TRƯỜNG ĐẠI HỌC SƯ PHẠM HÀ NỘI

Chương 3
Lập trình hướng đối tượng

Nguyễn thị Quỳnh Hoa - Khoa CNTT- ĐH Sư phạm Hà Nội

NỘI DUNG

- Lập trình truyền thống
- Lập trình hương đối tượng
- Lớp – Thuộc tính – Phương thức của đối tượng
- Trừu tượng hóa dữ liệu
- Tính kế thừa trong PHP
- Các mức truy cập
- Hàm khởi tạo và hàm hủy trong PHP

LẬP TRÌNH TRUYỀN THỐNG

- Lập trình không có cấu trúc
- Lập trình hướng thủ tục
- Lập trình hướng modun

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

- Đối tượng là những sự vậy, hiện tượng có những thuộc tính, phương thức giống nhau
- Lập trình hướng đối tượng (OOP-Object-Oriented Programming) : Là phương pháp lập trình giúp tăng năng suất, đơn giản hóa độphức tạp khi bảo trì, mở rộng phần mềm bằng cách cho phép lập trình viên tập trung vào các đối tượng phần mềm giống như trong thực tế

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

- Các tính chất :
  - Trừu tượng hóa : là quấ trình đơn giản hóa một đối tượng,là việc xác định những thuộc tính, phương thức cần thiết cho một chương trình
  - Thừa kế: Là kỹ thuật cho phép lớp này có thể kế thừa các phương thức và thuộc tính của lớp khác
  - Đa hình : là kỹ thuật cho phép lớp này có thể viết lại các thuộc tính hay phương thức của lớp khác
  - Đóng gói : Là tính chất không cho phép người dùng hay đối tượng khác thay đổi dữ liệu thành viên của đối tượng nội tại. Chỉ có các thành viên trong đối tượng đó mới được phép thay đổi

ĐỐI TƯỢNG VÀ LỚP ĐỐI TƯỢNG

- Các đối tượng (objects) được định nghĩa thông qua :
  - Các thông số cơ bản của đối tượng (các thuộc tính) được thể hiện thông qua các biến
  - Các hành vi (phương thức) được thể hiện thông qua các hàm
- Class (lớp) định nghĩa các thuộc tính và các hành vi của các đối tượng có chung tên các biến và hàm

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PHP

- Có 2 dạng :
  - Các lớp đối tượng đã được xây dựng sẵn
    - Simple XML
    - PDO
    - SOAP
    - DOM
    - …
  - Các lớp đối tượng do người dùng định nghĩa

LỚP TRONG PHP

- Khai báo

```
class tên_lớp
    {
        // Danh sách các biến, lớp... (thuộc tính)
        // Danh sách các hàm (phương thức)
    }
```

- Sử dụng :
  - $đối_tượng=new tên_class
  - //Sử dụng $đối_tượng-> thuộc tính, phương thức trong class

TRỪU TƯỢNG HÓA DỮ LIỆU

- Xét bài toán : Xây dựng chương trình quản lý sinh viên
- Đối tượng : Các sinh viên
- Dữ liệu : Hộ tên, ngày sinh, địa chỉ, giới tính, học lớp,…
- Hành động : Thêm sinh viên, Đuổi sinh viên, Đăng ký học phần cho sinh viên,…

TRỪU TƯỢNG HÓA DỮ LIỆU

- Các cơ chế :
  - Public : Các đối tượng từ bên ngoài class có thể truy cập, hỗ trợ sự kế thừa. Khai báo : **public $tên_biến;**
  - Protected :Các đối tượng từ bên ngoài class không thể truy cập, hỗ trợ sự kế thừa. Khai báo : **protec $tên_biến;**
  - Private : Các đối tượng từ bên ngoài class không thể truy cập, không hỗ trợ kế thừa. Khai báo : **private $tên_biến;**
  - Set, get : Cho phép các đối tượng bên ngoài class truy cập, xử lý các cơ chế private và protected

VÍ DỤ

TÍNH KẾ THỪA TRONG PHP

- Giả sử có 2 lớp **Động Vật** và **Con Trâu** có những thuộc tính và phương thức sau

Động Vật	Con Trâu
Thuộc Tính: - Mắt - Mũi - Miệng - Chân - Giới tính **Phương Thức:** - Ăn - Ngủ - Chạy - La hét	**Thuộc Tính:** - Mắt - Mũi - Miệng - Chân - Giới Tính - Sừng **Phương Thức:** - Ăn - Ngủ - Chạy - La Hét

- Cú pháp :

```
1  class classCon extends classCha {
2  }
```

TÍNH KẾ THỪA TRONG PHP

- Để kế thừa một lớp trong PHP, ta dùng từ khóa **extends**, theo sau là tên lớp cha :

```
Class con extends cha {

}
```

