面向对象第3天

[学习目标 1](#_Toc522077779)

[最终类和最终方法 2](#_Toc522077780)

[1、概述 2](#_Toc522077781)

[2、实例：最终类和最终方法演示 2](#_Toc522077782)

[抽象类和抽象方法 2](#_Toc522077783)

[1、概述 2](#_Toc522077784)

[2、实例：抽象类和抽象方法实例演示 2](#_Toc522077785)

[接口技术 3](#_Toc522077786)

[1、接口的基本概念？ 3](#_Toc522077787)

[2、接口定义和实现要点 3](#_Toc522077788)

[3、实例：接口的定义和实现演示 3](#_Toc522077789)

[4、实例：创建手机类并实现小灵通接口、MP3接口、MP4接口 3](#_Toc522077790)

[类的自动加载 3](#_Toc522077791)

[1、为什么需要类的自动加载 3](#_Toc522077792)

[2、类文件的命名规范 4](#_Toc522077793)

[3、类的自定义加载函数：spl\_autoload\_register() 4](#_Toc522077794)

[（1）spl\_autoload\_register()何时调用？ 4](#_Toc522077795)

[（2）语法格式 4](#_Toc522077796)

[（3）使用普通函数作为参数 4](#_Toc522077797)

[（4）使用匿名函数作为参数 5](#_Toc522077798)

[对象克隆 5](#_Toc522077799)

[1、什么是对象克隆？ 5](#_Toc522077800)

[2、实例：对象克隆的演示 5](#_Toc522077801)

[3、实例：魔术方法\_\_clone()在克隆对象中的使用 5](#_Toc522077802)

[对象遍历 5](#_Toc522077803)

[常用魔术方法 5](#_Toc522077804)

[1、\_\_toString() 5](#_Toc522077805)

[2、\_\_invoke() 6](#_Toc522077806)

[面向对象的设计模式 6](#_Toc522077807)

[1、什么是对象设计模式？ 6](#_Toc522077808)

[2、常用的设计模式有哪些？ 6](#_Toc522077809)

[3、单例设计模式的要求(三私一公) 6](#_Toc522077810)

[4、实例：单例设计模式演示 6](#_Toc522077811)

# 学习目标

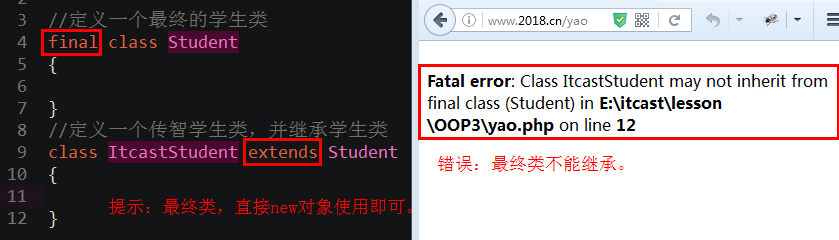
* 最终类和最终方法
* 抽象类和抽象方法
* 接口描述
* 类的自动加载
* 对象克隆和对象遍历
* 单例设计模式

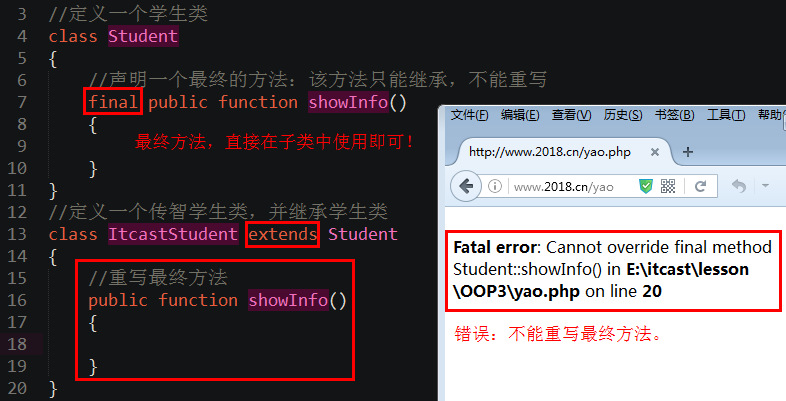
# 最终类和最终方法

## 1、概述

* Final关键字修饰的类，就是最终类；
* Final关键字修饰的方法，就是最终方法；
* 最终类：该类只能实例化，不能被继承。该类十分完善了，不需要升级和扩展。
* 最终方法：该方法可以被继承，但不能重写。该方法十分完善了，不需要重写了。

