目 录

第一篇 进入 VFP 6.0 大门

第 1 章 概述 …… 2
1.1 从 dBASEⅡ到 Visual FoxPro 6.0 …… 2
1.1.1 xBASE 系列数据库的发展 …… 2
1.2 Visual FoxPro 6.0 的特点 …… 3
1.2.1 总体特点 …… 3
1.2.2 用户界面的特点 …… 3
1.2.3 数据库引擎性能的改进 …… 4
1.2.4 语言上的特点 …… 4
1.2.5 方便的数据库容器 …… 4
1.3 小结 …… 5
思考与练习 …… 5
第 2 章 打开 Visual FoxPro 6.0 的大门 …… 6
2.1 Visual FoxPro 6.0 系统的安装 …… 6
2.1.1 安装的系统软硬件需求 …… 6
2.1.2 从 CD-ROM 上安装的过程 …… 6
2.1.3 从网络上用 CD-ROM 安装的过程 …… 6
2.1.4 启动与退出 Visual FoxPro 6.0 …… 7
2.2 Visual FoxPro 6.0 系统的配置 …… 8
2.2.1 配置 Visual FoxPro 6.0 …… 8
2.2.2 通过“选项”卡配置系统 …… 9
2.3 Visual FoxPro 6.0 的系统结构 …… 11
2.3.1 整体结构与功能 …… 11
2.3.2 使用集成环境 …… 12
2.3.3 使用菜单系统 …… 13
2.3.4 使用工具栏 …… 16
2.3.5 自己定制工具栏 …… 17
2.3.6 使用命令窗口 …… 17
2.3.7 使用设计器 …… 18

2.3.8　使用向导 …… 18
2.3.9　使用生成器 …… 19
2.4　小结 …… 20
思考与练习 …… 20

第二篇　基础篇

第 3 章　VFP 6.0 可视化数据库管理 …… 22
3.1　使用“项目管理器”管理项目 …… 22
3.1.1　项目管理器简介 …… 22
3.1.2　管理项目的“数据” …… 23
3.1.3　管理项目的“文档” …… 23
3.1.4　在“项目管理器”中添加、移去文件 …… 24
3.1.5　在“项目管理器”中新建、修改文件 …… 25
3.1.6　“浏览”项目中“表”的数据 …… 26
3.1.7　“项目信息”的显示与编辑 …… 26
3.1.8　给“项目”中的文件添加说明 …… 26
3.1.9　不同“项目”之间的文件共享 …… 27
3.1.10　定制自己的“项目管理器” …… 27
3.2　建立数据库文件 …… 28
3.2.1　建立数据库文件 …… 28
3.3　建立表和索引 …… 28
3.3.1　创建表 …… 28
3.3.2　浏览表中的数据 …… 31
3.3.3　编辑表中的数据 …… 33
3.3.4　向表中添加数据 …… 33
3.3.5　从表中删除数据 …… 34
3.3.6　定制浏览数据窗口 …… 34
3.3.7　修改表的结构 …… 36
3.3.8　给表加过滤器 …… 36
3.3.9　给表加索引 …… 38
3.3.10　用索引给表排序 …… 39
3.3.11　对多个字段排序 …… 40
3.3.12　筛选记录 …… 40
3.4　用“数据库”管理数据 …… 41
3.4.1　数据库的设计过程 …… 41
3.4.2　数据库的简单操作 …… 42
3.4.3　向数据库中添加删除表 …… 43

3.4.4 在数据库中查找表 …… 44
3.4.5 使用长表名与注释 …… 44
3.4.6 使用长字段名、标题与注释 …… 45
3.4.7 使用默认值 …… 46
3.4.8 设置字段级规则 …… 47
3.4.9 设置记录级规则 …… 48
3.4.10 设置触发器 …… 49
3.4.11 建立、编辑永久性关系 …… 50
3.4.12 建立、编辑临时性关系 …… 52
3.4.13 建立参照的完整性 …… 52
3.4.14 使用多个数据库 …… 53
3.5 小结 …… 53
思考与练习 …… 53
第4章 信息查询与更新 …… 55
4.1 使用“查询”查询数据 …… 55
4.1.1 创建查询 …… 55
4.1.2 为查询输出字段建立别名 …… 57
4.1.3 为查询结果排序 …… 57
4.1.4 筛选查询结果 …… 58
4.1.5 查询结果的分组 …… 58
4.1.6 确定查询去向 …… 59
4.1.7 验证查询语句 …… 60
4.1.8 给查询添加备注 …… 61
4.1.9 运行查询 …… 61
4.2 使用“视图”浏览更新数据 …… 61
4.2.1 创建本地视图 …… 61
4.2.2 与远程数据连接 …… 63
4.2.3 建立远程视图 …… 64
4.2.4 用视图更新数据 …… 64
4.2.5 控制更新数据的条件 …… 65
4.2.6 控制视图字段的显示与输入 …… 66
4.2.7 控制视图更新的方法 …… 67
4.2.8 为视图添加筛选表达式 …… 67
4.2.9 为视图传递参数 …… 68
4.2.10 设置与服务器的连接时间 …… 68
4.2.11 优化视图 …… 68
4.3 查询更新多个表 …… 69
4.3.1 向查询中添加表或视图 …… 69
4.3.2 通过联接选择记录 …… 70

4.3.3 删除修改表间的联接 …… 71
4.3.4 将多个本地表加入本地视图 …… 72
4.3.5 将多个远程表加入远程视图 …… 73
4.3.6 将本地表和远程表同时加入视图 …… 73
4.4 小结 …… 74
思考与练习 …… 74
第 5 章 信息的显示与输出 …… 75
5.1 设计报表和标签 …… 75
5.1.1 规划你的报表布局 …… 75
5.1.2 创建报表的布局 …… 76
5.1.3 创建标签的布局 …… 79
5.1.4 修改布局 …… 81
5.1.5 规划数据的显示位置 …… 82
5.1.6 用“快速报表”添加控制 …… 82
5.1.7 设置数据源 …… 84
5.1.8 向报表中添加控制 …… 85
5.1.9 报表控件的布局 …… 90
5.1.10 给布局增加分组 …… 92
5.1.11 定制报表的页面 …… 95
5.1.12 设置报表的带区 …… 96
5.1.13 改变“域控件”和“标签”的字体 …… 97
5.1.14 改变控件的颜色 …… 98
5.1.15 打印和预览报表 …… 98
5.2 使用表单 …… 99
5.2.1 创建表单 …… 99
5.2.2 向表单中快速添加数据字段 …… 101
5.2.3 设置表单数据环境 …… 103
5.2.4 向表单添加控件 …… 105
5.2.5 使用“生成器”向表单添加控件 …… 106
5.2.6 设置控件的属性和方法 …… 107
5.2.7 向表单同时加入多个控件 …… 110
5.2.8 设置字段映象 …… 110
5.2.9 向表单中快速添加字段的控件 …… 112
5.2.10 复制删除表单中的控件 …… 113
5.2.11 为控件设置 Tab 顺序 …… 114
5.2.12 定制表单 …… 115
5.3 小结 …… 117
思考与练习 …… 117

第三篇 设计篇

第 6 章 VFP 6.0 程序设计简介 …… 120
6.1 Visual FoxPro 程序设计框架 …… 120
6.1.1 程序设计的特点 …… 120
6.1.2 Visual FoxPro 程序设计机制 …… 121
6.1.3 使用命令窗口 …… 121
6.1.4 在代码窗口中建立程序 …… 121
6.1.5 用 Visual FoxPro 的工具建立程序代码 …… 122
6.2 Visual FoxPro 程序设计过程 …… 122
6.2.1 Visual FoxPro 程序设计流程 …… 122
6.2.2 确定任务分析数据 …… 123
6.2.3 分解任务与模块划分 …… 123
6.2.4 建立原形系统 …… 123
6.2.5 组装模块进行整体测试 …… 123
6.3 面向对象程序设计的新特性 …… 123
6.3.1 从面向过程到面向对象 …… 123
6.3.2 面向对象程序设计的基本概念 …… 124
6.4 小结 …… 125
思考与练习 …… 125
第 7 章 VFP 6.0 程序语言概论 …… 126
7.1 语法和命名规则 …… 126
7.1.1 变量的命名 …… 126
7.1.2 对象的命名 …… 127
7.1.3 表字段的命名 …… 128
7.1.4 变量的命名 …… 129
7.2 数据类型与字段类型 …… 129
7.2.1 数据类型与字段类型 …… 129
7.2.2 数据类型与字段类型的说明 …… 130
7.3 数据的存储类型 …… 132
7.3.1 数据的存储类型 …… 132
7.3.2 常量 …… 132
7.3.3 变量 …… 133
7.3.4 数组 …… 135
7.3.5 字段 …… 135
7.3.6 记录 …… 136
7.3.7 对象 …… 136
7.4 操作符 …… 136

7.4.1 字符操作符 …… 136
7.4.2 日期和时间操作符 …… 136
7.4.3 逻辑操作符 …… 137
7.4.4 关系操作符 …… 137
7.4.5 数值操作符 …… 137
7.5 表达式 …… 138
7.5.1 表达式的名称命名规则 …… 138
7.5.2 字符表达式 …… 138
7.5.3 日期表达式 …… 138
7.5.4 数值表达式 …… 139
7.5.5 逻辑表达式 …… 139
7.5.6 名称表达式 …… 139
7.5.7 使用宏 …… 139
7.6 程序的流程控制 …… 140
7.6.1 流程控制介绍 …… 140
7.6.2 条件分支 …… 140
7.6.3 循环 …… 142
7.7 过程与自定义函数 …… 144
7.7.1 什么是自定义函数 …… 144
7.7.2 过程与函数的定义 …… 145
7.7.3 对过程和函数的调用 …… 145
7.7.4 向过程和函数传递参数 …… 145
7.7.5 函数的返回值 …… 146
7.8 筛选字段和记录 …… 146
7.8.1 记录作用域 …… 146
7.8.2 使用 FOR 子句 …… 147
7.8.3 使用 WHILE 子句 …… 147
7.9 使用数组 …… 148
7.9.1 从表向数组传递数据 …… 148
7.9.2 从数组向表中传递数据 …… 150
7.10 使用 NULL 值 …… 152
7.10.1 NULL 值的特点 …… 152
7.10.2 用 NULL 作为值 …… 152
7.10.3 表达式中使用.NULL. …… 153
7.10.4 用.NULL.作为参数 …… 153
7.11 小结 …… 153
思考与练习 …… 154
第 8 章 VFP 6.0 面向对象的程序设计 …… 155
8.1 Visual FoxPro 中的类和对象 …… 155

8.1.1 Visual FoxPro 中的对象 …… 155
8.1.2 对象的属性、事件和方法 …… 155
8.1.3 Visual FoxPro 中的类 …… 156
8.1.4 Visual FoxPro 中类的层次 …… 156
8.1.5 容器类和非容器类 …… 157
8.2 用类设计器设计类 …… 158
8.2.1 什么时候使用类 …… 158
8.2.2 从基类派生出新类 …… 159
8.2.3 用类设计器创建类 …… 160
8.2.4 用类设计器创建自定义类 …… 161
8.2.5 用类设计器修改类 …… 168
8.2.6 创建组件 …… 169
8.2.7 修改和保护属性和方法 …… 175
8.2.8 复制删除类库中的类 …… 179
8.3 用编程的方法设计类 …… 179
8.3.1 程序中定义类的语法 …… 179
8.3.2 保护类成员 …… 180
8.3.3 从类中创建对象 …… 180
8.3.4 向容器类中添加对象 …… 181
8.3.5 用程序创建按钮组件 …… 181
8.3.6 用程序定义表格控件 …… 186
8.3.7 把类成员定义成数组 …… 188
8.3.8 创建对象数组 …… 188
8.3.9 使用对象存储数据 …… 189
8.4 对类和对象的引用 …… 190
8.4.1 容器中对象的引用层次 …… 190
8.4.2 对象的相对引用 …… 191
8.4.3 属性的设置 …… 192
8.4.4 方法的调用 …… 192
8.4.5 事件的响应 …… 193
8.4.6 创建对对象的引用 …… 193
8.4.7 数据和对象的集成 …… 194
8.5 对父类方法的引用 …… 194
8.5.1 对子类的方法进行扩充 …… 194
8.5.2 修改属性的默认值 …… 196
8.6 小结 …… 196
思考与练习 …… 196
第 9 章 设计应用程序界面 …… 197
9.1 菜单系统的设计 …… 197

9.1.1 创建菜单系统的步骤 …… 197
9.1.2 菜单系统的规划 …… 198
9.1.3 创建菜单系统 …… 199
9.1.4 创建快捷菜单 …… 200
9.1.5 对菜单项进行分组 …… 202
9.1.6 指定菜单的快速访问键 …… 202
9.1.7 创建菜单的键盘快捷键 …… 203
9.1.8 设定菜单项的启用状态 …… 203
9.1.9 标记菜单项的使用状态 …… 204
9.1.10 设置对菜单项的响应 …… 204
9.1.11 增加初始化和清理代码 …… 206
9.1.12 在程序运行时控制菜单 …… 207
9.1.13 菜单系统的测试和调试 …… 208
9.1.14 在状态栏中显示提示信息 …… 209
9.1.15 设置菜单标题出现的位置 …… 209
9.1.16 保存恢复菜单 …… 210
9.1.17 给菜单系统加入默认过程 …… 210
9.2 为应用程序创建工具栏 …… 211
9.2.1 创建工具栏 …… 211
9.2.2 向工具栏中添加对象 …… 211
9.2.3 用表单设计器向表单集中添加工具栏 …… 211
9.2.4 用代码向表单集中添加工具栏 …… 212
9.2.5 创建用户的工具栏 …… 212
9.2.6 让工具栏按钮执行相应操作 …… 214
9.2.7 创建和工具栏协调工作的菜单 …… 215
9.3 使表单协同工作 …… 216
9.3.1 应用程序中的表单和表单集 …… 216
9.3.2 单文档和多文档的用户界面 …… 216
9.3.3 设置表单的不同类型 …… 217
9.3.4 给主表单添加菜单 …… 218
9.3.5 创建表单集 …… 218
9.3.6 对象集合和对象数量属性 …… 219
9.4 使用控件 …… 220
9.4.1 控件和数据 …… 220
9.4.2 选择合适的控件 …… 220
9.4.3 使用选项按钮组 …… 221
9.4.4 使用列表框和下拉列表框 …… 224
9.4.5 使用复选框 …… 229
9.4.6 使用文本框 …… 230

9.4.7 使用编辑框 …… 232
9.4.8 使用组合框 …… 234
9.4.9 使用微调控件 …… 236
9.4.10 使用命令按钮和命令按钮组 …… 237
9.4.11 计时器控件 …… 238
9.4.12 使用图像和标签 …… 240
9.4.13 使用形状和线条 …… 241
9.4.14 表单的图形方法 …… 242
9.4.15 使用表格 …… 242
9.4.16 使控件的使用更加简单 …… 246
9.4.17 允许鼠标拖放 …… 249
9.4.18 使用页框 …… 254
9.5 小结 …… 256
思考与练习 …… 257
第 10 章 VFP 6.0 事件驱动模型 …… 258
10.1 在 Visual FoxPro 中使用事件 …… 258
10.1.1 Visual FoxPro 中的核心事件 …… 258
10.1.2 容器和对象的事件 …… 258
10.1.3 类与控件的事件 …… 260
10.2 Visual FoxPro 的事件发生顺序 …… 261
10.2.1 跟踪事件的发生顺序 …… 261
10.2.2 Visual FoxPro 的事件发生顺序 …… 261
10.2.3 给事件添加代码 …… 263
10.3 Visual FoxPro 的事件分类 …… 263
10.3.1 鼠标事件 …… 263
10.3.2 键盘事件 …… 264
10.3.3 焦点事件 …… 264
10.3.4 表单事件 …… 264
10.3.5 其他常用的事件 …… 265
10.4 小结 …… 265
思考与练习 …… 265
第 11 章 VFP 6.0 应用系统的集成和测试 …… 266
11.1 应用程序框架的构造 …… 266
11.1.1 Visual FoxPro 应用程序结构 …… 266
11.1.2 建立设置应用程序主程序 …… 267
11.1.3 设置应用程序的执行环境 …… 268
11.1.4 显示初始的程序界面 …… 268
11.1.5 控制事件的循环 …… 268
11.1.6 恢复应用程序的初始环境 …… 269

11.1.7 建立编译应用程序 …… 269
11.2 应用程序的测试与调试 …… 271
11.2.1 Visual FoxPro 中的测试和调试 …… 271
11.2.2 创建测试的环境 …… 272
11.2.3 使用调试器 …… 273
11.2.4 设置程序断点 …… 274
11.2.5 观察运行变量 …… 277
11.2.6 处理运行中的错误 …… 278
11.2.7 处理过程中的错误 …… 279
11.2.8 处理类和对象中的错误 …… 280
11.3 应用程序的优化 …… 283
11.3.1 对表和索引的优化 …… 283
11.3.2 使用 Rushmore 技术 …… 284
11.3.3 对索引使用 Rushmore 技术 …… 285
11.3.4 关闭 Rushmore …… 285
11.3.5 优化 Rushmore 表达式 …… 286
11.3.6 优化表单和控件 …… 288
11.3.7 优化程序 …… 290
11.3.8 多用户环境下的应用 …… 292
11.3.9 优化对远程数据的提取 …… 292
11.4 小结 …… 293
思考与练习 …… 293

第四篇 基于 WEB 数据库开发

第 12 章 Web 中的数据库开发 …… 296
12.1 Web 中的数据库开发 …… 296
12.1.1 开发基于 Web 的应用程序 …… 296
12.1.2 在 Web 上运行应用程序 …… 296
12.2 Visual FoxPro 的 Web 开发 …… 298
12.2.1 为何使用 Visual FoxPro 开发 Web 应用程序 …… 298
12.2.2 建立基于 Visual FoxPro 的 WWW 查询网页 …… 298
12.2.3 建立基于 Visual FoxPro 的 WWW 网上发布 …… 306
12.3 小结 …… 312
思考与练习 …… 312

第一篇

进入 VFP6.0 大门

本篇导读

本篇引导读者进入 Visual FoxPro 的世界。尽管 Visual FoxPro 在 90 年代才出现，但是它的历史以及它在微机市场上的号召力，却不是一个新软件所能比拟的。

本篇主要讲述以下内容：

第 1 章论述 Visual FoxPro 的历史和特点，讲述 Visual FoxPro 从 dBASE II 到 Visual FoxPro 6.0 的发展，以及 Visual FoxPro 6.0 的总体结构、用户界面、数据库的性能、数据库的语言等方面的特点。

第 2 章主要讲述系统的安装及系统的配置。其中系统安装部分讲述了系统的软硬件需求和安装过程。系统的配置部分讲述了系统的配置过程、方法及应该注意的地方，系统的结构讨论了 Visual FoxPro 6.0 在整体结构、集成环境、菜单系统、工具栏、命令窗口、设计器、生成器和使用向导等方面的功能。

第1章 概 述

本章将通过对 Visual FoxPro 6.0 的简要介绍，使用户了解 Visual FoxPro 6.0 的历史、数据库的概念、Visual FoxPro 6.0 的特点以及 Visual FoxPro 6.0 的使用方法。

本章的主要内容有：

(1) 从 dBASE II 到 Visual FoxPro 6.0；

(2) Visual FoxPro 6.0 的特点。

1.1 从dBASE II 到Visual FoxPro 6.0

1.1.1 xBASE系列数据库的发展

80 年代，随着 PC 及其兼容机的广泛应用，dBASE II 一出现便成为微机市场上的主流数据库管理程序。在鼎盛时期，Ashton-Tare 的这一数据库产品曾占领了微机数据库市场的80%~90%。但由于 dBASE II 是解释型的而不是编译型的，随时需要一个解释器，运行程序之前都需要编译，从而使得大型文件的运行效率十分低下，不利于开发大型的商用化软件。

Fox Software 的 FoxBase，不但与 dBASE 高度兼容，而且被认为是 dBASE II 系列中最好的编译器。当性能优越的 FoxPro 首次推出时，便赢得了广泛欢迎，FoxPro2.0 采用了 Fox 公司的 Rushmore 技术，使得运行的效率提高了上千倍。Watcom C 发展公司为 FoxPro2.0 设计的 Distribution Kit 使程序员能生成 EXE 文件，极大地方便了大型商用程序的开发。

软件界巨头 Microsoft 公司收购了 Fox 公司后，对 FoxPro 进行了革命性的改进，为 FoxPro 加进了 OOP 技术、可视化方法，从而使它的新产品 Visual FoxPro 成为与 Visual Basic 同等地位的产品。

如今，最新的 Visual FoxPro 6.0 出现了，它令广大 xBASE 用户耳目一新。并且它有望成为客户机/服务器结构中前端的主要开发工具，它新增的 Web 发布功能，积极地迎接网络时代的到来。

说 明 什么是数据库？每天我们都接触大量的信息，如：货物清单、财务报表、工资表格、人事档案……，这些让人应接不暇的数据，如果采用某种方式利用计算机存成电子数据，把数据分门别类，提供查找的方法，可以随时检索、找到所需要的数据。这些数据大多存在二维的数据表格中，例如职员的信息可以存在表 1.1 中。

假如公司中也像表 1.1 那样数据库非常多，且数据量又非常庞大，这时就需要借助于计算机将这些数据存于“数据库”中。目前广为流行的是关系型数据库，它将相关的表

（Table）、视图（View）、关系（Relation）等组织起来并达到很高的层次，从而方便用户对数据的管理。Visual FoxPro 便是这样一个出色的关系型数据库管理系统，它为用户对数据进行查询和维护带来了极大方便。

表1.1 职员信息表

职员代号	职员名称	性别	电话	出生日期
10001	张三	男	12345678	1967-10-2
10002	李四	男	234567456-789	1972-3-4
20001	王五	女	23450787	1954-10-3
20002	赵六	女	52345567	1976-3-8
...	...	...	...	...

了解某一软件的历史与对这个软件的学习似乎没有太大关系。然而如果你是一个商业用户，并且有整个决策的权利，那么对软件历史的了解、市场情况、公司背景等还是需要知道的，因为这对于正确的决策至关重要。

1.2 Visual FoxPro 6.0 的特点

1.2.1 总体特点

Visual FoxPro 6.0 使用户对数据的组织、定义数据库及相关规则、建立应用系统变得更为方便简单。用可视化的工具或向导你能很快建立表单、查询和报表。

如果能充分利用 Visual FoxPro 提供的集成环境、强大的面向对象的编程工具、客户机/服务器功能和对 OLE 与 ActiveX 的支持，可以使用户在建立复杂应用系统时更为简单而方便。它在速度、能力和灵活性多方面的改进，使 xBASE 数据库进入了一个新的时代，其全新的对象和事件模型，使应用程序在创建过程、代码的添加与管理上比以前更为快速。Visual FoxPro6.0 在总体结构上比以前的版本更为清晰，用户操作更为方便，它提供的独立调试工具使应用程序的调试变得更加方便。

1.2.2 用户界面的特点

通过对界面的改进，如今的 Visual FoxPro 6.0 在向导、生成器、工具栏和设计器的帮助下，应用系统的开发变得十分简单。

项目管理器负责集中管理整个应用程序的框架和所需要的各种元素，如图 1.1 所示，为中文 Visual FoxPro 的环境。

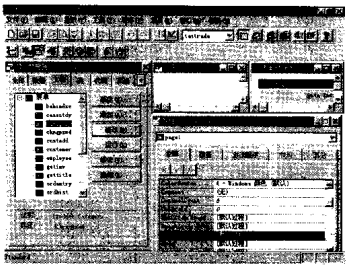


图1.1　Visual FoxPro 6.0 的界面特点

1.2.3　数据库引擎性能的改进

Visual FoxPro 数据库改善了原来的数据引擎，从而支持客户机/服务器、支持 NULL、提供了事务处理的能力。因而 Visual FoxPro 可作为前端开发强大的客户机/服务器程序。

Visual FoxPro 支持数据字典、本地视图、远程视图，支持 NULL 值、事务处理，通过 ODBC 对任何数据源进行访问，为开发客户机/服务器程序提供了强大功能。

Visual FoxPro 采用了 Rushmore 技术，可以大幅度地提高查询的效率。

说明　什么是 Rushmore 技术？Rushmore 技术是一种数据索引技术，通过标准的 Visual FoxPro 索引可以优化数据的检索过程。在 Visual FoxPro 中可以对任何索引用 Rushmore 技术，包括结构复合索引（.CDX）、非结构复合索引（.CDX）和独立索引（.IDX）。它们通过压缩技术，只相当于旧的索引结构的六分之一，由于磁盘的读取时间减少了，从而减少了索引的读取时间。它可以将大量的索引内容读入内存从而有了更高的效率。但当内存空间较小，读取一个大的数据表时，内存可能不足，此时虽然不会丢失数据，但 Rushmore 技术对数据的查询不会带来任何好处。

1.2.4　语言上的特点

Visual FoxPro 语言拥有 1000 多个命令和函数。它虽然对 xBASE 的面向过程的编程方式提供支持，但最大的变化是它是一个真正的面向对象的编程语言。

Visual FoxPro 对象的模型包括了面向对象的程序设计的所有功能，如封装、继承、多态和子类，从而极大地扩展了传统 xBASE 的功能。在 Visual FoxPro 中可以实现真正的事件驱动程序，无须用 READ 语句和事件处理的代码。Visual FoxPro 为你提供了自动处理事件的事件模型，不必再使用 READ 语句或编写自己的事件处理程序。

1.2.5　方便的数据库容器

Visual FoxPro 提供了数据库容器为交互式的用户、应用程序的开发者提供了对数据库

的集中管理。Visual FoxPro 中，数据库包括表、关系、视图和数据字典等，为用户对数据的管理提供了极大方便，使得用户对数据的完整性、参照的完整性的维护变得非常容易。

1.3 小 结

本章论述了 Visual FoxPro 的历史和特点，讲述了 Visual FoxPro 从 dBASE II 到 Visual FoxPro 6.0 的发展，Visual FoxPro 6.0 在总体结构、用户界面、数据库的性能、数据库的语言等方面的特点。使用户在总体上对 Visual FoxPro 有了一个大概的了解。

思考与练习

1. 请简单论述一下你对数据库的理解。
2. 什么是 Rushmore 技术？它具有什么作用？

第2章　打开Visual FoxPro 6.0 的大门

本章主要介绍 Visual FoxPro 6.0 的安装条件，如何安装、配置系统，Visual FoxPro6.0 的总体框架结构，如何最快捷地掌握 Visual FoxPro。

本章的主要内容有：

(1) Visual FoxPro 6.0 系统的安装；

(2) Visual FoxPro 6.0 系统的配置；

(3) Visual FoxPro 6.0 的系统结构。

2.1　Visual FoxPro 6.0 系统的安装

2.1.1　安装的系统软硬件需求

Visual FoxPro 6.0 尽管功能强大，但对于系统的硬件要求并不苛刻，它的硬件安装的要求如下：

一台IBM PC 486 50MHz 计算机或其兼容机；

一个鼠标；

10MB内存，推荐16MB以上；

最小化安装需要15MB硬盘空间，典型安装需要100MB，最大安装需要240MB空间；

若进行网络安装需要一个支持Windows的网络和一个带硬盘的服务器；

采用支持VGA或更高分辨率的显示器；

系统需要运行Windows95或Windows NT 3.51或更高版本。

2.1.2　从CD-ROM上安装的过程

目前绝大多数的计算机都安装了CD-ROM驱动器，这里主要介绍从CD-ROM对Visual FoxPro的安装，软盘的安装请查阅相关资料。

从CD-ROM上安装的过程如下：

开始 将 CD 插入 CD-ROM 驱动器。

下一步 从“资源管理器”的目录中选择光区，找到 setup.exe 文件并运行。

下一步 按照安装向导选择安装形式并完成安装。

完成 退出安装。

2.1.3　从网络上用CD-ROM安装的过程

对于网络上的用户，可以实现资源的共享，可以按照如下方法进行安装：

开始 将 CD 插入与网络相连的工作站的任何共享的 CD-ROM 驱动器中。

下一步 从“资源管理器”的目录中选择“映射网络驱动器”将 CD-ROM 进行映射。

下一步 从“资源管理器”的目录中选择映射的驱动器，找到 setup.exe 文件并运行。

下一步　按照安装向导选择安装形式并完成安装。

完成　退出安装。

2.1.4　启动与退出Visual FoxPro 6.0

Visual FoxPro 6.0 的启动：

从“开始”菜单找到“Microsoft Visual FoxPro”组，选择“Microsoft Visual FoxPro 6.0”项，出现如图 2.1 所示菜单。

图2.1　启动Visual FoxPro 6.0

技巧　为了方便地启动，可以在自己的桌面上加入自己的启动快捷键，方法如下：

在桌面上单击鼠标右键，选择快捷菜单中的“新建”，然后选择“快捷方式”。

(1) 在“创建快捷方式”对话框上选择“浏览”命令按钮，然后在“浏览”窗口中找到 Visual FoxPro 6.0 所在的目录，找到 Vfp.exe，选“打开”，如图 2.2 所示。

(2) 在“创建快捷方式”对话框中选择“下一步”。

(3) 在“为程序选择标题”对话框中“选定快捷方式（S）”中输入你所要的快捷方式的名称，然后选择“完成”。

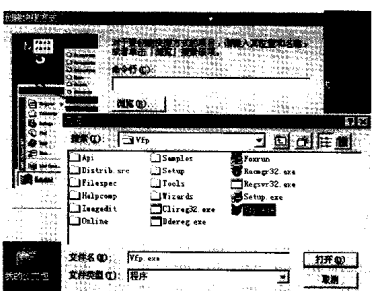


图2.2　创建Visual FoxPro 6.0 快捷方式

Visual FoxPro 6.0 退出有以下方法：

(1) 在“命令”窗口打“quit”命令，如图2.3所示。

(2) 直接按 Alt+F4。

(3) 在“文件”菜单选择“退出”命令。

(4) 双击主窗口左上角的控制菜单。

(5) 在主窗口控制菜单中选择“关闭”，如图 2.4 所示。

图2.3　从命令窗口退出

图2.4　从控制菜单退出

本节介绍了如何安装、启动与退出系统。这些工作是学习应用Visual FoxPro的第一步，即使你的系统已经安装完毕，你也可以再次运行setup程序来重新增加新的内容。

2.2　Visual FoxPro 6.0系统的配置

2.2.1　配置Visual FoxPro 6.0

当你打开运行 Visual FoxPro 时，程序读取注册信息，并按照设置配置你的系统。读完配置信息后，Visual FoxPro 读取配置文件，在配置文件中可以改变在注册表中的默认值。Visual FoxPro 启动后，可以用选项对话框或者 Set 命令进行设置。下面是 Visual FoxPro 启动与配置的过程：

开始　Visual FoxPro 启动并读取注册表，如图 2.5 所示。

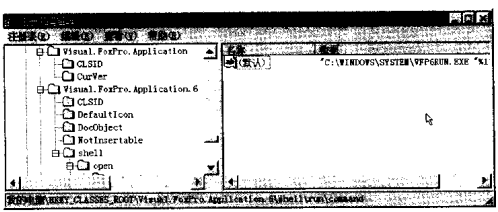


图2.5　Visual FoxPro 6.0 启动并读取注册表

下一步　可以通过配置文件修改配置信息或通过“选项”卡进行修改，见图 2.6 所示。

完成　在“选项”卡选择“确定”设置为当前配置，选择“设置为默认值”设置为默认配置。

注　对运行时间（run-time）版的 Visual FoxPro，启动时它并不读取 Windows 注册表。如果你试图用运行时间（run-time）库发布你的产品，你可以通过两种方法来产生配置信息：通过一个配置文件，或通过一个程序来修改用户计算机上的注册信息。

技巧　打开并运行 Visual FoxPro SAMPLES\CLASSES 目录下的程序 REGISTRY.PRG，里面包含了大量的方法，它们基于 Windows API 的调用，使你能方便地操纵 Windows 注册表。

Visual FoxPro 包含一个资源文件 FOXUSER.DBF，它存储了诸如当前命令窗口的大小和位置、当前的工具条显示方式等信息。FOXUSER.DBF 是一个普通的 Visual FoxPro 数据表，你可以根据你应用的需要加以修改。

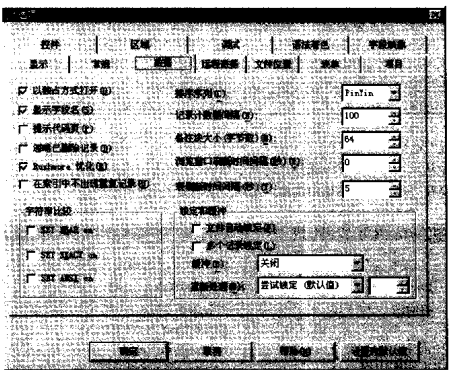


图2.6　通过选项修改Visual FoxPro 6.0 配置信息

说明　系统配置让你有权决定你的 Visual FoxPro 的外观和行为，你可以设定文件的缺省位置和路径，源代码在编辑窗口的外观，以及日期、时间的格式定义。这种配置可以是暂时的或永久的，对于暂时的配置，它们存于内存中，并在退出时失效。永久配置将修改 Windows 的注册表。

注册表是 Windows 保存配置信息的数据库，包括操作系统、应用程序、OLE、ODBC 等信息。同样 Visual FoxPro 也将应用的配置信息存于注册表中。

2.2.2　通过“选项”卡配置系统

Visual FoxPro 6.0“选项”对话框包含了12个选项卡，分别对应了不同的设置。表2.1列出了各选项的功能。

表2.1　选项对话框的选项及功能

选项卡	访问的功能
显示	选择界面形式
数据	对数据库进行设置
常规	设置数据输入和程序设计选项
远程数据	与远程数据库的连接选项
文件位置	设置文件的位置，如目录、位置等
表单	表单设计器选项

续表

选项卡	访问的功能
项目	项目管理器选项
控件	可视类库和 OLE、ActiveX 选项
区域	时间、日期、货币格式选项
调试	调试窗口选项
语法着色	编辑窗口显示方式
字段映像	设置不同字段的编辑控制的编辑模式

“常规”选项卡设置过程：

开始 从“工具”菜单中选择“选项...”。

下一步 在“选项”对话框中选择“常规”项，如图 2.7 所示。

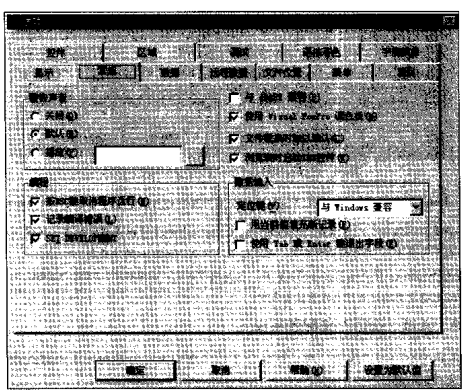


图2.7 通过选项卡修改常规配置信息

下一步 在“常规”选项卡中修改“数据输入”或“编程”选项。

完成 选择“确定”为临时设置、选择“取消”取消当前设置、选择“设置为默认值”将设置写入系统注册表作为永久设置。

本节论述如何将 Visual FoxPro 设置为你所喜爱的形式，随着学习的不断深入，你对这方面的要求会越来越多，之所以放到最前面来讲述，是为了提醒你，在 Visual FoxPro 6.0 的世界里，你是“自由的”。

2.3 Visual FoxPro 6.0的系统结构

2.3.1 整体结构与功能

Visual FoxPro 提供了如下的强大的功能：

(1) 功能强大的项目和数据库管理功能。可以应用源代码管理的产品，如“Microsoft Visual SourceSafe”，数据库容器允许多个用户在同一数据库里创建、修改对象。通过查看数据库设计器能方便地浏览数据库中的对象。数据引擎支持对规则的触发，“悲观”式的记录缓冲能更有效地实现记录锁定，NULL现在能作为关键字使用。

(2) 对调试工具作了改进。可以更为有效地对应用组件进行跟踪监视。你可以选择在Visual FoxPro的主窗口，或者选择用分开的调试窗口，从而不与你的工作空间发生干扰。新的调试工具可以设置断点、跟踪事件、载入执行代码，更接近于Visual C 的调试环境。为了观看变量的值，你只要将光标放在跟踪窗口的变量名上面。

(3) 更简单的表设计器和扩充的数据字典功能。表设计器在你创建字段的同时能方便地建立索引，设置各种默认值，使设计变得更为快速简单。当你将字段加入表单时，能立即产生你所需要的控制。而输入掩码（InputMask）和格式（Format）属性使你能更好地控制你数据的显示方式。一个与ODBC管理器相结合的，提供了优化功能的连接设计器使连接变得更为简单。

(4) 功能强大的查询和视图设计。在查询和视图设计器中可以定义外连接、为列设置别名（aliases）、用白分比来选择记录。在视图设计器中，你可以定义一个默认的控制类、输入掩码和格式。如果你用游标（cursor）选择记录，可以用新的NOFILTER创建一个物理文件，可被后面的了查询加以引用。以前，如果查询有过滤（filter），游标便不能被引用。

(5) 设计表单变得更容易，功能变得更强大。随着数据字典功能的增强，表单设计器的功能更加强大。可以采用单文档（SDI）和多文档（MDI）两种方式进行设计。应用SDI可以创建作为窗口桌面子窗口的应用窗口。一个新的支持热键的菜单设计器，可以设计你的热键菜单，将它用于右击鼠标的事件处理中。表单和控制增加了属性和方法，使得对表单外观与功能的控制更为强大。你可以选择一组控制，在属性窗口定义或改变它们的公共属性。而垂直、水平对中等对齐属性使控件更容易布置。键盘的导航在属性等窗口的应用使得不同对象间的变换变得更为快捷。新的编辑器对代码提供了格式控制、对不同代码可以选择不同的颜色，具有更强地查找和替换功能。

(6) 功能强大的向导功能。应用向导功能使你的应用程序的建立变得易如反掌。“应用程序向导”（Application Wizard）帮助你建立应用程序。“交叉表向导”（Cross-tab Wizard）帮助建立表的查询。“文档向导”（Documenting Wizard）帮助从你的工程和程序中产生格式化的文本文件。“表单向导”（Form Wizard）可以产生表单。“图形向导”（Graph Wizard）产生图形。“报表向导”（Group/Total Report Wizard）设计报表。“导入向导”（Import Wizard）用来导入添加数据。“标签向导”（Label Wizard）产生标签。“本地视图向导”（Local View Wizard）帮助建立本地视图。“邮件合并向导”（Mail Merge Wizard）实现邮件的合并。“一对多表单向导”（One-To-Many Form Wizard）建立一对多表单。“一对多报表向导”（One-To-Many Report Wizard）建立一对多报表。“Oracle 升级向导”（Oracle

Upsizing Wizard）将Visual FoxPro数据库转化为具有相同结构的Oracle数据库。“查询向导”（Query Wizard）定义查询。“远程视图向导”（Remote View Wizard）建立远程视图。“报表向导”（Report Wizard）产生报表。“安装向导”（Setup Wizard）产生供发布的磁盘。“SQL Server 升级向导”（SQL Upsizing Wizard）将Visual FoxPro数据库转化为具有相同结构的SQL Server数据库。“表向导”（Table Wizard）产生表。“WWW 查询页向导”（WWW Search Page Wizard）从你的Visual FoxPro表中产生可供查询的Web页。

(7) OLE 和ActiveX功能。Visual FoxPro现在可以作为OLE服务器，因而其它程序可以很容易地使用Visual FoxPro的强大功能。LSimpleFrame扩展支持更为广泛的ActiveX控件。Visual FoxPro同时支持你自己的本地或远程的OLE服务器应用。

(8) 新增加的应用例子Solutions集成了各种应用组件，其中各种组件的例子你都可以直接用于你的应用系统。

2.3.2 使用集成环境

如图 2.8 为 Visual FoxPro 的集成环境。

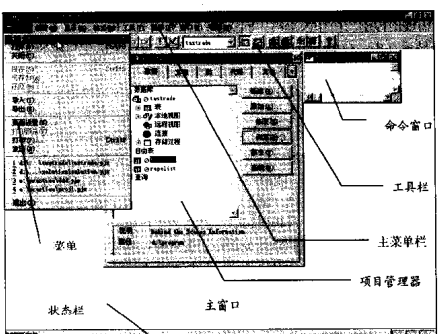


图2.8　Visual FoxPro 6.0 集成环境

说明　集成环境主要包括对以下几部分内容：

主窗口外框：管理 Visual FoxPro 主窗口的控制菜单，及最大、最小化和关闭按钮。

主菜单栏：主窗口最上的主菜单栏可以完成用户的大部分功能和操作。

主窗口：显示输出结果。

命令窗口：输入并执行 Visual FoxPro 的命令、函数。

状态栏：显示当前操作的状态，各种菜单、控件的功能说明。

工具栏：以图标的方式来执行对应的菜单命令。

2.3.3　使用菜单系统

Visual FoxPro 的大部分功能可以通过菜单系统来实现，工具栏和菜单系统相结合完整的体现了 Visual FoxPro 的强大功能，下面分别加以介绍。

1. 文件菜单(如图2.9所示)

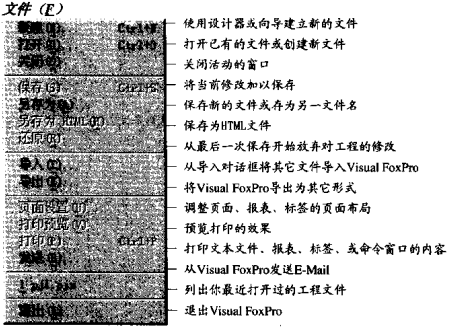


图2.9　文件菜单

2. 编辑菜单(如图2.10所示)

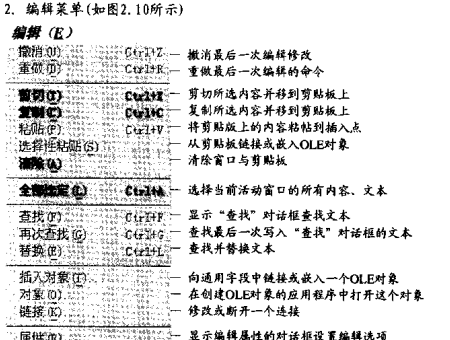


图2.10　编辑菜单

3. 显示菜单(如图2.11所示)

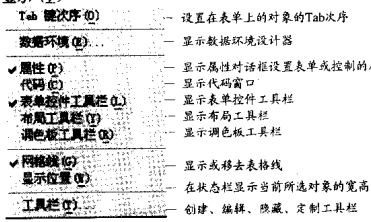


图2.11　显示菜单

4. 格式菜单(如图2.12所示)

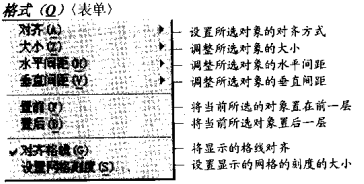


图2.12　格式菜单

5. 工具菜单(如图2.13所示)

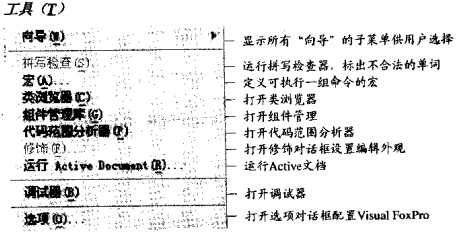


图2.13　工具菜单

6. 格式菜单(如图2.14所示)

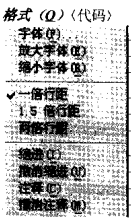


图2.14　格式菜单

7. 程序菜单(如图2.15所示)

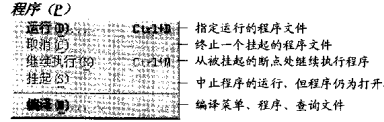


图2.15　程序菜单

8. 窗口菜单(如图2.16所示)

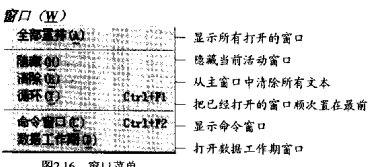


图2.16　窗口菜单

9. 帮助菜单(如图2.17所示)

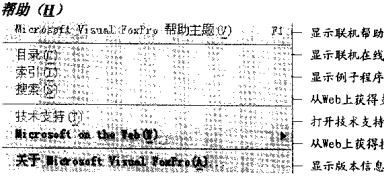


图2.17　帮助菜单

技巧　选择菜单和菜单命令的方法有两种：

(1) 用鼠标。把鼠标移动到菜单的标题上单击，然后移动鼠标到所需的菜单命令上单击。

(2) 用键盘。按 Alt 键，同时按下菜单标题后面的字母，用光标键移动光标，在需要的菜单命令上按 Enter 键。

2.3.4　使用工具栏

在工具栏上显示的按钮，往往代表了最为常用的命令，有效地利用工具栏，能大大方便程序开发工作，Visual FoxPro 为我们提供了十几个工具栏，这些工具栏可以随时打开和关闭。

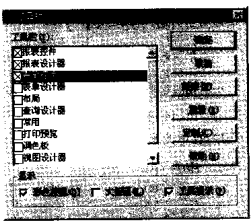


图2.18　工具栏对话框

打开工具栏的方法：

开始　从“显示”菜单中选择“工具栏...”，显示如图2.18所示的工具栏对话框。

完成　在“工具栏”对话框中选择要加入的工具栏，在其前打×，按“确定”按钮完成设置。

技巧　工具栏可以显示为窗口形式，只须将鼠标放在工具栏上而不是按钮上，按鼠标左键，将鼠标“拖拽”下来，并可以将工具栏放在任何位置，如图 2.19 所示。将鼠标在按钮上放一会，该按钮的说明文字便会出现。

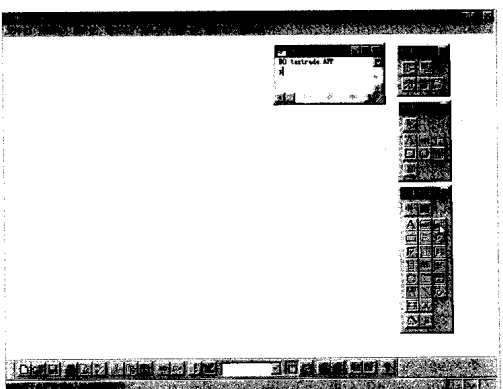


图2.19　自己确定工具栏位置

2.3.5　自己定制工具栏

Visual FoxPro 提供的工具栏上的按钮，我们虽然不能删除或添加，但可以根据任务的需要将它们重新组合，创建出自己的工具栏，创建方法如下：

图2.20　新工具栏对话框

开始 从“显示”菜单中选择“工具栏...”，显示如图 2.20 所示的工具栏对话框。

下一步 选择“新建”按钮，在新工具栏对话框中输入新工具栏的名称，如图 2.21 所示，然后选择“确定”。

下一步 选择合适的按钮，将它拖拽到“我的新工具栏”中，如图 2.21 所示。

下一步 选择“关闭”完成新工具栏的定义。

完成 再次进入“工具栏”对话框，选择“我的新工具栏”，按“删除”删除已经不需要的自定义工具栏。

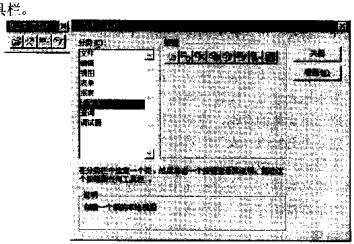


图2.21　拖拽生成自定义工具栏

2.3.6　使用命令窗口

在 Visual FoxPro 里，主菜单所列出的只是最为常用的命令，而 Visual FoxPro 提供的命令窗口，能直接接受用户输入的命令，成为你与 Visual FoxPro 交流的主要渠道。

当你选择主菜单时，所选择的选项对应的命令行在命令窗口中显示出来。自从你进入 Visual FoxPro 系统，你从主菜单或命令行窗口中输入的命令在退出系统以前具有有效性。用户只需要将光标移动到命令之上，然后按 Enter 键，所选命令将再次执行。用户也可以在命令窗口中将以前输入的命令加以修改，然后再次执行。

在命令行窗口中，可以使用 Visual FoxPro 的命令和函数，因而熟练地掌握 Visual FoxPro 的命令与函数将大大提高你的工作效率。

使用命令行窗口的操作步骤如下：

开始 将光标移到命令窗口，使其成为当前活动窗口。

下一步 在命令窗口中输入 DIR，按 Enter 键。

下一步 在命令窗口中输入 CLEAR，按 Enter 键。

下一步 选择“帮助”菜单的“示例应用程序”项，选择 Tasmanian Traders 的 Open 选

择项。

下一步 在窗口中输入 select * from category where category_name='Seafood'，按 Enter 键。

下一步 关闭查询窗口。

下一步 在命令窗口中输入?date()，按 Enter 键。

完成 在命令窗口中输入 CLEAR，关闭“项目管理器”。

技巧 当你编制应用程序时，有些程序并不一定等编完了再进行调试，可以在命令窗口中先验证程序中的语句是否正确。这种方法尤其适用于查询语句。当通过验证，证明语句是正确的，就可以将验证过的语句剪切复制到代码窗口。

2.3.7 使用设计器

通过“项目管理器”你可以很快地进入 Visual FoxPro 的设计器，这些设计器帮助你快速地建立“表”、“表单”、“数据库”、“查询”、“报表”等。你也可以从主菜单的“文件”选项下选择“新建”进入相关的设计器，下面介绍 Visual FoxPro 中设计器的种类和功能。

表设计器：建立“表”，为“表”建立“索引”。

查询设计器：为本地表建立、运行“查询”。

视图设计器：建立可更新的查询，建立运行对远程数据源的访问。

表单设计器：建立查看修改“表”中数据的“表单”。

报表设计器：为显示打印数据建立“报表”。

数据库设计器：建立数据库，为数据库中的表建立“关系”。

连接设计器：为远程视图建立一个连接。

2.3.8 使用向导

“向导”能帮助你实现日常的工作，如建立表单、格式化报表，建立查询等。通过在向导对话框中回答一些问题或选择一些选项，向导会按照你的要求建立一个文件或执行一项任务。例如当你选择了“报表向导”时，你可以选择需要创建的“报表”类型，然后会询问需要加入的“表”，并可以选择输出的格式。下面介绍 Visual FoxPro 中向导的使用。

使用向导的操作步骤如下：

开始 在“项目管理器”选择你想建立的文件类型，然后选择“新建（N）...”按钮。

下一步 也可以在主菜单中选择“文件”菜单中的“新建”，然后选择新建的文件类型。

完成 选择“向导”按钮进入相应的向导对话框。

说明 在进入向导对话框后，只需要按顺序回答在每一屏幕上出现的问题，然后选择“下一步”进入下一屏。如果你输入的有错误，或对先前的输入需要改动，可以按“上一步”按钮回到原来的屏幕进行更正。选择“取消”将放弃当前向导产生的结果。如果在执行中出现了问题，可以按 F1 取得在线帮助。当到了向导的最后一屏时，选择“完成”便可以实现你的要求了。

技巧 在进入向导的最后一屏时，根据你所选的向导的类型，可能有选项让你保存、浏览、修改或打印等选项。用“预览”功能可以时你在完成向导之前看到结果。如果你想进行不同的选择，你可以在一次

运行向导。当你对向导产生的结果满意后，就可以按“完成”按钮了。

2.3.9　使用生成器

“生成器”是一些带选项卡的对话框，用来简化对“表单”、一些“控制”及参考代码的修改。每个生成器有若干选项卡，帮助你设置所选对象的属性。下面介绍 Visual FoxPro 中“生成器”的使用。

1. 使用“表单生成器”

开始从主菜单的“表单”项选择“快速表单（Q）...”，显示图 2.22 所示的“表单生成器”的对话框。

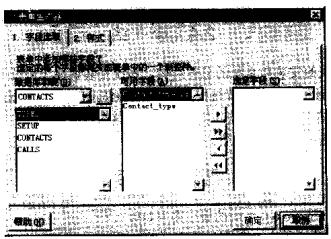


图2.22　表单生成器

下一步在“表单生成器”对话框中输入表单的属性。

完成选择“表单生成器”的“确定”或“取消”按钮。

2. 使用“自动格式生成器”

开始从“表单”设计器的“表单”上选择需要定制的控制。

下一步在“表单工具栏”上选择自动格式按钮，出现自动格式生成器对话框，如图 2.23 所示。

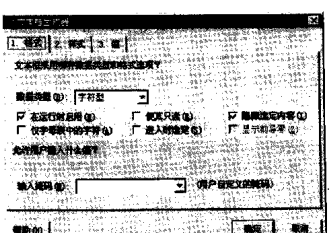


图2.23　文本框自动格式生成器

完成 选择“自动格式生成器”的“确定”或“取消”按钮。

说明 在 Visual FoxPro 中的“生成器”有以下几种：

复合框生成器、命令框生成器、编辑框生成器、表单生成器、列表框生成器、选项生成器、表格生成器、文本框生成器、自动格式生成器、参照完整性生成器。

本节介绍了 Visual FoxPro 系统的框架结构，其中的有些名词如 ActiveX、OLE、客户机/服务器等对有些人来说可能比较生疏。如果这样先不去管它，你渐渐就会明白。本章对 Visual FoxPro 6.0 所提供的功能作了总体上的叙述，对在应用中用到的一些知识和内容都做了简单的提及。看完本章后，你大概已经明白 Visual FoxPro 6.0 的功能与系统的特点了，在下面的篇章中将详细论述应用系统的构造。

2.4 小 结

本章论述了系统的安装和系统的总体结构。系统安装讲述了系统的软硬件需求和安装过程，系统的配置讲述了系统的配置过程、方法及应该注意的地方，系统的结构讨论了 Visual FoxPro 6.0 在整体结构、集成环境、菜单系统、工具栏、命令窗口、设计器、生成器和使用向导等方面的功能及所包括的内容。

思考与练习

1. 安装 Visual FoxPro 6.0 系统的最小需求是什么？
2. “选项”卡有何作用？如何用选项卡来配置系统？
3. 启动 Visual FoxPro 6.0 对照屏幕指明 Visual FoxPro 集成环境主要包括哪几部分？
4. 如何使用工具栏？什么是向导和生成器，如何使用它们？

第二篇

基 础 篇

本篇导读

本篇介绍了 Visual FoxPro 6.0 程序设计的基础内容，主要介绍 Visual FoxPro 6.0 的功能强大的数据库管理功能，数据库信息的显示与输出。这些数据库的基础知识都是进行 Visual FoxPro 设计的基础知识，学会使用这些功能，用户通过使用 Visual FoxPro 数据库的管理功能便可以执行简单的应用。Visual FoxPro 6.0 中对于数据库的管理都有相应的可视化的操作界面，而对于那些习惯于使用命令的人可以通过“命令窗口”实现对数据库的操作。

本篇主要讲述：

第 3 章论述了 Visual FoxPro 6.0 的可视化的数据库管理，如何通过“项目管理器”管理数据库，如何建立数据库文件，如何建立表和索引以及如何用数据库的功能更好的管理数据。

第 4 章主要论述了怎样对数据库中的数据进行查询和更新。包括使用“查询”，使用“视图”以及如何实现对多表的查询和更新操作。

第 5 章介绍了数据库中信息的显示和输出。通过使用“报表”，用户可以方便地实现数据库中相关信息的打印和输出，通过“表单”用户可以方便地实现数据库中信息的显示和编辑。

第3章　VFP 6.0 可视化数据库管理

对于用户，数据库的操作在 Visual FoxPro 6.0 中变得十分简单直观，大量可视化的设计器、向导在加上功能强大的项目管理器，使得应用项目的数据库设计、数据表的创建、索引的创建及将所有这些组织进入一个完整的数据库。同时对这些数据的查询、检索，以及报表的生成在 Visual FoxPro 6.0 中都可以通过可视化的工具来实现，本章着重叙述关于项目的建立，数据库的建立，数据表、索引、视图的建立以及如何对它们进行管理。所有这些工作都是建立应用系统的前提条件。

基础数据库的建立就象大厦的地基一样重要，而 Visual FoxPro 6.0 为我们提供了建立地基的很好的工具。

本章的主要内容有：

(1) 使用“项目管理器”管理项目；

(2) 建立数据库文件；

(3) 建立表和索引；

(4) 用“数据库”管理数据。

3.1　使用“项目管理器”管理项目

3.1.1　项目管理器简介

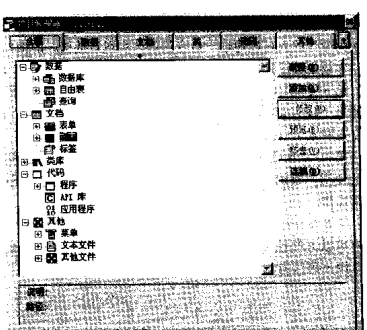


图3.1　Visual FoxPro 6.0“项目管理器”

Visual FoxPro 6.0 的“项目管理器”的组织结构如图 3.1 所示，它为我们提供了处理数据和对象的工具。通过“项目管理器”，你能很快地熟悉 Visual FoxPro 的世界。“项目管理器”提供了简单的、可视化的数据组织和编程环境。通过“项目管理器”能方便地实现对数据表、表单、数据库、报表、查询以及相关文件的管理。

说明　一个项目是文件、数据、文档的集合，Visual FoxPro 的对象被存于具有后缀 .PJX 的文件当中，下面就各部分功能加以介绍。

3.1.2　管理项目的“数据”

项目管理器的“数据”卡负责管理项目的数据库、自由表、查询和视图等数据内容，见图 3.2 所示。

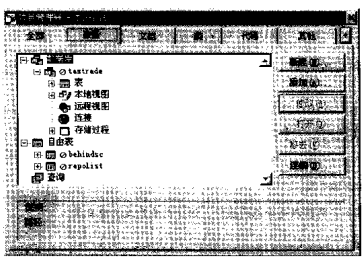


图3.2　“项目管理器”的“数据”卡

下面简要说明它们的组成：

数据库：它由数据表组成，它们通常由公共的字段建立相互关系。为了支持这些“表”和“关系”，你也可以在数据库建立相应的“视图”、“连接”、“存储过程”、“规则”和“触发器”。用“数据库设计器”可以建立数据库，在数据库中加入“表”。数据库文件的后缀为.DBC。

自由表：它不是数据库的一部分，存于后缀为.DBF 的文件里，如果需要可以将自由表加入数据库。

查询：它是用来实现对存于表中的特定数据的查找。通过查询设计器，你可以按照一定的查询规则从“表”中得到数据。采用 SQL-Select 的查询可以存于后缀为.QPR 的文本文件中。

视图：它执行特定的查询，从本地或远程数据源中获取数据，并允许你对所返回的数据进行修改。视图依数据库而存在，并不是独立的文件。

3.1.3　管理项目的“文档”

如图 3.3 所示，项目管理器的“文档”卡负责对数据的所有文档的管理。如用于数据

输入浏览的“表单”，用于打印查询结果的报表和标签。

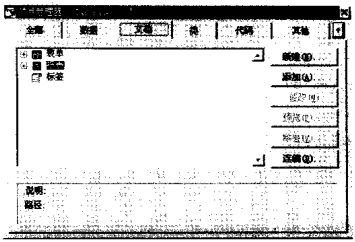


图3.3　“项目管理器”的“文档”卡

表单：显示和修改数据“表”中的内容。可以用“表单设计器”设计表单，从而实现对数据的管理。

报表：“报表”文件实现对 Visual FoxPro 数据表查询结果的格式化打印输出。用“报表设计器”可以实现对报表和标签的设计。

说明　其它选项，如“类”、“代码”、“其它”用于产生针对最终用户的应用程序。它们的用法将在以后的“程序设计”中加以介绍。

技巧　“项目管理器”的组织是分级的组织，你可以展开或折叠目录级别，如果在某一项的下面还有其它项，这一项有“+”的标志。点击“+”使展开此项的所有内容。例如当点击“自由表”前的“+”号时便会将此项目中的所有“自由表”列出。

建立一个新的项目的操作步骤如下：

开始　从“文件”菜单中选择“新建”。

下一步　选择“项目”然后选择“新建文件”，如图 3.4 所示。

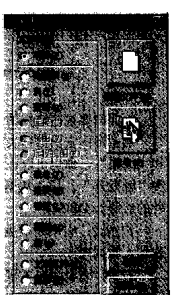


图3.4　新建“项目”

下一步　在“创建”对话框中输入“项目”名称“我的项目”。

完成　在“创建”对话框中选择“保存”完成创建。

3.1.4　在“项目管理器”中添加、移去文件

从“项目管理器”中可以向项目中加入或移去一个已有的文件。

1. 向项目中加入一个新文件

开始　在“项目管理器”中选择你想加入的文件类型，如“自由表”。

下一步　单击“项目管理器中”中的“添加（A）...”按钮。

下一步　在“打开”对话框中输入或选择要加入的“表”的名称。

完成 在“打开”对话框中选择“确定”完成添加。

2. 从项目中移去一个文件

开始 在“项目管理器”中选择你想删除的文件，如“自由表”下的一个表。

下一步 单击“项目管理器中”中的“移去（V）...”按钮，见图3.5所示。

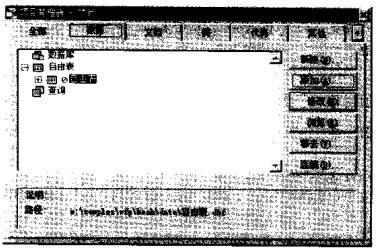


图3.5　从“项目管理器”中删除文件

完成 从所显示的对话框中选择“移去（r）”按钮，如果你想从计算机中删除文件，选择“删除”，如图3.6所示。

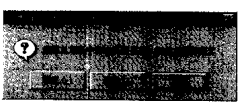


图3.6　从“项目管理器”中“移去”或“删除”文件

注意 当加入一个文件时，这个文件必须是“自由”的，即没有被其它数据库引用，否则会提示报错信息，此时需要将它先从它所属的数据库中移出，才能将它加入当前的数据库中。

3.1.5　在“项目管理器”中新建、修改文件

“项目管理器”使文件的“新建”和“修改”变得简单易行，只要选择所要创建的文件类型，然后按“新建”或“修改”，Visual FoxPro 帮助你选择合适的设计器，使你方便地完成设计任务。

1. 在项目中“新建”一个文件

开始 在“项目管理器”中选择你想加入“新建”的文件类型，如“数据库”。

下一步 选择“新建（N）...”按钮。

完成 按照所出现的设计器类型“新建”文件。

对于已经建立的文件可以通过“项目管理器”进行修改。

2. 用“项目管理器”修改一个文件

开始 在“项目管理器”中选择你想修改的文件，如“自由表”下的一个表。

下一步 单击“项目管理器中”中的“修改（M）”按钮。

完成〉按照所出现的设计器类型“修改”文件。

3.1.6　“浏览”项目中“表”的数据

在“项目管理器”中可以方便的浏览“表”或“自由表”中的数据。

浏览“表”中数据的操作步骤如下：

开始〉在“项目管理器”中选择你想浏览的数据“表”。

下一步〉选择“浏览（B）”按钮显示浏览结果。

下一步〉关闭查询结果窗口完成浏览。

完成〉返回“项目管理器”。

3.1.7　“项目信息”的显示与编辑

每一个“项目”都有说明性的文件，它记载了“项目”的作者、路径、所包含的文件和服务器的信息。

显示修改“项目信息”的操作步骤如下：

开始〉在主菜单中选择“项目”，选取“项目信息（I）...”，显示如图 3.7 所示的“项目信息”对话框。

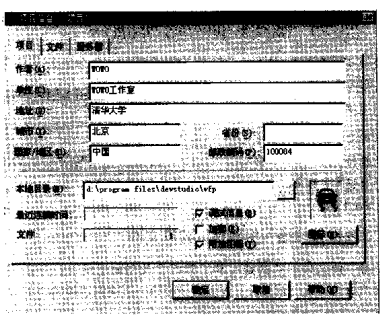


图3.7　“项目信息”对话框

下一步〉选择相应的选项卡，浏览或编辑项目信息。

完成〉按“确定”按钮完成整个项目信息的设置。

3.1.8　给“项目”中的文件添加说明

可以给“项目”中的文件添加简单的说明，从而为整个项目的管理带来方便。这样在几年以后，只要打开项目，每个文件的功能便一清二楚。

给项目文件添加说明的操作步骤如下：

开始〉在“项目管理器”选择所要加入说明的文件。

下一步 在主菜单中选择“项目”或按鼠标右键显示快捷键，选取“编辑说明（T）...”，显示见图 3.8 所示的“说明”对话框。

下一步 输入或修改文件的说明。

完成 按“确定”按钮完成操作。

图3.8　“说明”对话框

3.1.9　不同“项目”之间的文件共享

通过对不同“项目”的文件实现共享，可以实现代码和文件的复用。这些文件并没有被复制，因为项目只是存储了对文件的引用。一个文件可以同时被多个项目所引用。

实现项目间的文件共享的操作步骤如下：

开始 从“文件”菜单同时打开具有共享文件的两个项目。

下一步 从“项目管理器”中选择需要共享的文件。

完成 将所选择的文件“拖拽”到另一个项目的“项目管理器”的相关位置上。

3.1.10　定制自己的“项目管理器”

“项目管理器”是作为一个独立的窗口存在的。根据用户的不同需要，可以移动它的位置，改变它的大小与外观，也可以将它打开或折叠起来。

定制“项目管理器”的操作步骤如下：

开始 单击“项目管理器”右上角的上箭头，如图 3.9 所示，“项目管理器”被折叠起来。

图3.9　折叠后的“项目管理器”

下一步 选定“项目管理器”的一个选项卡，将此卡脱离“项目管理器”，如图 3.10。

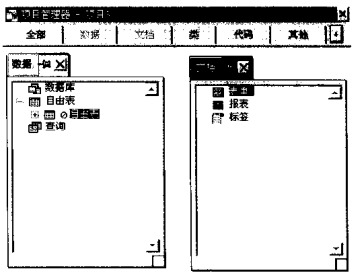


图3.10　“项目管理器”选项卡的脱离

技巧　当选项卡处于浮动状态时，在选项卡中单击鼠标右键可以访问快捷菜单中的菜单项，单击选项卡上的图钉图标，可以使该选项卡始终至于最前端，多个选项卡可同时具有“最前状态”。再次单击图钉图标可以取消“顶层显示”的设置。

下一步　单击“项目管理器”右上角的下箭头，“项目管理器”被还原。

完成　按“确定”按钮完成整个项目信息的设置。

3.2　建立数据库文件

3.2.1　建立数据库文件

上一节建立了项目文件我的项目.PJX，并熟悉了 Visual FoxPro 的环境。在 Visual FoxPro 中，“数据库”是存储数据的地方。

项目中建立数据库的步骤如下：

开始　在“文件”菜单中选择“打开”。

下一步　在“打开”对话框中输入“我的项目”，按“打开”。

下一步　在“资源管理器”中选择“数据库”，然后按“新建”。

下一步　在“创建”对话框中输入“业务”，然后按“保存”，出现“数据库设计器”，如图 3.11 所示。

完成　关闭“数据库设计器”完成数据库创建。

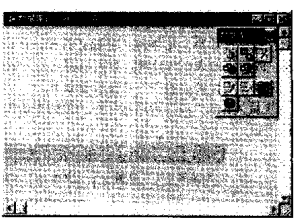


图3.11　“数据库设计器”

3.3　建立表和索引

上一节在“我的项目”中建立了数据库“业务”。在 Visual FoxPro 中，“表”是组织数据的最基本的单位。本节讲述如何用“表”来组织数据，以及学习“表”的建立和“表”的修改等内容。

3.3.1　创建表

注意　字段是组成表的关键，在创建新表之前要注意进行以下工作：

(1) 仔细分析你将要用表所收集的数据，确定各表的字段。每个表的字段应同属于同一范畴，不同范畴的信息应该属于不同的表。

(2) 争取先整理好你所需要的全部信息，然后进行表的划分，对那些通过计算可以得到的结果不要再分配字段。

(3) 表中要具有唯一标识单个记录的关键字段，或字段的组合。

(4) 根据字段的内容仔细确定每个字段的数据类型。

说明　Visual FoxPro 中的字段的数据类型、它们的示例如表 3.1 所示。

表3.1　数据类型

数据类型	中文名称	说明	示例
Character	字符型	字母、数字文本	住址
Currency	货币型	货币单位	购买价格
Numeric	数值型	整数或小数	购买数量
Float	浮点型	同数值型	
Date	日期型	年、月、日	出生日期
DateTime	日期时间型	年、月、日、时、分、秒	工作时间
Double	双精度型	双精度数值	实验数据
Integer	整型	整数	职员数量
Logical	逻辑型	真或假	定单是否用完
Memo	备注型	不定长的字母、文本、数字	职员简历
General	通用型	OLE	Word文档
Character（Binary）	字符型（二进制）	同前字符型，但是当代码页更改时字符值不变	保存在表中的口令，用于不同国家
Memo（Binary）	备注型（二进制）	同前备注型，但是当代码页更改时备注值不变	用于不同国家的登录脚本

下面假设要创建一个关于“客户”的表，它具有的字段及其类型如下：

客户代号——字符型　　公司名称——字符型　　联系人——字符型
联系人职务——字符型　　地址——字符型　　城市——字符型
邮政编码——字符型　　国家——字符型　　电话——字符型
传真——字符型　　销售额——货币型　　销售区域——字符型
公司简介——备注型

创建新“表”的途径有两条：通过“表向导”或“表设计器”。

1. 使用表向导创建表

开始　在“项目管理器”选择“自由表”。

下一步　选择“新建”。

下一步　选择新建表对话框中选择“表向导（W）”，如图 3.12 所示。

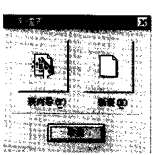


图3.12　选择“表向导”

下一步　按照“表向导”提供的提示信息一步一步建立“表”，必要时按 F1 取得帮助。

完成　执行完“表向导”。

2. 用表设计器创建表

开始　从“文件”菜单中选择“新建”，或从常规工具栏选择新建按钮。

下一步 在“新建”对话框中选择“表”，然后选择“新建文件（N）...”。

下一步 在“创建”对话框中输入表名“客户”，然后选择“保存”。

下一步 在显示的“表设计器”中的“字段名”下面键入“客户代号”，这时的“类型”默认值为“字符型”，选择默认值。在“宽度”下输入“8”。索引采用“升序”。其它字段按表 3.2 输入。

表3.2　客户表结构

字段名	类型	宽度	小数位数	索引	NULL
客户代号	字符型	8	无	升序	否
公司名称	字符型	50	无	无	否
联系人	字符型	15	无	无	否
联系人职务	字符型	12	无	无	可
地址	字符型	50	无	无	否
城市	字符型	10	无	无	否
邮政编码	字符型	10	无	无	否
国家	字符型	10	无	无	否
电话	字符型	10	无	无	否
传真	字符型	10	无	无	可
销售额	货币型	8	无	无	可
销售区域	字符型	10	无	无	可
公司简介	备注型	4	无	无	可

下一步 输入完所有字段后，按“确定”按钮，如图 3.13 所示。

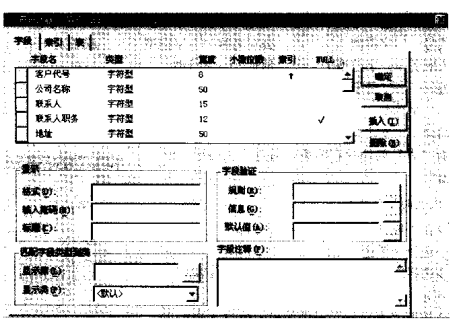


图3.13　用“表设计器”建立表

说明 如果“类型”为“浮点型”或“数值型”，可以在“小数位数”的地方确定小数位。如果需要加入“索引”，在“索引”部分加入排序方式。如果需要输入为“空值”，选择“NULL”项。

下一步 在“是否输入数据”的对话框中选择“是”。

下一步　在“是否输入数据”的对话框中选择“是”。

下一步　在出现的输入界面中可以输入数据，见图 3.14 所示。

完成　关闭输入数据的对话框完成。

说明　“表”是以“行”和“列”的格式来存储你的数据，就如同日常用的工作表格。“表”中的列用字段来表示，而每一行表示了一条记录。在 Visual FoxPro 中可以创建两种“表”：数据库“表”和“自由表”

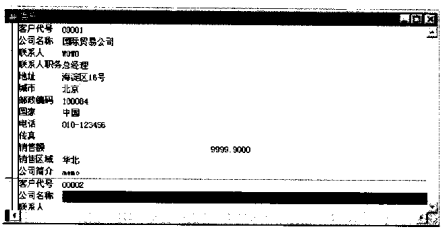


图3.14　向“表”中输入数据

3.3.2　浏览表中的数据

在建“表”的过程中，可以选择输入数据，也可以在以后向表中输入数据。

1. 浏览“表”并向“表”中加入数据

开始　从“文件”菜单中选择“打开”。

下一步　在“打开”的对话框中输入“我的项目.pjx”，按“打开”。

下一步　选择“数据库”的“业务”下面的“表”并选择“客户”，然后选择“浏览”，如图 3.15 所示。

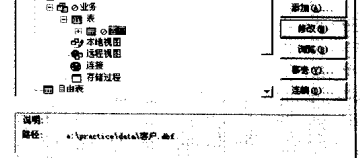


图3.15　“浏览”表中数据数据

下一步 在出现的“客户”窗口中，用户可以浏览或编辑添加数据。

完成 关闭“客户”窗口完成浏览数据。

说明 浏览“表”中的数据也可以从主菜单中选择“打开”你所要浏览的数据表，然后选择“浏览”，便可以查看并修改数据。

技巧 在浏览状态下，浏览窗口中的每一列表示了数据库中的一个字段，每一行表示一条记录，用浏览窗口中的滚动条可以滚动数据，用上、下、左、右箭头键可以实现光标在各编辑框中的移动。浏览窗口有“浏览”、“编辑”和“追加”三种种状态，在主菜单的“显示”项可以实现这三种状态的切换，三种状态的意义如下：

