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.

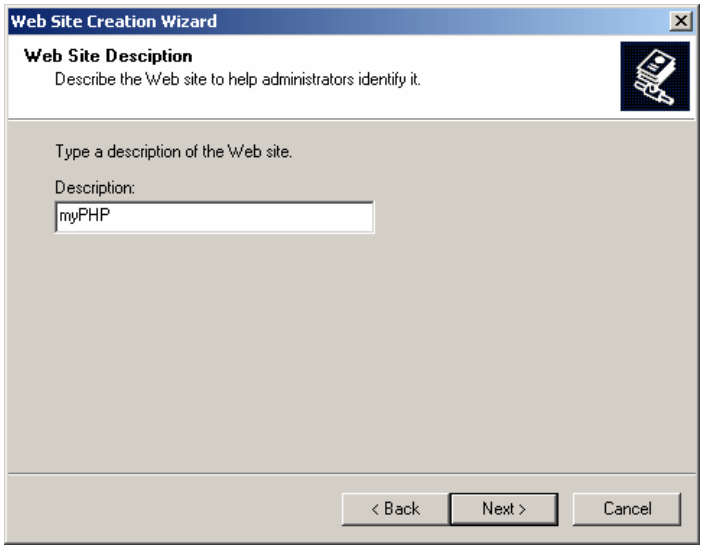
3. CẤU HÌNH ỨNG DUNG PHP

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:

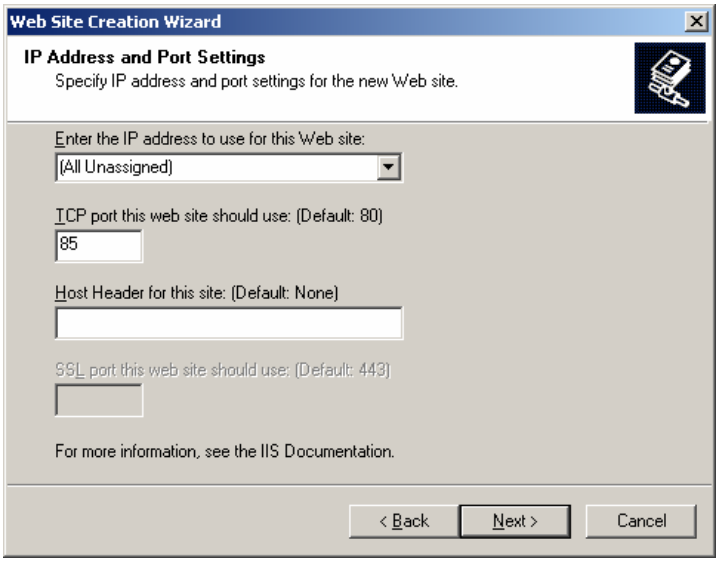
1. Tạo một thư mục có tên myPHP để lưu trữ các tập tin PHP
2. Khởi động IIS (tự động khởi động nếu Windows NT/2000)

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

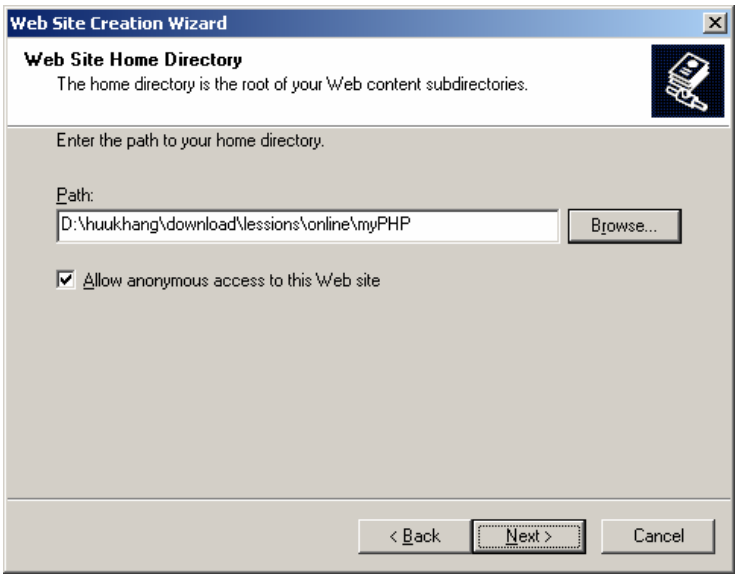
7. Chọn nút Next và khai báo IP và port, trong trường hợp bạn không sử dụng port 80 cho ứng site khác thì chọn giá trị mặc định. Tuy nhiên nếu có nhiều ứng dụng trước đó đã cấu hình trong IIS thì bạn có thể thay đổi port khác, ví dụ chọn port 85 như hình 1-2.



Hình 1-2: Khai báo IP và Port

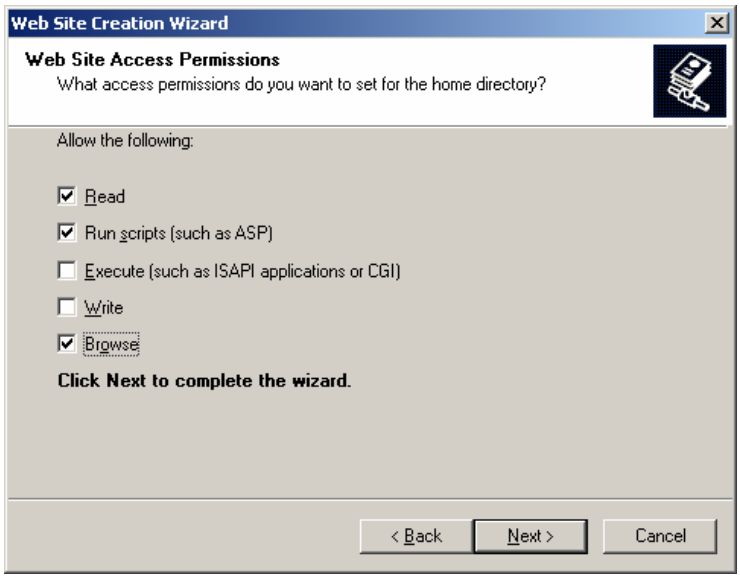
Lưu ý rằng, port 80 là port chuẩn điều này có nghĩa là khi triệu gọi trên trình duyệt bạn không cần gõ port, ví dụ http://localhost/. Đối với trường hợp port khác thì bạn phải gõ tương tự như http://localhost:85/

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.



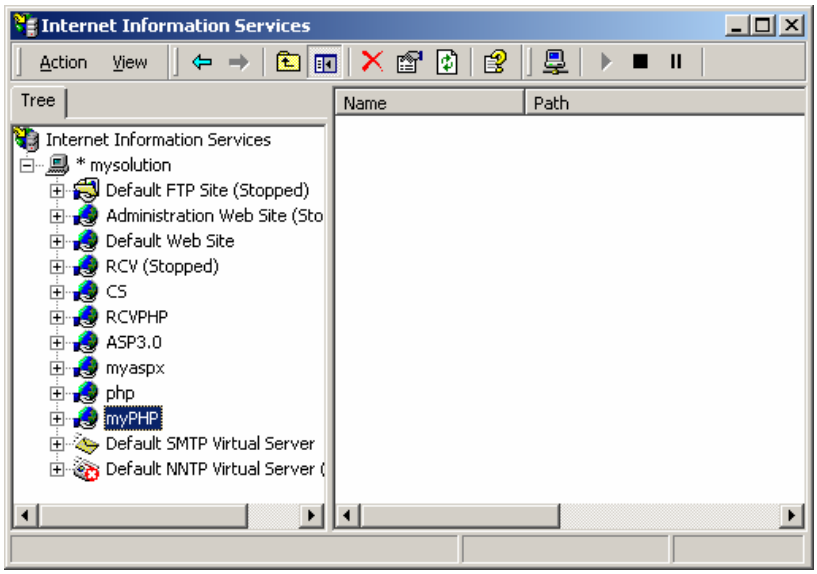
Hình 1-3: Chọn thư mục myPHP

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.



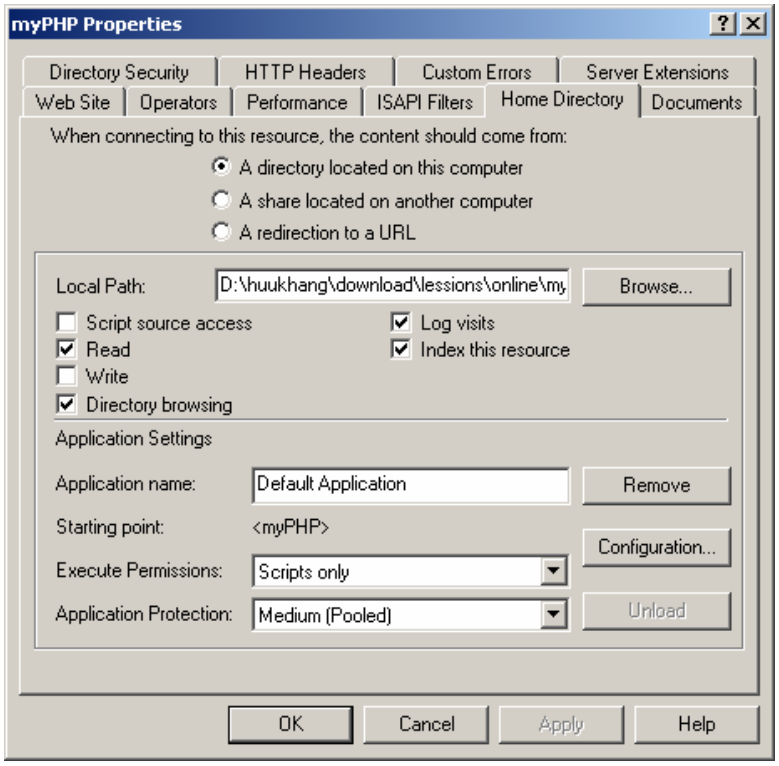
Hình 1-4: Quyền truy cập

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.



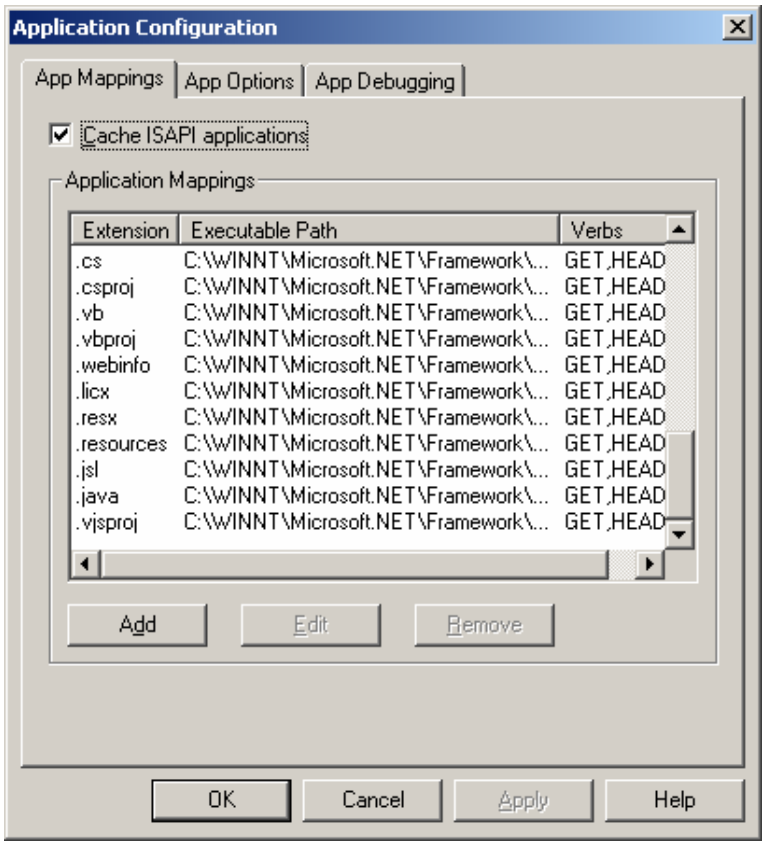
Hình 1-5: Tạo thành công ứng dụng PHP trong IIS

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



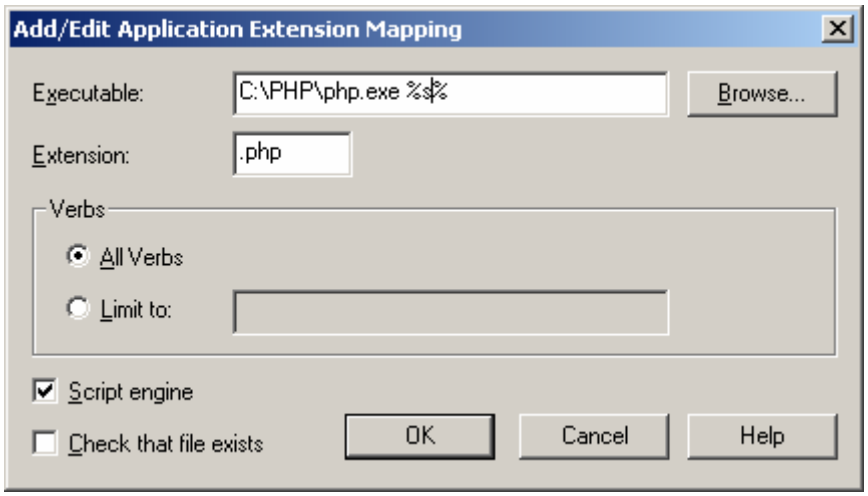
Hình 1-5: Cấu hình PHP trong IIS

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



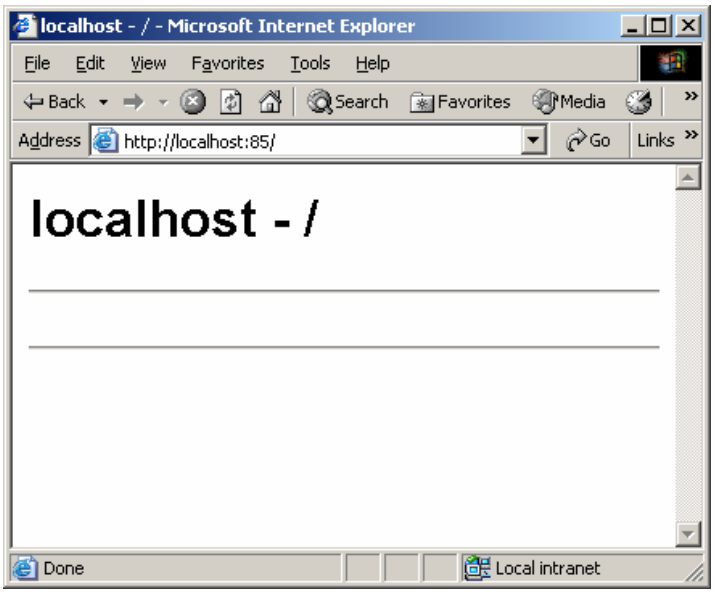
Hình 1-6: Thêm PHP Engine

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

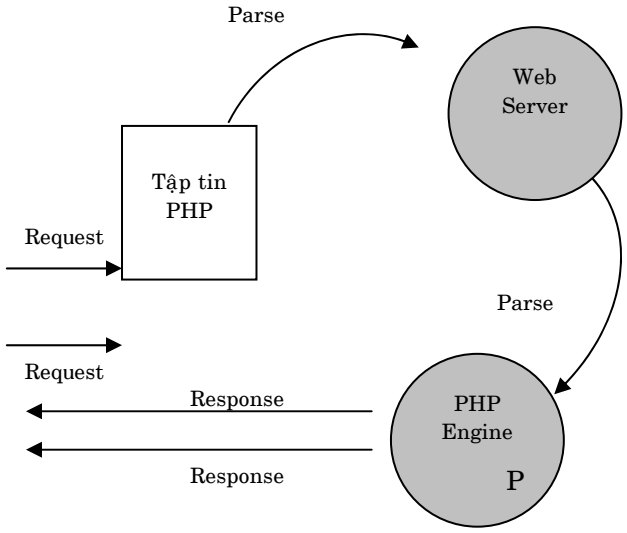
PHP dựa trên cú pháp của ngôn ngữ lập trình C, chính vì vậy khi làm việc với PHP bạn phải là người có kiến thức về ngôn ngữ C, C++, Visual C. Nếu bạn xây dựng ứng dụng PHP có kết nối cơ sở dữ liệu thì kiến thức về cơ sở dữ liệu MySQL, SQL Server hay Oracle là điều cần thiết.

