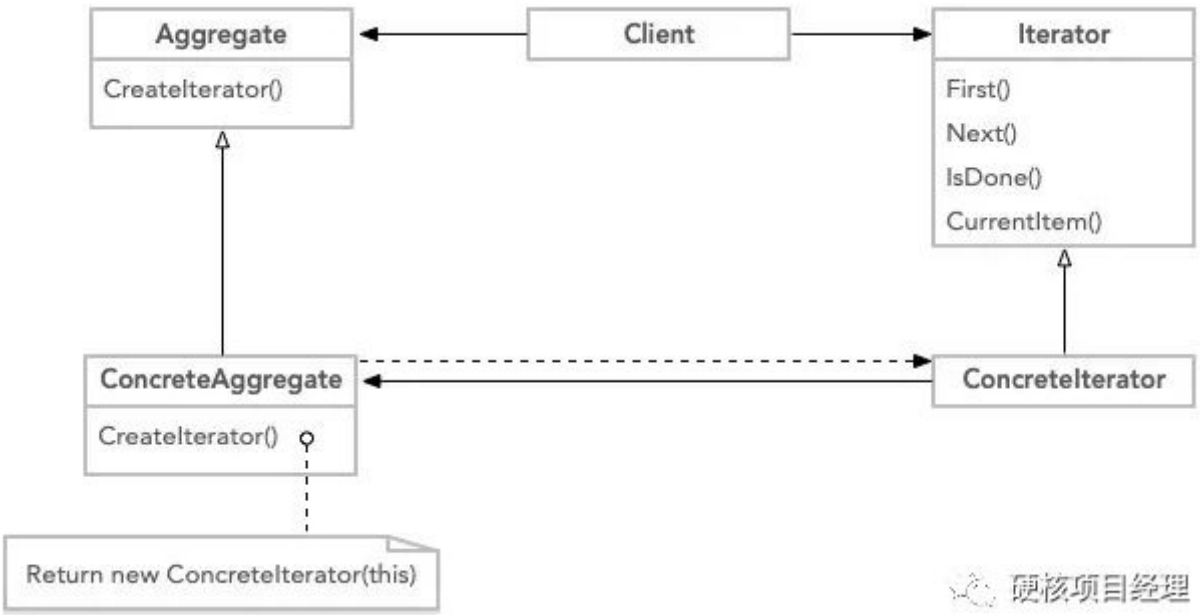
PHP设计模式之迭代器模式

一说到这个模式，就不得不提循环语句。在《大话设计模式》中，作者说道这个模式现在的学习意义更大于实际意义，这是为什么呢？当然就是被foreach这货给整得。任何语言都有这种类似的语法可以方便快捷的对数组、对象进行遍历，从而让迭代器模式从高高在上的23大设计模式中的明星慢慢成为了路人。特别是我们这门PHP语言，PHP的强大之处就在于对于数组的灵活操作，本身就是hashmap的结构，自然会有各种方便的数组操作语法，而foreach也是我们最常用的语句，甚至比for还常用。

Gof类图及解释

GoF定义：提供一种方法顺序访问一个聚合对象中各个元素，而又不需暴露该对象的内部表示

> GoF类图



> 代码实现

```php
interface Aggregate{
    public function CreateIterator();
}

class ConcreteAggregate implements Aggregate{
    public function CreateIterator()    {
        $list = [
            "a",
            "b",
            "c",
            "d",
```

```
        ];
        return new ConcreteIterator($list);
    }
}
```

首先是聚合类，也就是可以进行迭代的类，这里因为我是面向对象的设计模式，所以迭代器模式针对的是对一个类的内容进行迭代。在这里，其实我们也只是模拟了一个数组交给了迭代器。

```
interface MyIterator{
    public function First();
    public function Next();
    public function IsDone();
    public function CurrentItem();
}

class ConcreteIterator implements MyIterator{
    private $list;
    private $index;
    public function __construct($list)    {
        $this->list = $list;
        $this->index = 0;
    }
    public function First()    {
        $this->index = 0;
    }

    public function Next()    {
        $this->index++;
    }

    public function IsDone()    {
        return $this->index >= count($this->list);
    }

    public function CurrentItem()    {
        return $this->list[$this->index];
    }
}
```

