

目 录

第1章 创建第一个VB应用程序	(1)
1.1 VB 6.0简介	(1)
1.1.1 VB的基本概念	(1)
1.1.2 VB 6.0版本简介	(1)
1.1.3 VB 6.0的功能和特点	(2)
1.2 启动和退出VB 6.0	(3)
1.2.1 启动VB 6.0	(3)
1.2.2 退出VB 6.0	(3)
1.3 创建第一个VB应用程序	(5)
1.4 对象与事件的基本概念	(10)
1.4.1 对象的属性和方法	(10)
1.4.2 事件及事件过程	(12)
1.5 本章小结	(12)
第2章 VB 6.0的集成开发环境	(13)
2.1 认识VB 6.0的集成开发环境	(13)
2.1.1 主窗口	(13)
2.1.2 窗体编辑器窗口(Form窗口)	(14)
2.1.3 工具箱	(14)
2.1.4 属性窗口	(16)
2.1.5 工程资源管理器窗口	(16)
2.1.6 代码编辑窗口	(16)
2.1.7 对象浏览器	(17)
2.1.8 窗体布局窗口	(18)
2.1.9 调试窗口	(18)
2.1.10 菜单编辑窗口	(19)
2.1.11 调色板窗口	(19)
2.2 工作环境的设置	(19)
2.2.1 【编辑器】选项卡	(19)
2.2.2 【编辑器格式】选项卡	(22)
2.2.3 【通用】选项卡	(22)
2.2.4 【可连接的】选项卡	(23)
2.2.5 【环境】选项卡	(23)
2.2.6 【高级】选项卡	(23)
2.3 本章小结	(24)

第3章 工程的管理	(26)
3.1 工程的基本概念	(26)
3.1.1 工程资源管理器	(26)
3.1.2 工程结构	(26)
3.2 创建、打开和保存工程	(28)
3.2.1 创建新的工程	(28)
3.2.2 保存和打开工程文件	(29)
3.3 添加、删除和保存文件	(29)
3.3.1 添加文件	(30)
3.3.2 删除文件	(30)
3.3.3 保存文件	(30)
3.4 生成可执行文件	(31)
3.5 使用应用程序向导和外接程序	(33)
3.5.1 外接程序管理器	(33)
3.5.2 使用向导	(33)
3.6 本章小结	(36)
第4章 VB 6.0 的编程语言	(38)
4.1 程序的书写规则	(38)
4.1.1 注释	(38)
4.1.2 断行	(38)
4.1.3 将多行语句写在一行上	(39)
4.1.4 良好的编程习惯	(39)
4.2 变量和常量	(40)
4.2.1 变量的命名规则	(40)
4.2.2 变量的声明	(41)
4.2.3 变量的数据类型	(43)
4.2.4 变量的作用范围	(47)
4.2.5 变量的显式声明和隐式声明	(48)
4.2.6 常量	(49)
4.3 编写简单语句	(50)
4.3.1 赋值语句	(50)
4.3.2 使用各种运算	(50)
4.4 流程控制语句	(51)
4.4.1 条件判断语句	(51)
4.4.2 循环语句	(53)
4.5 过程和函数	(55)
4.5.1 定义和调用过程	(56)
4.5.2 定义和调用函数	(57)
4.5.3 过程和函数的参数	(57)
4.5.4 退出过程	(59)
4.6 本章小结	(59)

第 5 章 窗体设计	(60)
5.1 窗体的属性和事件	(60)
5.1.1 窗体的属性	(60)
5.1.2 窗体的事件	(62)
5.2 窗体的设计	(63)
5.2.1 创建新窗体	(63)
5.2.2 向窗体上添加控件	(64)
5.2.3 对控件的操作	(65)
5.2.4 设置窗体位置	(68)
5.2.5 设置启动窗体	(68)
5.2.6 窗体设计的基本原则	(69)
5.3 窗体的生命周期	(69)
5.3.1 窗体的创建	(70)
5.3.2 窗体的加载	(70)
5.3.3 窗体的显示	(70)
5.3.4 窗体的卸载	(71)
5.3.5 结束应用程序	(73)
5.4 MDI 窗体的基本概念	(73)
5.5 本章小结	(74)
第 6 章 VB 6.0 的基本控件	(75)
6.1 控件的基本概念	(75)
6.1.1 控件的分类	(75)
6.1.2 焦点的概念	(77)
6.2 命令按钮(CommandButton 控件)	(77)
6.2.1 向窗体添加命令按钮	(77)
6.2.2 命令按钮的属性和事件	(78)
6.2.3 带图案的命令按钮	(78)
6.3 文本框(TextBox 控件)	(79)
6.3.1 使用 TextBox 中的文本	(79)
6.3.2 多行 TextBox	(80)
6.3.3 创建密码文本框	(81)
6.3.4 创建只读文本框	(81)
6.4 标签(Label 控件)	(81)
6.4.1 设置标签的标题	(82)
6.4.2 用标签创建访问键	(82)
6.5 选项按钮(OptionButton 控件)	(83)
6.5.1 创建选项按钮组	(83)
6.5.2 选定或禁止选项按钮	(83)
6.5.3 选项按钮的事件和属性	(84)
6.5.4 选项按钮示例	(84)
6.6 复选框(CheckBox 控件)	(86)
6.6.1 复选框的属性和事件	(86)

6.6.2	CheckBox 示例程序	(87)
6.7	列表框(ListBox 控件)	(88)
6.7.1	Click 和 Double-Click 事件	(88)
6.7.2	向列表增减项目	(88)
6.7.3	获取列表内容	(89)
6.8	组合框(ComboBox 控件)	(90)
6.8.1	组合框的样式	(90)
6.8.2	增减列表中的项目	(91)
6.8.3	获取列表内容	(92)
6.9	滚动条	(93)
6.9.1	滚动条的主要事件和属性	(93)
6.9.2	滚动条控件示例	(94)
6.10	ActiveX 控件简介	(95)
6.11	本章小结	(96)
第 7 章	对话框	(97)
7.1	使用函数生成对话框	(97)
7.1.1	消息框	(97)
7.1.2	输入框	(100)
7.2	通用对话框	(101)
7.2.1	通用对话框概念	(101)
7.2.2	打开和另存为对话框	(103)
7.2.3	字体对话框	(105)
7.2.4	颜色对话框	(106)
7.2.5	打印对话框	(107)
7.2.6	帮助对话框	(108)
7.3	定制对话框	(109)
7.4	本章小结	(111)
第 8 章	菜单设计	(112)
8.1	菜单的基本概念	(112)
8.2	菜单编辑器	(113)
8.2.1	菜单编辑器的显示与隐藏	(113)
8.2.2	菜单编辑器的组成	(114)
8.3	用菜单编辑器创建菜单	(115)
8.3.1	在菜单编辑器中创建菜单控件	(115)
8.3.2	分隔菜单项	(116)
8.3.3	赋值访问键和快捷键	(116)
8.3.4	编写菜单控件的代码	(117)
8.3.5	创建子菜单	(118)
8.4	运行时创建和修改菜单	(119)
8.4.1	使菜单命令有效或无效	(119)

8.4.2	使菜单控件不可见	(119)
8.4.3	在菜单中使用复选标记	(120)
8.5	在菜单中添加文件列表	(120)
8.5.1	创建菜单控件数组	(120)
8.5.2	添加文件列表	(121)
8.5.3	保存文件列表	(121)
8.6	快捷菜单	(122)
8.7	本章小结	(124)
第9章	工具栏和状态栏	(125)
9.1	关于 ActiveX 控件	(125)
9.2	创建工具栏	(126)
9.2.1	使用 ImageList 控件	(126)
9.2.2	利用 ToolBar 控件创建工具栏快捷按钮	(128)
9.2.3	为工具栏编写代码	(129)
9.2.4	灵活使用 ToolBar 控件	(130)
9.2.5	手工创建工具栏	(131)
9.3	创建状态栏	(132)
9.4	本章小结	(134)
第10章	使用图形	(135)
10.1	使用图形控件	(135)
10.1.1	给应用程序添加图形	(135)
10.1.2	窗体和控件的图形属性概述	(137)
10.2	坐标系统	(139)
10.2.1	VB 的坐标系	(139)
10.2.2	Scale 方法及其属性	(140)
10.3	绘图方法	(142)
10.3.1	绘制文本	(142)
10.3.2	画直线	(143)
10.3.3	填充形体	(145)
10.3.4	使用 Circle 方法	(146)
10.3.5	画曲线	(148)
10.4	在图形中使用颜色	(149)
10.4.1	定义颜色	(149)
10.4.2	定义渐变	(150)
10.4.3	取得颜色值	(150)
10.5	本章小结	(151)
第11章	多文档界面	(152)
11.1	多文档界面的结构	(152)
11.2	多文档界面的设计	(154)

11.2.1 菜单设计	(154)
11.2.2 使用 MDI 窗体及其子窗体	(156)
11.3 MDI NotePad 应用程序	(160)
11.4 本章小结	(161)
第 12 章 调试程序	(163)
12.1 错误类型和程序模式	(163)
12.1.1 VB 程序中的错误类型	(163)
12.1.2 三种程序模式	(164)
12.2 进入中断模式	(165)
12.2.1 在程序中设置断点	(165)
12.2.2 使用 Stop 语句进入中断模式	(167)
12.2.3 其它方式	(167)
12.3 跟踪应用程序的执行	(168)
12.3.1 逐语句执行代码	(168)
12.3.2 跳过代码中的过程	(168)
12.3.3 从过程中跳出	(168)
12.3.4 运行到光标处	(168)
12.3.5 设置下一条要执行的语句	(169)
12.4 使用调试窗口	(169)
12.4.1 使用监视窗口	(169)
12.4.2 使用【调用堆栈】对话框	(171)
12.4.3 使用本地窗口	(172)
12.4.4 使用立即窗口	(172)
12.5 本章小节	(173)
第 13 章 文件操作	(174)
13.1 文件系统的概念	(174)
13.2 用于处理文件系统的语句和函数	(174)
13.3 文件系统控件	(177)
13.3.1 驱动器列表框	(177)
13.3.2 目录列表框	(178)
13.3.3 文件列表框	(179)
13.4 文件的读写	(180)
13.4.1 顺序文件	(181)
13.4.2 随机文件	(183)
13.4.3 使用二进制文件	(186)
13.5 文件管理	(187)
13.5.1 拷贝和移动文件	(187)
13.5.2 文件的更名	(187)
13.6 本章小结	(188)

第 14 章 多媒体(MCI)编程	(189)
14.1 多媒体中的图形	(189)
14.1.1 LoadPicture 函数	(189)
14.1.2 Paintpicture 方法	(190)
14.1.3 SavePicture 语句	(191)
14.1.4 MSChart 控件	(192)
14.1.5 关于颜色	(197)
14.2 使用音频和视频	(200)
14.2.1 使用 Animation 控件	(200)
14.2.2 Multimedia MCI 控件	(201)
14.2.3 MCIWnd 控件	(205)
14.3 本章小结	(206)
第 15 章 数据库	(207)
15.1 数据库编程概述	(207)
15.1.1 关系数据库概述	(207)
15.1.2 VB 数据库体系结构	(207)
15.2 数据库相关控件及其编程	(208)
15.2.1 DATA 控件	(209)
15.3 DAO 编程	(217)
15.3.1 创建数据库	(217)
15.3.2 修改数据库	(219)
15.3.3 使用记录和字段	(221)
15.4 SQL 简介	(230)
15.4.1 SQL 简介	(230)
15.4.2 SQL 和定位的比较	(230)
15.4.3 SQL 部件	(231)
15.5 数据库的维护	(234)
15.5.1 映射当前数据库结构的方法	(234)
15.5.2 压缩数据库	(236)
15.5.3 修复数据库	(237)
15.6 本章小结	(237)
第 16 章 WEB 浏览	(239)
16.1 编写 WEB 浏览器	(239)
16.1.1 使用 Browser 窗体编写 WEB 浏览器	(239)
16.1.2 使用 WebBrowser 控件编写 WEB 浏览器	(241)
16.2 VB 的 Internet 编程	(243)
16.3 VB 的 Internet 应用程序简述	(243)
16.3.1 在 Internet 应用程序中使用 ActiveX 文档	(243)
16.3.2 在 Internet 应用程序中使用 ActiveX 控件	(245)
16.3.3 在 Internet 应用程序中使用 ActiveX 代码部件	(247)

16.4 本章小结	(248)
附录 A 安装和卸载 VB 6.0 中文版	(249)
附录 B 菜单总汇	(255)
附录 C 编码约定	(263)

第 1 章 创建第一个 VB 应用程序

VB 6.0 中文版是 Microsoft 公司最新推出的跨世纪的产品,适合于 Windows 95 / 98 和 Windows NT 平台。它简单易用,适用面广,无论是通信、数据库,还是多媒体以及普通的 Windows 应用程序都可以用 VB 进行开发,而且方便快捷。在深入学习 VB 6.0 中文版之前,首先了解一下 VB 编程的基本概念是十分必要的。

1.1 VB 6.0 简介

本节作为全书的开篇,向读者讲述什么叫 VB,以及 VB 6.0 中文版的功能特点。

1.1.1 VB 的基本概念

Microsoft VB 提供了开发 Microsoft Windows 应用程序的最迅速、最简捷的方法。不论是 Microsoft Windows 应用程序的资深专业开发人员还是初学者,VB 都为他们提供了整套工具,以方便开发应用程序。

何谓 VB? “Visual”在字面上的意思是“看的、视觉的、用于看的”,引申到计算机程序设计,意思是:“可视化程序设计”,指的是开发图形用户界面 (GUI) 的方法。使用这种方法,用户不需编写大量代码去描述界面元素的外观和位置,而只要把预先建立的对象拖放到屏幕上的一点即可。

“Basic”指的是 BASIC (Beginners All Purpose Symbolit Instruction Code) 语言,它是一种在计算机技术发展历史上应用得最为广泛的语言。VB 在原有 BASIC 语言的基础上进一步发展,至今包含了数百条语句、函数及关键词,其中很多和 Windows GUI 有直接关系。专业人员可以用 VB 实现其他任何 Windows 编程语言的功能,而初学者只要掌握几个关键词就可以建立实用的应用程序。

1.1.2 VB 6.0 版本简介

VB 6.0 中文版是 Microsoft 公司在 VB 5.0 之后推出的最新版本,它有三种版本,各自满足不同的开发需要。

- **学习版**

学习版使编程人员轻松开发 Windows 95 / 98 和 Windows NT 的应用程序。该版本包括所有的内部控件,连同 Grid、Tab 和 Data.. Bound 控件。

- **专业版**

专业版为专业编程人员提供了一整套进行程序开发的功能完备的工具。该版本包括学习版的全部功能,以及 ActiveX 控件,还包括 Internet 控件和 Crystal Report Writer。

- **企业版**

企业版使得专业编程人员能够开发功能强大的组内分布式应用程序。该版本包括专业版的全部功能,以及自动化管理器、部件管理器、数据库管理工具、Microsoft Visual SourceSafe(TM)

面向工程版的控制系统等等。

1.1.3 VB 6.0 的功能和特点

VB 6.0 是 VB 5.0 的升级版本,它不仅继承了 VB 5.0 版本的诸多优点,而且增加了许多新特性。

首先,VB 6.0 有与以往的 VB 版本同样的特点,它们主要是:

1. 本机代码(专业版和企业版)

为了更快地执行,可将 VB 工程编译成本机代码。使用新的本机代码选项配置本机代码的编译,并可使用 Visual C++ 环境调试本机代码。

2. 创建自己的 ActiveX 控件(专业版和企业版)

VB 组合现有的控件,或从中创建自己的控件,用 VB 创建的 ActiveX 控件可以有事件、数据绑定支持、许可证支持、属性页、Internet 特征等多种功能。

3. 多工程(所有版本)

在 VB 的同一个实例中可打开多个工程,这对调试 ActiveX 控件很有用

4. 创建 ActiveX 文档(专业版和企业版)

就像设计传统的 VB 应用程序那样既简单又直观,ActiveX 文档将 VB 应用程序推进到 Internet 和 Intranet 浏览器窗口中。

5. MDI/SDI/ 资源管理器样式的界面选项(所有版本)

VB 6.0 能够创建单文档界面、多文档界面或 Microsoft 资源管理器样式的文档界面应用程序。

6. 向导(所有版本)

在学习版中,应用程序向导是新的,并且安装向导已经增强,它能为标准工程创建从属文件。在专业版中,ActiveX 控件接口向导、ActiveX 文档移植向导、数据窗体向导和属性页向导都是新的。

7. 引用和处理自己的事件(所有版本)

VB 6 部件提供的对象可以引用由其他应用程序的事件。使用 WithEvents 变量,也可以处理由其他应用程序或自己的对象引用的事件。

8. 代码编辑器的增强功能(所有版本)

“块注释和解除块注释”可对文本选择块的每一行添加和删除注释字符。“属性/方法列表”为控件显示一个可用属性的下拉列表。“自动快速信息”为语句和函数显示语法。“边距”指示器标记断点和当前语句。“过程查看”和“全模块查看”按钮使显示模块中选定的过程或全体代码变得更加容易。

9. 多线程和线程安全 ActiveX 部件(专业版和企业版)

这两个功能支持建立可缩放的使用多线程技术的 ActiveX 部件。在分布式应用程序中对未想到的执行过程(非用户界面的元素)建立 ActiveX 部件,这将允许在多线程环境中使用该部件。

10. Internet 部件下载(专业版和企业版)

可以利用安装向导为 Web 上部署的部件打包。

11. 全局对象(专业版和企业版)

为了简化重用代码库的创建过程,VB 允许把代码部件中的对象标记为全局的。

12. 枚举(所有版本)

VB 可以把相关的命名常数组成枚举型进行定义。对于专业版,枚举可以包含在 ActiveX 部件的类型库中,使用该部件的开发者可以使用它。

13. OLE 拖放(所有版本)

VB 6 的多数控件现在都支持 OLE 应用程序之间的拖放操作,例如:把 Word 文档中的内容拖到 TextBox 控件。

其次,VB 6.0 还具有许多新的特性,特别是增强了数据库功能。如:

- 能创建超高速的应用程序的本地代码编辑器。
- 新增了创建 Internet 应用程序的服务器端编程模型。
- 使用新增的创建可重用基于组件的 ActiveX Data Object(ADO)的环境,可简便访问远程数据。
- 集成了可视化浏览、创建、修改数据库方案的企业版 Visual Database Tools。
- 可快速访问 Oracle 和 Microsoft SQL Server 数据库。
- 单独版本的 MSDN Library,是用户使用 Microsoft 技术创建解决方案的基本资源,包括:超过 1.1GB 的技术编程信息、示例代码、技术文章和各种文档。

1.2 启动和退出 VB 6.0

用户在自己的计算机上安装了 VB 6.0 后,就可以启动它并在它的集成环境中编程了。

提示:关于如何安装 VB 6.0,可参阅附录一。

1.2.1 启动 VB 6.0

要创建 VB 应用程序,首先要启动 VB 6 的集成开发环境。方法是在 Windows 95 / 98 或 Windows NT 的桌面上,单击【开始】按钮,然后将鼠标指针依次指向【程序】、【Microsoft VB 6.0 中文版】及其下一级菜单项【Microsoft VB 6.0 中文版】,并在菜单项上单击鼠标左键,如图 1.1 所示。

接着系统开始运行 VB 6.0,然后打开如图 1.2 所示的界面,出现在最前面的是【新建工程】对话框。利用该对话框用户可以创建多种应用程序。在以后章节中,读者会逐渐了解各个图标的作用。

提示:如果想在启动 VB 6.0 时,不显示【新建工程】对话框,可选中【不再显示这个对话框】复选框。

1.2.2 退出 VB 6.0

退出 VB 6.0 的方法很简单,而且有多种,要退出 VB 集成环境,可执行以下操作之一:

1. 单击标题栏上的关闭按钮。
2. 从【文件】菜单中选择【退出】项。
3. 按快捷键【Ctrl + F4】。
4. 单击标题栏的最左边的图标,在打开的控制菜单中选择【关闭】项。

执行上述操作之后,如果有修改过的文件没有保存,VB 会弹出如图 1.3 所示的对话框,提

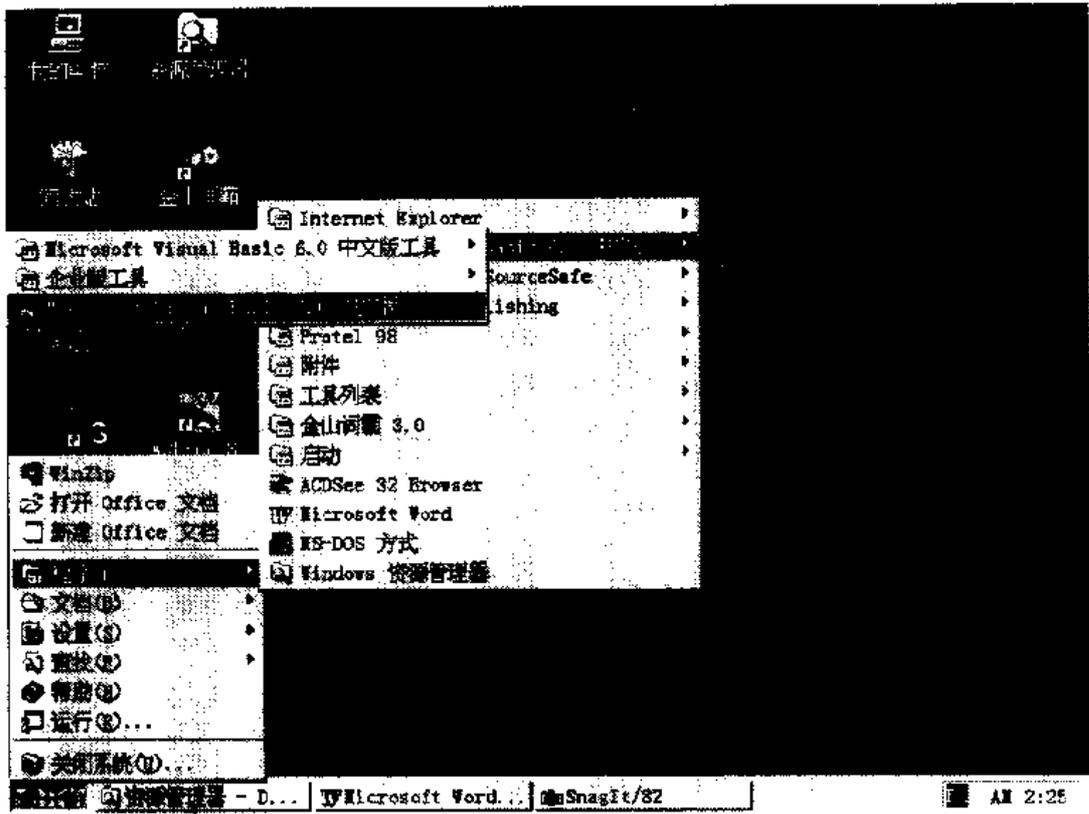


图 1.1 启动 VB 6.0

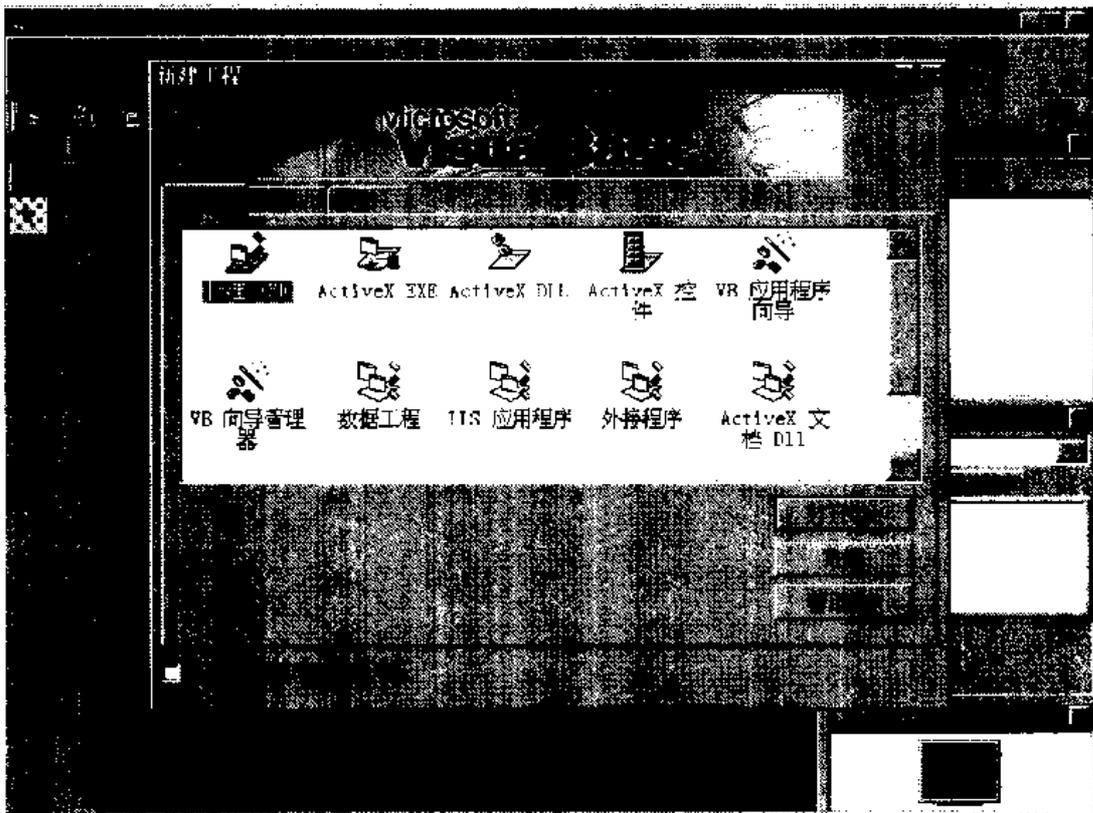


图 1.2 启动 VB 6.0 后的界面

示用户是否保存修改。

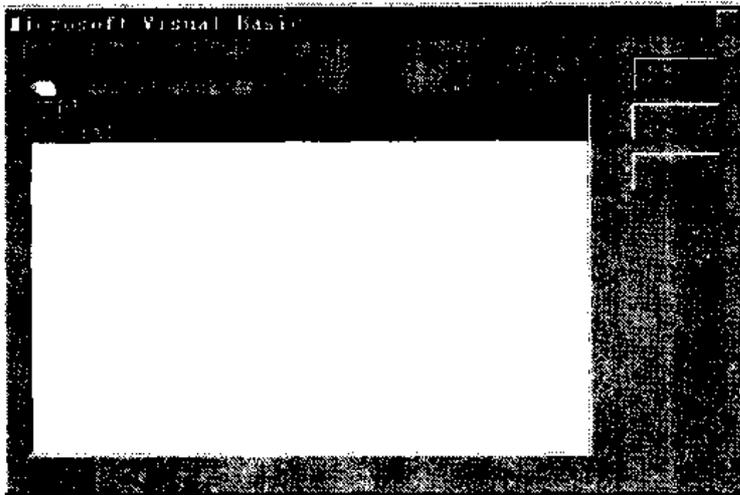


图 1.3 退出 VB 6.0 时弹出的提示对话框

1.3 创建第一个 VB 应用程序

在本节我们将建立一个简单的 VB 程序,从而建立关于 VB 编程最基本的概念。在下一节将介绍这些基本概念。

VB 的编程,不同于 DOS 下编程,不是一行接一行的编写代码,而是首先创建应用程序界面,即完成“Visual”,然后再编写程序代码,即完成“Basic”编程。

具体来讲,创建 VB 应用程序有三个主要步骤:

1. 创建应用程序界面。
2. 设置属性。
3. 编写代码。

为了说明这一实现过程,按照以下步骤创建一个简单应用程序,该应用程序由一个文本框和两个命令按钮组成,一个叫【画圆】,另一个叫【清除】。单击【画圆】按钮,文本框中会出现“在窗体上画圆”字样,同时,窗体上显示一个圆,单击【清除】按钮,文本框中出现“圆消失了”字样,同时,窗体上的圆消失。

1. 创建应用程序界面

在上一节我们已经看到,启动 VB 6.0 后,会打开【新建工程】对话框,选择【标准 EXE】图标后,单击【确定】按钮,系统建立一个默认的名叫“Form1”的窗体,如图 1.4 所示。窗体是创建应用程序的基础,通过使用窗体可将窗口和对话框添加到应用程序中。

(1) 用鼠标单击工具箱上的 CommandButton(命令按钮)控件,如图 1.5 所示。将鼠标指针移到窗体上,该指针变成十字线。

(2) 将十字线放在控件的左上角处,拖动十字线画出适合需要的控件大小的方框(拖动的意思是按住鼠标左键,用鼠标指针移动对象),释放鼠标按钮,控件出现在窗体上。如图 1.6 所示。

