**Câu hỏi phỏng vấn chưa trả lời được**

**I. PHP:**

1. **Sự khác nhau đặc biệt giữa abstract class và Interface:**

|  |  |
| --- | --- |
| Abstract | Interface |
| Nó được xem như là một bản thiết kế cho class. Các class con chỉ có thể extent một abstract class | Được xem như là một bản thiết kế cho một chức năng. Nó ko cần cùng về bản chất. Các class con có thể implement nhiều các Interface |

1. **Giới hạn quyền của thuộc tính và phương thức trong php: private, protected, public:**
   * **Private**: Giới hạn hẹp nhất của phương thức và thuộc tính, Nó chỉ sử dụng đc **bên** **trong** class khai báo nó. **Bên** **ngoài** class không thể nào sử dụng được kể cả lớp kế thừa. Nếu muốn truy cập vào nó ở bên ngoài class thì phải sử dụng phương thức là public

VD: class person{

private $name;

private function run(){

return “người chạy bằng 2 chân”

}

//Khi khai báo class muốn truy cập đến phương thức và thuộc tính là private như kia thì ko thể. Do đó ta phải thêm phương thức là public gọi đến phương thức và thuộc tính priave kia.

}

//Tạo phương thức public gọi đến thuộc tính private $name:

Public function setName($name){

$this->name = $name;

}

Public function getName(){

Return $this->name;

}

//Tạo phương thức public gọi đến phương thức priave run():

Public function getRun(){

Return $this->run();

}

//Khởi tạo đối tượng và đối tượng khởi tạo có thể gọi đc các phương thức và thuộc tính private kia nhờ các phương thức khai báo là public

$a = new person();

$a->setName(“Huy”);

Echo $a->getName();//Xuất ra màn hình là Huy;

Echo $a->getRun();//Xuất ra màn hình “Người chạy bằng 2 chân”’;

* **Protected:** Rộng hơn private, khi phương thức và thuộc tính khai báo là protected thì nó ngoài việc sử dụng trong class mà các class con kế thừa nó cũng có thể sử dụng được. Nó không thể sử dụng ngoài class. Để sử dụng nó ngoài class thì trong class cha hoặc con phải sử dụng phương thức là public gọi đến thuộc tính và phương thức protected. Do đó khi khởi tạo đối tượng thì đối tượng khởi tạo có thể gọi các thuộc tính và phương thức public;

VD: class person{

Protected $name;

Protected function run(){

Return “Người chạy bằng 2 chân”;

}

}

Class Male extends person{

Public function setName($name){

$this->name = $name // $this->name là gọi đến thuộc tính name //của class cha

}

Public function getName(){

Return $this->name;

}

Public function getRun(){

Return $this->run();

}

}

$a = new Male();

$a->setName(“Huy”) ;

$a->getName(); // Xuất ra màn hình: Huy

$a->getRun(); // Xuất ra mà hình : Người chạy bằng 2 chân;

* **Public:** Đây là mức truy cập rộng nhất. Các phương thức hay thuộc tính khai báo là public thì nó có thể sử dụng được cả trong hay ngoài class