迭代器闪亮登场，主要实现了四个方法来对集合数据进行操作。有点像学习数据结构或数据库时对游标进行的操作。用First()和Next()来移动游标，用CurrentItem()来获得当前游标的数据内容，用IsDone()来确认是否还有下一条数据。所以，这个模式也另称为**游标模式**。

```php
$agreegate = new ConcreteAggregate();
$iterator = $agreegate->CreateIterator();

while (!$iterator->IsDone()) {
    echo $iterator->CurrentItem(), PHP_EOL;
    $iterator->Next();
}
```

客户端直接使用while来进行操作即可。

- 大家一定很好奇，为什么我们的迭代器接口类不用Iterator来命名？试试就知道，PHP为我们准备好了一个这个接口，实现之后就可以用foreach来使用这个实现了Iterator接口的类了，是不是很高大上。我们最后再看这个类的使用。
- 不是说好对类进行遍历吗？为啥来回传递一个数组？开发过Java的同学一定知道，在一个名为Object类的JavaBean中，会写一个变量List类型的变量如List myList，用来表示当前对象的集合。在使用的时候给这个List添加数据后，下次就可以直接用Object.myList来获得一组数据了。比如从接口中获得的json数组内容就可以这样存在一个Bean中。这时，我们使用迭代器就可以只针对自己这个对象内部的这个数组来进行操作啦！
- 上述Java的内容其实是笔者在做Android开发时经常会用到的，有时数据库的JavaBean也会出现这种数组来存储外键。但在PHP中一般很少使用，因为PHP中大部分的AR对象和Java中的Bean概念还是略有不同。有兴趣的同学可以了解下！

我们的手机工厂不得了，自己组装了一条生产线，这条生产线主要是做什么的呢？成型机我们已经交给富X康来搞定了，我们这条线就是给手机刷颜色的。当我们把所有已经交货的手机（Aggregate）放到不同的生产线后（Iterator），就会一台一台的帮我们刷上当前生产线的颜色，是不是很强大！！科技不止于换壳，这条线还在，我们就可以再做别的事儿，比如加点挂绳什么的，反正只要能一台一台的通过我就能装上东西，你说好用不好用！！

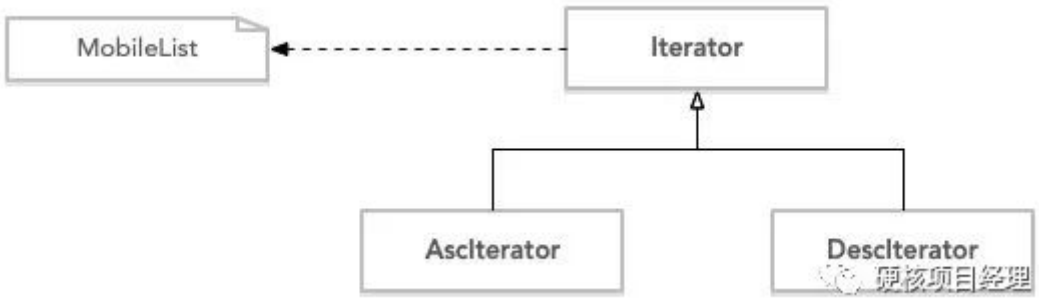
完整代码：https://github.com/zhangyue0503/designpatterns-php/blob/master/07.iterator/source/iterator.php

实例

实例还是围绕着我们的短信发送来看。这一次，我们的业务需求是尽快的发一批通知短信给用户，因为活动的时候可不等人啊。在之前我们会使用多个脚本来把用户手机号分成多组来进行发送。现在我们可以用swoole来直接多线程的发送。所要达到的效果其实就是为了快速的把成百上千的短信发完。