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:

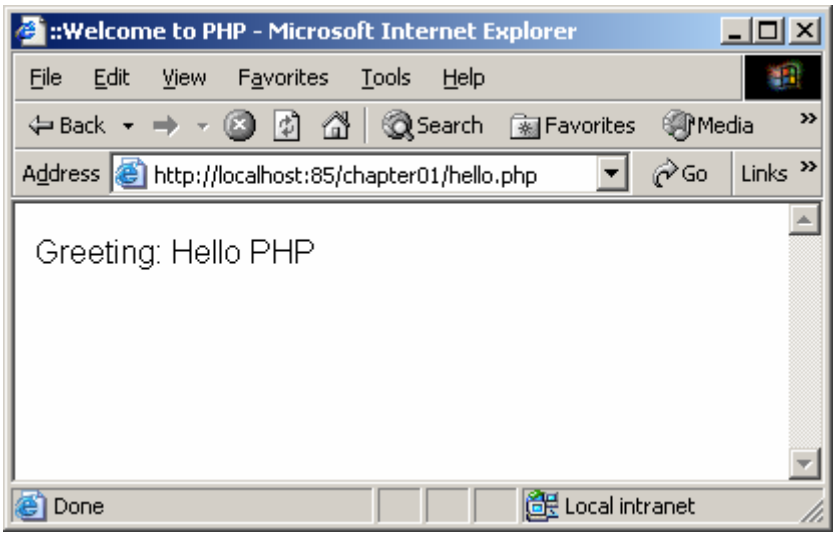
```
<br>
1-Giá trị biến Str: <?=$groupid?>
2-Giá trị biến i: <?=$i?>
3-Giá trị cũ thể: <?=10?>
```

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
```

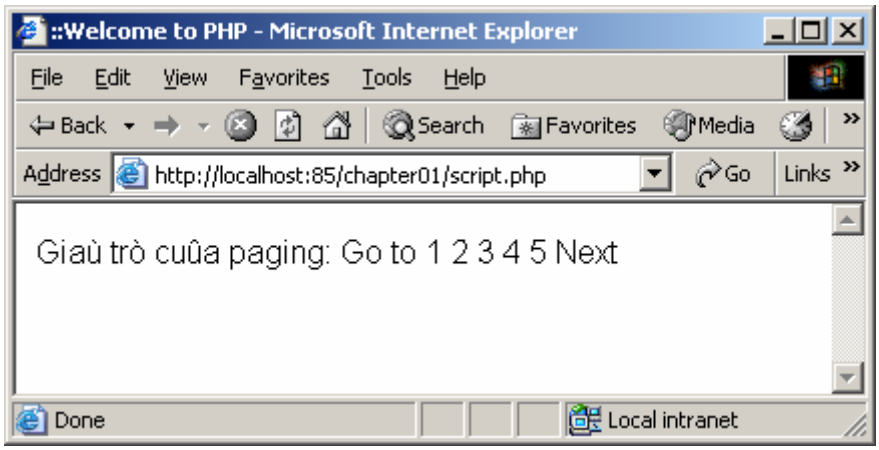
Lưu ý rằng, kết thúc mỗi câu lệnh phải dùng dấu ;

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:

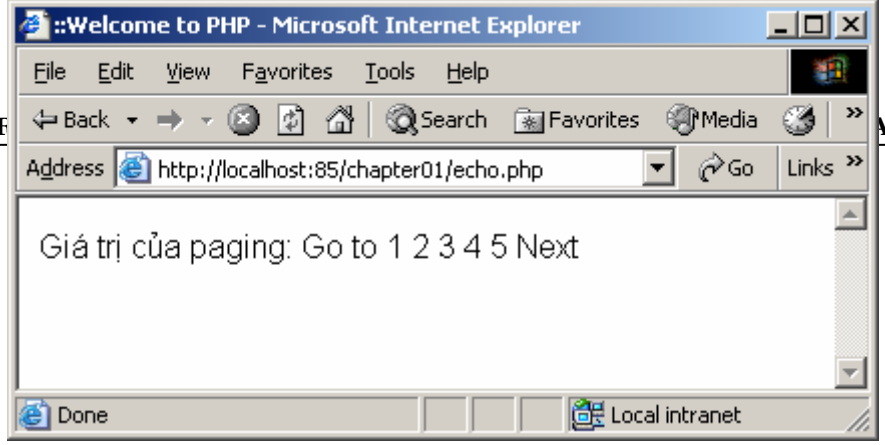
```
<?php
            $stSQLs=”select * from Customers”;
            echo $stSQLs;
?>
```

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	a = b a++ a-- a += b a -= b a *= b a /= b a %= b a |= b a &= b a ^= b a .= b
	++	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	
	.=		
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>
...
```

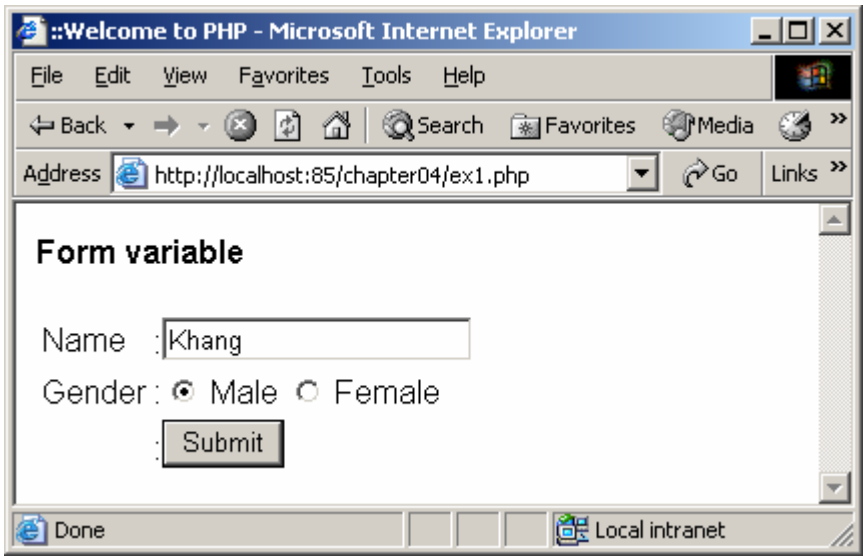
Khi người sử dụng nhập giá trị vào phần Name và chọn giới tính Male hay Female như hình 4-1, nếu nhấn nút submit thì trang ex1-1.php sẽ triệu gọi, trong trang này bạn có thể lấy giá trị nhập từ trang ex1.php bằng cách sử dụng biến form như ví dụ 4-1-1.

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?>
```

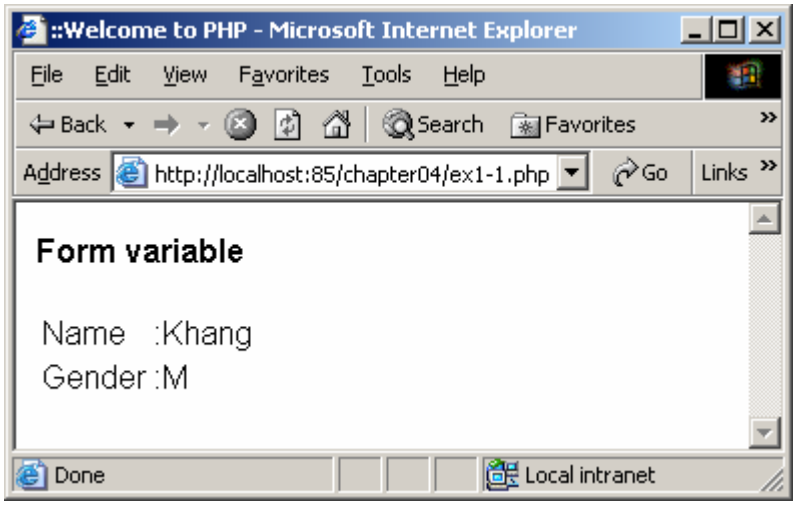
```
</td></tr>
<tr><td>Gender</td>
<td>
  :<?=$gender?>
</td></tr>
</table>
</BODY>
</HTML>
```

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

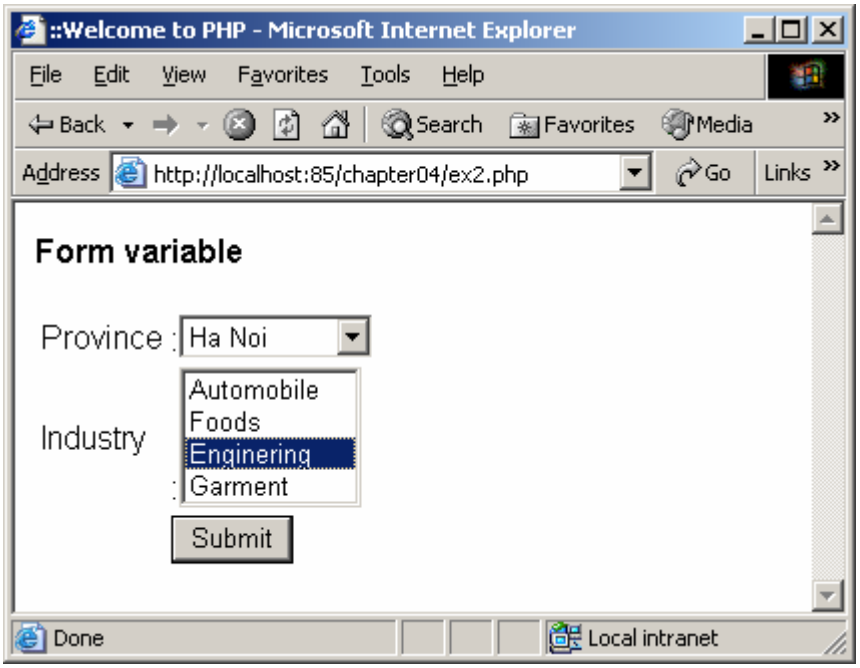
1.2. Biến form từ form được submit với phương thức GET

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

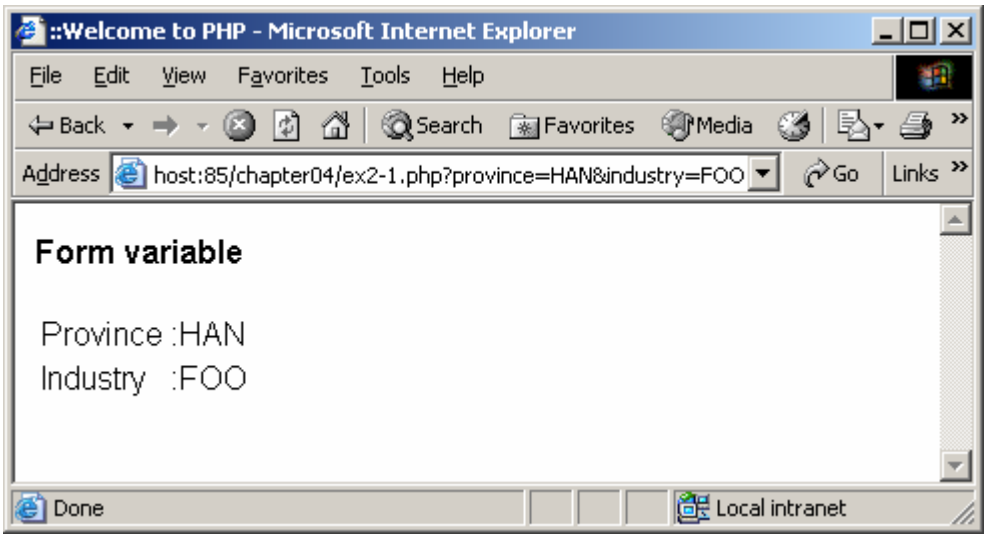
```
...
<form action=ex2-1.php method=get>
<tr><td>Province</td>
<td>
   :<select name=province>
   <option value=HAN>Ha Noi</option>
   <option value=HCM>Ho Chi Minh</option>
   <option value=HUE>Hue</option>
   </select>
</td></tr>
<tr><td>Industry</td>
<td>
   :<select name=industry multiple>
   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



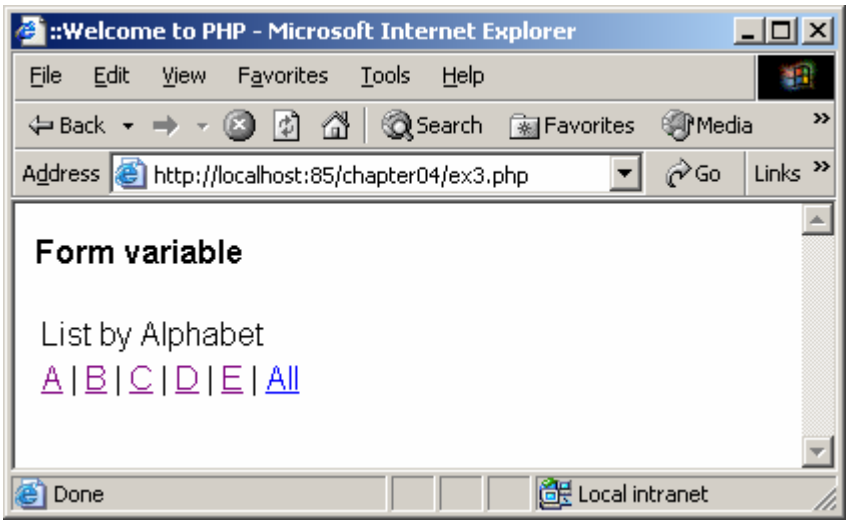
Hình 4-2-1: Biến form với phương thức GET

