互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

第12章 中介者模式

复杂的代码不会受到任何人的赞赏。大量的耦合、易管理的相关性和整体式代码流能够很好地证明编程人员出众的能力、辛勤的工作和高超的智慧。我无法相信复杂的代码会得到恶评。现在，无论是初级的还是经验丰富的编程人员，都应当忘记所有的讽刺并尽快振作起来。在很多情况下，为了在处理软件应用程序时保持清醒的头脑，编程人员必须保持这样的心态。增加的功能、扩张的范围以及相关人员众多都会导致代码库的交织。术语“面向对象(object oriented)”似乎已经消失，这种不可分开的代码一度是非常流行、快速和模块化的系统。不过，系统内部的类似对象相互的查找以及彼此应用的更新过多，并且耦合通常过于紧密。但是，我们还是有希望解决这种问题的。在许多并非紧密耦合的类似对象需要接受改变时，中介者设计模式就可以提供帮助。

名称：中介者

中介者设计模式用于开发一个对象，这个对象能够在类似对象相互之间不直接交互的情况下传送或调解对这些对象的集合的修改。

12.1 问题与解决方案

当对象的耦合过于紧密时，面向对象编程的优点就会显现出来。虽然仍旧在处理对象，但是这种样式开始向程序化转换。基本代码变得整体化且较为笨重。在能够应用解决方案之前，我们需要深入研究这个特定的问题。理解根本的原因是十分重要的。中介者设计模式解决的相同问题会缓慢地蔓延至具体代码的其他实例中。

在对象并未被专门设计为在开始就意识到相互之间的关系时，问题就会接踵而至。现在，对象时常被创建用于处理子对象或集合。这是一个非常完美的体系结构选项。然而，当可交换的对象或不基于相同架构的对象开始具有相互依赖性时，问题也就随之而来——往往是在对象被更新或对象接口发生变化的时候，或者是需要对某个对象应用更新的时候。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

为此，我们似乎只要添加一个新方法，以便使用相同的信息更新类似的对象。但是，如果添加这个方法，指定对象就会理解和联系不需耦合的另一个对象。最直接的后果是由另一对象公共方法的改变造成的。此时，不仅需要修改另一个对象，而且也需要更新与之无关的指定对象的方法。

基于中介者设计模式的对象在这些有关系但不耦合的对象之间提供了一个非常必要的通信中心。一个类似但是无关的对象会受到某个改变的影响，它会向中介者对象提及这个改变。通过将该改变应用于其他所有对象，中介者对象随后进行调解以促成这些对象能够接受该改变。最开始发生改变的对象并不知道有多少其他的对象也进行了相同的改变。除了应当应用改变之外，其他类似的对象并不知道更新的来源。

通过雇员和老板就可以阐明这种行为在现实生活中的实例。某位雇员决定在家自学一项新的综合技能。在学习新的综合技能时，他并未联系其他任何雇员。学习完成后，这位雇员向老板告知了自己所掌握的新技能。老板觉得这种做法很好，并且通告其他所有雇员都应当学习这项新技能。随后，其他雇员就按照自己的方式开始学习新的技能。

通过销售系统，我们也可以了解中介者设计模式的具体表现。假设销售系统被用于出售某个音乐店的吉他。通过创建一个吉他对象就可以访问库存的每把吉他。在销售管理界面上，卖主可以选择对特定的吉他进行打折。因为某些供应商可能想进行特别的推销活动，所以卖主也能够通过选择一个复选标记对特定品牌的所有吉他进行打折。如果选择了复选项，那么更新就能够被正常地应用。不过，中介者(也就是卖主)必须被告知这个改变(也就是举办的推销活动)以及应当对其他吉他对象应用该改变。

在源对象的改变应当被传递给其他有联系但不耦合的对象时，就应当使用基于中介者设计模式的对象来管理这些更新。

12.2 UML

如图 12-1 所示，该 UML 图详细说明了一个使用中介者设计模式的类设计。

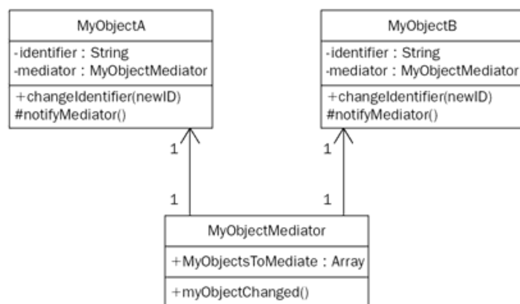

图 12-1

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

下面是对图 12-1 的说明：

- 图 12-1 中存在两个相似的类：MyObjectA 和 MyObjectB。这两个类显现的外形是相同的。它们的差异可能是用私有变量 identifier 表示的标识符。此外，所有函数都是相似的。
- 在对象创建期间，MyObjectMediator 的实例被存储在内部。随后，如果通过调用公共方法 cahngeIdentifier( )请求针对某个对象的变化，那么参数 newID 通过更新私有标识符字符串就能够被应用于该对象。接下来，调用保护方法 notifyMediator( )接可以对其余对象应用中介。
- MyObjectMediator 是一系列对象的中心所在。这些对象存储在数组 MyObjectsToMediate 中。MyObjectsToMediate 接到通知时就会执行 myObjectChanged( )方法，该方法负责解析数组 MyObjectsToMediate 并对其他所有对象应用指定的改变。

12.3 代码示例

示例 Web 站点不仅允许乐队进入和管理他们的音乐合辑，而且还允许乐队更新他们的配置文件、修改乐队相关信息以及更新其 CD 信息。现在，艺术家可上传 MP3 集合并从 Web 站点上撤下 CD。因此，Web 站点需要保持相对应的 CD 和 MP3 彼此同步。