## 2、实例：最终类和最终方法演示





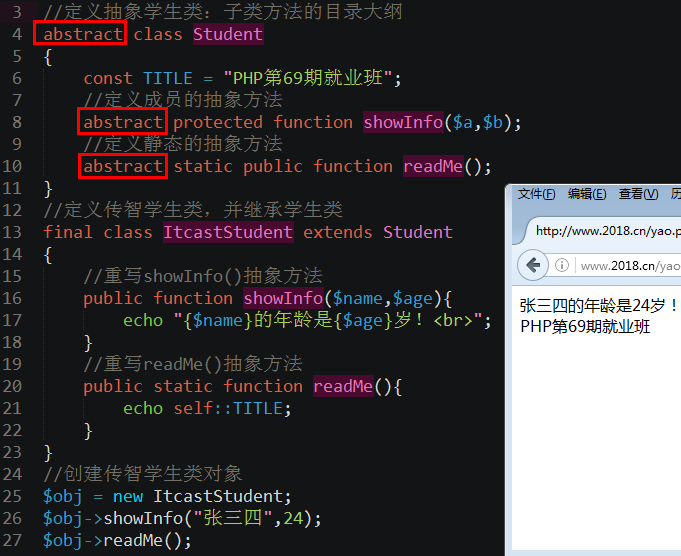
提示：类是我们自己设计的，我们怎么设计，他们就怎么用。

# 抽象类和抽象方法

## 1、概述

* abstract关键字修饰的类，就是抽象类；
* abstract关键字修饰的方法，就是抽象方法；
* 抽象类：该类不能直接实例化，必须先继承后再实例化。常用在基础类。
* 抽象方法：方法必须先继承后重写。
* 抽象方法就是方法的命名规范、命名规则、方法大纲，也可以理解为一种监督机制。
* 所有的抽象方法都必须重写，少一个都不行。
* 抽象方法没有方法体，必须在子类重写后，再定义方法体。
* 如果一类中有一个抽象方法，该类必须是抽象类。
* 抽象方法权限不能是private，因为要先继承再重写。
* 在PHP7中，抽象方法可以是成员方法，也可以是静态方法。
* 抽象类中，可以包含其它成员：常量、成员属性、成员方法、静态属性、静态方法。

## 2、实例：抽象类和抽象方法实例演示



# 接口技术

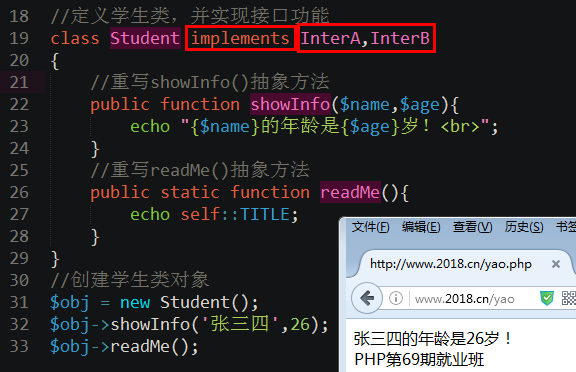
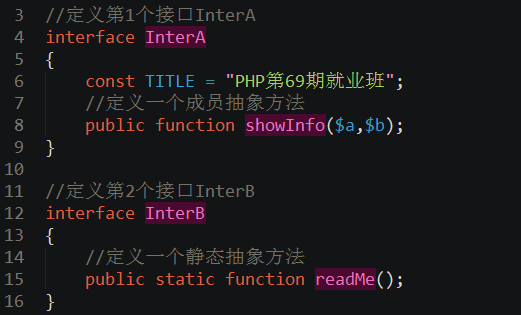
## 1、接口的基本概念？

