# Thuật ngữ cơ bản

## Class – Lớp

Định nghĩa: [Class là một tập hợp các đối tượng và mô tả của các đối tượng có chung các thuộc tính và hành động. Nó chứa các đặc tính của đối tượng như các thuộc tính, các hành động hoặc các hành vi](https://www.google.com/search?q=class+l%C3%A0+m%E1%BB%99t+t%E1%BA%ADp+h%E1%BB%A3p+c%C3%A1c+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng(objects)+v%C3%A0+m%C3%B4+t%E1%BA%A3+c%E1%BB%A7a+c%C3%A1c+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng+c%C3%B3+chung+c%C3%A1c+thu%E1%BB%99c+t%C3%ADnh+v%C3%A0+h%C3%A0nh+%C4%91%E1%BB%99ng.+N%C3%B3+ch%E1%BB%A9a+c%C3%A1c+%C4%91%E1%BA%B7c+t%C3%ADnh+c%E1%BB%A7a+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng+nh%C6%B0+c%C3%A1c+thu%E1%BB%99c+t%C3%ADnh(attributes)%2C+c%C3%A1c+h%C3%A0nh+%C4%91%E1%BB%99ng+ho%E1%BA%B7c+c%C3%A1c+h%C3%A0nh+vi+(behaviors)&oq=class+l%C3%A0+m%E1%BB%99t+t%E1%BA%ADp+h%E1%BB%A3p+c%C3%A1c+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng(objects)+v%C3%A0+m%C3%B4+t%E1%BA%A3+c%E1%BB%A7a+c%C3%A1c+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng+c%C3%B3+chung+c%C3%A1c+thu%E1%BB%99c+t%C3%ADnh+v%C3%A0+h%C3%A0nh+%C4%91%E1%BB%99ng.+N%C3%B3+ch%E1%BB%A9a+c%C3%A1c+%C4%91%E1%BA%B7c+t%C3%ADnh+c%E1%BB%A7a+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng+nh%C6%B0+c%C3%A1c+thu%E1%BB%99c+t%C3%ADnh(attributes)%2C+c%C3%A1c+h%C3%A0nh+%C4%91%E1%BB%99ng+ho%E1%BA%B7c+c%C3%A1c+h%C3%A0nh+vi+(behaviors)&aqs=chrome..69i57.636j0j7&sourceid=chrome&ie=UTF-8).

class Animal

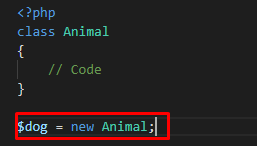
{

    // Code

}

## Object – Đối tượng

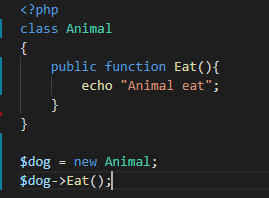
Định nghĩa: Đối tượng trong một ngôn ngữ OOP là các kết hợp giữa mã và dữ liệu mà chúng được nhìn nhận như là một đơn vị duy nhất. Mỗi đối tượng có một tên riêng biệt và tất cả các tham chiếu đến đối tượng đó được tiến hành qua tên của nó.



## Method – Phương thức

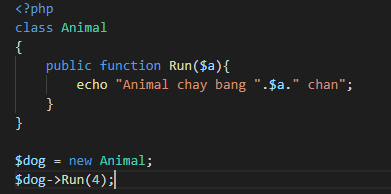
Định nghĩa: Phương thức giống với định nghĩa hàm trong lập trình cấu trúc. Phương thức là các hàm thể hiện hành vi của một đối tượng thuộc lớp. Có các loại phương thức của có đối, phương thức không đối, phương thức trả về, phương thức khởi tạo và phương thức hủy.

* + 1. Phương thức không đối

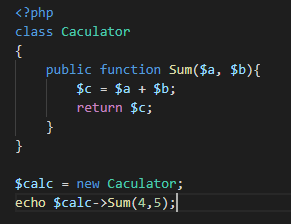




* + 1. Phương thức có đối



* + 1. Phương thức trả về





## Attribute – Thuộc tính

Thuộc tính của một lớp bao gồm các biến, các hằng, hay tham số nội tại của lớp đó. Ở đây, vai trò quan trọng nhất của các thuộc tính là các biến vì chúng sẽ có thể bị thay đổi trong suốt quá trình hoạt động của một đối tượng. Các thuộc tính có thể được xác định kiểu và kiểu của chúng có thể là các kiểu dữ liệu cổ điển hay đó là một lớp đã định nghĩa từ trước.

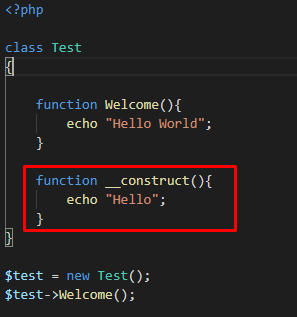
* + 1. Biến
    2. Hằng
    3. Mảng
    4. Object

## Constructor – Destructor

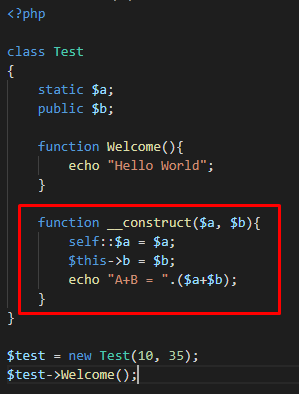
### Construct

* Có hai loại là construct có đối và construct không đối. Và hai phương thức construct này không thể được khai báo cùng lúc.
* Hàm construct là hàm thể hiện cho lớp đó. Và luôn được khai báo mặc định khi ta gọi đối tượng của lớp đó.

VD: Hàm construct không đối số

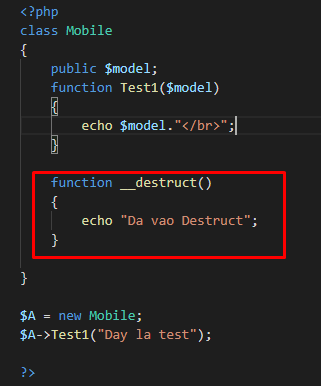


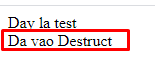
Hàm construct có đối số



### Destruct

* Hàm hủy, hàm này luôn dùng cho đóng collection của database

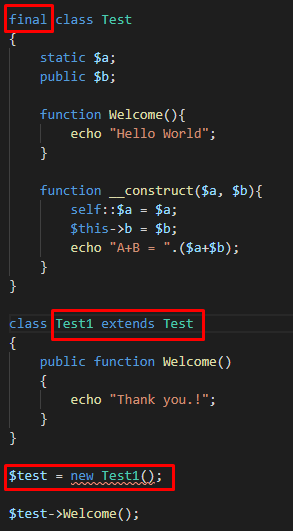




## Final – Static

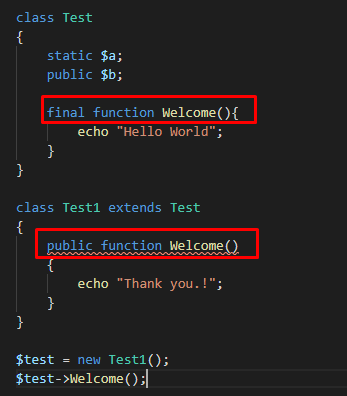
### Final

* Khi khai báo final class thì không lớp nào có thể kế thừa nó mà nó chỉ có thể khởi tạo được thôi





* Tương tự như class khi ta khai báo final cho phương thức thì các phương thức được kế thừa từ lớp con không thể ghi đè lên phương thức đó được.

