**面向对象软件开发的十大原则**

对于开发人员来说，时时回过头来检查一下应用程序是非常有用的。考虑一下这个应用程序是否达到了预期目的?是否遵循了最佳方案?要想实现目的是否还存在更好的方法?温故而知新，也许你会从回顾中发现新的思路。

本文就提出10种顶尖的面向对象的编程技术来帮助你对应用程序进行评价，希望你使用这10个顶尖技术从面向对象的编程中获得最大利益。

**1. 重新审视对象的声明操作**

如何声明对象变量会给对象的使用和性能带来重大的影响。你应该使用这个语法：

Private m\_oCust As Ccustomer

注意一定要使用正确的范围。在大多数情况下，你希望对象变量是私有的。如果只是在一个特定的程序中需要这个对象，就可以用Dim来声明它是属于那个程序私有的。如果在整个类或表单模块中都需要它的话，就应该用Private 在那个类或表单的顶部声明它。按照习惯，在模块层声明的变量要加一个前缀 "m\_"，以表示用于模块层或成员变量。如果需要从类外部使用对象变量，就不要将变量设成Public，宁可定义一个Property Get程序来使用对象变量，这样就防止了其它代码不用你不希望的方法去使用变量。

在声明中，总是使用类名称，而不要使用 "As Object"，因为使用 "As Object" 会使对象变量成为后期捆绑(late-bound)。这就意味着在运行时间之前，Visual Basic不知道你将使用的对象类型，因此它不能给你提供自动列表成分(对象属性和方法的下拉列表)。更重要的是，在运行时会对早期捆绑(early-bound)对象造成明显的性能冲击。

最后，在声明中不要使用"New"这个关键字。使用关键字"New"时，将在引用它的任何时间自动创建对象，这对性能会造成轻微的影响，因为每个对象的属性和方法调用都必须进行校验以确定对象是否被创建了。使用"New"这个关键字还会给调试带来噩梦，因为在对象被破坏之后，如果你不经意地再次引用它的话，应用程序会自动地重新创建对象。当你准备移植到.NET时，理解这一点更加重要，因为在那里关键字"New"有一个不同的意义。在.NET中，关键字"New"允许你在声明时创建一个对象，比如下面这条语句将创建一个新的Customer对象：

Private mo\_Cust as New CCustomer()

**2. 重视对象清除行为**

当不再需要对象变量时，一定要释放它们，然后VB就会破坏与变量相联系的对象，从而释放应用程序中的内存。使用下面的代码释放一个对象：

Set x = Nothing

**3. 封装的重大作用**

Encapsulate(封装)从字面上讲就是放入封套或"放在一个壳子中"。从面向对象编程 (OOP) 的角度来看，这意味着将所有与一个对象相关的数据和处理都放在它的类中。

封装有时候指的是信息隐藏，因为你将对象的数据藏在了类的内部。如果其它的类需要使用数据，可以使用属性程序(.NET 中的属性声明)来暴露这些数据。这个行为为类提供了控制， 它控制用户可以如何观看或刷新数据，并且防止其它对象误用数据。

封装使得对象之间的交互作用变得简单化。一个对象不用知道另一个对象的全部数据或者其数据是如何管理的，就能够使用这个对象。例如，一个Customer对象可能有名字和地址属性以及电话号码、信用历史和其它相关域。如果一个Invoice 对象想要使用一个 Customer 对象，它可以为发票请求名字和地址，而不需要知道Customer是从哪里获取那个数据的，或者其它数据元素是什么。

对于如何为可扩展的标记语言(XML)将文档对象模型 (DOM) 封装在一个包装类中的解决方案，请参考此文，这使你能够更容易地与一个XML文件进行交互作用，而不需要知道DOM的细节。

**4. 状态监视问题**

良好的基于组件的开发技术会频繁地要求无状态的对象。无状态对象没有保持的属性，只有方法。你可以访问一个无状态的组件，执行方法，然后不需要获取方法调用间的任何数据就可以完成处理。

有一些OOP方面的文章提出没有属性的对象就不是对象。 但是我们现在忽略这个语义上的问题，你可以创建有方法而没有属性的类。当你从事COM+或Web开发时，经常会需要这样做。一个ASP页面可以在一个组件上执行一个方法来完成一个特殊的处理，然后再释放组件，不用为管理对象状态而费心。

