# LẬP TRÌNH WEB ĐỘNG VỚI

# PHP / MySQL

- GUESTBOOK
- CATALOG
- FORUM
- SHOPPING CART

## PHẦN 3

Tống Phước Khải (tổng hợp & biên dịch)

# BIẾN (variables)

## và các phép xử lý trên biến

# PHP

1- Biến và cách sử dụng Biến

2- Xử lý dữ liệu từ FORM

3- Tìm hiểu sâu hơn về Biến

Bạn đọc thân mến,

Vừa qua tôi đã soạn xong phần 1 và phần 2 của giáo trình tự học PHP/MySQL. Tôi đã nhận được email của những bạn quan tâm, chờ đợi phần 3 của giáo trình này. Đáng lẽ phần 3 đã cho ra đời sớm nhưng vì bận rộn quá nhiều công việc (hiện tôi đang phụ trách và có rất nhiều công việc trong nhóm Hanosoft - software Hán Nôm) nên việc biên soạn sách tự học này ít nhiều bị trì hoãn.

Do hoàn cảnh trên, chắc chắn công việc biên soạn này không tránh được những sai sót. Nhưng dù sao đi nữa, biết được các bạn có thể áp dụng giáo trình này vào thực tế thì tôi phần nào cũng lấy đó làm sự khích lệ cho riêng mình.

Đúng lý ra phần 3 này là dành trọn cho việc nói về ngôn ngữ SQL nhưng phần 2 đã bàn về CSDL rồi, nếu phần này nếu cũng bàn về nó thì có vẻ hơi nhàm phải không các bạn? Với ý nghĩ này tôi đã dành trọn phần 3 để nói về biến trong PHP.

Hẳn nhiên tôi biết một số bạn mong mỏi những gì trong đây có thể áp dụng liền thì đỡ chán hơn. Nhưng theo tôi nghĩ trước tiên hết bạn cần phải nắm rõ mọi ngóc ngách của PHP và MySQL thì mới có thể thiết kế được những chương trình đạt tiêu chuẩn. Do vậy mong các bạn hãy kiên nhẫn khi đọc những chương hướng dẫn suông như thế này! Đừng nản lòng và nên ghi nhớ đây là cội rễ cho các ứng dụng thực tế của các bạn.

Để bắt đầu chương này ít nhất bạn cũng phải có chút đỉnh kiến thức về **Cơ Sở Lập Trình**. Tôi nghĩ nếu bạn đã học qua một khoá lập trình căn bản thì bạn có thể hiểu được. Nếu không, đòi hỏi bạn cần phải động não hoặc tìm tòi hơi nhiều. Nào, chúng ta bắt đầu đi thôi!

PHP xử lý các biến rất linh động. Nó có thể nhận biết được kiểu của biến và làm cho cú pháp câu lệnh đơn giản hơn. Ai đã từng lập trình với C, Java hoặc Perl sẽ cảm thấy rất dễ dàng khi sử dụng PHP. Tuy nhiên việc dễ dãi này cũng gây ra một số trở ngại nhất định.

**Tất cả những biến khai báo trong PHP đều được bắt đầu với dấu đô la** ($). Dù cho biến của bạn kiểu chuỗi, nguyên hay thập phân hoặc thậm chí là mảng thì chúng không có gì khác biệt nhau. PHP chỉ theo dõi dữ liệu chứa trong biến thay đổi như thế nào thôi.

Nói chung, khi làm việc với PHP bạn sẽ quan tâm đến 3 vị trí khác nhau của biến đó là: (1) *khai báo ngay trong mã lệnh PHP*, (2) *chuyển tiếp từ một trang HTML* hoặc (3) là *biến sẵn có trong của hệ thống PHP*.

**Chúng ta sẽ tìm hiểu về từng loại trên ở phần tiếp theo. Nên lưu ý rằng biến cũng có thể được chuyển tiếp từ các nơi khác như từ các URL hoặc từ các SESSION.**

# Gán biến trong một Script

Bạn không cần phải khai báo EXPLICIT cho biến như trong một số ngôn ngữ khác. Chỉ cần khai báo tên biến là nó sẽ sẵn sàng làm việc. Bạn hãy xét các ví dụ sau để hiểu cách khai báo biến trong PHP **uyển chuyển** như thế nào:

```
$a = "Toi thich hoc PHP"; //day la bien chuoi
$b = 4; //day la bien so
$c = 4.837; //day la bien so thuc
$d = "2"; //day cung la bien chuoi
```

Để ý rằng dấu = là dấu dùng để gán. Còn khi thực hiện phép so sánh bằng thì bạn dùng hai dấu bằng (= = ). Ví dụ: IF ($x= =1)

PHP rất thông minh trong việc biến đổi kiểu. Ví dụ, bạn thực hiện phép cộng một số nguyên với một chuỗi chứa ký tự số (trong ví dụ trên là $b và $d).


```
$a = "Toi thich hoc PHP"; //day la bien chuoi
$b = 4; //day la bien so
$c = 4.837; //day la bien so thuc
$d = "2"; //day cung la bien chuoi
```

```
$e = $b + $d;
echo $e;
```

PHP sẽ nhận ra rằng bạn muốn xem chuỗi trong $d (chuỗi "2") như là một số nguyên. Thế là nó sẽ hoán chuyển sang trị nguyên và thực hiện phép toán cộng cho ra kết quả là $e = 6. Ngoài ra, PHP còn có thể hiểu được các chuỗi vừa số vừa chữ như ví dụ sau:

```
$a = 2;
$b = "2 con heo con";
$c = $a + $b;
```