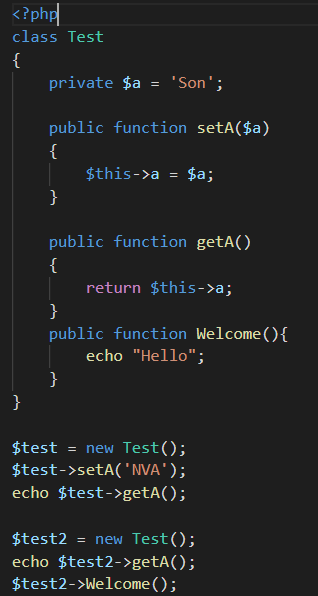




### Static

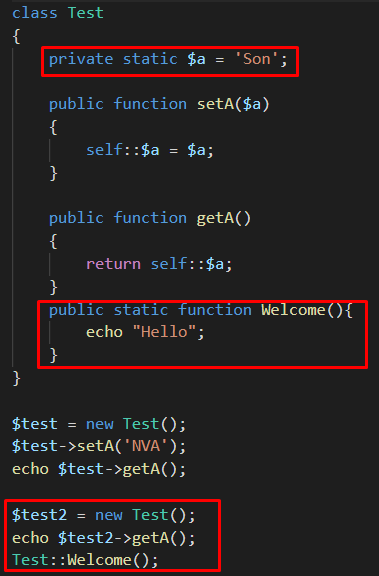
* Là một thành phần tĩnh mà nó hoạt động như một biến toàn cục. Nó lưu lại giá trị cuối cùng mà nó thực hiện

VD: Không khai báo thuộc tính static





VD: Với khai báo static



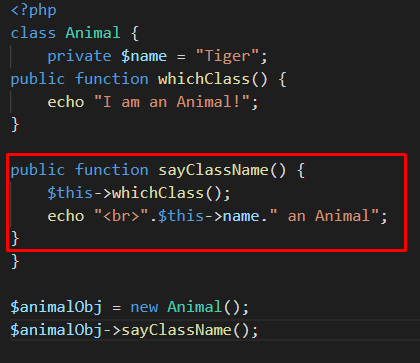


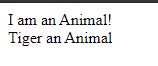
## This - self

### This

* Từ khóa this dùng để tham chiếu đến các thuộc tính, phương thức thường

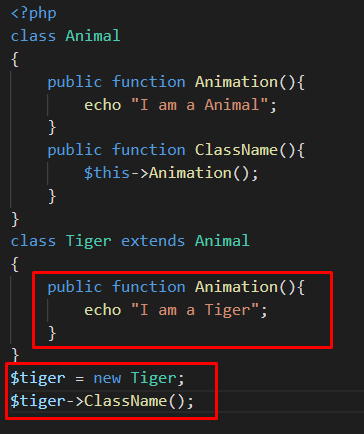
VD:





* Từ khóa this dùng để tham chiếu đến đối tượng (object).

VD:





Trong ví dụ này ta thấy khi gọi phương thức ClassName() bằng đối tượng $tiger mà trong phương thức ClassName() của lớp Animal lại dùng this để gọi đến phương thức Animation() nên nó sẽ hiểu là chuyển đến phương thức Animation() của lớp Tiger. Tức là this luôn luôn tham chiếu đến đối tượng.

* Từ khóa this không sử dụng được bên trong các thành phần tĩnh

VD

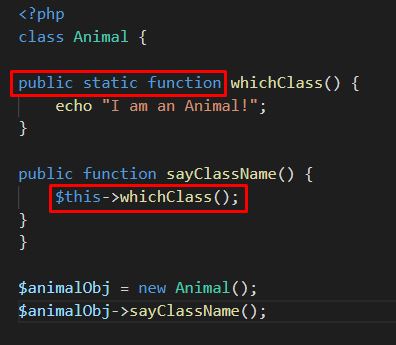




Như chúng thấy nó sẽ báo lỗi. Điều này chứng tỏ nó không có tác dụng khi được khai báo bên trong các phương thức tĩnh.

* Có thể giọi các hàm static bằng từ khóa this

VD:

* Không thể gọi các thuộc tính static bằng từ khóa this

VD:

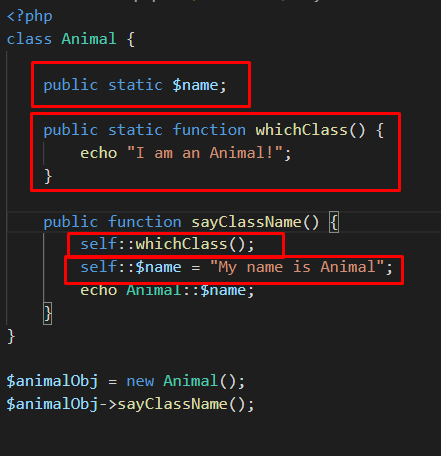




### Self

* Từ khóa self dùng để tham chiếu đến các thuộc tính, phương thức static

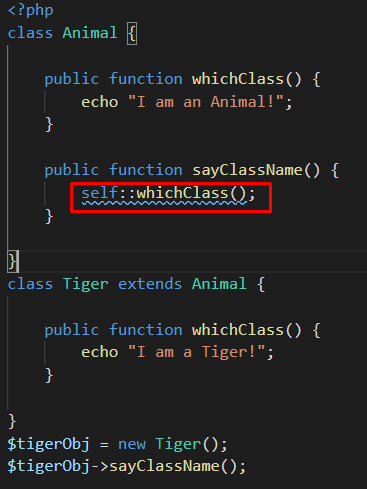
VD:





* Từ khóa self dùng để truy cập đến chính class

VD:

Trong ví dụ này khi đối tượng gọi hàm sayClasName() nhưng class Tiger không có phương thúc này nên nó sẽ lấy ở lớp cha nhưng do trong phương thức này gọi hàm wichClass() bằng từ khóa self nên nó gọi phương thức của chính lớp cha.

* Từ khóa self sử dụng được bên trong các thành phần thường

VD:

* Có thể giọi các hàm static bằng từ khóa self

VD



* Không thể gọi các thuộc tính thường bằng từ khóa self

|  |  |
| --- | --- |
| **This** | **Self** |
| Từ khóa this dùng để tham chiếu đến các thuộc tính, phương thức thường | Từ khóa self dùng để tham chiếu đến các thuộc tính, phương thức static |
| Từ khóa this dùng để tham chiếu đến đối tượng (object) | Từ khóa self dùng để truy cập đến chính class |
| Từ khóa this không sử dụng được bên trong các thành phần tĩnh | Từ khóa self sử dụng được bên trong các thành phần thường |
| Có thể giọi các hàm static bằng từ khóa this | Có thể giọi các hàm static bằng từ khóa self |
| Không thể gọi các thuộc tính static bằng từ khóa this | Không thể gọi các thuộc tính thường bằng từ khóa self |
| Làm việc với các hàm và thuộc tính thường | Làm việc với các hàm và thuộc tính static |