* 接口就是特殊的抽象类。
* PHP 类是单继承，也就是不支持多继承。
* 当一个类需要多个类的功能时，单继承就无能为力了，为此 PHP 引入了类的接口技术。
* 多人合作开发项目时，需要规范各个功能的名称，就需要用到接口技术。
* 接口就是一种标准，一种规范。类的功能实现，按照标准接口实现即可；

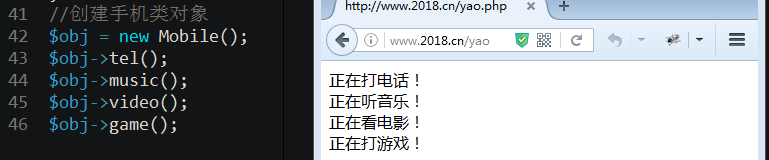
## 2、接口定义和实现要点

* interface关键字定义接口；
* Implements关键字用来实现接口；
* 接口中方法权限必须是public；
* 接口中方法默认是抽象的，所以不需要在方法名前面加 abstract ；
* 接口中方法可以是成员方法，也可以是静态方法；
* 接口中也可以定义常量，但常量不能重写；
* 类可以实现(implements)多个接口（相当于把多个功能集于一身，如手机实现了小灵通、MP3、MP4的功能）；
* 接口也可以继承(extends)接口，类可以继承(extends)类，但是类只能实现(implements)接口。

## 3、实例：接口的定义和实现演示



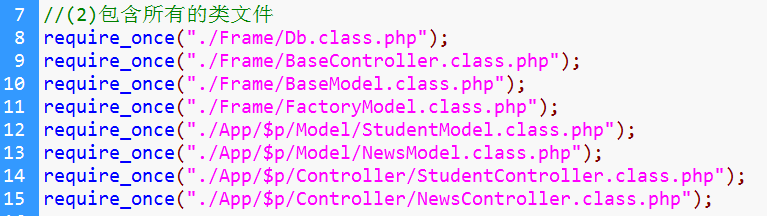
## 4、实例：创建手机类并实现小灵通接口、MP3接口、MP4接口



# 类的自动加载

## 1、为什么需要类的自动加载

很多开发者写面向对象的应用程序时，对每个类的定义，都建立一个独立的 PHP 类文件，方便类文件的统一管理，这无可厚非。但一个很大的烦恼是，不得不在每个脚本开头，写一个长长的包含文件列表（每个类一个文件）。这样一来，就增加了很多负担、占用了很多的内存，对于后期维护也不方便。



**解决方案：按需要加载类文件，而不是把所有类全部包含进来**。

## 2、类文件的命名规范

* 一个类要单独定义成一个独立的类文件；
* 类文件扩展名，要以".class.php"结尾，是一种约定，不是必须的；
* 类文件主名，要与类名一致；
* 例如：Db.class.php、UserController.class.php、UserModel.class.php