Kết quả cho ra là $c = 4. Nếu một số nguyên hay thập phân đứng ở vị trí đầu một chuỗi thì PHP có thể hiểu được như ví dụ trên. Tương tự, PHP thực hiện tương tự đối với các kiểu số khác nhau:

```
$f = 2; //$f la mot so nguyen
$g = 1.444; // $g la mot so thuc
$f = $f + $g; //$f tu bien doi thanh so thuc
```

Việc xử lý này thật là hay, nhưng nó có thể dẫn đến một số rắc rối đó là sẽ có những lúc bạn không biết ở tại thời điểm nào bạn sẽ làm việc với kiểu của biến là kiểu gì. Tôi sẽ trình bày trong phần Kiểm Tra Biến.

# Qui định về chuỗi

Trong các ví dụ trên, bạn thấy tất cả các chuỗi đều được bao trong dấu nháy đôi. Có hai cách khác để bạn thể hiện một cho PHP hiểu đó là chuỗi.
Trong một chuỗi mà bạn đã bao lại bằng cặp nháy đôi "...", xong bạn chèn một biến vào giữa, thì PHP vẫn hiểu được biến đó. Ví dụ:

```
$my_name = "Jay";
$phrase = "Hello, my name is, $my_name";
echo $phrase;
```

Kết quả cho ra là: Hello, my name is, Jay. Thật khác thường phải không các bạn?! (Đáng lẽ ra dấu nháy " thứ hai phải sau chữ *is* rồi đặt một dấu cộng chuỗi với biến *$my_name*)

Trong trường hợp sau đây, tôi muốn xuất ra một chuỗi: *Tôi đăng ký hosting hết $20* thì phải làm sao? Bởi vì trong chuỗi này có chứa $, điều này sẽ làm cho PHP hiểu đó là một biến mới. Chúng ta xem cách giải quyết như sau:
Nếu như trong chuỗi bạn muốn có chứa các ký tự đặc biệt như: dấu nháy đôi "", dấu slash \, dấu đô la $ , bạn phải sử dụng đến ký tự chuyển (gọi là dấu escape) đó là dấu slash (\).
*Tôi quen đọc dấu / là "dấu suyệt trái" và \ là "dấu suyệt phải".*

Giả sử, để xuất ra màn hình một dòng chữ: *<form action="mypage.php" method="get">*, như bạn thấy trong đó chứa tới 4 dấu nháy đôi - thuộc dạng ký tự đặc biệt. Ta phải sử dụng tới 4 dấu suyệt phải như sau:

```
echo "<form action=\"mypage.php\" method=\"get\">";
```

Thì đến khi chạy chương trình mới mong cho ra kết quả như mong muốn.

***Tác dụng của dấu nháy đơn đối với PHP:***
Bạn sẽ thấy dấu nháy đơn trong PHP có tác dụng hơn dấu nháy đôi như thế nào! Nếu chuỗi của bạn có chứa các biến (bắt đầu bằng $), bạn bao lại bằng dấu nháy đơn thì biến đó sẽ bị biến thành chuỗi luôn, chớ không được hiểu là một biến như cách bao bằng dấu nháy đôi:

```
$my_name = "Jay";
echo 'Hello, my name is, $my_name';
```

Kết quả cho ra là *Hello, my name is, $my_name* chứ không phải Hello, my name is Jay.

Cuối cùng, trong PHP4 bạn có thể sử dụng dấu Here Documents. Đây là một loại ký hiệu tương tợ hai loại nháy đơn và nháy đôi. Trong một số trường hợp khi sử dụng nó bạn sẽ cảm thấy rất tiện lợi. Here Docs xác định giới hạn ở đầu chuỗi với 3 dấu nhỏ hơn <<< và ký hiệu nhận dạng (trong sách này tôi sử dụng ký hiệu nhận dạng EOQ) Chuỗi được kết thúc cũng với ký hiệu nhận dạng như vậy và kèm theo là dấu chấm phẩy (;). Sau đây là ví dụ chuỗi *Toi thich hoc PHP* được gán cho biến $mystring được xác định bằng cách sử dụng Here Doc.

```
$my_string = <<<EOQ
Toi thich hoc PHP.
EOQ;
```

Sử dụng Here Doc, các biến sẽ chỉ ảnh hưởng trong chuỗi cho nên khi thể hiện dấu nháy đôi trong chuỗi thì không cần sử dựng dấu escape.

```
$element = <<<EOQ
<textarea name="$name" cols="$cols" rows="$rows"
wrap="$wrap">$value</textarea>
EOQ;
```

Như ví dụ trên các bạn thấy không cần phải hao phí nhiều dấu suyệt (\), chúng ta vẫn có thể có được một chuỗi chứa các ký hiệu dạng biến không có tầm ảnh hưởng ra bên ngoài.