## Ngoại lệ

* Khi có một bài toán nếu như người dùng nhập những phép tính không hợp lệ thì php sẽ ném ra một lỗi và gây ra việc chương trình bị dừng lại nếu lỗi đó không được sửa. Và một lỗi được ném ra phải được sửa, đôi khi người viết code không sửa được lỗi được ném ra mà vẫn muốn chạy chương trình nên ngoại lệ được định nghĩa.
* VD: Một bài toán rút gọn phân số. Nhưng nếu mẫu số bằng 0 thì phân số không hợp lệ. Khi chạy chương trình người dùng cố ý nhập mẫu bằng 0 thì xảy ra lỗi.
* Tham khảo thêm tại: <https://docs.google.com/document/d/1yDuotFu01eJFHOOddfd1pW38Gwa0KcduuM5WA8iwFEQ/edit?usp=sharing> hoặc <https://niithanoi.edu.vn/vi-du-huong-dan-su-dung-try-catch-xu-ly-loi-va-ngoai-le-trong-php.html>

## Magic Method

* Magic methods là các phương thức đặc biệt được tạo ra nhằm giải quyết các vấn đề về sự kiện trong chương trình (cụ thể là với class), và đối với PHP cũng thế.
* Nó gồm có các ưu điểm như sau:
  + Giúp cho chúng ta tùy biến được các hành vi.
  + Nó giúp cho chúng ta có thể khởi tạo một đối tượng theo cách mình muốn.
* Các magic method cơ bản:
  + \_\_construct(): gọi khi khởi tạo đối tượng.
  + \_\_destruct(): goij khi hủy đối tượng.
  + \_\_set(): gọi khi ta truyền dữ liệu cho một thuộc tính không được phép truy cập.
  + \_\_get(): khi đọc dữ liệu từ một thuộc tính không được phép truy cập.
  + \_\_isset(): được gọi khi gọi hàm isset() hoặc empty() trên một thuộc tính không được phép truy cập.
  + \_\_unset(): được gọi khi hàm unset() được sử dụng trong một thuộc tính không được phép truy cập.
  + \_\_call():được gọi khi ta gọi một phương thức không được phép truy cập trong phạm vi của một đối tượng.
  + \_\_callstatic(): được kích hoạt khi ta gọi một phương thức không được phép truy cập trong phạm vi của một phương thức tĩnh.
  + \_\_toString(): phương thức này giúp class chỉ định xem sẽ in ra cái gì khi nó được dùng.
  + \_\_invoke():phương thức này được gọi khi một lệnh cố gắng gọi một đối tượng như một hàm.
  + \_\_sleep(): được gọi khi serialize() một đối tượng.
  + \_\_wakeup: được gọi khi unserialize() đối tượng.
  + \_\_set\_state():
  + \_\_clone(): được sử dụng khi chúng ta clone một object.
  + \_\_debugInfo(): được gọi khi chúng ta sử dụng hàm vardump().

## Namespace

* Namespace giúp tạo ra một không gian tên cho hàm và lớp trong lập trình nói chung và trong PHP nói riêng.
* Cú pháp: namespace tenNamespace;
* Mỗi file chứa 1 namespace phải được khai báo trên cùng trước các code khác trong file. Cách thực hành tốt nhất là mỗi file chứa 1 class với tên class trùng với tên file đồng thời có 1 namespace mô tả được đường dẫn tới file đó
* Cú pháp sử dụng: use tenNamespace\tenClass
* Ta cũng có thể định danh lại cho namespace nếu tên của nó quá dài: use tenNamespace as tenMoi;

VD: Các bước thực hiện autoload bằng composer sử dụng namespace:

Bước 1: Tạo thư mục OOP

├── composer.json

├── Ex1.php

└── src

   ├── SP1

   │   └── SP1.php

   ├── SP2

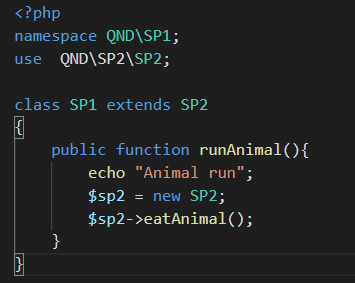
   │   └── SP2.php

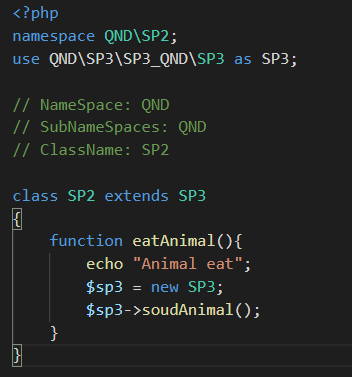
   └── SP3

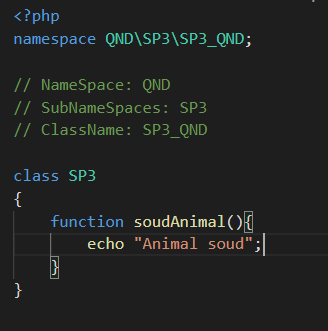
└──SP3\_QND

   └── SP3.php

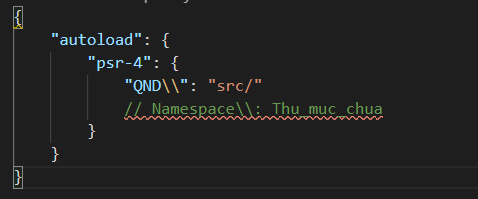
Bước 2: Tạo các file SP1.php, SP2.php, SP3.php

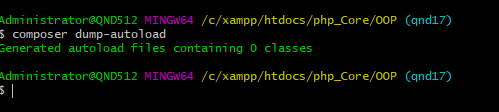


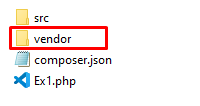




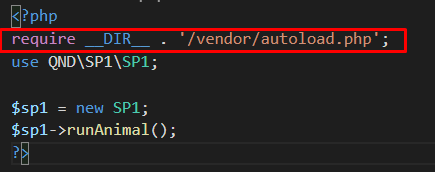
Bước 3: Tạo composer.json thực hiện autoload sau đó chạy lệnh: composer dump-autoload







Bước 4: Trong file Ex1.php gọi tới autoload để load các namespace



Bước 5: Chạy chương trình



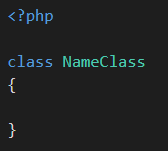
## Quy chuẩn code

### PSR-1

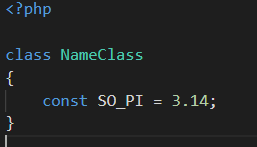
* Dùng thẻ <?php hoặc <?=

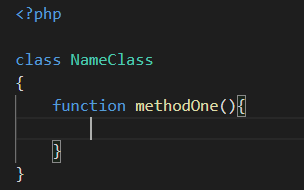
* Các file phải dùng duy nhất UTF-8 không BOM cho code PHP
* Các file nên hoặc định nghĩa kí hiệu (lớp, hàm, hằng số, vv.) hoặc đưa ra tác dụng phụ (e.g. sinh ra output, thay đổi .ini settings, vv.) nhưng không nên làm cả 2
* Các namespace và các class phải tuân theo các chuẩn PSR-4
* Tên class phải được định nghĩa ở dạng StudlyCaps



* Hằng số phải được định nghĩa bằng chữ cái in hoa và cách nhau bằng dấu gạch dưới.

