**[状态模式(State pattern)](http://www.cnblogs.com/lds85930/articles/643014.html)**

**一、引子**   
  
　　状态模式自身结构非常简单——前面刚刚介绍了几个结构比较简单的设计模式，和他们一样，状态模式在具体实现上留下了可变换的余地。我前面已经介绍过它的孪生兄妹策略模式了，大家可以两者比较着阅读。本文将会讨论两者的区别。  
  
　　**二、定义与结构**  
  
　　GOF《设计模式》中给状态模式下的定义为：允许一个对象在其内部状态改变时改变它的行为。这个对象看起来似乎修改了它的类。看起来，状态模式好像是神通广大——居然能够“修改自身的类”！  
  
　　能够让程序根据不同的外部情况来做出不同的响应，最直接的方法就是在程序中将这些可能发生的外部情况全部考虑到，使用if else 语句来进行代码响应选择。但是这种方法对于复杂一点的状态判断，就会显得杂乱无章，容易产生错误；而且增加一个新的状态将会带来大量的修改。这个时候“能够修改自身”的状态模式的引入也许是个不错的主意。  
  
　　状态模式可以有效的替换充满在程序中的if else语句：将不同条件下的行为封装在一个类里面，再给这些类一个统一的父类来约束他们。来看一下状态模式的角色组成吧：  
  
　　1) 使用环境（Context）角色：客户程序是通过它来满足自己的需求。它定义了客户程序需要的接口；并且维护一个具体状态角色的实例，这个实例来决定当前的状态。   
  
　　2) 状态（State）角色：定义一个接口以封装与使用环境角色的一个特定状态相关的行为。  
  
　　3) 具体状态（Concrete State）角色：实现状态角色定义的接口。  
  
　　类图如下，结构非常简单也与策略模式非常相似。

|  |
| --- |
| http://dev.yesky.com/imagelist/05/10/u104af948942.jpg    http://dev.yesky.com/imagelist/05/10/wt1710i261bf.jpg |

　　**三、实现**  
  
　　由于状态模式结构非常简单，所以在这里罗列一些反映状态模式实现结构的代码没有什么太大的作用。如果你有兴趣的话可以按照上面类图来编写一下。  
  
　　在引子中已经提到，状态模式在具体实现上存在不同的方案。因此这里重点就这些不同的实现方式进行介绍和讨论。  
  
　　首先，实现时是否将状态角色、具体状态角色暴露给客户程序？按照GOF的建议是不希望将状态角色暴露给客户程序的，与客户程序打交道的仅仅是使用环境角色，客户是不知道系统是怎么实现的，更不关心什么有几个具体状态。但是当使用环境角色中的初始状态紧紧依赖于客户程序时，适乎暴露是在所难免的——这就与策略模式异常相似了！  
  
　　具体状态角色中的行为一般是与使用环境角色密切相关的。因此这里便有了一个小细节：我们把使用环境角色作为参数传递进入具体状态角色后，是在具体状态角色中来实现状态响应行为；还是仅仅调用在使用环境角色中已经实现了的方法？由于这些行为往往与使用环境角色相关，所以按照《重构》一书的“指导”——后一种实现方法是比较地道的。  
  
　　从定义可知，状态模式是要应对状态转换的。那么状态的转换在哪里定义呢？你可以选择在使用环境角色的代码中来表现出来，当然这便意味着状态转变的规则就固定下来了。GOF还给出了另外一种稍微灵活一点的实现方式：在每一个具体状态角色中来指定后续状态以及何时进行转换。  
  
　　其实在java强大的反射机制的支持下，我们还可以将状态的转换做的更加灵活——我们可以将状态转换的规则写在.xml等等的配置文件里面甚至是数据库中，我们姑且叫做状态转换表。进行转换前，根据状态转换表来读取下一个状态，然后利用反射获得具体的状态对象……。看起来很不错的样子，只是效率可能低一些，在企业应用中这应该不是最重要的。   
  
　　状态模式已经被我们想象着“实现”了一番。那么状态模式的引入会给我们的程序带来哪些优势呢？前面我们已经说过：状态模式的引入免除了代码中复杂而庸长的逻辑判断语句。而且具体状态角色将具体状态和它对应的行为封装了起来，这使得增加一种新的状态变得简单一些。而且如果设计合理得话，具体状态角色可以被重用（和策略模式一样，可以考虑使用享元模式来实现）。  
  