## 3、类的自定义加载函数：spl\_autoload\_register()

PHP7以下版本，使用\_\_autoload()实现类的自动加载；

PHP7以上版本，使用spl\_autoload\_register()实现类的自动加载。

### （1）spl\_autoload\_register()何时调用？

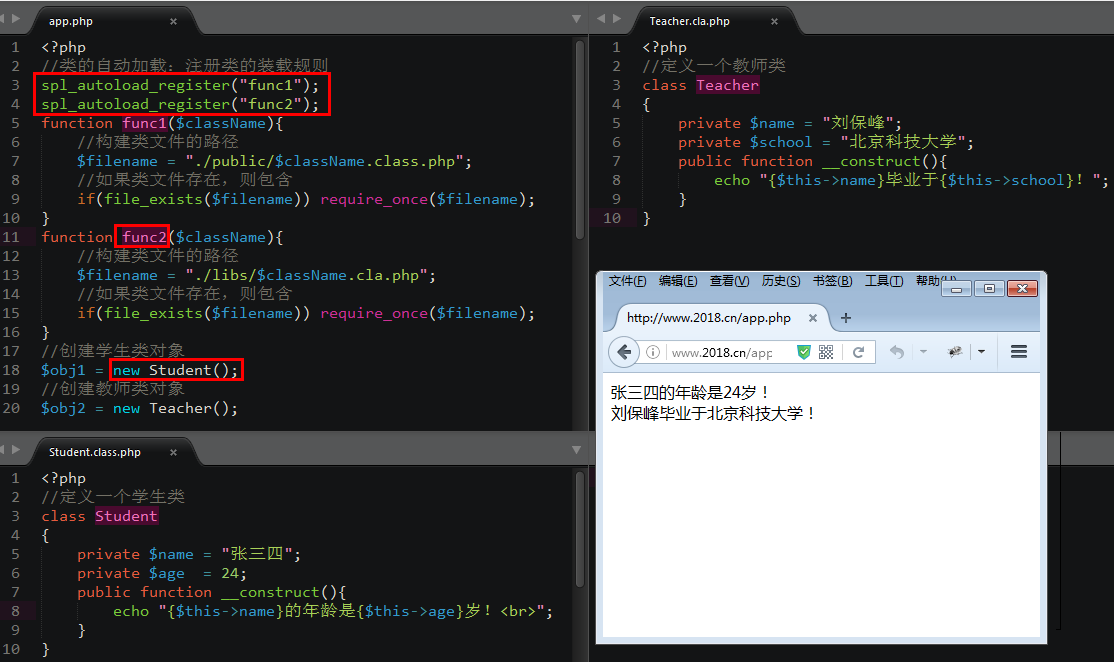
当试图使用未定义的类时spl\_autoload\_register自动调用，**使用一个类有以下几种情况：**

* 使用new关键字创建不存在类的对象时，spl\_autoload\_register自动调用；例如：$obj = new Student()
* 当使用静态化方式访问一个不存在的类时，spl\_autoload\_register自动调用，例如：Student::show()；
* 当继承一个不存在的类时，spl\_autoload\_register自动调用，例如：class Stu extends Parent{}
* 当实现一个不存在的接口时，spl\_autoload\_register自动调用，例如：class Stu implements Inter

