Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang

# Hướng dẫn học lập trình

# PHP &MySQL

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang



## Cài đặt Web server - bước đầu tiên để học PHP!

"Trường học" vừa khai giảng, trong lúc đợi bài học đầu tiên ra lò, cho phép tớ được "múa rìu qua mắt thợ cái"!

Như bác ngocha85 đã nói, để học PHP và MySQL, một trong những thứ cần chuẩn bị là web server chạy trên PC của mình. Để cho nhanh chóng, theo tớ tốt nhất nên cài bộ XAMPP.

**Câu hỏi 1: XAMPP là gì?**

Trả lời: XAMPP giống với WAMP, nghĩa là người mới học ko cần phải biết cách cài đặt riêng lẻ từng thành phần như Apache, PHP và MySQL. Chỉ cần download một gói về là xong.

Các tính năng có trong XAMPP:

1. Apache 2 => server
2. PHP 5 => ngôn ngữ lập trình
3. MySQL => cơ sở dữ liệu
4. Webalizer => quản lý statistic của site
5. Mercury => giả lập gửi email
6. FileZilla => giả lập FTP server
7. Rất nhiều tính năng chuyên sâu khác...

**Câu hỏi 2: Tại sao ko dùng WAMP?**

Trả lời: Vì cái này tớ chưa dùng bao giờ + Cái này bác ngocha85 chắc sẽ giới thiệu => tránh đụng hàng là hơn. Với lại cái XAMPP này theo tớ cũng rất hay, thậm chí ko cần cài đặt, chỉ cần copy và chạy.

**Câu hỏi 3: Down XAMPP ở đâu?**

Trả lời: Ở đây: http://www.apachefriends.org/en/xampp.html
Có đủ bộ XAMPP cho Windows, Linux, MacOS và cả Solaris, vì vậy mọi hệ điều hành nó đều chấp tất!

**Câu hỏi 4: Cài đặt và sử dụng XAMPP như thế nào?**

Trả lời: Sau khi tải về, bạn sẽ có 1 file zip. Giải nén file đó ra 1 thư mục bất kỳ, ví dụ C:\XAMPP. Để chạy web server, bạn kích hoạt file xampp_control.exe, bấm nút Start bên cạnh Apache và nút close để XAMPP Control tự động chuyển xuống system tray.

Ngoài ra, bạn có thể khởi động MySQL nếu dùng cơ sở dữ liệu, FileZilla nếu dùng FTP và Mercury nếu dùng email.

Để biết chắc web server đã chạy đúng, bạn mở trình duyệt web của mình, gõ http://localhost vào thanh Address, sau đó enter. Một trang thông báo sẽ hiện ra, cho biết quá trình "cài đặt" đã hoàn tất.

**Câu hỏi 5: Làm thế nào để chạy các script viết bằng PHP?**

Trả lời: Bạn cho script vào thư mục C:\XAMPP\htdocs\ sau đó gọi file đó qua URL http://localhost/script_name.php

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang



Vậy là hết "bài chuẩn bị cho bài khởi động" của bác ngocha85 sẽ post trong vài ngày tới. Tèn tén ten!

Thừa thắng xông lên, tớ làm luôn bài "Hello World".

## Bài 1: Nói "hello world" với PHP

### Cần chuẩn bị những gì?

1. Web server cần đảm bảo sẵn sàng. Apache được khởi động theo bài post ở trên.
2. Một script editor. Cái này có rất nhiều, như PHP Designer, Dev-PHP, ... Thậm chí dùng notepad cũng được. Nhưng tốt nhất nên dùng một editor có hỗ trợ unicode. Như tớ dùng SCiTE.
*Bạn vào trang này để xem list và review các PHP editor: http://www.php-editors.com/*
3. 5 phút thời gian rảnh rỗi.

### Bắt đầu!!!

1. Tạo một file mang tên "helloworld.php" trong thư mục htdocs. Mở file đó bằng script editor.
2. Gõ đoạn code sau vào editor:

PHP Code:

```
<?php
   echo "Hello World!";
?>
```

3. Mở trình duyệt web, gõ http://localhost/helloworld.php [enter].
4. Nhắm mắt lại trong 0.0001 giây. Nếu mở mắt ra mà bạn thấy dòng chữ Hello World là đã thành công rồi đó!

### Giải thích

1. Dòng thứ nhất của file helloworld.php là "<?php" và dòng cuối cùng là "?>". Đây là 2 thẻ (tag) để báo cho server biết điểm bắt đầu và kết thúc của một đoạn code PHP. Nói cách khác, bằng cách này bạn có thể nhúng code PHP trong bất cứ file HTML có sẵn nào. Khi thực thi file PHP, web server sẽ chỉ thực hiện những đoạn code đặt trong 2 thẻ này và bỏ qua tất tần tật những phần còn lại.
2. Dòng 2 là một lệnh của PHP: Lệnh echo. Lệnh này làm nhiệm vụ in một xâu ra ngoài màn hình. Cần nhớ một lệnh PHP luôn kết thúc bằng dấu chấm phẩy ";". Nếu thiếu dù chỉ một dấu chấm phẩy, code của bạn sẽ ko chạy và dừng lại biểu tình ngay.
3. Cũng ở dòng 2, xâu "Hello World" được đặt trong dấu ngoặc kép. Nếu ko, sẽ có lỗi.