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>
```

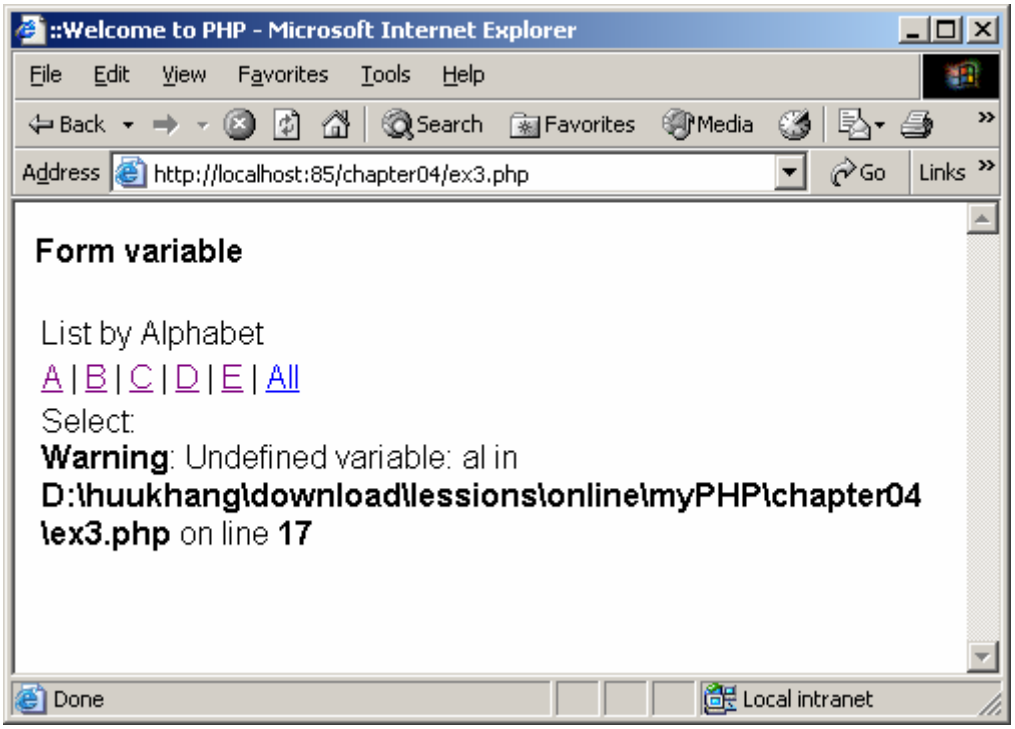
Tuy nhiên, lần đầu tiên triệu gọi trang này mà không có tham số trên QueryString, khai báo biến form sẽ phun ra lỗi như hình 4-3-1.

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

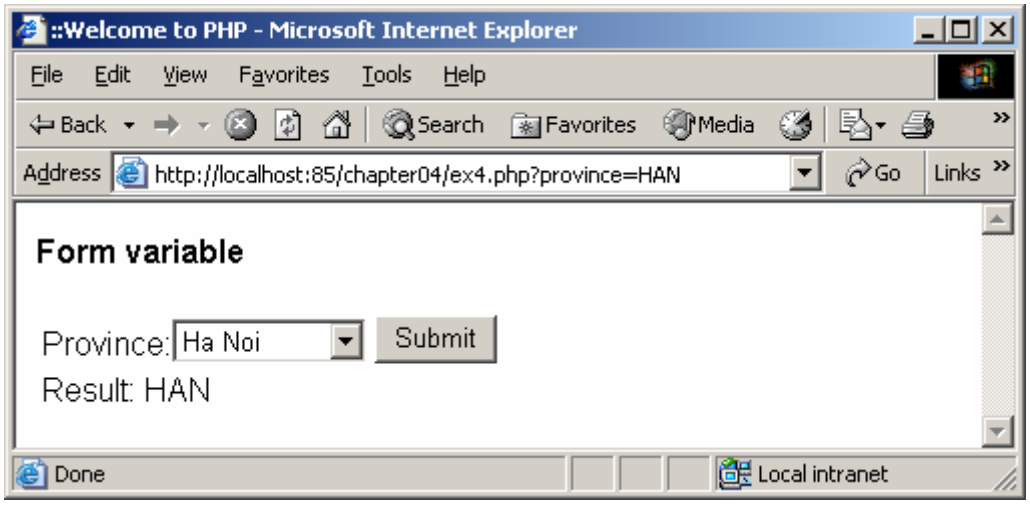
Ngoài cách sử dụng biến form trong trường hợp lấy giá trị từ tham số của QueryString, bạn có thể sử dụng hàm $HTTP_GET_VARS. Ví dụ, chúng ta khai báo trang PHP như ví dụ 4-4.

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

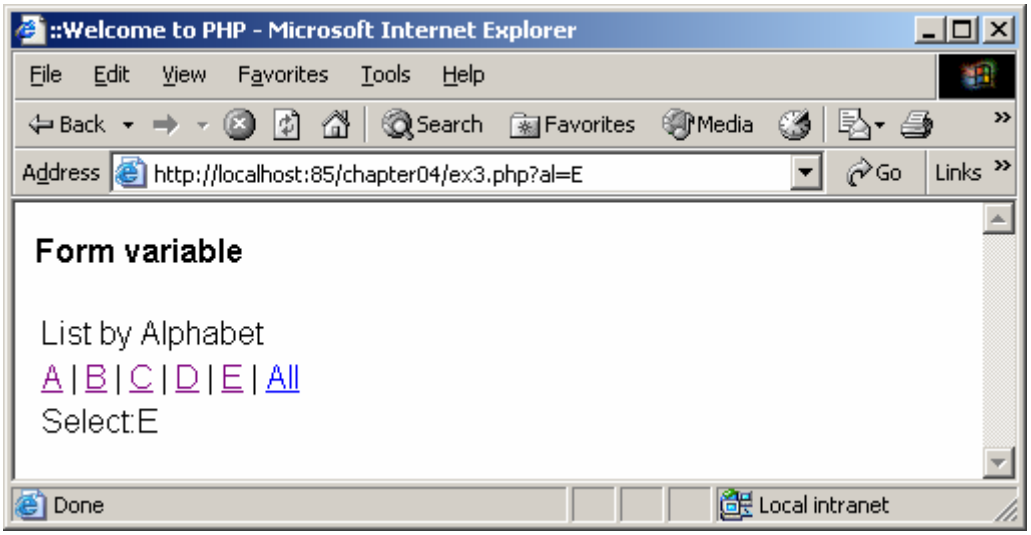
Tương tự như vậy trong trường hợp bạn không sử dụng thẻ form mà giá trị lấy từ chuỗi QueryString bằng cách sử dụng $HTTP_GET_VARS như ví dụ 4-5.

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

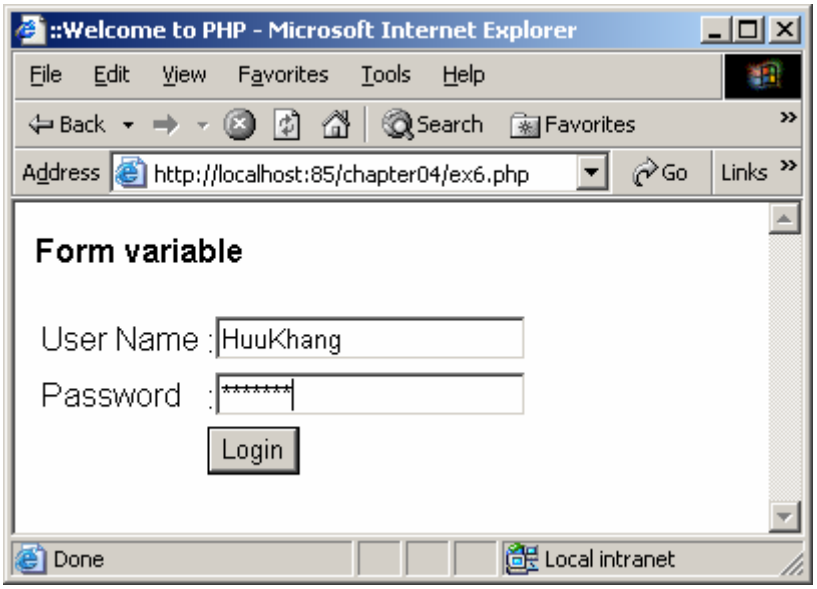
Tương tự như $HTTP_GET_VARS nhưng $HTTP_POST_VARS cho phép bạn lấy giá trị lấy từ các thẻ nhập liệu của thẻ form trong traang submit trước đó. Ví dụ, bạn khai báo trang nhập liệu như ví dụ 4-6.

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

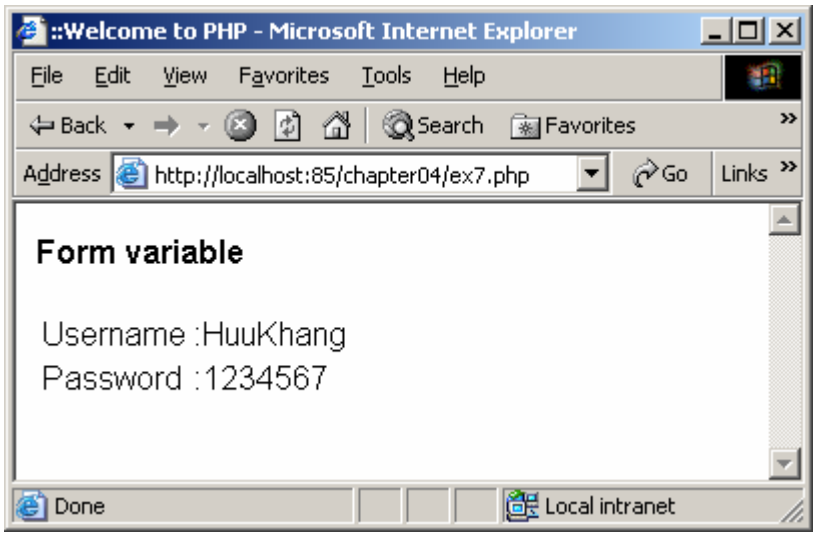
Bằng cách sử dụng $HTTP_POST_VARS để lấy giá trị username và password như ví dụ 4-7.

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();
```

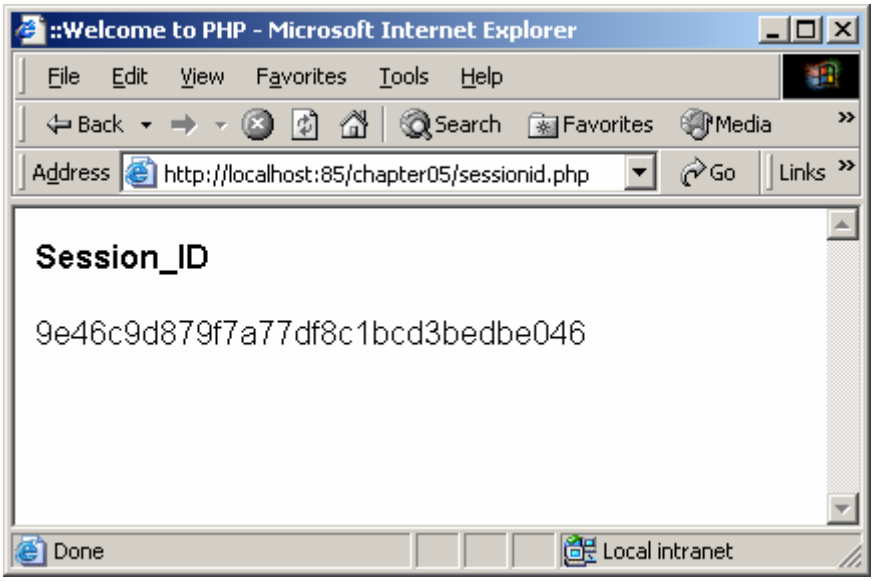
Chẳng hạn, bạn khai báo báo để lấy giá trị session_id trong trang sessionid.php như ví dụ 5-1.