Các phần tử mảng sử dụng khoá liên hợp (bạn sẽ tìm hiểu ở phần tiếp theo) không thể sử dụng Here Doc được. Ví dụ sau đây sẽ xuất hiện lỗi:

```
$array = array ("fname"=>"jay", "lname"=>"greenspan");
$str = <<<EOQ
print my string $array["fname"]
EOQ;
```

# Mảng (array) trong PHP

Mảng là một dạng của biến trong đó có chứa nhiều giá trị. Ví dụ một dạng đơn giản của mảng là tháng:

```
$thang = array("Gieng", "Hai", "Ba", "Bon", "Nam",
"Sau", "Bay", "Tam", "Chin", "Muoi", "Muoi Mot", "Muoi Hai");
```

Mảng này có chứa 12 phần tử, và bạn có thể định vị chúng bằng thứ tự ở trong mảng, bắt đầu bằng vị trí 0. Do đó lệnh echo $thang[0] sẽ cho ra là Gieng và echo $thang[11] sẽ cho ra Muoi Hai. Để truy xuất được tất cả các phần tử trong mảng, bạn có thể tính ra chiều dài của mảng và thực hiện vòng lặp:

```
for ($i=0; $i<count($months); $i++)
{
echo $thang[$i] . "<br>\n" ;
}
```

Chi tiết về vòng lặp sẽ được trình bày ở các phần sau. Bạn có thể gán giá trị vào mảng với một phép toán đơn giản như sau:

```
$dogs = array();
$dogs[0] = "kiki";
$dogs[1] = "lulu";
```

Nếu bạn không xác định chỉ số bên trong ngoặc vuông thì giá trị sẽ được gán cho phần tử cuối mảng. Trong ví dụ sau "nana sẽ được gán vào $dogs[2]:

```
$dogs[] = "nana";
```

# Mảng liên hợp

Cũng giống như các ngôn ngữ khác, PHP tận dụng khả năng của mảng liên hợp (associative array). Có thể bạn cảm thấy mới mẻ với khái niệm này. Để tôi nói sơ qua một chút: Mỗi phần tử trong mảng liên hợp mang khoá(key) riêng. Các phần tử của mảng sẽ được truy cập thông qua khoá. Điều này giống như cách thức truy xuất trong các query khi làm việc với Database. Trong ví dụ sau, bạn sẽ thấy các phần tử first_name, last_name, e-mail sử dụng các key:

```
$person = array (
"first_name" => "Jay",
```

```
"last_name" => "Greenspan",
"e-mail" => "jgreen_1@yahoo.com"
);
```

Nếu như bạn muốn thêm phần tử vào mảng, bạn có thể gán tiếp một giá trị khác. Dòng lệnh sau sẽ thêm một số nguyên vào trong mảng, do đó mảng này sẽ chứa tất cả 4 phần tử.

```
$person["age"] = 32;
```

Nếu bạn muốn truy cập cả khoá và giá trị của một mảng liên hợp, bạn sẽ dùng **list() = each()** như sau:

```
while (list($key, $value) = each($person))
{
echo "<b>key :</b> $key, value = $value <br>\n";
}
```

Các chương sau này tôi sẽ nói kỹ về `list() = each()` một cách chi tiết hơn. Trên cơ bản `each()` truy xuất được cả khoá và giá trị của phần tử trong mảng. List() giữ các giá trị

và gán vào $key và $value. Tiến trình này tiếp tục cho đến khi mỗi phần tử trong mảng được truy cập. Nếu bạn muốn duyệt qua hết mảng bạn cần phải sử dụng reset($person).

Nếu bạn chỉ muốn sử dụng giá trị của phần tử trong mảng mà thôi hoặc bạn muốn sử dụng mảng không liên lợp và vẫn muốn sử dụng cấu trúc list()=each() bạn phải thực hiện như sau:

```
while (list( , $value) = each($person))
{
echo "value = $value <br>\n";
}
```

Hoặc bạn chỉ muốn truy xuất khoá, bạn sẽ làm như sau:

```
while (list($key) = each($person))
{
echo "key = $key <br>\n";
}
```

*Hãy nhận định về mảng trong PHP như sau:*
*- Tất cả các mảng trong PHP đều là mảng liên hợp. Tại vì sao? Bởi vì những mảng không phải là liên hợp thì PHP cũng sẽ tự động gán cho chúng các key. Ví dụ: $x= array ("pug", "poodle"), PHP sẽ tự gán cho $x các khoá là các con số nguyên theo thứ tự bắt đầu từ số 0. Bạn sẽ được tìm hiểu kỹ ở chương 6.*

# Mảng đa chiều

PHP cũng hỗ trợ mảng đa chiều. Mảng đa chiều thường sử dụng nhất đó là mảng hai chiều. Chúng chứa thông tin dựa trên hai khoá. Giả sử, nếu chúng ta chứa thông tin hai người trở lên thì mảng hai chiều sẽ hỗ trợ việc này rất tốt. Chúng ta sẽ xác lập một mảng $people. Trong mảng $people lại chứa mảng cho từng cá nhân:

```
$people = array (
"khai" => array ("ho_lot" => "tongphuoc", "tuoi" => 30),
"minh" => array ("ho_lot" => "leanh" , "tuoi" => 52)
);
```

Ta thấy $people chứa các thông tin của 2 người, Khai và Minh. Để truy cập một trị trong bất kỳ thông tin của cá nhân nào bạn sẽ phải dùng cả hai khoá. Ví dụ để truy xuất tuổi của Minh bạn sẽ thực hiện lệnh như sau:

```
echo $people["minh"]["tuoi"];
```

Bạn có thể truy cập tất cả các phần tử trong mảng hai chiều bằng cách sử dụng vòng lặp trên cả hai chiều của mảng:

```
while(list($person, $person_array) = each($people))
{
echo "<b>Ban biet gi ve $person</b><br>\n";
while(list($person_attribute, $value) = each($person_array))
{
echo "$person_attribute = $value<br>\n";
}
}
```

# Biến gán từ trình duyệt (web browser)

Quan điểm chung của việc sử dụng PHP cũng như các ngôn ngữ khác là cung cấp khả năng nhập thông tin theo ý muốn của khách. Thông thường các thông tin này được nhập vào thông qua một form HTML. Nhưng cũng có thể chúng xuất phát từ các nguồn khác như: HTML, cookie, session.