### Một vài câu hỏi

1. Có cần thiết phải trình bày như trên ko?
=> Ko. Bạn có thể trình bày code theo bất cứ cách nào bạn muốn. Lùi vào 10 dấu cách, mỗi dòng cách nhau 3 hàng, ... Điều đó là tùy bạn. Tuy nhiên cần phải viết code cho thật dễ đọc và dễ hiểu để tiện cho việc sửa đổi và chia sẻ code sau này.

2. Có cách nào báo hiệu một đoạn code PHP ngoài cách dùng <?php ko?
=> Có. Nhiều cách là đằng khác. Ví dụ bạn có thể viết

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang



PHP Code:

```
<?
   // Code ở đây
?>
```

Tuy nhiên các cách khác đều ít thông dụng và được khuyến cáo ko nên sử dụng.

3. Có thể đặt xâu Hello World trong dấu ngoặc đơn ko?
=> Có thể. Bạn có thể dùng dấu ngoặc kép và dấu nháy đơn để chứa xâu. Sự khác nhau giữa chúng sẽ được thảo luận sau.

4. Nếu trong xâu cũng có dấu ngoặc / xâu là một đoạn văn bản rất dài thì sao?
=> Ko có gì phải lo lắng. Cái gì cũng có cách giải quyết. Vấn đề là cách đó ko nằm trong bài học hôm nay. Hết 5 phút rồi, bạn hãy nghỉ ngơi đã. :P

**Bài tập**

Vì Bài 1 hết sức đơn giản, chỉ theo tinh thần Hello World nên bài tập cũng sẽ chỉ có bài, và cũng rất rất đơn giản.

Hãy cho biết lỗi sai trong các đoạn code sau:

1.

PHP Code:

```
echo "Hello World!";
```

2.

PHP Code:

```
<?php
    echo "Hello World!"
?>
```

3.

PHP Code:

```
<?php
    echo "Hello World!';
?>
```

4.

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang


PHP Code:

```php
<?php
    echo "Hello World!";
?>
```

## Bài 2 - Mục 1: Lưu trữ dữ liệu trong PHP. Vài điều cần nói về biến.

Trước khi bắt đầu bài 2, tớ xin trình bày về cách chú thích (comment) trong PHP. Đây có thể coi là một kỹ năng cũng được, vì bạn rất KHÔNG NÊN viết code mà ko có chú thích. Có thể đoạn code rất dễ hiểu vào thời điểm viết, nhưng nếu ko có chú thích, chỉ vài tháng sau bạn có thể quên ngay mình đã viết cái gì. Viết chú thích ngay vào thời điểm code là cách tốt nhất.

Trong PHP, một dòng chú thích được đặt sau 2 dấu sổ chéo //

Ví dụ

PHP Code:

```php
// Đây là một dòng chú thích
```

Nếu chú thích của bạn dài hơn 1 dòng, bạn có thể để nó trong 1 block, mở đầu bằng /* và kết thúc bằng */

PHP Code:

```php
/*
   Chú thích dòng thứ nhất
   Thứ 2
   Thứ 3
   Vân vân...
*/
```

Còn một cách nữa, KHÔNG phổ biến (ít ra là tớ thấy thế), đó là chú thích đặt sau dấu #. Chú thích này cũng chỉ cho phép 1 dòng giống như //

Một điều khác cũng rất cần chú ý đó là PHP ko cho phép đặt chú thích trong chú thích (nested comment).

Ta bắt đầu vào Bài 2.

## Bài 2: Lưu trữ dữ liệu trong PHP

Khi bắt tay vào lập trình một chương trình, hiển nhiên ta sẽ cần phải lưu trữ dữ liệu. Cụ thể, dữ liệu có thể được lưu trữ bằng biến (variable). Khác với các ngôn ngữ lập trình khác, trong PHP các biến ko cần phải khai báo (declare) trước khi sử dụng. Để sử dụng biến, bạn chỉ cần gán (assign) cho nó một giá trị (value). Biến sẽ tự động được tạo. Cực kỳ đơn giản và nhanh chóng!

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang



### 1. Biến. Khai báo. Đặt tên.

Biến trong PHP bắt đầu bằng dấu dollar ($), theo sau là tên biến. Tên biến có thể bắt đầu bằng dấu gạch dưới (_ gọi là underscore) hoặc chữ cái. Tiếp sau đó là các chữ cái, số hoặc lại là dấu gạch dưới. Một số ký tự mở rộng (extended character) có thể được sử dụng, nhưng tốt nhất là nên tránh.

Một số ví dụ về biến ĐÚNG: $uds, $update_softs, $uds_has_more_than_26000_members

Biến sai: abc vì thiếu dấu dollar, $124adfd vì bắt đầu bằng số

Cũng cần thảo luận thêm một chút: Cũng vì sự dễ dãi trong việc ko phải khai báo biến nên sẽ có lúc bạn gõ nhầm tên biến. Ví dụ $uds gõ thành $usd (ặc!)

Ví dụ:

PHP Code:

```php
<?php
   $uds = "Welcome to UDS!";
   echo $uds;
?>
```

May mắn làm sao, từ bản PHP 5 trở lên, sẽ có một cảnh báo (warning) khi bạn chạy script, cho biết bạn chưa gán giá trị cho biến $usd.

À, còn một vấn đề chưa nói đến: Đó là trong PHP, tên biến CÓ phân biệt chữ hoa chữ thường (case-sensitive). Nghĩa là $uds hoàn toàn khác với $UDS hay $uDs. Nói chung nên tránh việc đặt tên biến chỉ khác nhau cách viết hoa thường này, vừa đỡ mất công giữ Shift, vừa đỡ nhớ nhầm tên biến.

