# PHP设计模式-单例模式

## 概念

* 单例模式，是一种常用的软件设计模式。
* 在它的核心结构中只包含一个被称为单例的特殊类。
* 通过单例模式可以保证系统中一个类只有一个实例。即一个类只有一个对象实例。

### 要点主要有三个：

* 一个类只能有一个对象；
* 必须是自行创建这个类的对象；
* 要向整个系统提供这一个对象；

## 从具体实现角度来说，就是以下三点：

* 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

 PHP代码：

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: guoju  
 \* Date: 2018/12/7  
 \* Time: 22:34  
 \*/***class** Single{  
 *//存放实例* **private static** *$\_instance* = **null**;  
  
 *//私有化构造方法* **private function** \_\_construct()  
 {  
  
 }  
  
 *//私有化克隆方法* **private function** \_\_clone()  
 {  
 *//* ***TODO: Implement \_\_clone() method.*** }  
  
 *//公有化获取实例方法* **public static function** getInstance(){  
 **if** (!(**self**::*$\_instance* **instanceof** Single)){  
 **self**::*$\_instance* = **new** Single();  
 }  
 **return self**::*$\_instance*;  
 }  
  
}

//获取并实例化

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: guoju  
 \* Date: 2018/12/7  
 \* Time: 22:52  
 \*/***include 'single.php'**;  
$objSingle = Single::*getInstance*();

## 为什么要使用PHP单例模式

* + 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。
  + 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。
  + 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

数据库层的应用-代码同上

* 优点

实例控制

* 单例模式会阻止其他对象实例化其自己的单例对象的副本，从而确保所有对象都访问唯一实例。只需调用getInstance方法，获取先前实例化的对象，无需重新实例化。

灵活性

* 因为类控制了实例化过程，所以类可以灵活更改实例化过程。
* 缺点
* 开销
* 虽然数量很少，但如果每次对象请求引用时都要检查是否存在类的实例，将仍然需要一些开销。可以通过使用静态初始化解决此问题。
* 可能的开发混淆
* 使用单例对象（尤其在类库中定义的对象）时，开发人员必须记住自己不能使用new关键字实例化对象。因为可能无法访问库源代码，因此应用程序开发人员可能会意外发现自己无法直接实例化此类。

## 使用Trait关键字实现类似于继承单例类的功能