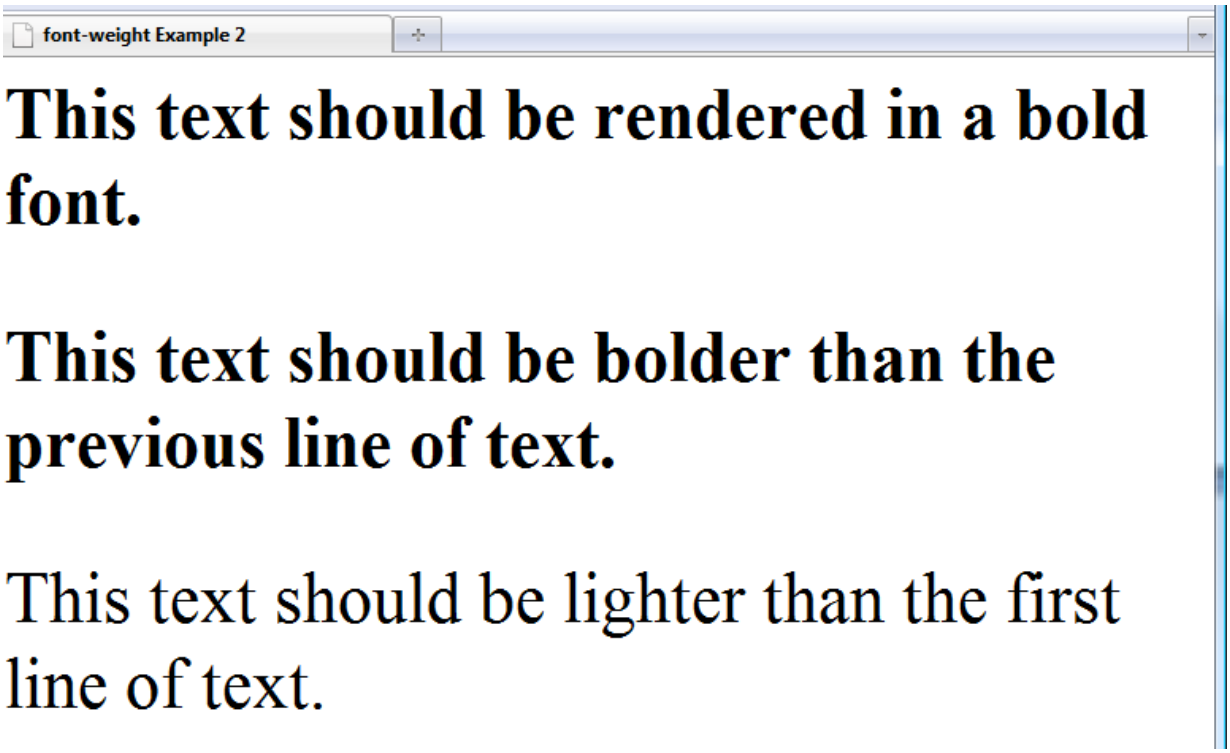


d. Thuộc tính font-weight:

Thuộc tính **font-weight** mô tả cách thức thể hiện của font chữ là ở dạng bình thường (normal) hay in đậm (bold). Ngoài ra, một số trình duyệt cũng hỗ trợ mô tả độ in đậm bằng các con số từ 100 – 900.

Ví dụ 16

```
<html>
<head>
<title>font-weight Example 2</title>
	<style>
		body {font-weight: 600; font-size: 45px}
	</style>
</head>
<body>
<p>This text should be rendered in a bold font.</p>
<p style="font-weight: bolder">This text should be bolder than the previous line of text.</p>
<p style="font-weight: lighter">This text should be lighter than
the first line of text.</p>
</body>
</html>
```



4. CSS Link

Trong CSS, ta có thể áp dụng các thuộc tính định dạng như font chữ, gạch chân, màu chữ,… cho một liên kết. Ngoài ra, CSS còn cung cấp một số hiệu ứng định dạng cho một đối tượng liên kết ở một trạng thái xác định như :

- Liên kết chưa được thăm (a:link)
- Rê chuột lên liên kết (a:hover)
- Liên kết được thăm (a:visited)
- Liên kết đang được kích hoạt – đang giữ nhấn chuột (a:active)

Ví dụ 17

```
<html>
<head>
<style type="text/css">
      a:link {color:#00FF00; font-size:14px; text-decoration:none;}
      a:hover { color:#FF00FF; font-size:1.2em; text-decoration:blink;}
      a:visited { color:#FF0000; text-decoration:none; background-color:#ff704d; }
      a:active { color:# 662D91; font-variant:small-caps; }
</style>
</head>
<body>
        <p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
</body>
</html>
```

5. Định kiểu danh sách

CSS cũng cung cấp một số thuộc tính để định dạng danh sách, làm việc trình bày trang web trở nên phong phú hơn. Các thuộc tính về danh sách của CSS cho phép ta:

- Định dạng kí hiệu cho danh sách có thứ tự
- Định dạng kí hiệu cho danh sách không thứ tự
- Dùng hình ảnh đánh dấu danh sách

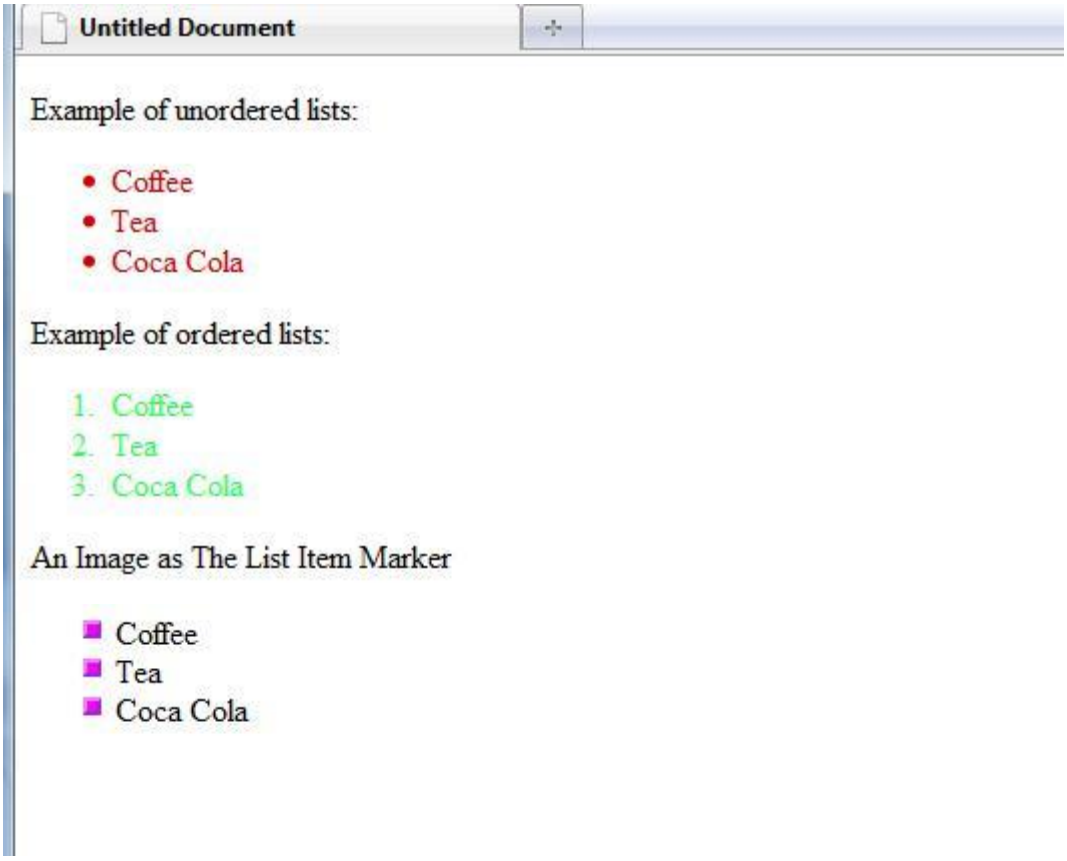
Thuộc tính **list-style-type** cho phép ta định dạng các kí hiệu đầu danh sách

Ví dụ 18

```
<html>
<head>
      <style type="text/css">
            ul{color:#D40000;}
            ol{color:#2AFF55;}
      </style>
```

```
</head>
<body>
<p>Example of unordered lists:</p>
<ul list-style-type:circle; >
     <li>Coffee</li>
     <li>Tea</li>
     <li>Coca Cola</li>
 </ul>
<p>Example of ordered lists:</p>
<ol list-style-type: upper-roman>
     <li>Coffee</li>
     <li>Tea</li>
     <li>Coca Cola</li>
   </ol>
<p>Example of ordered lists:</p>
<ul list-style-image:url('bullet.gif') >
     <li>Coffee</li>
     <li>Tea</li>
     <li>Coca Cola</li>
   </ul>
</body>
</html>
```

Kết quả



6. Định kiểu bảng

Với CSS, ta có thể thay đổi định dạng các thuộc tính của bảng như đường biên, độ rộng, độ cao của cột, màu,…

Các thuộc tính CSS dùng làm việc với bảng :

a. Border:

Dùng để định độ rộng đường viền của bảng

Ví dụ 19

table, th, td { border: 1px solid black; }

Border-collapse: có 3 giá trị

- Collapse: cho phép ta trộn các đường biên trong bảng
- Separate: cho phép ta tách các đường biên trong bảng
- Inherit:tương tự separate

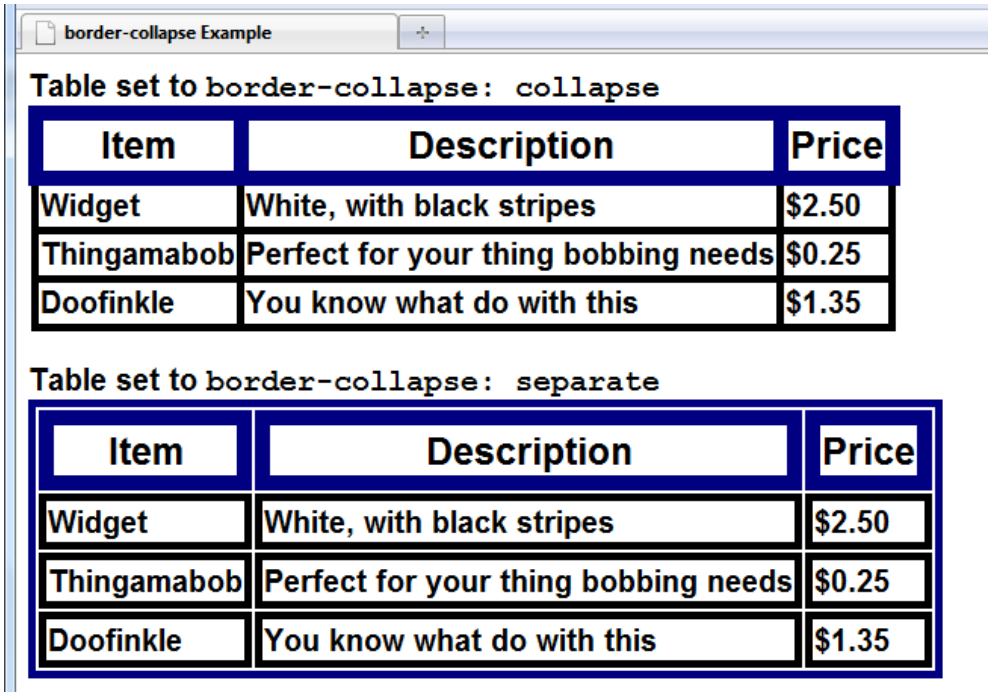
Ví dụ 20

```
<html>
<head>
<title>border-collapse Example</title>
```

```
<style>
	th {border: 10px solid navy; font: bold 25px Arial, Helvetica,
	sans-serif}
	td {border: 5px solid black; font: bold 20px Arial, Helvetica,
	sans-serif}
	body {font: bold 20px Arial, Helvetica, sans-serif}
</style>
</head>
<body>
	Table set to <code>border-collapse: collapse</code>
	<table style="border-collapse: collapse; border: 5px solid navy">
		<tr>
<th>Item</th>
<th>Description</th>
<th>Price</td>
		</tr>
		<tr>
<td>Widget</td>
<td>White, with black stripes</td>
<td>$2.50</td>
		</tr>
		<tr>
<td >Thingamabob</td>
<td >Perfect for your thing bobbing needs</td>
<td >$0.25</td>
		</tr>
		<tr>
<td>Doofinkle</td>
<td>You know what do with this</td>
<td>$1.35</td>
		</tr>
	</table>
	<p>
		Table set to <code>border-collapse: separate</code>
	<table style="border-collapse: separate; border: 5px solid navy">
		<tr>
<th>Item</th>
<th>Description</th>
<th>Price</td>
		</tr>
		<tr>
<td>Widget</td>
<td>White, with black stripes</td>
<td>$2.50</td>
		</tr>
```

```
            <tr>
<td >Thingamabob</td>
<td >Perfect for your thing bobbing needs</td>
<td >$0.25</td>
            </tr>
            <tr>
<td>Doofinkle</td>
<td>You know what do with this</td>
<td>$1.35</td>
            </tr>
        </table>
        </p>
</body>
</html>
```

Kết quả hiển thị



b. Width:

Định độ rộng của bảng, cột, ô

c. Height:

Định độ cao của bảng, cột, ô

Ví dụ 21

```
table { width:100%;}
th { height:50px; }
```

d. Text-align:

Căn lề cho text theo phương ngang trong ô của bảng gồm các giá trị như left, right, center

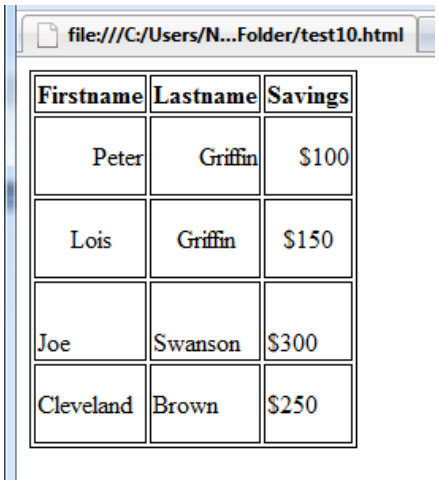
e. Vertical-align:

Căn lề cho text theo phương đứng trong ô của bảng gồm các giá trị như top, bottom, middle

Ví dụ 22:

```
<html >
<head>
<style type="text/css">
table, td, th { border:1px solid black;}
td{ height:50px;}
</style>
</head>
<body>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Savings</th>
</tr>
<tr style="text-align:right;">
<td>Peter</td>
<td>Griffin</td>
<td>$100</td>
</tr>
<tr style="text-align:center">
<td>Lois</td>
<td>Griffin</td>
<td>$150</td>
</tr>
<tr style="vertical-align:bottom;">
<td>Joe</td>
<td>Swanson</td>
<td>$300</td>
</tr>
<tr style="vertical-align:middle;">
<td>Cleveland</td>
<td>Brown</td>
<td>$250</td>
</tr>
</table>
</body>
</html>
```

Kết quả hiển thị



f. Padding:

Xác định khoảng cách giữa text và đường biên của ô trong bảng

Ví dụ 23:

```
td{ padding:15px; }
```

g. Background-color:

Màu nền ô

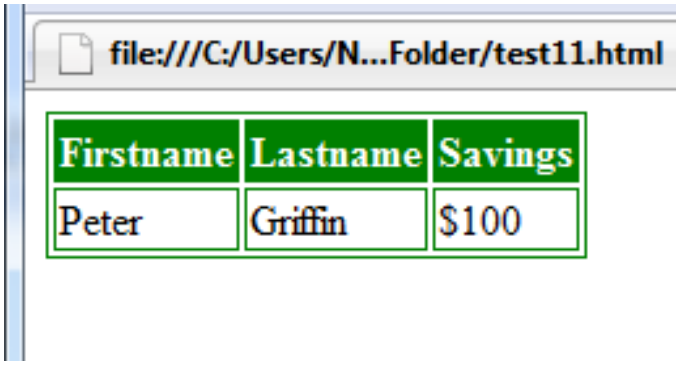
h. Color:

Màu chữ

Ví dụ 24:

```
<html>
<head>
<style type="text/css">
        table, td, th { border:1px solid green; }
        th { background-color:green; color:white; }
</style>
</head>
<body>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Savings</th>
</tr>
<tr>
<td>Peter</td>
<td>Griffin</td>
<td>$100</td>
```

```
</tr>
</table>
</body>
</html>
```



7. Thuộc tính Id và class của thẻ

Khi áp dụng một thuộc tính CSS cho một thành phần như p, a, img,… thì toàn bộ các thành phần này trong trang web đều nhận thuộc tính này. Nếu ta muốn một thành phần nào đó như liên kết trên menu có các thuộc tính khác với liên kết trong phần nội dung thì ta sẽ nhóm các thuộc tính đó vào trong id hoặc class

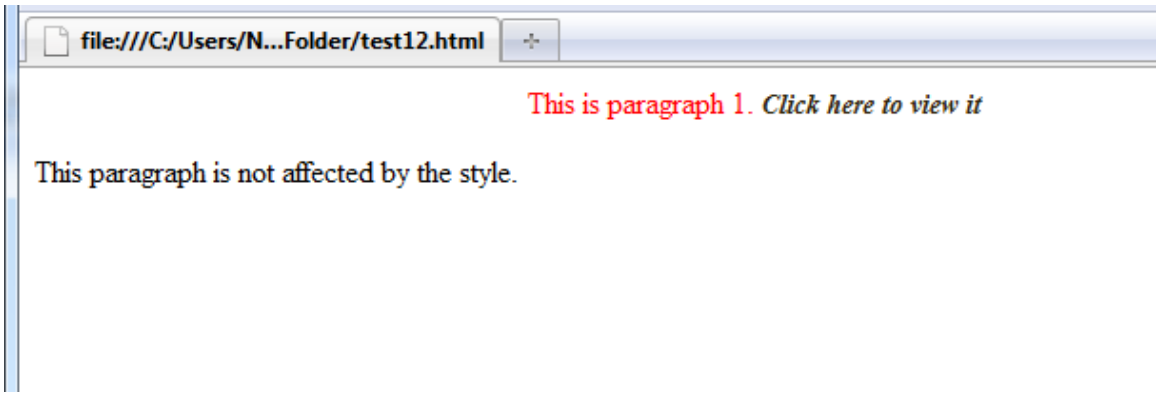
a. Thuộc tính Id

Id được dùng cho một đối tượng riêng biệt, và được xác định bởi ký hiệu "#".

Ví dụ 25

```
<html>
<head>
<style>
#para1
{
	text-align:center;
	color:red;
}
#para1 a
{
	text-decoration:none;
	color:#2e260f;
	font-weight:bold;
	font-style:italic;
	font-size:14px;
}
</style>
</head>
<body>
	<p id="para1">This is paragraph 1. <a href="id.html">Click here to view it</a></p>
```

```
    <p>This paragraph is not affected by the style.</p>
</body>
</html>
```



b. Thuộc tính Class

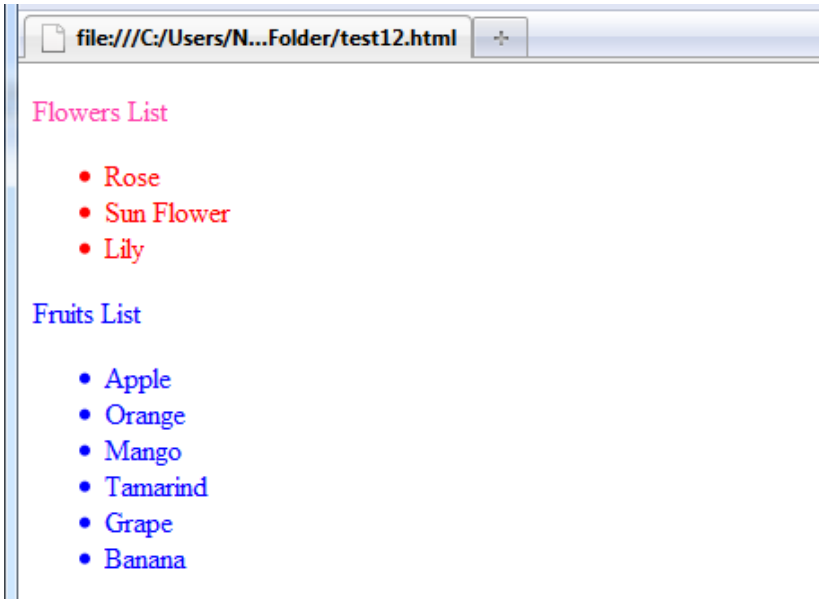
Class được dùng để nhóm một số thành phần có những thuộc tính đặc biệt. Nhưng khác với id, class được sử dụng cho nhiều đối tượng khác nhau và được xác định bởi ký hiệu "."

Ví dụ sau sẽ cho thấy rõ hơn sự khác biệt giữa id và class

Ví dụ 26

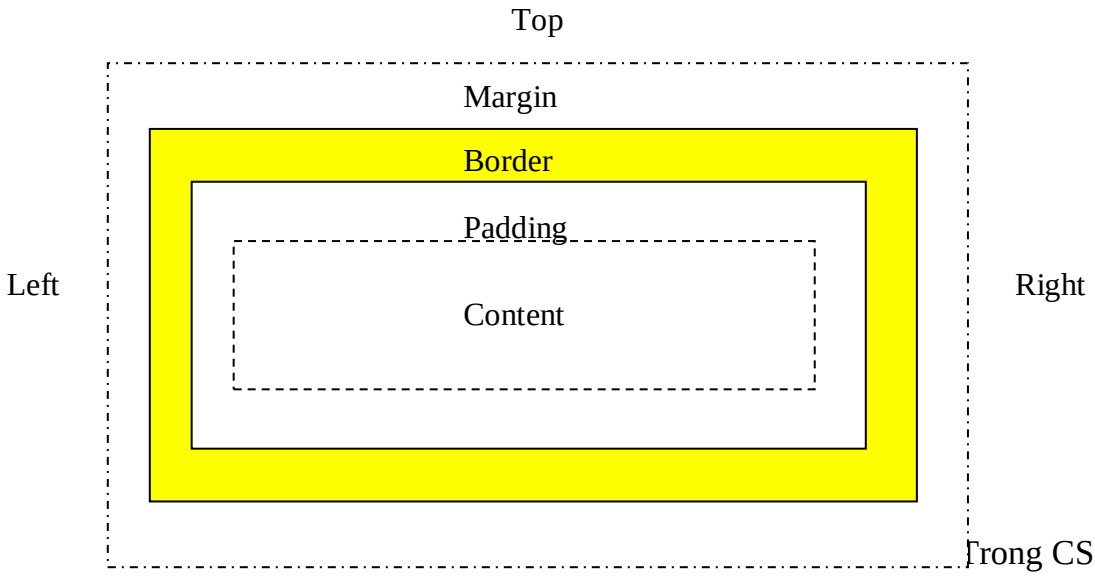
```
<html>
<head>
<style>
    #flower {color:#ff0000;}
    #flower p{ color:#ff3faa;}
    .fruit { color:#0000FF }
</style>
</head>
<body>
<div id="flower">
    <p>Flowers List</p>
    <ul>
    <li>Rose</li>
    <li>Sun Flower</li>
    <li>Lily</li>
  </ul>
</div>

<p class="fruit">Fruits List</p>
<div class="fruit">
<ul>
    <li >Apple</li>
    <li >Orange</li>
    <li >Mango</li>
```

```
        <li >Tamarind</li>
        <li >Grape</li>
        <li >Banana</li>
</ul>
</div>
</body>
</html>
```



* Lưu ý: không nên dùng kí tự đầu là chữ số đặt tên cho class và id

8. Mô hình hộp



Trong CSS, box model (mô hình hộp) mô tả cách mà CSS định dạng khối không gian bao quanh một thành phần. Nó bao gồm padding (vùng đệm), border (viền) và margin (canh lề) và các tùy chọn. Hình bên dưới mô tả cấu trúc minh họa mô hình hộp cho một thành phần web.

a. Thuộc tính margin:

Thuộc tính margin được dùng để canh lề cho một thành phần web hay cả trang web so với các đối tượng bên ngoài. Các thuộc tính về margin gồm:

- margin-top
- margin-right
- margin-bottom
- margin-left

Các đơn vị được dùng với thuộc tính margin:

- Auto: trình duyệt tự động thiết lập margin. Kết quả được thể hiện tùy thuộc vào trình duyệt
- Pixel, pt, em,…: thiết lập khoảng cách lề theo độ dài
- %: thiết lập khoảng cách lề theo phần trăm

Ví dụ 27

```
Body
{
	margin-top:80px;
	margin-bottom:40px;
	margin-left:50px;
	margin-right:30px;
	border:1px dotted #FF0000
}
Hoặc:
	body { margin:80px 30px 40px 50px; border:1px dotted #FF0000 }
```

b. Thuộc tính padding

Thuộc tính padding dùng để tạo khoảng cách giữa các nội dung bên trong với đường biên của khối. Các thuộc tính về padding gồm:

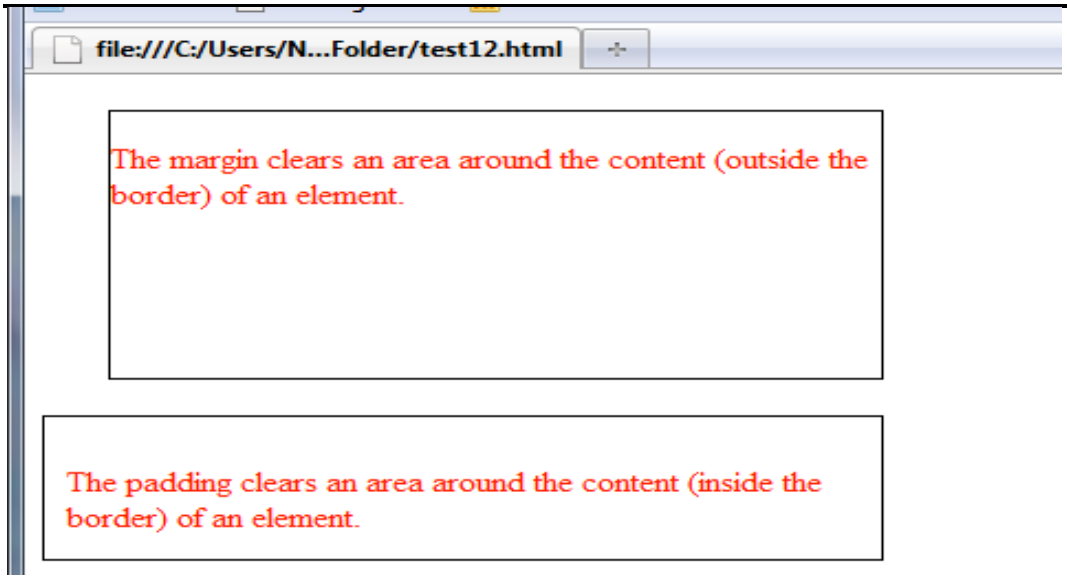
- padding-top
- padding-right
- padding-bottom
- padding-left.