Mục 2 sẽ mang tên **Một số kiểu dữ liệu trong PHP**. Mọi người đón đọc nhá!

**Bài tập**

Trong các biến sau đây, biến nào được đặt tên đúng, biến nào bị đặt tên sai:

1. this_is_a_variable
2. $yet another variable
3. $simplevariable
4. $blah_blah_blah_123456789_____
5. $123456789_____abacabadfskdjsfksdfkdserwuewrjfdksj fdksljf
6. $^^
7. $___A___VARIABLE____
8. $THiS_iS_THe_LaST_ONe

## Bài 2 - Mục 2: Lưu trữ dữ liệu trong PHP. Một số kiểu dữ liệu cơ bản [updated]

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang



*Ngoài lề một chút: Lúc đầu tớ cũng ko định tách Bài 2 ra làm mấy thread, nhưng nếu để như thế kia thì dài quá, sợ đọc theo các bác mệt mắt => nản lòng.*

Ta có một số kiểu dữ liệu cơ bản sau đây trong PHP:

a. Kiểu số (number)
b. Kiểu xâu (string)
c. Kiểu boolean (boolean)

**a. Kiểu số**

Trong kiểu số (lại) có 2 kiểu cơ bản khác: Số nguyên (int) và số thực (float). Số nguyên có thể biểu diễn bằng số thập phân (hệ 10 - decimal), hệ 8 (octal) và hệ 16 (hexadecimal).

Ví dụ ta gán giá trị cho một số biến kiểu NGUYÊN như sau:

PHP Code:

```
<?php
   $a = 27;
   $b = -27;
   $c = 027;
   $d = -027;
   $e = 0x27;
   $f = -0x27;
?>
```

Ở ví dụ trên, cả 6 biến từ $a đến $f đều có giá trị là 27 hoặc -27. Tuy nhiên, với biến $a và $b, ta dùng kiểu biểu diễn số thập phân (viết như số ta viết hàng ngày). Với $c và $d, dùng kiểu số hệ 8 (bắt đầu với chữ số 0). Với $e và $f dùng kiểu hệ 16 (bắt đầu với chữ số 0 và chữ cái x).

Nếu đã từng học qua Pascal, chắc chắn bạn sẽ hỏi tớ: Thế nếu tớ dùng 1 biến kiểu int, gán cho nó một giá trị cao bằng max của int, thì khi đem số đó cộng với 1, giá trị có bị chuyển thành âm do tràn số (overflow) ko?

Câu trả lời là ko. Một biến kiểu int có giá trị cực lớn trong PHP là 2147483647, khi cộng 1 vẫn sẽ trả giá trị đúng là 2147483648, nhưng lần này sẽ thuộc kiểu float. Nói cách khác, PHP tự chuyển số bị tràn lên kiểu float.

Nếu thích đặt câu hỏi, chắc chắn (lại một lần nữa) bạn sẽ hỏi tớ: Sao cậu biết điều ấy?

Câu trả lời rất đơn giản: Bạn hãy cùng tớ làm ví dụ với đoạn code sau:

PHP Code:

```
<?php
   $a = 2147483647;
   var_dump($a);
   $a = $a + 1;
```

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang


```
  var_dump($a);
?>
```

Sau khi chạy script, kết quả trả về sẽ là

int(2147483647) float(2147483648)

=> Đúng như tớ nói nhá!

Tớ xin giải thích như thế này:

Ở dòng thứ nhất, ta đem gán giá trị 2147483647 cho $a. Đây là một giá trị cực to, nhưng vẫn nằm trong int, vì vậy $a sẽ thuộc kiểu int.
Dòng thứ 2 và thứ 4, ta dùng lệnh var_dump($a); Đây là lệnh in ra kiểu và giá trị của một biến trong PHP. Chú ý nhé, lệnh này khá phổ biến và hay được dùng để debug code.
Ở dòng thứ 3, ta dùng lệnh gán $a = $a + 1; Với các bạn đã học lập trình, điều này chẳng có gì khó hiểu. Sau khi thực thi lệnh, $a sẽ mang giá trị của $a cộng thêm với 1. Còn nếu (chẳng may) bạn chưa học lập trình bao giờ, thì tớ (lại) xin giải thích như thế này:
- Dấu bằng ở đây là lệnh gán, đem giá trị của vế phải gán cho vế trái, chứ ko phải là dấu bằng trong biểu thức toán học mà mình vẫn học. Do đó, ko có gì là trái với lẽ tự nhiên cả. :P

Một điều khác mà bạn nên nhớ, đó là hãy THẬT cẩn thận khi sử dụng số kiểu float trong PHP. Nó luôn chỉ là những giá trị xấp xỉ, và ko hề chính xác tuyệt đối. Do đó tốt nhất là chuyển số float sang int khi có thể. Cách làm sẽ được thảo luận sau.

Giờ ta sang kiểu xâu.

**b. Kiểu xâu**

Định nghĩa một cái nào: Xâu là một chuỗi các ký tự. Một câu tớ xì pam là một xâu. Cả cái bài viết này cũng có thể là một xâu.

Để sử dụng xâu, có 3 cách (hic, bắt đầu phức tạp rồi => mọi người đứng dậy vươn vai cái cho tỉnh táo! :P):

Cách 1 là dùng nháy đơn.
Cách 2 là dùng ngoặc kép (hay gọi là nháy kép gì cũng được).
Cách 3 là dùng kiểu HEREDOC.