浏览：在浏览窗口中显示“表”中的数据，此时允许你查看和修改表或视图中的数据。这种对以浏览、编辑、追加方式的视图有效。它的窗口如图 3.16 所示。

客户代号	公司名称	联系人	联系人职务
00001	国际贸易公司	WDO	总经理
00002	北京WDO建筑公司	WDO	业务员

图3.16 “浏览”窗口的“浏览”状态

编辑：显示编辑所选的表或视图，这种对以浏览、编辑、追加方式的视图有效。它的窗口形式如图 3.17 所示。

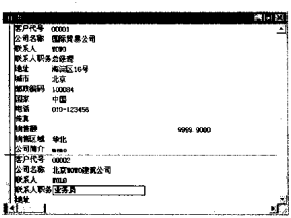


图3.17 “浏览”窗口的“编辑”状态

追加：当选择此项时自动在当前编辑的表或视图的末尾添加一条空记录，你可以在“浏览”或“编辑”两种方式下追加数据，这种对以浏览、编辑、追加方式的视图有效。

2. 在“表”中的数据记录间移动

你可以用滚动条、箭头键或 Tab 键实现数据的移动。

开始 从“表”菜单中选择“转到记录（G）...”，见图 3.18 所示。

完成 在下一级子菜单中选择移动方式。

如果选择了“记录号(R)...”会出现如图3.19所示的“转到记录”对话框，输入记录号然后按“确定”按钮即可。

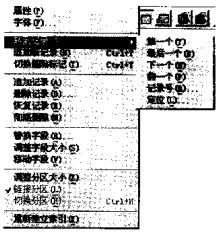


图3.18 “表”菜单

图3.19 “转到记录”对话框

3.3.3 编辑表中的数据

编辑“表”中当前所选的字符型、数值型、逻辑型、日期型或时间日期型的字段信息，只需将光标移到相应的位置输入即可。

编辑备注型的信息，在浏览窗口中双击字段或按CTRL+PGDN键，此时出现一个编辑窗口用来编辑备注字段。

一个用于嵌入和链接OLE对象的通用型字段，在浏览窗口中双击该字段，可以直接编辑所链接的文档，或进入父程序进行编辑。

编辑“表”中数据的操作步骤如下：

开始 从主菜单中选择“帮助”，然后选择“示例程序”。

下一步 在出现的帮助窗口中，选择“Tasmanian Traders”然后选择“Open”。

下一步 在从“项目管理器”中选择“表”employee，单击“浏览”按钮。

下一步 在出现的employee浏览窗口选择所要编辑的字段并编辑。

下一步 在employee浏览窗口中双击Notes字段编辑备注型字段。

下一步 在employee浏览窗口中双击Photo字段编辑通用型字段。

完成 关闭employee浏览窗口。

3.3.4 向表中添加数据

快速地向“表”中添加数据，可以将“浏览”或“编辑”窗口设为“添加方式”在文件的最底部出现一个空白记录。

向“表”中添加数据的操作步骤如下：

开始 按上几节的方法打开“我的项目”。并将“客户”表定为“浏览”方式。

下一步 在主菜单中，选择“显示”然后选择“追加方式”。

下一步 在“浏览”窗口中输入新的记录，按Tab键实现字段间的移动，如图3.20所示。

完成 关闭客户表的浏览窗口。

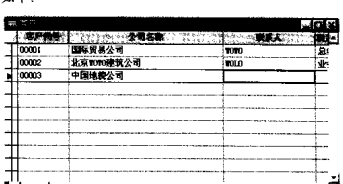


图3.20 “追加”记录

3.3.5　从表中删除数据

从“表”中删除数据分为两步走，首先在表的浏览窗口做出删除标记。然后从“表”菜单中选择“删除”进行真正的删除。

在“表”中删除数据的操作步骤如下：

开始　按上几节的方法打开“我的项目”。并将“客户”表定为“浏览”方式，并加入几条记录。

下一步　用鼠标点击“浏览”记录窗口中所要删除的记录的左面的空白，作出黑色的删除标记，如图 3.21 所示。

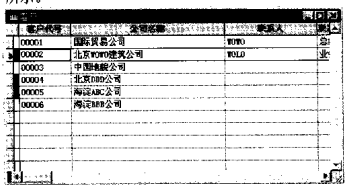


图3.21　为记录做删除标记

下一步　在主菜单中，选择“表”然后选择“彻底删除（M）”，然后在所出现的对话框中选择“是”。

下一步　在主菜单中，选择“表”然后选择“删除记录（D）…”。

下一步　在出现的“删除”窗口中输入删除的规则，如图 3.22 所示，在 For 或 While 处输入删除条件。

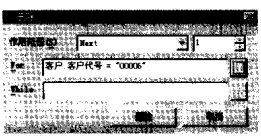


图3.22　“删除”对话框

完成　在“删除”对话框中单击“删除”按钮。

3.3.6　定制浏览数据窗口

按照自己的要求可以重新定制浏览窗口，可以改变列的宽度，显示或隐藏表格线，也可以将窗口分割为两部分。

定制“浏览”窗口的操作步骤如下：

开始　按上几节的方法打开“我的项目”。并将“客户”表定为“浏览”方式。

下一步　用鼠标点击“浏览”记录窗口最上面的标题栏，把鼠标放在所要移动的标题的上面，按下左键，然后拖动此字段到需要的位置，完成字段位置的重新排列。

下一步 在主菜单中，选择“表”然后选择“移动字段”，然后按左、右箭头键实现字段的移动，按 Enter 键完成字段的移动定位。

下一步 将鼠标移到“浏览”记录窗口最上面的标题栏的字段交界位置，当出现单竖线时，按下鼠标左键拖动，可以改变字段的列宽，如图 3.23 所示。

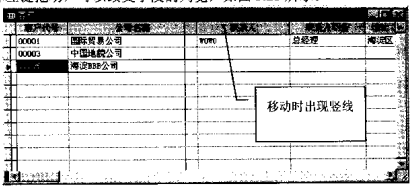


图3.23　改变列的宽度

下一步 在主菜单中，选择“表”然后选择“调整字段大小”，然后按左、右箭头键实现字段宽度的变化，按 Enter 键完成字段的宽度的设置。

说 明 浏览窗口中列的宽度的变化并不是物理数据库中字段的宽度真的变了，而只是一种显示上的变化，要真的实现表中字段宽度的变化需要在表设计器中修改。

下一步 在主菜单中，选择“显示”然后选择“网格线”，可以显示或隐藏浏览窗口中的网格。

下一步 将鼠标移到“浏览”记录窗口左下角的分割窗口工具上，当出现单竖线时，按下鼠标左键拖动，可以将浏览窗口分割为两部分，同时具有浏览和编辑窗口，如图 3.24 所示。

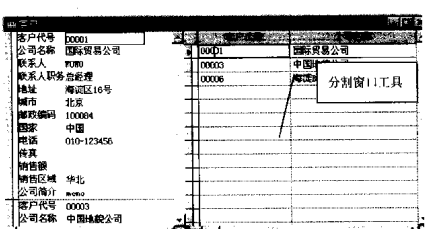


图3.24　改变列的宽度

下一步 在主菜单中，选择“表”然后选择“切换分区”，可以在两个窗口之间切换焦点。

说明　浏览窗口中的两个窗口的默认值是相互关联的，当你选择其中一个窗口中的记录时，另一个窗口所显示的记录也是一样的。要取消这种关联，可以从“表”菜单中选择“链接分区”项，从而可以在链接和非链接状态之间来回切换。

下一步　在主菜单中，选择“表”然后选择“调整分区大小”，然后按住左、右箭头键实现分区宽度的调整。

3.3.7　修改表的结构

当需要对已经存在的数据库表的结构的修改时，可以通过表设计器来实现删除字段改变字段类型和宽度，也可以重新设置索引。如果你正在修改的表是数据库的一部分，可以增加字段并修改属性。

修改“表”结构的操作步骤如下：

开始　从“文件”菜单中选择“打开”。

下一步　在“打开”的对话框中输入“我的项目.pjx”，按“打开”。

下一步　选择“数据库”的“业务”下面的“表”并选择“客户”，然后选择“修改”。

下一步　在“表设计器”中，选择“公司名称”字段，然后将“宽度”改为40。

下一步　在“表设计器”中，选择“销售区域”字段，然后按“删除”，如图3.25所示。

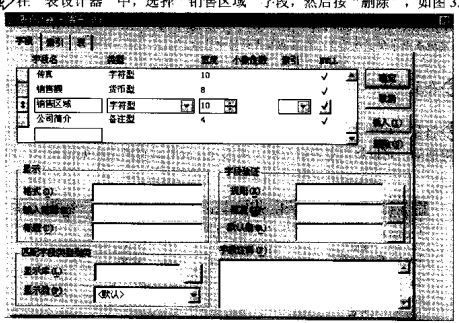


图3.25　修改表的结构

下一步　在“表设计器”中，选择“联系人职务”字段，然后按“插入”按钮，在出现的新字段，依次输入：字段名——联系人性别，类型——字符型，宽度——4。

下一步　在“表设计器”中，按“确定”按钮。

完成　在出现的对话框中选择“是”完成对表结构的更改。

3.3.8　给表加过滤器

你能按照自己的要求重新定制表，使只有符合“过滤”条件的记录才被显示出来。

说明 设置过滤，是为了限制在浏览窗口中显示的记录，当你只想查看销售额大于某一限度的客户时，就可以使用过滤器了。通常过滤器是在“工作区属性”对话框中设置的。

1. 用过滤器限制记录

开始 从主菜单中选择“帮助”，然后选择“示例程序”。

下一步 在出现的帮助窗口中，选择“Tasmanian Traders”然后选择“Open”。

下一步 在从“项目管理器”中选择“表”customer，单击“浏览”按钮。

下一步 在主菜单中选择“表”，然后选择“属性”，出现属性对话框，如图 3.26 所示。

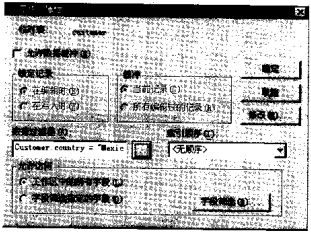


图3.26　设置过滤器

下一步 在“数据过滤器”中输入所要的过滤表达式，如 Customer.country = "Mexico"。

技巧 可以单击“...”按钮，此时进入“表达式设计器” 如图 3.27 所示。先从字段栏选择“country”，然后双击它，在“SET FILTER 表达式：”出现“Customer.country”，然后从“逻辑”处选择“=”，从“字符串”选择“文本”，最后在“”之间输入 Mexico，表达式构造完毕，当然你也可以直接输入表达式。

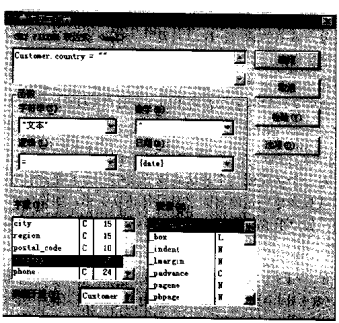


图3.27　用表达式设计器构造过滤器表达式

下一步 在“数据过滤器”对话框中选择“确定”按钮。

完成 此时 customer 浏览窗口中显示的都是国家在"Mexico"的客户。

2. 用过滤器限制字段

如果你只想显示所选字段，可以在过滤器中加以设置。

开始 从主菜单中选择“表”，然后选择“属性”。

下一步 在“工作区属性”对话框，在允许访问处选择“字段筛选指定的字段”，选择“字段筛选”按钮。

下一步 从“字段选择器”中选择所选的字段，然后按“确定”，如图 3.28 所示。

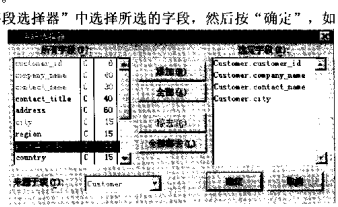


图3.28 用字段选择器选择字段

完成 当再次进入浏览窗口时，便会显示你选择的字段。

3.3.9 给表加索引

当表建成以后，你可以给表添加索引，索引能使你对记录的显示、查询打印更为迅速。你也可以限制记录是否重复，支持不同表之间的“关系”。

索引按照一定的顺序查找记录，从而提高检索速度，它对于在数据库的表之间建立关系也十分重要。就象一本书的索引一样，“表”的“索引”用一系列值记录了特定的记录集。并决定了记录的读取顺序。对一个表你可以建立多个索引，每种索引代表了一种获取数据的顺序。你所建立的索引被保存在复合索引文件里，当“表”打开、更新时，索引同时被打开和更新。索引文件的名字与表名相同，并具有.CDX 后缀。

注 索引的创建十分简单，但请你不要对每个字段都建立索引，否则会降低程序的运行效率，对一些很少用的索引可以考虑用其它索引文件的格式，这在以后会加以叙述。

对于表，可以对一个字段或一个表达式建立索引。为了使索引更为有效，对于那些经常用于对表、视图或报表建立过滤器的字段最好建立索引。如果你对不经常用于过滤或查询的字段建立索引，如“街道地址”，它有可能降低运行效率。

说 明 索引的种类

Visual FoxPro 中有三种索引：结构复合索引（.CDX）、非结构复合索引（.CDX）、独立索引（.IDX）其中结构复合索引是所有索引中最为重要的索引，本书中的索引大多是结构复合索引，它的特点是：

(1) 在表打开时自动打开。

(2) 在同一个索引文件中可以有多种排序方式，具有多个索引关键字。

(3) 在对表进行添加、更改、删除时索引文件自动维护。

建立索引的操作步骤如下：

开始 在“项目管理器”，中选择想要加入字段的“表”，然后选择“修改”。

下一步 在所显示的“表设计器”中，选择“索引卡，如图 3.29 所示。

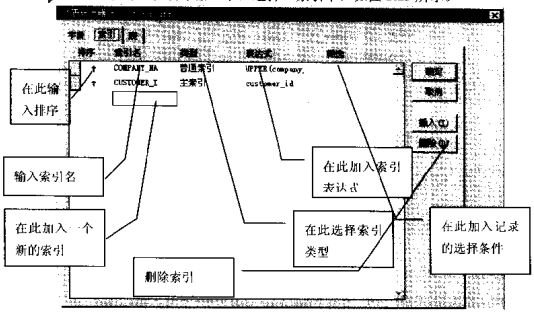


图3.29　“表设计器”的索引卡

下一步 在“索引名”对话框，输入索引名。

下一步 在“类型”处选择索引类型。

说明 索引的类型主要有四种，这些索引控制表字段和记录中是否允许或者禁止重复值，还可以用于在永久关系中建立参照的完整性：

主索引（Primary Indexes）：在字段输入的值是唯一的，不允许重复，对于属于一个数据库的表都可以建立一个主索引，一个表只能有一个主索引。

候选索引（Candidate Indexes）：也是具有唯一值的索引，在数据库表和自由表中都可建立候选索引。一个表可以有多个候选索引，必要时它可以当作主索引。

普通索引（Regular Indexes）：普通索引可以决定字段的处理顺序，它允许字段中有重复值，一个表中的普通索引可以有多个。

唯一索引（Unique Indexes）：为了保证与以前版本的兼容性，Visual FoxPro 中可以使用唯一索引。唯一索引允许出现重复，但唯一索引只存储索引文件中重复值第一次出现的记录。“唯一”指索引文件对每一个特定的关键字只存储一次，而忽略了重复值第二次或以后的记录。

下一步 在“表达式”框输入“索引表达式”。

下一步 如果想要选择记录，可以从“筛选”处进入“表达式生成器”建立过滤器。

完成 按“确定”完成索引的创建。

3.3.10　用索引给表排序

一旦建立了索引，你就可以用它为记录进行排序。

使用索引的操作步骤如下：

开始 在“项目管理器”，中选择已经建立索引的“表”，然后选择“浏览”。

下一步 从主菜单中选择“表”，然后选择“属性”。

下一步 在“工作区属性”对话框的“索引顺序”处选择一个索引，如图 3.30 所示。

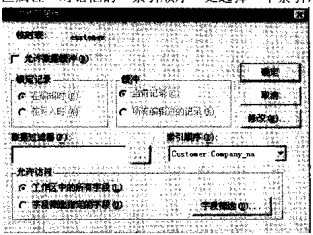


图3.30　选择一个索引

完成 按“确定”完成索引的引用。

技巧 按照工作性质的不同，你可以应用不同的索引，索引的使用最好遵循以下原则：

(1) 为了提高显示、查询、打印的速度用“普通索引”、“候选索引”或“主索引”。

(2) 为了控制字段的重复值或对记录进行排序，对数据库“表”用“主索引”或“候选索引”，对于“自由表”用“候选索引”。

3.3.11　对多个字段排序

为了提高对多个字段建立过滤器的查询或视图的运行效率，可以在“索引表达式”中对多个字段建立索引，实现对记录的排序工作。这种索引的排序是按照表达式的值进行的而不是按照字段。

为多个字段建立索引的操作步骤如下：

开始 在“项目管理器”，中选择已经建立索引的“表”，然后选择“修改”。

下一步 在“表设计器”中，选择索引选项卡，输入索引的名称。

下一步 在“索引名”处输入索引名称。

下一步 在“表达式”框输入你所想要的对多个索引的表达式，例如你想使记录按照“国家”、“邮政编码”和“公司名称”进行排序，可以用如下的表达式：

客户.国家+客户.邮政编码+ 客户.公司名称

说 明 你也可以对不同数据类型的字段建立索引，并在表达式中使用函数，例如你可以按照最大定货数量和公司名称进行排序，“最大定货数量”字段是货币型字段，而公司名称是字符型，表达式如下：

STR（客户.最大定货数量，20，4）+客户.公司名称

完成 按“确定”完成对多个字段索引的建立。

3.3.12　筛选记录

通过为索引建立筛选表达式，可以实现对记录的控制。

筛选记录的操作步骤如下：

开始 在“项目管理器”中选择已经建立索引的“表”，然后选择“修改”。

下一步 在“表设计器”中，选择索引选项卡，选择已建立的索引或建立新的索引。

下一步 在“筛选”处输入过滤表达式，例如你想输入“国家”在墨西哥的记录，可以建立下面的表达式：

客户.国家="Mexico"。

完成 按“确定”完成筛选表达式的建立。

3.4 用“数据库”管理数据

单独的使用“表”，可以方便地存储、浏览数据。可是只有将这些“表”加入数据库，Visual FoxPro 的强大功能才能发挥出来。把表放进数据库能减少数据的容余并提高了数据的完整性。例如不必为每个定单保存客户的名称和地址，你可以将客户的名称和地址存于一张“表”中，然后与定单的“表”建立一定的关系。如果用户的地址发生变化，你只需要在“客户”表中修改一条记录。为了更有效地操作记录和对远程数据进行访问，可以为数据库建立视图和连接。

3.4.1 数据库的设计过程

数据库是一种工作环境，它存储了一个“表”的集合，在表之间可以建立关系，对数据字段可以设置属性和触发规则，从而保证表之间数据的完整性。一个数据库文件具有.DBC的后缀。

一个精心设计的数据库可以使信息的访问变的十分方便，并不具有数据冗余。对 Visual FoxPro 来说，不同主题内容的信息保存在不同的表当中，为了使数据库的设计更为合理，数据库的设计可以采用如下步骤：

开始 确立“数据库”的功能：确定收集信息的范围，并仔细收集这些信息。

下一步 确定“表”的种类：根据信息确定不同信息的范畴，提取公共信息组成公共信息表，并按照不同的主题确定表的个数和每个“表”包含的信息。

下一步 确定“表”的结构：根据每个表的信息，确定表中的“字段”，将字段作为“表”中的一列。

下一步 确定“表”之间的关系：将所创建的“表”加以分析，确定各个字段之间的“关系”，要明确为“一对一”、“一对多”关系，对于“多对多”关系最好对相关的表加以分解，添加新表或字段，从而转化为“一对一”、“一对多”关系。

完成 将所设计的数据库结构加以审核，看是否满足数据的独立性和完整性，必要时改进设计。

数据库设计是整个应用系统建立的基石，一个设计良好的数据库可以使应用系统的建立变的更为简单，反之很可能使应用系统开发一半在重新改进数据库，造成巨大损失。

说明 在建立数据库的过程中，对于“表”的建立你很可能在自由“表”还是“数据库表”之间犹豫。“自由表”之间没有必要的关联，如果只存储相对独立的信息，没有依靠其它表的信息或被其他“表”

所引用，可以使用“自由表”。

“数据库表”有更为强大的功能，它可以使用“长表名”和“长字段名”，表中的字段可以有“标题”和“注释”，表中的字段可以设置“默认值”，能设置字段级和记录级“规则”，对于插入、删除、修改等数据库操作可以设置“触发器”，它还可以实现同远程数据源的连接，创建本地视图和远程视图。

3.4.2　数据库的简单操作

对数据库的管理可以通过“项目管理器”，“表设计器”和“数据库设计器”等工具加以实现。

打开数据库的操作步骤如下：

开始 在“项目管理器”中选择想要打开的数据库。

下一步 选择“修改”按钮，可以进入“数据库设计器”。

完成 可以从主菜单中选择“文件”然后选择“打开”，选择所要打开的数据库，从而完成数据库的打开。

当一个数据库被打开后，在“数据库设计器”中将显示数据库所包含的所有“表”和“关系”。数据库设计器工具栏，可以实现对数据库各种功能的快速访问。“数据库”菜单可以帮助你实现数据库的命令。在“数据库设计器”上按住鼠标右键可以访问快捷菜单。

“数据库设计器”大小可以改变，可以展开和折叠数据字段和索引，从而可使“数据库设计器”中只显示表的名称。这些设置当你的“数据库设计器”中包含了大量的“数据表”的时候十分有用。

“数据库设计器”操作步骤如下：

开始 从主菜单中选择“帮助”，然后选择“示例程序”。

下一步 在出现的帮助窗口中，选择“Tasmanian Traders”然后选择“Open”。

下一步 在从“项目管理器”中选择“tastrade”数据库，单击“修改”按钮，显示“数据库设计器”，如图 3.31 所示。

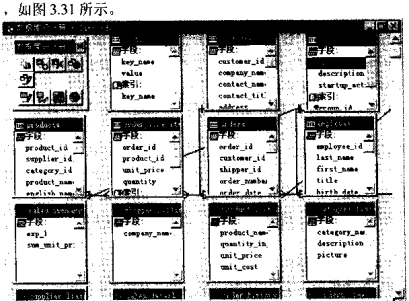


图3.31　“数据库设计器”

下一步 在“数据库设计器”上右点击鼠标，在出现的快捷键中选择“全部折叠”，此时的“数据库设计器”如图 3.32 所示。

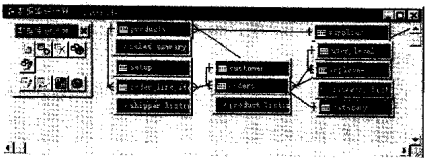


图3.32　折叠后的“数据库设计器”

下一步 在“数据库设计器”上右点击鼠标，在出现的快捷键中选择“全部展开”。

下一步 在“数据库设计器”中将鼠标置于“表”上，然后右点击鼠标，在出现的快捷键中选择“折叠”，然后再选择“展开”。

下一步 在主菜单中选择“数据库”，然后选择“重排”，在出现的“重排表和视图”的对话框中确定重排方式，然后按“确定”，如图 3.33 所示。

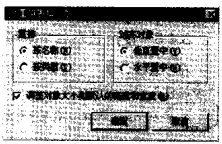


图3.33　重排表的大小位置

下一步 在“数据库设计器”中右点击鼠标，在出现的快捷键中选择“属性”，然后在数据库属性对话框中可以输入数据库的属性、选择数据库设计器中显示的类型，见图3.34，然后选择“确定”。

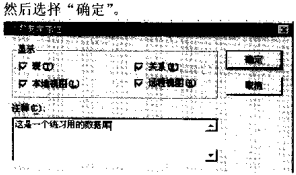


图3.34　输入数据库属性

完成 关闭“数据库设计器”。

3.4.3　向数据库中添加删除表

在 3.2 节中曾建立了项目“我的项目”并创建了数据库“业务”。本节介绍如何向数据库中添加“表”。

你可以向数据库中加入任何已经建立的“表”，但是这个“表”不能属于其它的数据库，因为“表”在同一时间只能属于一个数据库。如果你想要把属于其它数据库的“表”加入一个新的数据库，你首先把“表”从原来的数据库中“移出”。

在数据库中添加、删除“表”的操作步骤如下：

开始 从主菜单中选择“数据库”，选“添加表（A）…”。

下一步 在“打开”对话框中选择所要加入的表名，按“确定”。

下一步 选择刚加入的“表”，从主菜单中选择“数据库”，然后选“移去（R）”。

完成 在出现的对话框中选择“移去”。

3.4.4 在数据库中查找表

在数据库中很可能有大量“表”和“视图”，如果想很快的找到一个特定表，可以使用查找命令。

在数据库中查找“表”的操作步骤如下：

开始 从主菜单中选择“数据库”，选“查找对象（F）…”。

下一步 在“查找表或视图”对话框中选择所要查找的表或视图，按“查找”。

完成 如果只想显示表或视图，可以在主菜单中选择“数据库”然后选择“属性”，在对话框中选择所显示的类型。

3.4.5 使用长表名与注释

在“表设计器”中可以使用长表名和表的注释，在“表名”框中输入长表名，然后选“表属性”设置表的注释。

说明 在 Visual FoxPro 数据库中有一个数据字典，从而使数据库的设计和修改变的更为方便、灵活。这个数据字典是数据库所特有的，正是因为有了数据字典，数据库表的功能才大大高于“自由表”。数据字典为你提供了以下内容：

(1) 长表名。

(2) 为每个字段、表和数据库提供注释。

(3) 长字段名。

(4) 在浏览窗口显示的字段和表格列表头的标题。

(5) 字段的默认值。

(6) 主关键字和候选关键字。

(7) 字段级和记录级规则。

(8) 触发器。

(9) 数据库表间的永久关系。

(10) 存储过程。

(11) 到远程数据源的连接。

(12) 本地视图和远程视图。在下面几节中将分别加以介绍。

在 Visual FoxPro 中，表名可以由字母、数字、下划线或汉字组成，但表名的第一个字符必须是字母、下划线或汉字。数据库表和自由表的存储具有默认的表名及.DBF 文件名。对于一个数据库表，可以建立一个长表名。长表名最多可以包含 128 个字符，并且可以用来代替短表名来标识数据库表。如果定义了长表名，表在界面（如数据库设计器、查询设计器、视图设计器）中，将显示长表名。表的注释可以使表的功能易于理解，尤其对于一个人的应用项目来说，注释会给你带来极大方便。

使用长表名和注释的操作步骤如下：

开始　在“项目管理器”中先选择数据表，然后选择“修改”。

下一步　在“表设计器”中，选择“表”选项卡，如图3.35所示。

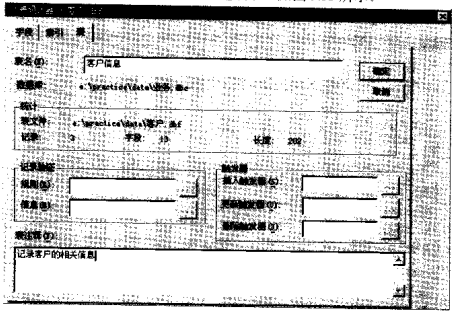


图3.35　“表”选项卡

下一步　在“表名”框输入长表名，在“表注释”框输入该表的注释。

完成　按“确定”完成长表名和注释的输入。

3.4.6　使用长字段名、标题与注释

与长表名一样，字段也可以有长字段名、标题与注释，它们都可以在“表设计器”中设置。

注意　在创建新表时应指定字段名，自由表的字段名最多可包含 10 个字符，数据库的字段名最多可以包含 128 个字符。如果从数据库中移去一个表，那么此表的长字段名将被截断为 10 个字符。如果长字段名的前 10 个字符在表中如果不唯一，Visual FoxPro 将取长字段名的前几个字符，而后在后面追加顺序号，共同形成 10 个字符长的字段名。如 customer_contact_name 的短名是 customer_c，customer_contact_address 的短名是 customer-2，依此类推。但是对于汉字的短字段名就无法区分。例如：“客户联系人”的短字段名为“客户联系人”，“客户联系人地址”的短字段名是“客户联系 ”，“客户联系人电话”的短字段名也是“客户联系”，因此应该尽量用前五个汉字将字段区分开。当一个表和数据库相关联时，必须使用长字段名来引用该表中的字段。如果从数据库中移去具有长字段名的表，长字段名就会丢失。

说明　数据库中建立表以后，能为每个字段添加字段的说明，使表更容易被理解更新。在“项目管理器”中选择字段以后，Visual FoxPro 便会显示该字段的注释文本。

数据库表中的每个字段可以有一个标题，Visual FoxPro 在浏览窗口中的字段的标题将显示为你输入的标题文字。

使用长字段名、标题和注释的操作步骤如下：

第一步 在“项目管理器”中先选择数据表，然后选择“修改”。

第二步 在“表设计器”中，选择“字段”选项卡，如图 3.36 所示。

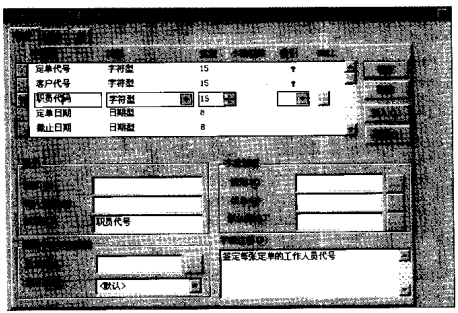


图3.36　为“字段”设置标题、长字段名、注释

第三步 在“字段名”框输入长字段名，在“标题”框输入字段的标题，在“字段注释”框输入该字段的注释。

第四步 按“确定”完成输入。

3.4.7　使用默认值

所谓“默认值”，就是在向数据库表中添加新的记录时，该字段可以是预先准备好的数值或字符串。无论在“表单”、数据浏览窗口或“视图”中输入数据，默认值都起作用，并一直存在于字段中，直到它被新值修改。默认值可以是除了通用型类型以外的任何数据类型。

说明　使用默认值能加快数据的输入速度，除非想输入不同的值，否则可以跳过该字段。例如对于“定单”表的“客户代号”一项，当前业务往来最频繁的“客户”可以被设为“默认值”，当有新的更为频繁的客户出现时，可以修改默认值与之相对应。

‘默认值’“可以是一个数量值，也可以是一个等于数量值的表达式。表达式可以是符合 Visual FoxPro 规则的任何 xBASE 表达式，但它的返回值的类型应该和字段的数据类型相一致。如果“默认值”表达式的数据类型与相关字段的类型不匹配，Visual FoxPro 将报错。如果表达式中使用了用户自定义函数，Visual FoxPro 将不对表达式求值，也不会报错。

当执行“追加记录”、“插入记录”的操作时，将计算默认值，并将其放入对应的字段中去。

注意　如果从数据库中“移去”或“删除”一个表，该表所有的默认值将从数据库中删除，但在存储过程中的对“默认值”的引用依然存在。

当没有设置“默认值”时，若 SET NULL 设置为 ON，则会插入一个空值，如果该字段允许使用 null 值，你也可以使用.NULL.来作为字段的“默认值”。

使用默认值的操作步骤如下：

开始 在“项目管理器”中先选择数据表，然后选择“修改”。

下一步 在“表设计器”中，选择“字段”选项卡，该卡的“字段验证”部分有“默认值”一项，如图 3.37 所示。

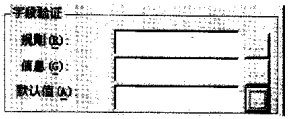


图3.37　为“字段”设置默认值

下一步 在“默认值”框输入该字段的默认值，必要时可以按...按钮进入“表达式生成器”完成默认值表达式的构造。

完成 按“确定”完成输入。

3.4.8　设置字段级规则

为了控制数据库中表字段的数据，可以创建字段级和记录级规则，这些规则被称为“有效性规则”（Validation Rules）。字段级和记录级规则把用户输入的值与所定义的规则表达式的值进行比较，如果两者不相匹配，则拒绝该值。

字段级和记录级规则能够控制输入到表中的信息类型，用户对字段的内容可以始终如一的进行有效性的检查。所有的用户在数据库规则面前地位均等，用户通过程序来输入不合法数据的可能性降低了。

说明 为了实现商业规则和参照完整性的约束，可以用“主索引”、“触发器”等多种方式，表 3.3 列出了使用这些方式的激活级别和时机。

表3.3　约束方式

约束方式	级	激活时机
NULL有效性	字段	当在浏览窗口中离开字段，或执行INSERT或REPLACE语句更改字段值的时候
字段级规则	字段	当在浏览窗口中离开字段，或执行INSERT或REPLACE语句更改字段值的时候
记录级规则	记录	记录更新（UPDATE）时
候选/主索引	记录	记录更新（UPDATE）时
VALID子句	表单	移出记录时时
触发器	表	表中的值发生改变时，在发生INSERT、UPDATE、DELETE事件时

用字段级有效性规则可以检查用户输入到字段中的信息类型或其它字段的值。利用字段级和记录级有效型规则可以实现数据的完整性。例如可以建立这样一条规则限制项目表中的“客户编号”字段的输入值：若“客户表”中不具有一个特定的“客户编号”，则在“项目表”中拒绝该“客户编号”的输入。

说明 字段级规则当字段值发生改变时便发生了作用，即使在数据缓冲区中，字段级规则也可以被激活。在“浏览窗口”、“表单”和其它用户输入数据窗口中处理数据时，如果焦点从字段上移开，Visual FoxPro便开始检查记录级规则、如果字段值没有改变，规则不被触发，进行字段级检查的情况如表 3.4 所示。

表3.4　输入方式的检查规则

数据输入方式	窗口或命令	检查字段级规则
用户界面	浏览界面 表单 其它用户界面 窗口	当从字段上移开时，如果字段值已经改变，则检查字段级规则（如果字段值不改变，则不检查字段规则）
不指定字段的命令	APPEND APPEND GENERAL APPEND MEMO BROWSE CHANGE DELETE EDIT GATHER APPEND BLANK INSERT INSERT –SQL	在字段值改变时检查字段级规则，按字段的定义顺序进行检查 在追加或插入记录时，检查字段级规则
指定字段的命令	UPDATE UPDATE-SQL REPLACE	根据命令中指定字段的顺序检查字段级规则

设置字段级规则的操作步骤如下：

开始 在“项目管理器”中先选择数据表，然后选择“修改”。

下一步 在“表设计器”中，选择“字段”选项卡，该卡的“字段验证”部分有“规则”和“信息”框，如图 3.38 所示。

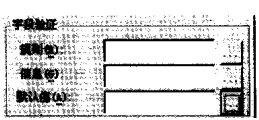


图3.38　为“字段”设置规则

下一步 在“规则”框输入该字段的规则，必要时可以按...按钮进入“表达式生成器”完成规则表达式的构造。

下一步 在“信息”框输入该字段级规则被违反时所显示的提示信息。

完成 按“确定”完成字段级规则的输入。

3.4.9　设置记录级规则

记录级有效性规则属于表的有效性规则，用记录级有效性规则，可以控制用户输入到

记录中的信息类型。记录级有效性规则检查不同字段在同一记录中的限制，从而保证不违反数据库的商业原则。例如：为了确保项目的开始日期在截止日期之前，可以设置一个记录级规则。

说明　记录级规则当记录值发生改变时被激活。当记录指针离开记录时，Visual FoxPro 检查记录级规则。如果记录值没有改变，记录级规则不被触发。如果修改了记录，并没有移动记录指针，此时关闭浏览窗口，记录级规则仍被检查，并对错误的发生提出警告，然后关闭浏览窗口。与触发器不同，即使数据在缓冲区中，记录级规则也会被激活，如果发现违反了规则，就需要有错误处理的代码，直到用户改正错误或者取消更改用户才能进入下一个操作。

在有效性规则中，不要包括在当前工作区中移动记录指针的命令和函数。如果规则中包含了象 SEEK、LOCATE、SKIP、APPEND、APPEND BLANK、INSERT、AVERAGE、COUNT、BROWSE和REPLACE FOR这样的函数和命令，规则将被循环触发，从而产生错误。

如果从数据库中删除、移去一个数据库表，所有与它相关的规则将同时从数据库中删除。然而，由被移去、删除的规则所引用的存储过程将继续存在，这是因为存储过程可以被多个规则引用。

设置记录级规则的操作步骤如下：

开始　在“项目管理器”中先选择数据表，然后选择“修改”。

下一步　在“表设计器”中，选择“表”选项卡，该卡的“记录验证”部分有“规则”和“信息”框，如图 3.39 所示。

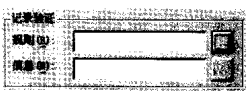


图3.39　设置记录级规则

下一步　在“规则”框输入该记录的规则，必要时可以按...按钮进入“表达式生成器”完成记录级规则表达式的构造。

下一步　在“信息”框输入该记录级规则被违反时所显示的提示信息。

完成　按“确定”完成记录级规则的输入。

3.4.10　设置触发器

触发器（Trigger）是针对“表”的表达式，当表中的任何记录被指定的操作命令修改时，出发器被激活。出发器能执行数据库应用程序要求的其它操作。出发器可以执行：对数据库记录的修改，进行参照完整性的检查。

触发器是作为表的特定属性来存储的，如果从数据库中删除一个表，相关的出发器也将被删除。当进行了其它有效性检查后，触发器被激活，触发器不对缓冲区数据起作用。

设置触发器的操作步骤如下：

开始　在“项目管理器”中先选择数据表，然后选择“修改”。

下一步　在“表设计器”中，选择“表”选项卡，该卡的“触发器”框，如图 3.40 所示。

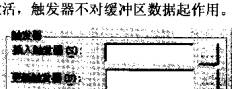


图3.40　设置记录级规则

下一步　在“插入触发器”框输入插入的触发规则，在“更新触发器”输入更新触发规则，在

“删除触发器”框输入删除引发的触发器规则，必要时可以按...按钮进入“表达式生成器”完成触发器表达式的构造。

完成 按“确定”完成触发器的输入。

3.4.11　建立、编辑永久性关系

当表建成以后，信息已经加以分类。然而为了使这些信息表达的更为合理，还需要给表建立关系。在各个独立的表之间都存在着关系，例如想了解当前定单的客户的信息，在“定单”表和“客户”表之间便存在着关系。

要建立两个表之间的关系，可以把其中一个表的主关键字添加到另一个表中，使两个表都具有该字段。在表与表（表 A 与 B）之间可以有三种关系：“一对多”、“一对一”、“多对多”。下面分别加以介绍：

一对多关系：它是一种最为普通的关系。在“一对多”关系中，表 A 中的一条记录在表 B 中有多条记录与它对应，而表 B 中的一条记录在表 A 中只有一条。在“一对多”关系中，相同的字段在“一”方要建立主关键字或候选关键字，“多”方要使用普通索引。例如对于“定单”与“客户”来说每个客户可以具有多个定单。

多对多关系：此时表 A 的一条记录在表 B 中可以有多条记录，反之亦然。此时，要使数据库正常工作，必须改变数据库的设计，从而转化为“一对一”或“多对多”关系。例如，对于一个部门可以与多个客户有业务联系，因而对于部门中的每一条记录，在客户表中可以有多条记录。反之，对于每个客户，也可能与多个部门合作。这时需要建立一个中间表——一项目表，从而将多对多的关系分解为两个一对多的关系。对于部门表与项目表，每个部门可以有多个项目，但每个项目只能有一个部门。对于客户表和项目表，每个项目可以有多个项目，每个项目只指向一个客户。从而它们都是一对多的关系。

一对一关系：这种关系在表 A 中的一条记录在表 B 中只能有一条记录，对表 B 的每一条记录也在表 A 也只能有一条记录与之对应，这种关系并不经常使用。

说明 为了使建立的关系更为合理，可以按以下方法建立关系：

(1) 确定哪个表是“一”，哪个表是“多”。

(2) 对于“一”端的表，添加一个整数字段，并使这个字段成为主索引。

(3) 对于“多”端的表，添加和“一”表主索引相同的字段，并对这个字段建立普通索引。

注意 两个表的索引表达式应该是相同的，即“一”表的索引的表达式如果包含了一个函数，在“多”表中的表达式也应该用相同的函数。

建立、编辑永久性关系的操作步骤如下：

开始 如果需要建立关系的表的索引还没有建立，请先用“表设计器”为相应的表建立关系，并打开数据库设计器。

下一步 在“数据库设计器”上将鼠标放于“一”表的主索引上，按下鼠标将该主索引拖拽到“多”表的与其对应的普通索引之上，此时在两索引之间建立了一条“一对多”的连线，表示两表之间的索引，如图 3.41 所示。

下一步 在“数据库设计器”上将鼠标放于“一对多”的连线上，单击使其变粗表示选择，然后按鼠标右键，在出现的快捷菜单上选择“编辑关系”，如图3.42所示。

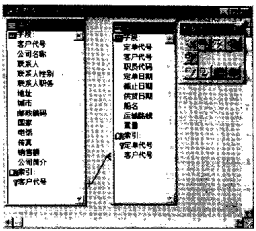


图3.41　建立关系

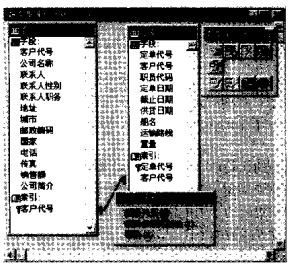


图3.42　编辑关系

下一步 在“编辑关系”对话框中输入关系的条件，如图3.43所示。

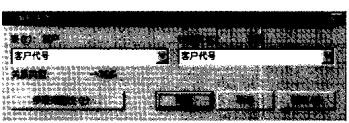


图3.43　编辑关系对话框

说明 建立的关系的类型是由子表的索引类型决定的，当子表为主索引或候选索引时，建立的关系为“一对一”关系，当子表的索引为普通索引或唯一索引时，建立的关系是“一对多”关系。

数据库表之间的关系分为“永久性关系”和“临时关系”。永久性关系是存于数据库文件中的数据库表间的关系，它有以下特点：在查询设计器和视图设计器中，自动作为默认连接条件，在数据库设计器中，显示为联系表索引之间的连线，作为表单和报表之间的连线，在数据环境设计器中显示，它的功能是实现表参照的完整性。

完成 按“确定”完成关系的编辑。

3.4.12　建立、编辑临时性关系

永久性关系并不控制表内的记录指针间的关系，因而还需要建立“临时关系”。临时关系会引起一个表（子表）的记录指针自动随另一表（父表）的记录指针移动。这样便允许当关系中的“一”方（父方）选择一个记录时，会自动的在“多”方（子表）定位与之相关的记录。

建立、编辑临时性关系的操作步骤如下：

开始 如果需要建立关系的表的索引还没有建立，请先用“表设计器”为相应的表建立关系，并打开数据库设计器。

完成 在“数据库设计器”上，将鼠标放于“一”表的主索引上，按下鼠标将该主索引拖拽到“多”表的与其对应的普通索引之上，此时在两索引之间建立了一条“一对多”的连线，表示两表之间的索引。

3.4.13　建立参照的完整性

参照完整性（Referential Integrity）涉及一系列规则，从而当输入或删除记录时，保持已经定义的表之间的关系。从而防止下面的情况发生：① 当父表中没有关联记录时在子表中添加相关记录。② 主表中的相关字段的值发生改变，从而使子表出现无关记录。③ 删除主表中的相关记录。

用存储过程和触发器的代码可以实现参照的完整性。Visual FoxPro 的参照完整性生成器（RI）能帮助你实施规则类型、要实施规则的表和导致参照完整性检查的系统事件。RI 生成器是进行级联删除和更新的一个工具。

建立参照完整性的操作步骤如下：

开始 打开数据库设计器。

下一步 在“数据库”菜单选择“编辑参照完整性”或进入“数据库设计器”双击关系的连线，在“编辑关系”对话框单击“参照完整性”按钮，也可以在连线上单击鼠标右键。在出现的快捷菜单中选择“参照完整性”。此时出现 RI 生成器对话框，如图 3.44 所示。

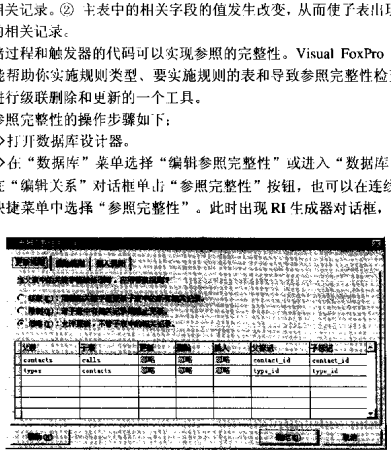


图3.44　RI生成器

完成 输入相应的规则后完成参照完整性建立。

使用 RI 为数据库生成规则时 Visual FoxPro 把生成的代码作为引用的存储过程保存起来。但是当更改数据库的设计以后，那么应该在运行数据库之前，重新运行 RI 生成器。这样存储过程和实施参照完整性的触发器将被修改，反映出最新的设计变化。如果不运行 RI 生成器，可能会出现意外结果。

3.4.14 使用多个数据库

同时使用多个数据库有两种方法：其一是同时打开多个数据库，另一种方法是不打开数据库而引用其中的表。

数据库被打开以后，表之间的关系由数据库中的信息来控制。当打开一个数据库以后，如果不关闭其它数据库，原来打开的数据库将仍保持打开状态。此时可以选择当前的数据库，并使用当前数据库中的表。

使用多个数据库的操作步骤如下：

开始 打开“我的项目”。

下一步 在“项目管理器”中新建一个数据库“人事”。

下一步 在“项目管理器”中选择“人事”数据库，然后按“修改”，或者选择“打开”，对于已经打开的数据库，“打开”按钮变为“关闭”。

下一步 在“项目管理器”中选择“业务”数据库，然后按“修改”，或者选择“打开”。

下一步 此时在“标准工具栏”中可以选择当前的数据库。

下一步 在出现的“数据库设计器”可以选择“人事”或者“业务”数据库，然后用鼠标选取，使其成为当前表。

完成 在“项目管理器”中选择数据库，然后按“关闭”按钮。

3.5 小 结

本章论述了 Visual FoxPro 6.0 的可视化的数据库管理，如何通过“项目管理器”管理数据库，如何建立数据库文件，如何建立表和索引以及如何用数据库的功能更好地管理数据。用户可以使用大量可视化的设计器、向导和功能强大的项目管理器，使得数据库设计、数据表的创建、索引的创建以及将所有这些组织进入一个完整的数据库变的十分容易。

Visual FoxPro 6.0 提供了建立基础数据库的很好的工具。通过本章的学习，用户应掌握以下内容：使用“项目管理器”管理项目，建立数据库文件，建立表和索引，使用“数据库”管理数据。

思考与练习

1. Visual FoxPro 项目管理器有何作用？如何使用项目管理器来管理数据、文档？
2. 在创建表之前需要做好哪些准备工作？什么是关键字段？
3. 过滤器有什么作用？怎样设置过滤器的过滤字段？
4. Visual FoxPro 中的索引有哪几种？如何确定索引的排序？索引的使用应该遵守哪几条原则？

5. “自由表”和“数据库表”有何区别？什么是数据字典，Visual FoxPro 的数据字典提供了哪些功能？

6. “默认值”、“规则”和“触发器”起什么作用？

7. 什么是字段级规则？什么是记录级规则？它们有什么区别？如何设置它们？

8. “一对一”、“一对多”、“多对多”关系各代表什么？它们对具有关系的表的索引有什么要求？

第4章　信息查询与更新

数据库的应用大大提高了对数据的处理能力，在浩如烟海的数据中找到自己关心的数据变的非常简单。在 Visual FoxPro 6.0 中，使用“查询设计器”能方便地生成一个“查询”，从而获得你所需要的数据。“视图”能帮助你从本地或远程数据源中获取相关数据，此时，可以对这些数据进行修改并更新，Visual FoxPro 6.0 将自动完成对“源”表的更新。同时你还可以实现对多个“表”生成一个“查询”或“视图”，这多个“表”既可以是本地的，也可以是远程的。

本章的主要内容有：

(1) 使用“查询”查询数据；

(2) 使用“视图”浏览更新数据；

(3) 查询更新多个表。

4.1　使用“查询”查询数据

4.1.1　创建查询

“查询”是为了更快地获得数据，Visual FoxPro 6.0 中的“查询设计器”可以按照用户的要求快速地建立“查询”。从而获得“表”和“视图”中的数据。

建立查询的步骤可以用图 4.1 所示的过程来表示。

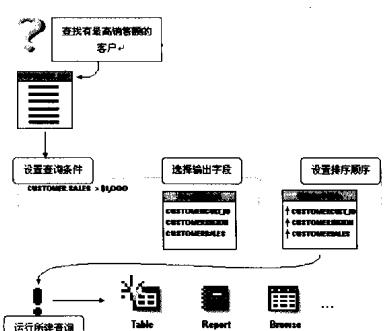


图4.1　创建“查询”的过程

在“查询设计器”中可以指定从“表”或“视图”查询记录的条件，然后按照想得到的数据的输出类型定向输出到“浏览窗口”、“报表”、“表”和“标签”等。查询可以用文件加以保存，它存为后缀为.QPR 的文件中。

1. 用“查询设计器”创建“查询”

开始 从主菜单中选择“帮助”，然后选择“示例程序”。

下一步 在出现的帮助窗口中，选择“Tasmanian Traders”然后选择“Open”。

下一步 在从“项目管理器”中选择“数据”选项卡，然后选择“查询”。

下一步 选择“新建”按钮。

下一步 在出现的对话框中选择“新建查询”按钮。

下一步 在“添加表或视图”对话框中选择“customer”表，按“添加”或双击，如图 4.2 所示。

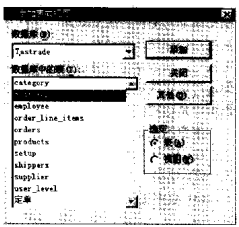


图4.2　创建“查询”的过程

说明 在“添加表或视图”对话框中，“数据库”框可以选择当前使用的数据库，“其它”按钮可以选择使用不属于数据库中的表。

下一步 单击“添加表或视图”对话框的“关闭”按钮，此时出现如图 4.3 所示的“查询设计器”。

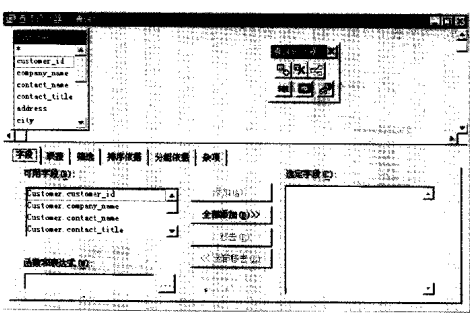


图4.3　查询设计器

说明 在“查询设计器”中，“字段”选项卡用来选择需要包含在查询结果中的字段，如果想改变信息行的排序，可以选择“排序依据”选项卡。

下一步 在“字段”选项卡按“全部添加”，也可以选择字段后按“添加”，在“选定

字段”框中会显示查询输出的字段。

下一步　在“字段”选项卡按“全部添加”，也可以选择字段后按“添加”，在“选定字段”框中会显示查询输出的字段。

完成　关闭“查询设计器”并在输入查询名称对话框中输入“客户查询”。

2. 用“查询向导”建立查询

开始　从“项目管理器”中选择“数据”选项卡，然后选择“查询”。

下一步　选择“新建”按钮。

下一步　在出现的对话框中选择“查询向导”按钮。

下一步　选择想要建立的查询的类型。

完成　按照“查询向导”对话框建立查询。

4.1.2　为查询输出字段建立别名

为了使你的查询结果更容易被理解，可以给字段加入一个描述性的标题。例如你可能希望在查询结果窗口中显示“定单总量”，从而代替表达式 SUM（MaxOrdAmount）或一个意义模糊的字段名。

用查询输出字段加别名的操作步骤如下：

开始　从“项目管理器”中选择“数据”选项卡，然后选择“查询”。

下一步　选择查询，然后按“修改”。

下一步　在“查询设计器”中的“函数和表达式”框中输入字段的名称，然后在 AS 后加入别名，下面是表达式的例子：

SUM(maxorderamt) AS SumMaxOrd

下一步　选择想要建立的查询的类型“添加”按钮，将所选的表达式加入“选定字段”。

完成　关闭“查询设计器”输入查询名称。

4.1.3　为查询结果排序

排序决定了查询输出结果中记录或输出行的先后次序。

为查询结果排序的操作步骤如下：

开始　从“项目管理器”中选择“数据”选项卡，然后选择“查询”。

下一步　选择查询“查询客户”，然后按“修改”。

下一步　在“查询设计器”中选“排序依据”选项卡。

下一步　在“选定字段”框中选择排序字段。

下一步　按“添加”按钮，将字段加入“排序条件”框中，见图 4.4 所示。

下一步　选择排序条件中的字段左侧按钮上下拖拽，可以设置排序的顺序。

说明　在“排序条件”框中字段的排序顺序决定了该字段排序的重要性，第一个字段是主排序次序，为了调整排序字段的重要性，可以在“排序条件”框中，将左侧的按钮拖拽到相应位置上，通过设置“排序选项”中的按钮，可以确定排序是升序还是降序，在“排序条件”框中的字段的左侧，有上下箭头表示排序的升降顺序，各个字段可以有不同的排序顺序。

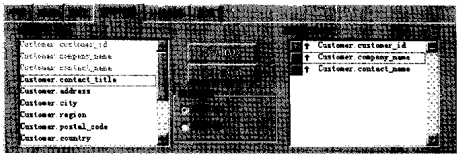


图4.4　查询设计器

完成 关闭“查询设计器”。

4.1.4　筛选查询结果

为了得到想要的查询结果，必须设置查询条件。查询设计器中的“筛选”选项卡可以用来限制想要检索的记录。

查询条件是为了选择所有数据的一个所需的子集，如查询所有工作时间超过 25 年的职员等。使用“筛选”选项卡可以对字段输入筛选表达式。

筛选所需记录的操作步骤如下：

开始 从“项目管理器”中选择“数据”选项卡，然后选择“查询”。

下一步 选择查询“查询客户”，然后按“修改”。

下一步 在“查询设计器”中选“筛选”选项卡。

说明 在“筛选”选项卡中，“字段名”选择筛选字段，“否”和“条件”构造运算符。例如当选择了“否”，然后在“条件”框选择“=”的意思是“不等于”，在“逻辑”框表示了各个表达式之间的关系是“逻辑或”或者是“逻辑与”。

下一步 对“字段名”、“否”、“条件”、“实例”、“大小写”和“逻辑”进行设置，构造出筛选表达式，如图 4.5 所示。

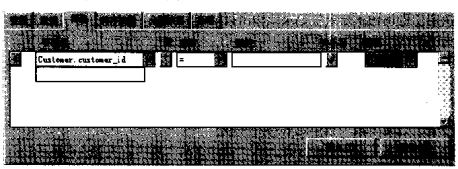


图4.5　构造筛选表达式

完成 关闭“查询设计器”。

4.1.5　查询结果的分组

分组就是为了把类似的记录压缩为一个结果记录，从而完成基于一组记录的计算。例如你可能想要知道一个地区的所有定单的总和，你不必一个一个地查看所有记录，而只需

要将所有同一地区的记录压缩为一个记录，从而得到那个地区定单的总数。

在“查询设计器”中，是通过“分组依据”选项卡来实现对记录的分组操作的。当应用积聚函数，如 SUM、COUNT、AVG 时，分组功能十分有用。

使用分组的操作步骤如下：

开始 从主菜单中选择“帮助”，然后选择“示例程序”。

下一步 在出现的帮助窗口中，选择“Tasmanian Traders”然后选择“Open”。

下一步 从“项目管理器”中选择“查询”，单击“新建”按钮。

下一步 选择“查询向导”，类型选择“Query Wizard”，单击“ok”按钮。

下一步 在“查询向导——Step1”的“databases and tables”框选择“ORDERS”，然后加入所有字段，单击“Finish”按钮。

下一步 在“查询向导——Step5”选择“Save Query”，按“Finish”按钮。

下一步 在对话框中输入“定单查询”，按“保存”按钮。

下一步 从“项目管理器”中选择“定单查询”，单击“修改”按钮。

下一步 在“查询设计器”中选择“字段”选项卡，单击“全部移去”按钮。

下一步 在“查询设计器”的“函数和表达式”框单击“...”按钮。

下一步 在出现的“表达式生成器”中的“数学”框选择“SUM（）”。

下一步 选择“数学”框选择“VAL（）”。

下一步 在“字段”框选择“discount”，构造后的表达式为：SUM(VAL(Orders.discount))。

下一步 按“确定”，在“查询设计器”按“添加”按钮。

下一步 在“查询设计器”中选择“分组依据”选项卡，如图 4.6 所示。

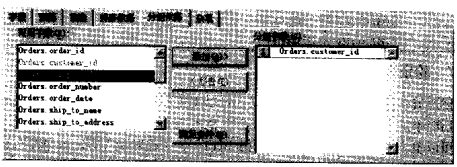


图4.6　建立分组依据

下一步 在“分组依据”选项卡的“可用字段”选择“customer-id”字段，单击“添加”。

更直 关闭“查询设计器”。

4.1.6　确定查询去向

查询的结果可以输出到不同的去向，如“浏览窗口”、“临时表”等。如果没有选定输出去向，查询结果将目的地默认为浏览窗口。

选择查询去向的操作步骤如下：

开始 从“项目管理器”中选择“数据”选项卡，然后选择“查询”。

下一步 选择查询“定单查询”，然后按“修改”。

下一步 在“查询设计器”中选查询工具栏中的“查询去向”按钮。

下一步　在“查询去向”对话框中选择输出去向，如图 4.7 所示。

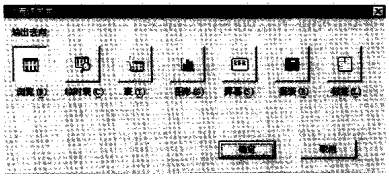


图4.7　选择查询去向

完成　按照“查询去向”的要求输入条件，然后按“确定”。

说明　为查询的结果指定去向，可以实现如表 4.1 所示功能。

表4.1　查询结果去向

输出去向	实现功能
浏览窗口	将查询结果在浏览窗口中显示出来
临时表	将你的查询结果存于你命名的只读的临时表中
表	将查询结果存于你命名的表中
图形	将查询结果与 Microsoft Graph 一起应用
屏幕	将查询结果显示于 Visual FoxPro 的主窗口中或当前活动输出窗口中
报表	输出到报表文件（.FRX）中
标签	输出到标签文件（.LBX）中

4.1.7　验证查询语句

如果你想确认查询是否被正确的定义，可以浏览由查询设计器所生成的代码，也可以为查询的功能添加注释。所添加的注释也会在 SQL 窗口中显示出来。

结果可以输出到不同的去向，如“浏览窗口”、“临时表”等。如果没有选定输出去向，查询结果将目的地默认为浏览窗口。

在建立查询的任何时刻，都可以选择“查询”菜单中的“查看 SQL（V）...”或从查询工具栏选择“显示 SQL 窗口”，查看 SQL 代码。

SQL 代码窗口是一个只读窗口，显示了查询的 SQL 语句，见图 4.8 所示。

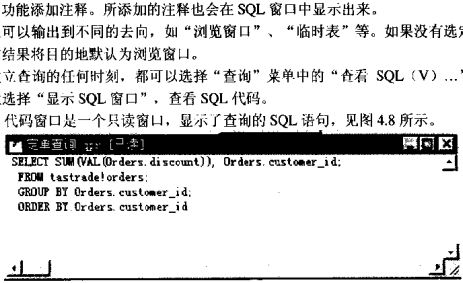


图4.8　SQL语句窗口

4.1.8　给查询添加备注

如果你想为你的查询加上注释，使得它更容易识别，可以为查询添加备注。

给查询加备注的操作步骤如下：

开始 从“查询”菜单中选择“备注”。

下一步 在备注对话框中输入备注，如图 4.9 所示。

完成 按“确定”完成备注的添加。

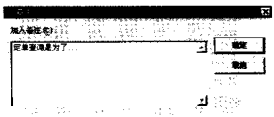


图4.9　给查询加备注

4.1.9　运行查询

查询设计完毕，并指定了输出目的以后，可以用“运行”按钮启动运行查询。Visual FoxPro 执行查询设计器产生的SELECT-SQL语句，并将结果定向为你指定的去向。

4.2　使用“视图”浏览更新数据

4.2.1　创建本地视图

“视图”是一组记录，它的来源可以是本地表、其它视图、存于服务器上的表或者远程数据源，如通过 ODBC 和在服务器上的 SQL Server 相连。当你对视图加以修改和更新时，这种修改和更新可以反映到数据源中。

说明 创建视图的过程和创建查询的过程相似，其中的主要区别是视图是可以更新的，而查询不能更新。如果只想获得一组只读的查询结果，可以利用查询并把查询保存于一个“QPR”文件中。如果想从本地或远程表中提取一组数据，并且想更新这组数据，就需要使用视图。

查询、视图和远程视图的特征见图 4.10 所示。

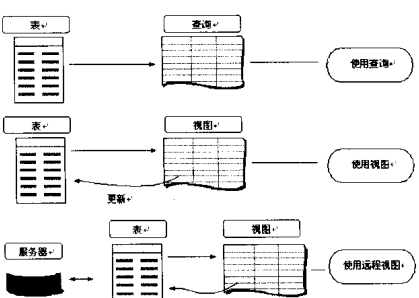


图4.10　查询、视图和远程视图

创建“视图”，可以通过“视图向导”或“视图设计器”。

1. 用视图向导建立视图

开始 按照前面介绍的打开“Tasmanian Traders”示例。

下一步 从“项目管理器”中选择“数据库”，然后选择“本地视图”或者“远程视图”。

下一步 按“新建”，然后选择“视图向导”。

完成 按照向导要求建立视图。

使用视图设计器可以创建本地视图。本地视图包括本地的 Visual FoxPro 表，使用.DBF 格式的表和存于本地服务器上的表。本地视图在“项目管理器”中在“数据库”选项中，如图 4.11 所示。

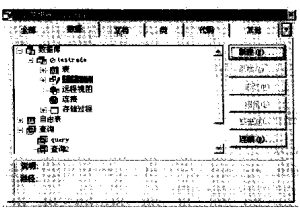


图4.11　本地视图的位置

2. 用视图设计器建立本地视图

开始 从“项目管理器”中选择“数据库”，然后选择“本地视图”。

下一步 按“新建”，然后选择“新视图”。

下一步 在“添加表或视图”对话框中选择需要建立视图的表或视图。

下一步 关闭“添加表或视图”对话框，进入“视图设计器”，如图 4.12 所示。

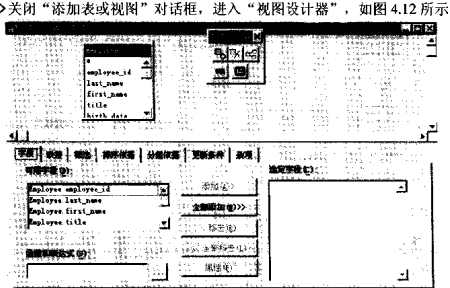


图4.12　本地视图的位置

下一步　在“视图设计器”的“字段”选项卡选择在视图中显示的字段，按“添加”按钮。

完成　在主菜单中选择“查询”，然后选择“运行查询”可以运行已经建立的视图。

说　明　“视图设计器”和“查询设计器”几乎一样，只是“视图设计器”中多了一个“更新条件”选项卡，从而可以定义更新的条件。

4.2.2　与远程数据连接

通过远程视图，你可以从 ODBC 服务器上提取一部分数据，而不用将所有的数据都载入本地计算机上。在本地对所选择的记录进行更改或者添加后，其结果可以返回到远程的数据源上。

有两种方式可以实现和远程数据的连接：直接通过注册在你计算机上的 ODBC 数据源或者用“连接设计器”自行连接。ODBC 驱动必须已经安装到了你的计算机上，并已经注册了 Visual FoxPro 数据源。

建立远程数据连接的操作步骤如下：

开始　从“项目管理器”中选择“数据库”，然后选择“连接”。

下一步　按“新建”，出现“连接设计器”，见图 4.13 所示。

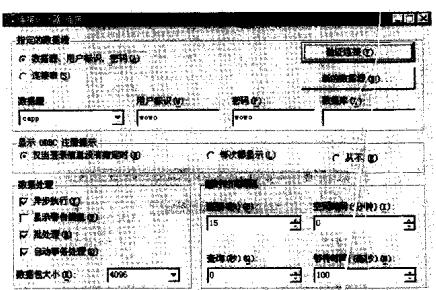


图4.13　建立远程连接

下一步　按“数据源”框中选择数据源。

下一步　如果有必要输入“用户标识”。

下一步　如果有必要输入“密码”。

下一步　如果有必要输入“数据库”项。

说　明　在“连接设计器”的“数据源”中如果没有你需要的数据源，可以按“新的数据源”进入 ODBC 数据源管理器”设置数据源，也可以按“验证连接”来验证你的连接是否能执行。

你也可以通过“文件”菜单中的“新建”，选择“连接”来建立一个新的连接。

重复 关闭“连接设计器”在“保存”对话框中输入连接的名称，如“我的连接”。

4.2.3 建立远程视图

建立远程视图，可以通过已经建立的连接或新建立一个“连接”来实现。

建立远程视图的操作步骤如下：

开始 从“项目管理器”中选择“数据库”，然后选择“远程视图”。

下一步 按“新建”，出现“选择连接或数据源”对话框，如图 4.14 所示。

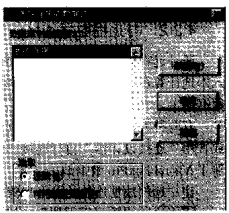


图4.14　选择连接或数据源

下一步 按“选取”区的“连接”在“数据库中的连接”框显示你当前建立的连接，按“可用的数据源”可以列出你的计算机上的可用的ODBC数据源，选择一个“连接”或“数据源”，然后按“确定”。

说明 本例连接到一个 SQL Server 数据源中，设置的连接“我的连接”时已经确定了用户名和密码，如果设置不全，会出现登录对话框，见图 4.15 所示，输入用户名和密码即可。

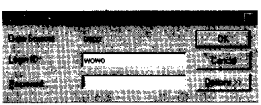


图4.15　进行SQL Server登录

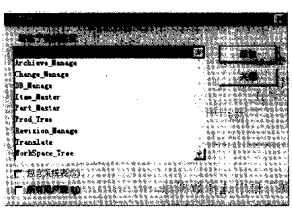


图4.16　打开远程数据表

下一步 在出现的如图 4.16 所示的对话框中选择相应的数据表，按“添加”。

重复 在出现的“视图设计器”中设计你的视图。

4.2.4 用视图更新数据

视图中数据的更新和表中数据的更新类似。但用视图可以实现对数据源的更新。

“数据库设计器”中的“更新条件”选项卡可以控制对数据的修改，修改可以反映到数据源中，也可以打开和关闭对表中指定字段的更新，并设置合适的SQL语句来完成更新。在“视图设计器”中的视图，其默认设置通常允许视图被更新。

更新视图的操作步骤如下：

开始　从“项目管理器”中选择“数据库”，然后选择一个视图。

下一步　按“修改”按钮，出现“视图设计器”，选择“更新条件”选项卡，如图4.17

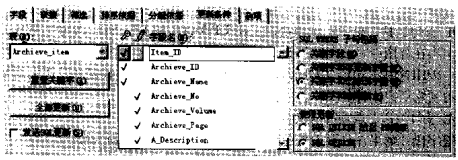


图4.17　设置更新条件

所示。

下一步　选择“发送SQL更新”复选框，使表变为可更新。

说明　如果需要在视图上的修改返回到数据源中，必须至少设置一个关键字段才能使用这个选项。当在视图设计器中首次打开一个表时，“更新条件”选项卡会自动显示哪些字段被定义为关键字段。

下一步　如果要设置关键字段，在“更新字段”选项卡中单击字段名旁边的“钥匙”开关列，如果想把设置恢复到源表的初始设置，按“重置关键字”按钮。

完成　如果要使字段可更新，在“更新字段”选项卡中单击字段名旁边的“笔”开关列，如果想把所有字段都更新，按“全部更新”按钮，此时表中必须有已经定义的关键字段。

说明　在表中可以设置只允许某些字段可以更新，如果使表中的任何字段可更新，在表中必须有已经定义的关键字段。如果某一个字段没有标注为可更新的，用户在表单中或者浏览窗口中修改这个字段，但修改的结果不会反映到数据源中去。

4.2.5　控制更新数据的条件

如果你工作于多用户环境下，与多个用户共同访问在服务器上的数据库，有可能出现多个客户同时改变在远程服务器上的数据的情况。在Visual FoxPro中可以用更新条件选项卡来设置更新的条件。

在“SQL子句包括”框中，可以设置当多个用户获得同一数据时的更新原则，在更新操作被允许之前，Visual FoxPro检查在远程数据源中的表，看它们从被提取时算起是否已经改动。如果在数据源中的数据已经改动，更新不被允许。“SQL子句包括”框的选项如

图 4.18 所示。

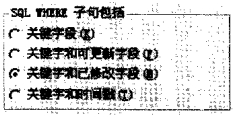


图4.18　设置远程更新条件

“SQL 子句包括”框中的选项决定了在 UPDATE 和 DELETE 语句中 WHERE 子句包含的字段，当在视图中对数据发生改动时，Visual FoxPro 向远程数据源或表发送什么样的更新指令，WHERE 子句用来判断从你的视图从数据库中提取数据到发送更新指令期间，这些记录被其它用户进行的操作。

“SQL 子句包括”选项的含义如下：

主关键字段：如果在“源”表中的主关键字段被修改则更新失败。

关键字和可更新字段：如果在远程数据表中的可更新字段被修改，则更新失败。

关键字和已修改字段：如果你在本地修改的字段在“源”表中被变更，则更新失败。

关键字和时间戳：如果从第一次提取数据开始，字段的时间戳（timestamp）发生改变，则更新失败。

4.2.6　控制视图字段的显示与输入

在视图中可以包含表达式、设置输入的提示、也可以设置如何与服务器进行数据通信。

因为视图属于数据库的一部分，因而可以使用数据库的一些特性。如可以给字段添加标题、添加注释、设置触发规则。

控制视图字段显示的操作步骤如下：

开始　从“项目管理器”中选择“数据库”，然后选择一个视图。

下一步　按“修改”按钮，这时出现“视图设计器”对话框，选择“字段”选项卡，如图 4.19 所示。

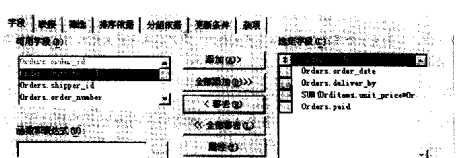


图4.19　设置视图的字段

下一步　在“选定字段”列表框选择一个字段，然后按“属性”，出现图 4.20 所示的“视图字段属性”对话框。

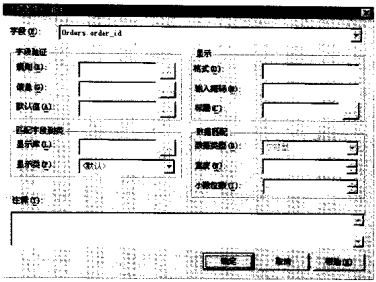


图4.20　设置视图的字段属性

下一步 在“视图字段属性”对话框中输入显示、触发、字段验证等属性。

完成 按“确定”完成属性的设置。

4.2.7　控制视图更新的方法

为控制视图主键字段在视图中的更新，可以使用“视图设计器”中“更新条件”选项卡的“使用更新”选项组，如图 4.21 所示。这组选项决定当主键字段更新时，发往服务器或“源”表的更新语句使用的 SQL 语句命令。

图4.21　使用更新选项

更新的方式有两种：可以先把旧的记录删除，然后用在视图中输入的值代替（SQL DELETE 然后 INSERT），也可以直接用服务器上的 UPDATE 函数来实现服务器上记录的更新（SQL UPDATE）。

4.2.8　为视图添加筛选表达式

和查询一样，视图中也可以加入表达式和函数来实现筛选条件。

给视图添加表达式的操作步骤如下：

开始 从“项目管理器”中选择“数据库”，然后选择一个视图。

下一步 按“修改”按钮，出现“视图设计器”，选择“筛选”选项卡，如图 4.22 所示。

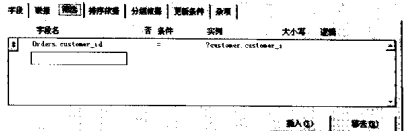


图4.22　为视图添加筛选表达式

下一步 在“筛选”选项卡中仿照“查询”筛选表达式的设计，构造出筛选表达式。

完成 退出“视图设计器”完成设置。

说明 如果视图是基于远程数据源，在“表达式设计器”中的函数为支持服务器上数据库的函数，Visual FoxPro 并不对你的表达式进行语法分析，而只是将它传递给远程的服务器。因此仔细阅读关于服务器数据库上的文档，看它支持哪些函数。

4.2.9 为视图传递参数

在建立视图时，可以为它传递一个参数，从而完成一个特定的查询。例如，你想查看在一个特定国家的客户，可以为国家字段定义为一个参数，参数名可以是字母、数字或下划线。

给视图传递参数的操作步骤如下：

开始 从“项目管理器”中选择“数据库”，然后选择一个视图。

下一步 按“修改”按钮，出现“视图设计器”，选择“筛选”选项卡，见图 4.22。

下一步 在“实例”框，用？代表参数名来构造表达式。

完成 在“查询”菜单中选择“运行查询”，会出现输入参数的对话框，如图 4.23 所示。

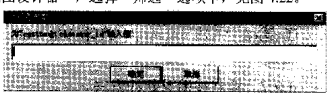


图4.23　为视图输入参数

4.2.10 设置与服务器的连接时间

如果你想调整 Visual FoxPro 与服务器的连接时间，可以使用时间设置。

设置连接时间的操作步骤如下：

开始 从“项目管理器”中选择“数据库”，然后选择一个连接。

下一步 按“修改”按钮，出现“视图设计器”，选择“超时时间间隔”区域，如图 4.24 所示。

完成 在“超时时间间隔”区域设置好连接时间，从“文件”菜单选择“保存”。

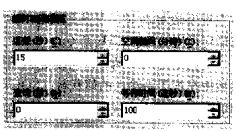


图4.24　设置连接时间

4.2.11 优化视图

在“高级选项”对话框，对视图如何获取数据，如何修改服务器或数据源中的数据可以进行优化。

优化视图的操作步骤如下：

开始 从“项目管理器”中选择“数据库”，然后选择一个连接。

下一步 从“查询”菜单中选择“高级选项”，出现“高级选项”对话框，见图 4.25 所示。

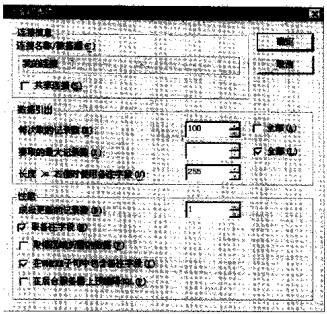


图4.25　高级选项对话框

下一步　在“高级选项”对话框中设置连接和获取数据的条件。

完成　选择“确定”完成设置。

4.3　查询更新多个表

4.3.1　向查询中添加表或视图

在 Visual FoxPro 中可以实现对多表和多视图的查询，也可以将本地视图和远程视图合并到一个视图中。当你的数据存于两个或多个表中，或者被分别存储于本地表和远程数据源中时，这种能力是非常有效的。

当你需要从两个或多个表中获取信息时，只需要将所有相关表加入你的查询，然后通过公共字段将它们联接在一起，之后你就可以实现对所有记录的查询了。实现多表查询的表可以是数据库表、自由表、本地或者远程视图。

当你将表或视图加入你的查询中时，Visual FoxPro 按照字段名相同的原则，将在表或视图之间建立联接。例如如果你将在 SAMPLED\DATA 中的表 Customer 加入查询设计器然后加入表 Orders，Visual FoxPro 将按照两表的相同字段 Customer.cust_id 和 Orders.cust_id 自动建立一个联接。

如果在你的数据库中对表和视图定义了永久性关系，Visual FoxPro 将把这些关系设为默认联接，如果你想将表或视图加入你的数据库中去，你首先应该打开相应的数据库。如果是对远程数据的访问，应先将服务器上的数据库服务器程序启动。

向查询中添加表或视图的操作步骤如下：

开始　从“项目管理器”选择“查询”，然后进入“查询设计器”。

下一步　在“查询设计器”工具栏上选“添加表”。

下一步 在“添加表或视图”对话框，在“数据库”框中选择相应的数据库，在“选定”框中选择“表”或“视图”，选择想要加入的表或视图，按“添加”，如图 4.26 所示。