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;
```

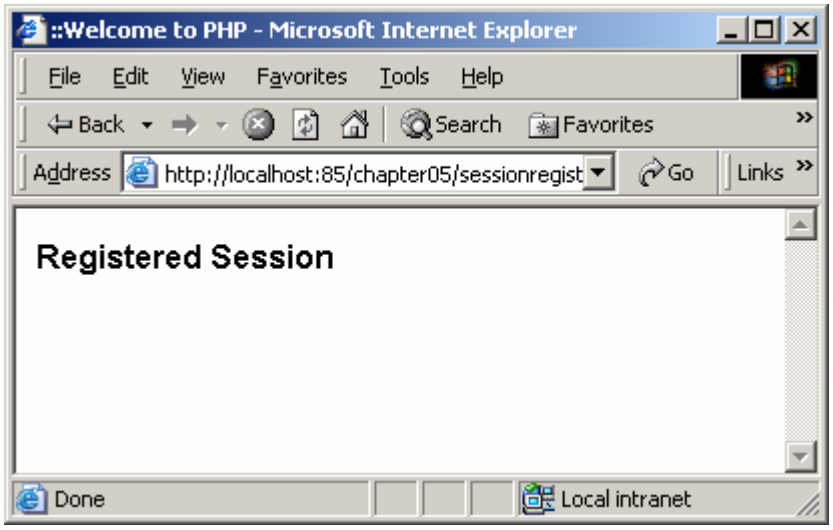
```
$sessioname3=value3;
session_register("sessioname1","sessioname2","sessioname3");
```

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

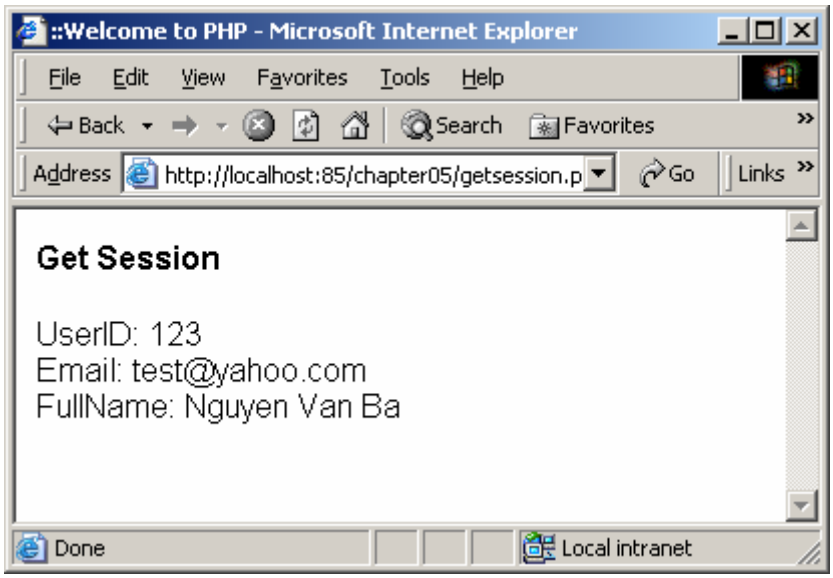
1.3. Lấy giá trị từ session

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

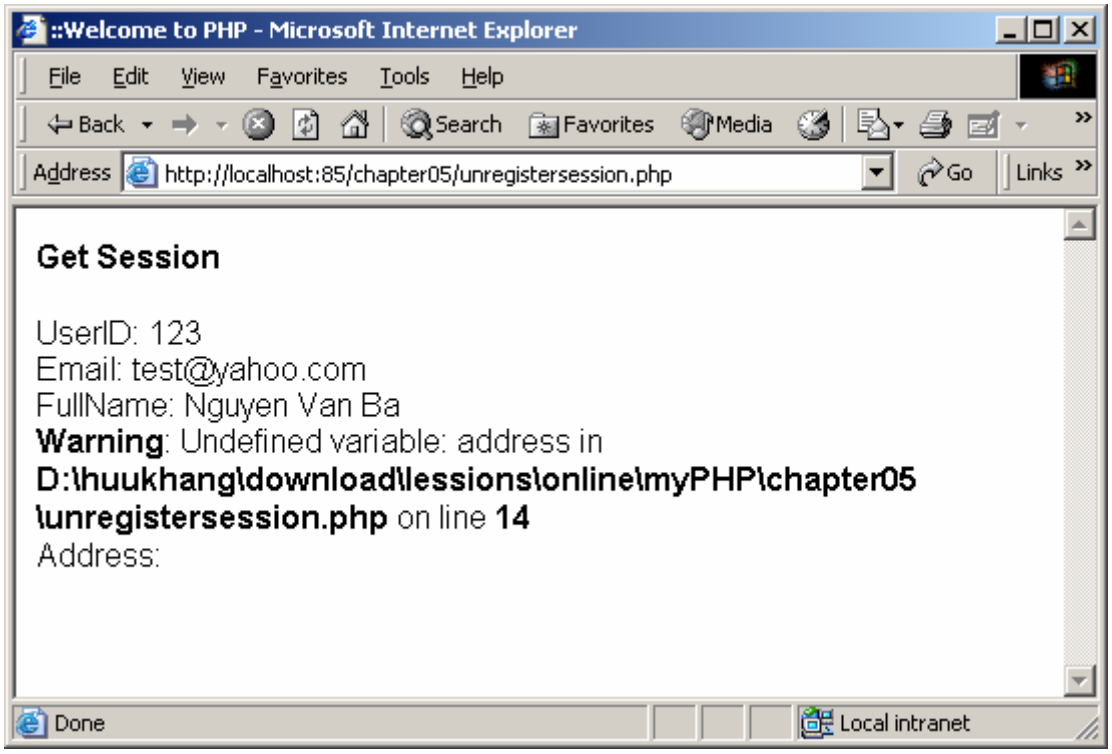
Tuy nhiên, trong trường hợp bạn truy cập một biến session chưa khởi tạo trước đó thì l64i sẽ phát sinh. Ví dụ trong trường hợp này chúng ta truy cập biến session có tên $address như ví dụ 5-4.

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

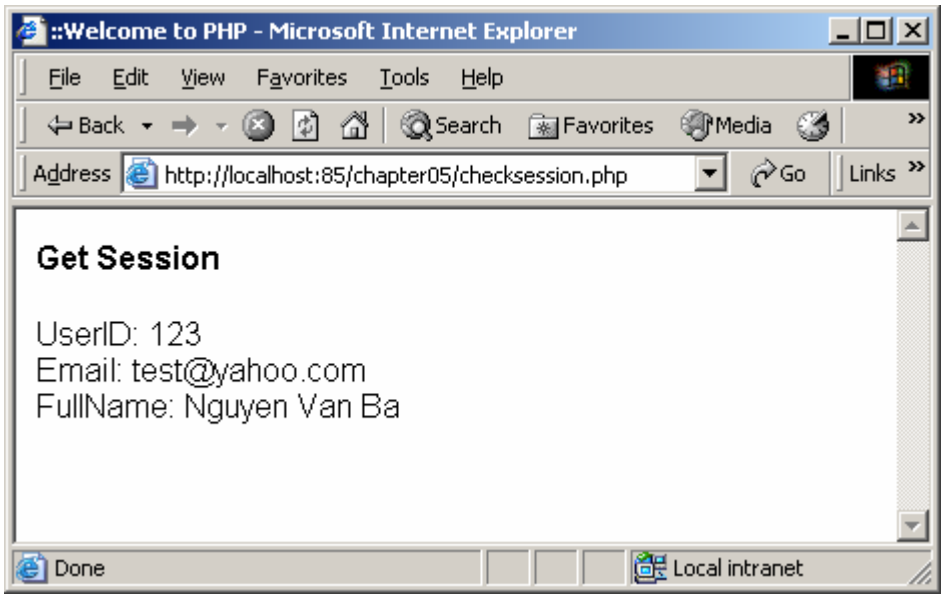
Để kiểm tra session đó có tồn tại hay chưa bạn sử dụng hàm session_is_register trong trang checksession.php. Đối với trường hợp này chúng ta cần kiểm tra 4 session trước khi truy cập đến chúng như ví dụ 5-5.

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

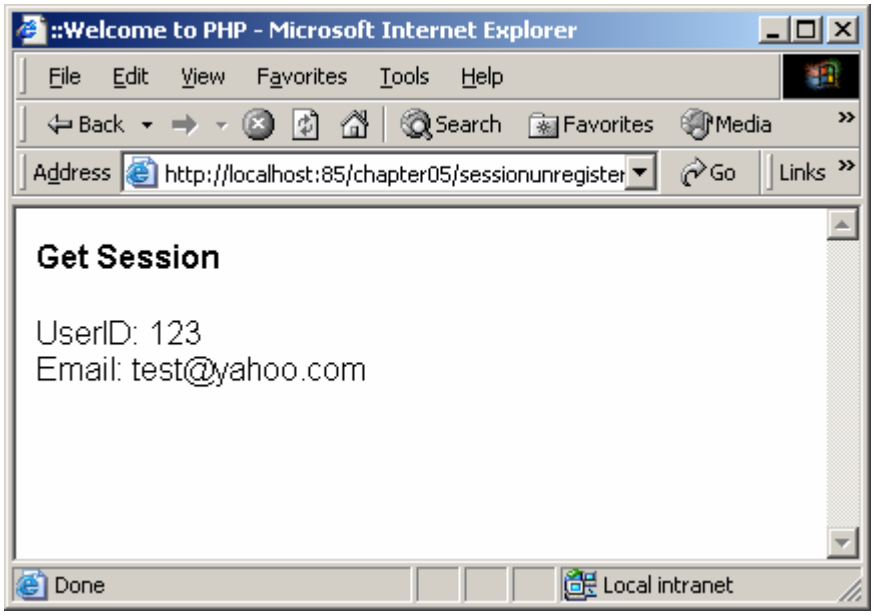
Khi không có nhu cầu sử dụng session nữa thì bạn sử dụng hàm session_unregister để loại session đó. Chẳng hạn, trong trường hợp này chúng ta muốn loại bỏ session có tên là fullname bạn khai báo trong trang sessionunregister.php như ví dụ 5-6.

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

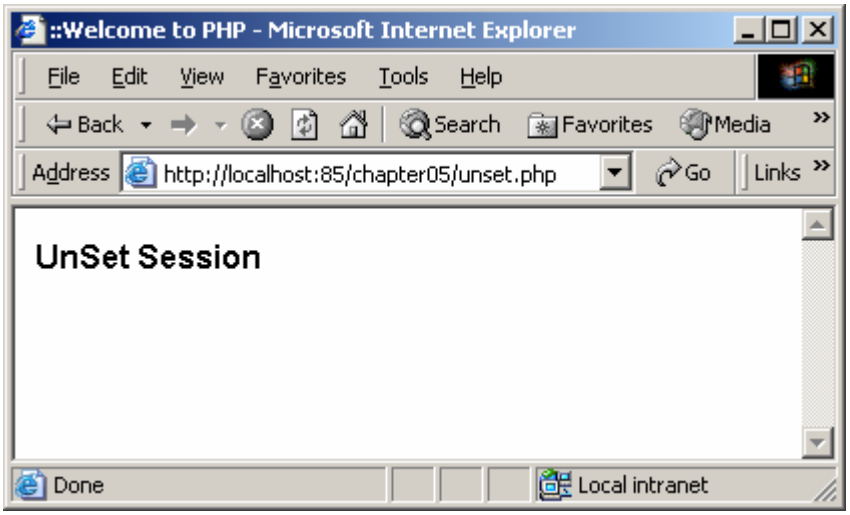
Trong trường hợp loại bỏ tất các session đang tồn tại thì sử dụng hàm session_unset(). Ví dụ dùng hàm này để lạoi bỏ session và dùng hàm sessin_destroy để huỷ tất cả session đó khai báo trong trang unset.php như ví dụ 5-7.

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

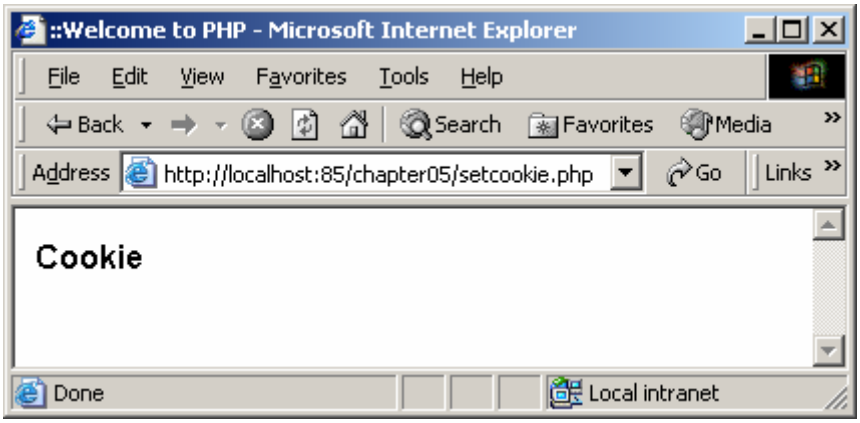
2. COOKIE

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

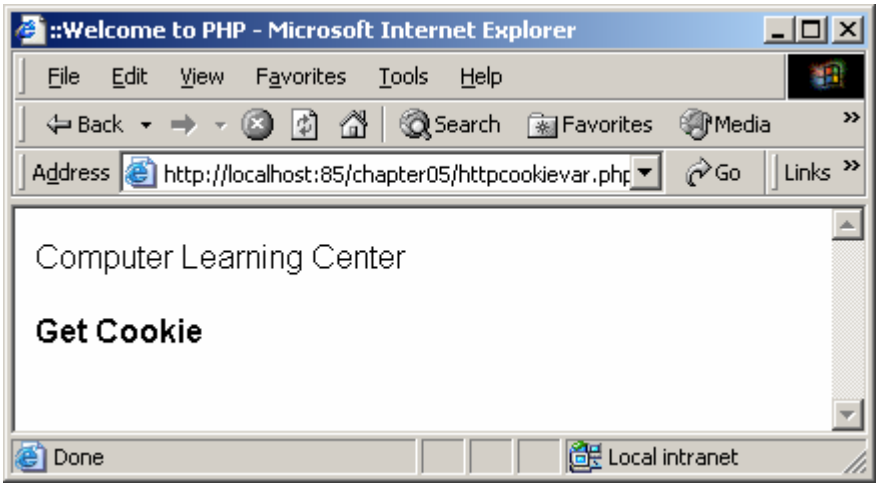
```
<?php
     session_start();
     $myvalue="Online Recruitment";
     session_set_cookie_params($myvalue);
?>
<HTML>
<HEAD>
<TITLE>::Welcome to PHP</TITLE>
</HEAD>
<BODY>
<h4>Session-Cookie</h4>
</BODY>
</HTML>
```

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, ...

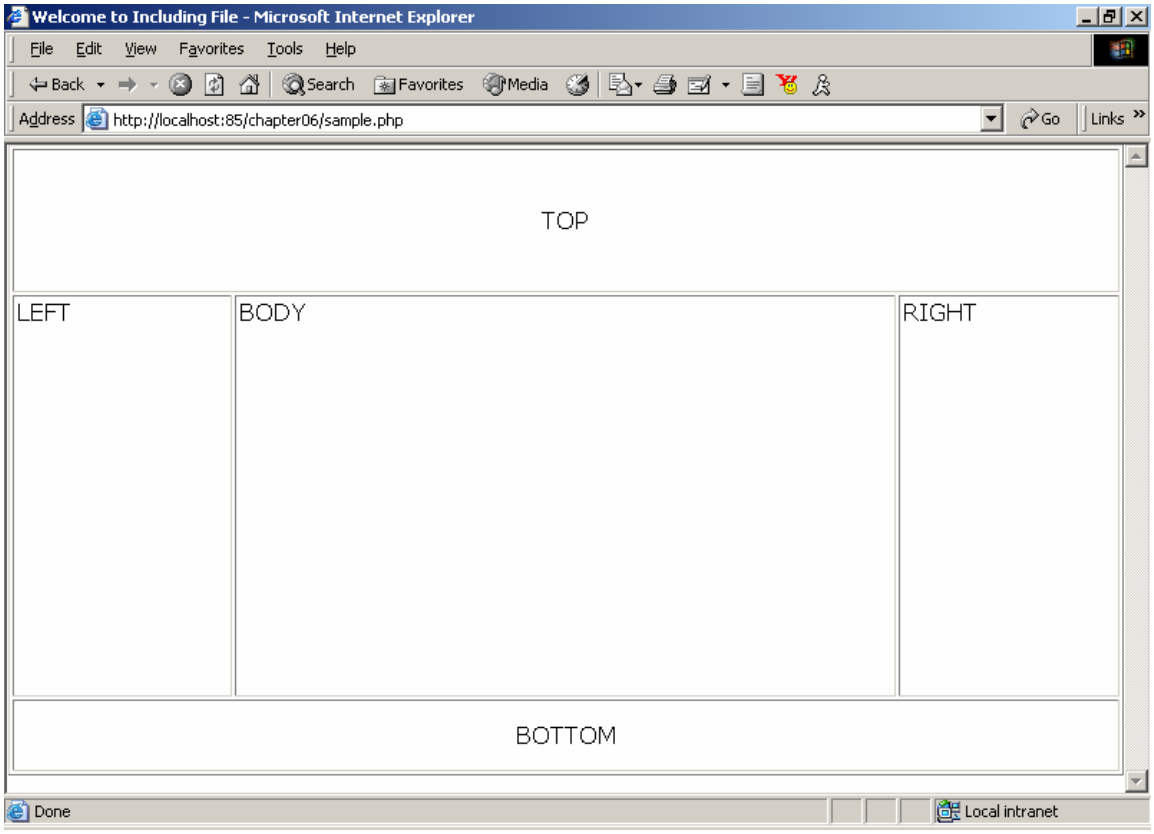
Phần *bottom* thường thông tin liên lạc của công ty, chủ nhân của *web site* và bản quyền. Ngoài ra, phần *bottom* đôi khi là danh sách các *menu* con khác.

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%.*

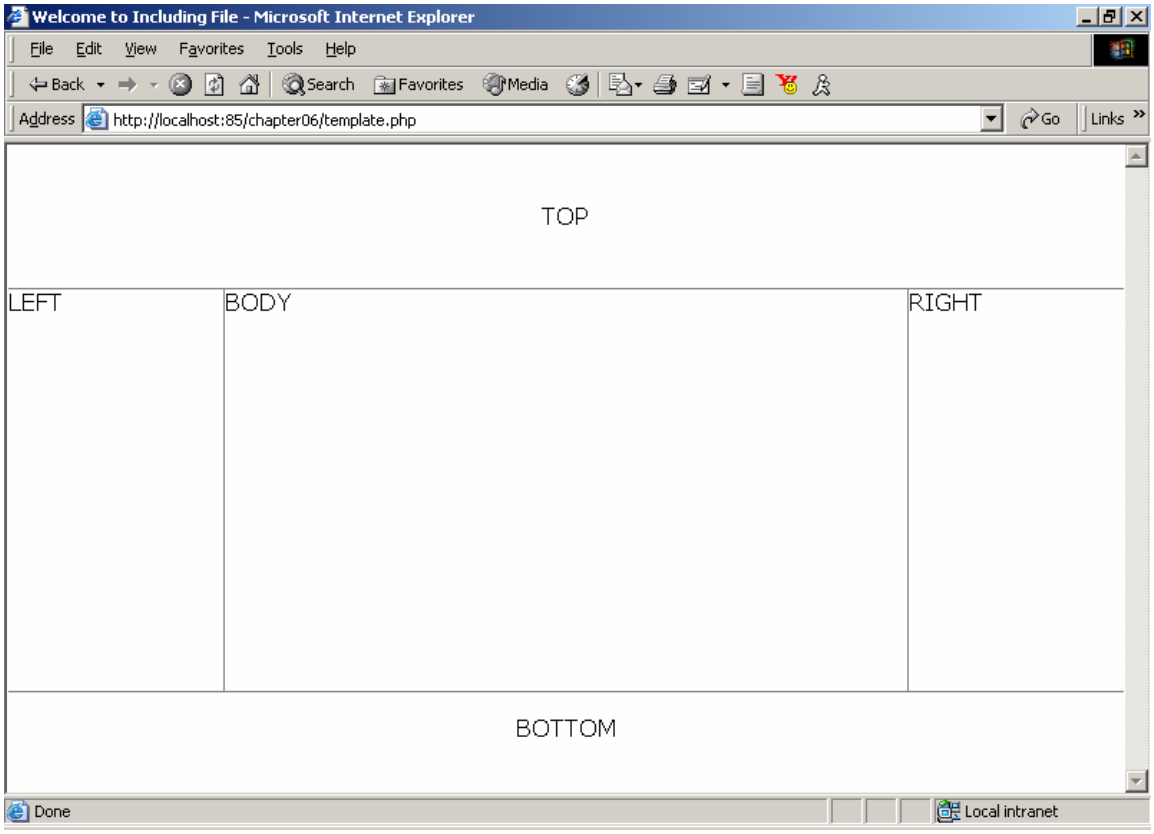
Để có giao diện như trang *sample.php* như trên, bạn có thể khai báo như ví dụ 6-3.

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”);
?>
```

