PHP (Part 1)

1. Tổng quan về PHP

1.1. Giới thiệu PHP

- **PHP** là viết tắt của chữ "Hypertext Preprocessor", đây là một ngôn ngữ lập trình được sử dụng để viết ở phía máy chủ (server side). Và PHP là một open source, nên chính vì thế nó có tính cộng đồng của nó cao và đồng thời cũng sẽ có rất nhiều các framawork, CMS hỗ trợ như Laravel, Wordpress.

1.2. Cài đặt PHP

Để có thể chạy được các ứng dụng viết bằng PHP thì chúng ta cần phải có 1 server được cài đặt phần mềm biên dịch PHP ra ngôn ngữ máy, nếu chúng ta chạy trên nền Internet thì cần phải thuê hosting tại các nhà cung cấp dịch vụ Internet và đăng ký tên miền (domain) sau đó upload trang PHP lên hosting, tuy nhiên để học PHP thì chúng ta cần cài Server ảo bằng các phần mềm server như XAMPP, WAMP, VERTRIGO

2. Cài đặt Webserver (Xampp)

2.1. Localhost

Hầu hết các trang web hiện nay đều viết bằng PHP & MySql, để chạy được các trang web này chúng ta cần đặt source code trên một host (là 1 vùng đĩa trên server) tại 1 máy chủ có hỗ trợ PHP & MySql, đối với môi trường Internet để mọi người có thể truy cập được trang web thì chúng ta cần thuê host tại các nhà cung cấp dịch vụ internet, tuy nhiên để thực hành tại máy cá nhân thì chúng ta cần tạo Localhost. Có rất nhiều phần mềm hỗ trợ việc này, tuy nhiên Xampp là 1 lựa chọn tối ưu.

Localhost là từ ghép của hai chữ “local” và “host“. Local dịch theo nghĩa IT là máy tính của chúng ta, Host theo nghĩa IT là máy chủ. Vậy localhost nghĩa là một máy chủ được vận hành trên máy tính của chúng ta.

Localhost bao gồm nhiều ứng dụng đi kèm với nhau và tất cả các ứng dụng đó sẽ kết hợp với nhau để tạo ra một môi trường có thể chạy các mã nguồn PHP & MySql, (chẳng hạn phần sau chúng ta sẽ cài WordPress) trên máy tính của cá nhân bao gồm:

* Phần mềm Webserver tên Apache, đây là webserver thông dụng nhất.
* Phần mềm PHP để xử lý mã PHP vì WordPress viết bằng ngôn ngữ PHP.
* Phần mềm MySQL Server để lưu trữ và xử lý cơ sở dữ liệu, do WordPress sử dụng MySQL làm nền tảng cơ sở dữ liệu. Cơ sở dữ liệu thường được mình viết theo chữ tiếng Anh là database.
* Phần mềm PHPMyAdmin để xem và quản lý cơ sở dữ liệu MySQL.

2.2. Localhost vận hành như thế nào

Khi cài đặt Localhost vào máy tính rồi, thì máy tính của chúng ta đã có một phần mềm Webserver để chạy ứng dụng website với địa chỉ là http://127.0.0.1. Đây là địa chỉ IP dạng localhost, ngoài ra chúng ta cũng có thể chạy localhost với đường dẫn là http://localhost.

Thông thường khi cài Localhost, mỗi khi cần sử dụng chúng ta sẽ cần mở bảng điều khiển của localhost lên và kích hoạt cho nó khởi động các ứng dụng đi kèm.

2.3. Lưu ý trước khi cài đặt

Xoá toàn bộ ứng dụng liên quan tới localhost

Nếu chúng ta có cài đặt các phần mềm liên quan đến việc làm localhost như PHP, MySQL thì hãy xoá hết. Và chúng ta không nên cài XAMPP trên Windows Server đã cài đặt IIS.

Lưu ý nếu dùng Skype

Nếu máy chúng ta đang cài đặt phần mềm Skype thì localhost sẽ không hoạt động được do Skype đã chiếm quyền sử dụng cổng mạng 80, đây là cổng mặc định của webserver. Do đó, chúng ta hãy mở Skype -> Tools -> Connection Options -> và bỏ chọn phần “Use port 80 and 443…..” rồi nhập một cổng bất kỳ để Skype sử dụng.

Tắt tường lửa

Nếu máy chúng ta có cài đặt tường lửa từ Windows hay từ một phần mềm Antivirus nào khác thì hãy tắt nó đi vì có thể nó sẽ chặn cổng 80 hoặc các ứng dụng webserver.

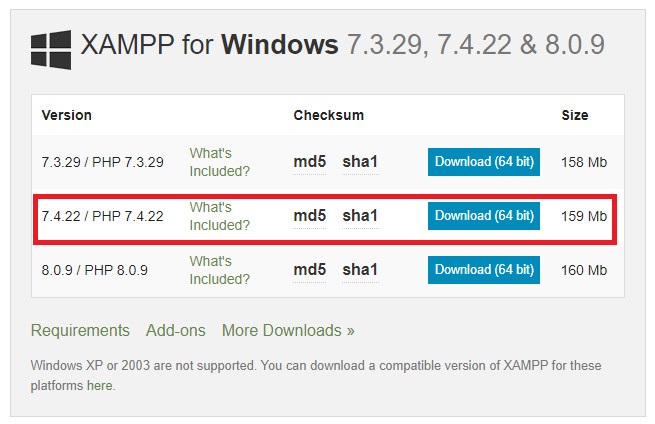
Tắt UAC trên Windows

Nếu máy của chúng ta đang dùng Windows và có bật chức năng User Account Control thì hãy tắt nó đi khi dùng localhost để tránh các vấn đề bị giới hạn quyền.

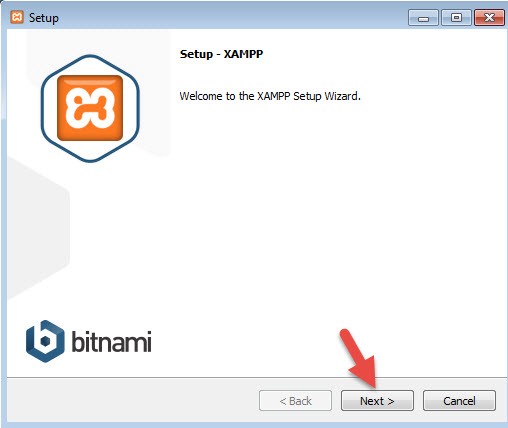
2.4. Qui trình cài đặt

Tải XAMPP, đầu tiên chúng ta truy cập vào địa chỉ <https://www.apachefriends.org/download.html>

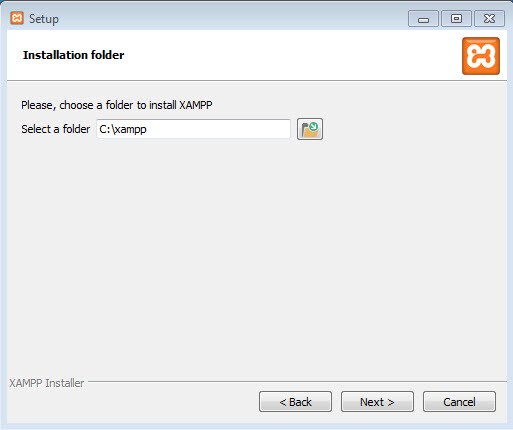
Sau đó chọn phiên bản XAMPP phù hợp với hệ điều hành của máy tính chúng ta đang sử dụng, và chúng ta nên chọn phiên bản PHP 7 trở lên (Nên chọn phiên bản 7.4.22 để sau này cài WordPress thuận tiện) Lưu ý là XAMPP chỉ có phiên bản cho hệ điều hành 32bit nhưng 64bit vẫn hoạt động bình thường.



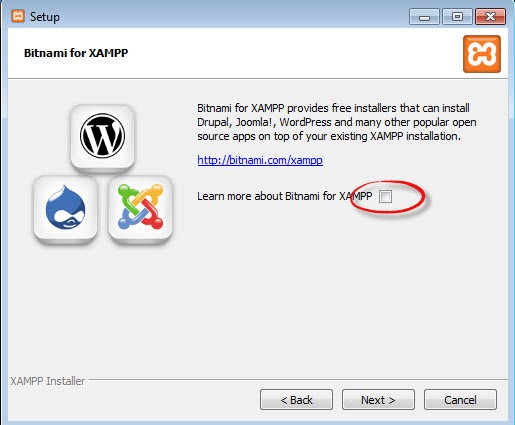
Sau khi tải file cài đặt về xong, hãy chạy nó, sau đó chọn Next.



Ở phần chọn đường dẫn, chúng ta hãy chọn đường dẫn cần lưu cài đặt của XAMPP. Lưu ý rằng đường dẫn này chúng ta phải nhớ vì khi cài đặt web lên localhost, chúng ta phải truy cập vào thư mục này. Chúng ta nên để mặc định là c:\xampp. Tiếp tục ấn Next.