提示:在窗体上添加控件的另一个简单方法是双击工具箱中的控件按钮,这样会在窗体中央创建一个尺寸为缺省值的控件;然后再将该控件移到窗体中的其他位置。

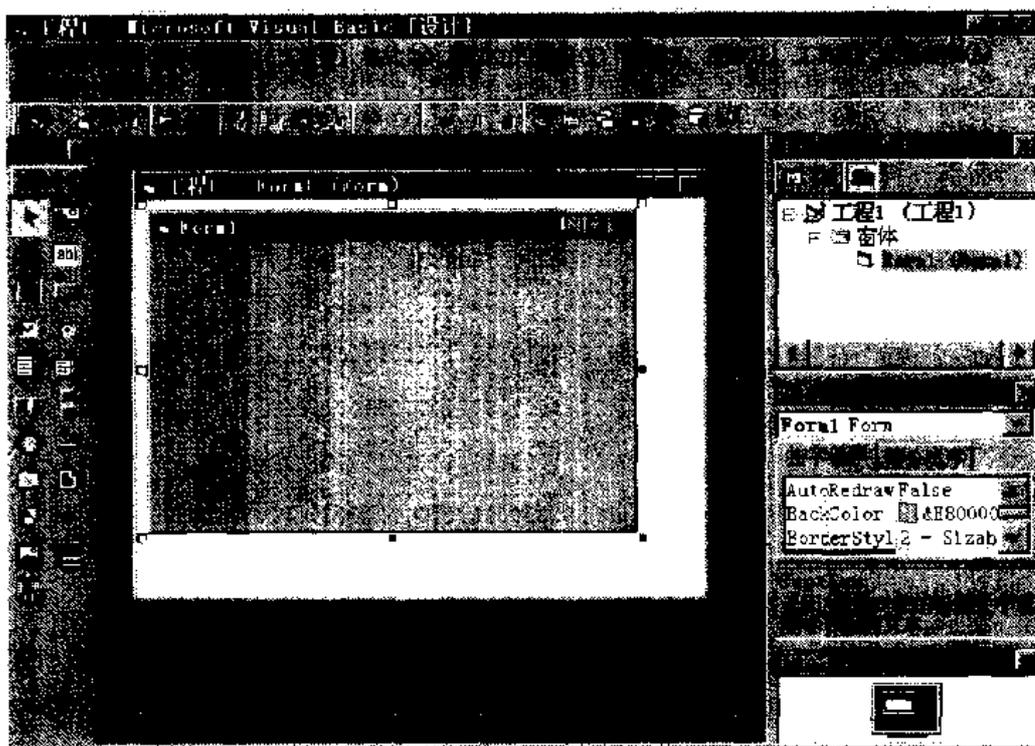


图 1.4 VB 6.0 的集成开发环境

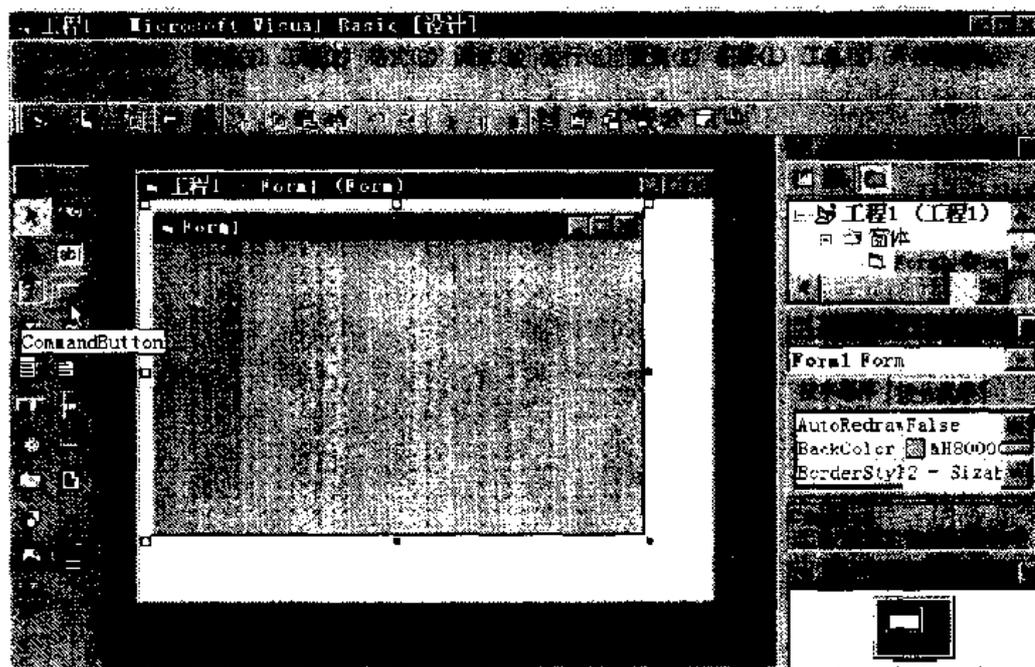


图 1.5 用工具箱添加命令按钮控件

(3) 按照第二步的方法,在窗体上添加第二个按钮。

(4) 用鼠标单击工具箱上的 TextBox(文本框)控件,将鼠标指针移到窗体上,该指针变成十字线,将十字线放在控件的左上角处,拖动十字线画出适合需要的控件大小的方框。

现在已生成了应用程序的界面,如图 1.7 所示。

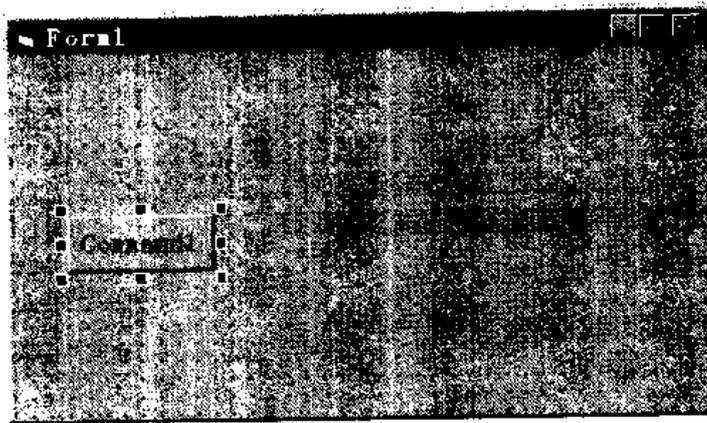


图 1.6 向窗体上添加的命令按钮

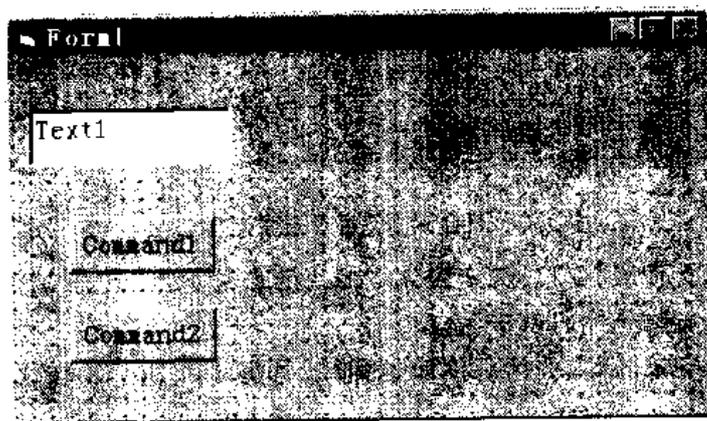


图 1.7 整个应用程序界面

2. 设置属性

下一步是给创建的对象设置属性。属性窗口给出了设置所有的窗体对象属性的简便方法。

(1) 创建的两个按钮上有 Command1, Command2 的名字, 我们可以在属性窗口中改变显示的文字。首先单击要设置属性的按钮(如 Command1), 选中这个按钮(此时按钮周围有 8 个方块状的大小句柄), 然后属性窗口中就会出现按钮 Command1 的属性列表。

(2) 从属性列表选定 Caption 属性, 并将右列的“Command1”设为“画圆”, 这时按钮上显示“画圆”, 如图 1.8 所示。

(3) 用同样的方法将按钮 Command2 的 Caption 属性设为“清除”, 将文本框 Text1 的 Text 属性设置为空(即删除 Text 属性中的字符), 将窗体的 Caption 属性设为“我的第一个应用程序”。

注意: 在设置对象的属性时, 首先要选中该对象, 然后在属性窗口中设置相应的属性。

提示: 如果没有显示属性窗口, 可在【视图】菜单中选择【属性窗口】命令, 或单击工具栏上的【属性窗口】按钮来打开属性窗口。

3. 编写代码

代码编辑器窗口是编写应用程序的 VB 代码的地方。使用代码编辑器, 可以快速查看和编辑应用程序代码的任何部分。要打开代码窗口可以用多种方法:

- 双击要编写代码的窗体或控件。

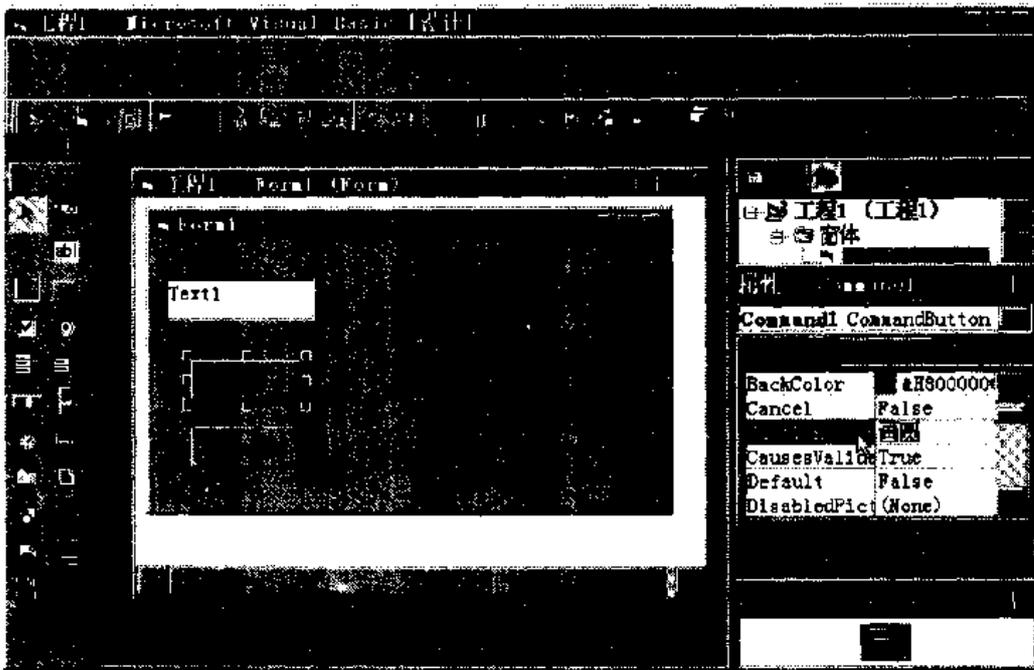


图 1.8 设置对象属性

- 从【工程资源管理器】窗口,选定窗体或模块的名称,然后单击【查看代码】按钮。
- 从【视图】菜单选择【代码窗口】选项。

(1) 双击【画圆】按钮,出现的代码编辑器窗口,如图 1.9 所示。在代码编辑器窗口,出现如下代码:

```
Private Sub Command1_Click()
End
```

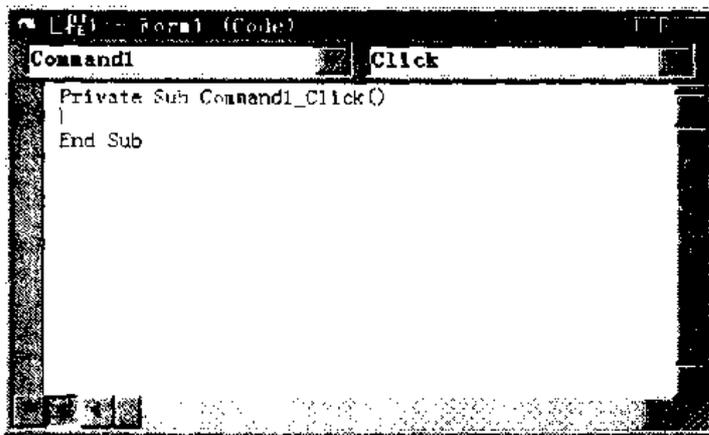


图 1.9 代码编辑器窗口

(2) 在这两条语句之间,添加如下的程序代码:

```
Form1.Circle (2400,1000),800
Text1.Text = " 在窗体上画圆"
```

(3) 在左边的“对象”列表框中,选定“Command2”选项(如图 1.10 所示),在右边的“过程”列表框中,选择“Click”选项(如图 1.11)所示。此时,代码编辑器窗口中会添加这样的语句:

```
Private Sub Command2_Click()
```

```
End
```

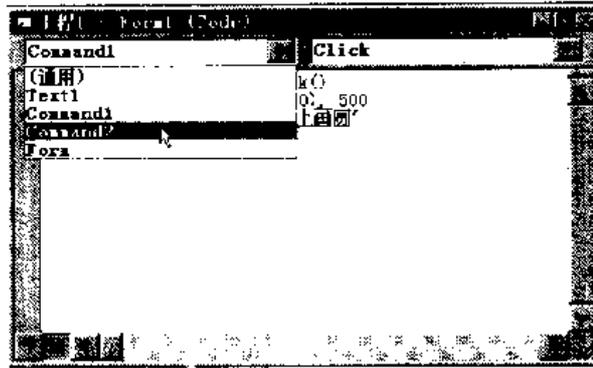


图 1.10 选择对象

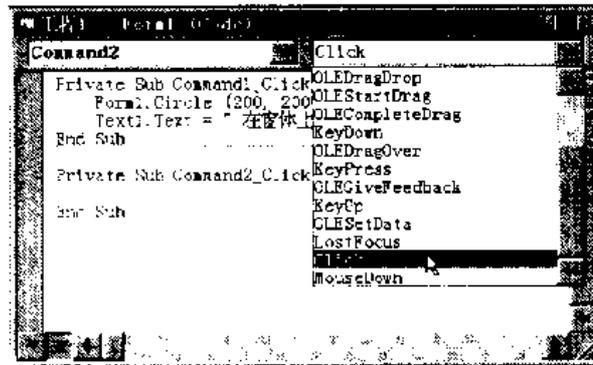


图 1.11 选择事件过程

在这两条语句之间输入下面的代码：

```
Form1.Cls
```

```
Text1.Text = "圆消失了"
```

至此，整个程序代码为：

```
Private Sub Command1_Click ()  
    Form1.Circle (200,200),500  
    Text1.Text=" 在窗体上画圆"  
End Sub
```

```
Private Sub Command2_Click()  
Form1.Cls  
Text1.Text = "圆消失了"  
End
```

4. 运行应用程序

检查一下代码输入是否有误，确保输入正确后，可以从【运行】菜单中选择【启动】选项，或者单击工具栏中的【启动】按钮，或按 F5 键来运行应用程序，结果如图 1.12 所示。

单击【画圆】按钮后，窗体显示如图 1.13 所示。单击【清除】按钮后，窗体将恢复到如图 1.14 的形式。

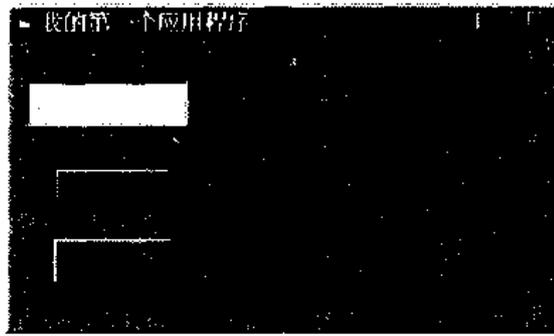


图 1.12 程序运行结果

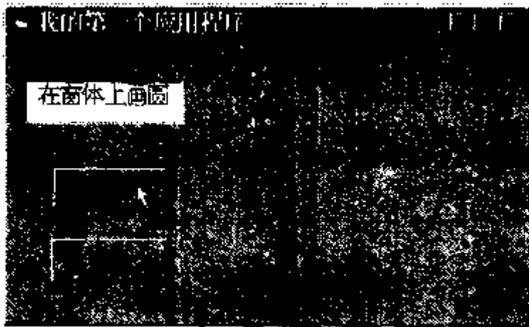


图 1.13 单击【画图】按钮后的窗体

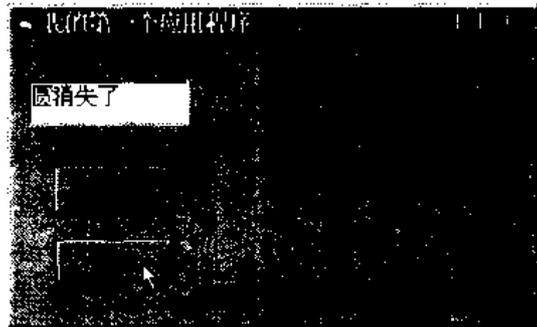


图 1.14 单击【清除】按钮后的窗体

1.4 对象与事件的基本概念

对象是面向对象程序设计的核心,对象的概念来源于日常生活中,如:我们学习用的钢笔、书本等都是对象,在刚刚建立的应用程序中,文本框和命令按钮也都是对象。

对象都有自己的状态和行为,如:钢笔的大小、颜色就是它的状态,而书本可以折叠起来,就是一种行为。在 VB 的程序设计中,对象的状态用数据表示,称为对象的属性。对象的行为用程序代码来实现,称为对象的方法。而事件就是能被对象所识别的动作,如:使用鼠标单击和双击就是常见的事件。

VB 的窗体和控件是具有自己的属性、方法和事件的对象。可以把属性看作一个对象的性质,把方法看作对象的动作,把事件看作对象的响应。

1.4.1 对象的属性和方法

日常生活中的对象,如气球同样具有属性、方法和事件。气球的属性包括可以看到的一些性质,如它的直径和颜色。其他一些属性描述气球的状态(充气的或未充气的)或不可见的性质,如它的寿命。通过定义,所有气球都具有这些属性;这些属性也会因气球的不同而不同。

气球还具有本身所固有的方法,如:充气方法、放气方法和上升方法等,所有的气球都具备这些能力。

1. 设置对象属性

属性是对象中的数据,用来表示对象的状态,设置对象属性可以有两种方法。

(1) 在程序设计时设置对象属性

如果要在程序设计时设置对象属性,那么可在属性窗口中进行。首先选中要修改属性的

对象,然后在属性窗口中选择要修改的属性,在列表右列输入或选择新的数据即可。如:在上面的例子中,我们将按钮 Command1 和 Command2 的 Caption 属性分别设置为“画圆”和“清除”,将窗体的 Caption 属性设置为“我的第一个应用程序”。

(2) 在程序运行时设置对象属性

如果要在程序运行时设置对象的属性,应使用程序代码完成,它是一条赋值语句,格式为:

对象名.属性 = 属性值

其中,“对象名.属性”是 VB 引用对象属性的方法。例如在上面的例子中,我们使用了赋值语句给文本框对象的属性进行了设置:

```
Text1.Text = “在窗体上画圆”
```

```
Text1.Text = “圆消失了”
```

注意:程序代码的语法首先是对象 (Text1),接着是属性 (Text),然后是赋值 (“圆消失了”)。

Caption 和 Name(名称)属性的区别:

(1) Caption 属性是许多对象都具有的属性,它的主要作用是设置对象在窗体上显示的文字,向用户提示该对象的功能。对象刚刚创建时,Caption 属性与 Name 属性相同。在程序设计时,一般都要设置对象的 Caption 属性,例如:将按钮的 Caption 属性由“Command1”改为“画圆”,标明单击此按钮将在窗体上画圆。

(2) Name 属性是对象的名字,在程序中用到对象的属性和方法时,都要用到它,例如在上面的例子中,引用文本框的属性时:

```
Text1.Text
```

其中 Text1 就是文本框对象的名字。

提示:如果将文本框的 Name 属性改为:txtDisplay,那么引用 Text 属性时应这样写:

```
txtDisplay.Text
```

2. 对象的方法

方法决定对象的动作,它必须由程序代码实现。使用对象的方法时,应为如下格式:

对象.方法

它的语法与属性的语法相似,对象(一个名词)后面紧跟着方法(一个动词)。在上面的例子中我们使用了窗体的两种方法,第一个方法是在窗体上画圆:

```
Form1.Circle (2400,1000), 800
```

该方法有一些附加项,称为参数,表示画圆的圆心(2400,1000)和半径 800。

提示:一些方法可能有一个或多个参数,它们对执行的动作做进一步的描述。

另一个方法是清除窗体上的内容:

```
Form1.Cls
```

这个方法没有参数。

1.4.2 事件及事件过程

气球有预定义的对某些外部事件的响应。例如,气球对刺破它的事件响应是放气,对放手事件的响应是升空。

“可视化”和“事件驱动”是使用 VB 进行 Windows 程序设计的关键,在程序中流动的是事件而不是数据。在 VB 中,事件就是能被对象识别的动作,键盘的输入、鼠标的移动和单击、窗体的装载等,都是事件。

不同的对象所识别的事件类型不同,如命令按钮可以识别鼠标单击(Click)、双击(DbClick)等事件。但相同的事件发生在不同对象上,产生的效果不一定相同,例如上面的例子中,就是在不同的命令单击时,得到的结果也不一样。

每个对象对每个可以识别的事件都有一个事件过程,事件过程的语法如下:

```
Sub 对象名_事件()  
    处理事件的程序代码  
End Sub
```

在上一节的示例中有两个事件过程,分别处理鼠标在不同按钮上单击的事件,第一个事件过程是在窗体上画圆且在文本框中显示“在窗体上画圆”字样:

```
Private Sub Command1_Click()  
    Form1.Circle(200,200),500  
    Text1.Text = "在窗体上画圆"  
End Sub
```

另一个事件过程是清除窗体上的圆且文本框中显示“圆消失了”字:

```
Private Sub Command2_Click()  
    Form1.Cls  
    Text1.Text = "圆消失了"  
End
```

1.5 本章小结

本章向读者讲述了 VB 的基本概念以及 VB 6.0 中文版的功能特点,然后通过一个简单的程序实例介绍有关对象、属性、方法、事件、过程等概念,使读者对 VB 编程有一个感性认识,同时也能激起继续学习 VB 6.0 中文版的兴趣。

学完本章后,读者应掌握以下内容:

1. 什么叫 VB,VB 6.0 中文版有那些功能特点。
2. 如何启动和退出 VB 6.0。
3. 了解编写 VB 程序的基本过程。
4. 了解对象、对象的属性、对象的方法的概念。
5. 了解事件和事件过程的概念。

第 2 章 VB 6.0 的集成开发环境

VB 6.0 的工作环境是集成开发环境(IDE),它集 VB 程序的编写、修改、调试和生成于一体,使用非常方便。了解 VB 集成开发环境,对于进一步学习 VB 编程将是十分重要的。本章主要介绍集成开发环境中各个组成元素的作用和使用方法,同时还介绍如何设置工作环境,以满足用户需要。

2.1 认识 VB 6.0 的集成开发环境

在第一章,我们已经与 VB 6.0 的集成开发环境接触了不少,图 2.1 显示了 VB 6.0 界面中一些最经常出现的窗口元素。

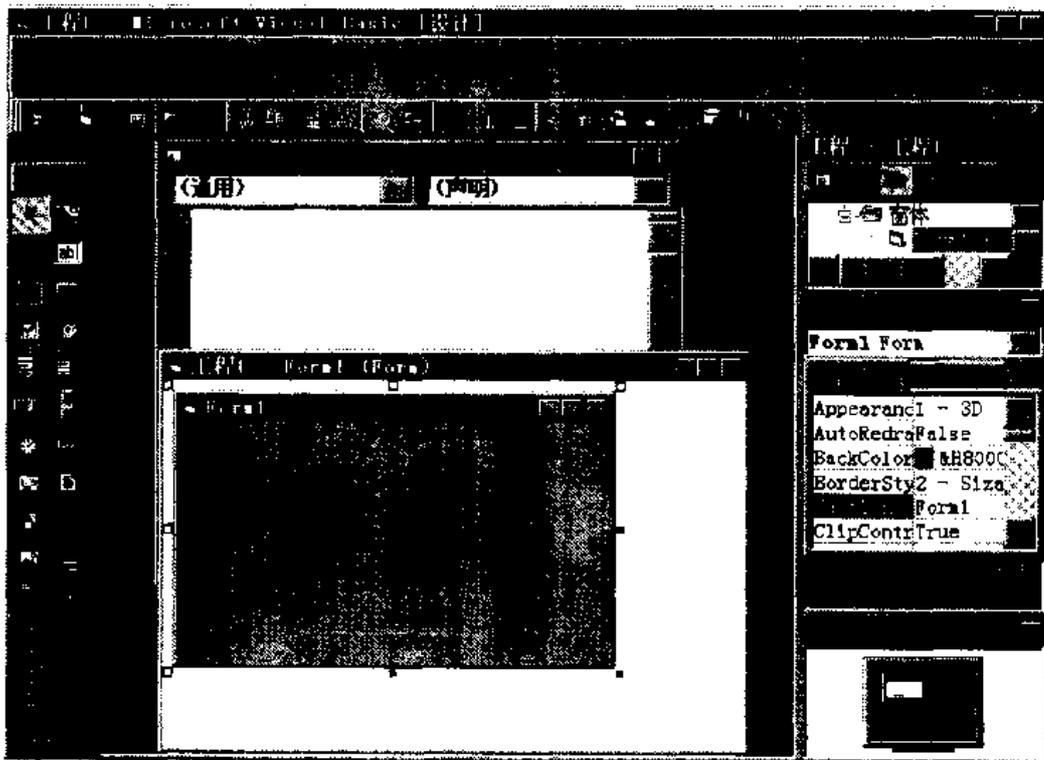


图 2.1 VB 6.0 集成开发环境

从图 2.1 可以看出,VB 6.0 的集成开发环境主要包括主窗口(有标题栏、工具栏、菜单栏组成)、工具箱、工程管理器窗口、属性窗口、窗体布局窗口、窗体设计器窗口、代码编辑器窗口等。此外,还有许多未显示在界面上的窗口,包括:调试窗口(立即、本地和监视窗口)、调色板窗口、菜单编辑器窗口、对象浏览器窗口等。下面对各个组成元素作一介绍:

2.1.1 主窗口

与其他 Windows 应用程序一样,VB 6.0 主窗口由标题栏、工具栏、主菜单三部分组成,如图

2.2 所示。

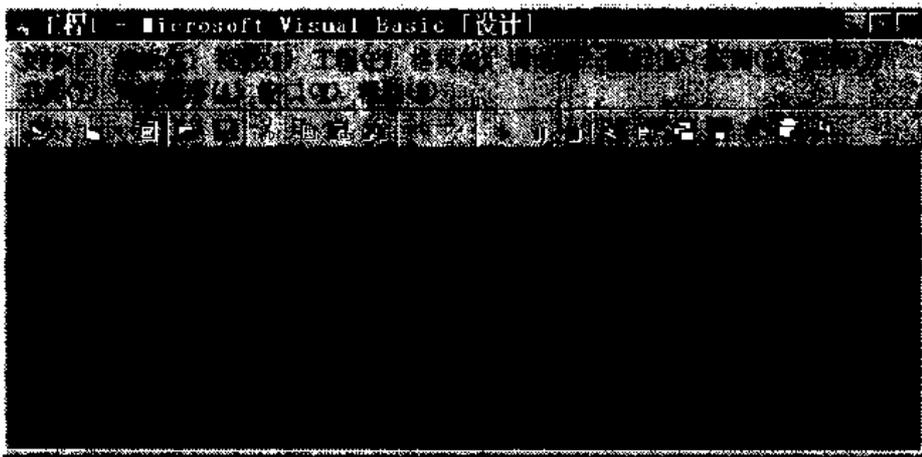


图 2.2 VB 6.0 的主窗口

窗口的最上面是标题栏,标题栏中的“工程 1”是当前打开的工程文件名,由于未给此工程命名,VB6.0 使其缺省为“工程 1”;方括号中的“设计”表示当前处于设计状态,当用户运行程序或调试程序时,会分别是“运行”或“中断”。标题条左边是 Windows 窗口的控制菜单,右边是窗口最小化、窗口最大化、关闭窗口按钮。

标题栏下面是主菜单,分别有【文件(F)】、【编辑(E)】、【视图(V)】、【工程(P)】、【格式(O)】、【调试(D)】、【运行(R)】、【查询(U)】、【图表(T)】、【工具(T)】、【外接程序(A)】、【窗口(W)】、【帮助(H)】,共 13 项菜单。附录二中列出了常用的菜单项及其功能。

工具栏位于主菜单下面,工具栏中的按钮所代表的命令都是我们经常用到的,工具栏的设置给用户提供了一种访问命令的快速方法,单击工具栏上的按钮,将执行按钮代表的命令的功能。

提示: VB 6.0 缺省显示的工具栏为标准工具栏,如果要显示或隐藏其他工具栏,可通过【视图】菜单的【工具栏】选项进行选择。

2.1.2 窗体编辑器窗口(Form 窗口)

窗体编辑器作为用户自定义窗口,用来设计应用程序的界面,如图 2.3 所示。用户可以在窗体中添加控件、图形图片和菜单,来创建所期望的外观。应用程序中每个窗体都有自己的窗体编辑器窗口,一个应用程序可以有多个 Form 窗口,各窗口都有不同的名字,以免发生混淆。缺省时,被命名为 Form1, Form2, Form3 等。当然用户也可以自己命名这些窗口名。所有的 Form 都是可视的,但用户也可把这些窗口设为不可视,这要根据实际情况而定。

2.1.3 工具箱

工具箱中包含标准的 VB 控件和 ActiveX 控件,如图 2.4 所示。除了缺省的工具箱布局外,还可通过从快捷菜单中选定【添加选项卡】,并在结果选项卡中添加控件来创建自定义布局。工具箱中的内部控件如表 2.1 所示,以后章节将详细介绍各控件的具体用法。