- Gọi các phương thức và thuộc tính của lớp cha
  - Gọi bên trong lớp con : $this->thuoctinh, $this->phuong>thuc
  - Gọi từ bên ngoài lớp : $đối_tượng = new tên_class
    - $đối_tượng->thuộc tính , phương thức trong class

NẠP CHỒNG

- Cho phép viết lại các phương thức có cùng tên với phương thức của lớp cha
- Có thể tham chiếu tới các phương thức của lớp cha đã bị ghi đè bằng cách sửa dụng cú pháp

parent::tên_phương_thức(tham số)

- Từ khóa **Final** đặt trước tên phương thức của lớp cha sẽ giúp cho phương thức đó ko bị ghi đè

VÍ DỤ

- Xét ví dụ sau :

```php
class con_nguoi{
	public function an(){
		return "Anh ấy đang ăn cơm";
	}
}
class sinh_vien extends con_nguoi{
	public function an(){
		return "Anh ấy đang không có gì để ăn";
	}
}

$sinhVien = new sinh_vien();
echo $sinhVien->an();

// kết quả thu được là "Anh ấy đang không có gì để ăn"
```

```php
class con_nguoi{
	public function an(){
		echo "Anh ấy đang ăn cơm";
	}
}
class sinh_vien extends con_nguoi{
	public function an(){
		parent::an();
		return "Anh ấy đang không có gì để ăn";
	}
}

$sinhVien = new sinh_vien();
echo $sinhVien->an();

// kết quả thu được là "Anh ấy đang ănAnh ấy đang không có gì để ăn"
```

LỚP TRỪU TƯỢNG TRONG PHP

- Cho phép định nghĩa các lớp và các phương thức một cách trừu tượng
- Sử dụng từ khóa **abstract** ở trước các lớp và các phương thức trừu tượng
- Không thể tạo ra một thể hiện của một lớp trừu tượng
- Ở lớp cha, các phương thức trừu tượng chỉ có tên và phải được đặt ở chế độ **public** hoặc **protected**. Các phương thức trừu tượng sẽ được định nghĩa chi tiết ở lớp con với các chế độ bảo vệ tương ứng như ở lớp cha

LỚP TRỪU TƯỢNG TRONG PHP

- Khai báo lớp Abstract

```php
abstract class BaseClass
{
    // phương thức ở mức protected
    abstract protected function hello();

    // Phương thức ở mức public
    abstract public function hi();
}
```

```php
abstract class BaseClass
{
    // Phương thức này sai vì hàm hello là
    // hàm abstract nên không được code bên trong nó
    abstract protected function hello()
    {
        // dòng code
        echo 1;
    }

}
```

LỚP TRỪU TƯỢNG TRONG PHP

```php
abstract class BaseClass
{
    abstract protected function hello();
}

// Sai vì BaseClass là lớp Abstract nên không
// khởi tạo mới được
$base = new BaseClass();
```

```php
abstract class Person
{
    protected $ten;
    protected $cmnd;
    protected $namsinh;

    abstract public function showInfo();
}

// Lớp này sai vì chưa viết lại hàm showInfo
class CongNhan extends Person
{

}

// Lớp này đúng vì ta đã khai báo, viết lại
// đầy đủ các hàm abstract
class SinhVien extends Person
{
    public function showInfo(){

    }
}
```

HÀM VÀ LỚP FINAL

- Lớp **Final** là lớp được khai báo là lớp cuối cùng, không một lớp nào có thể kế thừa nó.

```php
// Lớp Filnal
final class Person
{
    protected $ten;
    protected $cmnd;
    protected $namsinh;
    public function showInfo()
    {
        echo 'freetuts.net';
    }
}

// Hàm này sẽ bị báo lỗi vì lớp SinhVien
// đã kế thừa một lớp Final, điều này là không thể
class SinhVien extends Person {
}

// Đoạn code này đúng vì lớp Final được
// sử dụng bình thường như các lớp khác
// chỉ có điều là không được kế thừa
$person = new Person;
$person->showInfo();
```

HÀM VÀ LỚP FINAL

- Ví dụ hàm **Final :**

```php
class Person
{
    protected $ten;
    protected $cmnd;
    protected $namsinh;
    final public function showInfo()
    {
        echo 'freetuts.net';
    }
}

// Lớp này đúng vì lớp Person không phải
// là một lớp final
class SinhVien extends Person {

    // Hàm này sai vì hàm showInfo
    // là hàm final trong lớp Person
    // nên không thể Override lại
    public function showInfo(){

    }

    public function Go()
    {
        // Đoạn code này đúng vì hàm final được
        // sử dụng bình thường
        $this->showInfo();
    }
}
```

