面向对象基础加强

属性与方法的注意点：

属性声明的时候如果不赋值，则为null

类中属性的值必须是一个 “直接的值”，不能是：表达式3-1，函数的返回值如time();

直接的值也就是8种基本数据类型

但是在java中属性的值是可以为表达式，函数返回值

java中一个文件可以有多个类class,只有一个主函数作为程序入口，有关逻辑处理部分主要是在主函数中完成，其他的class都只是各种工具类，主要负责完成某种功能

PHP中类范围之外的函数不能重复定义，在js中是可以的

但是，在PHP中定义在类中的函数和全局的普通函数不是一回事，是可以重名的。这就是所谓的方法的重载

假如类内部定义了一个time()方法，那么如何知道调用的是系统的time()还是类内部的time()呢？

主要是看是否有$this->time(),没有this说明调用的就是全局的！！！

PHP中设置浏览器解释字符集：header('Content-type:text/html;charset=utf-8')

构造方法就是让对象在创建时就赋予某些属性

public function \_\_construct($name,$sex){

$this->name=name;

$this->sex=sex;

}

每当new对象时，\_\_construct()就开始发挥作用

$zmh = new People('朱明浩',22);

$zmh->name;//朱明浩

$zmh->sex;//22

注意：new对象时的参数个数必须要与\_\_construct()中的参数个数一致，默认是不需要参数的

与java中的方法不一样，构造方法以及其他所有方法都是不能重载的，否则会提示 can't redeclare 错误

在java中，调用一个函数时，可以根据参数的个数和类型来调用同名的不同函数，因为java是支持重载的，也就是同一个函数名，可以根据参数的不同，允许存在多个函数

析构函数\_\_destruct()对象销毁时自动执行

unset($obj);销毁对象

在不启动Apache服务器的情况下，可以配置环境变量，直接在命令行中运行php程序执行php文件；可以这样：php 01.php

当一个对象的名 被赋予了其他值，就相当于这个对象被销毁了，就会执行\_\_destruct()函数；

对象的销毁：

1.可以显式的销毁unset(),赋值为null,false

2.php是脚本语言，在代码执行到最后一行时，所有申请的内存都要释放掉，自然，对象那段内存也要释放掉，对象也就被销毁了

PHP中的对象内存回收机制：

对象存放在内存中的堆中，对象名其实就是对象的引用，存放的是对象的地址，把一个对象赋给另外一个对象，实际上就是把这个对象名赋给其他变量，内存中始终只有一个对象，只是有多个引用（变量）指向这个对象罢了

内存中有一张专门记录对象引用次数的表，当对象引用次数=0时，没有引用指向该对象，PHP就会销毁该对象

经典案例：

$a = new Obj();

$d=$c=$b=$a;

unset($a);

问：\_\_destruct()函数执行的多少次，什么时候执行？

因为只有一个对象，所以只执行一次，uset()并不会导致对象销毁，因为还有其他的变量指向这个对象，这个对象不能被销毁

PHP的垃圾回收机制与java的比较相似！！！

$this的绑定：当一个对象调用其方法时，在执行该方法之前，先完成一个绑定，$this->绑定到调用此方法的对象

也就是那个对象调用方法，$this就绑定哪个对象

在类中方法中的变量如果不加$this 则理解为方法内部的一个局部变量

封装:封装与修饰符是相互相成的

public 是公共的，大家都可以访问、操作

private 表示私有的，只能通过公共的public setXXX getXXX方法调用访问,其中getXXX()必须要有返回值return，setxxx只是用来设置、改变值，并不需要返回值

private $money;

public function getMoney(){return $money;}

public function setMoney(){}

private定义的变量只能在本类中访问

封装：对于一个对象，对外界开放一个借口，调用接口时，内部进行的操作，不需要让外界知道，隐藏了内部的一些细节

权限修饰符：用来说明属性和方法的权限特点，写在属性和方法前，java中可以写在类名前

封装一个mysql工具类：

首先用private定义连接数据库所需要的各种信息

private HostName;

private User;

private Password;

private DBName;

private charset;

然后针对PHP原生的mysql方法进行深度封装，就是用一个新的function将其包裹起来，像mysql\_connect(),mysql\_query(),mysql\_close(),mysql\_select\_db()

在mysql类的\_\_construct()方法中调用$this->connect($this->HostName,$this->User,$this->Password) 来连接，调用$this->switchDB($this->DBName);来选择数据库，调用setchar($this->charset);来设置字符集