Nói rõ nhé:

Cách 1: Xâu được đặt trong dấu nháy đơn.

PHP Code:

```
<?php
  echo 'Đây là xâu đặt trong dấu nháy đơn';
?>
```

Sẽ có bạn hỏi tớ (sao hỏi nhiều thế!): Trong xâu có thể đặt dấu nháy đơn được ko? Kiểu như

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang


xâu là I'm a student ý.
Câu trả lời là bạn phải thêm một dấu sược (hay suộc gì ý) trước dấu nháy đơn "bất thường" ý. Như thế này:

PHP Code:

```
echo 'Trong nháy đơn lại có một nháy đơn như thế này \', và như thế này nữa \'.';
```

Đặt cái dấu đó (\) gọi là "escape the character". Nói nhỏ nhá: Bài tớ viết hay chèn tiếng Anh vào là để các bạn đỡ "bỡ ngỡ" khi đọc tut hay doc bằng Eng.

Một lần nữa, (lại) có một câu hỏi được đặt ra: Nếu trong xâu cũng có một dấu \ thì sao?
Câu trả lời cũng rất giản dị: Dùng thêm một dấu \ nữa ngay trước dấu \ ý. Như thế này \\

Lần này, sẽ ko có một câu hỏi, mà sẽ là một tiếng thở dài: Sao lắm thứ thế? Còn cái dấu nào phải "escape" như dấu \ và ' ko?
Có. Đó là:
1. \n : Báo hiệu xuống dòng trong PHP. Giống như <br /> trong HTML.
2. \t : Thay mặt cho Tab
3. \$ : Dấu dollar (tránh "cạnh tranh lành mạnh" với tên biến mà! :P)
4. ... Để gặp nói sau. Nói nhiều e "tẩu hỏa nhập ma" chết!

Quên mất, trừ \' và \\, mấy cái escape này chỉ dùng trong trường hợp xâu đặt trong dấu ngoặc kép.

Hờ hờ, lại quên một điều phải nói trước khi chuyển qua phần kế tiếp: Nếu trong xâu ta ko thèm dùng dấu \, cũng kóc thèm dùng dấu ', mà dùng cả \' cho "dân chơi" thì sao? 
Trả lời: Thì cứ làm như bình thường thôi. Như thế nè: \\\'. Dấu \ thứ 1 để escape cho dấu \ thứ 2. Dấu \ thứ 3 để escape cho dấu ' cuối cùng. Thường thôi!

Cách 2: Xâu được đặt trong dấu ngoặc kép (hay nháy kép - whatever)

Trường hợp này rất giống với sử dụng dấu nháy đơn đã nói ở trên.

PHP Code:

```
<?php
   echo "Xâu này đặt trong dấu ngoặc kép";
?>
```

Sở dĩ nói RẤT giống mà ko phải HOÀN TOÀN giống vì giữa chúng có điểm khác nhau: Khi thực thi, PHP sẽ tìm và thay thế trong xâu những ký tự đặc biệt được escape (như \n, \t...) như đã nói ở trên, cùng với các biến (nếu có) trong xâu.

Ví dụ:

PHP Code:

```
<?php
   $a = 1;
```

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang



```
   echo "Biến \$a có giá trị là $a";
?>
```

Sẽ cho ta kết quả: Biến $a có giá trị là 1

Trong khi đó, nếu sử dụng dấu nháy đơn:

PHP Code:

```
<?php
   $a = 1;
   echo 'Biến \$a có giá trị là $a';
?>
```

Lại in ra: Biến \$a có giá trị là $a

Điều đó cho thấy: Khi sử dụng dấu nháy đơn, giá trị của biến trong xâu, cùng với các ký tự đặc biệt cần escape sẽ ko được in ra. Các bạn nhớ kỹ điều này nhé!

Ta sang cách thứ 3: Xâu đặt trong cấu trúc HEREDOC

Ở cách 1, PHP sẽ nhận thấy 1 xâu được bắt đầu với dấu nháy đơn thứ nhất và kết thúc với dấu nháy đơn thứ 2. Tương tự với cách 2, nhưng là dấu ngoặc kép.

Ở cách 3 này, PHP sẽ coi một xâu bắt đầu bằng 3 dấu nhỏ hơn viết liền nhau <<<, đi kèm với 1 tên định danh (identifier) tùy ý bạn đặt tên, ví dụ là HERE, kết thúc là tên đó kèm theo dấu ;

Nghe có vẻ hơi phức tạp, nhưng bạn hãy cùng tớ gõ ví dụ sau: (chú ý là chữ HERE có thể thay bằng bất cứ chữ gì, tên bạn chẳng hạn, miễn là nó tuân theo nguyên tắc đặt tên biến của PHP. À, mà nhớ là mở bằng <<<HERE thì phải đóng bằng HERE; nhé, ko được mở cửa ra vào, đóng cửa sổ đâu!)

PHP Code:

```
<?php
   echo <<<HERE
            Xâu &#273;ược ghi ở dòng thứ nhất
   Dòng th&#7913; 2
      Dòng th&#7913; 3
   Vân vân
HERE;
?>
```

Nhìn vào ví dụ trên, bạn có nhận xét gì?

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang


Thứ nhất, xâu ko nhất thiết phải thuộc một dòng. Nó ko nhất thiết phải ngắn gọn, mà có thể dài "tràng giang đại hải" ra mấy chục dòng cũng được. Điều này rất tiện nếu bạn muốn echo một lúc cả một bài thơ chẳng hạn!