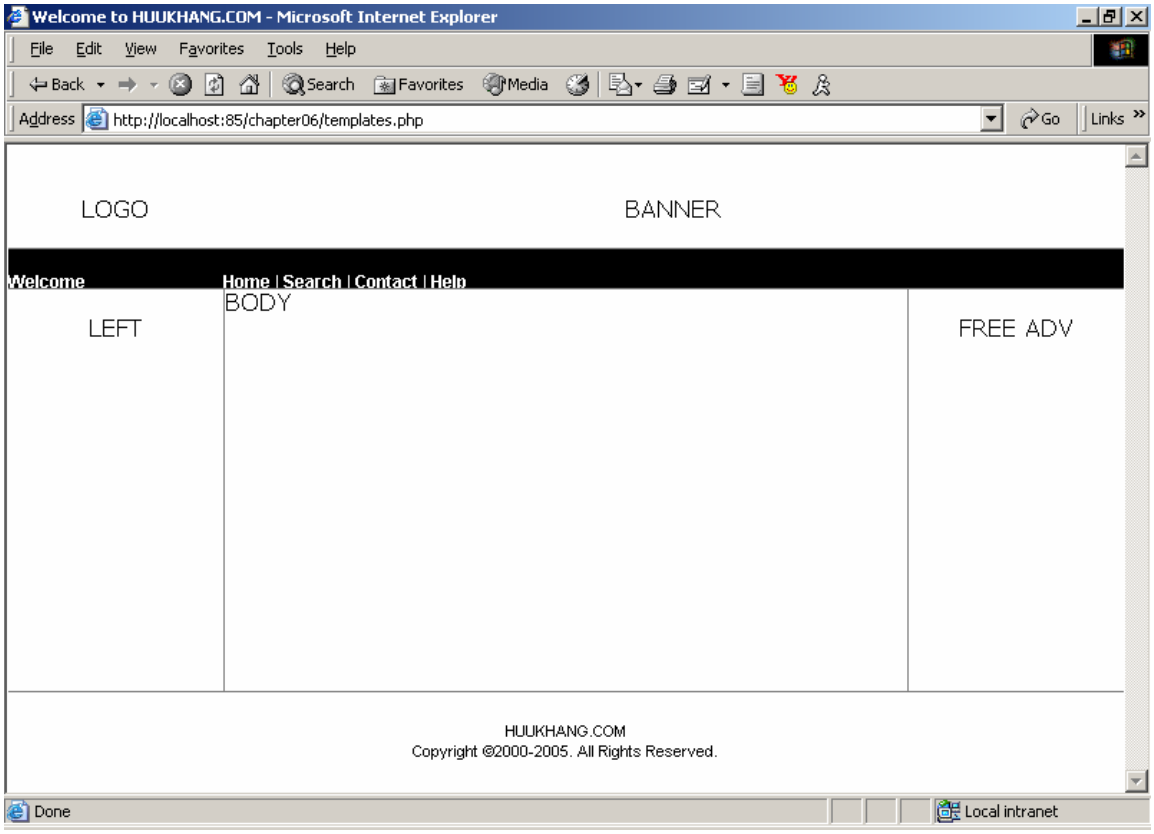
Trong đó trang *templates.PHP* khai báo chèn *top.htm, left.htm, right.htm* và *bottom.htm* như ví dụ 6-5.

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.

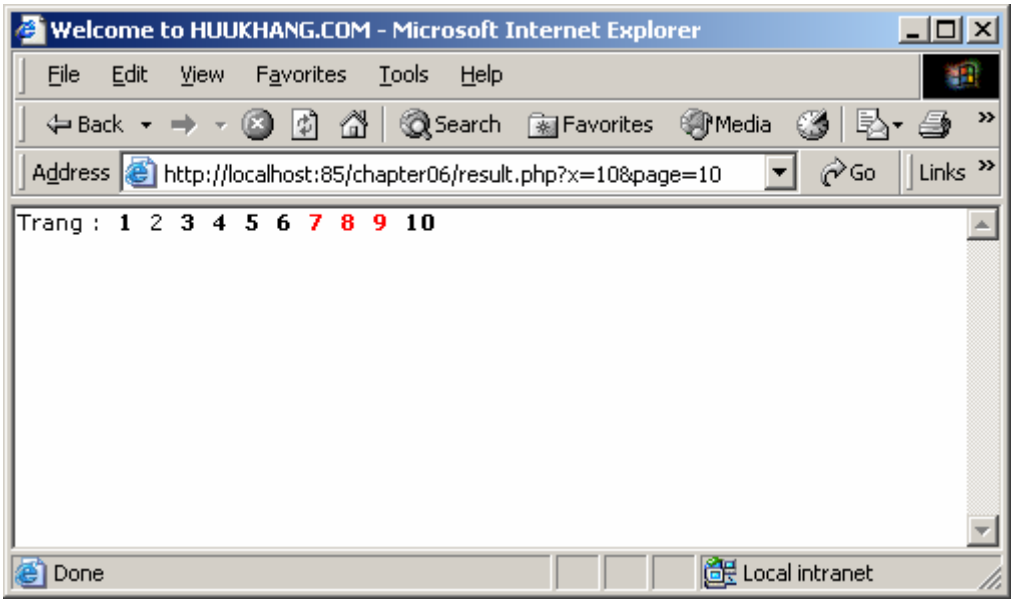
Ví dụ trong trường hợp này chúng ta muốn sử dụng chung hàm có tên getPaging nhận 5 tham số $totalRows (tổng số mẩu tin), $curPg (số trang hiện hành), $pg (số trang trình bày), $re (số mẩu tin trên 1 trang), $file (trang php cần gọi) trong tập tin paging.php.

```
<?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

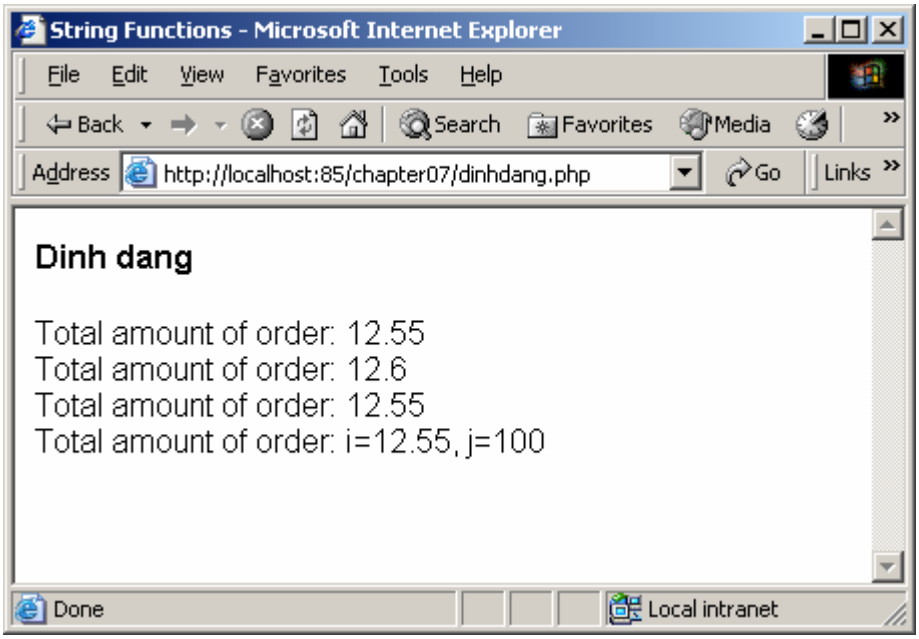
PHP là kịch bản được xem là tốt nhất cho xử lý chuỗi, bằng cách sử dụng các hàm xử lý chuỗi, bạn có thể thực hiện các ý định của mình khi tương tác cơ sở dữ liệu, tập tin hay dữ liệu khác.

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.
```

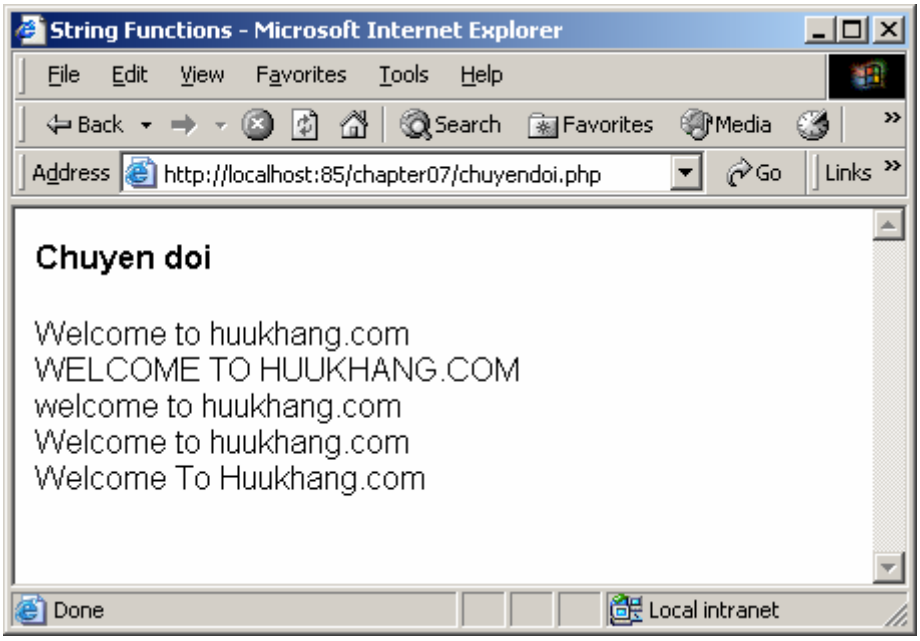
1.2. Hàm chuyển đổi chuỗi

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;
```

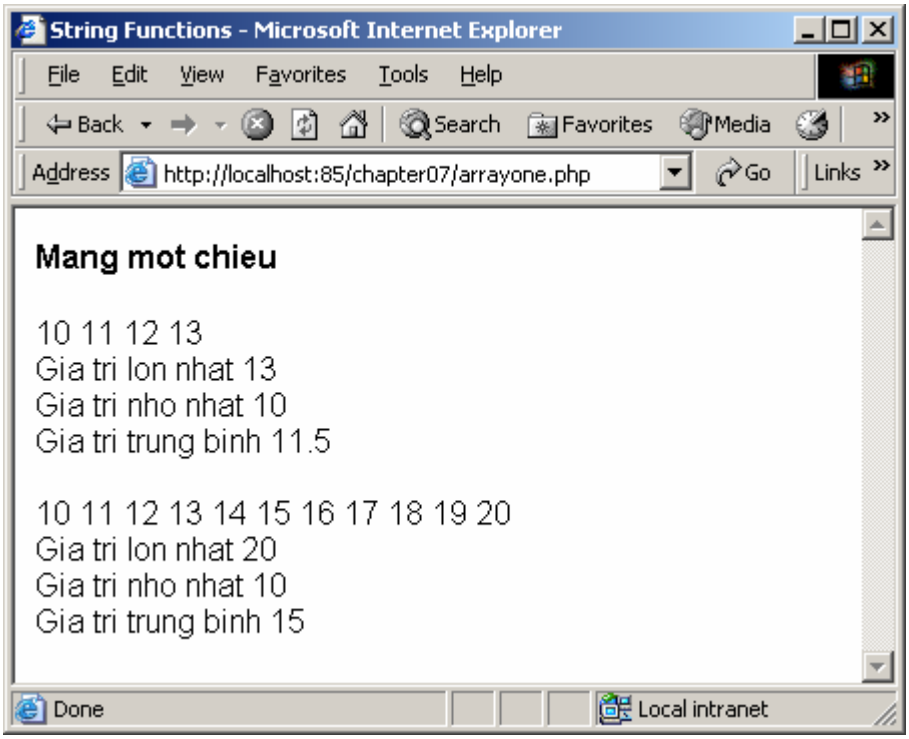
Lấy giá trị của phần tử mảng, bạn cũng thực hiện tương tự như trường hợp truy cập mảng phần tử.

