Visual FoxPro 6.0

实用管理系统开发

实例剖析

高国宏 编著

中国民航出版社

Visual FoxPro 6.0 实用管理系统开发实例剖析

高国宏　编著

图书在版编目（CIP）数据

Visual FoxPro 6.0 实用管理系统开发实例剖析/高国宏编著.
-北京：中国民航出版社，1999.11
ISBN 7-80110-364-5

Ⅰ.V… Ⅱ.高… Ⅲ.关系数据库-数据库管理系统，
FoxPro 6.0 Ⅳ.TP311.13

中国版本图书馆 CIP 数据核字（1999）第 61787 号

内容简介

本书按照对象编程的习惯编排，向读者解剖展示了 Visual FoxPro 6.0 一个实际应用程序 TasTrade 订单管理应用系统的核心技术。它是作者经历十一个月专业研究 Visual FoxPro 6.0 的经验总结，也是任何一个 Visual FoxPro 专业程序开发人员的超值参考书。其中，从目录开始就不断地点明了许多技术参考点、研究心得和体会，体现了专业开发人员对开发专业应用程序的忘我追求和不断探索的精神。

本书比较适合 Visual FoxPro 6.0 爱好者、大专院校师生，特别是企业管理电脑化开发人员阅读参考，相信本书将是 VFP 专业开发人员案边一本很有益的参考书。

Visual FoxPro 6.0 实用管理系统开发实例剖析

高国宏 编著

出版 中国民航出版社
社址 北京市朝阳区光熙门北里甲 31 号楼（100028）
发行 中国民航出版社
印刷 广东出版技校彩印厂印刷
开本 787×1092 毫米 1/16
印张 12
字数 285 千字
版本 1999 年 11 月第 1 版 1999 年 11 月第 1 次印刷
印数 1-1900 册

书号 ISBN 7-80110-364-5 / G·170
定价 30.00 元

前言

本人长期致力于企业电脑化管理的商业实践，对于企业电脑化管理实现过程中的困绕是比较清楚的。其中之一，就是如何快速实现目标系统的功能，缩短开发周期和开发成本。

过去，代码编程的年代里企业不得不聘请十几个专业人员，花上几年的时间，才开发出几个通用化程度又很低的应用系统。现在，Visual FoxPro 6.0 给我们这些软件开发人员提供了一个无比强大的开发工具，它使得我们可以单枪匹马地在短期内开发出专业级的应用系统程序，这不仅给我们开辟了广阔的发展空间，而且减轻了企业实现电脑化管理的开支，使企业快速电脑化管理成为可能。笔者曾独自在半年内为几家外资工厂开发了专业级的物料需求计划（MRP）和制造工厂定单计划自动下达系统、工资管理、采购管理、仓库管理系统。电脑化管理可以大幅度地减少企业浪费，提高工作效率，赢得更多的商机，使企业更加有实力进行更大范围的电脑化，以此为快刀利器参入现代化的激烈商业竞争。

Visual FoxPro 6.0 不同于它的前辈，它已经抛弃了代码过程，代之以对象编程。在 Visual FoxPro 6.0 中文简体专业版中，它是以一个名字叫 TasTrade 的实际商业订单管理系统为例来讲解其对象编程技术的。由于对象编程本身已经需要大家换一个写程序的观点，再加上语法也同以前的版本不同，一般开发人员很难在短期内获得要领。笔者就是在经过十一个月的艰苦奋斗之后才可以轻松的同你交流学习过程中的体会，因为，新手往往会在其中找不到下手的途径，而徒见示例软件在潇洒地运行。本书就是把实际的例子解剖给您看，引导您快速进入 Visual FoxPro 6.0 的新境界，届时您会有一种抛下自行车换上 BENZ 跑车的感觉，清凉、舒服、耳目一新！而这个例子，您在任何一个简体中文版的 Visual FoxPro 上都可以找到，对照学习，必使您事半功倍，收益无穷。

Visual FoxPro 6.0 不同于 Microsoft 的其他产品，它是从 FOX 公司购买回来的，不存在 XX 软件产品必须在 X 版之后才能用的说法。Visual FoxPro 不同于 VB、VC++的特征在于：它拥有快速数据模型的构造能力，这是其他对象开发工具所不具备的。对于量身定制的企业管理系统而言，Visual FoxPro 可以说是开发应用管理系统（MIS）的最佳平台。