Thứ hai, chữ HERE; ở dòng cuối cùng tớ ko căn lề với chữ echo ở dòng 1. Đó là LUẬT, dù tớ thấy hơi "bất công" và "nghiệt ngã" một tí:
- Sau <<<HERE phải xuống dòng. Ko được phép có dù chỉ 1 ký tự trắng (dấu cách ý)
- Trước và sau HERE; cũng thế. Ko được phép có dù chỉ 1 ký tự trằng. Nói cách khác, đừng dại gì căn lề cho dòng này.

Cái gì là LUẬT thì phải THEO, cãi ko được Còn nếu ko theo, PHP sẽ báo lỗi:

Quote:

```
Parse error: parse error, unexpected T_SL in E:\XAMPP\htdocs\test.php on line 2
```

Tớ nói dông dài như vậy là vì đã từng mất bao nhiêu thời gian mới tìm ra được lỗi sai của mình. Chỉ vì một dấu cách mà chương trình đình công, ko thèm chạy! Kinh nghiệm xương máu!

Còn một ý nữa: Nếu từ nãy đến giờ bạn chỉ đọc "chay", ko thực hành thì (chưa chắc) đã nhận thấy: Khi chạy chương trinh, thay vì in ra mấy dòng như trên, PHP lại in mọi thứ ra cùng 1 dòng:

Quote:

```
Xâu được ghi ở dòng thứ nhất Dòng thứ 2 Dòng thứ 3 Vân vân
```

Sửa chữa điều này cũng khá đơn giản:

PHP Code:

```
<?php
   $s = <<<HERE
            Xâu &#273;ược ghi ở dòng thứ nhất
   Dòng th&#7913; 2
      Dòng th&#7913; 3
   Vân vân
HERE;
   echo nl2br($s);
?>
```

Có gì bí ẩn ở đây ko? Thay vì echo thẳng mấy dòng kia ra, ta đi "vòng vèo" một chút bằng cách gán xâu chứa mấy dòng đó cho biến $s, sau đó echo nl2br($s) ra màn hình.

nl2br() được gọi là một hàm (function). Nó nhận một xâu làm tham số (parameter), ở đây là xâu $s, sau đó in ra theo luật: Cứ gặp dấu xuống dòng trong code là chuyển thành dấu xuống dòng trong HTML.

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang


Cái tên nl2br cũng chẳng phải thần chú gì khó nhớ, nó rất giản dị: chỉ là viết tắt của new-line-to-br. New-line là dấu xuống dòng trong code, 2 là to , br là <br /> (thẻ xuống dòng trong HTML).

Vậy là vấn đề đã được giải quyết. Kết quả in ra đúng như mong đợi:

Quote:

```
Xâu được ghi ở dòng thứ nhất
Dòng thứ 2
Dòng thứ 3
Vân vân
```

Kiểu dữ liệu cơ bản cuối cùng mà tớ sẽ nói tới chính là Kiểu boolean.

**c. Kiểu boolean**

Đây là kiểu dữ liệu đơn giản nhất trong PHP (đỡ quá!). Ý tưởng rất đơn giản: Mọi thứ chỉ thuộc vào 1 trong 2 loại: Đúng hoặc Sai, Có và Không, 1 và 0. Không có ngoại lệ. Anh ko là True thì sẽ là False. Ở đây ko có chỗ cho người ba phải!!!

Giá trị của biến kiểu boolean là TRUE hoặc FALSE. Hai từ này hoàn toàn ko phân biện hoa thường, vì vậy có thể viết như thế nào cũng được: TRue, tRUe, true, ...

Ví dụ:

PHP Code:

```php
<?php
   $a = TRUE;
   $b = false;
?>
```

Một kiểu dữ liệu đơn giản đồng nghĩa với việc ko cần giải thích nhiều về ví dụ của nó.

Nhưng nó lại ko đồng nghĩa với việc: Kiểu boolean chẳng có gì đáng nói! Thực tế là kiểu này rất hay dùng trong PHP, ví dụ khi tính toán một biểu thức và xem giá trị của nó có lớn hơn một số nào đấy hay ko... (biểu thức điều kiện)

Xin được kết thúc Bài 2 tại đây. Cảm ơn quý vị đã quan tâm theo dõi...

COMING UP NEXT: **Một số hàm cần thiết khi debug code**

## Bài 2 - Mục 3: Các kiểu dữ liệu quan trọng khác

Tiếp sau mục 2: Các kiểu dữ liệu cơ bản, tớ xin giới thiệu thêm một vài kiểu dữ liệu quan trọng khác của PHP: Mảng, Đối tượng, Null và Resource (sozy vì 2 kiểu cuối ko rõ dịch như thế nào)

**1. Mảng (array)**

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang



Mảng được sử dụng khi bạn muốn lưu trữ một số lượng lớn các biến. Một ví dụ hết sức đơn giản: Một lớp có 50 học sinh, và bạn muốn quản lý cả 50 học sinh đó. Để đại diện cho một học sinh, tất nhiên bạn sẽ muốn 1 biến. Nhưng nếu đặt tên là hs1, hs2, ... hs50 thì quả là quá mất thời gian! Và đây chính là lý do để mảng có "đất dụng võ".

Mảng chứa rất nhiều giá trị (value), mỗi giá trị được truy cập nhờ khóa (key). Khóa có thể chỉ là những số đếm thông thường như 1, 2, 3, hay có thể là xâu, như "abc", "def", "ghi". Mảng có khóa là xâu như vậy được gọi là associative array.

