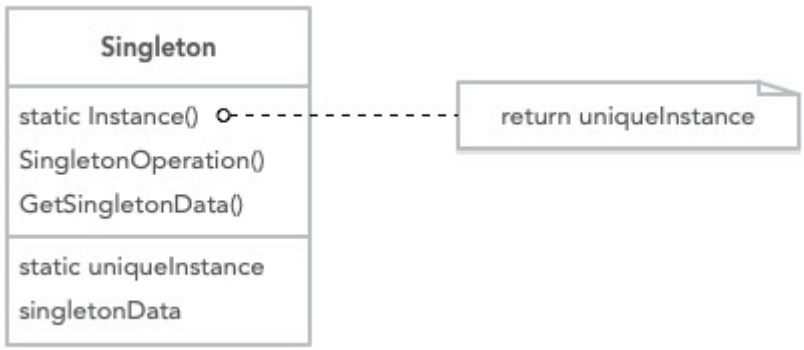
PHP设计模式之单例模式

单例模式绝对是在常用以及面试常问设计模式中排名首位的。一方面它够简单，三言两语就能说明白。另一方面，它又够复杂，它的实现不仅仅只有一种形式，而且在Java等异步语言中还要考虑多线程加锁的问题。所以在面试时，千万不要以为面试官出单例模式的问题就放松了，这个模式真的是可深可浅，也极其能体现一个开发者的水平。因为只要工作过一段时间，不可避免的就会接触到这个模式。

Gof类图及解释

GoF定义：保证一个类仅有一个实例，并提供一个访问它的全局访问点。

> GoF类图



> 代码实现

```php
class Singleton
{
    private static $uniqueInstance;
    private $singletonData = '单例类内部数据';

    private function __construct()
    {
        // 构造方法私有化，外部不能直接实例化这个类
    }

    public static function GetInstance()
    {
        if (self::$uniqueInstance == null) {
            self::$uniqueInstance = new Singleton();
        }
        return self::$uniqueInstance;
```

```
    }

    public function SingletonOperation(){
        $this->singletonData = '修改单例类内部数据';
    }

    public function GetSigletonData()
    {
        return $this->singletonData;
    }

}
```

没错，核心就是这样一个单例类，没别的了。让静态变量保存实例化后的自己。当需要这个对象的时候，调用GetInstance()方法获得全局唯一的一个对象。

```
$singletonA = Singleton::GetInstance();
echo $singletonA->GetSigletonData(), PHP_EOL;

$singletonB = Singleton::GetInstance();

if ($singletonA === $singletonB) {
    echo '相同的对象', PHP_EOL;
}
$singletonA->SingletonOperation(); // 这里修改的是A
echo $singletonB->GetSigletonData(), PHP_EOL;
```

客户端的调用，我们会发现$singletonA和$singletonB是完全一样的对象。

- 没错，从代码中就可以看出，单例最大的用途就是让我们的对象全局唯一。
- 那么全局唯一有什么好处呢？有些类的创建开销很大，而且并不需要我们每次都使用新的对象，完全可以一个对象进行复用，它们并没有需要变化的属性或状态，只是提供一些公共服务。比如数据库操作类、网络请求类、日志操作类、配置管理服务等等
- 曾经有过面试官问过，单例在PHP中到底是不是唯一的？如果在一个进程下，也就是一个fpm下，当然是唯一的。nginx同步拉起的多个fpm中那肯定就不是唯一的啦。一个进程一个嘛！
- 单例模式的优点：对唯一实例的受控访问；缩小命名空间；允许对操作和表示的精化；允许可变数目的实例；比类操作更灵活。
- Laravel中在IoC容器部分使用了单例模式。关于容器部分的内容我们会在将来的Laravel系列文章中讲解。我们可

以在Illuminate\Container\Container类中找到singleton方法。它调用了bind方法中的getClosure方法。继续追踪会发现他们最终会调用Container的make或build方法来进行实例化类，不管是make还是build方法，他们都会有单例的判断，也就是判断类是否被实例化过或者在容器中已存在。build中的if (!$reflector->isInstantiable())。

公司越来越大，但我们的全部公司的花名册都只有一份（单例类），保存在我们的OA系统中。怕的就是各个部门拥有各自己的花名册后会产生混乱，比如更新不及时漏掉了其他部门新入职或者离职的员工。其他部门在需要的时候，可以去查看全部的花名册，也可以在全部花名册的基础上建立修改自己部门的部分。但是在OA系统中，其实他们修改的还是那一份总的花名册中的内容，大家维护的其实都是保存在OA系统服务器中的那唯一一份真实的花名册

完整代码：https://github.com/zhangyue0503/designpatterns-php/blob/master/21.singleton/source/singleton.php

实例

既然上面说过数据库操作类和网络请求类都很喜欢用单例模式，那么我们就来实现一个Http请求类的单例模式的开发。记得在很早前做Android的时候，还没有现在这么多的框架，Http请求都是自己封装的，网上的教程中大部分也都是采取的单例模式。

> 缓存类图

完整源码：https://github.com/zhangyue0503/designpatterns-php/blob/master/21.singleton/source/singleton-http.php

```php
<?php

class HttpService{
    private static $instance;

    public function GetInstance(){
        if(self::$instance == NULL){
            self::$instance = new HttpService();
        }
        return self::$instance;
```

```php
    }

    public function Post(){
        echo '发送Post请求', PHP_EOL;
    }

    public function Get(){
        echo '发送Get请求', PHP_EOL;
    }
}

$httpA = HttpService::GetInstance();
$httpA->Post();
$httpA->Get();

$httpB = HttpService::GetInstance();
$httpB->Post();
$httpB->Get();

var_dump($httpA === $httpB);
// 发送Post请求
// 发送Get请求
// 发送Post请求
// 发送Get请求
// bool(true)

$httpA = new HttpService();
$httpA->Post();
$httpA->Get();

$httpB = new HttpService();
$httpB->Post();
$httpB->Get();

var_dump($httpA === $httpB);
// 发送Post请求
// 发送Get请求
// 发送Post请求
// 发送Get请求
// bool(false)
```

```
51
```

> 说明

- 是不是依然很简单，这里就不多说这种形式的单例了，我们说说另外几种形式的单例
- 在Java等静态语言中，静态变量可以直接new对象，在声明$instance的时候直接给他赋值，比如 private static $instance = new HttpService();。这样可以省略掉GetInstance()方法，但是这个静态变量不管用不用都会直接实例化出来占用内存。这种单例就叫做饿汉式单例模式。
- 我们的代码和例子很明显不是饿汉式的，这种形式叫做懒汉式。你要主动的来用GetInstance()获取，我才会创建对象。
- 懒汉式在多线程的应用中，如java多线程或者PHP中使用swoole之后，会出现重复创建的问题，而且这多次创建的都不是同一个对象了。这时一般会使用双重检测来来确保全局还是只有唯一的一个对象。具体代码大家可以去自己找一下。饿汉式不会有问题，饿汉式本身就已经给静态属性赋值了，不会再改变。具体可以参考静态类相关文章(公众号内查询《PHP中的static》或掘金https://juejin.im/post/5cb5b2926fb9a068664c1ac5)。
- 还有一种方式是静态内部类的创建方式。这种平常就不多见了，它的资源利用率高。将静态变量放在方法内，使静态变量成为方法内的变量而不是类中的变量。它可以让单例对象调用自身的静态方法和属性。