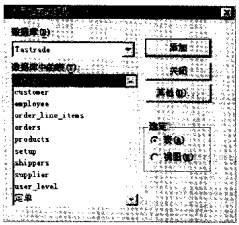


图4.26　添加表或视图

说明 如果想加入一个不属于数据库的表，单击“其他”按钮。在“打开”对话框中选择需要的数据表文件，然后按“确定”。

下一步 在“查询设计器”中的“联接条件”对话框中，验证联接，见图 4.27 所示。

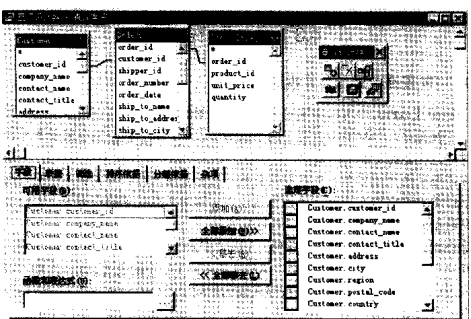


图4.27　向查询设计器中添加多个表

完成 “关闭”添加表或视图对话框。

4.3.2　通过联接选择记录

通过在查询中多个表之间添加联接或修改联接，可以控制你的查询中的记录。通过联接条件对话框，你能更改表之间的联接类型。

当你添加表时，联接自动出现，然而如果相关联的字段的字段名并不完全相符合，你可能需要在表之间自己建立联接。在查询设计器中，你可以在表之间拖动字段来建立联接，也可以在查询设计器工具栏中选择“添加联接”，然后在“联接条件”对话框中设定联接。

在表之间建立联接的操作步骤如下：

开始 从“项目管理器”选择“查询”，然后进入“查询设计器”。

下一步 在“查询设计器”工具栏上选“添加表”，添加两个或多个表到“查询设计器”。

下一步 在“查询设计器”工具栏，选择“添加联接”。

下一步 在“联接条件”对话框中选择在两个表之间需要建立联接的字段名。

下一步 在“联接条件”对话框中选择在两个表之间需要建立联接的字段名，如图 4.28 所示。

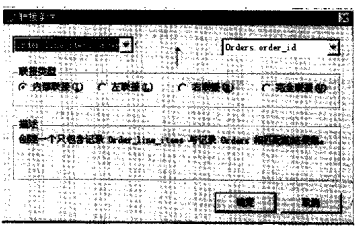


图4.28　向查询设计器中添加多个表

注意 所联接的字段的列的类型和字段的长度必须完全相同。

下一步 在“联接条件”对话框中选择联接类型。

说明 各种类型的联接具有不同的功能，下面加以介绍：

内部联接：是最常用的联接，只有当两个表的联接条件同时满足时，才能提取数据。

左联接：提取联接条件中左侧的表的所有数据，满足联接条件的右侧的表的数据。

右联接：提取联接条件中右侧表中的所有数据，满足联接条件的左侧表中的数据。

完全联接：不管是否满足联接条件，从两个表中提取所有数据。

完成 按“确定”完成联接条件的建立。

4.3.3　删除修改表间的联接

你也可以修改已经存在的联接。

删除、修改联接的操作步骤如下：

开始 从“项目管理器”选择“查询”，然后进入“查询设计器”。

下一步 在“查询设计器”中单击一条联接连线，使它变粗。

下一步 从“查询”菜单中选择“移去联接条件”。

下一步 在“查询设计器”中选择“联接”选项卡，如图 4.29 所示。

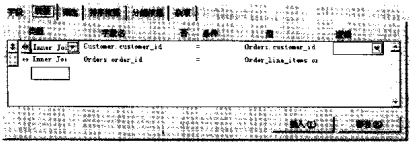


图4.29　修改联接条件

完成 在“联接”选项卡中修改你所想要修改的“联接”。

说明 除了筛选和联接类型外，你可以通过更改联接条件来控制查询的结果。联接并不要求相连的字段完全匹配，你可以用 Like、>=、>、in 等操作来确定联接条件。

联接条件像筛选条件，它们都是通过对满足条件的记录的值的比较来实现的。筛选是将字段的值同一个筛选种类型的联接具有不同的功能，而联接条件是比较一个表中一个字段和另一个表中的字段的值。

4.3.4　将多个本地表加入本地视图

你可以将两个或多个本地表加入本地视图。建立多表视图和建立多表查询类似，但视图可以具有对数据源表进行更新操作。例如你可以将 SAMPLES\DATA 表 Customer 和 Orders 建立一个视图。当从一个“浏览窗口”或者“表单”中对视图的结果集进行修改时，这种修改可以反映到数据源表中。

将多个表加入本地视图的操作步骤如下：

开始 从“项目管理器”选择“本地视图”，然后进入“视图设计器”。

下一步 在“视图设计器”中加入所需要的表。

下一步 在“视图设计器”工具栏选择“添加联接”，进入“联接条件”对话框，按要求建立联接条件，然后选择“确定”。

下一步 在“视图设计器”中选择“更新条件”选项卡，如图 4.30 所示，并加入关键字和可更新字段，然后选择“发送 SQL 更新”。

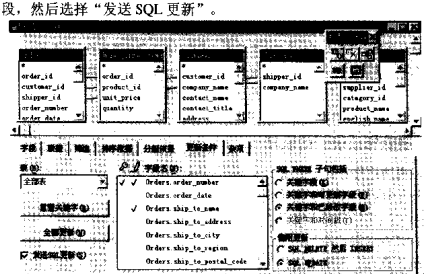


图4.30　确定更新条件

完成 关闭“视图设计器”完成多表“本地视图”的建立。

4.3.5　将多个远程表加入远程视图

当和远程数据源连接后，可以访问远程数据源中的相互关联的表，为了得到相关信息你可以将两个或多个远程表加入远程视图。

将多个表加入本地视图的操作步骤如下：

开始 从“文件”菜单中选择“新建”，然后选择“远程视图”，选择“新建文件”。

下一步 在“选择连接或数据库”对话框中，选择一个预先定义好的“连接”或“数据源”，如果有必要需要进行登录，然后按“确定”。

下一步 在“打开”对话框，选择你想加入的远程表，见图4.31所示。

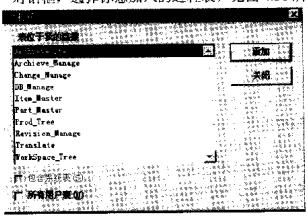


图4.31　在“打开”对话框中选择远程图表

下一步 当加表时，选择设置默认“联接”，或在“联接条件”对话框中输入必要的联接条件，如图4.32所示。

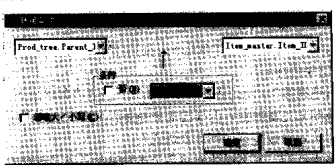


图4.32　建立联接条件

完成 按“确定”进入“视图设计器”，设置“更新条件”及其它属性，完成多表“远程视图”的建立。

4.3.6　将本地表和远程表同时加入视图

如果你存储的数据既存储于数据源中又存储于一个相关的本地表中，你可以将本地和远程数据连接到同一个视图中。

首先建立一个“远程视图”，从服务器上得到你需要的相关记录，然后用一个“本地视图”将本地表和“远程视图”连接在一起。

例如，假设你在服务器中有一个存储客户信息的数据库，本地表中存储了国家代码的信息。你可以先建立一个远程视图，与服务器相连接，并在客户表中选择相关数据，然后建立一个“本地视图”加入本地表和已经建立的“远程视图”。当在“本地视图”添加新的定单信息时，你可以更新“远程视图”和本地表中的数据。

将远程数据和本地表加入本地视图的操作步骤如下：

开始 建立一个“远程视图”，并从远程服务器添加一个或多个表。

下一步 建立一个新的“本地视图”并加入刚刚建立的“远程视图”。

下一步 将相关的本地表加入视图，并用相同的字段将它们相联。

下一步 进入“视图设计器”的“筛选”选项卡，设置相关的筛选表达式，然后运行。

下一步 更新视图的结果将同时更新本地表和远程视图。

完成 关闭“本地视图”，然后关闭“远程视图”更新远程服务器上的数据。

4.4 小　结

数据库的应用大大提高了对数据的处理能力，本章主要论述了怎样对数据库中的数据进行查询和更新。包括使用“查询”，使用“视图”以及如何实现对多表的查询和更新操作。Visual FoxPro 6.0 中，使用“查询设计器”能方便地生成一个“查询”，获得你所需要的数据。“视图”能帮助你从本地或远程数据源中获取相关数据，可以对这些数据进行修改并更新，Visual FoxPro 6.0 将自动完成对“源”表的更新。你还可以实现对多个“表”生成一个“查询”或“视图”，这多个“表”既可以是本地的，也可以是远程的。

通过本章的的学习，你应该掌握以下内容：使用“查询”查询数据，使用“视图”浏览更新数据，查询更新多个表。

思考与练习

1. Visual FoxPro 中建立查询有哪几步？如何用“查询设计器”和“查询向导”建立查询？如何为查询结果排序？
2. 什么是“本地视图”，什么是“远程视图”？两者有何区别？
3. 如何使用视图来更新数据？怎样控制更新条件和需要显示的字段？
4. 如何为视图传递参数和添加筛选表达式？
5. 如何建立多表查询？怎么样才能将多个本地表加入本地视图？
6. 如何将多个远程表加入远程视图？
7. 如何将本地表和远程表同时加入视图？

第5章　信息的显示与输出

为了更有效地实现对数据的管理，你可能将结果打印出来或显示于一个窗口中，Visual FoxPro 数据库提供了“报表”和“标签”，从而方便对表中数据和查询结果的打印。而“表单”提供了一个方便、快捷的数据浏览和编辑的方式。

本章的主要内容有：

（1）设计报表和标签；

（2）使用表单。

5.1　设计报表和标签

5.1.1　规划你的报表布局

利用“报表设计器”或者“报表向导”，你可以设计如发票之类的报表文件，然后你可以在打印页中显示后者打印你的报表或标签。通常报表的设计可以分为四步，如图 5.1 所示。

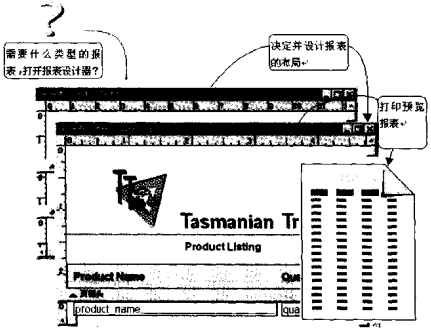


图5.1　设计报表的过程

开始 > 决定需要创建什么类型的“报表”。

下一步 > 创建“报表”布局文件。

下一步 > 修改并定制布局文件。

完成 > 预览打印“报表”。

“报表”和“标签”可以把数据以文档的形式进行显示输出。报表由两部分组成：“数据源”（Data Source）和“布局”（Layout）。数据源通常为数据库中的表，也可以是查询、视图或者临时表。而“布局”决定了报表的打印输出格式。当定义了一个表、视图或查询以后，你就可以创建一个表和标签了。

说明　每张报表都有一定格式，统一存储于报表的格式文件里，在 Visual FoxPro 中格式文件具有.FRX 的后缀。每个报表文件还应该伴随一个后缀为.FRT 的文件。报表文件统一规定了将要打印的字段、相关文本以及它们在页面上的输出位置。在报表文件中并不存储将要输出的数据库字段的具体的值，只是存储了它们在报表上的位置及格式。标签格式文件的后缀是.LBX 相关文件后缀为.LBT。

所有的报表都有一定格式，如图 5.2 所示。各种报表的复杂程度也不一样，复杂的如商用发票，简单的如一个商品列表，因此报表的布局应该符合你的报表的类型。

图5.2　报表的类型

报表的布局应符合一定的原则，表 5.1 所示为关于各种报表布局应遵循的原则。

表5.1　报表布局的原则

布局类型	布局说明	例子
列报表	每行输出一条记录，记录字段的值水平放置	存货清单 学生登记表 统计报表
行报表	每条记录的字段在一侧放置	货物列表
一对多报表	一条记录对应的多条记录	发票 财务报表
多栏报表	多列的记录，记录的字段沿左边放置	电话簿 名片
标签	多列的记录，记录沿左边放置	邮政标签

5.1.2　创建报表的布局

创建报表的布局有三种方法：

通过“报表向导”来创建单表或者多表报表。

采用“快速报表”从单表中创建报表。

用“报表设计器”修改、建立报表。

报表的修改是通过报表设计器来实现的。通过“报表向导”可以使用户在“报表设计

器”中对报表的定义程式化，通过回答一个个屏幕上的提问，可以很快地创建需要的报表。“快速报表”建立报表最为迅捷。而通过“报表设计器”，你可以任意定制报表，但这需要做一定的工作。

用报表向导创建报表的操作步骤如下：

开始 从“项目管理器”选择“报表”，选择“新建”。

下一步 在“新报表”对话框中选择“报表向导”。

下一步 在随后的“选择类型”对话框中选择向导类型，如“报表向导”类型，如图 5.3 所示。

图5.3　选择向导类型

下一步 按“确定”，进入如图 5.4 所示的选择字段对话框中选择“Contacts”表，并选择所有字段。

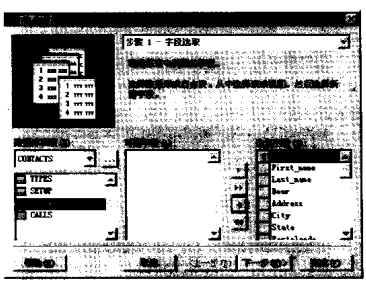


图5.4　选择字段

下一步 按“下一步”，进入选择报表类型对话框，如图 5.5 所示，选择“选择报表样式”，选择“带区型”，然后按“下一步”。

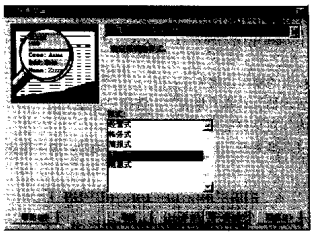


图5.5　选择报表样式

下一步 在选择报表布局对话框设置报表的布局，如图 5.6 所示。

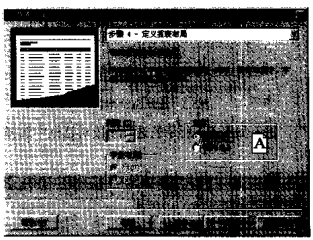


图5.6　定义报表布局

下一步 在排序对话框输入需要排序设置的字段，如图 5.7 所示。

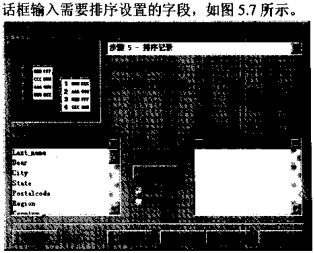


图5.7　选择记录排序

下一步 按“下一步”，进入完成对话框，此时可以预览报表的格式，如图 5.8 所示。

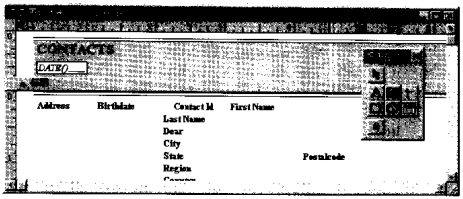


图5.8　预览报表的结果

下一步　如果打印预览的报表满足设计，按“Finish”完成报表的设计。

完成　在“另存为”对话框输入报表格式文件的名称和路径，按“保存”完成。

用“报表设计器”建立一个空白报表的操作步骤如下：

开始　从“项目管理器”选择“报表”，选择“新建”。

下一步　在“新报表”对话框中选择“新报表”。

完成　在“报表设计器”中设计你的报表。

5.1.3　创建标签的布局

标签具有特定的多列格式，如同多列报表布局。“标签”的建立和“报表”的建立类似，可以通过“标签向导”或者“标签生成器”来创建标签。

用标签向导创建标签的操作步骤如下：

开始　从“项目管理器”选择“标签”，选择“新建”。

下一步　在“新标签”对话框中选择“标签向导”。

下一步　在随后的选择表对话框中选择需要建立标签的“表”，如图5.9所示。

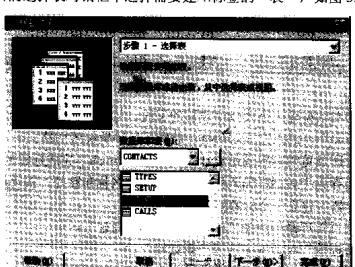


图5.9　创建标签选择表对话框

下一步　按“下一步”，进入如图5.10所示的选择标签类型对话框，选择标签的类型。

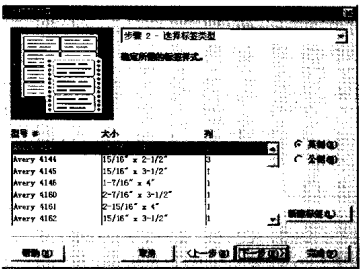


图5.10　选择标签类型

下一步 在定义标签布局对话框设置标签的布局，如图 5.11 所示。

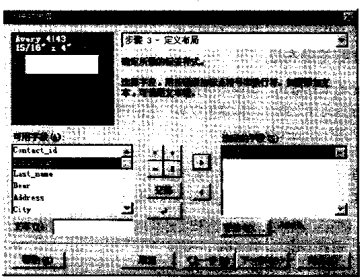


图5.11　定义标签的布局

下一步 在排序对话框输入需要排序设置的字段，与图 5.7 近似。

下一步 按“Next”，进入完成对话框，此时可以预览标签的格式。

下一步 如果打印预览的标签满足设计需要，按“Finish”完成标签的设计。

完成 在“另存为”对话框输入标签格式文件的名称和路径，按“保存”完成。

用“标签设计器”建立一个标签的操作步骤如下：

开始 从“项目管理器”选择“标签”，选择“新建”。

下一步 在“新标签”对话框中选择“新标签”。

下一步 在“新标签布局”对话框中选择标签布局，然后按“确定”。

下一步 进入如图 5.12 所示的“标签设计器”。

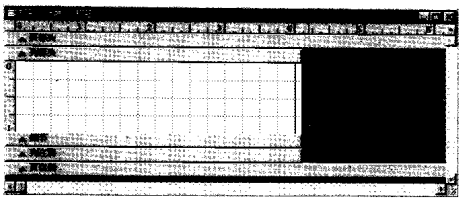


图5.12　标签设计器

完成　在“标签设计器”中设计你的标签。

说明　“标签设计器”和“报表设计器”相似，只是在默认页和纸张的规定略有不同。“报表设计器”利用了标准纸张的整页，而“报表设计器”使用的是标准的标签纸的大小。

5.1.4　修改布局

如果你已经用向导或快速报表生成了一个报表，或已经建立了一个“报表”或“标签”的原形，则在“报表设计器”或者“标签设计器”中，可以对它们进行修改。

在“报表设计器”或“标签设计器”中，可以对你想输出的“报表”或“标签”的格式和内容进行控制，可以加入一些“控制”，如“标签、直线、矩形、图片”等控制，也可以使用域、变量和表达式，另外还可以使用 OLE 绑定控制。这样定制出来的报表形式如图 5.13 所示。

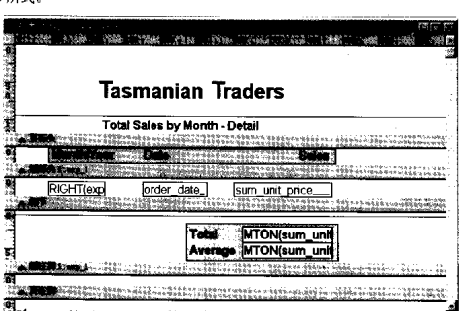


图5.13　修改定制报表布局

修改布局的操作步骤如下：

开始　从“项目管理器”选择一个“报表”或一个“标签”。

下一步 选择“修改”按钮。

更高 进入“报表设计器”或“标签设计器”进行修改。

5.1.5 规划数据的显示位置

在“报表设计器”中有若干个带区，每个带区实现的功能各有不同，通过设置不同的带区，可以控制数据在页面上的打印位置。在图 5.13 中列出了部分带区。在每一张报表中，可能会出现多个分组带区、多个列标头和注脚带区，在页面上，可以添加所需要的带区，它们的功能、建立方法可见表 5.2 描述。

说明 每个带区的属性、尺寸、特征在“报表设计器”中都可以加以改变，用鼠标可以拖动带区的大小，“标尺”可以指示当前带区的大小。此外，控制的高度一定要小于带区的大小，可以先对控制修改大小，然后再改变带区的大小。

表5.2 报表中各带区功能和建立方法

带区	出现的范围与频率	建立方法
标题	每报表打印一次	从“报表”菜单选择“标题/总结”
页面标头	每页打印一次	默认
列标头	每列打印一次	从“文件”菜单选“页面设置”， 设置“列数”>1
组标头	每组打印一次	从“报表”菜单，选择“数据分组”
细节	每记录打印一次	默认
组注脚	每组打印一次	从“报表”菜单，选择“数据分组”
列注脚	每列打印一次	从“文件”菜单选“页面设置”， 设置“列数”>1
页面注脚	每页打印一次	默认
总结	每报表打印一次	从“报表”菜单选择“标题/总结”

5.1.6 用“快速报表”添加控制

每一种控制都对应了一种数据类型的数据，在表 5.3 中列出了在“报表”和“标签”中可以使用的控制。

表5.3 控制与显示的数据

控制类型	显示的内容
域	表的字段、变量和表达式
标签	静态文本
直线	直线
直角长方形	填充区域方框
圆角长方形	圆角方框或填充区、圆、椭圆
图片/OLE绑定	位图或通用型字段

加到报表的控制的格式、大小、颜色、位置等都可以改变，每一个控制也可以具有一个注释，在打印时它并不出现，只是为了给设计人员提供方便。

用“快速报表”添加控制的操作步骤如下：

步骤1 从“项目管理器”选择“报表”。

步骤2 选择“新建”按钮。

步骤3 在“新报表”对话框中选择“新报表”。

步骤4 从主菜单的“报表”选择“快速报表”。

步骤5 在“打开”对话框中选择一个“表”。

步骤6 在“快速报表”对话框中输入标题、字段布局并可以添加一个别名，如图 5.14 所示。

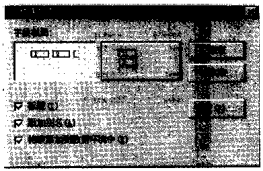


图5.14　快速报表对话框

步骤7 按“字段”按钮进入“字段选择器”，如图 5.15 所示。

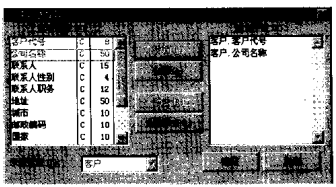


图5.15　字段选择器

步骤8 在“字段选择器”中选择所输出的字段，然后按“确定”。

步骤9 在“快速报表”对话框中选择“确定”，此时出现如图 5.16 所示的快速报表，相关的控制已经加入到了报表中。

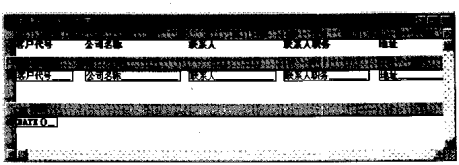


图5.16　用快速报表生成的报表格式

注意 通用字段用“快速报表”无法加入到报表中去。

步骤10 关闭“报表生成器”，在“项目设计器”中选择刚刚设计的报表，然后选择“预览”，可以预览刚才设计的报表，如图 5.17 所示。

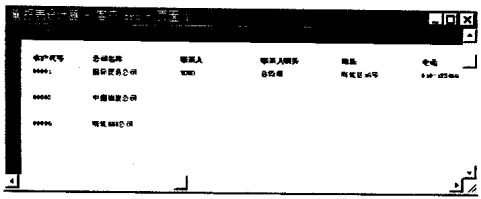


图5.17　生成报表的预览

完成　关闭“预览”窗口，返回“项目设计器”。

5.1.7　设置数据源

在 Visual FoxPro 中，“数据环境”的使用为用户对数据的调用提供了巨大方便，关于数据环境的详细情况，以后将详细介绍。在这里你只要这样理解就行了，假使你正在设计一个表单或报表，在“数据环境”中加入当前应用所需要处理的数据的“表”或“视图”则就不必一次次地去查找所需的数据了，“数据环境”为当前的应用建立了“数据源”，，“数据环境”为你执行以下任务：在打开或运行报表或表单时，自动的打开表或者视图，这些表或视图是你当前应用所需要的数据的集合，当关闭报表或表单时，数据环境自动关闭已经打开的"表"或“视图”。

数据环境中添加的“表”或“视图”，可以为一个表添加索引，从而使得数据的输出更为有序。

向数据环境中添加表或视图的操作步骤如下：

开始　从“项目管理器”选择“报表”，并选择一个已建的报表。

下一步　选择“修改”按钮，进入“报表设计器”。

下一步　在“显示”菜单选择“数据环境”，出现如图 5.18 所示的报表的“数据环境”。

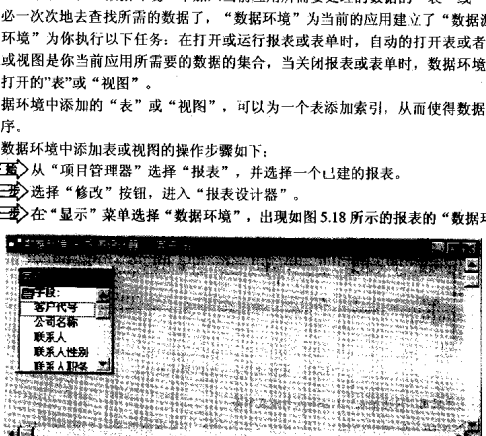


图5.18　报表的数据环境

下一步　在“数据环境”菜单选择“添加”或在“数据环境”上右击鼠标，在菜单中选

择“添加”。

下一步 在“添表或视图”对话框中选择需要加入的表，按“添加”按钮，如图 5.19 所示。

下一步 在“数据环境”按鼠标右键，从快捷键中选择“属性”对话框，如图 5.20 所示。

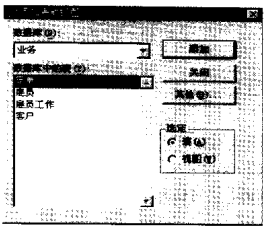


图5.19　选择向数据环境中添加的表或视图

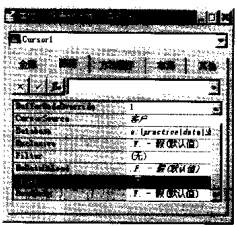


图5.20　属性窗口

下一步 在属性窗口中，选择最上面的下拉对话框，从中选择对象“Cusor1”。

下一步 选择“数据”选项卡，并选择“Order”属性。

下一步 在“Order”属性中输入索引名称，也可以从“数据”选项卡最上面的属性下拉列表中选取。

完成 关闭“属性”窗口。

5.1.8　向报表中添加控制

在 Visual FoxPro 中，“报表”中可以添加各种控制，如域、标签、直线、图片等。下面通过实例分别介绍加入控制的各种方法。

向报表中添加“域”控制的操作步骤如下：

说明 “域控制”用来加入数据库中的字段、表达式或计算的结果。

开始 从“项目管理器”选择一个“报表”，选择“修改”。

下一步 在“显示”菜单中选择“数据环境”。

下一步 在“数据环境”中选择想要加入的字段，按鼠标左键将该字段拖到“报表设计器”中需要加入字段的位置。

下一步 在“报表设计器”工具栏中选择“报表控件工具栏”。

下一步 在“报表控件”工具栏中选择“域控制”。

下一步 在“报表设计器”中“报表”的相应带区内加入“域控制”。

下一步 在“报表表达式”对话框输入需要的表达式和格式、“域控件”的位置等信息，如图 5.21 所示。

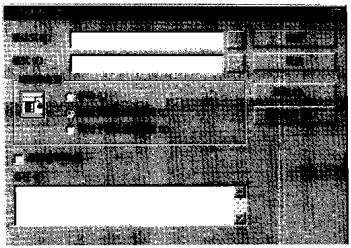


图5.21　为域控件设置表达式

➡ 如果需要，单击“表达式”框后的…，可以进入“表达式生成器”构造所需的表达式。

➡ 输入表达式后，单击“格式”框后的…，可以进入“格式”对话框设置所需的报表字段的输出格式，见图 5.22 所示。

说明　域控件的域表达式的格式有三种：字符型、数值型、日期型。对于每一种类型会出现不同的编辑选项。格式决定了报表或者标签打印时的打印格式。

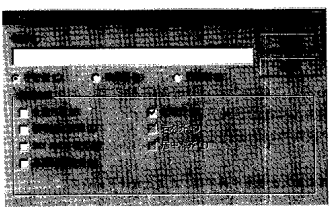


图5.22　设置格式

➡ 在“报表表达式”对话框中的“备注”框，对刚刚加入的域控制的功能加以说明。

➡ 按“确定”完成“域控制”的添加。

➡ 选择刚刚添加的“域控制”。

➡ 从“格式”菜单中选择“文本对齐方式”，从下一级菜单中选择一种合适的对齐方式。

说明　域控件文本的“对齐方式”只是改变了域控件中文本在输出时打印的位置，而不改变域控件在报表中的大小和位置。

结束按“确定”完成“域控制”的添加。

1. 向报表中添加“标签”控制

说明“标签”使得在报表中出现文字文本，从而对输出的数据的内容、范围作出说明。

开始从“项目管理器”选择一个“报表”，选择“修改”。

下一步在“报表控件工具栏”中选择“标签”控制A。

下一步在“报表设计器”中的相应带区内单击鼠标左键，相应位置出现光标，如图 5.23 所示。

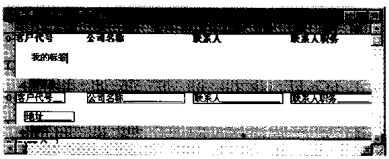


图5.23　加入标签控制

下一步在光标位置输入需要的文字，此时就象在文本编辑器中一样，可以进行换行、剪切、复制等操作，将鼠标左键再次单击完成输入。

下一步双击标签或者单击鼠标右键在快捷菜单中选“属性”，进入图 5.24 所示的“文本”属性对话框设置文本属性值，包括位置、注释、打印条件。

结束按“确定”完成“标签”的修改。

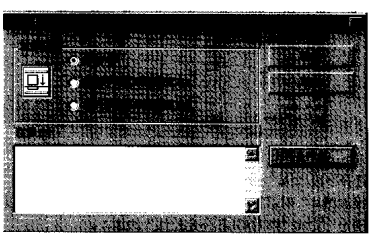


图5.24　设置标签属性

2. 向报表中添加“图片/OLE绑定控件”

开始从“项目管理器”选择一个“报表”，选择“修改”。

下一步在“报表控件工具栏”中选择“图片/OLE 绑定控件”。

下一步在“报表设计器”中的相应带区内单击鼠标左键，出现如图 5.25 所示的“报表图片”对话框。

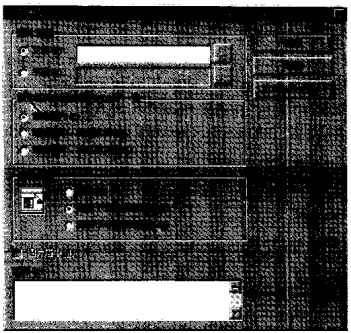


图5.25　报表图片对话框

下一步 在“图片来源”框中选择“字段”，输入字段名称，或者单击右侧的...按钮，进入“选择字段/变量”对话框，如图 5.26 所示。

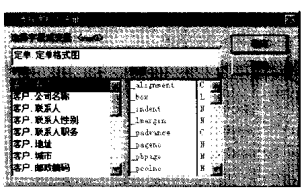


图5.26　选择通用字段

下一步 在“选择字段/变量”对话框中选择具有通用类型的“字段”，然后按“确定”。

下一步 在“报表图片”对话框中选择图片大小、对象位置必要时可以输入注释。

下一步 按“确定”完成“图片/OLE 绑定控件”的加入。

下一步 双击“图片/OLE 绑定控件”或者在“图片/OLE 绑定控件”控件上单击鼠标右键在快捷菜单中选“属性”，可以重新设置该控件的属性。

完成 按“确定”完成“图片/OLE 绑定控件”属性的修改。

3. 向报表中添加直线、长方形、圆控制

说明 在报表中加入直线、长方形和圆等控件，可以使你的报表更为美观，数据的划分更为直观，也可以用它们强调报表中的重要数据。

开始 从“项目管理器”选择一个“报表”，选择“修改”。

下一步 在“报表控件工具栏”中选择“直线”控制 ＋。

下一步 在“报表设计器”中的相应带区内用鼠标画出“直线”。

下一步 选择“直线”并拖拽直线两头的“小黑点”，可以改变直线的长度和位置。

下一步 选择“格式”菜单，然后选择“绘图笔”，在下一级子菜单中可以选择直线的类型，如粗细，是否为点划线等。

下一步 双击直线，出现“矩形/线条”对话框，如图 5.27 所示。

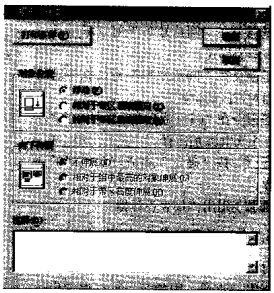


图5.27　设置矩形/直线的打印条件

下一步 在“矩形/线条”对话框中设置直线的打印位置和打印条件。

下一步 如果有必要在“矩形/线条”对话框中的“注释”框输入注释。

下一步 在“报表控件工具栏”中选择“矩形”控制 □。

下一步 在“报表设计器”中的相应带区内用鼠标画出“矩形”。

下一步 选择“矩形”并按住鼠标左键拖拽矩形四边头的“小黑点”，可以改变矩形的大小。

下一步 选择“格式”菜单，然后选择“绘图笔”，在下一级子菜单中可以选择直线的类型，如粗细，是否为点划线等。

下一步 选择“格式”菜单，然后选择“填充”，在下一级子菜单中可以选择矩形的填充方式。

下一步 选择“格式”菜单，然后选择“方式”，在下一级子菜单中可以选择矩形是否透明以及下面的控件是否可见。

下一步 选择“格式”菜单，然后选择“置前”或“置后”，可以调整矩形的显示“层”。

下一步 在“报表控件工具栏”中选择“圆角矩形”控制 ○。

下一步 在“报表设计器”中的相应带区内用鼠标画出“圆角矩形”。

下一步 选择“圆角矩形”并按住鼠标左键拖拽矩形四边头的“小黑点”，可以改变矩形的大小。

下一步 选择“格式”菜单，然后选择“绘图笔”，在下一级子菜单中可以选择直线的类型，如粗细，是否为点划线等。

下一步 选择“格式”菜单，然后选择“填充”，在下一级子菜单中可以选择圆角矩形的填充方式。

下一步 选择“格式”菜单，然后选择“方式”，在下一级子菜单中可以选择圆角矩形是否透明以及下面的控件是否可见。

下一步 双击圆角矩形，出现“圆角矩形”对话框，如图 5.28 所示。

下一步 在“圆角矩形”对话框中选择圆角矩形的样式。

下一步 在“圆角矩形”对话框中选择对象相对带区的位置。

下一步 在“圆角矩形”对话框中的“注释”框中为其添加注释。

下一步 按“确定”完成圆角矩形的属性设置。

完成 退出“报表设计器”。

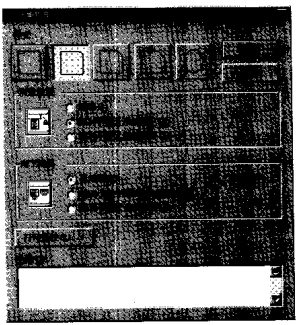


图5.28　圆角矩形的属性

5.1.9　报表控件的布局

对于加入“报表”中的各种控制，如域、标签、直线、图片等。可以对它们进行修改，或调整它们的布局，这就需要对它们进行选择、移动、调整大小、对齐、分组等操作。下面通过实例分别介绍对加入控件的操作方法。

调整控件的布局的操作步骤如下：

开始 从“项目管理器”选择一个“报表”，选择“修改”。

下一步 在“细节”带区单击想要选择的控件，此时控件被选中，如图 5.29 所示。

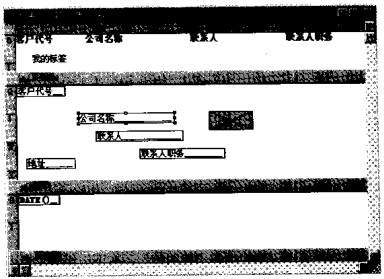


图5.29　选择相关控件

下一步 按住键盘的“Shift”键，然后用鼠标点击想要选择的控件，此时多个控件同时可以被选中。

下一步 从“格式”菜单中选择“分组”，刚才选择的“控件”便成为一组而存在，它们可以设置相同的属性，并可以一次选中同时移动、对齐。

从“格式”菜单中选择“取消分组”，分组的“控件”又成为单一的控件。

选择一个控件，用鼠标左键按住四边的选取框的“小黑点”拖动，可以调整控件的大小。

选择一组控件，如图 5.30 所示，然后从“格式”菜单中选择“对齐”，选择一种对齐方式。

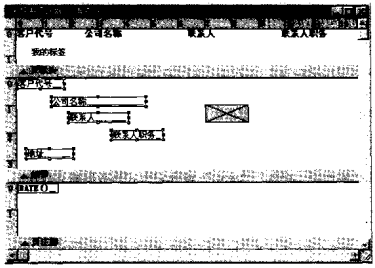


图5.30　选择控件并对齐

从“格式”菜单中选择“大小”，选择“调整到最宽”，然后再选择“调整到最高”，调整后的控件见图 5.31 所示。

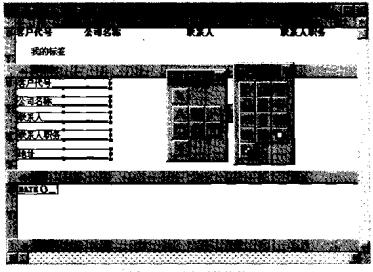


图5.31　对齐后的控件

也可以选择用“布局”工具栏来实现控件的布局。对齐方式可以选择上、下、左、右、中间对齐，调整宽度可以用等宽、等高、等大小，位置可以用水平居中、垂直居中。

选择一组控件，从“编辑”菜单中选择“复制”，然后选择“粘贴”或者直接按“Ctrl+C”键，此所选的控件组的另一个副本会出现，然后将它们移动到指定位置。

下一步 选择一组控件，从“编辑”菜单中选择“删除”或直接按“Ctrl+X”键，所选控件便被删除。

完成 关闭“报表设计器”完成对控件的设置。

说明 本小节讲述了如何对控制进行必要的调整，使它们具有所需要的外观。有效地利用“布局”工具栏将大大加快报表的设计速度。

5.1.10　给布局增加分组

对所输出的数据进行分组，可以使具有相同属性的数据或字段放在一起，便数据规范化。在每一个报表中可以具有多个分组，每一个分组可以有自己的“组标头”和组带区。分组使得数据的显示更为明显。组的分割建立在“分组表达式”之上。当为报表添加分组后，报表将自动加入组标头和组注脚带区，这些带区可以加入所需要的控件。

数据的分组需要对数据进行排序，如果在报表的“数据环境”中只有没有“索引”的表，那么需要为它加上索引，也可以使用带“索引”的“视图”。所有的数据都必须具有排序的规则，即必须使用“视图”、“索引”或其它方式为数据建立排序。

在布局中添加单一分组的操作步骤如下：

说明 “单一分组”是具有输入表达式的一级数据分组。例如，你可以打印出所有按“国家”区分的客户的记录，具有相同国家的记录放在一起，并按国家进行打印，此时“国家”字段必须是“数据源”中的排序字段。

开始 从“项目管理器”选择一个“报表”，选择“修改”。

下一步 在“报表”菜单选择“数据分组”将出现“数据分组”对话框，如图5.32所示。

下一步 在“分组表达式”框输入分组表达式，或者按[...]进入“表达式设计器”构造

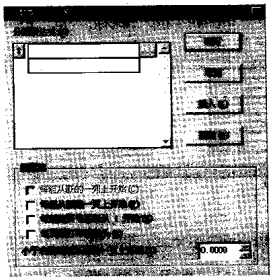


图5.32　添加单一分组

分组表达式。

下一步 在“组属性”框中，选择你需要的组的属性。

“组属性”选项框规定了组之间的分隔方式，分隔方式有四种方式：每组从新的一列

开始，每组从新的一页上开始，每组的页号重新从1开始，每页都打印组标头。输入表达式以后，可以从“组属性”选择这些选项。

下一步　按“确定”，此时在“报表设计器”中加入了你建立的新组，并自动加入了“组标头”和“组注脚”，如图5.33所示。

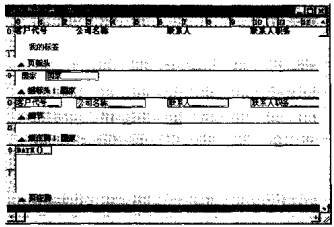


图5.33　加入单一组后的报表

在布局中添加多个分组的操作步骤如下：

在每一个报表中可以添加多达20个的数据分组，对于需要做大量的需要不同总计表达示和需要用不同方式来组织数据的报表，多级分组十分有用。

分组层次的选择，通常是按照更改频度的大小来进行的，通常将更改频度最大的作为一级分组。例如，可以按国家和城市为客户进行分组，在两个分组中城市字段的更改频度要大于国家，因此将“城市”组作为第一个，而将“国家”组当作第二个组。此时在数据环境的相关表中，必须用“国家+城市”为该表做排序索引。

开始　打开以前建立的项目“我的项目”。

下一步　从“项目管理器”选择一个“客户”表，选择“修改”。

下一步　在“表设计器”选择“索引”选项卡，为它添加一个“国家-城市”的普通索引，索引的表达式为“国家+城市”，如图5.34所示。

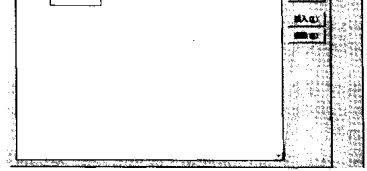


图5.34　添加“国家+城市”索引

下一步 选“确定”进入“报表设计器”，选择“客户”报表，然后按“修改”。

下一步 从“报表”菜单，选择“数据分组”，进入“数据分组”对话框，输入新的分组“客户.城市”。

下一步 在“分组表达式”框输入分组表达式“客户.城市”，或者按 ... 进入“表达式设计器”构造分组表达式。

下一步 按住“分组表达式”列表框的“客户.城市”分组左侧的按钮 ↕，将它拖动放到第一组的位置，从而改变了分组的次序，如图 5.35 所示。

下一步 在“组属性”框中，选择你需要的“组属性”。

下一步 按“确定”，此时在“报表设计器”中加入了你建立的新组，并自动调整了分组的顺序，加入了“组标头”和“组注脚”，见图 5.36 所示。

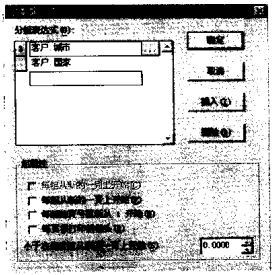


图5.35　添加多个分组

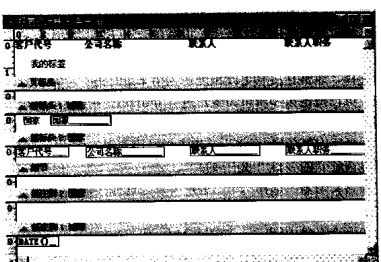


图5.36　两级分组后的报表

说明 从图 5.34 可知，“组”按照被创建时的顺序，即“数据分组”对话框中的顺序编号，“报表设计器”中组带区的名字为缩短的分组表达式，各组按级别从标头开始，具有最大编号的组离“细节”带区最近，而最小编号的组标头和组注脚离细节最远。

下一步 从“报表”菜单，选择“数据分组”，进入“数据分组”对话框，选择一个分组，然后更改“分组表达式”，然后按“确定”，可以实现对分组的修改。

下一步 从“报表”菜单，选择“数据分组”，进入“数据分组”对话框，选择一个分组，然后按“删除”按钮，数据分组被删除，同时在改组上加入的控件也同时被删除。

完成 关闭“报表设计器”完成对分组的设置。

5.1.11　定制报表的页面

“报表”的布局建立后，可以对它进行定制，报表的布局除了上面的关于控件、分组、数据环境以外，还包括页面的设置。“在报表设计器”中，除了可以用数据环境、数据分组以外，还可以利用“报表控件”、“布局”和“调色板”，图 5.37 列出了“报表设计器”可能用到的工具。

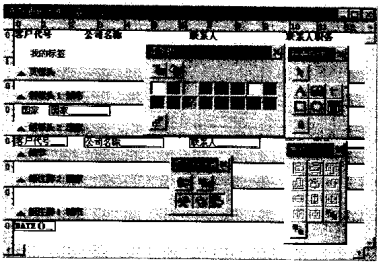


图5.37　报表设计器和相应工具

“报表设计器”中，“数据环境”规划出当前报表的数据，“页面设置”定义了报表的总体布局，如页边距、页面方向、纸张类型等。通过设置页边距、页面的大小和方向，可以得到页面的整体外貌，而页面上的各种控制、带区的设置决定了数据具体的打印输出方式。

设置纸张的边距、大小和方向的操作步骤如下：

开始在“文件”菜单中选择“页面设置”，出现如图 5.38 所示的页面设置对话框。

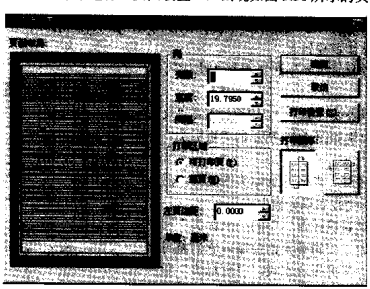


图5.38　页面设置对话框

下一步 在“页面设置”对话框中的左页边距，输入当前的左页边距的值，在“页面布局”处将显示当前页面的整体外观。

下一步 在“页面设置”对话框中按“打印设置”按钮，可以进入打印设置对话框，如图 5.39 所示。

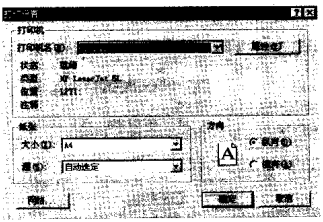


图5.39　打印设置对话框

下一步 在“打印设置”对话框的“纸张”框选择纸张的大小。

下一步 在“打印设置”对话框的“方向”框选择纸张的方向。

下一步 在“打印设置”对话框按“属性”按钮可以进入“文档属性”对话框进行高级的页面设置和纸张大小的设置。

下一步 在“打印设置”对话框按“确定”按钮完成纸张设置返回“页面设置”。

完成 按“确定”完成页面大小和方向的设置。

5.1.12　设置报表的带区

“报表”中，各带区的高度可以改变，报表带区的高度决定了每个报表带区在页面边距以内的区域中使用空间的大小。页面中的页面标头、注脚，细节带区都可以进行设计。此外，还可以添加页面和总结带区。下面通过一个例子学习如何使用带区。

设置带区的操作步骤如下：

开始 从“项目管理器”选择一个“报表”，选择“修改”。

下一步 在“报表设计器”中，用鼠标双击“页标头”带区栏，出现如图 5.40 所示的“页标头”对话框。

下一步 在“高度”框输入页标头的高度，然后按“确定”返回“报表设计器”。

下一步 在“报表设计器”中的“页标头”和“页注脚”内添加、编辑必要的控制，它们将每页出现一次。对于多页报表来说，“页标头”和“页注脚”通常包含报表的名称、页码、日期、标签等。

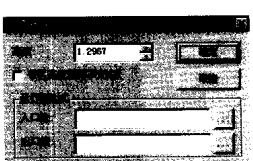


图5.40　页标头设置

下一步 在“报表设计器”中，用鼠标双击“细节”带区栏，出现与图 5.38 相似的“细节”对话框，在“高度”框输入页标头的高度，然后按“确定”返回“报表设计器”。

下一步 在“报表设计器”中的“细节”带区加入控件，从而实现对每一条记录的打印。

下一步 从“报表”菜单中选择“标题/总结”。

下一步 在出现的“标题/总结”对话框中选取“标题带区”和“总结带区”，如图 5.41 所示。

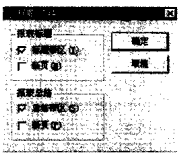


图5.41　添加“标题/总结”带区

下一步 如果希望标题或者总结带区单独出现在一页中，请选择“新页”。

下一步 按“确定”返回“报表设计器”。

下一步 “报表设计器”中的带区都将显示出来，见图 5.42 所示。

下一步 如果需要，可以双点击带区，在带区对话框的“高度”框中可以输入带区的高度。

下一步 如果需要，在相应的带区内加入控件，完成相关信息的输出。

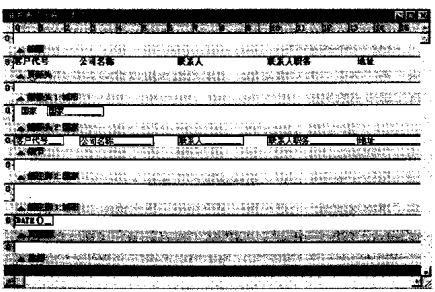


图5.42　加入标题和总结带区后的报表

完成 关闭“报表设计器”完成报表各带区的设计。

5.1.13　改变“域控件”和“标签”的字体

“报表”中，“域控件”和“标签”的字体可以改变。

改变控件的字体的操作步骤如下：

开始 从“项目管理器”选择一个“报表”，选择“修改”。

下一步 在“报表设计器”中选择一个控制。

下一步 从“格式”菜单中选择“字体”。

下一步 在“字体”对话框中选择相应的“字体”。

下一步 按“确定”完成字体的改变。
下一步 从“报表”菜单选择“默认字体”。
下一步 在“字体”对话框中设置默认字体。
下一步 按“确定”完成默认字体的改变。
说明 默认字体改变后，只有从改变后加入报表的控制才采用默认字体。
完成 关闭“报表设计器”完成字体设置。

5.1.14 改变控件的颜色

“报表”中，控件的颜色可以改变。
改变控件的颜色的操作步骤如下：
开始 从“项目管理器”选择一个“报表”，选择“修改”。
下一步 在“报表设计器”中选择一个控制。
下一步 从“显示”菜单中选择“调色板工具栏”。
下一步 在“调色板工具栏”中选择“前景色”工具，或选择“背景色”工具，然后选取对应的颜色，按“其它”可以选择其它颜色。
完成 关闭“报表设计器”完成颜色设置。

5.1.15 打印和预览报表

当“报表”和“标签”布局确定后，可以进行预览或打印。此外，在对“报表”布局进行设计过程中也可以预览。
“预览”可以不用完全的打印便能看到整个报表的外观和格式，从而发现设计的不足，通过对报表的放缩，可以发现设计的不足，及时加以修正。
预览报表的操作步骤如下：
开始 从“项目管理器”选择一个“报表”，选择“预览”。
下一步 在“报表设计器”的预览窗口中将显示出报表格式，如图 5.43 所示。

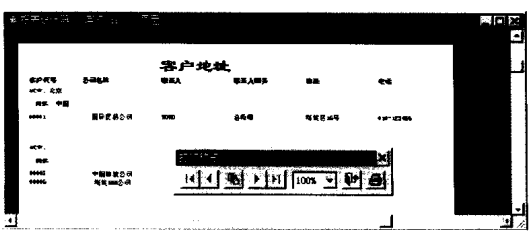


图5.43　打印预览

说明 使用打印预览工具栏的工具可以使打印的页从一页转到另一页。

完成 在“打印预览”工具栏中，选择“关闭预览”。

“报表设计器”中设计的报表只是形成了一个格式文件，只是定义了报表的外观和数据的打印位置。打印时将根据“数据源”对报表记录进行排序和分组处理。如果在数据环境中没有对数据进行分组、排序，数据就不会在布局中分组排序。

打印报表的操作步骤如下：

开始 从“项目管理器”选择一个“报表”，选择“修改”。

下一步 在“报表”菜单中选择“运行报表”，或在“文件”菜单中选择“打印”。Visual FoxPro将把报表发送到打印机。

完成 打印完成。

注意 如果用户没有设置数据环境，将出现“打开”对话框，选择想要输出的“表”，然后按“确定”。

5.2　使用表单

“表单”是Visual FoxPro中最为常用的数据的显示及编辑界面。在Visual FoxPro中通过“报表”和“标签”可以实现数据的打印输出。而“查询”、“视图”和“表单”实现数据的显示和编辑。与“查询”和“视图”相比，“表单”具有更大的灵活性，功能也更为强大。

5.2.1　创建表单

Visual FoxPro中“表单”的创建可以通过“表单向导”或者“表单设计器”。其中“表单设计器”使用户的工作变的十分简单、容易。

在Visual FoxPro中，“表单”和“表单集”对象具有属性、事件和方法，通过表单设计器可以对它们进行设置，必要时可以加入代码。

用表单向导创建表单的操作步骤如下：

开始 从“项目管理器”选择“文档”选项卡，选择“表单”，按“新建”。

下一步 在“新建表单”对话框中选择“表单向导”。

下一步 在随后的“选择类型”对话框中选择表单向导类型，如“表单向导”类型，如图5.44所示。

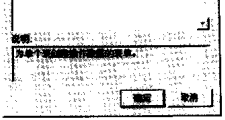


图5.44　选择向导类型

第三步　按“确定”，进入如图 5.45 所示的选择字段对话框，中选择“CONTACTS”表，并选择所有字段。

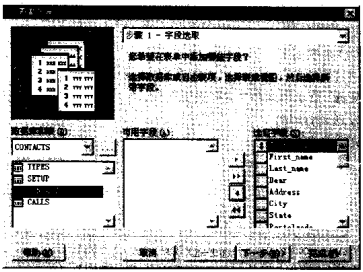


图5.45　选择表单的字段

第四步　按“下一步”，进入如图 5.46 所示的选择表单风格对话框，选择不同风格，在预览框中会出现相应的显示方式，在按钮类型选择框中有四种按钮风格，可以选择你所喜欢的按钮风格。

图5.46　选择表单风格

第五步　按“下一步”，进入如图 5.47 所示的记录排序对话框，选择排序字段。

第六步　按“下一步”，进入完成对话框，选择“完成”完成“表单”的创建。

第七步　在“另存为”对话框输入表单格式文件的名称和路径，表单文件具有“.SCX”后缀，按“保存”完成。

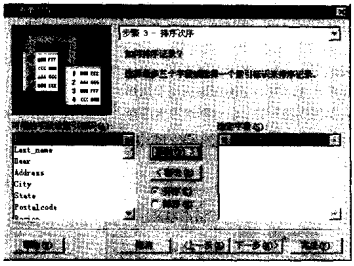


图5.47　选择记录排序

用“表单设计器”建立一个空白表单的操作步骤如下：

开始 从“项目管理器”选择“表单”，选择“新建”。

下一步 在“新建表单”对话框中选择“新建表单”。

下一步 在“表单设计器”中设计你的表单，如图 5.48 所示。

完成 关闭“表单设计器”。

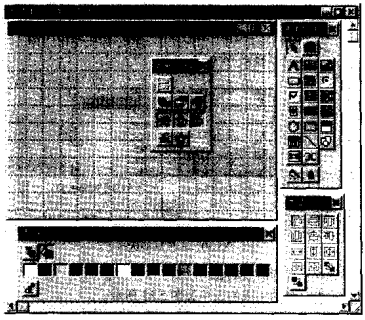


图5.48　新建表单

5.2.2　向表单中快速添加数据字段

当空白表单建立后，可以向表单中添加控件。为了将字段快速地加入“表单”，可以采用下例所示的方法。

向表单中快速添加字段的操作步骤如下：

开始 从“表单设计器”工具栏中，选择“表单生成器”，或者从“表单”菜单中选择“快速表单”。

下一步 在“表单生成器”的字段选择选项卡选择“数据库”。

下一步 在“表单生成器”的字段选择选项卡选择“表”。

下一步 在“表单生成器”的字段选择选项卡选择“字段”，如图 5.49 所示。

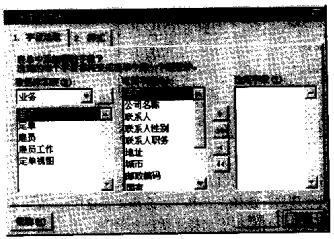


图5.49 设置字段选择

下一步 选择“表单生成器”的风格选项卡，如图 5.50 所示。

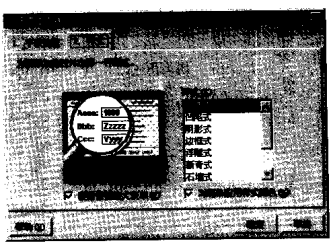


图5.50 表单风格选择

下一步 在风格选项卡选择所需的表单风格。

下一步 按“确定”，自动生成的表单如图 5.51 所示。

下一步 关闭“表单设计器”，在是否保存对话框中选择“是”。

下一步 在“另存为”对话框中输入表单名。

完成 在“另存为”对话框中按“保存”完成表单的快速生成。

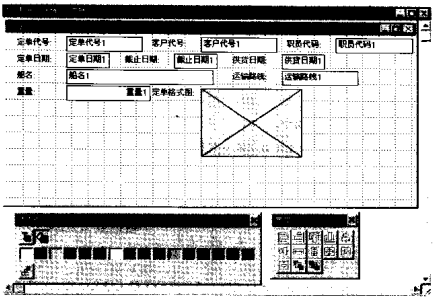


图5.51　表单生成器自动生成的“定单”表单

5.2.3　设置表单数据环境

同“报表”和“标签”的数据环境类似，“表单”的数据环境包括了与“表单”打交道的所有的“表”和“视图”，以及它们之间的关系。在“数据环境设计器”中，你可以方便地建立表单的数据环境。它具有如下的特征：当“表单”打开或运行时，数据环境自动打开表或视图。数据环境中加入的“表”或“视图”的字段，将自动地出现在控件的属性窗口的“Control Source”属性的设置框中，供用户选择相应的数据源。当表单关闭时数据环境自动关闭“表”。

建立设置数据环境的操作步骤如下：

开始　从“项目管理器”选择“表单”，并选择或新建一个表单。

下一步　选择“修改”按钮，进入“表单设计器”。

下一步　在“显示”菜单选择“数据环境”，出现如图5.52所示的表单的“数据环境”。

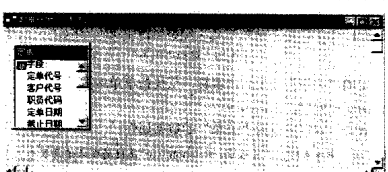


图5.52　数据环境设计器

下一步　按鼠标右键，选择快捷菜单中的“添加”。

下一步　在出现的“添加表或视图”对话框中选择可以添加的表或视图，然后按“添加”。

下一步 按“关闭”关闭“添加表或视图”对话框。

下一步 在“数据环境”上按鼠标右键，选择快捷菜单中的“属性”。

下一步 在“属性”窗口中选择“数据”选项卡，如图 5.53 所示，设置“数据”属性。

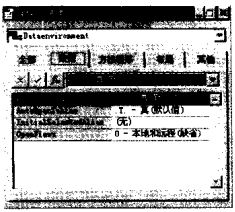


图5.53 设置数据环境“数据”属性

说明 “数据环境”对象的属性中与数据表和视图有关的有四个，它们的功能特点如下：

AutoCloseTables：默认值为“.T.-真”，如果为“真”当“表单”关闭时，自动关闭表和视图。

AutoOpenTables：默认值为“.T.-真”，如果为“真”当“表单”打开或运行时，自动打开表和视图。

InitialSelectedAlias：在设计时为“”。如果没有指定，在运行表单时，首先加到“数据环境”的临时表最先被选定。

OpenViews：缺省值为 0，设置打开的视图的类型。

下一步 在“数据环境”上选择一个“表”或“视图”，按鼠标右键，选择快捷菜单中的“移去”，“表”或“视图”将被移去，与之相关的关系也将被移去。

下一步 在“数据环境”中添加两个具有永久关系的表或视图，“数据环境”中将自动的显示这些关系。

下一步 在“数据环境”中用鼠标单击“关系”线选择关系，如图 5.54 所示。

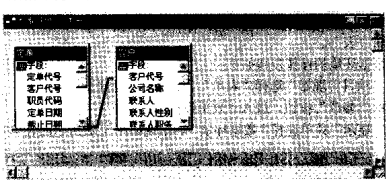


图5.54 选择“关系”

下一步 在选定的“关系”线上右击鼠标，从快捷菜单中选择“属性”，出现如图 5.55 所示的关系对象的属性窗口。

下一步 在“属性”窗口中设置“关系”的数据属性。

说明 “数据环境”中关系对象的属性中，RelationExpr 属性的默认值设置为主表中主关键字段的名称。如果相关表是用表达式当作索引，RelationExpr 属性的值就应该设置为这个表达式。如果关系不是“一对多”关系，OneToMany 属性要设置为“.F.-假”。如果关系是“一对多”关系，OneToMany 属性必须要设置为“.T.-真”。

完成　关闭“数据环境”。

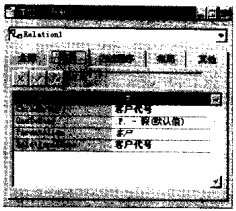


图5.55　关系的属性窗口

5.2.4　向表单添加控件

在 Visual FoxPro 中，“报表”中可以添加各种控件，如域、标签、直线、图片等。同样“表单”中也可以加入对象。Visual FoxPro 中对象的类型有四种：

控件：包括“复选框”、“组合框”、“标签”等；

容器：包括“页框”、“表格”等；

用户自定义类：用户自己定义的对象；

OLE 对象

下面通过实例介绍加入控件方法，在以后的章节中会介绍其它对象的处理。

向表单中添加控件的操作步骤如下：

说明　通过标准的表单控件的工具栏，可以十分方便地向表单中加入标准的控件。

开始　从“项目管理器”选择“表单”，选择“新建”。

下一步　选择“新建表单”。

下一步　在“表单设计器”中选择想要加入的控件，如“标签” A。

下一步　在“表单”上选择想要加入的位置，画出控件，或直接单击选择默认的大小。

下一步　选择“属性”菜单，然后选择“属性”，出现属性窗口。

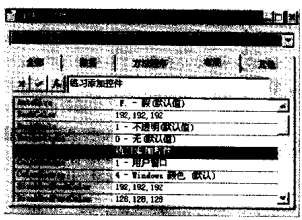


图5.56　更改控件属性

下一步　在“表单设计器”中选择刚刚加入的“标签”，此时属性窗口显示该控件的属性。

下一步　在“属性”窗口中选择“布局”选项卡然后选择“Caption”，在上面的输入框中输入“练习添加控件”如图 5.56 所示。

下一步　在“属性”窗口中选择“布局”选项卡的“FontName”，在上面的输入框中选择“隶书”。

下一步　在“属性”窗口中选择“布局”选项卡的“FontSize”，在上面的输入框中选择“24”。

下一步　在“属性”窗口中选择“其他”选项卡的“Name”，在上面的输入框中输入“lblPractice”。

下一步　选择“表单设计器”中的“标签”，在“布局”工具栏按“水平居中”按钮 和“垂直居中”按钮 。

下一步　选择“表单设计器”中的“标签”，在“布局”工具栏按“水平居中”按钮

[确定]，然后按“垂直居中”按钮[垂直居中]。

[下一步]此时“表单设计器”中的表单如图 5.57 所示。

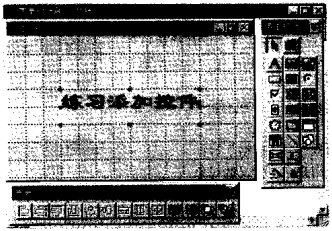


图5.57　向表单中添加控件

[完成]关闭“表单设计器”。

5.2.5　使用“生成器”向表单添加控件

Visual FoxPro 中，通过“向导”可以方便的建立各种应用，而通过它提供的又一个强大的工具“生成器”，用户可以按照一步一步的提示信息把所需要的控件的属性设置好，加入“表单”中。下面通过例子说明生成器的使用。

用生成器向表单中加入控件的操作步骤如下：

[开始]从“表单控件”工具栏中选择“生成器锁定”按钮。

[下一步]从“表单控件”工具栏中选择“文本框”按钮。

[下一步]在“表单”上的适当位置画出文本框。

[下一步]在“文本框生成器”中选择格式选项卡，如图 5.58 所示。

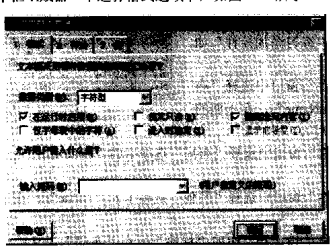


图5.58　设置文本框格式

[下一步]在格式选项卡中输入数据类型和输入掩码。

下一步 在“文本框生成器”中选择样式选项卡，如图5.59所示。

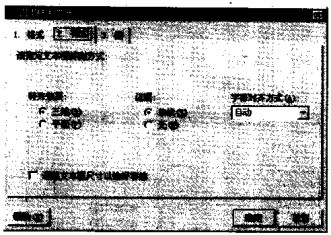


图5.59　设置文本框风格

下一步 在风格选项卡选择文本框显示的方式，是选择“三维”还是“平面”，是否具有边框，以及文本框中文本的对齐方式等。

下一步 在“文本框生成器”中选择值“值”选项卡，如图5.60所示。

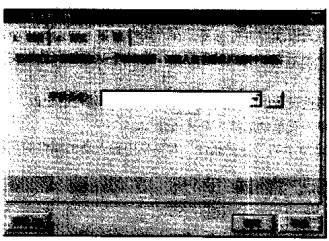


图5.60　设置文本框数据源

下一步 在值选项卡中，可以选择文本框中显示的数据源，如果你已经定义了数据环境，在“字段名”框会出现所有可用的字段，如果所需要的字段不在其中，可以按右侧的 ... 按钮选择需要的“表”。

更成 按“OK”中完成文本框用文本框生成器加入表单的操作。

说明 上面通过生成器添加文本框的方法，也同时适用于其它控件，只是控件生成器的种类不同。

5.2.6　设置控件的属性和方法

在上面的例子中，控件的属性是通过属性窗口设置的。在Visual FoxPro中，可以添加控件的新的属性和方法。

下面首先介绍一下“属性窗口”的结构，如图5.61所示。

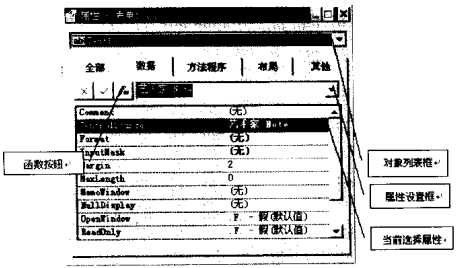


图5.61　属性对话框

添加并修改属性与方法的操作步骤如下：

开始 从“项目管理器”选择上一节建立的“表单”。

下一步 选择“修改”按钮。

下一步 选择“练习添加控件”标签，在“属性窗口”中选择“布局”选项卡，从中选择“Caption”属性，然后在“属性设置框”中将“练习添加控件”文本更改为“练习设置属性”。

下一步 点击“属性设置框”左侧的 ✓ 按钮，完成属性的更改。

下一步 从“表单”菜单中选择“新建属性”。

下一步 在图 5.62 所示的“新建属性”对话框中输入新的属性的名称和该属性的说明。

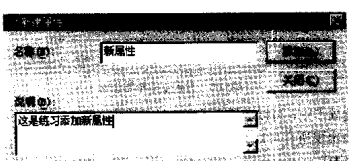


图5.62　添加新属性

下一步 按“确定”完成新属性的添加。

下一步 在“属性窗口”中选择“其他”选项卡。

下一步 在“其他”选项卡中选择“新属性”，然后在“属性设置框”中输入属性值“我的新属性值”。

下一步 从“表单”菜单中选择“新方法程序”。

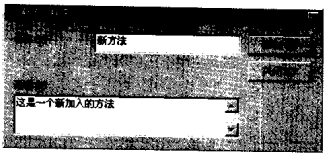


图5.63　添加新方法

在图 5.63 所示的“新方法程序”对话框中输入新方法的名称和该方法的说明。

按“确定”完成新方法的添加。

在“属性窗口”中选择“方法”选项卡。

在“方法”选项卡中选择“新方法”，然后双点击该方法。

在出现的“新方法”代码窗口中加入该方法的代码，如图 5.64 所示。

图5.64　设置新方法的代码

选择标签“练习设置属性”。

单击“调色板”工具栏中的“前景色”按钮，然后选择“红色”，此时“标签”的前景色变为红色。

说明　对于表单上的控制，通常有几种设置方法。可以在表单设计器窗口中拖动对象的大小和位置，用鼠标拖动出长方形包围表单上的多个控件，可以选择控件，然后用对齐工具可以重排控件的位置。用“调色板”工具栏可以设置控件的前景色和背景色。通过“属性窗口”设置属性。

从“表单控件”工具栏中选择“命令按钮”。

在“表单设计器”中画出命令按钮控件。

选择“命令按钮”，然后在属性对话框中设置“Caption”属性为“新按钮”，“Name”属性为“cmdNew”。

双击“新按钮”，出现代码窗口。

在“代码窗口”，对象框选择“cmdNew”，过程框选择“Click”。

在“代码窗口”中输入如图 5.65 所示的代码。

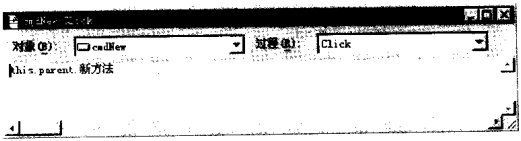


图5.65　输入代码

下一步 关闭“代码窗口”。

下一步 从“表单”菜单中选择“执行表单”，出现如图 5.66 所示的表单。

下一步 在“表单”单击“新按钮”，表单上的标签变为如图 5.67 所示的表单。

完成 关闭运行表单。

通过上面的例子我们对如何设置表单的属性，如何为表单添加属性和方法，如何使用这些方法有了一个全面的了解。对于表单中的控件有多种，这里只是添加了三个控件，设置了它们的属性，并用简单的代码说明了它们之间的协调工作。在以后的程序设计中将详细论述如何建立你的应用程序。

图5.66　运行表单

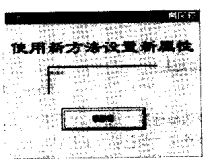


图5.67　单击新按钮后的表单

5.2.7　向表单同时加入多个控件

“表单”中可以同时加入多个控件。

添加多个控件的操作步骤如下：

开始 从“项目管理器”选择一个“表单”或新建一个表单。

下一步 选择“修改”按钮。

下一步 在“表单控件”工具栏，选择“按钮锁定”工具按钮。

下一步 选择工具栏中的“文本框”。

下一步 用鼠标反复单击表单，表单上将加入多个文本框控件。

下一步 在“表单控件”工具栏，选择“按钮锁定”工具按钮，取消选择状态。

完成 关闭“表单设计器”。

5.2.8　设置字段映象

当从“数据环境”中向“表单”中拖拽“字段”或“表”时，将自动出现与字段相对应的控件。字段类型和控件类型的对应可以由用户进行设置。

设置字段映象的操作步骤如下：

开始 从“项目管理器”选择一个“表单”或新建一个表单。

下一步 选择“修改”按钮。

下一步 从“工具”菜单，选择“选项”。

下一步 在“选项”对话框中，选择“字段映象”选项卡，如图5.68所示。

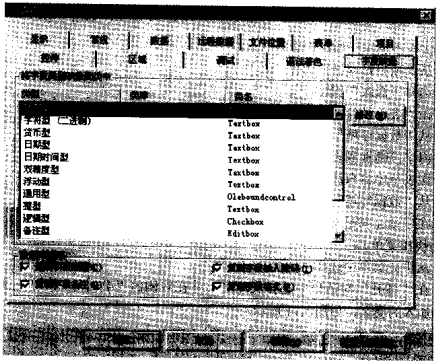


图5.68　设置字段映象

下一步 如果修改某一类型对应的控件的类名，选择该类型，然后按“修改”按钮。

下一步 在“字段映象”对话框中，如图5.69所示，选择对应的库。

下一步 在“字段映象”对话框中的“类名”框选择字段对应的控件的类名，然后按“应用”按钮。

下一步 在“字段映象”对话框中的“类名”框选择字段对应的控件的类名，然后按“应用”按钮。

下一步 在“字段映象”选项卡中，有“数据库选项”选择所需的选项。

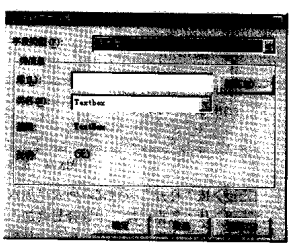


图5.69　修改字段类型的映象

说明 “数据库选项”用来设置数据库字段被加入时所复制的控件类的属性，它们的意义如下：

拖放字段标题：当将表或字段加入表单或者容器时，自动为绑定的控件添加一个标签。

复制字段备注：将把绑定型控件的备注项自动设置为“表设计器”中字段的备注栏的内容。

复制字段输入掩码：把绑定控件的输入掩码属性自动设置为“表设计器”中的字段掩码。

复制字段格式：把绑定控件的格式属性自动设置为表设计器中的字段格式。

第五步 关闭“选项”对话框完成设置。

5.2.9 向表单中快速添加字段的控件

设置了字段映象后，可以向表单中快速的添加所需要的字段的控件，并自动地加入数据源。

向表单快速添加控件的操作步骤如下：

开始 从“项目管理器”选择“表单”。

下一步 选择“修改”按钮。

下一步 在“表单设计器”单击鼠标右键，在快捷菜单中选择“数据环境”。

下一步 从“数据环境”中选择想要加入的字段。

下一步 将字段拖拽到表单中你想要加入的位置。

下一步 在表单中将自动加入与字段映象设置一致的控件。

下一步 从“数据环境”中选择想要加入的字段。

下一步 在字段上按住鼠标右键并将字段拖拽到表单中，出现图 5.70 的快捷菜单。

图5.70　加入控件

下一步 从快捷菜单中选择“创建其它控件”。

下一步 在“打开”对话框中选择类库名，并在右侧的“类名”框选择加入的类名，如图 5.71 所示。

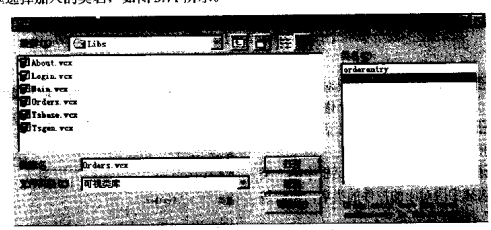


图5.71　选择为字段其它类名

下一步 按“打开”按钮完成向表单加入一个控件。

下一步 从“数据环境”中选择想要加入的表。

下一步 将“表”拖拽到“表单”中你想要加入的位置。

下一步 在“表单”中将自动加入与字段映象设置一致的表的控件通常是一个“网格”控件。

下一步 在“表单”中将自动加入与字段映象设置一致的表的控件通常是一个“网格”控件，如图 5.72 所示。

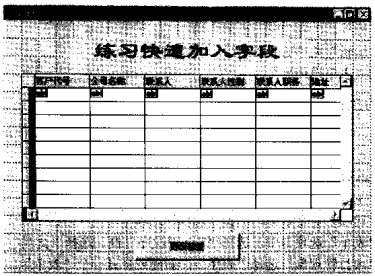


图5.72　给表加入网格控件

说明　此处的表单是对上一节表单的改进，先在数据环境中加入“表”或“视图”，然后将“表”拖到表单中，修改标签的“Caption”属性为“练习快速加入字段”，并用鼠标拖拽控件调整大小，用“布局”工具栏中的工具调整控件相对表单的位置，这些方法在以前都有介绍。

