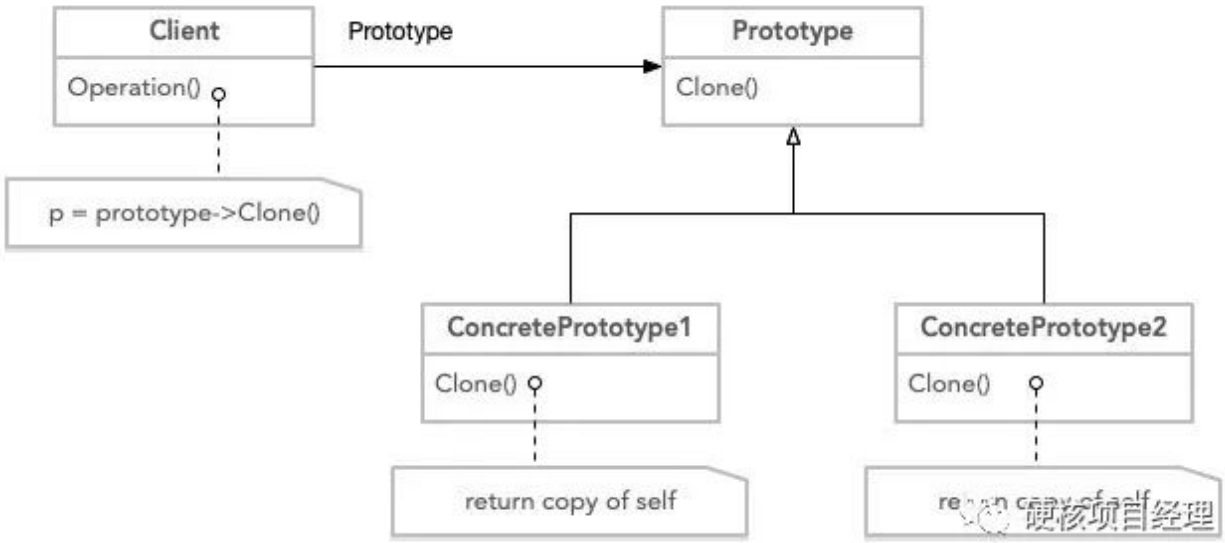
PHP设计模式之原型模式

原型模式其实更形象的来说应该叫克隆模式。它主要的行为是对对象进行克隆，但是又把被克隆的对象称之为最初的原型，于是，这个模式就这样被命名了。说真的，从使用方式来看真的感觉叫克隆模式更贴切一些。

Gof类图及解释

GoF定义：用原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象

> GoF类图



> 代码实现

```php
abstract class Prototype{
    public $v = 'clone' . PHP_EOL;

    public function __construct()    {
        echo 'create' . PHP_EOL;
    }

    abstract public function __clone();
}
```

首先我们通过模拟的方式定义了一个原型，这里主要是模拟了__clone()这个方法。其实这是PHP自带的一个魔术方法，根本是不需要我们去进行定义的，只需要在原型类中进行实现就可以了。当外部使用clone关键字进行对象克隆时，直接就会进入这个魔术方法中。在这个魔术方法里面我们可以对属性

进行处理，特别是针对引用属性进行一些独特的处理。在这个例子中，我们只使用了一个值类型的变量。无法体现出引用类型的问题，我们将在后面的实例中演示对引用类型变量的处理。

```
class ConcretePrototype1 extends Prototype{
    public function __clone()    {
    }
}

class ConcretePrototype2 extends Prototype{
    public function __clone()    {
    }
}
```

模拟的具体实现的原型，其实就是主要去具体的实现__clone()方法。后面我们看具体的例子时再说明。

```
class Client{
    public function operation()    {
        $p1 = new ConcretePrototype1();
        $p2 = clone $p1;

        echo $p1->v;
        echo $p2->v;
    }
}

$c = new Client();
$c->operation();
```

客户端使用clone来复制p1,可以看到p2也具有相同的$v属性。

- 原型模式看似就是复制了一个相同的对象，但是请注意，复制的时候，__construct()方法并没有被调用，也就是当你运行这段代码的时候，create只输出了一次。这也就带出了原型模式最大的一个特点——减少创建对象时的开销。
- 基于上述特点，我们可以快速的复制大量相同的对象，比如要给一个数组中塞入大量相同的对象时。
- 复制出来的对象中如果都是值类型的属性，我们可以任意修改，不会对原型产生影响。而如果有引