```
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

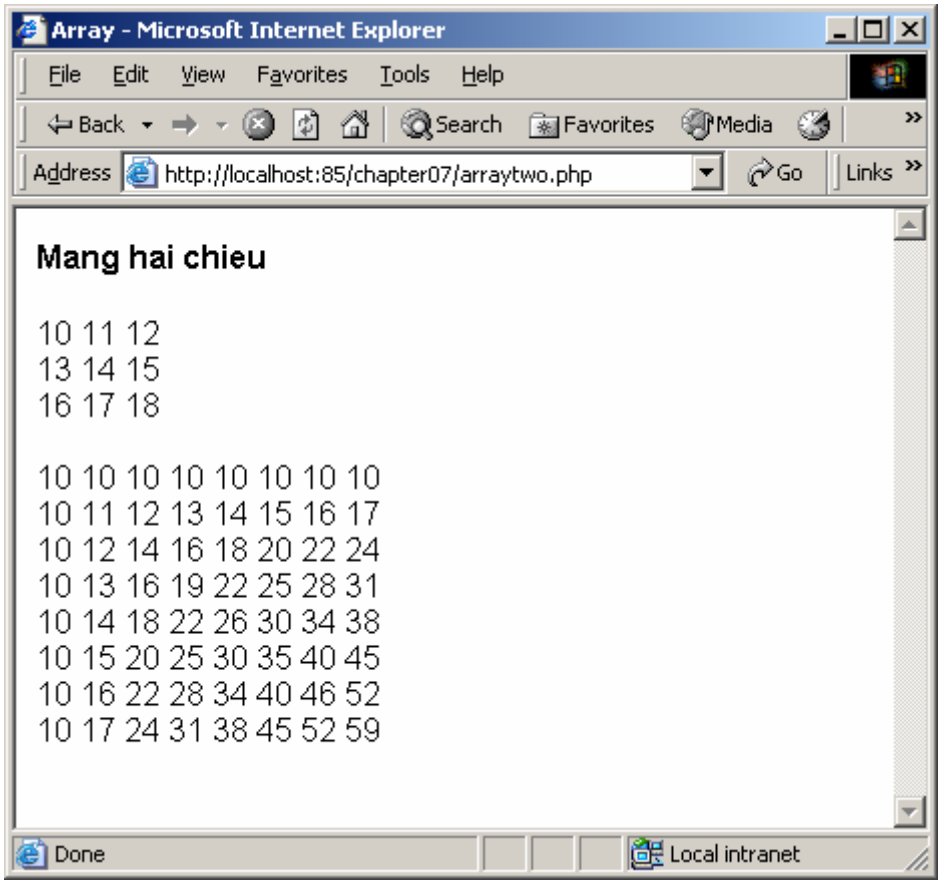
2.2. Mảng hai chiều

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.



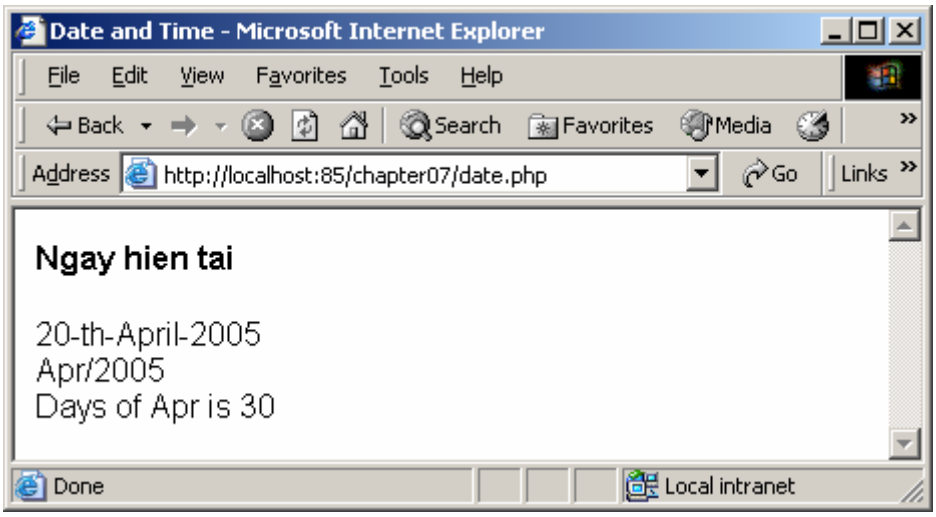
Hình 7-8: Mảng hai chiều

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
```

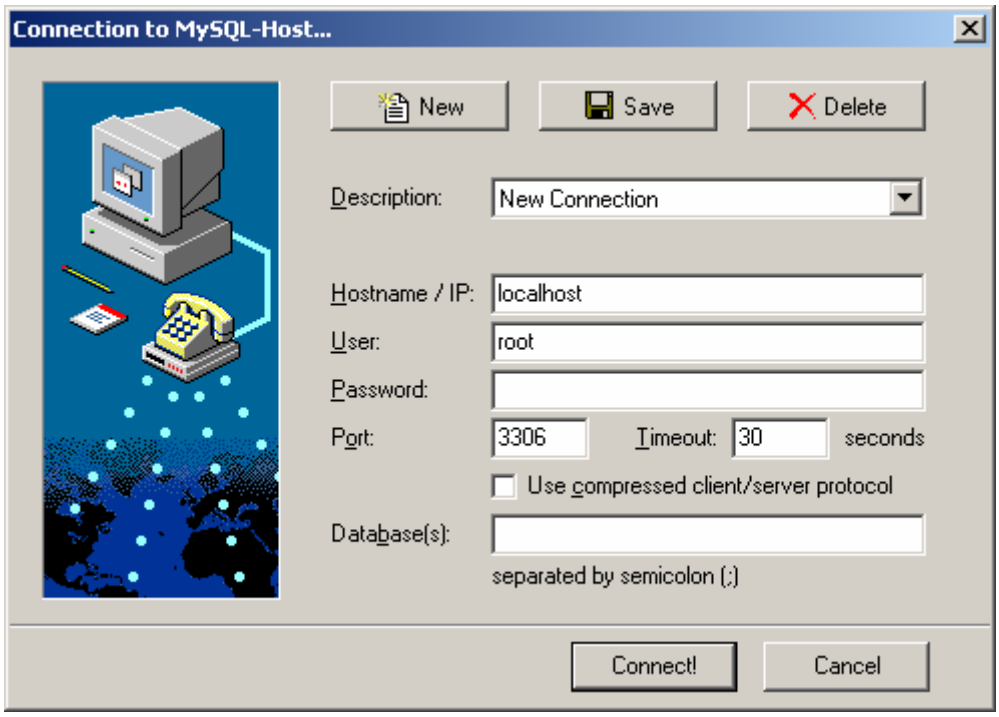
3. TAO CƠ SỞ DỮ LIỆU VÀ NGƯỜI DÙNG

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).

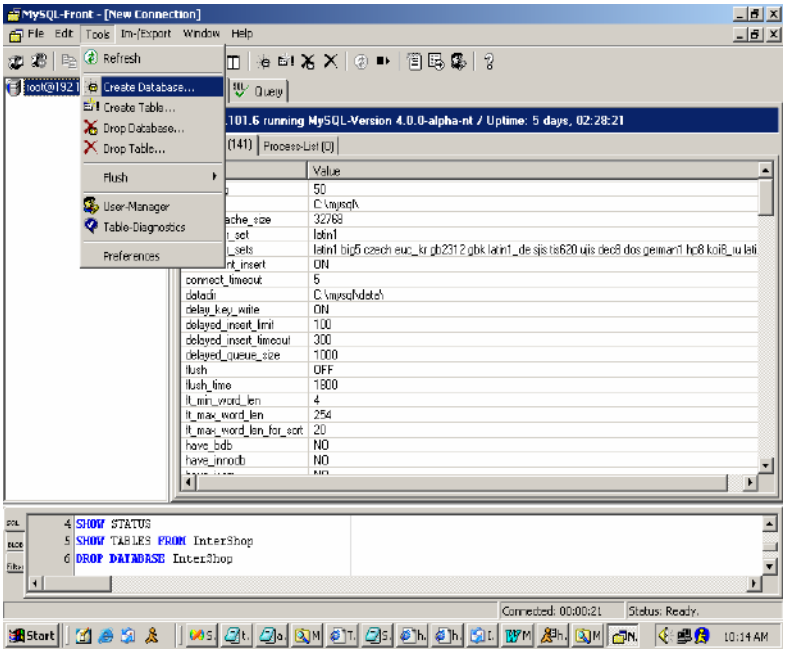
Trong trường hợp lần đầu tiên, bạn không cần cung cấp tên cơ sở dữ liệu, bạn có thể tạo chúng sau khi kết nối.



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

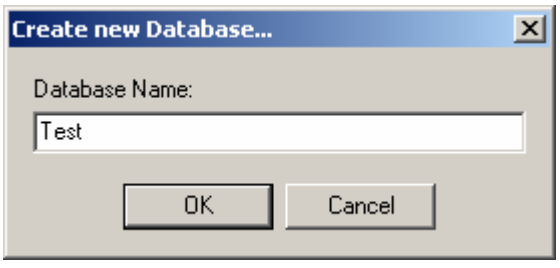
Sau kết nối cơ sở dữ liệu thành công, trình điều khiển cơ sở dữ liệu *MySQL* có giao diện như hình 8-2, cộng việc đầu tiên bạn phải thực hiện là tạo cơ sở dữ liệu.

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.



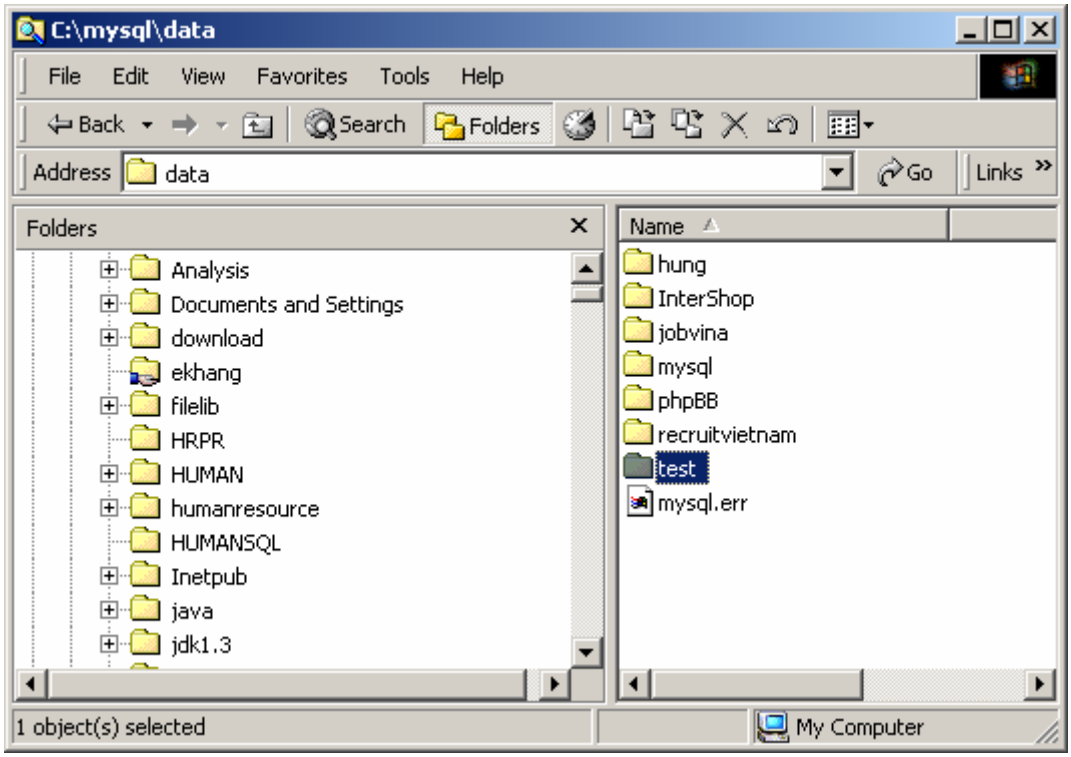
Hình 8-2: Giao diện điều khiển cơ sở dữ liệu MySQL

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'
```