1. **Isset và Empty** 
   * **Isset:** Kiểm tra biến có tồn tại ko. Tồn tại thì trả về True còn ko tồn tại thì trả về False. Nếu biến có giá trị là null thì vẫn coi là ko tồn tại.
     + **Các trường hợp sử dụng:** 
       - Sử dụng trong form: khi lấy thông tin trong form phải kiểm tra biến có tồn tại ko rồi lấy. Nếu ko ktra thì người dùng sử dụng firebug thay đổi một số name của các thẻ input dẫn đến chương trình lỗi.
       - Kiểm tra biến trong thuật toán phân trang. Trong thuật toán phân trang ta lấy page trên URL để xác định record trên trang đó. Chính vì hiển thị trên URL nên nó rất nguy hiểm, nếu ta ko kiểm tra biến có tồn tại hay ko thì người dùng chỉ cần thêm “page=x” chương trình sẽ lỗi ngay
   * **Empty:** Kiểm tra một biến nào đó có giá trị rỗng hoặc chưa được khởi tạo hay không. Rỗng trả về True, Ko rỗng trả về False
     + **Các trường hợp trả về biến rỗng:**
       - $a = 0 hoặc $a = ‘0’
       - $a = null
       - $a = “”
       - $a = FALSE
       - $a không tồn tại.
     + **Các trường hợp sử dụng:** 
       - Sử dụng khi validate dữ liệu trong form
2. **Sự tối ưu giữa câu lệnh điều kiện switch vs if else:**
   * Hầu như ko có sự khác biệt giữa switch vs if else. Nhưng:
   * Nếu có nhiều if else liên tục thì ta nên sử dụng switch case vì:
     + Switch – case sẽ giúp cấu trúc chương trình rõ ràng, dễ đọc dễ hiểu hơn if else
     + Switch – case sẽ kiểm tra 1 biến với 100 trường hợp khác nhau, còn if else thì kiểm tra 100 điều kiện với 100 trường hợp khác nhau. Chính vì thế nếu ta có 1000 điều kiện thì dùng if else sẽ rất phức tạp
3. **Upload file:** Để Upload file ta cần lưu ý những điều sau:
   * Trong Form upload cần phải có **method=”POST”** và đặc biệt phải có **enctype=”multipart/form-data”**
   * Input phải có **type=”file**” thì mới có thể upload được file
   * Để xử lý file ta cần phải kiểm tra biến toàn cục có tồn tại hay rỗng không (isset hoặc empty)**: $\_POST[‘thuộc tính name của button type=”submit” ’]**. Tiếp theo ta kiểm tra các biến toàn cục của file: **$\_FILE[‘filename’][‘error’]** xem upload file có lỗi ko rồi tiếp đếm sử dụng các $\_FILE[‘’] khác;
   * Trong $\_FILE[‘’] có các thành phần sau:
     + **$\_FILE[‘filename’][‘tmp\_name’]**: Thư mục tạm thời lưu file
     + **$\_FILE[‘filename’][‘name’]:** Tên của file
     + **$\_FILE[‘filename’][‘type’]:** Kiểu file
     + **$\_FILE[‘filename’][‘error’]:** Mã lỗi. Nếu error == 0 thì ko có lỗi
     + **$\_FILE[‘filename’][‘size’]:** kích thước của file được tính bằng chia cho 1024
   * Khi ta Upload file lên server thành công thì ta phải di chuyển file sang một thư mục khác vì nếu ko di chuyển file đó sẽ mất trong một khoảng thời gian. Để di chuyển file ta phải cần: Tạo một thư mục trong chương trình sau đó sử dụng hàm **move\_upload\_file($tmp\_name, thư mục ta tạo)**
     + VD: Tạo một thư mục là uploads:
       - Sử dụng hàm **move\_upload\_file($\_FILE[‘filename’][‘tmp\_name’], ‘uploads/’ . $\_FILE[filename][‘name’])**
4. **Tham trị và tham chiếu:**
   * **Tham trị**: là một biến bình thường. Khi ta tác động và thay đổi nó trong hàm thì khi kết thúc hàm biến đó vẫn ko thay đổi giá trị.
     + **VD:**  $a = 10;

Function tong($b){

Return $b \* $b;

}

Echo tong($a) // Sử dụng biến a trong hàm // Hiển thị ra 100

Echo $a// gọi biến a ngoài hàm và nó ko thay đổi giá trị = 10

* **Tham chiếu:** một biến được tham chiếu bởi một biến khác, biến này thay đổi giá trị thì biến kia thay đổi theo. Kí hiệu biến là tham chiếu **&$**. Sử dụng trong trường hợp số lượng tham số quá lớn và muốn tốc độ nhanh, tránh việc cấp phát dữ liệu thì nên dùng tham số.

**II. LARAVEL:**