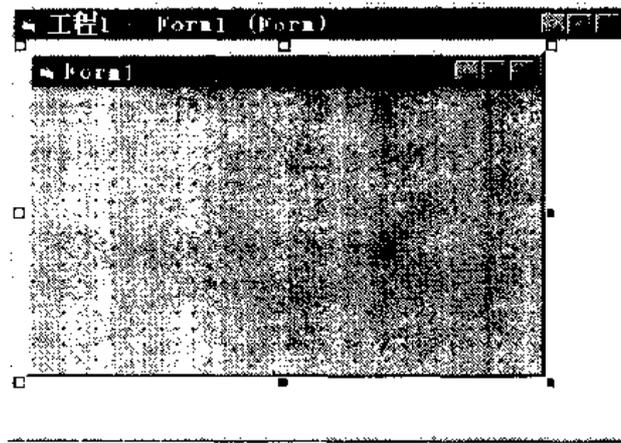


图 2.3 窗体编辑器窗口

表 2.1 VB 工具箱中的内部控件

图 标	类 名	名称及功能
		用 J 选中界面上的控件
	PictureBox	图片框,用于装载图片
	Label	标签,显示文本但不能被编辑
	TextBox	文本框,用于运行时显示用户输入的信息
	Frame	框架,将其他控件分成可标识的控件组
	CommandButton	命令按钮,用于开始、中断或结束一个过程,单击它将调用已写入 Click 事件过程中的命令
	CheckBox	复选框,用于选择一个或多个选项
	OptionButton	单项按钮,用于选择一个符合要求的选项
	ComboBox	组合框,同时具有文本框和列表框的功能
	ListBox	列表框,显示项目列表以供用户选中一个或多个项目
	HScrollBar	水平和垂直滚动条。对于不能自动提供滚动条的控件,允许用户为它们添加滚动条(这些滚动条与许多控件的内建滚动条不同)
	VScrollBar	
	Timer	定时器,按指定时间间隔执行定时器事件
	DriveListBox	驱动器列表框,显示有效的磁盘驱动器并允许选择
	DirListBox	目录列表框,显示目录和路径并允许选择
	FileListBox	文本列表框,显示文本列表并允许选择
	Shape	形状,向窗体、框架或图片框添加矩形、正方形、椭圆或圆形
	Line	线形,在窗体上添加线段
	Image	图像,显示位图、图标或 Windows 图元文件、Jpeg 或 gif 文件;单击时类似命令按钮
	Data	数据控件,能与现有数据库连接并在窗体上显示数据库中的信息
	OLE	OLE 容器,将数据、周期放入 VB 应用程序中



图 2.4 工具箱

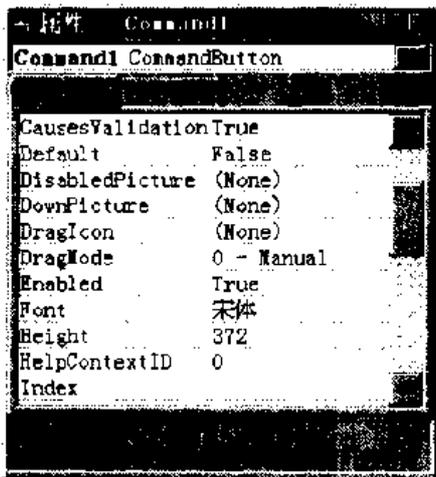


图 2.5 属性窗口

2.1.4 属性窗口

属性窗口用于列出选定窗体和控件的属性设置值,同时也可在属性窗口中进行属性值的设定。属性窗口如图 2.5 所示。

属性窗口最上面为标题栏,标题栏下面的文本框为对象框。在对象框中通过下拉箭头可选择不同的对象。在对象框下面有两个选项卡,分别代表显示属性的两种方法:【按字母序】、【按分类序】。选项卡下面即为属性列表,它分为两栏,左边一栏列出各属性名称,右面一栏列出相对应的属性值。当选定某一属性时,将光标移到右面对应的设置属性区,单击鼠标,将出现一活动光标,若想改变设置,先将原属性值删除,再键入新设置的值后按

回车即可。也有一些属性值可以通过对话框获取,或者选取一个可能值。在属性窗体最下面是提示框,当用户选中某一属性时,提示框会简要地给出该属性的功能。

2.1.5 工程资源管理器窗口

工程是用于建立一个应用程序的所有文件组成的集合。在 VB 中用工程管理器来管理工程中的窗体和各种模块。工程资源管理器窗口如图 2.6 所示。工程资源管理器采用资源管理器样式的界面,列出当前工程中所有的文件,包括窗体模块、类模块、标准模块、资源文件及 ActivX 文档,各种文件的位置以很鲜明的层次显现出来。当创建、添加或从一个工程中移除可编辑文件时,工程管理器窗口都会发生相应的变化。在窗口内,双击文件名,即可打开相应的文件。

提示:工程资源管理器窗口的标题栏下面有三个工具按钮,分别是:【查看代码】、【查看对象】和【切换文件夹】,单击它们可以查看代码、查看对象和切换文件夹。

2.1.6 代码编辑窗口

应用程序中的每一个窗体和模块都有独立的代码窗口。代码窗口用于编写、显示和修改 VB 代码,用户可同时打开多个代码窗口。代码窗口如图 2.7 所示。

窗口中含有对象框、过程/事件框、代码编辑区、过程查看图标和全模查看图标。对象框用于显示对象的名字,如图中“Form1”,单击右边下拉箭头可选择其他对象。

过程/事件框,用以列出与对象有关的事件。当选择一个事件时,代码框就显示该事件过程。

代码编辑区位于窗口正中央,它显示了与对象和事件相关的代码,用户可以方便地以文本方式进行编辑代码过程。

查看图标位于窗口左下角,它的作用是使代码编辑区每次只显示一个过程代码。在它旁边的是全模查看图标,它用于在代码编辑区内显示所有过程。

打开代码窗口的方法有多种:

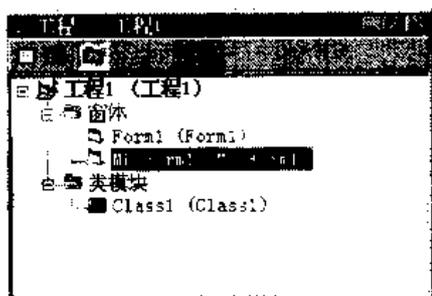


图 2.6 工程资源管理器窗口

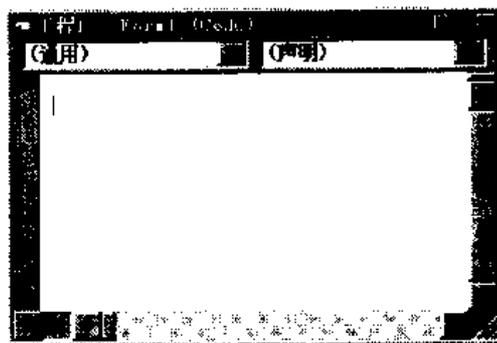


图 2.7 代码编辑窗口

1. 双击一个控件或窗体,这种方法最常用。
2. 从工程管理器中选择要查看的窗体或模块,再单击【查看代码】按钮。
3. 用 F7 键。
4. 选择【视图】菜单中的【代码窗口】菜单项。

2.1.7 对象浏览器

对象浏览器用于显示对象库和过程中可用的类、事件、方法、属性和常量。通过它可以方便地寻找和使用用户建立的对象,也可以使用其他程序中的对象。选择【查看】菜单中的【对象浏览器】选项可以显示对象浏览器。对象浏览器如图 2.8 所示。

各个组成部分说明如下:

(1) 属性/库显示框。位于标题栏下面,图中显示<所有库>,单击右边下拉箭头,可以在列表中选择其他库或属性。可通过【工程】菜单中的【引用】选项,以对话框的方式往此列表中添加对象库或类库。

(2) 搜索文本框。位于属性/库显示框的下面,在此可以输入要搜索的文本,输入字符串可以使用标准的 VB 通配符。

(3) 类列表。显示库或工程中所有可用的类,它以<全局>开始,在类列表中可以通过鼠标或键盘选择某一特定的类。

(4) 成员列表。当在类列表中选择一个类后,可以显示它的成员,右击成员列表,通过快捷菜单中的【组成员】选项可改变显示的顺序。

(5) 明细板。在窗口最下面,显示成员的定义。

(6) 向右(左箭头),向前(右箭头)按钮。向右按钮为返回上一次选择的类和成员,每点一次就向上一步,直到最开始处。向前按钮为重复原来选择的类和成员。

(7) 复制到剪切板按钮。将选中对象复制到剪切板。

(8) 查看定义按钮。单击此按钮,使光标移到代码窗口内成员或类的定义处。

(9) 帮助按钮。类和成员的在线帮助,F1 键有同样功能。

(10) 搜索按钮。搜索与搜索文本框中指定的文本匹配类库或属性、方法、事件和常量。

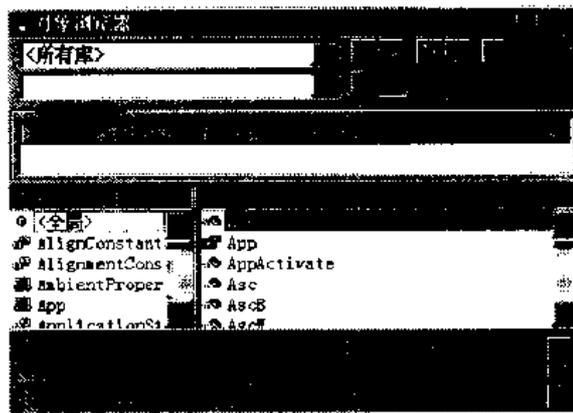


图 2.8 对象浏览器窗口

(11) 显示/隐藏结果列表,显示搜索结果。

2.1.8 窗体布局窗口

窗体布局窗口的作用是:用一个小屏幕来显示窗体在屏幕上的位置。显示器用以表示整个屏幕,显示器中的白色部分表示窗体,它们的相对位置就是运行时窗体与屏幕的相对位置。可以通过移动窗体来调整窗体布局。这个窗口在多窗体应用程序(MDI)中很有用,通过调整各窗口之间的位置关系,以达到最佳的视图效果或目的。窗体布局窗口如图 2.9 所示。

2.1.9 调试窗口

调试窗口包括【立即】、【本地窗口】和【监视】,这些窗口只有在运行应用程序时才有效。

【立即】窗口如图 2.10 所示。可在【立即】窗口内直接输入 VB 语句,还可以用立即窗口来测试有问题或新写的过程代码;在执行一个应用程序时查询和更改某个变量值;在程序运行时查看调试输出;在执行一个应用过程中查询或更改一个属性值;调用任何在过程代码中想调用的过程。

提示:在立即窗口内键入或粘贴一行代码,然后按回车键即可执行该代码。

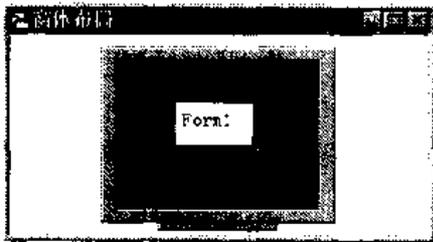


图 2.9 窗体布局窗口

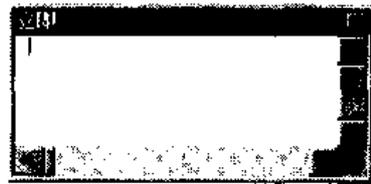


图 2.10 【立即】窗口

【本地】窗口如图 2.11 所示。在【本地】窗口内可自动显示所有在当前过程中的变量声明及变量值。

- 【表达式】字段列出变量的名称。
- 【值】字段列出对应变量的值,可以对【值】字段中的数值进行编辑。
- 【类型】字段列出变量的类型,用户不能在此处编辑数据。

【监视】窗口是用来监视各种变量和表达式的,如图 2.12 所示。通过将变量和表达式加到监视表达式列表中,就可选出想要 VB 监视的表达式。

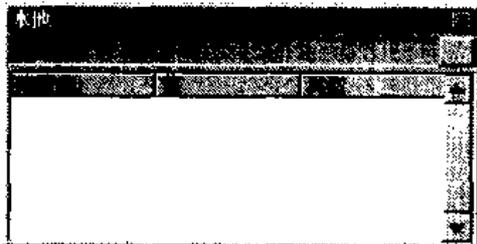


图 2.11 【本地】窗口



图 2.12 【监视】窗口

• 【表达式】字段列出监视表达式,并在最左边列出监视图标。可通过(调试)菜单“中的【添加监视】选项加入一系列监视表达式。

- 【值】字段列出表达式的值。
- 【类型】字段列出表达式的类型。
- 【上下文】字段列出监视表达式的内容。

提示：拖动一个变量到监视窗口，就能使该变量加到列表中。

2.1.10 菜单编辑窗口

菜单编辑窗口严格意义应称为【菜单编辑器】对话框，是用来增删和修改一个窗体上的菜单的。可通过【工具】菜单中的【菜单编辑器】选项来打开该窗口，在本书第八章，将对菜单编辑器作详细介绍。

2.1.11 调色板窗口

调色板窗口如图 2.13 所示。调色板窗口可通过【查看】菜单中的【调色板】选项来激活。用户可以通过调色板来改变窗体或控件的颜色，并可建立自定义颜色。



图 2.13 调色板窗口

窗口的左上面有两个方框，上面的正方形方框显示当前选定的前景颜色和背景颜色，用鼠标单击内部小方框，所选颜色就成为指定对象前景色，单击外部大方框，所选颜色就成为指定对象的背景色。下面的长方形方框用来显示着色效果，字母 Aa 的颜色即为前景色，后面的颜色即为背景色。窗口中间是一组可供用户选择的几十种颜色，若用户对这些颜色不满意，可先单击中间颜色框的最下面一排中的任一框，然后单击窗口右边【定义颜色】按钮来定义自己满意的颜色。

提示：改变对象的颜色还可以通过【属性】窗口来实现。

2.2 工作环境的设置

针对程序设计者的不同习惯，VB 提供了一些设置工作环境的功能，用来调整适合用户的程序开发环境。要对各项功能进行设置，可选择【工具】菜单的【选项】，打开【选项】对话框，在该对话框中有六个选项卡，用户可以对编辑器、编辑器格式、环境等进行设置，以适合自己的爱好。下面对各个选项卡逐一介绍。

2.2.1 【编辑器】选项卡

打开【选项】对话框后，缺省选中的选项卡就是【编辑器】，如图 2.14 所示。该选项卡中一些重要项目的说明如下：

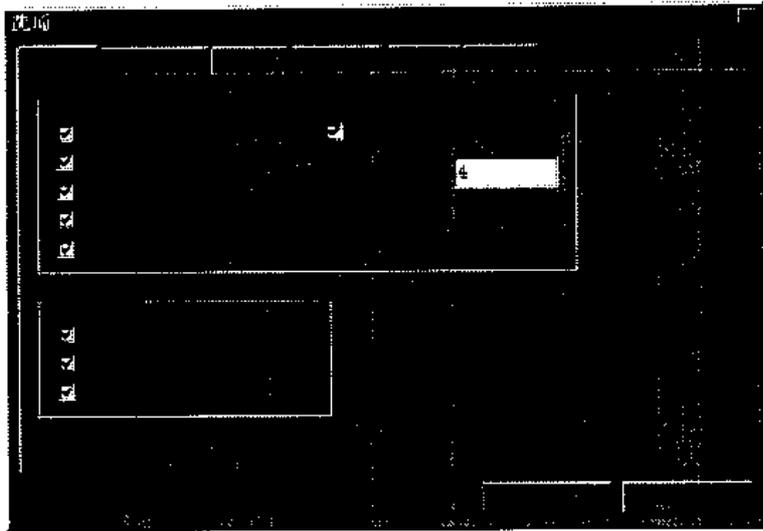


图 2.14 【编辑器】选项卡

1. 【代码设置】区

(1) 【自动语法检测】复选框

选中该复选框后,用户如果完成一程序代码的输入,转到其他行时,VB 会自动对此行程序代码进行语法检查。一旦出现语法错误,就会弹出一个消息框,提示用户出错信息,如图 2.15 所示。如果没有选中该复选框,出现语法错误时,将不显示消息框,但也会将该行代码以红色字体显示,以此提示用户。

(2) 【要求变量声明】复选框

选中该复选框后,用户开始进入代码编辑器时,VB 自动在模块声明段添加语句:

```
Option Explicit
```

此时,如果用户在程序中使用了未经声明的变量,程序一旦运行 VB 就弹出消息框并将该变量反白显示以提示用户,所以用户应选中该复选框,防止由于不小心将变量输入错误。

(3) 【自动列出成员】复选框

选中该复选框后,用户如果在代码编辑器中输入控件的名称时,输入句点后,VB 会自动弹出该控件在运行模式下可用的属性和方法,如图 2.16 所示。此时可以选择某个属性或方法后,再按回车键,即可将该项目插入到当前位置,也可双击该项目将其插入。

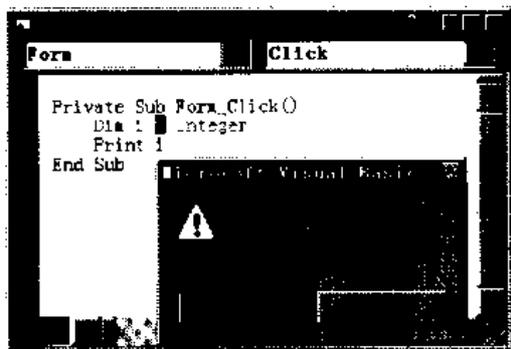


图 2.15 语法错误消息框

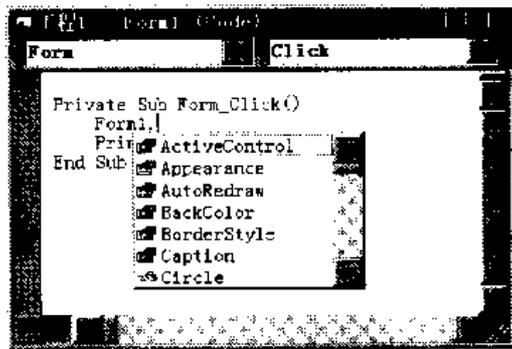


图 2.16 自动列出成员的功能

(4) 【自动显示快速信息】复选框

选中该复选框后,用户在进行代码编辑过程中,输入数组变量或函数名称时(例如:输入 InputBox 函数),VB 会即时提示相关信息,并将正要输入的项目以粗体字显示,如图 2.17 所示。

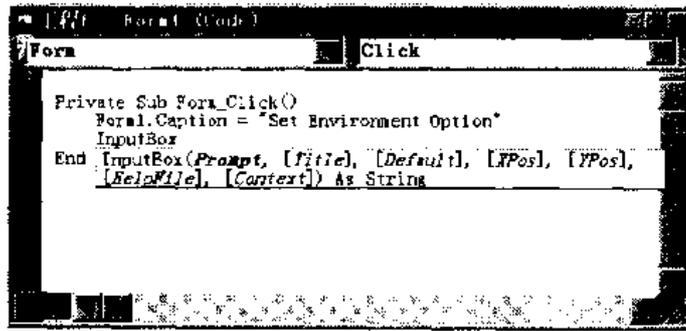


图 2.17 自动显示快速信息的功能

(5) 【自动显示数据提示】复选框

选中该复选框后,用户在调试程序过程中,只要把鼠标指针放在某个变量上,VB 会自动显示该变量的值。

(6) 【自动缩进】复选框

选中该复选框后,用户在进行代码编辑过程中,VB 有自动缩进程序代码的功能。

(7) 【Tab 宽度】文本框

该文本框用于设置按 Tab 键时,光标所跳过的字符间隔。

2. 【窗口设置】区

(1) 【编辑时可拖放文本】复选框

选中该复选框后,如果选中一段文本,就可以用鼠标拖动这段文本到其他的位置。

(2) 【缺省为整个模块查阅】复选框

选中该复选框后,可在代码编辑窗口中看到所有模块的程序代码。

(3) 【过程分隔符】复选框

选中该复选框后,VB 会将各个过程的程序代码以分隔线分开,如图 2.18 所示。

注意:必须在选中【缺省为整个模块查阅】复选框的前提下,【过程分隔线】复选框才可使用,否则程序窗口不会显示所有过程的代码,也就不存在过程分隔线之说。

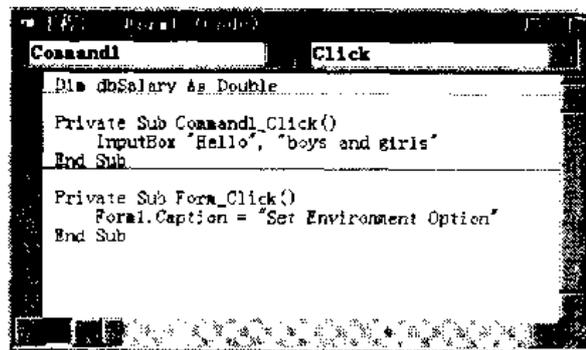


图 2.18 各个过程代码以分隔线分开

2.2.2 【编辑器格式】选项卡

这个选项卡上的各个选项主要用来设置程序代码文本的颜色、字体和大小等,如图 2.19 所示。

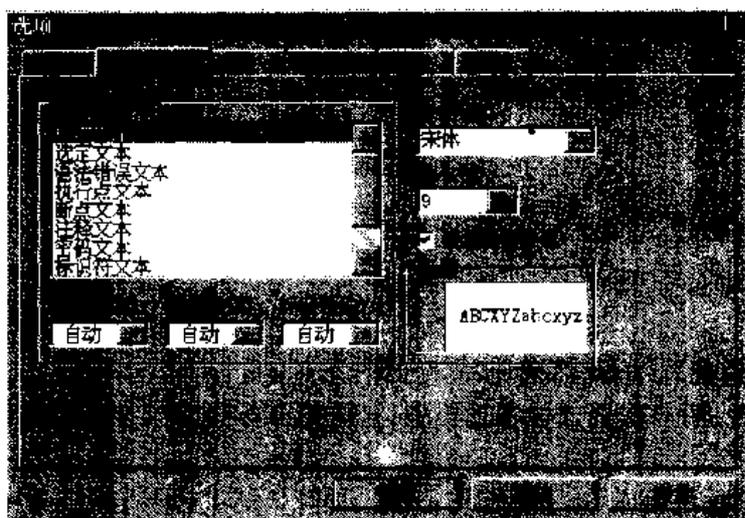


图 2.19 【编辑器格式】选项卡

(1) 【代码颜色】区

用户可在这个区域选择各种情况下代码文本的显示颜色,包括:前景色(即文本的颜色)、背景色和标识色。

(2) 其他选项

用户可在【字体】下拉列表框选择字体,在【大小】下拉列表框选择字体大小,【示例】框会即时显示用户所作的修改。

2.2.3 【通用】选项卡

【通用】选项卡的各个选项主要用来设置一些通用的系统选项,如图 2.20 所示。

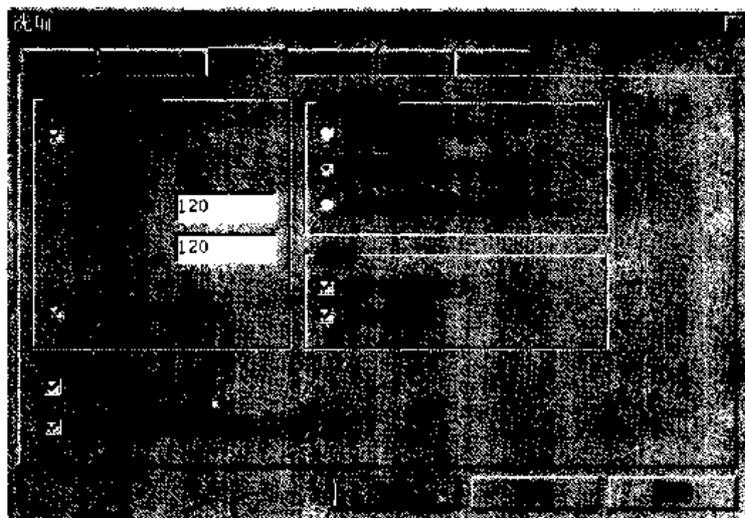


图 2.20 【通用】选项卡

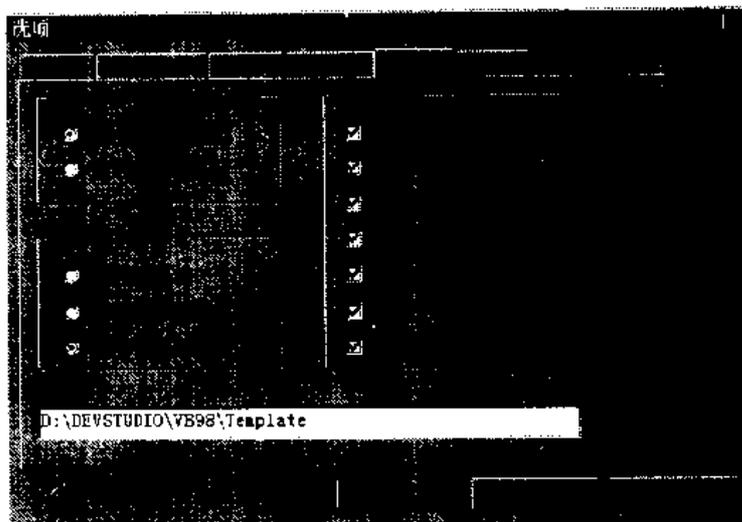


图 2.22 【环境】选项卡

设置。【扩展 HTML 编辑器】文本框显示 HTML 编辑器所采用的应用程序,缺省情况时为 Windows 的记事本程序(Notepad.exe)。

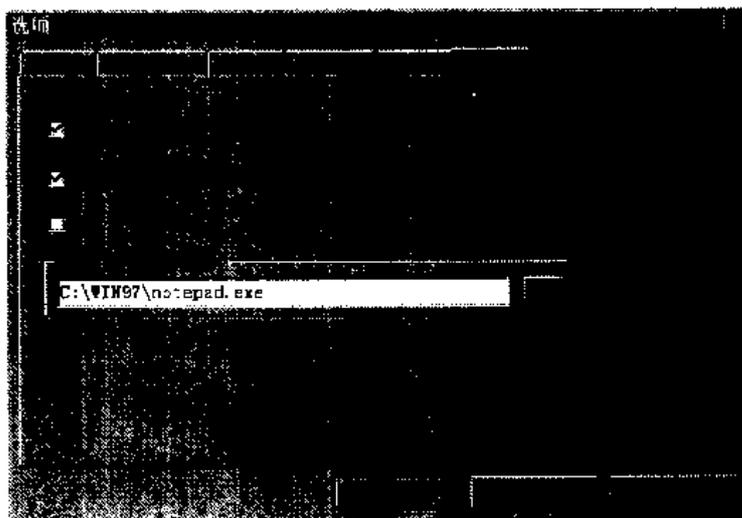


图 2.23 【高级】选项卡

提示:在【高级】选项卡中,用户可单击“...”按钮选择 HTML 编辑器。例如,可选择写字板程序作为 HTML 编辑器。

2.3 本章小结

VB 6.0 的工作环境是集成开发环境(IDE)。本章系统而详细地介绍了此开发环境内各个组成部分,包括:

- 主窗口
- 窗体编辑器
- 工具箱
- 属性窗口
- 工程资源管理器窗口
- 代码编辑窗口
- 对象浏览器
- 窗体布局窗口
- 调试窗口
- 菜单编辑窗口
- 调色板窗口

另外本章还介绍如何利用【选项】对话框设置工作环境,以满足用户需要。

通过学习本章,读者能更深刻理解集成开发环境(IDE)的含义。熟练掌握用集成开发环境来开发程序的方法,为今后方便、灵活地开发程序打下良好的基础。

第 3 章 工程的管理

要创建一个应用程序,用户只需设计好界面元素,然后再编写一些必要的控制代码就可以了。其实在幕后,VB 集成环境还做了很多其他的工作,例如,为这个应用程序创建一系列的文件来保存应用程序的有关信息。这些用来建造应用程序的文件的集合就被称为工程。VB 使用工程来管理构成应用程序的所有不同的文件,在工程的所有部件被汇集在一起并完成代码编写之后,便可以编译工程,创建一个可执行文件。

3.1 工程的基本概念

在开发应用程序时,要使用工程来管理构成应用程序的所有不同的文件。一个工程包括:

- 跟踪所有部件的工程文件 (.vbp)。
- 每个窗体的文件 (.frm)。
- 每个窗体的二进制数据文件 (.frx),它含有窗体上控件的属性数据。对含有二进制属性(例如图片或图标)的任何 .frm 文件都是不可编辑的,这些文件都是自动产生的。
- 每个类模块的一个文件 (.cls),该文件是可选项。
- 每个标准模块的一个文件 (.bas),该文件是可选项。
- 一个或多个包含 ActiveX 控件的文件 (.ocx),该文件是可选项。
- 单个资源文件 (.res),该文件是可选项。

工程文件就是与该工程有关的全部文件和对象的清单,也是所设置的环境选项方面的信息。每次保存工程时,这些信息都要被更新。所有这些文件和对象也可供其他工程共享。当完成工程的全部文件之后,即可将此工程转换成可执行文件 (.exe):从【文件】菜单中,选取【制作工程.exe】。

注意:使用 VB 的专业版和企业版,还可以创建其他类型的可执行文件,例如, .ocx 和 .dll 文件。本章假定所论及的是标准的 .exe 工程。

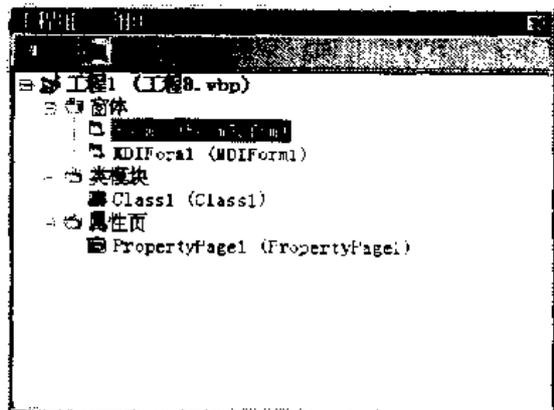


图 3.1 工程资源管理器窗口

3.1.1 工程资源管理器

当创建、添加或从一工程中删除可编辑文件时,VB 会反映工程资源管理器窗口中发生的变化,该窗口包含此工程的当前文件的列表。图 3.1 所示的工程资源管理器窗口,列出了可以纳入 VB 工程的一些文件类型。

