清松电脑系列丛书

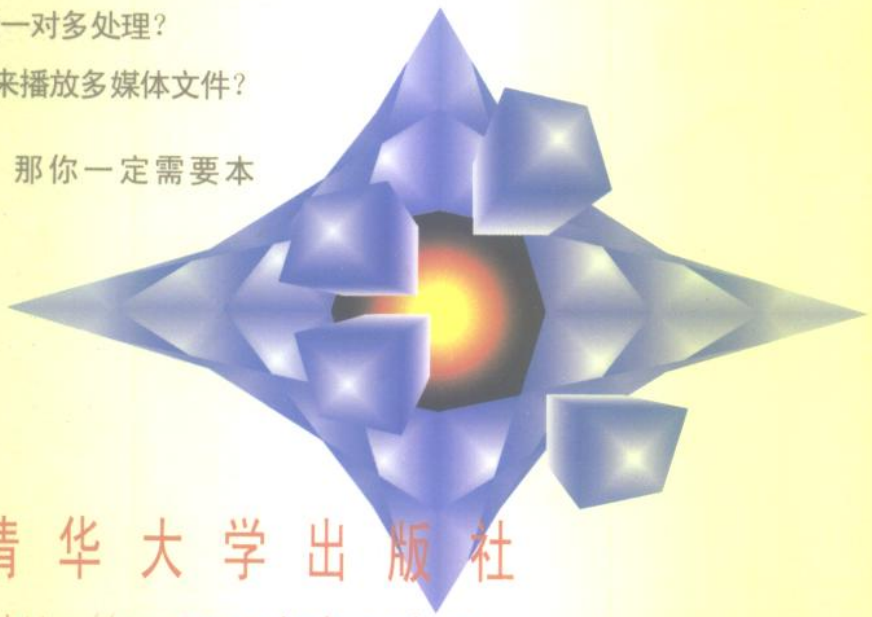
深入 Visual FoxPro 6.0 面向对象程序设计

- 在Visual FoxPro 6.0环境下如何用面向对象设计方法编程?
- 如何从过程设计方法转向面向对象的设计方法?
- 如何设计出让用户感到亲切、易用的界面?
- 如何在Visual FoxPro 6.0中调用EXCEL电子表格?
- 如何利用GRID控件实现表的一对多处理?
- 如何用Multimedia MIC控件来播放多媒体文件?

...

当你不能解决这些问题时，那你一定需要本书!

沈惠璋
马英骐 编著
吴继泽



清华大学出版社
http://www.tup.tsinghua.edu.cn

深入 Visual FoxPro 6.0
面向对象程序设计

沈惠璋　马英骐　吴继泽　编著

清华大学出版社

(京) 新登字 158 号

内 容 简 介

本书是一本介绍在 Visual FoxPro 6.0 环境下如何使用面向对象编程技术，开发应用程序的书籍。全书由三部分组成：面向对象基础、表单布局设计和实例分析。书中不仅介绍面向对象程序设计的概念，还提供了 32 个面向不同应用的实例，使读者能够全面地掌握面向对象编程技术。

书　　名：深入 Visual FoxPro 6.0 面向对象程序设计
作　　者：沈惠璋　马英骐　吴继泽　编著
出 版 者：清华大学出版社（北京 清华大学校内，邮政编码：100084）
因特网址：http://www.tup.tsinghua.edu.cn
责任编辑：童隆裴
印 刷 者：北京市清华园胶印厂
发 行 者：新华书店总店北京科技发行所
开　　本：787×1092 1/16　印张：19.25　字数：453 千字
版　　次：1999 年 10 月第 1 版　　2000 年 1 月第 2 次印刷
书　　号：ISBN 7-302-03746-9/TP·2096
印　　数：5001—10000
定　　价：29.00 元

前　言

FoxPro是我国应用最广泛、用户群最大的微机数据库系统，有成千上万的管理软件是使用FoxPro作为开发平台的。这些软件都在各自的环境中起着重要的作用，创造了巨大的经济效益。

目前，FoxPro的最新版本是Visual FoxPro 6.0中文版。新的Visual FoxPro 6.0不仅提供了更好的图形操作界面，更重要的是还提供了新的程序设计方法——面向对象的程序设计。因此，为了让更多的人掌握这个新的编程工具，作者编写了此书。

本书不仅详细介绍了面向对象程序设计的原理，还提供了大量的面向对象编程的范例，来帮助读者尽快地掌握FoxPro环境下面向对象的程序设计。对一个问题往往可以有多种解决方法，作者希望本书能起到抛砖引玉的作用：不仅提供给读者几种典型的应用实例，更重要的是启发读者进入一种更高的思维境界。