下一步　在“数据环境”、“数据库设计器”或“项目管理器”中选择一个表的多个字段，按住鼠标右键并将表拖拽到表单中，出现如图5.73的快捷菜单。

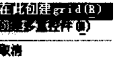


图5.73　创建多重控件

下一步　从快捷菜单中选择“创建多重控件”。

下一步　表单中按照“字段映象”将出现与所选择的字段相对应的控件。

重复　关闭“表单设计器”。

5.2.10　复制删除表单中的控件

当对设计的“表单”进行修改时，可能需要复制一个或多个控件，也可能需要删除一个或多个控件。

复制删除控件的操作步骤如下：

开始　从“项目管理器”选择“表单”，新建或打开一个“表单”。

下一步　在“表单设计器”中用鼠标选择一个控件。

下一步　从“编辑”菜单选择“复制”。

下一步　从“编辑”菜单选择“粘贴”，将控件拖到需要的位置。

下一步　按住“Shift”键，然后用鼠标选择多个控件。

下一步　从“编辑”菜单选择“复制”。

下一步　从“编辑”菜单选择“粘贴”，将多个控件拖到需要的位置。

下一步　用鼠标按住左键拖拽出一个长方形，把需要的控件选定。

下一步　从“编辑”菜单选择“复制”。

下一步 从“编辑”菜单选择“粘贴”，将选定控件拖到需要的位置。

下一步 选择控件。

下一步 从“编辑”菜单选择“剪切”或“清除”。

完成 关闭“表单设计器”。

5.2.11 为控件设置Tab顺序

对用户来说，一个习惯的按 Tab 键的输入顺序十分重要。在表单中可以用多种方法来设置按 Tab 时焦点在控件中的移动顺序。

设置控件 Tab 顺序的操作步骤如下：

开始 从“工具”菜单选择“选项”。

下一步 在“选项”对话框中，选择“表单”选项卡。

下一步 在“表单”选项卡中，选择“Tab 键次序”框，选择“交互”。

下一步 从“项目管理器”选择“表单”，新建或打开一个“表单”。

下一步 在“显示”菜单中，选择“Tab 次序”，此时“表单”中的控件显示当前的 Tab 顺序，如图 5.74 所示。

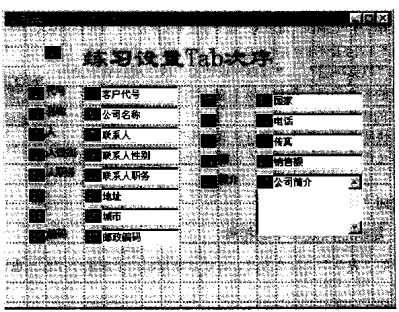


图5.74 设置Tab次序

下一步 上点击想要作为 Tab 顺序第一个的控件。

下一步 按“Tab 顺序”点击表单上的控件，控件左侧将显示当前的顺序。

下一步 用鼠标点击表单中没有控件的位置完成设置或者按 ESC 键取消设置。

下一步 从“工具”菜单选择“选项”。

下一步 在“选项”对话框中，选择“表单”选项卡。

下一步 在“表单”选项卡中，选择“Tab 键次序”框，选择“按列表”。

下一步 在“显示”菜单中，选择“Tab 次序”，此时出现“Tab 键次序”对话框，见图 5.75 所示。

下一步 选择“按行”按钮。

下一步 选择“按列”按钮。

下一步 在列表框中用鼠标拖动按钮的位置，可以调整 Tab 的次序。

下一步 按“确定”完成 Tab 次序的设置。

完成 关闭“表单设计器”。

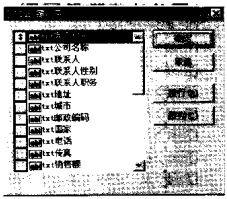


图5.75　按列表设置Tab次序

5.2.12　定制表单

为了使你的表单更加漂亮，显示出专业水准。可以按要求对表单进行定制工作，例如改变表单的前景色、改变表单的显示字体、添加线条等修饰。

1. 用控件定制表单

开始 从“项目管理器”选择“表单”，新建或打开一个“表单“。

下一步 在“表单”中加入一个标签。

下一步 在“属性”窗口中设置标题“Caption”为“定制表单”。

下一步 在“属性”窗口中设置字体“FontName”为“隶书”。

下一步 在“属性”窗口中设置字体“FontSize”为“24”。

下一步 在“属性”窗口中设置字体“FontColor”为“255，0，255”粉红色。

下一步 在“数据环境”中加入“客户”表。

下一步 选择“数据环境”中的“客户”表，按住鼠标右键，拖拽表字段到表单。

下一步 选择快捷菜单“创建多重控件”。

下一步 在表单中选择控件，用“布局”工具栏对标签与文本框进行重新布置，如图 5.76 所示。

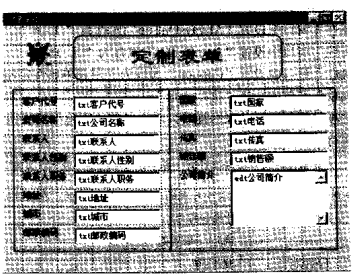


图5.76　定制表单

下一步 从“表单控件”工具栏中选择“形状”按钮。

下一步 在“表单”上的“定制表单”标签处画出一个长方形。

下一步 从“格式”菜单选择“置后”将长方形放在后面。
下一步 从“属性”窗口将长方形的“Curvature”属性设为“20”。
下一步 从“表单控件”工具栏中选择“直线”按钮。
下一步 在“表单”上画出五条直线，见图 5.75。
下一步 从“表单控件”工具栏中选择“图像”按钮。
下一步 在“表单”上的画出图像大小及位置。
下一步 在“属性”窗口设置图像的“Picture”属性，选择一个图像文件。
下一步 在“属性”窗口设置表单的“FillColor”属性为“128，128，192”。
下一步 从“表单”菜单中选择“执行表单”，运行的结果如图 5.77 所示。

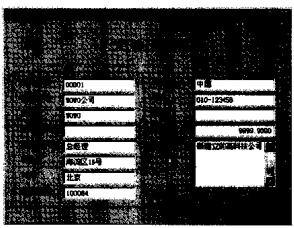


图5.77　表单定制后的运行结果

下一步 关闭“表单”。

2. 用表单属性定制表单

开始 打开上例的“表单”。
下一步 在“属性”窗口中设置“AlwaysOnTop”属性为“.T.-真”。
下一步 在“属性”窗口中设置“AutoCenter”属性为“.T.-真”。
下一步 在“属性”窗口中设置“Caption”属性为“用表单属性定制表单”。
下一步 从“表单”菜单中选择“执行表单”。
下一步 激活“项目管理器”并来回拖动，看运行的“表单”的状态。
下一步 关闭运行的“表单”。

说明 当“表单”在表单设计器中进行设计时，表单处于“活动”状态，及对“表单”外观属性的任何变动都立即能反映到表单上，及所有对表单外观的改动是“可视”的，在设计时，表单的常用属性见表 5.4 所示，它们定义了表单在整体上的外观和行为。

表5.4　表单的外观属性

属性	默认值	属性说明
AlwaysOnTop	.F.	该窗口是否总位于其它打开窗口的上面
AutoCenter	.F.	表单运行时是否在 Visual FoxPro 主窗口居中位置
BackColor	255,255,255	表单窗口的背景色

续表

属性	默认值	属性说明
BorderStyle	3	是否具有边框，3 为系统边框
Caption	Form1	表单标题栏显示的文本
Closable	.T.	用户能否双击“关闭”框关闭表单
DataSessoion	1	表单具有的表是私有还是在全局工作区打开
MaxButton	.T.	是否有最大化按钮
MinButton	.T.	是否有最小化按钮
Movable	.T.	表单是否可以移动
ScaleMode	由“选项”设置决定	控制对象的尺寸和位置的度量单位
WindowState	0-正常	控制表单是最小化、最大化还是正常
WindowType	0-非模式	表单是“模式”表单还是“非模式”表单

5.3　小　结

在这一章介绍了数据库中信息的显示和输出。通过使用“报表”用户可以方便地实现数据库中相关信息的打印和输出，通过“表单”用户可以方便地实现数据库中信息的显示和编辑。为了更有效地实现对数据的管理，你可能将结果打印出来或显示于一个窗口中，Visual FoxPro 数据库提供了“报表”和“标签”，方便你对表中数据和查询结果的打印。而“表单”提供了一个方便、快捷的数据浏览和编辑的方式。

通过本章学习，你应掌握：设计报表和标签、使用表单。

思考与练习

1．报表布局的原则有哪些？建立报表布局有几种方法？各类报表布局应用的典型范围是什么？如果让你分别建立一个学生登记表、一个电话簿、一张名片的报表布局，你将如何选择报表的布局？

2．“报表设计器”中的带区有哪几类？它们在每一页出现的打印范围和频率？

3．“报表”和“标签”中可以使用的控制有哪些？怎样添加这些控制？

4．什么是“数据环境”，它起到了什么作用？什么是“数据源”？它起到了什么作用？

5．如何调整“报表”的布局？如何给布局分组？

6．“表单”、“报表”和“标签”各有什么区别？使用什么方法可以向“表单”中快速添加字段？

7．表单的“数据环境”和报表的“数据环境”意义相同么？如何设置表单的“数据环境”？

8．设置“字段映象”起到了什么作用？

9．如何定制“表单”？

第三篇

设 计 篇

本篇导读

在前面几章中，我们已经学会了如何建立数据库，如何在数据库中建立表和索引。并学会了如何从表中查询和编辑数据，如何建立查询、视图和表单。学会使用这些 Visual FoxPro 最基本的功能后，可以进入 Visual FoxPro 6.0 的另一个更为广阔的天地——Visual FoxPro 程序设计。

在 Visual FoxPro 6.0 中程序设计不在是一件复杂而又枯燥的事情，程序员不必为程序中没完没了的臭虫“Bug”而苦恼。通过一整套全新的开发工具，Visual FoxPro 中的应用程序可以实现自动的信息分类，并对相应的工作进行跟踪和处理。Visual FoxPro 6.0 中提供的交互输入输出界面已经为用户对数据的处理带来了巨大方便，然而，为了使用户的应用功能得到充分发挥，掌握 Visual FoxPro 的设计语言是必要的。

本篇主要包括：

Visual FoxPro 6.0 程序设计简介
Visual FoxPro 6.0 程序语言概论
Visual FoxPro 6.0 面向对象程序设计
Visual FoxPro 6.0 应用程序界面设计
Visual FoxPro 6.0 事件驱动模型

第6章　VFP 6.0 程序设计简介

程序设计是为了完成一项任务而编写的指令的集合，在某种程度上 Visual FoxPro 中的程序设计同传统的程序设计有很大的不同。在 Visual FoxPro 中可以同时应用面向过程和面向对象的编程方法。

如果你刚刚学习编程，请从本章读起。如果你已经学会了一种编程语言或者对可视化编程有一定了解，可以略读本章，快速地进入下一章。

本章主要介绍以下内容：

（1）Visual FoxPro程序设计框架；

（2）Visual FoxPro程序设计过程；

（3）面向对象程序设计的新特性。

6.1　Visual FoxPro程序设计框架

6.1.1　程序设计的特点

既然 Visual FoxPro 为我们提供了那么多建立、查询和编辑数据的工具，为何还要学习编程呢？请看下面通过 Visual FoxPro 工具和通过程序实现查询记录的不同。

如果你想查询在“客户”表中的“联系人”为“WOWO”的所有客户的列表：

1. 使用Visual FoxPro工具手工查询

开始 从“文件”菜单中选择“打开”。

下一步 在“打开”对话框的“文件类型”列表框中选择“表（*.dbf）”。

下一步 在文件列表框中选择“客户.dbf”。

下一步 从“显示”菜单中选择“浏览”。

完成 在“浏览”记录窗口，查找“联系人”为“WOWO”的记录。

2. 使用Visual FoxPro的命令在命令窗口中输入如下的命令

```
USE e:\practice\data\客户.dbf
LOCATE FOR 联系人=“WOWO”
BROWSE
```

当用 USE 命令打开表时，如果默认目录设置为“客户”表所在的目录，则可以不带路径，否则必须指明表所在的路径，或者在“选项”中设置默认路径。

在很多时候，某些任务需要反复执行，并且步骤繁多。在此种情况下，编写一段程序，并反复验证它为正确的，然后通过程序来执行任务，既快速又减少了出错率。

因此在许多情况下使用程序，而不采用工具来执行任务，有如下的特点：

（1）对于需要反复执行的复杂任务，使用程序会快速无误地自动执行任务；

（2）程序可以不断地调试、修改并反复执行，直到运行正确，从而使得大型的复杂应用可以在不断改进中完善起来；

（3）程序之间可以相互的调用，一个进程或者一个存储过程也可以被多个程序调用；

（4）通过程序的流程控制，使得使用程序执行复杂任务成为可能。

6.1.2　Visual FoxPro程序设计机制

在 Visual FoxPro 中，程序由代码组成，代码中可以有 SQL 语句、各种指令和函数。所有这些程序指令可以出现于以下的位置：

（1）命令窗口（Command Window）；

（2）程序文件（.PRG）；

（3）表单设计器的事件/代码窗口；

（4）菜单设计器的过程代码窗口；

（5）报表设计器的过程代码窗口。

6.1.3　使用命令窗口

Visual FoxPro 提供的命令窗口可直接运行程序，也可以直接键入命令，然后按 Enter 键执行命令，对已经执行的命令，在命令窗口中会自动保留，如果需要运行一个键入旧命令，可以将光标移到命令所在的行，然后按 Enter 键，也可对命令做修改，然后按 Enter 键。

命令窗口可以看作是一个编辑窗口，在其中进行命令的编辑时，可以进行插入、删除、剪切、复制和粘贴等操作，也可以将命令窗口的内容剪贴到其他的程序段中。

当利用 Visual FoxPro 的工具，如菜单、工具栏各种设计器、查询等的操作可以马上转化为 Visual FoxPro 的命令，并显示于命令窗口中，用户可以将这些命令复制粘贴到程序中。从而为程序的构造带来了极大方便。

6.1.4　在代码窗口中建立程序

Visual FoxPro 中的程序是由一系列命令组成的文本文件，具有.PRG 的后缀，下面就如何建立单一的简单程序加以说明。

应用程序的建立与执行的操作步骤如下：

开始 从“文件”菜单中选择“新建”。

下一步 选择“新建文件”。

下一步 在出现的“程序 1”的窗口中输入如图 6.1 所示的命令。

```
*查找客户的记录
&&程序注释

SET DEFAULT TO e:\practice\data      &&设置默认目录
OPEN DATABASE 业务                   &&打开数据库
USE 客户                             &&打开表
SET ORDER TO   客户代号              &&排序
SEEK "00001"                         &&查找记录
BROWSE                               &&浏览表
```

图6.1　输入“查找客户”程序

下一步 从“文件”菜单选择“保存”。
下一步 在“另存为”对话框，在“保存文档为”框输入“查找客户”。
下一步 按“保存”保存程序文件。
下一步 从“程序”菜单选择“执行”。
下一步 关闭“查找客户”程序窗口。
下一步 从“命令窗口”输入“MODIFY COMMAND 查找客户”。
下一步 在“查找客户”程序窗口中修改程序。
完成 关闭程序窗口。

上面的例子介绍了程序的建立、修改、保存和运行的所有操作，通过它可以学习到程序建立的整个过程。

6.1.5 用Visual FoxPro的工具建立程序代码

当前一个良好的用户化的应用程序界面，对用户来说十分重要。Visual FoxPro 为用户建立应用程序骨架提供了强有力的工具。

通过“菜单设计器”可以方便地生成菜单，通过“表单设计器”可以快速地生成输入和编辑数据的界面，通过“报表设计器”可以快速地生成“报表”。然而对于具体任务的执行还需要由程序代码来实现。

通过 Visual FoxPro 的工具，用户可以为任务建立框架，同时可以方便地为框架添加相应的应用代码，使得框架丰满起来。程序设计的目的就是为了执行某一项具体的任务，程序本身实际是对现实中实际工作的一种抽象。框架的构造要符合整个工作流程，而具体每一步的任务由具体的程序算法来实现。

6.2 Visual FoxPro程序设计过程

6.2.1 Visual FoxPro程序设计流程

对于 Visual FoxPro 的应用程序，它们的构造过程都是近似的，只在具体的应用代码上有一定的差别。从编写第一行应用程序起，就不要忘记注释和调试。而在编程时要时刻注意特殊情况，设计程序应该完成哪些测试。

程序设计的步骤主要包括：

开始 分析所要解决的任务，找出相关的数据。
下一步 对任务进行分解，进行模块的划分，同时分析相关的数据。
下一步 设计程序的整体模块，并建立原形系统。
下一步 编制具体的模块，在编制中随时测试。
完成 组装各模块进行整体的测试。

注意 程序设计是一个不断反复的过程，为了使这一过程更为合理一定要注意程序初始的准备工作，作好需求分析和初步设计，从而为以后的编码打下良好的基础。另外在开发过程中不断的调试并测试已经编好的代码对开发过程来说十分重要。

6.2.2 确定任务分析数据

在大多数的情况下，数据都是以原始报表的形式存在的。当用户提出了一个新任务时，首要的工作就是确定任务处理信息的范围，并对所涉及的数据进行分析处理，将数据归类，同时确定任务本身所要实现的具体目标，围绕这个目标需要哪些具体的工作。

对于数据的分析要仔细，剔除那些重复的数据，对一些特殊的复杂的数据，要根据情况进行分解。

6.2.3 分解任务与模块划分

利用一个函数或过程来完成一个大而复杂的任务是不现实的。对所给的任务，需要按照工作的流程划分为多个相对独立的步骤。

按照每一个具体的步骤，又可以进一步细化，直到无法继续划分。

6.2.4 建立原形系统

提出任务的用户与应用程序的编制人员，即使紧密配合，在许多问题的理解上也可能出现差异。采用 Visual FoxPro 的强大的可视化的工具，能方便地建立一个“原形系统”，它是程序粗略的骨架，加入了菜单和代表功能的对话框。它本身并不实现具体的功能，只是体现了程序模块功能的划分和程序的外观。

利用“原形系统”用户可以基本了解程序完成后的操作方式、程序大概的外观、各部分大概的功能。对那些在理解上有区别的地方可以很快地提出修改意见，使程序编制人员在起步阶段便和用户保持一致。

6.2.5 组装模块进行整体测试

当某一个独立具体的模块编制完成以后，要将它们嵌入“原形系统”，这是一个给骨架添“肉”的过程。编程人员常说，每一个应用项目，都是一个美丽的少女，当还没有见到她时，时时刻刻想到她的样子，这便是“原形系统”。等她真的出现，它是成为一个讨人喜欢的公主还是成为一个讨厌的丑婆，就要看在程序编制和程序完成后的调试与测试工作是否全面负责了。

常可以看到，一些程序的编制人员“下笔如有神”，代码突飞猛进，可是一句注释也见不到。在正式的开始编程以前，一定要记住：“做好需求、搞好注释、随时调试”是应用程序最后成功的保证。

在 Visual FoxPro 中注释的符号是“*”和“&&”，也可以用“NOTE”来表示注释的内容。

6.3 面向对象程序设计的新特性

6.3.1 从面向过程到面向对象

目前，程序设计的方法已经从传统的面向过程的方法向面向对象的方法转化。前几年的结构化的程序设计方法如今渐渐被面向对象的方法所替代。

在结构化程序设计，是一个自顶向下的逐步求精的设计过程。对许多经验不足的开发

人员，这种方法往往变成了一个一个过程和函数的互相调用，随着项目的增大、复杂程度的增加，用这种方法构造出的程序在调试、维护和代码复用等方面存在许多的弊病。程序修改后再调试变的十分困难，尤其对那些不是本人编写、注释又少的程序，对它们的修改简直是一场灾难。

软件复杂度在不断地增加，程序的规模也不断地扩大。为了提高软件的开发效率，增加代码的利用效率。当前普遍采用了面向对象的编程方法，反映到程序设计上便是可视化编程方法的流行。

Visual FoxPro 完全支持面向对象的程序设计方法，但同时又提供对面向过程的支持。从而为程序设计带来巨大的方便。

6.3.2 面向对象程序设计的基本概念

面向对象的方法，采用从问题领域来解决问题的方法。“对象”成为反映客观世界的事物抽象的实体，具有与它相关的数据和方法。

面向对象的方法涉及到许多基本的概念，如果从广义上理解，它对事物的描述将更容易理解。

对象：对具体的客观事物的表示；

类：一组具有相同特性的对象的抽象定义；

方法：对象的功能，自身的能力；

属性：对象的特征特点；

继承：对象具有的相同结构和差异的变化；

消息：实现对象之间的通讯。

因此，可以这样地理解面向对象，它是现实世界的缩影，一个对象具有自己特有的属性（私有数据和方法），同时又具有为自身或他人执行的功能（方法），通过对象之间的通讯，对象之间实现协同工作，共同解决实际的问题。

在 Visual FoxPro 中系统提供了大量对象，如“表单”、“报表”、各种控件。它们都是程序可以使用的对象，用户也可以根据需要建立自己的对象，并为对象提供相应的属性和方法。下面简单介绍 Visual FoxPro 中常用的几个有关面向对象程序设计的术语。

1. 对象（Object）

对象是私有数据和对这些数据进行处理的操作（方法）相结合的实体。一个“表单”可以看作是一个对象，“表单”中的一个“命令按钮”一张图片也可以看作对象。

2. 属性（Property）

属性定义了对象所具有的数据，它是对象所有特征数据的集合。

3. 方法（Method）

方法是对象功能的定义，是实现对象具体操作的代码，方法和消息相对应，从而完成对象的“如何做？”和“怎么做？”的问题。

4. 消息（Message）

消息是要求对对象进行具体的操作的指令，通过对象之间的消息传递，可以实现整个

系统的协调工作。

5. 事件（Event）

事件是用户或系统的动作所引发的事情，例如单击鼠标就发生了一个“Click”事件。为了响应事件可以为事件加入响应的代码，也可以执行某个方法。

6. 类（Class）

类是一种类型的定义，它将属性和方法相统一，把具有相似特征和行为的对象形成一个个结构。类是具有相同或近似特性的对象的抽象，对象是类的具体的实例。类可以具有子类（Subclass），子类继承了父类的属性和方法，并可以附加自己的特定的属性和方法。父类和子类具有继承关系，从而为软件开发带来巨大的方便。

面向对象的编程方法，使用户从原来的对过程的分析转化为对系统对象的分析。程序的开发重心变成了对对象的抽象和类的定义，从而更加符合现实世界中人们的思维方式。

6.4 小 结

Visual FoxPro 中的程序设计同传统的程序设计有很大的不同。通过本章的学习读者可以了解到 Visual FoxPro 中有面向过程和面向对象两种编程方法，并对这两种方法有一个大概的了解。

通过本章应掌握以下内容：Visual FoxPro 的程序设计框架结构、Visual FoxPro 程序设计过程、Visual FoxPro 面向对象程序设计的新特性。

思考与练习

1. Visual FoxPro 中为什么要使用程序来执行某些特定的任务？
2. Visual FoxPro 中的程序设计要经过哪几步？要注意哪些问题？
3. 面向对象的程序设计有哪些新的特点？什么是对象、类、属性、方法、事件和消息？

第7章　VFP 6.0 程序语言概论

Visual FoxPro 虽然提供了大量的工具，能方便地完成对数据库的操作，实现用户对数据信息的分类和查询。然而仅仅有这些是不够的，Visual FoxPro 还提供了许多高级语言的特性。

同其它高级语言类似，Visual FoxPro 提供了多种数据类型，可以将这些数据类型的数据存储于表、数组、变量和其它的数据容器中。可以使用各种操作符，也可以使用 Visual FoxPro 提供的丰富的函数和各种命令。

在本章将对 Visual FoxPro 语言的特性做简单的说明，在语言上它同其它语言十分类似。

本章主要包括：

（1）语法和命名规则；
（2）数据和字段类型；
（3）数据的存储类型；
（4）操作符和表达式；
（5）自定义函数和过程；
（6）程序的流程控制。

7.1　语法和命名规则

7.1.1　变量的命名

变量可以是任何数据类型的数据，用户根据需要可以对变量进行赋值，它存储于内存，当退出程序时，变量所占的内存自动释放。

变量的命名遵循如下的原则：

语法：[Scope]TypeName

说明　*[Scope]* 可选项，是变量的前缀，用来提醒变量的应用范围。TypeName 变量的名称。

[Scope]常用的类型如表 7.1 所示。

表7.1　数据变量的范围

Scope	范围含义	描述	举例
L	Local	本地数据	lnCounter
P	Private	私有数据	pnStatus
G	Public (global)	全局数据	gnOldRecno
T	Parameter	参数数据	tnRecNo

说明　Type 用来变量的数据类型，前缀的推荐方式为表 7.2 所示。

表7.2　数据的类型

Type	代表数据类型	描述	举例
a	Array	数组型	aMonths
c	Character	字符型	cLastName
y	Currency	货币型	yCurrentValue
d	Date	日期型	dBirthDay
t	Datetime	日期时间型	tLastModified
b	Double	双精度型	bValue
f	Float	浮点型	fInterest
l	Logical	逻辑型	lFlag
n	Numeric	数值型	nCounter
o	Object	对象型	oEmployee
u	Unknown	未知型	uReturnValue

上面介绍的命名规则不是必须的，而只是一种推荐你养成一种良好的编程习惯，这种编程习惯为大多数程序员所遵守。

用于范围的前缀，不是必须的。

7.1.2　对象的命名

对象的命名遵循以下原则：

语法：PrefixName

说明　Prefix 为前缀是可选项表明对象的类型，Visual FoxPro 中常用对象的前缀如图 7.3 所示。

表7.3　对象的前缀

Prefix	代表对象类型	描述	举例
chk	CheckBox	复选框	chkReadOnly
cbo	ComboBox	组合框	cboEnglish
cmd	CommandButton	命令按钮	cmdCancel
cmg	CommandGroup	命令组控件	cmgChoices
cnt	Container	容器控件	cntMoverList
ctl	Control	控制控件	ctlFileList
<user-defined>	Custom	用户自定义对象	user-defined
edt	EditBox	编辑框	edtTextArea
frm	Form	表单	frmFileOpen
frs	FormSet	表单集	frsDataEntry
grd	Grid	表格	grdPrices
grc	Column	列对象	grcCurrentPrice

续表

Prefix	代表对象类型	描述	举例
grh	Header	标头对象	grhTotalInventory
img	Image	图像控件	lblHelpMessage
lbl	Label	标签控件	lblHelpMessage
lin	Line	直线控件	linVertical
lst	ListBox	列表框	lstPolicyCodes
olb	OLEBoundControl	OLE 绑定型控件	olbObject1
ole	OLE	OLE 控件	oleObject1
opt	OptionButton	选项控件	optFrench
opg	OptionGroup	选项组控件	opgType
pag	Page	页面对象	pagDataUpdate
pgf	PageFrame	页面框架集	pgfLeft
sep	Separator	分隔符对象	sepToolSection1
shp	Shape	形状控件	shpCircle
spn	Spinner	微调控件	spnValues
txt	TextBox	文本框控件	txtGetText
tmr	Timer	时间控件	tmrAlarm
tbr	ToolBar	工具栏对象	tbrEditReport

7.1.3　表字段的命名

表字段的命名遵循如下的原则：

语法：Alias.TypeName

说明　Alias 为表的别名，Type 代表了表中的字段类型。Type 常用的类型如表 7.4 所示。

表7.4　字段的命名类型

Type	类型名称	描述	举例
c	Character	字符型	Customer.cLastName
d	Date	日期型	Customer.dBirthDay
t	Datetime	日期时间型	Customer.tLastMod
b	Double	双精度型	Customer.bRate
f	Float	浮点型	Customer.fValue
g	General	通用型	Customer.gPicture
l	Logical	逻辑型	Customer.lSellMail
m	Memo	备注型	Customer.mComments
y	Currency	货币型	Customer.nItems
n	Numeric	数值型	Customer.nItems
i	Integer	整型	Customer.iCustID

7.1.4 变量的命名

变量可以是任何数据类型的数据，用户根据需要可以对变量进行赋值，它存储于内存，当退出程序时，变量所站的内存自动释放。

变量的命名遵循如下的原则：

语法：[Scope]TypeName

说明　*[Scope]* 可选项，是变量的前缀，用来提醒变量的应用范围。TypeName 变量的名称。

[Scope]常用的类型如表 7.5 所示。

表7.5　数据变量的范围

Scope	范围含义	描述	举例
l	Local	本地数据	lnCounter
p	Private	私有数据	pnStatus
g	Public (global)	全局数据	gnOldRecno
t	Parameter	参数数据	tnRecNo

7.2 数据类型与字段类型

7.2.1 数据类型与字段类型

在 Visual FoxPro 中所有的数据都具有类型，类型是对值允许的值和值的范围的说明。一旦为数据指明了类型，这些数据便能够被存储，能够通过数据变量和数组对数据进行处理。当建立“表”时，对表中的每个字段都需要指定类型名称。

Visual FoxPro 中的数据类型如表 7.6 所示。

表7.6　Visual FoxPro数据类型

类型	中文名称	类型说明	大小	范围
Character	字符型	任意文本	最多 254 个字符	任意字符
Currency	货币型	货币值的数量	8 字节	922337203685477.5808 到 922337203685477.5807
Date	日期型	年、月、日的日期数据	8 字节	01/01/100 到 12/31/9999
DateTime	日期时间型	年月日和时分秒的时间数据	8 字节	日期从 01/01/100 到 12/31/9999,时间从 00:00:00 a.m. 到 11:59:59 p.m.
Logical	逻辑型	“真”或“假”的布尔型	1 字节	“真”(.T.) 或“假”(.F.)
Numeric	数值型	整数或分数	8 字节	.9999999999E+19 到.9999999999E+20

Visual FoxPro 中只用于表中字段的数据类型如表 7.7 所示。

表7.7　Visual FoxPro数据字段类型

类型	中文名称	类型说明	大小	范围
Double	双精度型	双精度浮点数	8 字节	从 +/-4.94065645841247E-324　到 +/-8.98846567431 15E307
Float	浮点型	与数值型相同	内存中 8 字节 表中 1-20 字节	从.9999999999E+19 到.9999999999E+20
General	通用型	OLE 对象的引用	表中 4 字节	受可用的内存空间的限制
Integer	整型	整数值	4 字节	从-2147483647 到 2147483646
Memo	备注型	引用大的数据块	表中占 4 字节	受可用的内存空间的限制
Character (Binary)	字符型（二进制）	任何未经过代码页修改的字符数据	254 字符	任意字符
Memo (Binary)	备注型（二进制）	任何未经过代码页修改的备注字段数据	表中占 4 字节	受可用的内存空间的限制

7.2.2　数据类型与字段类型的说明

上一小节介绍了在 Visual FoxPro 中的数据类型，下面对每种类型加以详细的说明，介绍每种数据类型的特点。

1. 字符型

字符型的变量或字段保存字符数据，如姓名、地址等文本数据。有些数据虽然由数字组成，但主要是处理的文本信息，也最好定义为字符型，例如电话号码和邮政编码等。

字符型常量的引用，是通过将字符加上单引号“'”或者加上双引号“""”来引用的。例如：客户.客户代号="00001"。

2. 货币型

请采用货币型来保存货币值。如果在货币型表达式的小数为超过了 4 位，Visual FoxPro 将在处理之前对它进行四舍五入到四位。

如果给货币型数据赋值，要在数值前加上“$”符号，如：

收入 = $50.33

3. 日期型

日期型保存不带时间的日期值，日期变量采用“yyyymmdd”的字符格式保存。

给日期型变量赋值，应该把日期放于括号内，如：

dLastAppointment = {10/14/94}

如果使用空的日期值，请用如下的格式赋值：

STORE {} to dBlankdate0

STORE { } to dBlankdate1

STORE {/} to dBlankdate2

日期型的格式取决于对SET DATE, SET MARK TO, 和 SET CENTURY的设置。

4. 日期时间型

当既需要保存日期又需要保存时间时，要使用日期时间型。日期时间型的数据存于两个四字节整数的 8 个字节中，前一个四字节保存日期，后一个保存时间。时间从午夜算起，最小的计时单位百分之一秒。如果不指定日期值，则系统默认为为 1899 年 12 月 30 日。如果没有指定时间值，默认值为午夜值。

给日期时间型赋值采用如下方式：

datetime = {4/1/95 10:00am}

timeonly = {10:00am}

如果指定空的时间值，请在括号中加入“:”号：

STORE {/:} to tBlankdate1

日期时间值的日期取决于 SET DATE, SET MARK 和 SET CENTURY 的设置，时间的格式取决于 SET HOURS 和 SET SECONDS 的设置。

对于日期和时间型来说，都有如下的等价关系：

{00:00:00AM} 等价于午夜 {12:00:00AM}

{00:00:00PM} 等价于中午 {12:00:00PM}

从{00:00:00} 到 {11:59:59} 等价于 {12:00:00AM} 到 {11:59:59AM}

从{12:00:00} 到 {23:59:59} 等价于 {12:00:00PM}到 {11:59:59PM}

5. 双精度型

双精度型的存储精度比数值型高，可以存储一些精密的实验数据。在表中输入双精度型数据时，是由你自己决定小数点输入的位置的。

6. 浮点型

浮点型同于数值型，它是为了提供兼容性而设置的类型。

7. 通用型

通用型用来保存 OLE 对象。通用型字段中并没有保存真正的 OLE 对象，而只是保存了一个 10 个字节的引用，也可以认为是一个指针，它指向真正的内容：文档、图片等。而 OLE 对象的实际大小与类型取决于建立 OLE 对象的服务器程序，以及你是采用联接还是嵌入的方式引入对象。如果是联接方式，只是存储了一个指向对象的引用值。如果是嵌入方式，数据表中还容纳了一个应用的副本。通用型字段的大小只是受磁盘大小的限制。

8. 整型

如果对性能和表的存储空间的限制很严格，可以用整型存储整数信息。在表中，整型字段存储的是一个 4 字节的二进制值，因而比其它的数值类型占据更少的空间，并且不需要 ASC II 转换。

9. 逻辑型

逻辑型用来存储逻辑值“真”（.T.）和“假”（.F.），这对于只需要存储是、否，对、错，真、假的数据十分有效。

10. 备注型

备注型的字段可以用来存储一个数据块，备注字段存储了一个 10 个字节的引用，而备注的大小，取决于你输入了多少数据。记录中的备注字段的数据存储于另一个与表名相同的文件中，这个文件具有.FPT 后缀。备注字段的大小只取决于磁盘空间的大小。

Visual FoxPro 中的备注字段可以存储任意类型的数据，可以为空值、编译代码等。

11. 数值型

数值型用来存储数量值，可以是正数，也可以是负数。可以是整数，也可以是小数。Visual FoxPro 中支持 16 进制的数值，例如 255 可以用 0xFF 代替。

对于数值型字段，在设计时可以决定小数位数，小数位的长度是整个字段长度的一部分。例如，如果数值字段的长度定为 6 位，小数定为 4 位，则字段的最大值可以是 9.9999。

7.3 数据的存储类型

7.3.1 数据的存储类型

选择数据的存储类型，需要根据数据的类型以及数据容量的大小。在程序中，数据的声明方法、位置，声明的类型决定了数据的访问方式。数据的作用域规定了数据的访问范围和有效性。

Visual FoxPro 中，你不但可以将数据存储于常量、变量和数组中，还可以将数据存储于记录和对象中。

Visual FoxPro 中的数据容器如表 7.8 所示。

表7.8　数据容器的类型

数据容器	中文名称	作用域	定义方法
Constants	常量	私有	#DEFINE ERRSTR "Error!"
Variables	变量	全局、私有、局部	Var = 7
Arrays	数组	全局、私有、局部	ArrayName[1,1] = "John Brown"
Fields	字段	永久存储，表打开时可以访问	REPLACE name WITH "John Brown"
Object Properties	对象属性	通过对象和对象容器被引用	txtCustomer.Value = "John Brown"

7.3.2 常量

常量在程序的整个操作过程中保持不变。例如圆周率 π 的值是 3.1415926，它是数值型常量的代表。

在 Visual FoxPro 应用程序中创建的常量，在操作中保持不变。在程序的设计阶段，可以改变程序的某一个常量值而影响整个的应用程序。

常量可以是任何一种类型的数据。

给常量赋值可以用以下方式：

#DEFINE TABLERR1 "此表错误！！"

取消已经定义的常量可以用#UNDEFINE 命令，释放常量而节省内存，如：

#UNDEF TABLERR1

7.3.3 变量

变量在程序的整个操作过程中可以改变，可以存储任何类型的数据并可以在任何时候改变它们。

变量的赋值可用如下的方法：

1) 用“STRORE” 命令：STORE 666 TO nMyVariable

2) 用“=”操作符：nMyVariable=666

变量只有在应用程序创建时才发生作用，并且具有作用域的规则。Visual FoxPro 中可以用 LOCAL 、PRIVATE 和 PUBLIC 来指定变量的作用域。

(1) LOCAL（局部变量）：此种类型创建的变量只能在当前的模块或过程中使用和修改，而不能被其他的过程所调用。当创建进程时，变量被分配内存，所在的进程结束时自动地释放内存。

用 LOCAL 定义局部变量的语法：

LOCAL VarList

或者：LOCAL [ARRAY] ArrayName1

(*nRows1* [, *nColumns1*])[, ArrayName2 (*nRows2* [, *nColumns2*])] ...

说明 *VarList*：一个或多个局部变量的列表。

[ARRAY] ArrayName1 (nRows1 [, nColumns1])[, ArrayName2 (nRows2 [, nColumns2])] ... ：声明一个或多个局部数组

(2) PRIVATE（私有变量）：私有变量在调用程序中将自身隐藏。从而使得在当前程序段中对变量的引用不会影响变量先前的定义。当拥有私有变量的程序段结束时，所有的被声明为私有的变量或者数组都可以重新被访问。

用 PRIVATE 定义私有变量的语法：

PRIVATE *VarList*

或者：PRIVATE ALL[LIKE

Skeleton | EXCEPT *Skeleton*]

说明 *VarList*：一个或多个私有变量的列表。

ALL LIKE *Skeleton*：将所有与 *Skeleton* 定义的类型相匹配的数据隐藏起来，*Skeleton* 中可以包含? 和*.

ALL EXCEPT *Skeleton*：将所有与 *Skeleton* 定义的类型不相匹配的数据隐藏起来，*Skeleton* 中可以包含? 和*.

用 PRIVATE 进行声明并不产生新的变量，只是把在高层程序中声明的变量，在当前的程序中隐藏起来。

用 PRIVATE 声明的例子：

程序 7.1　PRIVATE 声明变量

```
***使用 PRIVATE ***
SET TALK OFF
val1 = 10                                          && 给变量赋初值
val2 = 15

DO down
? val1, val2                                       && 显示 10, 100

PROCEDURE down                                     && down 过程演示 PRIVATE 范围作用
PRIVATE val1                                       &&将 val1 从上一层隐藏起来
val1 = 50
val2 = 100                                         &&val2 值发生改变
?'        Val1        Val2'
? val1, val2                                       && 显示 50, 100
RETURN
```

(3) PUBLIC（全局变量）：全局变量在当前的 Visual FoxPro 工作器中，任何程序都可以调用和修改全局变量和全局数组，命令窗口中创建的变量和数组将自动地被赋予全局属性。

用 PUBLIC 定义全局变量的语法：

PUBLIC MemVarList

或者：PUBLIC [ARRAY] ArrayName1

(*nRows1* [, *nColumns1*])[, ArrayName2 (*nRows2* [, *nColumns2*])] ...

说 明　*MemVarList:*　一个或多个被声明为全局变量的列表。

[ARRAY] ArrayName1 (nRows1 [, nColumns1])[, ArrayName2 (nRows2 [, nColumns2])] ...: 将要声明的一个或多个全局数组。

用 PUBLIC 声明的全局变量在没有赋值时被初始化为（.F.），如果你想使用 PUBLIC 的变量，在赋值以前必须加以声明。如果你先赋值然后在声明为 PUBLIC，则 Visual FoxPro 将会抱错。

用 PUBLIC 命令声明的例子：

程序 7.2　PUBLIC 声明变量

```
***使用 PUBLIC***
SET TALK OFF
PUBLIC val1,val2
val1 = 10                                          && 给全局变量赋初值
val2 = 15
```

```
DO down
? val1                                          && 显示 10
? val2                                          && 显示 100

RELEASE ALL                                     && 只释放私有的变量
DISPLAY MEMORY LIKE val?                        && 显示全局变量
RELEASE val1,val2                               && 公共变量必须直接引用加以释放
DISPLAY MEMORY LIKE val?

PROCEDURE down                                  && down 过程演示 PUBLIC 作用域
PRIVATE val1
val1 = 50
val2 = 100
? val1                                          && 显示 50
? val2                                          && 显示 100
RETURN
```

Visual FoxPro 中，字段的优先权大于变量的优先权，如果变量与字段同名，可以在变量的前面加上“m.”或者“m->”前缀对变量加以引用，如下所示：

```
?  m.cName
?  m->cName          &&打印变量
?  cName             &&打印字段
```

7.3.4　数组

数组是存于内存中的一系列元素值，通过数组元素的下标可以对数组中的元素加以引用。由于数组元素存于内存中，你可以快速、简单地对数组元素进行定位和处理。

数组和变量具有相同的作用域原则，一个简单的 Visual FoxPro 数组可以存储任何类型的数据。

声明一个数组可以使用 DIMENSION 或 DECLARE 命令，赋值可以直接引用数组的下标，例如：

```
DIMENSION ArrayName[5,2]
ArrayName[1,2] = 966789
```

数组提供了一种对信息快速排序的方法，当信息存于数组中时，你可以方便地对它们进行查询、排序等操作。可以从数组中提取或保存数据，也可以将表中的数据转化为数组的形式或者将数组中的数据存于表中。

Visual FoxPro 对数组的大小和类型不加任何的限制，唯一的限制是内存空间的大小。

7.3.5　字段

字段是在记录中，具有特定的名称和数据类型的一个特定位置，字段可以是 Visual

FoxPro 中的任何数据类型和字段类型。字段的名称是在数据表建立时给定的。

7.3.6 记录

记录是表中的一行，每一条记录都是由字段组成。在每条记录中最多可以容纳 255 个字段。在一张数据表中，“行”可以被认为是“记录”，而“列”则可以被认为是字段。对每一列，具有相同的字段名和大小。每一行都是同一个记录，代表了一个具体的实例。记录的添加可以使用 APPEND 命令，记录的插入可以采用 INSERT-SQL 命令。

7.3.7 对象

每一个具体的对象是类的一个实例，类是对具有相同特征的对象集的特征描述。Visual FoxPro 中的系统对象包括“表单”、“控件”或“数据环境”等。利用对象的属性和方法可以方便地实现应用程序的构造，用户通过自己定义对象，可以有效地提高程序的可靠性，增加代码的重用率。

记录是表中的一行，每一条记录都是由字段组成。在每条记录中最多可以容纳 255 个字段。在一张数据表中，“行”可以被认为是“记录”，而“列”则可以被认为是字段。对每一列，具有相同的字段名和大小。每一行都是同一个记录，代表了一个具体的实例。记录的添加可以使用 APPEND 命令，记录的插入可以采用 INSERT-SQL 命令。

7.4 操作符

7.4.1 字符操作符

字符操作符用来联接比较字符串数据，表 7.9 按优先级列出了字符操作符及其含义。

表7.9 字符操作符

操作符	引发动作	例子代码
+	联接两个字符串、一个字符串和一个字段或者一个字符串和一个变量	?'努力' + '学习'
-	将两个联接串的尾部空格删除并联接两个字符串、一个字符串一个变量或者一个字符串和一个字段	? 客户.性别 - 客户.名称
$	在一个字符串中查找另一个字符串，看第一个字符串是否在第二个中出现	?'努力'$'努力学习' ?'WOWO'$ 客户.名称

7.4.2 日期和时间操作符

表 7.10 为日期和时间操作符。

表7.10 日期时间操作符

操作符	引发动作	例子代码
+	相加	tNewTime = tTime1 + nSeconds dNewDate = dDate1 + nDays
-	相减	nSeconds = tTime1 - tTime2

7.4.3　逻辑操作符

表 7.11 按优先权列出逻辑操作符。

表7.11　逻辑操作符

操作符	引发动作	例子代码
(　　)	分组表达式	cVar AND (cVar2 AND cVAR3)
NOT, !	逻辑“非”	IF NOT cVarA = cVarB IF ! nVar1 = nVar2
AND	逻辑“与”	lVar0 AND lVar9
OR	逻辑“或”	lVarX OR lVarY

7.4.4　关系操作符

表 7.12 为关系操作符。

表7.12　关系操作符

操作符	引发动作	例子代码
<	小于	? 1 < 2
>	大于	? 1 > 2
=	等于	? cVar1 = cVar
<>, #, !=	不等于	.T. <> .F.
<=	小于等于	? {01/01/92} <= {01/01/92}
>=	大于等于	? 32 >= nHisAge
==	比较是否相等	? status == "Open"

7.4.5　数值操作符

表 7.13 为按优先权列出顺序数值操作符。

表7.13　数值操作符

操作符	引发动作	例子代码
(　　)	分组	(4-3) * (12/nVar2)
**, ^	幂运算	? 3 ** 2 ? 3 ^ 2
*, /	乘除	? 2 * 7 ? 14 / 7
%	模运算	? 15 % 4
+, -	加减	? 4 + 15

7.5 表达式

7.5.1 表达式的名称命名规则

表达式中的变量、对象与函数参数都需要名称。Visual FoxPro 中的命名规则如下所示：

（1）名称可以包含中英文字符、数字、下划线；

（2）名称的开头只能是中英文字符和下划线；

（3）名称长度可以为1到254个字符，但自由表的字段名和索引名最多只能有十个字符大小；

（4）名称不要使用Visual FoxPro保留字。

例如：

合法名称：名称，Name，First_Last，S1，_BeginTime

非法名称：REPLACE，123 号，3Men，we　type

7.5.2 字符表达式

字符表达式用字符操作符和下面列出的 Visual FoxPro 的元素构成：

（1）字符型的字段；

（2）返回值为字符型的函数；

（3）存储字符值的变量或数组元素；

（4）字符串常量。

用([ ])将字符串包含起来，可以实现对(" 和 ')的引用，例如：

```
STORE [Robert's Diner] TO cCompanyName
STORE [他说："多好的天气啊！"] TO cBanner
```

例如：

合法表达式：名称+地址，First_Last$Name，客户.姓名-"("+客户.电话+")"，

```
    名称+SUBSTR("Hello, world",n,5)    -地址
```

非法表达式：REPLACE+名称，123+"中国"

7.5.3 日期表达式

日期表达式用日期操作符和下面列出的 Visual FoxPro 的元素构成：

（1）日期或日期时间型的字段；

（2）返回值为日期或日期时间型的函数；

（3）存储日期或时间值的变量或数组元素；

（4）日期或时间型的常量。

例如：

合法表达式：时间间隔 = 终止时间 - 起始时间，

```
    DATE（） + nDays，tTime1 – nSeconds
    职员.下班时间-职员.上班时间
```

非法表达式："时间" +DATETIME（）

7.5.4　数值表达式

数值表达式用数值操作符和下面列出的 Visual FoxPro 的元素构成：

（1）数值型的字段；

（2）返回值为数值型的函数；

（3）存储数值的变量或数组元素；

（4）数值型的常量。

例如：

合法表达式：5**3+6/3+33%6

（9-7）*（30*nVarible）*客户.数量

SUM(MaxOrdAmount)-500

7.5.5　逻辑表达式

逻辑表达式用逻辑操作符和下面列出的 Visual FoxPro 的元素构成：

（1）逻辑型的字段；

（2）返回值为逻辑型的函数；

（3）存储逻辑值的变量或数组元素；

（4）结果为逻辑值的表达式。

Visual FoxPro 中逻辑表达式的值只有“真（.T.）”和“假（F.）”，表达式从左到右计算逻辑表达式的值。

例如：

合法表达式：cVar AND (cVar2 OR cVAR3)

.T. AND .T. AND .F. AND .T. AND .T. AND .T.

7.5.6　名称表达式

名称表达式可以用来字符型的变量或者数组元素的值。因为许多 Visual FoxPro 的命令和函数要求提供名称，在 Visual FoxPro 中可以创建名称表达式。

当把名称存于变量或数组元素中时，用小括号将内存变量括起来，可以将名称替换成命令或者函数。

例子：REPLACE 命令需要一个字段的名称，将字段名存于某一变量当中，然后可以使用该名称表达式：

```
STORE '名称' TO cVarName
REPLACE (cVarName) WITH '中国'
```

7.5.7　使用宏

在程序中可以使用宏替换的方法，用宏替换名称。名称表达式的运算速度比宏的运算速度要快。

宏替换的语法：& VarName[.cExpression]

使用宏替换的例子：

程序 7.3　使用宏替换

```
x = "Fox"
? "&x.Pro"                                          &&显示 FoxPro
STORE  '客户'  TO  gcTableName
STORE  '公司'  TO  gcTagName
USE  &gcTableName  ORDER  &gcTagName                &&使用宏替换
USE (gcTableName) ORDER (gcTagName)                 &&名称变量
```

7.6　程序的流程控制

7.6.1　流程控制介绍

在 Visual FoxPro 中，有两类流程的控制：条件分支和循环。它们的应用形式和其它的程序设计语言类似。

下面通过一个实例来描述流程的应用。

使用流程的例子如下：

说明　本实例完成给 10000 个雇员加薪的任务. 在这 10000 名雇员中，年薪大于或等于 300 的雇员涨 5%的工资，年薪低于 300 的雇员涨 10%的工资。本例使用了循环和分支两种方法来实现对流程的控制。

程序 7.4　使用流程

```
SCAN                                  &&在 SCAN 和 ENDSCAN 之间的代码的执行次数与表中具有的记录
                                      &&个数相等当执行完一次以后记录的指针移到下一条记录。
  IF salary >= 300.00                                  &&当工资高于 300 时工资涨 5%
    REPLACE salary WITH ;                              &&表示下一行为本行继续执行
        salary * 1.05
    ELSE                                               &&当工资底于 300 时工资涨 10%
      REPLACE salary WITH ;
          salary * 1.10
  ENDIF                                                &&结束 IF
ENDSCAN                                                &&完成对表的更改结束 SCAN
```

7.6.2　条件分支

Visual FoxPro 中的分支有两种：

（1）用 IF...ELSE...ENDIF

语法：IF *lExpression* [THEN]
　　Commands
　　[ELSE
　　Commands]

ENDIF

说明　*lExpression* 逻辑表达式，如果该表达式为(.T.)则执行下面的语句，而不执行 ELSE 语句。如果表达式的值为(.F.)并且包含 ELSE 语句，则执行 ELSE 后的语句。如果如果表达式的值为(.F.)并且不包含 ELSE 语句，则不执行任何内容。

IF…ENDIF 实现分支程序的例子如下：

说明　本实例用 IF…ENDIF 语句实现程序的分支程序，通过打开示例程序的数据库并执行查找操作。

程序 7.5　使用 IF…ENDIF 实现分支

```
CLOSE DATABASES
OPEN DATABASE (HOME( ) + 'practice\data\业务')          &&打开数据库
USE 客户                                                &&  打开客户表

GETEXPR '输入查询的条件' TO gcTemp;
TYPE 'L' DEFAULT '公司 = ""'
LOCATE FOR &gcTemp                                      &&  输入查找的表达式
IF FOUND( )                                             &&  找到么?
   DISPLAY                                              &&如果找到显示记录
ELSE                                                    &&  如果没有找到
   ? '没有找到 ' + gcTemp + ' 的记录 '                  &&  显示未找到的信息
ENDIF
USE                                                     &&关闭当前工作区
```

（2）用 DO CASE …ENDCASE

语法：DO CASE

```
    CASE lExpression1
        Commands
    [CASE lExpression2
        Commands
    ...
    CASE lExpressionN
        Commands]
        [OTHERWISE
        Commands]
     ENDCASE
```

说明　*lExpression1 Commands* …当第一个 CASE 的表达式为(.T.)，执行它后面的语句。直到遇到下一个 CASE 语句或者 ENDCASE 语句。如果 CASE 表达式为（.F.），则忽略与下一个 CASE 语句之间的命令集。此命令只执行一组 CASE 为（.T.）的命令。如果所有的表达式均为（.F.）并包含 OTHERWISE 语

句，则执行 OTHERWISE 后的语句。

当需要多路判断时，用 DO CASE...ENDCASE 语句比用 IF 语句更为简捷、清楚。

DO CASE...ENDCASE 实现分支程序的例子如下：

说明　本实例用 DO CASE...ENDCASE 语句实现程序的分支程序，通过对每一个 CASE 语句进行匹配，执行分支处理。

程序 7.6　使用 DO CASE...ENDCASE 实现分支

```
STORE CMONTH(DATE( )) TO month                          &&今天所在的月份
DO CASE                                                 &&  循环开始
   CASE INLIST(month,'January','February','March')      &&开始分支判断
        STORE '第一季度' TO rpt_title
   CASE INLIST(month,'April','May','June')
        STORE '第二季度' TO rpt_title
   CASE INLIST(month,'July','August','September')
        STORE '第三季度' TO rpt_title
   OTHERWISE
        STORE '第四季度' TO rpt_title
ENDCASE                                                 &&  循环结束
WAIT WINDOW rpt_title NOWAIT
```

7.6.3　循环

Visual FoxPro 中的循环有三种：

（1）用SCAN...ENDSCAN

语法：SCAN [NOOPTIMIZE]
　　[Scope] [FOR lExpression1] [WHILE lExpression2]
　　[Commands]
　　[LOOP]
　　[EXIT]
　　ENDSCAN

说明　NOOPTIMIZE:　不使用 SCAN 的 Rushmore 技术。

Scope: 设定记录的范围，只有在范围内的记录才被查询。范围从句可以是 ALL, NEXT *nRecords*, RECORD *nRecordNumber*, 和 REST，缺省的范围值是 ALL。

FOR *lExpression1*：只有当 *lExpression1* 表达式的值为（.T.）时，才执行命令。

WHILE *lExpression2*：只有当表达式 *lExpression2* 的值为(.T.)，才执行它后面的语句。

Commands: 要执行的 Visual FoxPro 命令。

LOOP：直接返回到 SCAN 进行下一个循环，它可以放在 SCAN 和 ENDSCAN 之间的任何位置。

EXIT：从 SCAN ...ENDSCAN 之间跳到 ENDSCAN 后的第一条语句，EXIT 可以放在 SCAN 和 ENDSCAN 之间的任何位置。

ENDSCAN：结束 SCAN 循环

1. SCAN...ENDSCAN实现循环程序

说明　本实例将显示出在北京的所有公司。

程序 7.7　使用 SCAN...ENDSCAN 实现循环

```
CLOSE DATABASES
OPEN DATABASE (HOME( ) + 'Practice\data\业务')
USE 客户                                          && 打开客户表
CLEAR                                             &&清除屏幕
SCAN FOR UPPER(地址) = '北京'
? 联系人, 公司, 地址                              &&显示地址在北京的记录
ENDSCAN
```

（2）用FOR ... ENDFOR

语法：FOR *Var* = *nInitialValue* TO *nFinalValue* [STEP *nIncrement*]

Commands

[EXIT]

[LOOP]

ENDFOR | NEXT

说明　*Var*：用于记数的变量或数组元素，它们不一定要存在。

nInitialValue TO nFinalValue：nInitialValue 是计数器的初始值，nFinalValue 是计数器的终止值。

STEP *nIncrement*：*nIncrement* 是记数的步长增量，如果 *nIncrement* 为负值则步长递减，如果忽略了 STEP 则步长默认为 1。

Commands：要执行的 Visual FoxPro 命令，它可以是多条名令。

EXIT：从 FOR ... ENDFOR 循环之间跳到 ENDFOR 后的第一条语句，EXIT 可以放在 FOR 和 ENDFOR 之间的任何位置。

LOOP：直接返回到 FOR 进行下一个循环，不执行 LOOP 和 ENDFOR 之间的语句，它可以放在 FOR 和 ENDFOR 之间的任何位置。

ENDFORINEXT：进入下一循环

（3）用DO WHILE ... ENDDO

语法：DO WHILE lExpression

Commands

[LOOP]

[EXIT]

ENDDO

说明　*lExpression*：逻辑表达式，当它的值为（.T.）时，DO WHILE 和 ENDDO 之间的语句被执行。

Commands：当 *lExpression* 的值为（.T.）时，要执行的 Visual FoxPro 命令，它可以是多条命令。

LOOP：直接返回到 DO WHILE 进行下一个循环，不执行 LOOP 和 ENDDO 之间的语句，它可以放在 DO WHILE 和 ENDDO 之间的任何位置。

EXIT：从 DO WHILE ... ENDDO 循环之间跳到 ENDDO 后的第一条语句，EXIT 可以放在 DO WHILE 和 ENDDO 之间的任何位置。

ENDDO：进入下一循环或结束循环。

2. DO WHILE...ENDDO实现循环程序

说明　本实例来所有价格在 20 以上的产品的数目总和，直到文件结束。此时跳出循环并显示所求的总量。

程序 7.8　使用 DO WHILE...ENDDO 实现循环

```
CLOSE DATABASES
OPEN DATABASE (HOME( ) + 'Practice\data\业务')
USE 产品                                          && 打开产品表
SET TALK OFF
gnStockTot = 0
DO WHILE .T.                                      && 开始循环
   IF EOF( )
      EXIT
   ENDIF
   IF 价格 < 20                                   && 如果价格少于 20 跳到下一条记录
     SKIP
     LOOP
   ENDIF
   gnStockTot = gnStockTot + 库存量               && 计算价格在 20 以上库存量
   SKIP
ENDDO                                             && 结束循环
CLEAR
? '价格超过 20 的所有产品:'
?? gnStockTot
```

7.7　过程与自定义函数

7.7.1　什么是自定义函数

在 Visual FoxPro 中，自定义函数（UDF）和过程把经常使用的一种功能的代码独立出来。当在程序中需要相同功能的程序时，可以反复调用已经编好的代码。从而有利于程序的维护，对程序的修改只需修改过程与函数，对它们的所有调用便自动完成修改。

自定义函数和过程是有区别的，自定义函数可以具有返回值，而过程没有返回值，但它们的区别不大。

通常过程与函数的调用可以出现在 Visual FoxPro 对象的方法和事件代码中，从而将面向对象的程序设计和结构化的程序设计相结合，提高了程序设计的有效性。对函数和过程的调用次数没有太大的限制。

7.7.2　过程与函数的定义

在 Visual FoxPro 中，过程的定义如下：

```
PROCEDURE myprocedure
  * 在此处可以填写过程的代码
ENDPROC
```

在 Visual FoxPro 中，函数的定义如下：

```
FUNCTION myfunction
  * 在此处添加函数的执行代码
ENDFUNC
```

函数和过程可以保存在一个单独的文件中或者放在一般程序代码的底部，但主程序的代码不能放在过程和函数的后面。当把过程或者函数放在一个单独文件中时，对过程的调用可以用 SET PTOCEDURE TO 命令，如：

```
SET PROCEDURE TO funtionprocedure.prg
```

7.7.3　对过程和函数的调用

在 Visual FoxPro 的程序中调用过程和函数用两种方法：

（1）用DO命令，如：DO myprocedure或直接用myprocedure；

（2）在函数名后加上（），如：myfuntion（）。

7.7.4　向过程和函数传递参数

通过参数可以向函数和过程传递参数，下面的过程可以接受一个单一的参数：

程序 7.9　用过程接受单一参数

```
PROCEDURE myproc( cString )
* 在消息框中显示信息
MESSAGEBOX ("过程：" + cString)
ENDPROC
```

当函数或过程被调用时，向函数或过程传递参数可以用一个变量也可以用一个具体的值。

传递参数

程序 7.10　传递参数

```
DO myprocedure WITH cTestString                  &&使用变量为过程传递参数
DO myprocedure WITH "练习字符串"                 &&使用一个具体的字符串为过程传递参数
myfunc("练习字符串")                             &&使用变量为函数传递参数
```

```
myfunct( cTestString )                    &&使用一个具体的字符串为过程传递参数
```

可以给过程或者函数传递不同的参数，但参数之间要用“，”隔开，例如下面的例子向过程传递了三个不同类型的参数。

传递多个参数给过程

程序 7.11　传递多个参数

```
PROCEDURE myproc( dDate, cString, nTimesToPrint )        &&给过程传递三个参数
FOR nCnt = 1 to nTimesToPrint
   ? DTOC(dDate) + " " + cString + " " + STR(nCnt)
ENDFOR
ENDPROC
```

调用上例的过程可以用如下的代码：

程序 7.12　调用过程

```
DO myproc WITH DATE(), "Hello World", 10
```

7.7.5　函数的返回值

可以通过函数返回任何值，一个函数的默认的返回值是（.T.）

接收函数的返回值的程序如下：

程序 7.13　接收函数返回值

```
FUNCTION plus2weeks
   PARAMETERS dDate
   RETURN dDate + 14
ENDFUNC
```

调用上例的过程可以用如下的代码：

程序 7.14　调用函数

```
dDeadLine = plus2weeks(DATE())
```

7.8　筛选字段和记录

7.8.1　记录作用域

函数具有作用域的规则，同样 Visual FoxPro 中的 Scope 语句，可以指定记录的作用范围从而缩短查询的范围，它们分别是：

ALL：选择表中的所有记录；

NEXT nExpr：从当前记录开始的具有由nExpr指定数目的记录；

RECORD nNumber：只对nNumber指定的记录起作用；
REST：从当前的记录指针开始一直到表的结尾。
使用记录的作用域的代码如下：

程序 7.15　使用记录作用域

```
CLOSE ALL
DISPLAY FIELD 客户代号, 公司 OFF NEXT 10
REPLACE 客户.公司 WITH "北京" RECORD 5
REPLACE 客户.公司 WITH "北京" REST
```

7.8.2 使用FOR子句

使用 FOR 子句同样可以缩小记录的查询范围，从而提高命令的执行速度，只要是符合逻辑条件的任何记录都将是选定范围。

语法：FOR cExpr

举例：

程序 7.16　使用 FOR 子句

```
DELETE FOR 地址 = '北京'                    && 删除所有地址在北京的记录
```

Visual FoxPro 中使用 FOR 子句的命令有：

APPEND FROM ARRAY，COUNT，LOCATE，APPEND FROM，DEFINE PAD，RECALL，AVERAGE，DELETE，REPLACE FROM ARRAY，BLANK，DISPLAY，REPLACE，BROWSE，EXPORT，REPORT，CALCULATE，FOR()，SCAN...ENDSCAN，CHANGE，INDEX，SORT，COPY TO ARRAY，LABEL，SUM，COPY TO，LIST

7.8.3 使用WHILE子句

程序中使用 WHILE 子句，将选择所有逻辑条件为（.T.）的记录。当逻辑条件表达式为（.F.）时，则剩下的记录将不被执行。

语法：WHILE cExpr

说明　当 FOR 和 WHILE 一起使用时，WHILE 的优先权大于 FOR，即程序总是先执行 WHILE 的筛选条件，然后在对满足条件的记录用 FOR 进行筛选。

举例：

程序 7.17　使用 WHILE 子句

```
REPLACE ALL 联系人 WITH '鸿运' FOR 营业额>100000;
    WHILE 地址='北京'
```

Visual FoxPro 中使用 WHILE 子句的命令有：

AVERAGE，COUNT，RECALL，BLANK，DELETE，REPLACE，BROWSE，DISPLAY，REPLACE FROM ARRAY，CALCULATE，EXPORT，REPORT，CHANGE，LABEL，SCAN ... ENDSCAN，COPY TO ARRAY，LIST，SORT，COPY TO，LOCATE，SUM

7.9　使用数组

7.9.1　从表向数组传递数据

Visual FoxPro 中的数组的大小没有太大的限制，只取决于磁盘与内存空间的大小，数组在内存中的运行速度很快。

下面的命令将表中的数据从表传递到数组：

（1）SCATTER

语法：SCATTER

[FIELDS *FieldNameList*

| FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*] [MEMO]

TO *ArrayName* | TO *ArrayName* BLANK | MEMVAR | MEMVAR BLANK

| NAME *ObjectName*

说明　FIELDS *FieldNameList*：确定传给数组或变量的字段，忽略此子句将选中所有的字段。字段的列表中可以具有“备注”型字段，但需要在列表后面加上 MEMO 关键字。

FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*：用 LIKE 和 EXCEPT 可以选择将要传递给数组或变量的字段。如果包含了 LIKE 则所有与 LIKE 后 *Skeleton* 表达式相匹配的字段将传给数组，如果包含了 EXCEPT 则所有与 EXCEPT 后 *Skeleton* 表达式相匹配的字段将被排除。

Skeleton 表达式支持通配符例如将所有以 A 和 P 开头的字段传递给数组可以用如下的命令：

SCATTER FIELDS LIKE A*,P* TO myarray

LIKE 和 EXCEPT 子句可以共同使用，如：

SCATTER FIELDS LIKE A*,P* EXCEPT PARTNO*　TO myarray

MEMO：包括字段中的“备注”字段，默认值是不包括，如果你将“备注”字段传给数组或者变量，则必须要有足够的内存，如果“备注”字段太长，内存中无法容纳，将有内存不足的错误提示，“备注”将不被传入。

TO *ArrayName*：将被来存储记录的数组名，SCATTER 将按照字段列表的顺序将字段的值按顺序拷贝到数组中，如果数组元素的个数大于字段的个数，剩下的元素不变。如果数组元素的个数小于字段的个数，或者数组没有定义，Visual FoxPro 将自动的新建一个数组，数组元素和对应的表字段具有相同的大小和数据类型。

TO *ArrayName* BLANK：创建一个具有空元素的数组，它的大小和数据类型和表中的字段相对应。

MEMVAR：把数据传给变量，而不是数组，每一个变量和一个字段相对应，变量将和字段具有相同的名称、大小、数据类型。当在内存中有相同的字段名和变量名时，字段的优先级大于变量，要引用变量可以加上“M.”前缀，在 MEMVAR 前请不要加“TO”否则 Visual FoxPro 将把它作为数组处理。

MEMVAR BLANK：建立一个空的变量组，每一个变量的名称、大小和数据类型和字段相对应。

NAME *ObjectName*：创建一个对象，它的属性的名字和表中的字段中的名称一致，从而每一个对象的属性正好和表中的字段相应。对对象的引用有点象对表的引用，必要时在前面加（M.）、表示引用的是对象，例如：

程序 7.18　对象的引用

```
USE 客户
SCATTER NAME 客户
?客户.公司                                    && 引用的是表
?M.客户.公司                                  && 引用的是对象的属性
```

下面的例子说明SCATTER的使用

说明　本程序将表中的数据传递给变量。

程序 7.19　传递表中的数据

```
CREATE TABLE Test FREE ;
(Object C(10), Color C(16), SqFt n(6,2))          &&创建一个自由表
SCATTER MEMVAR BLANK                              &&在内存中建立一个表的变量结构
m.Object="Box"                                    &&给内存中的变量赋值
m.Color="Red"
m.SqFt=12.5
APPEND BLANK                                      &&给表添加新空白字段
GATHER MEMVAR                                     &&把内存中的值传递给新添加的字段
BROWSE
```

（2）COPY TO ARRAY

语法：COPY TO ARRAY *ArrayName*
[FIELDS *FieldList*]
[*Scope*] [FOR *lExpression1*] [WHILE *lExpression2*]
[NOOPTIMIZE]

说明　*ArrayName*：确定拷贝出的数组的名称。

FIELDS *FieldList*：只有在字段列表 *FieldLis* 中的字段才被拷贝到数组中去。如果此项缺省则将所有的字段拷贝。

Scope：确定所要拷贝的记录的范围。可以使用 ALL, NEXT *nRecords*, RECORD *nRecordNumber*, 和 REST 子句。

FOR *lExpression1*：只有满足 *lExpression1* 条件表达式的记录才被拷贝到数组中。如果 *lExpression1* 用了优化的表达式，则 COPY TO ARRAY 将使用 Rushmore 技术对查询提供优化。

WHILE *lExpression2*：只有当 *lExpression2* 条件表达式的值为（.T.）时才执行拷贝。

下面的例子说明COPY TO ARRAY的使用

说明 本实例打开客户表，创建一个二维的数组并用 COPY TO ARRAY 将表中的前三个记录值拷贝到数组。

程序 7.20 使用 COPY TO ARRAY

```
CLOSE DATABASES
OPEN DATABASE (HOME( ) + 'Practice\data\客户')
USE 客户                                          && 打开客户表
DIMENSION gaTemp(3,10)                            &&创建二维数组
COPY NEXT 3 TO ARRAY gaTemp                       &&将三个记录拷贝到数组
DISPLAY MEMORY LIKE gaTemp                        &&显示内存中的数组中的值
```

（3）SELECT-SQL

有关使用 SELECT-SQL 的内容请见后面 SQL 语言的章节。

注意 SCATTER 和 COPY TO ARRAY 有以下几个不同点：

SCATTER 传递的是当前表的当前记录的数据，COPY TO ARRAY 可以传递当前表中的多条记录的数据。

SCATTER BLANK 可以自动的产生和表的字段结构相同的数组，数组中的元素和表中的字段具有相同的大小和数据类型，数组中的元素为空值。

用 SCATTER VAR 选项可以自动的建立和表具有相同的大小、名称和数据类型的内存变量。

7.9.2 从数组向表中传递数据

Visual FoxPro 中从数组向表中传递数据的命令有：

（1）GATHER

语法：GATHER FROM *ArrayName* | MEMVAR | NAME *ObjectName*
[FIELDS *FieldList* | FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*]
[MEMO]

说明 FROM *ArrayName*：确定替代表中的当前记录的数组名称。数组中的元素按照顺序替换在记录中相对应的字段的内容，如果数组的元素比表中的字段的个数少，忽略剩下的字段。如果数组的元素比表中的元素多，忽略多余的数组元素。

MEMVAR：规定是从变量或者数组中拷贝值到当前的记录。具有和字段相同名字变量的值将拷贝到对应的字段，如果名称不同将不与拷贝。

NAME *ObjectName*：确定具有和表中的字段名相同的属性的对象。当前的记录的每一个字段的值被具有相同名称的对象的属性值代替。如果字段名和属性名称不同，不与替代。

FIELDS *FieldList*：确定被变量或者数组元素替代的字段，只有在 *FieldList* 中规定的字段才被替代，否则不与替代。

FIELDS LIKE *Skeleton* | FIELDS EXCEPT *Skeleton*：通过 LIKE 或者 EXCEPT 子句可以有选择的替代字段的值。如果加入 LIKE *Skeleton*，则和 *Skeleton* 相匹配的字段将被替代。如果包含 FIELDS EXCEPT *Skeleton* 子句则拷贝所有与 *Skeleton* 不匹配的字段值。*Skeleton* 支持通配符“*和？”例如可以使用如下的名令将所有以 A 和 P 开头的字段替换：

GATHER FROM gamyarray FIELDS LIKE A*,P*

MEMO：将字段中的“备注”字段用数组或者变量中的元素替代。如果略去 MEMO 则“备注”字段将不被拷贝，即使包含 MEMO，通用型字段在 GATHER 命令中也将不被替换。

下面的例子说明GATHER命令的使用

说明　本实例用向表中拷贝一条新记录。

程序 7.21　使用 GATHER 命令

```
CREATE TABLE 练习 FREE;                                    &&创建临时表
      (字段 1 C(10),字段 2 C(16), 字段 3 n(6,2))
SCATTER MEMVAR BLANK                                       &&在内存中建立一个相同的变量结构
m.字段 1="北京"
m.字段 2="海淀区"
m.字段 3=25.6
APPEND BLANK                                               &&在自由表中添加一个空记录
GATHER MEMVAR                                              &&向空记录中拷贝内存中的字段
BROWSE
```

（2）APPEND FROM ARRAY

语法：APPEND FROM ARRAY *ArrayName*
　　[FOR lExpression]
　　[FIELDS *FieldList*]

说明　*ArrayName*：确定向表中记录拷贝数据的数组名称。

FOR *lExpression1*：只有满足 *lExpression1* 条件表达式的数组中的数据才被拷贝到表中。*lExpression1* 中必须包含条件表达式中的相关字段。在数组元素加入表中以前，先看是否满足条件表达式，只有满足表达式的数组元素才被加入到表中。

FIELDS *FieldList*：只有在 *FieldList* 中的字段才被数组中的数据更新。字段列表的顺序和数组元素的顺序是一致的。

下面的例子说明APPEND FROM ARRAY命令的使用

说明　本实例用 APPEND FROM ARRAY 命令向表中添加一条新记录。

程序 7.22　使用 APPEND FROM ARRAY 命令

```
CREATE TABLE 练习 FREE;                                    &&创建自由表
      (字段 1 C(10),字段 2 C(16), 字段 3 n(6,2))
SCATTER TO 新记录 BLANK                                    &&  按表的结构产生一个新的数组
新记录[1]="北京"                                           &&为数组赋值
新记录[2]="海淀区"
新记录[3]=25.6
```

```
APPEND FROM ARRAY 新记录                    && 从数组中添加元素到表中
```

（3）INSERT – SQL

关于 INSERT –SQL 的内容请参阅 SQL 语言部分

GATHER 和 APPEND FROM ARRAY 与 INSERT - SQL 有以下几个不同点：

GATHER 将数组中的数据传递给当前表中的当前记录。GATHER MEMVAR 将内存变量中的数据传递给当前记录。

APPEND FROM ARRAY 先在当前表的末尾加一新的记录，然后将数组中的数据传递给这个新的记录中。INSERT – SQL 在表中添加一条新的记录，然后将数组中的数据传递给这个记录。与 GATHER 和 APPEND FROM ARRAY 不同的是它可以向任何打开的表中添加数据而不需要一定在当前工作区中打开的表。

7.10　使用NULL值

7.10.1　NULL值的特点

在 Visual FoxPro 中支持 NULL 值，从而能更为方便地与 Microsoft Access 和其它的 SQL 数据库一同工作，NULL 值具有如下的特点：

等价于不具有任何值；

与0、空字符串或者空格不同；

排序时具有最大的优先权；

可以用于计算和大多数的函数中。

NULL 值将影响到命令、函数和表达式的执行。Visual FoxPro 中支持 ANSI 标准规定，可以应用于任何具有值或者表达式的地方。

7.10.2　用NULL作为值

在 Visual FoxPro 中，用".NULL."或者在交互的方式下键入 CTRL+0 都可以赋给 NULL 值。NULL 值不同于空值、空字符串或者 0。

对于空字符串，用 EMPTY（）和 ISBLANK（）检验都将返回（.T.），但是如果采用 ISNULL（）函数返回（.F.）。

在数组和 STORE、GATHER 和 SCATTER 函数中可以使用 NULL 值，例如：

```
DIMENSION X[4]
STORE .NULL. TO X
```

NULL 值不是一种数据类型，当把 NULL 赋给字段或者变量时，该字段或者变量的数据类型不发生变化，只是值变为.NULL.，如：

```
STORE 50 TO nX
nX = .NULL.
? TYPE ("nX")          && 数据类型仍然是数值型
```

在函数和命令中，不同的数据类型对 NULL 有不同的解释：

Logical（逻辑型）：如果逻辑型表达式的结果为.NULL.，则它返回.NULL.值或出错信

息。而函数 EMPTY（）、ISBLANK（）和 ISNULL（）除外。

Numeric（数值型）：数值型表达式结果为.NULL.则返回.NULL.值，如果传递一个为.NULL.的值给数值型函数则返回的结果也是.NULL.。

Date（日期型）：含 NULL 值的日期型表达式的返回值是.NULL.。

7.10.3 表达式中使用.NULL.