Visual FoxPro 6.0 也不同于 Delphi、Visual Pro、PROMAKE 等开发平台，它不需要您再学习一种新语言，用惯 XBASE 的开发者，比如 Clipper、FoxPro、dBase 的开发者，您现在就可以上机实践了。许多新电脑软件开发使用后，需要一段时间检验的，Fox 系列大概可以减轻您对开发平台自身安全的恐惧感！试问您乐意把辛苦开发的应用系统开发在您尚不熟悉的平台上去冒险吗？

Visual FoxPro 6.0 在 Windows NT 上是非常安全的，网络速度也比较理想；如果您十分高兴使用 Windows 95 的立体界面风格，您可以在您的网络终端上使用 Win 95。Visual FoxPro 本来就是一个 32 位的开发平台，在这种配置下，您可以用 PC 的价钱，创造小型机的效率。企业电脑化管理之最合理的性能价格比配置，至少在目前应是如此。

笔者极力向您推荐的，就是这样一个开发平台：它既可以继承传统的 XBASE 成果，又具备其他对象开发平台的绝大多数功能，还具有自身的快速模型功能和报表功能。如果您目前手上就有一个客户需要开发管理应用系统，那么，请参考此书立即着手吧！您还可以演示一

下 TasTrade，告诉客户：您未来的应用程序就是这种专业 Windows 风格的精工产品，并且让您一用就会！

Visual FoxPro 6.0 的入门书已经很多了，本书与其他图书不同。本书的目标不是重复 Visual FoxPro 6.0 的语法和帮助文件，而是把 Visual FoxPro 6.0 使用手册上没有专题讲解的实例应用程序解剖给您，看看 Microsoft 是如何开发一套应用管理系统的，并期望您能够开发出专业的管理软件交到客户手上！让您跨入 Windows 下专业开发者的行列里来。

本书的完成应归功于我的那些很好合作的客户们，我为能够在短期内给他们一些帮助而高兴，并一定会开发更好的软件和功能提供给他们，因为，我已经了解了他们所在行业的真正需求和他们所面临的管理困扰，并且大家已经成为好朋友。还要感谢我的助手们，大家辛苦了。

希望能与同仁交流或合作，我的 E_mail 是：mrpiso@public.guangzhou.gd.cn

高国宏　1999 年 8 月 20 日

写于客户企业管理系统开发过程的间隙时间里

佛山,广州两地成书

目 录

目 录

第一章　通用主程序的写法

要点：

在本章中你要明确 VFP 如何使用 API 调用，如何开始事件循环如何设置运行环境。如果你要增强 VFP 功能跟 VB 斗一斗，就一定要搞懂 API。

1.1 主程序代码

一个应用程序总应该有一个起点，在这个起点处作一点准备功夫，这个起点就是我们的主程序。在我们分析的这个主程序里，包含了对环境的设定和建立主不可见类，所以，您应该在研究主程序时同时研究一下 TasTrade 类或 Application 类，是他们调用了环境设定类，并开始了事件循环 READ EVENT 命令。

main.prg 中的代码是本应用程序的主程序，在运行本应用程序时首先执行 main.prg 于中的代码，此代码用于：

1. 声明读写应用程序.INI 文件的 API 函数。

2. 保存某些环境设置，这些环境设置（curdir（），Path 和 ClassLib）必须在本应用程序对象实例化之前设置好。

3. 在用户运行 main.fxp 而不是 tastrade.app 时，调整路径。

4. 将类库设置为 main 和 tsgen。

5. 创建一个基于 mair.vcx 中 TasTrade 的应用程序对象。

```
oApp = CREATEOBJECT("TasTrade")
```

此对象创建时发生以下行为：

1. 产生父类（Application）的 Init 事件，与此事件关联的代码创建一个环境对象并保存其他环境设置：

```
THIS.AddObject("oEnvironment", "Environment")
THIS.oEnvironment.Set( )
```