所以在创建一个mysql对象时就已经连接好数据库，我们只需要组织sql语句，通过对象调用相应封装好的方法即可

主要实现的功能有：

查询多行多列：

$arr = array();

$res=$this->query($sql);//执行查询，返回一个资源，查不到返回false

if(!$res){

return false;

}

//只有确实返回了资源才能执行取出资源到一行

while($row=mysql\_fetch\_assoc($res)){

$arr[] = $row;

}

return $arr;//返回循环之后的二维数组

查询一行，与上面主要的不同是，不需要数组来循环接收结果，直接$row = mysql\_fetch\_assoc($res);return $row;

查询共有多少行：

$sql = "select count(\*) from table\_name";

$res = $this->query($sql);

$row = mysql\_fetch\_row($res);

return $row[0];

总结：组织sql语句，执行sql语句返回结果集，取出结果集到数组或者变量，释放资源bool mysql\_free\_result ($res )，关闭连接(mysql\_close($conn))

mysql\_query()是个万能函数，可以执行所有sql语句，包括 insert select delete update

int mysql\_affected\_rows ([ resource $link\_identifier ] )

取得最近一次与 link\_identifier 关联的 INSERT，UPDATE 或 DELETE 查询所影响的记录行数。

可以用来判断是否插入成功。

if (!$link) {

die('Could not connect: ' . mysql\_error());

}

mysql\_num\_rows — 取得结果集中行的数目

int mysql\_num\_rows ( resource $result )

垮数据库查询表

强烈建议用户在 sql 查询中使用 database.table 。

mysql\_error ([ resource $link\_identifier ] )

返回上一个 MySQL 函数的错误文本，如果没有出错则返回 ''（空字符串）。如果没有指定连接资源号，则使用上一个成功打开的连接从 MySQL 服务器提取错误信息。

protected 定义的属性或者方法可以在子类中被访问，子类是不能直接访问父类的private属性的，要通过parent::xxx

private私有的属性非常特殊，能继承过来，只是无法访问，相当于一个牌坊，是不能被子类操作的

就像一个人，有些东西是私有的private，有的是公有的public，有的只能是留给儿子的protected

继承时，子类能继承父类的所有public 和 protected定义的属性和方法，并有权进行各种操作、更改

父类没有的子类可以添加

父类private属性、方法能被子类继承但是不能进行操作，可以打印出自子类对象来验证：print\_r(subObj);

可以在子类中通过parent::fun();来调用父类的方法！！！

继承深入理解：其实子类继承父类是全部继承的，只是私有的属性和方法不能被子类修改，但在内存中他们是存在子类中的，所以可以通过parent::xxx来访问父类的各种属性或方法！！！

案例如下：

<?php

class People{

private $wife = '小甜甜';

public function hshow(){

echo $this->wife;

}

}

class stu extends People{

private $wife = '凤姐';

public function sshow(){

parent::hshow();

echo $this->wife;

}

}

$zmh = new stu();

print\_r($zmh);

// stu Object 说明子类是继承了父类的所有属性和方法，包括私有的

/\*(

[wife:stu:private] => 凤姐

[wife:People:private] => 小甜甜

)\*/

// exit;

$zmh->sshow();

?>

$this-> 只能访问该类内的方法\属性

protected和public：父类有的，全部继承过来

继承过来的方法、属性，子类可以修改\重写

如果父类没有的，子类可以添加

总之：继承过来就完全属于子类的的，可以进行各种操作

继承时的权限变化：子类属性和方法的属性只能不变或者越来越宽松，但不能权限变小，比如不能修改父类的public为protected

构造方法的继承：

构造方法也是可以继承的，否则将无法创建对象，但是如果子类也有自己的构造方法，那么这个子类的构造方法就会覆盖父类的构造方法，父类的构造方法是public的，可以被重写

在子类的构造方法中调用父类的构造方法：

class stu extends Human{

public function func\_name(){

parent::\_\_construct();

echo "我是小明<br>";

}

}

在类中不能像下面这样：属性值必须是直接的值，不能使函数返回值

public $str = parent::\_\_construct();

所以在子类中要执行父类的构造函数最好用一个新的函数将其封装起来，在外面通过对象调用

\_\_construct()方法多用于mysql类中初始化数据库连接，通用的做法是：先做一个MySQL基类model，然后继承这个基类进行额外功能的添加

但如果子类覆盖了父类的构造方法，岂不是无法初始化数据库了？其实，这时候我么只需要在子类的构造方法中调用一次父类的构造方法parent::\_\_construct;，再添加新的功能模块即可！！！