表达式中的.NULL.值在多数情况下是不变的，在表 7.14 中是.NULL.在逻辑表达式中的值。

表7.14 .NULL.对逻辑表达式值的影响

逻辑表达式	X=TRUE 时	X=FALSE 时	X=.NULL.时
X AND .NULL.	.NULL.	FALSE	.NULL.
X OR .NULL.	TRUE	.NULL.	.NULL.
NOT .NULL.	FALSE	TRUE	.NULL.

在条件表达式中如果是.NULL.值，则该条件为假。

7.10.4 用.NULL.作为参数

在 Visual FoxPro 中，如果使用.NULL.作为参数，许多命令和函数会受到影响。下面是有关.NULL.作为参数传递时的一些规则：

（1）当传递一个.NULL.值给一个命令，会产生错误；

（2）函数的参数如果认为.NULL.值是有效值，则函数的结果也是.NULL.值；

（3）如果向数值型参数赋给.NULL.值，将会产生错误；

（4）如果ISBLANK(), ISDIGIT(), ISLOWER(), ISUPPER(),ISALPHA( )和 EMPTY（）传递.NULL.值返回值是（.F.），向ISNULL（）传递.NULL.值，返回值为（.T.）；

（5）通过IS NULL和IS NOT NULL子句，INSERT-SQL和SELECT-SQL语句处理NULL值；

（6）SQL合计函数将忽略NULL值，如果所有的合计值均是.NULL.值，则结果是.NULL.值；

7.11 小 结

Visual FoxPro 提供了许多高级语言的特性。和其它高级语言类似，Visual FoxPro 提供了多种数据类型，可以将这些数据类型的数据存储于表、数组、变量和其它的数据容器中。可以使用各种操作符，也可以使用 Visual FoxPro 提供的丰富的函数和各种命令。

本章对 Visual FoxPro 语言的特性作了较为详细的说明，主要介绍了如下的内容：语法和命名规则、数据和字段类型、数据的存储类型、操作符和表达式、自定义函数和过程、程序的流程控制。

思考与练习

1. Visual FoxPro 中数据变量的作用范围有哪几类？数据类型有哪几种？

2. Visual FoxPro 中有哪几类对象？它们的命名为什么要遵守一定的命名规则？这样有什么好处？

3. “表”字段的命名遵守什么样的约定？如何引用它们？

4. “变量”命名有哪些约定？

5. Visual FoxPro 中的数据类型和字段类型各有哪些？这些类型各用在什么样的场合？

6. 变量的存储类型有哪几类？定义方法是什么？

7. Visual FoxPro 中字符操作符有哪三种？它们引发的操作是什么？

8. Visual FoxPro 中日期和时间操作符有哪几种？它们引发的操作是什么？

9. Visual FoxPro 中逻辑操作符有哪几种？它们引发的操作是什么？

10. Visual FoxPro 中关系操作符有哪几种？它们引发的操作是什么？

11. Visual FoxPro 中数值操作符有哪几种？它们引发的操作是什么？

12. Visual FoxPro 中日期和时间操作符有哪三种？它们引发的操作是什么？

13. Visual FoxPro 中表达式有哪几类？它们都应用于什么条件？

14. 在 Visual FoxPro 中实现分支和循环各有哪些语句？

15. “FOR“子句和”WHILE“子句起到了什么作用？

16. Visual FoxPro 中为什么要使用数组？如何使用数组向表中传递数据？

17. 为什么要使用“NULL“值，它有什么特点？

第8章　VFP 6.0 面向对象的程序设计

Visual FoxPro 6.0 不但支持传统的面向过程的编程方法，而且是一种功能强大的面向对象的编程语言。

面向对象的编程方法与原来主要对流程和程序模块的设计方法有所不同，这种编程方法主要是考虑对象的构造以及与对象有关的属性和方法的设计。对象为了实现自己的功能要具有自己的函数，另外也需要有暴露在外部的接口，用来完成其他程序的调用。

在本章主要介绍：

（1）VFP 6.0 中的类和对象；

（2）用类设计器设计类；

（3）用编程的方法设计类；

（4）对类和对象的引用；

（5）对父类方法的引用。

8.1　Visual FoxPro中的类和对象

8.1.1　Visual FoxPro中的对象

在 Visual FoxPro 中，对象具有属性、事件和方法，通过它们可以实现一个对象的功能和处理它们的外部接口。

通过面向对象的程序设计方法，在应用程序中有效地处理对象，并建立自己的类库。可以使你的程序具有以下的优点：

（1）代码更为精练；

（2）对象可以很容易的组装成为应用程序，而不必把太多的精力用于关心每一个对象的细节；

（3）代码的维护和代码的复用更为方便，大型程序的构造更为简单。

面向对象的应用程序实际上是在“组装”程序，每个对象中的代码对于许多编程人员来说是透明的，编程人员更为关心的是功能和接口，也就是对象所具有的属性和方法。

Visual FoxPro 中，有两个密切相关的概念：类和对象。类和对象是有区别的，类作为一个对象的整体定义而存在，它是对象的轮廓和蓝图。这就如同一台机床和图纸的关系，图纸决定了某一种类型的机床的功能和外观，而一台具体的机床可以看作一个对象而存在。某种类型的机床，根据它的初始设计可以派生出一个系列的产品，这就如同由一个父类派生出若干子类一样，虽然父类和子类之间有相同点但也存在不同的地方。属于某个系列的某一台具体的机床可以看作是一个具体的对象。

8.1.2　对象的属性、事件和方法

在 Visual FoxPro 中，“属性”封装了数据，每个对象都具有“属性”。“属性”值可

以在设计阶段设置，也可以在运行阶段更改，但有些“属性”是只读属性，不可改变。

对象可以具有“事件”，“事件”是预先定义的动作，由用户或者系统激活。Visual FoxPro 中的事件通常包括键盘“事件”和鼠标“事件”，如单击或双击鼠标，移动鼠标，按键等。

“方法”是指对象为实现一定功能而编写的代码，“方法”和对象相连。事件中的代码可以认为是基于事件的响应方法。但“方法”可以独立于事件的单独代码，必要时可以由其它方法或事件代码进行显式的调用。

事件通常已经预先由系统定义好了，不能随便的扩充，而方法和属性却可以无限的扩展。事件的发生具有一定的顺序，这在以后章节中会逐渐介绍。

8.1.3 Visual FoxPro中的类

所有的属性、事件和方法都由“类”进行定义。Visual FoxPro 中，“类”通过封装和继承使得代码更容易维护和复用。

通过对事物的抽象，由类定义的对象可以隐藏自己内部的复杂性。通过在对象中封装数据和方法，可以实现对象的功能，通过封装对象的属性可以实现数据的封装。

子类可以继承父类的所有的属性和方法，也可以根据需要加入新的属性和方法。如图 8.1 所示，“电话”种类可以有很多中，但都具有基类电话的基本属性。

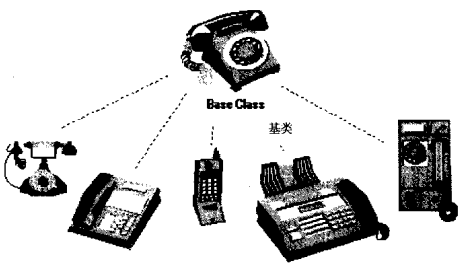


图8.1　类和子类的继承关系

8.1.4 Visual FoxPro中类的层次

Visual FoxPro 中的类有一定的层次，这样当建立你自己的类时，可以从 Visual FoxPro 中已经定义的类派生。Visual FoxPro 中对象类的标准类层次图见图 8.2 所示。

从图中可以看到 Visual FoxPro 中对象可以分为容器对象和控件对象，它们代表了具有不同特点的 Visual FoxPro 对象。

图8.2　Visual FoxPro的类层次

8.1.5　容器类和非容器类

Visual FoxPro 中的类分为两种类型，它们分别是容器类和非容器类或控件。

容器类可以包含其他的对象，并且允许对所包含的对象的访问，例如创建具有两个列表和两个命令按钮的容器类，然后把控件加入表单中。那么在设计和运行时，都可以对容器里的列表和按钮进行访问。

控件的封装比容器严密。控件在设计和运行时，可以作为一个整体进行处理，但组成控件的组件不能单独的修改。

因此，容器类可以当作其他对象的父类。控件可以包含在容器中，但不能作为其他对象的父类，及不能包含其它的对象。

容器类和非容器类的区别见图 8.3 所示。

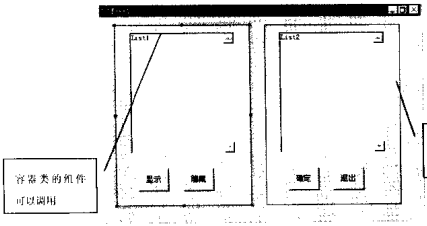


图8.3　容器和控件

表 8.1 列出了每种容器类可以包含的对象。

表8.1　容器类所能包含的对象

容器类型	包含的对象
命令按钮组	命令按钮
容器	任何控件
控制	任意控件
自定义控件	任何控件、页框控件、其它自定义控件
表单集	表单、工具栏
表单	页框、任何控件、容器、自定义控件
表格列	标头和除表单集、表单、工具栏、时钟和其他列控件以外的对象
表格	表格列
选项按钮组	选项按钮
页框	页面
页面	任何控件、容器和自定义控件
工具栏	任何控件、页框和容器

8.2　用类设计器设计类

8.2.1　什么时候使用类

如果想使用类来简化应用程序的结构，必须弄清楚什么时候使用类。当然，你可以为 Visual FoxPro 中的每一个控件都建立一个类，但这样做毫无意义。

类是用于对公共任务的封装。例如在表单中，每次都需要处理诸如记录的前移、后移、保存、关闭等通用的命令。此时就可以定义一个通用的按钮类来处理这些每个表单都会遇到的功能，在需要的时候从这个按钮类创建一个对象加入表单。

每个人的应用程序都应是一件精心设计的杰作。对于程序的界面，首先要符合大众的操作习惯和公共的编程习惯，但必须要给予它们自己的个性。这就要求应用程序界面具有

统一的外观和风格。为了达到这个目的，可以给控件建。立各自的模板及设计出具有特点的控件，在程序中使用这些类可以方便的实现。

8.2.2 从基类派生出新类

对于 Visual FoxPro 中的容器和控件，可以从它们派生出子类。你可以为这些子类设置属性，加入方法代码，也可以根据需要添加新的属性和方法。

表 8.2 列出了可以作为基类的 Visual FoxPro 的对象：

表8.2　Visual FoxPro基类对象

复选框 (CheckBox)	编辑框 (EditBox)	列表框 (ListBox)	形状 (Shape)
列[1] (Column)	表单 (Form)	OLE 绑定控件 (OLEBoundControl)	微调 (Spinner)
命令按钮 (CommandButton)	表单集 (FormSet)	OLE 容器控件 (OLEContainerControl)	文本框 (TextBox)
命令组控件 (CommandGroup)	表格 (Grid)	选项按钮[1] (OptionGroup)	时间 (Timer)
复选框 (ComboBox)	标头[1] (Header)	选项组 (OptionGroup)	工具栏 (ToolBar)
容器 (Container)	图像 (Image)	页面[1] (Page)	控制 (Control)
标签 (Label)	页框 (PageFrame)	自定义 (Custom)	直线 (Line)
分隔符[1] (Separator)			

注意　表 8.2 中具有上角标 1 的类是父容器的一部分，在类设计器中不能基于它们创建子类。

对于所有这些可以作为基类的对象，都具有下面的最小事件集：

Init（）：对象创建时激活。

Destroy（）：对象从内存中释放时激活。

Error（）：在事件和方法的过程中发生错误时发生。

对于所有这些可以作为基类的对象，都具有下面的最小属性集：

Class：该类的类型。

BaseClass：该类的基类。

ClassLibrary：该类所在的类库。

ParentClass：创建的对象所基于的类，如果直接从 Visual FoxPro 派生，则它和 BaseClass 的属性值相同。

对于控件，通过派生子类可以建立新的具有封装功能的控件。例如，你想建立一个按

钮，当单击它时可以自动的释放表单。此时可以在 Visual FoxPro 的命令按钮的基础上建立它的一个子类，在 Click（）事件中加入下面的命令：

RELEASE THISFORM

可以将这个按钮加到任何一个表单中，如图 8.4 所示。

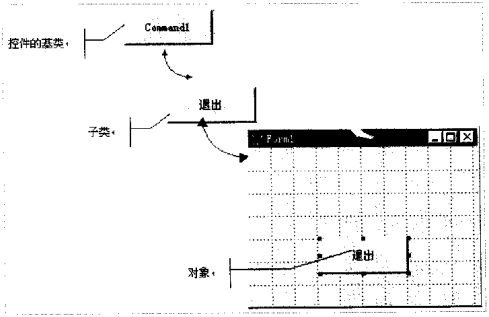


图8.4　类、子类和对象的建立

8.2.3　用类设计器创建类

创建类有两种方法：通过类设计器和通过代码。

当使用类设计器时，需要对新类的基类进行说明，指明类库的名称、对于新建的类的说明等。在类设计器中创建的类，可以在“类设计器”中见到每个对象的最终的外观。

用类设计器创建新类的操作步骤如下：

开始 从“项目管理器”选择“类”选项卡。

下一步 按“新建”按钮。

下一步 在“新类”对话框中的“新类”框输入类名。

下一步 在“新类”对话框的“派生于”框输入或按…选择基类名。

下一步 在“新类”对话框中的“存储于”框输入或按…选择类库名。

说明 所有的类都存储于“类库”当中，Visual FoxPro 中的类库文件具有.VCX 后缀。

类库的创建可以有三种：

(1) 在创建类时，在“存储于”框中直接输入一个新库的文件名。

(2) 在用程序创建类时用 CREATE CLASS 命令。

(3) 用 CREATE CLASSLIB 命令。

下一步 在“新类”对话框中按“确定”，如图8.5所示。

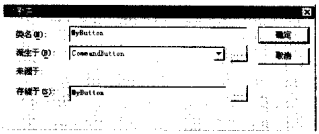


图8.5　创建新类

下一步 这时会出现如图8.6所示的类设计器。

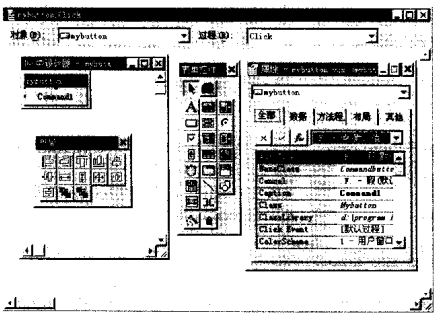


图8.6　类设计器

说明 “类设计器”和表单设计器类似，你可以在“属性窗口”中观看编辑类的属性，用“代码窗口”编辑事件和方法的代码，用“表单控件”工具栏可以向“容器”中添加控件。

完成 关闭“类设计器”。

8.2.4　用类设计器创建自定义类

自定义类在运行时是不可见的，但当定义时可以在“类设计器”中给自定义类建立自己的属性和方法。

用类设计器创建自定义类的操作步骤如下：

说明　在这里所定义的自定义控件，用来对“按钮组件”进行处理，有关“按钮组件”的建立及应用在后面将做详细介绍，在本例中，你主要是学习如何建立自定义控件、如何给自定义控件添加自己的属性和方法，关于对自定义控件的引用在以后会做详细说明。

开始　从“项目管理器”选择“类”选项卡。

下一步　按“新建”按钮。

下一步　在“新类”对话框中的“新类”框输入类名“CustomChecker”。

下一步　在“新类”对话框的“派生于”框中选择“Custom”。

下一步　在“新类”对话框中的“存储于”框输入“MyCustom”。

下一步　在“新类”对话框中按“确定”。

下一步　在“类”菜单中选择“新方法程序”。

下一步　在“新方法程序”对话框中的“名称”框输入“CheckConflicks”，按“添加”按钮。

下一步　在“新方法程序”对话框中的“名称”框输入“HanderRecord”，按“添加”按钮。

下一步　在“新方法程序”对话框中的“名称”框输入“String”，按“添加”按钮。

下一步　在“新方法程序”对话框中的“名称”框输入“VerifyChanges”，按“添加”按钮。

下一步　在“新方法程序”对话框中的“名称”框输入“VerifyEachChange”，按“添加”按钮。

下一步　在“属性”窗口中，选择“方法程序”，双击“CheckConflicks”方法。

下一步　在“代码”窗口中，输入“CheckConflicks”如下的代码。

程序 8.1　“CheckConflicks”方法

```
*-------------------------------------------------------------------------------------------*
*  首先检查是否有其它程序使用所要处理的表中的数据，如果有则调用 HandleRecord 方法
*  由用户决定如何进行下一步操作。
*
*  返回的数值型的数值：
*    0 — 当前的值没有发生变化
*    1 — 成功执行用户的更改
*    2 — 不能成功更改用户表中的数据
*-------------------------------------------------------------------------------------------*
#define NOBUFFI_LOC '没有充足的数据缓冲区.'
LOCAL lnSuccess, llnRec
m.lnSuccess = 0

DO CASE
   CASE INLIST(CURSORGETPROP('Buffering'), 2,3)                    && 获得行缓冲区
```

```
    IF '2' $ GETFLDSTATE(-1)                                    && 数据发生变化
        m.lnSuccess = THIS.HandleRecord(0)
    ENDIF
CASE INLIST(CURSORGETPROP('Buffering'), 4,5)                    && 获得表缓冲区
    m.llnRec = GETNEXTMODIFIED(0)                               && 获得被修改的记录
    DO WHILE m.llnRec > 0
        GO m.llnRec                                             && 记录指针移动到被修改的记录
        m.lnSuccess = IIF(m.lnSuccess != 2, THIS.HandleRecord(0), 2)
                                                                &&调用 HandleRecord 处理修改记录
        m.llnRec = GETNEXTMODIFIED(m.llnRec)
    ENDDO
OTHERWISE                                                       && 没有缓冲区
    WAIT WINDOW NOBUFF_LOC NOWAIT
ENDCASE
RETURN m.lnSuccess                                              && 返回值
```

下一步 在“代码”窗口中，在“过程”框选择“HandlerRecord”，添加如下的代码。

程序 8.2　“HandlerRecord” 代码

```
*-------------------------------------------------------------------------------*
*  本方法由 CheckConflicts 方法和 VerifyEachChange 进行调用，通过一个循环
*  将每个当前记录的值和表中的值进行比较。
*  如果将 1 作为参数传递给本方法，方法将在用户对记录做修改以前将当前值和
*  字段中的值进行比较
*
*  返回的数值型的数值：
*   0 — 当前的值没有发生变化
*   1 — 成功执行用户的更改
*   2 — 不能成功更改用户表中的数据
*-------------------------------------------------------------------------------*
LPARAMETERS lnScope
*-- lnScope 的正确使用的参数有:
*   0 - 只处理冲突                                              && 默认值
*   1 - 也处理值的改变

* 验证参数
IF TYPE("m.lnScope") != "N"
    m.lnScope = 0
ENDIF
```

```
IF !BETWEEN(m.lnScope, 0, 1)
    #define WINDMSG_LOC "错误的传递给 handlerecord 方法参数"
    WAIT WINDOW WINDMSG_LOC
ENDIF

* 声明常量 &变量
#define CR_LOC CHR(13)
#define SAVE_LOC "你想用当前值进行修改么?" + CR_LOC + "(选择 '取消' 恢复原值.)"
#define CONFLICT_LOC "数据冲突"
#define VERIFY_LOC "验证更改"
#define ORG_LOC "原值: "
#define CUR_LOC "当前值: "
#define CHG_LOC "你的更改: "
#define MEMO_LOC " 是一个内存字段."
#define FIELD_LOC "字段: "
#define RECORD_LOC "Record Number: "
#define VALCHG1_LOC "另一个用户改变了一个值."
#define VALCHG2_LOC "值修改完毕."

LOCAL lnChoice, lnField, lcField, luOldVal, luCurVal, luField, llMadeChange, llSuccess
m.llMadeChange = .F.
m.llSuccess = .T.

*在检查冲突之前刷新当前视图
IF CURSORGETPROP('SourceType') != 3                    && 不是一个本地表
  =REFRESH()
ENDIF
* 对每条记录中的字段检查是否发生冲突和值的变化
FOR m.lnField = 1 to FCOUNT()
    m.lnChoice = 0
    m.lcField = FIELD(m.lnField)
    IF TYPE(m.lcField) = "G"                           &&如果是通用型字段
        LOOP                                           && 不能检查通用型字段
    ENDIF
    m.luOldVal = OLDVAL(m.lcField)
    m.luCurVal = CURVAL(m.lcField)
    DO CASE
        * -----< 只检查冲突 >--------
        CASE m.lnScope = 0
```

```
            IF m.luOldVal != m.luCurVal
                m.llMadeChange = .T.
                m.lnChoice = MESSAGEBOX(VALCHG1_LOC + CR_LOC + FIELD_LOC;
                        + lcField + CR_LOC + RECORD_LOC + ALLTRIM(STR(RECNO())) + ;
                        IIF(TYPE("m.lcField") != "M", CR_LOC + CR_LOC + ORG_LOC;
                        + THIS.String(m.luOldVal) + CR_LOC + CUR_LOC;
                        + THIS.String(m.luCurVal) +CR_LOC+CHG_LOC;
                        +THIS.String(EVAL(m.lcField)), CR_LOC + CR_LOC;
                        + m.lcField + MEMO_LOC) + CR_LOC + CR_LOC;
                        + SAVE_LOC, + 3+48+0, CONFLICT_LOC)
            ENDIF
        * -----< 检查冲突和更改 >--------
        CASE m.lnScope = 1                                    && 检查所有的更改
            m.luField = EVAL(m.lcField)
            IF m.luOldVal != m.luField OR m.luCurVal != m.luField
                m.llMadeChange = .T.
                m.lnChoice = MESSAGEBOX(VALCHG2_LOC + CR_LOC + FIELD_LOC;
                    + m.lcField + CR_LOC + RECORD_LOC + ALLTRIM(STR(RECNO()));
                    +IIF(TYPE("m.lcField") != "M", CR_LOC + CR_LOC + ORG_LOC;
                    + THIS.String(m.luOldVal) +CR_LOC + CUR_LOC ;
                    + THIS.String(m.luCurVal) +CR_LOC + CHG_LOC;
                    +THIS.String(EVAL(m.lcField)), CR_LOC+CR_LOC;
                    + m.lcField + MEMO_LOC) +CR_LOC + CR_LOC;
                    + SAVE_LOC, + 3+48+0, VERIFY_LOC)
            ENDIF
    ENDCASE
    DO CASE
        CASE m.lnChoice = 7                                   && 不存储更改
            REPLACE (m.lcField) WITH m.luCurVal
        CASE m.lnChoice = 2                                   && 取消，恢复原值
            REPLACE (m.lcField) WITH m.luOldVal
     ENDCASE
ENDFOR
IF m.llMadeChange
    m.llSuccess = TABLEUPDATE(.F., .T.)
    RETURN IIF(m.llSuccess, 1, 2)
ELSE
    RETURN 0
ENDIF
```

下一步 在“代码”窗口中，在“过程”框选择“String”如下的代码。

程序 8.3　“String”过程代码

```
*-------------------------------------------------------------------------------*
*本方法由 HandleRecord 方法调用它的返回值与传入的参数相等价。
*如果传入的是备注字段，将只是显示一项提示而不是把所有的备注文本显示在对话框中
*-------------------------------------------------------------------------------*

LPARAMETERS luValue

m.uType = TYPE('m.luValue')
DO CASE
    CASE m.uType = 'C'                                    &&字符型
        RETURN ALLTRIM(m.luValue)
    CASE INLIST(m.uType, 'N', 'Y')                        &&数值型
        RETURN ALLTRIM(STR(m.luValue))
        CASE m.uType = 'D'                                &&日期型
        RETURN DTOC(m.luValue)
    CASE m.uType = 'T'                                    &&日期时间型
        RETURN TTOC('m.luValue')
    CASE m.uType = 'L'                                    &&逻辑型
        RETURN IIF(m.luValue, '.T.', '.F.')
    CASE uType = 'M'                                      &&备注型
        RETURN '备注字段'
ENDCASE
```

下一步 在“代码”窗口中，在“过程”框选择“Verifychanges”如下的代码。

程序 8.4　“Verifychanges”过程代码

```
*-------------------------------------------------------------------------------*
*如果表或者记录发生了更改，应该给用户以提示，从而保存更改。
*如果用户选则了“是”，保存所有的更改。
*
*  返回的数值型的数值：
*    0 — 当前的值没有发生改动
*    1 — 成功执行所有用户的更改
*    2 — 不能成功更改用户表中的数据
*-------------------------------------------------------------------------------*
```

```
* 声明常量 &变量
#define SAVECHG_LOC '想保存你的更改么?'
#define SAVECHG2_LOC '保存更改'
#define NOBUFF_LOC2 '没有数据缓冲区.'

LOCAL lnChoice, llMadeChange, lnSuccess
m.llMadeChange = .F.
m.lnSuccess = 0

* 如果用户做了任何改动，提示是否保存
DO CASE
    CASE INLIST(CURSORGETPROP('Buffering'), 2,3)                    && 行缓冲区
        IF '2' $ GETFLDSTATE(-1)
            m.llMadeChange = .T.
        ENDIF
    CASE INLIST(CURSORGETPROP('Buffering'), 4,5)                    && 表缓冲区
        IF GETNEXTMODIFIED(0) > 0
            m.llMadeChange = .T.
        ENDIF
    OTHERWISE
        WAIT WINDOW NOBUFF_LOC NOWAIT
ENDCASE
IF m.llMadeChange
    m.lnChoice = MESSAGEBOX(SAVECHG_LOC, 4+32, SAVECHG2_LOC)
    IF m.lnChoice = 6                                               && 是
        m.lnSuccess = IIF(TABLEUPDATE(.T.,.T.), 1, 2)
    ELSE
        =TABLEREVERT(.T.)
    ENDIF
ENDIF
RETURN m.lnSuccess
```

下一步　在“代码”窗口中，在“过程”框选择“VerifyEachchange”如下的代码。

程序 8.5　“VerifyEachchange”过程代码

```
*-------------------------------------------------------------------------------------------*
*对于表或者记录中的任何改动，显示原来的值和新值，然后提示是否进行更改。
*冲突处理包括在 HandleRecord 方法中
```

```
*  返回的数值型的数值：
*    0 — 当前的值没有发生改动
*    1 — 成功执行所有用户的更改
*    2 — 不能成功更改用户表中的数据
*-----------------------------------------------------------------------------*

#define NOBUFF_LOC3 '没有数据缓冲区.'
LOCAL lnSuccess, lnRec
m.lnSuccess = 0

DO CASE
    CASE INLIST(CURSORGETPROP('Buffering'), 2,3)                    && 行缓冲
        IF '2' $ GETFLDSTATE(-1)                                    &&数据发生改变
            m.lnSuccess = THIS.HandleRecord(1)
        ENDIF
    CASE INLIST(CURSORGETPROP('Buffering'), 4,5)                    && 表缓冲
        m.lnRec = GETNEXTMODIFIED(0)
        DO WHILE m.lnRec > 0
            GO m.lnRec
            m.lnSuccess = IIF(m.lnSuccess != 2, THIS.HandleRecord(1), 2)
            m.lnRec = GETNEXTMODIFIED(m.lnRec)
        ENDDO
    OTHERWISE                                                       && 无缓冲
        WAIT WINDOW NOBUFF_LOC NOWAIT
ENDCASE
RETURN m.lnSuccess
```

完成 关闭“类设计器”。

8.2.5 用类设计器修改类

当类建立以后，可以对它们进行修改，修改的结果将反映到所有派生于它的子类。也可以根据需要加入新的属性和方法，对错误或不合理的代码进行修改，这种修改将影响所有派生于它的子类，使得子类自动获得改动后的新的特征。

用类设计器创建自定义类的操作步骤如下：

开始 从“项目管理器”选择“类”选项卡。

下一步 选择一个已经建立的类库，并在类库中选择一个类。

下一步 按“修改”按钮。

下一步 在“类设计器”中修改类。

完成 关闭“类设计器”。

注意　在对类修改时，不要修改类的名称属性“Name”，它很可能已经被应用的组件引用。

8.2.6　创建组件

Vixual FoxPro 中有两种类型的类可以包含多个不同的对象，它们是容器类和控制类。对于加在容器类中的对象，在进行表单设计时在表单设计器中可以对每一个对象单独操作，在程序运行时可以用程序对其中的对象单独控制。对于控制类，只有在类设计器中才可以对它进行操作，不能在类设计器之外的表单设计器等地方处理对象中的控件。

在应用程序中，常常需要建立一些可以被大多数应用程序引用的组件，有了这些组件对应用程序的设计将变成简单的组件的组装。

下面通过一个完整的组件构造的例子学习以下内容：

（1）学习使用容器类；

（2）学习在容器类中添加对象；

（3）为新的组件建立属性和方法；

（4）如何进行错误代码的处理。

用类设计器创建一个命令按钮组件的操作步骤如下：

说明　在本例中，将用类设计器建立一个按钮组件，通过该组件用户可以实现表单中的记录的移动操作，如移到上一条、移到首记录等操作。

开始　从“项目管理器”选择“类”选项卡。

下一步　按“新建”按钮。

下一步　在“新类”对话框中的“新类”框输入类名“ViewButton”。

下一步　在“新类”对话框的“派生于”框中选择“Container”。

下一步　在“新类”对话框中的“存储于”框输入“MyCustom”。

下一步　在“新类”对话框中按“确定”。

下一步　在“类设计器”的表单控件工具栏中选择“查看类”。

下一步　在出现的菜单中选择“添加”。

下一步　在“打开”对话框中选择在 8.7 中定义的“custom.vcx”类库，此时“表单控件”工具栏中显示出该类库中的所有类。

下一步　选择“custom.vcx”类库中的“CustomChecker”类。

下一步　在“类设计器”中的 ViewButton 窗体单击鼠标左键，窗体中加入了自定义控件。

下一步　在“类设计器”的表单控件工具栏中选择“命令按钮”。

下一步　在“类设计器”的容器“ViewButton”中画出命令按钮。

下一步　在“类设计器”中选择画出的命令按钮。

下一步　从“标准”工具栏中单击“Copy”按钮。

下一步　从“标准”工具栏中单击三次“Paste”按钮。

下一步　在“类设计器”中调整容器类的大小，并选定四个命令按钮，用“布局”工具栏使它们能整齐的排列，如图 8.7 所示。

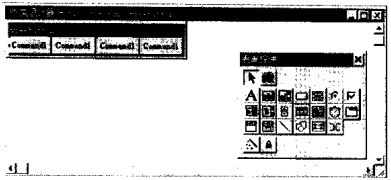


图8.7　加入四个按钮的容器类

从“显示”菜单中选择“属性”。

从“类设计器”中选择第一个命令按钮，在“属性”窗口的“布局”选项卡选择“Caption”属性，在“属性输入框”中输入“首记录”，将“属性”窗口的“Name”属性改为“cmdFirst”。

修改第二个按钮的“Caption”属性为“上一记录”，将“Name”属性改为“cmdPrevious”。

修改第三个按钮的“Caption”属性为“下一记录”，将“Name”属性改为“cmdNext”。

修改第四个按钮的“Caption”属性为“尾记录”，将“Name”属性改为“cmdLast”，完成按钮的外观设计，见图 8.7 所示。

双击“首记录”按钮，在“代码窗口”中的“对象”框，选择“cmdFirst”，在“过程”框中选择“Error”，在代码窗口中输入如下的代码。

程序 8.6　“cmdFirst”的“Error”过程代码

```
Parameters nError, cMethod, nLine
This.Parent.Error(nError, cMethod, nLine)          &&调用容器的错误处理代码
```

在“代码窗口”中的“对象”框，选择“cmdLast”，在“过程”框中选择“Error”，在代码窗口中输入如下的代码。

程序 8.7　“cmdLast”的“Error”过程代码

```
Parameters nError, cMethod, nLine
This.Parent.Error(nError, cMethod, nLine)          &&调用容器的错误处理代码
```

在“代码窗口”中的“对象”框，选择“cmdPrevious”，在“过程”框中选择“Error”，在代码窗口中输入如下的代码。

程序 8.8　“cmdPrevious”的“Error”过程代码

```
Parameters nError, cMethod, nLine
This.Parent.Error(nError, cMethod, nLine)          &&调用容器的错误处理代码
```

下一步 在“代码窗口”中的“对象”框，选择“cmdNext”，在“过程”框中选择“Error”，在代码窗口中输入如下的代码。

程序 8.9　“cmdNext”的“Error”过程代码

```
Parameters nError, cMethod, nLine
This.Parent.Error(nError, cMethod, nLine)          &&调用容器的错误处理代码
```

下一步 双击“首记录”按钮，在“代码窗口”中的“对象”框，选择“cmdFirst”，在“过程”框中选择“Click”，在代码窗口中输入如下的代码。

程序 8.10　“cmdFirst”的“Click”事件代码

```
THIS.Parent.BeforeRecordPointerMoved               &&预处理
GO TOP                                             &&记录移动到首
THIS.Parent.RecordPointerMoved
THIS.Parent.EnableDisableButtons                   &&设置各个按钮的激活状态
```

下一步 在“代码窗口”中的“对象”框，选择“cmdLast”，在“过程”框中选择“Click”，在代码窗口中输入如下的代码。

程序 8.11　“cmdLast”的“Click”事件代码

```
THIS.Parent.BeforeRecordPointerMoved
GO BOTTOM                                          &&记录移动到尾
THIS.Parent.EnableDisableButtons
THIS.Parent.RecordPointerMoved
```

下一步 在“代码窗口”中的“对象”框，选择“cmdPrevious”，在“过程”框中选择“Click”，在代码窗口中输入如下的代码。

程序 8.12　“cmdPrevious”的“Click”事件代码

```
THIS.Parent.BeforeRecordPointerMoved
SKIP -1                                            &&记录移动到上一条
IF BOF()                                           &&如果到了首记录则执行 GO TOP
    GO TOP
ENDIF
THIS.Parent.RecordPointerMoved
THIS.Parent.EnableDisableButtons
```

下一步 在“代码窗口”中的“对象”框，选择“cmdNext”，在“过程”框中选择“Click”，在代码窗口中输入如下的代码。

程序 8.13　"cmdNext" 的 "Click" 事件代码

```
THIS.Parent.BeforeRecordPointerMoved
SKIP 1                                    &&记录移动到上一条
IF EOF()                                  &&如果到了尾记录则执行 GO BOTTOM
   GO BOTTOM
ENDIF
THIS.Parent.RecordPointerMoved
THIS.Parent.EnableDisableButtons
```

下一步 在“类”菜单中选择“新方法程序”。

下一步 在“新方法程序”对话框中的“名称”框输入“BeforeRecordPointerMoved”，按“添加”按钮。

下一步 在“新方法程序”对话框中的“名称”框输入“EnableDisableButtons”，按“添加”按钮。

下一步 在“新方法程序”对话框中的“名称”框输入“RecordPointerMoved”，按“添加”按钮。

下一步 在“类”菜单中选择“新建属性”。

下一步 在“新建属性”对话框中的“名称”框输入“Skiptable”。

下一步 在“新建属性”对话框中的“可视性”框选择“公共”，如图 8.8 所示。

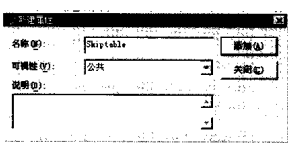


图8.8　添加新属性

下一步 在“新建属性”对话框中按“添加”按钮。

下一步 在“新建属性”对话框中的“名称”框输入“EnableDisableOninit”。

下一步 在“新建属性”对话框中的“可视性”框选择“公共”，按“添加”按钮。

下一步 在“代码窗口”中的“对象”框，选择“ViewButton”，在“过程”框中选择“BeforeRecordPointerMoved”，在代码窗口中输入如下的代码。

程序 8.14　"ViewButton" 的 "BeforeRecordPointerMoved" 代码

```
IF !EMPTY(This.SkipTable)
   SELECT (This.SkipTable)
ENDIF
```

下一步 在“代码窗口”中的“对象”框，选择“ViewButton”，在“过程”框中选择“EnableDisableButtons”，在代码窗口中输入如下的代码。

程序 8.15 "ViewButton"的"EnableDisableButtons"代码

```
LOCAL nRec, nTop, nBottom
IF EOF()                                        && 为空表或者所选的记录为0个
   THIS.SetAll("Enabled", .F.)                  &&使所有的控件处于激活状态
   RETURN
ENDIF
nRec = RECNO()
GO TOP
nTop = RECNO()
GO BOTTOM
nBottom = RECNO()
GO nRec
DO CASE
   CASE nRec = nTop                             &&如果是首记录
      THIS.cmdFirst.Enabled = .F.
      THIS.cmdPrevirous.Enabled = .F.
      THIS.cmdNext.Enabled = .T.
      THIS.cmdLast.Enabled = .T.
   CASE nRec = nBottom                          &&如果是尾记录
        THIS. cmdFirst.Enabled = .T.
        THIS. cmdPrevirous.Enabled = .T.
        THIS.cmdNext.Enabled = .F.
        THIS. cmdLast.Enabled = .F.
      OTHERWISE
        THIS.SetAll("Enabled", .T.)             &&使所有的控件处于激活状态
ENDCASE
```

下一步 在"代码窗口"中的"对象"框，选择"ViewButton"，在"过程"框中选择"Error"，在代码窗口中输入如下的代码。

程序 8.16 "ViewButton"的"Error"代码

```
Parameters nError, cMethod, nLine
#define NUM_LOC "错误代号: "
#define PROG_LOC "过程: "
#define MSG_LOC "错误信息: "
#define CR_LOC CHR(13)
#define SELTABLE_LOC "所选的表:"
#define OPEN_LOC "打开"
#define SAVE_LOC "你想保存更改么?"
```

```
#define CONFLICT_LOC "无法处理冲突."
DO CASE
CASE nError = 13                                          && 没有发现别名
*-----------------------------------------------------------------------*
*当表还没有打开时，如果用户想移动记录，将显示一个打开表的对话框
*-----------------------------------------------------------------------*
cNewTable = GETFILE('DBF', SELTABLE_LOC, OPEN_LOC)
IF FILE(cNewTable)
   SELECT 0
   USE (cNewTable)
   This.SkipTable = ALIAS()
ELSE
    This.SkipTable = ""
ENDIF
CASE nError = 1585
*-----------------------------------------------------------------------*
* 由 CustomChecker 处理更新操作.
*-----------------------------------------------------------------------*
nConflictStatus = THIS.CustomChecker1.CheckConflicts()
IF nConflictStatus = 2
   WAIT WINDOW CONFLICT_LOC
ENDIF
OTHERWISE
*-----------------------------------------------------------------------*
* 显示一个没有定义的错误.
*-----------------------------------------------------------------------*
   lcMsg = NUM_LOC + ALLTRIM(STR(nError)) + CR_LOC + CR_LOC + ;
           MSG_LOC + MESSAGE( )+ CR_LOC + CR_LOC + ;
           PROG_LOC + PROGRAM(1)
   lnAnswer = MESSAGEBOX(lcMsg, 2+48+512)
   DO CASE
      CASE lnAnswer = 3                                   &&退出
         CANCEL
      CASE lnAnswer = 4                                   &&重试
         RETRY
      OTHERWISE
         RETURN
   ENDCASE
ENDCASE
```

下一步 在“代码窗口”中的“对象”框，选择“ViewButton”，在“过程”框中选择“Init”，在代码窗口中输入如下的代码。

程序 8.17　“ViewButton”的“Init”代码

```
IF THIS.EnableDisableOnInit
   THIS.EnableDisableButtons
ENDIF
```

下一步 在“代码窗口”中的“对象”框，选择“ViewButton”，在“过程”框中选择“RecordPointerMoved”，在代码窗口中输入如下的代码。

程序 8.18　“ViewButton”的“RecordPointerMoved”代码

```
IF TYPE('_VFP.ActiveForm') = 'O'
   _VFP.ActiveForm.Refresh
ENDIF
```

完成 关闭“类设计器”。

8.2.7　修改和保护属性和方法

对于已经建立的类的属性和方法，可以进行修改也可以将属性和方法加以保护[1]。

技巧 当为一个用户定义的类设置属性时，用户可能会输入非法的属性，这样很可能会引起程序的错误。为了避免这种情况发生，在“新建属性”对话框的“说明”框中请输入属性的正确设置的说明，例如当属性只能设置成 0、1、2 时，请加以说明。必要的时候，在代码中还需要对属性值进行确认。当建立类时，对每个类最好加以说明，关键的应用要重点突出。

在需要的时候，很可能需要将一些属性或方法保护起来，这样在“表单设计”或其它的关于这个类的对象的使用时，便看不到受到保护的属性和方法。

属性和方法的保护的操作步骤如下：

说明 在本例中将学习如何修改已经有的属性和方法，如何为类建立必要的说明，如何浏览类的信息，如何保护类的属性和方法。

开始 从“帮助菜单”选择“示例应用程序”。

下一步 在“帮助对话框”中选择“Solutions Samples”，然后按“Open”。

下一步 在“项目管理器”中选择“类”选项卡，然后选择“Samples”类库，然后选择“stopwatch”类，按“修改”按钮。

下一步 在“类”菜单中，选择“类信息”。

下一步 在“类信息”对话框中选择“类”选项卡，查看当前类的特性，如图 8.9 所示。

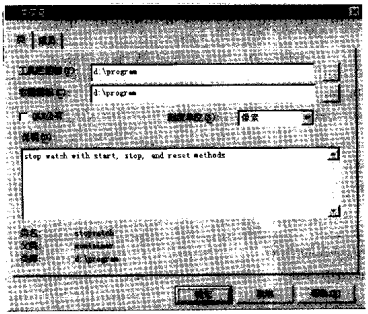


图8.9 浏览类信息

说明 在“类”选项卡中“工具栏图标”指明了当类被加入到“表单控件”工具栏中时的图标，容器图标指明了在类浏览中的图标形式，“OLE 公有”复选框选择时，当类在项目管理器中编译时，会产生一个自动化的 OLE 服务器，一个全局的标识或 GUID 也将在程序中产生。说明对话框中可以加入对类的说明信息，在最下面将显示当前的类名、父类名和所在的类库名。

下一步 在“类信息”对话框中选择“成员”选项卡，将显示类中的所有属性和方法的信息，见图 8.10 所示。

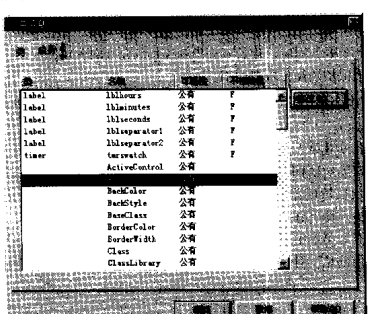


图8.10 类成员信息

说明　在“成员”选项卡中，“名称”指明了当前类的属性或者方法名称。“可视性”指明了当前的属性或方法是“公有”、“保护”还是“隐藏”的。“不初始化”表示对象被加入时是否执行 Init 方法，当此值为真（.T.）时，Init 不被执行。

步骤1　在“成员”选项卡中选择一个类如“Timer”，然后选择一个属性，如“nhour”，然后按“修改”按钮。

步骤2　在出现的“编辑属性/方法程序”对话框中，可以对选中的“nhour”属性的可视性进行修改，如图 8.11 所示。

步骤3　在“说明”框中加入对“nhour”的说明“小时数”。

步骤4　按“应用”完成对“nhour”属性的修改。

步骤5　在“属性或方法程序”对话框中选择“nmin”，在“说明”框输入“分钟数”。

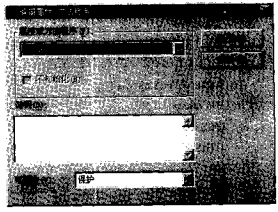


图8.11　编辑属性/方法程序

步骤6　在“属性或方法程序”对话框中选择“nsec”，在“说明”框输入“秒数”。

步骤7　按“关闭”完成对属性的修改。

步骤8　在“类信息”对话框中按“确定”。

步骤9　在“类设计器”中双点击“stopwatch”，在出现的代码窗口中“对象”选择“stopwatch”，过程选择“updatedisplay”，代码窗口有如下的代码（这里由作者加上了必要的注释）。

说明　在“stopwatch”类中，有一个保护方法“updatedisplay”和三个保护属性“nhour、nmin 和 nsec”，在类中有三个自定义方法“Start、Stop、Reset 和 increment”。

程序 8.19　“stopwatch”的“updatedisplay”代码

```
*-------------------------------------------------------------------------------*
*本方法用来设置三个标签的标题，反映出所耗时间。
*为了确保正确的功能，本方法设为保护方法，防止用户从外部修改调用此方法
*-------------------------------------------------------------------------------*
*将数值型转化为字符型
cSecDisplay = ALLTRIM(STR(This.nSec))
cMinDisplay = ALLTRIM(STR(This.nMin))
cHourDisplay = ALLTRIM(STR(This.nHour))
*设置标题的标签，如果数值的属性值小于 0，标题从 0 开始
This.lblSeconds.Caption = IIF(This.nSec < 10, "0" + cSecDisplay , cSecDisplay)
This.lblMinutes.Caption = IIF(This.nMin < 10, "0" + cMinDisplay , cMinDisplay)
This.lblHours.Caption = IIF(This.nHour < 10, "0" + cHourDisplay , cHourDisplay)
```

下一步 在代码窗口中“对象”选择“stopwatch”，过程选择“increment”，代码窗口有如下的代码。

程序 8.20　“stopwatch”的“increment”代码

```
*-------------------------------------------------------------------------------*
*每一次计时器事件触发（每秒一次）将调用本方法，用来
*设置三个计时属性“nhour、nmin 和 nsec”的值
*-------------------------------------------------------------------------------*
THIS.nSec = THIS.nSec + 1                          &&每秒钟将 nsec 属性值加 1
IF THIS.nSec = 60                  &&如果 nsec 属性值到达 60，nsec 值置 0，nmin 属性值加 1
   THIS.nSec = 0
   THIS.nMin = THIS.nMin + 1
ENDIF
IF THIS.nMin = 60                  &&如果 nmin 属性值到达 60，nmin 值置 0，nhour 属性值加 1
   THIS.nMin = 0
   THIS.nHour = THIS.nHour + 1
ENDIF
THIS.UpdateDisplay                          &&设置属性值后调用 UpdateDisplay 方法
THIS.ElapsedSeconds = (THIS.nHour * 3600) + ;          &&设置当前经过的总秒数
                      (THIS.nMin * 60) + (This.nSec)
```

下一步 在代码窗口中“对象”选择“stopwatch”，过程选择“start”，代码窗口有如下的代码。

程序 8.21　“stopwatch”的“start”代码

```
This.tmrSWatch.enabled = .T.                                    &&启动跑表
```

下一步 在代码窗口中“对象”选择“stopwatch”，过程选择“stop”，代码窗口有如下的代码。

程序 8.22　“stopwatch”的“stop”代码

```
This.tmrSWatch.enabled = .f.                                    &&终止跑表
```

下一步 在代码窗口中“对象”选择“stopwatch”，过程选择“Reset”，代码窗口有如下的代码。

程序 8.23　“stopwatch”的“Reset”代码

```
This.nSec = 0
This.nMin = 0
This.nHour = 0
This.UpdateDisplay                                              &&重新设置跑表
```

下一步 在代码窗口中“对象”选择“tmrSWatch”，过程选择“Timer”，代码窗口有如下的代码。

程序 8.24　“stopwatch”的“Timer”代码

```
THIS.Parent.Increment          &&通过调用在 stopwatch 中的方法每秒钟响应一次 Timer 事件
```

下一步 在属性窗口中“对象”框选择“tmrSWatch”，属性选择“Iterval”，它的值为1000，表示每毫秒触发一次 Timer 事件。

下一步 在属性窗口中“对象”框选择“stopwatch”，选择“其他”框选择“Elapsedseconds”，它的属性默认为0。

说明 所有的新属性在创建时的默认值为（.F.），给属性设置一个不同的默认值，可以使用“属性”窗口的“其他”选项卡，单击属性并修改属性值，当把类添加到表单中时，这个值将作为属性的默认值。

完成 关闭“类设计器”。

8.2.8　复制删除类库中的类

把类库加入项目后，可以十分容易地实现把一个类从类库中删除或者从一个库中复制到另一个库。

复制删除类库中的类的操作步骤如下：

开始 从“项目管理器”选择“类”选项卡。

下一步 选择类库，按左侧的“田”按钮“+”。

下一步 选择一个类，并用鼠标拖拽到另一个类库中，实现了类的复制。

下一步 选择一个类，按“移去”按钮。

下一步 在消息对话框中选择“移去”，从类库中删除类。

完成 关闭“项目管理器”

说明 可以用 RENAME CLASS 命令更改类库中的类的名称。但类库名称修改后，在其他的类库文件中的有关这个类的子类仍然使用旧的名称。

在“工具”菜单中选择“类浏览器”，打开相应的类库，可以浏览该类库中类的所有信息。

8.3　用编程的方法设计类

8.3.1　程序中定义类的语法

在程序中，可以用编程的方法定义类，存于程序文件.PRG 中。程序的代码必须在类定义之前，不能先定义类在写程序代码。在 Visual FoxPro 中用程序创建类的语法是：

```
DEFINE CLASS ClassName1 AS cBaseClass
  [[PROTECTED PropertyName1, PropertyName2 ...]
      PropertyName = eExpression ...]
  [[HIDDEN PropertyName1, PropertyName2 ...]
```

```
    PropertyName = eExpression ...]
  [ADD OBJECT [PROTECTED] ObjectName AS ClassName [NOINIT]
    [WITH cPropertylist]]
  [[PROTECTED| HIDDEN] FUNCTION | PROCEDURE Name
    cStatements
  [ENDFUNC | ENDPROC]]
ENDDEFINE
```

8.3.2 保护类成员

如果你想保护或者隐藏类对象中的属性或方法，可以使用 DEFINE CLASS 命令中的 PROTECTED 和 HIDDED 关键字。

例如在雇员类中，如果你不想有人从外部改动受雇日期“HireDate”属性，可以把它保护起来，而通过一个方法 GetHireDate 来获得雇员的受雇日期。这个类的定义如下面类定义的代码。

程序 8.25　定义类

```
DEFINE CLASS employee AS CUSTOM
PROTECTED HireDate                          &&将受雇日期保护起来
    First_Name = ""
    Last_Name = ""
    Address = ""
    HireDate = {//}

PROCEDURE GetHireDate
    RETURN This.HireDate
ENDPROC
ENDDEFINE
```

8.3.3 从类中创建对象

当已经建立了可视类，在程序中可以用 CREATEOBJECT( )函数创建一个对象。

采用代码建立对象的程序如下所示，介绍如何从存于类库 FORMS.VCX 的类库中的类中创建一个表单对象，并指出如何引用这个表单。

程序 8.26　从类库中创建对象

```
*首先打开表单类所在的类库，ADDITIVE 避免在设置类库时关闭其他的已经打开的类库
SET CLASSLIB TO Forms ADDITIVE
frmTest = CREATEOBJECT("TestForm")      &&从类库中创建基于 TestForm 类的表单 frmTest。
frmTest.Show                                                    &&显示表单
```

8.3.4　向容器类中添加对象

在定义类时使用 ADD OBJECT 子句或者 AddObject 方法可以向容器类中添加对象。

1. 向容器中添加对象

说明　本实例介绍基于“表单”类的一个类，采用 ADD OBJECT 命令向表单中加入了两个命令按钮。

程序 8.27　向容器中添加对象

```
DEFINE CLASS myform AS FORM
    ADD OBJECT cmdOK AS COMMANDBUTTON
    ADD OBJECT PROTECTED cmdCancel AS COMMANDBUTTON
ENDDEFINE
```

2. 用AddObject向容器对象中添加对象

说明　如果程序中已经创建了容器对象，则需要用 AddObject 方法向容器中添加对象，在下面的程序代码中创建了一个表单对象，并向它加入两个命令按钮。

程序 8.28　用 AddObject 向容器对象中添加对象

```
frmMessage = CREATEOBJECT("FORM")
frmMessage.AddObject("txtText1", "TEXTBOX")
frmMessage.AddObject("txtText2", "TEXTBOX")
```

3. 在类中用AddObject向容器中添加对象

说明　如果在类定义时向程序中添加代码，可以在类的方法中使用 AddObject 方法向容器中添加对象，下面的程序代码用 AddObject 向表格的列中添加控件。

程序 8.29　在类中用 AddObject 向容器中添加对象

```
DEFINE CLASS mygrid AS GRID                          &&定义表格类
ColumnCount = 3                                      &&设定列数
PROCEDURE Init                                       &&在初始化表格时加入控件
    THIS.Column2.AddObject("cboClient", "COMBOBOX")
    THIS.Column2.CurrentControl = "cboClient"
ENDPROC
```

8.3.5　用程序创建按钮组件

前面采用“类设计器”建立了一个按钮组件，本小节采用程序的方法建立与之相似的 NavButton 的定义。

建立 NavButton 组件的操作步骤如下：

说明　本实例用命令按钮实现用户在表中移动记录指针，如上移一条、下移一条，移到首记录，移到尾记录等操作。

开始 设计定位按钮。因为所有的按钮的操作都具有共同的特点，因而首先定义一个具有相同的外观、功能的定位按钮类 NavButton，另外需要由 NavButton 派生的四个子类按钮 navFirst、navLast、navPrevious、navNext 来完成记录的移到首记录，移到尾记录，上移一条记录，下移一条记录的操作。最后需要建立 vcr 容器类，将按钮加入到容器类 vcr 中。该容器可以加入到表单或工具栏。

下一步 从“文件”菜单中选择“新建”。

下一步 选择“程序”，然后选择“新建文件”。

下一步 定义 NavButton 类。

程序 8.30　定义 NavButton 类

```
DEFINE CLASS Navbutton AS COMMANDBUTTON                    &&定义定位按钮的父类
Height = 25
Width = 100                                  &&定义按钮的大小和一个自定义属性 TableAlias
TableAlias = ""                               && TableAlias 用来存储执行操作的表的别名
PROCEDURE Click
IF NOT EMPTY(This.TableAlias)                 &&如果设置了 TableAlias 在子类定位代码执行前，
    SELECT (This.TableAlias)                  &&父类选择具有该别名的表，否则假定为当前工作区
ENDIF
ENDPROC
PROCEDURE RefreshForm          &&用_SCREEN.ActiveForm.Refresh 代替 THISFORM.Refresh，
  _SCREEN.ActiveForm.Refresh                  &&从而允许在表单或工具栏中添加类
ENDPROC
ENDDEFINE&&结束类定义.
```

下一步 定义 NavFirst，NavLast，NavPrevious 和 NavNext 类。

程序 8.31　定义 NavFirst，NavLast，NavPrevious 和 NavNext 类

```
*-------------------------------------------------------------------------------*
*定义 navFirst 类
*-------------------------------------------------------------------------------*
DEFINE CLASS navFirst AS Navbutton
  Caption = "首记录"                                        &&定义 Caption 属性.
  PROCEDURE Click                                  &&建立 Click 事件发生时的执行代码
    DODEFAULT( )                          &&执行基类 Navbutton 代码选择 TableAlias 属性的工作区
    GO TOP                                                  &&将记录移动到表头
    THIS.RefreshForm                  &&调用父类的 RefreshForm 方法，这里没有使用(::)操作符，
                                                  &&因为在子类中没有和父类重名的方法
  ENDPROC                                                   &&结束 Click 事件的过程
ENDDEFINE                                                   &&结束类定义
```

```
*-------------------------------------------------------------------------------*
*定义 navLast 类
*-------------------------------------------------------------------------------*
DEFINE CLASS navLast AS Navbutton
   Caption = "尾记录"                                    &&定义 Caption 属性.
   PROCEDURE Click                                       &&建立 Click 事件发生时的执行代码
      DODEFAULT( )                                       &&执行基类 Navbutton 代码选择 TableAlias 属性的工作区
      GO BOTTOM                                          &&将记录移动到表尾
      THIS.RefreshForm                                   &&调用父类的 RefreshForm 方法，这里没有使用(::)操作符，
                                                         &&因为在子类中没有和父类重名的方法
   ENDPROC                                               &&结束 Click 事件的过程
ENDDEFINE                                                &&结束类定义
*-------------------------------------------------------------------------------*
*定义 navPrevious 类
*-------------------------------------------------------------------------------*
DEFINE CLASS navPrevious AS Navbutton
   Caption = "上一记录"                                  &&定义 Caption 属性.
   PROCEDURE Click                                       &&建立 Click 事件发生时的执行代码
      DODEFAULT( )                                       &&执行基类 Navbutton 代码选择 TableAlias 属性的工作区
      SKIP -1                                            &&将记录移动到上一条
      IF BOF( )                                          &&如果到文件首，指针指到文件首
         GO TOP
      ENDIF
      THIS.RefreshForm                                   &&调用父类的 RefreshForm 方法，这里没有使用(::)操作符，
                                                         &&因为在子类中没有和父类重名的方法
   ENDPROC                                               &&结束 Click 事件的过程
ENDDEFINE                                                &&结束类定义
*-------------------------------------------------------------------------------*
*定义 navNext 类
*-------------------------------------------------------------------------------*
DEFINE CLASS navNext AS Navbutton
   Caption = "上一记录"                                  &&定义 Caption 属性.
   PROCEDURE Click                                       &&建立 Click 事件发生时的执行代码
      DODEFAULT( )                                       &&执行基类 Navbutton 代码选择 TableAlias 属性的工作区
      SKIP  1                                            &&将记录移动到上一条
      IF EOF( )                                          &&如果到文件尾，指针指到文件尾
         GO BOTTOM
      ENDIF
      THIS.RefreshForm                                   &&调用父类的 RefreshForm 方法，这里没有使用(::)操作符，
```

```
                                        &&因为在子类中没有和父类重名的方法
    ENDPROC                             &&结束 Click 事件的过程
ENDDEFINE                               &&结束类定义
```

下一步 定义 vcr 容器类。

程序 8.32　定义 vcr 容器类

```
*-------------------------------------------------------------------------*
*定义 vcr 容器类，该加入了四个定位按钮
*-------------------------------------------------------------------------*
DEFINE CLASS vcr AS CONTAINER
    Height = 25                         && 容器类的 Height 定义为四个命令按钮的高度
    Width = 400                         &&容器类的 Width 设为 400，是四个按钮宽度的和
    Left = 3
    Top = 3
    ADD OBJECT cmdTop AS navFirst ;     &&加入四个按钮，每个按钮由 Left 决定初始位置
        WITH Left = 0
    ADD OBJECT cmdPrior AS navPrevious ;
        WITH Left = 100
    ADD OBJECT cmdNext AS navNext ;
        WITH Left = 200
    ADD OBJECT cmdBot AS navBottom ;
    WITH Left = 300
PROCEDURE SetTable(cTableAlias)         &&设置 TableAlias 属性，该属性在 NavButton 定义
    IF TYPE("cTableAlias") = 'C'
        THIS.cmdTop.TableAlias = ;
            cTableAlias
        THIS.cmdPrior.TableAlias = ;
            cTableAlias
        THIS.cmdNext.TableAlias = ;
            cTableAlias
        THIS.cmdBot.TableAlias = ;
            cTableAlias
        ENDIF
    ENDPROC
ENDDEFINE                               &&结束类定义
```

下一步 建立基于 vcr 的子类 vcr2。

说明 你可以建立基于 vcr 的子类，在子类中可以附加诸如“保存、编辑、退出”等按钮，如图 8.12

所示。

图8.12 创建基于父类vcr的子类

程序 8.33 定义子类 vcr2

```
DEFINE CLASS vcr2 AS vcr                    &&从 vcr 派生出一个新类 vcr2
ADD OBJECT cmdQuit AS COMMANDBUTTON WITH ;  &&向其中加入一个按钮
    Caption = "退出",;                       &&定义 Caption 属性.
    Height = 25, ;
    Width = 100
    Width = THIS.Width + THIS.cmdQuit.Width
    cmdQuit.Left = THIS.Width - ;
                THIS.cmdQuit.Width
PROCEDURE cmdQuit.CLICK                     &&当用户单击 cmdQuit 时代码释放表单
    RELEASE THISFORMENDPROC
ENDDEFINE                                   &&结束类定义
```

下一步 向表单中加入 vcr。

vcr可以作为一个控件，加入到表单中。本例的NAVCLASS.PRG中代码定义了一个加入vcr的表单。

程序 8.34 向表单中加入 vcr

```
DEFINE CLASS NavForm AS Form
   ADD OBJECT oVCR AS vcr
ENDDEFINE
```

下一步 将以上定义的所有代码存于 NAVCLASS.PRG。

下一步 从“文件”菜单中选择“新建”。

下一步 选择“程序”，然后选择“新建文件”。

下一步 加入如下加载 vcr 类定义的代码。

程序 8.35 加载 vcr 类

```
SET PROCEDURE TO navclass ADDITIVE
```

下一步 创建基于 NavForm 类的对象。

程序 8.36 创建基于 NavForm 类的对象

```
frmTest = CREATEOBJECT("navform")
```

下一步 用 Show 显示表单。

程序 8.37　用 Show 显示表单

```
frmTest.Show
```

完成 将以上定义的代码存于 SHOWFRM.PRG。

8.3.6　用程序定义表格控件

在 Visual FoxPro 中表格是一个复杂的控件，它由列组成，列又包括标头和其他的控件。在列中，默认的编辑控件是一个文本框。但是通过自己定制表格，可以扩充它的功能。

在图 8.13 中显示的是一个含有表格对象的表单。表单中的第二列为复选框，可以输入逻辑字段值。

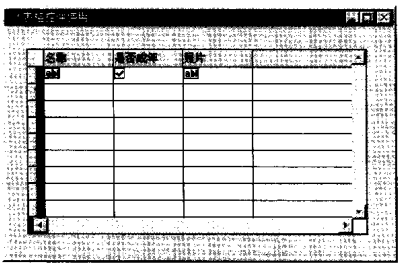


图8.13　含有表格控件的表单

建立表格控件的程序如下：

说 明 本实例用程序实现用户自定义的表格类，并设计标头和编辑的控件，在表中移动记录指针，如上移一条、下移一条，移到首记录，移到尾记录等操作。

开始 从“文件”菜单中选择“新建”。

下一步 选择“程序”，然后选择“新建文件”。

下一步 输入含有复选框的表格类的定义，如下代码所示。

程序 8.38　定义表格类

```
DEFINE CLASS grdPerson AS Grid
    Left = 24                                    &&设置表格的外观属性
    Top = 10
    Width = 295
    Height = 210
    Visible = .T.
```

```
    RowHeight = 28
    ColumnCount =3                          && ColumnCount 属性为 3，表示有三列
Column1.ControlSource ="住户.名称"          &&设置列的数据源
Column2.ControlSource ="住户.是否成年"      &&是否成年字段为逻辑型
Column3.ControlSource ="住户.照片"
Column2.Sparse = .F.   &&Column2 含有复选框，设置 Sparse 属性为 .F.使得所有的复选框为可视
Procedure Init
    THIS.Column1.Width = 100                &&设置列宽和标头的标题
    THIS.Column2.Width = 68
   THIS.Column3.Width = 100
    THIS.Column1.Header1.Caption = "名称"
    THIS.Column2.Header1.Caption = "是否成年"
    THIS.Column3.Header1.Caption = "照片"
    THIS.Column2.AddObject("chk1", "checkbox")   &&为第二列加入复选框对象
    THIS.Column2.CurrentControl = "chk1"         &&设置当前的控件为复选框
    THIS.Column2.chk1.Visible = .T.
    THIS.Column2.chk1.Caption = ""
 ENDPROC
ENDDEFINE                                   &&结束类定义
```

下载定义含有表格类的表单类，如下代码所示。

程序 8.39　定义表单类

```
DEFINE CLASS GridForm AS FORM
    Width = 330                             &&设置表单类的外观属性
    Height = 250
    Caption = "Grid Example"
    ADD OBJECT grid1 AS grdProducts         &&添加表格类对象
PROCEDURE Destroy                           &&创建该类的对象使用 READ EVENTS
    CLEAR EVENTS                &&在表单的 Destroy 事件中使用 CLEAR EVENTS 关闭表单
ENDPROC
ENDDEFINE                                   &&结束类定义
```

下载在程序类定义的最前端加入如下的代码，创建基于 GridForm 的对象，并发出 READ EVENTS 命令。

程序 8.40　创建基于 GridForm 的对象

```
CLOSE DATABASE
OPEN DATABASE (SYS(2004) + "Pratice\data\人事.dbc")
USE 住户
```

```
frmTest= CREATEOBJECT("GridForm")
frmTest.Show
READ EVENTS
```

完成 将以上定义的代码存于 SHOWGRID.PRG。

8.3.7 把类成员定义成数组

可以用数组定义类的成员。

用数组定义类的成员的程序如下：

说明 本实例介绍如何把类成员定义成数组，Choices 是关于一组控件的数组。

程序 8.41 用数组定义类的成员

```
DEFINE CLASS MoverListBox AS CONTAINER
   DIMENSION choices[3]                              &&定义对象数组
   ADD OBJECT lstFromListBox AS LISTBOX
   ADD OBJECT lstToListBox AS LISTBOX
   ADD OBJECT choices[1] AS COMMANDBUTTON            &&给对象数组添加控件
   ADD OBJECT choices[2] AS COMMANDBUTTON
   ADD OBJECT choices[3] AS CHECKBOX
   PROCEDURE choices.CLICK                           &&单击按钮数组的响应过程
      PARAMETER nIndex
      DO CASE
         CASE nIndex = 1
            *按第一个命令按钮的响应代码
         CASE nIndex = 2
            *按第二个命令按钮时的响应代码
         CASE nIndex = 3
            * 按第三个复选框时的响应代码
      ENDCASE
   ENDPROC
ENDDEFINE                                            &&结束类的定义
```

8.3.8 创建对象数组

可以创建对象数组。

创建对象数组的程序如下：

程序 8.42 创建对象数组

```
DIMENSION MyArray[5]                                 &&声明对象数组
FOR x = 1 TO 5
```

```
MyArray[x] = CREATEOBJECT("COMMANDBUTTON")          &&给数组成员添加对象
ENDFOR
```

注意 创建对象数组时，要注意以下几点：

数组中的每个成员需要单独的赋给对象的值。

数组中成员的调用要一个成员一个成员的调用，不能作为一个整体调用，例如代码：Marry.Enable=.F.将发生错误。

在 Visual FoxPro 中数组的维数可以重新设定，当新的维数比初始化的数组维数大时，所有新增的元素将被定为.F.。当新维数小于初始化数组的维数时，那些下标大于新下标中最大值的数组将被释放。

8.3.9　使用对象存储数据

面向对象的语言中最主要的是类的构造，在类中定义了和每个实体对象相关的数据和方法。例如，在一个关于“客户”的类的定义中，可以定义与这个客户有关的信息，同时也可以定义计算这个顾客的年龄的方法。

通常 Visual FoxPro 将数据存储于数据库的表中，应用程序运行时，首先是从数据库表中通过查询等操作提取相关的信息。对于得到的信息，将它们转化为一个对象来处理，将带来巨大的方便，此时数据和实体是相对应的。

用对象来处理数据的操作步骤如下：

说明 本实例定义了用户类 NewUser，在数据库表中存储了所有用户的信息，包括用户的 ID 号，用户的口令等，用户登录后，创建一个新用户对象 oUser，同时调用新的用户的方法进行处理，判断用户登录的正确性。

下一步 从“文件”菜单中选择“新建”。

下一步 选择“程序”，然后选择“新建文件”。

下一步 在代码窗口输入程序 NEWUSER.PRG。

程序 8.43　程序 NEWUSER.PRG

```
DEFINE CLASS NewUser AS CUSTOM                          &&定义用户类
    PROTECTED LogonTime, AccessLevel&&定义保护的类变量
    UserId = ""                                         &&用户标识
    PassWord = ""                                       &&口令
    LogonTime = { / / : : }                             &&登录时间
    AccessLevel = 0                                     &&用户级别
    PROCEDURE  Logon
        DO FORM LOGON WITH ;                  &&  调用你已经定义好的登录窗口的表单
            This.UserId, ;                              &&登录表单参数
            This.PassWord, ;
            This.AccessLevel
        This.LogonTime = DATETIME( )
```

```
    ENDPROC
* 创建获得保护属性的方法.
    PROCEDURE GetLogonTime                          &&获得登录时间
        RETURN This.LogonTime
    ENDPROC
    PROCEDURE GetAccessLevel                        &&获得用户级别
        RETURN This.AccessLevel
    ENDPROC
ENDDEFINE                                           &&结束类定义
```

更成 保存 NEWUSER.PRG 程序。

主程序中创建新用户的对象用如下：

程序 8.44　主程序中创建新用户的对象

```
oUser = CREATEOBJECT('NewUser')
oUser.Logon
```

当需要当前用户的信息时，可以通过执行oUser对象的方法来获得。如：

程序 8.45　通过执行 oUser 对象获得当前用户的信息

```
IF oUser.GetAccessLevel() >= 4
    DO ADMIN.MPR
ENDIF
```

8.4　对类和对象的引用

8.4.1　容器中对象的引用层次

类建立以后，如果向表单或容器中添加了对象，对对象的引用需要知道对象引用的层次，图 8.14 指出了 Visual FoxPro 中的对象的层次关系。

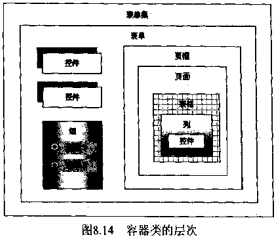


图8.14　容器类的层次

如果要引用对象，可以采用如下的命令：

程序 8.46 引用对象

```
frsFormSet.frmForm1.cmdButton1
```

如果使表格列中的控件无效，需要按照如下的层次提供引用：

程序 8.47 使表格列中的控件无效

```
Formset.Form.PageFrame.Page.Grid.Column.Control.Enabled = .F.
```

对于应用程序对象(_VFP)的 ActiveForm 属性允许在不知道表单名称的情况下处理当前的活动表单。例如：

程序 8.48 引用当前的活动表单

```
_VFP.ActiveForm.BackColor = RGB(255,255,255)
```

与之类似，ActiveControl 属性允许你处理当前活动表单的活动控件，例如：

程序 8.49 引用当前的活动表单的活动控件

```
_VFP.ActiveForm.ActiveControl.Name
```

8.4.2 对象的相对引用

除了采用绝对的名称引用对象以外，还可以采用相对的方式引用容器中的对象。在 Visual FoxPro 中，有一些属性和关键字，采用它们可以更为简捷方便地从类的层次中引用对象。在表 8.3 中列出了这些属性和关键字。

表8.3 相对引用的属性和关键字

属性或关键字	引用的含义
Parent	包含该对象的容器
THIS	当前对象
THISFORM	当前表单
THISFORMSET	当前表单集

对象的相对引用的程序如下：

说明 下面的代码使用 THISFORMSET, THISFORM, THIS,和 Parent 实现对对象属性的相对引用。

程序 8.50 对象的相对引用

```
THISFORMSET.frm1.cmd1.Caption = "OK"     &&表单集中的任何表单中的控件的事件或属性代码
THISFORM.cmd1.Caption = "OK"             &&在 cmd1 所在的任何表单中的事件或属性代码中
THIS.Caption = "OK"                      &&对需要改变标题的控件的事件或属性代码中
THIS.Parent.BackColor = RGB(192,0,0)     &&将控件所在的表单的背景色设置为暗红色
```

8.4.3 属性的设置

在运行时设计属性，请采用如下的语法：

Container.Object.Property = Value

1. 在运行时设置对象属性

说 明 下面的代码设置表单 frmPhonelog 中的 txtDate 的属性。

程序 8.51 运行时设置对象属性

```
frmPhoneLog.txtDate.Value = DATE( )          && 显示当前日期
frmPhoneLog.txtDate.Enabled = .T.            && 使得控件有效
frmPhoneLog.txtDate.ForeColor = RGB(0,0,0)   &&文本为黑色
frmPhoneLog.txtDate.BackColor = RGB(192,192,192)   && 背景为灰色
```

如果设置多个属性可以采用 WITH...ENDWITH 结构简化设置属性的代码。

2. 用WITH...ENDWITH结构设置多个属性

说 明 下面的代码设置表单集中表单的表格列的多个属性的值。

程序 8.52 用 WITH...ENDWITH 结构设置多个属性

```
WITH THISFORMSET.frmForm1.grdGrid1.grcColumn1
    .Width = 5
    .Resizable = .F.
    .ForeColor = RGB(0,0,0)
    .BackColor = RGB(255,255,255)
    .SelectOnEntry = .T.
ENDWITH
```

8.4.4 方法的调用

在运行时方法的调用，请采用如下的语法：

Parent.Object.Method

在运行时调用方法的代码如下：

说 明 本实例调用方法显示表单，并将焦点设置于命令按钮上。

程序 8.53 运行时调用方法

```
frsFormSet.frmForm1.Show
frsFormSet.frmForm1.txtGetText1.SetFocus
```

如果方法具有返回值，则必须以圆括号结尾，例如：

Form1.Caption = Form1.GetNewCaption( )

注意　传递给方法的参数必须放在方法名后的圆括号中，如 Form1.Init（cParam）。

8.4.5 事件的响应

事件通常包括键盘事件和鼠标事件，当事件发生，事件中的代码被执行。在编程时，用户可以调用相关的事件中的过程。例如，使用 MOUSE 命令，可以产生 Click, DblClick, MouseMove, 和 DragDrop 鼠标事件。下面的语句调用表单 frmPhoneLog 的 Activate 事件代码，但是并不真正地激活这个表单。

```
frmPhoneLog.Activate
```

如果要真正地显示表单，需要调用 Show 方法，此时将调用 Activate 事件的代码。

```
frmPhoneLog.Show
```

8.4.6 创建对对象的引用

对象可以被复制，也可以通过引用来执行对象的过程或方法。

可以用编程的方法建立对象的引用。

引用对象的操作步骤如下：

说明　本实例定义了类 InvoiceForm 并创建了对它的一个引用。

下一步　从“文件”菜单中选择“新建”。

下一步　选择“程序”，然后选择“新建文件”。

下一步　在代码窗口输入程序 NEWWINV.PRG。

程序 8.54　程序 NEWWINV.PRG

```
*-----------------------------------------------------------------*
*--程序名：NEWINV.PRG
*--功能：返回对一个新的发票表单的引用
*-----------------------------------------------------------------*
frmInv = CREATEOBJECT("InvoiceForm")
RETURN frmInv                          &&返回一个 InvoiceForm 的表单对象
DEFINE CLASS InvoiceForm AS FORM
    ADD OBJECT txtCompany AS TEXTBOX
    *其他的代码
ENDDEFINE
```

下一步　保存文件为 NEWWINV.PRG。

下一步　从“文件”菜单中选择“新建”。

下一步　选择“程序”，然后选择“新建文件”。

下一步　在代码窗口输入定义 REFERENCE.PRG 的程序如下。

程序 8.55　定义 REFERENCE.PRG 的程序

```
frmInvoice = NewInv()                  && 将对表单对象的引用存于一个变量
```

```
frmInvoice.SHOW
IF TYPE ("frmInvoice")="O" AND NOT ISNULL(frmInvoice)        &&检查对象是否存在
    txtCustName = frmInvoice.txtCompany                      &&对表单中的对象的引用
    txtCustName.Value = "Visual FoxPro 新客户"
ELSE
    WAIT WINDOW "创建对象错误！"
ENDIF
```

说明　使用 TYPE（）和 ISNULL（）函数可以检查对象是否存在。当关闭表单后，表单对象变量存一个.NULL.值，但它的类型仍然是 "O"，所以必须还要用 ISNULL（）判断表单是否关闭。

```
RELEASE txtCustName     &&释放对象引用
RELEASE frmInvoice &&释放原始对象引用
```

注意　如果对象的引用存在，用 RELEASE 命令不能完全从内存中释放它。如上例中的 RELEASE txtCustName 首先释放了对 txtCustName 的引用，然后才用 RELEASE frmInvoice 释放对整个对象的引用。

完成　保存 REFERENCE.PRG 程序。

8.4.7　数据和对象的集成

在 Visual FoxPro 中的属性可以在设计或运行时进行更改，这些属性中有一些可以实现 Visual FoxPro 中的对象和数据的集成，如表 8.4 所示。

表8.4　对象类中和数据库集成的属性

对象类	数据属性
表格	RecordSource, ChildOrder, LinkMaster
其他控件	ControlSource
列表框和组合框	ControlSource, RowSource
表单和表单集	DataSession

8.5　对父类方法的引用

8.5.1　对子类的方法进行扩充

Visual FoxPro 中，子类继承了父类的方法和属性。多数情况下，子类不但继承了父类的功能，还需要在父类的基础上进行扩充。

通常，可以使用 Visual FoxPro 中的域操作符（::）来实现对父类方法的调用，它的语法是：

Parent.Class::Method（）

1. 用（::）引用父类方法

说明　本实例定义了两个表单类 FirstForm 和 SecondForm，第二个类 SecondForm 的 Click 首先执行第一个类 FirstForm 的 Click 的方法，然后执行自身的代码。

程序 8.56 用(::)引用父类方法

```
DEFINE CLASS FirstForm AS FORM                    &&定义第一个表单类 FirstForm
    PROCEDURE Click
        WAIT WINDOW "这是第一个表单的任务"
    ENDPROC
ENDDEFINE
DEFINE CLASS SecondForm AS FirstForm              &&定义第二个表单类 SecondForm
    PROCEDURE Click
        FirstForm::Click
        WAIT WINDOW "这是第二个表单的任务"
    ENDPROC
ENDDEFINE
```

同时也可以使用函数 DODEFAULT（）来实现对父类方法的引用。

2. 继承父类的方法

说明　在本例中，cmdCancel 按钮类是 cmdOk 的类的子类，它通过继承父类的方法，同时又添加了新的判断而实现了自身的功能。

开始　从“帮助菜单”选择“示例应用程序”。

下一步　在“帮助对话框”中选择“Solutions Samples”，然后按“Open”。

下一步　在“项目管理器”中选择“类”选项卡，然后选择“Buttons”类库，然后选择“cmdOk”类，按“修改”按钮。

下一步　在“类设计器”中，双点击“OK”按钮。

下一步　在“代码窗口”中，查看“Click”的代码如下。

程序 8.57　“Click”的代码

```
IF TYPE("THISFORM.PARENT") = 'O'                  &&释放表单或者表单集
    RELEASE THISFORMSET
ELSE
    RELEASE THISFORM
ENDIF
```

下一步　关闭代码窗口和类设计器。

下一步　在“Buttons”类库，选择“cmdCancel”类，按“修改”按钮。

下一步　在“显示”按钮，选择“属性”。

下一步　在“属性”窗口中，选择“其他”选项卡，选择“ParentClass”属性，它的属性值为“cmdOk”，即它的父类是“cmdOk”。

下一步　在“类设计器”中，双点击“cmdCancel”按钮。

下一步　在“代码窗口”中，在“Click”事件下加入如下代码。

程序 8.58 “cmdCancel”按钮的“Click”事件代码

```
IF USED( ) AND CURSORGETPROP("Buffering") != 1
   TABLEREVERT(.T.)
ENDIF
DODEFAULT( )
```

说明 在更改未写入数据表中之前，被写入表缓冲，因此用 TABLEREVERT(.T.)可以取消已经做的变动。然后用 DODEFAULT( )函数调用父类的方法释放表单。

完成 关闭代码窗口和类设计器。

8.5.2 修改属性的默认值

Visual FoxPro 中，当把对象加入表单时，可以修改对象在类的定义中没有被保护的属性。当属性被修改，默认值将被覆盖，以后在“类设计器”中对属性的默认值进行修改将不改变表单中属性的值。如果表单中的属性使用默认值，则对象的属性将随类在类设计器中的改变而改变。

8.6 小 结

本章主要介绍了面向对象的编程方法，它与原来主要对流程和程序模块的设计方法有所不同，这种编程方法主要是考虑对象的构造以及与对象有关的属性和方法的设计。对象为了实现自己的功能要具有自己的函数，另一方面也需要暴露在外面的接口，用来完成其它程序的调用。

本章主要介绍了：VFP 6.0 中的类和对象、用类设计器设计类、用编程的方法设计类、对类和对象的引用、对父类方法的引用。

思考与练习

1. 如何理解 Visual FoxPro 中的对象？使用面向对象的方法有哪些好处？
2. 什么是对象的“属性”和“方法”？基类和子类有何关系？
3. 容器类和非容器类有何区别？
4. 组件有什么作用，如何创建组件？
5. 用编程的方法设计类要注意哪些问题？
6. 为什么使用对象存储数据？这样有什么好处？
7. 如何调用对象的方法？
8. 如何实现对父类方法的调用？从中体会类带来的好处？

第9章　设计应用程序界面

对任何程序设计者来说，界面变的越来越重要。Visual FoxPro 6.0 最为常见的用于显示和编辑的界面便是表单。然而，对整个应用程序的导航设计仅仅有表单是不够的。对大多数用户而言，首先见到的是菜单，在菜单的导航支持下才进入一个个的表单或对话框，在对话框中，又可以通过各种控件来实现各种功能。

本章主要介绍如下内容：

（1）菜单系统的设计；

（2）为应用程序创建工具栏；

（3）使表单协同工作；

（4）使用控件。

9.1　菜单系统的设计

9.1.1　创建菜单系统的步骤

Visual FoxPro 提供了“菜单设计器”，通过它，用户可以方便地建立自己的菜单系统。在 Visual FoxPro 中，应用程序可以建立多个菜单，通过主菜单的导航完成对整个系统的控制。

要建立一个菜单系统通常包括以下的几个步骤：

开始 设计规划你的菜单系统：确定系统的各部分应用程序都需要哪些导航菜单，它们出现在界面的什么地方，每个菜单所具有的子菜单等内容。

下一步 创建菜单和子菜单：Visual FoxPro 提供了如图 9.1 所示的“菜单设计器”通过它，用户可以方便地建立菜单的标题、菜单项和子菜单。

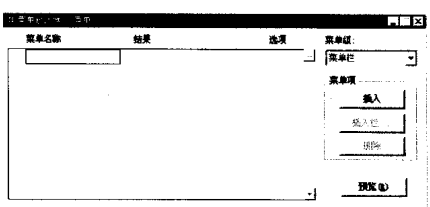


图9.1　菜单设计器

下一步 指定各部分菜单的任务：各部分的任务可以是显示一个表单，执行一个应用程序，也可以显示一个对话框。在适合的地方可以建立初始化的代码，也可以加入清理代码。

初始化代码在程序执行前声明常量、变量，打开必要的文件。清理代码可以加入用户的自定义过程，从而确定菜单的访问状态等内容。

下一步 浏览菜单系统：通过浏览菜单找出当前设计的不足，及时修改。

下一步 生成菜单程序：如果使用“菜单设计器”将自动生成菜单程序，也可以通过用户自定义方式，通过应用程序建立菜单程序。Visual FoxPro 将用一个扩展名为“.MNX”的文件中保存菜单系统，而另一个提示用户输入文件名后缀为“.MPR”的文件用来保存生成的菜单程序。

最后 运行菜单程序，进行测试。

9.1.2 菜单系统的规划

在应用系统中，菜单往往提供了对整个系统的导航支持，菜单设计是否符合用户的操作习惯，是否清晰明了，是否符合方便、快捷的原则都将影响系统的总体印象。因此菜单的设计要十分慎重，遵循一定的准则。

通常菜单系统在设计时，要注意以下几方面的内容：

(1) 按照用户执行任务的顺序组织菜单结构，不要按照程序的执行层次组织菜单。最终使用菜单的是用户，因此菜单设计时要始终考虑用户的操作顺序和操作习惯，经常和用户进行交流，搞清用户在执行任务时的思路、完成任务的过程和方法，从而按照用户的要求去设计菜单。

(2) 菜单的标题要具有实际的应用意义，并要符合各个应用领域的专业所属命名习惯。由于应用系统所属的行业不同，菜单系统在设计时要充分考虑到大众习惯和专业习惯，根据不同的需要设计出含义明确的菜单标题。

(3) 菜单项的布置要有一定的顺序。菜单中的菜单项的顺序可以按照菜单的执行频率、任务的执行顺序或按照字母顺序来组织。

(4) 菜单项按照不同的逻辑范围，要用分隔线进行分组。通过将菜单进行逻辑分组，可以方便用户对菜单进行快速查找，将执行某一方面任务的菜单放在一起也符合用户的习惯。

(5) 在一个屏幕中的菜单项不要过于拥挤，菜单项应在一个屏幕内。对菜单进行逻辑分类，通过给菜单建立子菜单使得系统菜单的布置在整体上简捷、明了。

(6) 选择合适的快捷键。快捷键的选择要符合大众习惯，例如要用ALT+F作为“文件”的快捷键而不要使用ALT+X。

(7) 菜单项的描述要符合日常习惯和所属的专业习惯。在进行菜单项功能的描述时，应尽量避免使用计算机术语，而尽量使用日常和应用程序所属专业的语汇。对于功能动作菜单应尽量选择合适的动词不要使用静态的名词。

(8) 混合使用大小写字母。菜单的快捷字母使用大写，强调的部分使用大写，而其他字母要使用小写。

技巧 可以通过执行实例程序 Tasmanian Traders 来体会菜单系统的设计和规划准则，从“帮助”菜单选择“示例应用程序”、然后选择“Run” Tasmanian Traders.

