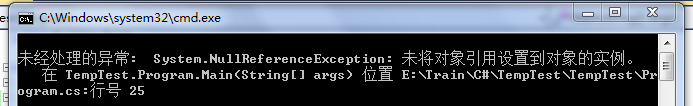
[**.NET中的设计模式——利用空模式构造安全代码**](http://www.cnblogs.com/niyw/archive/2010/09/04/1818082.html)

题记：

　　《.NET中的设计模式》系列随笔停下有一段时间了，一则总结个东西不容易，另一则，不想写相同的内容(如果朋友们没有在我的随笔中看到新东西，我认为是浪费大家的时间，也是一种失败)。

　　今天开题之前先让大家见一个老朋友，相信没有一个人会不认识它:-)大家是不是觉得很眼熟啊。程序中产生一个错误的原因很简单，解决方法也很简单，我也相信写过两年代码的人只要稍微细心一点就很少遇到这个问题。抛去原因，相信很多人是这么解决此类问题的：

**if(obj!=null){}**

然而一个项目中，各种类似变量成千上万，如果每个地方都这么判断，不仅浪费体力，而且让代码丑陋，甚至因为遗漏而出现低级Bug。相信这都不是大家愿意看到的。那么有没有好的解决办法呢？答案就是**空模式**！

　　空模式不属于《GoF23种设计模式》，最早接触它，是在一本Java中的设计模式书中，用Java的人应该都知道(如果用Java几年还不知道就无语了...)。然而既然是设计模式，.NET中一样可以用（工具无优劣，挫人论高低）。

**空模式，其目的是为了保证代码的安全和优雅，避免丑陋的代码。**

　　还回到已开始的错误场景，其实我们最希望的解决方式不是解决Bug,而是Bug不存在，那么空模式怎么解决此类问题呢？先看下面的示例代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | //为业务对象定义一个不做任何事情的基类,  public class BusinessBase {      //为该类型定义一个Empty静态成员      public static BusinessBase Empty = new BusinessBase();      public virtual void DoSomething() {          Console.WriteLine("This is a null object,it can't do anything.");      }  }  //在扩展类型做具体的事情  public class BusinessA : BusinessBase {      public override void DoSomething() {          Console.WriteLine("Niyw is doing something.");      }  } |

通常我们的代码会这么写

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | static void TestNormal() {      Console.WriteLine("测试传统方法.");      BusinessBase bb = GetBusinessNormal(true);      if(bb!=null)          bb.DoSomething();      else          Console.WriteLine("出错啦.");      bb = GetBusinessNormal(false);      if (bb != null)          bb.DoSomething();      else          Console.WriteLine("出错啦.");  }  static BusinessBase GetBusinessNormal(bool flag) {      try {          if (flag)              return new BusinessA();          throw new Exception("出错啦.");      }      catch {          return null;      }  } |

使用空模式时，代码如下:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | static void TestEmptyPattern() {      Console.WriteLine("测试空模式.");      BusinessBase bb = GetBusinessEmptyPattern(false);      bb.DoSomething();      bb = GetBusinessEmptyPattern(true);      bb.DoSomething();  }  static BusinessBase GetBusinessEmptyPattern(bool flag) {      try {          if(flag)              return new BusinessA();          throw new Exception("出错啦.");      }      catch {          return BusinessA.Empty;      }  } |

通过比较上面的两段代码,一般情况下**四行的业务代码**，为了容错，**至少要写6行的容错代码**，而**使用空模式时**，**不用增加任何一行容错代码**。是不是发现用了空模式后代码更加优雅简单,并且安全性提高了很多.

小结：空模式是一种很简单的模式，甚至不算是模式。然而它代表了一种很好的编码习惯，在具体项目开发过程中，如果用好类似的习惯，相信开发效率会更高，个人技能也会更高。

分类: [.NET与设计模式](http://www.cnblogs.com/niyw/category/255528.html)