Các đơn vị được dùng với thuộc tính padding: %, px, pt, em,…

Ví dụ 28

```
<html>
```

```
<head>
<style type="text/css">
	.padding
	{
		border:1px solid #000000;
		padding: 10px 20px 20px 10px;
		color:#FF1F00;
		height:50px;
		width:350px;
	}
	#margin
	{
		border:1px solid #000000;
		margin-top: 20px;
		margin-right:30px;
		margin-left:30px;
		margin-bottom:20px;
		color:#FF1F00;
		height:150px;
		width:350px;
	}
</style>
</head>
<body>
	<div id="margin">
		<p>The margin clears an area around the content (outside the border) of an element.</p>
	</div>
	<div class="padding">
		<p>The padding clears an area around the content (inside the border) of an element.</p>
	</div>
</body>
</html>
```



c. Border

Trong ví dụ trên ta thấy thuộc tính border được dùng để đóng khung, trang trí cho một đối tượng, phân cách các đối tượng giúp trang web trông dễ nhìn hơn. Border có các thuộc tính sau:

- **border-width:** độ rộng cho viền, có các giá trị thin (mảnh), medium (vừa), thick (dày), hay là một giá trị đo cụ thể như pixels
- **border-color:** màu viền
- **border-style:** kiểu viền, có 8 giá trị dotted, dashed, solid, double, groove, ridge, inset và outset. Ngoài ra, hai giá trị none hay hidden dùng để ẩn đường viền

Chúng ta cũng có thể dùng riêng các thuộc tính border-top, border-right, border-bottom hay border-left để chỉ định viền riêng cho các đối tượng.

Ví dụ 29

```
#border
{
border-top-width:thin;
border-top-color:#FF0000;
border-top-style:solid;
border-right-width:thick;
border-right-color:#AFAFAF;
border-right-style:dotted;
}
```

d. Thuộc tính Width và Height

Thuộc tính Width

Width quy định độ rộng cho một thành phần web, ngoài ra ta còn có một số thuộc tính đi kèm

- max-width: quy định chiều rộng tối đa cho một thành phần web.
- min-width: quy định chiều rộng tối thiểu cho một thành phần web.

Thuộc tính height:

Height: quy định chiều cao cho một thành phần web, ngoài ra ta còn có một số thuộc tính đi kèm

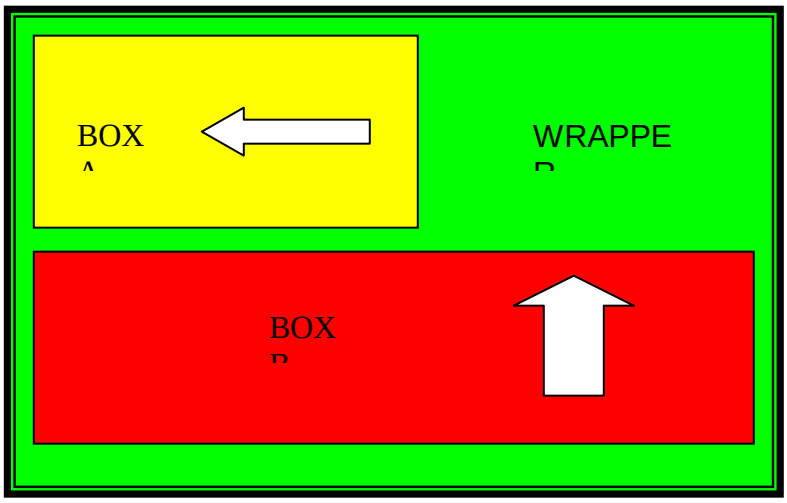
- max-height: quy định chiều cao tối đa cho một thành phần web.
- min-height: quy định chiều cao tối thiểu cho một thành phần web.

e. Thuộc tính float và clear

Thuộc tính Float:

Thuộc tính float dùng để cố định một thành phần web về bên trái hay bên phải không gian bao quanh nó. Đây là một thuộc tính rất cần thiết khi dàn trang, hiển thị văn bản thành cột, định vị trí ảnh và text

Trong hình minh họa dưới đây ta thấy 2 khối BOX A và BOX B được đặt trong khối WRAPPER. Khi ta sử dụng thuộc tính float cho BOX A cố định về phía trái thì BOX B sẽ tràn lên để lấp khoảng trống phía trên



Thuộc tính float có 3 giá trị:

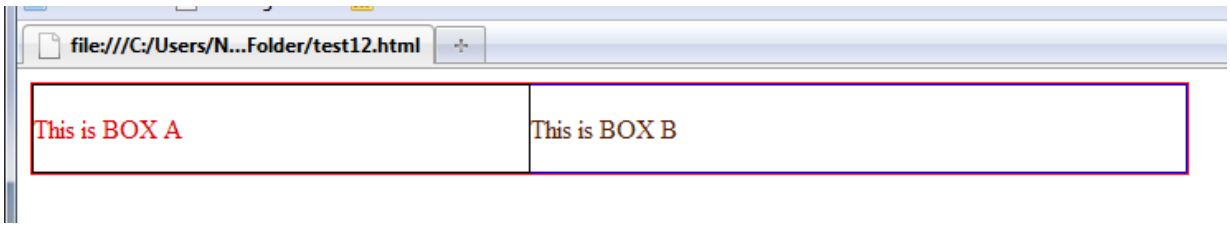
- Left: Cố định phần tử về bên trái.
- Right: Cố định phần tử về bên phải.
- None: Bình thường.

Ví dụ 30

```
<html>
<head>
<style type="text/css">
```

```
        #wrapper
        {
                border:1px solid #ff0000;
                width:700px;
        }
        #box_a
        {
                border: 1px solid #000000;
                width: 300px;
                float:left;
                color:#d40000;
        }
        #box_b
        {
                border: 1px solid #0000ff;
                color:#551f00;
        }
</style>
</head>

<body>
        <div id="wrapper">
                <div id="box_a">
                        <p>This is BOX A</p>
                </div>
                <div id="box_b">
                                <p>This is BOX B</p>
                </div>
        </div>
</body>
</html>
```

Kết quả hiển thị



Thuộc tính clear

Thuộc tính clear thường đi kèm với float, được dùng xử lý các phần tử liên quan tới phần tử đã được float để quyết định hướng xử sự của phần tử này. Ở ví dụ trên, khi BOX A được float qua trái thì mặc nhiên BOX B sẽ được tràn lên để lắp vào chỗ trống. Nhưng khi chúng

ta sử dụng thuộc tính clear cho BOX B thì chúng ta có quyền quyết định xem phần văn bản đó có được tràn lên hay không.

Thuộc tính clear có 4 giá trị: left, right, both và none.

Thuộc tính position

Bên cạnh thuộc tính float, clear, CSS cung cấp cho ta thuộc tính position để xác định tọa độ của một đối tượng nào đó trên cửa sổ trình duyệt. Ta có 2 cách để xác định tọa độ:

- Định vị tuyệt đối: position sẽ nhận giá trị absolute
- Định vị tương đối: position nhận giá trị relative

Ví dụ 31

```
#img1{position:absolute; top:50px; left:70px }
#img2 { position:relative; bottom:70px; right:50px }
```

Chương 4: Giới thiệu ngôn ngữ kịch bản Javascript

I. Giới thiệu về Javascript

- Javascript là ngôn ngữ kịch bản được sử dụng nhiều trên các website, và được hỗ trợ trên một số trình duyệt như: Internet Explorer, FireFox, Chrome, Opera, và Safari.

- Để học ngôn ngữ này bạn cần hiểu cơ bản về HTML hoặc XHTML.

Vậy javascript là gì?

- javascript là ngôn ngữ kịch bản được nhúng vào trong tài liệu html.

II. Ngôn ngữ javascript

1. Chèn mã lệnh javascript vào trong tài liệu HTML

a. Chèn mã lệnh trên vùng <body>

Cách chèn mã lệnh nay chỉ áp dụng khi mã lệnh javacript được chèn thực hiện một mục đích nào đó trên tài liệu html tại vị trí cần chèn vào.

Ví dụ 4.1: Chèn mã lệnh javascript vào trong tài liệu html

```
<body>
<script language="javascript" type="text/javascript">
// code here.
</script>
</body>
```

b. Chèn mã lệnh trên vùng <head>

Cách chèn mã lệnh nay thường khi mã lệnh javacript được chèn thực hiện một mục đích nào tại nhiều vị trí khác nhau trên tài liệu html.

Ví dụ 4.2: Chèn mã lệnh javascript vào trong vùng thẻ head

```
<head>
<script language="javascript" type="text/javascript">
// code here.
</script>
</head>
```

c. Chèn mã lệnh trực tiếp vào trong các thẻ HTML

Cách chèn mã lệnh nay chỉ áp dụng khi mã lệnh javacript được chèn thực hiện một mục đích nào đó trên thẻ html được chèn vào.

Ví dụ 4.3: Chèn mã lệnh javascript vào trong thẻ html

```
<p onclick="alert('Xin chao cac ban');"> Click here!!!</p>
```

d. Chèn mã lệnh bằng một tập tin riêng trên vùng <head>

Cách chèn mã lệnh nay thường khi mã lệnh javacript được chèn thực hiện một mục đích nào tại nhiều vị trí khác nhau trên tài liệu html, bên cạnh đó mã lệnh trong tập tin này không chỉ áp dụng cho một trang bất kỳ mà có thể áp dụng cho toàn bộ website.

Ví dụ 4.4: Chèn mã lệnh bằng một tập tin riêng có tên my_javascript.js

```
<head>
<script language="javascript" src="my_javascript.js" type="text/javascript">
// code here.
</script>
</head>
```

2. Lời chú thích

Chúng ta có thể thêm những khối ghi chú để biết phần mã lệnh tương ứng thực hiện điều gì. Các ghi chú được trình duyệt bỏ qua và chỉ thấy trong mã nguồn.

Cú pháp câu ghi chú:

- Dòng ghi chú nằm trên một dòng văn bản.

```
// dòng ghi chú trên 1 dòng.
```

- Dòng ghi chú nằm trên nhiều dòng văn bản.

```
/* dòng ghi chú thứ nhất
 dòng ghi chú khác….  */
```

3. Biến và cách xuất thông tin lên trình duyệt

a. Biến và cách khai báo biến

Biến trong javascript được sử dụng từ khóa var để khai báo, khi khai báo nhiều biến chúng ta phân cách chúng bởi dấu (,).

Tên biến không có khoảng cách trống, không được trùng tên với từ khóa, bắt đầu bằng ký tự, đặc điểm tên biến phân biệt chữ hoa và chữ thường.

Ví dụ 4.5:

```
var so_a, so_b;
var chuoi;
```

Sau khi khai báo biến chúng ta có thể khởi gán giá trị cho tên biến bằng cách dùng toán tử gán(=), khi đó kiểu dữ liệu của biến là kiểu của giá trị được khởi gán.

Ví dụ 4.6:

so_a = 9; //kiểu của so_a là kiểu số nguyên.

chuoi = "Hello" // kiểu của chuoi là kiểu chuỗi ký tự.

Để kiểm tra xem kiểu dữ liệu của tên biến ta sử dụng hàm typeof(<tên_biến>).

Chuyển đổi chuỗi số thành kiểu số ta sử dụng hàm parseInt() hoặc parseFloat().

Ví dụ 4.7:

```
chuoi = "123.45";
so_a = parseInt(chuoi); //khi đó so_a = 123;
so_b = parseFloat(chuoi); //khi đó so_b =123.45;
```

Chuyển đổi số thành chuỗi ta sử dụng hàm toString()

Ví dụ 4.8:

```
chuoi = toString(so_a); // khi đó chuoi = "123";
```

b. Xuất thông tin lên trình duyệt web

Để xuất thông tin lên trình duyệt web ta sử dụng Cú pháp sau:

document.write(<nội dung>);

hoặc document.writeln(<nội dung>);

Ví dụ 4.9:

```
document.write("Chào các bạn"); // xuất ra chuỗi.
document.write(chuoi); // xuất ra biến chuỗi.
```

4. Các phép toán

Trong ngôn ngữ lập trình javascript các phép toán tương tự như các phép toán trong C.

Các phép tính toán học:

+	Phép cộng
-	Phép trừ
*	Phép nhân
/	Phép chia
%	Phép lấy phần dư
++	Tăng giá trị lên 1 đơn vị
--	Giảm giá trị xuống 1 đơn vị

Các phép gán:

Phép toán	Ví dụ	Tương tự	Kết quả
=	x=y		x=5

+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

Phép toán + : được sử dụng để cộng chuỗi(nối chuỗi)

Ví dụ 4.10: Cộng hai chuỗi

```
st1="Hôm nay là";
st2="đẹp  trời";
st=st1+ “  ”+st2; // Kết quả st = “Hôm nay là một ngày đẹp trời”;
```

Phép cộng giữa chuỗi và số: khi cộng một chuỗi với một số kết quả sẽ là chuỗi.

Ví dụ 4.11: Cộng chuỗi và số

```
//Cộng số với số
x=5+5;
document.write(x);
// Cộng chuỗi với chuỗi
x="5"+"5";
document.write(x);
// Cộng số với chuỗi
x=5+"5";
document.write(x);
// Cộng chuỗi với số
x="5"+5;
document.write(x);
```

Phép so sánh:

==	Bằng
===	Bằng chính xác (giá trị và kiểu)
!=	Không bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng

Phép toán logic:

&&	Và
||	Hoặc
!	Phủ định

Phép toán điều kiện:

ten_bien =(<điều kiện>)?gia_tri_true:gia_tri_sai

5. Câu lệnh rẽ nhánh If...Else

Câu lệnh rẽ nhánh có 3 dạng:

Dạng 1: Câu lệnh if dạng khuyết.

Câu lệnh if dạng này được sử dụng để kiểm tra điều kiện, nếu điều kiện thõa mãn thì thực hiện một nhiệm vụ nào đó.

Cú pháp:

```
if (<Điều kiện>)
 {
   // mã lệnh nếu biểu thức điều kiện đúng
  }
```

Ví dụ 4.12: Lấy ra giờ hệ thống, nếu giờ nhỏ hơn 10 giờ thì in ra trình duyệt "Chào buổi sáng"

```
<script type="text/javascript">
var d=new Date(); // Lớp d là kiểu dữ liệu giờ
var time=d.getHours(); // lấy ra giờ
if (time<10)
 {
  document.write("<b>Chào buổi sáng</b>");
 }
</script>
```

Dạng 2: Câu lệnh if dạng đầy đủ

Câu lệnh if dạng này được sử dụng để kiểm tra điều kiện, nếu điều kiện thõa màn thì thực hiện một công việc này còn nếu sai thì thực hiện nhiệm vụ khác.

Cú pháp:

```
if (<Điều kiện>)
  {
```

```
 // Mã lệnh nếu điều kiện đúng;
 }
else
 {
 //Mã lệnh nếu điều kiện sai
 }
```

Ví dụ 4.13: Lấy ra giờ hệ thống, nếu giờ hiện tại nhỏ hơn 10 thì xuất ra màn hình là "Good morning!", ngược lại "Good day!"

```
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time < 10)
 {
 document.write("Good morning!");
 } else {
  document.write("Good day!");
 }
</script>
```

Dạng 3: Câu lệnh if lồng nhau:

Câu lệnh if dạng lồng được sử dụng khi điều kiện đưa ra có thể xẩy ra hơn hai trường hợp.

Cú pháp:

```
if (<Điều kiện1>)
 {
  // Mã lệnh nếu biểu thức <Điều kiện 1> đúng;
 } else if (<Điều kiện2>)
 {
  // Mã lệnh nếu biểu thức <Điều kiện 2> đúng;
 } else
 {
  // Mã lệnh nếu biểu thức <điều kiện 1> và <điều kiện 2> sai.
  }
```

Ví dụ 4.14:

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
```

```
 {
 document.write("<b>Good morning</b>");
 }
else if (time>10 && time<16)
 {
 document.write("<b>Good day</b>");
 }
else
 {
 document.write("<b>Hello World!</b>");
 }
</script>
```

6. Câu lệnh lựa chọn Switch

Câu lệnh switch được sử dụng khi biểu thức có thể trả về nhiều giá trị khác nhau, mỗi giá trị như vậy thì thực hiện một công việc khác nhau.

Cú pháp:

```
switch(<biểu_thức>)
{
case gt1:
 //Thực hiện lệnh nếu biểu_thức = gt1
 break;
case gt2:
 //Thực hiện lệnh nếu biểu_thức = gt2
 break;
default:
 //Thực hiện lệnh nếu biểu_thức không bằng gt1 hoặc gt2
}
```

Ví dụ 4.15:

```
<script type="text/javascript">
var d=new Date();
theDay=d.getDay();
switch (theDay)
{
case 1:  document.write("Thứ hai");
 break;
case 2:  document.write("Thứ ba");
```

```
 break;
case 3:  document.write("Thứ tư");
 break;
case 4:  document.write("Thứ năm");
  break;
case 5:  document.write("Thứ sáu");
 break;
case 6:  document.write("Thứ bảy");
 break;
default:
  document.write("Chủ nhật");
}
</script>
```

7. Định nghĩa hàm

Ngoài những hàm javascript đã định nghĩa sẵn, chúng ta có thể định nghĩa để thực hiện một nhiệm vụ nào đó.

Cấu trúc của hàm:

```
function <tên hàm>(<tham số nếu có>)
{
	// mã lệnh.
}
```

8. Hộp thông báo

Khi người dùng nhập thông tin hoặc tác động đến trang web thì một thông báo hiện ra cảnh báo hay nhắc nhở người.

Kiểu thông báo: Hộp thông báo alert() chỉ có nút lệnh OK.

Cú pháp:

```
alert("Dòng thông báo");
```

Ví dụ 1.16:

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("I am an alert box!");
```

```
}
</script>
</head>
<body>
<input type="button" onclick="show_alert()" value="Show alert box" />
</body>
</html>
```

Kiểu thông báo confirm() có hai nút lệnh nút lệnh OK và Cancel.

Nếu Click vào Ok thì kết quả trả về là true, nếu click vào Cancel thì kết quả trả về là false.

Cú pháp:

```
confirm("Dòng thông báo");
```

Ví dụ 4.17:

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
 {
  document.write("You pressed OK!");
  } else   {
  document.write("You pressed Cancel!");
  }
}
</script>
</head>
<body>
<input type="button" onclick="show_confirm()" value="Click here" />
</body>
</html>
```

Kiểu thông báo prompt() cho phép người dùng nhập vào giá trị và chỉ có hai nút lệnh OK và Cancel.

Nếu click vào Ok thì kết quả trả về là giá trị được nhập từ hộp prompt(), nếu click vào Cancel thì giá trị trả về là null.

Cú pháp:

```
prompt("Dòng thông báo","Giá trị nhập vào mặc định");
```

Ví dụ 4.18:

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
 {
 document.write("Hello " + name + "! How are you today?");
 }
}
</script>
</head>
<body>
<input type="button" onclick="show_prompt()" value="Show prompt box" />
</body>
</html>
```

9. Câu lệnh lặp For

Câu lệnh for dùng để lặp lại công việc với số lần lặp được xác định trước.

Cú pháp:

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
 // Mã lệnh thực hiện;
}
```

Ví dụ 4.19:

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
```

```
document.write("<br />");
}
</script>
</body>
</html>
```

10. Câu lệnh lặp While

Câu lệnh lặp While:

Câu lệnh while dùng để lặp lại công việc với số lần lặp chưa xác định trước, số lần lặp phục thuộc vào điều kiện.

Cú pháp:

```
while (<biểu thức điều kiện>)
 {
   // Thực hiện mã lệnh;
  }
```

Kiểm tra điều kiện, nếu biểu thức điều kiện còn đúng thì thực hiện mã lệnh, điều kiện sai thì thoát khỏi vòng lặp.

Ví dụ 4.20:

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
 {
 document.write("The number is " + i);
 document.write("<br />");
 i++;
 }
</script>
</body>
</html>
```

Câu lệnh do ... while

Thực hiện mã lệnh, sau đó kiểm tra điều kiện nếu điều kiện còn chưa đúng thì thực hiện thì thực hiện mã lệnh, điều kiện đúng thì thoát khỏi vòng lặp.

Cú pháp:

```
do
 {
 // Thực hiện mã lệnh;
 }
while (var<=endvalue);
```

Ví dụ 4.21:

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
while (i<=5);
</script>
</body>
</html>
```

11. Câu lệnh lặp For...In

Câu lệnh for …in là một câu lệnh đặc biệt không có trong C chỉ có trong javascript, nó được dùng để duyệt các phần tử trong một đối tượng trong javascript. Ví dụ như trong đối tượng mảng.

Cú pháp:

```
for (variable in object)
  {
  // Mã lệnh thực hiện;
  }
```

Ví dụ 4.22:

```
<html>
<body>
<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
```

```
mycars[1] = "Volvo";
mycars[2] = "BMW";
for (x in mycars)
 {
  document.write(mycars[x] + "<br />");
 }
</script>
</body>
</html>
```

12. Sự kiện trong Javascript

a. Sự kiện onLoad và onUnload

Sự kiện	Chức năng
onFocus	
onBlur	Sự kiện onBlur được thực hiện khi người dùng con trỏ di chuyển ra khỏi thẻ.
onChange	
onClick	
onActive	
onLoad	Sự kiện onLoad được thực hiện khi người dùng tải dữ liệu lên trình duyệt web.
onUnload	Sự kiện onUnload được thực hiện khi người dùng đó lại quá trình tải dữ liệu lên trình duyệt web.
onKeypress	
onSubmit	Sự kiện onSubmit được thực hiện khi người dùng thực hiện submit một form.
MouseOver	Sự kiện onMouseover được thực hiện khi người dùng đưa chuột lên thẻ.
onMouseOut	Sự kiện onMouseout được thực hiện khi người dùng đưa chuột ra khỏi thẻ.

13. Câu lệnh Try...Catch

Câu lệnh này kiểm tra một đoạn mã lệnh có bị lỗi Cú pháp hay không, nếu bị lỗi thì thông báo lỗi phát sinh ra.

Cú pháp:

```
try
  {
  //mã lệnh
  }
catch(err)
  {
  //thông báo lỗi
```

```
 }
```

Khi chạy thử "mã lệnh" nếu mã lệnh bị lỗi Cú pháp thì "thông báo lỗi" sẽ đưa ra.

Ví dụ 4.23:

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Error description: " + err.description + "\n\n";
  txt+="Click OK to continue.\n\n";
  alert(txt);
 }
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

14. Câu lệnh Throw

Cú pháp:

```
throw(exception)
```

Ví dụ 4.24:

```
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 0 and 10:","");
```

```
try
  {
  if(x>10)
    {
    throw "Err1";
    }
  else if(x<0)
    {
    throw "Err2";
    }
  else if(isNaN(x))
    {
    throw "Err3";
    }
  }
catch(er)
  {
  if(er=="Err1")
    {
    alert("Error! The value is too high");
    }
  if(er=="Err2")
    {
    alert("Error! The value is too low");
    }
  if(er=="Err3")
    {
    alert("Error! The value is not a number");
    }
  }
</script>
</body>
</html>
```

15. Ký tự đặc biệt Text

Mã	Hiện ra
\'	Dấu nháy đơn (’)

\"	Dấu nháy kép (”)
\&	Dấu và (&)
\\	Dấu xổ phải (\ )
\n	Xuống dòng mới
\r	carriage return
\t	Nhảy tab
\b	Khoảng trắng
\f	form feed

III. Đối tượng trong javascrip

Javascript là một ngôn ngữ lập trình hướng đối tượng, cho phép chúng ta định nghĩ ra đối tượng và sử dụng đối tượng đó, bên cạnh đó javascript cũng định nghĩa cho chúng ta một số đối tượng.

1. Đối tượng String

Đối tượng chuỗi được tạo ra để lưu trữ chuỗi các ký tự.

Ví dụ 4.25:

hoten = “Hồ Diên Lợi”;

Các khái báo chuỗi: Đối tượng chuỗi được khai báo như sau:

var ten_chuoi = “chuỗi ký tự”;

Ví dụ 4.26:

var hoten = “Hồ Diên Lợi”;

Một phương thức của đối tượng chuỗi:

- Độ dài của chuỗi: length
- Cộng hai chuỗi ký tự (+)
- Chuyển chuỗi ký tự thành chuỗi chữ in hoa: toUpperCase()
- Chuyển chuỗi ký tự thành chuỗi chữ thường: toLowerCase( )
- Cắt lấy chuỗi con từ vt1 đến vt2: substring(vt1,vt2)
- Cắt lấy một chuỗi: substr(vt, number)
- indexOf( )
- match(re);

- replace(re, *replacementString*);
- charCodeAt(3)

2. Đối tượng Date

Tạo một đối tượng date ta sử dụng các Cú pháp sau:

```
var myDate = new Date();
var myDate = new Date(yyyy, mm, dd, hh, mm, ss);
var myDate = new Date(yyyy, mm, dd);
var myDate = new Date("monthName dd, yyyy hh:mm:ss");
var myDate = new Date("monthName dd, yyyy");
var myDate = new Date(epochMilliseconds);
```

Một số phương thức đối với đối tượng Date:

getTime( )	setTime(val)	0-...
getSeconds( )	setSeconds(val)	0-59
getMinutes( )	setMinutes(val)	0-59
getHours( )	setHours(val)	0-23
getDay( )	setDay(val)	0-6
getDate( )	setDate(val)	1-31
getMonth( )	setMonth(val)	0-11
getFullYear( )	setFullYear(val)	1970-...