9.1.3　创建菜单系统

对于一个完整的菜单系统通常包括菜单、菜单项和子菜单等部分，各部分的介绍见图9.2所示。

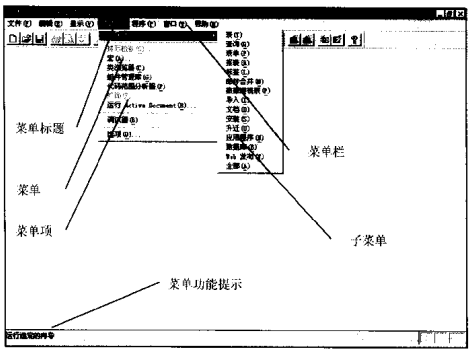


图9.2　完整的菜单系统

创建菜单系统的操作步骤如下：

说　明　如果创建菜单系统可以从 Visual FoxPro 系统出发，也可以直接开发自己的菜单系统。在本例中我们将学习如何建立“快速菜单”，如何在菜单中添加菜单项，如何添加子菜单。

开始　从“文件”菜单中选择“新建”。

下一步　新建一个项目。

下一步　在“项目管理器”中选择“其他”选项卡。

下一步　在“其他”选项卡中“菜单”。

下一步　在“其他”选项卡中“菜单”。

下一步　选择“新建”。

下一步　在“新菜单”对话框中选择“菜单”。

下一步　从“菜单”菜单中选择“快速菜单”，此时“菜单设计器”中出现已经定制好的菜单系统，见图9.3所示。

下一步　选择“窗口”菜单。

下一步　在“菜单项”按钮组框中单击“插入”按钮。

下一步　在“菜单名称”框中输入“客户”。

下一步　在“结果”框中选择“子菜单”。

下一步　拖动“客户”菜单前面的移动按钮，将它移动到“窗口”和“帮助”菜单之

间。

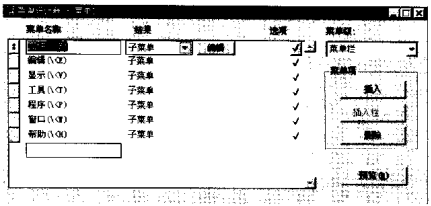


图9.3　生成的快速菜单

下一步 单击“了菜单”右侧的创建按钮 创建 。

下一步 在新出现的设计窗口中，输入必要的菜单项。

下一步 从“菜单级”下拉框中选择“菜单栏”返回第一级菜单。

说明 对于每一级菜单，如果菜单项的“结果”选择的是“子菜单”，则都可以按右侧的“创建”按钮来创建下一级子菜单，如果子菜单已经创建，则它的提示为“编辑”，可以进入下一级菜单的设计器进行设计修改。

下一步 在“菜单”菜单中选择“生成”。

下一步 在出现的对话框中选择“是”。

下一步 选择输出文件的路径，然后按“产生”，如图 9.4 所示。

图9.4　生成菜单

下一步 单击“预览”按钮，此时 Visual FoxPro 的菜单变为你所设计的当前菜单。

下一步 在“预览”对话框中按“确定”。

完成 关闭“菜单设计器”。

9.1.4　创建快捷菜单

Windows 程序中，在对象上单击鼠标右键便会出现关于这个对象的菜单操作。这种菜单给用户带来了极大的方便。在 Visual FoxPro 中同样可以建立这样的快捷菜单。

创建快捷菜单的操作步骤如下：

开始 在“项目管理器”中选择“其他”选项卡。

下一步 在“其他”选项卡中选择“菜单”，按“新建”。

下一步 在“新菜单”对话框中选择“快捷菜单”。

下一步 在“菜单设计器”中，输入第一个菜单项，“菜单名称”为“日期...”。

下一步 在“结果”框中选择“过程”。

下一步 按“过程”右边的 创建 按钮。

下一步 在“代码”窗口中输入如图 9.5 所示的过程代码。

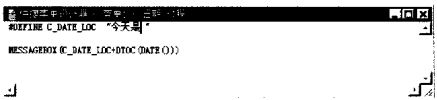


图9.5　“日期”的过程代码

下一步 关闭“日期过程”代码窗口。

下一步 在“日期”的前面加上“\<D”。

下一步 在“菜单设计器”中，输入第二个菜单项，“菜单名称”为“\<X 退出”。

下一步 在“结果”框中选择“过程”。

下一步 按“过程”右边的 创建 按钮。

下一步 在“代码”窗口中输入下面的过程代码。

```
MESSAGEBOX("退出请按“退出”按钮")
```

下一步 关闭代码窗口。

下一步 从“菜单”菜单中选择“生成”，将生成的菜单文件的名称设定为“快捷菜单”。

下一步 关闭“菜单设计器”。

下一步 从“项目管理器”中选择“文档”选项卡。

下一步 选择“表单”，单击“新建”按钮，然后按“新建表单”。

下一步 在“表单设计器”中添加一个“编辑框”控件和一个“命令按钮”控件。

下一步 将“命令按钮”控件的“Caption”属性设置为“退出”，如图 9.6 所示。

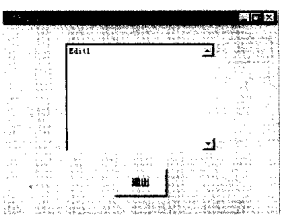


图9.6　创建快捷菜单响应表单

下一步 从“显示”菜单中选择“代码”。

下一步 在代码窗口中，选择“command1”的“Click”事件，加入如下的代码。

```
RELEASE THISFORM
```

下一步 在代码窗口中，选择“Edit1”的“RightClick”事件，加入如下的代码。

```
DO e:\practice\menus\快捷菜单.mpr          &&执行快捷菜单
```

下一步 关闭代码窗口。

下一步 从“表单”菜单中选择“执行表单”。

下一步 在表单中的编辑框中单击鼠标右键，出现快捷菜单，如图 9.7 所示。

下一步 在快捷菜单中选择“日期”将出现显示日期的对话框。

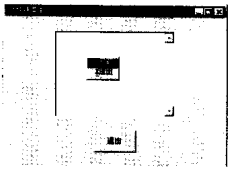


图9.7　执行具有快捷键的表单

关闭显示日期的对话框。

在表单中的编辑框中单击鼠标右键，出现快捷菜单。

在快捷菜单中选择“退出”将出现显示退出操作的对话框。

关闭显示退出操作的对话框。

按“退出”按钮退出表单。

9.1.5　对菜单项进行分组

根据功能的需要，有时需要将菜单项进行分组。

对菜单进行分组的操作步骤如下：

打开上一小节建立快捷菜单的项目。

在“项目管理器”中选择“其他”选项卡。

在“其他”选项卡中选择“菜单”，选择“快捷菜单”，按“修改”按钮。

在“菜单设计器”中选择“退出”菜单项，然后按“插入”。

在“菜单设计器”中的“菜单名称”栏输入“\-”。

从“菜单”菜单选择“生成”。

在“项目管理器”选择“快捷菜单”表单，然后按“运行”按钮。

在“快捷菜单”表单中单击鼠标右键，出现带分组分隔符的快捷菜单，如图 9.8 所示。

图9.8　加分组分隔符的菜单

关闭“快捷菜单”表单。

说明　如果对菜单进行分组，只需在需要分组的地方插入一个菜单项，将这个菜单项的菜单名称设置为“\-”即可。

9.1.6　指定菜单的快速访问键

菜单可以具有快速访问键，从而方便用户的操作。

对菜单进行分组的操作步骤如下：

打开已经建立的“快捷菜单”的项目。

在“项目管理器”中选择“其他”选项卡。

在“其他”选项卡中选择“菜单”，选择“快捷菜单”，按“修改”按钮。

在“菜单设计器”中选择“日期”菜单项，然后将它修改为“日期（\<D）”。

在“菜单设计器”中选择“退出”菜单项，然后将它修改为“退出（\<X）”。

从“菜单”菜单选择“生成”。

在“项目管理器”选择“快捷菜单”表单，然后按“运行”按钮。

此时出现如图 9.9 所示的菜单。

在键盘上按“D”键。

关闭出现的消息窗口。

关闭“快捷菜单”表单。

图9.9　指定快速访问键的菜单

建立菜单的快速访问键，在快速访问键的前面加上“\<”。

9.1.7　创建菜单的键盘快捷键

另一种快速访问菜单的方法是建立菜单的键盘快捷键，从而提供了一种不需要访问菜单便执行菜单命令的方法。通常快捷键是CTRL和一个键盘字母的组合。

创建键盘快捷键的操作步骤如下：

开始 打开已经建立的“快捷菜单”的项目。

下一步 在“项目管理器”中选择“其他”选项卡。

下一步 在“其他”选项卡中选择“菜单”，选择“快捷菜单”，按“修改”按钮。

下一步 在“菜单设计器”中选择“日期”菜单项，然后单击“选项”栏中的按钮。

下一步 出现“提示选项”对话框后，按下“CTRL+D”键，此时“键标签”框中和“键说明”框中自动出现快捷键的对应文本，“键说明”框中可以输入菜单项旁边出现的提示文本，通常“键标签”和“键说明”的内容是一致的，如图9.10所示。

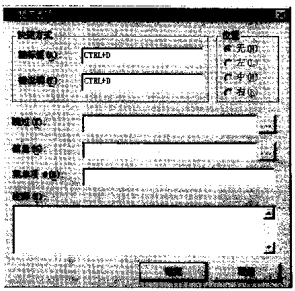


图9.10　提示选项对话框

下一步 按“确定”。

下一步 在“菜单设计器”中选择“退出”菜单项，然后单击“选项”栏中的按钮。

下一步 出现“提示选项”对话框后，按下“CTRL+X”键。

下一步 按“确定”。

下一步 从“菜单”菜单选择“生成”。

下一步 在“项目管理器”选择“快捷菜单”表单，然后按“运行”按钮。

下一步 此时出现如图9.11所示的菜单。

下一步 单击表单取消快捷菜单。

下一步 按“CTRL+X”键，然后关闭出现的消息对话框。

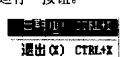


图9.11　具有键盘快捷键的菜单

完成 退出“快捷菜单”表单。

9.1.8　设定菜单项的启用状态

每个菜单选项可以根据一定的逻辑条件设定它是处于启用状态还是废止状态。

设定菜单项的状态的操作步骤如下：

开始 打开已经建立的“快捷菜单”的项目。

下一步 在“项目管理器”中选择“其他”选项卡。

下一步 在“其他”选项卡中选择“菜单”，选择“快捷菜单”，按“修改”按钮。

下一步 在“菜单设计器”中选择“日期”菜单项，然后单击“选项”栏中的√按钮。

下一步 出现“提示选项”对话框后，选择“跳过”框后的…。

下一步 在“表达式生成器”中构造逻辑表达式，它的值决定了菜单项的启用状态。

说明 当逻辑表达式的值为“真”（.T.）时，则菜单或菜单项处于启用状态。当表达式的值为“假”时，则菜单或菜单项处于废止状态。

完成 关闭“表达式生成器”，关闭“菜单设计器”。

9.1.9 标记菜单项的使用状态

标志菜单的使用状态，可以提示用户当前使用的菜单是哪一个。

标记菜单的使用状态的操作步骤如下：

开始 从“帮助”菜单中选择“实例应用程序”，并运行“Solutions Sample”。

下一步 在对话框中打开“Menus”，选择“Disable or display a check beside a menu item”，然后按“See Code”。

下一步 在出现的“表单设计器”中选择“item1”，并双击它。

下一步 在代码窗口中，对象为“chk1”，过程为“ItemactiveChange”，代码窗口的代码如下。

程序 9.1 对象“chk1”的“ItemactiveChange”代码

```
SET MARK OF BAR 1 OF checkitems TO THIS.Value          &&标记菜单项的使用状态
THISFORM.imgCheck1.visible = THIS.VALUE                &&在表单上显示标记
```

下一步 在代码窗口中，对象为“chkEnabled1”，过程为“ItemactiveChange”，代码窗口的代码如下。

说明 Visual FoxPro 中，应用程序在运行时，可以使用 SET SKIP OF 命令设置菜单或菜单项的启用状态，使用 SET MARK OF 命令可以在菜单项旁边设置复选标记。

程序 9.2 对象“chkEnabled1”的“ItemactiveChange”代码

```
SET SKIP OF BAR 1 OF checkitems !THIS.Value&&标记菜单项的启用状态
THISFORM.imgItem1.visible = !THIS.value&&在表单上当前启用状态下菜单的显示形式
```

下一步 关闭代码窗口和表单设计器，然后选择“Run Sample”。

下一步 对出现的表单进行操作，看看设置菜单项的启用状态和设置菜单项的使用状态对菜单项的影响。

完成 关闭所有打开的表单。

9.1.10 设置对菜单项的响应

当用户选择一个菜单的时候，将执行菜单所对应的任务，它可以是显示一个表单或者

激活另一个菜单系统。这需要执行菜单对应的命令、调用相关的过程等操作。

设置对菜单项的响应的操作步骤如下：

开始 打开已经建立的“快捷菜单”的项目。

下一步 在“项目管理器”中选择“其他”选项卡。

下一步 在“其他”选项卡中选择“菜单”，选择“快捷菜单”，按“修改”按钮。

下一步 在“菜单设计器”中增加一个新的菜单项，“菜单名称”为“定单（\<O）...”，“结果”选择“子菜单”，用“选项”设置快捷键为“CTRL+O”。

下一步 在“菜单设计器”中增加一个新的菜单项，“菜单名称”为“显示客户（\<K）...”，“结果”选择“命令”，用“选项”设置快捷键为“CTRL+K”。

下一步 在“菜单设计器”中的“显示客户”菜单“命令”后的空白框中输入命令“DO FORM e:\practice\forms\客户”，如图9.12所示。

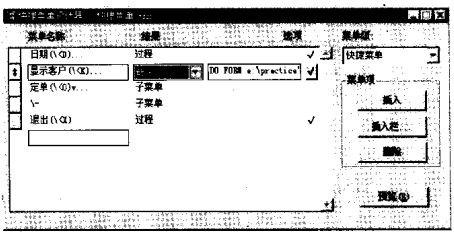


图9.12　使用命令完成菜单项的任务

说明 Visual FoxPro 中，可以使用命令完成菜单项的响应，这个命令可以是 Visual FoxPro 中的任何有效的命令，当使用命令显示表单、表单集或对话框时，请在菜单选项后加上“...”从而表明菜单的响应是可以进一步输入信息，表明操作是可以挽救的。

下一步 在“菜单设计器”中选择“日期”菜单项，然后单击 编辑 按钮。

下一步 在代码窗口中查看代码。

说明 对菜单项的响应可以为菜单或菜单项建立一个过程，Visual FoxPro 自动为每个过程加上 PROCEDURE 关键字，用户不必在代码窗口中加入这个命令。

下一步 在“菜单设计器”中选择“定单”菜单项，然后单击 编辑 按钮。

下一步 在新的“菜单设计器”中输入两个菜单项，如图9.13所示。

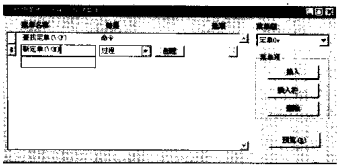


图9.13　定单菜单的子菜单

下一步 在“菜单级”框中选择“定单”。

下一步 从“显示”菜单中选择“菜单选项”，出现“菜单选项”对话框，如图 9.14 所示。

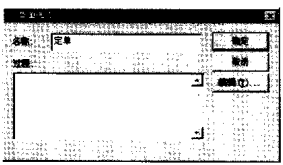


图9.14　菜单选项对话框

下一步 在“菜单选项”对话框中的“过程”框中编写后调用过程，或者单击“编辑”然后按“确定”按钮，在出现的过程窗口中编辑过程代码。

说明 为含有子菜单的菜单项指定过程，要指定菜单的级别。如果某个菜单不包含菜单项，而只执行命令或过程，在菜单名后加上“！”表明菜单只是执行任务。

下一步 保存菜单的设置，返回“项目管理器”。

下一步 在“项目管理器”中创建如图 9.15 所示的“客户”表单。

下一步 在“项目管理器”中选择“快捷菜单”表单，然后单击“运行”按钮。

下一步 操纵显示的表单，通过执行快捷键的命令体会菜单的响应。

完成 关闭所有打开的表单。

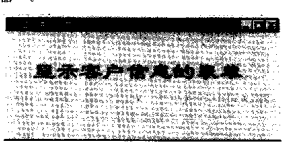


图9.15　客户表单

9.1.11　增加初始化和清理代码

通常通过向菜单系统添加初始化的代码能实现如下的功能：创建环境、定义变量、打开文件以及保存和恢复菜单系统的操作。同样为了减少菜单系统的大小，可以使用清理代码，从而在菜单初始化时启用或废止菜单中的菜单项。清理代码放在初始化代码和菜单的

定义之后，而位于过程代码之前。

增加初始化和清理代码的操作步骤如下：

开始打开已经建立的“快捷菜单”的项目。

下一步在“项目管理器”中选择“其他”选项卡。

下一步在“其他”选项卡中选择“菜单”，选择“快捷菜单”，按“修改”按钮。

下一步从“显示”菜单中选择“常规选项”，出现如图 9.16 所示的“常规选项”对话框。

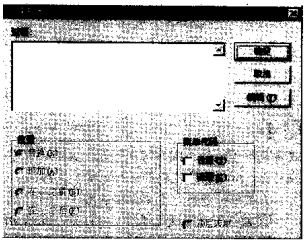


图9.16　常规选项对话框

下一步从“菜单代码”区域选择“设置”复选框，出现一个独立的代码窗口，按“确定”，在代码窗口中输入初始化代码。

下一步从“菜单代码”区域选择“设置”复选框，出现一个独立的代码窗口，按“确定”，在代码窗口中输入初始化代码。

下一步关闭“设置”代码窗口。

下一步从“显示”菜单中选择“常规选项”。

下一步从“菜单代码”区域选择“清理”复选框，出现一个独立的清理代码窗口，按“确定”，在清理代码窗口中输入清理代码。

说明如果设计的应用程序的主菜单，则在清理代码中需要加入 READ EVENTS 命令，并且给退出菜单系统的命令指定 CLEAR EVENTS。

重点关闭“清理”代码窗口。

9.1.12　在程序运行时控制菜单

Visual FoxPro 中，菜单在用户的界面上可以使用一个名称，在生成的菜单程序（.MPR 文件）中可以使用另一个名称或编号。同样，每个菜单项也都有一个名称和编号。

在生成的菜单程序中，可以使用“填充名称”，引用一个菜单的标题，如图 9.17 所示。

图9.17　在生成菜单程序中使用“填充名称”

在生成的菜单程序中，菜单项的引用可以使用菜单项编号，生成的应用程序是否使用编号是可选的，如果没有指定编号，则由 Visual FoxPro 自动设置，见图 9.18 所示。

图9.18　在生成菜单程序中使用“菜单项#”

注意　在应用程序代码中，不要使用 Visual FoxPro 生成的菜单程序中提供的主菜单名称或者菜单项编号，因为每次重新生成菜单程序时，它们的值都可能更改。如果引用了编号或名称，应用程序代码的运行可能会失败。

为菜单标题指定主菜单名的操作步骤如下：

开始　在“项目管理器”中打开已经建立的“菜单”。

下一步　在“菜单设计器”中选择需要设定的菜单项。

注意　此时“结果”框中必须选择“命令”，“子菜单”或“过程”，而不应选择“填充名称”。

下一步　选择“选项”按钮，进入“提示选项”对话框。

下一步　在“提示选项”对话框中的“主菜单名”框输入你确定的主菜单名称。

下一步　按“确定”返回“菜单设计器”。

说明　设置菜单项编号的步骤和设置主菜单名的步骤相同，只是在“结果”框中不能选择“菜单项#”，此外如果使用“快速菜单”建立的菜单或菜单项的名称不要随便改动它们的主菜单名或者菜单项名称。

完成　关闭“菜单设计器”。

9.1.13　菜单系统的测试和调试

在设计菜单时，可以对菜单系统进行预览。如果要预览菜单，只需要在“菜单设计器中按“预览”按钮，便可以出现菜单的预览对话框，见图 9.19。

此时系统菜单变为当前的设计菜单，当你选择菜单中的任何一项时，预览窗口中将会显示当前菜单选项的属性。

如果要测试菜单系统，可以按如下步骤进行：

开始　从“菜单”菜单中选择“生成”，如果用户对菜单作了修改，将提示用户是否保存它。

下一步　在随后的对话框中选择“是”，然后在“另存为”对话框中选择或输入菜单文件（.MNX），按“保存”，然后在“生成菜单”对话框中输入生成菜单的文件名（.MPR）。

下一步　按“产生”，生成一个菜单程序文件。

下一步　在“程序”菜单选择“执行”，运行刚刚生成的菜单程序，进行测试。

完成　如果菜单程序运行不正确，则修改调试应用程序。

注意　如果你对生成的菜单程序（.MPR）进行了修改，而后在“菜单设计器”中又重新生成菜单程序，则所有的修改将全部丢失。

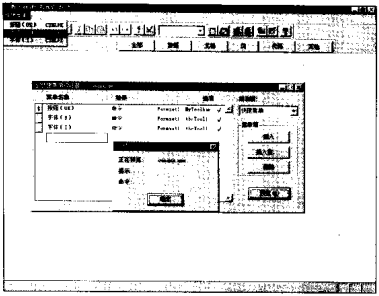


图9.19　预览菜单系统

9.1.14　在状态栏中显示提示信息

当用户选择一个菜单或者菜单项时，在状态栏中可以显示当前所选择的菜单项的提示信息，从而帮助用户对菜单项进行选择。

为菜单项设置提示信息的操作步骤如下：

开始>在“项目管理器”中打开已经建立的“菜单”。

下一步>在“菜单设计器”中选择需要设定提示信息的菜单项。

下一步>选择“选项”按钮，进入“提示选项”对话框。

下一步>在“提示选项”对话框中的“信息”框中输入需要显示的提示信息，单击…可以进入“表达式生成器”构造提示信息的表达式。

注意　提示信息应该用双引号将字符串括起来。

下一步>按“确定”返回“菜单设计器”。

完成>关闭“菜单设计器”。

9.1.15　设置菜单标题出现的位置

应用程序中可以设置菜单标题出现的位置。

1. 设置菜单标题的相对位置

开始>在“项目管理器”中打开已经建立的“菜单”。

下一步>在“菜单设计器”中选择需要设定提示信息的菜单项。

下一步>选择“显示”菜单，选择“常规选项”。

下一步>在“常规选项”对话框的“位置”选项组选择“替换、追加、在...之前、在...之后”，见图9.20所示。

完成 关闭“常规选项”对话框，设定完成后，将重新排列标题的位置。

图9.20　设置菜单标题的位置

此外，你也可以当应用程序编辑对象时，显示并设置菜单标题的位置。如果你的应用程序包含了一个对象并激活了它，你所加入的菜单标题将不显示出来，除非你想让它们显示出来。

2. 对对象可视化编辑时设置菜单标题的位置

开始 在“项目管理器”中打开已经建立的“菜单”。

下一步 在“菜单设计器”中选择需要设定提示信息的菜单项。

下一步 选择“显示”菜单，选择“常规选项”。

下一步 在“常规选项”对话框的“位置”选项组选择“无、左、中、右”，如图 9.21 所示。

图9.21　设置菜单标题的位置

说明 无、左、中、右的设置对菜单标题的影响如下：

无：不把菜单标题放在菜单上。

左：将菜单标题放在菜单标题组的最左侧。

中：将菜单标题放在菜单标题组的中间

右：将菜单标题放在菜单标题组的最右侧

完成 关闭“常规选项”对话框。

9.1.16　保存恢复菜单

应用程序中可以使用 PUSH MENU 将当前菜单存入堆栈中，可以使用 POP MENU 命令将堆栈中保存的菜单还原。当你想短暂的移去当前菜单，用替换菜单执行完任务后再恢复原来的菜单，这种入栈、出栈的方法十分有用。

注意 菜单占用的内存较大，可以使用 SYS(1016)系统函数检查可用的内存数量，在菜单进入堆栈前调用 SYS(1016)函数，然后在菜单进入堆栈后调用 SYS(1016)，两者的差为当前菜单占用的内存。

9.1.17　给菜单系统加入默认过程

可以创建一个全局过程，用于对整个菜单系统的响应。当一个菜单项没有指定的执行过程时遍运行此过程。

设置菜单默认过程的操作步骤如下：

开始 在“项目管理器”中打开已经建立的“菜单”。

下一步 选择“显示”菜单，选择“常规选项”。

下一步 在“常规选项”对话框中，在“过程”框输入或调用一个过程。

说明 也可以按 [illegible] 按钮，然后按“确定”按钮，进入过程代码窗口输入过程。

完成 关闭“常规选项”对话框。

9.2　为应用程序创建工具栏

9.2.1　创建工具栏

与菜单相结合，在应用程序中添加用户自定义的工具栏，用来执行经常执行的任务，从而简化用户的操作。

在 Visual FoxPro 应用程序中，如果使用工具栏，必须要为它定义一个工具栏类，从而建立你所需的工具栏。

创建工具栏类的操作步骤如下：

开始　从“项目管理器”中选择“类”选项卡，选择“新建”。

下一步　在“类名”框中输入“MyToolbar”。

下一步　在“派生于”框中选择“Toolbar”。

下一步　在“存储于”框中输入路径名，如“e:\practice\libs\Toolbarv.vcx”，如图 9.22 所示。

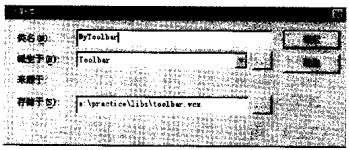


图9.22　创建工具栏类

完成　按“确定”进入“类设计器”。

说明　创建工具栏类也可以从“文件”菜单中选择“新建”或者使用 CREATE CLASS 命令，还可以用 DEFINE CLASS 用程序建立工具栏类。

9.2.2　向工具栏中添加对象

工具栏类定义好以后，可以向工具栏中添加任何 Visual FoxPro 支持的对象。在“类设计器”中可以对加入的对象调整大小、移动位置、删除对象或加入分隔符。

向工具栏加入对象的操作步骤如下：

开始　从“项目管理器”中选择“类”选项卡，选择“MyToolbar”类，按“修改”。

下一步　从“表单控件”工具栏中选择“计时器”控件和“分隔符”控件。

下一步　向工具栏中加入“标签”控件和“命令按钮”。

下一步　选择“类设计器”中“MyToolar”的“标签”控件，按 DELETE 键将它删除。

下一步　在“属性窗口”中设置工具栏类“MyToolar”的属性。

下一步　保存工具栏类“MyToolar”。

完成　关闭“类设计器”。

9.2.3　用表单设计器向表单集中添加工具栏

工具栏类定义好以后，可以用这个类创建一个工具栏。在“表单设计器”中，可以向表单集中添加工具栏。

向表单集中添加工具栏

开始 新建一个“表单集工具栏”项目。

下一步 在“项目管理器”中选择“类”选项卡，加入上一节创建的工具栏类库“Toolbar”。

下一步 在“项目管理器”中选择“表单”，然后选择“新建”。

下一步 选择“新建表单”，进入“表单设计器”，在“表单设计器”中设置表单的“Caption”属性为“工具栏表单”，“Name”属性设置为“frmForm1”。

下一步 从“表单”菜单中选择“创建表单集”，建立一个新的表单集。

下一步 从“表单控件”工具栏中选择“查看类”按钮。

下一步 在出现的对话框中选择类库文件“Toolbar.vcx”。

下一步 从“表单控件”工具栏中选择“MyToolbar”按钮。

下一步 鼠标单击表单设计器，加入“MyToolbar”工具栏，并把它拖到合适的位置。

说明 向表单集中加入工具栏通常可以有三种方法：

(1) 直接在“项目管理器”中将工具栏类拖拽到“表单设计器”。

(2) 注册工具栏类从表单控件工具栏中向表单集加入工具栏。

(3) 在表单集的 Init 事件中用 AddObject 方法加入工具栏。

完成 关闭“表单设计器”。

9.2.4　用代码向表单集中添加工具栏

除了使用“表单设计器”向表单集中添加工具栏外，也可以使用代码向表单集中添加工具栏。

用代码向表单集中添加工具栏通常在 Init 事件中使用 SET PROCEDURE TO 命令。

用代码向表单集中添加工具栏的程序如下：

说明 本例在表单集的 Init 事件中加入代码，从而向表单集中加入工具栏。

程序 9.3　用代码向表单集中添加工具栏

```
SET CLASSLIB TO Toolbar
THIS.AddObject("tbrMyTool","MyToolbar")
tbrMyTool.Show
```

9.2.5　创建用户的工具栏

通过代码可以定义用户的工具栏。下面通过一个例子建立一个具有两个按钮的工具栏，如图 9.23 所示。

图9.23　工具栏按钮

创建用户自定义工具栏的操作步骤如下：

说明 本例通过选择两个工具栏按钮，改变表单集中的表单 frmForm1 的属性。

开始 打开“表单集工具栏”项目。

下一步 在“项目管理器”中选择“代码”。

下一步 在“代码”选项卡中选择“程序”，按“新建”。

下一步 在代码窗口中输入如下的代码。

程序 9.4　定义 myToolBar 工具栏类

```
*-------------------------------------------------------------------------------*
*  定义 myToolBar 工具栏类
*-------------------------------------------------------------------------------*
DEFINE CLASS myToolBar AS TOOLBAR        &&定义工具栏，包含两个命令按钮和一个分隔符
ADD OBJECT cmdBold AS COMMANDBUTTON
ADD OBJECT scp1 AS SEPARATOR
ADD OBJECT cmdItalic AS COMMANDBUTTON
Left = 1                                                    &&设置工具栏属性
Top  = 1
Width = 100
Height= 40
Caption = "Form 属性"
cmdBold.Caption = "B"                                       &&设置控件的属性
cmdBold.Height = 20
cmdBold.Width = 40
cmdItalic.Caption = "I"
cmdItalic.Height = 20
cmdItalic.Width = 40
cmdItalic.FontBold = .F.                                    && FontBold 的默认属性为.T.
PROCEDURE Activate                                  &&工具栏被激活时设置两个命令按钮的字体属性
    THIS.cmdBold.FontBold = ;
        THISFORMSET.frmForm1.FontBold
    THIS.cmdItalic.FontItalic = ;
        THISFORMSET.frmForm1.FontItalic
ENDPROC
PROCEDURE cmdBold.CLICK                                     &&用户单击 cmdBold 时
    THISFORMSET.frmForm1.FontBold = ;                  &&将 frmForm1 的 FontBold 属性求反
      !THISFORMSET.frmForm1.FontBold
    THIS.FontBold = ;                                  &&将 cmdBold 的属性和表单集的一致
      THISFORMSET.frmForm1.FontBold
ENDPROC
PROCEDURE cmdItalic.CLICK&&用户单击 cmdItalic 时
    THISFORMSET.frmForm1.FontItalic = ;                &&将 frmForm1 的 FontItalic 属性求反
      !THISFORMSET.frmForm1.FontItalic
    THIS.FontItalic = ;                                &&将 cmdItalic 的属性和表单集的一致
        THISFORMSET.frmForm1.FontItalic
    ENDPROC
```

```
ENDDEFINE                                    &&类定义结束
```

下一步 保存文件到 e:\practice\progs\toolbar.prg 中。

下一步 在“项目管理器”中选择“文档”选项卡，选择“工具栏表单”然后按“修改”按钮。

下一步 在“表单设计器”中双点击表单，进入代码窗口。

下一步 在表单集“frsFormset1”的“Init”事件中加入如下的代码。

程序 9.5　“Init”事件代码

```
*-------------------------------------------*
SET PATH TO e:\practice\progs
SET PROCEDURE TO   toolbar ADDITIVE
THIS.AddObject("tbrTool1","mytoolbar")&&  在当前表单集中添加工具栏
THIS.tbrTool1.Show
```

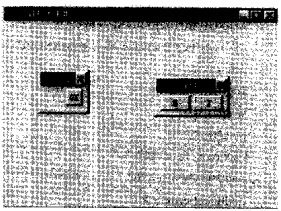


图9.24　带工具栏的表单

下一步 保存代码。

下一步 选择“表单”菜单，然后选择“执行表单”，将出现如图 9.24 所示的带有两个工具栏的表单。

下一步 在“Form 属性”工具栏中选择 B 按钮，然后选择 I 按钮，看字体发生的变化。

完成 关闭表单。

说明　自定义工具栏在设计时，可以设置它的属性，可以使用方法和事件代码来定义工具栏的操作。例如设置“Movable”属性允许工具栏可以移动，使用“Dock”方法停放或者移出工具栏。

9.2.6　让工具栏按钮执行相应操作

应用程序中，工具栏的每一个按钮都有一定的功能。因此工具栏定义好以后，必须定义工具栏和相关对象的操作。

定义工具栏按钮的操作步骤如下：

开始 打开“表单集工具栏”项目。

下一步 在“项目管理器”中选择“类”选项卡，选择类库“Toolbar”。

下一步 选择“MyToolbar”类，按“修改”。

下一步 在“类设计器”中选择“OK”命令按钮。

下一步 进入属性窗口，选择“方法程序”选项卡。

下一步 选择“Click Event”方法双击它。

下一步 在“代码窗口”中输入如下代码。

```
RELEASE THISFORMSET
```

下一步 关闭“代码窗口”和“类设计器”。

下一步 运行表单“工具栏表单”。

完成 在“工具栏表单”的“ToolBar”工具栏中单击“OK”。

9.2.7　创建和工具栏协调工作的菜单

工具栏的按钮通常都是经常使用的菜单操作。在实际应用中，工具栏和响应的菜单项的命令必须是同步的，用户创建的菜单的菜单项和对应的工具栏按钮执行相同的工作，并且启用状态、使用状态必须保持一致。

为了使菜单和工具栏协调工作，要遵循如下的步骤：

开始 通过定义一个工具栏类创建一个工具栏，添加适当的命令按钮，在命令按钮的“Click”事件中加入相应要执行的方法。

下一步 创建相应的菜单。

完成 将工具栏和菜单加到表单集中。

创建和工具栏协同工作的菜单的操作步骤如下：

说明 本实例建立一个快捷菜单，它和工具栏一同加入表单集，并执行相关的任务。如何向表单集中添加工具栏，在9.2.3节中已经作了说明。

开始 打开“表单集工具栏”项目。

下一步 在“项目管理器”中选择“其他”选项卡，选择“新建”。

下一步 选择“快捷菜单”进入“菜单设计器”。

下一步 在“菜单设计器”中的“菜单名称”新建三个菜单项“按钮（OK）、字体（B）、字体（I）”，“结果”都选择“命令”。

下一步 在“按钮（OK）”的命令框输入“Formset1. MyToolbar.command1.Click”。

下一步 在“字体（B）”的命令框输入“Formset1. tbrTool1. cmdBold.Click”。

下一步 在“字体（I）”的命令框输入“Formset1. tbrTool1. cmdItalic.Click”。

下一步 单击“选项”按钮进入“提示选项”对话框设置菜单的启用状态和快捷键等信息。

下一步 生成菜单程序名为“CoMenu.mpr”。

下一步 修改“工具栏表单”，加入“frmForm1”的“RightClick”事件的代码如下。

```
DO e:\practice\menus\comenu.mpr
```

说明 如果向表单集中添加的是一个菜单，通常需要如下步骤：

(1) 在表单集的“Load”事件加入类似下面的代码：

```
PUSH MENU _MSYSMENU        &&保存现有的菜单
DO menuname.mpr            &&运行所需的菜单
```

(2) 在表单集的“UnLoad”事件加入类似下面的代码：

```
POP MENU _MSYSMENU  &&恢复原始菜单
```

下一步 运行表单“工具栏表单”，右击鼠标键，表单中出现快捷菜单，如图9.25所示。

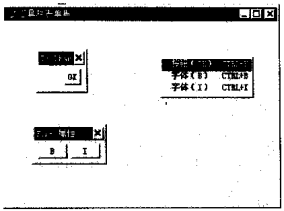


图9.25 与工具栏协调工作的表单

下一步 执行快捷菜单。

完成 关闭所有表单。

9.3 使表单协同工作

9.3.1 应用程序中的表单和表单集

在 5.2 节中已经介绍了如何应用表单来显示编辑表中的数据。表单的功能不仅如此，它可以包含大量的对象，通过对这些对象的事件进行处理，调用相应的方法，用户在应用程序中可以建立一系列相关的表单，从而完成一个或多个完整的任务。

在 Visual FoxPro 中，还可以把一系列相关内容加入表单集，从而同时显示隐藏在同一个表单集中的表单，并可以可视化的排列多个表单的位置。在同一表单集中的表单只有一个数据环境，从而可以同时控制在多个表单中的记录指针。当在一个表单中更改了父表的指针时，在与之相关的同子表相连的表单中的数据将自动显示更新。

9.3.2 单文档和多文档的用户界面

Visual FoxPro 中有两种应用程序界面：多文档（MDI）和单文档（SDI）。

多文档界面（MDI）：包含一个单一的主窗口，其他的应用窗口在主窗口内部或者浮动于主窗口之上。Visual FoxPro本身便是一个MDI的窗口，包括了命令窗口、代码窗口、属性窗口等。

单文档界面（SDI）：包含一个或多个相互独立的窗口，每一个窗口单独的显示。

通常包含一个窗口的应用为 SDI 形式，但有时还需要将 SDI 和 MDI 混合使用。例如 Visual FoxPro 中的 Debug 窗口是一个 SDI 却包含了它自己的 MDI 应用。

为了对两种界面提供支持，Visual FoxPro 的表单具有不同的类型，它们分别是：

子表单（Child Form）：用于创建MDI窗口中的一个表单，它不能超出主表单的范围，最小化时显示在MDI窗口的底部。如果主窗口最小化，它也随之最小化。

浮动表单（Floating form）：属于一个主表单，但并不包含在主表单内部。它可以移动到屏幕的任何位置，但不能移动到主窗口的背后。最小化时，它的最小化图标位于桌面的底部。当MDI最小化时，它也随之最小化。

顶层表单（Top-level form）：没有父表单的独立表单，用来创建SDI应用，或者作为MDI应用中的主表单存在。它将显示于Windows的任务栏。

9.3.3　设置表单的不同类型

表单在第一次建立的时候都是一样的，通过更改表单的属性可以将表单设置为你所需要的类型。

设置表单类型的操作步骤如下：

说明　本例设置表单的类型，并演示了表单的调用关系。

开始　新建一个项目“表单演示”。

下一步　在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步　在“表单设计器”中，设置“ShowWindow”属性为“3-作为顶层表单”，“name”属性为“frmMDI”，“Caption”属性为“MDI 表单”。

说明　表单 ShowWindow 属性共有三个选项，它们分别为：

0-在屏幕中：产生子表单、Visual FoxPro 窗口将成为它的主窗口。

1-在顶层表单中：子表单的父表单将是顶层表单，它将被限制在顶层表单的内部。

2-作为顶层表单：设置一个没有父表单的独立表单，它可以作为 MDI 的主窗口或 SDI。

下一步　保存表单为“MDIForm.scx”。

下一步　新建一个表单，设置“ShowWindow”属性为“1-在顶层表单中”，设置“MDIForm”属性为“.T.”，“name”属性为“frmChild”，“Caption”属性为“子表单”。

下一步　保存表单为“ChildForm.scx”。

下一步　新建一个表单，设置“ShowWindow”属性为“0-在屏幕中”，设置“Desktop”属性为“.T.”，“name”属性为“frmFloat”，“Caption”属性为“浮动表单”。

下一步　保存表单为“FloatForm.scx”。

说明　表单是主表单、子表单还是浮动表单，取决于 ShowWindow、MDIForm、Desktop 的属性值，如表 9.1 所示。

表9.1　主表单、子表单和浮动表单的属性设置

表单类型	ShowWindow 属性	MDIForm	Desktop
主表单	2-作为顶层表单	.F.	.F.
子表单	0-在屏幕中 1-在顶层表单中	.T.-最大化时子表单和父表单相连 .F.-最大化时子窗口为分开窗口	.F.
浮动表单	0-在屏幕中 2-作为顶层表单	.F.	.T.

下一步　修改表单“MDIForm.scx”。

下一步　在“表单设计器”中向 MDIForm 表单添加一个“命令按钮”控件。

下一步　设置“命令按钮”控件的“Caption”属性为“显示子表单”。

下一步 双击“命令按钮”，在代码窗口中向它的“Click”事件中添加如下的代码。

```
DO FORM e:\practice\forms\ChildForm
```

下一步 从“表单”菜单中选择“执行表单”，将出现图 9.26 所示的主表单。

下一步 在“MDI 表单”中单击“显示子表单”命令按钮显示“子表单”。

注意 当显示子表单时，主表单必须已经处于激活状态，因此你不能在主表单的“Init”事件中显示子表单，因为此时主表单还没有被激活。

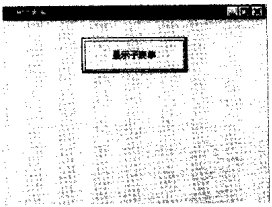


图9.26　MDI表单

完成 关闭所有打开的表单。

9.3.4　给主表单添加菜单

必要的时候，可以为主表单添加菜单。

给主表单添加菜单的操作步骤如下：

开始 打开 “表单演示” 项目。

下一步 在“项目管理器”中用“快速菜单”方法新建一个菜单“MDI 菜单.mnx”，并生成菜单程序“MDI 菜单.mpr”。

下一步 在“项目管理器”中，选择“MDIForm”表单，然后按“修改”按钮。

下一步 在“表单设计器”中，双击“MDIForm”表单。

下一步 在代码窗口中为表单的“Init”事件加上如下的代码。

```
DO e:\practice\menus\MDI 菜单.mpr
```

说明 添加菜单的语句可以表示为：DO menuname.mpr WITH oForm, lAutoRename，其中 menuname.mpr 为所要加入的菜单文件名，oForm 是对表单对象的引用，如果在表单的 Init 事件中执行此代码可以用 THIS，lAutoRename 确定是否为菜单设置一个唯一的标识。

下一步 保存表单，然后从“表单”中选择“执行表单”。

完成 关闭所有打开的表单。

9.3.5　创建表单集

表单集是一个或多个表单的容器，在“表单设计器”中可以创建表单集。

创建表单集的操作步骤如下：

说明 本实例介绍如何建立表单集，如何向表单集中添加表单以及如何从表单集中删除表单。

开始 打开 “表单演示” 项目。

下一步 在“项目管理器”中，选择“MDIForm”表单，然后按“修改”按钮。

下一步 从“表单”菜单中，选择“创建表单集”。

下一步 在“属性”窗口中设置表单集的属性，“Name”设置为“frsMDI”。

下一步 从“表单”菜单中，选择“添加新表单”。

下一步 此时“表单设计器”中出现一个新的表单，添加完毕。

说明　当你运行一个表单集时，你可能不想见到表单集中的所有表单，只需要将不需要见到的表单的“Visible”属性设为(.F.)，当需要显示表单时在将它的属性设为(.T.)。

下一步　在“表单设计器”中选择新加的表单。

下一步　从“表单”菜单中，选择“删除表单”可以从表单集中删除表单。

完成　从“表单”菜单中，选择“删除表单集”可以删除表单集。

9.3.6　对象集合和对象数量属性

在 Visual FoxPro 中，容器对象都具有对象集合（Collection）和对象数量（Count）属性。对象集合（Collection）属性是一个对象数组，每个元素包含了一个对象。对象数量（Count）属性是一个数值属性，它记录了容器中对象的书目。

每个容器的对象集合（Collection）和对象数量（Count）属性根据容器对象的不同具有不同的名称，它们的名称如表 9.2 所示。

表9.2　容器中的对象集合和对象数量

容器类型	对象集合	对象数量
应用（Application）	Objects Forms	Count FormCount
表单集（FormSet）	Forms	FormCount
表单（Form）	Objects Controls	Count ControlCount
页框（PageFrame）	Pages	PageCount
页（Page）	Controls	ControlCount
表格（Grid）	Columns	ColumnCount
命令组（CommandGroup）	Buttons	ButtonCount
列（Column）	Controls	ControlCount
工具栏（ToolBar）	Controls	ControlCount
容器（Container）	Controls	ControlCount
控件（Control）	Controls	ControlCount

对象集合属性和对象数量属性允许你通过循环来处理容器中的对象。

通过循环来处理容器中对象的程序如下：

说明　本实例用编程的方法，通过调用表格对象的对象数量和对象集合属性，利用循环改变表格的背景色设置。

程序 9.6　通过循环来处理容器中对象

```
o = THISFORM.grd1                                    &&得到当前表格对象
FOR i = 1 to o.ColumnCount
    IF i % 2 = 0                                     && 如果是偶数列
        o.Columns(i).BackColor = RGB(0,255,0)        && 变为绿色
    ELSE                                             && 如果是奇数列
```

```
        o.Columns(i).BackColor = RGB(255,0,0)          && 变为红色
    ENDIF
ENDFOR
```

9.4 使用控件

9.4.1 控件和数据

Visual FoxPro 的表单中可以使用两种控件：绑定型控件和非绑定型控件。对于绑定型控件，具有数据源属性，数据直接存入表、字段或者变量中。

对于绑定型控件，需要设置数据源属性“ControlSource”，对于表格（Grid）为“RecordSource”。表 9.3 中列出了控件和它的数据源属性的功能。

表9.3　控件的数据源属性的功能

空间类型	数据源属性的功能
复选框 （Check box）	如果 ControlSource 属性为表中的一个字段，则当表的指针发生移动时，字段值是 NULL 值、逻辑真（.T.）、逻辑假（.F.）或者为 0、1、2 时分别表示选中、清除、变灰状态。
列 （Column　）	如果 ControlSource 属性是一个表字段，用户对列中数值的编辑相当于直接编辑表中的字段。如果将表和表格绑定设置 RecordSource 属性
列表框或组合框 （Listbox 或 combo box）	如果 ControlSource 是一个变量，则在列表中的选项将存入这个内存变量。如果 ControlSource 属性是一个表字段，在列表中的选项将存入当前记录指针的字段中，表字段中的值和列表中的值相匹配，将自动在列表中选定此项。
选项按钮 （Option button）	如果 ControlSource 是一个数值型字段，字段值 0、1 表示按钮是否被选中。如果 ControlSource 是一个逻辑型字段，逻辑值（.T.）、（.F.）表示按钮是否被选中。当表中的指针移动时，选项按钮的状态自动随字段的值更新，如果选项组控件的 ControlSource 属性是一个字符字段，则选项按钮被选中时，它的标题自动存入字段。注意选项按钮的 ControlSource 不能为字符型。
微调 （Spinner）	微调的值取决于字段或变量的数值
文本框或编辑框 （Text box 或 edit box）	文本框中显示的值就是表中字段的值，它的值如果发生改变将自动反映到表字段中。移动记录指针，则当前记录的字段值为文本框的值

9.4.2 选择合适的控件

每一种控件都能完成某个特定的任务，但根据任务种类的不同，Visual FoxPro 中应保持任务种类和控件类型的一致性。

表 9.4 列出了 Visual FoxPro 中功能和所选控件的对应关系。

表9.4　功能和所选控件对应关系

功能类型	应该选择的控件类型
提供给用户预先设定的选择	选项按钮组、列表框和下拉列表框、复选框
接受不能预先设定的用户输入	文本框、编辑框、组合框
接受有一定值范围的输入	微调
执行具有一定时间间隔的特定任务	计时器
显示信息	图像、标签、文本框、编辑框、形状

9.4.3　使用选项按钮组

选项按钮组

选项按钮组控件是包含选项按钮的控件。它允许用户对对话框中的一组选项作出选择，每一次只能选中一个选项。

使用选项按钮组控件的操作步骤如下：

说明　通过向表单中加入“选项按钮组”控件，用户通过本实例可以学习如何设置选项按钮的数目，如何设置按钮的属性、如何选定当前的按钮。

开始　新建一个项目“使用控件”。

下一步　在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步　从“表单控件”工具栏中选择“选项按钮组”控件，将它加入表单中。

下一步　在“属性窗口”设置选项按钮组的“ButtonCount”属性为“4”。

说明　选项按钮组中选项的数目默认值为 2，如果要改变按钮组的数目，可以更改选项按钮组的“ButtonCount”属性。

下一步　鼠标右键单击“选项按钮组”控件，在快捷菜单中选择“编辑”，此时可以选择“选项按钮组”控件中的选项按钮进行编辑，如图 9.27 所示。

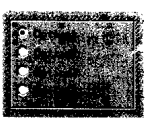


图9.27　编辑选项按钮

下一步　在属性对话框中设置“Option1”的“Caption”属性设为“选项按钮 A”。

下一步　双击选项按钮组进入代码窗口，在“OptionGroup1”的“Click”事件加入如下的代码。

程序 9.7　“OptionGroup1”的“Click”事件代码

```
THISFORM.Optiongroup1.Option2.Caption="选项按钮 B"
THISFORM.Optiongroup1.Buttons(3).Caption="选项按钮 C"
```

在程序中设置按钮的属性，可以通过指定选项按钮组中选项按钮的名称来设置。另一种方法是通过使用选项按钮组的 Buttons 属性，指定选项按钮在选项按钮组中的索引号来设置。

下一步　在代码窗口中，在“OptionGroup1”的“DblClick”事件加入如下的代码。

程序 9.8 “OptionGroup1”的“DblClick”事件代码

```
THISFORM.Optiongroup1.Buttons(4).Caption="选项按钮 D"
THISFORM.Optiongroup1.SetAll("Enabled",.F.,"OptionButton")
```

如果要设置所有选项按钮的属性，可以使用 SetAll 方法，上面的代码是将选项按钮组中的所有选项的启用状态设置为废止状态。

下一步 在代码窗口中，在“OptionGroup1”的“RightClick”事件加入如下的代码。

程序 9.9 OptionGroup1”的“RightClick”事件代码

```
cSelected=THISFORM.Optiongroup1.Buttons(THISFORM.Optiongroup1.Value).Caption
Messagebox("你选择了："+cSelected)
```

为了判断当前选定的按钮，可以使用选项按钮组的“Value”属性，它的值为当前选择的按钮在选项按钮组中的索引值，上面的代码先将当前选择的按钮的“Caption”属性保存于变量 cSelected 中，然后用消息框将它显示出来。

下一步 关闭代码窗口，保存表单名为“表单 Option”。

下一步 从“表单”菜单中选择“执行表单”，选项按钮组初始状态如图 9.28 所示。

下一步 在“表单”中鼠标左键单击选项按钮组，单击鼠标后的状态（见图 9.28）。

下一步 在“表单”中鼠标左键单击选项按钮组，选择一个合适的选项按钮，然后鼠标右键单击选项按钮组，出现消息对话框。

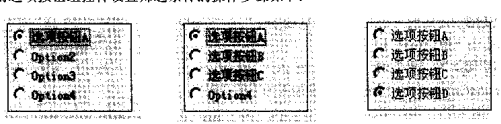
下一步 关闭显示的消息对话框。

下一步 在“表单”中鼠标左键双击选项按钮组，双击鼠标后的状态。状态变化如图 9.28 所示。

完成 关闭所有打开的表单。

通过选项按钮，可以实现对数据库中的数据表记录的筛选功能。

用选项按钮组控件设置筛选条件的操作步骤如下：



初始状态　　单击鼠标后　　双击鼠标后

图9.28 选项按钮组在执行时的状态变化

说明 通过向表单中加入“选项按钮组”控件，用户通过本实例可以学习如何设置选项按钮的数目、如何设置按钮的属性、如何选定当前的按钮。

开始 打开项目“使用控件”。

下一步 在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步 在“表单设计器”设置表单的数据环境，在数据环境中加入以前建立的表“客

户”。

在表单设计器中，向表单加入一个列表框。

在表单设计器中，向表单中加入一个选项按钮组控件。

在属性窗口中设置选项按钮组控件的“ButtonCount”属性为“3”。

在属性窗口中，设置列表框的“Name”属性为“lstCustomers”，然后设置它的属性，如表 9.5 所示。

表9.5　列表框的属性

对象	属性	属性值
lstCustomers	RowSourceType	2-别名
lstCustomers	RowSource	客户

在属性窗口中，设置选项按钮组控件的属性，如表 9.6 所示。

表9.6　选项按钮组控件的属性

选项按钮	“Name”属性	“Caption”属性
Option1	optAll	所有客户
Option2	optBJ	北京的客户
Option3	optSH	上海的客户

进入代码窗口，设置用户选择选项按钮的筛选事件的代码，如表 9.7 所示。

表9.7　用选项按钮设置筛选条件

对象	过程	代码
optAll	Click	SET FILTER TO GO TOP THISFORM.lstCustomers.Requery
optBJ	Click	SET FILTER TO 客户.城市="北京" GO TOP THISFORM.lstCustomers.Requery
optSH	Click	SET FILTER TO 客户.城市="上海" GO TOP THISFORM.lstCustomers.Requery

关闭代码窗口。

运行表单，如图 9.29 所示，选择不同的选项按钮，看对列表框中显示的数据的影响。

关闭表单。

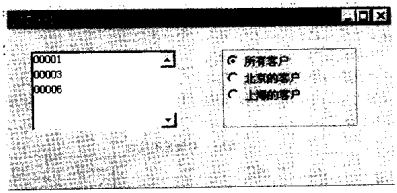


图9.29　用选项按钮组控件筛选记录的表单

9.4.4　使用列表框和下拉列表框

列表框　　组合框

列表框和下拉列表框（Style 属性为“2-下拉列表框”的组合框）提供了一个可以滚动的列表。列表框中可以显示多个列表项，而下拉列表框中只显示一个项，只有当鼠标单击右侧的下拉按钮时，才显示所有的列表。

使用列表框和下拉列表框的操作步骤如下：

说明　通过向表单中加入“选项按钮组”控件，用户通过本实例可以学习如何设置选项按钮的数目、如何设置按钮的属性、如何选定当前的按钮。

开始　从“帮助”菜单运行示例程序“Solutions Sample”。

下一步　选择“controls”中的示例“Fill a list with different sources”。

下一步　选择“Run Samples”按钮，出现如图 9.30 所示的表单。

下一步　对表单进行操作，看对“RowSourceType”下拉列表框做不同的选择时，列表框的值的变化。

下一步　退出表单，然后选择“see code”按钮。

下一步　双击表单，进入代码窗口，按照表 9.8 所示的内容查看代码。

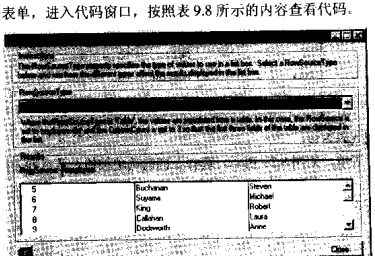


图9.30　列表框和下拉列表框表单

表9.8　查看示例代码

对象	过程	从代码中查看内容
cboTypes	Init	组合框的 Value 属性，AddItem 方法
cboTypes	InteractiveChange	LstDisplay 的 ColumnCount 属性 lstDisplay 的 RowSourceType 属性 lstDisplay 的 RowSource 属性

下一步　如果有兴趣可以查看并分析其他的代码。

完成　关闭所有窗口

常用的列表框的属性如表 9.9 所示。

表9.9　常用的列表框的属性

属性	属性的使用说明
ColumnCount	列表框中的列数
ControlSource	用户在列表框中选择的值存在什么地方
MoverBars	列表框的左侧是否显示移动按钮
MultiSelect	是否可以在列表框中选择多个项
RowSource	列表数据的来源
RowSourceType	确定数据源 RowSource 的值是：一个值、一个表、一条 SQL 语句、一个查询、一个数组、一个文件列表、一个字段列表

说明　列表框的 Value 属性默认值是字符型，但它可以是数值型或者字符型。当 RowSource 属性是一个字符型，并希望反映列表框选择的字符串，那么 Value 属性的要设置为空字符串，先按 SPACEBAR，然后按 BACKSPACE 键可以在属性窗口中输入空白字符串。

常用的列表框的方法如表 9.10 所示。

表9.10　常用的列表框的方法

方法	方法的使用说明
AddItem	当 RowSourceType 属性是 0 时给列表框添加一项
RemoveItem	从 RowSourceType 属性是 0 的列表框移去一项
Requery	数据源 RowSource 的值改变时更新列表

RowSourceType 属性决定了列表框或者下拉列表框数据源的类型，用户总是先选择数据源的类型，然后在设置数据源属性 RowSource。下面列出各种类型的数据源可能选择的数据源。

（1）　0-无

当 RowSourceType 设置为“0-无”时，只能使用 AddItem 方法向列表中添加项，如：

```
frmForm1.lstMyList.RowSourceType = 0
frmForm1.lstMyList.AddItem("第一项")
frmForm1.lstMyList.AddItem("第二项")
```

用 RemoveItem 方法可以从列表中移去一项，如：

```
frmForm1.lstMyList.RemoveItem(2)
```

（2） 1-值

当 RowSourceType 设置为“1-值”时，RowSource 属性可以指定多个需要在列表中显示的值。如果用属性窗口设置 RowSource 属性，列表项应该用“，”号分隔。如果在程序中设置要加上双引号并用逗号隔开，如：

```
Form1.lstMyList.RowSourceType = 1
Form1.lstMyList.RowSource = "第一,第二,第三,第四"
```

（3） 2-别名

当 RowSourceType 设置为“2-别名”时，列表框中可以显示包含表中的一个或多个字段。当 ColumnCount 属性设置为 0 或者 1 时，列表框显示第一个字段的值。ColumnCount 属性为 3，列表框中将显示最前面的三个字段。列表框中的字段显示次序和表中的次序是一致的。

（4） 3-SQL语句

当 RowSourceType 设置为“3-SQL 语句”时，RowSource 属性设置为一条“SELECT-SQL”语句，当在程序中使用“SELECT-SQL”语句设置 RowSource 属性时要加上引号，例如：

```
THISFORM.lstDisplay.RowSource = "SELECT first_name,last_name from employee;
    where first_name='A' into cursor temp"  &&从职员表中选取记录到临时表 temp
```


默认情况的 Visual FoxPro 不带 INTO 子句的 SELECT 语句将在浏览窗口中立即显示临时表，因此要加上 into cursor 子句，从而避免这种情况。

（5） 4-查询（.QPR）

当 RowSourceType 设置为“4-查询（.QPR）”时，将使用查询结果填充列表框。此时 RowSource 属性设置为一个查询文件（.QPR 文件）。例如：

```
THISFORM.List1.RowSource = "城市.qpr"
```

（6） 5-数组

RowSourceType 设置为“5-数组”时，将使用数组元素填充列表框。此时 RowSource 属性设置为给表单或表单集创建的数组属性，或者在应用程序中创建的数组。

列表框的 RowSource 属性不是在设置它的时候进行设置，而是在应用程序需要的时候进行设置，因此要注意你所创建的数组的作用域，当把 RowSource 属性设置为表单或表单集的数组属性时，应该将它设置为相对于列表框的引用，而不是方法。例如把名为 arrayprop 的表单数组属性，在 Init 方法中设置 RowSource 属性有如下的正确和错误的设置方法：

```
THIS.lst1.RowSource = "THIS.arrayprop"          &&  错误
THIS.lst1.RowSource = "THISFORM.arrayprop"      &&  正确
```

如果使用多维数组设置列表的数据源，请按照如下步骤进行：

开始 把 RowSourceType 设置为“5-数组”。

下一步 把 RowSource 属性设置为多维数组。

下一步 把 ColumnCount 属性设置为需要显示的列数。

完成 把 ColumnWidths 属性设置为每一列的宽度。

（7） 6-字段

RowSourceType 设置为“6-字段”时，将使用一个或多个用逗号分隔的字段值填充列表框。例如：

```
联系人,公司名称,城市
```

当此 RowSourceType 设置为“6-字段”时，RowSource 属性可以包含如下的几种类型的信息：

字段

别名.字段

别名.字段,字段,字段,...

注意 当需要在列表框中显示来自多个表中的字段时，应将 RowSourceType 设置为“3-SQL 语句”。当 RowSourceType 设置为“6-字段”时，列表框中的字段位置可以自行设定。

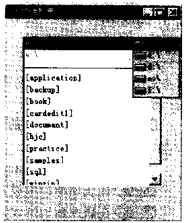


图9.31　文件名填充列表

（8） 7-字

RowSourceType 设置为“7-字段”时，将使用当前目录下的文件填充列表框。如图 9.31 所示。

此时列表框中可以列出文件、路径、驱动器，用户可以自行选择文件的列表。RowSource 属性可以设置为文件类型名，例如当将 RowSource 属性设置为“*.DBF”时将显示所有的后缀为.DBF 的文件。

（9） 8-结构

RowSourceType 设置为“8-结构”时，将使用 RowSource 属性指定的表中的字段名来填充列表框。当需要用一系列字段来查找表中的值或者对表进行排序时，可以将 RowSourceType 属性设置为“8-结构”。

（10） 9-弹出菜单

RowSourceType 设置为“9-弹出菜单”时，可用一个定义好的弹出菜单来填充列表框。

创建使用列表框组合框的操作步骤如下：

说明 本实例介绍了如何允许列表框中显示多列，如何允许在列表框中选择多个选项并处理被选择的项，如何在程序中向列表中添加项。本例中还包括了文本框的使用。

开始 新建一个项目“使用控件”。

下一步 在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步 从“表单控件”工具栏中选择“列表框”控件，将它加入表单中。

下一步 从“表单控件”工具栏中选择“标签”控件 A，将它加入表单中，设置它的“Caption”属性为“被选项数目”。

下一步 从“表单控件”工具栏中选择“文本框”控件，将它加入表单中。

从“表单控件”工具栏中选择“组合框”控件，将它加入表单中，如图 9.32 所示。

在“表单设计器”设置表单的数据环境，在数据环境中加入以前建立的表“客户”。

在属性窗口中设置 list1 的 ColumnCount 属性为“3”。

说明　列表框列的默认值是一列，但它可以设置为多列，只需要重新设置 ColumnCount 属性为所需的列数。当设置为多列时，可以一次选中一行。

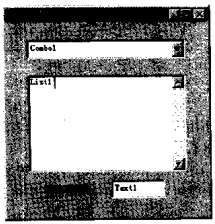


图9.32　创建使用列表框和组合框

在代码窗口中的“FORM1”的“Init”事件下加入如下代码。

```
THISFORM.list1.ColumnWidths="100,50,50"
```

在属性窗口中设置 list1 的 RowSourceType 属性为“6-字段”。

在属性窗口中设置 list1 的 RowSource 属性为“公司名称,联系人,城市”。

在属性窗口中设置 list1 的 MultiSelect 属性为“.T.-真”。

说明　列表框列的默认情况下一次只能选择一列，如果要选择多个列将 MultiSelect 属性为“.T.-真”。

在代码窗口中的“list1”的“InteractiveChange”事件下加入如下代码。

程序 9.10　“list1”的“InteractiveChange”事件代码

```
nNumberSelected = 0                                    && 用来存储被选项数目的变量
THISFORM.Combo1.Clear                                  &&将复选框清空
FOR nCnt = 1 TO THIS.ListCount
    IF THIS.Selected(nCnt)                             &&如果被选择
        nNumberSelected = nNumberSelected + 1          &&被选项数目加一
        THISFORM.Combo1.Additem (THIS.List(nCnt))      &&并将它加入组合框
    ENDIF
ENDFOR
THISFORM.Text1.Value = nNumberSelected
```

关闭代码窗口，在“表单”菜单中选择“执行表单”。

在表单的列表框中选择多行多列，看组合框和文本框的值的变化，见图 9.33 所示。

在代码窗口中的“Text1”的“KeyPress”事件下加入如下代码。

程序 9.11　“Text1”的“KeyPress”事件代码

```
LPARAMETERS nKeyCode, nShiftAltCtrl
IF nKeyCode = 13        && Enter 键
    THISFORM.Combo1.Additem(This.Value)
    THIS.Value = ""
```

ENDIF

说明 用户可以交互的向列表框中添加项目，此时要使用AddItem方法。上面的代码，当用户在文本框中输入一个字符串，并按 Enter 键后，文本框中的文本被自动的加入到组合框（下拉列表框）中，文本框的文本自动清空。

下一步 关闭代码窗口，在“表单”菜单中选择“执行表单”。

下一步 在表单的文本框输入文本，然后按“Enter”键，查看组合框中的列表项。

完成 关闭表单，将它存为“表单 List”。

图9.33　运行中的表单

通过在列表框选择一项转到相关的记录：

很多时候你可能根据列表框中的选择而查看或编辑相关的记录。Visual FoxPro 中有多种方式实现这一功能，具体采用什么方法取决于数据源的类型，如表 9.11 所示。

表9.11　列表框选项与对应的记录选择方式

RowSourceType 属性	记录选择方式
2 – 别名 6 – 字段	当从列表框中选择一项时，将自动的将记录指针设置为当前选项。只需在列表框的 InteractiveChange 事件中加入 THISFORM.Refresh 方法，其它控件的值将自动变为当前选择的记录。
1 – 值 3 – SQL 语句 4 – 查询（.QPR） 5 – 数组	在列表框的 InteractiveChange 事件中首先选择数据表，然后根据需要搜索值。例如： SELECT 客户 LOCATE FOR THIS.Value = “北京” THISFORM.Refresh

9.4.5　使用复选框

复选框

复选框可以设置布尔状态：真或假，开或关。有时这种判断是不确定的，因此还存在一种不做判断的状态。

通常根据复选框的 Value 属性的值，复选框可以具有四种状态，如表 9.12 所示。

表9.12　列表框的Value值和状态关系

显示状态	Value 属性设置
Check1	0 或.F.
Check1	1 或.T.
Check1	2
Check1	.NULL.

