BÀI 5

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PHP

GV: Lê Thị Bích Hằng
Khoa CNTT – Trường ĐHNT

Nội dung

- Class và object
- Khai báo class, member và method
- Hàm tạo và hàm hủy
- Sử dụng constant member
- Sử dụng static member/method
- Lớp kế thừa

Khai báo Class và Object

- Khai báo lớp (class)

```
class tên_lớp
{
    //các thuộc tính và phương thức
}
```

- Tạo và hủy một đối tượng (object)
  - Tạo một đối tượng:

```
$tên_biến = new tên_lớp();
```

  - Hủy một đối tượng:

Đối tượng sẽ tự động bị hủy khi không còn tham chiếu nào đến nó, hoặc dùng lệnh:

```
$tên_biến = NULL;
```

Khai báo member và method

```
<?php
    class Cart
    {       var $items;  // Items in our shopping cart
            // Add $num articles of $artnr to the cart
            function add_item ($artnr, $num){
                        $this->items[$artnr] += $num;
            }
            // Take $num articles of $artnr out of the cart
            function remove_item ($artnr, $num){
                 if ($this->items[$artnr] > $num) {
                       $this->items[$artnr] -= $num;
                        return true;
                  } else  return false;

            }
    }
?>
```

- Lớp Cart ở đây là một kiểu dữ liệu, vì vậy có thể tạo một biến có kiểu này với toán tử new

```
$cart = new Cart;
$cart->add_item("10", 1);
```

Khai báo member và method

- Các từ khóa khai báo

  public: có thể sử dụng bên ngoài lớp

  private: chỉ sử dụng cục bộ bên trong lớp

  protected: sử dụng được bởi các lớp kế thừa

- Một số quy tắc chung

  - Không thể khai báo hai phương thức (method) trùng tên

  - Method phải được khai báo ngay bên trong khai báo lớp

  - Dùng biến giả **$this** để truy xuất các thuộc tính (member) và method trong lớp

  - Dùng toán tử -> để truy xuất đến member và method

Ví dụ - public, private, protected property

```
<?php
class Myclass {
    public $font_size ="18px";
    public $font_color = "blue";
    public $string_name = "w3resource";
    public function customize_print() {
        echo "<p style=font-size:“ . $this->font_size . ";color:“ .
                    $this->font_color.";>". $this->string_name."</p>";
    }
}
$f = new MyClass;
$f->font_size = "20px";
$f->font_color = "red";
$f->string_name = "Object Oriented Programming";
echo $f->customize_print();
?>
```

Ví dụ - public, private, protected property

```php
class Myclass
{
 // Declare $font_size as Public property
 public $font_size ="18px";
 // Declare $font_color as Private property
 private $font_color = "blue";
 // Declare $string_name as Protected property
 protected $string_name = "w3resource";
 // Declare a method to print properties value. This is public.
 function property_print()
 {
 echo $this->font_size;
 echo $this->font_color;
 echo $this->string_name;
 }
}
$obj = new MyClass;
echo $obj->font_size; //Display 18px
echo $obj->font_color; //Fatal error: Cannot access private property
Myclass::$font_color in F:\wamp\..
echo $obj->string_name; //Fatal error: Cannot access protected property
Myclass::$string_name in F:\wamp\..
$obj->property_print(); //Display 18pxbluew3resource
?>
```

Ví dụ - public, private, protected method

```php
<?php
class Myclass
{   public function my_public_method()
    {   echo "This is a Public method";
    }
    private function my_private_method()
    {   echo "This is a Private method";
    }
    protected function my_protected_method()
    {   echo "This is a Protected method";
    }
 // This is public
    function test()
    {   $this->my_public_method();
        $this->my_private_method();
        $this->my_protected_method();
    }
}
 $obj = new MyClass;
 $obj->my_public_method(); //Display This is a Public method
 $obj->my_private_method();//Fatal error: Call to private method Myclass::my_private_method() ...
 $obj>my_protected_method();//Fatal error: Call to undefined function my_protected_method() in ...
 $obj->test(); //Display This is a Public methodThis is a Private methodThis is a Protected method
?>
```

Hàm tạo và hàm hủy

Khai báo

```
public function __construct([danh sách tham số])
{
    // khởi tạo giá trị các member;
}

public function __destruct()
{
    // dọn dẹp;
}
```

- *constructor được tự động thực hiện khi đối tượng được tạo*
- *destructor được tự động thực hiện khi đối tượng bị hủy*

Ví dụ - constructor

```php
<?php
class Myclass
{
    private $font_size;
    private $font_color;
    private $string_value;
    function __construct($font_size,$font_color,$string_value)
    {  $this->font_size = $font_size;
       $this->font_color = $font_color;
       $this->string_value = $string_value;
       $this->customize_print();
    }
   // Declare a method for customize print
    function customize_print()
    {  echo "<p style=font-size:".$this->font_size.";color:".$this->font_color.";>".$this->string_value."</p>";
    }
}
$a = new MyClass('30px','red','Object Oriented Programming');
$b = new MyClass('25px','blue','Object Oriented Programming');
$c = new MyClass('20px','green','Object Oriented Programming');
?>
```