**5. 传递参数的可扩展性**

参数提供了类的方法之间传递信息的途径。你需要用参数向方法传递任何必要的数据，特别是当你需要无状态对象的时候。

例如，无状态对象Customer 的PlaceOrder方法要求把客户的信息作为参数传递给它，然后PlaceOrder方法就用这个信息来完成必要的处理。

当定义方法的参数时，一定要使它们可以扩展。例如，下面这行代码是不可扩展的：

Public Function PlaceOrder(sLastName as String, sFirstName as String, sAddress as String)

要想调用这个方法你必须传递这3个参数。但是如果你以后决定在定单上还需要电话号码，就必须修改函数签名，这就破坏了兼容性以及每个调用此方法的代码段。为了防止这个问题的发生，一个更好的解决方法是：在一个容器中传递参数。你可以使用记录集、变量数组或 XML字符串来传递更普通的参数。

当转移到.NET时，这个技巧就不是十分必要了，因为在.NET中你可以进行函数装载，这样就可以具有两套不同参数的同一个函数：

Public Function PlaceOrder(sLastName \_

as String, sFirstName as String, sAddress as String)

Public Function PlaceOrder(sLastName \_

as String, sFirstName as String, \_

sAddress as String, sPhone as String)

**6. 现在就使用XML**

XML提供了一个管理状态和在应用程序的组件之间传递它的很好方法。 你可以用任何喜欢的格式来定义XML，然后可以从XML字符串插入、更新、删除或回顾任何信息。

XML最好的一点是它的跨平台性并拥有独立的销售商。XML既不是Microsoft的技术也不是Sun的技术，是万维网联盟W3C控制着这个标准，具体细节请参见原文。

XML成为了.NET中的数据存取标准，所以现在就使用会使你一路领先。

**7. 定义灵活的界面**

界面(Interfaces)提供了一个在运行时间插入组件的很好途径。你可以定义一个界面，建立一个执行它的类，然后就可以在任何时间，很容易地用任何执行同一界面的其它类来代替这个类。

例如，你可以开发一个邮件列表的标签打印组件，并且定义这个组件所要求的标准界面。在这个例子中，那个界面包括名字和地址信息。然后，任何执行这个界面的类都可以使用这个组件，而不需要对其进行任何修改。一个执行标签打印界面的客户类可以使用这个组件来打印客户邮件标签，一个执行标签打印界面的职员类可以使用这个组件来打印薪水标签等。

**8. 继承功能**

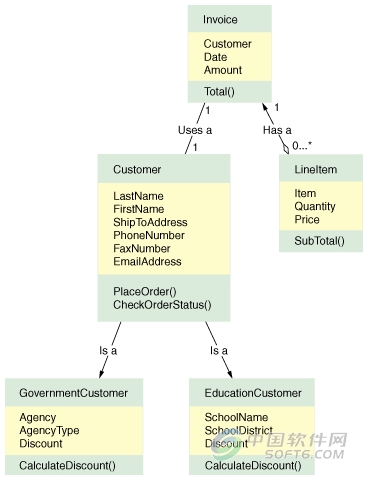
继承是OOP的一个关键原则，它允许你定义一个对象的基础功能，然后将那个功能应用于对象的特定子类中。

例如，你可以为一个客户(Customer)对象定义基础功能，可以恢复和保存数据以及计算一个折扣。然后就可以定义一个政府客户对象，让它继承客户对象的所有功能，但是屏蔽掉折扣计算，因为给政府客户的折扣更高。

在VB.NET中，你很快就能实现这些。

**9. 按照模型进行处理**

开发一个描述交易的域模型对于理解应用程序的目的以及它应该解决的问题是非常有用的。 请参见下图：



但是有一个常见的错误是取得域的模型并试图为它编码。相反，你应该融合域模型到一个执行模型中，这个执行模型定义了如何建立类的方法。融合过程中，需要考虑许多因素，请参阅建立对象模型。

**10. 了解OOP，使用OOP**

所有的.NET都是面向对象的。现在你对OOP了解得越多，理解得越好，将来转移到.NET就越容易。在所有项目中都要准备好使用类模型。如果你不了解OOP，或者虽然你了解它但是并不真正理解它，可以采用一个OOP类。OOP技术很快就会成为你的第二个自我，你会发现想记起原来是怎么编码的都很困难。