3. Đối tượng Array

- Concat(): nooid chuỗi

Ví dụ 4.27: Nối chuỗi

```
<html>
<body>
<script type="text/javascript">
var parents = ["Jani", "Tove"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(children);
document.write(family);
</script>
</body>
</html>
```

- Sort

Ví dụ 4.28:

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.sort());
</script>
</body>
</html>
```

Ví dụ 4.29:

- Split
- Length
- splice

4. Đối tượng Math

- Hàm round()

```
<html>
<body>
<script type="text/javascript">
document.write(Math.round(0.60) + "<br />");
document.write(Math.round(0.50) + "<br />");
document.write(Math.round(0.49) + "<br />");
document.write(Math.round(-4.40) + "<br />");
document.write(Math.round(-4.60));
</script>
</body>
</html>
```

- Hàm random()

```
<html>
<body>
<script type="text/javascript">
//return a random number between 0 and 1
document.write(Math.random() + "<br />");
//return a random integer between 0 and 10
document.write(Math.floor(Math.random()*11));
</script>
</body>
</html>
```

Chương 5: Ngôn ngữ PHP

I. Tổng quan về PHP

1. Cú pháp PHP

Cách 1 : Cú pháp chính:

<?php Mã lệnh PHP **?>**

Cách 2: Cú pháp ngắn gọn

<? Mã lệnh PHP **?>**

Cách 3: Cú pháp giống với ASP.

<% Mã lệnh PHP **%>**

Cách 4: Cú pháp bắt đầu bằng script

<script language=php>

.....

</script>

Mặc dù có 4 cách thể hiện. Nhưng đối với 1 lập trình viên có kinh nghiệm thì việc sử dụng cách 1 vẫn là lựa chon tối ưu.

2. Xuất giá trị ra trình duyệt

Để xuất dữ liệu ra trình duyệt chúng ta sử dụng các hàm hàm sau:

- Hàm **echo()**

Cú pháp:

```
echo (<thông tin>);
```

Trong đó:

+ Thông tin có thể là hằng, biến hay biểu thức.

+ Có thể viết **echo** <thông tin>;

Ví dụ 5.1: Xuất ra trình duyệt nội dung "Chào các bạn"

```
<?php
    echo "Chào các bạn";
    echo ("Chào các bạn");
?>
```

- Hàm **print()**

Cú pháp:

```
print(<thông tin>);
```

Trong đó:

+ Thông tin có thể là hằng, biến hay biểu thức.

+ Có thể viết print <thông tin>;

Ví dụ 5.2: Xuất ra trình duyệt nội dung "Chào các bạn"

```
<?php
    print "Chào các bạn";
    print ("Chào các bạn");
?>
```

3. Lời chú thích

Để giải thích các đoạn mã hay câu lệnh trong PHP thực hiện chức năng, nhiệu vụ ngông ngữ đã có pháp người dùng viết các ghi chú.

Lời ghi chú không thực thi khi chạy đoạn mã.

Trong HTML, các ghi chú được đặt trong <!-- phần ghi chú -->

Trong PHP có ba dạng ghi chú:

Dạng 1: # đây là ghi chú.

Dạng này chỉ áp dụng ghi đó chỉ nằm trên một dòng văn bản

Dạng 2: // đây là ghi chú.

Dạng này cũng chỉ áp dụng ghi đó chỉ nằm trên một dòng văn bản

Dạng 3: /* đây là một ghi chú dài

Áp dụng cho nhiều hàng */

Dạng ghi chú này thường áp dụng cho đoạn ghi chú dài gồm nhiều dòng văn bản

4. Biến trong PHP

Biến là một ô nhớ trong bộ nhớ chính giúp biểu diễn thông tin thực tế trong chương trình.

a. Khai báo biến

- Biến trong PHP được bắt đầu bằng ký tự $ và theo sau là tên biến.

- Nguyên tắc đặt tên biến: tên biết bắt đầu bằng một ký tự hoặc là dấu _theo sau có thể là ký tự, số hoặc dấu _

Chú ý:

PHP không yêu cầu khai báo biến trước khi sử dụng, tuy nhiên chúng ta nên khai báo và khởi gán giá trị ban đầu cho biến trước khi sử dụng.

Không khai báo tên biến trùng với tên hàm, tên biến phân biệt chữ HOA và chữ thường.

Ví dụ 5.3: Khai báo biến

```
<?php
        $chuoi = "Hello world";
        $_123 = 123.4567;
?>
```

b. Gán giá trị cho biến.

Khi ta gán giá trị cho biến, nếu biến chưa được khai báo lúc này biến vừa được khai báo vừa được khởi tạo, nếu biến đã được khai báo thì biến này sẽ có giá trị bằng với giá trị được gán.

- Biến có thể khởi gán giá trị trực tiếp:

```
$tên_biến = <giá trị của biến>;
```

Ví dụ 5.4: Khởi gán giá

```
<?php
        $number = 120;
        echo $number;
?>
```

- Biến có thể khởi gán giá trị của biến khác hoặc của một biểu thức.

```
$tên_biến =$tên_biến_khác hoặc biểu thức;
```

Ví dụ 5.5: Khởi gán giá trị của một biểu thức.

```
<?php
        $number = 120;
        $gia =1000;
        $thanh_tien =$number * $gia;
        echo $thanh_tien;
?>
```

c. Phạm vi hoạt động của biến

Một biến trong PHP có phạm vi hoạt động như sau:

- Biến cục bộ: Khi một biến được khai báo trong một hàm thì nó được xem là biến cục bộ và nó chỉ có ý nghĩa sử dụng trong hàm đó. Khi gán giá trị cho biến bên ngoài thì biến ngoài hàm này sẽ được xem như một biến hoàn toàn khác với biến trong hàm cho dù cùng tên. Khi ra khỏi hàm có biến cục bộ được khai báo thì biến và giá trị cho nó sẽ được hủy bỏ.

Ví dụ 5.6: Biến cục bộ

```
<?php
$a= 10; //biến toàn cục
function test()
{
	echo $a; // biến cục bộ
}
test(); // không cho kết quả
echo $a; // kết quả 10
?>
```

- Biến toàn cục: Là biến có thể truy cập ở bất kỳ nơi nào trong chương trình. Tuy nhiên, để có thể sử dụng và cập nhật được biến toàn cục thì phải được khai báo toàn cục (khai báo với global hay $_GLOBALS) trong hàm mà nó được sử dụng.

Ví dụ 5.7: Biến toàn cục

```
<?php
$a = 10;
$b = 20;
function sum()
{
	global $a, $b;
	$b = $a+$b;
}
sum();
echo $b; // kết quả 30
?>
```

- Biến static: khác với biến cục bộ, biến static không mất giá trị của nó khi ra khỏi hàm và nó giữa nguyên giá trị đó khi hàm đó được gọi thêm một lần nữa.

Ví dụ 5.8: Biến static

```
<?php
function hien()
{
	static $a =0;
```

```
        echo $a."</br>";
        $a++;
}
hien(); // kết quả 0
hien(); // kết quả 1
hien(); // kết quả 2
?>
```

5. Hằng

a. Khái báo hằng

Hằng là một giá trị không thể chỉnh sửa được thông qua việc thực hiện chương trình, quy tắc đặt tên hằng cũng giống như quy tắc đặt tên biến.

Chúng ta có thể định nghĩa hằng bằng cách sử dụng hàm **define()**. Một khi hằng được định nghĩa, nó không bị thay đổi.

Chỉ có các kiểu dữ liệu boolean, integer, float, string mới có thể chứa hằng.

Cú pháp:

```
define("TÊN_HẰNG", "giá trị");
```

Ví dụ 5.9: Tính diện tích đường tròn

```
<?php
define("PI", 3.14);
$r = 10;
echo "Diện tích đường tròn". 2*PI*$r*$r;
?>
```

Chú ý: PHP cung cấp một lượng lớn các hằng đã được định nghĩa trước để bất kỳ trang nào có thể thực thi được.

- _FILE_ : tên của script file đang được thực hiện.
- _LINE_: số dòng của mã script đang được thực hiện trong script file hiện tại.
- _PHP_VERSION_: version của PHP
- TRUE
- FALSE
- E_ERROR: báo hiệu có lỗi.
- E_PARSE: báo lỗi sai khi biên dịch.

- E_NOTICE: một vài sự kiện có thể là lỗi hoặc không.
- E_ALL: tất cả các lỗi.
- ….

b. Sử dụng hằng

Đối với hằng đã khai báo, chúng ta dùng tên hằng mỗi khi sử dụng.

Ví dụ 5.10: Tính diện tích và chu vi hình tròn

```
<?php
        define('PI',3.14); // định nghĩa PI = 3.14
        $r = 10;
        $chu_vi = 2*PI*$r; // 62.8
        echo $chu_vi."</br>";
        $dien_tich = 2*PI*$r*$r; //628
        echo $dien_tich;
?>
```

Những điểm khác nhau giữa hằng và biến:

- Phía trước tên hằng không có dấu $
- Hằng chỉ có thể được khai báo bằng hằng **define()**
- Không khai báo lại hằng khi đã được thiết lập.

6. Kiểu dữ liệu

a. Kiểu dữ liệu

Kiểu dữ liệu	Mô tả
boolean	Chỉ có một trong hai giá trị TRUE và FALSE
integer	Kiểu số nguyên, giá trị có thể là số trong hệ thập phân, thập lục phân và bát phân.
float/double	Kiểu số thực
string	Kiểu dữ liệu chuỗi, ký tự. Trong đó, mỗi ký tự chiếm 1 byte. Mỗi chuỗi có thể chứa một hay nhiều ký tự thuộc 256 ký tự khác nhau. Không có vấn đề gì xảy ra khi chuỗi quá lớn vì chuỗi không có giới hạn về kích thước. Mỗi chuỗi được ghi theo những cách sau: - Dùng dấu nháy đơn ' ' để bao chuỗi. - Dùng dấu nháy đôi " " để bao chuỗi.
array	Kiểu dữ liệu là mảng các phần tử.
object	Kiểu dữ liệu là đối tượng của lớp

Ví dụ 5.11: Kiểu dữ liệu số

```
<?php
        $kieu_b = TRUE;
        echo $kieu_b."</br>"; // kết quả là 1
        $inta = 1234;
        echo $inta."</br>"; // kết quả là 1234
        $intb = -123;
        echo $intb."</br>"; // kết quả là  -123
        $intc = 0123;
        echo $intc."</br>"; // kết quả là 83
        $intd = 0x1A;
        echo $intd."</br>"; // kết quả là 26
        $float_a = 1.234;
        echo $float_a."</br>"; // kết quả là  1.234
        $float_b = 1.2e3;
        echo $float_b."</br>"; // kết quả là 1200
        $float_c = 7E-10;
        echo $float_c."</br>"; // kết quả là 7.0E-10
?>
```

Ví dụ 5.12: Kiểu dữ liệu string

```
<?php
        $name = 2010;
        $chuoi1 ='Chúc mừng năm mới năm $name';
        $chuoi2 ="Chúc mừng năm mới năm $name";
        echo $chuoi1."</br>"; // Chúc mừng năm mới năm $name
        echo $chuoi2."</br>"; // Chúc mừng năm mới năm 2010
?>
```

Ví dụ 5.13: Kiểu dữ liệu mảng

```
<?php
$mang = array(1,2,3,4,5);
print_r($mang);
// hàm in ra giá tri của mảng theo dạng Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 )
?>
```

Ví dụ 5.14: Kiểu object

```
<?php
class ten_class
```

```
{
	function xuatchao()
	{
		echo "Hello my class!";
	}
}
$a = new ten_class();
$a ->xuatchao(); // kết quả Hello my class
?>
```

b. Chuyển đổi kiểu dữ liệu

Trong quá trình tính toán chúng ta có thể thực hiện việc chuyển đổi kiểu dữ liệu cho biến bằng cách ghi tên kiểu dữ liệu mà biến muốn chuyển đổi vào phía trước biến.

Thường chúng ta chuyển đổi kiểu dữ liệu cho biến vì trong quá trình tính toán kiểu dữ liệu cũ của biến có thể không còn phù hợp nữa.

Ví dụ 5.15: Chuyển đổi kiểu dữ liệu

```
<?php
	$don_gia = 5000;
	$so_luong = 10000;
	$thanh_tien = (double)($so_luong * $don_gia);
	echo $thanh_tien;
?>
```

7. Các toán tử

a. Toán tử toán học

Toán tử toán học gồm các phép toán sau:

+ (cộng), - (trừ), * (nhân), / (chia), % (chia lấy phần dư), ++ (tăng giá trị biến lên 1 đơn vị)
- - (giảm giá trị biến 1 đơn vị)

b. Toán tử nối chuỗi

c. Toán tử gán kết hợp

Phép toán	Ví dụ	Tương đương
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y

*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

d. Toán tử so sánh

Toán tử so sánh gồm các phép toán sau:

==	Bằng
!=	Không bằng
<>	Khác
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng

e. Toán tử logic

Toán tử logic gồm các phép sau:

&&	Và
||	So sánh
!	Phủ định

f. Toán tử @

Trong trương hợp biểu thức hay phép toán của chúng ta phát sinh lỗi, nếu chúng ta không muốn xuất hiện ra thông báo lỗi thì chúng ta dùng toán tử @

Ví dụ 5.16: Khi chưa sử dụng toán tử @

```
<?php
    $a = 10;
    $b = 0;
    $c =$a/$b;
    echo  "Kết quả :".$c;
?>
```

Màn hình xuất hiện:

Warning: Division by zero in **C:\wamp\www\vd.php** on line **4**

Kết quả :

Ví dụ 5.17: Khi sử dụng toán tử @

```
<?php
    $a = 10;
```

```
    $b = 0;
    $c =@($a/$b);
    echo  "Kết quả :".$c;
?>
```

Màn hình xuất hiện:

Kết quả :

g. Tham chiếu &

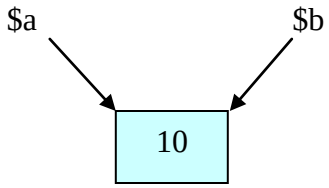
Trong PHP tham chiếu có nghĩa là lấy cùng giá một trị bằng nhiều tên biến khác nhau.

Ký hiệu tham chiếu là &.

Ví dụ 5.18: Biến $a và $b tham chiếu đến ô nhớ lưu giá trị biến $a

```
<?php
    $a = 10;
    $b = &$a;
    echo  "Kết quả :".$b;
?>
```

Ví dụ trên có nghĩa, $b không lấy giá trị của biến $a mà biến $b và biến $a cùng lấy một nội dung trong cùng một ô nhớ.



8. Các hàm kiểm tra giá trị

a. Kiểm tra tồn tại isset()

Hàm isset() dùng để kiểm tra xem biến có giá trị hay không.

Hàm này có thể dùng để kiểm tra sự tồn tại của một hay nhiều biến khác nhau. Nếu tất cả các biến đều có giá trị kết quả trả về bằng true, ngược lại trả về giá trị false.

Cú pháp:

```
isset(<tên biến 1>, <tên biến 2>,…)
```

Ví dụ 5.19: Kiểm tra xem người dùng có nhập vào tên đăng nhập hay chưa, nếu đã nhập thì in ra "Xin chào <tên đăng nhập>" ngược lại thì in ra "Vui lòng nhập tên đăng nhập".

```
<?php
```

```
if (isset($_POST[“tên_đăng_nhập”]))
{
      echo “Xin chào: ”.$_POST[“tên_đăng_nhập”];
} else {
      echo “Vui lòng nhập tên đăng nhập”;
}
?>
```

Chú ý: Nếu muốn in kết quả của hàm isset() thì ta có thể dùng hàm var_dump().

Ví dụ 5.20: Kiểm tra tên biến có phải là

```
<?php
$chuoi ="abc";
var_dump(isset($chuoi)); // kết quả bool(TRUE)
$a = NULL;
$b =123;
var_dump(isset($a));// kết quả bool(FALSE)
var_dump(isset($b));// kết quả bool(TRUE)
var_dump(isset($a,$b));// kết quả bool(FALSE)
?>
```

b. Kiểm tra giá trị rỗng empty()

Hàm empty() dùng để kiểm tra biến có giá trị rỗng hay không.

Nếu biến có giá trị NULL, chuỗi rỗng hoặc O thì kết quả trả về là TRUE, ngược lại trả về giá trị là FALSE.

Hàm này ngược lại với hàm isset(), và thường được dùng để kiểm tra xem người dùng có nhập giá trị vào một đối tượng nào đó trên form hay không.

Những kết quả dưới đây được xem là rỗng:
+ “” : chuỗi rỗng.
+ 0: 0 khi kiểu là integer.
+ NULL
+ FALSE
+ array(): mảng rỗng.
+ var $var: một biến được khai báo nhưng không có giá trị trong lớp.

Cú pháp:

```
empty(<tên_biến>)
```

Ví dụ 5.21: Kiểm tra dữ liệu có rỗng hay không

```
<?php
if(empty($_POST["ten_dang_nhap"]))
{
	echo "Vui long nhap vao tên dang nhap";
	exit;
} else {
echo "Xin cho: ".$_POST["ten_dang_nhap"];
}
?>
```

c. Kiểm tra giá trị số is_numeric()

Hàm is_numeric() kiểm tra biến có kiểu giá trị kiểu số hay không.

Nếu giá trị của biến không phải là kiểu số thì kết quả trả về là TRUE, ngược lại thì kết trả về là FALSE.

Cú pháp:

```
is_numeric(<tên_biến>)
```

Ví dụ 5.22: Kiểm tra dữ liệu nhập vào có phải là kiểu số không?

```
<?php
if(is_numeric($_POST["so_luong"]))
{
	$so_luong = $_POST["so_luong"];
	$don_gia =$_POST["don_gia"];
	$thanh_tien = $so_luong * $don_gia;
	exit;
} else {
echo "So luong phai la kieu so"];
}
?>
```

d. Kiểm tra kiểu giá trị của tên biến

- is_int() và is_long()

Hàm is_int() hoặc is_long()kiểm tra giá trị của biến có phải là số nguyên hay không.

Nếu giá trị của biến là số nguyên thì kết quả trả về là TRUE, ngược lại trả về giá trị là FALSE.

Cú pháp:

```
is_int(<tên_biến>); hoặc is_long(<tên_biến>);
```

Chú ý: Trong trường hợp biến vượt quá phạm vị của số nguyên thì chúng ta có hàm is_long() dùng để kiểm tra giá trị của biến có phải là kiểu long hay không. Hàm này có Cú pháp tương tự như hàm in_long().

Ví dụ 5.23: Kiểm tra kiểu dữ liệu của biến có phải là số nguyên không.

```
<?php
    $a ="15";
    $b =15;
    echo is_int($a); // kết quả trả về là 0
    echo is_int($b); // kết quả trả về là 1
?>
```

- is_string()

Hàm is_string() kiểm tra giá trị của biến có phải là kiểu chuỗi hay không.

Nếu giá trị là kiểu chuỗi thì kết quả trả về là true, ngược lại trả về giá trị là false.

Cú pháp:

```
is_string(<tên_biến>)
```

Ví dụ 5.24: Kiểm tra kiểu dữ liệu nhập vào là kiểu chuỗi không?

```
<?php
    $a ="Hello";
    $b =15.5;
    echo is_string($a); //kết quả là 1
    echo is_string($b); // kết quả là 0
?>
```

- is_double()

Hàm is_double() kiểm tra giá trị của biến có phải là kiểu số có dấu chấm động.

Nếu giá trị của biến là kiểu số dấu chấm động, số lẽ thì giá trị trả về là true, ngược lại thì trả về là false.

Cú pháp:

is_double(<tên biến>)

Ví dụ 5.25: Kiểm tra dữ liệu nhập vào có phải là số thực không?

```
<?php
    $x =4.1123;
```

```
    echo is_double($x); // kết quả trả về là 1
?>
```

e. Xác định kiểu dữ liệu biến

Hàm gettype() kiểm tra kiểu dữ liệu của biến, hoặc giá trị là kiểu nào: integer, string, double, array, object, class…

Kết quả trả về của hàm là kiểu của giá trí hay kiểu của biến.

Cú pháp:

gettype(<tên biến>)

Ví dụ 5.26: Kiểm tra kiểu dữ liệu nhập.

```
<?php
    $n = "day la chuoi";
    $a =123;
    $b =123.456;
    $mang =array(1,2,3);
    echo gettype($n)."</br>";
    echo gettype($a)."</br>";
    echo gettype($b)."</br>";
    echo gettype($mang);
?>
```

II. Câu lệnh điều khiển

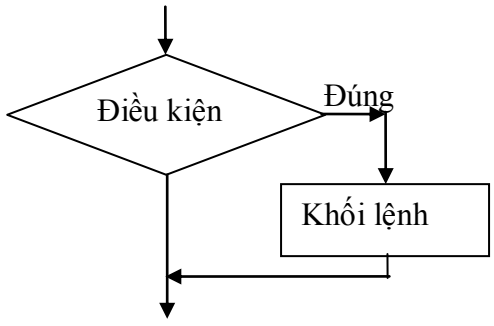
1. Câu lệnh rẽ nhánh If...Else

Dạng 1: Câu lệnh if dạng khuyết

Nếu điều kiện đúng thì thực hiện khối lệnh bên trong if sẽ được thực hiện. Ngược lại thì bỏ qua.

Cú pháp:

```
if ( <điều kiện>)
{
    // khối lệnh
}
```



Điều kiện có thể là biểu thức so sánh giá trị TRUE/FALSE hoặc là một giá trị số. Nếu giá trị số khác 0 thì giá trị trả về là TRUE, ngược lại trả về giá trị FALSE.

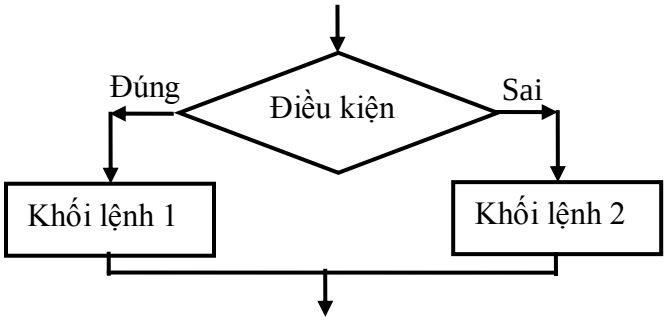
Khối lệnh: các lệnh sẽ được thực hiện khi điều kiện có giá trị là TRUE.

Dạng 2: Câu lệnh if dạng đầu đủ

Nếu điều kiện đúng thì thực hiện khối lệnh 1 bên trong if sẽ được thực hiện. Ngược lại thì thực hiện khối lệnh 2.

Cú pháp:

```
if ( <điều kiện>)
{
        // khối lệnh1
} else {
        // khối lệnh 2
}
```



Toán tử ?

Toán tử dấu ? dùng để thay thế câu lệnh if … else với một câu lệnh bên trong.

(<điều kiênh>)? <kết quả khi điều kiện đúng>: <kết quả khi điều kiện sai>;

Ví dụ 5.27: Tìm số lớn nhất trong hai số

```
<?php
if(strlen($_POST['a'])&& strlen($_POST['b']))
{
        $a = $_POST['a'];
        $b =$_POST['b'];
        $kq = ($a>$b)?$a:$b;
} else {
$kq = "Bạn chưa nhập thông tin vào";
}
?>
```

Dạng 3: Câu lệnh if lồng

Trong trường hợp có nhiều điều kiện thì chúng ta sử dụng câu lệnh if lồng nhau.

Cú pháp:

```
if (<điều kiện 1>)
{
        // khối lệnh 1
} elseif (<điều kiện 2>)
{
        // khối lệnh 2
```

```
}
…
else
{
      // khối lệnh khi không thỏa các điều kiện trên
}
```

2. Câu lệnh lựa chọn switch

Trong trường hợp có nhiều điều kiện xẩy ra. Trong trường hợp muốn so sánh giá trị của biến với biểu thức, và đối với mỗi giá trị này sẽ có những xử lý khác nhau thì ta dùng switch … case.

Cú pháp:

```
switch (<biểu thức>)
{
case <giá trị 1> :
        // khối lệnh khi biểu thức thõa mãn điều kiện 1
        break;
case <giá trị 2> :
        // khối lệnh khi biểu thức thõa mãn điều kiện 2
        break;
….
default:
      // khối lệnh khi không thõa tất cả các case trên.
}
```

3. Câu lệnh lặp

a. Cấu trúc for/foreach

- Cấu trúc for

For được dung khi chúng ta biết trước số lần lặp, biến đếm chạy trong khoảng giới hạn của vòng lặp, và giá trị lặp.

Vòng lặp sẽ kết thúc khi biến đếm vượt qua giới hạn của vòng lặp.

Cú pháp:

```
for ($biến chạy = <giá trị đầu>; <điều kiện của vòng lặp>; <giá trị lặp>)
{
      // Khối lệnh
}
```

- Câu trúc foreach

Cấu trúc foreach thường được dùng đẻ duyệt tập hợp(mảng). Cấu trúc này sẽ duyệt từ phần tử đầu tiên đến phần tử cuối cùng của tập hợp(mảng).

Cú pháp:

```
foreach ($tên_mảng as $giá_trị)
{
      // khối lệnh
}
```

Ví dụ 5.28: Vòng lặp foreach

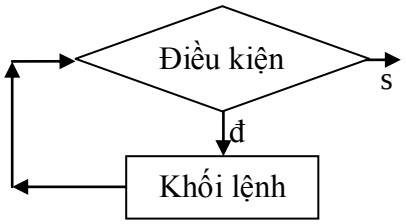
```
<?php
if(strlen($_POST['mang']))
{
      $mang =explode(",",$_POST['mang']);
      foreach($mang as $pt)  {
        echo $pt."  ";
       }
}
?>
```

b. Cấu trúc while

Khi chúng ta không xác định được số lần lặp(số lần lặp phụ thuộc vào điều kiện tại thời điểm thực thi) thì chúng ta sử dụng cấu trúc whlie.

Cú pháp:

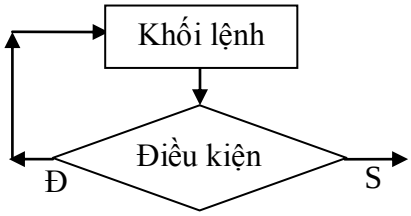
```
while (<điều kiện>)
{
      // Khối lệnh
}
```



c. Cấu trúc do … while

Cú pháp:

```
do
{
      // khối lệnh
}
while (<điều kiện>);
```



4. Sử dụng break và continue trong cấu trúc lặp

a. Lệnh break

break cho phép ta thoát khỏi cấu trúc điều khiển dựa trên kết quả của biểu thức logic

Ví dụ 5.28: Kiểm tra số nguyên tố

```
<?php
$so = 15;
$kq =true;
for ($i=2; $i<=$so; $i++)
{
        if($so%$i==0)
        {
                $kq= false;
                break;
        }
}
?>
```

b. Lệnh continue

Khi gặp continue, các lệnh bên dưới continue tạm thời không thực hiện tiếp, khi đó con trỏ sẽ nhảy về đầu vòng lặp để kiểm tra giá trị của biểu thức điều kiện còn đúng hay không. continue thường đi kèm với một biểu thức logic

Ví dụ 5.29: Tính tổng các phần tử lẻ từ 1 đến 10