由于面向对象编程是一个内容较多、涉及面很广的话题，特别是作者水平有限，虽然本书力求避免错误，但不当之处还是在所难免，欢迎广大读者、朋友批评指正。或者如果有更好的想法，可以与作者联络，共同提高。

本书由张磊研究室统一策划，沈惠璋、张治文、吴继泽编写。其中沈惠璋完成了本书的第1～5章的编写；马英骐完成了本书第6、7章的编写；吴继泽完成了本书第8章的编写。

张磊研究室对本书提供技术支持，有关书中的问题可访问：www.zhanglei.com，或者发送电子邮件到：supports@zhanglei.com。

作　者

1999/6/16

目　　录

第 1 章　面向对象程序设计概述

本章介绍程序员在使用 Visual FoxPro 提供的面向对象程序设计方法之前需知的基本原理。当了解这些原理并掌握它们提供的巨大益处之后，就会明白 Visual FoxPro 和所有其他重要开发平台向面向对象转移的原因了。

面向对象的程序设计方法与编程技术不同于标准的过程化程序设计。程序设计人员在进行面向对象的程序设计时，不再是单纯地从代码的第一行一直编到最后一行，而是考虑如何创建对象、利用对象来简化程序设计和提供代码的可重用性。对象可以是应用程序的一个自包含组件：一方面具有私有的功能，供自己使用；另一方面又提供公用的功能，供其他用户使用。

1.1　什么是面向对象程序设计

面向对象使程序员的观点从程序设计语言如何工作，转向注重于执行程序设计功能的对象模型，而不是着重于每个程序代码如何与程序的其他部分和系统的交互作用上。面向对象程序着重于建立能够模拟需要解决的现实世界问题的对象。

在面向对象的程序设计中，对象是组成软件的基本元件。每个对象可看成是一个封装起来的独立元件，在程序里担负某个特定的任务。因此，在设计程序时，不必知道对象的内部细节，只要在需要时，对对象的属性进行设定和控制即可。图 1-1 示范了对象和应用程序的关系。

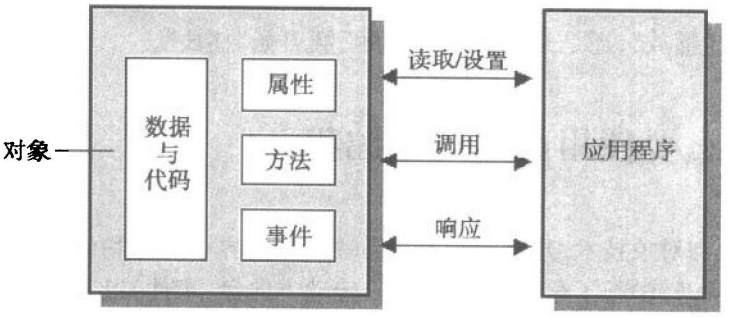


图 1-1

例如，不妨把一台录音机看成一个对象。要使用录音机，只要知道操作方法就行了。当要录音或播放声音时，只需将录音带放进录音机内，按下有关的功能操作键，录音机就会知

道是播放还是录制。而普通的用户根本就不需要去了解其内部的运转方式,更不需要知道其内部的电路板是如何焊接及解码芯片是如何运算的。

不过,程序设计者在使用对象时,虽然无需知道对象的内容,但是必须要了解对象对外所提供的属性、方法和事件,就好比用户必须知道录音机面板上按键的作用和操作程序,才能够享用它。也就是说,要先知道怎么用,才能享受到它所提供的功能或服务。

属性

所谓属性就是对象表现出来的特征、状态或行为。就像录音机有型号、尺寸、颜色、出厂日期等特征一样。不同的对象可以拥有各种相同或不同的属性,其中有些属性是只读并且无法改变的,而有些则可以通过设定来改变。这就好像录音机的出厂日期、型号等属性是无法改变的,但操作面板上所显示的时间则可通过设定来改变。

方法

方法是用来处理或操纵对象的途径。对象通常会提供一些方法,以便应用程序可以使用对象所提供的服务。例如,录音机提供了"播放"、"停止"、"暂停"、"快进"、"快退"等操作按钮,而这些按钮其实就相当于录音机提供的方法。用户只要按下这些按钮,就可以得到录音机所提供的播放、停止、暂停播放、快进等服务。

同理,只要通过对象对外提供的方法,就可以得到它的服务,根本不需要知道对象内部的实际运作方式。所以,用面向对象的程序设计方法来开发应用软件,不仅可以提高效率,更重要的是可以保证软件的质量。因为,用户仅需知道怎样调用对象提供的服务(功能)就可以了,而不必从头开始设计和编写应用软件中需要的所有功能。