Để khai báo một mảng, chúng ta có thể sử dụng cách như ví dụ sau:

PHP Code:

```php
<?php
   $a = array(1, 2, 3, 4);
   $b = array("a", "b", "c");
   $c = array(1, "a", array(3, 4));
?>
```

Như ở ví dụ trên, $a, $b, $c đều là mảng. Mảng $a chứa các số từ 1 đến 4, mảng $b chứa các xâu "a", "b", "c". Còn mảng $c sành điệu hơn, chứa cả số lẫn xâu, thêm cả một mảng ở bên trong nó nữa.

Sau khi khởi tạo giá trị trong mảng $a, mặc định mỗi phần tử (element) trong nó sẽ được gán cho một khóa là số nguyên. Nó bắt đầu từ 0, ko phải là 1. Do đó, phần tử thứ 0 sẽ là 1, thứ 1 sẽ là 2, vân vân.

Ví dụ:

PHP Code:

```php
<?php
   echo $a[2];
?>
```

Sẽ in ra màn hình giá trị 3 - tức là phần tử mang khóa là 2 trong mảng $a.

Như tớ đã nói ở trên, một khóa có thể là một xâu, nghĩa là người ta có thể truy cập mảng $d (chẳng hạn) bằng cách dùng $d["blah"]. Vậy ta khởi tạo giá trị của $d như thế nào?

Rất đơn giản, ta sử dụng toán tử (operator) =>

PHP Code:

```php
<?php
   $d = array("blah" => 1, "abc" => 2, "def" => "ghi");
?>
```

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang


Có thể dễ dàng đoán được: Nếu dùng lệnh echo $d["def"] sẽ cho ra kết quả là "ghi".

Tìm hiểu sâu thêm về Mảng, kiểu dữ liệu mạnh mẽ của PHP, sẽ là phần việc của một Bài học sau này.

**Đối tượng (object)**

PHP5 là một ngôn ngữ lập trình hướng đối tượng (OO - Object Oriented). Nói một cách đơn giản nhất (nhưng vẫn nghe ù tai nếu bạn chưa nghe về đối tượng bao giờ) thì lập trình hướng đối tượng (OOP - Object Oriented Programming) là việc tạo ra một kiểu dữ liệu mới (đối tượng - object hay lớp - class). Thay vì việc phải tạo một dãy các hàm liên quan đến đối tượng đó, bạn sử dụng thuộc tính (properties) và phương thức (method) trực tiếp của đối tượng ý.

Hãy nhắm mắt vào tưởng tượng. Bạn có một quả bóng bay. Quả bóng ý có những thuộc tính gì? À, rất đơn giản thôi: Đó có thể là kích thước, màu sắc hay độ căng - xẹp của bóng.

Còn phương thức: Quả bóng có thể căng lên, hoặc xẹp đi. Rất dễ dàng phải ko?

Giờ hãy tưởng tượng, bạn có một đối tượng mang tên QB (quả bóng ). Để tạo ra một quả bóng, bạn dùng lệnh:

PHP Code:

```
<?php
   $bong = new QB();
?>
```

Quả bóng có kích thước (KT), màu sắc (MS) và độ căng - xẹp (CX). Để $bong mang màu đỏ, bạn có thể viết:

PHP Code:

```
<?php
   $bong->MS = red;
?>
```

Tương tự, nói đến kích thước, độ căng - xẹp của quả bóng, ta có thể dùng $bong->KT, $bong->CX.

Thế còn phương thức? Như đã nói, quả bóng có thể căng lên (CL) hoặc xẹp đi (XD). Để thực thi các phương thức này, ta làm như ví dụ sau:

PHP Code:

```
<?php
   $bong->CL();
?>
```

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang


Tạm dừng việc "cưỡi ngựa xem hoa" phần đối tượng tại đây.

**3. Null**

Một biến được coi là NULL (ko có giá trị) nếu nó thỏa mãn cả 3 điều kiện sau:
1. Nó được gán là NULL (ko phân biệt hoa thường)
2. Nó chưa bao giờ "được" (hay "bị") gán giá trị.
3. Nó đã bị "xử đẹp" bằng unset - hàm hủy bỏ các biến chỉ định.

Để kiểm tra một biến có là NULL hay ko, ta có thể sử dụng hàm is_null(biến). Ví dụ:

PHP Code:

```
<?php
   $test = NULL;
   echo is_null($test);
?>
```

Cho ra kết quả là 1.

**4. Resource**

Có những lúc PHP cần xử lý các đối tượng như kết nối cơ sở dữ liệu hay các đối tượng của hệ điều hành. Chúng sẽ được coi là resource.

Nói chung trong hầu hết các trường hợp, bạn thậm chí ko nhận ra việc mình có phải đang làm việc với resource hay ko.

## Bài 3: Kết hợp PHP và HTML

Nói đến PHP, người ta nói đến lập trình web. Nói đến HTML, người ta cũng nói đến làm web. Vậy ko có lý gì HTML và PHP lại ko đi được cùng với nhau! Bài 3 sẽ đề cập tới một vấn đề rất phổ biến khi lập trình PHP: Kết hợp mã PHP với HTML.

Trước hết, chúng ta hãy dành ít phút tìm hiểu cách thức hoạt động của World Wide Web (WWW).