2. 用 GetPrivString 读取 tastrade.ini 文件中的默认节，以此确定初始显示屏是否显示。

3. 如果已在.INI 中指定，则显示 IntroForm（初始显示屏）。

4. 在 Tastrade 的 Init 事件中调用此类的 Login（）方法。

5. Login（）调用此类的方法 DoFromRetVal（）。

6. DoFormRetVal（）从 Login 中获取参数，创建并显示 LogInPicture 类的一个实例。

7. 在用户登录时，返回用户的访问级别，并作为应用程序对象的 cUserLevel 属性保存。

8. 调用本应用程序对象中的 Do（）方法：

```
oApp.Do( )
```

在方法 Do（）中的代码作如下操作：

1）运行本应用程序的菜单程序：

```
THIS.DoMenu( )
```

2）根据用户的访问级别确定初始时本应用程序要运行的组件。

```
lcAction = THIS.GetStartupAction( )
```

此方法 GetStartupAction 返回此用户在字段 Action 中的值

3）运行组件，如输入订单：

```
IF !EMPTY(lcAction)
&lcAction
ENDIF
```

4）用 READ EVENTS 建立事件等待状态。

示例主程序如下：

```
*****宣布头文件***********
#INCLUDE "INCLUDE\TASTRADE.H"
*****宣布 API 函数**********
****读写私有 INI 文件 API***
*-- DECLARE DLL statements for reading/writing to private INI files
DECLARE INTEGER GetPrivateProfileString IN Win32API  AS GetPrivStr ;
String cSection, String cKey, String cDefault, String @cBuffer, ;
Integer nBufferSize, String cINIFile
DECLARE INTEGER WritePrivateProfileString IN Win32API AS WritePrivStr ;
String cSection, String cKey, String cValue, String cINIFile
*****系统注册 API**********
*-- DECLARE DLL statements for reading/writing to system registry
DECLARE Integer RegOpenKeyEx IN Win32API ;
Integer nKey, String @cSubKey, Integer nReserved,;
Integer nAccessMask, Integer @nResult
DECLARE Integer RegQueryValueEx IN Win32API ;
Integer nKey, String cValueName, Integer nReserved,;
Integer @nType, String @cBuffer, Integer @nBufferSize
DECLARE Integer RegCloseKey IN Win32API ;
Integer nKey
******WINDOWS3.1 索取字串 API*******
*-- DECLARE DLL statement for Windows 3.1 API function GetProfileString
DECLARE INTEGER GetProfileString IN Win32API AS GetProStr ;
String cSection, String cKey, String cDefault, ;
String @cBuffer, Integer nBufferSize
******清理环境**********
CLEAR
*******确认项目管理器关闭,防止热键冲突*******
*-- Ensure the project manager is closed, or we may run into
*-- conflicts when trying to KEYBOARD a hot-key
```

```
DEACTIVATE WINDOW "Project Manager"
*****建立对象后,下列公用参数将被清除*********
*-- All public vars will be released as soon as the application
*-- object is created.
PUBLIC gcOldTalk, gcOldDir, gcOldPath, gcOldClassLib, gcOldEscape
IF SET('TALK') = 'ON'
SET TALK OFF
gcOldTalk = 'ON'
ELSE
gcOldTalk = 'OFF'
ENDIF
gcOldEscape    = SET('ESCAPE')
gcOldDir       = FULLPATH(CURDIR())
gcOldPath      = SET('PATH')
gcOldClassLib = SET('CLASSLIB')
*****主要部分:设置路径、建立对象实例**************************
*-- Set up the path so we can instantiate the application object
IF SetPath()
  PUBLIC oApp
  oApp = CREATEOBJECT("TasTrade")
  IF TYPE('oApp') = "O"
    *-- Release all public vars, since their values were
    *-- picked up by the Environment class
    RELEASE gcOldDir, gcOldPath, gcOldClassLib, gcOldTalk, gcOldEscape
    oApp.Do()
  ENDIF
ENDIF
CLEAR DLLS
CLEAR ALL
RELEASE ALL EXTENDED
*****防止从\PROGS 子目录执行主程序***
FUNCTION SetPath()
  LOCAL lcSys16, ;
        lcProgram
  lcSys16 = SYS(16)
  lcProgram = SUBSTR(lcSys16, AT(":", lcSys16) - 1)
  CD LEFT(lcProgram, RAT("\", lcProgram))
  *-- If we are running MAIN.PRG directly, then
  *-- CD up to the parent directory
```

```
  IF RIGHT(lcProgram, 3) = "FXP"
    CD ..
  ENDIF
******设置路径和类库**********************
  SET PATH TO PROGS, FORMS, LIBS, ;
        MENUS, DATA, OTHER, ;
        REPORTS, INCLUDE, HELP, ;
        BITMAPS
  SET CLASSLIB TO MAIN, TSGEN
ENDFUNC
```