事件与事件响应

事件就是对象所碰到的情况,例如有录音带被放进录音机,或者是录音带播完,这样的情况就是一个"事件"。当一个事件发生后,就需要对该事件进行响应。也就说,可以事先指定当事件发生时,对象要做出什么样的反应。例如,当"录音带放进录音机"的事件发生时,可以指定是直接"播放",还是先"快进"一小段后再开始"播放"。

1.2 为什么要使用面向对象编程

通过使用面向对象技术,开发者能建立反映真实世界中的事物的应用程序。以汽车为例,用户买了一辆车,学习了车的性能(属性)、转动方向盘(事件)以及将拐向所希望的方向(方法)。用户可以理解车的这些特点,但并不知道车的内部构造以及为什么车会有这样的功能。

同样的例子,来看一下代码的重复使用性。制造商制造了一辆基本的模型车。如果希望汽车有额外的功能,如空调、动力自动驾驶、遮阳装置等等,则可以增加这些功能。通过增加这些额外的功能,就创建了这个基本模型车的一个子类。基本模型车是售车商用来给顾

客作演示的。从演示中,顾客可以决定他们是否喜欢这辆车的性能。通过这个简单的运作,开发商既可以销售这种基本模型车,也可以对车的现有外观进行小的改动,修改车的特性,而不用对整辆车重新设计。

1.3 充分理解事件和状态

对象行为和事件本身并不难理解。简单讲,对象行为是指对象如何操作及如何响应特殊操作,对象行为是用事件和状态来定义的。

1.3.1 什么是状态

在应用 Visual FoxPro 开发之前,需要对状态这一概念有个深刻的理解。状态之所以重要,有以下几个原因:

- 状态是定义一个事件的关键
- 确定对象如何行为的关键是事件
- 对象行为对于 Visual FoxPro 面向对象的结构非常重要

对象状态最基本的问题是对象是否存在。如果存在,它就会与决定其存在的其他对象发生关系。一个不与世界上任何事物相关而孤立存在的对象是不可想象的。实际上,逻辑学禁止设想一个没有任何关系的对象。因为只要设想一个对象,就必定要建立一个同设想相关的关系。所以,更重要的问题是,对象怎样与自身相关、怎样彼此相关以及怎样同世界上其他事物相关?

简单定义,状态是一个 Visual FoxPro 对象同其他对象及系统的组合关系。因此,对象的关系的总和便决定了它的状态。虽然这个定义很粗略、很简单,但它却是理解概念的一个基础。另外,这个定义没有完全表述真实性,而是表述了对它的理解。

根据定义,Visual FoxPro 事件同时间有关。因为当对象状态改变时,它们才出现。由于对象状态是其他所有对象关系的总和,就有允许制定对象变化进入有意义事件的时间。否则,如果不考虑时间,就只能知道对象的全部状态,即对象存在期间所有关系的总和。

现在必须清楚地知道,状态不是对象,状态是关系的映射,因此对象是概念上的事件模型。

1.3.2 什么是事件

无论在科学上,还是在生活中,事件是普遍存在的。例如,两岁的女儿突然哭着要把正在工作的妈妈叫回来;海底火山爆发,诞生了一块新的陆地;单击鼠标按钮等等。所有这些都是事件的例子。然而只有后者才是 Visual FoxPro 软件开发人员所关心的。

一个事件可以简单定义为对象状态或条件的变化。状态变化可以是两个 Visual FoxPro 对象间的关系变化,也可能会像超级明星那样显著。只有当状态变化重要到足以引起注意

时,程序员才把它们当作事件来看。

事件起因于状态的变化

不分析状态——这个现在新出现的却是很基本的程序概念——要想理解事件是不可能的。所有的事件都起因于引起状态变化的进程。正是这些状态的变化,使人们能够理解事件。例如,在炎热的夏天,要使室内达到恒温,空调同外部有两种重要的关系:首先,它与制冷机有关,可通过控制一个开关来决定是否制冷;其次,它又与房间的空气温度有关,可监控温度的变化。

使用前面事件的定义可以知道,只要空气温度有部分变化,哪怕只有 1°,由于空调同变化的空气温度相联系,那个变化就是一次事件。然而并不需要制冷机在温度每变化 1°左右时都打开或关闭。例如,将室温设定到 23°时,除非温度高于这一设定值,制冷机才开始工作,否则制冷机对于温度的变化不作任何控制。事件随时发生,而只有当监视到满足编程规则的条件时,制冷机才会起作用。在这个例子中,规则"在 23°时改变"被称作一个触发规则。