用类型的变量，则需要在__clone()方法进行一些处理，否则修改了复制对象的引用变量中的内容，会对原型对象中的内容有影响。

我们的手机操作系统（也可以想象一下PC电脑的操作系统），都是怎样安装到设备中呢？其实都是不停的复制拷贝最初的那一套系统。用微软的例子非常好说明这个问题，当年微软能够成为一个帝国，其实也是因为他不停的将winodws操作系统拷贝复制到光盘中，然后卖给千家万户（当然，这里没中国什么事儿）。而中国市场呢，大量的高手破解了windows之后也是由这一份文件不停的复制拷贝才装到了我们的电脑中。手机、智能设备等各类产品的操作系统、软件都是如此。一次开发无限拷贝正是软件行业暴利的原因。毕竟我们的系统也是由不少的工程师日以继夜的996在Android原生系统的基础上开发出来的，赶紧不断的复制到即将出厂的手机上吧！！

完整代码：https://github.com/zhangyue0503/designpatterns-php/blob/master/08.prototype/source/prototype.php

实例

同样还是拿手机来说事儿，这次我们是根据不同的运营商需要去开发一批定制机，也就是套餐机。这批手机说实话都并没有什么不同，大部分都是相同的配置，但是运营商系统不同，而且偶尔有一些型号的CPU和内存也可能存在不同。这个时候，我们就可以用原型模式来进行快速的复制并且只修改一部分不相同的地方啦。

> 原型模式生产手机类图

完整源码：https://github.com/zhangyue0503/designpatterns-php/blob/master/08.prototype/source/prototype-phone.php

```php
<?php
interface ServiceProvicer{
    public function getSystem();
}

class ChinaMobile implements ServiceProvicer{
    public $system;
    public function getSystem(){
        return "中国移动" . $this->system;
    }
}
class ChinaUnicom implements ServiceProvicer{
    public $system;
    public function getSystem(){
        return "中国联通" . $this->system;
```

```php
    }
}

class Phone {
    public $service_province;
    public $cpu;
    public $rom;
}

class CMPhone extends Phone{
    function __clone()    {
        // $this->service_province = new ChinaMobile();
    }
}

class CUPhone extends Phone{
    function __clone()    {
        $this->service_province = new ChinaUnicom();
    }
}


$cmPhone = new CMPhone();
$cmPhone->cpu = "1.4G";
$cmPhone->rom = "64G";
$cmPhone->service_province = new ChinaMobile();
$cmPhone->service_province->system = 'TD-CDMA';
$cmPhone1 = clone $cmPhone;
$cmPhone1->service_province->system = 'TD-CDMA1';

var_dump($cmPhone);
var_dump($cmPhone1);
echo $cmPhone->service_province->getSystem();
echo $cmPhone1->service_province->getSystem();


$cuPhone = new CUPhone();
$cuPhone->cpu = "1.4G";
$cuPhone->rom = "64G";
```

```php
$cuPhone->service_province = new ChinaUnicom();
$cuPhone->service_province->system = 'WCDMA';
$cuPhone1 = clone $cuPhone;
$cuPhone1->rom = "128G";
$cuPhone1->service_province->system = 'WCDMA1';

var_dump($cuPhone);
var_dump($cuPhone1);
echo $cuPhone->service_province->getSystem();
echo $cuPhone1->service_province->getSystem();
```

> 说明

- 打印了很多东西呀，不过主要的还是看看移动手机，也就是CMPhone中的__clone()方法，我们没有重新去初始化一个新对象。这时，复制的$cmPhone1对象中的service_province和$cmPhone中的是同一个对象。没错，这就是引用的复制问题。引用只是复制了引用的地址，他们指向的是同一个对象。当$cmPhone1修改service_province对象里面的属性内容时，$cmPhone里面的service_province对象里面的属性也跟着改变了。
- 在CUPhone中，我们重新new了一个新的service_province对象。这次外面的$cuPhone1对该对象中的属性修改时就不会影响$cuPhone中引用对象的值。
- 原型模式中最主要的就是要注意上述两点，而普通的值属性会直接进行复制，不会产生这个问题。这里又牵涉出另外两个概念：浅复制和深复制
- 浅复制，是指被复制对象的所有变量都含有与原来对象相同的值，而所有的对其他对象的引用都仍然指向原来的对象
- 深复制把引用对象的变量指向复制过的新对象，而不是原有的被引用的对象
- 关于引用和值的问题，我们将在其他的文章中进行讲解，请关注微信或掘金号