1.2 通用函数库

在 Clipper、FoxPro 2.x 的设计过程中，我们总是要设计大量的函数；今天，在面向对象编程中，我们依然可以设定函数库，不过已经不是必须的了，而且，函数设计数量也少的可怜。正如您的经验所猜到的一样，我们是为了方便后面的示例讲解才在此提及他们的。如果您认为可以忽略，那就直接过去好了，不过，当您在下面讲解时遇到它们时，请回来再看一遍。不要认为简单而放弃了它，按部就班往往有好效果，一日千里往往是走马观花。

```
*-- General purpose utility functions independent of any classes
*-- for better performance and accessibility
****宣布头文件******
#INCLUDE "INCLUDE\TASTRADE.H"

****************************************
FUNCTION IsTag (tcTagName, tcAlias)
  *-- Receives a tag name and an alias (which is optional) as
  *-- parameters and returns .T. if the tag name exists in the
  *-- alias. If no alias is passed, the current alias is assumed.

  ***确认一个索引档,如果索引档存在于此表中,返回.T.;否则假定为当前库的索引
  LOCAL llIsTag, ;
        lcTagFound

***若没有明确别名,则定为当前表的别名
  IF PARAMETERS() < 2
    tcAlias = ALIAS()
  ENDIF
  ****若表别名为空,则 返回 .F.
  IF EMPTY(tcAlias)
    RETURN .F.
```

```
  ENDIF

*****在组合索引中查找索引,注意 TAG（）函数的使用参数***
  lIIsTag = .F.
  tcTagName = UPPER(ALLTRIM(tcTagName))

  lnTagNum = 1
  lcTagFound = TAG(lnTagNum, tcAlias)
  DO WHILE !EMPTY(lcTagFound)
    IF UPPER(ALLTRIM(lcTagFound)) == tcTagName \\精确查询索引的名字
      lIIsTag = .T.
      EXIT
    ENDIF
    lnTagNum = lnTagNum + 1
    lcTagFound = TAG(lnTagNum, tcAlias)
  ENDDO

  RETURN lIIsTag
ENDFUNC

FUNCTION NotYet()
  *-- Used during construction of Tastrade to indicate those
  *-- parts of the application that were not yet completed.
  ***构造例程时,若未完成给予提示*************************
  =MESSAGEBOX(NOTYET_LOC, MB_ICONINFORMATION)
  RETURN
ENDFUNC

FUNCTION FileSize(tcFileName)
  *-- Returns the size of a file. SET COMPATIBLE must be ON for
  *-- FSIZE() to return the size of a file. Otherwise, it returns
  *-- the size of a field.
  ****返回文件大小,SET COMPATIBLE ON 需要先设定,否则返回的是栏位大小
  LOCAL lcSetCompatible, lnFileSize

  lcSetCompatible = SET('COMPATIBLE')
  SET COMPATIBLE ON
  lnFileSize = FSIZE(tcFileName)
  SET COMPATIBLE &lcSetCompatible
  RETURN lnFileSize
```

```
ENDFUNC

FUNCTION FormIsObject()
  *-- Return .T. if the active form is of type "O" and its baseclass
  *-- is "Form".
  ****打开表单对象
  RETURN (TYPE("_screen.activeform") == "O" AND ;
          UPPER(_screen.ActiveForm.BaseClass) = "FORM")
ENDFUNC

FUNCTION OnShutdown()
  *-- Custom message called via the ON SHUTDOWN command to indicate
  *-- that the user must exit Tastrade before exiting Visual FoxPro.
  ****退出提示:先退出子例程,才能推出开发平台
  =MESSAGEBOX(CANNOTQUIT_LOC, ;
              MB_ICONEXCLAMATION, ;
              TASTRADE_LOC)
ENDFUNC
```