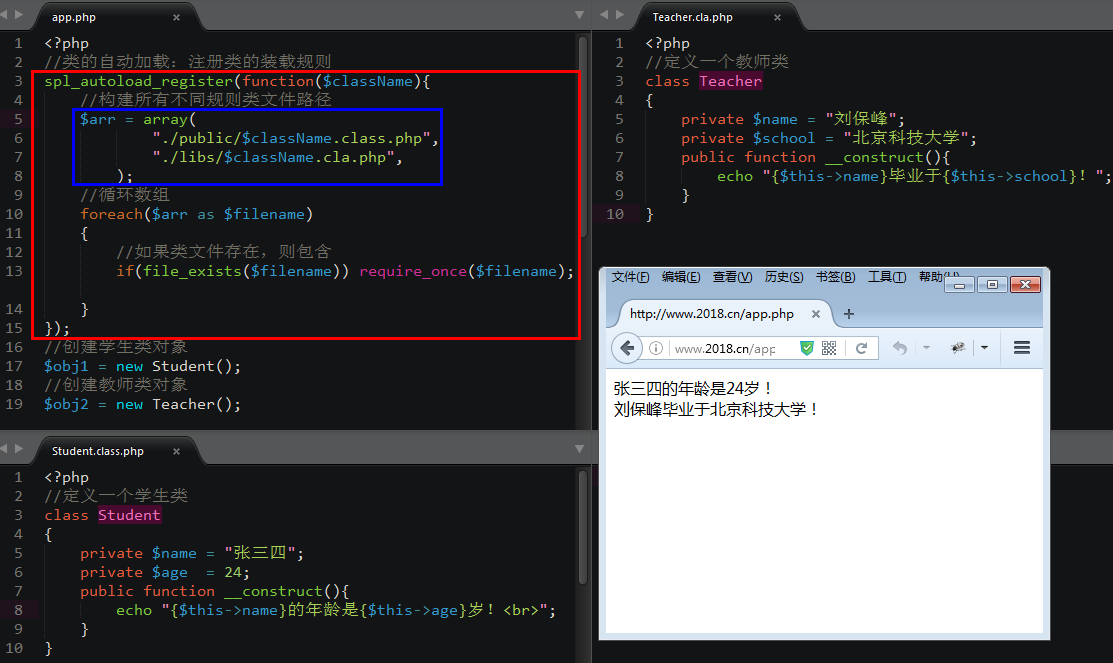
### （2）语法格式

* 描述：将函数注册到SPL(标准PHP库)的\_\_autoload函数队列中。如果该队列中的函数尚未激活，则激活它们。它实际上创建了 autoload 函数的队列，按定义时的顺序逐个执行。
* 语法：**bool spl\_autoload\_register ([ callback $autoload\_function ] )**
* 参数：$autoload\_function，欲注册的自动装载函数，可以是匿名函数，也可以是字符串的函数名称。$autoload\_function有一个传递过来的类名形参，用于在函数中构建类文件路径。
* 返回：成功时返回 TRUE， 或者在失败时返回 FALSE。

### （3）使用普通函数作为参数



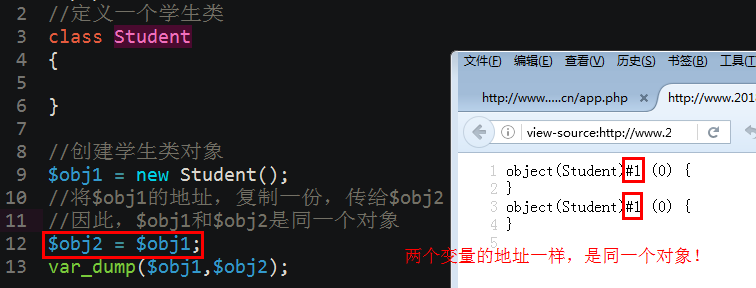
### （4）使用匿名函数作为参数



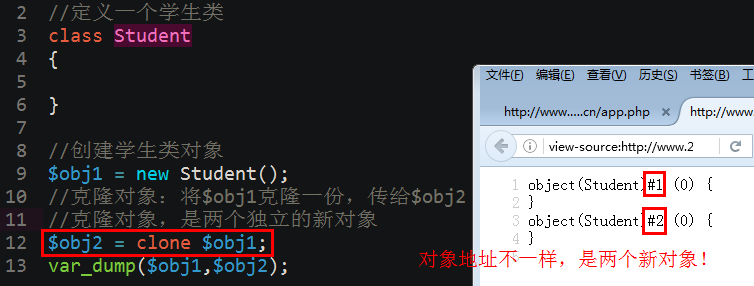
# 对象克隆

## 1、什么是对象克隆？

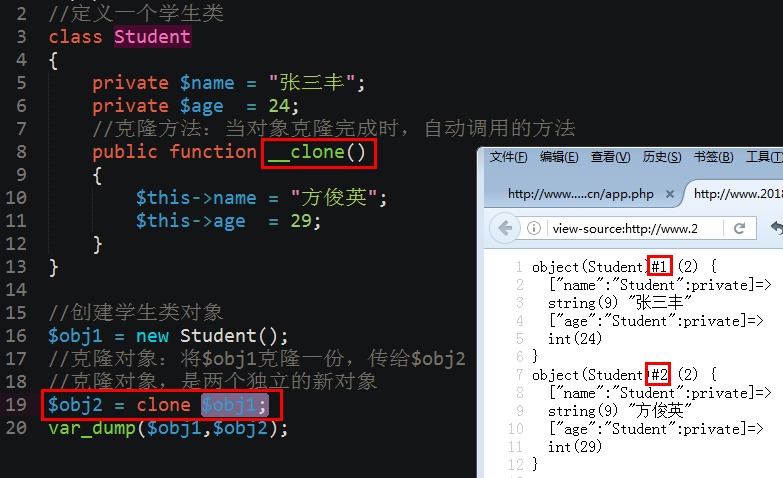
* 如果已存在了一个对象，而还想再创建一个新对象，并且，两个对象的属性值不一样，或者属性比原来多 ，怎么实现呢？ $obj2 = $obj1 无法实现！
* $obj2 = $obj1，这不是复制对象，而是将$obj1和$obj2指向了同一个对象地址。
* 创建新对象有两种方式：a. 使用new关键字； b. 使用 clone 关键字