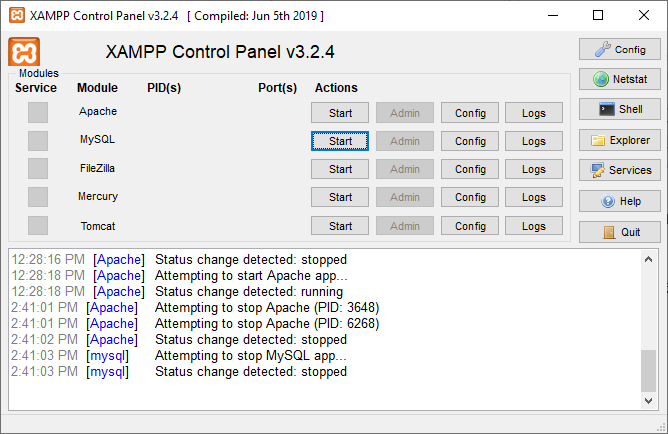
Ở trang kế tiếp, chúng ta bỏ chọn phần “Learn more about Bitnami for XAMPP“. Và ấn Next 2 lần nữa để bắt đầu quá trình cài đặt XAMPP.



Sau khi cài xong, ấn nút Finish để kết thúc cài đặt và mở bảng điều khiển của XAMPP. Tuy nhiên, hãy khởi động lại máy sau khi cài đặt xong để tránh tình trạng không khởi động được localhost.

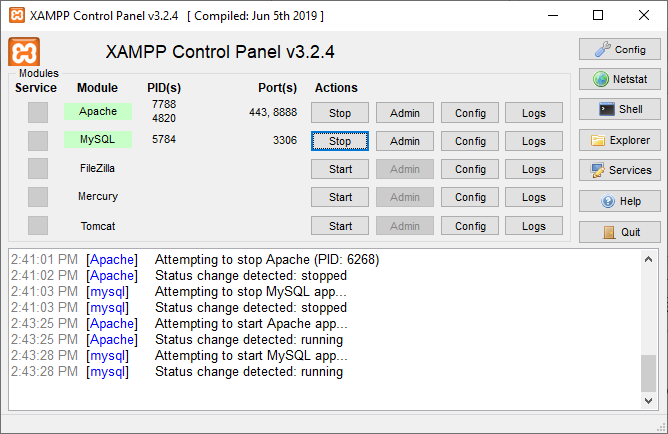
2.5. Khởi động Localhost

Bây giờ chúng ta hãy vào thư mục c:\xampp và mở file xampp-panel.exe lên để bật bảng điều khiển của XAMPP.



Chúng ta để ý sẽ thấy hai ứng dụng Apache và MySQL có nút Start, đó là dấu hiệu bảo 2 ứng dụng này chưa được khởi động, hãy ấn vào nút Start của từng ứng dụng để khởi động Webserver Apache và MySQL Server lên thì mới chạy được localhost.

Nếu cả hai ứng dụng chuyển sang màu xanh như hình dưới là đã khởi động thành công.



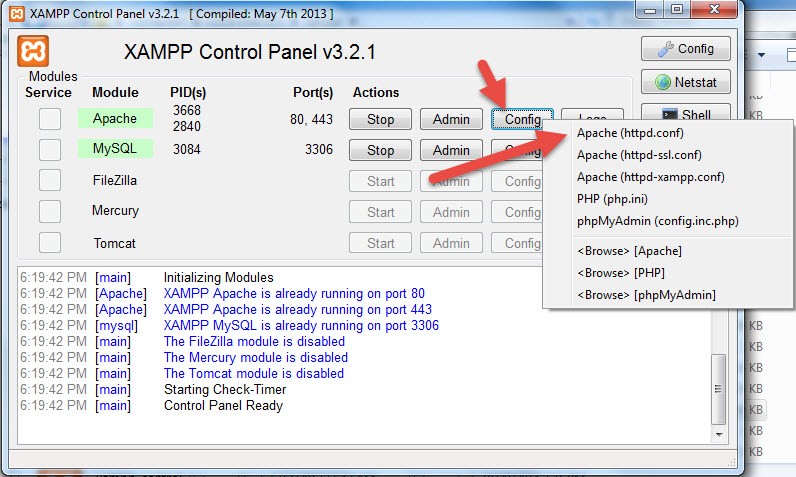
Để tắt Xampp chúng ta cần ấn vào 2 nút stop để tắt dịch vụ và sau đó click vào nút Quit (Lưu ý Xampp cần chạy ở quyền Admin (**Run as Administrator**) thì khi thoát sẽ không bị lỗi ghi vào các file)

2.6. Cách đổi cổng mạng cho Localhost

Mặc định Localhost sẽ sử dụng cổng 80, bởi vì khi chúng ta gõ tên miền như http://localhost thì tức là nó đã sử dụng cổng 80 để đọc các dữ liệu web trong localhost. Tuy nhiên nếu như chúng ta đã dùng cổng 80 cho một ứng dụng khác, hoặc đơn giản là không khởi động Apache được thì chúng ta nên thiết lập cho Apache trong Localhost sử dụng một cổng khác, như 8080 chẳng hạn.

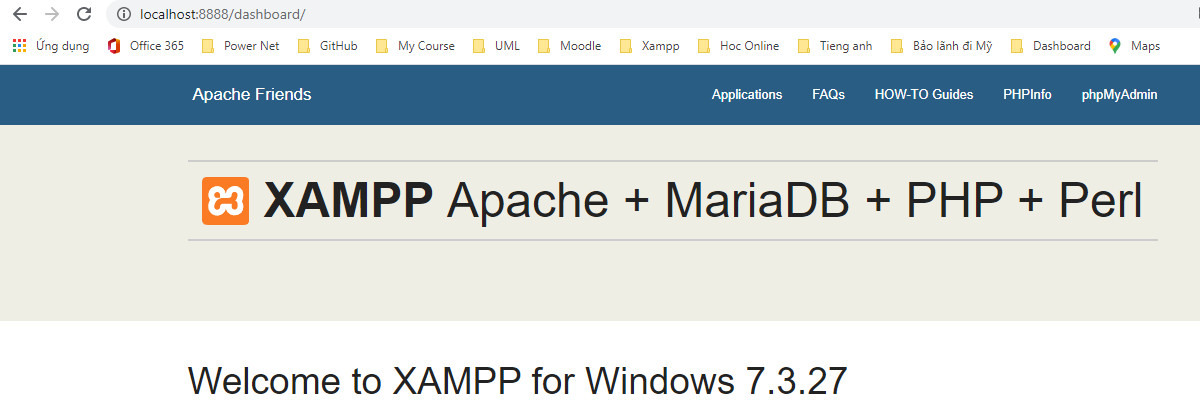
Trước khi đổi, mình cần các chúng ta lưu ý là sau khi đổi xong thì chúng ta phải truy cập vào website với tên miền http://localhost:8080 thay vì chỉ là http://localhost.

Để đổi cổng, chúng ta mở bảng điều khiển XAMPP lên và chọn nút Config của Apache, sau đó chọn Apache (httpd.conf).



Sau đó chúng ta tìm dòng: Listen 80 và đổi thành Listen 8888 chẳng hạn

Sau đó chúng ta Stop cái Apache và Start lại rồi thử truy cập vào localhost theo đường dẫn http://localhost:8888, nếu truy cập được thì chúng ta đã làm thành công. Và cũng nên lưu ý rằng, sau khi đổi cổng thì mỗi khi truy cập chúng ta phải sử dụng đường dẫn có kèm theo số cổng chúng ta vừa đổi sang vì mặc định nếu không điền thì nó sẽ sử dụng cổng 80.



2.7. Thao tác trên localhost

2.7.1. Làm việc với thư mục và tập tin

Một điều khá cơ bản nhưng rất quan trọng khi chúng ta làm việc với website mà bất kể là localhost hay hosting đó là hiểu cơ chế phân thư mục của Website.

Bây giờ chúng ta hãy vào thư mục C:\xampp\htdocs\ và tạo một thư mục tên “**hungpham**“, thư mục này sẽ chứa website của mình.

Như ở trên, đã hướng dẫn chúng ta tạo một thư mục tên là **hungpham** trong thư mục **htdocs** và chúng ta có thể sử dụng đường dẫn http://localhost/hungpham để chạy. Vậy bây giờ mình có thể nói, thư mục C:\xampp\htdocs\hungpham chính là thư mục gốc của tên miền http://localhost/hungpham

Bây giờ chúng ta thử copy một tập tin nào đó vào trong thư mục C:\xampp\htdocs\hungpham rồi chạy tên miền http://localhost/hungpham, chúng ta sẽ thấy nó liệt kê file mà chúng ta vừa copy vào.

Nếu chúng ta click vào file ảnh đó thì trình duyệt sẽ hiển thị ảnh với đường dẫn là http://localhost/hungpham/tên-tập-tin. Đây được xem là một link ảnh trong website của chúng ta.

Tương tự, hãy thử tạo một thư mục bất kỳ rồi copy một tập tin nào đó vào, thì bây giờ chúng ta có thể truy cập xem ảnh với đường dẫn là http://localhost/hungpham/tên-folder/tên-tập-tin.

Như vậy chúng ta có thể hiểu, cái đường dẫn trên website nó sẽ phân thứ cấp tùy theo cấu trúc thư mục và file trong đó.