当把复选框的“ControlSource”属性设置为数据表中的一个逻辑字段时，则当复选框被选中时，记录中的字段值为“真（.T.）”。当复选框未被选中时，记录中的字段值为“假

（.F.）”。当复选框为灰色时，记录中的字段值为“.NULL.”。

9.4.6 使用文本框

文本框

文本框可以添加、编辑表中的非备注字段的数据。文本框中的数据保存在“Value”属性中。当设置了文本框的数据源 ControlSource 属性时，文本框中的值和表中的字段的值是绑定的，文本框中的值将和数据表中当前记录字段的值保持一致。

使用文本框的操作步骤如下：

说明　通过本例用户学习使用文本框的基本属性，包括判断输入文本的有效性，选择选定的文本，对文本框进行格式编辑，用文本框接受口令等内容。

开始　打开项目“使用控件”。

下一步　在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步　在“表单设计器”中向表单加入四个文本框和四个标签。

下一步　设置文本框和标签的属性如表 9.13 所示。

表9.13　文本框和标签的属性设置

对象	属性	属性值
Label1	Caption	日期文本框
Label2	Caption	字符文本框
Label3	Caption	格式文本框
Label5	Caption	口令文本框
Text1	Name	txtDate
Text2	Name	txtChar
Text3	Name	txtMask
Text4	Name	txtPassword

下一步　将四个标签的“FontSize”属性设置为“12”。

下一步　进入代码窗口在“txtDate”对象的“Valid”事件加入如下的代码。

程序 9.12　“txtDate”对象的“Valid”事件代码

```
IF CTOD(THIS.Value) < DATE( )
    = MESSAGEBOX("请输入将来的时间",1)
    RETURN 0
ENDIF
```

说明　判断文本框的输入的值是否有效，应该在文本框的“Valid”事件中加入判断的代码。如果所输入的值为无效值，给出提示信息并返回“.F.”或“0”，上面的代码判断在文本框中输入的值是否为正确的日期。

下一步　在代码窗口在“txtChar”对象的“GotFocus”事件加入如下的代码。

程序 9.13　“txtChar”对象的“GotFocus”事件代码

```
NODEFAULT                                        &&不执行默认过程
TextBox::GotFocus
THIS.BackColor=RGB(255,128,255)                  &&背景色设为粉红色
THIS.SelStart=0
THIS.SelLength=LEN(ALLTRIM(THIS.Value))          &&将选择字符设为全选
```

下一步 在代码窗口在“txtChar”对象的“LostFocus”事件加入如下的代码。

程序 9.14　“txtChar”对象的“LostFocus”事件代码

```
THIS.BackColor=RGB(255,255,255)                  &&背景色恢复为白色
```

用户可以改变所选择的文本框的背景颜色，只需对 GotFocus 和 LostFocus 事件进行处理。也可以在选择文本框时自动选择其中的所有文本。

下一步 在属性窗口中设置“txtMask”的“InputMask”属性为“$”。

文本框的“Format”属性决定了显示方式，InputMask属性决定了用户输入的值的范围，设置的语法是：

```
Control.InputMask[ = cMask]
```

其中 cMask 的可以用表 9.14 所示的替代符组成。

表9.14　文本框和标签的属性设置

设置符	说明
X	任何的输入字符
8	数字和符号，如“？”
#	数字、空格和符号
$	输入当前设置的货币型数值（用 SET CURRENCY 设置）
$$	在文本框或微调旁边显示一个浮动的货币型符号
*	在值的左侧显示一个星号
.	在小数点位置输入“.”
,	用来分隔小数点左侧的数字

下一步 在属性窗口中设置“txtPassword”的“PasswordChar”属性为“*”。

说明　口令由于安全的需要，常常要将输入的字符不显示出来，只需要将文本框的“PasswordChar”属性设置为替代的字符。

下一步 运行表单，在日期文本框中输入值，看它对正确输入和错误输入的反应。

下一步 在字符文本框中输入值看它的背景色变化。

下一步 在格式文本框中输入值。

下一步 在口令文本框中输入值，见图 9.34 所示。

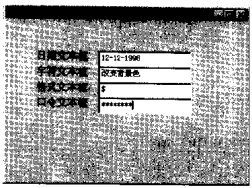


图9.34　文本框的使用

完成　关闭表单。

文本框的常用属性见表 9.15 所示。

表9.15　文本框常用属性说明

属性	属性说明
Alignment	设置文本框文本的显示处对左、对中、对右还是自动
ControlSource	在文本框中显示的表字段或变量
InputMask	设置输入掩码
SelectOnEntry	文本框获得焦点时是否自动选择所有文本
TabStop	是否可用 Tab 选择

9.4.7　使用编辑框

编辑框

对于长字段或者备注字段的文本可以用编辑框进行编辑，它可以实现自动换行并可以使用方向键和滚动条来浏览文本。

当用编辑框编辑备注字段时，只需要将该字段的 ControlSource 属性设置为该备注字段即可。

使用编辑框的操作步骤如下：

说明　通过本例用户学习使用编辑框的基本属性，如何向编辑框中输入数据，编辑框的属性设置等。

开始　打开项目“使用控件”。

下一步　在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步　在“表单设计器”中向表单加入一个编辑框和三个命令按钮。

下一步　设置文本框和命令按钮的属性如表 9.16 所示，设计好的表单如图 9.35 所示。

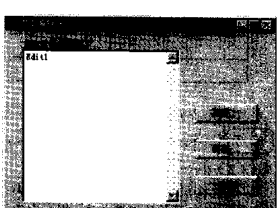


图9.35　文本编辑框表单

表9.16　文本框和命令按钮的属性设置

对象	属性	属性值
Label1	Caption	文本文件编辑框
Command1	Caption	打开…
Command2	Caption	保存…
Command3	Caption	确定
Command1	Name	cmdOpen
Command2	Name	cmdSave
Command3	Name	cmdOK
Form1	Caption	文本编辑框

下一步 给表单“Form1”增加自定义属性“cString”。

下一步 设置“Edit1”的“ControlSource”属性为“THISFORM.cString”。

下一步 在代码窗口设置“cmdOK”的“Click”代码如下。

程序 9.15　“cmdOK”的“Click”代码

```
RELEASE THISFORM          &&释放表单
```

下一步 在代码窗口设置“cmdOpen”的“Click”代码如下。

程序 9.16　“cmdOpen”的“Click”代码

```
*-------------------------------------------------------------------------------*
*首先建立一个内存中的 CURSOR，具有
*一个字符字段和一个备注字段，用来保存
*文件名和文本文件
CREATE CURSOR textfile (filename c(35), mem m)
APPEND BLANK                                    &&给游标添加一个空白记录
REPLACE textfile.FileName WITH ;                &&用 GETFILE( )函数得到文本文件的文件名
    GETFILE("TXT")                              &&并将它存于文件名字段中
IF EMPTY(textfile.FileName)                     &&如果用户选择“取消”则文件名为空
    RETURN
ENDIF
APPEND MEMO mem FROM ;                          &&将文本文件的内容添加到备注字段
   (textfile.FileName) OVERWRITE
THISFORM.Edit1.ControlSource = ;                &&设置表单中的文本框的数据源
   "textfile.mem"
THISFORM.Refresh
THISFORM.cmdSave.Enabled = .T.                  &&将“保存…”按钮启用
```

下一步 在代码窗口设置“cmdSave”的“Click”代码如下。

程序 9.17　“cmdSave”的“Click”代码

```
COPY MEMO textfile.mem TO ;
    (textfile.filename)                    &&保存文本文件
```

下一步 运行表单，单击“打开...”按钮，文本框中加入文本，如图 9.36 所示。

下一步 选择“保存...”，将有是否保存更改的对话框，保存文件。

完成 按“确定”退出表单。

编辑框和文本框有三个属性用于处理对选定文本的处理：SelLength、SelStart 和 SelText。

在程序中使用 SelLength、SelStart 属性可以从程序中选择文本。下面的代码可以在编辑框中选择第一个单词：

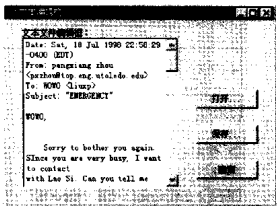


图9.36　打开文本文件加入编辑框中

程序 9.18　使用 SelLength、SelStart 属性

```
Form1.Edit1.SelStart = 0
Form1. Edit1.SelLength = AT(" ", Form1. Edit1.Text) - 1
```

下面的代码使用 SelText 将选定的文本变为大写：

程序 9.19　使用 SelText 属性

```
Form1.edtText.SelText = UPPER(Form1.edtText.SelText)
```

编辑框中常用的设计属性见表 9.17 所示。

表9.17　编辑框常用属性说明

属性	属性说明
AllowTabs	是否允许用户在编辑框中使用 Tab 键，而不是调到下一个控件
HideSelection	编辑框失去焦点时选定的文本是否仍然是选定状态
ReadOnly	文本框中的文本是否可以被修改
ScrollBars	文本框是否具有垂直滚动条

9.4.8　使用组合框

组合框

组合框可以认为是文本框和列表框的结合。通过设置组合框的 Style 属性可以得到两种类型的组合框-“0-下拉组合框”和“2-下拉列表框”。

下拉组合框允许用户在组合框的文本框中输入新的项。

使用组合框的操作步骤如下：

说明　通过本例用户学习使用组合框的基本属性，通过两个不同类型的组合框，学习判断输入文本，添加项等内容。

开始　打开项目“使用控件”。

下一步　在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步　在“表单设计器”中向表单加入两个组合框。

下一步　在属性窗口将“Combo1”的“Style”属性设置为“0-下拉组合框”，将“Combo2”的“Style”属性设置为“2-下拉列表框”。

下一步　在代码窗口在“Combo1”的“Valid”事件下加入如下的代码。

程序 9.20　“Combo1”的“Valid”事件代码

```
lItemExists = .F. &&  假设输入的值不在列表中.
FOR i = 1 to THIS.ListCount
   IF THIS.List(i) = THIS.Text                          &&如果在列表中有输入的值
      MessageBox("列表中已经具有此项！！")
      lItemExists = .T.                                 &&退出
      EXIT
   ENDIF
ENDFOR
IF !lItemExists
   THIS.AddItem(THIS.Text)                              &&将输入的信息加入列表
   THISFORM.Combo2.AddItem(THIS.Text)
ENDIF
```

说明　在向列表添加新的一项之前，最好先检查当前的列表中是否已经存在此项，如果包含则需不在添加。

下一步　保存表单然后运行，如图 9.37 所示。

下一步　在第一个组合框中输入几项，按“Enter”键。

下一步　进入第二个组合框看它和第一个的联系和区别。

下一步　在第一个组合框中输入已有的项，看它的消息判断。

完成　退出表单。

组合框中常用的设计属性见表 9.18 所示。

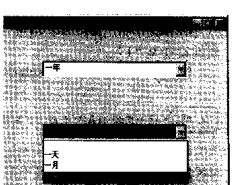


图9.37　使用列表框

表9.18 组合框常用属性说明

属性	属性说明
ControlSource	设置表的字段，输入用于保存输入或选择的值
InputMask	用于下拉组合框，设置用户可以输入的值的类型
IncrementalSearch	在用户输入每一个字母以后是否和列表中的值相匹配
RowSource	设置组合框的项的来源
RowSourceType	设置组合框的数据源类型它与列表框设置的意义相同
Style	设置组合框是下拉组合框还是下拉列表框

9.4.9 使用微调控件

微调

微调控件可以接受给定范围的值的输入。它可以通过设置微调的值来选择，也可以在微调的框中输入值，从而确保输入的值在给定的范围。

使用微调的操作步骤如下：

说明 通过本例用户学习使用微调控件的基本属性，对于经常用到的给定用户输入范围的输入，采用微调控件是一个很好的解决方案。

开始 打开项目“使用控件”。

下一步 在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步 在“表单设计器”中向表单加入两个微调。

下一步 在属性窗口将“Spinner1”的“KeyboardHighValue”属性和“SpinnerHighValue”属性设置为“100”。

下一步 在属性窗口将“Spinner1”的“KeyboardLowValue”属性和“SpinnerLowValue”属性设置为“50”。

说明 通过设置“KeyboardHighValue”属性和“SpinnerHighValue”属性可以控制微调中的最大的输入值，通过设置“KeyboardLowValue”属性和“SpinnerLowValue”属性可以控制微调中的最小的输入值。

下一步 在属性窗口将“Spinner2”的“KeyboardHighValue”属性和“SpinnerHighValue”属性设置为“5”。

下一步 在属性窗口将“Spinner2”的“KeyboardLowValue”属性和“SpinnerLowValue”属性设置为“1”。

下一步 在属性窗口将“Spinner2”的“Increment”属性设置为“-1”。

说明 对于用微调表示诸如“优先级”设置值，单击▲按钮表示优先级从“2”到“1”，此时可以设置增量选择为“-1”。如果要想让微调输入非数值型的值，可以调整微调的大小，使它只显示按钮，并在微调旁边加入文本框，然后在微调的 UpClick 和 DownClick 事件中加入处理代码，与文本框发生联系即可。

完成 运行表单。

微调中常用的设计属性见表9.19所示。

表9.19　微调常用属性说明

属性	属性说明
Interval	当用户单击↑或↓时的增加或减少的值
KeyboardHighValue	用户从文本框中输入的最大值
KeyboardLowValue	用户从文本框中输入的最小值
SpinnerHighValue	用户单击上按钮时在文本框中显示的最大值
SpinnerLowValue	用户单击上按钮时在文本框中显示的最小值

9.4.10　使用命令按钮和命令按钮组

命令按钮　命令按钮组

在对话框中经常可以见到“取消”、“确认”、“退出”等命令按钮控件，它们只是为了完成一定的任务，执行特定的操作，而不是为了输入值。

使用命令按钮和命令按钮组的操作步骤如下：

开始打开项目“使用控件”。

下一步在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步在“表单设计器”中向表单加入三个命令按钮和一个命令按钮组。

下一步在属性窗口设置它们的属性如表9.20所示。

表9.20　命令按钮和命令按钮组的属性设置

对象	属性	属性值
Command1	Caption	确定
Command2	Caption	取消
Command3	Caption	退出
Commandgroup1	ButtonCount	3
Commandgroup1. Command1	Caption	第一个按钮
Commandgroup1. Command2	Caption	第二个按钮
Commandgroup1. Command3	Caption	第三个按钮

下一步在属性窗口设置 Command3 的“Default”属性为“.T.-真”，此时的表单如图9.38所示。

说明当命令按钮的“Default”属性设置为“.T.-真”，它的边框要比其他的按钮多一条粗线的边框，当用户按“Enter”键时，将自动执行这个按钮的“Click”事件。

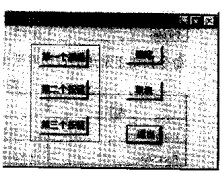


图9.38　使用按钮与按钮组控件

下一步在代码窗口中将“Command3”的“Click”过程加入如下代码。

```
RELEASE THISFORM
```

下一步在代码窗口中在“Commandgroup1”的“Click”事件中加入如下的代码。

程序 9.21 "Commandgroup1" 的 "Click" 事件代码

```
DO CASE
    CASE THIS.Value = 1
        WAIT WINDOW "你选择了" + THIS. Command1.Caption NOWAIT
        * 其他任务代码
    CASE THIS.Value = 2
        WAIT WINDOW "你选择了" + THIS. Command2.Caption NOWAIT
        * 其他任务代码
    CASE THIS.Value = 3
        WAIT WINDOW "你选择了 " + THIS.Command3.Caption NOWAIT
        * 其他任务代码
ENDCASE
```

说明 在命令按钮组的 "Click" 事件响应了命令按钮组中命令按钮的所有单击事件。命令按钮的 Value 的值指明了单击的按钮，如果单击命令按钮组而没有击中按钮组中的按钮，按钮组的 Value 值仍为上一次的值。如果为按钮组中的按钮指定了事件响应代码，则将执行按钮的事件代码，按钮组相对应的代码不执行。

下一步 运行表单，看在按钮组中单击鼠标时的响应。

重点 关闭表单。

命令按钮常用的设计属性见表 9.21 所示。

表9.21 命令按钮常用属性说明

属性	属性说明
Cancel	设置当用户按 ESC 时与命令按钮相连的代码
Caption	按钮的标题
DisabledPicture	当按钮禁止时显示的.BMP 文件
DownPicture	当按钮被按下时显示的.BMP 文件
Enabled	按钮是否可以被选择
Picture	按钮显示的图片

命令按钮组常用的设计属性见表 9.22 所示。

表9.22 命令按钮组常用属性说明

属性	属性说明
ButtonCount	按钮组中按钮的数目
BackStyle	设置按钮是透明的还是不透明的

9.4.11 计时器控件

计时器控件

计时器控件可以在用户指定的时间间隔执行操作并检查它的数值。计时器检查系统时

钟，在一定的时间内重复的触发时钟事件，从而执行某种操作。

使用计时器控件的操作步骤如下：

开始 打开项目“使用控件”。

下一步 在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步 在“表单设计器”中向表单加入一个计时器控件，一个标签控件。

说明 计时器在设计时是可见的，从而使用户方便地完成属性的设置，而在运行时，计时器是不可见的，因此它在设计时的位置和大小都无关紧要。

下一步 在属性窗口设置计时器和标签的属性如表 9.23 所示，此时的表单如图 9.39 所示。

图9.39　秒表表单

表9.23　计时器和标签属性设置

对象	属性	属性值
Label1	Caption	
Label1	Name	lblTime
Timer1	Interval	500
Timer1	Enable	True
Form1	Caption	秒表

说明 计时器的 Interval 属性，指定了计时器事件的发生间隔。当计时器为启用状态时，它将在相应的时间间隔中触发一个 Timer 事件。Interval 的间隔范围是 0~2147483647，因此它的最长的间隔为 596.5 小时。本例中时间间隔设为 500，表示每半秒执行一次 Timer 事件。

下一步 在代码窗口中为“Timer1”的 Timer 事件添加如下的代码。

程序 9.22　“Timer1”的 Timer 事件代码

```
IF THISFORM.lblTime.Caption != Time()
    THISFORM.lblTime.Caption = Time()          &&设置标签的属性为当前的时间
ENDIF
```

下一步 运行表单。

完成 关闭表单。

计时器控件常用的设计属性见表 9.24 所示。

表9.24　计时器控件常用属性说明

属性	属性说明
Enabled	当它的值为“.T.-真”时，表单在加载时计时器便开始工作。如果设置为“.F.-假”则需要由外部的事件启动计时器。
Interval	计时器事件的间隔的毫秒数

注意 计时器的时间间隔不能太短，否则处理器将占用大量的时间来处理响应事件，从而降低系统的效率。

9.4.12　使用图像和标签

图像　　标签

图像和标签分别用来添加图片和文本。图像控件在设计时可以动态地更改它的大小，也可以在程序运行时在事件或方法代码中对图像进行处理。

使用图像和标签的操作步骤如下：

开始　打开项目“使用控件”。

下一步　在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步　在“表单设计器”中向表单加入一个图像，一个标签。

下一步　在属性窗口设置属性如表 9.25 所示。

表9.25　图像和标签的属性设置

对象	属性	属性值
Label1	Caption	职员照片：
Label1	FontSize	16
Label1	FontName	隶书
Image1	Picture	..\samples\vastrade\bitmaps\buchstev.bmp
Image1	BorderStyle	1-固定单线

下一步　运行表单，如图 9.40 所示。

图9.40　标签与图像表单

说明　标签和文本框有许多不同的地方：

(1) 标签不具有数据源属性。

(2) 标签不能够被直接编辑。

(3) 标签不能用 Tab 键进行选择。

完成　关闭表单。

图像的常用属性有：

Picture：需要显示的.BMP 文件。

BorderStyle：图像是否具有可见边框。

Stretch：Stretch 为“0-剪裁”时，超出的图像部分将不显示。为“1-等比填充”时，图像保持原来的横纵比进行填充。为“1-变比填充”时，图像填充整个的图像区域。

在设计时标签常用的属性见表 9.26 所示。

表9.26　标签控件设计常用属性说明

属性	属性说明
Caption	在标签中显示的文本
AutoSize	标签是否自动根据标题调整大小
BackStyle	标签是否透明
WordWrap	标签上的文本是否可以换行

9.4.13　使用形状和线条

形状　线条

形状和线条用于可视化的将表单中的组件分类分组，也可以用于对界面的装饰。

使用形状和线条的操作步骤如下：

开始 打开项目“使用控件”。

下一步 在“项目管理器”中打开在 9.4.12 小节中建立的图像标签表单。

下一步 在“表单设计器”中向表单加入一个形状，一个线条。

下一步 在属性窗口设置属性如表 9.27 所示。

表9.27　形状和线条的属性设置

对象	属性	属性值
Image1	Stretch	2-变比填充
Line1	BorderColor	64,128,128
Shape1	BackColor	0,128，192
Shape1	Curvature	20
Label1	Alignment	2-中央

下一步 调整表单中的控件的位置，如图 9.41 所示。

下一步 将 Line1 的 LineWidth 属性设置为 5。

下一步 将 Shape1 的“Curvature”属性重新设置为 0。

下一步 将 SpecialEffect 属性设置为“0-3 维”，如图 9.42 所示。

图9.41　表单中加入形状和线条

图9.42　属性变化对表单的影响

完成 关闭表单。

直线控件在设计时常用的属性如表 9.28 所示。

表9.28　直线控件设计常用属性说明

属性	属性说明
BorderWidth	指定线宽
LineSlant	直线的倾斜方向，有效值为“\”或“/”

形状控件在设计时常用的属性如表 9.29 所示。

表9.29　形状控件设计常用属性说明

属性	属性说明
Curvature	0（直角）-99（圆或椭圆）的值
FillStyle	形状是透明的还是具有指定填充的方案
SpecialEffect	当 Curvature 为 0 时控件是二维还是三维的
WordWrap	标签上的文本是否可以换行

9.4.14　表单的图形方法

除了使用线条和形状以外，Visual FoxPro 还提供了图形化方法，如表 9.30 所示。

表9.30　表单的图形方法使用说明

属性	属性说明
Circle	在表单上画一个圆或圆弧
Cls	清除表单上的图文信息
Line	在表单上画一条直线
Pset	设置表单上的点为指定颜色
Print	在表单上打印字符串

9.4.15　使用表格

表格

表格是 Visual FoxPro 中最具价值的控件之一。它是一个容器对象，可以显示操作多行多列数据。它可以包含“列”，列具有“标头”和“控件”。

使用表格创建一对多表单的操作步骤如下：

说明　本例使用表格建立“一对多”表单，复习使用组合框，学习使用表格控件来浏览表中的数据，以及如何把组合框和表格相结合。

开始　新建一个项目“使用控件”。

下一步　在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步　设置表单的数据环境，在“数据环境”设计器中添加以前建立的“客户”表和“定单”表，如果它们没有建立关系，请建立“一对多”的关系。

下一步　在“表单设计器”中添加一个组合框、一个列表框、两个标签，它们的位置见图 9.43 所示。

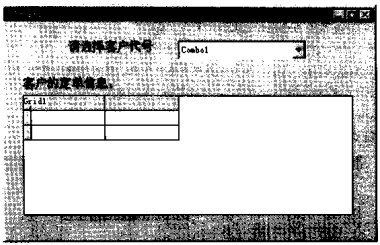


图9.43　加入组合框和表格的表单

说明　表格最为常用的用途是当父表中的记录发生变化时，表格中显示的子表的内容也随之变化。通常父表用文本框或组合框来显示相关的父记录。

要显示“一对多”表单，通常需要在数据环境中加入具有一对多关系的相关的表。此时可以将父表中作为选择的字段从数据环境中拖拽到表单中，将相关的子表直接拖拽到表单中，就可以直接建立一对多的表单。

下一步　布置控件，并在属性窗口修改“Combo1”和标签的属性如表 9.31 所示。

表9.31　组合框的属性设置

对象	属性	属性值
Combo1	RowSourceType	6-字段
Combo1	RowSource	客户.客户代号
Combo1	ControlSource	客户.客户代号
Combo1	Style	2-下拉列表框
Label1	Caption	请选择客户代号
Label2	Caption	客户的定单信息

下一步　布置控件，并在属性窗口修改“Grid1”的属性如表 9.32 所示。

表9.32　组合框的属性设置

对象	属性	属性值
Grid1	RecordSourceType	1-别名
Grid1	RecordSource	定单
Grid1	LinkMaster	客户
Grid1	ChildOrder	客户代号
Grid1	RevlationExpr	客户代号

说明　表格中的各项相关的属性设置按如下进行：RecordSource 设置为相关的子表的名称。把

LinkMaster 属性设置为主表名称。ChildOrder 属性设置为相关表的索引标识的名称，它和主表中关系表达式一致。将 RevlationEpr 设置为连接相关表和主表的表达式。

通常设置表格中要显示的表的数据源按照如下的步骤进行：

选择表格的 RecordSourceType 属性，如果需要用表格打开表，则将属性设置为“0-表”，如果在表中直接放入打开表的字段，将 RecordSourceType 属性设置为“1-别名”。

在属性窗口的 RecordSource 属性下输入数据源的表名或者表的别名。

第三步 运行表单，结果如图 9.44 所示。

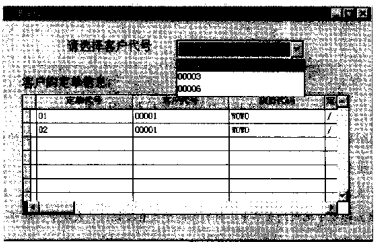


图9.44 运行中的一对多表单

第四步 关闭表单。

第五步 在表单的属性窗口中设置“ColumnCount”属性为“4”。

说明 “ColumnCount”属性为当前的表格窗口所需的列数，此时属性窗口的“Grid1”下面显示出当前的所有“列”。

第六步 在“属性窗口”中设置“Column4”的属性，此时表格的边框被圈起，表示编辑表格中的对象，设置 ControlSource 属性为“定单.是否发货”。

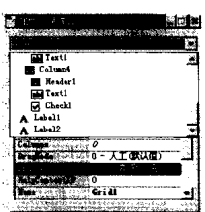


图9.45 表格列属性层次

说明 设置表格的列的数据源将在当前列中显示所选字段的内容。

第七步 在“表单控件”工具栏选择“复选框”，然后单击第四列，此时查看属性窗口，看对象的包含层次是否正确，如图 9.45 所示。

说明 表格的“列”中除了可以显示字段还可以嵌入控件，如文本框、复选框、组合框等控件。如果需要将这些加入的控件居中排列，可以先创建一个容器类，将控件放入容器类中，调整控件在容器类中的位置，最后将容器类加入到表格的相关列中，并将 ControlSource 数据源属性设置为需要的字段。如果加入的是一个复选框，那么应该将复选框的 Caption 属性设置为空，并将所属列的“Sparse”属性设置为“.F.-假”。

下一步 在属性窗口将“Check1”的“Caption”属性设置为空。

下一步 在属性窗口将“Column4”的“Sparse”属性设置为“.F.-假”。

下一步 在属性窗口将“Column4”的“ControlSource”属性设置为“定单.是否供货”。

下一步 在属性窗口将“Column4”的“CurrentControl”属性设置为“Check1”。

说明 如果需要在表格中移去列的控件，在属性窗口的“对象下拉列表框”中选择所要移去的属性，然后按 Delete 键。

下一步 在属性窗口中设置其他的对象属性如表 9.33 所示。

表9.33　列表框的列的属性设置

对象	属性	属性值
Column1	ControlSource	定单.定单代号
Column2	ControlSource	定单.船名
Column3	ControlSource	定单.运输路线
Column1.Header1	Caption	定单代号
Column2.Header1	Caption	船名
Column3.Header1	Caption	运输路线
Column4.Header1	Caption	是否供货
Column1	Width	80
Column2	Width	90
Column3	Width	140
Column4	Width	80

下一步 当设置标头的 Caption 和 Width 属性时注意表格的外观的变化。

下一步 在代码窗口的“Grid1”的“Init”事件加入如下代码。

程序 9.23　“Grid1”的“Init”事件代码

```
THIS.Column2.AddObject("cboShipName", "COMBOBOX")
THIS.Column2.CurrentControl = "cboShipName"
* 下面的代码确保控件在每一行都是可见的
* and is diplayed in every row in the grid
THIS.Column2. cboShipName.Visible = .T.
THIS.Column2.Sparse = .F.
```

下一步 运行表单，如图 9.46 所示。

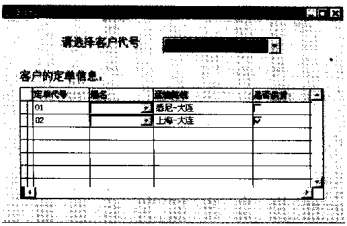


图9.46　表格中加入多个控件后的表格

说明　使用 AddObject 方法可以用代码向列中添加控件，列中的标头和控件不直接从列中继承属性，因此它们的属性需要一一设定。表格中的列可以设置特定的格式，如前景色背景色和输入的限制等。

完成　关闭表单。

在设计时常用的表格属性如表 9.34 所示。

表9.34　设计时常用的表格属性说明

属性	属性说明
ChildOrder	与主表主键相连的子表外来键
ColumnCount	列数，当设置为-1 时，表格中的列数和表的列数相同
LinkMaster	表格中所显示子表的父表
RecordSource	表格所显示数据的数据源
RecordSourceType	数据源的类型：可以是“0-表、1-别名、3-查询、4-SQL 说明”

在设计时常用的表格的列的属性如表 9.35 所示。

表9.35　设计时常用的表格列的属性说明

属性	属性说明
ControlSource	列中显示的数据源，通常为表中的字段
Sparse	当它的属性为“.T.-真”时，则只有在被选中的单元才显示控件，其他的单元将显示数据值。此时，用户在快速滚动表格中的数据时可以实现快速重画。
CurrentControl	当前表格中的活动控件。它的默认值为文本框“Text1”，当在列中加入了新的控件时，可以指定它的 CurrentControl 属性进行新的设置。

9.4.16　使控件的使用更加简单

表单中加入大量的控件后，如何让用户更为方便地操纵这些控件，也是一个值得仔细考虑的问题。除了适当的按功能为控件分组，进行控件的合理布局以外，还可以设置快速访问键、设置 Tab 次序、设置提示文本或者设置控件的启用废止状态等方法。

定制控件的访问操作步骤如下：

说明　通过本例，学习如何为控件设置快速访问键、如何设置控件的 Tab 顺序、如何设置控件的使用

提示文本，如何启用和废止菜单，如何允许设置拖放操作等多方面的内容，和以往一样，我们通过一个具体的实例进行学习。

开始 打开项目“使用控件”。

下一步 在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步 在“表单设计器”中向表单加入一个“标签”控件。

下一步 在“表单设计器”中向表单加入两个“图像”控件。

下一步 在“表单设计器”中向表单加入两个“命令按钮”控件。

下一步 在“表单设计器”中向表单加入一个“文本框”控件，然后调整表单的布局，设置好的表单布局如图 9.47 所示。

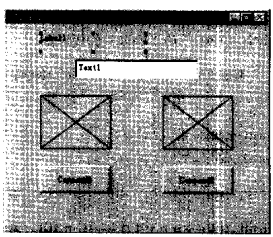


图9.47　表单控件的布置

下一步 在属性窗口中设置控件的属性，如表 9.36 所示。

表9.36　表单中控件的属性设置

对象	属性	属性值
Form1	Caption	定制表单控件
Label1	Caption	请输入“确定”：
Label1	FontSize	16
Label1	FontName	隶书
Image1	Stretch	2-变比填充
Image2	Stretch	2-变比填充
Image1	Picture	..\samples\tastrade\bitmaps\condimen.bmp
Image1	Picture	..\samples\tastrade\bitmaps\confect1.bmp
Command1	Caption	确定(\<Y)
Command1	Caption	退出(\<N)

说明 如果需要为表单中的控件设置快速访问键，只需要将控件的“Caption”属性设置中在需要作为访问键的字母的前面键入“\<”，而后在表单运行中可以按 ALT 键和快速字母来访问控件，例如按“ALT+Y”可以直接访问“确定按钮”。

下一步 从“显示”菜单选择“Tab 键次序”，单击控件“Text1”旁边的框，它在表单打开时具有最初的焦点。

下一步 按照需要的“Tab”键次序依次按“SHIFT+Click”。

表单中默认的控件的“Tab”键次序是控件加入表单的顺序。

下一步 在属性窗口设置“Text1”的“ToolTipText”属性为“输入“确定””。

下一步 在属性窗口设置“Form1”的“ShowTip”属性为“.T.-真”。

控件的“ToolTipText”属性设置以后，当用户的鼠标在控件上停留时，将自动显示“ToolTipText”属性指定的文本，同时表单的“ShowTips”属性决定是否显示用户设置的控件的工具提示文本。

下一步 在属性窗口设置“Command1”和“Command2”的“Enabled”属性为“.F.-假”。

下一步 在代码窗口设置“Text1”的“Valid”的代码如下。

程序 9.24　“Text1”的“Valid”的代码

```
IF THIS.Text="确定"
    THISFORM.Command1.Enabled=.T.
    THISFORM.Command2.Enabled=.T.
ENDIF
```

说明 如果需要设置按钮的启用或废止状态只要将按钮的“Enabled”属性设置为“.T.-真”或“.F.-假”，上面的代码在初始设置时命令按钮为“废止”状态，只有当用户在文本框中输入了“确定”时，命令按钮才成为启用状态。

下一步 在代码窗口的“Command1”和“Command2”的“Click”事件下加入如下的代码。

```
RELEASE THISFORM
```

下一步 保存表单于“表单 Define”，运行表单，如图 9.48 所示。

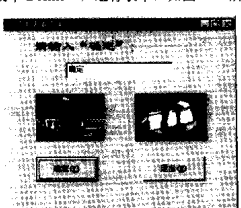


图9.48　定制表单中的控件

完成 关闭表单。

9.4.17　允许鼠标拖放

用户对于鼠标的拖放可能并不陌生，当拖拽时，Visual FoxPro 提供了相同的尺寸和灰色的外框。

关于鼠标拖拽的属性、事件和方法，如表 9.37 所示。

表9.37　设计时常用的表格列的属性说明

属性	属性说明
DragMode	确定拖拽方式是人工还是自动
DragIcon	当拖拽时的图标
DragDrop	何时将控件放到对象上
DragOver	识别对象拖过时的状态
Drag	开始和终止人工拖动

要理解拖动，首先要弄清拖动的“源”和“目标”：

源：指将被拖动的控件。

//标：将被拖拽到的对象，它可以是表单或可以识别 DragDrop 事件的控件。

实现鼠标的拖拽的操作步骤如下：

说明　通过本实例可以学习鼠标的拖放操作。

开始　打开项目“使用控件”。

下一步　在“项目管理器”中新建一个表单，进入“表单设计器”。

下一步　在“表单设计器”中向表单加入两个“列表框”控件，在加入两个标签。

下一步　在属性窗口设置控件的属性，如表 9.38 所示。

表9.38　表单中控件的属性设置

对象	属性	属性值
Label1	Caption	拖拽源：
Label2	Caption	拖拽目的：
Label1	FontSize	12
Label2	FontSize	12
List1	Name	lstSource
List2	Name	lstSelected
lstSource	DragIcon	..\samples\classes\grid\ dragmove.cur
lstSelected	DragIcon	..\samples\classes\grid\ nodrop01.cur
lstSource	MultiSelect	.T.-真
lstSelected	MultiSelect	.T.-真

下一步　在表单的“Init”事件加入如下的代码。

程序 9.25　表单的“Init”事件代码

```
#define ONE_LOC "第一项"
#define TWO_LOC "第二项"
#define THREE_LOC "第三项"
#define FOUR_LOC "第四项"
#define FIVE_LOC "第五项"
#define SIX_LOC "第六项"
#define SEVEN_LOC "第七项"
#define EIGHT_LOC "第八项"
#define NINE_LOC "第九项"
#define TEN_LOC "第十项"
* 向拖拽源添加所需的项
This.lstSource.AddItem (ONE_LOC)
This.lstSource.AddItem (TWO_LOC)
This.lstSource.AddItem(THREE_LOC)
This.lstSource.AddItem (FOUR_LOC)
This.lstSource.AddItem (FIVE_LOC)
This.lstSource.AddItem (SIX_LOC)
This.lstSource.AddItem (SEVEN_LOC)
This.lstSource.AddItem (EIGHT_LOC)
This.lstSource.AddItem (NINE_LOC)
This.lstSource.AddItem (TEN_LOC)
```

下一步　给表单添加两个新的属性“MouseX”，“MouseY”。

说明　用来计算鼠标所在的位置。

下一步　在代码窗口中设置“lstSource”的“DblClick”事件加入如下的代码。

程序 9.26　“lstSource”的“DblClick”事件代码

```
* 当在拖拽源列表框双击鼠标时从源列表中移去该项.
* 并在目的列表框加入该项
THIS.Parent.lstSelected.AddItem(This.List(This.ListIndex))
This.RemoveItem(This.ListIndex)
```

下一步　在代码窗口中设置“lstSource”的“KeyPress”事件加入如下的代码。

程序 9.27　“lstSource”的“KeyPress”事件代码

```
LPARAMETERS nKeyCode, nShiftAltCtrl
IF nKeyCode = 63 AND nShiftAltCtrl = 1
    THIS.Parent.SelectAll(THIS)
```

```
ENDIF
```

下一步 在代码窗口中设置“lstSource”的“MouseDown”事件加入如下的代码。

程序 9.28　“lstSource”的“MouseDown”事件代码

```
LPARAMETERS nButton, nShift, nXCoord, nYCoord          &&保存当前鼠标按下的位置
THIS.Parent.MouseX = nXCoord
THIS.Parent.MouseY = nYCoord
```

下一步 在代码窗口中设置“lstSource”的“DragDrop”事件加入如下的代码。

程序 9.29　“lstSource”的“DragDrop”事件代码

```
LPARAMETERS oSource, nXCoord, nYCoord
IF oSource.Name != THIS.Name                           &&是放到自己上么
    THISFORM.LockScreen = .T.
    nCnt = 1
    DO WHILE nCnt <= THIS.Parent.lstSelected.ListCount
        IF THIS.Parent.lstSelected.Selected(nCnt)   &&如果选择了一项，用代码控制将它进行移动
            THIS.Parent.lstSource.AddItem(THIS.Parent.lstSelected.List(nCnt))
            THIS.Parent.lstSelected.RemoveItem(nCnt)
        ELSE
            nCnt = nCnt + 1
        ENDIF
    ENDDO
    THISFORM.LockScreen = .F.
ENDIF
```

说明 “DragDrop”事件接受三个参数：oSource，nXCoord，nYCoord。其中 oSource 是对拖放到目的的空件对象的引用，nXCoord 和 nYCoord 为当前鼠标的水平和垂直坐标

下一步 在代码窗口中设置“lstSource”的“DragOver”事件加入如下的代码。

程序 9.30　“lstSource”的“DragOver”事件代码

```
LPARAMETERS oSource, nXCoord, nYCoord, nState
DO CASE
    CASE nState = 0                                    && 进入
        oSource.DragIcon = "d:\program files\devstudio\vfp\samples\classes\dragmove.cur"
    CASE nState = 1                                    && 离开
        oSource.DragIcon = "d:\program files\devstudio\vfp\samples\classes\nodrop01.cur"
ENDCASE
```

说明　当用户进行拖拽时，可以定义哪些位置可以释放控件哪些地方不能执行释放操作。在“DragDrop”事件中可以通过改变 DragIcon 的属性实现可视的提示用户当前的拖拽的区域的域属性。

在代码窗口中设置“lstSource”的“MouseMove”事件加入如下的代码。

程序 9.31　“lstSource”的“MouseMove”事件代码

```
LPARAMETERS nButton, nShift, nXCoord, nYCoord
IF nButton = 1                                          && 鼠标左键
   IF ABS(nXCoord - THIS.Parent.MouseX) > 8 OR ;
      ABS(nYCoord - THIS.Parent.MouseY) > 8
      THIS.Drag
   ENDIF
ENDIF
```

说明　Visual FoxPro 的 DragMode 属性有两种方式，一种是“0-人工”，另一种是“1-自动”。当使用“0-人工”设置时，可以控制什么时候可以进行拖拽。“0-人工”控制允许在开始识别一个 MouseDown 事件，从而记录鼠标的位置。

从代码中进行拖动，需要将 DragMode 属性设置为“0-人工”，然后用 Drag 方法，语法为

container.control.Drag(*nAction*)

在代码窗口中设置“lstSelected”的“DblClick”事件加入如下的代码。

程序 9.32　“lstSelected”的“DblClick”事件代码

```
* 当在拖拽源列表框双击鼠标时从源列表中移去该项，
* 并在目的列表框加入该项
THIS.Parent.lstSource.AddItem(This.List(This.ListIndex))
This.RemoveItem(This.ListIndex)
```

在代码窗口中设置“lstSelected”的“KeyPress”事件加入如下的代码。

程序 9.33　“lstSelected”的“KeyPress”事件代码

```
LPARAMETERS nKeyCode, nShiftAltCtrl
IF nKeyCode = 63 AND nShiftAltCtrl = 1
THIS.Parent.SelectAll(THIS)
ENDIF
```

在代码窗口中设置“lstSelected”的“MouseDown”事件加入如下的代码。

程序 9.34　“lstSelected”的“MouseDown”事件代码

```
LPARAMETERS nButton, nShift, nXCoord, nYCoord                    &&保存当前鼠标按下的位置
THIS.Parent.MouseX = nXCoord
THIS.Parent.MouseY = nYCoord
```

下一步 在代码窗口中设置“lstSelected”的“DragDrop”事件加入如下的代码。

程序 9.35　“lstSelected”的“DragDrop”事件代码

```
LPARAMETERS oSource, nXCoord, nYCoord
IF oSource.Name != THIS.Name
    THISFORM.LockScreen = .T.
    * ListCount 的值在 RemoveItem 方法被调用时发生变化
    nCnt = 1
    DO WHILE nCnt <= THIS.Parent.lstSource.ListCount
        IF THIS.Parent.lstSource.Selected(nCnt)
            THIS.Parent.lstSelected.AddItem(THIS.Parent.lstSource.List(nCnt))
            THIS.Parent.lstSource.RemoveItem(nCnt)
        ELSE
            nCnt = nCnt + 1
        ENDIF
    ENDDO
    THISFORM.LockScreen = .F.
ENDIF
```

下一步 在代码窗口中设置“lstSelected”的“DragOver”事件加入如下的代码。

程序 9.35　“lstSelected”的“DragOver”事件代码

```
LPARAMETERS oSource, nXCoord, nYCoord, nState
DO CASE
    CASE nState = 0                                              && 进入
        oSource.DragIcon = "d:\program files\devstudio\vfp\samples\classes\dragmove.cur"
    CASE nState = 1                                              && 离开
        oSource.DragIcon = "d:\program files\devstudio\vfp\samples\classes\nodrop01.cur"
ENDCASE
```

下一步 在代码窗口中设置“lstSelected”的“MouseMove”事件加入如下的代码。

程序 9.36　“lstSelected”的“MouseMove”事件代码

```
LPARAMETERS nButton, nShift, nXCoord, nYCoord
IF nButton = 1                                                   && 鼠标左键
```

```
        IF ABS(nXCoord - THIS.Parent.MouseX) > 8 OR ;
            ABS(nYCoord - THIS.Parent.MouseY) > 8
            THIS.Drag
        ENDIF
    ENDIF
```

下一步 关闭代码窗口，运行表单，如图9.49所示。

下一步 在“拖拽源”列表框双击列表项，该项自动加到“拖拽目的”列表框。

下一步 从“拖拽源”列表框拖拽列表项到“拖拽目的”列表框。

下一步 从“拖拽目的”列表框拖拽列表项到“拖拽源”列表框。

下一步 从“拖拽源”列表框按“SHIFT”键选择多项拖拽到“拖拽目的”列表框。

完成 关闭表单。

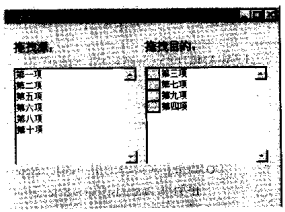


图9.49　执行拖拽的表单

9.4.18　使用页框

Visual FoxPro 中页框是包含了页面的容器。页框、页面和控件之间的关系如图 9.50 所示。

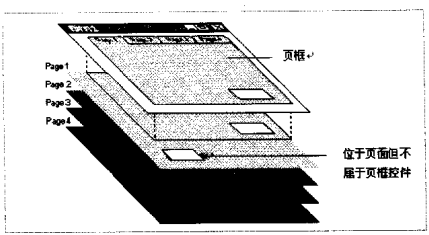


图9.50　页面上的页框容器控件

从图 9.50 可以看出，页框可以看作页面的扩展，它有一定的位置。页框的主要作用是创建带有选项卡的的表单和对话框。

使用页框的操作步骤如下：

说明 通过本实例可以学习页框、OLE 容器和 OLE 绑定控件的使用。

开始 打开项目“使用控件”。

下一步 在“项目管理器”中新建一个表单，进入“表单设计器”。

第三步　在“表单设计器”中向表单加入一个“页框”控件，两个命令按钮。

第四步　在属性窗口设置页框的“PageCount”属性为3。

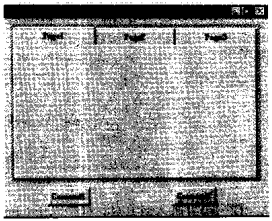


图9.51　具有三个页面的页框

说明　页框的“PageCount”属性决定了页框的页面数量。

第五步　从“表单设计器”选择页框，然后从快捷键选择“编辑”，此时页框的边缘发生变化，它被激活为容器控件，如图9.51所示。

如果向页框中添加控件，必须首先选择控件，然后通过快捷键或者从属性窗口的对象列表中选择页框容器内包含的页面，使页框的边缘出现虚框，才能向页框中添加相应的控件。

第六步　单击选项卡的标题，可以选择不同的页面，也可以在属性窗口选择相应的对象。

第七步　在属性窗口选择“Page1”将“Caption”属性设置为“这是一个选项卡的长标题。

第八步　在属性窗口将“Pageframe1”的“TabStretch”属性设置为“1-单行”。

说明　当选项卡的标题太长时，可以设置页框的“TabStretch”属性来决定选项卡的标题显示形式，页框的“TabStretch”属性有两种选择它们的含义如下：

多重行：选项卡将重叠起来，从而显示所有的选项卡的标题内容。

单行：这是默认值，此时选项卡的标题将被剪裁，从而选项卡的标题只显示一部分内容。

第九步　在属性窗口选择“Pageframe1.Page1”，并向此页面加入一个命令按钮。

第十步　在属性窗口设置各个对象的属性如表9.39所示。

表9.39　页框表单的属性设置

对象	属性	属性值
Command1	Caption	确定
Command2	Caption	退出
Page1.Command1	Caption	进入 OLE 容器页面
Page2	Caption	OLE 容器
Page2	Caption	OLE 绑定控件

第十一步　进入代码窗口，在“Page1.Command1”的“Click”事件下加入如下的代码。

```
THISFORM.Pageframe1.ActivePage=2
```

第十二步　在代码窗口的“Pageframe1.Command1”和“Pageframe1.Command2”对象的“Click”事件下加入如下的代码。

```
RELEASE THISFORM
```

第十三步　选择页面“Page2”，从“表单控件”工具栏选择“OLE 容器控件”按钮，在页面“Page2”上点击鼠标，此时出现插入对象对话框，见图9.53所示。

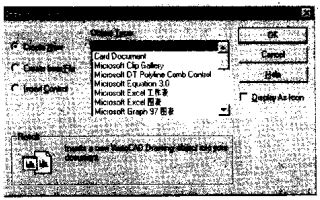


图9.52　插入对象对话框

下一步 在插入对象对话框中选择“Insert Control”，插入“Calender Control”并调整它的大小，此时的表单页面“Page2”见图9.53所示。

说明 随着面向对象技术的成熟，软件行业向构件化发展。作为适应这一趋势的成熟技术 OLE，具有很大的实用价值。Visual FoxPro6.0 中的“OLE 容器控件”可以包容大量的 OLE 对象，也可以根据需要加入所有注册的“.OCX”控件。

下一步 从快捷键进入“数据环境设计器”，向数据环境中添加具有通用型字段（Generral）的数据表，如存储有雇员照片的雇员表。

下一步 选择页面“Page3”，从“表单控件”工具栏选择“OLE 绑定型控件”按钮，在页面“Page3”上点击鼠标，页面上出现“OLE 绑定型控件”，根据需要调整它的大小，在“Page3”上加入一个标签控件，将它的“Caption”属性设置为“雇员照片”并修改字体属性，如图 9.54 所示。

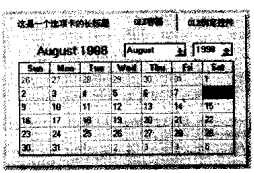


图9.53　加入日历OCX控件的页面

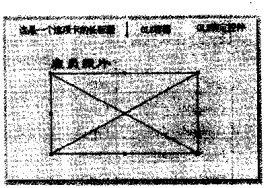


图9.54　加入OLE绑定型控件的页面

下一步 在属性窗口设置“OLEBoundControl1”的“ControlSource”属性设置为“雇员.雇员照片”。

说明 “OLE 绑定型控件”可以将表中的通用型字段在表单中显示出来。要将通用型字段和“OLE 绑定型控件”相连，只要设置“OLE 绑定型控件”的“ControlSource”属性设置为表的通用型字段。

完成 关闭表单。

9.5　小　结

本章介绍了如何设计一个良好的用户界面，Visual FoxPro 6.0 最为常见的用于显示和

编辑的界面是表单。然而，对整个应用程序的导航设计只有表单是不够的。对大多数用户而言，首先见到的是菜单，在菜单的导航支持下才进入一个个的表单或对话框，在对话框中，又可以通过各种控件来实现各种功能。

本章介绍了以下内容：菜单系统的设计、为应用程序创建工具栏、使表单协同工作、如何使用控件。

思考与练习

1. 请自己设计一个菜单，并按照9.1介绍的方法在“菜单设计器”中建立菜单？
2. 如何创建多个用户的工具栏？怎样将工具栏和菜单协调工作？
3. 表单和表单集的应用范围是什么？表单有哪几种？各有什么特点？
4. 绑定型控件和非绑定型控件有什么区别？
5. 在表单中可以使用的控件有哪些？如何使用？
6. 页框、页面和控件之间有什么关系？如何使用页框？

第10章　VFP 6.0 事件驱动模型

通过 Visual FoxPro 6.0 功能完善的事件处理机制，用户能十分方便地实现对操作的响应。目前的应用程序大多以与用户交互的方式进行工作，Windows 的事件驱动程序的设计方法，正是为了适合用户对交互功能的要求。

在传统的程序设计中，应用程序的执行是面向过程的顺序过程，程序的运行往往独占系统的资源。而在事件模型中，任务的执行是对事件的响应，应用程序始终在运行中等待事件的发生，然后执行相关的操作。整个的事件的循环由 Windows 操作系统本身来控制，因而可以实现多任务的并行操作，多个程序同时启动，从而实现用户的资源的外围设备的共享。

在本章主要介绍：

（1）在Visual FoxPro中使用事件；

（2）Visual FoxPro中事件的发生顺序；

（3）Visual FoxPro的事件的分类。

10.1　在Visual FoxPro中使用事件

10.1.1　Visual FoxPro中的核心事件

在 Visual FoxPro 中，当用户进行键盘或鼠标操作，从而触发一系列事件的发生，执行相关的代码。

Visual FoxPro 中的常用的事件如表 10.1 所示。

表10.1　Visual FoxPro的核心事件集

事件名称	触发操作
GotFocus	对象接受焦点时，如单击鼠标，使用 Tab 键或者在代码中时用 SetFocus 方法。
LostFocus	对象失去焦点时，如在其他控件对象单击鼠标，使用 Tab 键或者在代码中使用其他对象的 SetFocus 方法。
KeyPress	按下键盘键
MouseDown	在一个对象上按下鼠标键
MouseMove	在一个用户对象上移动鼠标
MouseUp	在一个对象上释放鼠标键

10.1.2　容器和对象的事件

当在表单控件上进行移动鼠标、单击鼠标等操作时，该对象将独立地接受自己的鼠标事件。

注意　对控件的事件作出响应，Visual FoxPro 有如下的规则：

(1) 容器对象不处理它本身所包含的对象所发生的事件。

(2) 当控件本身不对事件作出处理时，Visual FoxPro 将在类层次上检测是否具有相关的处理代码。

容器与对象事件响应的操作步骤如下：

开始　从“项目管理器”新建一个项目“事件模型”。

下一步　在“项目管理器”中新建一个表单。

下一步　在“表单设计器”中向表单加入两个命令按钮。

下一步　在属性窗口设置两个命令按钮的“Caption”属性，如图 10.1 所示。

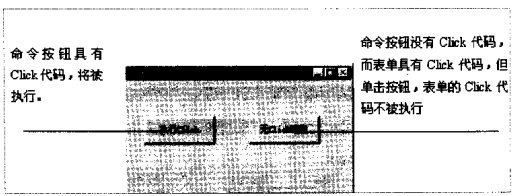


图10.1　对Click事件的响应

下一步　在“代码窗口”中的“Command1”的“Click”事件加入如下的代码。

```
Messagebox("Click 事件发生了!")
```

下一步　在“代码窗口”中的“Form1”的“Click”事件加入如下的代码。

```
RELEASE THISFORM
```

下一步　运行表单，先单击“执行 Click”按钮，然后单击“无 Click 响应”按钮，最后单击表单。

说明　通过上面的操作，可以了解容器和对象的事件响应的相互对立的原则，这条原则也同样适用于表格控件。

下一步　新建一个表单，向表单中加入一个命令按钮组控件，并设置它的命令按钮的“Caption”属性，如图 10.2 所示。

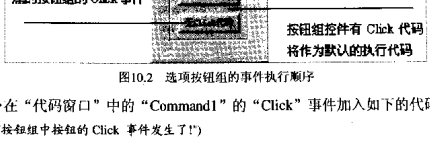


图10.2　选项按钮组的事件执行顺序

下一步　在“代码窗口”中的“Command1”的“Click”事件加入如下的代码。

```
Messagebox("按钮组中按钮的 Click 事件发生了!")
```

下一步 在“代码窗口”中的“Commandgroup1”的“Click”事件加入如下的代码。

```
Messagebox("按钮组的 Click 事件发生了!")
```

下一步 在“代码窗口”中的“Form1”的“Click”事件加入如下的代码。

```
RELEASE THISFORM
```

完成 运行表单，先单击“有 Click 代码”按钮，然后单击“无 Click 代码”按钮，，然后单击命令按钮组，最后单击表单。

说明 对于命令按钮组或者选项按钮组，如果组中的对象没有响应的代码，它将执行组中的相应的事件代码，因此组中的代码是默认的代码响应。

在“表格-列-标头或控件”的三层关系中，只有和事件相关连的对象才处理它的代码，如图 10.3 所示，为对“MouseMove”响应代码所在的对象。

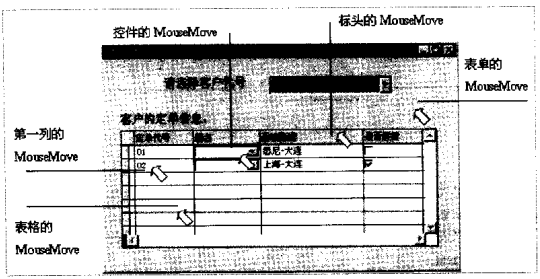


图10.3　表格的事件响应

10.1.3　类与控件的事件

用户常常定义自己的类，然后将它加入表单中。由于类的继承性，表单中的基于类的控件，如果没有添加事件代码，则将按照从底到高的顺序检查父类，直到遇到相关的父类代码并执行它。这种响应的关系如图 10.4 所示。

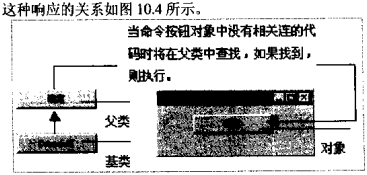


图10.4　类和控件的事件响应关系

10.2　Visual FoxPro的事件发生顺序

10.2.1　跟踪事件的发生顺序

Visual FoxPro 中的事件有些发生的顺序是固定的，如表单的“Load”和“Destroy”事件，有些事件的发生顺序是独立的，如“Click”事件。大多数的事件是由用户的交互操作引发的一系列事件中的一个。

在 Visual FoxPro 中的调试窗口，可以跟踪事件的发生顺序，充分理解事件的发生顺序有助于合理的安排任务代码。

跟踪事件发生顺序的操作步骤如下：

开始 从“工具”菜单选择“调试器”，进入调试窗口。

下一步 在“调试器”窗口选择“工具”菜单，然后选择“事件跟踪”。

下一步 在“事件跟踪”窗口中选择“开启事件跟踪”选项，如图 10.5 所示。

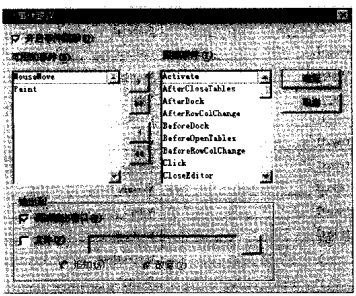


图10.5　设置事件的跟踪顺序

下一步 从“窗口”菜单选择“输出”，出现“调试输出”窗口。

下一步 从“调试”菜单选择“运行”。

下一步 从“运行”对话框中选择一个需要运行的表单。

下一步 操作表单，并在“调试输出”窗口中查看事件的发生顺序。

完成 关闭表单和“调试窗口”。

10.2.2　Visual FoxPro 的事件发生顺序

Visual FoxPro 中的事件很多发生的顺序是固定的，当一个表单的“数据环境”的 AutoOpenTables 属性设置为“.T.-真”时，一些事件的发生顺序如表 10.2 所示。而其他事件的发生顺序和用户的操作有关。

表10.2　Visual FoxPro的事件的发生顺序

对象	发生事件
数据环境（Data environment）	BeforeOpenTables
表单集（Form set）	Load
表单（Form）	Load
数据环境游标（Data environment cursor(s)）	Init
数据环境（Data environment）	Init
表单中的对象（Objects[1]）	Init
表单（Form）	Init
表单集（Form set）	Init
表单集（Form set）	Activate
表单（Form）	Activate
表单中的对象1（Object1[2]）	When
表单（Form）	GotFocus
表单中的对象1（Object1）	GotFocus
表单中的对象1（Object1）	Message
表单中的对象1（Object1）	Valid[3]
表单中的对象1（Object1）	LostFocus
表单中的对象2（Object2[3]）	When
表单中的对象2（Object2）	GotFocus
表单中的对象2（Object2）	Message
表单中的对象2（Object2）	Valid[4]
表单中的对象2（Object2）	LostFocus
表单（Form）	QueryUnload
表单（Form）	Destroy
表单中的对象5（Object[5]）	Destroy
表单（Form）	Unload
表单集（Form set）	Unload
数据环境（Data environment）	AfterCloseTables
数据环境（Data environment）	Destroy
数据环境游标（Data environment cursor(s)）	Destroy

说　明　以上的上角标的含义如下：

Objects[1]：所有的内部对象和外部的容器对象。

Object1[2]：Tab 顺序排在第一个的对象。

Object2[3]：下一个获得焦点的对象。

Valid[4]：当对象失去焦点时。

Object[5]：所有的容器和非容器对象。

10.2.3　给事件添加代码

你可以为每一个 Visual FoxPro 中的事件添加代码，也可以所有的事件都没有代码响应。但多数情况下是为特定的事件添加特定的代码。

Visual FoxPro 中事件的发生顺序决定了添加代码的一些规则，在添加代码时，有如下的规则可以遵循：

在表单中的控件的Init事件发生于表单控件的Init事件之前，从而可以在表单的Init事件中加入对控件的必要的初始化操作代码。

列表框、组合框或者复选框值发生变化时，将发生InteractiveChange事件，而不管值是否发生变化都可能引发Click事件。

当你拖拽控件时，一些鼠标控件的事件可能不发生，例如MouseUp和MouseMove事件在进行拖拽和释放操作时将不发生。

Valid和When事件将返回一个值，默认值为“.T.-真”。如果你从When事件返回一个“.F.”或“0”值，控件将无法得到焦点，如果从Valid事件返回一个“.F.”或“0”值，控件将无法失去焦点。

10.3　Visual FoxPro的事件分类

10.3.1　鼠标事件

Visual FoxPro 中的鼠标事件是最为常见的事件，表 10.3 列出了常用的鼠标事件和对事件应用的说明。

表10.3　Visual FoxPro的鼠标事件

事件	事件说明
Click	鼠标放在控件上，按下并释放鼠标左键，更改控件的特定的值。使用程序触发 Click 事件的代码。在表单上单击鼠标时发生。
DblClick	两次快速的单击鼠标的左键时发生，当从列表框或组合框选择选项并按“ENTER”键时也发生此事件。
MouseDown 和 MouseUp	鼠标按下时发生 MouseDown 事件，鼠标释放时发生 MouseUp 事件。通过对参数的判别，MouseDown 和 MouseUp 事件可以区分鼠标的左、中、右键，还可以判断 SHIFT、CTRL 和 ALT 键与鼠标的组合。
MouseMove	当用户在对象上移动鼠标时发生。
DragDrop	当用户完成鼠标的拖放操作，释放鼠标时发生。在此事件可以加入对象的移动代码。
DragOver	鼠标拖动对象移动过当前的对象时发生。用户可以通过此事件，监控鼠标进入、离开或经过对象的操作。
DropDown	单击组合框的下拉箭头，列表框下拉前发生。此时可以使用 AddItem 和 RemoveItem 对组合框进行更新。

10.3.2 键盘事件

Visual FoxPro 中的键盘事件扩展了用户的操作，常用的键盘事件为：

KeyPress：当用户按下释放键盘键时发生此事件。它常常用来获得用户输入到控件的输入，从而判断键盘输入的有效性。

10.3.3 焦点事件

所谓的“焦点”（Focus）就是当前操作的对象的状态。Visual FoxPro 中的焦点事件如表 10.4 所示。

表10.4　Visual FoxPro的焦点事件

事件	事件说明
When	控件在接受焦点以前发生此事件。当 When 事件的返回值为 ".T." 时，默认的控件将接受焦点，否则控件没有接到焦点。
GotFocus	获得焦点事件 GotFocus 当用户通过操作或者通过程序代码时控件对象得到焦点时发生。只有当控件的 Enabled 属性和 Visible 属性为 ".T." 时，对象才触发 GotFocus 事件。
Valid	在控件失去焦点以前发生。Valid 事件有一个返回值，当它的返回值为 ".T." 时表示对象已经失去了焦点，当返回值为 ".F." 时表示控件没有失去焦点。
LostFocus	失去焦点事件 LostFocus 当用户失去焦点时发生，它的发生顺序由控件的类型决定。

10.3.4 表单事件

Visual FoxPro 中常用的表单事件如表 10.5 所示。

表10.5　Visual FoxPro的表单事件

事件	事件说明
Load	Load 事件发生于创建表单或表单集之前发生，它的发生顺序见表 10.2.1 所示。它通常加入用于初始化的代码。
Activate	当表单、表单集或页面对象被激活时，发生该事件，它的发生顺序和控件的类型有关。
Deactivate	当容器对象中包含的对象失去了焦点，而使得当前容器处于了非活动状态时，发生此事件。当表单卸载时不发生 Deactivate 事件，而只有在当前应用程序中移动焦点时，才触发 Activate 和 Deactivate 事件。
Unload	表单集或表单释放的时候发生此事件，它发生在所有包含的对象被释放和 Destroy 事件之后。
Paint	表单或工具栏发生重画时（如调整大小、移动）发生。
Resize	对象的大小发生变化时发生。

10.3.5　其他常用的事件

Visual FoxPro 中其他常用的事件如表 10.6 所示。

表10.6　Visual FoxPro其他常用事件

事件	事件说明
Init	容器对象的 Init 事件发生于容器中包含的对象的 Init 事件之后，从而允许在容器对象的 Init 事件中对它所包含的对象进行处理。
Destroy	当释放一个对象的实例时发生。容器的 Destroy 事件发生于所有的它所包含的对象的 Destroy 事件之前
Timer	每间隔 Interval 所指定的时间间隔发生此事件。
Error	当发生错误时发生此事件，可以在此事件加入错误处理代码。
InteractiveChange	当用键盘或鼠标更改控件的值时发生此事件。

10.4　小　结

本章介绍了 Visual FoxPro 功能完善的事件处理机制，通过它用户能十分方便的实现对操作的响应。在事件模型中，任务的执行是对事件的响应，应用程序始终在运行中等待事件的发生，然后执行相关的操作。整个的事件的循环由 Windows 操作系统本身来控制，因而可以实现多任务的并行操作，多个程序同时启动，从而实现用户的资源的外围设备的共享。

本章共介绍了如下内容：在 Visual FoxPro 中使用事件、Visual FoxPro 中事件的发生顺序、Visual FoxPro 的事件的分类。

思考与练习

1．Visual FoxPro 中的鼠标和键盘的核心事件有哪些？它们将触发哪些操作？
2．表格中控件的事件响应顺序是怎样的？
3．如何跟踪事件的发生顺序？Visual FoxPro 中事件的发生顺序是怎样的？
4．Visual FoxPro 中的事件分为哪几类？分别有哪些？

第11章　VFP 6.0 应用系统的集成和测试

在前面几章已经详细的论述了构造应用程序所需要的所有要素。利用这些要素条件，可以利用 Visual FoxPro 提供的功能强大的向导功能，你可以十分方便地建立应用程序的框架，并为这个框架创建需要的界面和所需要的数据，通过查询和报表使用户得到必要的数据，最后可以将整个的应用程序编译通过形成最终的程序。在编译和创建过程中，用户的每一个组件都需要进行严格的测试和调试，从而保证程序运行的正确性。Visual FoxPro6.0 功能强大的调试器可以帮助用户方便的完成调试。为了使你的应用程序变的小巧、高效，在应用程序构造当中要考虑到程序的优化。

在本章主要介绍：

（1）应用程序框架的构造；

（2）应用程序的测试与调试；

（3）应用程序的优化。

11.1　应用程序框架的构造

11.1.1　Visual FoxPro应用程序结构

一个典型的 Visual FoxPro 应用程序的结构如图 11.1 所示。

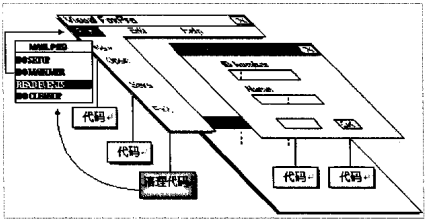


图11.1　Visual FoxPro应用程序的典型结构

从图 11.1 中可见，一个典型的 Visual FoxPro 应用程序由数据库结构、应用程序界面、报表和查询等部分组成。它具有一个程序的主控菜单，必要时要有各级的子菜单。提供表单，用户可以方便地浏览和编辑数据，对表单或菜单中的控件或命令的相关事件加入事件的处理代码，完成相关的任务。提供查询和报表，用户能方便地从数据库中提取相关的信息。所有的这些组成部分集成到一个程序框架当中，并为整个的程序设置一个起始点用于程序的启动。

11.1.2　建立设置应用程序主程序

每一个应用程序都需要设置一个主文件作为应用程序的起始点，这个主程序可以是一个程序、一个表单或者一个查询。

最常见的是用一个主程序去调用程序框架的各部分组件，从而实现对整个应用程序的控制。

主程序通常由如下的几个部分顺序操作完成：

开始 设置整个应用程序的执行环境。

下一步 确定应用程序的初始界面。

下一步 用 READ EVENTS 事件建立对程序事件的响应循环。

下一步 从一个命令按钮或菜单命令执行 CLEAR EVENTS 命令。

完成 退出应用程序，恢复整个程序的执行环境。

注意 如果使用应用程序向导建立一个主程序，可以对创建的主程序进行修改，而不必建立新的主程序。

建立设置应用程序的主程序的操作步骤如下：

开始 从“项目管理器”新建一个项目“应用程序”。

下一步 从“代码”选项卡选择“程序”，然后按“新建”。

下一步 在“代码”窗口中输入如下的代码。

程序 11.1　初始化程序

```
DO SETUP.PRG                          &&设置当前应用程序执行环境
DO MAINMENU.MPR                       &&显示菜单建立初始应用程序界面
READ EVENTS                           &&建立一个事件循环
```

注意 必须在另外的程序中，如 MAINMENU.MPR 中执行 LEAR EVENTS 命令。

```
DO CLEANUP.PRG                        &&恢复环境退出程序
```

下一步 保存应用程序，文件名为“Main.PRG”。

完成 从主菜单中选择“Main.PRG”文件，从“项目”菜单中查看“设置主文件”菜单项，它的前面有使用标记，说明它已经是一个主程序，如图 11.2 所示。

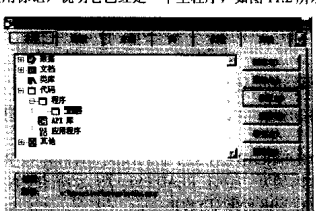


图11.2　设置一个文件为主文件

说明 在每一个项目文件中，只能有一个文件可以作为主文件，它在项目管理器中用粗体字显示出来。

11.1.3 设置应用程序的执行环境

因为 Visual FoxPro 的默认的开发环境并不是应用程序的最佳执行环境，应用程序的主文件建立以后，要建立一个文件用来保存原来的环境并设置应用程序的环境。

为应用程序建立环境的程序通常有以下的几部分：

（1）保存原来的环境设置；

（2）初始化程序变量；

（3）设置默认的路径；

（4）设置外部的库文件和需要的过程文件。

例如当需要用 SET TALK 命令改变 TALK 设置为"OFF"，可以用下面的代码：

程序 11.2 用 SET TALK 命令

```
IF SET('TALK') = "ON"                    &&如果原来设置为 "ON"
    SET TALK OFF                         &&关闭 TALK
    cTalkVal = "ON"                      &&保存原来设置
ELSE
    cTalkVal = "OFF"                     &&否则设置不变
ENDIF
```

11.1.4 显示初始的程序界面

Visual FoxPro 中的初始程序界面可以是一个登录窗口、一个主菜单、一个表单或其他的组件。

如果将主表单作为应用程序的启动界面，可以按如下的步骤进行：

开始 在主表单的 Load 事件中加入设置初始环境的代码。

下一步 建立主表单的数据环境。

下一步 建立主表单的事件响应代码。

完成 在主表单的 Unload 事件下加入代码恢复环境设置。

在程序的主程序中用"DO"命令可以启动程序的主菜单程序，用"DO FORM"命令可以启动表单程序。

11.1.5 控制事件的循环

设置好应用程序的环境，初始化初始界面后，需要一个事件循环对用户的操作作出响应。

Visual FoxPro 中使用 READ EVENTS 命令开始对事件的响应。READ EVENTS 命令在主程序的位置十分重要，因为一旦执行 READ EVENTS 命令所有主程序中的文件将处于"挂起"状态，直到执行 CLEAR EVENTS 为止。

当事件循环开始以后，应用程序将控制权交给最后一次的显示界面来处理。例如在主程序中包含了下面的两条命令，应用程序将显示表单 STARTUP.SCX，并将控制权交给它。

程序 11.3　执行表单

```
DO FORM STARTUP.SCX
READ EVENTS
```

使应用程序退出事件循环用 CLEAR EVENTS 命令。通常是将 CLEAR EVENTS 命令放在主菜单或者在一个表单的命令按钮中。CLEAR EVENTS 命令将中断 Visual FoxPro 的事件循环，并返回到执行 READ EVENTS 命令行的位置。

在你开始事件循环之前需要考虑一种退出循环的方式，要确信在你的应用程序界面中有“退出”的位置并执行了 CLEAR EVENTS 命令。

11.1.6　恢复应用程序的初始环境

使用宏替换并结合 SET 命令，用户可以方便地实现对原有的变量环境的恢复。例如，如果将原来的 SET TALK 设置为一个全局变量 cTalkVal，则执行下面的命令可以恢复原来的环境。

```
SET TALK &cTalkval
```

宏替换的变量名不能具有“M.”前缀，因为“.”代表了变量的连接，将产生语法错误。

当用户的环境设置和环境恢复的代码不在同一个过程文件时，要确保恢复环境文件能够获得以前保存的环境变量的值。通常是将这些参数保存于全局变量、用户自定义类或者作为应用程序对象的一个属性。

11.1.7　建立编译应用程序

Visual FoxPro 的应用程序向导为我们建立一个应用程序框架提供了强有力的工具。通过它可以方便地建立一个包括项目文件、数据表、报表、表单和特定的类库的应用。

通过运行应用程序向导，建立一个完整的应用程序框架，然后做必要的修改进行客户化的工作。

完整的运行应用程序向导通常包括以下几部分：

选定项目位置（Choose Project Location）。

选择数据库（Choose Database）。

选择文档（Choose Documents）。

定制菜单（Configure Menu）。

完成（Finish）。

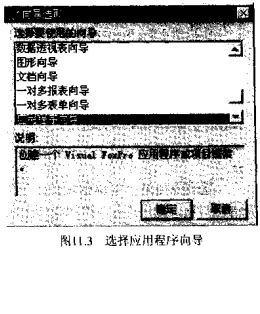


图11.3　选择应用程序向导

用向导建立并编译应用程序的操作步骤如下：

开始　从“工具”菜单选择“全部”，进入“向导选取”窗口，选择“应用程序向导”，如图 11.3 所示。

下一步　按“确定”。

下一步 在“选定项目位置”窗口中选择“项目类型”的“仅生成框架”。

下一步 在“项目名”框输入项目的名称，在“项目文件”框输入项目文件名和路径，见图 11.4 所示。

下一步 按“下一步”。

下一步 在“完成”窗口中选择“完成”。

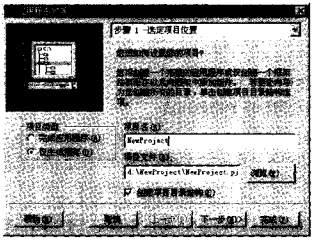


图11.4　选定项目位置

下一步 从“文件”菜单中选择“打开”，打开刚才建立的项目文件。

下一步 从“项目管理器”的“全部”选项卡可以展开向导为当前的项目创建的框架程序，如图 11.5 所示。

下一步 从“项目”菜单，选择“连编”菜单项。

下一步 在“连编选项”对话框中选择“重新连编项目”，并选择“重新编译所有文件”和“显示错误信息”对话框。

下一步 选择“确定”。

说明　项目可以编译成为“.EXE”文件和“.APP”文件两种。如果用户具有一个 Visual FoxPro 环境，可以执行“.APP”文件，否则需要执行“.EXE”文件。如果运行“.EXE”文件用户需要具有两个 Visual FoxPro 的动态链接库(VFP500.DLL 和 VFPxxx.DLL)从而为整个的程序提供一个 Visual FoxPro 的运行环境。