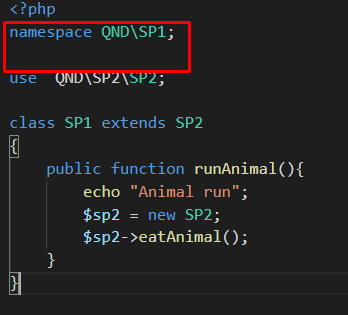


* Các phương thức phải đặt tên dạng camelCase

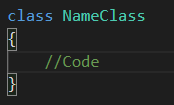


### PSR-2

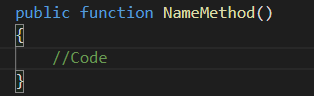
* Tuân thủ PSR-1
* Code phải sử dụng 4 ký tự space để lùi khối
* Mỗi dòng code phải dưới 120 ký tự, Nên dưới 80 ký tự
* Phải có 1 dòng trắng sau namespace, và phải có một dòng trắng sau mỗi khối code



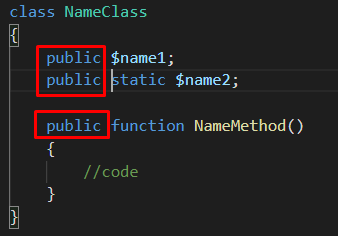
* Ký tự mở lớp { phải ở dòng tiếp theo, và đóng lớp } phải ở dòng tiếp theo của thân class



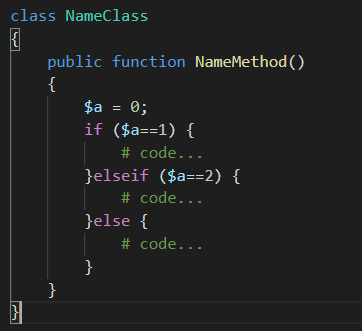
* Ký tự { cho hàm phải ở dòng tiếp theo, và ký tự } kết thúc hàm phải ở dòng tiếp theo của thân hàm



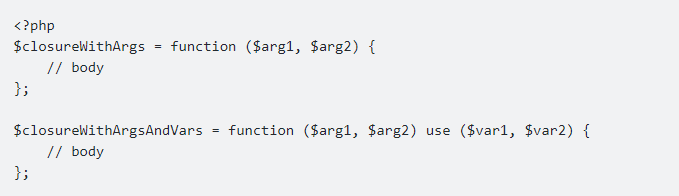
* Các visibility (public, private, protected) phải được khai báo cho tất cả các hàm và các thuộc tính của lớp



* Các từ khóa điều khiển khối(if, elseif, else) phải có một khoảng trống sau chúng; hàm và lớp thì không được làm như vậy.



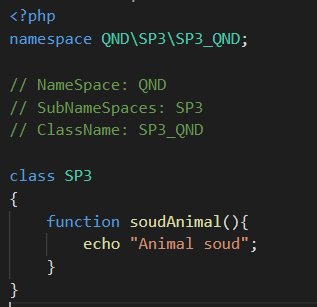
* Mở khối { cho cấu trúc điều khiển phải trên cùng một dòng; và đóng khối này với } ở dòng tiếp theo của thân khối.
* Hằng số true, false, null phải viết với chữ thường.
* Từ khóa extends và implements phải cùng dòng với class Implements nhiều lớp, thì mỗi lớp trên một dòng Keyword và không được dùng sử dụng khai báo property.
* Tên property không nên có tiền tố \_ nhằm thể hiện thuộc tính protect hay private.
* Tham số cho hàm, phương thức: Không được thêm space vào trước dấu ‘,’ và phải có một space sau ‘,’.



* Các tham số có thể trên nhiều dòng, nếu làm như vậy thì phải mỗi dòng 1 tham số. Abstract, final PHẢI đứng trước visibility, còn static phải đi sau

### PSR-4

* Cú pháp: \<NamespaceName>(\<SubNamespaceNames>)\*\<ClassName>
* Trong đó: Mỗi lớp phải xây dựng sao cho có thể được tham chiếu đến bởi dòng code ba thành phần: Namespace, Các SubNamespaceNames con, và tên lớp ClassName
  + NameSpace : Tiền đố đầu tiên bắt buộc phải có - được hiểu là tên vendor. Tên này do bạn tự đặt, sao cho không xung đột tên các thư viện khác.
  + SubNameSpaces: Các namespace con. Có một hoặc nhiều tùy bạn. Nhưng bắt đầu từ SubNameSpace thì nó tương ứng với cấu trúc thư mục lưu trữ code.
  + ClassName: Bắt buộc phải có và phải có tên file PHP trùng tên ClassName ở thư mục tương ứng với namespace cuối cùng (ClassName.php), trong file đó sẽ định nghĩa nội dung của code của lớp.

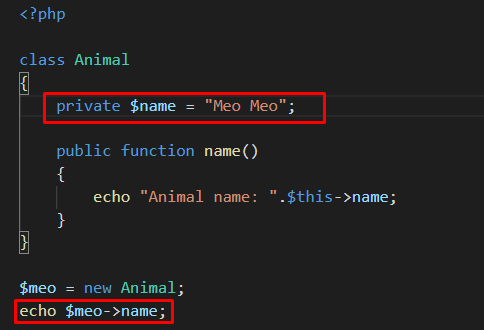


# Tính đóng gói – Encapsulation

## Định nghĩa

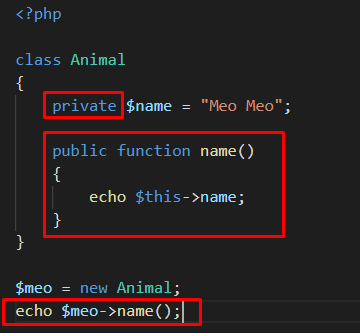
- Tính đóng gói (Encapsulation) là cơ chế của che giấu dữ liệu bởi chúng được lớp (class) che giấu đi (ở dạng private) một số dữ liệu, hàm và phương thức để đảm bảo rằng các dữ liệu đó sẽ được truy cập và sử dụng đúng mục đích, đúng cách thông qua các hàm và phương thức ở dạng public mà class cung cấp. Đó là lý do bạn không thể truy cập đến các thuộc tính private hoặc gọi đến phương thức private của class từ bên ngoài class đó.

VD:





Tính đóng gói được thể hiện khi mỗi đối tượng mang trạng thái là private ở bên trong một class và những đối tượng khác không thể truy cập trực tiếp vào phạm vi này. Thay vào đó họ chỉ có thể gọi các hàm mang phạm vi public được gọi là phương thức.





## Access Modifier

- Trong PHP, có các access modifier quy định phạm vi sử dụng của thuộc tính và phương thức:

+ Private: các thuộc tính và phương thức private chỉ có thể truy cập từ bên trong class

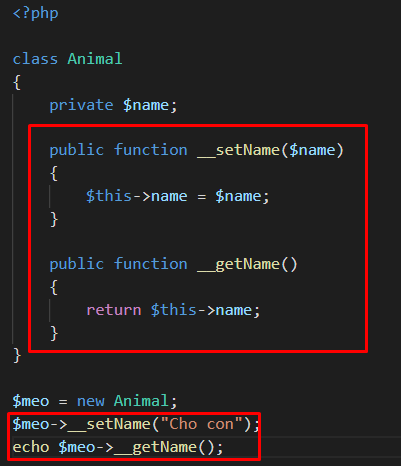
+ Protected: các thuộc tính và phương thức protected có thể truy cập từ bên trong class và các class kế thừa nó

+ Public: các thuộc tính và phương thức public có thể truy cập từ bên ngoài

## Geter – Seter

- Trong lập trình hướng đối tượng, với các phương thức có cách hiển thị khác với public (bao gồm private hoặc protected) thì chúng ta sẽ thường sẽ khai báo thêm các phương thức setter() hoặc getter() vào class để có thể thực hiện các tác vụ lấy ra (hoặc gán) giá trị cho các property này.