***Tip: Chúng ta có thể đổi thư mục Webroot mặc định bằng cách***

*1. Hãy tạo 1 thư mục bất kỳ trên đĩa nào đó chẳng hạn F:/ltphp*

*2. Vào file httpd.conf*

*3. Tìm đến đoạn DocumentRoot và sửa thư mục webroot lại theo địa chỉ mới*

*DocumentRoot "F:\ltphp"*

*<Directory "F:\ltphp">*

*4. Stop các dịch vụ và khởi động lại Xampp*

2.7.2. Tạo cơ sở dữ liệu MySQL (Database)

Thao tác này chúng ta sẽ phải làm trong bước cài đặt Website trên localhost

Khi nhắc đến database, chúng ta phải nhớ là nó bao gồm 3 thành phần chính là:

* Tên user của database.
* Mật khẩu của user database.
* Tên database.
* Database Host (thường thì điền là localhost, bất kể là chúng ta dùng ở localhost hay host bình thường).

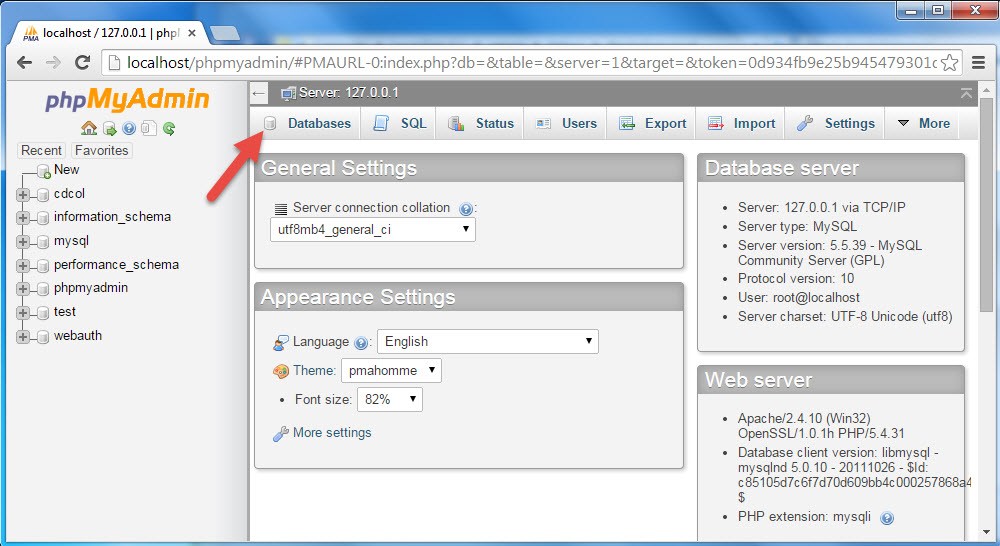
Chúng ta hãy tưởng tượng rằng, user của database sẽ có nhiệm vụ đọc và ghi dữ liệu vào database nên khi sử dụng các mã nguồn PHP, chúng ta phải khai báo cùng lúc cả user của database và tên database.

Đối với localhost, chúng ta không cần tạo user cho database mà sẽ sử dụng thông tin user như sau:

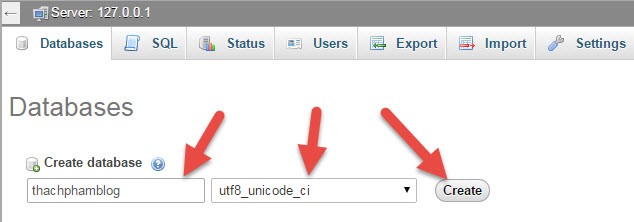
* Tên user database: root
* Mật khẩu: bỏ trống

Do vậy, chúng ta chỉ cần tạo database là đủ.

Để tạo database, chúng ta hãy truy cập vào localhost với đường dẫn http://localhost/phpmyadmin. Sau đó chúng ta nhấp vào menu Databases.



Sau đó ở phần Create databsae, chúng ta nhập tên database cần tạo vào ô Database name, phần Collation chúng ta hãy chọn là utf8\_unicode\_ci như hình dưới rồi ấn nút Create kế bên.



Tạo xong hãy nhìn bên menu tay trái, nếu nó xuất hiện tên database vừa tạo là thành công. Vậy bây giờ, chúng ta tạm có một databse với các thông tin như:

* **Database Host**: localhost
* **Database user**: root
* **Database password**: trống
* **Database name**: hungphamblog

3. Nền tảng của PHP

3.1. Cú pháp PHP

3.1.1. Bắt đầu PHP.

- Như đã giới thiệu ở bài trước thì trong file PHP có thể chứa cả code HTML,CSS,.. nên chính vì điều đó server phải có một quy tắc để phân biệt giữa code PHP và code khác.

Để PHP có thể chạy được thì bắt buộc tên của files phải có phần mở rộng là .php (Nếu muốn chỉnh sửa chúng ta phải có kiến thức server).

Tất cả các code PHP phải nằm trong Tag: **<?php ?>** hoặc **<?= ?>**

3.1.2. Comment trong PHP.

- Comment trong lập trình là các đoạn chú thích cho dòng code của chúng ta các đoạn chú thích này không ảnh hưởng đến tốc độ xử lý hay code của chúng ta.

- Để comment trong php các chúng ta có 2 cách:

* Comment một dòng( comment inline): Để comment 1 dòng trong PHP các chúng ta sử dụng cặp thẻ // noi dung comment.
* Comment nhiều dòng( comment multiline): Để comment nhiều dòng trong PHP các chúng ta sử dụng cặp thẻ /\* noi dung \*/

3.1.3. Hiển thị dữ liệu trong PHP.

Trong bất kỳ một ngôn ngữ nào thì nó đều có input và output và đối với PHP thì các chúng ta xuất dữ liệu dạng text, số ra màn hình với hàm echo theo các cách sau:

echo "noidung";

//hoặc

echo ("noidung");

//hoặc

echo 'noi dung';

3.1.4. Chương trình Helloworld.

Đầu tiên các chúng ta tạo một file hello.php trong thư mục htdocs của xampp và viết 1 trong 2 đoạn code sau:

<?php

echo "hello world!";

?

Hoặc sử dụng tag echo.

<?= 'Hello world'; ?>

Bây giờ các chúng ta khởi động xampp lên và chạy với đường dẫn http://localhost:8888/hello.php

và nếu làm đúng như trên thì Kết quả hiển thị sẽ là: hello world!

3.2. Biến, hằng, kiểu dữ liệu

3.2.1. Biến

- Biến là một định danh, nó dùng để lưu trữ các giá trị và nó có thể dùng phép gán để thay đổi giá trị. Cú pháp của biến bắt đầu bằng dấu đô la **$** và tiếp theo là các chữ, số, dấu gạch dưới. Ký tự đầu tiên của tên biến phải là chữ hoặc là dấu gạch dưới, không được là số.

*Ví dụ*

<?php

$sinhvien = ''; //đúng

$\_sinh\_vien = ''; //đúng

$sinh\_vien90 = ''; //đúng

$90sinhvien = ''; //sai

?>

*Lưu ý: PHP là một ngôn ngữ có phân biệt chữ hoa chữ thường. Ví dụ: $sinhvien khác $SinhVien*

- Gán giá trị cho biến chúng ta vẫn dùng phép gán = như các ngôn ngữ khác

$hello = 'Hello Word';

- Hiển thị giá trị của biến ra màn hình

<?php

$sinhvien = 'Nguyen Van A';

echo $sinhvien; // Xuất ra màn hình

?>

3.2.2. Hằng

Hằng cũng là một biến nhưng chúng ta không thể thay đổi giá trị của nó. Tuy nhiên cách khai báo biến và hằng số thì lại khác nhau.

Cú Pháp: define(‘ten\_hang’, ‘gia\_tri’);

*Trong đó:*

* define: hàm tạo biến hằng
* ten\_hang: là tên biến hằng
* gia\_tri: giá trị của hằng

<?php

/\* Tạo một hằng số có tên là SDT và gán giá trị cho nó là 0909090909\*/

define('SDT', '0909090909');

echo SDT; // xuất ra màn hình giá trị của hằng.

?

3.2.3. Kiểu dữ liệu

Trong php có tổng cộng 7 kiểu dữ liệu:

* Kiểu INT
* Kiểu Boolean
* Kiểu Số Thực (float, double)
* Kiểu Chuỗi
* Kiểu Mảng (array)
* Kiểu NULL
* Kiểu Đối Tượng (object)

***a. Kiểu INT:*** Chữ INT là viết tắt của chữ INTEGER, là một kiểu dữ liệu dạng số và có thể ở viết ở nhiều cơ số khác nhau.

*Ví dụ 1*

<?php

$thap\_phan = 123; // Số thập phân

$so\_am = -123; // Số âm

$bat\_phan = 0123; // số bát phân

$thap\_luc\_phan = 0x1A; // và số thập lục phân

?>

Kiểu số INT Chúng ta không dùng dấu nháy để bao quanh nó và kích thước của kiểu INT là 32bit. Trong PHP không hỗ trợ nhiều kiểu Unsigned Integer (Số nguyên dương) nên nếu chúng ta sử dụng vượt quá giới hạn của nó thì mặc nhiên trình biên dịch sẽ hiểu đây là kiểu Float (số thực), tuy nhiên không phải lúc nào điều này cũng đúng cho trường hợp số dương.

*- Khai báo biến kiểu INT*

Để khai báo một biến kiểu INT chúng ta sẽ gán giá trị cho nó là số nguyên (kể cả số âm).