```
<?php
$tong = 0;
for ($i=1; $i<=10; $i++)
{
        if($i%2==0)
        {
                continue;
        }
        $tong = $tong+$i;
}
echo $tong;
```

```
?>
```

5. Kiểu mảng

a. Khái niệm mảng

Mảng nói chung là một biến đặc biệt, nó bao gồm một dãy các ô nhớ có nhiều ô nhớ con cho phép biểu diễn thông tin dạng danh sách trong thực tế. Các phần tử mảng có thể có kiểu dữ liệu khác nhau.

b. Khai báo mảng và sử dụng mảng

Cách 1: Khai báo mảng chưa biết số phần tử mảng

Cú pháp: $ten_mang = array();

Ví dụ 5.30: Khai báo mảng chưa biết số phần tử mảng

```
<?php
$mang = array();
for($i=0; $i<10; $i++)
        $mang[$i] = $i;
?>
```

Cách 2: Khai báo biết trước số phần tử mảng

Cú pháp: $ten_mang = array(<số phần tử mảng>);

Ví dụ 5.31: Khai báo mảng biết trước số phần tử mảng

```
<?php
$mang = array(20);
for($i=0; $i<20; $i++)
        $mang[$i] = $i;
?>
```

Cách 3: Nếu khai báo mảng biết trước giá trị của mảng thì chúng ta vừa khai báo vừa gán giá trị.

Cú pháp: $ten_mang = array([khóa =>] giá_trị_1, …);

Trong đó:

+ Khóa: có thể là số nguyên dương hoặc chuỗi.

+ Khóa không được trung nhau.

+ giá_trị_1, …: có thể dùng tất cả các kiểu dữ liệu.

Ví dụ 5.32:

```
<?php
    $mang_1 = array(1,2,3,4,5); // không tạo giá trị cho khóa
    $mang_2 = array(1=>"Một", 2 =>"Hai", 3=>"Ba", 4 =>"Bốn", 5=>"Nam");
    $mang_3 = array("mot"=>1, "hai" =>2, "ba"=>3, "bon" =>4, "nam"=>5);
?>
```

Cách 4: Gán giá trị cho từng phần tử mảng

Cú pháp:

```
$ten_mang[ ] =<giá trị>;
Hoặc  $ten_mang[<giá trị khóa>] =<giá trị>;
```

c. Truy xuất phần tử mảng.

Cú pháp:

```
$ten_bien = $ten_mang[<giá trị khóa>];
```

d. Các thao tác trên mảng

- Đếm số phần tử mảng

Để đếm số phần tử của mảng một chiều ta sử dụng hàm count(< tên_biến_mảng>)

$so_pt_mang = count($ten_mang);

- Duyệt mảng

+ Duyệt mảng có khóa tự động

Ví dụ 5.33:

```
<?php
    $n = count($mang_1);
    for($i  = 0; $i< =$n; $i++)
        echo "\t".$mang_1[$i];
?>
```

+ Duyệt mảng có khóa do người dùng tạo:

Ví dụ 5.34:

```
<?php
foreach( $mang_2 as $gia_tri)
{
    echo "\t $gia_tri";
}
?>
```

+ Duyệt để lấy cả giá trị của khóa và giá trị của phần tử:

Ví dụ 5.35:

```
<?php
    foreach( $mang_2 as $khoa => $gia_tri)
    {
        echo "<br> [$khoa] => $gia_tri";
    }
?>
```

e. Một số hàm

- Tìm kiếm trên mảng: array_search()

Hàm này sẽ tìm kiếm một giá trị trên mảng, nếu tìm thấy sẽ trả về khóa của phần tử chứa giá trị đó, nếu không tìm thấy sẽ trả về giá trị NULL

Cú pháp:

$khoa = array_search($gia_tri_can_tim, $mang);

Ví dụ 5.36:

```
<?php
    $mang = array(0=>'xanh', 1=>'đỏ', 2=>'tím', 3=>'vàng');
    $khoa = array_search('đỏ', $mang);
    echo $khoa;
?>
```

- Ghép mảng: array_merge()

Ghép hai mảng hay nhiêu mảng với nhau, kết quả trả về là một mảng mới được tạo ra từ các mảng.

Cú pháp:

$mang_ghep = array_merge($mang_1, $mang_2,…);

Chú ý: Khi các mảng dùng để ghép có khóa trùng nhau thì mảng ghép sẽ chỉ lấy phần tử có khóa trùng của mảng cuối cùng.

Ví dụ 5.37:

```
<?php
    $mang1= array("màu"=>"đỏ", 2, 4);
    $mang2 = array("a", "b", "màu"=> "xanh", "hình"=>"tròn", 4);
    $mang_chung = array_merge($mang1, $mang2);
    print_r($mang_chung);
?>
```

- Đếm số lần xuất hiện: array_count_values()

Dùng để đếm số lần xuất hiện của các phần tử trong mảng. Kết quả trả về là một mảng trong đó khóa chính là giá trị trên mảng cần đếm và giá trị sẽ là số lần xuất hiện của nó trong mảng.

Cú pháp:

$mang_slxh = array_count_values($ten_mang);

Ví dụ 5.38:

```
<?php
    $mang = array(1, "hello",1, "world", "hello", 2 , "xin chào",1);
    $mang_slxh = array_count_values($mang);
    print_r($mang_slxh);
?>
```

- Tạo mảng duy nhất: array_unique()

Hàm sẽ bỏ đi những giá trị lặp đi lặp lại trong mảng. Kết quả trả về là một mảng mới mà trong đó mỗi phần tử trong mảng chỉ xuất hiện một lần.

Cú pháp:

$mang_day_nhat = array_unique($ten_mang);

Ví dụ 5.39:

```
<?php
    $mang = array(1, 3, 1, 2, 5, 1, 3, 4);
    $mang_dn = array_unique($mang);
    print_r($mang_dn);
?>
```

- Tìm các giá trị khác nhau của một mảng so với mảng khác: array_diff()

Hàm sẽ so sánh giữa hai mảng và lọc ra những phần tử chỉ có trong mảng thức nhất mà không có trong mảng thứ hai. Kết quả trả về là một mảng mới với những phần tử chỉ xuất hiện duy nhất trong một mảng.

Cú pháp:

array_diff($mang_1, $mang_2);

Ví dụ 5.40:

```
<?php
    $mang_1 = array("a"=>"xanh", "đỏ", "tím", "vàng");
    $mang_2 = array("b"=>"xanh", "vàng", "đỏ");
    $mang =array_diff($mang_1, $mang_2);
    print_r($mang);
?>
```

III. Xây dựng hàm trong PHP

1. Hàm do người dùng định nghĩa

a. Khai báo hàm

Để khai báo hàm chúng ta sử dụng từ khóa function, tiếp sau đó là tên hàm và danh sách các tham số (nếu có), các lệnh của hàm được đặt trong cặp ngoặc { }.

Cú pháp:

```
function ten_ham([danh sách tham số])
{
    // khối lệnh bên trong hàm
    [returm gia_trị;]
}
```

Trong đó:

Nếu hàm cần tham số để xử lý thì truyền vào tham số để xử lý, ngược lại có thể bỏ trống.

Nếu hàm trả lại giá trị thì giá trị trả về của hàm là: return gia_trị, nếu không có giá trị trả về thì không có lệnh trả về.

Ví dụ 5.41: Xuất ra câu chào "Hello world";

```
<?php
function cau_chao()
{
	echo "Hello world";
}
?>
```

Ví dụ 5.42: Hàm trả về giá trị

```
<?php
function tinhtong($a,$b)
{
	$s =$a+$b;
	return $s;
}
$so_a =5;
$so_b = 10;
$tong = tinhtong($so_a,$so_b)+tinhtong(2,3);
echo "Tổng = $tong"; // kết quả trả về 20
?>
```

b. Sử dụng hàm

Hàm sau khi được tạo có thể được gọi lại thông qua tên hàm, nếu hàm có các thông tin bên trong thì cung cấp đầy đủ các thông tin, nếu hàm có giá trị trả về thì phải có biến để nhận giá trị của hàm.

Cú pháp:

```
ten_ham([danh sách các giá trị]);
```

Trong đó:

- ten_ham được gọi đúng với hàm được định nghĩa.
- Danh sách giá trị: cung cấp các thông tin cho các tham số của hàm.

Ví dụ 5.43: Gọi hàm xuất ra câu chào

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

2. Hàm trong thư viện hàm

a. Kiểu dữ liệu string

Kiểu dữ liệu string dùng để lưu trữ chuỗi các ký tự.

Ví dụ 5.44:

```
$hoten = "Hồ Diên Lợi";
```

Một số hàm xử lý chuỗi.

- Hàm **ltrim(str [,char]):** Xoá khoảng trắng từ bên trái của chuỗi, nếu có tham số char thì sẽ bỏ luôn các các ký tự bên trái trong char.

Ví dụ 5.45: Cắt bỏ các ký tự dư thừa bên trái chuỗi

```
<?php
    $st="aaaa   Hoàng Nam";
    $st = ltrim($st,'a');
    echo $st; //"Hoàng Nam"
?>
```

- Hàm **rtrim(str [,char]):**Xoá khoảng trắng từ bên phải của chuỗi, nếu có tham số char thì sẽ bỏ luôn các các ký tự bên phải trong char.

Ví dụ 5.46: Cắt bỏ các ký tự dư thừa bên phải

```
<?php
    $st="Hoàng Nam    aaaa  ";
    $st = rtrim($st,'a');
    echo $st; //"Hoàng Nam"
?>
```

- Hàm **trim($st [,char]):** Loại bỏ kí tự thừa ở đầu và cuối của xâu, nếu có tham số char thì sẽ bỏ luôn các các ký tự bên phải trong char.

Ví dụ 5.47: Cắt bỏ các ký tự dư thừa bên trái và bên phải

```
<?php
    $st="  aaaa     Hoàng Nam    aaaa  ";
    $st = trim($st,'a');
    echo $st; //"Hoàng Nam"
?>
```

- Hàm **addslashes($st):** định dạng dữ liệu trong chuỗi để lưu vào CSDL.

Để lưu chuỗi có các dấu nháy ' hay cặp ", dấu \, dấu \\ thì chúng ta dùng thêm dấu \ vào phía trước chúng như sau: \', \", \\, \\\.

Ví dụ 5.48: Định dạng dữ liệu

```
<?php
        $st ="Who're you?";
        echo $st."</br>";// Who're you?
        echo addslashes($st); //Who\'re you?
?>
```

Vì các dấu nháy khi lưu vào trong CSDL sẽ xuất hiện dấu \ trước các ký tự, khi đọc lên trình duyệt chúng ta cần loại bỏ các dấu đó. Để loại bỏ ta sẽ sử dụng hàm **stripslashes()**

- Hàm **stripslashes():**

Ví dụ 5.49: Loại bỏ các ký tự.

```
<?php
$st ="Who\'re you?";
echo stripslashes($st); //Who're you?
?>
```

- Hàm **ucfirst($st)** : Viết hoa kí tự đầu tiên của một xâu.

Ví dụ 5.50: Viết hoa ký tự đầu tiên của một xâu.

```
<?php
        $st = 'xin chào!';
        echo ucfirst($st);          // Xin chào!
?>
```

- Hàm **ucwords($st):** Viết hoa kí tự đầu tiên của mỗi từ.

Ví dụ 5.51: Viết hoa kí tự đầu tiên của mỗi từ.

```
<?php
        $st = 'xin chào!';
        echo ucwords($st); // Xin Chào!
?>
```

- Hàm **strtolower($st)** : Biến kí tự bất kỳ thành chữ thường.

Ví dụ 5.52: Biến kí tự bất kỳ thành chữ thường

```
<?php
        $st = 'HỒ dIên LợI!';
```

```
        echo strtolower($st); // hồ diên lợi
?>
```

- Hàm **strtoupper($st):** biến kí tự bất kỳ thành chữ hoa.

Ví dụ 5.53: Biến kí tự bất kỳ thành chữ hoa.

```
<?php
        $st = 'HỒ dIên LợI!';
        echo strtoupper($st); // Hồ DIÊN LỢI
?>
```

- Hàm **strlen($st)**: Kết quả trả về độ dài của xâu

Ví dụ 5.54: Xuất ra trình duyệt độ dài lớn

```
<?php
        $hoten = 'Hồ Diên Lợi';
        echo "Độ dài xâu  " . strlen($hoten); //Kết quả trả về là 13
?>
```

- Hàm **strcmp($str1, $str2)**: hàm so sánh chuỗi không phân biệt chữ hoa và chữ thường, hàm này trả về kết quả là:

+ = 0: nếu hai chuỗi bằng nhau

+ < 0: nếu chuỗi $str1 nhỏ hơn chuỗi $str2

+ > 0: nếu chuỗi $str1 lớn hơn chuỗi $str2

Ví dụ 5.55: So sánh chuỗi

```
<?php
        echo strcmp('chao','Chao'); // kết quả 0
        echo strcmp('chao chu','chao anh'); // kết quả là 1
?>
```

- Hàm tìm một chuỗi trong 1 chuỗi **strstr($st1, $st2) và strchr($s1, $st2)**: Hàm trả về kết quả là một chuỗi con của chuỗi $st1 được lấy từ vị trí xuất hiện đầu tiên của chuỗi $st2 đến hết chuỗi $st1 nếu tìm thấy chuỗi $st2 trong chuỗi $st1, nếu không tìm thấy thì trả về giá trị FALSE.

Ví dụ 5.56: Tìm kiếm chuỗi trong chuỗi

```
<?php
        $st="ho die nloi@yahoo.com";
        echo strstr($st, "@")."</br>"; //@yahoo.com
        echo strchr($st, "#"); //
```

```
?>
```

- Hàm tìm vị trí chuỗi con **strpos($st1,$st2):** Hàm trả về vị trí chuỗi con đầu tiên của cuổi $st2 trong chuỗi $st1, nếu không tìm thấy thì kết quả trả về là FALSE

Ví dụ 5.57: Trả về vị trí đầu tiên của chuỗi trong chuỗi

```
<?php
        $st="hodienloi@yahoo.com";
        echo strpos($st,"yahoo"); // vị trí 10
        echo strpos($st,"gmail"); //
?>
```

- Hàm tìm kiếm và thay thế **str_replace($st1, $st2, $st)**: Hàm tìm kiếm chuỗi $st1 trong chuỗi $st, nếu tìm thấy thì thay thế chuỗi $st1 bằng chuỗi $st2 trong chuỗi $st.

Ví dụ 5.58: Tìm kiếm và thay thế yahoo bằng gmail, hotmail bằng gmail

```
<?php
$st="hodienloi@yahoo.com";
echo str_replace('yahoo', 'gmail', $st); //hodienloi@gmail.com
echo str_replace('hotmail', 'gmail', $st); //hodienloi@yahoo.com
?>
```

- Hàm **strrev($st):** Đảo ngược 1 xâu.

Ví dụ 5.59: Đảo xâu ký tự

```
<?php
echo strrev("Hello world!"); // kết quả trả về là "!dlrow olleH"
?>
```

- Hàm tách chuỗi **explode($ch,$st):** Hàm tách chuỗi $st thành nhiều phần tử con bằng cách chỉ định chuỗi tách $ch và gán từng chuỗi con vào các phần tử của mảng

Ví dụ 5.60: Tách chuỗi thành nhiều phần tử mảng

```
<?php
$st="Xin chào tất cả các bạn trong lớp";
$mang = explode(' ',$st);
for($i=0 ; $i<count($mang); $i++)
      echo "Từ thứ $i là: ".$mang[$i]."</br>";
?>
```

- Hàm kết hợp chuỗi **implode($ch, $mang)**: Hàm kết hợp các phần tử của mảng thành một chuỗi các phần tử khi ráp thành chuỗi sẽ cách nhau bằng một chỉ thị $ch.

Vi dụ 5.61: Nối nhiều phần mảng thành một chuỗi.

```
<?php
        $array = array('Xin', 'chào', 'tất', 'cả', 'các', 'bạn');
        $st = implode(" ", $array);
        echo $st; //Kết quả xuất ra là:  Xin chào tất cả các bạn
?>
```

- Hàm đổi số thành ký tự trong bảng mã ASCII chr():

Ví dụ 5.62: Chuyển mã thành ký tự trong bảng mã ASCII

```
<?php
for($i =15 ;$i<25; $i++)
echo chr($i).","; //kết quả là: □ , □ , □ , □ , □ , □ , □ , □ , □ , □ ,
?>
```

- Phép toán nối chuỗi (.)

Ví dụ 5.63: Nối hai chuỗi

```
<?php
        $ho_dem = 'Hồ Diên';
        $ten = 'Lợi';
        $ho_ten =$ho_dem .' '. $ten;
        echo $ho_ten; // Hồ Diên Lợi
?>
```

b. Kiểu dữ liệu số

- Hàm giá trị tuyệt đối: **abs**(x)

Ví dụ 5.64: Giá trị tuyệt đố của -5

```
<?php
echo abs(-5);
?>
```

- Hàm làm tròn: **round**(x[,i])

Ví dụ 5.65: Hàm làm tròn số

```
<?php
        $so = 1234.56789;
        echo round($so,2) // kết quả là 1234.57
?>
```

- Hàm làm lấy phân nguyên: **floor**(x)

Ví dụ 5.66: Hàm lấy phần nguyên

```
<?php
        $so = 123.678;
        echo floor($so); // kết quả của hàm là 123
?>
```

- Hàm e^x : **exp**(x)

Ví dụ 5.67: Hàm e^x

```
<?php
        echo exp(3);
?>
```

- Hàm lượng giác: sin(x), cos(x), tan(x) với x radian

Ví dụ 5.68: Tính sin, cos, tan của 30 độ

```
<?php
        define("PI",3.14);
        echo sin(PI*30/180); // kết quả  0.4997701026431
        echo cos(PI*30/180);// kết quả 0.86615809440546
        echo tan(PI*30/180); //Kết quả 0.57699640039287
?>
```

- Hàm căn bậc hai: **sqrt**(x)

Ví dụ 5.69:

```
<?php
        $so = 36;
        echo sqrt($so); // Kết quả là 6
?>
```

- Hàm ngẫu nhiên trong khoảng: **rand**(n1,n2)

Ví dụ 5.70:

```
<?php
```

```
        echo rand(10,20);
?>
```

- Hàm logaric: **log**(x)

Ví dụ 5.71:

```
<?php
        $so = 20;
        echo log($so); // kết quả 2.995732273554
?>
```

- Hàm mũ: pow(a, x)

Ví dụ 5.72:

```
<?php
        $so = 10;
        echo pow($so,2);
?>
```

- Hàm pi(): Hàm trả về giá trị là số pi, hàm không có tham số.

Ví dụ 5.73:

```
<?php
      $so = pi();
      echo $so*10; // 31.415926535898
?>
```

- Hàm **range(gt1,gt2):** Hàm lấy giá trị nguyên trong khoảng gt1… gt2

Ví dụ 5.74:

```
<?php
foreach (range(0, 12) as $number)
{
   echo $number;
}
echo "</br>";
foreach (range(0, 100, 10) as $number)
{
   echo $number;
}
?>
```

- Hàm **number_format():** Định dạng số

Ví dụ 5.75:

```
<?php
$number = 1234.56;
echo number_format($number)."</br>"; //1.235
echo number_format($number, 2, ',', ' ')."</br>"; //1 234,46
$number = 1234.5678;
echo number_format($number, 2, '.', '');//1234.57
?>
```

c. Kiểu dữ liệu ngày, giờ

- Hàm **checkdate()** kiểm tra ngày nhập vào có hợp lệ không?

Cú pháp: **checkdate** ($month , $day, $year)

Trong đó: Hàm trả về giá trị đúng hoặc sai.

$month, $day, $year: là kiểu dữ liệu số nguyên, là tháng ngày năm được nhập vào để kiểm tra.

Ví dụ 5.76:

```
<?php
	$day = 29;
	$month = 2;
	$year = 2010;
	$kq=checkdate($month, $day, $year);
	if($kq)
	{
		echo "Ngay thang hop le";
	}
	else
	{
		echo "Ngay thang khong hop le";
	}
?>
```

- Hàm **date()**

Cú pháp: **date ($format [,$timestamp])**

Trong đó:

Hàm trả về một chuỗi được qui định bởi chuỗi định dạng $format

Nếu $format =

D	Ngày có dạng hai chữ số 01- 31
D	Thứ trong tuần có 3 ký tự Mon - Sun
J	Ngày được có dạng 1-31
L/l	Thứ trong tuần được viết đầy đủ Monday - Sunday
N	Cho ra số thứ tự của ngày trong tuần (1-7)1: Monday – 7: Sunday
W	Cho ra số thứ tự ngày trong tuần 0: Monday – 6: Sunday
Z	Cho ra ngày thứ máy trong năm(0 - 365)
W	Cho ra số thứ tự tuần trong 1 năm
F	Cho ra tên tháng đầy đủ từ January tới December trong năm
M	Cho ra tháng từ 01 - 12
M	Cho ra tên tháng chỉ có 3 ký tự đầu từ Jan đến Dec
N	Cho ra tháng từ 1 - 12
Y	Cho ra năm có 4 chữ số 2009
Y	Cho ra năm có 2 chữ số 09
A	Dạng AM và PM
A	Dạng am và pm
G	Định dạng đồng hồ 12
H	Cho ra giờ 0 -23 h
H	Cho ra giờ 1- 12h
I	Cho ra phút (0-59)
S	Cho ra giây(0-59)

- Lấy giá trị của ngày hiện tại: getdate()

Hàm trả về mảng gồm 11 phần tử để lưu trữ các giá trị(seconds, minutes, hours, mday, wday, mon, year, yday, weekday, month, 0) của ngày tháng năm hiện tại.

Ví dụ:

```
<?php
    $today = getdate();
    print_r($today);
?>
```

- Lấy thời gian hiện tại: time()

- Chuyển chuỗi thành thời gian: strtotime()

- Kiểm tra và chuyển thời gian sang đơn vị giây: mktime()

- Định dạng thời gian thành só nguyên: idate()

IV. Biểu mẫu form

1. Đặc điểm form

Form là một thành phần của trang web. Chúng ta sẽ xây dựng form bằng cách thêm vào form nội dung và các đối tượng thể hiện (textField, Textarea, Button, RadioButton, CheckBox, List/Menu) sau đó định dạng chúng.

Các thuộc tính cơ bản của form:

- name: tên form
- action: hành động
- method: phương thức
- vị trí: _top, _parent, _self, _black

Chú ý: khi muốn lấy được giá trị trên form đưa về xử lý ở trang nào thì action sẽ chỉ ra trang đó.

Phương thức là cách thức lấy giá trị trên form. Chúng ta có hai phương thức sau: POST và GET

+ POST: chuyển giá trị trên form và để nhận được các giá trị này chúng ta dùng biến $_POST hoặc $_REQUEST.

+ GET: chuyển giá trị trên form và để nhận được các giá trị này chúng ta dùng biến $_GET hoặc $_REQUEST.

2. Biểu mẫu sử dụng phương thức $_POST

a. Đặc điểm

- Biến $_POST được dùng để lấy các giá trị trên form thông qua phương thức POST. Thông tin được gửi từ form với phương thức này không giới hạn dung lượng thông tin gửi đi.

- Thông tin được gửi bằng phương thức POST sẽ không hiện thị lên địa chỉ URL nên người dùng không thể thấy được.

b. Cách sử dụng

Cú pháp lấy giá trị của một đối tượng trên form sau khi form submit:

$_POST[‘tên điều kiển’];

Ví dụ: Tạo biểu mẫu nhập vào 2 số tính tính tổng và cho ra kết quả form

```
<?php
if(strlen($_POST["so_a"])&& strlen($_POST["so_b"]))
{
    $a = $_POST["so_a"];
```

```
        $b = $_POST["so_b"];
        $kq = $a + $b;
}
else
{
        $kq = "Bạn chưa nhập giá trị vào textfield";
}
?>
<form name="form1" method="post" action="vd.php">
  <table width="600" border="0" align="center" cellpadding="6" cellspacing="4">
    <tr>
      <td colspan="2" align="center" bgcolor="#FFCCFF"><span class="style1">TÍNH TỔNG CỦA HAI SỐ VÀ TRẢ GIÁ TRỊ LẠI FORM</span></td>
    </tr>
    <tr>
      <td width="142" align="right" valign="middle" bgcolor="#FFCCFF">Số A</td>
      <td width="422" bgcolor="#FFCCFF"><label>
        <input  type="text" name="so_a" value="<?php echo $_POST["so_a"]; ?>">
      </label></td>
    </tr>
    <tr>
      <td align="right" valign="middle" bgcolor="#FFCCFF">Số B </td>
      <td bgcolor="#FFCCFF"><label>
        <input  type="text" name="so_b" value="<?php echo $_POST["so_b"]; ?>">
      </label></td>
    </tr>
    <tr>
      <td align="right" valign="middle" bgcolor="#FFCCFF">Kết quả</td>
      <td bgcolor="#FFCCFF"><label>
        <input name="kq" type="text" id="kq" value="<?php echo $kq;?>" size="30">
      </label></td>
    </tr>
    <tr>
      <td align="right" valign="middle" bgcolor="#FFCCFF">&amp;nbsp;</td>
      <td bgcolor="#FFCCFF"><label>
        <input type="submit" name="button" id="button" value="Submit">
      </label></td>
    </tr>
```

```
  </table>
</form>
```

3. Biểu mẫu sử dụng phương thức $_GET

Lấy giá trị bằng phương thức

Chương 6: Hướng đối tượng trong PHP

I. Khái niệm

Một lớp được hiểu là một kiểu dữ liệu đặc biệt bao gồm các thuộc tính và các phương thức được định nghĩa từ trước. Đây là sự trừu tượng hóa của đối tượng. Một đối tượng sẽ được xác lập khi nó được thực tế hóa từ một lớp. Khác với kiểu dữ liệu thông thường, một lớp là một đơn vị (trừ tượng) bao gồm sự kết hợp giữa các phương thức và các thuộc tính.

II. Tạo lớp

Lớp là tập hợp các biến và các hàm – theo thuật ngữ của OOP, lớp là tập hợp các thuộc tính và các phương thức. Ngoài các lớp thư viện được xây dựng sẵn, chúng ta có thẻ tạo ra các lớp riêng còn gọi là lớp do người dùng tự định nghĩa.

Cấu trúc của một lớp bao gồm các từ khóa class, tên_lớp, dấu ngoặc {} để bao lại các câu lệnh bên trong lớp:

Cú pháp:

```
class <tên_lớp>
{
    // Khai báo các thuộc tính của lớp.
    // Gán và lấy giá trị của thuộc tính.
    // Tạo đối tượng của lớp.
}
```

Trong đó:

- Khai báo các thuộc tính của lớp

Thuộc tính là thành phần lưu trữ các tính chất, đặc điểm của đối tượng. Ứng với mỗi thuộc tính chúng ta khai báo một biến bắt đầu bằng từ kháo var để lưu trữ giá trị của chúng.

Cách khai báo:

```
var <tên_thuộc_tính_1>;
……..
```

Chú ý: Chúng ta cũng có thể khởi gán giá trị cho tên biến.

Ví dụ:

```
Class vidu_lop
{
  var $text;
  ……..
}
```

- Lấy các giá trị cho các thuộc tính

+ Gán giá trị cho thuộc tính

Một thuộc tính ban đầu cần phải có giá trị, do đó chúng ta cần phải gán giá trị cho thuộc tính.

Cú pháp:

```
function set_name ($giá_trị)
{
	$this -><$tên_thuộc_tính> =$giá_trị;
}
```

Ví dụ: Gán giá trị cho thuộc tính name.

