[**http://www.runoob.com/design-pattern/singleton-pattern.html**](http://www.runoob.com/design-pattern/singleton-pattern.html)

**单例模式(Singleton Pattern)**：顾名思义，就是只有一个实例。作为对象的创建模式，单例模式确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例。

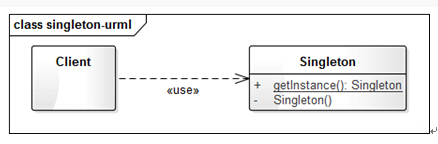
**（一）为什么要使用PHP单例模式**

1，php的应用主要在于数据库应用, 一个应用中会存在大量的数据库操作, 在使用面向对象的方式开发时, 如果使用单例模式,  
则可以避免大量的new 操作消耗的资源,还可以减少数据库连接这样就不容易出现 too many connections情况。

2，如果系统中需要有一个类来全局控制某些配置信息, 那么使用单例模式可以很方便的实现. 这个可以参看zend Framework的FrontController部分。

3，在一次页面请求中, 便于进行调试, 因为所有的代码(例如数据库操作类db)都集中在一个类中, 我们可以在类中设置钩子, 输出日志，从而避免到处var\_dump, echo

**（二）单例模式UML图**



                       Singleton Pattern

**（三）单例模式的实现**

1，私有化一个属性用于存放唯一的一个实例

2，私有化构造方法，私有化克隆方法，用来创建并只允许创建一个实例

3，公有化静态方法，用于向系统提供这个实例

**（四）代码实现**

[复制代码](javascript:void(0);)

class Singleton{

//存放实例

private static $\_instance = null;

//私有化构造方法、

private function \_\_construct(){

echo "单例模式的实例被构造了";

}

//私有化克隆方法

private function \_\_clone(){

}

//公有化获取实例方法

public static function getInstance(){

if (!(self::$\_instance instanceof Singleton)){

self::$\_instance = new Singleton();

}

return self::$\_instance;

}

}

$singleton=Singleton::getInstance();

[复制代码](javascript:void(0);)

优点：因为静态方法可以在全局范围内被访问，当我们需要一个单例模式的对象时，只需调用getInstance方法，获取先前实例化的对象，无需重新实例化。

**(五)使用Trait关键字实现类似于继承单例类的功能**

[复制代码](javascript:void(0);)

Trait Singleton{

//存放实例

private static $\_instance = null;

//私有化克隆方法

private function \_\_clone(){

}

//公有化获取实例方法

public static function getInstance(){

$class = \_\_CLASS\_\_;

if (!(self::$\_instance instanceof $class)){

self::$\_instance = new $class();

}

return self::$\_instance;

}

}

class DB {

private function \_\_construct(){

echo \_\_CLASS\_\_.PHP\_EOL;

}

}

class DBhandle extends DB {

use Singleton;

private function \_\_construct(){

echo "单例模式的实例被构造了";

}

}

$handle=DBhandle::getInstance();

//注意若父类方法为public，则子类只能为pubic，若父类为private，子类为public ，protected，private都可以。

[复制代码](javascript:void(0);)

补充，大多数书籍介绍单例模式，都会讲三私一公，公优化静态方法作为提供对象的接口，私有属性用于存放唯一一个单例对象。私有化构造方法，私有化克隆方法保证只存在一个单例。