*Ví dụ 2*

<?php

$tuoi = 12; // biến $tuoi là kiểu INT có giá trị = 12

?>

*- Ép dữ liệu sang kiểu INT*

Cú Pháp: (int)$ten\_bien;

*Ví dụ 3*

<?php

$tuoi = '98'; // biến tuổi là một chuỗi có giá trị bằng '98'

$tuoi = (int)$tuoi; // lúc này biến $tuoi là một kiểu int có giá trị 98

?>

Việc chuyển đổi này trong PHP đôi khi lại không cần thiết vì các kiểu dữ liệu trong php tự động chuyển các biến sang các kiểu thích hợp để thực hiện phép tính, tuy nhiên sau khi thực hiện tính toán thì biến đó sẽ tự chuyển lại kiểu dữ liệu ban đầu.

*Ví dụ 4*

<?php

$a = '123'; // Biến $a là kiểu chuỗi có giá trị bằng '123'

$b = 123; // Biến $b là kiểu INT có giá trị bằng 123

$c = $a + $b; // Biến C là kết quả của phép toán $a + $b và sẽ có giá trị là 246 nên nó là kiểu INT

var\_dump(is\_int($c)); // hàm is\_int($tenbien) dùng để kiểm tra một biến có phải là kiểu INT hay không

var\_dump(is\_int($a)); // kết quả là false vì biến $a là kiểu string

?>

Trong ví dụ này các chúng ta thấy biến $a là chuỗi còn biến $b là số, khi ta cộng 2 biến lại thì các biến sẽ tự động chuyển sang kiểu số INT thích hợp để cộng, và kết quả là kiểu INT gán vào biến $c. Để kiểm tra chúng ta dùng dòng lệnh var\_dump(is\_int($c));để xuất ra màn hình kết quả kiểm tra.

*Ví dụ 5*

<?php

$a = 'a123'; // biến $a có giá trị là chuỗi 'a123'

$a = (int)$a; // chuyển $a sang kiểu INT

echo $a; // kết quả xuất ra màn hình là số 0

?>

Chạy đoạn lệnh này các chúng ta sẽ thấy kết quả ra số 0. Tại sao? vì chúng ta thấy biến $a có ký tự đầu tiên không phải ở dạng số nên nó sẽ tự động cắt bỏ tất cả những ký tự đằng sau ký tự a nên chuỗi này rỗng, mà giá trị rỗng chuyển sang kiểu INT có giá trị bằng không.

*Ví dụ 6*

<?php

$a = '123a'; // biến $a có giá trị là chuỗi '123a'

$a = (int)$a; // chuyển $a sang kiểu INT

echo $a; // kết quả xuất ra màn hình là số 123

?>

Chạy đoạn code này kết quả xuất ra màn hình là 123, cũng như giải thích ở trên nó sẽ xóa các ký tự bắt đầu từ ký tự a nên chuỗi sẽ còn ’123′, chuyển sang kiểu INT thành 123.

*- Kiểm tra dữ liệu có phải kiểu INT.*

Để kiểm tra một biến nào đó có phải kiểu **INT** không chúng ta dùng 2 hàm **is\_int($bien)** hoặc **is\_integer($bien)**. kết quả trả về giá trị **True** nếu là kiểu INT và **False** nếu không phải kiểu INT.

***b. Kiểu boolean***

Đây là một kiểu dữ liệu đơn giản nhất trong các kiểu dữ liệu trong PHP, nó chỉ chứa 2 giá trị là đúng hoặc sai (TRUE hoặc FALSE). Để tạo biến kiểu boolean thì chúng ta gán giá trị cho nó là TRUE hoặc FALSE. Lưu ý TRUE, FALSE không phân biệt hoa thường, nghĩa là chúng ta gõ thế nào cũng được miễn là đúng.

*Ví dụ 1*

<?php

$is\_admin = false; // biến $admin là kiểu boolean có gái trị là false

?>

*- Ép dữ liệu sang kiểu boolean.*

Tương tự như kiểu INT chúng ta sử dụng (bool) hoặc (boolean) để ép kiểu sang kiểu bool. Như vậy trong PHP thì bool và boolean là 2 từ khóa có cùng một ý nghĩa.

*Ví dụ 2*

<?php

$bool = 1; // biến $bool là kiểu int

$bool = (bool)$bool; // lúc này biến $bool sẽ có kiểu boolean

// Hoặc

$bool = (boolean)$bool; // lúc này biến $bool sẽ có kiểu boolean

?>

Các ký tự 0, ký tự trống và null đều được quy về giá trị FALSE, các ký tự còn lại quy về TRUE. Việc chuyển đổi này đôi khi cũng không cần thiết vì php tự xem xét giá trị và quy về TRUE hay FALSE.

*Ví dụ 3*

<?php

$a = 123; // TRUE

$b = 0; // FALSE

$c = '0'; // FALSE

$d = 'a123b' // TRUE

$e = null; // FALSE

$f = ''; // FALSE

?>

*- Kiểm tra một biến kiểu boolean*

Để kiểm tra một biến có phải kiểu boolean chúng ta dùng hàm **is\_bool($bien)**;để kiểm tra, kết quả của hàm này trả về TRUE nếu là kiểu bool, ngược lại là false nếu không phải kiểu bool.

***c. Kiểu số thực***

Hiểu một cách nôm na kiểu số thực là những số có phần dư, còn kiểu INT là những số không dư phần nào, như số 1.234 là kiểu số thực, 1234 là kiểu số nguyên (INT). Kích cỡ của nó phụ thuộc xác định phụ thuộc vào từng platform nhưng giá trị lớn nhất xấp xỉ 1.8e308, các kiểu dữ liệu trong php của kiểu số thực gồm có kiểu float, double.

*Ví dụ 1*

<?php

$a = 1.234; // Kiểu số thực

?>

*- Ép dữ liệu sang kiểu số thực*

Chúng ta dùng (float), (double) để chuyển kiểu dữ liệu sang số thực cho một biến

*Ví dụ 2*

<?php

$a = 123; // biến $a kiểu int

$a = (float)$a; // Biến $a lúc này kiểu số thực (float)

$a = (double)$a; // Biến $a lúc này kiểu ố thực (double)

?>

*- Kiểm tra một biến kiểu số thực*

Để kiểm tra một biến phải kiểu số thực không chúng ta dùng hàm **is\_float($bien)** để kiểm tra cho kiểu **float**, **is\_double($bien)** để kiểm tra cho kiểu **double**. Kết quả 2 hàm này trả về TRUE nếu đúng, FALSE nếu sai.

***d. Kiểu chuỗi***

Kiểu chuỗi gồm 2 kiểu: kiểu string (chuỗi) và char (ký tự), mỗi ký tự là 1 byte và là một trong 256 ký tự khác nhau, để khai báo báo các chúng ta chỉ việc khai báo một biến và gán giá trị chuỗi cho nó, chuỗi phải được bao quanh bằng dấu nháy đơn ‘ hoặc dấu nháy kép “. Ép kiểu cũng như trên ta dùng (string) để chuyển sang kiểu chuỗi.

*Ví dụ 1*

<?php

$a = 123; // khai báo biến $a kiểu int có giá trị 123

$a = (string)$a; //Chuyển biến $a thành kiểu chuỗi và có giá trị là '123'

?>

*- Kiểm tra một biến kiểu string*

Để kiểm tra một biến kiểu chuỗi (string) ta dùng hàm is\_string($bien), kết quả hàm này trả về TRUE nếu đúng và FALSE nếu không đúng.

***e. Kiểu giá trị null***

Đây là kiểu đặc biệt trong PHP và cũng như các ngôn ngữ lập trình khác, nó mang giá trị rỗng.

Lúc chúng ta khởi tạo một biến và chúng ta gán = NULL thì sẽ hệ thông sẽ không tốn bộ nhớ để lưu trữ, nên việc sử dụng nó rất có lợi.

Kiểu NULL khi ép kiểu sang kiểu INT thì bằng 0, khi ép kiểu sang kiểu chuỗi thì = rỗng, và khi ép sang kiểu boolean thì mang giá trị FALSE.

*Ví dụ*

<?php

$a = null; // Khởi tạo biến $a và gán giá trị null

$b = (int)$a; // Biến $b có giá trị là ( 0 )

$c = (string)$a; // Biến $c có giá trị rỗng ( '' )

$d = (bool)$a; // Biến $d có giá trị FALSE

?>

*- Kiểm tra một biến có giá trị null*

Để kiểm tra một biến có giá trị null hay không ta dùng hàm is\_null($bien). Biến này trả về TRUE nếu đúng và FALSE nếu không đúng.

3.3. Kiểu mảng, kiểu đối tượng

***a. Kiểu mảng***

Mảng là danh sách các phần tử có cùng kiểu dữ liệu và nó là một trong các kiểu dữ liệu trong php có độ phức tạp tính toán cao. Có 2 loại mảng là mảng một chiều hoặc mảng nhiều chiều. Riêng với PHP thì các phần tử của mảng có thể không cùng kiểu dữ liệu, và các phần tử của mảng được truy xuất thông qua các chỉ mục(vị trí) của nó nằm trong mảng.

*- Khởi tạo và truy xuất các phần tử trong mảng*

Để khai báo mảng ta dùng cú pháp sau:

<?php

$ten\_mang = array(); // khởi tạo một mảng gán vào biến $ten\_mang

?>

*Ví dụ:* Giả sử có 2 sinh viên là Nguyễn Văn A và Nguyễn Văn B, giờ chúng ta sẽ khởi tạo một mảng $sinhvien để lưu 2 sinh viên này lại.