# Biến từ Form của HTML

Dạng thông thường nhất để khách có thể nhập thông tin riêng là thông qua một form HTML. Trong phần phụ lục A có trình bày chi tiết về các tạo một form HTML. Nếu bạn chưa biết gì về cách tạo form này thì hãy đọc phần phụ lục. Bạn hãy tạo trang sign.php chỉ chứa 100% mã lệnh HTML như sau (có thể đặt là sign.htm cũng được):

```
<form action=mypage.php action=post>
<input type=text name=email>
<input type=text name=first_name>
<input type=submit name=submit value=OK>
<input type=submit name=reset value=Cancel> </form>
```

Một khi khách nhấp chuột vào nút SUBMIT (chấp nhận) thì các biến như $email, $first_name, và $submit sẽ được chuyển giao sang trang action là mypage.php. Sau đó, trong trang mypage.php bạn sẽ xử lý các biến này tuỳ thuộc vào mục đích chương trình. Để ý rằng phần lớn các ứng dụng trong sách này đều sử dụng giá trị của nút lệnh SUBMIT.

Trong trang mypage.php bạn phải viết các lệnh để xử các thao tác của người truy cập. Bạn hãy xem cách xử lý trong trang mypage.php mẫu như sau:

```
<?php
if (isset($submit) && $submit=="OK")
{
echo "Cam on ban da gui thong tin cho chung toi.";
} else {
?>       <form action=mypage.php action=post>
         <input type=text name=email>
         <input type=text name=first_name>
         <input type=submit name=submit value=OK>
         <input type=submit name=reset value=Cancel> </form>

         <?php
         }
?>
```

Bạn hãy xem kỹ ví dụ trên, nếu như người truy cập nhập đủ thông tin và nhấn nút **OK** từ trang sign.php (chứa toàn mã lệnh html), thì nó chuyển sang trang mypage.php và xuất ra dòng thông báo: Cam on ban da gui thong tin cho chung toi. Ngược lại, nếu như nhấn nút **Cancel** thì nó sẽ thực hiện mã lệnh trong lệnh Else và sẽ hiển thị form để buộc nhập lại.

**Chú ý:** Bạn hãy xem lại cách thức submit trong ví dụ GuestBook ở tập một. Trong tập 1, nếu bạn không chọn Submit thì chương trình sẽ gọi lại trang sign.php là trang chứa Form nhập liệu

bằng lệnh include. Còn ở đây không gọi lại trang sign.php nữa, bởi vì chúng ta làm theo kiểu khác là gắn Form nhập liệu ngay trong file Action là mypage.php.

Các biến cũng có thể được truy xuất thông qua mảng $HTTP_POST_VARS hoặc $HTTP_GET_VARS, dựa vào method sử dụng trong form của bạn. Việc này rất thuận tiện, nếu các biến từ các forms có thể mang cùng tên với biến trong script của bạn, hoặc nếu bạn có các biến chưa định nghĩa được chuyển giao thì bạn sẽ tìm được ở đó.

Bạn có thể truy cập bất kỳ phần tử riêng biệt nào như đã làm trong mảng liên hợp ($HTTP_POST_VARS["e-mail"]). Hoặc bạn có thể tạo vòng lặp duyệt qua tất cả các phần tử của mảng:

```
while (list($key, $value) = each($HTTP_POST_VARS))
{
echo "variable = $key value = $value <br>";
}
```

# Truyền mảng

Có những trường hợp khi việc chuyển giao biến không thể thực hiện được. Ví dụ như khi bạn chọn cả hai giá trị cho cùng một biến. Việc này thường xảy ra khi làm việc với form có chứa listbox và có thể là bạn sẽ giữ phím Ctrl để chọn phần tử thứ 2 trong list. Ta giải quyết bằng cách sử dụng phép truyền mảng.

