PHP设计模式之简单工厂模式

先从简单工厂入门，不管是面试还是被他人面试，在问到设计模式的时候，大多数人都会提到工厂模式。毫无疑问，工厂相关的几种模式在设计模式中是最出名的也是应用比较广泛的一种模式。在GoF设计模式中也都是属于创建型的模式。

但是，能够说明白**简单工厂**、**工厂模式**、**抽象工厂模式**这三种模式的人还真能让面试官刮目相看。这里有个前提，是你真的能说明白，大部分人，包括在深入研究设计模式之前，我也没办法说清楚。不管是我去面试，还是面试其别人。当我面试别人的时候，能讲个大概也就差不多了。而我去面试的时候，也就是类似的讲个大概。经历不少挫折之后才有了想深入的研究研究设计模式的想法，于是便会产生这一系列的文章。从这篇简单工厂开始，我们一起再次深入的对设计模式进行学习。

当然，这里用的是PHP。

解释

简单工厂，也称静态工厂，不属于GoF23种设计模式。但是可以说是所有的设计模式中大家可能最容易理解，也可能在你的代码中早就已经用过不知道多少次的一种设计模式了。我们先从一个最最简单的代码段来看。

```php
// Factory
class Factory
{
    public static function createProduct(string $type) : Product
    {
        $product = null;
        switch ($type) {
            case 'A':
                $product = new ProductA();
                break;
            case 'B':
                $product = new ProductB();
                break;
        }
        return $product;
    }
}
```

没错，核心点就是中间那段简单的switch代码，我们在返回值类型中固定为Product接口的实现。

***在这段代码中，使用了PHP新特性，参数类型及返回值类型**

产品接口和产品实现

```php
// Products
interface Product
{
    public function show();
}

class ProductA implements Product
{
    public function show()
    {
        echo 'Show ProductA';
    }
}

class ProductB implements Product
{
    public function show()
    {
        echo 'Show ProductB';
    }
}
```

最后客户端的使用就很简单了

```php
// Client
$productA = Factory::createProduct('A');
$productB = Factory::createProduct('B');
$productA->show();
$productB->show();
```

从以上代码可以看出，其实这里就是一个工厂类根据我们传入的字符串或者其他你自己定义的标识符，来返回对应的产品（Product对象）。

形象化一点的比喻：我是一个卖手机的批发商（客户Client，业务方），我需要一批手机（产品Product），于是我去让富士康（工厂Factory）来帮我生产。我下了订单（$type变量）指明型号，

然后富士康就给我对应型号的手机，然后我就继续我的工作了，和富士康的合作还真是挺愉快的。
这里比较规范的写法可能是所有产品都会去实现一个统一的接口，然后客户端只知道接口的方法统一调用即可。不规范的话也可以不使用接口，返回各种不同的对象，类似于外观（Facade）模式进行统一的门面管理。

源码地址：简单工厂基础类图实现

实例

场景：短信发送功能模块。现在我们使用了三个商家的，分别是阿里云、蝶信、极光的短信服务，在不同业务中可能使用不同的短信发送商，使用简单工厂可以方便的完成这个需求。

> 类图

> 代码

```php
<?php

interface Message {
    public function send(string $msg);
}

class AliYunMessage implements Message{
    public function send(string $msg){
        // 调用接口，发送短信
        // xxxxx
        return '阿里云短信（原阿里大鱼）发送成功！短信内容：' . $msg;
    }
}

class BaiduYunMessage implements Message{
    public function send(string $msg){
        // 调用接口，发送短信
        // xxxxx
        return '百度SMS短信发送成功！短信内容：' . $msg;
    }
}

class JiguangMessage implements Message{
```

```
    public function send(string $msg){
        // 调用接口，发送短信
        // xxxxx
        return '极光短信发送成功！短信内容：' . $msg;
    }
}

Class MessageFactory {
    public static function createFactory($type){
        switch($type){
            case 'Ali':
                return new AliYunMessage();
            case 'BD':
                return new BaiduYunMessage();
            case 'JG':
                return new JiguangMessage();
            default:
                return null;
        }
    }
}

// 当前业务需要使用极光
$message = MessageFactory::createMessage('Ali');
echo $message->send('您有新的短消息，请查收');
```

源码地址：简单工厂实例-短信发送工厂

> 说明

- createMessage一定要使用static？不一定，看自己业务情况决定，需要常驻的全部static，按需实例化的就new完了再正常->去调用
- 三个message子类一定需要实现接口？也不一定，php本身就是弱类型语言，可以不去强制实现，但使用接口更加符合面向对象的规范（请参考多态），简单工厂本身其实是对多态的一种表述和应用
- 当需要增加发送消息的渠道时，添加新类继承Message接口，实现send()方法，修改MessageFactory()中createFactory()方法里的switch
- 思考上述修改违背了面向对象的什么原则？（提示：开放XX）
- 实例中没有使用返回值类型？本身这也是新语法，对于老版本兼容来说可以不需要去写，但是如果公司技术栈已经升级到7以上的话，建议这种设计模式架构类的代码还是按照上方解释中的新特性写法去书写，也就是带上参数类型和返回值类型，更加的符合规范，也更容易理解

- 实例中传错了$type返回NULL怎么办？亲，实际写代码的时候请处理好这个问题哟，返回一个默认的，或者上层捕获都是不错的解决方案，当然最好客户端那边提前判断好，没问题了再进工厂吧