这个时候我们也会做一些策略，比如数据库里是100条要送的短信，有个字段是发送状态，一个线程正序的发，一个线程倒序的发，当正序和倒序都发送到50条的时候其实已经同步的发完这100条了，不过也有可能会有失败的情况出现，这时，两个线程还会继续去发送那些上次发送不成功的信息，这样能够最大程度的确保发送的效率和到达率。

> 消息发送迭代器类图



完整源码：https://github.com/zhangyue0503/designpatterns-php/blob/master/07.iterator/source/iterator-msg.php

```php
<?php

interface MsgIterator{
    public function First();
    public function Next();
    public function IsDone();
    public function CurrentItem();
}

// 正向迭代器
class MsgIteratorAsc implements MsgIterator{
    private $list;
    private $index;
    public function __construct($list)    {
        $this->list = $list;
        $this->index = 0;
    }
    public function First()    {
        $this->index = 0;
    }

    public function Next()    {
```

```php
        $this->index++;
    }

    public function IsDone()    {
        return $this->index >= count($this->list);
    }

    public function CurrentItem()    {
        return $this->list[$this->index];
    }
}

// 反向迭代器
class MsgIteratorDesc implements MsgIterator{
    private $list;
    private $index;
    public function __construct($list)    {
        // 反转数组
        $this->list = array_reverse($list);
        $this->index = 0;
    }
    public function First()    {
        $this->index = 0;
    }

    public function Next()    {
        $this->index++;
    }

    public function IsDone()    {
        return $this->index >= count($this->list);
    }

    public function CurrentItem()    {
        return $this->list[$this->index];
    }
}

interface Message{
    public function CreateIterator($list);
```

```php
}

class MessageAsc implements Message{
    public function CreateIterator($list)    {
        return new MsgIteratorAsc($list);
    }
}
class MessageDesc implements Message{
    public function CreateIterator($list)    {
        return new MsgIteratorDesc($list);
    }
}

// 要发的短信号码列表
$mobileList = [
    '13111111111',
    '13111111112',
    '13111111113',
    '13111111114',
    '13111111115',
    '13111111116',
    '13111111117',
    '13111111118',
];

// A服务器脚本或使用swoole发送正向的一半
$serverA = new MessageAsc();
$iteratorA = $serverA->CreateIterator($mobileList);

while (!$iteratorA->IsDone()) {
    echo $iteratorA->CurrentItem(), PHP_EOL;
    $iteratorA->Next();
}

// B服务器脚本或使用swoole同步发送反向的一半
$serverB = new MessageDesc();
$iteratorB = $serverB->CreateIterator($mobileList);

while (!$iteratorB->IsDone()) {
```

```
102         echo $iteratorB->CurrentItem(), PHP_EOL;
103         $iteratorB->Next();
104     }
105
```

说明

- 其实就是两个迭代器，一个是正序一个是倒序，然后遍历数组
- 例子中我们还是对一个数组的操作，另外用两个类似于工厂方法模式的类来对迭代器进行封装
- 例子非常简单，但有时候这种用法也非常实用，比如一些搜索引擎排名的爬虫，多次确认某些关键词的排名，这时候我们就可以正着、反着来回进行验证

完整源码：https://github.com/zhangyue0503/designpatterns-php/blob/master/06.observer/source/spl_observer.php

> 彩蛋

PHP中的Iterator接口已经为我们准备好了一套标准的Iterator模式的实现，而且（这里需要画重点），实现这个接口的类可以用foreach来遍历哦！

文档：https://www.php.net/manual/zh/class.iterator.php

源码：https://github.com/zhangyue0503/designpatterns-php/blob/master/07.iterator/source/iterator-php.php

文档中相关的接口都可以看看，更重要的是，PHP的SPL扩展中，也为我们准备了很多常用的迭代器封装。要知道，面试的时候要是能说出这里面的几个来，那面试官可是也会刮目相看的哦！

SPL迭代器：https://www.php.net/manual/zh/spl.iterators.php