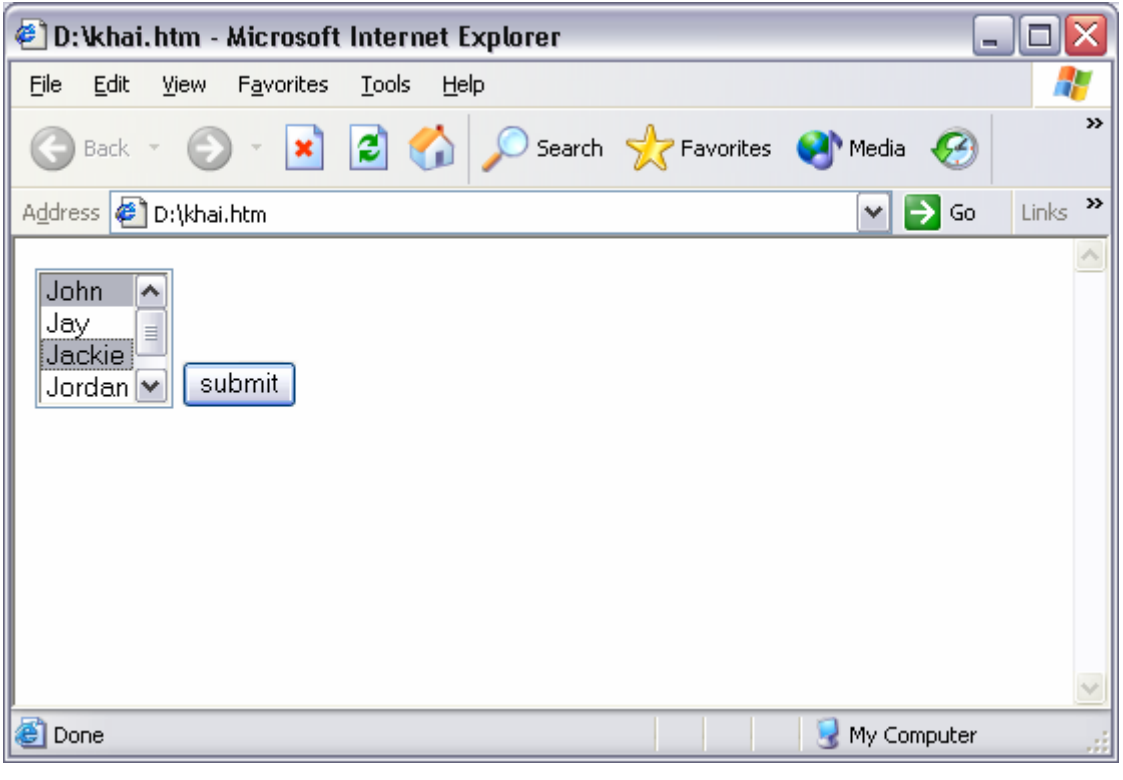
Các lệnh được sử dụng như sau:

```
<form action ="mypage.php" method="post">
<select name="j_names[]" size=4 multiple>
<option value="2">John
<option value="3">Jay
<option value="4">Jackie
<option value="5">Jordan
<option value="6">Julia
</select>
<input type="submit" value="submit">
</form>
```

Để ý rằng trong trong lệnh câu lệnh select name, tôi sử dụng dấu ngoặc vuông để bảo PHP biết rằng đây là một mảng. Nếu không sử dụng dấu [ ] thì sẽ có tới 2 giá trị gán cho cùng một biến.

Một khi được SUBMIT bạn có thể truy cập mảng như sau:

```
if (is_array($j_names))
{
echo "<b>the select values are:<br> <br>";
while(list($key, $value) = each($j_names))
{
echo $value . "<br>\n";
}
}
```


D:\khai.htm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address D:\khai.htm
Go
Links
John
Jay
Jackie
Jordan
submit
Done
My Computer

Việc truyền mảng rất thông dụng khi bạn Submit Form có một loạt các checkbox (tên các mặt hàng chẳng hạn). Khách truy cập có thể sẽ nhấp chuột vào nhiều checkbox hoặc không có checkbox nào. Trong chương 8, có ví dụ cho phép người quản trị có thể sử dụng checkbox để chọn và xoá các phần tử.
Mảng được chuyển giao từ form có thể có khoá liên hợp, ngay cả đối với mảng nhiều chiều. Tên của phần tử form thường có tên là name = "array_name[element_name]". Hoặc đối với mảng nhiều chiều là array_name[element_name] [subelement_name]".

# Cookies

Cookie là những **file nhỏ** chứa một số các thông tin truy cập Web. Các cookie do Websever phát sinh, lưu giữ lại, sau đó sẽ được đọc ở những lần truy cập về sau.
Cookie đơn thuần chỉ là *thông tin ghi nhận lại những động tác truy cập web của khách*. Khi tồn tại trên đĩa cứng, cookie trở thành các thỉnh cầu của giao thức HTTP, được gởi đến Webserver.
Để có thể phát sinh một cookie bạn cần phải sử dụng hàm setcookie() như sau:

```
setcookie(name, value, time_to_expire, path, domain, security
setting);
```

Chúng ta sẽ tìm hiểu chi tiết về cookie ở chương 6. Còn bây giờ bạn tìm hiểu sơ lược các chức năng thông qua một ví dụ:

```
setcookie("my_cookie",
"my_id",time()+(60*60*24*30),"/",".mydomain.com", 0)
```

Lệnh trên sẽ phát sinh một cookie với các chức năng sau:
- Chứa một biến tên là my_cookie
- Giá trị của mycookie my_id
- Cookie tồn tại trong vòng 30 ngày kể từ ngày nó phát sinh (`time()+(30*24*60*60)` ngày giờ hiện tại + 30 ngày được quy ra giây).
- Cookie có tác dụng đến tất cả các trang trong domain. Bạn có thể hạn chế lại bằng các chỉ ra đường dẫn đến một số trang nào đó trong domain.
- Nó sẽ hiện diện trong tất cả các website có địa chỉ http://mydomain.com
- Không có xác lập đặc biệt nào về bảo mật.

Một khi cookie được xác lập, các biến phát sinh từ cookie có tác dụng giống như biến phát sinh từ form mà chúng ta đã bàn trước đây. Chúng sẽ hiện diện với chức năng là biến global.

Sau khi script PHP xác lập cookie, các script khác trong domain có thể truy cập cookie một cách tự động.
Nếu như bạn muốn cẩn thận hơn để $mycookie không xung đột với một biến nào khác cũng có tên $mycookie, bạn có thể truy xuất nó thông qua mảng HTTP_COOKIE_VARS và sử dụng lệnh: HTTP_COOKIE_VARS["mycookie"].