## 2、实例：对象克隆的演示



## 3、实例：魔术方法\_\_clone()在克隆对象中的使用



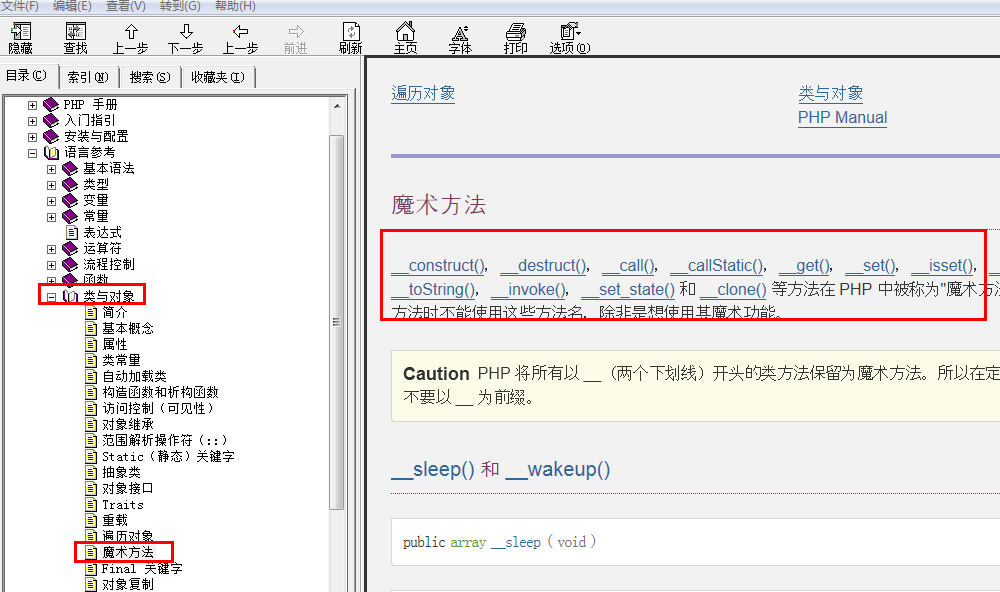
# 对象遍历

foreach既可以遍历数组元素，也可以遍历对象属性。



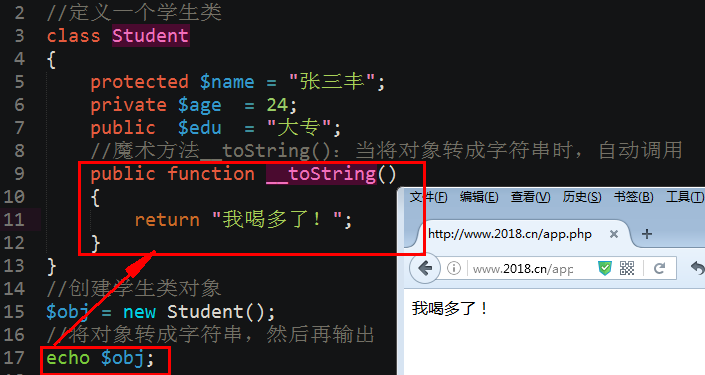


# 常用魔术方法



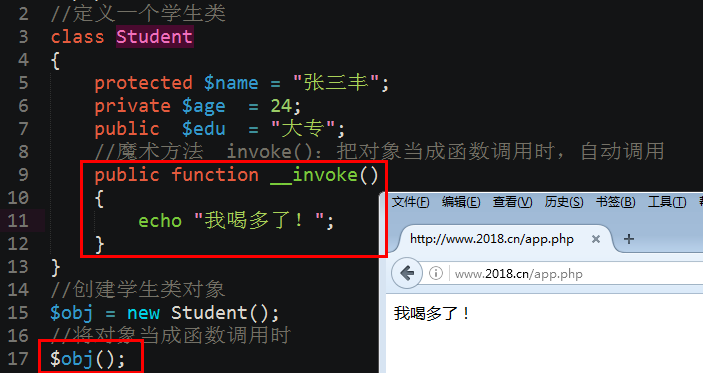
## 1、\_\_toString()

* 描述：将对象转成字符串时，\_\_toString()会自动调用。
* 语法：**public string \_\_toString ( void )**
* 注意：PHP不支持对象转字符串，因此，不能使用echo输出一个对象

。

## 2、\_\_invoke()

* 描述：当把一个对象当成函数调用时，\_\_invoke()会自动调用。
* 语法：mixed \_\_invoke ([ $... ] )