注意：CANNOTQUIT—LOC、MB—ICONEXCLAMATION 等常量含义已在头文件中宣布，具体可查看*.h 文件，千万别被它吓呆了。

第二章　标准应用程序类（不可视类的写法）

要点：

在对象编辑中，类是一个很重要的概念，也是 VFP 高级开发技术之一，写类水平的高低标志着你对一种对象的掌握程度，弄不懂这个类，不要向人讲你会 VFP 开发呀！

Application 类是一个典型的不可视类。

tsgen.vcx 中创建了一个通用的应用程序类：Application，它所具有的功能使其能够用于任何应用程序，这些功能包括：

1. 应用程序运行时保存 Visual FoxPro 6.0 环境。
2. 为应用程序建立环境。
3. 应用程序退出时恢复 Visual FoxPro 6.0 环境。
4. 运行与应用程序相关的主菜单并建立事件循环。
5. 管理应用程序工具栏。
6. 管理用户登录过程。
7. 管理表单。

main.vcx 中有一个 Application 的子类 TasTrade，它提供了 Tasmanian Traders 中专有的一些功能：

显示启动屏幕。

用户登录时运行合适的初始组件。

main.prg 中用 TasTrade 类中生成了一个对象：oApp。

2.1 一个重要的不可视类：Application 的设计

Application 类的功能描述。

1. 当应用程序运行时，应用程序对象将保存 Visual FoxPro 6.0 环境。

Visual FoxPro 6.0 环境包括 SET 命令的设置，如 PATH，TALK，CARRY，等等。在 Application 类的 Init 事件代码中向 Application 中添加了一个新的对象，该对象是基于 tsgen.vcx 中的类 Environment。在 Environment 类的 Init 事件代码中，已有的 Visual FoxPro 6.0 的环境设置被保存到类的属性中，以便以后恢复使用。此外，该代码还将 Visual FoxPro 6.0 的主窗口标题保存到 Apllication 类的属性 cOldWindCaption 中。当释放一个基于 Application 类的对象时，可以恢复这些已保存的设置。

Visual FoxPro 6.0 中另一个环境设置包括在运行一个应用程序时活动的标准工具栏。Application 的方法 ReleaseToolbars（）将系统工具栏的名称保存到 Application 类中的属性 aToolbars 的元素中，同时还保存它们在运行应用程序时是打开还是关闭的信息。Application 的 Init 事件代码还可以将当前系统菜单压入菜单栈中，以便以后恢复。

2. 应用程序对象可以为应用程序建立运行环境。

Application 类的 Init 事件代码中其他部分可以建立运行环境，此代码可以：

• 调用 Environment 类中的方法 Set（），设置类和过程库，设置是否显示已删除的记录（ClassLib，Procedure，Deleted），并设置其他特定应用程序的环境设置。

• 为应用程序打开数据库。

• 调用方法 ReleaseToolbars（）释放运行应用程序时活动的所有系统工具栏。

* 用保存在 Application 类中 cMainWindCaption 属性的值设置 Visual FoxPro 6.0 的主窗口标题。

3. 应用程序退出时，应用程序对象可以恢复 Visual FoxPro 6.0 的原有环境。

Application 类中方法 Cleanup（）的代码可以恢复 Visual FoxPro 6.0 的主窗口标题，关闭数据库，清除窗口，发布命令 CLEAR EVENTS，恢复初始菜单，并调用方法 Show Toolbars（）重新打开在 TasTrade.app 运行前已显示的系统工具栏。

当发布 CLEAR EVENTS 命令时，程序继续执行 READ EVENTS 并且 main.prg 中的代码释放 oApp。释放此应用程序对象时，将发生 Environment 类中的 Destroy 事件，因为它是 Application 类的一个成员。Enviroment 对象的 Destroy 事件代码将调用 Environment 对象中的方法 Reset（），恢复 Visual FoxPro 6.0 的环境设置，此环境设置是在创建这个 Environment 对象时就已经存在的。

4. 此应用程序对象运行与本应用程序相关的主菜单并建立事件循环。

一旦 Application 对象建立起来，main.prg 中应用程序类的 Do（）方法就将被调用，Do（）方法将运行主菜单程序并发布命令 READ EVENTS。

