web разработка

PHP ООП
Част 3

Съдържание

- Static, late static binding
- Is a, Has a
- interfaces

Static

Static

Статичните свойства или методи могат да се достъпват без да има обект на класа
Използват се за неща валидни за всеки обект или общи за класа.

Достъпват се чрез **::**, вместо $this->...

$this е забранен в статични методи (с цел защита)

Static - 2

```
class Car extends Vehicle{
      public static $legalMaxSpeed=130;
      }
$car1 = new Car();
$car1::$legalMaxSpeed
```

Или

```
echo Car::$legalMaxSpeed.
```

Late Static Binding

Late Static Binding

```
class Model
{
protected static $name = ‘Model’;
    public static function find()
    {
        echo static::$name;
    }
}
```

!!STATIC **not** SELF!!!

```
class Product extends Model
{
    protected static $name = 'Product';
}

Product::find();

?>
```

Outputs: 'Product'

Is a/Has a

Is a/Has a

Has-a: използваме друг клас като свойство на текущия клас
Is-a: наследяваме клас и разширяваме неговата функционалност

Cat and Dog **IS A** Animal.
Leopard **IS A** Cat.

Animal **HAS A** Fur, Feed.

ресурси

Is a/Has a - 2

1//Да се измисли ООП дизайн за хипермаркет, който да има характеристики: име, адрес, оборот, мениджър, тел. на мениджър, заплата на мениджър, щандове, брой стоки в щанд, доставчици за щанд.
И методи: open(), close(), hire_manager($manager), rise_manager($manager), fill_stall($employee, $products), stall_ad($ad).

Is a/Has a - 3

2// Да се имплементира ООП дизайн за GSM, който да има характеристики: марка, модел, цена, батерия, издържливост на батерията, тип батерия, зареденост на батерията, дисплей, размер дисплей, тип дисплей.
И методи: buyGSM(), makeCall($number), chargeBattery($time), changeDisplayResolution($sizeX, $sizeY).

Interfaces

Interfaces

Когато искаме няколко класа да притежават еднакви методи, но не е основателно да бъдат наследници на един и същи клас използваме **интерфейси/interfaces**

Interfaces - 2

Дефиниране на interface

```
interface NameInterface {
        public function getName;
        public function setName($name);
        }
```

Interfaces - 3

Декларираме, че класът ще използва interface

```
class Book implements NameInterface {
private $name;

public function getName(){
    return $this->name;
}
public function setName($bookName){
    return $this->name=$bookName;
}
}
```

Interfaces - 4

класът може да декларира използването на няколко интерфейса

interface iNameInterface {...}
interface iListen {...}
interface iWatch {...}

class DVD ***implements iNameInterface, iListen, iWatch {...}***

class CD ***implements iNameInterface, iListen {...}***

class Book ***implements iNameInterface {...}***

Interfaces - 5

Всеки интерфейс определя какви методи ***задължително*** трябва да имплементира даден клас.

Ако не имплементираме всички методи в заявените интерфейси или пропуснем, необходимите за методите параметри, PHP ще върне ***fatal error***.