Ví dụ - destructor

```php
<?php
class MyClass
{
 function __construct()
  {
    echo 'w3resource'.'<br>';
    $this->name = "MyClass";
  }
 function __destruct()
  {
    echo "Destroying " . $this->name . "<br>";
  }
}
$obj = new MyClass();
?>
```

Sử dụng constant member

- Khai báo

  const TÊN_HẰNG = giá trị;

- Truy xuất

  tên_lớp :: TÊN_HẰNG *// ngoài lớp*

  self :: TÊN_HẰNG *// trong lớp*

Ví dụ - Sử dụng constant member

```php
<?php
class MyClass
{
    const constant1 = 'PHP Class Constant';
    function PrintConstant() {
        echo  self::constant1 . "<br>";
    }
}

echo MyClass::constant1 . "<br>";
$classname = "MyClass";
echo $classname::constant1 . "<br>";
$class = new MyClass();
$class->PrintConstant();
echo $class::constant1."<br>";
?>
</body>
</html>
```

Sử dụng static member

- Dữ liệu tĩnh là loại dữ liệu được sử dụng ở dạng toàn cục, dù nó được xử lý ở bất kỳ file nào trong cùng một chương trình đều được lưu lại, ta có thể gọi chúng là thành viên tĩnh. Mỗi thành viên đều có các mức truy cập private, public và protected bình thường
- Khai báo

(public/private/protected) static $thuộc_tính;

- Truy xuất

tên_lớp :: $thuộc_tính // ngoài lớp

self :: $thuộc_tính // trong lớp

Ví dụ - Sử dụng static member

```php
class Animal
{    protected $_name = 'No name';
     function setName($name){
        $this->_name = $name;
    }
    function getName(){
        return $this->_name;
    }
}
$cat = new Animal();
$cat->setName('Cat');
echo $cat->getName();
//kết quả: Cat

$mouse = new Animal();
echo $mouse->getName();
//kết quả: No name
```

Ví dụ - Sử dụng static member

```php
class Animal
{    protected static $_name = 'No name';
     function setName($name){
         Animal::$_name = $name;
    }
    function getName(){
         return Animal::$_name;
    }
}
$cat = new Animal();
$cat->setName('Cat');
echo $cat->getName();
//kết quả: Cat

$mouse = new Animal();
echo $mouse->getName();
//kết quả: Cat
```

Sử dụng static method

- Khai báo

(public/private/protected) static function phương_thức(…)

- Truy xuất

tên_lớp :: phương_thức(…); *// ngoài lớp*

self :: phương_thức(…) ; *// trong lớp*

Sử dụng static method

- Một số vấn đề thông dụng khi sử dụng static method
  - Truy xuất trực tiếp không cần khởi tạo object
  - Gọi các hàm tĩnh trong nội bộ của class
  - Không sử dụng từ khóa $this
  - Kế thừa khi sử dụng static

Ví dụ - Sử dụng static method

Truy xuất trực tiếp không cần khởi tạo object

```php
class Animal
{
    protected static $_name = 'No name';

    public static function setName($name){
        self::$_name = $name;
    }

    public static function getName(){
        return self::$_name;
    }
}

Animal::setName('Cat');
echo Animal::getName();
// Kết quả: Cat
```

Ví dụ - Sử dụng static method

Gọi các hàm tĩnh trong nội bộ của class

```
class Animal
{
    protected static $_name = 'No name';
    public static function setName($name){
        self::$_name = $name;
    }
    public static function getName(){
        return self::$_name;
    }
    public static function all($name){
        self::setName($name);
        echo self::getName();
    }
}
Animal::all('Cat');
// Kết quả: Cat
```

Ví dụ - Sử dụng static method

Không sử dụng từ khóa $this

```
class Animal{
    protected $_age = '';
    protected static $_name = 'No name';
    public static function setInfo($name, $age)
    {
        // Đúng
        self::$_name = $name;

        // Sai vì $this không tồn tại
        $this->_name = $name;

        // Sai vì $this không tồn tại
        $this->_age = $age;

        // Sai vì thuộc tính $_age không phải là tĩnh
        Animal::$_age = $age;
    }
}
```

Ví dụ - Sử dụng static method

Kế thừa khi sử dụng static

```
class Animal{
    protected static $_name = '';
    public static function setName($name){
        self::$_name = $name;
    }
    public static function getName(){
        return self::$_name;
    }
}
class Cat extends Animal{
    public static function setName($name) {
        parent::setName($name);
    }
}

Cat::setName('Cat');
echo Cat::getName();
// Kết quả: Cat
```

Lớp kế thừa (Inheritance)

- Khai báo lớp con

```
class lớp_con extends lớp_cha
{
    các thuộc tính và phương thức
}
```

Tất cả các member và method được khai báo public hay protected trong lớp cha được thừa kế và có thể sử dụng trong lớp con

