php中常用的魔术方法并说出其含义

\_sleep serialize之前被调用  
\_\_wakeup unserialize时被调用  
\_\_toString 打印一个对象时被调用  
\_\_set\_state 调用var\_export时被调用，用\_\_set\_state的返回值作为var\_export的返回值  
\_\_construct 构造函数，实例化对象时被调用  
\_\_destruct 析构函数，当对象销毁时被调用  
\_\_call 对象调用某个方法， 若存在该方法， 则直接调用， 若不存在， 则调用\_\_call 函数  
\_\_get 读取一个对象属性时，若属性存在，则直接返回，若不存在，则调用\_\_get 函数  
\_\_set 设置一个对象的属性时， 若属性存在， 则直接赋值， 若不存在， 则调用\_\_set函数  
\_\_isset 检测一个对象的属性是否存在时被调用  
\_\_unsetunset一个对象的属性时被调用  
\_\_clone 克隆对象时被调用  
\_\_autoload 实例化一个对象时，如果对应的类不存在，则该方法被调用

php5中的构造方法和析构方法的含义与作用

构造方法：\_\_construct

析构方法：\_\_destruct

类的属性可以序列化后保存到session中，从而以后可以恢复整个类用到的函数是：

serialize() 和 unserialize()

php中的子类与父类有相同名称的函数，则子类的函数会替换父类的函数，请写出代码：

<?php

class A

{

function name1()

{

echo "come on";

}

}

class B extends A

{

public $tmp;

function name2()

{

echo"come on too";

}

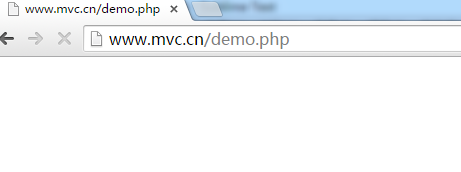
}

$obj = new B();

$a = $obj->name2();

echo $a;

?>



接口和抽象类的区别：

抽象类是一种不能被实例化的类， 只能作为其他类的父类来使用。抽象类是通过关键字  
abstract 来声明的。  
抽象类与普通类相似， 都包含成员变量和成员方法， 两者的区别在于， 抽象类中至少要  
包含一个抽象方法，  
抽象方法没有方法体，该方法天生就是要被子类重写的。  
抽象方法的格式为： abstractfunctionabstractMethod();  
接口是通过interface关键字来声明的， 接口中的成员常量和方法都是public的， 方法可  
以不写关键字public，  
接口中的方法也是没有方法体。接口中的方法也天生就是要被子类实现的。  
抽象类和接口实现的功能十分相似， 最大的不同是接口能实现多继承。在应用中选择抽  
象类还是接口要看具体实现。  
子类继承抽象类使用extends，子类实现接口使用implements

\_\_autoload函数和spl\_autoload\_register的区别

用这个魔术函数的基本条件是类文件的文件名要和类的名字保持一致。  
当程序执行到实例化某个类的时候， 如果在实例化前没有引入这个类文件， 那么就自动  
执行\_\_autoload()函数。

这个函数会根据实例化的类的名称来查找这个类文件的路径，当判断这个类文件路径下  
确实存在这个类文件后就执行include或者require来载入该类， 然后程序继续执行， 如果这  
个路径下不存在该文件时就提示错误

而spl\_autoload\_register()将普通的函数（方法）注册为自动加载的

能够写出Db.class.php:

<?php

final class Db

{

private static $obj = NULL;

private $db\_host;//主机名

private $db\_user;//用户名

private $db\_pass;//密码

private $db\_name;//数据库

private $charset;//字符集

//私有的构造方法：阻止类外new对象

private function \_\_construct($config)

{

$this->db\_host = $config['db\_host'];

$this->db\_user = $config['db\_user'];

$this->db\_pass = $config['db\_pass'];

$this->db\_name = $config['db\_name'];

$this->charset = $config['charset'];

$this->connMySQL();//连接数据库

$this->selectDb();//选择数据库

$this->setCharacter();//设置字符集

}

private function \_\_clone(){}

public static function getInstance($config){

if(!self::$obj instanceof self){

self::$obj = new self($config);

}

return self::$obj;//返回对象

}

private function connMySQL(){

$link = @mysql\_connect($this->db\_host,$this->db\_user,$this->db\_pass);

if(!$link) exit("PHP连接MySQL服务器失败！");

}

private function selectDb(){

if(!mysql\_select\_db($this->db\_name))

exit("选择数据库{$this->db\_name}失败！");

}

private function setCharacter(){

$this->exec("SET NAMES {$this->charset}");

}

public function exec($sql)

{

// "SELECT \* FROM student"

$sql = strtolower($sql);

if(substr($sql,0,6)=="select"){

exit("SELECT语句，只能调用query()方法！");

}

return mysql\_query($sql);

}

private function query($sql)

{

$sql = strtolower($sql);

if(substr($sql,0,6)=="select"){

return mysql\_query($sql);

}else

{

exit("非SELECT语句，请用exec()方法!");

}

}

public function fetchAll($sql,$type=3)

{

$types = array(

1 => MYSQL\_NUM,

2 => MYSQL\_BOTH,

3 => MYSQL\_ASSOC

);

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

while($row = mysql\_fetch\_array($result, $types[$type]))

{

$arr[] = $row;

}

return $arr;

}

public function \_\_destruct()

{

mysql\_close();

}

}

?>