VD:

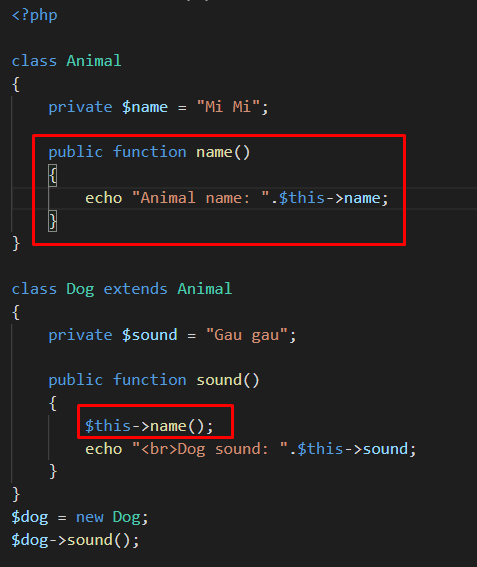


# Tính kế thừa – Inheritance

## Định nghĩa

- Đặc tính này cho phép một đối tượng có thể có sẵn các thuộc tính, phương thức mà đối tượng khác đã có thông qua kế thừa. Điều này cho phép các đối tượng có thể tái sử dụng hay mở rộng các đặc tính sẵn có mà không phải tiến hành định nghĩa lại.

- Trong PHP, một lớp có thể kế thừa từ một lớp khác. Đối tượng thuộc lớp con sẽ có các thuộc tính và phương thức protected và public của lớp mà nó kế thừa.

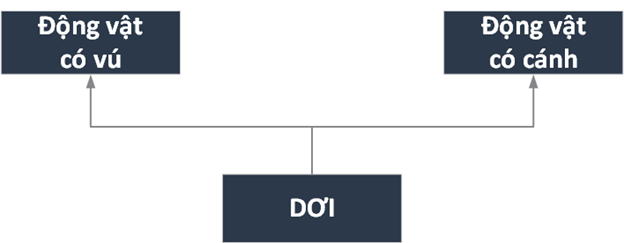


## Đa kế thừa

- Đa kế thừa là một class có thể kế thừa từ nhiều hơn 1 class khác.

- Trong PHP không hỗ trợ đa kế thừa

- Ta có thể thay thế tính chất đa kế thừa bằng cách sử dụng interface



## Traint

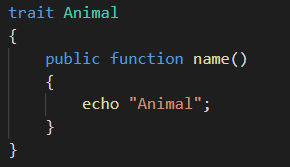
- Traits là cơ chế giúp cho lập trình viên có thể sử dụng lại các phương thức từ các class khác nhau một cách dễ dàng hơn.

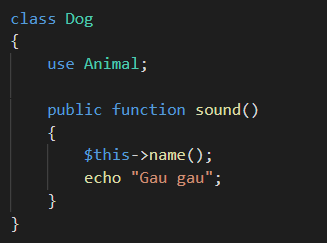
- Một trait tương tự như là 1 class nhưng chỉ nhằm mục đích nhóm chức năng lại. Và trait không thể khởi tạo giống class và trait sinh ra để bổ sung cho kế thừa truyền thống.

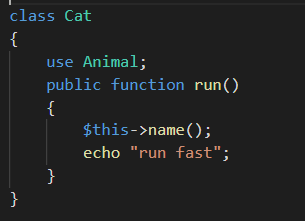
- Thay vì phải kế thừa 1 class hay interface để sử dụng lại 1 nhóm chức năng, thì với trait bạn không cần phải kế thừa vẫn có thể sử dụng được.

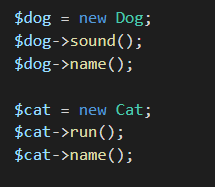
- Trait sẽ khắc phục được vấn đề không có đa kế thừa trong PHP

VD1:

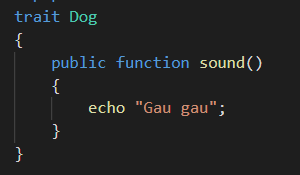


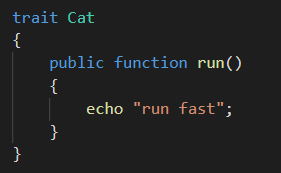


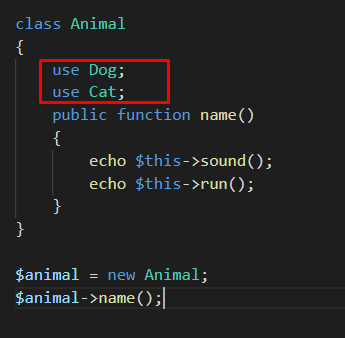




VD2:







- Không thể khởi tạo đối tượng cho trait như abstract class.

# Tính đa hình – Polymorphism

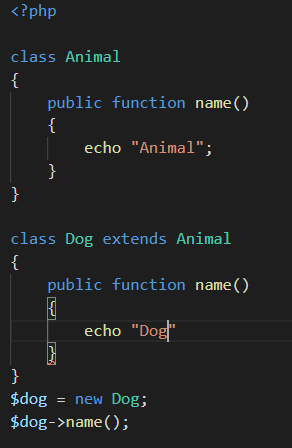
## Định nghĩa

- Thể hiện qua việc có thể định nghĩa một đặc tính, hoặc phương thức cho một loạt các đối tượng gần giống nhau. Nhưng khi thực hiện thì các đối tượng khác nhau sẽ có cách thực hiện khác nhau và cho ra kết quả khác nhau.

## Ghi đè

- Nếu một phương thức của lớp con trùng với một phương thức đã tạo của lớp cha thì đó là ghi đè.

VD:

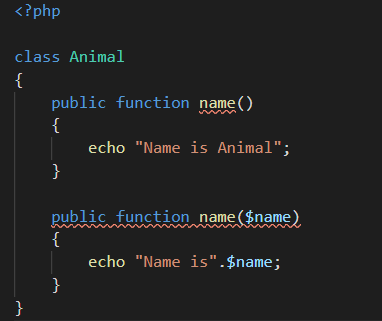


## Nạp chồng

- Các phương thức nằm trong cùng 1 lớp, có cùng tên với nhau nhưng có danh sách đối số khác nhau được gọi là các phương thức nạp chồng.

- Tùy theo ta gọi đối số thế nào mà nó sẽ gọi hàm tương ứng. Nạp chồng là hình thức đa hình trong quá trình biên dịch.

VD:



- Trong ví dụ này ta thấy khi nạp chồng phương thức name thì máy sẽ báo lỗi. Lý do là PHP không hỗ trợ khai báo như vậy mà chúng ta cần phải gọi hàm \_\_call

VD2:



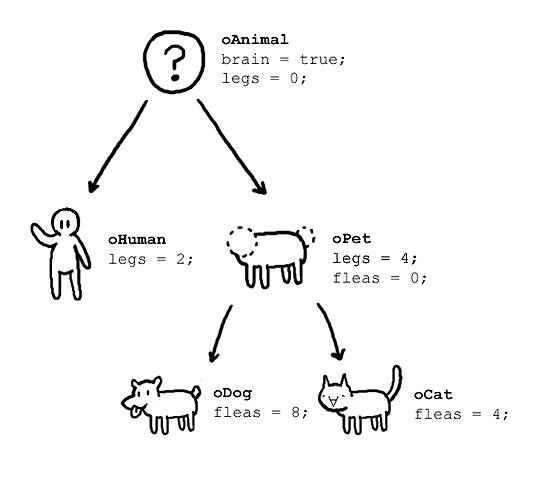
- Trong ví dụ này ta nạp chồng các phương thức bằng cách sử dụng hàm \_\_call. Kết quả nhận được:



# Tính trừu tượng – Abstraction

## Định nghĩa

- Tính trừu tượng (abstraction) trong lập trình hướng đối tượng giúp giảm sự phức tạp thông qua việc tập trung vào các đặc điểm trọng yếu hơn là đi sâu vào chi tiết.