现在把事件当作在对象状态中的显著变化,就需要知道更多的有关状态及状态改变方式的知识。

1.3.3 触发器

触发器是 Visual FoxPro 事件的功能显示,它是有效地实现 Visual FoxPro 最强大的面向对象特征事件驱动编程的基础,触发器可用来链接事件和进程。当开发 Visual FoxPro 应用程序时,对这个基本关系的理解是简单、有效地使用事件的关键。事件处理或事件代码是程序员设计特殊进程时用的术语,它用来决定是由一个唯一事件触发响应及响应效果,还是响应的触发。Visual FoxPro 提供了一套丰富的内嵌触发规则,这些规则正是非模态操作的基础。

触发器源于因果关系。因果关系是生活中非常熟悉的一种关系。几乎从诞生之日起,我们就认识了那些带来生命中值得留意的事件的因果关系。我们学习的重要一课是,事件触发的反应可以导致变化结果由不重要到重要。

解发器受到触发规则的控制,这是一个因果关系链。在一个事件生成触发器之前,需要指定一系列的条件。例如,单击鼠标按钮是一个事件,但为了使事件产生一个响应,光标必须预先在一个规定的位置上,如光标在复选框中,否则单击按钮也不会产生响应。如果光标不是在程序规定的位置上来产生响应,那么该事件本身就没有什么程序设计意义了。

下面的范例显示了触发器规则是怎样决定事件响应的。在图 1-2 所示位置上单击按钮,则会出现如图 1-3 所示的提示信息。而在图 1-4 所示位置上单击,则没有任何事情发生。

通过上面的操作可以看出,由于对按钮"Command1"设置了单击触发,所以在该按钮上单击,会触发用户定义的事件处理程序,从而显示新的对话框。而对表单中并没有定义单击的事件处理程序,则没有任何情况发生,系统也就忽略该事件。

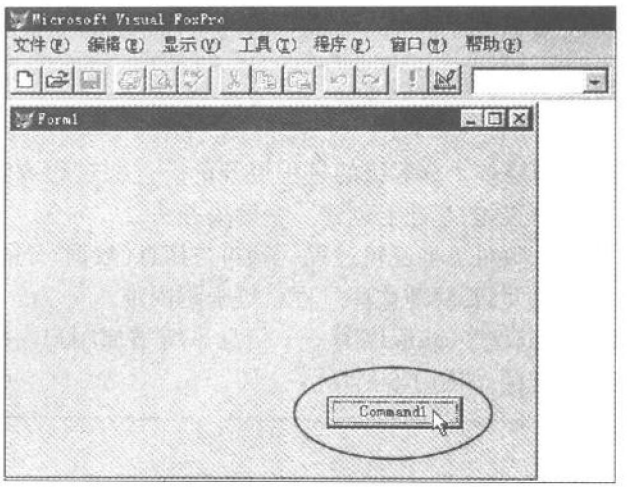


图 1-2

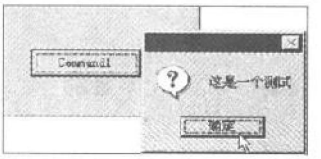


图 1-3

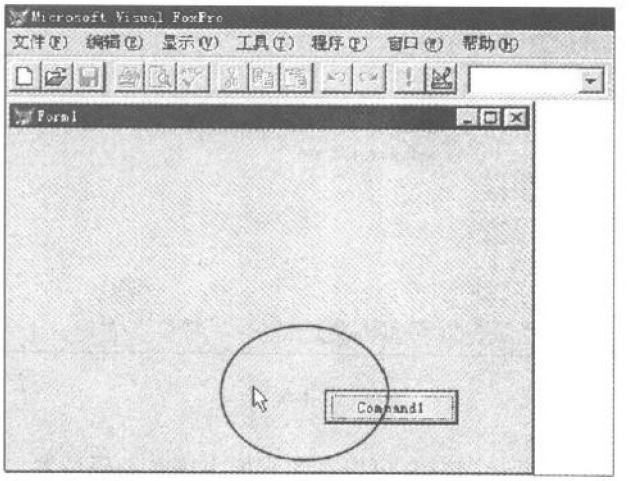


图 1-4

1.4 由过程转向面向对象

为什么由过程转向面向对象？当用户感觉已经很完备时，为什么 Microsoft 还要开发新的编程方法呢？答案在于开发应用程序的可复用性和应用程序的性能。为了说明采用面向