*NOTE: Chúng ta dùng hàm var\_dump($mang); để in ra các phần tử của mạng để test trong quá trình học. Hàm này có thể sử dụng được tất cả các kiểu dữ liệu trong php.*

*Cách 1:*

<?php

$sinhvien = array('Nguyễn Văn A', 'Nguyễn Văn B');

print\_r($sinhvien);

?>

*Cách 2:*

<?php

$sinhvien = array(

0 => 'Nguyễn Văn A',

1 => 'Nguyễn Văn B'

);

print\_r($sinhvien);

?>

*Cách 3:*

<?php

$sinhvien = array();

$sinhvien[0] = 'Nguyễn Văn A';

$sinhvien[1] = 'Nguyễn Văn B';

print\_r($sinhvien);

?>

*Cách 4:*

<?php

$sinhvien = array();

$sinhvien[] = 'Nguyễn văn A';

$sinhvien[] = 'Nguyễn Văn B';

print\_r($sinhvien);

?>

*Cách 5: giống cách 1 chỉ khác dùng [ ]*

$sinhvien=['Nguyễn văn A', 'Nguyễn văn B'];

*- Mảng có chỉ mục*

Là mảng có các phần tử được định danh một chỉ mục (kiểu số) và bắt đầu bằng số 0 và phần tử cuối cùng có chỉ mục là (n-1), trong đó n là tổng số phần tử của mảng. Điều này có nghĩa nếu mảng của chúng ta có 10 phần từ thì lần lượt các vị trí phần tử trong mảng là: [0] – [1] – [2] – [3] – [4] – [5] – [6] – [7] – [8] – [9]

Với 4 cách khai báo như trên:

Với cách 1: Chúng ta khởi tạo một mảng và gán trực tiếp 2 phần từ vào, vì mảng bắt đầu từ 0 nên nó tự hiểu phần tử đầu tiên có chỉ mục =0, và phần tử thứ 2 = 1.

Với cách 2: Chúng ta khởi taọ một mảng và gán trực tiếp 2 phần tử vào, nhưng lúc gán chúng ta có ghi rõ các chỉ mục cho từng phần tử.

Với cách 3: Ban khởi tạo một mảng rỗng. sau đó chúng ta dùng 2 lệnh để gán 2 phần tử vào, mỗi lệnh gán chúng ta có chỉ rõ chỉ mục.

Với cách 4: Chúng ta khởi tạo một mảng rỗng, sau đó chúng ta dùng 2 lệnh gán 2 phần tử vào nhưng chúng ta lại không chỉ rõ chỉ mục, lúc này PHP sẽ kiểm tra thấy mảng đang rỗng nên phần tử đầu tiên nó sẽ mặc định gán chỉ mục = 0, và phần tử tiếp theo sẽ bằng phần tử trước nó + 1 tức là sẽ = 1.

Để truy xuất các phần tử của mảng chỉ mục ta dùng cú pháp sau: **$tenmang[$index]**; trong đó **$index** là chỉ mục chúng ta muốn lấy.

*Ví dụ*

<?php

$sinhvien = array(

0 => 'Nguyễn Văn A',

1 => 'Nguyễn Văn B'

);

echo $sinhvien[0]; // Xuất ra màn hình phần tử 0 => Nguyễn Văn A

echo $sinhvien[1]; // Xuất ra màn hình phần tử 1 => Nguyễn Văn B

?>

*- Mảng kết hợp*

Là Mảng có các phần tử được định danh bằng một cái tên và đương nhiên vị trí các phần tử sẽ không có thứ tự.

*Ví dụ*

<?php

$sinhvien = array(

'sinhvien\_a' => 'Nguyễn Văn A',

'sinhvien\_b' => 'Nguyễn Văn B'

);

print\_r($sinhvien);

?>

Tương tự như những ví dụ ở phần mảng có chỉ mục, ta có thêm cách khai báo.

<?php

$sinhvien = array();

$sinhvien['sinhvien\_a'] = 'Nguyễn Văn A';

$sinhvien['sinhvien\_b'] = 'Nguyễn Văn B';

print\_r($sinhvien);

?>

*Xét ví dụ sau:*

<?php

$sinhvien = array();

$sinhvien['sinhvien\_a'] = 'Nguyễn Văn A';

$sinhvien['sinhvien\_b'] = 'Nguyễn Văn B';

print\_r($sinhvien);

?>

Trong ví dụ này điều đặc biệt là lúc gán sinh viên Nguyễn Văn B ta không truyền tên cho nó mà chỉ dùng dấu [] để thêm vào. Điều gì sẽ xảy ra? Câu trả lời như sau: Trình biên dịch sẽ chạy dòng thứ 1 khởi tạo một mảng rỗng, dòng thứ 2 thêm một phần tử cho mảng với tên sinhvien\_a, dòng thứ 3 nó sẽ thấy ko có truyền chỉ mục nên nó sẽ lưu giá trị Nguyễn Văn B dưới dạng chỉ mục. Nó xét thấy trong mảng này chưa có chỉ mục nào (vì dòng 2 truyền dạng kết hợp), nên sinh viên Nguyễn Văn B sẽ được lấy chỉ mục = 0.

Việc truy xuất các phần tử trong mảng kết hợp cũng tương tự như mảng chỉ mục ta dùng cú pháp sau: **$tenmang[$name]**, trong đó **$name** là tên của phần tử chúng ta muốn lấy ra.

*Ví dụ*

<?php

$sinhvien = array();

$sinhvien['sinhvien\_a'] = 'Nguyễn Văn A';

$sinhvien['sinhvien\_b'] = 'Nguyễn Văn B';

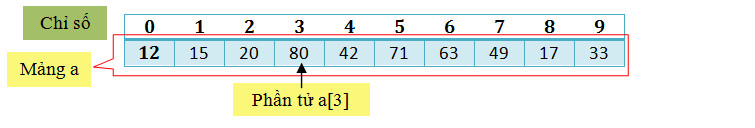
echo $sinhvien['sinhvien\_a']; // xuất ra màn hình sinh viên Nguyễn Văn A

echo $sinhvien['sinhvien\_b']; // xuất ra màn hình sinh viên Nguyễn Văn B

?>

Mảng một chiều

Tất cả những ví dụ ở trên gọi là mảng 1 chiều (gồm mảng 1 chiều chi mục, mảng một chiều kết hợp)

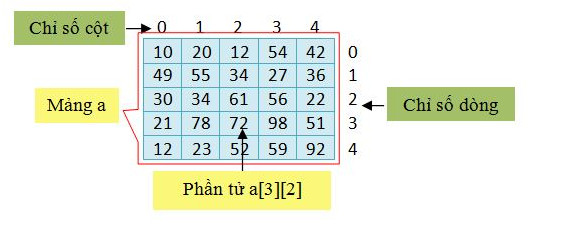


Mảng nhiều chiều

Là mảng có nhiều chỉ mục cho từng phần tử, ví dụ mảng 2 chiều thì mỗi phần tử có 2 chỉ muc, 3 chiều thì mỗi phần tử có 3 chỉ mục, …

Mảng nhiều chiều thực chất cũng chỉ là mảng 1 chiều nhưng được thể hiện dưới dạng nhiều chiều.

Xem hình minh họa mảng 2 chiều sau được biểu hiện bằng số dòng và số cột (nghĩa là 2 chiều giống trong hình học không gian 2 chiều), mỗi phần tử sẽ được định vị trí ở điểm giao nhau của chỉ số cột và dòng hiện tại.



*Ví dụ: thêm phần tử trong mảng 2 chiều*

<?php

$a = array();

$a[0][1] = 123;

$a[0][2] = 321;

?>

*Ví dụ: truy xuất phần tử trong mảng 2 chiều*

<?php

$a = array();

$a[0][1] = 123;

$a[0][2] = 321;

echo $a[0][1]; // in ra giá trị 123

echo $a[0][2]; // in ra giá trị 321

?>

*- Kiểm tra một biến kiểu mảng*

Để kiểm tra một biến có phải kiểu mảng (array) không ta dùng hàm **is\_array($bien)**, hàm này trả về TRUE nếu đúng và FALSE nếu không đúng.

***b. Kiểu Object***

3.4. Toán tử

3.4.1. Biểu thức

*Biểu thức là một tổ hợp các toán hạng và toán tử*. Toán tử thực hiện các thao tác như cộng, trừ, nhân, chia, so sánh, … Toán hạng là những biến hay là những giá trị mà các phép toán được thực hiện trên nó. Ví dụ $a + $b thì $a và $b được gọi là toán hạng, dấu + được gọi là toán tử, cả 2 kết hợp lại thành một biểu thức ($a + $b).

Mỗi biểu thức chỉ có một giá trị nhất định. Ví dụ ta có biểu thức ($a + $b) thì biểu thức này có giá trị là tổng của $a và $b.

*Ví dụ*

$ketqua = $a - $b;

$ketqua = 7 + 6;

$ketqua = 3\*$x + 4\*$y;

Toán tử gán (Assignment Operator):

Đây là toán tử thông dụng nhất trong mọi ngôn ngữ, ta dùng dấu = để gán giá trị cho một biến bất kỳ nào đó

$a = 12;

Nhiều biến có thể được gán cùng một giá trị qua một câu lệnh đơn gọi là gán liên tiếp.

