[**.NET中的设计模式---策略模式&状态模式**](http://www.cnblogs.com/niyw/archive/2010/07/17/1779575.html)

     凡是涉及过设计模式的人都应该听过一些模式的名字,其中包括策略模式和状态模式.它们就是我今天要阐述的对象.之所以放在一起,是因为二者有某种关系可以帮助大家真正了解设计模式.为什么没有第三种,因为那样子太多了,说不清.

对于这两种模式官方的定义如下:

**策略模式:** 定义一系列算法，把它们一个个封装起来，并使它们能相互替换。使得算法可以独立于它的客户而变化。

**状态模式:**允许一个对象在其内部状态改变时改变它的行为。对象看起来似乎修改了它的类。

这两个定义理解起来相当容易,二者大致可以概括为”对象根据不同的条件灵活选择不同的行为/方法”.这么看来策略模式和状态模式岂不是一样了?!现在看来确实是这么一回事.不过先别这么早的下结论,我们继续向下看.

这次看看二者的类图,如下:

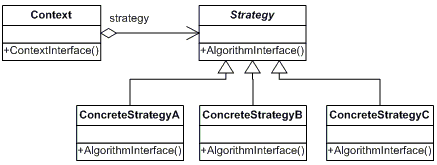


图1.策略模式

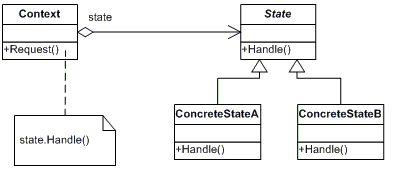


图2.状态模式

       这时候细心的人会发现这两种模式的UML类图完全一样.是不是觉得有点糊涂了?!这里先卖个关子,说点别的.

      在平时大家学习设计模式的时候也许不会发现今天我给大家举出的情况,也不会觉得自己对设计模式的理解有什么不妥.不过你真的理解了吗?网上有很多关于设计模式的帖子,不管内容还是论调几乎都一样,随随便便的拿个例子或代码说明自己学会了设计模式.但是如果仔细比对,发现他们的内容几乎和教科书上一样,并且只包含其中的一部分.这样的结果是人云亦云,糊涂的继续糊涂,不懂得继续装懂.

       好了,言归正传,回来继续说我们的策略模式和状态模式.通过上面说明,大家会发现不管从实现还是从设计,或者总结,二者几乎一模一样.那么二者真的是同一种模式吗?答案有两种,第一,是;第二不是.

       答案一.”是”:是从某些开发或设计的角度看的,就像上面的过程.不过这只是表面现象;

       答案二”不是”:很显然四人组比我们绝大多数人水平高的多,相比之下,我们出错的几率比他们高的多.那么这两种设计模式的区别到底在哪呢?

       看过书大家都知道策略模式和状态模式都属于Gof23种设计模式中的**行为模式**.二者外表一样,只是二者的关注点不一样.策略模式关注**行为的变化**,状态模式关注**对象状态的变化如何导致对象行为的变化**.

**策略模式中的两个角色:客户和策略.客户的职责是了解行为的差别,定义同一的接口.策略的职责是提供不同的实现,并根据客户的意图来选择.**至于有多少种行为,一般很难知道,因为随着发展会有不同的变化加入.现实中一个简单的例子就是根据客户的要求选择不同的加密算法.当有新的加密算法出现时,客户需要知道这个变化,此时策略此可以灵活的选择使用新的算法.

       状态模式一个形象的比喻是”饱了睡,饿了吃”.在这里”饱和饿”是两种状态,”睡和吃”是两种行为.另外一个典型的例子是银行账户.根据客户账户中余额的不同用户可以有不同的操作行为.这里要注意到,状态模式中状态与行为的对应关系.虽然不是一一对应,但潜藏了一些信息,那就是实际例子中行为与状态的有限和稳定,行为的唯一性**.有限和稳定是指对象的行为一般就那几种,除非业务需求变动,否则不会发生改变.唯一性只是对象只有一个吃的行为,二不会有第二个吃的行为.**

这时候策略模式与状态模式之间的区别就清楚了.

**策略模式关注行为的变化,但归根结底只有一个行为,变化的只是行为的实现.客户不关注这些.当新增变化时对客户可以没有任何影响.**

**状态模式同样关注行为的变化,但这个变化是由状态来驱动,一般来说每个状态和行为都不同.新增的状态或行为一般与已有的不同,客户需要关注这些变化.**

       下面举两个实际中的例子.

       一,在移动网络优化中,一般可以采用遗传算法,模拟退火算法和基于拓扑关系的算法.实际项目中,往往我们可以根据不同情况采用不同的算法,而客户对用什么算法往往并不关心.如果出现了第四种算法,工程师需要知道,而客户一样不需要知道.

       二,基于GIS组件的二次开发,出了已有地图鼠标操作外,我们往往需要添加自己的鼠标行为,此时一个良好的设计是我们扩展鼠标的状态,扩展鼠标的行为,让后将他们对应起来.同时不影响原来的鼠标行为.基因

       请仔细比较下面代码片断的细节差异.

　　 1.策略模式

　　  2.状态模式

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

public override void Action(string mouseState){  
 base.Action(mouseState);  
 if (mouseState == "Pan")  
 Pan();  
 else if (mouseState == "Select")  
 Select();  
 }  
 public void Pan() {/\*平移地图\*/}  
 public void Select(){/\*选择图形\*/}

[复制代码](javascript:void(0);)

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

public void Topology(){  
 //通过拓Topology式计算A获取结果B  
 }  
 public void Gene(){  
 //通过Gene方式计算A获取结果B  
 }  
 public void Simulation(){  
 //通过Simulation方式计算A获取结果B  
 }  
 public void Strategy()  
 {  
 string flag = "";  
 if (flag == "Topology")  
 Topology();  
 else if (flag == "Gene")  
 Gene();  
 else if (flag == "Simulation")  
 Simulation();  
 //work complated  
 }

[复制代码](javascript:void(0);)