　　使用状态模式也会带来一些问题。每个状态对应一个具体的状态类，使得整体分散，逻辑不太清晰。当然对于一个状态非常多的系统，状态模式带来的优点还是大于它的缺点的。  
  
　　由上面的分析就可以很明确的知道什么时候该使用状态模式了。下面是GOF在《设计模式》中给出的状态模式的适用情况：   
  
　　1) 一个对象的行为取决于它的状态, 并且它必须在运行时刻根据状态改变它的行为。  
  
　　2) 一个操作中含有庞大的多分支的条件语句，且这些分支依赖于该对象的状态。  
  
　　**四、状态VS策略**  
  
　　仔细对比状态模式和策略模式，难免会产生疑问：这两个明明是一个东西嘛！下面我们就来分析下两者区别。  
  
　　首先我要声明，在实际应用中只要能够使得你的代码灵活漂亮起来，何必计较这些方方面面的差别呢？  
  
　　Brandon Goldfedder在《模式的乐趣》里是怎么说的：“strategy模式在结构上与state模式非常相似，但是在概念上，他们的目的差异非常大。区分这两个模式的关键是看行为是由状态驱动还是由一组算法驱动，这条规则似乎有点随意，但是在判断时还是需要考虑它。通常，State模式的“状态”是在对象内部的，Strategy模式的“策略”可以在对象外部，不过这也不是一条严格、可靠的规则。”  
  
　　我很同意Brandon Goldfedder的观点。这两个模式的划分，就在于使用的目的是不同的——策略模式用来处理算法变化，而状态模式则是处理状态变化（好玄乎阿）。  
  
　　策略模式中，算法是否变化完全是由客户程序开决定的，而且往往一次只能选择一种算法，不存在算法中途发生变化的情况。从《深入浅出策略模式》中的例子可以很好的看出。  
  
　　而状态模式如定义中所言，在它的生命周期中存在着状态的转变和行为得更改，而且状态变化是一个线性的整体；对于客户程序来言，这种状态变化往往是透明的。

http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System;   
http://www.cnblogs.com/Images/OutliningIndicators/None.gif   
http://www.cnblogs.com/Images/OutliningIndicators/None.gifnamespace DesignPattern.State   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gif{   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*//// <summary>   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    /// 抽象状态 状态接口   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    /// </summary>   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    public interface ITcpState   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        void Open();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        void Close();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        void Acknowledge();   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*//// <summary>   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    /// 此类相当于Context   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    /// </summary>   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    public class TcpConnection   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        private ITcpState state;   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public ITcpState State   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif            set{this.state=value;}   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Open()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            this.State=new TcpEstablished();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            this.state.Open();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            this.State=new TcpListen();   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Close()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            this.state.Close();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            this.State=new TcpClosed();   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Acknowledge()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            this.state.Acknowledge();   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    public class TcpEstablished : ITcpState   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif           
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Open()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Console.WriteLine("Have opened,can not open again!");   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Close()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Console.WriteLine("Closing");   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Acknowledge()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Console.WriteLine("TcpEstablished!");   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    public class TcpListen : ITcpState   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Open()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Console.WriteLine("Have opened,can not open again!");   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Close()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Console.WriteLine("Closing");   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Acknowledge()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Console.WriteLine("TcpListen!");   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    public class TcpClosed : ITcpState   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Open()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Console.WriteLine("Openning");   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Close()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Console.WriteLine("Have closed,can not close again!");   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Acknowledge()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Console.WriteLine("TcpClosed!");   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    public class Client   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public static void Main()   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            TcpConnection tcpcon=new TcpConnection();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            tcpcon.Open();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            tcpcon.Acknowledge();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            tcpcon.Open();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            tcpcon.Close();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            tcpcon.Acknowledge();   
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            tcpcon.Close();   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }   
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}   
http://www.cnblogs.com/Images/OutliningIndicators/None.gif

　　**五、总结**  
  
　　比较笼统地介绍了下状态模式，并将它和非常相近的策略模式进行了比较。欢迎大家学习指正。