$a = $b = $c = $d = 12;

Biểu thức số học

Các phép toán thường được thực hiện theo một thứ tự cụ thể gọi là độ ưu tiên để đưa ra giá trị cuối cùng (sẽ đề cập đến sau). Các biểu thức số học trong các ngôn ngữ được thể hiện bằng cách sử dụng các toán tử số học (+ - \* / % )cùng với các toán hạng dạng số hoặc ký tự (biến). Những biểu thức này gọi là biểu thức số học.

$ketqua = $a + $b/2;

$ketqua = $a / 7;

$ketqua = $a + ($b = 5 + 6);

Như ta thấy trên toán hạng có thể là hằng, biến hay kết hợp cả 2, và mỗi biểu thức có thể kết hợp của nhiều biểu thức con. Định nghĩa hằng và biến như thế nào chúng ta đã đề cập ở Biến Và Hằng Số Trong PHP.

3.4.2. Toán tử quan hệ

Toán tử quan hệ cũng là một định nghĩa trong bài toán tử và biểu thức trong php, được dùng để kiểm tra mối quan hệ giữa hai biến hay giữa một biến và một hằng số. Ví dụ kiểm tra 2 biến $a và $b xem biến nào lớn hơn ta làm như sau: ($a > $b) và kết quả của biểu thức này sẽ trả về kiểu boolean TRUE hoặc FALSE.

Trong phần trước đã đề cập ở nội dung các kiểu dữ liệu trong php, kiểu boolean có giá trị là FALSE cho tất cả các giá trị bằng 0, ký tự rỗng hoặc null. Để so sánh 2 toán hạng ta làm như sau ($a == $b). Các chúng ta lưu ý rằng phép toán mà có 1 dấu bằng là phép gán, còn 2 dấu bằng là phép so sánh.

Các phép toán quan hệ trong PHP cũng giống như Javascript

> Lớn hơn

>= Lớn hơn hoặc bằng

< Nhỏ thua

<= Nhỏ thua hoặc bằng

== Bằng

!= Khác

*Ví dụ*

$a = 12; // Biến $a kiểu INT có giá trị = 12

$t = ($a == 12); // Biến $t có giá trị là TRUE vì biểu thức (12 == 12) đúng

$t = ($a > 12); // Biến $t có giá trị là FALSE vì biểu thức (12 > 12) sai

$t = ($a >= 12); // Biến $t có giá trị TRUE vief biểu thức (12 >= 12) đúng

$t = ($a != 12); // Biến $t có giá trị FALSE vì biểu thức (12 != 12) sai

*Lưu ý: Giống như Javascript chúng ta có phép toán === dùng để so sánh giá trị giữa các biến và hằng đúng theo giá trị và kiểu dữ liệu của nó, nếu ta sử dụng 2 dấu bằng == để so sánh ($a == $b) thì lúc này $a và $b chỉ so sánh giá trị và trả về true nếu $a bằng $b.*

3.4.3. Toán tử Logic

**&&** (phép AND)

**||** (phép OR)

**!** ( phép NOT)

*Lưu ý: giống Javascript*

*Ví dụ*

$a = 100;

$b = 200;

$tong = $a + $b;

$check = ($a < $b) && ($tong > 200);

Độ ưu tiên toán tử Logic

*Ví dụ:* Tính độ ưu tiên sau ( 7 > 5 && !(-5 > 1) || 10 == 10 ) **(1)**

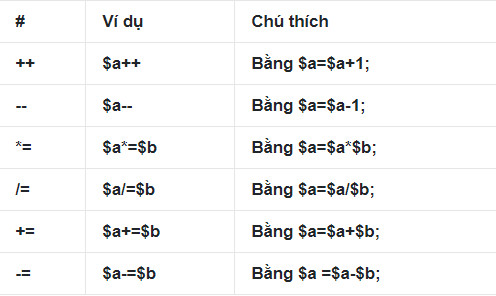
Bước 1: trong biểu thức này có một phép toán NOT đó là !(-5 > 1) nên ta tính trước phép này. Trong PHP cũng như các ngôn ngữ lập trình khác biểu thức !(biểu thức)cùng ý nghĩa với biểu thức (biểu thức) == false nên biểu thức ở trên ta biến đổi thành ( (-5 > 1 ) == false ). Biểu thức này trả về giá trị TRUE vì (-5 > 1) là sai.

Bước 2: Lấy kết quả bước 1 ta viết lại biểu thức (1) như sau: ((7>5) && true || 10 == 10) Theo độ ưu tiên thì ta tính phép AND trước tức là tính ((7 > 5) && true) trước. Phép tính này trả về TRUE bởi vì (7 > 5) = true suy ra true && true => true

Bước 3: Bước này lấy kết quả ở bước 2 ta ráp vào thì biểu thức (1) như sau: (true || 10 == 10). Phép OR sẽ trả về TRUE nếu một trong 2 biểu thức có giá trị true => biểu thức (1) là biểu thức có giá trị TRUE.

3.4.4. Toán tử kết hợp

Toán tử này thực ra là các biểu thức rút gọi của biểu thức (giống Javascript)



3.5. Các cấu trúc điều khiển

3.5.1. Câu lệnh if

Câu lệnh if cho phép ta đưa ra các quyết định dựa trên việc kiểm tra điều kiện nào đó đúng (true) hay sai (false). Cú pháp như sau:

**if ( $btdieukien )**

**{**

***// Các Câu Lệnh***

**}**

*Lưu ý: câu lệnh if của PHP giống Javascript*

*Ví dụ:*

$so\_can\_kiem\_tra = 12; (1)

$so\_du = $so\_can\_kiem\_tra % 2; (2)

if ($so\_du == 0){ (3)

echo 'Số '.$so\_can\_kiem\_tra.' Là Số Chẵn';

}

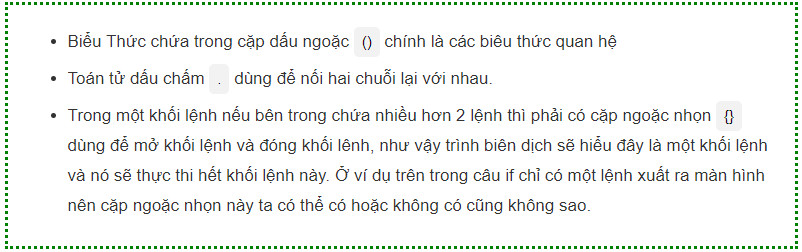
*Giải thích:*

Dòng thứ 1: Gán cho biến $so\_can\_kiem\_tra giá trị = 12

Dòng thứ 2: Chia biến $so\_can\_kiem\_tra / 2 và lấy số dư của phép chia (toán tử % dùng để chia lấy số dư, nếu chúng ta chưa biết toán tử này vui lòng đọc lại bài toán tử và biểu thức trong php).

Dòng thứ 3: Kiểm tra số dư có bằng 0 hay không? Nếu bằng không thì nó sẽ chạy dòng lệnh bên trong thẻ mở { và thẻ đóng }. Nội dung câu lệnh bên trong sẽ xuất ra màn hình thống báo là số chẵn.

Chạy chương trình này kết quả sẽ xuất ra là "Số 12 Là Số Chẵn". Bây giờ giả sử ta đổi giá trị 12 thành 13 thì kết quả sẽ không xuất ra màn hình vì số 13 chia cho 2 sẽ dư = 1, mà điều kiện để xuất ra câu thông báo là số dư phải bằng 0 => không đúng điều kiện nên trình biên dịch sẽ không chạy vào đoạn code bên trong lệnh if.



3.5.2. Câu lệnh if else

Lệnh **if** dùng để kiểm tra một điều kiện có đúng hay không? Giả sử trường hợp ngược lại điều kiện không đúng thì sẽ thực hiện điều gì ? để giải đáp câu hỏi này ta sẽ nghiên cứu đến lệnh **if else** trong php

**if ( $btdieukien )**

**{**

***// Các Câu Lệnh 1***

**} else {**

***// Các Câu Lệnh 2***

**}**

*Lưu ý: câu lệnh if else của PHP giống Javascript*

*Ví dụ: kiểm tra năm nay là năm chẵn hay năm lẽ, xuất ra màn hình kết quả chẵn hay lẽ*

$nam = 2014;

$so\_du = $nam % 2;

if ($so\_du == 0){

echo 'Năm ' . $nam . ' Là Năm Chẵn';

}

else{

echo 'Năm ' . $nam . ' Là Năm Lẻ';

}

3.5.3. Kết hợp nhiều câu lệnh if else trong php

Trong thực tế không phải lúc nào cũng chỉ có 2 điều kiện mà sẽ có hàng chục điều kiện khác nhau, lúc này ta phải kết hợp nhuần nhuyễn giữa 2 lệnh if và else để xử lý. Theo kiểu

if (**$btdk1**) { ***//nhóm lệnh 1*** } else if (**$btdk2**) { ***//nhóm lệnh 2*** } else if (**$btdk3**) { ***//nhóm lệnh 3*** } else { ***//nhóm lệnh khác*** }

*Lưu ý: câu lệnh if else if của PHP giống Javascript*

*Ví dụ: Nhập vào một màu và kiểm tra:*

* *Nếu là màu xanh thì xuất ra màn hình dòng chữ “Đây Là Màu Xanh”.*
* *Nếu là màu đỏ thì xuất ra dòng chữ “Đây là màu đỏ”.*
* *Nếu là màu vàng thì xuất ra dòng chữ “Đây là màu vàng”.*
* *Các màu còn lại thì xuất ra dòng chữ “Các màu khác”.*