Bạn có thể xác lập cookie cung cấp khả năng truy xuất như là một mảng:

```
setcookie("mycookie[first]","dddd",time()+2592000,"/","192.168.1.
1", 0);

setcookie("mycookie[second]","my_second_id",time()+2592000,"/","1
92.168.1.1", 0);
```

Cả hai biến trên đều có thể truy cập đến như là một mảng liên hợp.

## Sessions

PHP4 cũng giống như ASP và ColdFusion đều có hỗ trợ session, việc này giúp ích rất nhiều cho việc truy cập web. Vậy session là gì?

Đơn giản nó chỉ là **một cách thức để duy trì và truyền biến** trong khi chuyển tiếp giữa các trang web. Chương trình của bạn khai báo một session được bắt đầu với hàm start_session(). PHP đăng ký một SesssionID duy nhất, và thường thì ID này được gởi đến user thông qua một cookie. PHP sau đó tạo một tập tin trên server để theo dõi sự thay đổi của biến. Tập tin này có tên giống như tên của SessionID.

Một khi session được tạo, bạn có thể đăng ký bất kỳ số lượng biến. Các giá trị của những biến này được lưu giữ trong tập tin trên server. Cũng như sự tồn tại của cookie, các biến trong session sẽ hiện diện trên bất kỳ trang nào được truy cập đến trong phạm vi một domain. Việc xác lập này rất thuận tiện hơn là chuyển tiếp các biến từ trang này sang trang khác thông qua các phần tử ẩn trong form hay cookie.

Session nói chung là khá đơn giản. Hãy xem script sau sẽ đăng ký một biến session tên là $my_var, và sẽ gán cho nó một giá trị là "hello world".

```
<?
session_start();
session_register("my_var");
$my_var = "hello world";
?>
```

Trên những trang tiếp theo biến $my_var sẽ hiện diện, nhưng chỉ sau khi bạn chạy hàm sesssion_start(). Hàm này bảo PHP tìm kiếm một session xem có tồn tại hay không, rồi làm cho các biến session trở thành global. Nó có thể sử dụng câu lệnh IF để làm cho các biến session hoàn toàn có thể truy cập được. Hãy xem xét ví dụ sau:

```
<?php
session_start();
session_register("your_name");
//check to see if $your name contains anything
if(!empty($your_name))
{
echo "I already know your name, $your_name";
}
//this portion will probaby run the first time to
//this page.
elseif(empty($your_name) && !isset($submit))
{
echo "<form name=myform method=post action=$PHP_SELF>
<input type=text name=first_name> first name<br>
<input type=text name=last_name> last name<br>
<input type=submit name=submit value=submit>
</form>";
//if the form has been submitted, this portion will
```

```
//run and make an assignment to $your_name.
} elseif (isset($submit) && empty($your_name))
{
$your_name = $first_name . " " . $last_name;
echo "Thank you, $your_name";
}
```

Sau khi chạy chương trình này, chọn **refresh** trên trình duyệt. Bạn sẽ thấy script sẽ nhớ được rằng bạn là ai.

Các hàm setcookie() và session_start() nên ở vị trí gần đầu tập tin. Nếu bạn thử chuyển đến trình duyệt trước để xác lập một cookie bạn sẽ nhận được một thông báo lỗi.

# Biến sẵn có

Có rất nhiều biến sẵn có của PHP và Server. Bạn có thể liệt kê một danh sách đầy đủ bằng cách sử dụng lệnh phpinfo() để xem. Bạn hãy tạo một file php và cho chạy thử xem:

```
<?php
phpinfo();
?>
```

Bạn có thể sử dụng các biến này bằng nhiều cách thức khác nhau. Tôi sẽ trình bày một sau ngay sau đây, và sẽ chỉ ra bạn nên dùng vào trường hợp nào. Một số biến đến từ PHP engine, một số khác bắt nguồn từ Webserver.

# Biến sẵn có của PHP

## PHP_SELF

Biến này nhận giá trị là địa chỉ hiện tại của tập tin .php đang được duyệt. Địa chỉ này sẽ là địa chỉ đầy đủ từ gốc (bắt đầu từ http://) . Bạn sẽ sử dụng nó khi muốn truy cập lại chính trang web đang thi thành.

Xét ví dụ sau, đây là một form tương tự như form sign.php mà các bạn đã có dịp xét qua. Nếu khách thực hiện thao tác khác với submit thì chính form này sẽ được thi hành lại:

```
<?
if(isset($submit))
{
//Xuat ra thong bao tai day
echo "Cam on ban da submit";
} else {
?>
<form name=myform method=post action=<?=$PHP_SELF?>>
<input type=text name=first_name> first name<br>
<input type=text name=last_name> last name<br>
<input type=submit name=submit value=submit>
```

```
</form>
<?
}
?>
```

## HTTP_POST_VARS

Đây là một mảng chứa tất cả các biến được chuyển tiếp thông qua POST method từ một form. Bạn có thể truy cập từng biến riêng rẽ như là một phần tử của mảng liên hợp (ví dụ: $PHP_POST_VARS["myname"]).

## HTTP_GET_VARS

Đây là một mảng chứa tất cả các biến được chuyển tiếp thông qua GET method. Bạn có thể truy cập từng biến riêng rẽ như là một phần tử của mảng liên hợp (ví dụ: $PHP_GET_VARS["myname"]).

## HTTP_COOKIE_VARS

Tất cả các cookie chuyển đến trình duyệt đều có thể được truy xuất trong mảng liên hợp này. Nó bao gồm cả session cookie. Nếu bạn còn thắc mắc cookie sẽ thi hành như thế nào thì hãy xem hàm phpinfo() để biết được trình duyệt của bạn đang chuyển đến server những gì.