HÀM DỰNG (HÀM TẠO)

- Khái niệm : Là một phwong thức được tự động kích hoạt khi đối tượng được khởi tạo
- Cách dùng : Có 2 cách khởi tạo

Khai báo tên trùng tên lớp

```php
class SinhVien
{
    function SinhVien() {
        echo 'Lớp Sinh Viên vừa được khởi tạo';
    }
}

// khởi tạo lớp SinhVien
$sinhvien = new SinhVien();
```

Khai báo với tên __Contruct

```php
class SinhVien
{
    function __construct() {
        echo 'Lớp Sinh Viên vừa được khởi tạo';
    }
}

// khởi tạo lớp SinhVien
$sinhvien = new SinhVien();
```

HÀM DỰNG (HÀM TẠO)

- Hàm khởi tạo trong kế thừa:
  - Nếu lớp con có hàm khởi tạo và lớp cha cũng có hàm khởi tạo : Hàm khởi tạo của lớp con sẽ được chạy, còn hàm khởi tạo ở lớp cha không được chạy
  - Nếu lớp con không có hàm khởi tạo, lớp cha có hàm khởi tạo : Hàm khởi tạo ở lớp cha sẽ được chạy
  - Lớp con có hàm khởi tạo, lớp cha không có hàm khởi tọa : Trường hợp này hàm khởi tạo lớp con sẽ được chạy

HÀM HỦY

- Khái niệm : Phương thức đạc biệt được thực hiện khi hủy một đối tượng
- Cách dùng : Khai báo môt hàm với tên là **__destruct();**
- Cú pháp

```
public function __destruct(){
        //Lệnh thực thi
}
```

- Hàm hủy trong kế thừa : Tương tự như hàm khởi tạo trong kế thừa

LƯU Ý

- Hàm hủy và hàm dựng ở lớp cha sẽ không được thực thi nếu như lớp con cũng có hàm hhwngj (hàm hủy)
- Để gọi hàm dựng (hoặc hàm hủy) ở lớp cha, cần sử dụng

Parent:__construct();

Parent: __destruct();

HẰNG CỦA LỚP

- Hằng của lớp là giá trị không đổi thuộc về lớp, không phải đói tượng được tạo từ lớp khai báo với từ khóa const

  Const RATE = 1;

- Truy cập tới hàng của lớp
  - Truy cập từ tong lớp : **self::constName**
  - Truy cập từ bên ngoài lớp **className::constName**
- Thuộc tính hằng luôn là public
- Thường được sử dụng để xác định tập hợp các tùy chọn được truyền cho phương thức trong lớp

THUỘC TÍNH VÀ PHƯƠNG THỨC TĨNH

- Thuộc tính tĩnh hay phương thức tĩnh là thuộc về một lwps chứ không thuộc đối tượng được tạo ra từ lớp
- Khai báo với từ khóa **static**
  - **Public** static $number = 0;
  - **Public static function () {};**
- Truy cập :
  - Self::$number;
  - className::$number;

THUỘC TÍNH VÀ PHƯƠNG THỨC TĨNH

- Xét ví dụ

```php
// Lớp động vật
class Animal
{
    protected $_name = 'Chưa có tên';

    function setName($name){
        $this->_name = $name;
    }

    function getName(){
        return $this->_name;
    }
}

// Phần 1: Con Vịt
$con_vit = new Animal();
$con_vit->setName('Con Vịt');
echo $con_vit->getName();
// Kết quả: Con Vịt

// Phần 2: Con Heo
$con_heo = new Animal();
echo $con_heo->getName();
// Kết quả: Chưa có tên
```

THUỘC TÍNH VÀ PHƯƠNG THỨC TĨNH

- Xét ví dụ

```php
// Lớp động vật
class Animal
{
    protected static $_name = 'Chưa có tên';

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

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

// Phần 1: Con Vịt
$con_vit = new Animal();
$con_vit->setName('Con Vịt');
echo $con_vit->getName();
// Kết quả: Con Vịt

// Phần 2: Con Heo
$con_heo = new Animal();
echo $con_heo->getName();
// Kết quả: Con Vịt
```

SAO CHÉP ĐỐI TƯỢNG

- Sao chép là tạo một bản sao của đối tượng và gán vào biến mới
- **Cú pháp : $newobject= clone** $object

CÁC HÀM KIỂM TRA MỘT ĐỐI TƯỢNG

- **Class_exists** ($class) : trả về true nếu class đã được định nghĩa
- **Get_class** ($object) : trả về tên lớp của object
- **Is_a** ($object,$class) : Trả về true nếu object là một thể hiện của class
- **Property_exists** ($object,$property_: trả về true nếu object có property
- **Method_exists** ($object,$method) : trả về true nếu object có method

BÀI TẬP