对象编程程序设计方法的优点，先来看看下面的实例。

1.4.1 结构化的程序

结构化的设计方法是围绕多个分等级的过程展开的。这些过程或者按顺序执行，或者以一个问题为中心，并在此问题的基础上创建一个解决办法。

传统的结构化设计方法倾向于将逻辑过程(函数)与信息(数据)分隔开来。随着过程和函数的个数和复杂程度的增大，要管理这些过程就越来越困难。

例如，编写出下面的示范程序，该程序为一个信息系统增加用户。执行该程序时，它会显示一个窗口供用户输入信息，如图 1-5 所示。

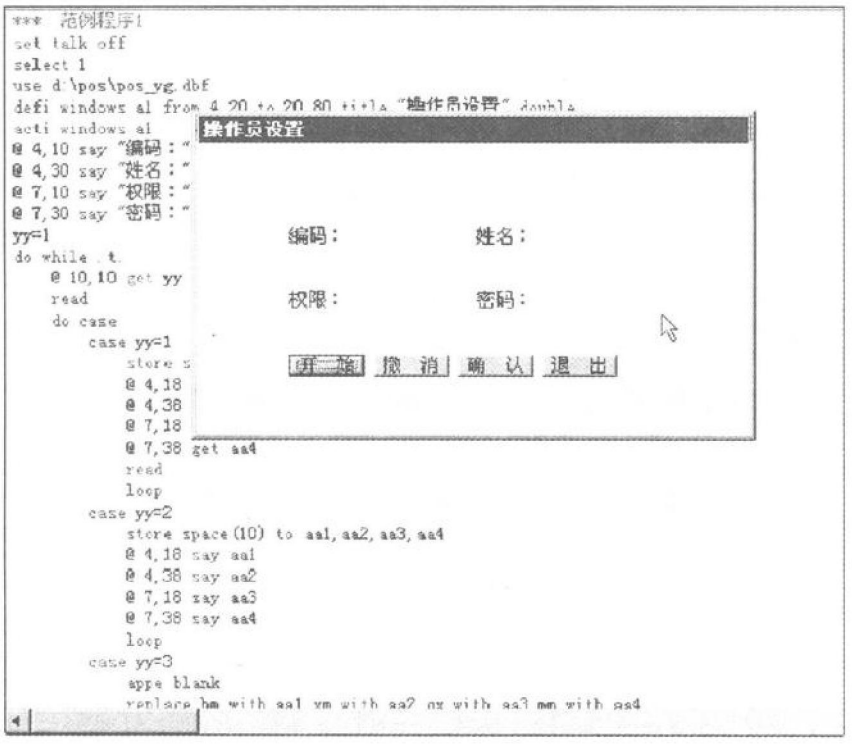


图 1-5

在该工作窗口中单击“开始”按钮，可以看到光标出现在编码位置。这时，用户可以输入需要的内容了。一切都很正常，但有一点不方便的是，如果需要临时中断输入过程，就必须把所有的字段全部输入后，才能将控制权交给四个按钮，这时才能够退出程序。

下面是实现该项功能的源程序：

```
* * 范例程序 1
set talk off
select 1
use d: \ pos \ pos_yg.dbf
```

```
defi windows a1 from 4,20 to 20,80 title "操作员设置" double
acti windows a1
@ 4,10 say "编码:"
@ 4,30 say "姓名:"
@ 7,10 say "权限:"
@ 7,30 say "密码:"
yy = 1
do while .t.
    @ 10,10 get yy func "*H 开 始;撤 消;确 认;退 出"
    read
    do case
        case yy = 1            * * 开始
          store space(10) to aa1,aa2,aa3,aa4
          @ 4,18 get aa1 valid jc_1( )
          @ 4,38 get aa2
          @ 7,18 get aa3
          @ 7,38 get aa4
          read
          loop
        case yy = 2            * * 撤消
          store space(10) to aa1,aa2,aa3,aa4
          @ 4,18 say aa1
          @ 4,38 say aa2
          @ 7,18 say aa3
          @ 7,38 say aa4
          loop
        case yy = 3            * * 确认
            appe blank
            replace bm with aa1,xm with aa2,qx with aa3,mm with aa4
            loop
          case yy = 4            * * 退出
            exit
    endcase
enddo
use
rele windows a1
return
****************************************
* * 检查编码是否重复
function jc_1
sele 1
locat for bm = trim(aa1)
if .not.eof()
    wait windows "编码重复,请重新输入!"
    return .f.
endif
return .t.
```

对于上例中的程序,可以画出如图 1-6 所示的流程图。同时,也可以看到程序完全采用由上至下的方法编写了全部的代码。