构造函数的重写一定要小心，保险起见，先调用一下父类构造方法！！！

但是在java中，父类和子类的构造函数都是会执行的，且父类的构造函数先执行，因为java中是重载，是允许多个同名函数存在的！！！

java中文件名=主类名 类的构造函数名=类名

子类有如下方法：

public function sshow(){

echo parent::$wife;//不能直接访问父类的私有属性，只能通过访问父类的public方法，通过这个方法访问父类的私有属性

}

多态详解：

在强类型编程语言中，函数返回值，参数类型，return语句的值，都是强制的。

面向对象：理解比记忆语法更重要

JAVA和c++中编译时是需要检测类型 归根结底都是表玻璃还是

java因为是强类型，所以要实现多态比较麻烦，也就是函数声明参数时，声明的是父类，而实际传参时，传的是子类，这样就实现了多态

java中多态其实是基于继承，或者实现抽象接口

但是在PHP中，PHP本身就是弱类型语言，所以声明函数是无需声明参数类型，传参时，可以随意传各种对象、变量，为了让他不那么灵活，可以用接口来做，接口就相当于一个规范，实现接口的类必须实现接口的所有抽象方法才行！！！

PHP也支持设置函数参数时声明参数类型

也可以说PHP本身就是多态，简直就是变态

子类是父类

里氏代换：可以用父类的地方都可以用子类

静态属性 静态方法

语法：在属性和方法前加 static修饰

为什么要用静态方法？有什么好处？该在什么时候使用静态static？

static定义的属性是类级别的，供所有对象共同使用

在内存中类是只有一份的，而static定义的变量是类级别的，所以static修饰的属性在内存中也只有一份，类声明完毕，static修饰的属性就已经存在，不依赖于具体的对象

普通属性是每个对象各自的属性，没个对象都有不一样的属性值

可以通过类名来调用static修饰的属性 class\_name::$var;

因为静态属性在内存中只有一个，所以 一改全改

也就是一个对象改变了static值，其他对象访问static修饰的属性时，得到的是改变后的值

设置错误级别提示：Error\_reporting(E\_ALL | E\_STRICT);

在PHP中，类名调用非静态方法是会出错的！！！

此所谓：静不能访问动，动可以访问静

动和静分别对应对象和类！！！

静->静 可以

静->动 不可以

对象->访问动态方法 可以

对象->访问静态方法 可以

总结：也就是静只能访问静，动可以访问所有

普通属性、方法就是动态属性、方法！！！

静态方法不绑定$this,还是原来的对象绑定到$this

self和parent

self：本类，自身

parent::父类的

self用法:self::$staticProperty;

self::staticMothod;

如果在类内部引用类内部的静态方法或者属性，最好用self ::xxx来代替 类名::xxx

如果只在类外部，就只能通过 类名::xxx 方法来调用

parent::xxx是子类中调用父类的属性和方法

final：在PHP中，final可以修饰类、方法，但不能修饰属性

final表示最终版本：

也就是final修饰类，此类不能够被继承；

final修饰方法，此方法不能被改写\重写

final方法可以继承

单例模式：这个类只能new一次对象，

实现方法：

两个对象的比较，实际上比较的是这个变量名所指向的内存地址，比如$s1==$s2,s1,s2都为对象

clone出来的对象是一个全新的对象，具有与源对象不同的内存地址

a instanceof b 判断a对象是否为b类的一个实例，返回bool值

魔术方法：在某种情况下会自动调用的方法就是魔术方法，其特点是以双下划线开头

\_\_construct() \_\_destruct() \_\_set() \_\_get()

\_\_clone() :对象在被克隆是被调用

\_\_get($property)访问对象未定义的属性，protected属性，private属性时自动调用，也就是访问非public属性时自动调用

流程：$man->private\_property ---》\_\_get(private\_property)

如果没有\_\_get()方法系统会直接报错，甚至fatal error ，通过\_\_get(),我们就可以自定义用户访问时的处理行为

老师总结：当我们调用一个权限不允许的属性，\_\_get()方法自动调用

其实对象就是一个属性集，在js中更明显

\_\_set()

public function \_\_set($p,$v){

echo '你想设置我的'.$p.'属性'.'<br/>';

echo "且属性值为".$v;