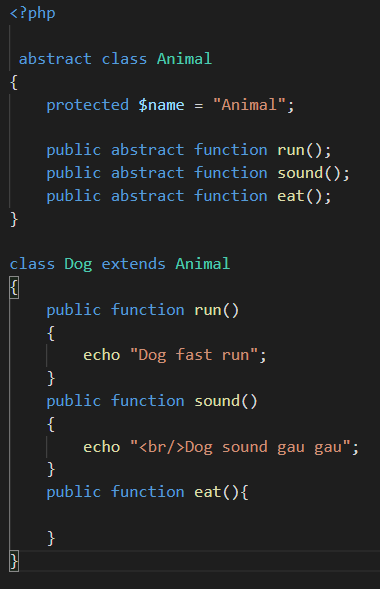
## Abstract class

- Lớp Abstract sẽ định nghĩa các hàm mà từ đó các lớp con sẽ kế thừa nó và Overwrite lại.

- Tất cả các phương thức của lớp abstract đều phải được khai báo là abstract và phải ở mức protected và public, không được ở mức private.

- Lớp Abstract có thể có thuộc tính nhưng thuộc tính không được khai báo là abstract, và bạn không thể khởi tạo một biến của lớp Abstract được.

VD:



## Intereface

- Interface trong hướng đối tượng là một khuôn mẫu, giúp cho chúng ta tạo ra bộ khung cho một hoặc nhiều đối tượng và nhìn vào interface thì chúng ta hoàn toàn có thể xác định được các phương thức và các hằng số sẽ có trong đối tượng implement nó

- Interface không phải là một đối tượng.

- Trong interface chúng ta chỉ được khai báo phương thức chứ không được định nghĩa chúng.

- Trong interface chúng ta có thể khai báo được hằng nhưng không thể khai báo biến.

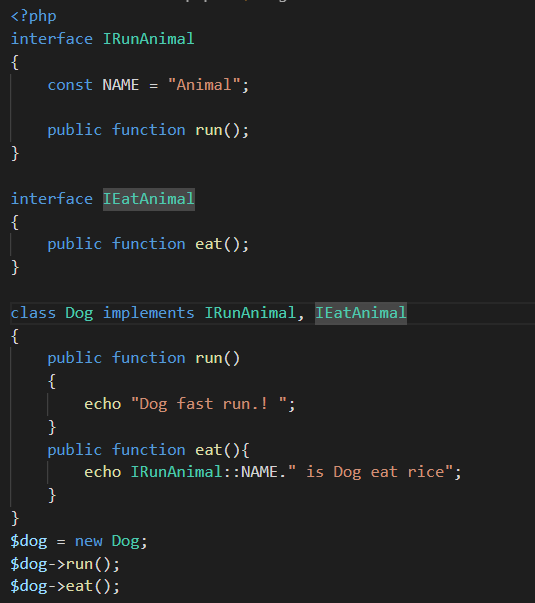
- Một interface không thể khởi tạo được (vì nó không phải là một đối tượng).

- Các lớp implement interface thì phải khai báo và định nghĩa lại các phương thức có trong interface đó.

- Một class có thể implement nhiều interface.

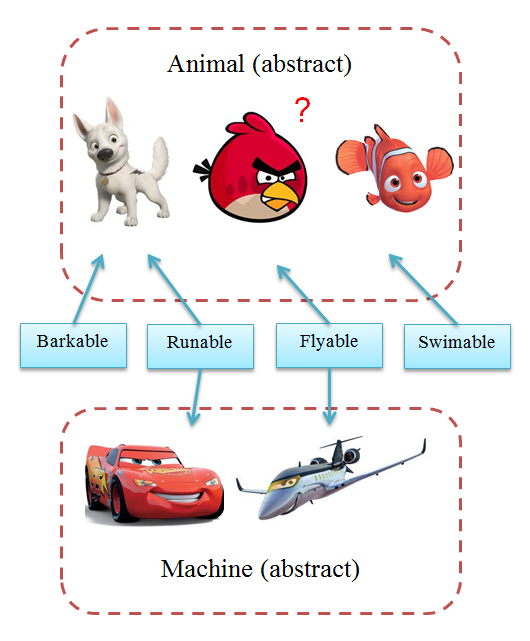
- Các interface có thể kế thừa lẫn nhau.

VD:





- Sự khác nhau giữa Interface và Abstract Class:



# Website PHP

## Debug PHP

- Hướng dẫn debug PHP với Visual Studio Code

Bước 1: Tải Xdebug

(Bạn phải download Xdebug đúng với phiên bản PHP bạn đang sử dụng)



File tải về sẽ có dạng Xdebug.dll và chép vào thư mục C:\xampp\php\ext

Bước 2: Thêm cấu hình Xdebug vào php.ini

Chèn đoạn code:

[XDebug]

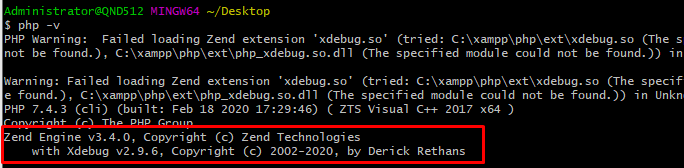
zend\_extension= C:\xampp\php\ext\php\_xdebug-2.9.6-7.4-vc15-nts-x86\_64.dll

xdebug.remote\_enable = 1

xdebug.remote\_autostart = 1

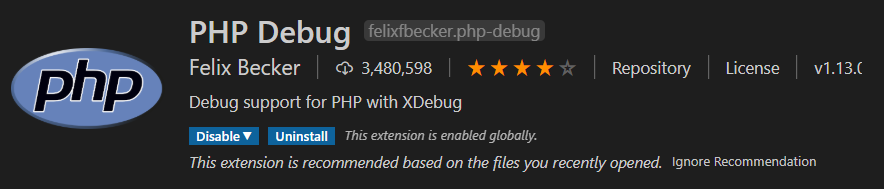
Vào cuối file php.ini

Sau đó vào cmd gõ lệnh php –v



Nếu có dòng chữ như trên là thành công.

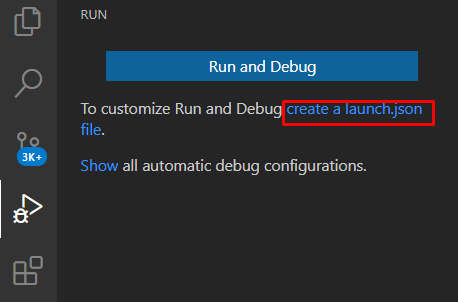
Bước 3: Cài đặt PHP Debug trên Vscode



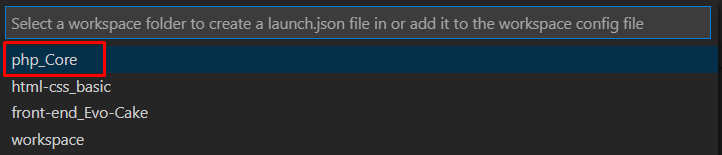
Reset lại Vscode là đã cài xdebug thành công.!