3.1.2 工程结构

每次保存工程,VB 都要更新工程文件 (.vbp)。工程文件包含文件列表,它与出现在工程资源管理

器窗口的文件列表相同,工程文件还引用工程中所使用的 ActiveX 控件和可插入对象。

通过双击一个现存工程的图标,或从【文件】菜单中选择【打开工程】,或拖动该文件并放入工程资源管理器窗口,可以打开这个现存工程文件。

1. 窗体模块

窗体模块(具有 .frm 文件扩展名)包含窗体及其控件的正文描述,包括它们的属性设置。它们也含有窗体级的常数、变量和外部过程(事件过程和一般过程)的声明。

2. 类模块

类模块(具有 .cls 文件扩展名)与窗体模块类似,只是没有可见的用户界面。可以使用类模块创建含有方法和属性代码的自己的对象。

3. 标准模块

标准模块(具有 .bas 文件扩展名)可以包含类型、常数、变量、外部过程和公共过程的公共的或模块级的声明。

4. 资源文件

资源文件(具有 .res 文件扩展名)包含着无需重新编辑代码便可以改变的位图、字符串和其他数据。例如,如果计划用一种外语将应用程序本地化,可以将用户界面的全部正文串和位图存放在资源文件里,然后将资源文件本地化,而不是将整个应用程序本地化。一个工程最多包含一个资源文件。

5. ActiveX 文档

ActiveX 文档(.dob)类似于窗体,但是在互联网资源管理器之类的互联网浏览器中是可以显示的。VB 的专业版和企业版能够创建 ActiveX 文档。

6. 用户控件和属性页模块

用户控件(.ctl)和属性页(.pag)模块也类似于窗体,但它们被用于创建 ActiveX 控件及与其关联的用来显示设计时属性的属性页。VB 的专业版和企业版能够创建 ActiveX 控件。

7. 部件

除文件和模块以外,还有几个其他类型的部件可以添加到工程中。

8. ActiveX 控件

ActiveX 控件(具有 .ocx 文件扩展名)是可选的控件,它可以被添加到工具箱中并在窗体里使用。当安装 VB 时,VB 包含的含有控件的那些文件被复制到一个公共目录(Windows 95 / 98 下面的子目录 \ Windows \ System)中。可以从多种源取得附加的 ActiveX 控件。也可以使用 VB 专业版和企业版创建自己的控件。

9. 可插入的对象

可插入的对象,例如 Microsoft Excel 的工作表对象,是可用作建造集成方案时建造块的部件。一个集成方案可以包含由不同的应用程序创建的不同格式的数据,例如工作表、位图和正文。

10. 引用

也可以添加能被应用程序使用的外部 ActiveX 部件的引用到工程中。通过访问【工程】菜单上的【引用】菜单项,使用如图 3.2 所示的【引用】对话框可指定引用。

11. ActiveX 设计器

ActiveX 设计器是类的设计工具,从类出发可以创建对象。窗体的设计界面是缺省的设计器,从其他的源可取得附加的设计器。

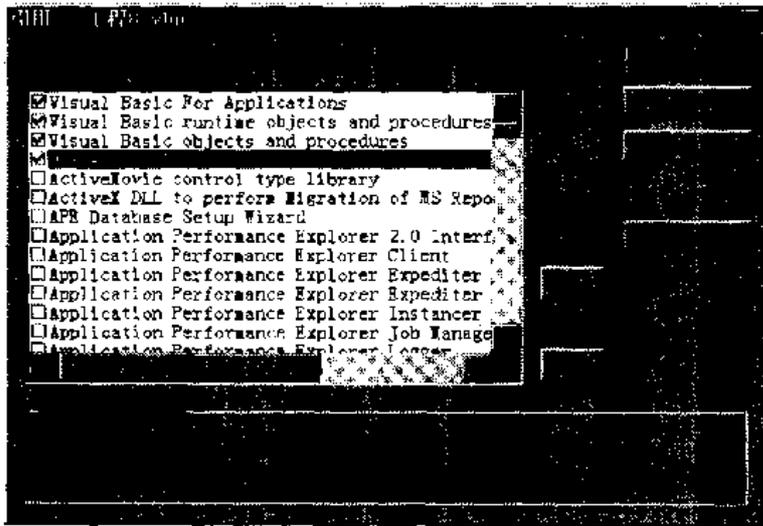


图 3.2 【引用】对话框

12. 标准控件

标准控件是由 VB 提供的,例如命令按钮或框架控件,它总是包含在工具箱里,与可从工具箱里删除、添加的 ActiveX 控件和可插入对象不同。

3.2 创建、打开和保存工程

用户要用 VB 编写应用程序,首先要创建工程,本节主要讲述如何创建、打开和保存工程。

【文件】菜单上的四个选项允许创建、打开和保存工程,它们分别是【新建工程】、【打开工程】、【保存工程】和【工程另存为】。

3.2.1 创建新的工程

要创建新的工程,可以有三个途径:

- 选择【文件】菜单中的【新建工程】,然后从【新建工程】对话框中选择需要的选项,再单击【确定】按钮,如图 3.3 所示。

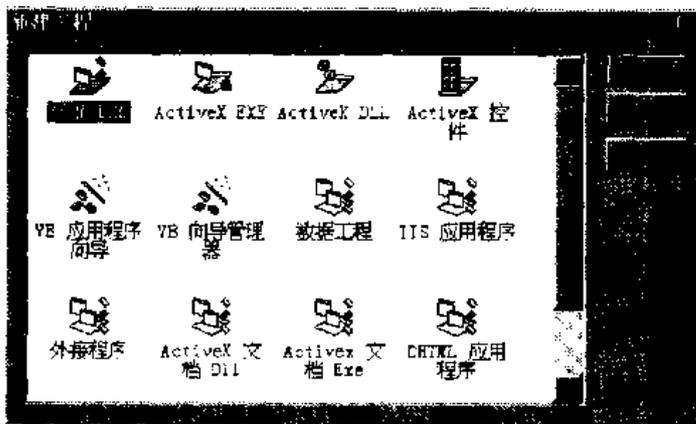


图 3.3 创建新的工程

- 单击【标准】工具栏的【新建工程】按钮。
- 单击【标准】工具栏的【新建工程】按钮右侧的下拉按钮,然后从菜单中选择合适的菜单

项,如图 3.4 所示。

3.2.2 保存和打开工程文件

选择【文件】菜单中的【保存工程组】项,或者单击【标准】工具栏中的【保存工程】按钮,可以保存当前的工程,集成环境会自动保存当前工程中的所有文件。

如果是第一次保存工程,或者选择了【文件】菜单中的【工程另存为】,集成环境会弹出如图 3.5 所示的【文件另存为】对话框,提示用户输入一个文件名来保存窗体或模块文件,并且提示用户保存所有修改过的窗口或模块。



图 3.4 【新建工程】下拉菜单

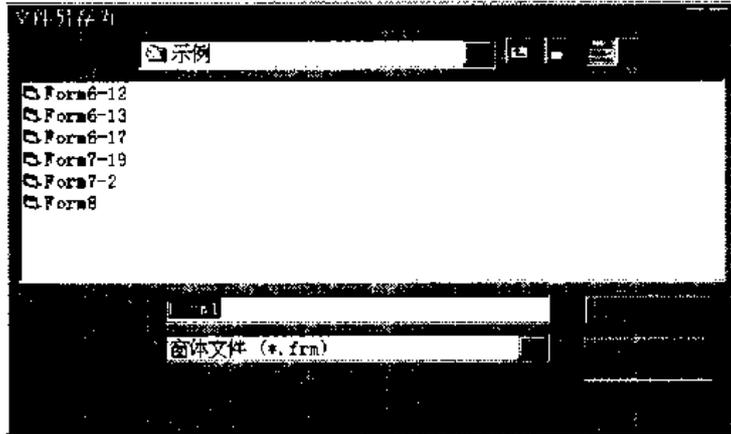


图 3.5 【文件另存为】对话框



图 3.6 【Source Code Control】对话框

接着系统打开【工程另存为】对话框,提示用户输入工程名。如果用户在安装 VB 6.0 的过程中,选择了自定义安装,并且在安装项目列表中选中了【Microsoft Visual SourceSafe 6.0】选项,那么用户单击该对话框中的确定按钮后,系统会弹出如图 3.6 所示的【Source Code Control】对话框,询问用户是否添加该工程到

SourceSafe 中。

提示:在工程间文件可以共享。像窗体这样的单个文件,可以是多个工程的组成部分,而且在一个工程中的窗体或模块所作的改变,将会传播到共享这个模块的所有工程。

要打开一个现存的工程文件,可以使用【文件】菜单中的【打开工程】,也可以单击【标准】工具栏的【打开工程】按钮。在打开一个工程之前,集成环境会关闭当前工程,并提示用户保存所有改动。

提示:在文件夹中双击一个现存工程的图标,或者拖动该文件并放入工程资源管理器窗口,也可以打开这个现存的工程文件。

3.3 添加、删除和保存文件

在进行应用程序开发时,不仅可以自行开发工程中的各个模块,还可以使用现成的模块文

件,方法是向工程中添加这些模块文件。

3.3.1 添加文件

用户如果要向工程中添加文件,可以按以下步骤进行:

1. 选择【工程】菜单中的【添加文件】选项,【添加文件】对话框(图 3.7)被显示。

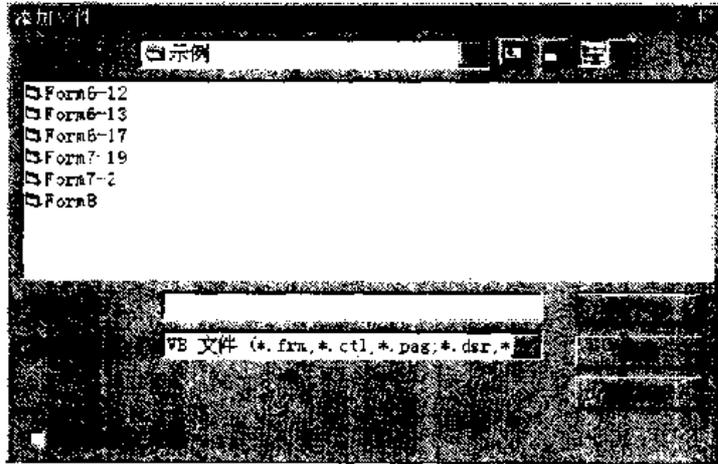


图 3.7 【添加文件】对话框

2. 选定一个现存的文件或一个新的文件类型,并单击【打开】按钮。

在工程中添加文件时,是简单地将该文件的引用纳入工程,而不是添加该文件的复制件。因此,如果更改文件并保存它,这个更改会影响包含此文件的任何工程。若想改变文件而不影响其他工程,应在【工程资源管理器】里选定该文件,然后从【文件】菜单选择【文件另存为】选项,以一个新的文件名保存此文件。

注意: 可以从 Windows 的【资源管理器】、【文件管理器】或【网上邻居】拖动文件并放入【工程】窗口,将它们添加到一个工程。

3.3.2 删除文件

用户如果要从工程中删除文件,可以按以下步骤进行:

1. 在【工程资源管理器】中选定该文件。
2. 从【工程】菜单中,选择【删除文件名】。

此文件将从工程中删除掉,但是仍存在于磁盘上。如果从工程中删除了文件,在保存此工程时 VB 更新此工程文件中的这个信息。但是,如果在 VB 之外删除一个文件,VB 不能更新此工程文件;因此,当打开此工程时,VB 将显示一个错误信息,警告一个文件丢失。

3.3.3 保存文件

如果用户只想保存文件而不保存工程,那么应执行以下操作:

1. 在【工程资源管理器】里选定此文件。
2. 从【文件】菜单选择【保存文件名】选项,在【文件另存为】对话框输入文件名。

另外,用户还可以将文本文件插入到代码中,在添加常数清单或添加可能保存在文本文件中的代码段时,这个功能很有用。方法如下:

1. 从【工程资源管理器】窗口中,选定要插入代码的窗体或模块。
2. 单击【查看代码】按钮,将光标移动到代码编辑器中要插入代码的地方。
3. 从【编辑】菜单,选取【插入文件】选项,显示如图 3.8 所示的【插入文件】对话框。
4. 在对话框中选择欲插入的文本文件名,然后单击【打开】按钮。

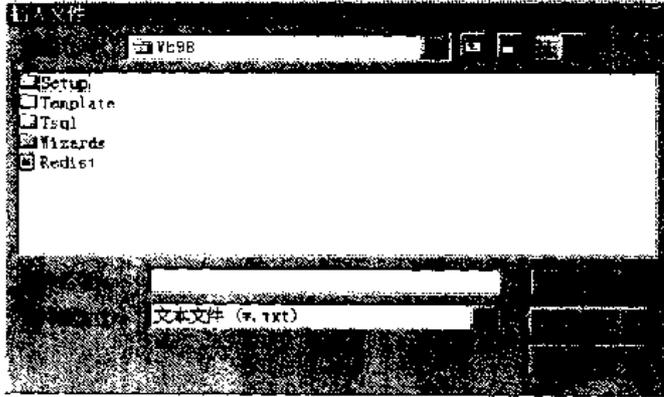


图 3.8 【插入文件】对话框

3.4 生成可执行文件

在此之前,我们运行程序都是使用工具栏的【启动】按钮或按下 F5 键,这样做只能看看程序执行的结果,而不能真正生成可执行文件(.EXE 文件)。如果要工程编译成可执行文件,可以按以下步骤进行:

1. 从【文件】菜单中选取【生成 projectname.exe】选项,这里 projectname 是工程的应用程序名,这时显示如图 3.9 所示的【生成工程】对话框。

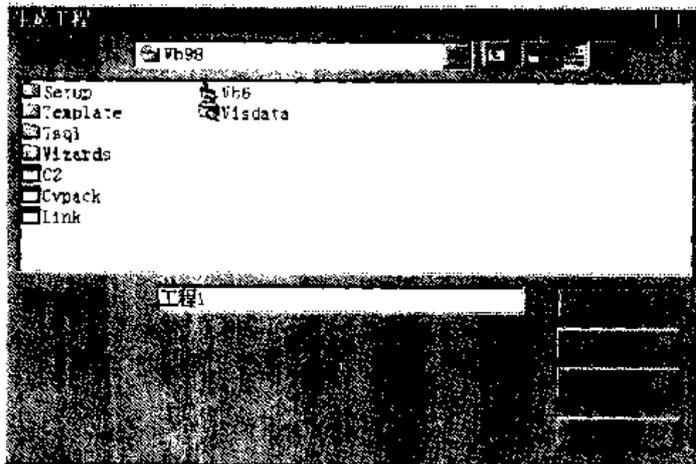


图 3.9 【生成工程】对话框

2. 在该对话框中,键入文件名或浏览有关目录,选定一个现有文件名。
3. 单击【选项】按钮,显示【工程属性】对话框,单击【生成】选项卡,如图 3.10 所示。可以在该对话框里,规定一些有关该可执行文件特定版本的详细资料。
4. 若要修改工程的版本号,则要设置合适的【主版本】、【次版本】和【修正】。选定【自动升级】复选框,那么每一次运行该工程的【生成 projectname.exe】命令时,【修正】都会自动增加。

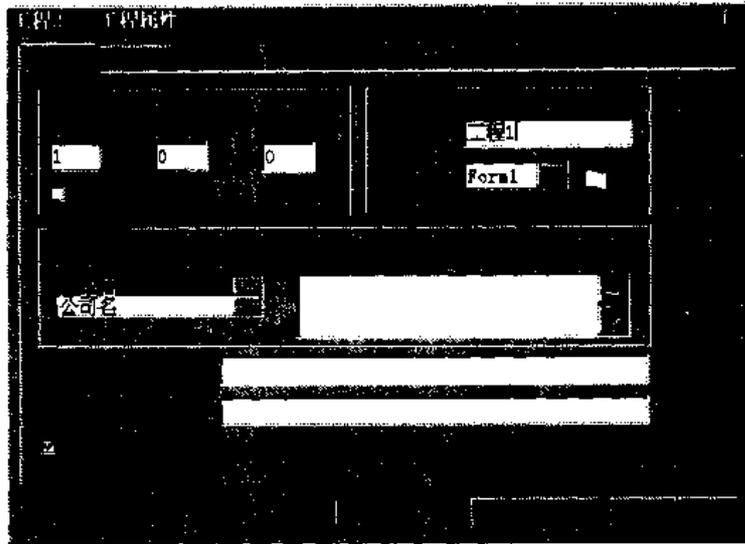


图 3.10 【工程属性】对话框的【生成】选项卡

5. 为了给应用程序指定新名,在【应用程序】域框中将新名键入【标题】文本框。如果要指定新图标,则从【图标】列表里选取一个。

6. 通过从列表框中选定主题并在文本框中输入信息,还可以输入【版本信息】框下的各种版本的版本专用注释(注释、公司名、商标和版权信息等等)。

7. 单击【确定】按钮,关闭【工程属性】对话框,再在【生成工程】对话框中单击【确定】按钮,编译和连接该可执行文件。

双击可执行文件的图标,像运行任何其他基于 Windows 的应用程序那样,可运行这个可执行文件。

提示: 在【工程属性】对话框中,单击【编译】选项卡,还可以设置一些编译的选项,例如用户可以选择生成 P- 代码或者本机代码,还可以对代码进行不同类型的优化。如图 3.11 所示。

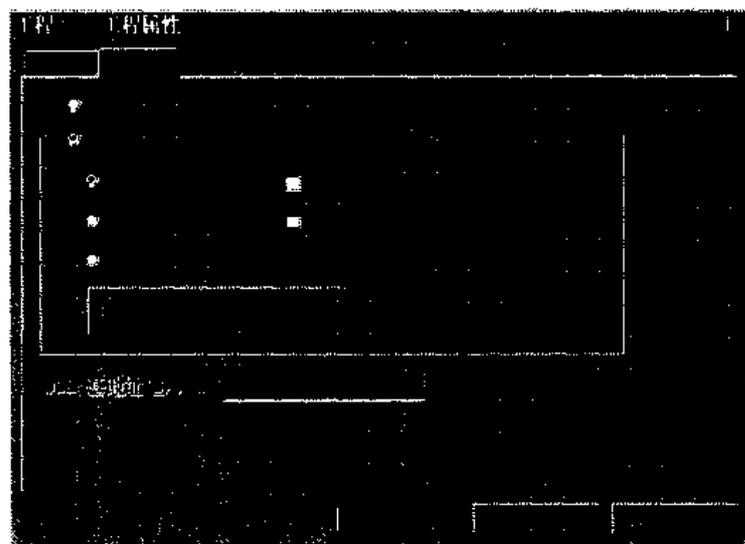


图 3.11 设置应用程序的编译选项

3.5 使用应用程序向导和外接程序

VB 6.0 允许选择和管理外接程序,这是对 VB 的扩充。这些扩充增强了 VB 开发环境的能力,例如,特殊的源代码控制能力。

3.5.1 外接程序管理器

Microsoft 和其他开发者创建了可以用于应用程序的外接程序,例如:向导就是一种外接程序,它可以简化某些任务(如创建窗体)。

使用外接程序管理器可以对工程添加或删除外接程序,从【外接程序】菜单中选择【外接程序管理器】选项,系统打开【外接程序管理器】对话框(如图 3.12 所示),其中列出可用的外接程序。

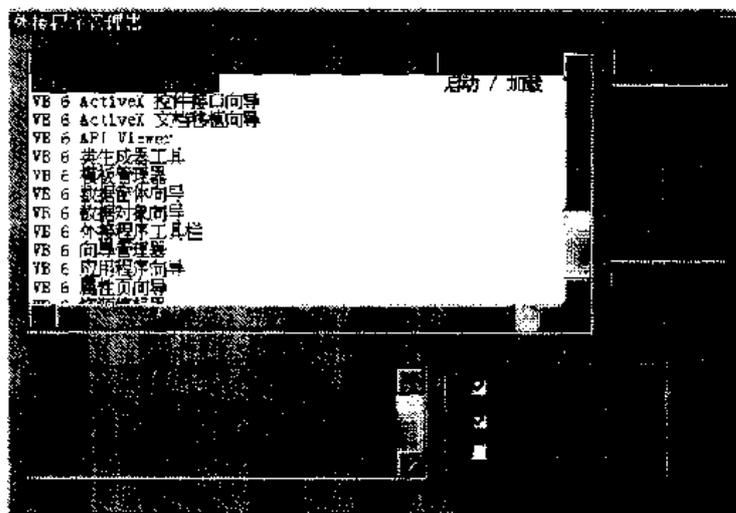


图 3.12 【外接程序管理器】对话框

注意:要使外接程序出现在【外接程序管理器】对话框中,外接程序的开发者必须保证它已正确安装。

如果用户要安装外接程序,可以按以下步骤进行:

1. 从【外接程序】菜单中,选择【外接程序管理器】选项。
2. 从可用外接程序列表中选定一个外接程序,然后在【加载行为】域中选定或清除各个复选框,指定该外接程序的加载行为。

3. 做完选定操作后,单击【确定】按钮。

这时,VB 连接被选定的外接程序,断开与被清除的外接程序的连接。

注意:选择一个外接程序后,会在 VB【外接程序】菜单中增加菜单项。

3.5.2 使用向导

应用程序向导通过提供具体任务的帮助的方法,使 VB 的使用更加容易。例如,VB 所包含的应用程序向导(如图 3.13 所示)通过提出一系列问题和选择的方法,为创建应用程序框架

的工作提供帮助。它根据用户的选择,生成窗体和窗体后面的代码,因此,用户需做的全部事情是为特有的功能添加代码。

VB 的专业版和企业版包含一些其他向导,其中有创建用于数据库的窗体的数据窗体向导,以及在互联网的应用程序里用于变换窗体的 ActiveX 文档向导。

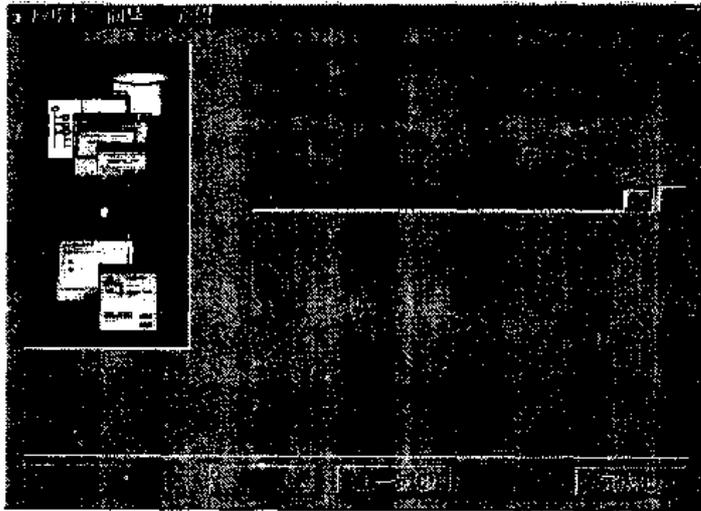


图 3.13 应用程序向导

使用外接程序管理器可安装或删除向导,一旦安装,它们将作为【外接程序】菜单上的选项出现。某些向导也能够以图标的形式出现在有关的对话框中。例如,可使用【新建工程】对话框中的【应用程序向导】图标访问【应用程序向导】。

提示: 用户可以用以下方法启动【应用程序向导】对话框:从【文件】菜单选取【新建工程】选项,再从【新建工程】对话框中选择【VB 应用程序向导】图标,最后单击【确定】按钮。

下面以用【应用程序向导】生成一个多文档界面程序为例,向用户介绍如何使用【应用程序向导】:

1. 从【文件】菜单中选择【新建工程】菜单项,从【新建工程】对话框中选择【VB 应用程序向导】图标,单击【确定】按钮。

2. 系统打开如图 3.13 所示的【应用程序向导—介绍】对话框,然后单击【下一步】按钮。

3. 这时系统打开如图 3.14 所示的【应用程序向导—界面类型】对话框,从中选择一种界面类型(单文档界面,多文档界面或资源管理器样式)。如:单击【多文档界面】单选框;并为应用程序起一个名字,如:“我的应用程序”。

4. 单击【下一步】按钮,系统打开如图 3.15 所示的【应用程序向导—菜单】对话框,为应用程序选择需要的顶层菜单和子菜单。如果列表中没有想要的菜单,可单击添加菜单按钮,在【添加新菜单】对话框中输入添加菜单的标题和名称。

5. 单击【下一步】按钮,系统打开【应用程序向导—自定义工具栏】对话框(如图 3.16 所示),为应用程序选择需要的工具按钮。添加工具按钮时,只需从左边列表框中的按钮选择一个,单击【右移所选按钮】,移到右边列表框即可。

6. 单击【完成】按钮,系统弹出一个消息框,提示用户已创建了应用程序;单击【确定】按钮,便完成了该应用程序的创建,最后整个 VB 界面显示如图 3.17 所示。

7. 单击【启动】按钮,运行程序,结果如图 3.18 所示。由此可见,用户只须用鼠标进行选

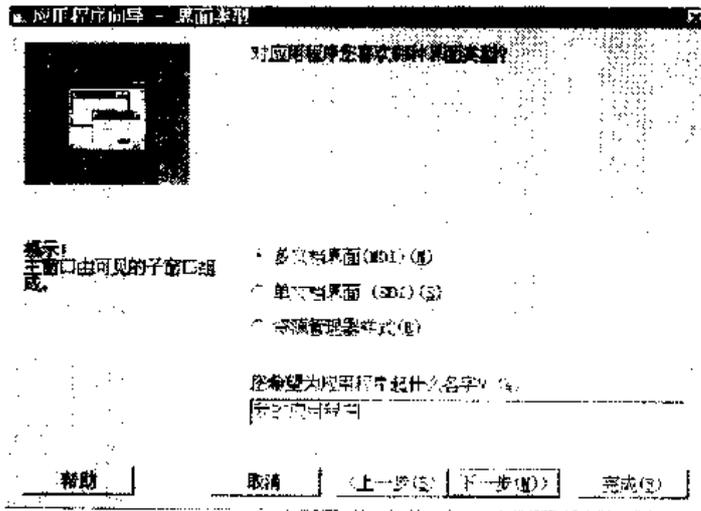


图 3.14 选择界面类型

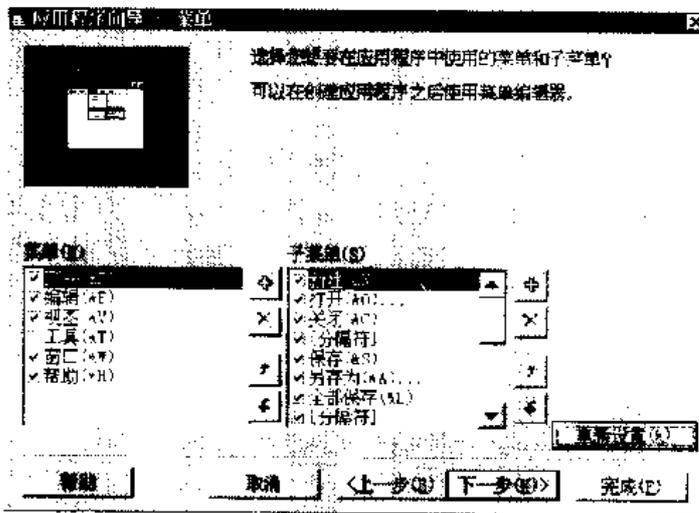


图 3.15 选择和添加菜单

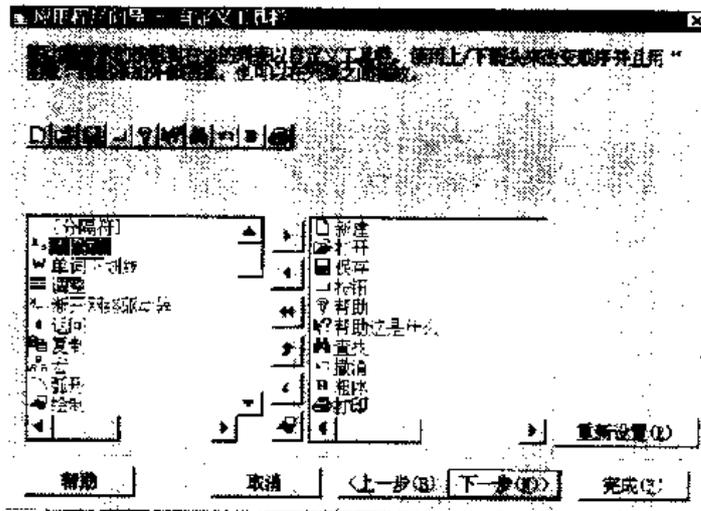


图 3.16 自定义工具栏

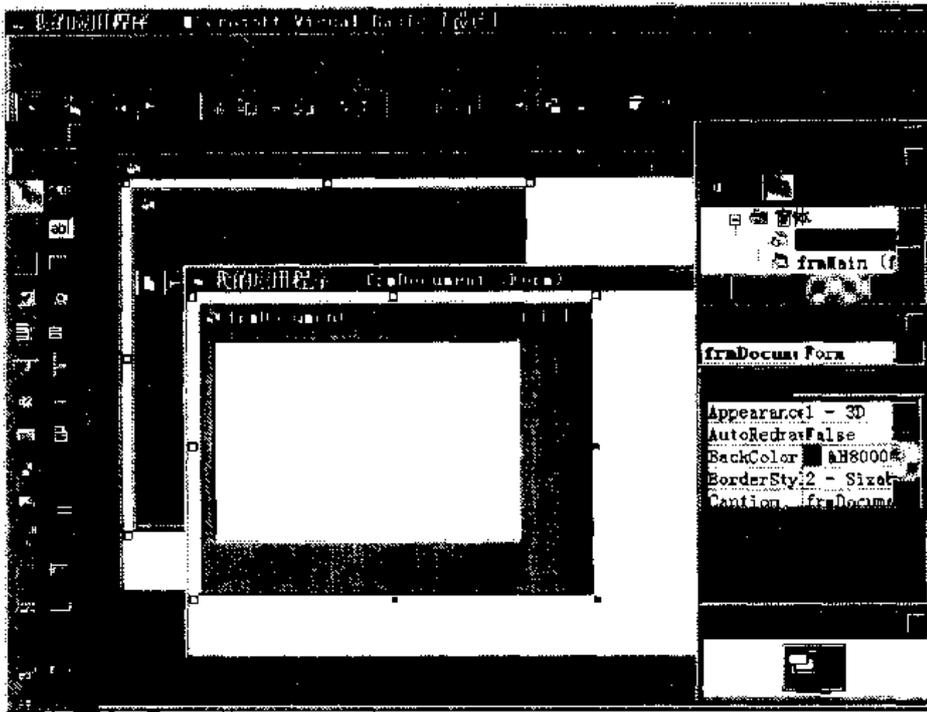


图 3.17 创建的应用程序

择和确定,就可创建美观的程序界面。但用这种方法创建的程序只是空壳,完成不了什么功能。要使该应用程序符合用户要求,还必须添加必要的界面元素和程序代码。随着以后的深入学习,读者会逐渐完善其功能。