}

总结set：当为无权操作的属性赋值时，或不存在的属性赋值时，自动调用，并自动传两个参数：属性 属性值

$man->sister='小红'; 属性：sister 值：小红，他们以参数形式传到\_\_set()函数中

\_\_isset() \_\_unset()

\_\_isset()方法：当isset()判断对象不可见的属性时（protected，private，不存在的属性时，会引发\_\_isset()来执行

可以使用unset()方法来销毁对象的属性

\_\_unset($property):当unset()方法试图销毁一个不可见的属性时，自动调用

总结：以上四个魔术方法都是调用或者操作不可见属性，由系统自动触发的！！！

所谓的不可见，也就是 未定义，private protected属性

虽然PHP不支持类似java中的重载，但是可以实现重载的效果；有一个函数可以判断函数参数的个数

func\_get\_args()返回一个包含函数参数列表的数组

public \_\_call ( $name , $arguments )

public static \_\_callStatic ( $name , $arguments )

在对象中调用一个不可访问方法时，\_\_call() 会被调用。

用静态方式中调用一个不可访问方法时，\_\_callStatic() 会被调用。

$name 参数是要调用的方法名称。 $arguments 参数是一个枚举数组，包含着要传递给方法 $name的参数。

类常量：const NAME = 'zmh';

说明：作用域在类内，又是不可改变的，相当于不可改变的静态属性，通过类名来调用，声明是不用加任何修饰符，全显示public，类外也可以访问

class MyClass

{

const constant = 'constant value' ;

function showConstant () {

echo self :: constant . "\n" ;

}

}

可以把在类中始终保持不变的值定义为常量。在定义和使用常量的时候不需要使用 $ 符号。

传统上常量标识符总是大写的。

define ( "FOO" , "something" );

bool define ( string $name , mixed $value )

bool defined ( string $name )

检查该名称的常量是否已定义。

Note:

如果你要检查一个变量是否存在，请使用 isset() 。 defined() 函数仅对constants有效。如果你要检测一个函数是否存在，使用 function\_exists() 。

mixed constant ( string $name )

通过 name 返回常量的值。

魔术常量：

1.无法手动修改他的值

2.但是他的值又是随环境变动的，所以叫魔术

\_\_FILE\_\_：当前文件的完整路径

\_\_LINE\_\_ 文件中的当前行号。

\_\_DIR\_\_ 文件所在的目录。

\_\_FUNCTION\_\_ 函数名称 要用在函数里面

\_\_CLASS\_\_ 类的名称 要用在类里面

\_\_METHOD\_\_ 类的方法名

bool file\_exists ( string $filename )

检查文件或目录是否存在。

延期绑定：

<?php

class A {

public static function who () {

echo \_\_CLASS\_\_ ;

}

public static function test () {

static:: who (); // 后期静态绑定从这里开始

}

}

class B extends A {

public static function who () {

echo \_\_CLASS\_\_ ;

}

}

B :: test ();

?>

类自动加载：

适用于大型网站，model比较多

function \_\_autoload($leiming)：如果调用不存在的类来创建对象时，在报错之前，这个魔术方法会自动调用，并把类名自动传给\_\_autoload()函数。所以我们可以在这个魔术方法中引入我们需要加载的类 require '';拼接一个文件名

require '';语句和 require();函数的作用是一样的！！

注意：函数内可以写任何合法PHP代码，包括类和函数

类的声明是全局有效的，不管他在多深的函数内

函数一定要调用才能执行，所以定义在函数内部的类，必须先调用函数才能创建对象！！！

可以通过spl\_autoload\_register()函数将自己的函数注册为自动加载函数，tp中也是这样做的！！！

spl\_autoload\_register — 注册给定的函数作为 \_\_autoload 的实现

异常处理：

函数最多只能有一个返回值，但是在new对象时，无论内部出现什么错误，都会返回一个对象，这时就需要我们捕获异常，并往外抛出异常，在类外面，通过try{}catch(Eception $e){}来处理异常

$e是返回的异常对象，可以通过这个对象的相关方法返回异常信息，比如getMessage() 。。。

异常在类内部是可以自己创建的，$e = new Exception();

if (! $x ) {

throw new Exception ( 'Division by zero.' );

}

即使出现异常，程序还是会继续往下执行的，而@xxx or die(xxx);程序终止，很不友善