Ví dụ - Inheritance

```php
<?php
class Myclass
 {   protected $font_size;
     protected $font_color;
     protected $string_value;
     function __construct($font_size,$font_color,$string_value)
     {   $this->font_size = $font_size;
         $this->font_color = $font_color;
         $this->string_value = $string_value;
         $this->customize_print();
     }
     public function customize_print()
     {  echo "<p style=font-size:".$this->font_size.";color:".$this->font_color.";>".$this->string_value."</p>";
     }
 }
// Define a subclass
class Mysubclass extends Myclass
 {  public function customize_print()
     {
         echo "<p style=font-size:".$this->font_size.";color:".$this->font_color.";text-decoration:underline;>".
         $this->string_value."</p>";
     }
 }
// Create objects and passes parameters
 $p = new Myclass('20px','red','Object Oriented Programming');
 $s = new Mysubclass('15px','green','Object Oriented Programming');
?>
```

Lớp kế thừa - phương thức nạp chồng

- **Gọi một method lớp cha**

parent::phương_thức(…)

Bằng cách định nghĩa lại một phương thức đã có ở lớp cha, tất cả các lời gọi đến phương thức này mà không chỉ định rõ như trên sẽ được hiểu là gọi phương thức có cùng tên của lớp con

Lớp kế thừa - phương thức nạp chồng

```
class Person{
        protected $name;
        protected $age;
        protected $sex;
        public function GetInfo(){
                echo $this->name;
                echo $this->age;
                echo $this->sex;
        }
}
 class Teacher extends Person{
        public function Teach(){
                $this->GetInfo();
        }
}
 class Student extends Person{
        public function Learn(){
        parent::GetInfo();
        }
}
```

Lớp kế thừa - Abstract

- Khai báo lớp trừu tượng

```
abstract class lớp_trừu_tượng
{
    // các thuộc tính
    abstract public function Function_Name(…);
    …
    // các phương thức khác
}
```

- ✓ *Lớp trừu tượng có thể chứa thuộc tính, các phương thức khác, và phải có phương thức trừu tượng*
- ✓ *Không thể tạo đối tượng trực tiếp từ lớp trừu tượng*
- ✓ *Lớp con bắt buộc phải định nghĩa các phương thức trừu tượng của lớp cha*

Ví dụ - abstract

```php
<?php
    abstract class employee{
          protected $empname;
          protected $empage;
          public function setdata($empname,$empage){
               $this->empname = $empname;
               $this->empage = $empage;
          }
          abstract function outputData();
    }

    class EmployeeData extends employee{
           public function __construct($name,$age){
                $this->setdata($name,$age);
           }
           function outputData(){
                echo $this->empname;
                echo $this->empage;
           }
    }
    $a = new EmployeeData("Hitesh","24");
    $a->outputData();
?>
```

Lớp kế thừa - ngăn kế thừa và nạp chồng

- Lớp không thể kế thừa

  final class Class_Name { … }

- Phương thức không thể nạp chồng

  final public function Function_Name(…) { … }

Lớp kế thừa - Interface

- interface MyInterface

```
{  function   method1();
   function    method2();
}
```

Lớp interface chỉ chứa các phương thức trừu tượng

- class MyClass implements MyInterface{

```
        function method1()
        { // definition of method1
        }
        function method2()
        { // definition of method2
        }
}
```

Lớp con kế thừa từ lớp interface phải cài đặt các phương thức ở lớp interface

Ví dụ - Interface

```php
<?php
        interface animal
        {
             function sound();
        }
        interface benefits
        {
             function protect();
        }
        class ABC implements animal, benefits{
        public function sound(){
             //Thực thi mã lệnh
        }
        public function protect(){
             //Thực thi mã lệnh
         }
}
?>
```

Một số lưu ý

- Phép gán đối tượng

```
$a = new lớp();
$b = $a; // $a và $b cùng trỏ đến một thực thể của lớp
```

- Nhân bản đối tượng

```
$b = clone $a;
// $b được tạo mới và sao chép giá trị từ $a
```

- Phương thức __clone():

  Sau khi sao chép toàn bộ các giá trị từ $a vào $b, phương thức này sẽ được tự động gọi nếu được định nghĩa trong lớp của $a và $b

Ví dụ - clone

```php
<?php
class MyClass
{
      public $x;
      private $y;
      function __construct($x, $y)
      {
            $this->x = $x;
            $this->y = $y;
      }
      function __clone()
      {     $this->x = "z";
      }
}
$a = new MyClass("w3resource", "com"); // create a new object
$b = clone $a; //clone of the object
var_dump($a);
echo '<br>';
var_dump($b);
?>
```

Kết quả:

object(MyClass)#1 (2) { ["x"]=> string(10) "w3resource" ["y":"MyClass":private]=> string(3) "com" }
object(MyClass)#2 (2) { ["x"]=> string(1) "z" ["y":"MyClass":private]=> string(3) "com“}