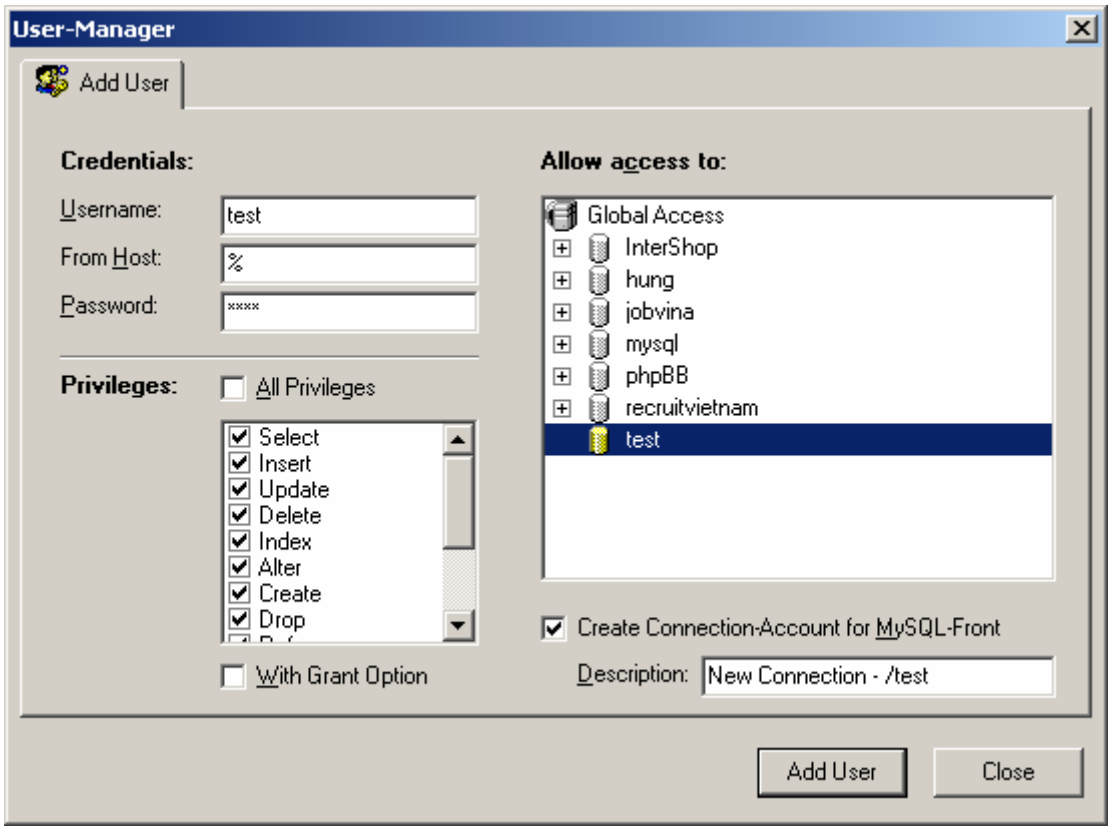
Trong phát biểu trên, vừa tạo ra *User* có tên *myis*, với *hostname* là cơ sở dữ liệu hiện hành, *password* là *1234* và được các đặt quyền *Select, Insert, Update, Delete, Index, Alter, Create*, *Drop* trên cơ sở dữ liệu hiện hành.

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
```

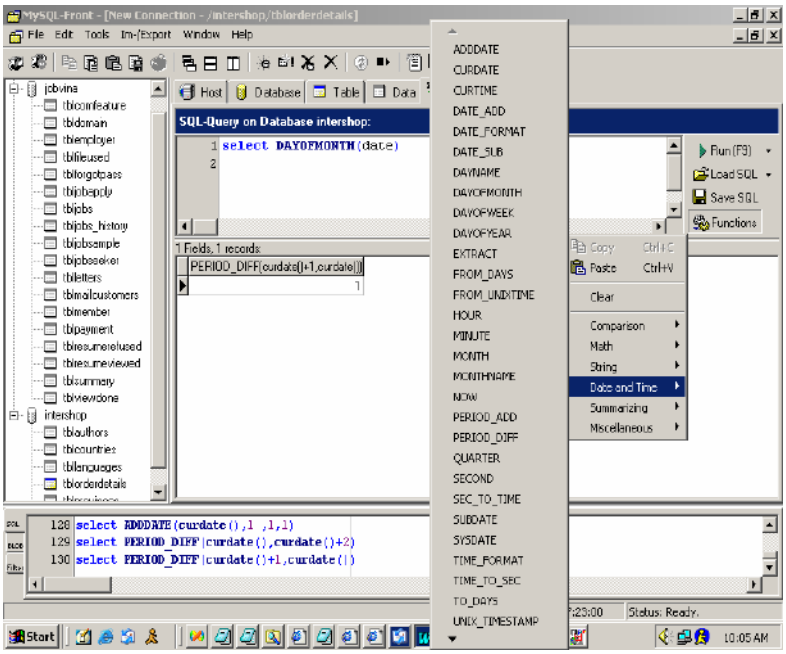
- Hàm *dayofmonth*: Hàm *dayofmonth* trả về ngày thứ mấy trong tháng:

```
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.

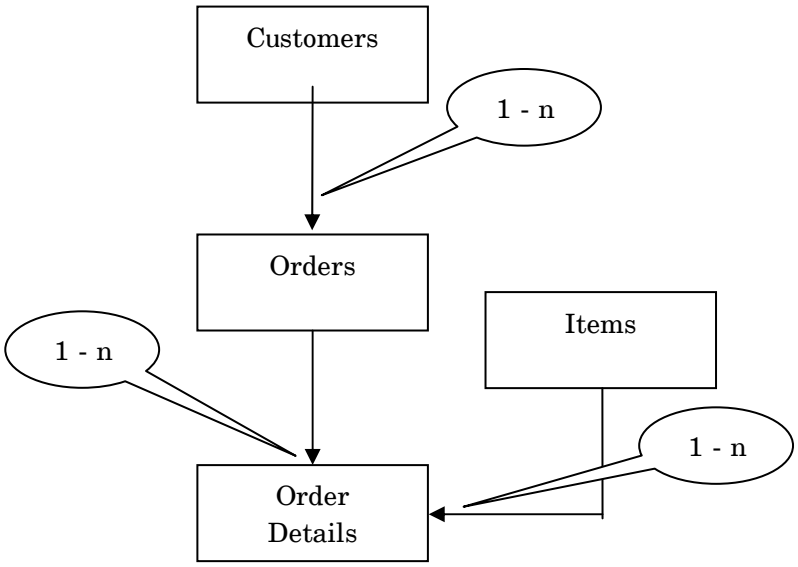
Một khi ứng dụng chạy chậm đi do cơ sở dữ liệu không tối ưu thì rất có thể bạn phải thiết kế và xây dựng lại từ đầu toàn bộ cấu trúc của chương trình và cơ sở dữ liệu.

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.

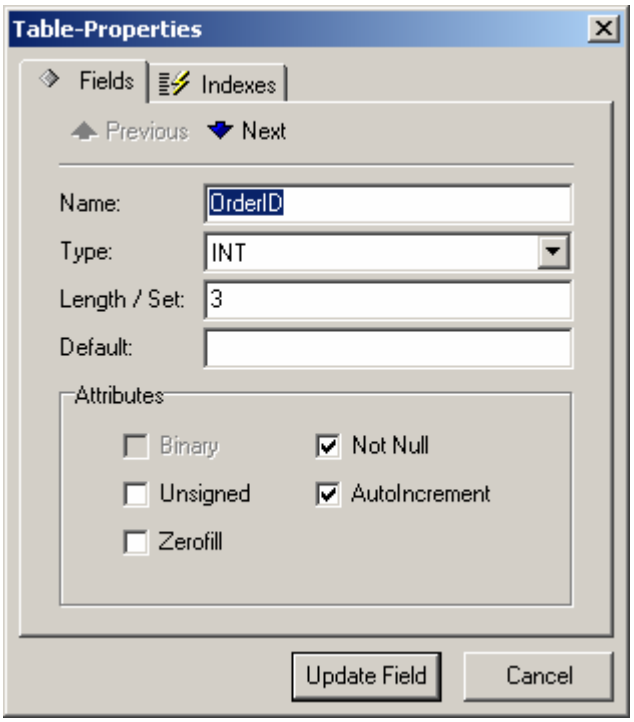
Trong trường hợp, bạn khai báo số tự động trong *SQL Server*, bạn cần phải khai báo thêm các thông số như *seed*. *Seed* là giá trị khởi đầu khi *SQL Server* tự động tăng giá trị, *Increament* là bước tăng, nó cho biết mỗi lần tăng cần bao nhiêu giá trị.

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

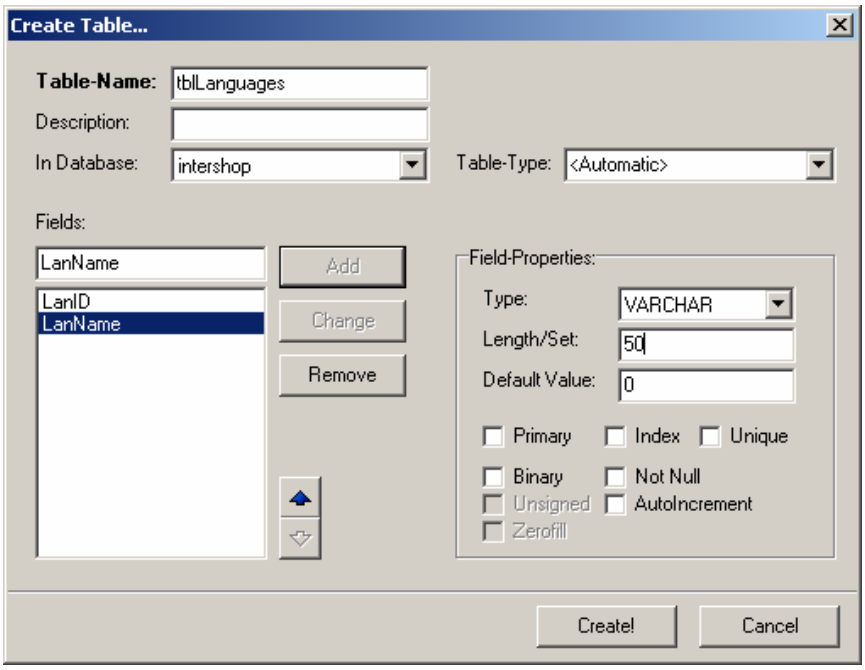
```
ALTER TABLE tblorders
CHANGE OrderDate
OrderDate DATETIME
DEFAULT "0000-00-00"
```

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:*

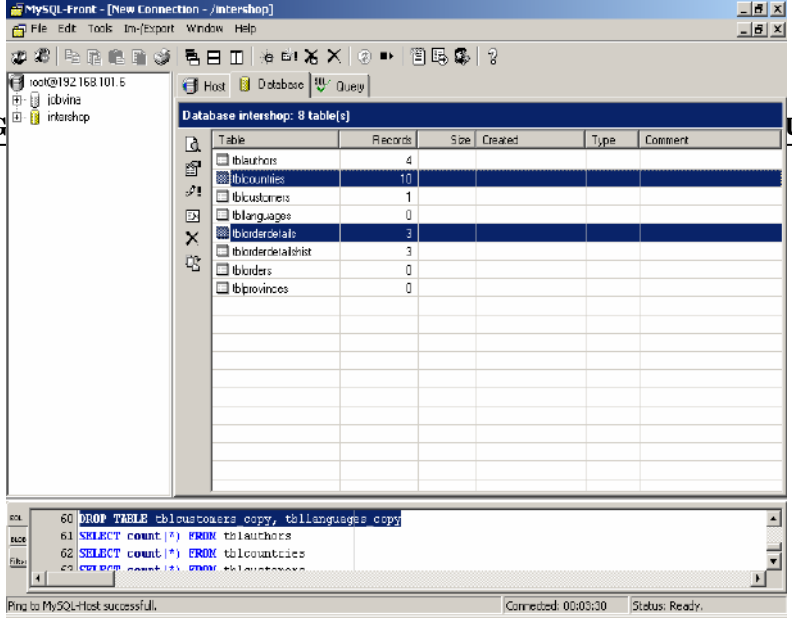
```
DROP <Object type> <Object name> [, .... n]
```

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

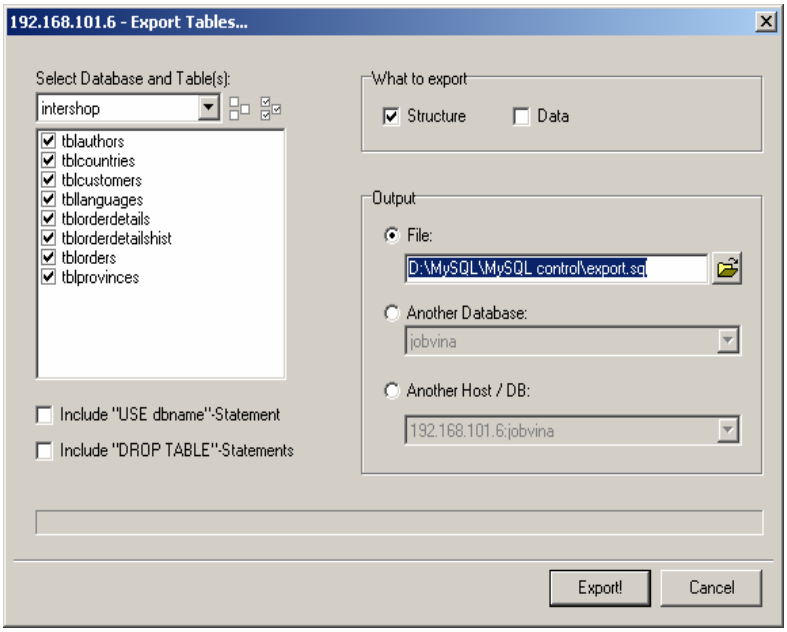
Thông thường khi xây dựng cơ sở dữ liệu để phát triển ứng dụng, đôi khi bạn cần chuyển cơ sở dữ liệu từ máy này sang máy khác, hay từ khu vực này hay đến khu vực khác.

