PHP设计模式之适配器模式

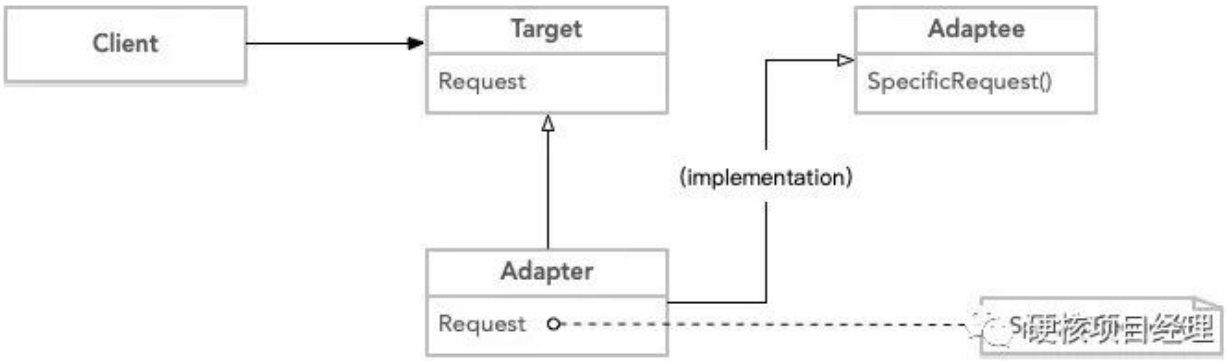
这个模式一直以来都有一个很经典的例子，那就是插座！没错，当我们从国外买回来电器，或者旅游出差去国外的时候，经常会需要一个电源适配器，因为我国的电压标准是220伏，而其他国家则有110伏的标准。而这个电源适配器正是适配器模式的一种标志。当对象不太符合要求的时候，给他加一个适配器呗！！

Gof类图及解释

GoF定义：将一个类的接口转换成客户希望的另外一个接口。Adapter模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作

> GoF类图

继承式



组合式

> 代码实现

```php
interface Target{
    function Request() : void;
}
```

定义一个接口契约，也可以是一个正常的有实现方法的类（后面的例子我们会用类）

```php
class Adapter implements Target{
    private $adaptee;

    function __constuct($adaptee){
        $this->adaptee = $adaptee;
```

```
    }

    function Request() : void {
        $this->adaptee->SpecificRequest();
    }
}
```

适配器实现这个接口契约，让Request()方法得以实现，但请注意，我们真正调用的其实是Adaptee类中的方法

```
class Adaptee {
    function SpecificRequest() : void{
        echo "I'm China Standard! ";
    }
}
```

- 适配器有两种形式，上方类图中给出了，我们代码实现的组合形式的
- 继承形式的在GoF书中是以C++为示例的，因为C++可以实现多重继承，但现在流行的大部分语言是以接口为形式的，也可以实现，但使用这种形式的适配器不多
- 其实还是面向接口编程的一种思维，类似于装饰器对旧功能的包装，我们这里就是直接去进行了替换，但对外的调用还是保持不变
- 适配器模式其实很好理解，代码真的就只有这么点

又说到我的手机工厂了，这回咱们的生意真的做大了哦！卖到泰国、新加坡、印度尼西亚去了，反正有咖喱的地方都有我们的身影了。据说是我们出了个咖喱色。换壳这事儿可不完全是因为受到诺X亚的影响，而是真的经过长期的调研我们发现不同颜色在不同的地方销量会更好。于是，富X康在原有的手机壳生产线（Target）上为我们加装了一个喷涂适配器（adapter），当我们需要其他颜色的壳时，只需要这个适配器换不同的颜料就好啦（adaptee），直接装上这个喷涂器，新的颜色的手机就诞生了。而当向另外一个国家扩展业务时，我们换颜料就行啦，用太久了不行就连喷头也换掉（是不是想起了连供打印机）