5. 应用程序对象管理应用程序工具栏。

表单级的属性指示与每个表单关联的工具栏，在这一个应用程序中，仅有一个工具栏，所以此属性对于本应用程序中的所有表单都是相同的。本应用程序中的表单调用类 Application 中的方法显示 ShownavToolbar（）显示此工具栏。本应用程序知道此工具栏是否已经存在，并只在必要时才显示它，同时增加属性 cFormInstanceCount 的值，当释放最后一个表单时，此表单的事件 Destroy 调用本应用程序中的方法 ReleaseNavToolbar（），减少 nFormInstanceCount 值，并移去此工具栏。

6. 本应用程序对象管理用户登录过程。

大部分应用程序的安全性即提供了一些登录过程的表单，类 Application 中的方法 Login（）调用方法 DoFormRetVal（），传送登录屏幕的类名。用户登录时，类 Application 的一个属性 cUserLevel，保存此用户的用户级类型，此属性用于菜单清理代码中移去那些用户不能查看的菜单项。

7. 本应用程序对象管理表单。

除了报表中用于为报表设置值范围的表单外，Tasmanian Traders 的其它表单均通过类 Application 中的方法显示，方法 DoForm（）接受一个表单的名，并将一个参数或选地传给此表单，而方法 DoFormRetVal（）接受一个表单类的名，并返回此表单指定的一个值。

```
*-- ParentClass:   custom
*-- BaseClass:    custom
```

此类实际上是一个真正合乎语法的 Visual FoxPro 主程序，其中包含了勾画一个主程

序的全部内容，比如：READ EVENTS 和 CLEAR EVENTS 语句，登录程序调用，主菜单的调用等等。在本书中，这是一个您必须下工夫细读的一个重要部分。

从类的写法来讲，这也是一种重要的类——不可见类的一个好例子，许多数据处理技巧是在这种类中实现的，比如：记录操作员的授权级别，口令，名称代码，并传递给其它操作表单。

Applicantion 类的代码清单如下：

```
#INCLUDE "c:\mainsamp\include\tastrade.h"
DEFINE CLASS application AS custom
Height = 34
Width = 92
*-- 应用程序主窗口的标题内容
cmainwindcaption = ""
*-- 运行时的主菜单（.MPR file）.
PROTECTED cmainmenu
cmainmenu = "MAIN.MPR"
*-- 应用程序的数据库
PROTECTED cdatabase
cdatabase = ""
*-- 应用程序运行前的窗口标题
PROTECTED coldwindcaption
coldwindcaption = ""
*-- 打开的事例数量
PROTECTED nforminstancecount
nforminstancecount = 0
*-- 定位工具栏参数
otoolbar = .NULL.
Name = "application"

*-- 环境已经清理的标志
PROTECTED lisclean

*-- 应用程序开始运行时的开发平台上的工具栏名阵列
PROTECTED atoolbars[1,1]

*-- Contains form names, object references, the number of current running instances, and the next available instance number.
DIMENSION ainstances[1,4]

*-- Puts up the main menu and runs the application.
```

```
PROCEDURE do
*-- Put up main menu
***调用主菜单*****
this.DoMenu()

*-- Start the event loop
***开始事件循环,标志一个应用程序的真正运行*******
READ EVENTS
ENDPROC

*-- Closes all windows, restores the main window caption, restores the VFP menu, etc.
PROCEDURE cleanup
*-- When we wish to end the application, we cannot just
*-- simply release the application object (oApp) and expect
*-- the Destroy method to run without first issuing a
*-- CLEAR EVENTS since the READ EVENTS was issued in the Do()
*-- method. Therefore, this method was created to
*-- clean up the environment before quitting the application.
*-- It also allows us to conditionally stop the user from
*-- exiting the program for whatever reason.
***关闭全部表单,注意下面的 FORMS（INFORMTOCLOSE）.RELEASE 的用法
LOCAL lnForm, lnFormToClose
lnFormToClose = 1
FOR lnForm = 1 TO _screen.FormCount
IF TYPE("_screen.Forms(lnFormToClose)") == "O"
IF _screen.Forms(lnFormToClose).QueryUnload()
_screen.Forms(lnFormToClose).Release()
ELSE
RETURN .F.
ENDIF
ELSE
lnFormToClose = lnFormToClose + 1
ENDIF
ENDFOR
***恢复原来的标题********
_screen.caption = this.cOldWindCaption
***关闭数据库*************
CLOSE DATA ALL
```