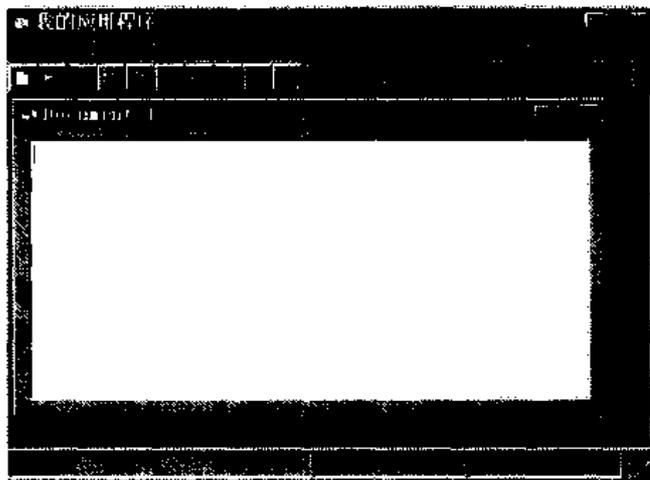


图 3.18 程序运行结果

3.6 本章小结

为了用 VB 创建应用程序,应当使用工程。一个工程是用来建造应用程序的文件的集合。本章叙述了如何创建和管理工程。学完本章,读者应了解以下内容:

1. 有关工程的基本概念;
2. 如何创建、打开和保存工程;

3. 如何向工程中添加、删除文件；
4. 如何只保存文件而不保存工程；
5. 使用应用程序向导和外接程序的基本方法。

第 4 章 VB 6.0 的编程语言

在这一章中,我们介绍的是 VB 6 源程序代码应了解的一些内容。VB 的窗体和控件提供程序的可视化界面,但是程序中大部分的实际工作是由程序代码来处理的。代码对用户和系统事件作出响应以执行任务。VB 6.0 提供强大的、使用相对容易的编程语言,这种编程语言以已经普遍使用多年的 Basic 语言为基础。但是,VB 语言已经得到扩展,用户可以通过判断和循环结构来轻松地控制程序;这种语言还得到了改编,用户能轻松地处理 VB 的对象和控件。

本章涉及 VB 中有关程序的书写规则、变量、流程控制、过程和函数等基本内容,如果用户曾经使用过其他的编程语言,那么会对本章内容相当熟悉。对于初次接触编程语言的读者,本章许多内容一时可能难以记住,这是非常正常的事情,随着学习的深入,读者会渐渐掌握这些内容的。

4.1 程序的书写规则

编写程序代码就像我们平时说话写字一样,都要依照一定的章法,否则,编写出的代码不能被计算机正确的识别,从而产生编译或者运算错误。

4.1.1 注释

在程序中使用注释是一个良好的习惯,用户可以使用注释来说明自己编写某段代码或声明某个变量的目的,以后读到这些注释就会想起当时的思路,既方便了开发者自己,也方便要阅读这些源代码的其他用户。

要添加注释,只要使用“'”符号作为注释文字的开关。“'”符号告诉 VB 忽略该符号后面本行的内容,这些内容是程序代码中的注释。VB 在编译程序时会自动跳过注释行。例如:

下面举出几个赋值语言和注释的示例

```
Form1.Width = 400           '将窗体的宽度设置为 400
Temp = Temp + 50           '变值 Temp 赋值为原值加 50
Form1.Caption = "Welcome!"  '将窗体的标题文字设置为 Welcome!
```

注释行一般写在同一行语句的后面,如果注释过长,一行写不下,可以另起一行,该行须以“'”开头。

提示: 在程序中的注释并不是越多越好,不必对每一条语句都进行注释,过多的注释会使程序显得十分罗嗦。用户只需对主要的过程和变量进行必要的说明即可。

4.1.2 断行

有些时候,一条 VB 语句会很长,如果将它写在一行,将给打印和阅读都带来不便。这种情况下,可使用续行符“_”(一个空格紧跟一条下划线),将长语句分成多行。例如:

```
StrTest = "白日依山尽," & _
```

“黄河入海流。”&_

“欲穷千里目,”&_

“更上一层楼。”

注意: 在同一行内,续行符后面不能加注释,续行符也不应将变量名和属性分隔在两行,续行符一般加在运算符的前后。

4.1.3 将多行语句写在一行上

VB 通常一行一条语句,而且在语句最后没有一个特殊的符号作为语句结束符(C 语言则以“;”符号作为语句结束符)。

如果想在一行中写下多条语句,则每条语句中间必须用冒号“:”作为分隔符号。例如:

```
Form1.Width = 400;Temp = Temp + 50;Form.Caption = "Welcome!"
```

当然,为了阅读方便,还是一行一条语句较好。

4.1.4 良好的编程习惯

良好的编程习惯,可以使应用程序的结构和编程风格标准化,以便于阅读和理解程序的代码。这种良好的编程习惯实际上就是一些编码约定,、可读性强且意义清楚,并且尽可能地直观。

1. 使用缩进

在编写程序代码时,经常要把一个控制结构放入另一个控制结构之内,在程序设计中叫做嵌套。编写程序代码的对齐方式与日常写文章是有些不同的,倘若总是把代码从最左一列开始写,很难看清嵌套关系。例如:

```
Private Sub Form_Click()  
    Dim QFont, MFont  
    For Each QFont In Screen.Fonts()  
        For Each MFont In Printer.Fonts()  
            If QFont = MFont Then  
                Print QFont  
            End If  
        Next QFont  
    Next MFont  
End Sub
```

上面一段程序中代码的嵌套关系很不直观。所以,在编写代码时,习惯上对过程、判断语句、循环结构的正文部分进行缩进,使程序代码的可读性大为改善。例如使用缩进格式重写上段代码:

```
Private Sub Form_Click()  
    Dim QFont, MFont  
    For Each QFont In Screen.Fonts()  
        For Each MFont In Printer.Fonts()  
            If QFont = MFont Then
```

```

Print QFont
End If
Next MFont
Next QFont
End Sub

```

使用缩进后,代码的嵌套关系和逻辑结构都变得非常清晰。

2. 添加注释

用户最怕接着别人的程序工作,因为要理解别人的思路和并强迫自己沿着这种路子走下去,实在有些勉为其难,如果能够在程序中适当添加一些注释,效果就会好得多。

3. 截断长代码行

当一行代码过长时,应使用下划线连接字符“_”将代码截短为多行代码,这样便于阅读打印和调试字符串。尤其是需要用一个比较长的字符串来显示一个消息框(MsgBox),或输入框(InputBox),或产生一个 SQL 字符串时,这一技术特别有用。

4. 变量命名

在程序中,常量和变量应该用一致的前缀来命名,这样很容易识别它们的数据类型。给变量加前缀可以指明它们的数据类型,例如:StrClserName 为字符型变量,dblTotal 为双精度型变量等等。而且变量或过程名的主体应该使用大小写混合形式,并且应该足够长以描述它的作用,如 SaveFile 或 RasterData。

4.2 变量和常量

在 VB 环境下进行计算时,常常需要临时存储数据,这些数据在开始是未知的,这就要将它们存储到变量中。

在程序处理数据时,用户把信息暂时存储在计算机的内存里。要存储信息,用户必须指定存储信息的单位,以便获取信息,这就是变量的功能。在所有的编程语言中,变量都为内存中的某个特定的位置命名,一旦定义了某个变量,该变量表示的都将是同一个内存位置,直到释放该变量。

4.2.1 变量的命名规则

为了区别存储着不同数据的变量,需要对变量命名。在命名变量时,用户拥有极大的灵活性。变量名可以简单,也可以描述字所包含的信息。例如用户可以把一个计数器变量简单地命名为 C,也可以用一个更具描述性的变量名,如 NumberOfRecord。在 VB 中,变量的命名要遵循下面的规则:



图 4.1 变量名错误时
VB 显示的消息框

1. 变量名必须以字母开头,例如 hax, count, lastone, x78 等变量名是合法的,而 2ab, \$ boot 等变量名是非法的。
2. 不能在变量名中出现句点“.”、空格或者嵌入下列字符:!, #, @, \$, %, &; 例如 ret, ret! ret%ret\$ 等变量名是合法的,而 ret., ret, r%et 等变量名就是不合法的,如果程序中出现了这种错误的变量名,VB 会显示如图 4.1 所示的消息框报告错误。

3. 变量名不能和关键字同名,关键字是 VB 使用的词,是语言的组成部分。其中包括预定

义语句(If Loop 等)、函数(Len、Abs 等)和操作符(Or、Mod 等)。

4. 变量名在有效的范围内必须是唯一的。有效范围就是引用变量可以被程序识别、使用的作用范围——一个过程，一个窗体等等。有关引用变量作用范围的内容，将在 4.2.4 节“变量的作用范围”中介绍。

5. 变量名的长度不得超过 255 个字符。

4.2.2 变量的声明

在使用变量前，最好先声明这个变量，也就是事先将变量的有关信息通知程序。声明变量要使用 Dim 语句，Dim 语句的格式为：

```
Dim 变量名【As 类型】
```

例如：在 Form_Load 事件过程中声明一个变量 Count，并将其赋值为 10：

```
Sub Form_Load()  
Dim Count  
Count = 10  
End sub
```

Dim 语句中用方括号括起来的“As 类型”子句表示是可选的，使用这一子句可以定义变量的数据类型或对象类型。数据类型定义了变量所存储信息的类型，可以是 String(字符串)、Integer(整型)、Currency(货币型)等。变量也可以是 VB 或其他应用程序的对象类型，如 Object(对象)、Form(窗体)和 TextBox(文本框)等。例如，我们可以将变量 Count 声明为整数类型：

```
Sub Form_Load()  
Dim Count As Integer  
Count = 10  
End Sub
```

注意：在过程内部用 Dim 语句声明的变量，只有在该过程执行时才存在，过程一结束，该变量的值也就消失了。此外，变量对过程而言是局部的，一个过程中不能访问另一个过程中的变量，所以在不同过程中可以使用相同的变量名。

1. 模块级变量

VB 应用程序是基于对象的，对象包含数据和代码。如果在同一窗体中的所有过程都分享同一个变量，就要把它声明为模块级变量。方法是首先单击代码窗口的对象列表框，从中选择【(通用)】选项(如图 4.2 所示)，进入窗体模块的声明段，然后在声明段中定义该变量。

在窗体模块的声明段中声明变量，既可使用 Dim 关键字，也可使用 Public 和 Private 关键字。用 Public 关键字声明模块级变量将使变量在整个应用程序中有效，不同模块中的过程都可以使用这个变量，因此使用 Public 关键字声明的变量又叫公用变量。例如可在声明段中这样声明变量：

```
Public Count As Integer
```

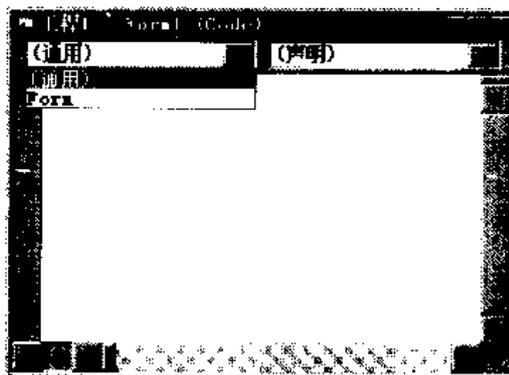


图 4.2 进入窗体模块的声明段

而用 Private 关键字声明变量,则本窗体模块中的过程可以访问它,而其他模块的代码不能访问该变量。例如,可在声明段中声明变量:

```
Private Count As Integer
```

注意:在过程中不能使用 Public 或者 Private 关键字声明变量,二者只能在模块的声明段中使用。

模块级变量在同一窗体的不同过程中都可以被访问,与此相对的是:用 Dim 关键字在过程中声明的变量只能在过程执行期间有效,其他过程中都不能使用,称为局部变量。例如在 Form_Load 事件过程中定义局部变量 Count,若在 Form_Click 事件过程中使用这个变量,VB 在程序运行时,会显示如图 4.3 所示的错误信息。

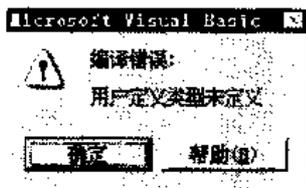


图 4.3 局部变量只能
在本窗体模块中使用

```
Private Sub Form_Load()  
Dim Count As Integer  
Count = 10  
End Sub  
Private Sub Form_Click()  
Count = 20  
End Sub
```

2. 静态变量

使用 Dim 关键字声明的局部变量,在过程执行结束后变量的值不能保留,在每一次过程重新执行时,变量值将被清零。如果用户希望在离开过程之后,还能保持过程中局部变量的值,就应该使用 Static 关键字将这个变量声明为静态变量。用 Static 关键字在过程中声明的局部变量,即使过程结束,变量的值仍然保留着。例如我们为一个窗体编写下面两种不同的程序:

程序一:

```
Private Sub Form_Click()  
Static Count As Integer  
Count = Count + 1  
Print Count  
End Sub
```

程序二:

```
Private Sub Form_Click()  
Dim Count As Integer  
Count = Count + 1  
Print Count  
End Sub
```

程序一的执行结果如图 4.4 所示,用户在窗体上每单击一次,显示的数字加 1。而程序二的运行结果如图 4.5 所示,用户在窗体上单击一次,显示的数字始终为 1。

注意:在模块的声明段中不能使用 Static 关键字,该关键字只能在过程中使用。

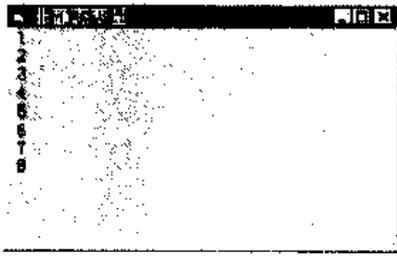


图 4.4 静态变量在过程结束后保留原值

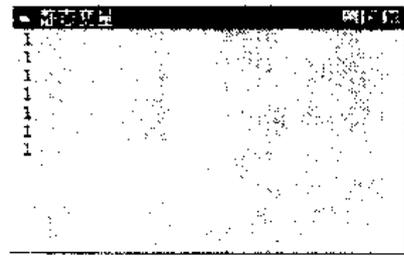


图 4.5 非静态变量在过程结束后不保留原值

4.2.3 变量的数据类型

在 VB 中,数据类型决定了如何将变量存储到计算机的内存中,所有的变量都具有数据类型,数据类型决定了变量能够存储哪种数据,不同类型的数据,要使用相应的类型。

1. 数字类型

VB 中数字型的数据类型特别多,它支持 6 种数字型的数据类型——Integer(整型)、Long(长整型)、Single(单精度浮点型)、Double(双精度浮点型)、Currency(货币型)和 Byte(字节型)。

如果用户事先已经知道变量要存放整数而不是带小数点的数字,就应当将它声明为 Integer 类型或 Long 类型,因为整数运算速度快,而且比其他数据类型占用的内存要少。如果变量包含小数,则可将它声明为 Single、Double 或 Currency 类型。Currency 数据类型支持小数点右面 4 位和小数点左面 15 位的定点数据。

如果在程序中要使用二进制数值,应使用 Byte 数据类型,但使用 Byte 类型不能表示负数为便于比较和理解,表 4.1 给出了几种数字型的数据类型说明。

表 4.1 VB 中数字型的数据类型

数据类型	存储位数	说 明
Byte	8 位(1 字节)	变量存储单精度型,无符号整型,8 位的数据,范围在 0~255 之间。Byte 数据类型在存储二进制数据时很有用
Currency	64 位(8 字节)	变量为整型的数值形式然后除以 10000 给出一个定点数,Currency 数据类型在货币计算与定点计算中很有用,是一个精确的定点数据类型
Double	64 位(8 字节)	变量存储为 64 位浮点数值形式,正数范围从 4.941E-324~1.798E308;负数范围从 -1.798E308~-4.941E-324
Integer	16 位(2 字节)	变量存储为 16 位的数值形式,其范围为 -32768~32767
Long	32 位(4 字节)	变量存储为 32 位有符号的数值形式,其范围为 -2147483648~2147483647
Single	32 位(4 字节)	变量存储为 32 位浮点数值的形式,正数范围是 1.401298E-45~3.402823E38,负数范围是 -3.402823E38~-1.401298E-45

2. Boolean 类型

Boolean(布尔)类型的变量主要用来进行逻辑判断,它的存储位数是 16 位,只能取两个值中的一个:True(真)或是 False(假)。例如:

```
Dim xBln As Boolean
```

```
xBln = True
```

```
xBin = False
```

提示：当把其他的数据类型转换为 Boolean 值时，0 会转成 False，其他值会变成 True；当把 Boolean 值转换为其他的数据类型时，False 变成 0，True 变成 1。

3. String 类型

String 类型变量的字符码范围是 0 ~ 255，字符集的前 128 个字符(0 ~ 127)对应于标准键盘上的字符与符号；而后 128 个字符(128 ~ 255)则代表了一些特殊字符。例如货币符号、重音符号、国际字符及分数。使用 String 类型可以声明两种字符串——变长与定长的字符串。

按照缺省规定，String 变量是一个可变长度的字符串，随着对字符串变量赋予新数据，它的长度可增可减。如果变量总是包含字符串而较少包含数值，就可将其声明为 String 类型，例如：

```
Private strTemp As String
```

然后可将字符串赋给这个变量。

```
strTemp = "一江春水向东流。"
```

如果要声明字符串具有固定长度，可用以下格式：

```
String * 字符串长度
```

例如，为了声明一个长度为 20 字符的字符串，可用下列语句：

```
Dim strTemp As String * 20
```

注意：这时如果赋给字符串 strTemp 的字符少于 20 个，则用空格将 strTemp 的不足部分填满，如果赋给字符串的长度太长，已超过 100 个字符，那么 VB 会直接截去超过部分的字符。

在 VB 中，数字和包含数字的字符串变量可以方便的互换类型。如果字符串表示数值，则可将字符串赋给数值变量，同时也可将数值赋给字符串变量，VB 会自动强制转换变量为适当的数据类型。例如在下面的程序中，就可以随意地将数字和字符串混合使用：

```
Dim intA As Integer
```

```
Dim strB As String
```

```
strB = 3212 '将数值赋值给字符串变量
```

```
intA = strB '将字符串传递给数值变量
```

```
'将字符串变量传递给正弦函数，再将正弦值赋给字符串变量
```

```
strB = Sin(strB)
```



图 4.6 转换字符串和数值时，
要求字符串中的值是数值

提示：如果传送字符串中的值不是数值，则在运行时出错，VB 弹出如图 4.6 所示的消息框显示错误信息。

4. Date 类型

Date 类型的变量用来保存日期，变量存储为 64 位浮点数值形式，可以表示的日期范围从公元 100 年 1 月 1 日到公元 9999 年 12 月 31 日，而时间可以从 0:00:00 到 23:59:59。用户可以使用下面示例的方式将日期赋给 Date 类型的变量，日期文字必须使用符号 # 括起来，否则，VB 不能

正确识别日期,例如:

```
Dim dateTemp As Date
dateTemp = # 12/02/74 #
dateTemp = # 1974 - 12 - 02 12:30:00PM #
dateTemp = # 74, 12, 02 #
dateTemp = # December 02, 1974 #
dateTemp = # 02 Dec 74 #
```

注意:不能这样赋值: dateTemp = # 1974 年 12 月 2 日 #, 因为 VB 不能正确识别这样包含汉字的日期文字。

当其他数值类型的变量转换为 Date 类型的变量时, VB 将以为小数点左边的值表示日期信息,而小数点右边的值则表示时间。VB 默认午夜为 0 而中午为 0.5, 而且使用负数表示 1899 年 12 月 30 日之前的日期。

例如下面的程序运行之后,用户在窗体上单击即可显示出一个由数字转化来的日期,结果如图 4.7 所示。

```
Private Sub Form_Click()
Dim dateTemp As Date
Dim dblTemp As Double
dblTemp = 27365.5
dateTemp = dblTemp
Print dateTemp
End Sub
```



图 4.7 窗体显示由数字转化来的日期

5. Object 类型

Object 变量存储为 32 位的数值形式,作为对象的引用。利用 Set 语句,声明为 Object 类型的变量可以赋值为任何对象的引用。

需要注意的是给 Object 类型的变量赋值必须用 Set 语句,用户很容易犯对一个 Object 类型的变量直接赋值的错误,例如:

```
Private Sub Form_Click()
Dim Temp As Object
Temp = Form1
End Sub
```



图 4.8 直接对 Object 类型的变量赋值而显示的错误信息

上面的例子中,直接将窗体对象 Form1 赋值给了 Object 类型的变量 Temp,这时 VB 会弹出如图 4.8 所示的信息框提示错误信息。

上例正确的程序代码应该是:

```
Private Sub Form_Click()
Dim Temp As Object
Set Temp = Form1
End Sub
```

6. Variant 类型

如果用户声明变量采用如下形式:

```
Static Temp
```

```
Dim Temp
```

此时,由于没有显式指明变量类型,VB 会缺省认为这个变量类型为 Variant 类型。

Variant 是一种特殊的数值类型,除了定长 String 数据和用户自定义类型外,它可以代表其他任何种类的数据。如果把它们赋值给 Variant 变量,则不必在这些数据的类型间进行转换;VB 会自动完成任何必要的类型转换。

当 Variant 类型的变量包含数值数据时,可以是任何整型或实型数,通常 Variant 变量会保持原来的数据类型。例如,如果把一个 Integer 类型的值赋值给 Variant 类型的变量,那么接下来的运算会把此 Variant 变量当作 Integer 来处理。但是,如果算术运算针对包含 Byte、Integer、Long 或 Single 的 Variant 变量,而且运算结果超过原来数据类型的正常范围时,则在 Variant 变量中的结果会提升到数值范围较大的数据类型,如 Byte 提升到 Integer, Integer 提升到 Long, Long 和 Single 会提升到 Double。而当 Variant 变量中的 Currency、Double 类型的值超过它们各自的范围时,就会发生错误。

除了可以包含各种类型的数据外,Variant 类型的变量还可以是 Empty、Error 及 Null 等特殊值。

(1) Null 值

Null 通常用于数据库应用程序,表示未知数据或丢失的数据,Null 具有以下几个特性:

- 对包含 Null 的表达式,计算结果总是 Null。可以说 Null 通过表达式传播。
- 将 Null、含 Null 的 Variant 变量或计算结果为 Null 的表达式作为参数传递给大多数的函数时,将会使函数返回 Null 值。如果要测试 Variant 变量是否包含 Null 值,可以使用 IsNull 函数。

例如:

```
If IsNull(x) And IsNull(y) Then  
Z = Null  
Else  
Z = 0  
End If
```

注意: 如果将 Null 值赋予 Variant 的任何其他类型变量,则将出错,而将 Null 值赋给 Variant 类型的变量则不会发生错误

(2) Empty 值

Empty 值用来标记尚未初始化的 Variant 变量。在赋值以前,Variant 变量具有值 Empty,值 Empty 不是 0,不是零长度字符串,也不是 Null 值。如果要测试 Variant 变量是否包含 Empty 值,可以使用 IsEmpty 函数。例如:

```
If IsEmpty(x) Then  
x = 0  
Else  
x = Empty
```

End If

只要将任何其他值(如 Null,0 等)赋值给 Variant 变量,Empty 值就会消失,除非将关键字 Empty 赋值给 Variant 变量:

提示: Empty 和 Null 不同:Empty 表示变量还未赋值,而 Null 表示 Variant 变量确实包含一个无效数据。

(3) Error 值

在 Variant 数据类型中,Error 是用来指示在过程中出现错误时的特殊值,用户或应用程序可以根据此错误值采取另外的行动,但程序并不产生普通的应用程序级的错误处理。

7. 数据类型转换

在进行程序设计时,有时需要进行变量的类型转换。VB 提供了几种类型转换函数,用户可以使用这些函数来将数值转换成特定数据类型。例如,可以用 CCur 函数将值转换成 Currency 类型:

```
Paper = CCur(time * piece)
```

在表 4.2 中,列出了一些常用的类型转换函数及其目标类型。在进行类型转换时,对目标数据类型,传递到转换函数的值必须是有效的,否则会发生错误。例如,若要把 Long 型转换成 Integer 型,那么,Long 型须在 Integer 数据类型的范围之内,即在 -32678 ~ 32767 之间。

表 4.2 VB 的类型转换函数

转换函数	目标类型
CByte	Byte
CCur	Currency
CBool	Boolean
CDbl	Double
Cdate	Date
Cint	Integer
CLng	Long
CSng	Single
CStr	String
Cvar	Variant
CVerr	Error

4.2.4 变量的作用范围

变量的作用范围确定了能够识别并使用变量的那部分代码。在一个过程内部声明变量时,只有过程内部的代码才能访问或修改那个变量的值。如果要使它对于同一模块内的所有过程都有效,或者对于整个应用程序的所有过程都有效,那么就需要声明模块级的变量。表 4.3 列出了变量作用范围的 3 个层次和它们的声明方式。

表 4.3 不同作用范围变量的声明方式

作用范围	局部变量	模块级变量	公用变量
声明方式	Dim, Static	Dim, Private	Public
变量的声明位置	过程内部	模块的声明段	模块的声明段
本模块中其他过程能否访问	不能	能	能
其他模块能否访问	不能	不能	能

当变量的作用范围不同时,变量的名字可以相同,在不同的过程可以使用相同名字的这些变量的类型可以相同,也可以不同。另外变量的作用范围还可以出现交叉,例如:可以定义名为 Temp 的公用变量,然后在过程中声明 Temp 为局部变量。这时,在过程内通过引用 Temp 来访问局部变量,而在过程外可以通过引用 Temp 来访问公用变量。请看下面的示例:

```
Public Temp As Double
Sub Form_Click()
Dim Temp As Integer
Temp = 12
End Sub
```

当变量名称相同而范围不同时,程序总优先访问局限性大的变量。而且,局部变量的局限性最大,其次为模块级变量,而公用变量的局限性最小。

注意:在相同的作用范围中不能定义相同的变量名,例如在下面的例子中,同一过程内重复定义了两个变量 x:

```
Sub Form_Load()
Dim x As Integer
Dim x As Currency
End Sub
```

那么在运行该程序时,VB 会弹出消息框显示错误信息。

4.2.5 变量的显式声明和隐式声明

在 VB 中,使用一个变量之前并不必须先声明这个变量。例如在下面的程序中,使用变量 Testday 和 Testweek 之前并没有声明它:

```
Sub Form_Click()
Testday = Now()
Testweek = WeekDay(Testday)
End Sub
```

这时,VB 用这个名字自动创建一个变量,使用这个变量时,可以认为它就是隐式声明的。

不过,使用这种隐式声明的方法时,如果把变量名拼错了,VB 遇到新的变量名时,将认为是用户又隐式声明了一个新变量,例如:

```
Sub Form_Click()
Testday = Now
Testweek = WeekDay(Tesday)
End Sub
```

这段代码的第三行与上段代码的第三行仅一个字母之差,也许是用户不小心把 Testday 错拼成 Tesday,将会导致运行结果不对。



图 4.9 变量未定义错误信息

因此,在使用变量前,最好先声明变量,同时为了确保在使用变量前已经进行了声明,只须在类模块、窗体模块或标准模块的声明段中加入这条语句:

```
Option Explicit
```