# BIẾN CỦA APACHE

Apache có sẵn rất nhiều biến. Tôi không trình bày đầy đủ tất cả các biến ra đây. Các biến bạn sử dụng, chúng tuỳ thuộc vào xác lập hiện tại của bạn như thế nào. Sau đây là một số biến mà có lẽ bạn sẽ sử dụng thường xuyên trong chương trình của bạn.

## DOCUMENT_ROOT

Biến này trả về đường dẫn của Webserver. Biến này được tôi sử dụng trong xuyên suốt quyển sách này. Hãy xét ví dụ sau:

```
include"$DOCUMENT_ROOT/book/functions/charset.php";
```

Bằng cách sử dụng biến DOCUMENT_ROOT thay vì dùng đường dẫn tuyệt đối, chúng ta có thể di chuyển toàn bộ một thư mục sang một Apache Server khác mà không lo lắng rằng đường dẫn sẽ bị sai lệch trong include path. Nên nhớ rằng nếu như bạn không sử dụng Apache Server thì biến này không sử dụng được. Nếu bạn sử dụng include_path trong tập tin php.ini,

bạn không cần phải lo lắng phải xác định đường dẫn như thế nào bởi vì PHP sẽ duyệt hết tất cả các thư mục và tìm ra tập tin bạn đã chỉ định.

## HTTP_USER_AGENT

Bất kỳ ai đã từng thiết kết Web site đều hiểu rằng tầm quan trọng của việc nhận dạng được trình duyệt của người sử dụng là gì. Một số trình duyệt thì không sử dụng được JavaScript, một số khác thì đòi hỏi dạng HTML đơn giản. Biến user_agent cung cấp cho bạn khả năng uyển chuyển đối với từng trình duyệt khác nhau. Một user_agent chuẩn có dạng như thế này:

```
Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)
```

Nếu bạn phân tích chuỗi này ra bạn sẽ biết được những gì bạn cần tìm. Có thể bạn chỉ thích hàm get_browser() của PHP. Về lý thuyết mà nói, hàm này định nghĩa khả năng cho phép của trình duyệt của user đang sử dụng. Cho nên bạn có thể biết được là chương trình của bạn đang phục vụ tốt hay không. Các sách PHP có những hướng dẫn về cách cài đặt và sử dụng get_browser(), nhưng tôi khuyên bạn không nên sử dụng nó. Bởi vì sử dụng get_browse bạn sẽ được bảo rằng IE 5 dùng cho PC và Netscape 4.01 dùng cho Mac có hỗ trợ CSS (cascading stylesheets) và JavaScript. Nhưng bất kỳ người sử dụng nào cũng biết rằng: viết lệnh DHTML

để chạy trên cả hai môi trường trình duyệt này là một công việc phức tạp. Thông tin bạn nhận được từ get_browser() có thể dẫn đến những tính năng giả trong bảo mật. Cách tốt nhất là bạn sử dụng HTTP_USER_AGENT và thực hiện quyết định của mình dựa trên trình duyệt hoặc platform xác định nào đó.

**REMOTE_ADDR**

Dùng để lấy địa chỉ IP của user. Tuy nhiên có những user am hiểu chuyện này và có thể họ thay đổi IP của máy mình. Cho nên không lấy gì để đảm bảo rằng: một địa chỉ IP chắn chắn là của một user nào đó. Bạn sử dụng biến này để theo dõi sự truy nhập của một user nhưng nó chỉ mang tính tương đối thôi.

**REQUEST_URI**

Biến này cũng giống như biến PHP_SELF. Ngoài ra nó còn chứa thêm tham số trong địa chỉ truy vấn . Nếu bạn truy cập vào địa chỉ:

`http://www.mydomain.com/info/products/index.php?id=6`

Thì biến REQUEST_URI của bạn có giá trị là: `info/products/index.php?id=6`

**SCRIPT_FILENAME**

Biến này chứa toàn bộ đường dẫn của tập tin.

# Kiểm tra biến

Ở trên chúng ta đã nói nhiều về Biến. Như các bạn biết đó, tên của một biến không quan trọng bằng giá trị mà nó chứa trong đó. Như tôi đã nói Biến trong PHP rất uyển chuyển. Điều này phát sinh sự bất lợi là bạn sẽ không biết ở tại một thời điểm nào đó thì biến này sẽ mang giá trị gì. Do đó bạn cần phải thực hiện thao tác kiểm tra biến.

## isset( )

Hàm này thực hiện việc kiểm tra biến có chứa giá trị hay không. Nó sẽ trả về giá trị TRUE hoặc FALSE. Nếu biến chưa được xác lập thì trị isset() sẽ là false.

