河北师大软件学院 *@Software College*

前端开发与HTML5
程序设计基础

王岩

2.9 面向对象

面向对象基本概念

什么是面向对象

- 对象 = 具体数据 + 操作数据的方法

归纳共同特征

类

面向对象的意义

- 更接近真实世界，代码结构更合理
- 高重用性，大大缩短开发时间
- 模块化的类结构，便于代码维护
- 更利于团队合作

PHP中的面向对象

- PHP语法：并不是纯面向对象
- 实际应用：以对象为基础构建程序架构

面向对象的基本概念

- 类的概念
  - 属性：静态特征
  - 方法：动态行为
- 声明一个类（定义一个类）

- 对象的概念（类的实例）
  - 类是抽象的，看不到摸不着的一个概念
  - 对象是具体的，看得到摸得着的一个实体
- 创建对象（类的实例化）

PHP面向对象基本写法

定义类

```
class className
{
    [访问限制符] $propertyName [ = value ] ;
    [访问限制符] function methodName([args])
    {
        //方法体
    }
}
```

访问限制符：

private：只有本类内部可以访问

protected：本类或其子类可以访问

public：类内和类外均可以访问

Demo2-9-1

- 创建一个类Person，代表现实世界中的“人”
- Person类具有的属性：姓名、性别、年龄、身高
- Person类具有的方法：吃饭、睡觉、学习、会武术
- 年龄是保密的使用private 、会武术也是保密的使用private

实例化对象

- 使用new关键字创建类的对象
- 使用符号 -> 来访问对象的属性和方法
- Demo2-9-2

对象的赋值传递

- 对象默认进行引用传递
- Demo2-9-3

构造方法

- 实例化对象时，自动调用类的构造方法执行
  - 用于类的初始化
- 统一构造方法
  - 方法名：__construct([args])
  - 在创建类的对象时自动调用构造方法
  - 在类的方法中，$this代表当前类的对象
- Demo2-9-4

析构方法

- 统一析构方法
- 方法名：__destruct()
- 当程序运行结束，需要使用unset( )销毁对象时，会自动调用析构方法

面向对象特征–封装

- 面向对象程序原则
  - 所有属性不对外暴漏（private或protected）
  - 通过属性存取器操作属性
- Demo2-9-5

类成员访问修饰符

- 访问限制修饰符
  - public：可以在类的外部访问
  - private：只能在本类中访问
  - protected：只能在本类和其子类中访问
- const 类常量修饰符
- static 类静态属性或方法修饰符
  - 静态属性：用static修饰的　类的属性
  - 静态方法：用static修饰的　类的方法

常量修饰符：const

- 类的公用常量（类的所有实例对象共享）
- 定义方法：const 常量名 = 常量值；

```
class  Calculator
{
     const PI = 3.14;
}
```

- 注意
  - 类常量通常全部使用大写字母
  - 一旦定义了一个类常量，它的值就不能改变

常量修饰符：const

- 类常量的访问
  - 类内：self :: 常量名

```
class  Calculator{
    const PI = 3.14;
    public function getPI(){
        return  self :: PI ;
    }
}
```

  - 类外：类名 :: 常量名

```
echo  Calculator :: PI ;
```

- Demo2-9-6

静态属性/方法

- 静态属性：类的公用变量（所有实例对象共享）
- 定义方法：static 访问限制符 属性名；

```
class  Person{
    private  $_name; //Person类每个对象有不同的_name
    static  public $maxAge = 130;//Person类所有对象的
                                 //maxAge是相同的
}
```

静态属性/方法

- 静态属性的访问
  - 在类内部：self :: 静态属性名

```php
class  Person{
    static  public  $maxAge = 130;
    public function __construct($age) {
        if ($age > self :: $maxAge)  return false;
    }
}
```

  - 在类外部：类名 :: 静态属性名

```php
echo  Person :: $maxAge ;
```

  - 注意：静态方法只能访问静态属性
- Demo2-9-7

$this与self的比较

	范围	表示	访问类型
$this	类的内部	当前对象	非静态、非静态属性和方法
self	类的内部	类本身	静态、常量属性和方法

继承

继承的基本写法

- 继承
  - 声明类时，指定该类继承某一个类
  - 该类可以继承父类的所有非私有属性和方法
- 继承的意义
  - 降低重复代码，程序结构更合理
- 基本语法
  - class ChildClass extends ParentClass{ }

继承的原则

- 继承原则：对于不同访问修饰符，只有public和protected类型属性或方法可以被继承

父类的访问限制符	子类中是否可以访问
public	可以
protected	可以
private	不可以

- 属性的继承：子类会继承或覆盖父类中的非私有属性
- 方法的继承：子类会继承或覆盖父类中的非私有方法

访问父类属性/方法

- 基本语法（在子类中调用父类的同名方法）
  - parent :: __construct( );

- Demo2-9-8

PHP继承特征

- 在PHP中，一个类只能继承自一个父类（单继承）
  - 优点：类结构清晰、便于代码维护

接口

- 使用接口模拟多继承
- 接口：一类具有共同性实体的约束规范
- 定义一个接口（interface）

```
interface   Runnable
{
    public function  run( );
}
```

- 接口中只能定义方法的原型和公用常量，不能定义属性
- 实现接口的类必须实现接口中定义的所有方法

接口

- 实现一个接口：使用implements关键字

```
class Person implements Runnable{
    private $_name;
    public function run(){
        echo  $this->_name . "is running";
    }
}
```

- 一个类可以继承多个接口，如

```
class Person implements Runnable, talkable { }
```

谢谢！