在添加 Option Explicit 语句后,VB 将自动检查程序中是否有未定义的变量,发现后将显示如图 4.9 所示的错误信息。

另外,用户还可以单击【工具】菜单的【选项】菜单项,系统弹出如图 4.10 所示的【选项】对

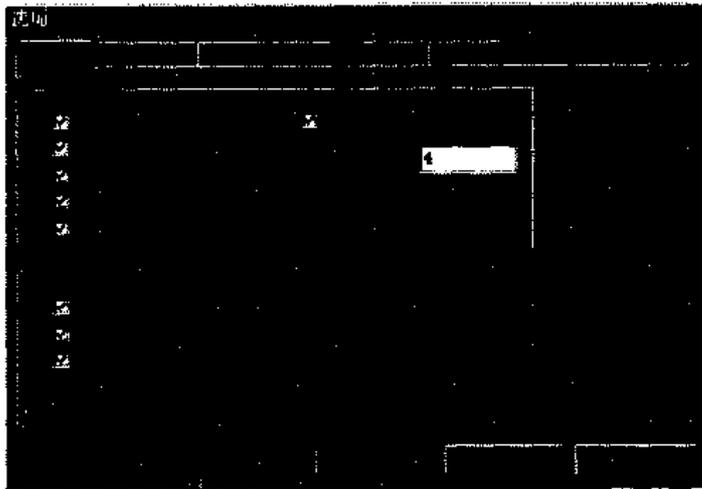


图 4.10 【选项】对话框

话框,单击该对话框中的【编辑器】选项卡,再选中【要求变量声明】复选框。这样 VB 就会在任何新模块中自动插入 Option Explicit 语句(如图 4.11 所示),但不会在已经建立的模块中自动插入,所以对于已经建立的模块,只能手工添加 Option Explicit 语句。

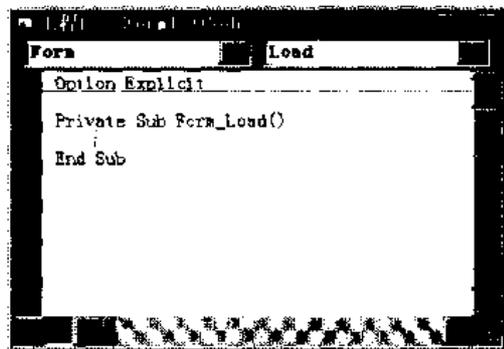


图 4.11 VB 自动添加 Option Explicit 语句

4.2.6 常量

变量只是在计算机的内存中存储信息的一种方法,另一种方法是使用常量。用户一旦定义了常量,在以后的程序中不能用赋值语句修改它们,否则,在运行程序时,VB 将生成一个错误。定义常量可以改进代码的可读性和可维护性,它通常是有意义的名字,用以取代程序运行中保持不变的数值或字符串。

VB 为不同的活动提供了多个常量集合,有颜色定义常量,数据访问常量,形状常量等等。如果用户想知道某个常量的值,可以单击【视图】菜单中的【对象浏览器】项或单击【对象浏览器】工具按钮,系统弹出如图 4.12 所示的对话框,可使用该对话框中的列表来找到所需的常量,选中常量后,对话框底端的文本区域将显示常量的值和功能。

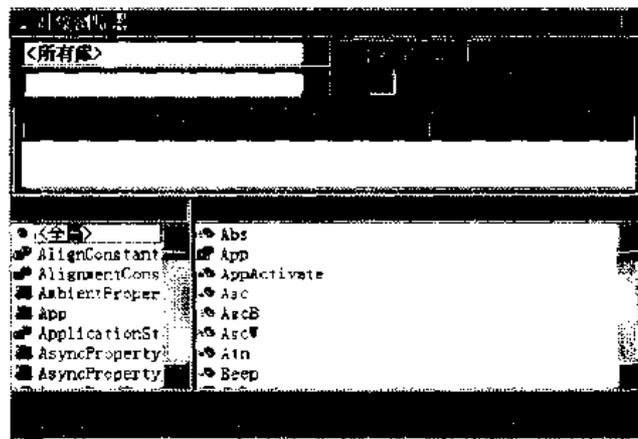


图 4.12 【对象浏览器】对话框

提示：可按 F2 键来打开【对象浏览器】对话框。

尽管 VB 为许多活动定义了大量的常量，但是有时候用户还要建立自定义常量，在 VB6 中声明常量的语法是这样的：

【Public/Private】Const 常量名【As 类型】= 表达式

参数【常量名】是有效的符号名，参数【表达式】由数值常数或字符串常数以及运算符组成。Const 语句可以表示数量、字符串或日期时间，例如：

```
Const herBirthday = # 12/2/94 #  
Public Const Pi = 3.1415926536  
Public Const TotalCount As Integer = 1000
```

提示：和变量声明一样，Const 语句也有作用范围，其规则与变量相同。如果要创建在整个应用程序中有效的常量，须在标准模块的声明段中声明，并在 Const 前放置 Public。在窗体模块或类模块中不能声明 Public 常量。

4.3 编写简单语句

前面讲了变量和常量的基本概念，了解了变量和常量能够存储信息以及如何声明变量和常量后，就要给常量和变量赋值以及进行常量和变量之间的运算。

4.3.1 赋值语句

VB 6 的程序代码由语句、常量和声明部分组成，赋值语句是最常用的语句。使用赋值语句可以在程序运行中改变对象的属性和变量的值，在前面的例子中已多次出现这样的语句：

```
Form1.Width = 400  
Temp = Temp + 50  
Form1.Caption = "Welcome!"
```

等等。赋值语句的语法是：

对象属性或者变量 = 表达式

其含义就是将等号右边表达式的值传送给等号左边的变量或对象属性。

4.3.2 使用各种运算

在进行程序设计时，要经常进行各种运算，如算术运算、逻辑运算、比较运算等。

1. 算术运算

算术运算是指通常的加减乘除以及指数这样的数学运算，在 VB 6 中，算术运算包括：加法（+）、减法（-）、乘法（*）、浮点数除法（/）、整数除法（\）、指数（^）、求余（Mod）。例如：

```
Temp = 23 + 74  
Temp = 23 / 74  
Temp = 2 ^ 3 'Temp = 2 的 3 次方 = 8  
Temp = 74 MOD 23 '结果为 5
```

2. 逻辑运算

逻辑运算可以表示比较复杂的逻辑关系,运算结果要么为 True,要么为 False。表 4.4 列出了 VB 中所有的运算符和它们表示的逻辑关系,在表中,True 用“1”代表,False 用“0”表示。

表 4.4 逻辑运算符和它们表示的逻辑关系

条件 A	条件 B	NOT A	A OR B	A AND B	A XOR B	A EQV B	A IMP B
0	0	1	0	0	0	1	1
0	1	1	1	0	1	0	1
1	0	0	1	0	1	0	0
1	1	0	1	1	0	1	1

其中,A IMP B 表示的逻辑关系为当 A 为 True,B 为 False 时结果才是 False,否则为 True。

3. 比较运算

比较运算就是比较大小,比较运算结果可以是 True 或 False。如果比较双方有一个为 Null,结果还为 Null。VB 6 中的比较运算有大于(>)、小于(<)、大于或等于(>=)、小于或等于(<=)、等于(=)、不等于(<>)。

4. 运算优先顺序

在一个表达式中进行若干运算时,每一部分都按预先确定的顺序进行计算求解,称为运算符的优先顺序。

在表达式中,当运算符不只一种时,要先进行算术运算,接着进行比较运算,然后再进行逻辑运算。若比较运算符的优先顺序相同,则按它们出现的顺序从左到右进行处理。算术运算符的优先顺序从高到低依次为:^、-(负数)、* 和 /、\、Mod、+ 和 -;逻辑运算符的优先顺序从高到低依次为:Not、And、Or、Xor、EQV、Imp。

提示:用户可以用括号改变优先顺序,强制括号内的表达式优先运算,因此使用括号控制运算的优先次序是一个良好的习惯,可以保证程序不易出错且清晰易懂。

4.4 流程控制语句

在 VB 中,控制程序执行的流程控制语句共有两种,即条件判断语句和循环语句。

4.4.1 条件判断语句

在要作出判断的许多情况下,有时希望只有在条件为真时才执行一条或多条语句。这时要用条件判断语句。VB 的条件判断结构可有以下几种:

1. If...Then 结构

使用 If...Then 结构,可有条件的执行某些语句。它的语法形式是:

If 条件 Then 语句

例如:

```
If YourHigh < 1.65 Then Print "你身高不够。"
```

这种语法形式只选择执行一条语句。如果要选择执行多条语句,则使用这样的语法:

```
If 条件 Then
语句 1
语句 2
...
End If
```

例如:

```
If YourHigh < 1.65 Then
Print“你的身高不够。”
Print“所以你不能加入该球队。”
End If
```

2. If...Then...Else 结构

使用 If...Then...Else 结构,可从几个流程分支中选择一个执行。它的基本语法是:

```
If 条件 1 Then
语句组 1
【Elseif 条件 2 Then
语句组 2】
...
【Else 语句组 n】
End If
```

执行到 If...Then...Else 结构时,VB 会首先测试条件 1,如果它为 False,VB 就测试条件 2,依次类推,直到找到为 True 的条件。一旦找到一个为 True 的条件时,VB 会执行相应的语句组,然后执行 End If 语句后面的代码。如果所有条件都是 False,那么 VB 便执行 Else 后面的语句组,再执行 End If 语句后的代码。例如我们可以用 If...Then...Else 结构编写一段程序,用户每单击一次窗体,便在窗体上依次显示一行诗,待显示完四行后,再循环显示。

```
Private Form_Click()
Static As Integer
Count = Count + 1
If Count = 1 Then
Print “离离原上草,”
Elseif Count = 2 Then
Print “一岁一枯荣。”
Elseif Count = 3 Then
Print “野火烧不尽,”
Elseif Count = 4 Then
Print “春风吹又生。”
Count = 0
Else
Count = 0
End Sub
```



图 4.13 一首小诗

程序执行结果如图 4.13 所示,用户在窗体上单击四次后,显示四行诗。

注意：由于要记忆单击窗体的次数 Count,所以变量 Count 须声明为静态变量。

3. Select Case 结构

当需要完成多重判定的任务时,可以使用 Select Case 结构,这种结构的语法是:

Select Case 测试条件

【Case 表达式 1

语句组 1】

【Case 表达式 2

语句组 2】

...

【Case Else

语句组 n】

End Select

将上面的例子改写成 Select Case 结构,则程序代码如下:

```
Private Sub Form1_Click()
```

```
Static Count As Integer
```

```
Count = Count + 1
```

```
Select Case Count
```

```
Case 1
```

```
Print "离离原上草,"
```

```
Case 2
```

```
Print "一岁一枯荣。"
```

```
Case 3
```

```
Print "野火烧不尽,"
```

```
Case 4
```

```
Print "春风吹又生。"
```

```
Count = 0
```

```
Case Else
```

```
Count = 0
```

```
End Select
```

```
End Sub
```

4.4.2 循环语句

利用循环控制程序结构可以使程序重复某项工作多遍,VB 主要有两种循环结构,即 Do...Loop 和 For...Next

1. Do...Loop 结构

使用 Do 循环重复执行一语句块,并计算测试条件以决定何时结束循环,循环条件必须是一个数值或者值为 True 或 False 的表达式。Do...Loop 语句有 4 种形式,其中一种循环形式为:

```
Do While 条件语句组
```

```
Loop
```

当 VB 执行该 Do 循环时首先测试循环条件,如果循环条件为 False 或零,则跳过循环语句;如果循环条件为 True 或非零,则进入循环体,执行完循环语句组后,再测试循环条件,直到循环条件为 False 时才退出循环。

Do...Loop 的第二种语句形式是:

```
Do  
语句组  
Loop While 循环条件
```

这种循环形式是先执行语句,然后在执行完循环语句后测试循环条件,这种形式的循环体内语句至少会执行一次。而第一种形式的循环结构当循环条件一开始便为 False 时,则一次都不执行循环内的语句。

还有两种 Do...Loop 的循环形式是:

```
Do until 循环条件  
语句组  
Loop 和 Do  
语句组  
Loop Until 循环条件
```

这两种形式当循环条件为 False 而非 True 时才能执行循环。

2. For...Next 结构

使用 Do 循环时,一般不知道要执行多少次循环,只能由循环条件决定是否继续循环。如果确切知道要执行多少次循环,宜用 For...Next 结构,这种循环使用一个计数器变量,每执行一次循环,计数器变量的值就会增加或者减少。For 循环的语法如下:

```
For 计数器 = 初值 To 终值 [Step 增量]  
语句组  
Next [计数器]
```

其中的参数:计数器,初值,终值和增量都必须是数值型的。

VB 在执行 For 循环时,将进行下列操作:

- (1) 设置计数器等于初值。
- (2) 测试计数器是否大于终值(若增量为负,则测试计数器是否小于终值),若是,则退出循环,若不是,进行操作(3)。
- (3) 执行循环语句。
- (4) 计数器增加增量(缺省时,增加 1)。
- (5) 重复(2)到(4)

提示: For 循环的增量参数可正可负。如果增量为正,则初值必须小于等于终值,否则,一次都不能执行循环内的语句。如果增量为负,则初值必须大于等于终值,否则,也一次都不能执行循环内的语句。增量若不设置,则缺省为 1。

3. 退出循环

有时候,在执行循环过程中满足了某种条件,而该条件在测试条件中没有给出,往往想不待循环完毕就退出循环以节省时间,这时可用 Exit 语句直接从 Do 循环或 For 循环中退出。它

的语法形式是：

```
Do [While | Unit 循环条件]
```

```
语句组
```

```
Exit Do
```

```
语句组
```

```
Loop
```

和

```
For 计数器 = 初值 To 终值 [Step 增量]
```

```
语句组
```

```
Exit For
```

```
语句组
```

```
Next [计数器]
```

Exit Do 和 Exit For 在查找数据时特别有用,一旦找到某数后可以立即退出循环,而不再循环下去。例如:

```
Dim x As Integer
```

```
For x = 1 To 100
```

```
If x = y Then Exit For
```

```
Next
```

在 1 到 100 间找一个等于变量 Y 的数字,找到后立即退出循环。

4.5 过程和函数

用户都见过生产机器(如汽车)的过程,整台机器并不是在同一个地方建造的。机器的不同组件常在不同地方建造测试,最后才将各组件组装到一起。这种大批量生产的模块化方法使整个生产操作更有效率。

编程也是这样,如果把一个庞大的程序分割成较小的逻辑部件,每个逻辑部件完成一定的功能,那么就可以简化程序设计任务。划分的逻辑部件叫过程,它是执行某一特定任务的代码段,可用于替代重复任务或共享任务。

使用过程和函数比用单个模块编写所有代码具有优越性。可以单独测试各个任务,过程中的代码量越小,调试就越容易;每次需要执行相同任务时调用过程而不重复程序代码,可以清除冗余代码。

VB 中,除了事件过程,还有 Sub 过程和 Function 过程。Sub 过程又称为子过程,它不返回值;Function 过程又称为函数,它可以返回值。为了与事件过程相区分,将自定义的 Sub 过程称为通用过程。

注意:通用过程必须由程序调用才能够运行,而事件过程除了响应用户引发的事件或系统引发的事件外,总是处于空闲状态,应用程序不能直接调用事件过程。本书以后章节的“过程”都指通用过程。

4.5.1 定义和调用过程

如果在程序设计中,有几个不同的事件过程要执行一个同样的任务,就可以将这个任务用一个通用过程来实现,并由事件过程来调用它。过程是在响应事件时执行的代码块,将模块中的代码分成过程后,在应用程序中查找和修改代码方便多了。

定义通用过程可用下面的语法形式:

```
【Private|Public】 【Static】Sub 过程名(参数列表)
语句组
End Sub
```

每次应用程序调用过程都会执行 Sub 和 End Sub 之间的语句组,缺省情况下,模块中的子过程都是公用的(Public),因此在应用程序中可随处调用它们。

下面举一个例子,我们可以创建一个应用程序,窗体上有一个按钮 Command1。当用户在窗体上单击时,窗体标题文字为【单击窗体】,当用户在按钮上单击时,窗体标题文字为【单击按钮】。用一个通用过程来完成改变窗体 Caption 属性的任务,并将该过程命名为:ChangeForm,而窗体标题可作为参数由调用者调入,且为字符串类型,设该参数为 FormTitle。那么这个通用过程是:

```
Sub ChangeForm(FormTitle As String)
Form1.Caption = Form Title
End Sub
```

在定义好了过程以后,还要在应用程序中调用该过程才能进行相应的操作,所以,可以在窗体 Form1 和按钮 Command1 的 Click 事件过程中调用 ChangeForm 过程,完整的程序代码为:

```
Option Explicit Private Sub ChangeForm(FormTitle As String)
Form1.Caption = FormTitle
End Sub
Private Sub Form1_Click()
Call ChangeForm("单击窗体")
End Sub
Private Sub Command1_Click()
ChangForm "单击按钮"
End Sub
```

程序运行结果如图 4.14 和图 4.15 所示。



图 4.14 单击窗体



图 4.15 单击按钮

提示：在上例的 Form1_Click() 和 Command1_Click() 事件过程中，分别使用了两种不同的方法调用过程 ChangeForm。一种是 Call ChangeForm("...")，另一种是 ChangeForm"..."。使用 Call 语法时，参数必须在括号内；省略 Call 关键字时，必须省略参数两边的括号，过程和参数间用空格隔开。

4.5.2 定义和调用函数

VB 包含了许多内部函数，如 Sin(正弦)、Sqr(开方)或 Abs(取绝对值)等。用户也可以用 Function 语句编写自己的函数过程。

函数过程的语法是：

```
【Private|Public】【Static】Function 函数名(参数列表)【As 数据类型】  
语句组  
End Function
```

Function 过程与 Sub 过程的不同之处主要是：

1. Function 过程可返回一个值到调用它的过程，而 Sub 过程不能返回值。
2. 调用方法不同。一般说来，在赋值语句右边的表达式中包含函数过程名和参数，这就调用了函数，而 Sub 过程常用 Call 调用。
3. 与变量一样，函数过程有数据类型，这就决定了返回值类型(缺省情况下，数据类型为 Variant)。

在 VB 中，调用 Function 函数和调用任何内部函数的方法是一样的，在表达式中直接写上它的名字和参数列表。

下面举一个例子，假设你的生日是 1974 年 12 月 2 日，可以调用函数 YourAge 来计算年龄，并在窗体上显示年龄。

```
Option Explicit  
Function YourAge(Birthday As Date) As Integer  
YourAge = (Now - Birthday) \ 365  
End Function  
Private Sub Form_Load()  
Dim intAge As Integer  
intAge = YourAge( #12/2/74 # )  
Print intAge  
End Sub
```

而且，在程序中还可以像调用 Sub 过程一样调用函数，但这种方法将会放弃返回值。例如：

```
YourAge #12/2/74 #  
Call YourAge( #12/2/74 # )
```

4.5.3 过程和函数的参数

过程和函数的参数是过程和函数与调用者之间进行信息交换的途径。过程的参数可以声明其数据类型，在缺省声明情况下，参数为 Variant 数据类型。在程序中给参数传递的是一个

表达式或者函数,而不是数据类型。VB 能自动计算表达式,并能按要求的类型将值传递给参数。

在 VB 中,参数缺省是按地址传递的,也就是使过程按照变量的内存地址去访问实际变量的内容。这样,将变量传递给过程时,通过过程可永远改变变量值。

例如,在下面的例子中,过程 try 通过地址将变量 Y 传入,在过程中改变了 Y 的值,当返回时,Y 值已经发生改变。

```
Option Explicit
Sub try(X As Integer)
x = x + 10
End Sub
Private Sub Form_Load()
Dim Y As Integer
Y = 10
Print "调用 try 前:Y = "&Y
try Y
Print "调用 try 后: Y = "&Y
End Sub
```

程序运行结果如图 4.16 所示。

而按值传递参数时,传递的只是变量的副本,即使过程改变了这个值,所作的改变只影响副本而不会影响变量本身。

按值传递参数时,必须在参数列表前加上 ByVal 关键字。例如把上例的 try 过程改写为按值传递参数:

```
Sub try(ByVal X As Integer)
X = X + 10
End Sub
Private Sub Form_Load()
Dim Y As Integer
Y = 10
Print "调用 try 前:Y = "&Y
Call try (Y)
Print "调用 try 后:Y = "&Y
End Sub
```

程序运行结果如图 4.17 所示,窗体上两次显示的 Y 值都是 10,没有发生改变。

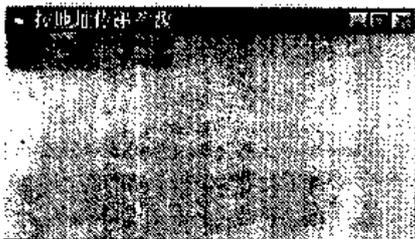


图 4.16 按地址传递参数改变实际变量值

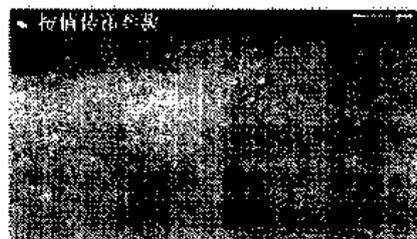


图 4.17 按值传递参数不改变实际变量值

4.5.4 退出过程

在特殊情况下,用户想在过程未执行完时中途退出可以使用 Exit Sub 或 Exit Function 语句。Exit Sub 或 Exit Function 语句可以出现在过程主体内的任何地方,它们的语法和 Exit For 以及 Exit Do 相似。

```
Sub|Function 过程名(参数列表)
语句组 1
Exit Sub|Function
语句组 2
End Sub|Function
```

下面的例子中,过程 Check 用来测试传递过来的数值是否小于 100,如果小于 100,则在窗体上显示该数值,否则,退出子过程 Check。

```
Option Explicit
Sub Check (X As Integer)
If X < 100 Then
Print x
Else
Exit Sub
Print x
End If
End Sub
Private Sub Form_Load()
Dim Y As Integer
Y = 200
Call Check(Y)
End Sub
```

运算该程序,发现窗体上什么都没有显示。这是因为 Y = 200 大于 100,在过程 Check 中,X = 200,程序跳过 If 后的语句,执行 Else 后面的语句。虽然 Else 后面的语句中有 Print X,但由于这之前有一条退出子过程的语句 Exit Sub,程序执行到这条语句时,直接从子过程中退出,不再执行其后的语句 Print X,因此窗体上并不显示任何内容。

4.6 本章小结

VB 编程语言继承了 Basic 语言易学易用的特点,本章着重讲述了 VB 编程语言的基本知识,主要包括以下内容:

1. 编写程序的常用规则和良好的编程习惯。
2. 如何使用变量和常量。
3. 在 VB 中进行各种运算。
4. VB 的两种流程控制语句:条件判定和循环控制。
5. 如何使用过程和函数。

第 5 章 窗体设计

窗体也称作表单(Form),是一种特定的类,它用于定义一个窗口。窗体是设计 VB 应用程序的一个基本平台,几乎所有的控件都是添加在窗体上的,而大多数应用程序都是由窗体开始执行的。所以介绍窗体的设计方法是介绍 VB 应用程序开发必不可少的重要内容。

在本章中,我们将集中介绍窗体的一些内容,包括有关窗体的属性,如何利用工具箱和窗体编辑器向窗体上添加控件,如何设置工程的启动窗体,窗体从生成到卸载的生命周期以及多文档界面 MDI 窗体的基本概念。

5.1 窗体的属性和事件

窗体对象是 VB 应用程序的基本构造模块,是运行应用程序时,与用户交互操作的实际窗口。窗体有自己的属性、事件和方法来控制窗体的外观和行为。例如,窗体的 Caption 属性确定显示在窗体对象标题栏中的内容,或最小化图标下的文本;而 Circle 方法则可以在窗体上画一个圆或一个椭圆。

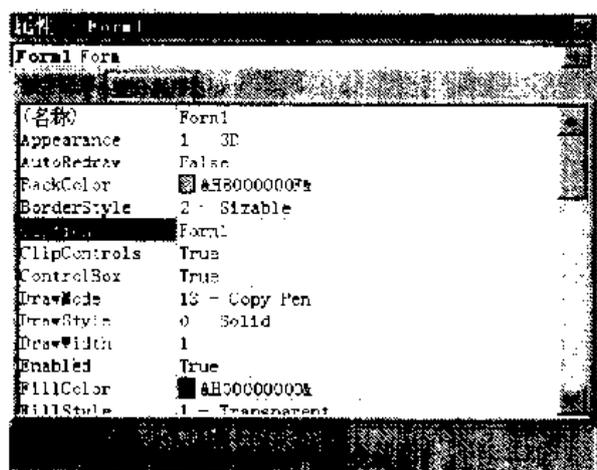


图 5.1 窗体的【属性】窗口

5.1.1 窗体的属性

设置窗体的第一步是设置它的属性,这可以在程序设计时在属性窗口中手工设置,也可以在程序运行时由代码实现。图 5.1 显示的是窗体的【属性】窗口。

窗体的属性很多,它们不仅影响着窗体的外观,还控制着窗体的位置、行为等其他方面。以下列出了窗体的一些经常使用的属性以及它们的说明。

1. Name 属性

Name 属性允许用户给窗体设置合适的名字,一个新窗体的缺省名是窗体 Form 加上一个特定的整数。例如,第一个新窗体是 Form1。窗体不能用系统中的关键字来命名,否则可能在用户的代码中引起冲突。引用窗体的 Name 属性的语法形式为:

```
Form1.Name
```

其中 Form1 为窗体名,下同。

2. Caption 属性

Caption 属性决定窗体标题栏中显示的文本。当用户创建一个新窗体时,其标题栏的缺省文本也是窗体 Form 加上一个特定的整数,例如 Form1 等。引用窗体的 Caption 属性的语法是:

```
Form1.Caption
```

3. BorderStyle 属性

BorderStyle 属性可以控制窗体边界类型及如何调整大小,缺省值为 2。允许用户通过窗体边缘的热点改变窗体的大小和形状。在代码中引用 BorderStyle 属性的方法是:

```
Form1.BorderStyle = [Value]
```

当 Value 值是 0 时,设置窗体无框架;Value 值为 1 时,设置窗体不可调整大小且具有单线框架;Value 值为 2 时,设置窗体可调整大小且具有双线框架;Value 值为 3 时,设置窗体不可调整大小且具有双线框架。

4. ControlBox 属性

当用户运行应用程序时该属性有效,用来在窗体标题栏左边设置一个控制框,单击控制框显示一个控制菜单,有【最大化】、【最小化】、【关闭】等菜单选项。ControlBox 属性缺省设置为 True,能够使窗体显示控制框。当窗体的 BorderStyle 属性设置为 0 时,控制框将不能被显示。

5. BackColor 和 ForeColor 属性

BackColor 属性决定窗体的背景颜色,ForeColor 属性决定窗体的前景颜色。引用这两种属性的语法是:

```
Form1.BackColor = [Color]
```

```
Form1.ForeColor = [Color]
```

6. AutoRedraw 属性

AutoRedraw 属性控制窗体图像的重建,可以设置成 True 或 False。在其他窗口覆盖某窗口后,又返回该窗口时,如果将 AutoRedraw 属性设置为 True,VB 将自动刷新或者重画该窗体的所有图形。如果将该属性设置为 False,VB 必须调用一事件过程来执行该项任务。此属性是使用图形方法如 Circle、Point、Cls 和 Print 的核心,设置 AutoRedraw 为 True,可以在窗体中重画这些方法的输出。

7. Height 和 Width 属性

Height 属性和 Width 属性可以确定窗体的初始高度和宽度,包括边框和标题栏。对于一个窗体,Height 和 Width 属性随用户或代码确定的窗体大小而改变,它们的最大值由系统决定。

8. Left 和 Top 属性

Left 和 Top 属性根据屏幕左上角确定窗体的位置。Left 属性确定窗体最左端和它的容器最左端之间的距离;Top 属性确定窗体最上端和它的容器最上端之间的距离。通常 Left 和 Top 属性在一个窗体中总是成对出现的,当通过用户或通过代码移动窗体时,这两个属性值都会随之改变。

9. MaxButton 和 MinButton 属性

MaxButton 和 MinButton 属性决定窗体是否能最大化或最小化。MaxButton 属性为 True 时,表明窗体有最大化按钮;为 False 时,表明窗体没有最大化按钮。MinButton 属性为 True 时,表明窗体有最小化按钮;为 False 时,表明窗体没有最小化按钮。要显示最大化或最小化按钮,BorderStyle 属性应设置为 1 或 2。当一个窗体被最大化时,最大化按钮会自动变为恢复按钮。

10. Icon 属性

Icon 属性决定在运行时窗体最小化时显示的图标。用户能够安排一个图标给窗体用来表示窗体的功能,如果用户没有给窗体指定图标,VB 会为窗体设置一个缺省图标。用户在为窗体设置图标时,可利用 VB 的图标库,该图标库在 Icons 子目录下。用户只要在属性窗口单击

Icon 属性右边的浏览按钮,从显示的对话框中选择其中一个目录,然后浏览是否有自己满意的图标。

11. WindowState 属性

WindowState 属性可以把窗体设置成在启动时最大化、最小化或正常大小。WindowState 属性为 0 时,窗体显示为正常大小;为 1 时,窗体最小化成图标;为 2 时,窗体最大化显示。

12. Enabled 属性

Enabled 属性决定窗体是否对用户产生的事件发生反应。Enabled 属性为 True 时,允许窗体对事件作出反应,为 False 时,禁止窗体对事件作出反应。

13. FontBold、FontItalic、FontStrikethru、FontUnderLine 和 FontTransParent 属性

前四种属性确定下列四种字形:黑体、斜体、删除线体和下划线体,FontTransParent 属性决定窗体中背景文字或图形是否包括在特殊字体的字符中。这些属性的设置有两种:True 和 False。True 设置按缺省的 FontBold 和 FontTransParent 把字型格式化,允许背景文字或图形通过字符字形显示。