***Code:***

$mau = 'màu xanh';

if ($mau == 'màu xanh'){

echo 'Đây là màu xanh';

}

else if ($mau == 'màu đỏ')

{

echo "Đây là màu đỏ";

}

else if ($mau == 'màu vàng'){

echo 'Đây là màu vàng';

}

else{

echo 'Các màu khác';

}

3.5.4. Câu lệnh if lồng

Đây là phần khó nhất trong tất cả các vấn đề của câu lệnh if else trong php, ở bên trên ta chỉ chạy câu lệnh if else một tầng, ở phần này ta sẽ nghiên cứu đến if else nhiều tầng lồng nhau, có nghĩa là câu if con nằm trong câu if cha. Theo dạng

if ($btđk\_cha)

{

*// Các câu lệnh thuộc về biểu thức cha;*

if ($btđk\_con){

*// Các câu lệnh thuộc về biểu thức con;*

}

}

*Ví dụ: Kiểm tra sô nhập vào có phải là số chẵn hay không? Nếu là số chẵn thì kiểm tra tiếp số đó có lớn hơn100 hay không, nếu lớn hơn 100 thì xuất ra màn hình “Số chẵn và lớn hơn 100″, ngược lại xuất ra màn hình “Số chẵn và nhỏ hơn 100″.*

$so = 80; // Nhập vào số 80

if ($so % 2 == 0) // Nếu số dư khi chia cho 2 = o, tức là số chẵn ta sẽ chạy tiếp câu lệnh bên trong

{

if ($so > 100){ // Nếu số lớn hơn 100 thì chạy lệnh bên trong

echo 'Số chẵn và lớn hơn 100';

}

else if ($so < 100){ // Ngược lại nếu số nhỏ hơn 100 thì chạy lệnh bên trong

echo 'Số chẵn và nhỏ hơn 100';

}

}

BÀI TẬP

Bài 1: Kiểm tra một số có chia hết cho 7 hay không?

Bài 2: Kiểm tra điểm học sinh có đủ điều kiện tốt nghiệp

Bài 3: Xếp hạng năng lực của học sinh cuối năm

3.5.5. Câu lệnh switch case

Câu lệnh switch trong php cho phép ta đưa ra quyết định có nhiều cách lựa chọn để giá trị của biểu thức truyền vào. Nếu giá trị biểu thức truyền vào trùng với các giá trị biểu thức điều kiện thì các câu lệnh bên trong biểu thức điều kiện sẽ được thực hiện.

*Cú pháp:*

switch ($variable) {

case $value\_1:

// chuỗi câu lênh

break;

case $value\_2:

// chuỗi câu lệnh

break;

default:

// chuỗi câu lệnh

break;

}

Trong đó lệnh switch, case và default là các từ khóa trong PHP. Các chuỗi câu lệnh có thể là lệnh đơn (1 lệnh) hoặc lệnh ghép (kết hợp nhiều lệnh) và không cần đặt trong cặp dấu ngoặc nhọn {}. Ở mỗi dòng lệnh lựa chọn (case) nó sẽ so sánh xem biến truyền vào $variable có bằng với biến điều kiện $value\_1, $value\_2 hay không, nếu trùng với case nào thì những câu lệnh bên trong case đó sẽ được thực hiện, đồng thời dòng lệnh break sẽ kết thúc câu lệnh switch. Nếu không có lựa chọn (case) nào đúng thì mặc định nó sẽ chạy vào chuỗi dòng lệnh trong default, lệnh default trong mệnh đề rẻ nhánh switch có thể có hoặc không.

Giá trị ở case chỉ chấp nhận các kiễu dữ liệu string, INT, boolean, null, foat hoặc là một biểu thức có kết quả trả về một trong năm loại dữ liệu đó và toán tử quan hệ so sánh trong switch luôn luôn là ==.

*Ví dụ: Viết chương nhập vào một số, dùng lệnh rẻ nhánh switch kiểm tra số đó nếu:*

* Bằng 0 thì xuất dòng lênh “Số không”
* Bằng 1 thì xuất dòng lệnh “Số một”
* Bằng 2 thì xuất dòng lệnh “Số hai”
* Bằng 3 thì xuất dòng lệnh “Số ba”
* Bằng 4 thì xuất dòng lệnh “Số bốn”
* Các số còn lại xuất hiện dòng lệnh “Không tìm thấy”

$number = 1;

switch ($number)

{

case 0 :

echo 'Số không';

break;

case 1:

echo 'Số một';

break;

case 2:

echo 'Số hai';

break;

case 3:

echo 'Số ba';

break;

case 4 :

echo 'Số bốn';

break;

default:

echo 'Không tìm thấy';

break;

}

3.5.6. Vòng lặp for

*Cú pháp*

for (bienkhoitao; dieukienthucthi; buocnhay) {

// code...

}

*Trong đó:*

* bienkhoitao: là giá trị khởi tạo ban đầu của vòng lặp.
* dieukienthucthi: là điều kiện mà vòng lặp được phép chạy(chú ý: Nếu bạn muốn lặp vô tận thì có thể bỏ trống).
* buocnhay: là khoảng đệm nhảy của mỗi vòng lặp.

*Lưu ý: câu lệnh for của PHP giống Javascript*

*Ví dụ*

<?php

for ($i = 0; $i <= 10; $i++) {

echo 'Tôi đang chạy đến số ' . $i . '<br/>';

}

?>

3.5.7. Vòng lặp foreach

Đây là một loại vòng lặp rất là hay dùng để lặp mảng, object và chính vì điều này nên nó sẽ được sử dụng rất nhiều về sau.

*Cú pháp*

foreach (**$variable** as **$key** => **$value**) {

// code...

}

hoặc nếu không muốn lặp key của mảng, object. Ta dùng cú pháp

foreach ($variable as $value) {

// code...

}

*Trong đó:*

* **$variable**: Là biến mà các bạn muốn lặp (có thể là array, object).
* **$key**: Là các key của mảng hoặc properties của object.
* **$value**: Là giá trị của mảng,object.

*Ví dụ*

<?php

//Khởi tạo mảng có key tự tăng.

$data = [

'A',

'B',

'C',

'D'

];

// Lặp cả key lẫn value của mảng

foreach ($data as $key => $value) {

echo "Đây là phần tử có key = $key và có giá trị= $value <br/>";

}

// Tạo khoảng phân cách để dễ phân biệt

echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ <br/>";

// Lặp value của mảng

foreach ($data as $value) {

echo "Giá trị phần tử= $value <br/>";

}

?>

3.5.8. For và foreach lồng

Cũng giống như trong câu lệnh rẽ nhánh, vòng lặp cũng hoàn toàn có thể lồng nhau được.

*Ví dụ for 1: vẽ tam giác vuông bằng vòng lặp for*

<?php

for ($i = 0; $i <= 10; $i++) {

for ($j = $i; $j <= 10; $j++) {

echo "\*";

}

echo "<br>";

}

?>

*Ví dụ for 2:*

<?php

$sv=['Lan','Hùng','Ánh','Sơn','Thuỷ','Minh'];

$sosv=count($sv);

for ($i = 0; $i <$sosv ; $i++) {

echo $sv[$i].'<br>';

}

?>

*Ví dụ foreach: duyệt mảng 2 chiều với vòng foreach lồng nhau*

<?php

$data = [0 => 1, 1 => [3 => 4, 5 => 6]];

foreach ($data as $val) {

if (is\_array($val)) {

foreach ($val as $v) {

echo $v . "<br/>";

}

} else

echo $val . "<br>";

}

?>

Hoặc viết đầy đủ

<?php

$data = [0 => 'a', 1 => [0=> 'd', 1=>'e',3 => 'b', 5 => 'c']];

foreach ($data as $k => $val) {

if (is\_array($val)) {

foreach ($val as $c=> $v) {

echo 'Phan tu '.$k.$c.' = '.$v . "<br/>";

}

} else

echo 'Phan tu '.$k.' = '.$val . "<br>";

}

?>

BÀI TẬP

1. Tạo mảng lưu thông tin sinh viên (Tên, email, tuổi) và dùng vòng lặp for in ra thông tin sinh viên trong mảng

*Hướng dẫn*

*a. Tạo mảng 2 chiều chúng ta có thể sử dụng 1 trong 2 mẫu sau:*

*mẫu 1:*

<?php

// Khởi tạo mãng

$students=[[

'name'=>'Hung',

'email'=>'hungcfi@gmail.com',

'age'=>48

],

[

'name'=>'Son',

'email'=>'soncfi@gmail.com',

'age'=>21

],

[

'name'=>'Thuy',

'email'=>'thuycfi@gmail.com',

'age'=>32

]];

echo $students[2]['name'].'<br>';

echo $students[2]['email'].'<br>';

echo $students[2]['age'].'<br>';

?>

*mẫu 2:*

<?php

$students = array(

array(

'name' => 'Nguyễn Văn Cường 1',

'email' => 'thehalfheart1@gmail.com',

'age' => '29'

),

array(

'name' => 'Nguyễn Văn Cường 2',

'email' => 'thehalfheart2@gmail.com',

'age' => '29'

),

array(

'name' => 'Nguyễn Văn Cường 3',

'email' => 'thehalfheart3@gmail.com',

'age' => '29'

),

array(

'name' => 'Nguyễn Văn Cường 4',

'email' => 'thehalfheart4@gmail.com',

'age' => '29'