# 面向对象的设计模式

## 1、什么是对象设计模式？

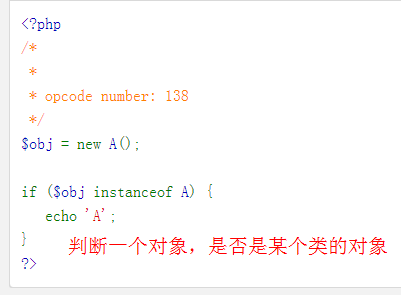
设计模式（Design pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。

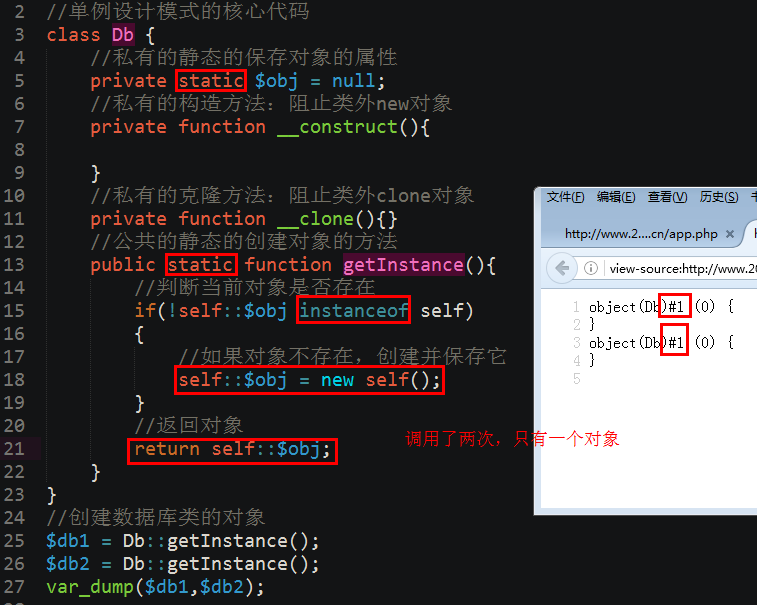
## 2、常用的设计模式有哪些？

* 单例设计模式：一个类只能创建一个实例对象，不管用什么办法都无法创建第2个对象；
* 工厂设计模式：生产不同类对象的工厂；
* 策略设计模式：定义一组算法，将每个算法都封装起来，并且使它们之间可以互换。
* 观察者设计模式：定义对象间一种一对多的依赖关系，使得每当一个对象改变状态，则所有依赖于它的对象都会得到通知并被自动更新。

## 3、单例设计模式的要求(三私一公)

* 一私：私有的静态的保存对象的属性。
* 一私：私有的构造方法，阻止类外new对象。
* 一私：私有的克隆方法，阻止类外clone对象。
* 一公：公共的静态的创建对象的方法。





## 4、实例：单例设计模式演示