下一步 如果“重新连编项目”没有产生错误信息，再次进入“连编选项”对话框，选择“连编应用程序”。

下一步 选择“确定”，形成

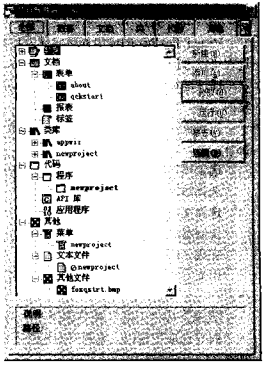


图11.5　用向导产生的框架程序包含的文件

NEWPROJECT.APP 文件。

下一步 在“命令窗口”输入以下命令。

```
do newproject.app
```

下一步 退出应用程序文件。

下一步 从“项目管理器”选择“连编”，进入“连编选项”对话框，选择“连编可执行程序”。

下一步 选择“版本...”按钮。

下一步 在“EXE 版本窗口”设置版本信息，如图 11.6 所示。

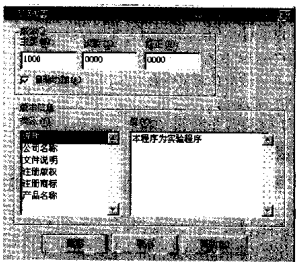


图11.6 设置EXE文件的版本信息

下一步 选择“确定”，形成 NEWPROJECT.EXE 文件。

下一步 在“命令窗口”输入以下命令。

```
do newproject
```

说明 如果执行程序不包含扩展名，则 Visual FoxPro 按照如下的顺序来查找执行版本程序：

(1) .EXE 可执行程序

(2) .APP 应用程序

(3) .FXP 已经编译版本

(4) .PRG 程序文件

用 DO 命令执行其他的程序，如菜单程序（.MPR）、表单程序（.SCX）或查询（.QPR）程序，则必须要包含扩展名。

完成 退出可执行程序文件。

11.2 应用程序的测试与调试

11.2.1 Visual FoxPro中的测试和调试

对 Visual FoxPro 程序进行测试是为了发现隐藏的问题，而调试是为了一个一个地发现问题出现的位置并对错误进行修正。通常在应用程序开发当中的各个阶段都需要进行测试

和调试工作，从而保证组成应用程序的各个组件都能正常运行。

在进行测试和调试以前，用户需要建立一个测试和调试的环境，它具有如下的特点：

（1）运行当中不能崩溃并产生错误提示；

（2）测试调试一定的范围，并能产生所需的动作或者错误提示信息；

（3）对于用户的不可预料的各种操作都可作出响应并能恢复初始状态。

Visual FoxPro 提供了功能强大的调试器帮助用户进行程序的调试工作，用户可以方便地发现错误并加以改正。但最为有效的方式还是在程序建立和编码阶段就对可能出现的问题有一个大概的预测，从而避免错误的发生。

具有一个良好的编程习惯，如使用标准的编程格式、给程序添加注释、符合通用的命名习惯等，能在你的程序中减少“Bugs”并且有助于程序的调试工作。为了使后来的测试调试工作更为容易，用户在代码的编写过程中要考虑如下的问题：

（1）程序中要建立一个良好的测试环境；

（2）程序中要有相应的判断；

（3）程序的逻辑与事件的发生顺序一致。

11.2.2 创建测试的环境

系统环境和数据环境对用户程序的测试和调试同样重要，在进行测试以前必须充分考虑到如下的问题：

(1) 系统所需的硬件和软件；

(2) 系统的路径和各个文件的属性；

(3) 整个系统文件的目录结构和文件的所在位置；

(4) 硬件和软件平台的选择。

为了保证应用程序的最大适应性，系统地开发平台应该选择最底层的平台，因此要考虑如下几方面的问题：

(1) 使用最底层的视频方式；

(2) 确定所需的最小的内存和存储空间的大小，包括所有需要并行运行的程序；

(3) 如果是网络版，需要考虑内存、文件和记录锁问题；

(4) 系统路径和文件属性。

应用程序运行前要确认是否能访问到所有所需的路径和文件，要注意以下问题：

(1) 需要公共的系统路径么；

(2) 文件的存储格式设置正确么；

(3) 每个用户的网络许可权限正确么；

(4) 目录结构和文件位置。

如果你在源代码中使用了绝对的路径名称和文件名称，这些路径和文件必须存在于当前的系统中，要注意：

(1) 使用Visual FoxPro的配置文件；

(2) 将源文件的目录和生成文件的目录的路径分开，从而确定发布所需的文件；

(3) 程序中尽量使用相对路径。

11.2.3　使用调试器

使用 Visual FoxPro 的调试器能方便地实现程序的调试。调试器可以有效地分离出错误并能方便地改正。在调试器中可以直接地输入测试的命令并监视代码的运行情况。

调试器具有五个窗口：跟踪、监视、局部、调试输出和调用堆栈。通过它们，用户可以观察整个代码的运行情况。

使用调试器的操作步骤如下：

说　明　通过本例学习使用调试器，包括打开调试器、执行程序、使用跟踪窗口和如何对需要的语句进行修改。

开始　打开项目“NewProject”。

下一步　在“工具”菜单选择“调试器”，进入调试器窗口，如图 11.7 所示。

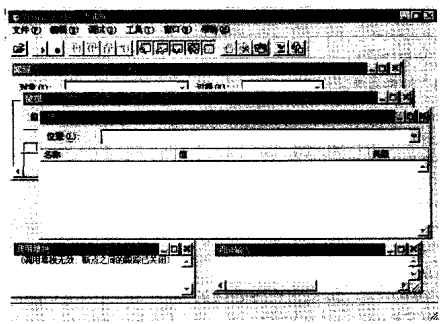


图11.7　调试器窗口

说　明　当程序执行到断点处，调试器将自动打开。

下一步　从“调试器”窗口的“调试”菜单选择“运行”。

下一步　运行“表单”“QckStart.scx”。

下一步　此时“跟踪”窗口中出现表单的代码和对象，如图 11.8 所示。

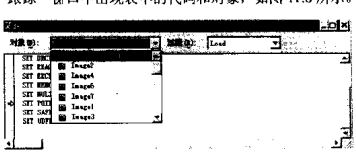


图11.8　跟踪窗口

下一步 从“调试”菜单选择“单步跟踪”或者从工具栏中按“单步”按钮，在代码左侧的箭头指明了当前执行到的语句。

说明 跟踪窗口可以一行一行的执行程序，并可以随时检查所有变量的值、属性和环境的设置。在调试器窗口中跟踪程序，可以采用如下的跟踪策略：

(1) 在程序中使用断点从而缩小需要单步跟踪的程序的范围。

(2) 通过“调试”菜单中的“设置下一条语句”，你可以跳过发生错误的语句直接执行下一条语句。

(3) 如果你有大量的对 Timer 事件响应的代码可以从 Visual FoxPro 主菜单中选择“工具”然后选择“调试”选项卡，清除“显示计时器事件”复选框。

下一步 运行到一个程序行，从“调试”菜单选择“定位修改”，将自动进入代码窗口当前语句的执行位置，可以进行修改。

说明 当选择“定位修改”后，所执行的程序将被终止，并打开代码窗口定位到当前的跟踪窗口中的执行语句，从而在发现错误的时候可以很快的进行修复。

完成 关闭代码窗口。

11.2.4 设置程序断点

使用 Visual FoxPro 的调试器能方便地为程序设置断点，从而将程序在所需要的地点挂起执行，查看变量和属性值以及环境变量。

设置程序断点的操作步骤如下：

开始 从菜单进入“调试器”。

下一步 选择“调试器”的“打开”菜单打开一个工程文件。

下一步 将光标移动到所需要的行，然后按“F9”，或者从工具栏中选择“切换断点”工具，或者鼠标双击需要断点的语句左侧的灰条，当语句的左侧出现了一个实心的红点●，表示断点设置完毕。

说明 当你调试对象时，可以从“对象”框选择对象，从过程框选择过程在需要的地方设置断点。

下一步 从“工具”菜单选择“断点”。

下一步 在“断点”对话框的“类型”框选择“在定位处中断”，如图 11.9 所示。

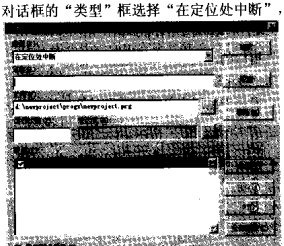


图11.9　断点对话框

说明　当选择了类型为“在定位处中断”时，在“定位”框和“文件”框输入不同的值具有不同的意义，表 11.1 列出了“定位”和“文件”的不同设置的含义。

表11.1　“定位”和“文件”框的设置的断点含义

定位	文件	断点意义
ErrHandler	d:\newproject\newproject.prg	在 newproject.prg 程序中执行名称为 ErrHandler 的过程的第一行
newproject.10	d:\newproject\newproject.prg	在名称为 newproject 程序的第 10 行
Click	d:\newproject\ Form.scx	在名称为 Form.scx 表单的名称为 Click 的过程、函数或事件的第一行
cmdOK.Click	d:\newproject\ Form.scx	在表单 Form.scx 中 cmdOK 相连的 Click 事件的第一行代码
cmdOK::Click		任何父类为 cmdOK 的控件的 Click 事件的第一行代码

下一步　在“断点”对话框的“类型”框选择“当表达式值改变时中断”，如图 11.10 所示。

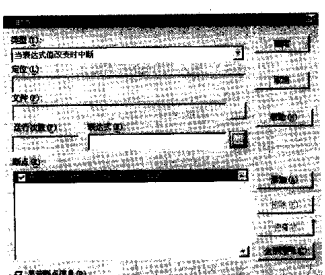


图11.10　断点对话框设置类型为“当表达式值改变时中断”

说明　当选择了类型为“当表达式值改变时中断”时，在“表达式”框输入不同的值具有不同的意义，表 11.2 列出了“表达式”的不同设置的含义。

表11.2　“表达式”框的输入不同值的断点含义

表达式	断点意义
RECNO(　)	当表的记录指针发生移动时中断
PROGRAM(　)	在程序、过程、方法事件的第一行中断
myform.Text1.Value	当文本框的值发生变化时发生改变

下一步 在“断点”对话框的“类型”框选择“当表达式值为真时中断”，如图 11.11 所示。

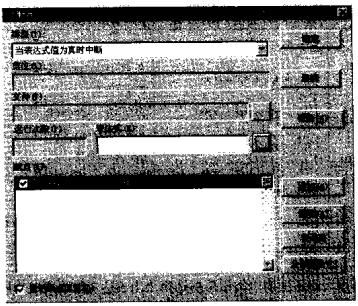


图11.11　断点对话框设置类型为“当表达式值为真时中断”

说明 当选择了类型为“当表达式值为真时中断”时，在“表达式”框输入不同的值具有不同的意义，表 11.3 列出了“表达式”的不同设置的含义。

表11.3　输入不同值的断点含义

表达式	断点意义
EOF( )	当表中的记录指针指到文件尾时发生中断
'CLICK'$PROGRAM( )	在 Click 或 DblClick 的代码的第一行发生中断
nReturnValue = 6	当一个消息框的返回值位于 nReturnValue 时，当用户选择“确认”时引发中断

下一步 在“断点”对话框的“类型”框选择“如果表达式值为真则在定位处中断”。

下一步 在“定位”框输入断点的位置。

下一步 在“表达式”框输入触发断点的表达式。

下一步 按“添加”按钮加入断点。

下一步 从“断点”对话框的“断点”框选择断点，然后按“删除”按钮，可以删除所设的断点。

下一步 进入“跟踪”窗口，在设置断点的行的左侧的●处双点击也可以删除当前的断点。

说明 在跟踪窗口中设置一个断点，然后进入“断点”对话框加入执行断点的条件，即从“类型”框选择“如果表达式值为真则在定位处中断”，然后加入表达式。

完成 从“调试”菜单选择“继续执行”观察所设置的断点是否执行。

11.2.5　观察运行变量

如果要观察程序中变量的值，可以通过“局部”窗口、“监视”窗口和跟踪窗口。

在“局部”窗口中可以显示当前程序的所有的变量、数组、对象和对象变量。在默认情况下，局部窗口中显示的是当前的执行程序的变量值。在“监视”窗口同样可以输入变量或表达式来查看当前的运行情况。

观察运行变量的操作步骤如下：

开始　从菜单进入“调试器”。

下一步　选择“调试器”的“打开”菜单打开一个工程文件。

下一步　将光标移动到所需要的行，然后按“F9”或双击左侧的灰色区域，设置断点。

下一步　从“调试”菜单选择“运行”。

下一步　打开“局部”窗口，并通过选取工具栏上的“跟踪”、“单步”、“跳出”和“运行到光标处”，可以控制当前的程序的执行位置，观察各种变量的情况。

下一步　从“局部”窗口的“位置”框选择需要跟踪变量的位置，从名称框可以选择对象和变量，查看它们的值，如图 11.12 所示。

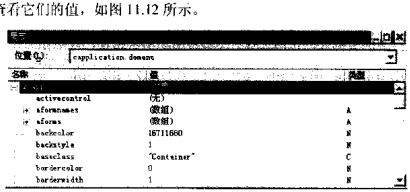


图11.12　从局部窗口查看变量的值

下一步　打开“监视”窗口，从“监视”窗口的“监视”框输入正确的表达式，然后按“ENTER”键，在下面的框中将显示所加入的变量表达式的名称、值和类型，如图 11.13 所示。

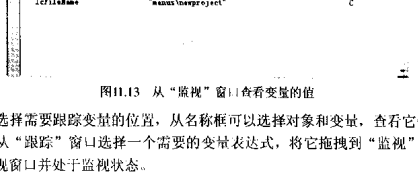


图11.13　从“监视”窗口查看变量的值

下一步　选择需要跟踪变量的位置，从名称框可以选择对象和变量，查看它们的值。

下一步　从“跟踪”窗口选择一个需要的变量表达式，将它拖拽到“监视”窗口，该变量被加入监视窗口并处于监视状态。

或从“监视”窗口中选择一个监视的表达式，然后按“DELETE”键，可以将该变量表达式从列表中删除。

或从“跟踪”窗口选择一个需要的变量或变量表达式，将鼠标在它的上面放片刻，该变量或变量表达式的当前值会自动的显示出来，如图 11.14 所示。

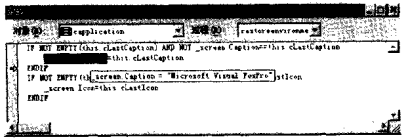


图11.14　自动查看变量表达式的值

或跟踪执行程序，必要时从“调试”菜单选择“取消”终止调试。

11.2.6　处理运行中的错误

程序在运行中会发生各种各样的错误，如：试图向一个并不存在的文件写数据，试图打开一个已经打开的数据表，试图选择一个已经关闭的数据表，被除数为 0，数据的类型相冲突等。

在表 11.4 中列出了处理错误的有用的命令和函数。

表11.4　处理错误的有用的命令和函数

命令和函数	说明
AERROR( )	将错误信息加入一个数组
DEBUG 或 SET STEP ON	打开调试跟踪窗口
ERROR	产生一个错误信号用来检测错误处理
ERROR( )	返回一个错误代号
LINENO( )	返回一个程序行号
MESSAGE( )	返回一个错误消息的字符串
ON ERROR	当发生错误执行的一条命令
ON( )	返回一个分派给错误处理命令的执行命令
PROGRAM( )或 SYS(16)	返回当前执行程序的名称
RETRY	重新执行前一条命令
SYS(2018)	返回当前错误消息的参数

要避免程序的运行错误，一个最好的办法是在程序错误还没有出现之前，预测可能出现的错误，加入必要的处理代码。例如，下面的命令实现表的指针移动：

```
SKIP
```

然而要保证记录的指针不应该越过最后一条记录，用下面的程序代码可以预先处理可能出现的错误，从而避免这种情况的发生。

程序 11.4　处理错误

```
IF !EOF()                                    &&如果没到文件尾
   SKIP
   IF EOF()                                  &&如果到了文件尾
      GO BOTTOM
   ENDIF
ENDIF
```

下面的程序段让用户在一个对话框中选择是否在工作区中打开一个表：

程序 11.5　在工作区中打开一个表

```
USE GETFILE('DBF') IN 0
```

如果用户在对话框中选择了取消或者输入了一个并不存在的数据表的名称，为了处理这些意外的情况可以用下面的代码来代替：

程序 11.6　在工作区中打开一个表

```
cNewTable = GETFILE('DBF')                   &&从文件选择框中选择一个数据库表文件名
IF FILE(cNewTable)                           &&如果文件存在
   USE (cNewTable) IN 0                      &&打开该表
ENDIF
```

11.2.7　处理过程中的错误

ON ERROR 命令可以用来处理过程中的错误代码，当没有对 ON ERROR 作出处理，Visual FoxPro 将用默认错误消息处理错误。

在 ON ERROR 命令后可以使用任何 Visual FoxPro 的命令或表达式，但最为常用的是在它的后面执行错误处理的过程或程序。

错误处理的操作步骤如下：

[五步] 在命令窗口中输入一个毫不相关的字符串“sjdgh”，然后按“ENTER”键，在窗口中出现错误提示号“16”，同时命令窗口变为如下的形式。

```
ON ERROR ?ERROR()
sjdgh
```

[下一步] 打开一个项目，并建立一个新的表单。

[下一步] 在表单上加入一个命令按钮。

[下一步] 在命令按钮的“Click”事件下加入如下的代码。

```
this.parent.error                            &&触发 Error 事件
```

[下一步] 在表单的“Error”事件下加入如下的代码。

程序 11.7　表单的“Error”事件代码

```
LPARAMETERS nError, cMethod, nLine
```

```
IF nError=.F.                                          && 当只触发 Error 事件时
  messagebox("测试错误代码！")
  nError=11200
ENDIF
DO CASE                                                && 对错误的代码进行处理
  CASE nError = 11000
    messagebox("错误 11000！")
  CASE nError = 11200
    messagebox("错误 11200！")
  OTHERWISE
    messagebox("其他错误！")
ENDCASE
```

说明　典型的 ON ERROR 的错误处理代码如下所示。

程序 11.8　ON ERROR 的错误处理代码

```
LOCAL lcOldOnError                                     &&用于保存原来的 ERROR 设置
lcOldOnError = ON("ERROR")
* 用一个过程来处理 ON ERROR 的错误
ON ERROR DO errhandler WITH ERROR(), MESSAGE()
* 其他的应用代码
ON ERROR &lcOldOnError                                 &&恢复原来的 ERROR 设置

PROCEDURE errhandler                                   &&处理错误代码的过程
LOCAL aErrInfo[1]
AERROR(aErrInfo)
DO CASE
    CASE aErrInfo[1] = 1                               && 显示文件不存在的信息
        *显示出适当的错误处理信息
        * 如果有必要可以运行修复程序用来修复错误
    OTHERWISE
        * 显示无法处理的未知错误
        * 向上一级提交错误代号
ENDPROC
```

完成　运行表单，单击按钮看程序的反映。

11.2.8　处理类和对象中的错误

如果 Visual FoxPro 的错误发生在对象的方法中，Visual FoxPro 将在该对象的 Error

事件处理错误。如果没有相关的处理代码，Visual FoxPro 一级一级向上查找 Error 事件的处理代码。如果在整个的类层次的继承结构中都没有找到处理错误的代码，则 Visual FoxPro 执行 ON ERROR 确定的错误处理。最后如果没有找到相关的错误处理，Visual FoxPro 显示默认的错误处理消息。

使用类来进行错误处理有很多优点，例如当你发现所有的继承类都有同一错误时，只需要在它们的父类中加入对应的错误处理，则所有的子类的对象都将自动得到该错误的错误处理。

类错误处理的操作步骤如下：

开始 从“帮助”菜单打开“Solutions 示例”。

下一步 从“类”选项卡选择“Buttons”类库的“vcr”类。

下一步 选择“修改”，在“类设计器”中双击按钮，进入代码窗口。

下一步 从代码窗口中选择“vcr”的“Error”过程，代码如下。

程序 11.9　“vcr”的“Error”过程代码

```
Parameters nError, cMethod, nLine
#define NUM_LOC "Error Number: "
#define PROG_LOC "Procedure: "
#define MSG_LOC "Error Message: "
#define CR_LOC CHR(13)
#define SELTABLE_LOC "Select Table:"
#define OPEN_LOC "Open"
#define SAVE_LOC "Do you want to save your changes anyway?"
#define CONFLICT_LOC "Unable to resolve data conflict."

DO CASE
    CASE nError = 13                                    && 当没有发现别名时
    *-----------------------------------------------------------------------*
    * 当用户试图在一个不存在或者没有打开的表中移动数据
    *或者表的指针移动的设置出现了错误
    *那么显示打开表的对话框.
    *-----------------------------------------------------------------------*
        cNewTable = GETFILE('DBF', SELTABLE_LOC, OPEN_LOC)
        IF FILE(cNewTable)
            SELECT 0
            USE (cNewTable)
            This.SkipTable = ALIAS()
        ELSE
            This.SkipTable = ""
        ENDIF
```

```
CASE nError = 1585
*-----------------------------------------------------------------------*
* 有 datachecker 类来处理冲突.
*-----------------------------------------------------------------------*
    nConflictStatus = THIS.DataChecker1.CheckConflicts()
    IF nConflictStatus = 2
        WAIT WINDOW CONFLICT_LOC
    ENDIF
 OTHERWISE
*-----------------------------------------------------------------------*
* 显示不可预料的错误信息.
*-----------------------------------------------------------------------*
   lcMsg = NUM_LOC + ALLTRIM(STR(nError)) + CR_LOC + CR_LOC + ;
       MSG_LOC + MESSAGE( )+ CR_LOC + CR_LOC + ;
       PROG_LOC + PROGRAM(1)
   lnAnswer = MESSAGEBOX(lcMsg, 2+48+512)
   DO CASE
      CASE lnAnswer = 3                                    &&忽略
         CANCEL
      CASE lnAnswer = 4                                    &&重试
         RETRY
      OTHERWISE
         RETURN
   ENDCASE
ENDCASE
```

说 明　所有的由“vcr”类中对四个按钮的操作都可能产生错误，错误代码不是分离的由各个按钮来处理，而是由它们所在的容器来统一处理，只要在各个按钮的“Error”事件加入如下的代码即可：

```
Parameters nError, cMethod, nLine
This.Parent.Error(nError, cMethod, nLine)
```

对于一些不可预料的错误，如果你没有加入必要的处理，将不会给任何的错误提示。对于有些错误信息，你可能需要加入对错误信息更为详细的说明、如用户获得帮助的途径等。

完 成　关闭所有的窗口。

说 明　当错误处理完成后，将执行产生错误的语句的下一条语句，如果你想重新运行产生错误的语句，可以使用 RETRY 命令。

11.3 应用程序的优化

11.3.1 对表和索引的优化

为了更快地获得表中的数据，可以使用索引和缓冲区，也可以使用 Rushmore 技术来优化查询。

如果你希望使用 Rushmore 技术，最好给表添加索引，这样不但可以增加查询的速度，而且可以实现记录的排序。当表中的记录具有唯一索引时，可以给表建立主索引或者候选索引，它们可以使 Visual FoxPro 性能更加优化。

除了为查找和排序建立的索引以外，在连接中也需要用到索引。如果将两个不具有索引的表连接起来，时间可能将慢几百倍。

Visual FoxPro 可以用索引表达式来建立索引，对于查询、排序或者几个表的连接可以使用到多个字段，也可以使用一些表达式，大大方便了实际应用。

当表的索引增加时，表的查询排序的速度可能会增加，然而对表中的数据的更新和插入的操作效率将降低。因为 Visual FoxPro 需要对所有的索引进行更新。

此外，不要在那些差异不大的字段上建立索引，如逻辑字段。因为它们对查询提供的优化是有限的，却加重了系统的负担，影响系统的整体效率。

（1）提高连接的效率

当使用 SELECT-SQL 语句对表进行连接时，请避免在连接的表出现如下的情况：

1. 连接表不具有主索引或唯一索引

2. 连接的表中含有空字段

为了避免这种情况的发生，可以在一个表的主索引和另一个表的外来关键字之间建立建立关系。当在不具有唯一索引的表之间建立连接时，最终可能出现两个表。例如下面的 SELECT-SQL 连接语句将产生非常大的结果集：

```
SELECT *;
 FROM  业务!客户 INNER JOIN 业务!定单 ;
   ON  客户.邮政编码 = 定单.邮政编码
```

上例中邮政编码确定了一个城市的位置，然而用它将客户和定单进行连接并没有实际的价值。邮政编码既不能唯一的确定一个客户，也不能唯一确定一张定单。因此用客户代号作为实现两表连接的条件来代替上面的语句要好的多：

```
SELECT *;
 FROM  业务!客户 INNER JOIN 业务!定单;
   ON  客户.客户代号 = 定单.客户代号
```

另外，当连接具有空字段的表时要小心，因为空字段在 Visual FoxPro 中是匹配的关系。然而，null 值在 Visual FoxPro 中不是一种匹配关系。

在对具有空字段的表进行连接时，可以使用对空值的判断条件。例如如果你认为在定单表中的客户代号可能为空值，可以在 SELECT-SQL 连接语句中加入如下的筛选条件，排除不具有客户代号的定单。

```
SELECT *;
FROM  业务!客户 INNER JOIN 业务!定单;
  ON  客户.客户代号= 定单.客户代号;
  WHERE 业务!定单<> ""
```

 也可以使用 EMPTY( )函数来测试空的字段，但是使用函数的运行速度比直接和常量的运行速度要慢。

使用项目管理器，可以将程序和过程组合成一个.APP 或 .EXE 文件。这将提高程序的运行效率。Visual FoxPro 将打开程序文件并是它处于打开的状态，从而，当在程序中使用 DO 命令时，Visual FoxPro 不需要再次打开文件。此外，一个应用程序只有一个或两个文件，从而减少了工作目录的文件数量，文件操作的速度将大大提高。

为了使应用程序的效率更高，更有效地使用表的索引时，要遵循以下的原则：

（1）如果没有使用表缓冲或者记录缓冲区，用户插入记录最好使用INSERT-SQL语句，而不要使用APPEND BLANK加REPLACE，尤其在多用户环境下更应如此，因为使用INSERT-SQL语句，索引只需更新一次；

（2）如果需要向一个具有索引的表添加大量记录，先移去索引，然后再添加记录，最后在重建索引，这样将更快。

由于函数对每一条记录都需计算，在SQL语句中应该避免使用函数，尤其对于那些返回多条记录的语句更是如此。使用名称表达式或者宏替换来代替函数，更好的策略是根据需要动态的建立SQL语句。

使用常用的顺序为索引建立排序。

在多用户环境下使用复合索引（.CDX），而不要用IDX文件。因为对一个复合索引（.CDX）的更新要比对多个单一索引的文件的更新速度要快。

11.3.2 使用Rushmore技术

Visual FoxPro 通过使用 Rushmore 技术可以大大提高数据的搜索速度。Rushmore 技术对标准的 Visual FoxPro 索引提供了优化，对 Visual FoxPro 中的任何一种索引都可以使用 Rushmore 技术，如 FoxPro 1.x(.IDX)索引，压缩索引(.IDX)和复合索引(.CDX)。

复合索引(.CDX) 和压缩索引(.IDX)都使用了压缩技术，使得它们的大小只是旧的格式的六分之一。因为使用了压缩技术，索引占用了更小的磁盘空间，内存中的缓冲区的索引容量也相应增大。

如果 Visual FoxPro 的运行环境的内存较小，而需要索引的表很大，可能没有足够的内存使用 Rushmore 技术。此时 Visual FoxPro 将显示警告的提示信息，此时你的程序仍可以正确运行，却没有通过 Rushmore 技术得到优化。

根据涉及的表的数量，可以使用 Rushmore 技术有不同的效果。处理单表数据时，可以用 FOR 子句进行优化。当处理多表数据时，使用 SELECT – SQL 语句。因而在使用 Rushmore 技术时，可以遵从下列原则：

从单表中获取数据时，在命令中使用FOR子句，例如对AVERAGE, BROWSE, 或者 LOCATE，或者使用SQL命令，如表11.5所示。

表11.5 可以使用FOR子句进行优化的命令

AVERAGE	BLANK	BROWSE
CALCULATE	CHANGE	COPY TO
COPY TO ARRAY	COUNT	DELETE
DISPLAY	EDIT	EXPORT TO
INDEX	JOIN WITH	LABEL
LIST	LOCATE	RECALL
REPLACE	REPLACE FROM ARRAY	REPORT
SCAN	SET DELETED	SET FILTER
SORT TO	SUM	TOTAL TO

从多表中取数据，可以使用SELECT-SQL语句、DELEDTE-SQL语句和UPDATE-SQL语句。

当使用 scope 子句时，如果想用 Rushmore 技术进行优化，则需要用 ALL 或 REST，NEXT 和 RECORD 子句都没有优化作用。因为多数的命令都把 ALL 作为默认的 scope 子句，当提供 scope 子句时自动地使用了 Rushmore 技术。

为了执行优化，不要使用排序功能。当创建索引或标识的时候，自动产生排序。因而对一个大的有序数据表使用 Rushmore 技术，应该使用 SET ORDER TO 关闭索引，然后使用 SORT 命令。

11.3.3 对索引使用Rushmore技术

Visual FoxPro 并不能对所有的索引使用 Rushmore 技术。当在 INDEX 命令中使用了 FOR 子句时，此时 Rushmore 技术并不提供优化。例如对于下面的带有 FOR 子句的命令，不能使用 Rushmore 技术。

```
INDEX ON ORDNUM FOR DISCOUNT > 10 TAG ORDDISC
```

对于含有 NOT 条件的表达式也不能提供优化。例如对于下面的命令，Visual FoxPro 提供了优化。

```
INDEX ON DELETED() TAG DEL
```

但下面的命令不具有优化作用。

```
INDEX ON NOT DELETED() TAG NOTDEL
```

当 SET DELETED 设置为 ON 时，使用第一个命令将大大提高运行的效率。

11.3.4 关闭Rushmore

当优化命令对 FOR 子句的索引关键字进行了修改的时候，基于 Rushmore 技术的记录可能会变的过时，此时需要关闭 Rushmore。

关闭 Rushmore 可以采用如下的方法：

使用NOOPTIMIZE子句

例如，对 LOCATE 命令下边的命令没有使用优化：

```
LOCATE FOR DueDate < {01/01/95} NOOPTIMIZE
```

使用SET OPTIMIZE命令全局的设置Rushmore

全局的关闭 Rushmore：

```
SET OPTIMIZE OFF
```

全局的打开 Rushmore：

```
SET OPTIMIZE ON
```

优化的全局的默认值是 ON。

说明 在如下的情况下不使用 Rushmore 技术进行优化：

(1) 当 Rushmore 技术不能对 FOR 子句的表达式进行优化时。

(2) 当使用 WHILE 子句时。

当内存不足时，数据检索将不使用 Rushmore 技术。

11.3.5 优化Rushmore表达式

在 FOR 子句或者 SQL WHERE 子句中的可优化表达式决定了 Rushmore 技术的形式。最基本的可优化的表达式既可以是一个整个的表达式，也可以是表达式的一部分，所以可以将基本的表达式组合成为复合的优化的表达式。

如果要创建基本的可优化的表达式可以用如下的语法：

```
eIndex  relOp  eExp
```

或者：

```
eExpr  relOp  eIndex
```

优化表达式具有如下的特点：

(1) eIndex与组成的索引表达式要精确的匹配。

(2) eExp是任何的表达式，包括从其他的不相关表中的所有的变量和字段。

(3) relOp是如<, >, =, <=, >=, <>, #, ==, 或!=的关系操作符。可以使用诸如ISNULL()、BETWEEN()或INLIST()的函数。

如果使用 BETWEEN()或 INLIST()函数，可以用如下的形式：

```
eIndex BETWEEN(eIndex, eExpr, eExpr)
```

或者

```
eExpr INLIST(eIndex, eExpr)
```

注意 对于 ISBLANK( )和 EMPTY( )函数，Visual FoxPro 不能提供优化。

当在“职员名称”、“职员代号”、“职员性别”和“雇佣日期”字段建立了索引时，下边的表达式是可以优化的：

```
职员名称 = "福勒"
职员代号 >= 1000
UPPER(职员名称) = "SMITH"
雇佣日期 < {12/30/90}
```

用户要了解什么时候查询将被优化，什么时候查询不被优化。只有表达式严格地按照

索引表达式构造时，Visual FoxPro 才进行 Rushmore 优化。

下面语句中的表达式是一个优化的表达式：

```
USE CUSTOMERS
INDEX ON UPPER(cu_name) TAG name
```

下面的表达式将不被优化，因为查询条件只是建立在 cu_name，而不建立在索引表达式之上：

```
SELECT * FROM customers WHERE cu_name ="ACME"
```

如果要对上面的语句实现优化可以用下面的语句替代：

```
SELECT * FROM customers WHERE UPPER(cu_name) = "ACME"
```

如果想检查 Rushmore 优化的级别，可以使用 SYS(3054)函数。

当 FOR 子句是一个优化的表达式，可以通过对优化表达式的连接，实现对带有 FOR 和 WHERE 子句的查询的优化。

使用逻辑运算符 AND, OR 和 NOT 将基本的表达式连接，可以组成一个优化表达式。这个表达式可以是一个完全优化的表达式。但如果连接的表达式不是优化的，连接后的复合表达式可能是部分优化的或非优化的表达式。如何确定表达式的连接是否是优化的，可以按照表 11.6 所示的表达式的组合方法来判断。

表11.6 基本表达式的组合优化结果

基本表达式	逻辑操作符	基本表达式	查询的优化结果
可优化	AND	可优化	完全可优化
可优化	OR	可优化	完全可优化
可优化	AND	不可优化	部分可优化
可优化	OR	不可优化	不可优化
不可优化	AND	不可优化	不可优化
不可优化	OR	不可优化	不可优化
——	NOT	可优化	完全可优化
——	NOT	不可优化	不可优化

通过“AND”操作符可以将两个可优化的基本表达式连接成为一个完全可优化的表达式，如：

```
职员姓名 = "WOWO" AND HIREDATE < {12/30/89}                    && 可被优化的表达式
```

如果使用“OR”将一个可优化的表达式和一个不可优化的表达式连接在一起，形成的表达式是不可优化的，如：

```
职员姓名 = "WOWO" OR "2" $职员代号                              && 不可优化
```

如果“NOT”一个可优化的表达式将形成一个完全可优化的表达式，如：

```
NOT 职员名称 = "WOWO"                                          && 完全可优化
```

与连接基本表达式一样，也可以连接复合表达式，形成一个完全优化的表达式、部分优化的表达式或一个不可优化的表达式。这些复合表达式的组合规则如表 11.7 所示。

表11.7　复合表达式的组合优化结果

基本表达式	逻辑操作符	基本表达式	查询的优化结果
完全可优化	AND	完全可优化	完全可优化
完全可优化	OR	完全可优化	完全可优化
完全可优化	AND	部分可优化	部分可优化
完全可优化	OR	部分可优化	部分可优化
完全可优化	AND	不可优化	部分可优化
完全可优化	OR	不可优化	不可优化
——	NOT	完全可优化	完全可优化
部分可优化	AND	部分可优化	部分可优化
部分可优化	OR	部分可优化	部分可优化
部分可优化	AND	不可优化	部分可优化
部分可优化	OR	不可优化	不可优化
——	NOT	部分可优化	不可优化
不可优化	AND	不可优化	不可优化
不可优化	OR	不可优化	不可优化
——	NOT	不可优化	不可优化

使用“OR”可以将两个完全可优化的表达式连接成为一个完全可优化表达式，如：

```
(FIRSTNAME = "FRED" AND HIREDATE < {12/30/89});
    OR (LASTNAME = "" AND HIREDATE > {12/30/88})
```

使用“AND”表达式可以将一个可优化的表达式和一个不可优化的表达式连接成为一个部分可优化的表达式，如：

```
(FIRSTNAME = "FRED" AND HIREDATE < {12/30/89});
    AND "S" $ LASTNAME
```

部分可优化的表达式同样可以组合成为一个部分可优化的表达式，如：

```
(FIRSTNAME = "FRED" AND "S" $ LASTNAME);
   OR (FIRSTNAME = "DAVE" AND "T" $ LASTNAME)
```

将不可优化的表达式连接可以形成一个不可优化的表达式，如：

```
("FRED" $ FIRSTNAME OR "S" $ LASTNAME);
   OR ("MAIN" $ STREET OR "AVE" $ STREET)
```

11.3.6　优化表单和控件

为了使表单和控件的性能得到优化，可以使用下面所述的方法：

(1) 使用数据环境：在“表单设计器”、“报表设计器”中使用数据环境，数据表打开的操作将比在表单的“Load”事件使用USE, SET ORDER, 和 SET RELATION命令的运行速度要快。

(2) 限制表单和表单集的数量：只有当一组表单共享同一个私有数据周期时，才使用表单集。当使用表单集时，Visual FoxPro将为所有在表单集中的表单和控件建立应用实例。

这对于那些并不共享同一个数据周期的表单来说，最好使用DO FORM命令在需要的时候打开其他的表单。

(3) 在页框中动态的加入页面控件：页框和表单集一样，当页框被载入时自动地为每一个页面加入所有的控件，因而当页框被载入时要花一定的时间。你可以为每个页面建立一个类，只是当需要时才载入页面的控件。

为页面载入控件可以按如下的步骤：

开始 按要求设计所有的表单的控件。

下一步 设计好以后，进入表单的第一页，将所有的控件保存为一个类。

下一步 在类设计器中打开所保存的类，看是否将所有的控件载入。

下一步 按照以上的方法将剩下的页面保存为类。

完成 在每个页面的 Activate 事件中加入对象，并使它们可视，如下面的代码。

程序 11.10 页面的 Activate 事件代码

```
IF THIS.ControlCount = 0
    THIS.AddObject("cnrpage1","cnrpage1")
    THIS.cnrpage1.Visible = .T.
    ENDIF
```

为控件动态的绑定数据：对于具有大量数据绑定控件的表单，可以在需要的时候动态地将数据控件和数据源相连。下面是进行连接设置的步骤：

开始 在表单的数据环境中加入所需的表单和视图。

下一步 在每一个数据绑定控件的 GotFocus 事件，加入绑定代码。例如将组合框的数据源设置为 customer.company 字段，可以用如下的代码。

程序 11.11 GotFocus 事件代码

```
*检查控件是否被绑定.
IF THIS.RecordSource = ""
    *将数据源的类型设置为正确的值
    * 并将控件与所需的字段相绑定
    THIS.RecordSource = "customer.company"
    THIS.RecordSourceType = 6
    THIS.Refresh
ENDIF
```

完成 设置每一个数据绑定控件的数据源。

延迟屏幕的刷新：有时由于需要，需要改变屏幕中控件的属性，如使控件可见还是不可见，改变控件的颜色，在绑定的控件中移动记录等。此时最好当所有的屏幕更改都完成后，再刷新屏幕。要延迟屏幕的刷新，可以用如下的方法：

开始 设置表单的 LockScreen 属性为“.T.-真”。

下一步 根据需要更改控件。

下一步 调用表单的 Refresh 方法。

完成 将表单的 LockScreen 属性设置为“.T.-假”。

下面的代码实现了上面的整个过程：

程序 11.12　GotFocus 事件代码

```
THISFORM.LockScreen = .T.                          &&锁定表单屏幕
THISFORM.MyButton.Caption = "保存"
THISFORM.MyGrid.BackColor = RGB (255, 0, 0)        &&  设置为红色
SKIP IN customers
SKIP IN orders
THISFORM.Refresh
THISFORM.LockScreen = .F.                          &&取消屏幕锁定，实现刷新
```

减少经常调用的方法的代码：因为 Refresh 和 Paint 事件被频繁调用，所以应尽量减少表单中相应事件的代码。同样，为了增加表单的载入速度，可以将 Init 事件中的部分代码，根据需要加入到不频繁使用的 Activate, Click, 和 GotFocus 事件中。

11.3.7　优化程序

为了提高 Visual FoxPro 的应用程序的效率，可以使用以下的方法。

在程序的编写过程中遵循如下的准则：

(1) 合理地选择数据类型，在需要的地方用“整型(Integer)”类型代替“数值型(numeric)型类型。在主键和外来键中尽量使用整型（Integer），从而减少表和索引文件的大小，并提高连接的速度。

(2) 避免频繁地打开文件。应该在打开文件的时候给打开的文件分配工作区，然后用SELECT命令选择特定的工作区。

(3) 使用“FOR...ENDFOR”语句来代替“DO WHILE...ENDDO”循环，因为它们更加快速。

(4) 当从多个字段读取数据时，“SCATTER TO ARRAY”的运行速度要快于“SCATTER MEMVAR”。

(5) 为了更有效地使用内存，应该在对象使用时再创建它们，使用完以后要及时地从内存中释放这些对象。

(6) 应该将输出内容输出到顶层窗口，对隐藏窗口的操作会大大减低程序的运行效率。

(7) 使用SET TALK OFF命令关闭状态显示，设置SET DOHISTORY为OFF避免在每次程序运行时都更新命令窗口。

使用名称表达式来代替宏替换：此时程序的运行效率将大大提高。例如当将一个值赋给变量cFile时，使用名称表达式要比使用宏替换要快。代码为：

程序 11.13　GotFocus 事件代码

```
cFile = "CUST"
use &cFile                                         &&  宏替换的速度较慢
use (cFile)                                        &&名称表达式的速度较快
```

更有效地引用对象的属性：为了更好地引用对象的属性，可以采用如下的方法：

(1) 对反复引用属性进行优化时，如果使用*object.property*语法实现对对象属性的反复引用，它的效率很底。为此，可以将属性值存于变量，对它加以改变，然后一次设置属性。例如，下面的代码通过在内存中创建一个数组用于存储对象的属性，只在最后一次的实现对属性的赋值。

程序 11.14 使用数组

```
* 将字符串赋给局部变量
lcChar = THISFORM.cCharString
LOCAL laCharArray[256]                                    && 产生局部数组
FOR nCounter = 1 to 256
    laCharArray[x] = SUBSTR(laChar,x,1)
ENDFOR
*最后将局部数组赋给对象数组
ACOPY(laCharArray,THISFORM.aCharArray)
```

(2) 如果对对象属性的更改多于一个，Visual FoxPro 将多次地对对象进行搜索。在下面的例子里 Visual FoxPro 将搜索四个对象(THISFORM, pgfCstInfo, pgCstName, 和 txtName)从而设置属性，因为代码设置两个属性，这种搜索将执行两次。

程序 11.15 搜索属性

```
THISFORM.pgfCstInfo.pgCstName.txtName.Value = "Fred Smith"
THISFORM.pgfCstInfo.pgCstName.txtName.BackColor = RGB (0,0,0)          & 黑色
```

为了避免上面的情况发生，可以使用“WITH...ENDWITH”命令。从而使 Visual FoxPro 只搜索一次对象，例如下面的代码的运行效率比上边的要快：

程序 11.16 使用“WITH...ENDWITH”命令

```
WITH THISFORM.pgfCstInfo.pgCstName.txtName
    .Value = "Fred Smith"
    .BackColor = RGB (0,0,0)                                   & 设置为黑色
ENDWITH
```

你也可以将对象存于一个对象的引用，然后用它来引用对象，如：

程序 11.17 引用对象

```
oControl = THISFORM.pgfCstInfo.pgCstName.txtName
oControl.Value = "Fred Smith"
oControl.BackColor = RGB (0,0,0)   & 设置为黑色
```

11.3.8 多用户环境下的应用

多用户环境下，必须要充分考虑到效率问题。当多个用户同时获取数据时，必须要充分考虑到并行和网络问题。

当你的应用试图锁住表或记录失败，可以尝试让 Visual FoxPro 经过一个小的时间间隔去重新锁定。然而每次这样的操作都会增加网络流量，如果网络本身的负荷已经很大，这种反复请求将使所有用户的效率降低。为了解决这个问题，可以调整请求锁定的时间间隔，通过使用一个比较长的请求时间间隔，减少网络的流量。

在每一次请求锁定时，调用 SYS(3051)函数可以设置某次请求的时间间隔的毫秒数。

另一种方法是使用事务。当事务开始时，事务的整个过程中，记录保持锁定状态，直到事务提交。即使你使用了 UNLOCK 命令，在 END TRANSACTION 或 ROLLBACK 命令之前也是锁定状态。

此外，向表中添加数据要求 Visual FoxPro 锁定表头。表头在事务执行当中保持锁定状态，禁止其他用户的操作。为了减少事务对操作的影响，最好事务只是包含数据的更新语句。

当在表单中使用事务时，不要先开始事务，然后运行表单，在关闭表单后提交事务，最好在表单的一个事件代码中执行事务的代码，如在“保存”按钮的“Click”事件加入如下的代码：

程序 11.18 “保存”按钮的“Click”事件代码

```
BEGIN TRANSACTION
UPDATE PRODUCTS SET reorder_amt = 0 WHERE discontinued = .T.
END TRANSACTION
```

11.3.9 优化对远程数据的提取

从远程数据库获取数据要付出很高的代价，为了从与远程服务器上的数据库中获得数据，必须按如下的过程进行：

开始 客户端向远程数据库发出查询请求。

下一步 服务器端对请求进行分析、处理。

下一步 服务器端产生查询的结果集。

下一步 服务器端通知客户机端查询完毕。

完成 客户机端通过网络从服务器端获取数据。

为了实现对远程数据提取的优化，可以采用如下的方法：

1. 只获取你所需要的数据

对于表单和报表来说，并不需要提取远程数据库的所有数据。因此，你可以通过创建远程视图，从而只提取所需要的数据字段和记录，这样可以减少网络上的数据流量。

从远程数据源获取数据，有如下的建议：

(1) 不要使用语句“SELECT * FROM …”，应该设定所需要的字段。

(2) 使用WHERE子句来限定传递数据的记录的数量。WHERE子句的设置越严格，传

递到你的计算机上的数据越少，查询的速度越快。

(3) 如果在设计时无法确定WHERE子句的值，可以在子句中使用参数。例如可以用如下的子句实现在运行时查询某一地区的记录。

程序 11.19 在运行时查询某一地区的记录

```
SELECT cust_id, company, contact, address ;
    FROM customers ;
        WHERE region = ?pcRegion
```

(4) 设置数据环境游标的 NoDataOnLoad 属性。这个属性通常当参数值来源于表单的控件值时，使视图参数化。

2. 更有效地更新远程表

当使用视图更新远程数据源中的表时，Visual FoxPro 必须检查要检查记录或记录集是否发生了变化。此时 Visual FoxPro 将服务器上的数据和你的本机数据进行比较，许多时候，这是一种费时的操作。

为了优化对远程数据源的优化，你可以设定 Visual FoxPro 对数据更改的检查方式。此时可以使用 WHERE 子句来控制记录的更新。

例如，在远程数据源上有一个基于客户表的视图，SELECT - SQL 语句可以进行如下构造：

程序 11.20 SELECT - SQL 语句

```
SELECT cust_id, company, address, contact ;
    FROM customers ;
    WHERE region = ?vpRegion
```

11.4 小 结

本章论述了如何利用前面几章详细论述的构造应用程序所需的所有要素。利用它们，通过 Visual FoxPro 提供的功能强大的向导功能，十分方便地建立应用程序的框架。创建需要的界面和所需要的数据，通过查询和报表使用户得到必要的数据，从而将整个的应用程序编译通过形成最终的程序。在编译和创建过程中，用户的每一个组件都需要进行严格的测试和调试，从而保证程序运行的正确性。Visual FoxPro6.0 功能强大的调试器可以帮助用户方便地完成调试。为了使你的应用程序变的小巧、高效，在应用程序构造当中要考虑到程序的优化。

本章主要介绍了：应用程序框架的构造、应用程序的测试与调试、应用程序的优化。

思考与练习

1. Visual FoxPro 应用程序的结构是怎样的？
2. 主程序起到了什么作用？如何设置？

3. 如何编译一个应用程序？
4. 用 Visual FoxPro 调试程序时要注意哪些问题？用户在编码时应考虑到哪些问题？
5. Visual FoxPro 调试器的五个调试窗口各有什么功能？
6. 设置程序断点要注意哪些问题？
7. 如何提高连接的效率？

第四篇

基于 WEB 数据库开发

本篇导读

每天，都有成百上千的人蜂拥到 Internet 上。各种 Web 站点和其他基于 Internet 的商业或科学的应用也日新月异。Visual FoxPro 的高速数据库引擎和灵活的面向对象的编程语言，使得它成为建立 Internet 应用程序的一个良好的工具。

第12章　Web 中的数据库开发

现今，WWW（World Wide Web）被广泛使用，对于在 Web 中开发数据库的应用程序有很大的需求。为了更好地发挥 Web 站点的灵活性，使用数据库作为存储信息，同时对信息进行及时更新，动态地显示当前信息的最新内容是十分必要的。

本章主要讲述如下内容：

（1）Web 中的数据库开发；

（2）将Visual FoxPro用于Web 开发。

12.1　Web 中的数据库开发

12.1.1　开发基于Web 的应用程序

为了使用 Visual FoxPro 开发基于 WWW 的应用程序，应具备如下的环境：

（1）TCP/IP的网络协议；

（2）一个Web 服务器（最好是运行于Windows NT Server的）；

（3）Web 连接器应用程序，一个脚本（Script）或一个连接器（Connector）；

（4）一台至少有32M RAM的奔腾高速计算机；

（5）一个Web 浏览器。

12.1.2　在Web 上运行应用程序

Internet 中的数据库连接的实质是客户机/服务器连接的一种特殊形式。用户拥有一个用于访问后端（Web 服务器）的前端（Web 浏览器），然后 Web 服务器和提供数据库访问的应用程序相连接。前端和后端之间通过 Internet 连接在一起，这便组成了 WWW 和 HTTP（Hypertext Transfer Protocol）。

其中，Web 浏览器不能直接访问数据，而 Web 服务器则不能直接访问向用户显示结果的界面。

在浏览器端具有脚本和 Active 控件。浏览器脚本是客户端浏览器的扩展，如 Netscape 的 JavaScript 和 Microsoft 的 VBScript。这些脚本语言提供了对用于输入的表单（Form）用户界面的控制，以及快速产生 HTML 输出的编程功能。脚本程序使得浏览器变得聪明起来，但浏览器仍然不能直接访问数据。

Active 控件是可以由 HTML 网页调用的并用浏览器脚本控制的外部控件。Active 控件包括 Java applet 和 ActiveX 自定义控件，它们可以由 Web 浏览器提供的脚本引擎来执行或者调用。脚本语言提供了基本的界面编程功能，而 Active 控件则优化提高了 Web 浏览器的性能。

然而即使使用高性能的 Active 控件也不能实现对服务器数据的直接访问。只有通过 Web 服务器才能实现应用程序和数据库的连接。

Web 服务器的作用如同一个中介人，协调应用程序对数据库的访问。Web 服务器调用服务器脚本来完成任务，其访问方法如图 12.1 所示。

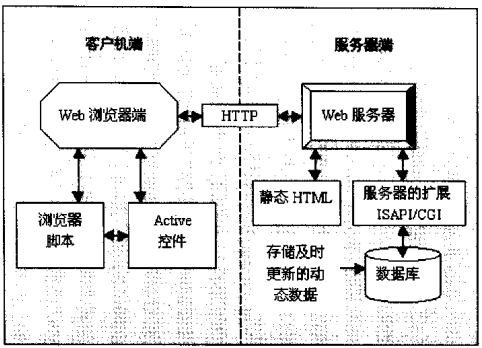


图12.1　通过浏览器访问Web服务器

Web 浏览器发送一个请求来执行服务器端的脚本程序，从而请求访问数据。脚本是运行于 Web 服务器中的程序，用来响应用户激活超链接或单击 HTML 表单的按钮。服务器端的脚本程序和客户机端的 JavaScript 或者 VBScript 是不同的。它可以是遵循 CGI（Common Gateway Interface）协议的可执行文件，也可以是服务器端的 API 接口，例如 Microsoft 的 IIS（Internet Information Server）中的 ISAPI（Internet Server API）。这些脚本可以是自我包含程序，也可以调用 Visual FoxPro 等其他的应用程序。

在每一个 Web 请求中，浏览器加入了自己的相关信息：浏览器的类型、Internet 地址、输入表单中所定义的字段值，并把这些信息发送到 Web 页中。当调用服务器端的脚本时，这些信息以及有关 Web 服务器的其他信息将会传给脚本。接着脚本负责运行它的处理任务，并产生与 Web 服务器兼容的输出。这些任务可以是运行数据库的查询，向表单中加入记录。

当运行服务器端的脚本程序，它便执行自定义的任务处理。但它总向 Web 服务器返回一个响应。多数情况下，这个响应显示了请求结果的 HTML 文档：查询结果、确认页或出错页等。

服务器脚本通常有如下功能：

(1) 从Web浏览器、Web服务器和HTML表单的变量检索信息。

(2) 建立实际的应用，如查询、向表中添加记录、处理各种有效性的检验等。

(3) 产生输出。多数情况下是HTML输出，但可以产生确认、重定向到另一个Web网页

或产生一个出错信息页。

12.2 Visual FoxPro的Web开发

12.2.1 为何使用Visual FoxPro开发Web应用程序

Visual FoxPro 具有的快速检索本地数据，灵活地访问远程数据，以及功能强大的面向对象的编程语言，使得 Visual FoxPro 成为很好的后端程序的开发平台。

在 Web 中访问 Visual FoxPro 可以采用如下的机制：

(1) 通过ODBC和一个基于Web服务器的工具。

(2) 利用Visual FoxPro OLE服务器响应Web请求。

(3) 将Visual FoxPro应用程序作为数据服务器处理Web服务器的请求。

使用 Visual FoxPro 作为 Web 的后端开发时具有如下的优点：

(1) Visual FoxPro具有更为高速的数据引擎。

(2) 可以利用真正的数据库开发工具，不必大量使用面向一般应用的脚本语言，从而可以建立更为复杂的商业应用。

(3) 面向对象的特点易于建立一个可以重用的处理框架，有利于系统的建立与维护。

使用 Visual FoxPro 作为数据库的后端，用户只需要稍微学习一些 HTML 的用法，不必受语义定义较为模糊的脚本语言的限制，用户可以使用它提供的功能强大的语言和数据库功能建立 Web 应用程序的逻辑。

12.2.2 建立基于Visual FoxPro的WWW查询网页

使用 Visual FoxPro 6.0 集成到菜单的 WWW 查询网页向导，用户可以方便地产生一个 HTML 文件作为输出，用户在 Web 浏览器中可以实现对 Visual FoxPro 中数据的查询。

向导使得复杂的网页在很短的时间里便可以完成。使用向导可以产生三个文件：.HTML 文件、.IDC 文件和.HTX 文件。每个文件都具有一个相同的文件名，由共享名加以区分。

使用向导建立 WWW 查询页的操作步骤如下：

说明　本例通过向导建立一个 WWW 数据查询的查询网页，首先使用数据库向导建立一个关于"唱片收藏的数据库，然后在浏览器中实现关于收藏的唱片的查询。并简要介绍了有关网页制作的一些简单知识。

开始　从“文件”菜单选择“新建”，选择向导。

下一步　在“应用程序向导”中，输入工程的名称，如图 12.2 所示，然后选择"确定"按钮。

下一步　在随后出现的“用程序生成器”中可以设置工程文件的属性，见图 12.3 所示。

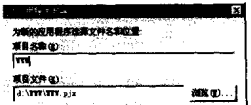


图12.2　应用程序向导

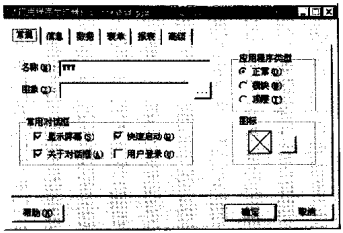


图12.3　设置应用程序属性

下一步 用“数据库向导”建立一个数据库，选择"Music Collection"数据库，如图 12.4 所示。

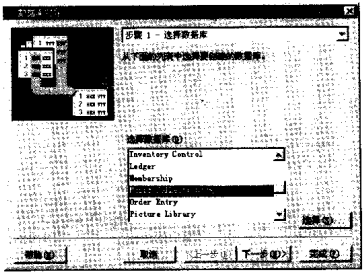


图12.4　用数据库向导建立数据库

下一步 在向导的第三步设置索引对话框中设置各表的主索引如下所示：

Music Category　：Musiccategoryid，Musiccategory

Recordingartists　：Recordingartistid，Recordingartistname，Birthdate

Recordings　：Recordingid，Recordingtitle，Recordingartistid，Musiccategoryid

Tracks　：Trackid

下一步 在“步骤 5 - 完成”对话框,选择"用示例数据填入表",按"完成"按钮,见图 12.5 所示。

下一步 保存数据库文件。

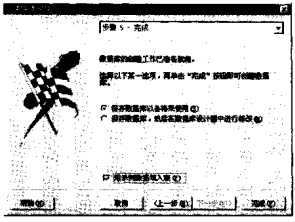


图12.5　选择将示例数据加入数据库

下一步 从“工具”菜单，选择“向导”，然后选择“全部”，选“WWW 搜索页向导”，进入如图 12.6 所示的对话框，选择“下一步”按钮。

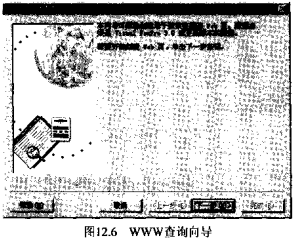


图12.6　WWW查询向导

下一步 从“数据库和表”框选择“Music Collection”，然后选择“RECORDING_ARTIST”，如图 12.7 所示，选择“下一步”按钮。

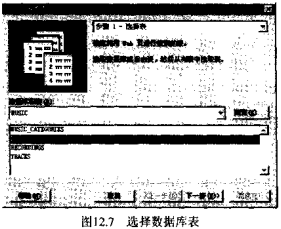


图12.7　选择数据库表

下一步 然后在选择查询字段对话框中选择查询的索引，如图 12.8 所示，选择“下一步”按钮。

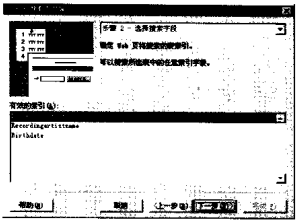


图12.8　选择查询的索引

下一步 在查询页标题处输入标题，并输入网页的说明，如图 12.9 所示，按“下一步”按钮。

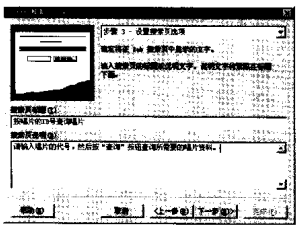


图12.9　输入网页的标题和说明

下一步 选择查询界面的背景图案和标题图案，如图 12.10 所示，按“下一步”按钮。

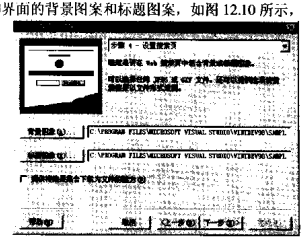


图12.10　选择标题图案和背景图案

下一步 选择查询结果显示的字段，如图 12.11 所示，按“下一步”按钮。

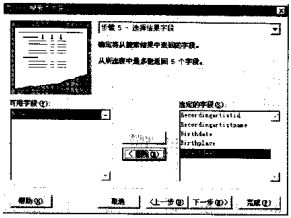


图12.11　选择查询结果显示字段

下一步 选择查询结果的背景图案和标题图案，设置显示的最多记录数和 ODBC 数据源，如图 12.12 所示，按“下一步”按钮。

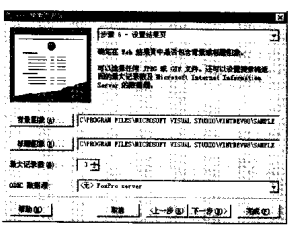


图12.12　设置查询结果页属性

下一步 按“Finish”完成查询网页的生成，保存生成的网页为 WWW.HTM。

下一步 进入 IE4.0 打开刚才生成的 WWW.HTM 文件，运行的结果如图 12.13 所示。

图12.13　生成的查询网页

下一步 从 IE4.0 的“查看”菜单选择“脚本调试程序”，进入“InterDev 6.0”可以对当前的脚本程序进行调试，如图 12.14 所示。

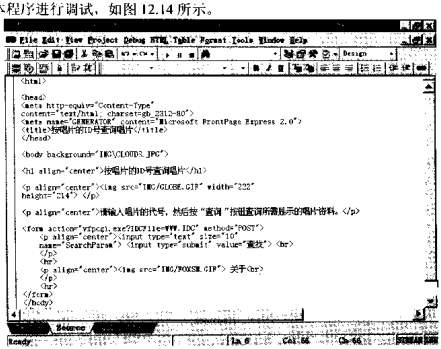


图12.14　进入InterDev 6.0可以修改调试脚本程序

下一步 在“InterDev 6.0”中显示的 HTML 程序的 WWW.HTM 文件如下：

程序 12.1　WWW.HTM程序

```
<HTML>
<HEAD>
<TITLE>按唱片的 ID 号查询唱片</TITLE>
</HEAD>
<BODY BACKGROUND="IMG/CLOUDS.JPG">
<CENTER><h1>按唱片的 ID 号查询唱片</h1></CENTER>
<CENTER>
<IMG SRC="IMG/GLOBE.GIF">
</CENTER>
<P>

</BODY>
<P>
<CENTER>
<P>　请输入唱片的代号，然后按“查询”按钮查询所需显示的唱片资料。</P>
<FORM ACTION="vfpcgi.exe?IDCFile=WWW.IDC" METHOD="POST">
<INPUT NAME="SearchParam" SIZE=10 VALUE="" >
```

```
<INPUT TYPE="SUBMIT" VALUE="Search">
<br>
</CENTER>
<hr><center>
<img src="IMG/FOXSM.GIF">
Generated by the Visual FoxPro WWW Search Page Wizard<br>
</center>
<hr>
</FORM>
</BODY>
</HTML>
```

说明 (1) 上面的 HTML 程序中，网页的标题将显示于当前浏览器的网页的最上方，标题通常显示了当前网页的功能。使用向导生成的 HTML 文件时，关于标题的说明被嵌在<TITLE> 与 </TITLE>之间，你可以在 "InterDev 6.0" 中对它进行修改：

```
<HEAD>
<TITLE — 欢迎你和我共同分享音乐天堂 — </TITLE>
</HEAD>
```

(2) 网页描述显示于标题的下方，用来提示网页的使用者当前站点的建设目的和查询相关信息的方法。当网页的使用者需要输入特定的查询信息时，通常需要加上网页的说明，和信息的输入格式等。修改本例的说明如下：

```
<P>
<CENTER>
<P>通过本网页，你可以和我共同分享我的音乐收藏，输入唱片的代号，你将得到该唱片的相关信息。</P>
```

(3) 你可以为你的网页设置标题图案和背景图案，在向导中为标题或背景选择.GIF 或.JEMP 图形文件，使可以在网页上显示当前的标题和背景图案，标题图案将显示在标题和描述之间，背景图案将填充当前的整个网页的背景。本例中查询网页的背景图案由如下语句设置：<BODY BACKGROUND="IMG/BANNER4.GIF">，标题图案是由如下语句设置：<IMG SRC="IMG/GLOBE.GIF">。

(4) 当选择了输出字段时，所有被选择的输出字段将以列表的形式显示出来，通过向导将产生.HTX 文件，当 HTML 文件需要返回结果时，这个文件将被 IIS 调用。被选择的字段将以变量的形式显示在%...%之间。本例生成的WWW.HTX文件如下：

程序 12.2 WWW.HTX 程序

```
<HTML>
<HEAD>
<TITLE>Visual FoxPro Query Return Page</TITLE>
</HEAD>
<BODY BACKGROUND="IMG/DSS.GIF">
```

```
<CENTER>
<IMG SRC="IMG/GLOBE.GIF">
</CENTER>

<HEAD><TITLE>Visual FoxPro WWW Data Server</TITLE></HEAD>
<BODY>
<CENTER><H1>Search Results</H1><H2>VFP WWW Data Server</H2><HR></CENTER>
<PRE>
<%BEGINDETAIL%>
<NOBR><B><%Recordingartistid%></B>: <%Recordingartistname%>, <%Birthdate%></NOBR>
<%ENDDETAIL%>
</PRE>
<HR>
<DL>
<%BEGINDETAIL%>
<DT><B><%Recordingartistid%></B>
<DD><%Birthplace%>
<DD><%Dateofdeath%>
<BR>
<%ENDDETAIL%>
</DL>
<HR>
</BODY>
</HTML>
```

(5) 查询结果页和查询页十分近似，你可以为查询结果页加标题图案和背景图案。它们的位置和查询页相同，在结果页中可以设置返回的查询结果的记录的最大行数，默认值是 10。

(6) 由向导产生的.HTM 文件是一个标准的网页文件，典型的其中有 FORMACTION 语句用来引用将被执行的 CGI 脚本文件。如下语句：

```
<FORM ACTION="vfpcgi.exe?IDCFile=WWW.IDC" METHOD="POST">
```

上面的语句向 CGI 脚本程序 vfpcgi.exe 发出 HTTP 服务请求，它带一个后缀为.IDC 的文件作为参数，它将被 Visual FoxPro 服务器使用：IDCFile=WWW.IDC。

(7) .IDC 文件基于将被 IIS 使用的文件格式。标准的.IDC 组件包括数据源，一个模板文件，一个 SQL 语句和将要返回的最大记录的行数。由向导生成的 WWW.IDC 文件如下所示：

程序 12.3 WWW.IDC 程序代码

```
Datasource:
Template: WWW.HTX
SQLStatement:
+SELECT   Recordingartistid, Recordingartistname, Birthdate, Birthplace, Dateofdeath
```

```
+ FROM 'RECORDING ARTISTS'
+ WHERE Recordingartistid = '%SearchParam%'
Maxrecords: 13
```

上面的程序中：

Datasource:由向导产生的文件使用固定的 Visual FoxPro 数据源，因此数据源的值为空值。

Template:模板参数引用一个后缀为.HTX 的文件，用来定制返回页的信息。.HTX 文件由向导产生，本文件中引用的是WWW.HTX文件。

SQLStatement: 参数包含一个 SQL SELECT 语句，它由向导根据用户在一、二、五步的选择来产生的。其中的%SearchParam%参数是由.HTM 文件返回的查询参数，包含了用户实际输入的值。

Maxrecords:中的值是由向导的第六步产生的值，表示结果网页显示的记录数。

完成 关闭 InterDev 6.0。

12.2.3　建立基于Visual FoxPro的WWW网上发布

现在，用户中为数众多的应用是在网上发布相关信息，Visual FoxPro 6.0 提供了网上发布向导，用来帮助用户把已知的信息在网上公布。

利用网上发布的功能，通过在网上随时显示数据库中的数据，使用户可以随时得到当前的最新信息。相关数据存储于中央数据库，用户通过浏览器访问当前所需的网页，网页中显示的数据与当前数据库中的数据完全一致，并可以随时根据数据库的更新情况加以修正。

下面通过一个具体的实例说明如何用 Visual FoxPro 建立基于网上发布的网页。

使用向导进行网上信息发布的操作步骤如下：

说 明 本例通过向导建立一个在网上发布艺术家信息的网页，使用上一小节建立的数据库，用向导将当前的数据库中存储的有关艺术家的相关信息在网上发布。

开始 打开上一节建立的 WWW 工程。

下一步 从“工具”菜单，选择“Web 发布”，进入如图 12.15 所示的对话框，选择需要发布的信息的表和字段，按“下一步”按钮。

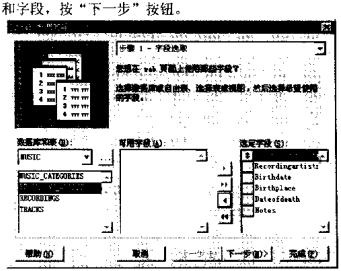


图12.15　选择需要显示信息的字段

下一步 在选择排序字段对话框选择排序的字段,如图 12.16 所示,按"下一步"按钮。

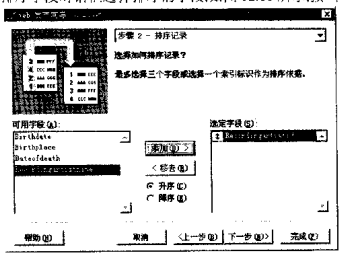


图12.16　选择数据列表的排序方式

下一步 在选择发布风格对话框，选择发布的风格，如图 12.17 所示。

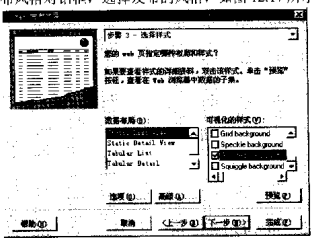


图12.17　选择数据的列表风格

下一步 在选择发布风格对话框按“选项...”按钮，进入如图 12.18 所示的对话框，选择背景颜色以及字体的颜色，然后按“确定”按钮。

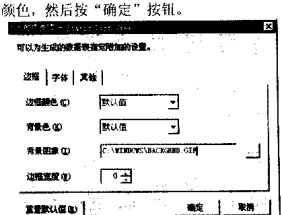


图12.18　设置网页的静态显示风格

下一步 在选择发布风格对话框按“高级...”按钮，进入如图 12.19 所示的“高级”对话框，设置网页的颜色和图案。

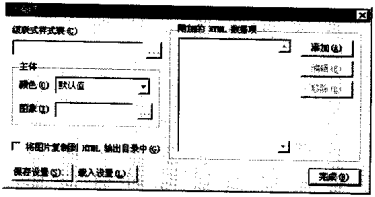


图12.19　“高级”对话框

下一步 在“高级”对话框中按“添加”按钮，进入如图 12.20 所示的“HTML 项”对话框，设置项的属性，加入两个项属性，属性值的确定如图 12.21 所示。

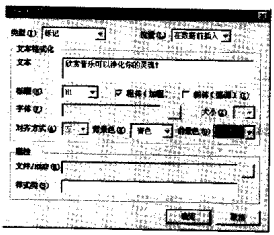


图12.20　设置插入网页元素标记的属性

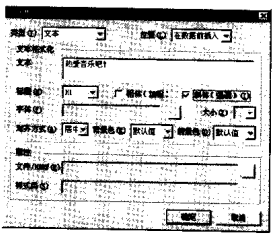


图12.21　设置插入的网页文本的属性

说明 各种属性的含义如下所示：

Type 类型框，用来选择在页面上插入的不同类型的 HTML 元素内容。

Text：显示在页面上的普通文本。

Hyperlink：用来产生超级链接，链接你需要访问的数据源。

Image：在一定的位置插入图像元素。

Marquee：在页面上产生可以移动的文本。

Horizontal Rule：在页面上产生一条横线。

Line Break：产生换行符。

Location 框，用来选择插入的 HTML 元素的位置。

Insert before data：将 HTML 元素插入到浏览器显示的数据的前面。

Insert after data：将 HTML 元素插入到浏览器显示的数据的尾部。

Insert in Header：插入 HTML 元素到标题位置。

Text Formating，用来设置文本的属性。

Text：文本的显示内容，只对 Text、Marquee、Hyperlink 三种类型有意义。

Heading：标题级别，共分为六级 H1-H6，主要用于设定文本字体的大小，设置时一般是由大到小排列，及由 H1 到 H6 的顺序顺次排列。

Bold：设置字体是否为粗体。

Italic：设置字体是否为斜体。

Font Name：设置字体的类型名称，如果选择了 Heading 则，此项失效。

Size：字体大小。

Alignment：设置当前文本相对浏览器中的页面的对齐方式。

Backcolor：设置当前插入元素的背景颜色。

Forecolor：设置当前的插入元素的字体颜色。

Atributes：用来设置当前的插入项的

File/HREF：设置超级链接链接的文件。

Style class：选择当前的插入项的样式类。

下一步 回到发布风格对话框按“下一步”按钮，进入如图 12.22 所示的对话框，然后按“完成”按钮，保存文件。

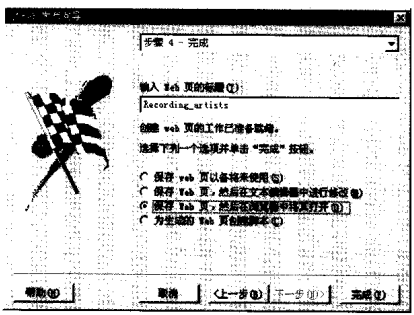


图12.22　完成网页生成

下一步 此时 Internet Explorer 4.0 将自动打开并显示当前由向导生成的.HTM 网页的数据，见图 12.23 所示。

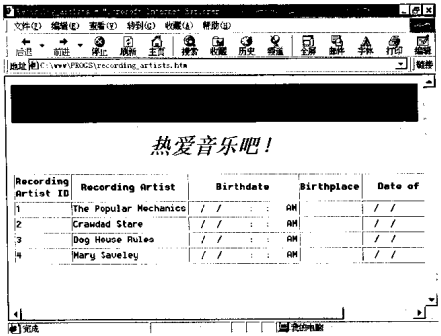


图12.23　生成的网上发布的网页

下一步从 IE4.0 的“查看”菜单选择“脚本调试程序”，进入“InterDev 6.0”可以对当前的脚本程序进行调试，.HTM 程序如下所示。

程序 12.4　.HTM 程序代码

```
<html>
<head>
<title>
Recording_artists
</title>
</head>
<body bgcolor="silver" background="BACKGRND.GIF">
<marquee bgcolor="Teal">
<h1>
<font color="Lime">
<strong>
欣赏音乐可以净化你的灵魂！
</strong>
</font>
</h1>
</marquee>
<h1 align="center">
```

```
<em>
热爱音乐吧!
</em>
</h1>
<table id="GenTable" border="1" background="BACKGRND.GIF" style="font-family: 8514fix; font-size:
4">
<tr>
<th>Recording Artist ID</th>
<th>Recording Artist</th>
<th>Birthdate</th>
<th>Birthplace</th>
<th>Date of Death</th>
<th>Notes</th>
</tr>
<tr>
<td>1</td>
<td>The Popular Mechanics</td>
<td>  /  /     ::   :  &nbs
p;AM</td>
<td> </td>
<td>  /  /     ::  ::  &nbs
p;AM</td>
<td> </td>
</tr>
<tr>
<td>2</td>
<td>Crawdad Stare</td>
<td>  /  /     ::  :  &nbs
p;AM</td>
<td> </td>
<td>  /  /     ::  :  &nbs
p;AM</td>
<td> </td>
</tr>
<tr>
<td>3</td>
<td>Dog House Rules</td>
<td>  /  /     ::  :  &nbs
p;AM</td>
```

```
    <td> </td>
    <td>  /  /     :  :  &nbs
p;AM</td>
    <td> </td>
    </tr>
    <tr>
    <td>4</td>
    <td>Mary Saveley</td>
    <td>  /  /     :  :  &nbs
p;AM</td>
    <td> </td>
    <td>  /  /     :  :  &nbs
p;AM</td>
    <td> </td>
    </tr>

    </table>
    </body>
    </html>
```

开成 关闭 InterDev 6.0。

12.3 小　结

本章介绍了基于 Web 的数据库开发的基本概念，如今基于 Web 数据库的应用程序有很大的需求，为了更好地发挥 Web 站点的灵活性，使用数据库作为存储信息，同时对信息进行及时更新，动态地显示当前信息的最新内容是十分必要的。

本章主要讲述如下内容：Web 中的数据库开发的基本概念、将 Visual FoxPro 用于 Web 开发。

思考与练习

1. Web 应用程序的结构是怎样的？
2. 在 Web 应用程序中，服务器程序和客户程序各承担了什么样的角色？
3. 如何利用 Visual FoxPro 建立 WWW 查询网页？
4. 如何利用 Visual FoxPro 进行 WWW 网上发布？

[General Information]
书名=Visual Foxpro 6.0 实例教程
作者=门槛创作室
页数=312
SS号=10028655
出版日期=1999年4月第1版