```
function set_vidu($text)
{
	$this ->name =$text;
{
```

+ Lấy giá trị của thuộc tính.

Trước khi muốn dùng thuộc tính thì chúng ta cần phải lấy giá trị của thuộc tính.

Cú pháp:

```
function get_name()
{
	return $this -> <tên_thuộc_tính>;
}
```

Ví dụ: Lấy giá trị của thuộc tính

```
function get_vidu()
{
	return $this ->name;
{
```

Chú ý: Trong lớp chúng ta có thể truy cập thông qua con trỏ $this. Con trỏ lớp dùng để chỉ lớp hiện tại đang làm việc.

III. Sử dụng lớp

Sau khi xây dựng xong lớp chúng ta có thế sử dụng lớp: trước tiên cần khởi tạo đối tượng, sau đó gán giá trị cho các thuộc tính và gọi sử dụng các phương thức của lớp.

- Tạo đối tượng:

Để khai báo một đối tượng chúng ta dùng từ khóa new tiếp đó là tên lớp.

Cú pháp:

$name_opp = new <tên_class>();

Ví dụ: Tạo đối tượng

```
$opp = new vidu_lop();
```

- Gán giá trị cho thuộc tính lớp.

$name_opp ->set_name(<giá trị>);

Ví dụ: Gán giá trị cho

```
$opp->set_vidu("Chào bạn");
```

- Sử dụng các phương thức lớp

Để sử dụng phương thức chúng ta chỉ cần gọi tên phương thức với các giá trị truyền vào cho tham số nếu có.

Cú pháp:

$name_opp ->get_name(tham số nếu có);

Ví dụ: Sử dụng phương thức

```
$opp->get_vidu();
```

Ví dụ: Tính tổng hai số

```
<?php
    class phep_tinh
    {
        var $so1;
        var $so2;
        // so thu nhat
            // Lay gia tri cua so thu nhat
            function lay_gt_1()
            {
                return $this->a;
            }
            // Gan gia tri cho so thu nhat
            function gan_gt_1($so1)
            {
                $this->a = $so1;
            }
        // so thu hai
            // Lay gia tri cua so thu hai
```

```
                function lay_gt_2()
                {
                        return $this->b;
                }
                // Gan gia tri cho so thu hai
                function gan_gt_2($so2)
                {
                        $this->b = $so1;
                }
        // Phuong thuc
                function tong()
                {
                        return $this->a + $this->b;
                }
    }
    $tinh = new phep_tinh();
    $tinh->gan_gt_1(6);
    $tinh->gan_gt_1(8);
    echo "a+b =".$tinh->tong();

?>
```

IV. Kế thừa

1. Khái niệm kế thừa

2. Chồng hàm

Chương 7: Tạo web động

I. Sử dụng tập tin dùng chung

Để sử dụng các đoạn mã bên ngoài, chúng ta có thể sử dụng khai báo tiền xử lý include và require. Cho phép chúng ta xây dựng các hàm các hằng số, và bất kỳ đoạn mã nào sau đó có thể chèn vào các đoạn kịch bản.

Require khác include là, nó có thể làm thay đổi nội dung của trang hiện tại khi biên dịch, các trang này dùng để khai báo các biến, các hằng số hay các đoạn mã đơn giản không có vòng lặp. Khi đó include cho phép thực hiện các câu lệnh phức tạp – có câu lệnh tạo chu trình. Nó chỉ sử dụng các hàm như những hàm ngoài của chương trình.

Ví dụ:

<table>
<tr><td colspan="3">Header</td></tr>
<tr><td colspan="3">Menu_main</td></tr>
<tr><td>Memu_left</td><td>Content</td><td>Hot information</td></tr>
<tr><td colspan="3">Fooder</td></tr>
</table>

1. REQUIRE

a. Cách sử dụng

Đối với phương thức require(), tất cả nội dung bên trong file được chèn vào sẽ được biên dịch.

Khi chúng ta muốn sử dụng đoạn chương trình đã được viết sẵn ở vị trí nào trong trang thì chúng ta chỉ cần dùng require() để chèn file chứa đoạn chương trình này ở vị trí đó.

Cú pháp:

require("tên và đường dẫn của tập tin");

Ví dụ:

Đoạn chương trình dưới đây được viết ở trang chao.php

```
<?php
```

```
    echo "Hello my class";
?>
```

Còn đoạn chương trình dưới được viết ở trong home.php

```
<?php
    echo "Đây là đoạn chương trình hướng dẫn học PHP và MYSQL<br>";
    require("chao.php");
    echo "<br>Đã đến với chương trình này";
?>
```

b. Các tập tin đương dùng trong require()

PHP không quy định cách đặt tên tập tin được chèn vào bằng require(). Vì vậy, chúng ta có thể đặt tên tập tin tùy ý. Khi chúng ta dùng require() để chèn tập tin này vào, nội dung của tập tin sẽ trở thành một phần trong trang web.

Thông thường, các câu lệnh PHP đopch viết trong các tập tin .html sẽ không thực hiện được. Chúng sẽ thực hiện được khi chúng được viết dưới trong các tập tin .php. Vì vậy, khi viết các câu lệnh PHP trong các tập tin được dùng để chèn này, ta nên chọn kiểu tập tin thích hợp như . inc hay . php để các lệnh này có thể biên dịch được khi thực thi.

c. Thẻ PHP và require()

Các lệnh PHP cần phải được bao bởi cặp thẻ PHP **<?php** và **?>**. Nếu chúng ta không có thẻ PHP **<?php** và **?>** khi viết lệnh PHP thì các lệnh PHP này sẽ trở thành dạng văn bản hoặc HTML và không thể thực thi được.

d. Dùng require() cho các template

Nếu ứng dụng web của chúng ta có cùng một mẫu thiết kế nhưng chỉ khác nhau về nội dung bên trong thì chúng ta sẽ tạo ra một mẫu template và chỉ cần khai báo các biến trình bày dữ liệu bên trong template.

Khi template này được thực thi, tất cả các biến này sẽ có giá trị và trình bày như một định dạng template được sử dụng nhiều lần.

Ví dụ: Thông thường trong ứng dụng web phần header và fooder thường được hiện thi ở hầu hết các trang, vì vậy ta sẽ tạo ra trang header.inc và fooder.inc để chứa định dạng và nội dung phần cuối trang. Sau đó, ở trang nào của ứng dụng có sử dụng header và fooder thì chúng ta sẽ chèn hai trang này vào.

Điểm đặc biệt và quan trọng nhất của kết hợp này là sau khi chúng ta đã thực hiện nhiều trang có chèn các trang header.inc và fooder.inc, chúng ta có thể dễ dàng thay đổi trang

header.inc và fooder.inc này. Và chỉ cần lưu lại các thay đổi thì tất cả các trang đã chèn các trang này sẽ tự động thay đổi.

2. INCLUDE

a. Cách sử dụng

include() cũng có cách sử dụng tương tự require(). Tuy nhiên, chúng có một điểm khác nhau cần phải lưu ý đó là khi nội dung bị lỗi thì dùng require() sẽ xuất hiện thông báo lỗi, trong khi đó dùng include() sẽ chỉ xuất hiện cảnh báo.

Trong những tập tin có dùng require() thì ta không nên sử dụng các câu trúc điều khiển vì sẽ không hiệu quả.

Cú pháp:

include("tên tập tin và đường dẫn đến tập tin");

Ví dụ: Trang tính toán có nội dung như sau:

```
<?php
if($a ==1)
{
	require("tinh_tong.php");
}
else
{
	require("tinh_tong.php");
}
?>
```

Trong ví dụ này khi thực thi trang tinh_toan.php, nội dung bên trong cả hai tập tin là tinh_tong.php và tinh_hieu.php đều được biên dịch, trong khi đó chỉ có một trong hai trường hợp đúng và chỉ cần biên dịch một tập tin khi thỏa mãn điều kiện đúng là đủ. Như vậy, dùng require() trong trường hợp này sẽ không còn thích hợp, thay vào đó chúng ta dùng include().

```
<?php
if($a ==1)
{
	include("tinh_tong.php");
}
else
{
	include("tinh_tong.php");
}
```

?>

b. require_once() và include_once()

Hàm require_once() và include_once() là hai dạng biến đổi của hàm require() và include().

Mục đích của hai hàm này trở nên hữu ích khi chúng ta bắt đầu sử dụng chúng để chèn các thư viện và các hàm vào, sử dụng các cấu trúc này giúp chúng ta tránh được việc chèn cùng một hàm hay thư viện lần thứ hai bởi khi hàm khai báo lại một hàm đã được xây dựng sẽ được phát sinh lỗi.

Việc chèn vào hai lần cùng một tập tin thường xảy ra khi xây dựng các ứng dụng lớn, khi nhiều tập tin thư viện khác nhau được chèn vào trong cùng một require_once() hay include_once() thì trong lần đầu tiên cách thức hoạt động của nó cũng giống như require() hay include(). Tuy nhiên, require_once() và include_once() nếu được gọi để chèn tập tin đã được chèn thì file này sẽ không chèn được chèn vào nữa. Hàm này là một công cụ thông minh cho việc tạo ra các thư viện dùng lại.

c. Đường dẫn của file được chèn

Sử dụng các hàm đã được giới thiệu ở trên để truy cập các thư viện có thể làm tăng tính mềm dẻo của ứng dụng. Tuy nhiên, vẫn còn có một vấn đề xảy ra.

d. Ví dụ

II. Mở tập tin và thư mục

1. Tập tin

a. Chế độ mở tập tin

Để mở một tập tin, chúng ta cần xác định chế độ mở. Có 3 tùy chọn cho chế độ mở file:

- Mở file ở chế độ read only, write only hay cả read và write.
- Mở file đã tồn tại: chúng ta có thể ghi đè hay ghi thêm vào nội dung đang có của file. Trong trường hợp ghi thêm vào nội dung đã có của file, có hai cách ghi là ghi vào đầu tập tin và ghi vào cuối của tập tin.
- Khi muôn ghi file hệ thống thì chúng ta cần chỉ định chế độ ghi file là nhị phân hoặc text.

Phân loại các chế độ mở file:

Chế độ	**Mô tả**
r	Chỉ đọc file, bắt đầu đọc đầu file
r+	Đọc và ghi file: Bắt đầu từ đầu file
w	Chỉ ghi file. Mở và xóa toàn bộ nội dung của file đã có hoặc tạo ra một file mới nếu file đó không tồn tại, sau đó ghi nội dung vào file.

w+	Đọc và ghi. Mở và xóa toàn bộ nội dung của file đã có hoặc tạo ra một file mới nếu file đó không tồn tại, sau đó ghi nội dung vào file
a	Chỉ ghi file. Mở và ghi nội dung vào cuối file hoặc tạo ra một file mới nếu file không tồn tại.
a+	Ghi và đọc dữ liệu. Mở và ghi nội dung vào cuối file hoặc tạo ra một file mới nếu file không tồn tại.
x	Tạo và mở file để ghi. Tạo ra một file mới và ghi nội dung vào file. Nếu file đã tồn tại, trả về giá trị FALSE và thông báo lỗi.
x+	Tạo và mở file để đọc và ghi. Tạo ra một file mới và ghi nội dung vào file. Nếu file đã tồn tại, trả về giá trị FALSE và thông báo lỗi.

b. Mở tập tin

Để mở tập tin chúng ta sử dụng hàm fopen().

Cú pháp:

fopen(<tên_tập_tin>,<chế độ mở>)

Ví dụ: Mở file vidu.txt trong thư mục vidu_web, với chế độ mở chỉ đọc.

```
<?php
$f = fopen("vidu.txt", "r");
?>
```

c. Đọc tập tin

- Kiểm tra kết thúc tập tin

Để kiểm tra trạng thái kết thúc tập tin hay chưa chúng ta sử dụng hàm feof()

Cú pháp:

feof($f)

Trong đó $f là biến khai báo nhận giá trị trả về của hàm fopen(). Hàm trả về kết quả là TRUE nếu con trỏ ở cuối file.

Ví dụ:

```
<?php
$f = fopen("vidu.txt", "r");
if(feof($f))
{
    echo "Đã kết thúc tập tin";
}
?>
```

- Duyệt và đọc từng dòng nội dung trong tập tin

Khi mở file chúng ta có thể đọc nội dung đã được mở theo từng dòng bằng hàm **fgets().** Hàm sẽ trả về là chuỗi có độ dài xác định, mặc định độ dài là 1024.

Cú pháp:

fgets($f [, int độ dài])

Ví dụ:

```
<?php
$f = fopen("vidu.txt", "r");
while(!feof($f))
{
	$noidung =fgets($f);
	echo $noidung. "</br>";
}
fclose($f);
?>
```

- Duyệt và đọc từ ký tự trong tập tin

Để đọc nội dung tập tin theo từng ký tự 1 ta sử dụng hàm **fgetc()**

Cú pháp:

fgetc($f)

Ví dụ:

```
<?php
$f = fopen("vidu.txt", "r");
while(!feof($f))
{
	$noidung =fgetc($f);
	echo $noidung;
}
fclose($f);
?>
```

- Đọc toàn bộ nội dung tập tin

Để đọc toàn bộ nội dung tập tin ta sử dụng hàm **readfile().**

Cú pháp:

readfile(<đường dẫn và tên tập tin>);

```
<?php
$f = "vidu.txt";
echo readfile($f);
?>
```

d. Định dạng tập tin.

Trước khi ghi chuỗi vào file, chúng ta cần phải định dạng lại chuỗi đó theo nhu cầu xuất dữ liệu trở lại khi đọc file.

Cách thức định dạng là do chúng ta tự thiết lập. Tuy nhiên, có một số định dạng được quy định sẵn như sau:

+ \t : nhảy tab

+ \n : xuống dòng

e. Ghi nội dung tập tin

Để ghi nội dung vào tập tin ta sử dụng hàm fwrite()

Cú pháp:

fwrite(<tập tin>, <nội dung [, <độ dài>]>);

Ví dụ:

f. Đóng tập tin.

Để đóng tập tin đã mở ta sử dụng hàm fclose()

Cú pháp:

fclose($f)

Trong đó $f là tên biến nhận giá trị trả về của hàm fopen()

g. Kiểm tra sự tồn tại của tập tin

Để mở file ta thường gặp hai trạng thái, file đó đã có hoặc file đó không tồn tại.

Để kiểm tra sự tồn tại của file ta sử dụng hàm file_exists() hoặc is_file()

Cú pháp:

file_exists(<tập tin>)

Trong đó: Hàm trả về kết quả TRUE hoặc FALSE, nếu tồn tại trả về giá trị là TRUE, ngược lại trả về giá trị FALSE

Ví dụ:

```
<?php
```

```
$f = "vidu.txt";
if (!file_exists($f))
{
	echo "không ton tai file";
	exit;
} else {
	echo readfile($f);
}
?>
```

h. Kiểm tra kích thước file

Để kiểm tra kích thước file ta sử dụng hàm filesize()

Cú pháp:

filesize(<đường dẫn và tên của file>);

Vi dụ: Kiểm tra kích thước tập tin

```
<?php
$f = "vidu.txt";
echo "Size :".filesize($f);
?>
```

k. Xóa tập tin

Để xóa tập tin ta sử dụng hàm unlink()

Cú pháp:

unlink(<tập tin>)

Ví dụ: Xóa tập tin vidu.txt

```
<?php
$f = "vidu.txt";
if (!unlink($f))
{
	echo "Khong xoa dc tap tin $f";
} else {
	echo "da xoa tap tin $f";
}
?>
```

2. Thư mục

a. Tạo thư mục

Để tạo thư mục ta sử dụng hàm mkdir()

Cú pháp:

mkdir(<tên thư mục>);

Ví dụ: Tạo thư mục

```
<?php
mkdir("DIENLOI");
?>
```

b. Kiểm tra thư mục

Để kiểm tra thư mục ta sử dụng hàm is_dir(), trả về TRUE nếu tồn tại thư mục, ngược lại là FASLE.

Cú pháp:

is_dir(<thư mục>)

Ví dụ: Kiểm tra sự tồn tại của thư mục

```
<?php
if(!is_dir("DIENLOI"))
{
	echo "Khong ton tai thu muc";
}
else
{
	echo "Ton tai thu muc";
}
?>
```

c. Mở thư mục

Để mở thư mục ta sử dụng hàm opendir(). Kết quả trả về sẽ là nguồn(chứa các thư mục và tập tin) của thư mục nếu thư mục mở thành công, ngược lại trả về giá trị False.

Cú pháp:

opendir(<tên thư mục>)

Ví dụ: Mở thư mục

```
<?php
```

```
$dir = opendir("Vidu_php");
?>
```

d. Đóng thư mục

Khi đã dùng xong thư mục, cần phải đóng thư mục bằng hàm closedir()

Cú pháp:

closedir(<tên thư mục>)

Ví dụ: Đóng thư mục

```
<?php
closedir("Vidu_php");
?>
```

e. Duyệt thư mục

Chúng ta duyệt thư mục bằng cách sử dụng vào lặp kết hợp với hàm readdir()

Cú pháp:

readdir(<tên thư mục>)

Ví dụ:

```
<?php
$dir =opendir("Vidu_php");
while(($file = readdir($dir))==true)
{
 echo $file."</br>";
}
closedir($dir)
?>
```

III. Upload tập tin lên server.

1. Giới thiệu

Trong hầu hết các ứng dụng web, thông thường người dùng có thể upload file lên server. Đối với ứng dụng web xây dựng bằng ngôn ngữ PHP chúng ta cũng có thể xây dựng cho người dùng upload file lên server.

2. Các bước upload file

Bưới 1: Tạo form upload file:

```
<form action ="upload_file.php" method ="POST" enctype ="multipart/form-data">
<label for ="file"> Tên file </label>
```

```
<input type ="file" name ="file_upload" id ="file" />
</br>
<input type ="submit" name ="submit" value ="Upload file" />
</form>
```

Chú ý:

+ Đối với form để upload file thì trên thẻ form chúng ta bổ sung thêm thuộc tính enctype ="multipart/form-data".

+ Method được sử dụng theo phương thức POST

+ Nếu muốn quyết định kích cỡ tối đa của tập tin upload thì trong thẻ input FileField upload chúng ta bổ sung thêm thuộc tính value = "kích thước tối đa" – đơn vị tính là byte, lúc này thuộc tính name của file field có giá trị là "MAX_FILE_SIZE".

Bước 2: Viết code thực hiện việc upload file

<?php

```
if($_FILES["file_upload"]["error"]>0)
{
  echo "Lỗi của file ".$_FILES["file_upload"]["error"]."</br>";
}
else
{
      echo "Upload:".$_FILES["file_upload"]["name"]."</br>";
      echo "Type:".$_FILES["file_upload"]["type"]."</br>";
      echo "Upload:".($_FILES["file_upload"]["size"]/1024)."Kb</br>";
      echo "Temp file: ".$_FILES["tmp_file"]."</br>";
      if (file_exists("upload/".$_FILES["file_upload"]["name"]))
      {
            echo $_FILES["file_upload"]["name"]."Đa ton tai";
      }
      else
      {
      move_uploaded_file($_FILES["file_upload"]["tmp_name"],"Upload/".$_FILES["file_
upload"]["name"]);
            echo "Lưu trữ "."Upload/".$_FILES["file_upload"]["name"];
      }
}
?>
```

Bằng cách sử dụng biến $_FILES[tên_đk_field] [tên thuộc tính] ta có thể upload file từ máy client sang server.

Trong đó, tham số thứ nhất là tên điều khiển filefield trên form, tham số thứ hai là một trong những thuộc tính như: name(tên file), type(loại file), size(kích thước byte – tính theo byte), tmp_name(tên tạm của file), error(lỗi)…

Với đoạn code trên, trước tiên ta kiểm tra file upload có bị lỗi hay không, nếu có lỗi thì thông báo lỗi, ngược lại in ra các thuộc tính của file upload. Sau đó kiểm tra xem trong thư mục upload đã có file này hay chưa, nếu đã có thì thông báo, ngược lại thì sử dụng hàm move_uploaded_file(file_tạm, nơi lưu trữ upload) để di chuyển thư mục về thư mục lưu trữ file trên server.

IV. PHP Cookies

1. Khái niệm

Cookie được sử dụng để xác định thông tin của người dùng. Cookie là một file nhỏ được server lưu trữ xuống từng máy tính của người dùng. Mỗi khi máy tính này yêu cầu một trang tới trình duyệt, nó cũng sẽ gửi theo cookie. Với PHP ta có thể tạo ra và sử dụng giá trị của biến cookie.

2. Khai báo cookie

Dùng hàm setcookie() để khai báo một biến cookie.

Chú ý: Hàm này đặt trên thẻ <html>

Cú pháp:

setcookie(name, value, expire [,path, domain]);

Trong đó:

+ Name: tên biến cookie

+ value : giá trị

+ expire: thời gian giới hạn dành cho cookie – đơn vị tính là giây. Nếu thời gian không được thiết lập trong hàm setcookie(), biến cookie sẽ còn hiệu lực cho đến khi người dùng xóa tập tin cookie.

+ path: đường dẫn

+ domain: tên miền của website.

Ví dụ:

```
setcookie("username","ho die n loi", time()+3600);
```

Chú ý: Giá trị của biến sẽ được tự động mã hóa khi gửi cookie đi, tự động giải mã khi nhận cookie về .

3. Sử dụng cookie

Khi người dùng muốn sử dụng biến cookie đã đăng ký chúng ta dùng biến $_COOKIE để đọc giá trị biến cookie.

Cú pháp:

$giá_trị = $_COOKIE[“tên biến cookie”];

Ví dụ:

```
$username = $_COOKIE["username"]; // ho dien loi
```

4. Hủy cookie

Khi hủy biến cookie chúng ta cần kiểm tra lại thời gian giới hạn dành cho biến cookie này(được thiết lập khi chúng ta tạo ra biến cookie). Sau đó chúng ta sử dụng hàm setcookie() để hủy bằng cách đặt gia trị cho biến cookie bằng "" và thời gian =- thời gian giới hạn.

Cú pháp:

setcookie(name, "" , time() – thời gian giới hạn);

Ví dụ:

```
setcookie("username","", time()-3600);
```

V. PHP Sessions

1. Khái niệm

Khi làm việc với 1 ứng dụng, chúng ta sẽ mở ứng dụng lên, làm việc và đóng ứng dụng lại. Máy tính sẽ biết được ta là ai, biết chúng ta mở ứng dụng và đóng ứng dụng lại. Nhưng trên trình duyệt web không biết ta là ai và chúng ta làm gì bởi vì giao thức HTTP không duy trì trạng thái.

Session giải quyết vấn đề này bằng cách cho phép chúng ta lưu trữ thông tin người dùng trên trình duyệt web. Tuy nhiên thông tin sesion chỉ tạm thời và sẽ bị xóa sau khi người dùng rời khỏi ứng dụng web. Nếu cần, chúng ta có thể lưu trữ trong CSDL.

2. Cách thức hoạt động

Session làm việc bằng cách tạo ra một địa chỉ duy nhất (UID) cho mỗi người sử dụng. UID có giá trị là một chuỗi số ngẫu nhiên. UID có thể được lưu trữ trong cookie hoặc được truyền lên URL.

Ngoài UID, bạn có thể khởi tạo và sử dụng một số biến session do người dùng khai báo, tất cả các session này có giá trị cho mỗi người sử dụng khi họ truy cập đến ứng dụng web.

3. Khởi động Session

Trước khi lưu trữ thông tin người dùng vào session, chúng ta phải khởi động session.

Chú ý: Hàm khởi động session phải đặt phía trên thẻ <html>

Cú pháp:

```
session_start();
```

hàm này sẽ đăng ký session với trình duyệt, cho phép chúng ta bắt đầu lưu trữ thông tin người dùng và đặt UID.

4. Đặt ký Session

Chúng ta dùng biến $_SESSION nhận và lưu trữ giá trị của biến session.

Cú pháp:

```
$_SESSION["ten_bien_session"] ="giá tri";
```

Ví dụ:

5. Sử dụng Session

Khi muốn sử dụng các biến session hoặc giá trị lưu trữ trong biến session đã đăng ký chúng ta cũng dùng biến $_SESSION để đọc giá trị biến session.

Cú pháp:

$gia_tri=$_SESSION["ten_bien_session"];

Ví dụ: Tạo form đăng nhập thành tài khoản thành viên

6. Hủy biến Session

a. Hủy toàn bộ các biến session

Khi chúng ta không dùng đến các biến session nữa chúng ta có thể hủy toàn bộ các biến session mà chúng ta đã đăng ký bằng cách sử dụng hàm session_destroy()

Cú pháp:

```
session_destroy();
```

b. Hủy một biến session

Khi chúng ta không cần dùng biến session nào thì chúng ta có thể dùng hàm unset() để hủy bỏ biến session đó.

Cú pháp:

unset($_SESSION[“tên biến session”]);

VI. Gửi E-mail trong PHP

PHP cho phép người dùng tạo ra form gửi mail đến địa chỉ mail xác định. Để gửi mail sử dụng hàm mail(to,subject,message,headers,parameters với các tham số sau:

To: Địa chỉ người nhận

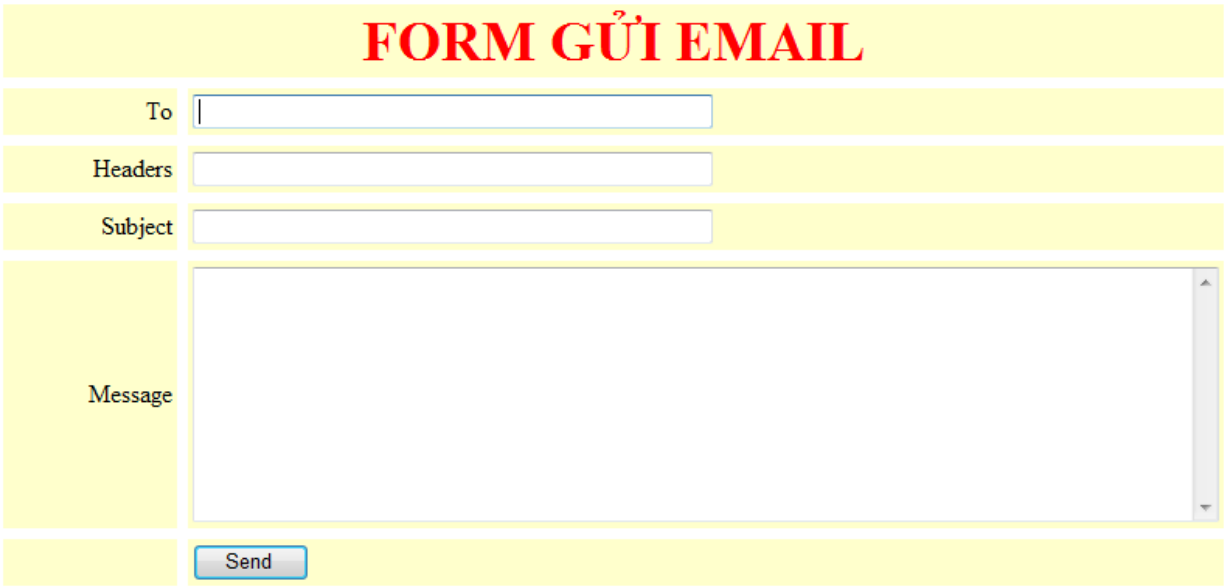
Subject: Tiêu đề của nội dung thư

Message: Nội dung thư

Headers Tiêu đề bổ sung: Người gửi email

Parameters:

Ví dụ: Form Mail



Ví dụ: Lấy thông tin từ Form

```
<?php
$to = $_POST['to'];
$headers =$_POST['headers'];
$subject=$_POST['subject'];
$message =$_POST['message'];
$headers ="Form: $headers";
mail($to,$subject,$message,$headers);
?>
```

Chương 8: CƠ SỞ DỮ LIỆU MYSQL

I. Tổng quan

1. Giới thiệu CSDL

a. Khái niệm

CDSL là một tập hợp dữ liệu được lưu trữ một cách có tổ chức nhằm giúp việc xem, tìm kiếm và lấy thông tin được nhanh chóng và chính xác, giúp giảm công sức và thời gian quả lý thông tin cần thiết.

b. Chức năng.

- Lưu trữ

Dữ liệu được lưu trữ trên đĩa và người dùng có thể chuyển đổi dữ liệu từ CSDL này sang CSDL khác.

Tùy theo quy mô của ứng dụng mà chúng ta có thể chọn CSDL lớn hay nhỏ. Nếu quy mô nhỏ thì chúng ta chọn Access, MySQL, … nếu quy mô lớn thì chúng ta có thể chọn SQL Server, Oracle, DB2,…

- Truy cập

Tùy thuộc vào mục đích và yêu cầu của người sử dụng mà có những mức độ truy cập khác nhau: cục bộ, chia sẽ, truy cập dữ liệu giữa các CSDL khác nhau.

- Tổ chức

Cách tổ chức CSDL, tùy thuộc vào mô hình CSDL, cách phân tích và thiết kế CSDL và các đặc điểm riêng của từng ứng dụng.

- Xử lý

Xử lý dữ liệu là việc sử dụng các truy vấn cùng các phép toán để truy xuất các kết quả theo yêu cầu của người dùng.

c. Các loại CSDL

- CSDL phân cấp(Hierachical Database)

CSDL phân cấp có cấu trúc cây, dữ liệu được tổ chức dưới dạng tập tin trên đĩa.

Ưu điểm: Tốc độ truy cập nhanh vì chúng có quan hệ trức tiếp với nhau.

Khuyết điểm: không dùng cho các ứng dụng có quan hệ phức tạp.

- CSDL hướng đối tượng(Object Oriented Database)

CSDL hướng đối tượng là CSDL mà trong đó một bảng dữ liệu có thể được khai báo như một field của bảng dữ liệu khác.

- CSDL quan hệ(Relation Database)

CSDL quan hệ là CSDL mà các bảng dữ liệu có quan hệ với các bảng khác thông qua các mối quan hệ.

d. Các đối tượng chính của CSDL

Tuy có rất nhiều CSDL khác nhau nhưng trong môn học này chúng ta chỉ tìm hiểu về CSDL quan hệ.

- Bảng dữ liệu(table)

Bảng dữ liệu là thành phần trung tâm của CSDL, được dùng để lưu trữ thông tin của CSDL.

Cách thiết kế các bảng dữ liệu có vai trò rất quan trọng vì nó quyết định tính hiệu quả trong việc lưu trữ thông tin.

Trong một CSDL có nhiều bảng, mỗi bảng dùng để lưu trữ một nhóm thông tin khác nhau.

Cấu trúc của bảng dữ liệu gồm hai thành phần dòng và cột

Cột: là một khối dữ liệu trong bảng, có cùng loại dữ liệu.

Mỗi cột có các thông tin chính sau:

+ Tên cột: dùng để phân biệt với các cột khác trong bảng, do vậy nó có tính duy nhất, tên cột không dùng các ký tự đặc biệt

+ Kiểu dữ liệu của cột: xác định loại kiểu dữ liệu nào được phép lưu trữ trong cột

Dòng: là tập hợp các thông tin của tất cả các cột trong bảng

- Quan hệ(relation)

Là thành phần được dùng để tạo mối liên kết giữa các bảng dữ liệu với nhau nhằm đảm bảo tính nhất quán, đúng đắn của dữ liệu trong CSDL.

e. Hệ quản trị CSDL

Hầu hết các CSDL đều dựa vào một hệ quản trị CSDL để quản lý các dữ liệu được lưu trữ bên trong các CSDL đó và làm cho CSDL dễ dàng đến được với người dùng khi cần truy cập các thông tin khác nhau.

Một hệ quản trị CSDL tối thiểu phải có khả năng lưu trữ dữ liệu và cho phép dữ liệu có thể trao đổi với các CSDL khác.

Tuy nhiên, hầu hết các hệ quản trị CSDL có nhiều tính năng hơn:

- Quản lý dữ liệu.

- Duy trì bảo vệ

- Duy trì dữ liệu

- Quản lý các giao dịch.

…

f. SQL (Structure Query Language)

SQL là một ngôn ngữ cho phép thực hiện các thao tác rút trích, tính toán, cập nhật trên các dữ liệu được lưu trữ trong CSDL.

2. CSDL MySQL

a. Giới thiệu

CSDL MySQL là tập hợp các đối tượng: bảng, bảng ảo … cho phép người dùng lưu trữ và xuất các thông tin đã được tổ chức và lưu trữ bên trong đó.

b. Đặc điểm

- MySQL được sử dụng cho các ứng dụng web có quy mô vừa và nhỏ.

- Người dùng có thể sử dụng giao diện đồ họa hay dùng dòng lệnh để thực hiện các thao tác trên CSDL.

c. Các tập tin vật lý lưu trữ CSDL

Mỗi bảng trong CSDL được tạo ra sẽ được lưu trữ dưới 3 tập tin vật lý:

- .frm: lưu cấu trúc của bảng

- .MYD: lưu nội dung của bảng

- .MYI: lưu chỉ mục của bảng.

Các tập tin dữ liệu này sẽ được tự động lưu trữ trong thư mục:

Wamp\mysql\data\tên_CSDL.

d. Quy tắc đặt tên cho CSDL, bảng, chỉ mục, cột và định danh

- Chiều dài của tên

Loại	Chiều dài tối đa (byte)	Chiều dài tối đa (ký tự không dấu)
Database	64	64
Table	64	64
Index	64	64
Column	64	64
Alias	255	255

- Quy tắc đặt tên

+ Tên không kết thúc bằng khoảng trắng.

+ Tên CSDL không có các ký tự '/', '\', '.', hoặc các ký tự không cho phép khi đặt tên cho thư mục (\, /, :, *, '', <, >)

+ Tên bảng không có các ký tự '/', '\', '.', hoặc các ký tự không cho phép khi đặt tên cho tập tin (\, /, :, *, '', <, >, |)

+ Chiều dài của tên tối đa là 64 ký tự không dấu. Nếu chúng ta sử dụng các ký tự đa byte thì chiều dài sẽ dựa trên tổng số byte của tất cả các ký tự được dùng.

e. Tạo CSDL

Có hai cách để tạo một CSDL là dùng giao diện đồ họa hoặc dùng dòng lệnh.

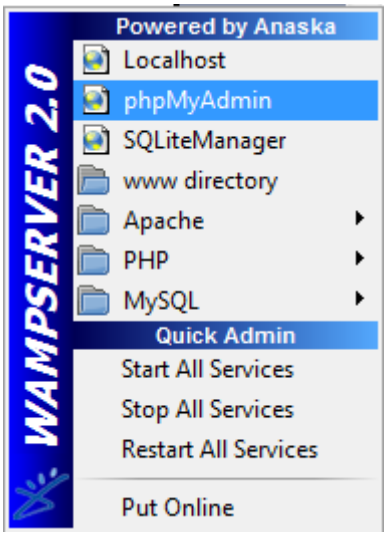
Các thuộc tính của CSDL

+ Tên CSDL: phải duy nhất trong hệ quản trị CSDL.

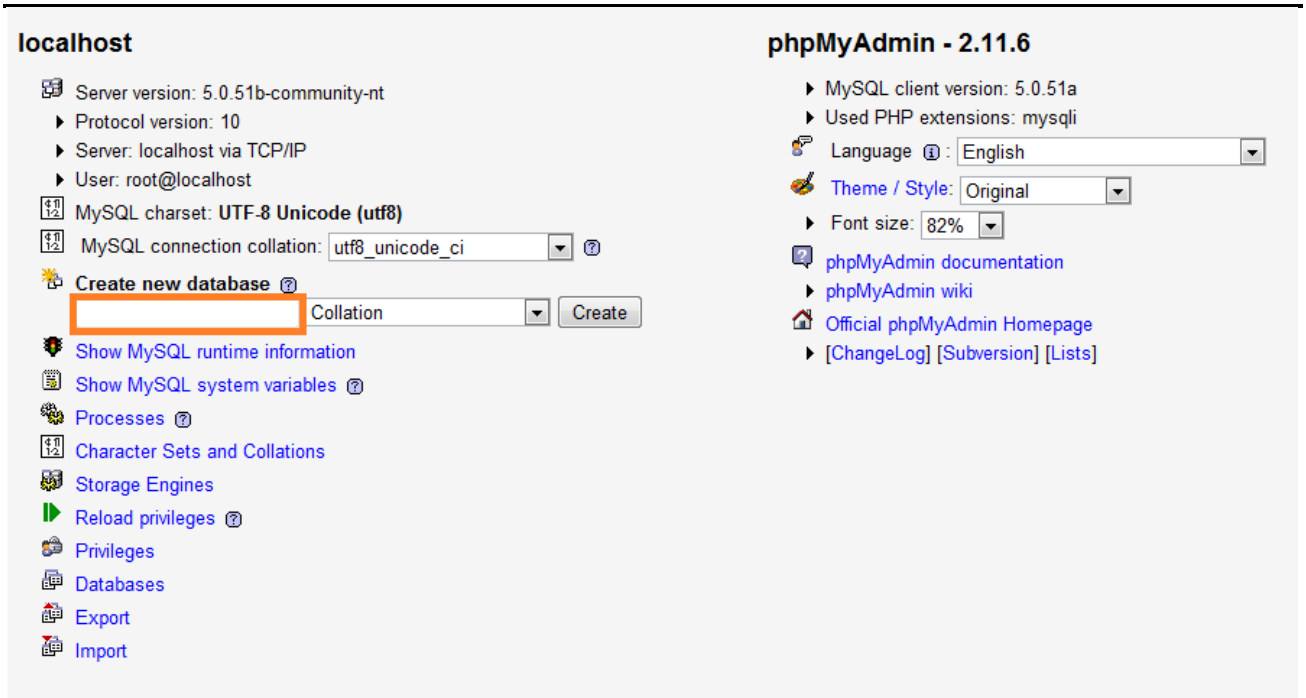
+ Vị trí lưu trữ: khi tạo mới một CSDL hệ thống sẽ tự động tạo ra một thư mục có tên là tên CSDL và được lưu trữ tại thư mục wamp\mysql\data\

Ví dụ: CSDL khoacntt

Bước 1: khởi động phpMyadmin



Bước 2: Trong màn hình giao diện đồ họa, nhập tên CSDL vào create new database và chọn các thông tin khác (nếu cần)



Bước 3: Nhấn Create để hoàn thành việc tạo CSDL.

- Sử dụng câu lệnh SQL

Cú pháp lệnh SQL tạo CSDL:

```
CREATE DATABSE name_database
[[DEFAULT] CHARACTER SET <character set name>]
[[DEFAULT] COLLATE <collation name>]
```

Trong đó:

+ CHARACTER SET : xác định bộ ký tự mặc định cho CSDL mới

+ COLLATE: Xác định bộ collation

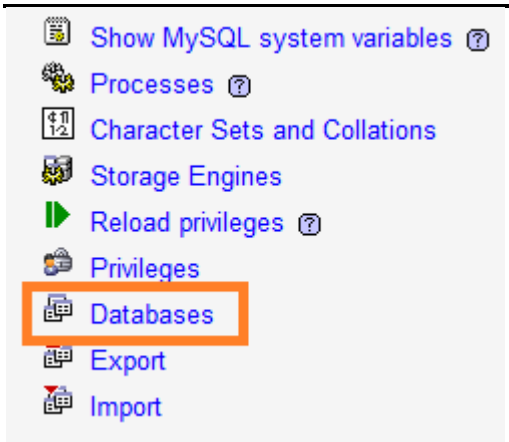
+ Character set name: Tên của một bộ mã bao gồm các ký tự, ký tự số, và biểu tượng để lưu trữ thông tin trong CSDL.

+ Collation name: tên một bộ mã tùy theo từng khu vực dựa trên bộ mã chuẩn character set name.

Cách thực hiện:

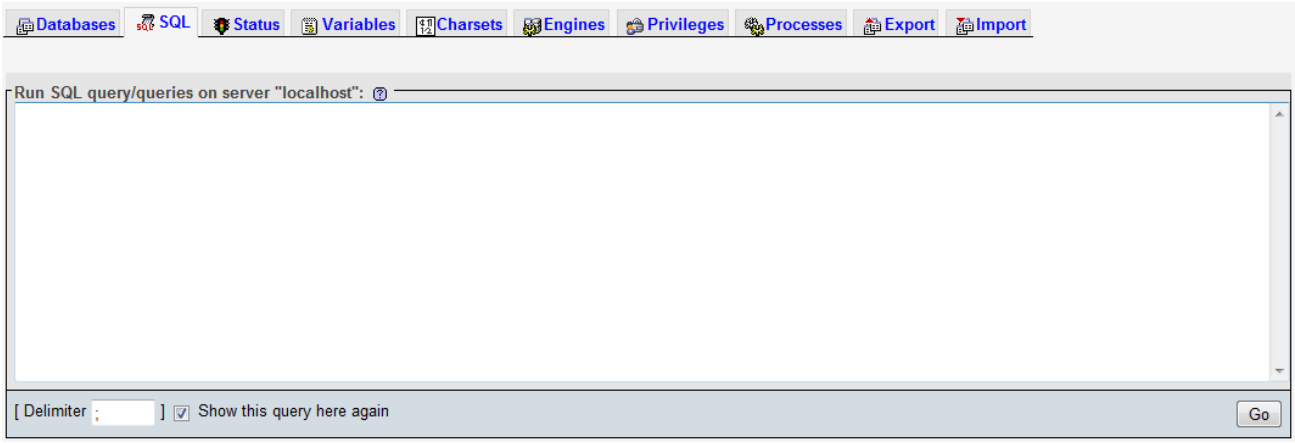
Bước 1: Khởi động phpMyAdmin

Bước 2: Chọn database



Bước 3: Chọn SQL

Bước 4: Viết lệnh SQL



Bước 5: Nhấn Go để kết thúc việc tạo CSDL

Ví dụ: Tạo CSDL khoacntt

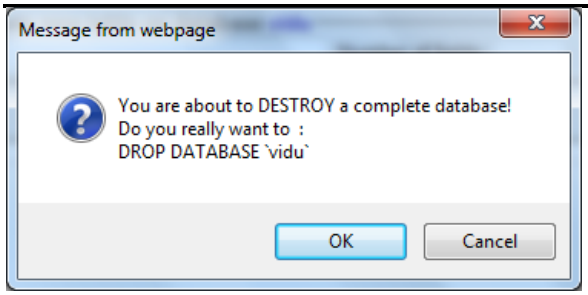
CREATE DATABASE `khoacntt1` ;

f. Xóa CSDL

- Xóa bằng giao diện đồ họa

Bước 1: Chọn CSDL cần xóa - nhấn Drop

Bước 2: Xác định lại việc xóa CSDL, sau đó chọn OK để xóa



- Sử dụng lệnh SQL

Cú pháp lệnh:

```
DROP DATABASE name_database
```

Ví dụ: Xóa CSDL khoacntt

```
DROP DATABASE `khoacntt`
```

II. Bảng(Table)

1. Khái niệm

Bảng trong MySQL dùng để lưu trữ thông tin của những đối tượng, thực thể trong thế giới thực muốn được lưu trữ vào trong máy tính.

Các thông tin trong bảng sẽ được tổ chức theo dạng dòng và cột.

2. Thuộc tính

a. Tên bảng

Tên bảng do người dùng đặt tên, tên bảng phải duy nhất trong một CSDL.

b. Các thuộc tính của cột trong bảng

- Tên cột: do người dùng đặt và tên cột là duy nhất trong bảng.

- Kiểu dữ liệu: Xác định kiểu dữ liệu lưu trữ trong cột, có các kiểu dữ liệu sau:

Kiểu số nguyên:

Kiểu dữ liệu	**Kích thước**	**Miền giá trị**
Tinyint	1 byte	-127 – 128 hay 0..255
Smallint	2 byte	-32768 – 32767 hay 0..65535
Mediumint	3 byte	-8388608 – 838860 hay 0..16777215
Int	4 byte	-2^{31} – 2^{31}-1 hay 0..2^{32}-1
Bigint	8 byte	-2^{63} – 2^{63}-1 hay 0..2^{64}-1

Kiểu dữ liệu true/false

Kiểu dữ liệu	**Kích thước**	**Miền giá trị**
Bool/ bloolean	1 byte	Có giá trị là true hoặc false

Kiểu số thập phân: decimal và numeric

Decimal và numeric là những kiểu dữ liệu được dùng để lưu trữ các giá trị số cụ thể. Giá trị của decimal và numeric được lưu trữ với một định dạng nhị phân.

Cú pháp:

Decimal(M[,N])

Trong đó:

+ M: tổng ký số.

+ N: số ký số thập phân, nếu N=0 được hiểu là không ký số thập phân và tương đương Decimal(M)

Các kiểu dữ liệu số thực

Kiểu dữ liệu	**Kích thước**	**Miền giá trị**
Float	4 bytes	3.402823466E-38 – 1.175493451E+38
Double	8 bytes	1.7976931348623157E-308 – 2.2250738585072014E+308

Kiểu dữ liệu ngày giờ

Kiểu dữ liệu	**Kích thước**	**Diễn giải**
Date		
Datetime		
Time		
Year[(2|4)]		
Timestamp[(kích cỡ định dạng)]		

Kiểu dữ liệu chuỗi

Kiểu dữ liệu	**Kích thước**	**Diễn giải**
Char	1÷ 255	Chuỗi cố định
Varchar	1÷ 255	Chuỗi động
TinyBlob	1÷ 255	Kiểu đối tượng nhị phân cỡ 255 ký tự
Tinytext	1÷ 255	Kiểu đối tượng chuỗi kích cỡ 255 ký tự
Blob	1÷ 65535	Kiểu blob
Text	1÷ 65535	Kiểu dạng văn bản cỡ 65535 ký tự
MediumBlob	1÷ 16777215(byte)	
Mediumtext	1÷ 16777215(ký tự)	

longBlob	$1 \div 2^{32}$-1(byte)	
longtext	$1 \div 2^{32}$-1(ký tự)	

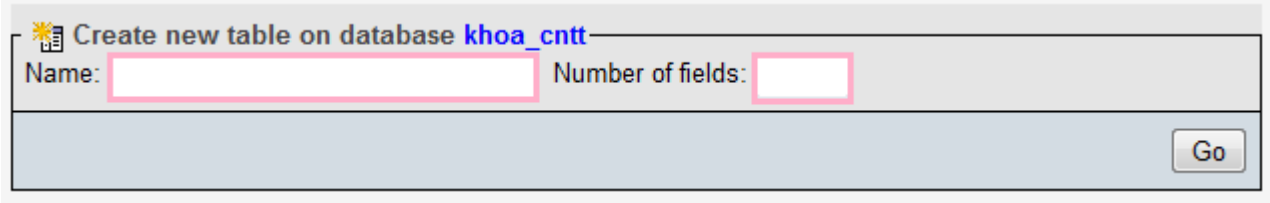
3. Thao tác với bảng

a. Tạo bảng

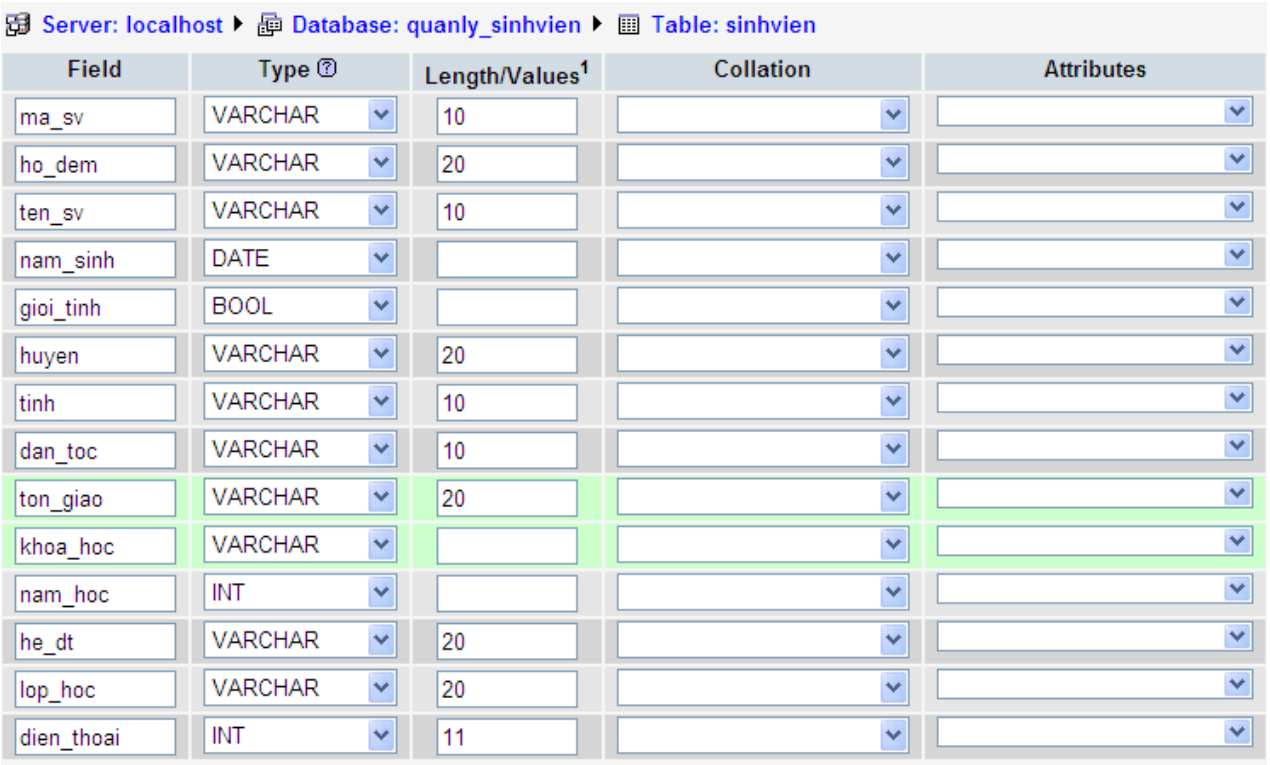
- Cách tạo bảng bằng giao diện đồ họa:

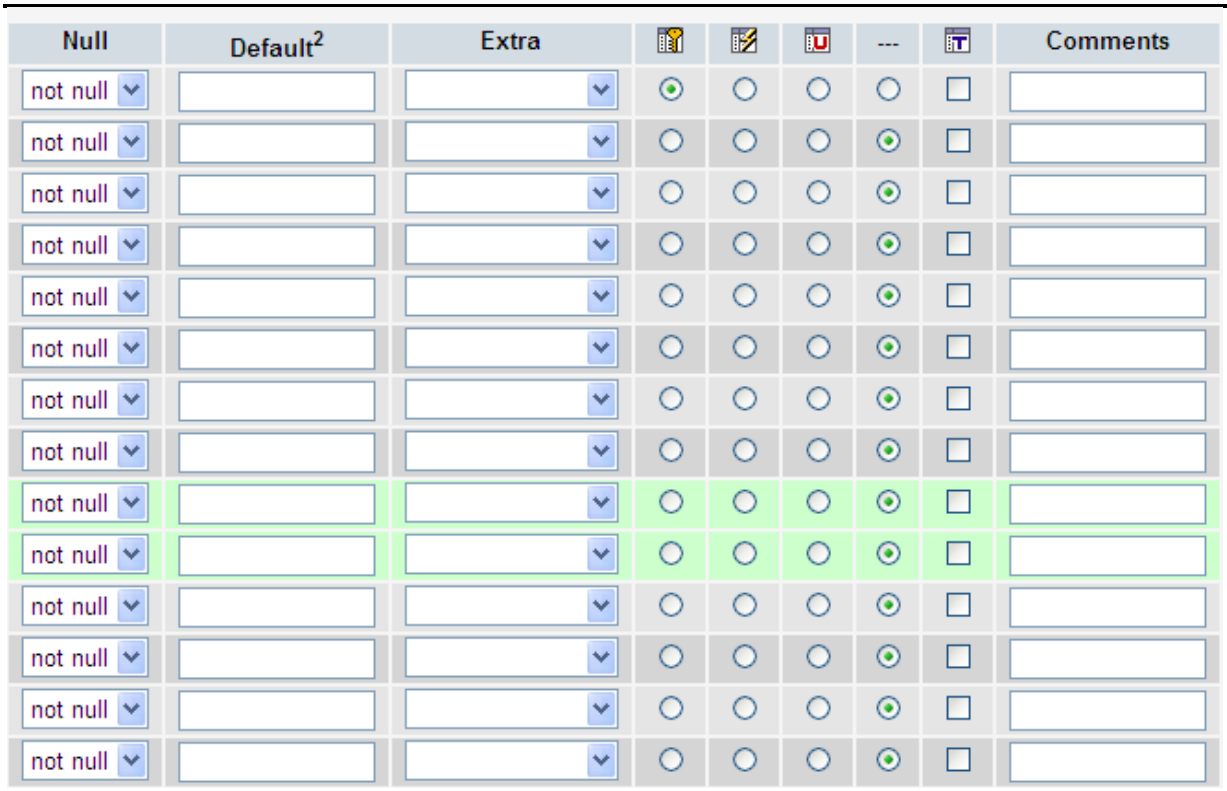
Bước 1: Chọn CSDL để tạo bảng

Bước 2: Nhập tên bảng vào trong Name và nhập số cột vào Number of fields sau đó nhấn Go



Bước 3: Nhập tên trường, chọn kiểu trường, độ dài của giá trị, ….





Bước 4: Nhấn vào Save để hoàn thành quá trình tạo bảng

- Tạo bảng bằng lệnh SQL

Cú pháp:

```
CREATE TABLE name_table
{
        Name_column_1 type_data[(size)] [<parameter>],
          Name_column_2 type_data[(size)] [<parameter>],
          …
}
```

Trong đó: Một số parameter sau:

+ NO NULL: không cho pháp dữ liệu trong cột để trống

+ DEFAULT giá_trị: cho phép cột có giá trị mặc định

+ PRIMARY KEY: thiết lập khóa chính của bảng

+ Auto_Increment: Xác định cột tăng giá trị tự động

Ví dụ: Tạo bảng sinhvien

```
CREATE TABLE `sinhvien` (
  `ma_sv` varchar(10) NOT NULL,
  `ho_dem` varchar(20) NOT NULL,
```

```
  `ten_sv` varchar(10) NOT NULL,
  `nam_sinh` date NOT NULL,
  `gioi_tinh` tinyint(1) NOT NULL,
  `huyen` varchar(20) NOT NULL,
  `tinh` varchar(10) NOT NULL,
  `dan_toc` varchar(10) NOT NULL,
  `ton_giao` varchar(20) NOT NULL,
  `khoa_hoc` varchar(10) NOT NULL,
  `nam_hoc` int(11) NOT NULL,
  `he_dt` varchar(20) NOT NULL,
  `lop_hoc` varchar(20) NOT NULL,
  `dien_thoai` int(11) NOT NULL,
  PRIMARY KEY  (`ma_sv`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

b. Thay đổi cấu trúc bảng

Trong trường hợp ta muốn thêm một hay nhiều cột vào bảng đã có, ta sẽ dùng câu lệnh ALTER TABLE.