14. Visible 属性

Visible 属性确定窗体是被显示还是被隐藏。设置为 True 时,能够使窗体可见;设置为 False 时,窗体将被隐藏。若要在启动时隐藏一个对象,可设置 Visible 属性为 FALSE。

15. FontSize 属性

FontSize 属性确定窗体中文本的字体大小,缺省时由系统定义。用户想要改变字体大小,可以设置 FontSize 属性为不同的数值。

16. Picture 属性

Picture 属性确定窗体中一个被显示的图片。该属性的设置值有:None、Bitmap、icon、metafile。none 设置是缺省值,用来确定窗体中没有图片。Bitmap、icon、metafile 等设置值用来指定一个图形。

17. ShowInTaskbar 属性

ShowInTaskbar 属性决定一个窗体对象是否出现在 Windows 95/98 任务栏中,该属性的值在运行时为只读状态。

18. Moveable 属性

Moveable 属性指定对象是否可移动。

5.1.2 窗体的事件

在 VB 中,事件就是能被对象所识别的动作,使用鼠标单击或者双击就是最常用的事件。此外,用户的键盘输入、鼠标的移动、窗体的载入,还有定时器产生的定时信号等等,都是事件。如果说属性决定了对象的外观,方法决定了对象的行为,那么事件就决定了对象之间联系的手段。其中与窗体有关的重要事件有以下几种:

1. Click 事件

用户在窗体上单击鼠标左键时触发 Click 事件,VB 将调用 Form_Click 事件过程。

2. DblClick 事件

用户在窗体上双击鼠标左键时触发 DblClick 事件。

3. Load 事件

一旦装载窗体,启动应用程序就自动产生该事件,Load 事件适用于在启动应用程序时对

属性和变量的初始化。

4. Unload 事件

删除窗体时发生 Unload 事件。当该窗体再被装载时,它的所有控件都要重新初始化。这个事件是由用户动作(用控件菜单关闭窗体)或一个 Unload 语句触发的。

5. GotFocus、LostFocus 事件

当窗体收到或失去焦点时,GotFocus 或 LostFocus 事件会发生。

6. Activate、Deactivate 事件

激活窗体时发生 Activate 事件,取消该活动窗体而激活另一个窗体时该窗体发生 Deactivate 事件。窗体可通过用户的操作变成活动窗体,如用鼠标单击窗体的任何部位或在代码中使用 Show 或 SetFocus 方法。

7. Paint 事件

重新绘制一个窗体时发生 Paint 事件。当移动、放大、缩小该对象或一个覆盖该对象的窗口移动后,该窗体暴露出来,就会发生此事件。

5.2 窗体的设计

窗体设计包括如何创建窗体以及如何往窗体上添加、移动或删除控件等内容,在进行窗体设计时,还要讲究一些原则以保证设计的窗体美观、易用。

5.2.1 创建新窗体

要创建一个新的 VB 应用程序必须启动一个新的工程文件。选择【文件】菜单中的【新建工程】命令选项,系统弹出如图 5.2 所示的【新建工程】对话框。

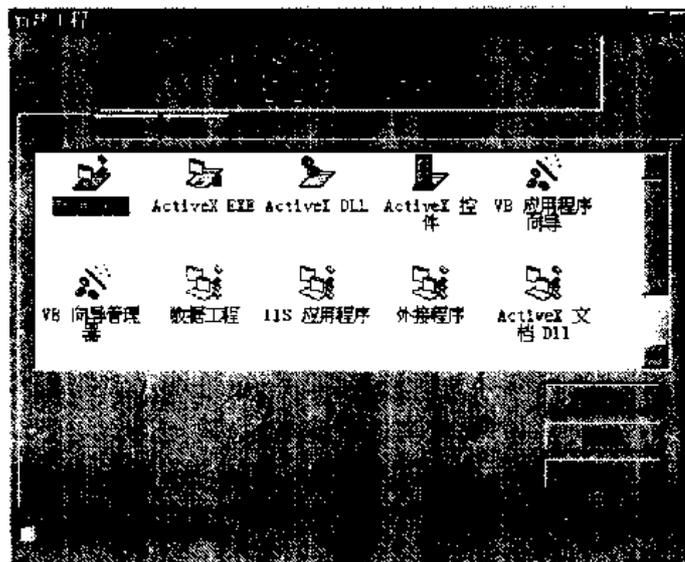


图 5.2 【新建工程】对话框

选中【标准.EXE】后,再单击对话框中的【确定】按钮,这时系统建立一个不含任何控件的窗体,并默认该窗体的 Name 属性为 Form1,Caption 属性也是 Form1。此时屏幕显示如图 5.3 所示。

用户可以看出,窗体编辑器的主体是一个空空的窗体,它的初始大小是缺省的,如果用户

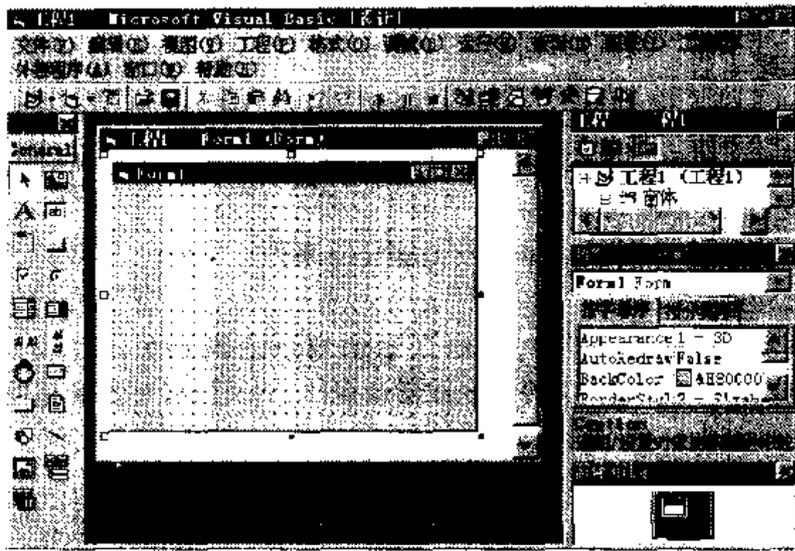


图 5.3 新建窗体后显示的屏幕

觉得窗体大小不合适,可以调整窗体大小。首先在窗体上单击来选中窗体,这时窗体周围出现 8 个小方块(控制柄),其中有 5 个是空心的,3 个是实心的,将鼠标指针移动到实心控制柄上,指针会变成双箭头形状,这时按住鼠标左键并拖动,就可以改变窗体的大小。

如果要精确设置窗体的大小,可以到属性窗口中设置窗体对象的 Width 和 Height 属性的值。

5.2.2 向窗体上添加控件

一个空空的窗体并不能完成什么工作,只有在窗体上添加不同的控件,并且使用代码进行控制,才能创建出美观实用的应用程序。向窗体上添加控件要用到控件工具箱和窗体编辑器。

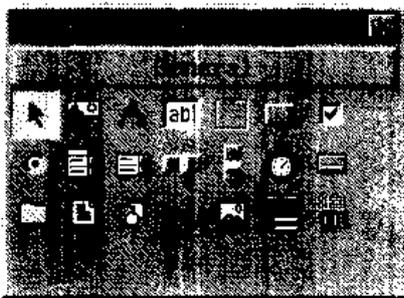


图 5.4 浮动的控件工具箱

在初次进入 VB 6 的集成环境中时,就可以看到默认的控件工具箱。同其他窗口一样,工具箱可以是连接状态,也可以是浮动状态(如图 5.4 所示)。有关各个控件的具体使用,在下一章中将有详细介绍。

如果 VB 6 集成环境中没有显示控件工具箱,可以单击工具栏上的【工具箱】按钮,或者选择【视图】菜单中的【工具箱】选项。如果要关闭工具箱,那么只要单击工具箱右上角的【关闭】按钮即可,也可在工具箱上单击鼠标右键,在弹出的快捷菜单中选择【隐藏】选项。

在工具箱中,除了【指针】按钮是用来选择控件的工具外,其余的都是控件按钮。使用工具箱添加控件要在窗体编辑器窗口中进行,用户可以单击【工程资源管理器窗口】中的查看对象按钮,或者选择【视图】菜单中的【对象】选项,还可以使用组合键 Shift + F7 来进入窗体编辑窗口,向窗体上添加控件有两种常用方法:

1. 在工具箱的某个控件按钮上单击,则该按钮会凹下去,而且鼠标移到窗体上时指针会变为“+”形状。用户可以在窗体中认为合适的位置按下鼠标左键,然后拖动鼠标来设置控件的大小,当松开鼠标左键后,控件就会出现在窗体上。

2. 在工具箱的控件按钮上双击,则窗体的中央就会出现一个相应的控件。控件大小为系

统默认尺寸。

注意：如果多次双击同一个控件按钮，则窗体的中央会重叠多个控件，用户如果不注意的话，会认为只有一个控件。所以，在添加了一个控件后，要随时将它移到其他合适的位置上去。

5.2.3 对控件的操作

当用户把控件都添加到窗体上以后，对各个控件的操作都在窗体编辑器中进行，下面以命令按钮为例，讲述控件的一些基本操作。

1. 控件的选择

在窗体编辑器中，可以对窗体上的控件进行各种操作，在进行操作之前，首先要选中控件。用户只要在相应控件上单击就可选择一个控件。被选中的控件周围会出现 8 个控制柄，如图 5.5 所示。用户可以使用这些控制柄改变控件大小。

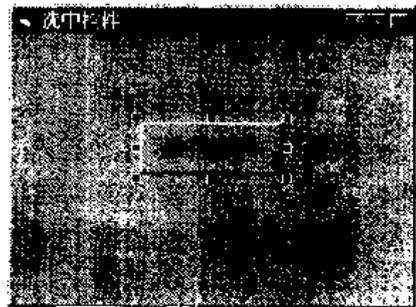


图 5.5 选中控件后，控件周围有控制柄

如果要同时选择多个控件，可以使用下列两种方法之一进行：

(1) 在一个控件上单击，选中该控件，然后按 Ctrl 键或 Shift 键不放，再单击其他要选择的控件，就可以同时选中多个控件。这种方法适用于选择多个不相邻的控件。

(2) 如果要选择一个区域内的所有控件，可以单击工具箱上的【指针】按钮，然后拖动鼠标在窗体上画一个框，包围所要选择的多个控件，释放鼠标按键后，框中所有控件都被选中。

如果要撤消对控件的选择，可以在窗体上单击，此时选中状态就会转移到窗体上。如果要撤消对多个选中控件中某一个控件的选择，需要按住 Ctrl 键或者 Shift 键，然后单击需要撤消选择的控件。

2. 调整控件大小

要调整控件大小，应当首先选中这个控件。然后将鼠标指针移动到控制柄上，指针会变成双箭头形状(如图 5.6 所示)，这时按住鼠标左键并拖动，可以改变控件大小。

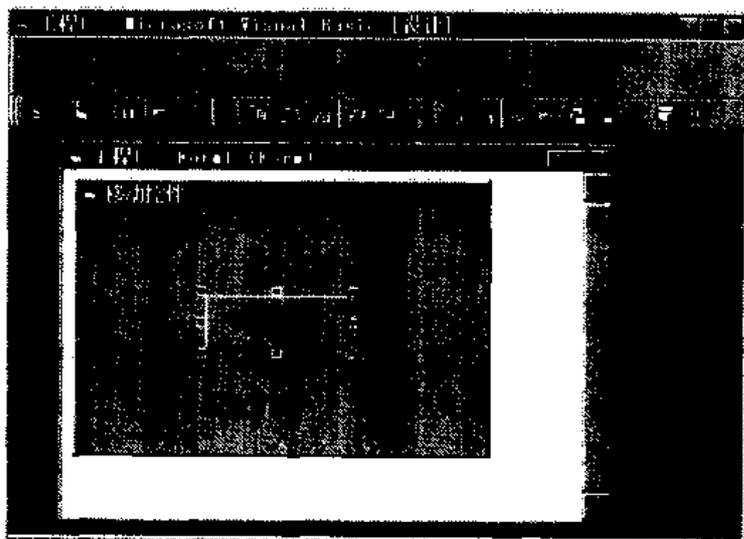


图 5.6 调整控件大小

注意：有的控件大小不能改变，如 Timer(定时器)控件；有的控件只能改变宽度，不能改变高度，如 DriverListBox 控件。

如果要设置控件的精确尺寸，应在相应的属性窗口中设定该控件的 Height 和 Width 属性。

如果要设置多个控件的大小，应先选中这些控件，在所选择的控件中，总有一个控件是被实心的控制柄包围着，通常这个控件是用户最后选择的一个控件，称为主控件。然后使用【格式】菜单下的【统一尺寸】子菜单下的选项，如图 5.7 所示。

VB 将按照主控件的相应尺寸来设置其他控件的大小。因此，用户可以改变主控件来设置控件大小的基准。如果要改变主控件，只需在已经选中的控件中单击某个控件，这时该控件便被实心控制柄包围，成为主控件。

3. 移动和删除控件

如果用户要删除某个控件，只要选中要删除的控件，然后按 Del 键即可。也可以在选中控件后，单击鼠标右键，弹出如图 5.8 所示的快捷菜单，再选择【剪切】或【删除】选项。

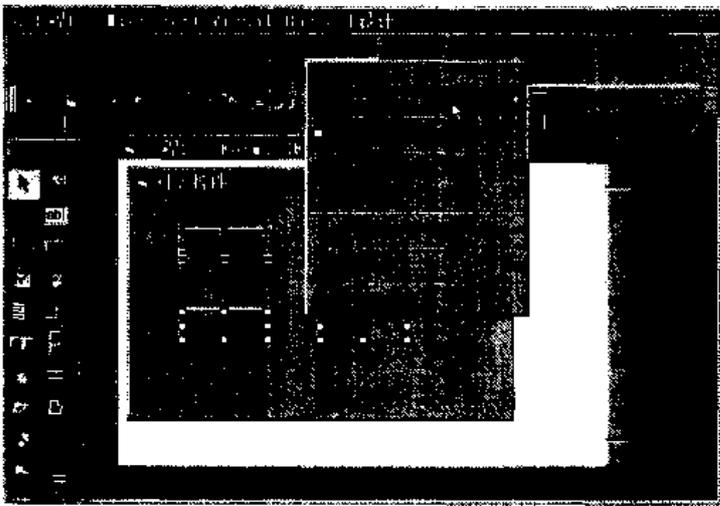


图 5.7 调整多个控件大小

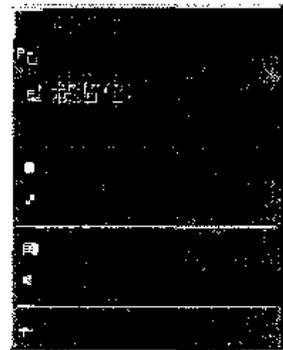


图 5.8 控件快捷菜单

提示：要删除一个窗体，不能使用 Del 键，而应先在【工程资源管理器窗口】中选中该窗体模块，然后选择【工程】菜单下的【移除 窗体名】选项。

如果用户要移动控件，只要将鼠标指针移动到相应控件上，按下鼠标左键并拖动到合适的位置。如果要同时移动多个控件，可以先选中这些控件，然后拖动其中的一个，就会使选中的控件都发生移动，并且它们之间相对位置不变。如果要对齐多个控件，可以先同时选中这些控件，然后选择【格式】菜单中的【对齐】子菜单中的选项来进行不同方式的对齐，对齐的基准是选中控件中的主控件，如图 5.9 所示。

在移动控件的过程中，用户可以发现控件的边缘总是与窗体上规则排列的小点对齐，这些小点被称作网格。用户可以根据自己的需要设置有关网格的选项，方法是选择【工具】菜单中的【选项】项，系统弹出【选项】对话框，然后选择对话框中的【通用】选项卡，如图 5.10 所示。在该选项卡中，用户可以自行设置是否显示网格以及网格间距，还可以决定控件是否对齐到网格。

4. 锁定控件

用户如果对控件在窗体上的布局已经满意了，那么应选择【格式】菜单中的【锁定控件】选

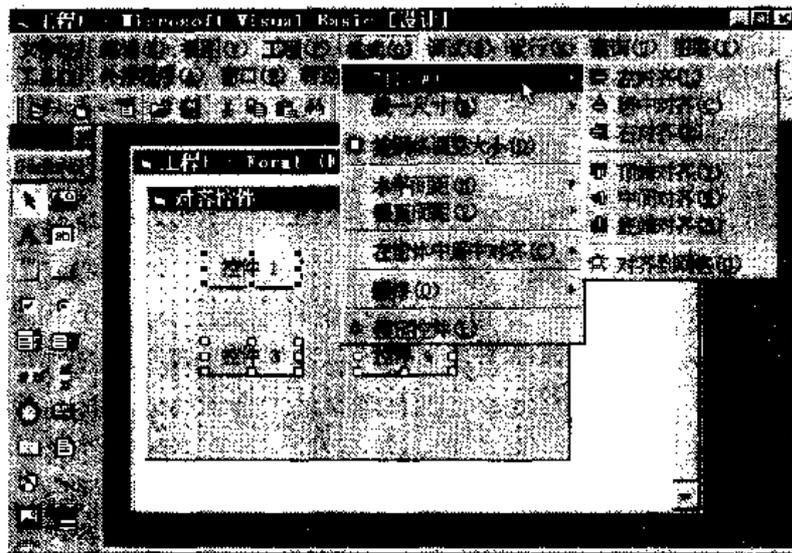


图 5.9 对齐控件

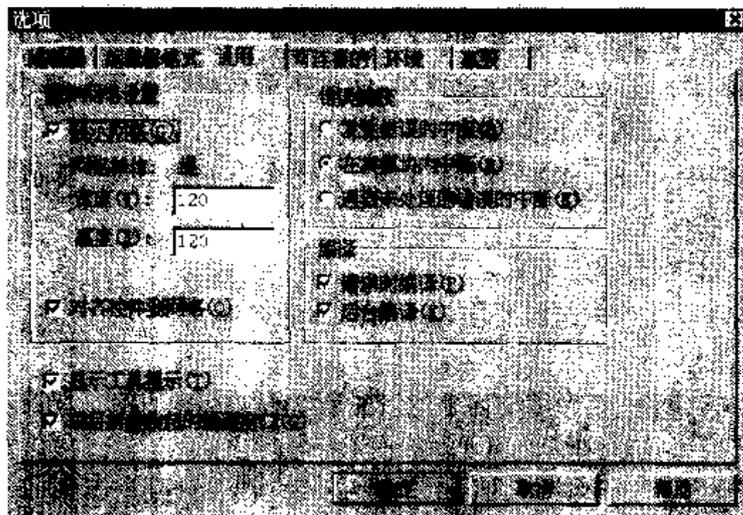


图 5.10 【选项】对话框的【通用】选项卡

项锁定当前窗体上的所有控件。这样,当它们处于指定位置时就不会由于不小心而被移动。

锁定控件的功能是全面性的,并不针对个别控件来设置,选择【锁定控件】菜单项时,窗体上所有控件全被锁定。选中控件后,八个句柄都呈白色显示(如图 5.11 所示)。如果要取消锁定,只要再选择一次【锁定控件】项即可。

5. 在程序中引用其他窗体上的控件

在开发比较大的应用程序时,经常会用到多个窗体,因此往往要在程序中引用其他窗体上的控件。如果要引用其他窗体上的控件,只要在控件的名称前加上窗体的名称,中间利用“.”或者“!”分隔。例如,在下面的语句中,可以控制窗体 Form2 中的文本框 txtName 显示的内容。

```
Form2.txtName.Text = "How are you!"
```

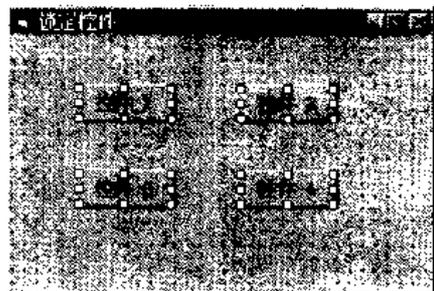


图 5.11 锁定控件

或者:

```
Form2! txtName.Text = "How are you!"
```

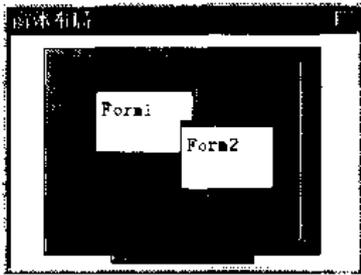


图 5.12 窗体布局窗口

5.2.4 设置窗体位置

用户可以使用以下两种方法来确定窗体在屏幕上出现的初始位置:

1. 在窗体的属性窗口中设置 Top 和 Left 属性值, Top 属性确定窗体最上端和它的容器最上端之间的距离; Left 属性确定窗体最左端和它的容器最左端之间的距离。

2. 使用窗体布局窗口设置窗体的位置更为直观。例如图 5.12 所示的布局窗口中,有两个窗体 Form1 和 Form2。用户可以用鼠标将窗体 Form2 拖动到布局窗口中虚拟屏幕的中央,这样在程序运行的时候 Form2 就会出现在真实屏幕对应的位置上。

5.2.5 设置启动窗体

每个应用程序都有自己的入口,即开始执行的地方,在缺省情况下,应用程序中的第一个窗体被指定为启动窗体。应用程序开始运行时此窗体就被显示出来。如果用户想在应用程序启动时显示别的窗体,可以按以下步骤改变启动窗体:

1. 在【工程】菜单中,选择【工程属性】选项,系统弹出如图 5.13 所示的【工程属性】对话框。
2. 在对话框中,选择【通用】选项卡。
3. 在【启动对象】列表框中选择要作为启动窗体的窗体。
4. 单击【确定】按钮。

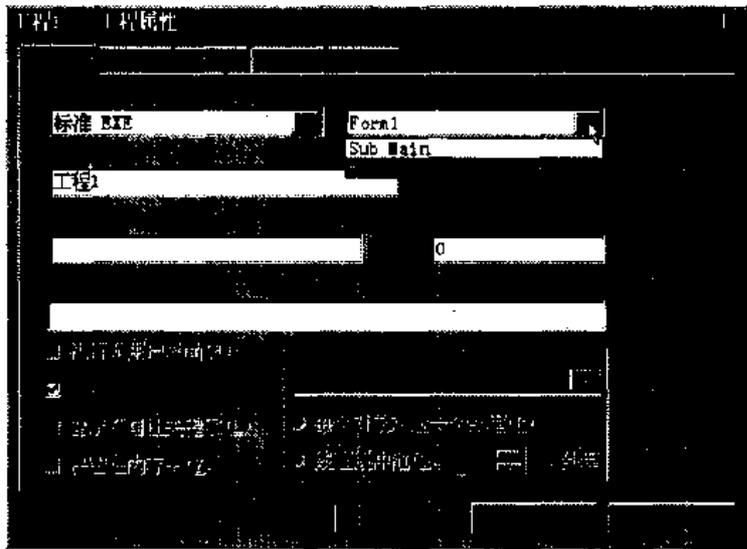


图 5.13 利用【工程属性】对话框【通用】选项卡设置启动窗体

有时候,也许要应用程序启动时不加载任何窗体。例如用户想先装入数据文件,然后加载窗体,或者先显示一个要求输入口令的对话框确认用户的身份。这时,可以在标准模块中创建一个名为 Main 的子过程,例如:

```
Sub Main()  
frmPassword.show '显示要求输入口令的对话框  
End Sub
```

Main 过程必须是一个位于标准模块中的子过程,不能在窗体模块内。将 Sub Main 过程设为启动对象的方法同上,在【工程属性】对话框的【启动对象】列表框中选择“Sub Main”即可。

5.2.6 窗体设计的基本原则

在开发应用程序时,首先要设计一个美观、易用的界面,比如:工具栏、状态栏、工具提示、上下文菜单及选项卡对话框等。因此在进行窗体设计时,用户要注意以下一些基本原则:

1. 符合 Windows 的界面准则

Windows 操作系统的主要优点就是为所有的应用程序提供了公用的界面,只有创建的应用程序界面与 Windows 界面一致时,才便于用户使用。例如菜单栏中,一般地,【文件】菜单在最左边,然后是【编辑】、【视图】等可选菜单,而【帮助】菜单则位于最右边。如图 5.14 所示就是一个典型菜单。

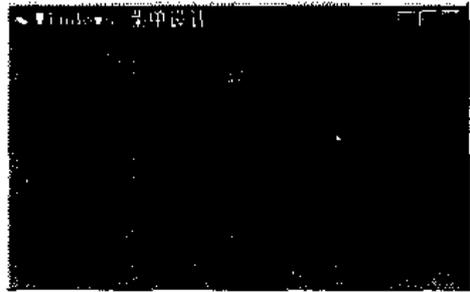


图 5.14 界面设计符合 Windows 准则

2. 注意控件的位置

在大多数界面设计中,不是所有的元素都一样重要,对那些重要的或者频繁访问的元素应当放在显著的位置上,而不太重要的元素就可以放到不太显著的位置上。另外,把元素和控件分成组也很重要,应尽量把信息按功能或者关系进行逻辑分组,还可以使用 Frame 控件来帮助加强控件间的联系。

3. 保持界面的简单明了

界面设计的一个重要原则是简单明了。对于应用程序而言,如果用户觉得界面复杂,则可能会认为程序本身很难,不易使用。

4. 使用图片和图标

在窗体中适当地使用图片与图标可以增加应用程序视觉趣味,而且图片和图标可以比文本更加形象地传达信息。特别是带有表示各种功能图标的工具栏,是一种很有用的界面设备。

5. 使用空白空间

在设计窗体时,在用户界面中使用空白空间有助于突出元素和改善可用性,而一个窗体上有太多的控件会导致界面杂乱无章。同时,各控件间的一致间隔以及垂直与水平方向元素的对齐也可使界面更美观可用,在【格式】菜单中有几个有用的选项。

6. 选取合适的字体

字体也是用户界面的重要部分,用户应选择不同的分辨率和不同类型的显示器上都容易阅读的字体,如一些标准的 Windows 字体:Arial、New Times Roman、宋体、黑体等。另外,选择不同大小、样式的字体也可使界面更丰富多彩。

5.3 窗体的生命周期

一般情况下,程序中不会经常使用处理外部输入的事件过程。但是窗体是控件的载体,是程序运行的主要场所。一些程序运行所必须的初始化操作和退出前的善后工作,往往要在窗

体创建以及退出时被激活的事件过程中进行。因此,用户在编写应用程序时,有必要了解一下窗体从创建到卸载的整个过程。

通常 VB 窗体在应用程序中有以下 4 种状态:

1. 创建状态
2. 加载状态
3. 可见状态
4. 卸载状态

5.3.1 窗体的创建

窗体创建状态开始的标志是 Initialize 事件,窗体创建时最先执行的代码应放在 Form_Initialize 事件过程中。

处于这种状态时,窗体是作为一个对象而存在,但还没有窗口,而且它的控件也不存在。此时只有窗体的代码部分在内存中,而窗体的可视部分还没有调入。其实任何窗体都要经过创建状态,例如:如果执行 Form2.Show,则窗体被创建并开始执行 Form_Initialize,一旦 Form_Initialize 执行完毕,该窗体被加载,便开始下一状态。

5.3.2 窗体的加载

Load 事件标志着加载状态的开始。一旦窗体进入加载状态,Form_Load 事件过程中的代码便开始执行。

Form_Load 事件过程开始执行后,窗体上的所有控件都被创建和加载,而且该窗体有了一个窗口,是通过窗口句柄和设备描述体完成的。另外,很多窗体会自动从创建状态进入加载状态,窗体如果满足下列条件之一便会被自动加载。

- 该窗体在【工程属性】对话框的【通用】选项卡中被指定为启动对象。
- 窗体中首先被调用的属性或方法是 Show 方法,如 Form1.Show。
- 首先被调用的窗体属性或方法是窗体内置的成员,例如调用了窗体的 Move 方法或者使用了窗体中某个控件的属性。
- 用 Load 语句加载窗体,如:Load Form1。

其中上面列举的第一和第二种情况下,一旦 Form_Load 执行完毕,窗体就直接可见,而后两种情况下,Form_Load 执行完毕后,窗体保持加载状态而不可见。

在 VB 编程中,常常加载了某一窗体但从未显示,这是因为:

- 用时钟控件产生计时事件。
- 用功能控件,而不是用户界面控件,如串行通信或访问系统文件。
- 执行 DDE 事务。

加载状态是窗体的一个根状态,在任何时候,只要隐藏了窗体,它就总是从可见状态回到加载状态。但是回到加载状态并不重新执行 Form_Load 事件过程,因为在窗体存活期中 Form_Load 事件过程只运行一次。

5.3.3 窗体的显示

使用窗体的 Show 方法,可以使窗体进入可见状态。一旦窗体可见,用户就能和它交互作用。

要使一个窗体可见,应调用 Show 方法:

Form1.Show

它与设置窗体的 Visible 属性为 True 具有相同的效果。

注意：任何窗体只有加载后才可见。

如果要隐藏一个窗体,应调用窗体的 Hide 方法。当一个窗体调用 Hide 方法后,该窗体就从屏幕上被删除,并且它的 Visible 属性被设置为 False,窗体返回加载状态。用户将无法访问隐藏窗体上的控件,但是对于运行中的 VB 应用程序,隐藏窗体的控件仍然是可用的。在程序中,要判断一个窗体是否处于可见状态,可以引用它的 Visible 属性,例如:

```
Private Sub Form_Load()  
If Form2.Visible Then  
Form2.Hide  
Else  
Form2.Show  
End If  
End Sub
```