完整代码：适配器模式

实例

继续发短信，看我能编到什么时候~~~

各位大拿在对接信息、支付类的接口时，经常会使用这些平台提供的SDK。特别是有了Composer之后，安装SDK就更加的方便了，但是，又有一个严重的问题，这帮人做的SDK虽说功能实现大同小异，但命名可是千差万别啊！！我们的系统原来一直使用的阿里云的业务，但是这回要增加极光和百

度云的信息功能，一来做个后备，二来根据不同业务使用不同的接口达到安全或节约的目的，有没有办法统一一下他们对外的接口，让我们使用他们的SDK时能够非常方便的和之前使用大家都已经很习惯的阿里云的接口一样呢？当然有，给他们各自都上个适配器呗，实例化的时候大不了外面再套个工厂返回不同的适配器就好啦，只要适配器里的实现方法和阿里云一样就OK啦！

> 短信发送类图

完整源码：短信发送适配器方法

```php
<?php

class Message{
    public function send(){
        echo "阿里云发送短信！" . PHP_EOL;
    }
    public function push(){
        echo "阿里云发送推送！" . PHP_EOL;
    }
}

class JiguangSDKAdapter extends Message{
    private $message;

    public function __construct($message){
        $this->message = $message;
    }

    public function send(){
        $this->message->send_out_msg();
    }
    public function push(){
        $this->message->push_msg();
    }
}

class JiguangMessage{
    public function send_out_msg(){
        echo "极光发送短信！" . PHP_EOL;
    }
```

```php
    public function push_msg(){
        echo "极光发送推送！" . PHP_EOL;
    }
}
class BaiduYunSDKAdapter extends Message{
    private $message;

    public function __construct($message){
        $this->message = $message;
    }

    public function send(){
        $this->message->transmission_msg();
    }
    public function push(){
        $this->message->transmission_push();
    }
}
class BaiduYunMessage{
    public function transmission_msg(){
        echo "百度云发送短信！" . PHP_EOL;
    }
    public function transmission_push(){
        echo "百度云发送推送！" . PHP_EOL;
    }
}

$jiguangMessage = new JiguangMessage();
$baiduYunMessage = new BaiduYunMessage();
$message = new Message();

// 原来的老系统发短信，使用阿里云
$message->send();
$message->push();


// 部分模块用极光发吧
$jgAdatper = new JiguangSDKAdapter($jiguangMessage);
$jgAdatper->send();
$jgAdatper->push();
```

```php

// 部分模块用百度云发吧
$bdAatper = new BaiduYunSDKAdapter($baiduYunMessage);
$bdAatper->send();
$bdAatper->push();
```

> 说明

- 在这个例子中，我们有两个适配器，因为有两个SDK需要我们去适配，谁说只能有一个电源转换器，万一哪个神奇的国度是用500伏的电压呢，所以还是多带个电源转换器吧
- 这里我们是继承的Message类，因为Message类是之前已经写好的代码，里面可能有一些可以公用的方法，所以并没有做接口抽象。可以考虑在重构代码的时候实现提取一个抽象接口，但在这里只是为了演示适配器不一定只是能去针对接口，只要和原对象保持一致，不去继承什么也是可以的，毕竟我们是弱类型语言，如果是类似于Java的强类型，那么继承或者实现还是很有必要的（多态性）
- 组合式的适配器与装饰器类似，都会维护一个外部对象，装饰器更多的会使用原来的类中的方法，对其进行增加功能的操作，而适配器则很少去增加功能，而是直接替换掉
- Laravel中的Filesystem模块，有一个FilesystemAdapter类，我觉得没啥可说的了，很明显的告诉大家咱用了适配器模式，好好研究一下吧
- 当你想使用一个类，但他提供的内容跟你的业务又不太匹配的时候；或者你想创建一个类，可以与其他不相关的类或不可预见的类协同工作的时候，不妨试试适配器模式吧