Cú pháp:

```
ALTER TABLE <name_table>
ADD <name_column>  <type_data>[(size) ][…]
```

Chú ý: tên cột mới thêm vào phải khác với tên cột đã có trong bảng.

Ví dụ: Thêm vào bảng khoa một trường có tên là dien_thoai sau trường giao_vu

```
ALTER TABLE `khoa` ADD `dien_thoai` VARCHAR( 11 ) NOT NULL AFTER `giao_vu` ;
```

c. Sửa đổi kiểu dữ liệu của cột

Khi chúng ta muốn sửa đổi kiểu dữ liệu cho cột đã có chúng ta có thể dùng lệnh ALTER TABLE

Cú pháp:

```
ALTER TABLE <name_table>
CHANGE <name_column_old>  <name_column_new> type_data_new [(size)]
```

Ví dụ: Thay đổi kích thước của trường dien_thoai

```
ALTER TABLE `khoa` CHANGE `dien_thoai` `dien_thoai` VARCHAR( 12 )
CHARACTER SET latin1 COLLATE latin1_swedish_ci NOT NULL
```

e. Hủy cột trong bảng

Khi không cần sử dụng cột trong bảng chúng ta sử dụng Cú pháp ALTER TABLE để hủy bỏ cột. Tuy nhiên, khi cột bị xóa thì dữ liệu bên trong cột cũng sẽ bị xóa mà không thể phục hồi được. Do đó cần phải cẩn thận khi hủy bỏ cột.

Cú pháp:

```
ALTER TABLE <name_table>
DROP  COLUMN <name_column>,…
```

f. Xóa bảng

Khi chúng ta không cần bảng hoặc cấu trúc bảng không phù hợp chúng ta có thể xóa bảng.

Cú pháp:

```
DROP TABLE <name_table>
```

III. Bảng ảo

1. Khái niệm

View bắt đầu được sử dụng từ phiên bản MySQL server 5.0.

View là một cách khác để hiện thị CSDL. Một view là một bảng ảo được lưu trữ trong CSDL nhưng không thật sự chứa dữ liệu. Thay vào đó, view là một đối tượng mà bên trong nó chỉ có một câu lệnh SELECT dùng để chọn lọc một cột, các dòng trong các bảng CSDL để người dùng có thể xem và truy cập.

2. Tạo bảng ảo

Tạo bảng ảo bằng lệnh CREATE VIEW

Cú pháp:

```
CREATE VIEW name_view [(name_column_view)]
AS
      command SELECT
[WITH CHECK OPTION]
```

Trong đó:

- Tên các cột trong view: là các tên được đặt tương ứng với các cột hay biểu thức tính toán trong câu lệnh SELECT.

- Câu lệnh SELECT: câu lệnh truy vấn, chọn lựa dữ liệu từ trong một hay trong nhiều bảng có liên kết với nhau.

- WITH CHECK OPTION: dùng để ngăn cản các thao tác cập nhật dữ liệu tác động vào bảng ảo có làm ảnh hưởng đến dữ liệu trong các bảng ảo được tạo ra bằng cách lấy nguồn dữ liệu từ bảng ảo này.

Ví dụ: Tạo một bảng ảo dùng để hiển thị thông tin tất cả các cột của bảng sinhvien, với mã sinh viên là "09Tin0012"

```
CREATE VIEW sinhvien_view
AS
SELECT * FROM sinhvien WHERE ma_sv= "09Tin0012"
```

Sau khi chúng ta muốn xem kết quả của view đã tạo, chúng ta sử dụng lệnh SELECT.

Cú pháp:

```
SELECT * FROM name_view
```

Ví dụ: Xem kết quả của view có tên sinhvien_view

```
SELECT * FROM sinhvien_view
```

Khi sử dụng bảng ảo để thực hiện tính toán thống kê dữ liệu, chúng ta có thể đặt tên cho các cột theo hai cách:

Cách 1: Đặt tên cột ngay sau câu lệnh CREATE VIEW

Ví dụ: Tạo một bảng ảo có tên là sinhvien_view_sum dùng để đếm số sinh viên. Dữ liệu hiển thị gồm các cột: ma_sv, tong_sv

```
CREATE VIEW sinhvien_view_sum (ma_sv, sum_sv)
AS
SELECT ma_sv, count(ma_sv) FROM sinhvien GROUP BY  ma_sv
```

Cách 2: Đặt tên cho cột tính toán ngay trong câu lệnh SELECT

Ví dụ: Tạo một bảng ảo có tên là sinhvien_view_sum dùng để đếm số sinh viên. Dữ liệu hiển thị gồm các cột: ma_sv, tong_sv

```
CREATE VIEW sinhvien_view_sum (ma_sv, sum_sv)
AS
SELECT ma_sv, count(ma_sv) as sum_sv FROM sinhvien GROUP BY  ma_sv
```

3. Cập nhật nội dung bảng ảo

Để sử đổi nội dung có trong bảng ảo ta sử dụng Cú pháp ALTER VIEW, Cú pháp này cũng tương tự như CREATE VIEW

Cú pháp:

```
ALTER VIEW name_view [(name_column_view)]
AS
    Câu lệnh SELECT mới
[WITH CHECK OPTION]
```

Ví dụ: Từ bảng ảo có tên là sinhvien_view, cho biết sinh viên có điểm tổng kết cao nhất của năm học

```
ALTER VIEW sinhvien_view
AS
SELECT ma_sv, count(ma_sv) as sum_sv, max(diemtk) as diem_max FROM sinhvien
GROUP BY as ma_sv
```

4. Xóa bảng ảo

Khi chúng ta không cần sử dụng bảng ảo nữa, chúng ta có thể xóa bỏ bảng ảo. Khi xóa bảng ảo, dữ liệu trong bảng nguồn không ảnh hưởng.

Cú pháp:

```
DROP VIEW name_view
```

Ví dụ:

IV. Toán tử

1. Khái niệm

MySQL cung cấp cho chúng ta các toán tử như: toán tử số học, toán tử so sánh, toán tử logic.

Các toán tử này được kết hợp vào bên trong các mệnh đề WHERE, HAVING, IF, CASE,…

2. Toán tử số học

Dùng để tính toán các phép tính: cộng, trừ, nhân, chia, chia lấy phần dư. Giá trị được đem tính toán phải là kiểu số.

Khi có nhiều phép tính thì chúng ta nên đưa từng biểu thức tính toán vào trong dấu ngoặc đơn () để việc tính toán đó được tường minh.

Toán tử toán học:

+	Cộng
-	Trừ
*	Nhân
/	Chia
%	Chia lấy phần dư

Chú ý: Toán tử toán học cho phép sử dụng các kiểu dữ liệu số, tuy nhiên đối với phép tính chia lấy phần dư thì chúng ta chỉ có thể sử dụng kiểu số nguyên.

3. Toán tử so sánh

Dùng để thực hiện các phép tính so sánh như: bằng, lớn hơn, nhỏ hơn, khác,… cho các biểu thức cần so sánh. Kết quả trả về của phép so sánh là đúng hoặc sai.

Toán tử so sánh được sử dụng cho nhiều kiểu dữ liệu khác nhau như kiểu số, kiểu chuỗi…

Toán tử so sánh:

=	So sánh bằng
< = >	So sánh bằng cả khi hai giá trị đem so sánh đều là NULL
<>, !=	So sánh khác
<	So sánh nhỏ hơn
<=	So sánh nhỏ hơn hoặc bằng
>	So sánh lớn hơn
>=	So sánh lớn hơn hoặc bằng

4. Toán tử logic

Để kết hợp các biểu thức so sánh đơn lẻ thành một biểu thức chung.

Toán tử logic:

AND, &&	Và
OR, ||	Hoặc
XOR	Nếu hai biểu thức cùng đúng thì trả về giá trị false ngược lại true
NOT, !	Phủ định

V. Phát biểu SQL

1. Câu lệnh SELECT

a. Truy vấn đơn giản SELECT … FROM

Câu lệnh này giúp chúng ta chọn ra dữ liệu của các cột có trong một bảng.

Cú pháp:

```
SELECT list_column
FROM name_table
```

b. Truy vấn có sắp xếp dữ liệu

Câu lệnh SELECT … FROM kết hợp với mệnh đề ORDER BY giúp chúng ta lấy dữ liệu của các cột bên trong bảng đồng thì sắp xếp lại dữ liệu theo thứ tự tăng dần hay giảm dần.

Cú pháp:

```
SELECT list_column
FROM name_table
ORDER BY  name_column_sort [DESC,…]
```

c. Truy vấn có điều kiện WHERE

Câu lệnh SELECT … FROM kết hợp với mệnh đề điều WHERE giúp chúng ta lọc các dòng dữ liệu bên trong bảng, dữ liệu này phải thỏa mãn điều kiện đưa ra trong mệnh đề WHERE.

Cú pháp:

```
SELECT list_column
FROM name_table
WHERE conditonal
[ORDER BY  name_column_sort [DESC,…]]
```

Các phép toán thường dùng trong điều kiện lọc

- Các phép so sánh
  - >, >= : so sánh lớn hơn, lớn hơn hoặc bằng
  - <, <= : so sánh nhỏ hơn, nhỏ hơn hoặc bằng
  - = : so sánh bằng
  - !=, <> : so sánh khác

- Các phép toán học
  - and : phép và
  - or : phép hoặc
  - not : phép phủ định
  - not in : phép phủ định tập hợp
  - between : kết quả phụ thuộc vào miền giá trị
  - like : phép toán so sánh gần giống, sử dung % để thay thế ký tự
  - not like : phép phủ định so sánh gần giống
  - in : phép so sánh trong một tập hợp

d. Nhóm dữ liệu GROUP BY

Lệnh SELECT … FROM kết hợp với mệnh đề GROUP BY giúp chúng ta nhóm dữ liệu của các dòng dữ liệu bên trong bảng và sử dụng thêm các hàm thống kê đi kèm để tính toán dữ liệu có tính chất thống kê.

Cú pháp:

```
SELECT list_column
```

```
FROM name_table
[WHERE conditional]
GROUP BY list_column_group
[ORDER BY  name_column_sort [DESC,…]]
```

e. Điều kiện lọc nhóm HAVING

Cú pháp:

```
SELECT list_column
FROM name_table
[WHERE conditional]
GROUP BY list_column_group
HAVING conditional
[ORDER BY  name_column_sort [DESC,…]]
```

f. Giới hạn mẫu tin LIMIT

Cú pháp:

```
SELECT list_column
FROM name_table
[WHERE conditonal]
[GROUP BY list_columns_group]
[HAVING conditonal]
[ORDER BY  name_list_sort [DESC,…]]
LIMIT n,m
```

2. Truy vấn con

Truy vấn con là một câu lệnh select được lồng vào trong các câu lệnh truy vấn khác nhằm thực hiện các truy vấn tính toán phức tạp.

Chú ý: Khi dùng truy vấn con cần tuân theo các quy tắc sau:

+ Truy vấn con phải đặt trong dấu ngoặc đơn ()

+ Truy vấn con chỉ có thể tham chiếu đến một cột hoặc một biểu thức.

Kết quả trả về của truy vấn con có thể là một giá trị hoặc một danh sách các giá trị.

a. Truy vấn con trả về giá trị

Truy vấn con trả về một giá trị là truy vấn mà kết quả trả về của nó là một giá trị duy nhất.

Ví dụ:

b. Truy vấn con trả về danh sách các giá trị

Truy vấn con trả về danh sách các giá trị là truy vấn con mà kết qảu trả về là tập hợp các giá trị.

Toán tử IN hoặc NOT IN thường được dùng trong trường hợp này vì nó so sánh một phần tử có thuộc (hay không thuộc) tập hợp các giá trị hay không.

Ví dụ

c. Làm việc với các toán tử so sánh

Các toán tử so sánh thương được sử dụng trong truy vấn con có thể là: >, >=, <, <=, =, <>.

Chú ý: Thông thường các toán tử so sánh được sử dụng khi truy vấn con trả về một giá trị.

d. Làm việc với toán tử truy vấn con

Các toán tử truy vấn con thường hay sử dụng là: ANY, SOME, ALL, IN, NOT IN, EXISTS, NOT EXISTS.

Chú ý: Thông thường các toán tử truy vấn con được sử dụng khi dùng truy vấn cón trả về tập hợp các giá trị.

Quy tắc:

IN ⇔ ANY

NOT IN ⇔ ALL

3. Câu lệnh thêm dữ liệu

Câu lệnh INSERT INTO cho phép chúng ta thêm mới một hay nhiều dòng dữ liệu vào bên trong một bảng.

a. Giá trị trực tiếp

Khi chúng ta có giá trị trực tiếp cần thêm vào một bảng thì chúng ta sử dụng câu lệnh INSERT.

Cú pháp:

```
INSERT INTO name_table [(list_columns)]
VALUES (list_values)
```

b. Lấy từ nguồn dữ liệu

Trong trường hợp chúng ta muốn lấy dữ liệu từ các bảng khác để thêm vào bảng thì chúng ta kết hợp giữa INSERT và SELECT.

Cú pháp:

```
INSERT INTO name_table [(list_columns_table)]
SELECT list_columns_values
FROM table_source
WHERE conditional
```

4. Câu lệnh cập nhật dữ liệu

Đôi khi chúng ta có nhu cầu thay đổi giá trị của dữ liệu bên trong bảng khi chúng không còn phù hợp nữa. Câu lệnh UPDATE cho phép chúng ta cập nhật dữ liệu đã tồn tại bên trong bảng.

Chú ý: Chúng ta cần cân nhắc khi cập nhật dữ liệu bởi vì dữ liệu khi cập nhật thì không thể khôi phục lại giá trị ban đầu được nữa.

a. Giá trị trực tiếp

Khi chúng ta muốn cập nhật giá trị trực tiếp hay một biểu thức có giá trị trả về cho mẫu tin bên trong bảng, chúng ta cần dùng câu lệnh UPDATE

Cú pháp:

```
UPDATE name_table
SET name_column = value (or expression)
WHERE conditional_update
```

b. Lấy dữ liệu từ các bảng khác

Khi chúng ta muốn lấy dữ liệu từ các bảng khác để cập nhật vào bảng thì chúng ta kết hợp giữa UPDATE và SELECT

```
UPDATE name_table
SET name_column = (SELECT … FROM … WHERE …)
WHERE conditional_update
```

5. Câu lệnh xóa dữ liệu

Khi dữ liệu trong bảng không còn cần sử dụng nữa chúng ta có thể huy bỏ các dòng dữ liệu này. Câu lệnh DELETE cho phép chúng ta xóa dữ liệu trong bảng.

a. Câu lệnh xóa dữ liệu đơn giản

b. Câu lệnh xóa dữ liệu có điều kiện được lấy từ bảng khác.

6. Sử dụng mệnh đề UNION trong truy vấn

Mệnh đề UNION dùng để kết nối dữ liệu của các câu lệnh truy vấn lại với nhau:

Cú pháp:

```
SELECT danh sách các cột 1
FROM tên bảng 1
[WHERE …]
[GROUP BY … [HAVING…]]
UNION
SELECT danh sách các cột 2
FROM tên bảng 2
[WHERE …]
[GROUP BY … [HAVING…]]
[ORDER BY…]
```

Chú ý:

+ Với truy vấn sử dụng UNION thì danh sách các cột trong các câu truy vấn phải tương ứng với nhau về số lượng, thứ tự và kiểu dữ liệu của các cột.

+ Khi dùng UNION, việc đặt tiêu đề cột được thực hiện ngay truy vấn đầu tiên.

+ Với UNION có thể kết hợp nhiều truy vấn với nhau.

Ví dụ:

7. Truy vấn dữ liệu từ nhiều bảng

Khi muốn liên kết các bảng có quan hệ với nhau để lấy ra dữ liệu chung chúng ta kết hợp lệnh SELECT … FROM với mệnh đề JOIN.

Khi sử dung JOIN để nối các bảng chúng ta cần phải lưu ý những bảng này phải có các cột liên hệ với nhau và thứ tự quan hệ chúng ta chỉ định giữa các bảng cũng sẽ làm ảnh hưởng tới kết quả truy vấn.

a. INNER JOIN

Khi kết nối các bảng dùng INNER JOIN, ta chỉ định việc so sánh giá trị trong các cột của các bảng là tương đương – dữ liệu đều có ở cả hai bảng.

Kết quả sau khi thực hiện truy vấn kết nối INNER JOIN là các mẫu tin thỏa điều kiện quan hệ ở cả hai bảng.

Cú pháp:

```
SELECT list_column
FROM name_table
INNER JOIN name_table_link ON conditional_link
[WHERE conditional]
[ORDER BY list_column_sort [DESC]]
```

Ví dụ:

b. LEFT JOIN, RIGHT JOIN

Khi kết nối các bảng dùng LEFT|RIGHT JOIN, ta chỉ định việc so sánh giá trị trong các cột của các bảng được ưu tiên cho mối quan hệ bên nhánh trái | phải. Việc đổi thứ tự ưu tiên này sẽ làm ảnh hưởng tới kết quả truy vấn.

Cú pháp:

```
SELECT list_column
FROM name_table
LEFT|RIGHT name_table_link
ON conditional_link
[WHERE conditional]
[ORDER BY list_column_sort [DESC]]
```

c. Mệnh đề liên kết dữ liệu nhiều bảng

Cũng với SELECT … FROM với JOIN, chúng ta có thể kết hợp nhiều bảng dữ liệu trong một câu lệnh truy vấn. Một bảng có thể liên kết với một hay nhiều bảng khác nhau trong cùng một câu truy vấn.

Cú pháp:

```
SELECT list_column
FROM name_table_1
INNER LEFT|RIGHT name_table_2
ON conditional_link_2
INNER LEFT|RIGHT name_table_3
ON conditional_link_2
…
[WHERE conditional]
[ORDER BY list_column_sort [DESC]]
```

8. Sử dụng hàm trong SQL

a. Các hàm cấu trúc điều khiển

- Hàm IF

Cú pháp:

IF(biểu_thức_so_sánh, biểu_thức_1, biểu_thức_2)

Kiểm tra biểu thức so sánh đúng thì kết quả trả về là một biểu thức 1, ngược lại kết quả trả về là biểu thức 2

Ví dụ:

- Hàm IFNULL

Cú pháp:

- Hàm NULLIF

Cú pháp:

- Hàm CASE

Cú pháp:

b. Các hàm chuyển đổi kiểu dữ liệu

- Hàm CAST
- Hàm CONVERT

c. Hàm xử lý chuỗi

- Hàm CHAR_LENGHT, hàm CHARACTER_LENGHT VÀ LENGTH
- Hàm CONCAT và hàm CONCAT_WS
- Hàm LOWER và hàm UPPER
- Hàm LEFT, hàm RIGHT, hàm MID và hàm SUBSTRING
- Hàm REPEAT
- Hàm REVERSE
- Hàm REPLACE
- Hàm ENCODE và hàm DECODE
- Hàm SPACE
- Hàm STRCMP

d. Các hàm sử lý số

- Hàm ABS()

- Hàm CEILING()/ hàm CEIL()
- Hàm FLOOR()
- Hàm MOD()
- Hàm PI()
- Hàm POW() và hàm POWER()
- Hàm ROUND()
- Hàm SQRT
- Hàm SIGN()
- Hàm RAND()

e. Các hàm xử lý thời gian

- Hàm ADDDATE/ DATE_ADD/ SUBDATE()/ DATE_SUB()
- Hàm CURDATE / CURRENT_DATE / CURTIME/ CURRENT_TIME/ NOW()
- Hàm DATE()/ MONTH() MONTHNAME() / YEAR()
- Hàm DAY()/DAYOFMONTH/ DAYNAME/ DAYOFWEEK/ DAYOFYEAR
- Hàm SECOND/ MINUTE/ HOUR/ TIME
- Hàm DATEDIFF/ TIMEDIFF

9. Import và export dữ liệu

a. Import dữ liệu

Nhập dữ liệu từ bên ngoài vào database trong MySQL

b. Export dữ liệu

Xuất dữ liệu từ database trong MySQL ra tập tin thuộc một trong các dạng sau:

- SQL
- LaTex
- Microsoft Excel 2000
- Microsoft Word 2000
- CSV for MS Excel
- CSV
- XML

Chương 9: PHP&MYSQL

I. Kết nối CSDL

1. Tạo kết nối

Trước khi chúng ta truy cập và làm việc với dữ liệu trong CSDL, chúng ta cần phải tạo kết nối đến CSDL.

Để thực hiện được công việc này, chúng ta sử dụng hàm **mysql_connect()**

Cú pháp:

```
mysql_connect(servername, username, password);
```

Trong đó:

+ servername: tham số tùy chọn, xác định server cần phải kết nối tới. Giá trị mặc định là localhost.

+ username: tham số tùy chọn, xác định tên người dùng đăng nhập vào hệ thống

+ password: tham số tùy chọn, xác định mật khẩu người dùng. Giá trị mặc định là "".

Chú ý: Hàm này có nhiều hơn 3 tham số, tuy nhiên những tham số trên cần thiết và rất quan trọng.

Ví dụ: Tạo kết nối

```
<?php
// tạo kết nối và lưu vào biến $conn
$conn =  mysql_connect("localhost","root","");
// kiểm tra kết nối
if(!$conn)
{
      die ("Can not connect database ".mysql_error());
      exit;
}
?>
```

2. Chọn CSDL

Sau khi đã tạo được kết nối, chúng ta cần phải chọn một CSDL để làm việc. Để thực hiện công việc này, chúng ta sử dụng hàm **mysql_select_db**().

Kết quả trả về của hàm này là TRUE nếu chọn CSDL thành công, ngược lại giá trị trả về là FALSE.

Cú pháp:

mysql_select_db(database, connection)

Trong đó:

+ database: tham số bắt buộc, xác định tên CSDL cần làm việc

+ connection: tham số tuy chọn, xác định kết nối. Nếu không xác định thì kết nối cuối cùng được mở bởi hàm **mysql_connect()** hoặc hàm **mysql_pconnect**() sẽ được sử dụng.

Ví dụ: Kết nối đến CSDL khoacntt

```
<?php
$conn =  mysql_connect("localhost","root","");
if(!$conn)
{
	echo "Can not connect database";
	exit;
}
// chọn CSDL khoacntt
$select_db = mysql_select_db("khoacntt",$conn);
// kiểm tra CSDL
if (!$select_db)
{
	die("Can not connect database".mysql_error());
}
?>
```

3. Truy vấn dữ liệu

Để thực hiện việc truy vấn dữ liệu, chúng ta dùng hàm **mysql_query()**.

Hàm **mysql_query()** sẽ trả về kết quả của cấu lệnh truy vấn nếu thực hiện thành công, ngược lại sẽ trả về FALSE.

Cú pháp:

mysql_query(query, connection) ;

Trong đó:

+ query: tham số bắt buộc, là câu lệnh truy vấn được gửi đi.

+ connection: tham số tùy chọn, xác định kết nối. Nếu không xác định thì kết nối cuối cùng được mở bởi hàm **mysql_connect()** hoặc **mysql_pconnect**() sẽ được sử dụng.