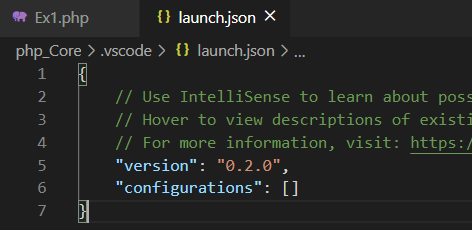
- Hướng dẫn sử dụng Xdebug

Bước 1: Tạo một launch.json

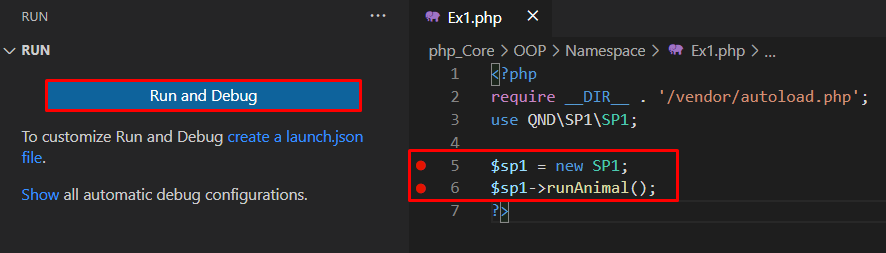


Bước 2: Chọn project cần debug

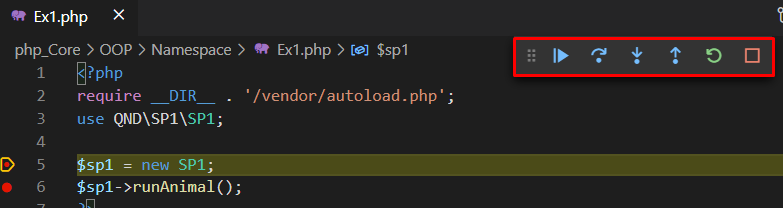




Lưu vào và ra file cần debug đặt breadpoint sau đó nhấn nút run and debug

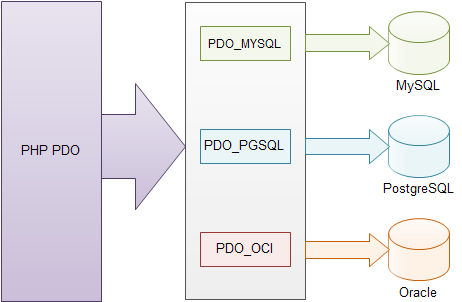


Bảng debug code và tiến hành debug



## PDO

* PDO là API cung cấp cho chúng ta để kết nối cũng như thao tác với CSDL, chuyển dữ liệu thành các đối tượng để chúng ta dễ thao tác. Hiện tai PDO hỗ trợ khoảng 12 loại hệ quản trị CSDL khác nhau.



* PDO chỉ xuất hiện trong PHP từ version 5.x trở đi, vì vậy nếu như server của các bạn chỉ support php version dưới 5., hãy chịu khó update lên 5.x.
* Sau khi cài đặt xong, chúng ta phải mở cái php.ini và enable thư viện này (nếu chưa có) cùng các trình điều khiển tới các CSDL liên quan.

VD dưới đây là dòng lệnh cấu hình trình điều khiển CSDL chạy trên Windows: extension=php\_pdo.dll extension=php\_pdo\_firebird.dll extension=php\_pdo\_informix.dll extension=php\_pdo\_mssql.dll extension=php\_pdo\_mysql.dll extension=php\_pdo\_oci.dll extension=php\_pdo\_oci8.dll extension=php\_pdo\_odbc.dll extension=php\_pdo\_pgsql.dll extension=php\_pdo\_sqlite.dll

* Các thao tác cơ bản:

1. Kết nối cơ sở dữ liệu

<?php

class DbConnect

{

    private $host = "localhost";

    private $dbName = "php\_pdo";

    private $username = "root";

    private $password = "";

    public function connect()

    {

        try {

            $conn = new PDO("mysql:host=$this->host; dbname=$this->dbName",

            $this->username, $this->password);

            // set the PDO error mode to exception

            $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

            echo "Connected successfully"; //test

return $conn;

        } catch (PDOException $e) {

            echo "Connection failed: " . $e->getMessage();

        }

    }

}

$obj = new DbConnect;

$obj->connect();

1. Lấy dữ liệu từ một bảng

$sth = $conn->prepare('select \* from table');

$sth->execute();

$result = $sth->fetchAll(PDO::FETCH\_OBJ);

PDO::FETCH\_ASSOC: Trả về dữ liệu dạng mảng với key là tên của column (column của các table trong database)

PDO::FETCH\_BOTH (default): Trả về dữ liệu dạng mảng với key là tên của column và cả số thứ tự của column

PDO::FETCH\_BOUND: Gán giá trị của từng column cho từng biến đã khởi tạo trước đó qua phương thức bindColumn()

PDO::FETCH\_CLASS: Gán giá trị của từng column cho từng thuộc tính (property/attribute) của một lớp Class theo tên column và tên thuộc tính.

PDO::FETCH\_INTO: Gán giá trị của từng column cho từng thuộc tính của một Class Instance (thể hiện của một lớp)

PDO::FETCH\_LAZY: Gộp chung

PDO::FETCH\_BOTH/PDO::FETCH\_OBJ PDO::FETCH\_NUM: Trả về dữ liệu dạng mảng với key là số thứ tự của column

PDO::FETCH\_OBJ: Trả về một Object của stdClass với tên thuộc tính của Object là tên của column.

1. Lấy dữ liệu có điều kiện
   1. Insert

public function insert()

    {

        $sql = "INSERT INTO $this->tableName VALUES(null, :name, :type, :pages, :price,

        :auther, :cdate)";

        $stmp = $this->dbConn->prepare($sql);

        $stmp->bindParam(':name',$this->name);

        $stmp->bindParam(':type',$this->type);

        $stmp->bindParam(':pages',$this->pages);

        $stmp->bindParam(':price',$this->price);

        $stmp->bindParam(':auther',$this->auther);

        $stmp->bindParam(':cdate',$this->created\_date);

        if ($stmp->execute()) {

            return true;

        }else{

            return false;

        }

    }

* 1. Load dữ liệu

public function getAllBooks()

    {

        $stmp = $this->dbConn->prepare("SELECT \* FROM $this->tableName");

        $stmp->execute();

        $books = $stmp->fetchAll(PDO::FETCH\_ASSOC);

        return $books;

    }

* 1. Update

$sth = $conn->prepare('update users set fullname = :fullname where id = :id');

$sth->execute(array(':fullname' => 'Dev funny', ':id' => 1));

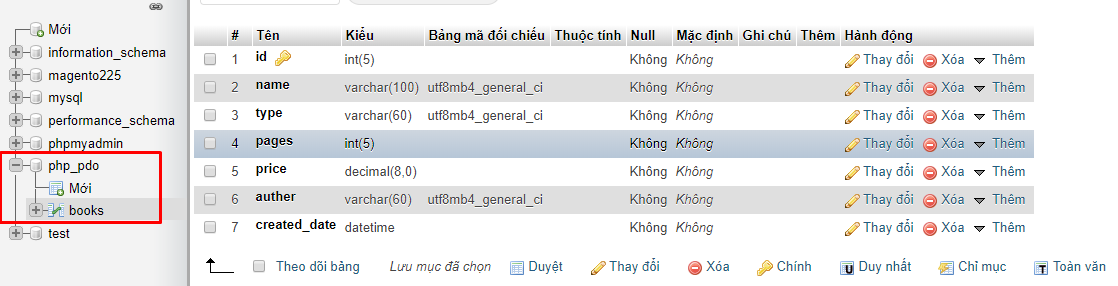
* 1. Delete

$sth = $conn->prepare('delete from users where id = :id');