Hãy tưởng tượng, bạn đang muốn truy cập trang web www.example.com/welcome.html. Bạn mở trình duyệt web, gõ vào ô địa chỉ: www.example.com/welcome.html và bấm Enter. Trang web sẽ hiện ra, gần như ngay tức khắc (ở đây ko nói đến mạng dial up siêu chậm nhá )

Vậy, điều gì đã xảy ra từ lúc bạn bấm Enter cho đến lúc trang web xuất hiện? Hãy cùng tớ xem xét những đoạn băng "behind the scene" này:

1. Ngay sau khi bạn bấm Enter, trình duyệt bạn đang dùng sẽ gửi một thông điệp (message) lên mạng, cho biết bạn đang muốn yêu cầu (request) trang www.example.com/welcome.html
2. Thông điệp đó được chuyển tới máy tính tại địa chỉ www.example.com/welcome.html
3. Máy chủ trên máy tính đó sẽ nhận được thông điệp và bắt đầu tìm kiếm file HTML được yêu cầu.
4. Máy chủ gửi file HTML đó về máy tính vừa yêu cầu (chính là máy tính của bạn). Nếu ko tìm

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang


thấy file HTML được yêu cầu, đơn giản là máy chủ sẽ trả lại một thông báo lỗi.
5. Trình duyệt của bạn, sau khi nhận về trang HTML, sẽ hiển thị nó ra màn hình.

Ở bước thứ 4, nếu file bạn yêu cầu là 1 file mang đuôi .php, thay vì gửi trả lại nội dung nguyên gốc của file, máy chủ sẽ lần lượt thực hiện thêm các bước:
1. Quét file trong chế độ HTML, gửi trả về nội dung HTML.
2. Ngay khi gặp <?php, máy chủ sẽ chuyển sang chế độ PHP, bắt đầu thực thi các lệnh PHP cho đến khi gặp ?>. Hiển nhiên nếu các lệnh PHP có output, máy chủ sẽ trả những output đó cho trình duyệt.
3. Kết thúc chế độ PHP (ra ngoài ?>), máy chủ quay lại chế độ HTML.
Quá trình cứ thế tiếp tục, cho đến khi kết thúc file .php.

Vậy là đã xong phần nói ngoài lề. Giờ ta bắt đầu vào Bài 3.

Ở Bài 1, tớ đã cùng các bạn viết chương trình đầu tiên, Hello World, bằng PHP. Giờ thử nhìn một file .php cũng mang nội dung Hello World:

HTML Code:

```
<html>
	<head>
		<title>Hello World</title>
	</head>
	<body>
		<p>Hello World!</p>
	</body>
</html>
```

Như ví dụ trên đây, các bạn có thể thấy: Đây chỉ đơn thuần là một file HTML, mang đuôi .php. Chẳng có gì đặc biệt! Và khi trình duyệt yêu cầu file này, máy chủ chỉ việc gửi trả nội dung nguyên gốc mà ko cần phải xử lý một chút lệnh nào cả.

Giờ hãy thử nâng cấp file .php đó bằng cách thêm vào nó một chút mã PHP:

PHP Code:

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <?php
      echo "<p>Hello World!</p>";
    ?>
```

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang


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

Khi chạy script này, kết quả khi view source code cũng ko khác gì ví dụ đầu tiên. Chỉ có cách làm là khác, thay vì chỉ sử dụng HTML, ta kết hợp cả PHP và HTML trong cùng một file.

Giờ, nếu ta muốn in ra màn hình chữ Hello ở một dòng, và World ở một dòng, ta sẽ làm ntn?

Nếu các bạn có biết về HTML, thì sẽ nghĩ ngay đến thẻ <br />:

PHP Code:

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <?php
      echo "<p>Hello<br />World!</p>";
    ?>
  </body>
</html>
```

Kết quả output thật mỹ mãn và chẳng có gì đáng nói.

Tuy vậy, nếu các bạn còn nhớ, tớ đã từng nói \n có thể dùng để xuống dòng trong PHP. Vậy, thừa thắng xông lên, bạn sẽ thay <br /> bằng \n:

PHP Code:

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <?php
      echo "<p>Hello\nWorld!</p>";
    ?>
  </body>
</html>
```

Kết quả? Thất bại. Trên màn hình, chữ Hello và World vẫn nằm cùng một dòng. Tại sao lại như

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang



vậy? Làm thế nào để giải quyết vấn đề này?

Trả lời: \n đúng là để xuống dòng, nhưng đó là xuống dòng trong PHP output, nó ko đảm bảo việc xuống dòng khi cái PHP output đó được trình duyệt xử lý dưới dạng mã HTML.

Để trình duyệt xử lý chính xác những vấn đề ntn, ta cho toàn bộ xâu đó vào thẻ <pre>, thẻ quyết định việc giữ nguyên định dạng của xâu:

PHP Code:

```
<html>
   <head>
      <title>Hello World</title>
   </head>
   <body>
      <?php
         echo "<p><pre>Hello\nWorld!</pre></p>";
      ?>
   </body>
</html>
```

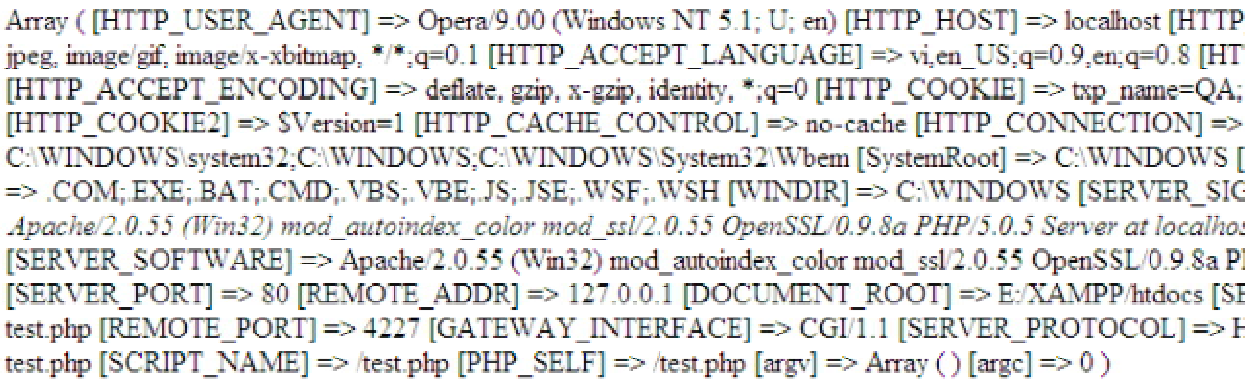
Một lần nữa, kết quả hiện ra thật mỹ mãn.

Ta xét thêm một ví dụ nữa.

PHP Code:

```
<?php
   print_r($_SERVER);
?>
```

Script trên làm trò gì vậy ta? Chưa cần biết print_r và $_SERVER là gì, bạn chỉ cần thấy đoạn nó output ra mấy dòng sau: (tớ phải post ảnh vì UDS ko cho phép đưa đoạn ý vào bài viết)



Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang


Phản ứng đầu tiên? Bạn sẽ thấy hơi chóng mặt phải ko ạ? Bạn sẽ tự hỏi: Nhiều thứ thế kia viết líu ríu vào nhau thì ai mà đọc được? Tại sao ko tách dòng ra chứ?

À, nói đến tách dòng, bạn sẽ nhớ ngay tới thẻ <pre> mà tớ nói bên trên. Bạn sẽ thêm nó vào script của mình:

PHP Code:

```
<?php
   echo "<pre>";
   print_r($_SERVER);
   echo "</pre>";
?>
```

Kết quả trả về ko thể nói là dễ đọc, mà phải nói là rất dễ đọc , tuy (có thể) bạn chẳng hiểu cái gì sất!

Hãy tạm hài lòng với những gì mình vừa làm được và thư giãn một chút trước khi ta bước vào Bài 4.

Bật mí trước: Chúng ta đã biết tới hàm var_dump in ra kiểu và giá trị của biến, hàm print_r (bạn đoán là) in ra các giá trị của một cái $_SERVER gì đó. Vậy trong Bài 4, ta sẽ cùng tìm hiểu print_r là gì, sử dụng ra sao, và còn những hàm nào như vậy nữa.

## Bài 4: Các toán tử

Để thực hiện việc tính toán các giá trị trong PHP, ta sử dụng toán tử (operator).

### 1. Gán (assignment)

Toán tử gán (dấu =) được sử dụng hết sức đơn giản. Ví dụ:

PHP Code:

```
<?php
   $a = 1;
   $b = 1;
   $c = "cool";
?>
```

Sau ví dụ, biến $a và $b mang giá trị 1, $c mang giá trị "cool".

Để cho ngắn gọn, thay vì phải mất 2 dòng khai báo $a và $b, ta có thể gộp:

PHP Code:

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang



```php
<?php
   $a = $b = 1;
   // Hoặc: $b = $a = 1;
?>
```

Kết quả vẫn đúng như mong đợi.

**2. Toán tử số học (arithmetic)**

Các toán tử này gồm có: + (cộng - addition), - (trừ - subtraction), * (nhân - multiplication), / (chia - division) và % (tính modul - modulus).

Ví dụ:

PHP Code:

```php
<?php
   $a = 10;
   $b = 5;
   $c = $a + $b; // $c = 15
   $d = $c - $a; // $d = 5
   $e = $a / $b; // $e = 2
   $f = $e * $b; // $f = 10
   $g = $a % $e; // $g = 0
?>
```

Ngoài ra, để sau khi tính toán, giá trị $a bằng $a nhân 2 chẳng hạn, thay vì viết $a = $a * 2; ta có thể viết ngắn gọn: $a *= 2;

Tương tự, có thể viết $a += 10; $a -= 1; $a /= 3; $a %= 1; Cấu trúc này rất giống C và C++, nên nếu bạn đã biết qua 2 ngôn ngữ này thì ko có gì phải bỡ ngỡ.

**3. Toán tử so sánh (comparision)**

Toán tử so sánh gồm những toán tử sau:

== Mang giá trị TRUE khi 2 vế mang cùng giá trị
=== Mang giá trị TRUE khi 2 vế mang cùng giá trị VÀ cùng kiểu
!= Mang giá trị TRUE khi 2 vế ko cùng giá trị
<> Mang giá trị TRUE khi 2 vế ko cùng giá trị
!== Mang giá trị TRUE khi 2 vế ko cùng giá trị HOẶC ko cùng kiểu
< Mang giá trị TRUE khi vế trái mang giá trị nhỏ hơn vế phải
> Mang giá trị TRUE khi vế trái mang giá trị lớn hơn vế phải
<= Mang giá trị TRUE khi vế trái mang giá trị nhỏ hơn hoặc bằng vế phải
>= Mang giá trị TRUE khi vế trái mang giá trị lớn hơn hoặc bằng vế phải