Ví dụ: Thực hiện truy vấn dữ liệu trong bảng sinhvien trong CSDL khoacntt.

```
<?php
```

```
$conn = mysql_connect("localhost","root","");
if (!$conn)
{
	echo "Can not connection database";
	exit;
}
mysql_select_db('khoacntt',$conn);
$sql ="SELECT * FROM sinhvien";
$result= mysql_query($sql);
while($r = mysql_fetch_array($result))
{
	echo "MaSV: ".$r['ma_sv']."  Ten sv: ".$r['ten_sv']."</br>";
}
?>
```

4. Thông báo lỗi

Trong quá trình làm việc với CSDL lỗi có thể phát sinh. Do đó, chúng ta cần thông báo các lỗi phát sinh này.

Để thực hiện công việc này, chúng ta sử dụng hàm **mysql_error()**.

Hàm này có kết quả trả về là câu thông báo lỗi nếu có lỗi phát sinh, ngược lại kết quả trả về sẽ là một chuỗi rỗng “”.

Cú pháp:

```
mysql_error(connection)
```

Trong đó:

connection: tham số tùy chọn, xác định kết quả kết nối. Nếu không kết nối thì kết quả cuối cùng được mở bởi hàm **mysql_connect()** hoặc **mysql_pconnect**() sẽ được sử dụng.

Ghi chú: Chúng ta thường kết hợp hàm **mysql_error()** với hàm **die**() hoặc hàm **exit**() để vừa thông báo lỗi vừa kết thúc công việc.

Ví dụ: Thông báo lỗi nếu không tạo kết nối

```
<?php
$conn = mysql_connect("localhost","root","");
if (!$conn)
{
	echo "Can not connection database";
	exit;
```

```
}
?>
```

5. Đóng kết nối

Sau khi đã làm việc xong với CSDL, chúng ta cần đóng kết nối đã mở.

Để thực hiện công việc này, chúng ta sử dụng hàm **mysql_close()**.

Hàm này có kết quả trả về là TRUE nếu đóng kết nối thành công, ngược lại sẽ trả về giá trị là FALSE nếu thất bại.

Cú pháp:

```
mysql_close(connection);
```

+ connection: tham số tùy chọn, xác định kết nối. Nếu không xác định thì kết nối cuối cùng được mở bởi hàm **mysql_connect() mysql_pconnect**() sẽ được sử dụng.

Ví dụ: Đóng kết nối

```
<?php
$conn = mysql_connect("localhost","root","");
……..
mysql_close($conn);
?>
```

II. Làm việc với CSDL MySQL

1. Đếm số lượng mẫu tin

Thông thường, chúng ta cần biết số lượng mẫu tin được truy vấn dữ liệu trước khi xử lý các công việc tiếp theo.

Để thực hiện công việc này, chúng ta sử dụng hàm **mysql_num_rows()**.

Hàm này có kết quả trả về là số lượng mẫu tin nếu thành công, ngược lại kết quả trả về là FALSE nếu thất bại.

Cú pháp:

```
mysql_num_rows(data);
```

Trong đó:

+ data: là tham số bắt buộc. Xác định con trỏ dữ liệu. Con trỏ dữ liệu là kết quả trả về của hàm **mysql_query()**

Ví dụ: Đếm số mẫu tin trong bảng sinhvien

```
<?php
```

```
$conn = mysql_connect("localhost","root","");
if (!$conn)
{
      echo "Can not connection database";
      exit;
}
mysql_select_db('khoacntt',$conn);
$sql ="SELECT * FROM sinhvien";
$result= mysql_query($sql);
echo "Số mẫu tin: ". mysql_num_rows($result);
mysql_close($conn);
?>
```

2. Hiển thị dữ liệu

a. Duyệt dữ liệu

Có nhiều cách để duyệt dữ liệu: duyệt dữ liệu theo dạng mỗi mẫu tin là một dòng, duyệt theo dạng mỗi mẫu tin là một mảng, duyệt theo dạng mỗi mẫu tin là một đối tượng.

Cách 1: Duyệt dữ liệu theo dạng mỗi mẫu tin là một dòng bằng hàm mysql_fetch_row().

Hàm này trả về một mảng (có chỉ số) chứa giá trị của một dòng dữ liệu với mỗi phần tử là nội dung của một cột.

Sau đó truy xuất bằng cách gọi từng phần tử của mảng $row[0], $row[1] ,$row[2],…

Cú pháp:

```
mysql_fetch_row(data)
```

+ data: là tham số bắt buộc. Xác định số con trỏ dữ liệu. Con trỏ dữ liệu là kết quả trả về của hàm **mysql_query()**.

Ví dụ: Duyệt dữ liệu trong bảng sinhvien sử dụng hàm **mysql_fetch_row()**

```
<?php
$conn =  mysql_connect("localhost","root","");
if (!$conn)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("khoacntt", $conn);
$result = mysql_query("SELECT * FROM sinhvien");
for($i=1;$i<=mysql_num_rows($result);$i++)
```

```
{
print_r(mysql_fetch_row($result));
echo "</br>";
}
mysql_close($conn);
?>
```

Cách 2: Duyệt theo dạng mỗi mẫu tin là một mảng bằng hàm mysql_fetch_array()

Hàm này trả về một mảng (có chỉ số chuỗi) chứa giá trị của một dòng dữ liệu với mỗi phần tử là nội dung của một cột.

Sau đó truy xuất bằng cách gọi từng phần tử của mảng $row[“tên cột 1”], $row[“tên cột 2”], $row[“tên cột 3”],…

Cú pháp:

```
mysql_fetch_array(data)
```

+ data: là tham số bắt buộc. Xác định số con trỏ dữ liệu. Con trỏ dữ liệu là kết quả trả về của hàm **mysql_query()**.

Ví dụ: Duyệt dữ liệu trong bảng sinhvien sử dụng hàm **mysql_fetch_array()**

```
<?php
$conn =  mysql_connect("localhost","root","");
if (!$conn)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("khoacntt", $conn);
$result = mysql_query("SELECT * FROM sinhvien");
while($row = mysql_fetch_array($result))
  {
  echo $row['ma_sv'] . " " . $row['ten_sv'];
  echo "<br />";
  }
mysql_close($conn);
?>
```

Cách 3: Duyệt theo dạng mỗi mẫu tin là một đối tượng bằng hàm mysql_fetch_object()

Hàm này có kết quả trả về là một mẫu tin trong bộ các mẫu tin như một đối tượng.

Sau đó truy xuất bằng cách gọi từng thuộc tính của đối tượng $tên_đôi_tượng ->tên cột 1, $tên_đôi_tượng ->tên cột 2, $tên_đôi_tượng ->tên cột 3,…

Cú pháp:

```
mysql_fetch_object(data)
```

+ data: là tham số bắt buộc. Xác định số con trỏ dữ liệu. Con trỏ dữ liệu là kết quả trả về của hàm **mysql_query()**.

Ví dụ: Duyệt dữ liệu trong bảng sinhvien sử dụng hàm **mysql_fetch_object()**

```
<?php
$conn =  mysql_connect("localhost","root","");
if (!$conn)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("khoacntt", $conn);
$result = mysql_query("SELECT * FROM sinhvien");
for($i=1;$i<=mysql_num_rows($result);$i++)
{
        $ob =mysql_fetch_object($result);
        echo $ob->ma_sv.", ".$ob->ten_sv."</br>";
}
mysql_close($conn);
?>
```

b. Hiển thị dữ liệu không định dạng

Khi chúng ta chỉ có nhu cầu hiện thị dữ liệu mà không cần định dạng thì cách đơn giản nhất là in dữ liệu theo dạng bảng.

Ví dụ: Hiển thị thông tin sinh viên trong bảng sinhvien

```
<p > <h1 align="center">THÔNG TIN SINH VIÊN</h1>
<table width="1000" border="1" cellspacing="2" cellpadding="2" align="center">
  <tr>
    <td width='100px'>Mã SV</td>
        <td width='250px'>Tên SV</td>
    <td width='80px'>Ngay sinh</td>
        <td width='70px'>Gioi tinh</td>
        <td width='300px'>Quen quan</td>
    <td width='100px'>Nam hoc</td>
        <td width='100px'>Lop hoc</td>
  </tr>
  <?php
  $conn =  mysql_connect("localhost","root","");
        if (!$conn)
```

```
        {
                die('Could not connect: ' . mysql_error());
        }
  mysql_select_db("khoacntt", $conn);
  $result = mysql_query("SELECT * FROM sinhvien");
  while($row = mysql_fetch_array($result))
  {
  echo "<tr>";
  echo "<td width='100px'>".$row[0]."</td>";
  echo "<td width='260px'>".$row[1]."</td>";
  echo "<td width='80px'>".$row[2]."</td>";
  echo "<td width='70px'>".$row[3]."</td>";
  echo "<td width='300px'>".$row[4]."</td>";
  echo "<td width='100px'>".$row[5]."</td>";
  echo "<td width='100px'>".$row[6]."</td>";
  echo "</tr>";
  }
  mysql_close($conn);
  ?>
</table>
</p>
```

c. Hiển thị dữ liệu có yêu cầu

Khi xây dựng một ứng dụng web nếu chúng ta hiển thị thông tin không có định dạng thì sẽ làm cho người xem cảm thấy nhàm chán và khó khăn trong quá trình đọc và tìm kiếm dữ liệu, chính vì vậy mà phát sinh các yêu cầu hiển thị dữ liệu khác nhau.

- Định dạng hiển thị dữ liệu

Yêu câu đơn giản nhất của một trang web là dữ liệu hiển thị phải được định dạng giúp cho người dùng dễ dàng đọc thông tin.

Các định dạng thông thường là định dạng các cột, dòng và kích thước, màu sắc, kiểu chữ, … cho nội dung hiển thị.

Chú ý: Để thực hiện được các yêu cầu về định dạng, chúng ta cần phải xem xét và tính toán các nội dung cần hiển thị.

Ví dụ:

- Tạo các cột tùy biến

Đôi khi, nội dung của một số cột trong CSDL khi hiển thị sẽ gây cho người dùng khó hiểu, hoặc chúng ta phải có dòng giải thích cho những thông tin đó. Chính vì vậy mà nhu cầu hiển

thị dữ liệu một cách rõ ràng, chi tiết, dễ hiểu trở thành một yêu cầu tất yếu khi thiết kế trang web.

Ví dụ:

- Phân trang

Đối với những bảng có số lượng mẫu tin lớn khi hiển thị dữ liệu sẽ làm cho người dùng cảm thấy khó xem. Phân trang dữ liệu giúp cho việc xem và tìm kiếm thông tin dễ dàng, nhanh chóng và thuận tiện.

Ví dụ:

- Liên kết trang có chuỗi tham số.

* Chuỗi URL có cấu trúc định dạng như sau:

http://<host>[:<port>][path][?string_parameter]

Trong đó:

string_parameter có Cú pháp như sau:

[?parameter_1 = value_1 [&parameter_2 = value_2[&…]]]

Chú ý: Trong trường hợp có nhiều tham số, cặp [parameter_1 = value_1] phân cánh nhau bằng dấu &

Ví dụ:

Để nhận giá trị truyền qua tham số này ta dùng biến $_REQUEST[“parameter”]

Ví dụ:

d. Chuyển đổi giá trị thời gian

Định dạng date time trong MySQL có dạng năm-tháng-ngày.

Ví dụ: Khi nhập ngày – tháng – năm ở PHP như sau: 25th January 2009 thì khi lưu trữ vào CSDL MySQL chúng sẽ được định nghĩa như sau: 2009-01-25

Khi nhập ngày tháng năm từ PHP vào MySQL ta có thể dùng hàm date() đã học hoặc sử dụng hàm DATE_FROMAT() trong MySQL

Để định dạng ngày tháng năm YYYY-MM-DD của MySQL sang định dạng MM-DD-YYYY có thể dùng hàm DATE_FORMAT() với Cú pháp như sau:

```
SELECT DATE_FORMAT (column_date ‘%M-%D-%Y’)
Hay
SELECT DATE_FORMAT (column_date, %M %D %Y)
```

Trong đó: %M và %D là định dạng tháng và ngày với 2 ký số, %Y là định dạng năm với 4 ký số.

Chú ý: Chúng ta cũng có thể căn cứ vào bảng định dạng thời gian sau để định dạng theo yêu câu:

Ký tự	Mô tả
%M	Tên tháng trong năm(January, February…)
%W	Tên thứ trong tuần (Sunday, Monday,…)
%D	Ngày trong tháng theo dạng 1^{st}, 2^{nd}, 3^{th}…
%Y	Năm với 4 ký số(2010)
%y	Năm với 2 ký số(10)
%m	Tháng với hai ký số (01-12)
%d	Ngày trong tháng với hai ký số (01-31)
%H	Giờ trong ngày với hai ký số(00-23)
%h	Giờ trong ngày với hai ký số(00-12)
%i	Phút trong giờ với hai ký số(01-59)
%r	12 giờ với định dạng (hh:mm:ss [AM|PM])
%T	12 giờ với định dạng (hh:mm:ss)
%S	Giây trong phút có hai ký số (00-59)
%p	AM hay PM

3. Lưu trữ dữ liệu mới vào CSDL

Để lưu thông tin từ trong PHP vào trong MySQL, chúng ta sẽ sử dụng hàm **mysql_query()** kết hợp với câu lệnh truy vấn INSERT INTO. Hàm này được dùng để gửi một truy vấn(hiển thị thông tin, thêm mới, xóa, cập nhật) tới một kết nối MySQL

Cú pháp:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
Hoặc
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Chú ý: Cú pháp INSERT INTO dùng để thêm một mẫu tin mới vào bảng trong CSDL

Ví dụ: Tạo form nhập thông tin của một mẫu tin(form_insert.php)

Các tên tương ứng

- masv
- tensv
- d, m, y
- gt
- quequan
- namhoc
- lophoc

Ví dụ: Trang lấy thông tin từ form (insert_sinhvien.php)

```
<?php
$masv = $_POST['masv'];
$tensv = $_POST['tensv'];
$d = $_POST['d'];
$m = $_POST['m'];
$y = $_POST['y'];
$gt = $_POST['gt'];
$quequan = $_POST['quequan'];
$namhoc = $_POST['namhoc'];
$lophoc = $_POST['lophoc'];
// xu ly dl ngay thang nam
if (checkdate($m,$d,$y))
{
	$ngaysinh =$d."/".$m."/".$y;
}
// kiem tra xem trong csdl co masv nay chua
function kt($masv)
{
	$conn = mysql_connect("localhost","root","");
	mysql_select_db("khoacntt",$conn);
	$sql="SELECT ma_sv FROM sinhvien WHERE ma_sv='$masv'";
	$result=mysql_query($sql);
	$kt1=true;
	while ($row = mysql_fetch_array($result))
	{
		$kt1=false;
```

```
        }
        return $kt1;
        mysql_close($conn);
}
if (kt($masv)&& checkdate($m,$d,$y))
{
        $conn = mysql_connect("localhost","root","");
        mysql_select_db("khoacntt",$conn);
        $sql = "INSERT INTO sinhvien
VALUES('$masv','$tensv','$ngaysinh','$gt','$quequan','$namhoc','$lophoc')";
        mysql_query($sql);
}
?>
```

4. Cập nhật dữ liệu

Để cập nhật thông tin từ PHP vào MySQL, chúng ta sẽ sử dụng hàm **mysql_query()** kết hợp với câu lệnh truy vấn UPDATE. Hàm này được dùng để gửi một truy vấn (hiển thị thông tin, thêm mới, xóa, cập nhật) tới một kết nối MySQL

Cú pháp:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Ví dụ: Form cập nhật dữ liệu bảng sinh viên

- Mã sinh viên được đọc từ CSDL với tên listbox là masv.
- Tên sinh viên mới có tên textfield là tensv

Ví dụ: Trang lấy thông tin từ form cập nhật dữ liệu

```
<?php
$conn = mysql_connect("localhost","root","");
mysql_select_db("khoacntt", $conn);
if(strlen($_POST['tensv']))
{
```

```
        $tensv = trim($_POST['tensv']);
        $masv = trim($_POST['masv']);
        mysql_query("UPDATE sinhvien SET ten_sv = '$tensv' WHERE ma_sv = '$masv'");
}
mysql_close($conn);
?>
```

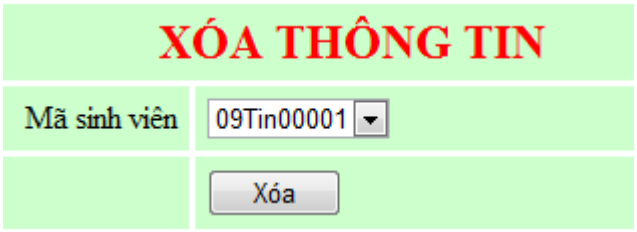
5. Xóa dữ liệu

Để xóa thông tin trong CSDL thông qua PHP, chúng ta sẽ sử dụng hàm **mysql_query()** kết hợp với câu lệnh truy vấn DELETE. Hàm này được dùng để gửi một truy vấn (hiển thị thông tin, thêm mới, xóa, cập nhật) tới một kết nối MySQL

Cú pháp:

```
DELETE FROM table_name
WHERE some_column = some_value
```

Ví dụ: Form xóa dữ liệu trong bảng sinhvien



- Chọn mã sinh viên cần xóa có tên listbox là masv, value được lấy trong CSDL

Ví dụ: Lấy thông tin từ form và xóa dữ liệu trong bảng sinh viên

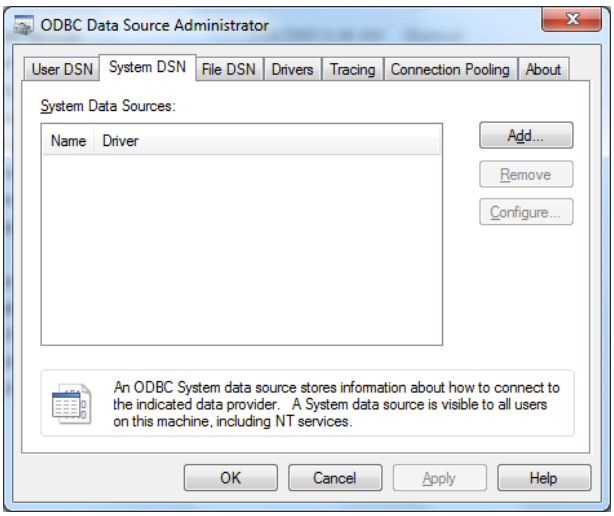
```
<?php
$conn = mysql_connect("localhost","root","");
mysql_select_db("khoacntt", $conn);
if(strlen($_POST['masv']))
{
        $masv = trim($_POST['masv']);
        mysql_query("DELETE FROM sinhvien WHERE ma_sv = '$masv'");
}
mysql_close($conn);
?>
```

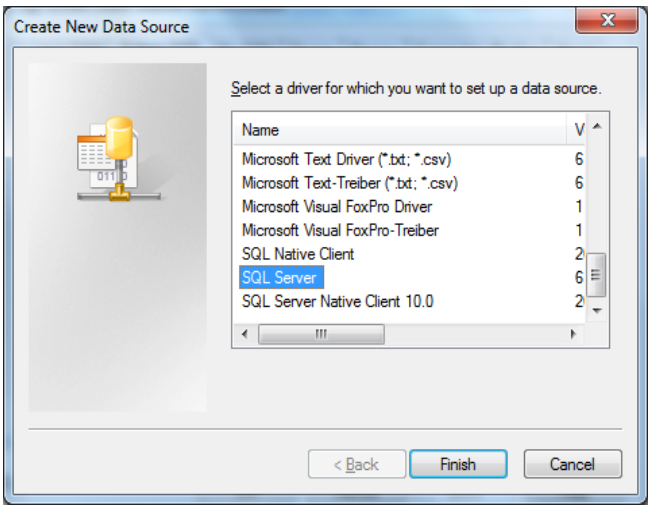
III. PHP kết hợp với các CSDL SQL Server

Bước 1: Mở Administrative Tools trong control Panel

Bước 2: Trong cửa sổ Administrative Tools chọn biểu tượng Data Sources(OBBC)

Bước 3: Trong cửa sổ ODBC Data Sources Administrator chọn tab System DSN

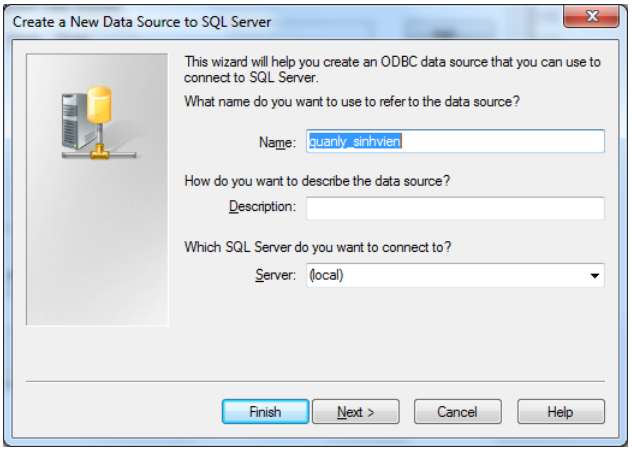


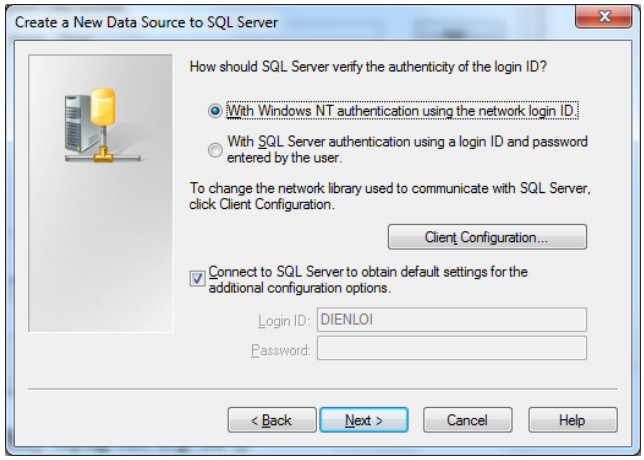


Bước 4: Nhấn vào nút Add chọn SQL Server chọn Finsh

Bước 5: Đặt tên cho Data Source (DSN) trong mục Name và chọn server cần kết nối tới trong mục server => nhấn nút next

Xác định cách thức kết nối tới SQL Server theo tài khoản của Window NT hay tài khoản của SQL Server => Nhấn nút next





Bước 6: Lựa chọn CSDL cần kết nối của SQL Server trong mục Change Default Database to => nhấn nút Next sau đó nhấn nút Test Data Source để kiểm tra việc kết nối với nguồn dữ liệu có thành công hay không => nhấn nút OK

Bước 7: Hoàn thành việc kết nối CSDL => nhấn nút OK

IV. Xây dựng các lớp xử lý

1. Một số phương thức trong lớp xử lý bảng

Lớp xử lý bảng dùng để xử lý các công việc liên quan tới kết nối, chọn CSDL và làm việc với CSDL,…

```
class connect_database
{
```

```
        // khai báo các thuộc tính
        // xây dựng các phương thức lớp xử lý bảng
}
```

a. Khai báo thuộc tính

Trong lớp xử lý bảng có một số thuộc tính thường xuyên được sử dụng nên chúng ta cần phải khai báo:

```
var $_sql ="";
var $_connection ="";
var $_result ="";
```

Trong đó:
+ $_sql : chứa nội dung của câu lệnh truy vấn
+ $_connection: chứa kết quả của hàm kết nối mysql_connect()
+ $_result: chứa kết quả của hàm mysql_query()

b. Kết nối CSDL

Trong hàm khởi tạo của lớp connect _database khai báo các thông tin kết nối đến CSDL.

```
{
        // Tạo và kiểm tra kết nối
        $this ->_connection =@mysql_connect("localhost","root","");
        // chọn CSDL
        $db ="name_database";
        mysql_select_db($db, $this->connection)
}
```

c. Gán giá trị cho thuộc tính $_sql

Hàm này sẽ gán giá trị cho thuộc tính $sql của lớp xử lý bảng

```
function setquery($sql)
{
        $this->_sql =$sql;
}
```

d. Lấy toàn bộ các dòng dữ liệu trong bảng

Hàm này có kết quả trả về là biến con trỏ chứa kết quả là các dòng dữ liệu trong bảng

```
function query()
{
$this->_result =mysql_query($this->_sql,$this->_connection);
return $this->_result;
```

```
}
```

Sau đó chuyển kết quả của biến $_result (gọi hàm query()) này vào mảng các dòng dữ liệu bằng hàm loadAllRow()

```
function loadAllRow()
{
	if(!($result = $this->query()))
	{
		return null;
	}$array =array();
	while ($row =mysql_fetch_assoc($result))
	{
		$array[] =$row;
	}
	mysql_free_result($result);
	return $array;
}
```

e. Đóng kết nối

Hàm này được dùng để đóng kết nối đang được mở

```
function close_connect()
{
	mysql_close($this->_connection);
}
```

2. Xây dựng lớp xử lý nghiệp vụ

Lớp xây dựng nghiệp vụ có các thành phần riêng của lớp đó. Lớp này kế thừa từ lớp xử lý bảng và có các hàm đọc, thêm, cập nhật, xóa dữ liệu.

```
class process_class extends connect_database
{
	// các thuộc tính
	// các phương thức
}
```

a. Các phương thức thường sử dụng

- Đọc dữ liệu

```
function doc_ds()
{
	$this ->setQuery("SELECT * FROM name_table");
```

```
        $result = $this->LoadAllRow();
        $this->close_connect();
        $return $result;
}
```

- Thêm dữ liệu

```
function them_moi(danh sach tham so)
{
        $this ->setQuery("INSERT INTO name_table VALUES (gia tri) ");
        $result = $this->query();
        $this->close_connect();
        $return $result;
}
```

- Cập nhật dữ liệu

```
function cap_nhat(danh sach tham so)
{
        $this ->setQuery("UPDATE name_table SET name_column=value, ...");
        $result = $this->query();
        $this->close_connect();
        $return $result;
}
```

- Xóa dữ liệu

```
function xoa(danh sach tham so)
{
        $this ->setQuery("DELETE FROM name_table WHERE ...");
        $result = $this->query();
        $this->close_connect();
        $return $result;
}
```

b. Các phương thức riêng cho các lớp

Ngoài các phương thức thường dùng được nêu ở trên, mỗi lớp xử lý nghiệp vụ còn có những phương thức đặc trưng, riêng biệt khác. Tùy theo yêu cầu mà chúng ta sẽ xây dựng các phương thức này.

Mục lục