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

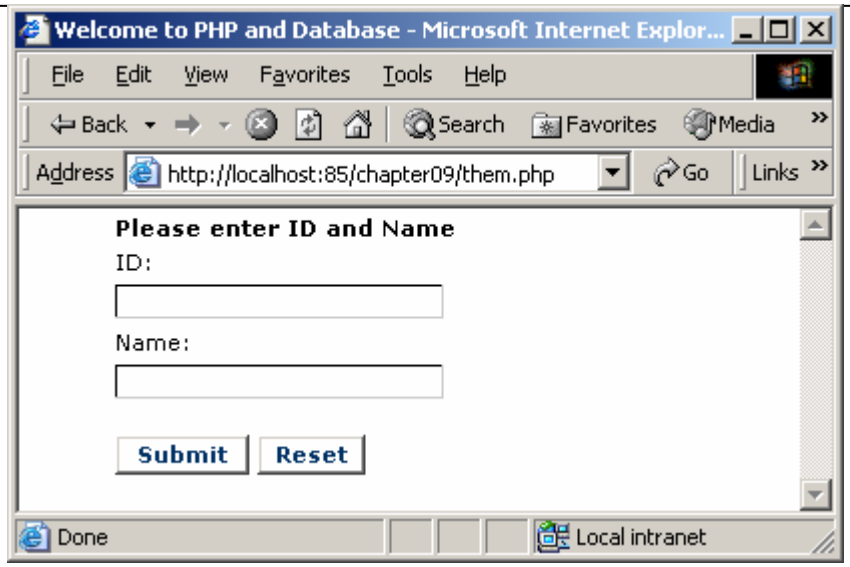
Để thêm mẩu tin, bạn sử dụng hàm mysql_query(chuỗi Insert). Chẳng hạn, chúng ta khai báo trang insert.php để thêm mẩu tin vào bảng tblships có hai cột dữ liệu là ShipID và ShipName như ví dụ trong trang insert.php.

```
<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)
```

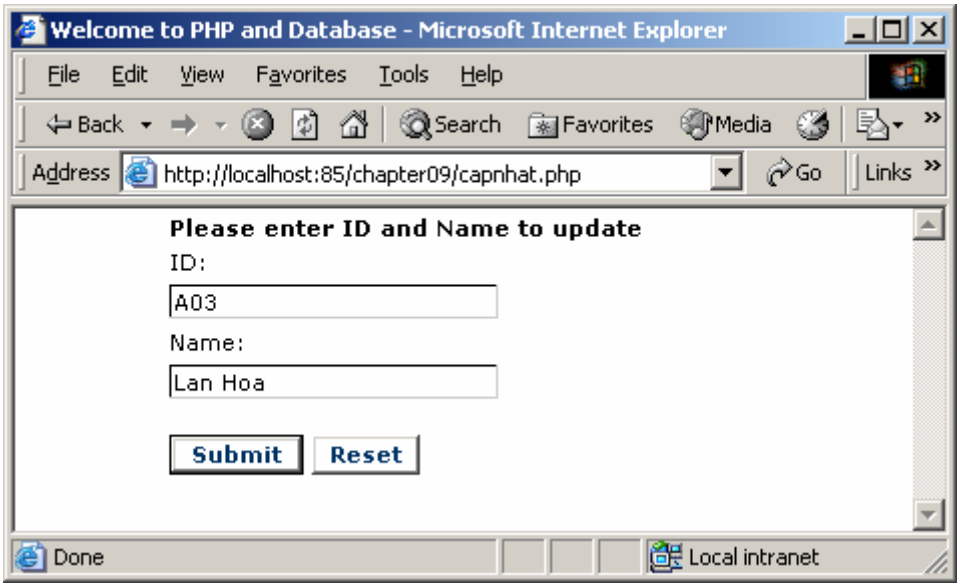
```
        $affectrow=mysql_affected_rows();
      mysql_close($link);
?>
So mau tin cap nhat <?= $affectrow?>
</BODY>
</HTML>
```

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);
?>
```

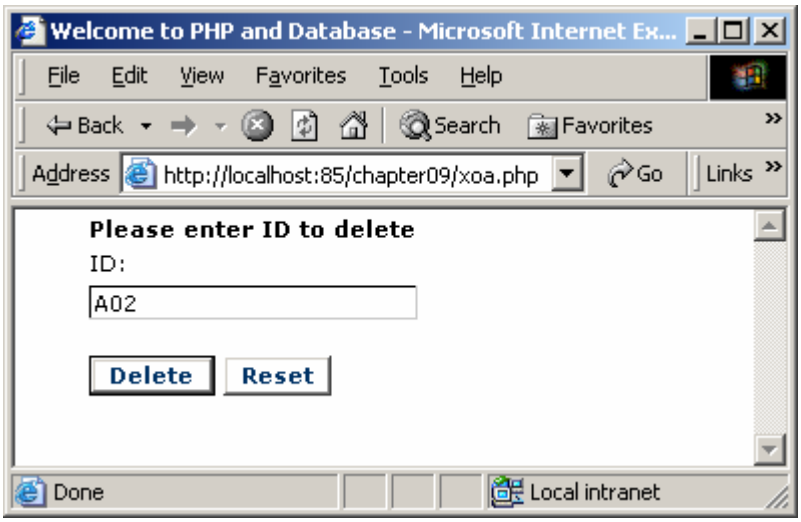
```
So mau tin cap nhat <?= $affectrow?>
</BODY>
</HTML>
```

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

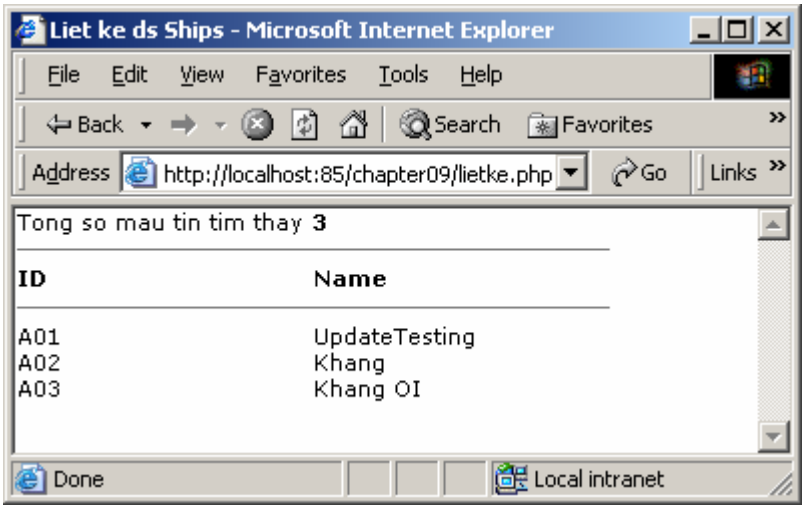
Sau khi nhập mã cần xoá, nếu người sử dụng nhấn nút Delete lập tức trang dodelete.php sẽ triệu gọi và xoá mẩu tin tương ứng.

```
<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

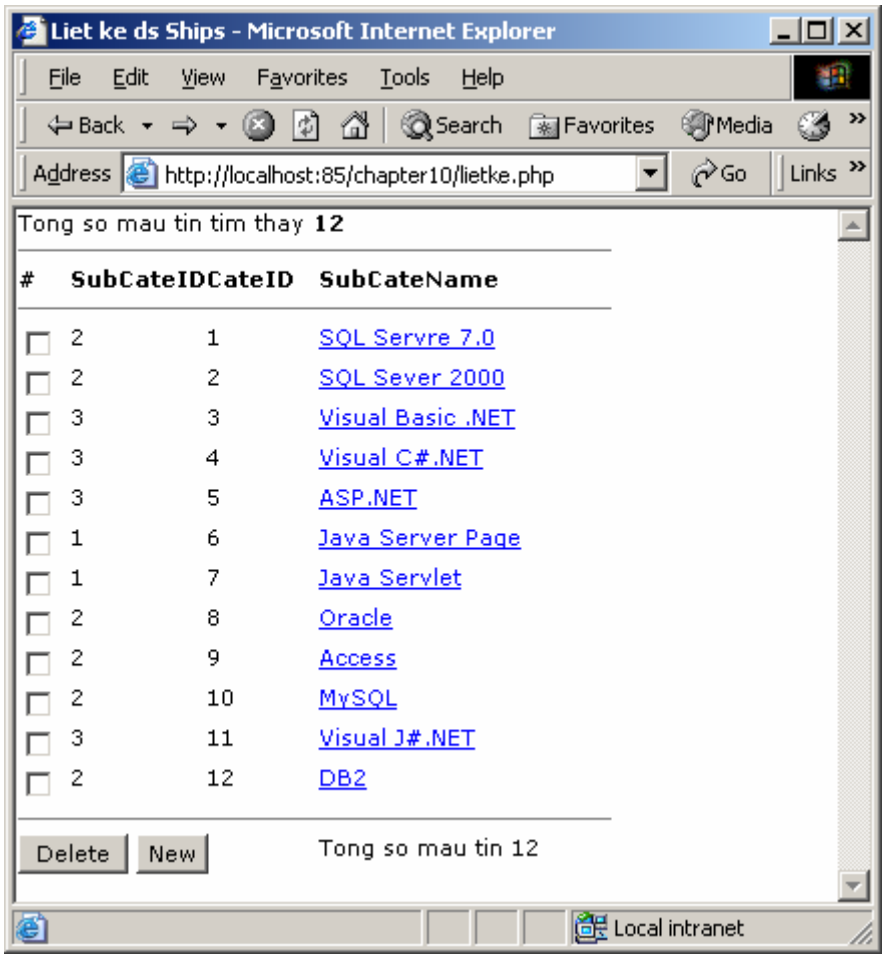
Trong bài trước chúng ta đã làm quen với cách xoá mẩu tin trong cơ sở dữ liệu mySQL. Đối với trường hợp xoá một lúc nhiều mẩu tin, chúng ta phải xây dựng trang PHP có sử dụng thẻ input dạng checkbox.

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="">
```

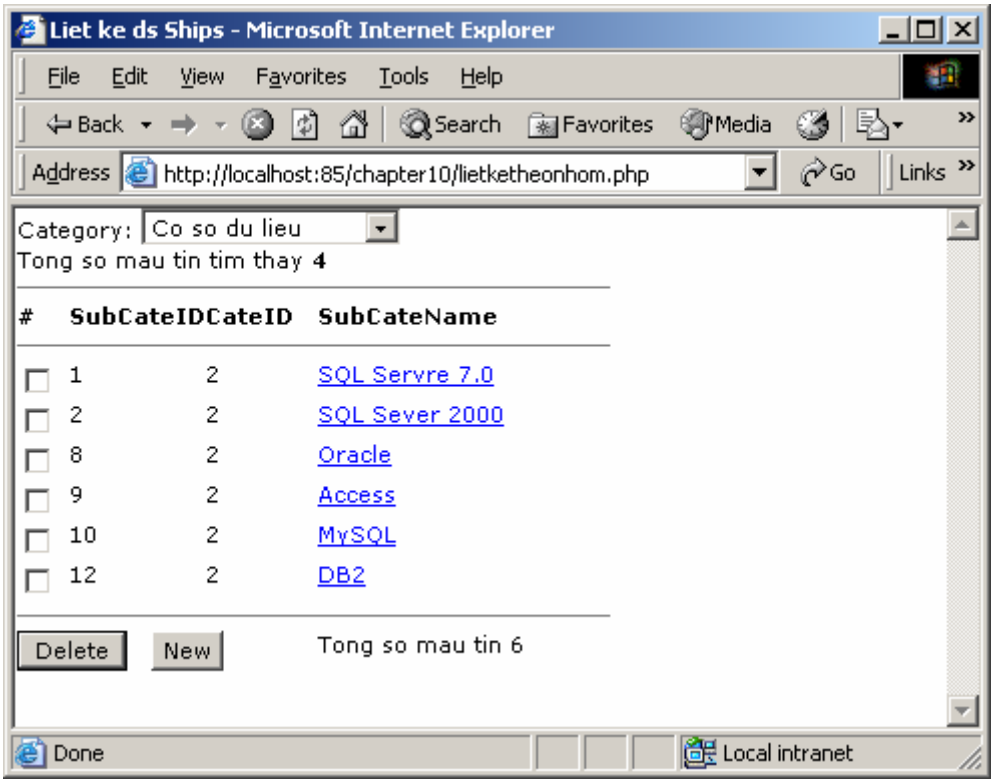
Cho biết bạn submit từ trang nào và loại xoá nhiều mẩu tin hay một mẩu tin đối với bảng tương ứng. Mục đích của vấn đề này là trang delete sử dụng chung cho nhiều bảng khác nhau và từ trang liệt kê (xoá nhiều) hoặc từ trang edit (1 mẩu tin cụ thể).

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>
```

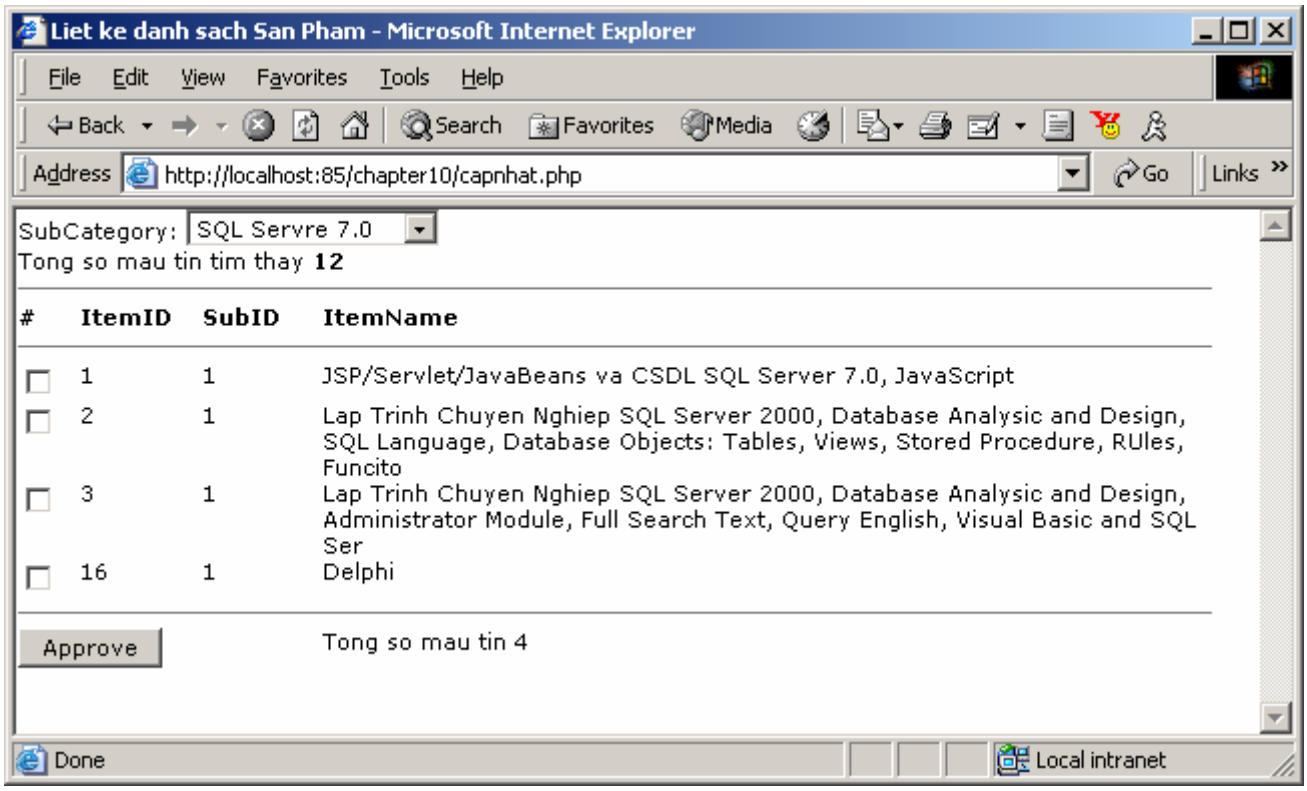
Sau đó gọi trong biến cố onsubmit của form như sau:

```
<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.