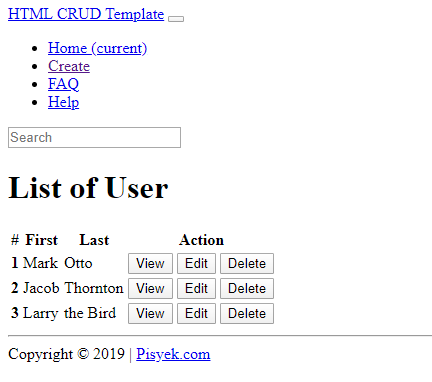
$sth->execute(array(':id' => 1));

### CRUD

**Bước 1**: Tạo cơ sở dữ liệu



**Bước 2**: Tạo file HTML CRUD



* Github: [git@github.com:quangnd512/php\_Core.git](mailto:git@github.com:quangnd512/php_Core.git)

**Bước 3**: Tạo kết nối PDO

* Tạo class DbConnect.php dùng để kết nối cơ sở dữ liệu.
* Truy cập: <http://www.kjetil-hartveit.com/blog/1/setter-and-getter-generator-for-php-javascript-c%2B%2B-and-csharp> để lấy set get các trường trong database

Tạo class Book.php để tạo các phương thức thêm sửa xoá với database

<?php

class Book

{

    protected $id;

    protected $name;

    protected $type;

    protected $pages;

    protected $price;

    protected $auther;

    protected $created\_date;

    private $tableName = 'books';

    private $dbConn;

    public function setId($id)

    {$this->id = $id;}

    public function getId()

    {return $this->id;}

    public function setName($name)

    {$this->name = $name;}

    public function getName()

    {return $this->name;}

    public function setType($type)

    {$this->type = $type;}

    public function getType()

    {return $this->type;}

    public function setPages($pages)

    {$this->pages = $pages;}

    public function getPages()

    {return $this->pages;}

    public function setPrice($price)

    {$this->price = $price;}

    public function getPrice()

    {return $this->price;}

    public function setAuther($auther)

    {$this->auther = $auther;}

    public function getAuther()

    {return $this->auther;}

    public function setCreated\_date($created\_date)

    {$this->created\_date = $created\_date;}

    public function getCreated\_date()

    {return $this->created\_date;}

    public function \_\_construct()

    {

        require\_once 'DbConnect.php';

        $db = new DbConnect;

        $this->dbConn = $db->connect();

    }

* Lưu ý: Các biến là các trường trong bảng. Và ta có thể khai báo các hàm thêm sửa xoá trong class này.

**Bước 4**: Liên kết các trường input trong html với database để khi nhấn nút submit các thông tin điền trong input sẽ được chuyển vào database.

* Tạo một class Action.php để điều hướng nút submit tới đó và xử lý dữ liệu lưu trong field và lưu vào trong database

<?php

require\_once('Book.php');

class Action

{

    function \_\_construct()

    {

        switch ($\_POST['submit']) {

            case 'insert':

                $objBook = new Book;

                $objBook->setName($\_POST['Iname']); //name field

                $objBook->setType($\_POST['Itype']);

                $objBook->setPages($\_POST['Ipages']);

                $objBook->setPrice($\_POST['Iprice']);

                $objBook->setAuther($\_POST['Iauther']);

                $objBook->setCreated\_date(date('Y-m-d H:i:s'));

                if ($objBook->insert()) {

                    header('location: index.php?insert=1');

                }else{

                    header('location: index.php?insert=0');

                }

                break;

            default:

                header('location: index.php');

                break;

        }

    }

}

if (isset($\_POST['submit'])) {

    $objAct = new Action;

}else{

    header('location: index.php');

}

**Bước 5**: Load dữ liệu từ database ra ngoài trang index.php

* Để kiểm tra các dữ liệu load trong database ta thêm return print\_r($books); exit; vào hàm load dữ liệu sau đó gọi hàm đó trong trang index.php.

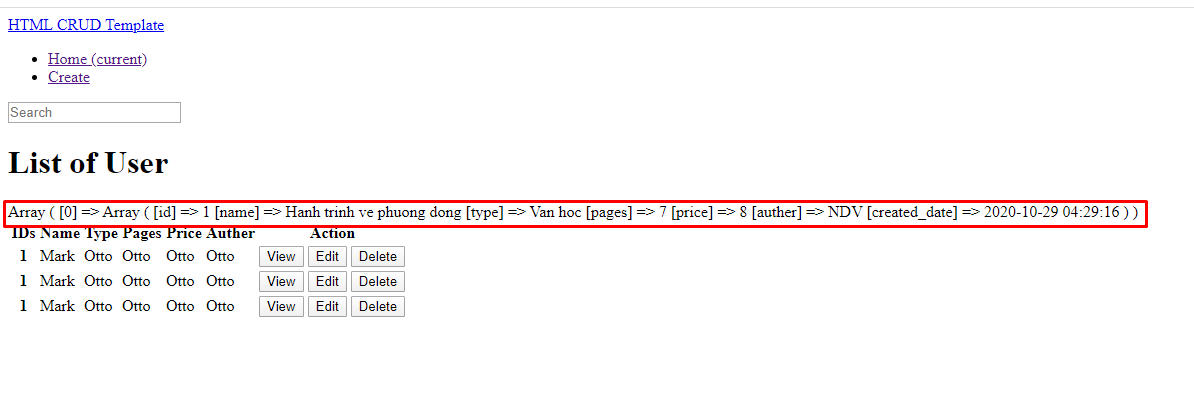
<?php

require\_once("Book.php");

    $objBook = new Book;

    $objBook->getAllBooks();

?>



* Để load data ta vào trang index.php để khai báo

<table class="table table-striped table-hover">

<thead>

     <tr>

         <th scope="col">IDs</th>

            <th scope="col">Name</th>

            <th scope="col">Type</th>

            <th scope="col">Pages</th>

            <th scope="col">Price</th>

             <th scope="col">Auther</th>

             <th scope="col">Action</th>

         </tr>

     </thead>

     <tbody>

      <?php

          require\_once("Book.php");

             $objBook = new Book;

             $books = $objBook->getAllBooks();

             foreach ($books as $book) {

              echo "<tr>";

                  echo "<th scope='row'>".$book['id']."</th>";

                     echo "<td>".$book['name']."</td>";

                     echo "<td>".$book['type']."</td>";

                     echo "<td>".$book['pages']."</td>";

                     echo "<td>".$book['price']."</td>";

                     echo "<td>".$book['auther']."</td>";

         ?>

                <td>

                 <a href="View/view.php"><button class="btn btn-primary btn-sm">View</button></a>

                    <a href="View/edit.php"><button class="btn btn-outline-primary btn-sm">Edit</button></a>

                    <button class="btn btn-sm">Delete</button>

                </td>

         <?php

          echo "</tr>";

             }

         ?>

     </tbody>

</table>

**Bước 6**: Tạo chỉnh sửa dữ liệu trong database

**Bước 7**: Xoá dữ liệu trong database

### ORM

### Doctrine

# Các mẫu thiết kế

## SOLID

## MVC

# Zend frameword