Web 站点的最初版本允许乐队从配置文件页面或单独的 CD 本身修改其乐队名称。如下所示，CD 对象具有一个接受乐队变化以及在数据库中进行更新的方法：

```
class CD
{
    public $band = '' ;
    public $title = '' ;

    public function save()
    {
        //stub - writes data back to database - use this to verify
        var_dump($this);
    }

    public function changeBandName($newName)
    {
        $this->band = $newName;
        $this->save();
    }
}
```

互动出版网
China-Pub.com
专业图书网上第一专营店
www.china-pub.com
独家提供样章

上面这个简单的类只是说明了 CD 对象可以具有乐队名和标题。接下来，函数 changeBandName( )接受一个新的乐队名参数。这个函数将该参数插入指定对象并调用 save( )方法。出于演示说明的目的，所以这里没有进一步说明 save( )方法。读者可以自己验证已完成的相应变化。

为了添加 MP3 归档文件，就需要创建另一个类似的对象来处理归档文件。艺术家必须也能够在 MP3 归档文件页面上修改其乐队名。同样，在与之关联的 CD 中也必须能够修改乐队名。

现在应当使用中介者设计模式。首先，为了使用该模式，必须修改 CD 类。随后，创建与 CD 类相似的 MP3 归档类。

```
class CD
{
    public $band = '' ;
    public $title = '' ;
    protected $_mediator;

    public function __construct($mediator = null)
    {
        $this->_mediator = $mediator;
    }

    public function save()
    {
        //stub - writes data back to database - use this to verify
        var_dump($this);
    }

    public function changeBandName($newName)
    {
        if (!is_null($this->_mediator)) {
            $this->_mediator-> change($this, array('band'=>$newName));
        }
        $this->band = $newName;
        $this->save();
    }
}

class MP3Archive
{
    public $band = '' ;
    public $title = '' ;
```

互动出版网
China-Pub.com
专业图书网上第一专营店
www.china-pub.com
独家提供样章


```
        protected $_mediator;

        public function __construct($mediator = null)
        {
            $this->_mediator = $mediator;
        }

        public function save()
        {
            //stub - writes data back to database - use this to verify
            var_dump($this);
        }

        public function changeBandName($newName)
        {
            if (!is_null($this->_mediator)) {
                $this->_mediator->change($this, array('band'=>$newName));
            }
            $this->band = $newName;
            $this->save();
        }
    }
```

对 CD 对象的第一个改变被添加入名为$_mediator 的保护变量，该变量用于存储中介对象的实例。MP3Archive 类中添加了构造函数。创建 CD 类的一个实例时，新的中介者对象应当被传递至这个类内。不过需要注意的是，$_mediator 变量的默认值为 null。这不仅允许我们创建用于只读功能的、实例中没有中介者的对象，而且也确保在使用中介者对象更新其他类时不会创建无限的循环。接下来，changeBandName( )方法会被修改。该方法用于查看中介者对象是否存在和不为空。如果存在且不为空，那么就会调用中介者对象的 change( )方法，从而传递入自身的实例并修改指定项的键控数组。

上述操作在对中介者对象本身应用更新之前发生。对于中介者来说，在修改指定项前获取其快照是十分重要的。

MP3Archive 对象几乎与 CD 对象完全相同。

如下所示，随后需要创建中介者类：

```
    class MusicContainerMediator
    {
        protected $_containers = array();

        public function __construct()
        {
```

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章


```
            $this->_containers[] = 'CD';
            $this->_containers[] = 'MP3Archive';
        }

        public function change($originalObject, $newValue)
        {
            $title = $originalObject->title;
            $band = $originalObject->band;

            foreach ($this->_containers as $container) {
                if (!($changedObject instanceof $container)) {
                    $object = new $container;
                    $object->title = $title;
                    $object->band = $band;

                    foreach ($newValue as $key=>$val) {
                        $object->$key = $val;
                    }

                    $object->save();
                }
            }
        }
    }
```

中介者对象知道其将要中介调解的所有音乐容器(Music Container)。构造函数用于构建将要中介调解的对象的内部数组。如果今后创建了新的音乐容器，那么中介者类中所需的唯一变化就是在保护数组$_containers 内添加一个新的元素。

更改方法将接受原始对象以及将要添加的新值。首先，原始对象中的标题和乐队名会被检索出来。接着，所有音乐容器会被循环遍历。如果$originalObject 不是这些容器的实例，那么就会创建相应的容器。将指定容器与原始对象进行比较的原因是为了减少重复。完成对中介者的操作后，变化就会应用于原始对象。此时，我们不需要创建和更改原始对象的副本，从而跳过了循环。

如果指定容器被创建为一个新的对象，那么就要根据原始对象设置标题和乐队名。最后，$newValue 的任何变化都会被循环遍历，然后应用于这个新的对象，接着该对象才会被保存。

上述整个处理过程之后，原始对象就会应用自己的更新并自动保存。

使用新的中介者对象，所采用的代码十分简单，如下所示：

```
$titleFromDB = 'Waste of a Rib';
```

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章


```
$bandFromDB = 'Never Again';

$mediator = new MusicContainerMediator();
$cd = new CD($mediator);
$cd->title = $titleFromDB;
$cd->band = $bandFromDB;


$cd->changeBandName('Maybe Once More');
```

处理具有类似属性并且属性需要保持同步的非耦合对象时，最佳的做法是使用基于中介者设计模式的对象。