Như ở phần cuối của bài static và final trong PHP mình có nói là *giữa static và self nó có khác nhau ở trong một số trường hợp*. Và để chứng minh cho điều đó thì bài hôm nay mình sẽ giới thiệu với mọi người những sự khác nhau đó.

**1, Tổng quan.**

Nhìn chung thì cả self và static đều dùng để gọi các thành phần tĩnh trong đối tượng, nhưng nếu chỉ đơn thuần như trong nội bộ class thì cả 2 keywords này đều cho ra kết quả tốt.

**VD**:

<?php

class ConNguoi

{

private static $name = 'ConNguoi';

public static function getName()

{

echo self::$name;

echo '<br>';

echo static::$name;

}

}

ConNguoi::getName();

**Kết Quả**:

ConNguoi

ConNguoi

Vậy liệu nó có cho ra các giá trị khác nhau khi chúng ta sử dụng tính kế thừa trong class? Để biết rõ hơn thì chúng ta sẽ tạ ra các ví dụ để so sánh nó.

**2, So Sánh và kết luận.**

Vẫn là class ConNguoi như trên nhưng chúng ta sẽ khai báo thêm một class NguoiLon kế thừa class ConNguoi và override lại thuộc tính $name như sau:

<?php

class ConNguoi

{

private static $name = 'ConNguoi';

public static function getName()

{

echo self::$name;

echo '<br>';

echo static::$name;

}

}

class NguoiLon extends ConNguoi

{

private static $name = 'NguoiLon';

}

NguoiLon::getName();

Sau khi chạy dòng trên thì mình thu được kết quả như sau:

ConNguoi

Fatal error: Cannot access private property NguoiLon::$name

-Như các bạn đã thấy: Đối với self thì kết quả chạy như bình thường, còn với static thì sao nó lại báo là không thể truy cập vào thuộc tính private mà lại là NguoiLon::$name, phải chăng thằng static này đại diện cho đối tượng hiện tại nên không thể truy xuất được đến thuộc tính $name. Thôi được rồi, để chắc ăn hơn thì mình thử đổi visibility của biến $name thành protected xem sao?

<?php

class ConNguoi

{

protected static $name = 'ConNguoi';

public function getName()

{

echo self::$name;

echo '<br>';

echo static::$name;

}

}

class NguoiLon extends ConNguoi

{

protected static $name = 'NguoiLon';

}

// NguoiLon::getName();

$a = new NguoiLon();

$a->getName();

Và đây là kết quả mình nhận được:

ConNguoi

NguoiLon

Giờ đã chạy được ngon lành, nhưng khi nó lại cho ra 2 kết quả khác nhau. Từ 2 ví dụ trên chúng ta **tạm đưa ra  kết luận** là static nó có nguyên tắc gần như [$this](http://toidicode.com/this-va-self-trong-php-111.html), là **đều truy xuất đến đối tượng hiện tại**.

Để ok hơn nữa thì mình sẽ tiếp tục tạo ra ví dụ sau và chạy nó:

<?php

class ConNguoi

{

protected static $name = 'ConNguoi';

public static function getSelf()

{

return new self;

// or

// return new static();

}

public static function getStatic()

{

return new static;

// or

// return new static();

}

}

class NguoiLon extends ConNguoi

{

}

echo get\_class(NguoiLon::getSelf()); //ConNguoi

echo get\_class(NguoiLon::getStatic()); //NguoiLon

Ok, nó đã chạy được và cho ra kết quả giống với kết luận ở trên. Lúc này thì chẳng cần phải ngại ngần gì nữa mà không dám kết luận khẳng định.

**Kết Luận**

* **Self**: Truy xuất đến class khai báo nó.
* **Static**: Truy xuất đến đối tượng hiện tại.