Bạn hãy xem xét ví dụ sau, nó thi hành một query MySQL. Bạn đã biết rằng một field trong database có thể chứa trị null hoặc chuỗi rỗng. Với việc sử dụng hàm isset() bạn sẽ kiểm tra và phân biệt được hai giá trị trên. Trong đoạn lệnh PHP bên dưới. Trong đó biến $query là một phát biểu SELECT lấy dữ liệu submit từ form của user.

```
$result = mysql_query($query) or
die (mysql_error());
$number_cols = mysql_num_fields($result);
```

```
echo "<b>query: $query</b><br>\n";
//layout table header
echo "<table border = 1>\n";
echo "<tr align=center>\n";
for ($i=0; $i<$number_cols; $i++)
{
echo "<th>", mysql_field_name($result, $i), "</th>\n";
}
echo "</tr>\n";//end table header
//layout table body
while ($row = mysql_fetch_row($result))
{
echo "<tr align=left>\n";
for ($i=0; $i<$number_cols; $i++)
{
echo "<td>";
if (!isset($row[$i])) //test for null value
{echo "NULL";}
else
{echo $row[$i];}
echo "</td>\n";
```

```
}
echo "</tr>\n";
}
echo "</table>";
```

Lưu ý rằng dấu chấm than (!) có nghĩa là phủ định.
Tức là nếu $var có giá trị null thì:
isset($var) cho ra giá trị False
!isset($var) cho ra giá trị True

## empty()

Hàm empty() có vẻ ngược ngạo so với hàm isset(). Nó sẽ cho ra trị True nếu $var có trị null, chuỗi rỗng hoặc số 0. Hàm này thường được sử dụng để kiểm tra xem user có nhập trị vào trong form hay không:

```
if(empty($first_name))
{
echo "Ban can phai nhap ten cua minh";
exit;
```

```
}
```

## is_int( )

Hàm này để kiểm tra biến có phải là số nguyên hay không. Có 2 cú pháp khác cho cùng kết quả như nó là: is_integer và is_long(). Bạn sử dụng hàm này khi không chắc rằng biến là một trị nguyên hay chuỗi. Ví dụ:

```
$a = "222";
$b = 22;
```

is_int($a) cho ra trị False
is_int($b) cho ra trị True

Tương tự bạn sẽ có một loạt hàm kiểm tra kiểu của biến sau đây:

## is_double()

Kiểm tra số kiểu double (dấu phẩy động). Hàm thay thế: `is_float()` và `is_real()`.

## is_string( )
Kiểm tra kiểu chuỗi.

## is_array( )
Kiểm tra kiểu mảng.

## is_bool( )
Kiểm tra kiểu boolean (TRUE và FALSE)

## is_object( )
Kiểm tra biến kiểu object. Bạn sẽ tìm hiểu kiểu object trong các phần sau.

## gettype( )
Hàm này sẽ cho bạn biết kiểu của biến như: string, double, integer, array, hoặc boolean. Ngoài ra nó có trả về các kiểu như object, class. Bạn sẽ khảo sát kỹ về việc lập trình hướng đối tượng trong các phần sau để biết thêm về object và class.

Lưu ý trị của hàn gettype() trả về luôn là một chuỗi: "string", "integer", "double" v.v.
Bạn hãy xem ví dụ sau:

```
$str = "Day la mot chuoi";
$type = gettype($str);
if ($type == "string")
{
echo "Dung vay";
}
```

# Đổi kiểu của biến

Bạn sẽ sử dụng 3 cách để đổi kiểu của biến.

## Phương pháp type casting

Phương pháp này rất đơn giản: Bạn chỉ cần ghi tên kiểu ra, đóng ngoặc đơn lại, rồi đặt trước biến. Tức khắc biến sẽ bị đổi theo kiểu mà bạn muốn.

Cách thức: (kiểu) $biến

Ví dụ:

```
$a = 1;
$b = (string)$a; //số 1 sẽ biến thành chuỗi 1
echo gettype($a), "<br>\n";
echo gettype($b), "<br>\n";
```

Kết quả cho ra là:

```
integer
string
```

## Sử dụng hàm settype( )

Hàm này có 2 đối số. Thứ nhất là tên biến, thứ nhì là kiểu. Ưu điểm của nó là nó có thể cho ra kết quả FALSE nếu như việc hoán đổi không được.

Cách thức: settype($biến, "kiểu")

Ví dụ:

```
$a = 1;
settype($a, "string");
```

## Sử dụng hàm intval( ), doubleval( ), và stringval( )

Phương pháp này thường để bạn áp dụng nhanh trong khi tính toán. Có lẽ nhìn tên hàm bạn cũng biết được chức năng của nó rồi. Hãy xét ví dụ sau:

```
$a = "43";//43 là kiểu chuỗi
$b = (intval($a) * 2);
```

# Biến của biến

Nghe qua có vẻ lạ lạ, nhưng đây là một "độc chiêu" của PHP. Với cách thức này bạn sẽ lấy giá trị của một biến để hình thành tên của một biến mới.
Cách thức: $$biến
Ví dụ:

```
$a = ‘khai’;
$$a = ‘Chao moi nguoi’;
```

Bạn sẽ thấy trong ví dụ trên một biến mới được hình thành đó là $khai chứa giá trị là "Chao moi nguoi"

Xét thêm ví dụ sau, trong đó $tacgia là một mảng liên hợp.

```
<?
$tacgia = array ("ho"=>"Tong", "ten"=>"Khai");
while (list($field,$value) = each($tacgia))
```

```
{
$field = "bien_$field";
$$field = $value;
}
echo $bien_ho, " ", $bien_ten;
?>
```

Khi chạy chương trình, các biến sau sẽ được tạo `$bien_ho, $bien_ten` và ghi ra màn hình: Tong Khai

# Tóm tắt

Bạn đã tìm hiểu các biến trong PHP. Bạn thấy PHP xử lý các biến linh hoạt hơn nhiều so với các ngôn ngữ khác. Còn một vấn đề khá quan trọng đối với biến đó là scope bạn cũng sẽ biết kỹ về nó ở trong các phần sau của giáo trình này.

**(Còn tiếp)**