),

array(

'name' => 'Nguyễn Văn Cường 5',

'email' => 'thehalfheart5@gmail.com',

'age' => '29'

),

array(

'name' => 'Nguyễn Văn Cường 6',

'email' => 'thehalfheart6@gmail.com',

'age' => '29'

)

);

?>

*b. Chương trình hoàn chỉnh*

<?php

$students =[[

'name'=>'Phạm Văn Hùng',

'email'=>'hungcfi@gmail.com',

'age'=>48

],

[

'name'=>'Mai Ngọc Sơn',

'email'=>'sonqn@gmail.com',

'age'=>21

],

[

'name'=>'Lê Thị Kim Thuỷ',

'email'=>'thuycfi.hctc@gmail.com',

'age'=>32

]];

?>

<table border="1" cellspacing="0" cellpadding="10">

<?php $sosv=count($students); ?>

<tr>

<th>Họ và tên</th>

<th>email</th>

<th>Tuổi</th>

</tr>

<?php for ($i = 0; $i < $sosv; $i++){ ?>

<tr>

<td><?php echo $students[$i]['name']; ?></td>

<td><?php echo $students[$i]['email']; ?></td>

<td><?php echo $students[$i]['age']; ?></td>

</tr>

<?php } ?>

</table>

**Giải thích cách viết kết hợp lệnh PHP & HTML**

PHP là một ngôn ngữ dành cho lập trình web. Như vậy, ta có thể suy ra là PHP và HTML sẽ phải đi cùng với nhau. Ở trong phần này, chúng ta sẽ cùng đi tìm hiểu làm thế nào để có thể kết hợp được mã PHP và HTML lại với nhau.

NHÚNG PHP TRONG HTML

Bước 1: Tạo một file có định dạng .php

Giả sử là index.php.

Tạo khung 1 trang HTML.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Nhúng PHP trong HTML</title>

</head>

<body>

<h1> <?php echo 'hello world!'; ?> </h1>

</body>

</html>

CÁCH HIỂN THỊ RA MÃ HTML VỚI PHP

Xét 3 dòng lệnh sau:

<h1>Hello world</h1>

<h1><?php echo “Hello world“ ?></h1>

<h1><= “Hello world” ?></h1>

Cả ba dòng lệnh trên khi hiển thị lên trang web sẽ có cùng một kết quả. Nhưng xử lý của dòng đầu tiên khi hiển thị ra khác với dòng thứ hai và thứ ba.

**Bước 1:**

Khi gặp dòng lệnh đầu tiên, trình thông dịch PHP của server sẽ không làm gì cả. Vì nó là đã là mã HTML rồi.

Còn khi gặp dòng lệnh thứ hai và thứ ba thì trình thông dịch PHP của server sẽ output chữ Hello word trong cặp ngoặc <?php ?> ra thành mã HTML trước khi xuất ra

**Bước 2:**

Server trả mã HTML về cho trình duyệt. Sau đó trình duyệt sẽ hiển thị mã HTML cho người dùng.

*Chú ý:*

**<?php** echo “Hello world” **?>** = **<=** “Hello world” **?>**

Do chúng ta output dữ liệu thường xuyên, PHP đã cung cấp 1 cách viết tắt cho câu lệnh **<?php** echo là **<=**

PHP không chỉ output dữ liệu ra được dưới dạng text như các ví dụ ở trên. Mà nó còn có thể output ra được dữ liệu dưới dạng mã css và js.

2. Vòng lặp for In ra các số chẵn từ 0 đến 100

3. Vòng lặp for tìm phần tử lớn nhất trong mảng

4. Vòng lặp for bước nhảy 2 in ra các số lẻ từ 0 dến 100

3.5.9. Vòng lặp while

Khác với vòng lặp for trong php, vòng lặp while và vòng lặp do while lặp đối với những bài toán không xác định được số lần lặp, còn vòng lặp for thì biết được số lần lặp mới lặp được. Tuy nhiên có những bài toán có thể dùng cả 3 vòng lặp for, while và do while để giải.

*Cú pháp:*

**while ( $btdieukien ) {**

**// dòng lệnh**

**}**

Trong đó **$btdieukien** là điều kiện để dừng vòng lặp. Nếu **$btdieukien** có giá trị false thì vòng lặp kết thúc, ngược lại vòng lặp sẽ tiếp tục lặp. Vòng lặp while sẽ lặp vô hạn nếu biểu thức điều kiện bạn truyền vào luôn luôn đúng.

*Ví dụ:* *Dùng vòng lặp while trong php liệt kê các số từ 1 tới 10.*

for ($i = 1; $i <= 10; $i++){

echo $i . ' - ';

}

3.5.10. Vòng lặp do while

Vòng lặp while sẽ kiểm tra điều kiện trước rồi thực hiện câu lệnh bên trong vòng lặp, còn vòng lặp do while thì ngược lại sẽ thực hiện câu lệnh bên trong vòng lặp trước rồi mới kiểm tra điều kiện. Nếu điều kiện đúng thì sẽ thực hiện tiếp vòng lặp kế tiếp, nếu điều kiện sai thì sẽ dừng vòng lặp. Vòng lặp do while trong php luôn luôn thực hiện ít nhất một lần lặp vì nó thực hiện trước rồi mới kiểm tra điều kiện.

*Cú pháp:*

**do {**

**// dòng lệnh**

**} while ( $btdieukien );**

*Lưu ý: Bắt buộc phải có dấu ; sau While*

*Ví dụ:*

$i = 1;

do{

echo $i;

$i++;

} while ($i <= 10);

[1. Tổng quan về PHP 1](#_Toc70146769)

[1.1. Giới thiệu PHP 1](#_Toc70146770)

[1.2. Cài đặt PHP 1](#_Toc70146771)

[2. Cài đặt Webserver (Xampp) 1](#_Toc70146772)

[2.1. Localhost 1](#_Toc70146773)

[2.2. Localhost vận hành như thế nào 1](#_Toc70146774)

[2.3. Lưu ý trước khi cài đặt 1](#_Toc70146775)

[Xoá toàn bộ ứng dụng liên quan tới localhost 1](#_Toc70146776)

[Lưu ý nếu dùng Skype 1](#_Toc70146777)

[Tắt tường lửa 2](#_Toc70146778)

[Tắt UAC trên Windows 2](#_Toc70146779)

[2.4. Qui trình cài đặt 2](#_Toc70146780)

[2.5. Khởi động Localhost 3](#_Toc70146781)

[2.6. Cách đổi cổng mạng cho Localhost 4](#_Toc70146782)

[2.7. Thao tác trên localhost 5](#_Toc70146783)

[2.7.1. Làm việc với thư mục và tập tin 5](#_Toc70146784)

[2.7.2. Tạo cơ sở dữ liệu MySQL (Database) 6](#_Toc70146785)

[3. Nền tảng của PHP 7](#_Toc70146786)

[3.1. Cú pháp PHP 7](#_Toc70146787)

[3.1.1. Bắt đầu PHP. 7](#_Toc70146788)

[3.1.2. Comment trong PHP. 7](#_Toc70146789)

[3.1.3. Hiển thị dữ liệu trong PHP. 8](#_Toc70146790)

[3.1.4. Chương trình Helloworld. 8](#_Toc70146791)

[3.2. Biến, hằng, kiểu dữ liệu 8](#_Toc70146792)

[3.2.1. Biến 8](#_Toc70146793)

[3.2.2. Hằng 8](#_Toc70146794)

[3.2.3. Kiểu dữ liệu 9](#_Toc70146795)

[3.3. Kiểu mảng, kiểu đối tượng 12](#_Toc70146796)

[Mảng một chiều 14](#_Toc70146797)

[Mảng nhiều chiều 14](#_Toc70146798)

[3.4. Toán tử 15](#_Toc70146799)

[3.4.1. Biểu thức 15](#_Toc70146800)

[Toán tử gán (Assignment Operator): 16](#_Toc70146801)

[Biểu thức số học 16](#_Toc70146802)

[3.4.2. Toán tử quan hệ 16](#_Toc70146803)

[3.4.3. Toán tử Logic 16](#_Toc70146804)

[Độ ưu tiên toán tử Logic 17](#_Toc70146805)

[3.4.4. Toán tử kết hợp 17](#_Toc70146806)

[3.5. Các cấu trúc điều khiển 17](#_Toc70146807)

[3.5.1. Câu lệnh if 17](#_Toc70146808)

[3.5.2. Câu lệnh if else 18](#_Toc70146809)

[3.5.3. Kết hợp nhiều câu lệnh if else trong php 18](#_Toc70146810)

[3.5.4. Câu lệnh if lồng 19](#_Toc70146811)

[BÀI TẬP 20](#_Toc70146812)

[3.5.5. Câu lệnh switch case 20](#_Toc70146813)

[3.5.6. Vòng lặp for 21](#_Toc70146814)

[3.5.7. Vòng lặp foreach 21](#_Toc70146815)

[3.5.8. For và foreach lồng 22](#_Toc70146816)

[BÀI TẬP 23](#_Toc70146817)

[NHÚNG PHP TRONG HTML 25](#_Toc70146818)

[CÁCH HIỂN THỊ RA MÃ HTML VỚI PHP 25](#_Toc70146819)

[3.5.9. Vòng lặp while 26](#_Toc70146820)

[3.5.10. Vòng lặp do while 26](#_Toc70146821)