PHP设计模式之：单例模式

**单例模式（职责模式）：**

简单的说，一个对象（在学习设计模式之前，需要比较了解面向对象思想）只负责一个特定的任务；

**单例类：**

1、构造函数需要标记为private（访问控制：防止外部代码使用new操作符创建对象），单例类不能在其他类中实例化，只能被其自身实例化；

2、拥有一个保存类的实例的静态成员变量

3、拥有一个访问这个实例的公共的静态方法（常用getInstance()方法进行实例化单例类，通过instanceof操作符可以检测到类是否已经被实例化）

另外，需要创建\_\_clone()方法防止对象被复制（克隆）

**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, 所以一个应用中会存在大量的数据库操作, 使用单例模式, 则可以避免大量的new 操作消耗的资源。

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ZF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。

**代码实现：**

<111?111php111

/1\*\*

\* 设计模式之单例模式

\* $\_instance必须声明为静态的私有变量

\* 构造函数和析构函数必须声明为私有,防止外部程序new

\* 类从而失去单例模式的意义

\* getInstance()方法必须设置为公有的,必须调用此方法

\* 以返回实例的一个引用

\* ::操作符只能访问静态变量和静态函数

\* new对象都会消耗内存

\* 使用场景:最常用的地方是数据库连接。

\* 使用单例模式生成一个对象后，

\* 该对象可以被其它众多对象所使用。

\*/

**class**Danli {

//保存类实例的静态成员变量

**private static**$\_instance;

//private标记的构造方法

**private function**\_\_construct(){

**echo**'This is a Constructed method;';

}

//创建\_\_clone方法防止对象被复制克隆

**public function**\_\_clone(){

trigger\_error('Clone is not allow!',E\_USER\_ERROR);

}

//单例方法,用于访问实例的公共的静态方法

**public static function***getInstance*(){

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

**self**::*$\_instance* = **new self**;

}

**return self**::*$\_instance*;

}

**public function**test(){

**echo**'调用方法成功';

}

}

//用new实例化private标记构造函数的类会报错

//$danli = new Danli();

//正确方法,用双冒号::操作符访问静态方法获取实例

$danli = Danli::*getInstance*();

$danli->test();

//复制(克隆)对象将导致一个E\_USER\_ERROR

$danli\_clone = **clone**$danli;