在显示窗体时,还可以在 Show 后面加上一个参数,以确定窗体是模态还是非模态。所谓模态窗体,就是指那些只能让用户在本窗体进行选择、输入,却不能切换到其他窗体中的窗体。而非模态窗体就可以允许用户随意在各个窗体之间切换。

显示非模态窗体方法为:

```
Form1.Show 0
```

显示模态窗体方法为:

```
Form1.Show 1
```

或

```
Form1.Show vbModal
```

窗体在可见状态下有两个十分重要的事件是 Activate 和 Deactivate 事件。当一个窗体变成活动窗体时,就会产生一个 Activate 事件;当另一个窗体或应用程序被激活时,该窗体就会产生 Deactivate 事件。在初始化或结束窗体行为时这两个事件特别有用。

5.3.4 窗体的卸载

窗体在卸载时可以是隐藏的,也可以是可见的。若没隐藏,则它将保持可见直到卸载完毕,内存和资源完全收回。

窗体卸载前要发生 Unload 事件,在这之前还要发生 QueryUnload 事件。QueryUnload 提供了停止窗体卸载的机会,如果某些数据希望保存,则此时可提示保存或忽略所作的修改。QueryUnload 的语法是这样的:

```
Private Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)
```

在 QueryUnload 事件中可以包含两个参数:

- Cancel 是一个整数,若将此参数设定为非零整数值,可在所有已装载的窗体中停止 QueryUnload 事件,并停止该窗体和应用程序的关闭。

- Unloadmode 是一个值或一个常数(值可取 0,1,2,3 或 4),指出引起 QueryUnload 事件的原因。

QueryUnload 事件的典型用法是在关闭一个应用程序之前用来确保包含在该应用程序中的窗体中没有未完成的任务。例如:如果还未保存某一窗体中的新数据,则应用程序会提示保存该数据。当一个应用程序正要关闭时,可使用 QueryUnload 事件过程将 Cancel 属性设置为 True 来阻止关闭过程。

注意: QueryUnload 事件是在任一个窗体卸载之前在所有窗体中发生,而 Unload 是在每个窗体卸载时发生。

另外,用户可以使用 QueryUnload 事件来判断窗体卸载的原因:

- Unloadmode = 0 或 vbFormControlMenu, 用户从窗体上的控制菜单中选择【关闭】选项。
- Unloadmode = 1 或 vbFormCode, Unload 语句被代码调用。
- Unloadmode = 2 或 vbAppWindows, 当前 Windows 操作环境会话结束。
- Unloadmode = 3 或 vbAppTaskManager, Windows 任务管理器正在关闭应用程序。
- Unloadmode = 4 或 vbFormMDIForm, MDI 子窗体正在关闭。

所以 QueryUnload 提供了取消关闭窗体的机会,同时也允许在需要时从代码中关闭窗体。下面举一个例子,当用户关闭窗体时,提示用户是否确认要关闭窗体:

```
Private Sub Form1_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    Dim StringM
    String
    Form1.Caption = "窗体卸载"
    If UnloadMode > 0 Then '如果正在退出程序
        StringM = "你确认要退出程序吗?"
        '如果正好关闭窗体
    Else
        StringM = "你确认要关闭窗体吗?"
    End If
    If MsgBox(StringM, vbQuestion + vbYesNo, Me.Caption) = vbNo Then
        Cancel = True '如果用户单击 NO 按钮,则停止 QueryUnload
    End If
End Sub
```

程序运行结果如图 5.15 所示,如果用户单击该窗体标题栏右上角的关闭按钮,则会弹出一个消息框(如图 5.16 所示)询问用户是否确认要关闭窗体。



图 5.15 程序运行结果



图 5.16 单击关闭按钮时弹出的提示框

注意: 在一些情况下,窗体不会接收到 QueryUnload 事件。比如使用了 End 语句结束程序,或者单击了工具栏中【结束】按钮。

5.3.5 结束应用程序

当所有窗体都已关闭并且没有代码正在执行时,事件驱动的应用程序就停止运行。如果最后一个可见窗体关闭时仍有隐藏窗体存在,那么应用程序因为没有可见的窗体而表面为结束了,但实际上应用程序仍在继续运行,直至所有隐藏窗体都关闭为止。这是因为对已卸载窗体的属性或控件的任何访问,都将导致隐含地、不予显示地加载那个窗体。

当应用程序只有一个窗体时,可以使用这样一个简单的语句来结束应用程序:

```
Unload Me
```

其中,Me 为 VB 的一个关键词,用来指定当前的窗体。

如果应用程序有一个以上窗体,可使用 Forms 集合和 Unload 语句。Forms 是一个集合,它包含了每一个在应用程序中加载的窗体。Forms 集合只有一个属性:Count,指定集合中元件的数目。例如,下面的程序就是使用 Forms 集合确保找到并关闭所有窗体。

```
Private Sub Form_Unload()  
Dim counter As Integer  
For counter = 0 to Forms.Count - 1  
Unload Forms(counter)  
Next  
End Sub
```

另外,还可以使用 End 语句来强行结束应用程序而不顾现存窗体或对象的状态。End 语句使应用程序立即结束:在 End 语句之后的代码不会执行,也不会再有事件发生。特别是 VB 将不执行任何窗体的 QueryUnload、Unload 事件过程,而且对象的引用都将被释放。

5.4 MDI 窗体的基本概念

多文档界面(MDI, Multiple Document Interface)允许创建在单个容器窗体中包含多个窗体的应用程序,像 Word 和 Excel 这样的应用程序就具有多文档界面。

MDI 应用程序由父窗口和子窗口组成,它允许用户同时显示多个文档,每个文档显示在它自己的窗口中。文档或子窗口被包含在父窗口中,父窗口为应用程序中所有的子窗口提供工作空间。例如,Word 允许创建并显示不同样式的多文档窗口,如图 5.17 所示。每个子窗口都被限制在 Word 父窗口的区域之内,当子窗体最小化时,它的图标显示在 MDI 窗体的工作空间之内,而不是在任务栏中。当最小化 Word 时,所有的文档窗口也被最小化,只有父窗口的图标显示在任务栏中。

要建立一个 MDI 窗体,应选择【工程】菜单中的【添加 MDI 窗体】选项。一个应用程序只能有一个 MDI 窗体,如果工程中已经有了一个 MDI 窗体,则该【工程】菜单上的【添加 MDI 窗体】选项将不可使用。

注意: 一个应用程序也可以包括标准的、不是包含在 MDI 窗体之内的非 MDI 窗体。MDI

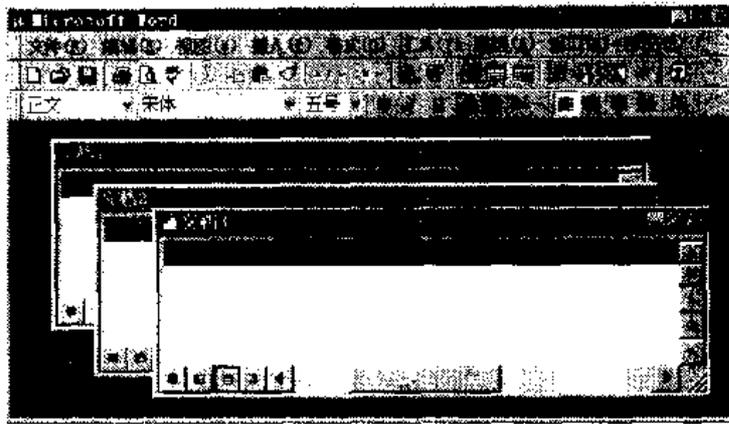


图 5.17 典型的 MDI 窗口

应用程序中标准窗体的典型用法是显示模态对话框。

另外,用户还可以使用应用程序向导快速生成一个 MDI 编辑器,其详细内容在《多文档窗体》一章中将具体介绍。

5.5 本章小结

窗体是设计 VB 应用程序的一个基本平台,本章详细介绍了有关窗体的一些内容,学完本章,读者应该掌握以下几点内容:

1. 窗体的主要属性设置和基本事件。
2. 如何往窗体上添加控件。
3. 在窗体编辑器中选择、移动、删除控件,以及如何在程序中引用其他窗体上的控件。
4. 怎样创建一个窗体。
5. 设置窗体大小和位置的方法。
6. 如何设置启动窗体。
7. 窗体存活期间的四种状态:创建状态、加载状态、可见状态、卸载状态。
8. MDI 窗体的基本概念。

在设计窗体时,为了使窗体美观、易用,还应该注意窗体设计的一些基本原则。

第 6 章 VB 6.0 基本控件

控件用来获取用户的输入信息和显示输出信息。应用程序中可用的控件包括文本框、命令按钮和列表框等。而通过另外一些控件可访问其他应用程序并处理数据,这时,那些远程应用程序就好像是代码的一部分。每个控件都有一组属性、方法和事件。本章介绍 VB 6.0 中的基本控件。

6.1 控件的基本概念

控件在 VB 的程序设计过程中占据重要地位,因为它不仅提供了一些事件过程供用户编写程序代码,完成程序各个流程应运行的工作,而且通过本身的属性设置会影响到窗体操作的界面外观。

在介绍 VB 6.0 的基本控件前,先介绍一下控件的分类以及焦点的基本概念。

6.1.1 控件的分类

VB 的控件有三种广义分类:

- 内部控件,例如 CommandButton 和 TextBox 控件等。这些控件都在 VB 的 .exe 文件中。内部控件总是出现在工具箱中,不像 ActiveX 控件和可插入对象那样可以添加到工具箱中,或从工具箱中删除。
- ActiveX 控件,是扩展名为 .ocx 的独立文件,其中包括各种 VB 版本提供的控件 (DBGrid、DBCombo、DBList 控件等等)和仅在专业版和企业版中提供的控件(例如 ListView、Toolbar、Animation 和标记对话控件),另外还有许多第三方提供的 ActiveX 控件。
- 可插入的对象,例如一个 Microsoft Word 文档对象。因为这些对象能添加到工具箱中,所以可把它们当作控件使用。其中一些对象还支持 OLE 自动化,使用这种控件就可在 VB 应用程序中编程控制另一个应用程序的对象。

1. 内部控件

表 6.1 总结了 VB 工具箱中的内部控件。

表 6.1 VB 6.0 的内部控件

控件名	类名	描述
复选框	CheckBox	显示 True/False 或 Yes/No 选项。一次可在窗体上选定任意数目的复选框
组合框	ComboBox	将文本框和列表框组合起来。用户可以输入选项,也可从下拉式列表中选择选项
命令按钮	CommandButton	在用户选定命令或操作后执行它
数据	Data	能与现有数据库连接并在窗体上显示数据库中的信息

续表

控件名	类名	描述
目录列表框	DirListBox	显示目录和路径并允许用户从中进行选择
驱动器列表框	DriveListBox	显示有效的磁盘驱动器并允许用户选择
文件列表框	FileListBox	显示文件列表并允许用户从中进行选择
框架	Frame	为控件提供可视的功能化容器
水平和垂直滚动条	HScrollBar VScrollBar	对于不能自动提供滚动条的控件,允许用户为它们添加滚动条(这些滚动条与许多控件的内建滚动条不同)
图像	Image	显示位图、图标或 Windows 图元文件、JPEG 或 GIF 文件;单击时类似命令按钮
标签	Label	为用户显示用户不可交互操作或不可修改的文本
线形	Line	在窗体上添加线段
列表框	ListBox	显示项目列表,用户可从中进行选择
OLE 容器	OLE	将数据嵌入到 VB 应用程序中
选项按钮	OptionButton	选项按钮控件与其他选项按钮组成选项组,用来显示多个选项,用户只能从中选择一项
图片框	PictureBox	显示位图、图标或 Windows 图元文件、JPEG 或 GIF 文件。也可显示文本或者充任其他控件的可视容器
形状	Shape	向窗体、框架或图片框添加矩形、正方形、椭圆或圆形
文本框	TextBox	提供一个区域来输入文本、显示文本
定时器	Timer	按指定时间间隔执行定时器事件

2. 标准 ActiveX 控件

VB 学习版包含若干 ActiveX 控件(称为标准 ActiveX 控件),有了这些控件就可在应用程序中引入高级功能。ActiveX 控件的文件扩展名为 .ocx,可手工将它们添加到工具箱中,以便在工程中使用。

表 6.2 总结了 VB 学习版提供的标准 ActiveX 控件。

表 6.2 标准 ActiveX 控件

控件名	类名	描述
公共对话框	CommonDialog	提供一组标准对话框,用于打开和保存文件,设置打印选项、选择颜色和字体等操作
数据绑定组合框	DBCombo	除了提供标准组合框控件的大多数功能之外还提供附加的数据访问功能
数据绑定网格	DBGrid	一个类似工作表的被连结控件,显示一组行和列,分别代表 Recordset 对象中的记录和字段
数据绑定列表框	DBList	除了提供标准 DBList 控件的大多数功能之外还提供附加的数据访问功能
Microsoft FlexGrid	MSFlexGrid	类似 DBGrid 控件,但还具有附加的格式化、分组

6.1.2 焦点的概念

焦点是接收用户鼠标或键盘输入的能力。当对象具有焦点时,可接收用户的输入。在 Windows 操作系统中,任一时刻可运行几个应用程序,但只有具有焦点的应用程序才有活动标题栏,才能接受用户输入。在有几个 TextBox 的 VB 窗体中,只有具有焦点的 TextBox 才显示由键盘输入的文本。

当对象得到或失去焦点时,会产生 GotFocus 或 LostFocus 事件。对象得到焦点时发生 GotFocus 事件;对象失去焦点时发生 LostFocus 事件。窗体和大多数控件支持这两个事件。

使用下列方法之一可以将焦点赋给对象:

- 运行时用鼠标选择对象。
- 运行时用快捷键选择对象。
- 在代码中用 SetFocus 方法。

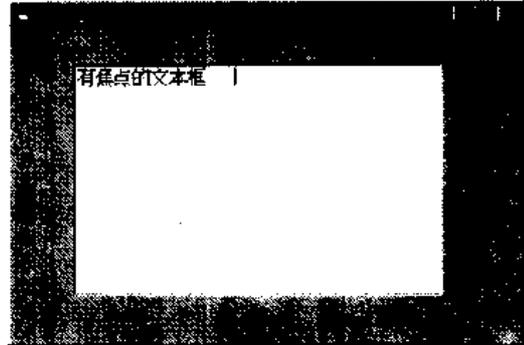


图 6.1 有焦点的文本框

对于大多数可以接收焦点的控件来说,当它们具有焦点时,它们的周围将显示一个虚线框,而当文本框具有焦点时,插入光标将在文本框中闪烁,如图 6.1 所示。

只有当对象的 Enabled 和 Visible 属性为 True 时,它才能接收焦点。Enabled 属性允许对象响应由用户产生的事件,如键盘和鼠标事件。Visible 属性决定了对象在屏幕上是否可见。

注意:所有的 Frame、Label、Menu、Line、Shape、Image 和 Timer 都不能接收焦点。只有不包含任何可接收焦点的控件的窗体,才能接收焦点。

6.2 命令按钮(CommandButton 控件)

CommandButton 控件被用来启动、中断或结束一个进程。单击它时将调用已写入 Click 事件过程中的命令。大多数 VB 应用程序中都有命令按钮,用户可以单击按钮执行操作。单击时,按钮不仅能执行相应的操作,而且看起来就像是被按下和松开一样,因此有时称其为下压按钮。

6.2.1 向窗体添加命令按钮

在应用程序中很可能要使用一个或多个命令按钮。用户可以用 VB 提供的 CommandButton 控件在窗体上添加命令按钮。并可用鼠标调整命令按钮的大小,也可通过设置 Height 和 Width 属性进行调整。

用 Caption 属性改变命令按钮上显示的文本。设计时,可在控件的【属性】窗口中设置此属性。Caption 属性最多包含 255 个字符。若标题超过了命令按钮的宽度,则会折到下一行。但是,如果控件无法容纳其全部长度,则标题会被截尾。

提示:可以通过设置 Font 属性改变在命令按钮上显示的字体。

运行时,可用鼠标或键盘通过下述方法选定命令按钮:

- 用鼠标单击按钮。

- 按 TAB 键,将焦点转移到按钮上,然后按 Spacebar 或 Enter 键选定按钮。
- 按命令按钮的访问键(Alt + 带有下划线的字母)。
- 若命令按钮是窗体的缺省命令按钮,则可按 Enter 键选定按钮;若已把焦点转移到其他控件上,按 Enter 键选定该按钮。
- 若命令按钮是窗体的缺省取消按钮,则可按 Esc 键选定按钮;若已把焦点转移到其他控件上,按 Esc 键取消该按钮。

另外,用户还可通过 Caption 属性创建命令按钮的访问键快捷方式,为此,只需在作为访问键的字母前添加一个连字符 (&)。例如,要为标题“Print”创建访问键,应在字母“P”前添加连字符,于是得到“&Print”。运行时,字母“P”将带下划线,同时按 Alt + P 键就可选定命令按钮。

注意: 如果不创建访问键,而又要使标题中包含连字符,应添加两个连字符 (&&)。这样一来,在标题中就只显示一个连字符而不显示下划线。

6.2.2 命令按钮的属性和事件

下面介绍 CommandButton 控件的主要属性和事件。

1. Default 和 Cancel 属性

在每个窗体上部可选择一个命令按钮作为缺省的命令按钮,也就是说,不管窗体上的哪个控件有焦点,只要用户按 ENTER 键,就已单击此缺省按钮。为了指定一个缺省命令按钮,应将其 Default 属性设置为 True。

也可指定缺省的取消按钮。在把命令按钮的 Cancel 属性设置为 True 后,不管窗体的哪个控件有焦点,按 ESC 键,就已单击此缺省按钮。

2. Value 属性

无论何时选定命令按钮都会将其 Value 属性设置为 True 并触发 Click 事件。若 Value 属性为 False(缺省),则指示未选择按钮。可在代码中用 Value 属性触发命令按钮的 Click 事件。例如:

```
cmdClose.Value = True
```

3. Click 事件

单击命令按钮时将触发按钮的 Click 事件,并调用已写入 Click 事件过程中的代码。单击命令按钮后也将生成 MouseDown 和 MouseUp 事件。如果要在这些相关事件中附加事件过程,则应确保操作不发生冲突。控件不同,这三个事件过程发生的顺序也不同。CommandButton 控件中事件发生的顺序为:MouseDown、Click、MouseUp。

注意: 如果用户试图双击 CommandButton 控件,则其中每次单击都将被分别处理;即 CommandButton 控件不支持双击事件。

6.2.3 带图案的命令按钮

命令按钮像复选框和选项按钮一样,可通过更改 Style 属性设置值后,用 Picture、DownPicture 和 DisabledPicture 属性增强视觉效果。要使用带图案的按钮,可按以下步骤进行:

1. 选中要加载图案的命令按钮,然后在属性窗口中将该控件的 Style 属性设置为 1。

2. 选择按钮的 Picture 属性,单击右边的浏览按钮,打开如图 6.2 所示的【加载图片】对话框,用户可以从 Graphics 子文件夹中选择一个图片放在按钮中。

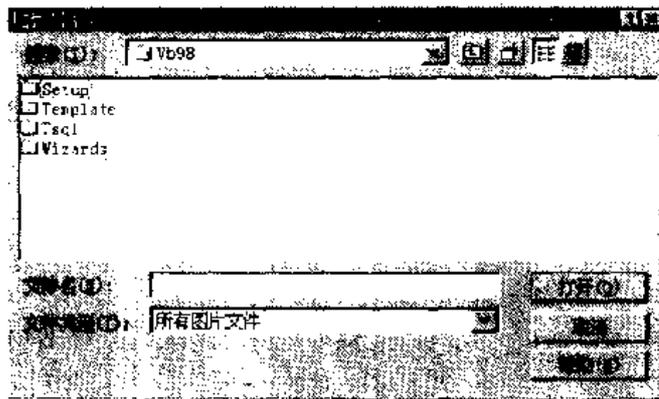


图 6.2 【加载图片】对话框

下面举一个使用 CommandButton 控件的例子。程序使用两个按钮,且有快捷键,单击任一按钮将运行一个游戏——纸牌或挖雷。

以下是程序的代码:

```
Private Sub Command1_Click()  
    Shell "c:\windows\sol.exe", vbNormalFocus  
End Sub  
Private Sub Command2_Click()  
    Shell "c:\windows\winmine.exe", vbNormalFocus  
End Sub
```

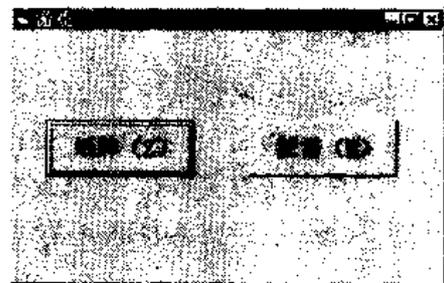


图 6.3 CommandButton 控件示例

图 6.3 显示了程序运行的情况,用户单击相应的按钮就可以调出 Windows 的两个游戏。

6.3 文本框(TextBox 控件)

TextBox 控件用来在运行时显示用户输入的信息,或者在设计或运行时为控件的 Text 属性赋值。TextBox 控件可用于编辑文本,也可将其 Locked 属性设置为 True 使其成为只读的。另外还可用文本框实现多行显示、根据控件的尺寸自动换行以及添加基本格式的功能。

TextBox 控件的一个重要属性是 Text 属性,它包含输入到 TextBox 控件中的文本。缺省时,文本框中输入的字符最多为 2048 个。若将控件的 MultiLine 属性设置为 True,则可输入多达 32K 的文本。

TextBox 中显示的实际文本是受 Text 属性控制的。Text 属性可以用三种方式设置:设计时在【属性】窗口进行,运行时通过代码设置,以及在运行时由用户输入。通过读 Text 属性,能在运行时检索 TextBox 的当前内容。

6.3.1 使用 TextBox 中的文本

利用 TextBox 的 SelStart、SelLength 和 SelText 属性,可以控制 TextBox 的插入点和选择行为。这些属性仅能在运行时使用。

当一个 TextBox 首次得到焦点时,TextBox 缺省的插入点和光标位置在文本的最左边。用户可以用键盘和鼠标移动它们。当 TextBox 失去焦点之后再得到时,插入点位置与用户最后设置的位置一样。在有些情况下,它可能与用户设置不一致。如:在字处理应用程序中,用户会希望新字符出现在已有文本后面;在数据项应用程序中,用户会希望他的输入替换原有条目。使用 SelStart 和 SelLength 属性,用户可以根据需要改变 TextBox 的行为。

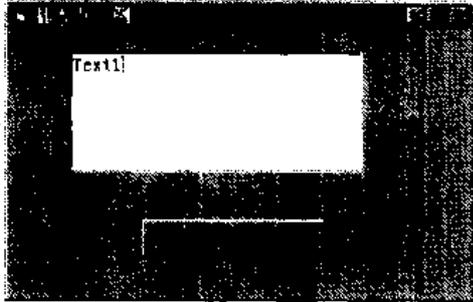


图 6.4 插入点示例

```
Text1.SetFocus  
End Sub
```

SelStart 属性是一个数字,指示文本串内的插入点,其中 0 表示最左边的位置。如果 SelStart 属性值大于或等于文本中的字符数,那么插入点将被放在最后一个字符之后,如图 6.4 所示。

上例的程序代码如下,单击【插入点置后】按钮,插入点将被放在最后一个字符之后:

```
Private Sub Command1_Click()  
Text1.Text = "How are you"  
Text1.SelStart = 15
```

SelLength 属性是一个设置插入点宽度的数值。把 SelLength 设为大于 0 的值,会选中并反白显示从当前插入点开始的 SelLength 个字符。

如果有一段文本被选中,此时用户键入的文字将替换被选中的文本。有些情况下,也可以用粘贴命令使新文本替换原有的文本。SelText 属性是一串文本,可以在运行时给它赋值以替换当前选中的文本。如果没有选中文本,将在当前插入点插入 SelText 文本。

6.3.2 多行 TextBox

TextBox 在缺省情况下只显示单行文本,且不显示滚动条(ScrollBar)。如果文本长度超过可用空间,则只能显示部分文本。通过设置 MultiLine 和 ScrollBars 两种属性(只能在设计程序时设置),可以改变 TextBox 的外观和行为。

注意: 不要把 ScrollBars 属性与 ScrollBar 控件混淆,ScrollBar 控件并不属于 TextBox,它有自己的属性集。

把 MultiLine 属性设为 True,可以使 TextBox 在运行时接受或显示多行文本。只要没有水平方向 ScrollBar,多行 TextBox 中的文本会自动按字换行,自动按字换行省去用户在行尾插入换行符的麻烦。当一行文本已超过所能显示的长度时,TextBox 自动将文本折回到下一行显示。

在设计时,不能在【属性】窗口输入换行符。在过程中,可以通过插入一个回车加上换行符来产生一个行断点。也可以用常量 vbCrLf 插入一个回车与换行符的组合。例如,下面的事件过程是在加载窗体时,把两行文本放入一个多行 TextBox (Text1) 中的示例。

```
Sub Form_Load()  
Text1.Text = "This is the first line"  
& vbCrLf & "This is the second line"  
End Sub
```

注意：当文本超过控件边界时可将 `MultiLine` 属性设置为 `True`，使控件自动换行，并可将 `ScrollBars` 属性设置成添加水平滚动条或垂直滚动条（或者两种都添加），由此即添加了滚动条。但是，如果添加滚动条，那么，由于出现滚动条而使水平编辑区域增大，自动文本换行功能就会失败。

6.3.3 创建密码文本框

密码框是一个文本框，允许在用户输入密码的同时显示星号之类的占位符。VB 提供 `PasswordChar` 和 `MaxLength` 这两个文本框属性，大大简化了密码文本框的创建。

`PasswordChar` 指定显示在文本框中的字符。例如，若希望在密码框中显示星号，则可在【属性】窗口或在程序中将 `PasswordChar` 属性指定为“*”。如图 6.5 所示，无论用户输入什么字符，文本框中都显示星号。

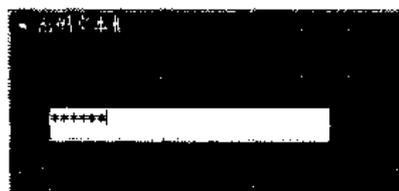


图 6.5 密码文本框示例

程序代码如下：

```
Private Sub Form_Load()  
    Text1.MaxLength = 6  
    Text1.PasswordChar = "*"   
End Sub
```

可用 `MaxLength` 设定输入文本框的字符数。输入的字符数超过 `MaxLength` 后，系统不接受多出的字符并发出嘟嘟声。

6.3.4 创建只读文本框

用户还可用 `Locked` 属性防止编辑文本框内容。将 `Locked` 属性设置为 `True` 后，用户就可滚动文本框中的文本并将其突出显示，但不能作任何变更。将 `Locked` 属性设置为 `True` 后，在文本框中可以使用【复制】命令，但不能使用【剪切】和【粘贴】命令。`Locked` 属性只影响运行时的用户界面。但这时仍可变更 `Text` 属性，方法是在运行时通过代码改变文本框的内容。

另外，引号（" "）有时出现在文本的字符串中，例如：

```
"It's nice to meet you !", said Miss QinQin.
```

因为赋予变量或属性的字符串都用引号（" "）括起来，所以对于字符串中要显示的一对引号，必须再插入一对附加的引号。VB 将并列的两对引号解释为嵌入的引号。

例如，要显示上面的字符串就应使用下述代码：

```
Text1.Text = ""It's nice to meet You !"" , said Miss QinQin."
```

也可用引号的 ASCII 字符（34）达到相同目的：

```
Text1.Text = " Chr(34) &It's nice to meet You ! & Chr(34), said  
Miss QinQin."
```

6.4 标 签 (Label 控件)

`Label` 控件显示的文本用户不能直接修改。在 `Label` 中实际显示的文本是由 `Caption` 属性

控制的,该属性可以在设计时在【属性】窗口中设置,或者在运行时用代码赋值。

在缺省情况下,标题是 Label 控件中唯一的可见部分。但是,如果把 BorderStyle 属性设为 1(也可以在设计时进行),那么 Label 就有了一个边框,这时看起来像一个 TextBox。还可以通过设置 Label 的 BackColor、BackStyle、ForeColor 和 Font 属性,改变 Label 的外观。

使用标签的情况很多,而且目的也不相同。通常用标签来标注本身不具有 Caption 属性的控件。例如,可用 Label 控件为文本框、列表框、组合框等控件添加描述性的标签。也可用它们为窗体添加说明文字,例如向用户提供帮助信息。还可编写代码改变 Label 控件显示的文本内容以响应运行时的事件。例如,若应用程序需要用几分钟处理某个变更,则可用标签显示处理状况消息。因为 Label 控件不接受焦点,所以被用来为其他控件创建显示出来的访问键。

6.4.1 设置标签的标题

为了改变 Label 控件中显示的文本,可使用 Caption 属性。设计时,可从控件的【属性】窗口中设置此属性。Caption 属性的长度最长可设置成 1024 字节。

缺省情况下,当输入到 Caption 属性的文本超过控件宽度时,文本会自动换行,而且在超过控件高度时,超出部分将被裁剪掉。

如果要输入一个较长的或在运行时可能变化的标题,可使用 Label 提供的两种属性: AutoSize 和 WordWrap,帮助用户改变控件尺寸以适应较长或较短的标题。

AutoSize 属性决定控件是否自动改变尺寸以适应其内容。如该属性设为 True,Label 就会根据其内容进行水平方向变化,如图 6.6 所示。

WordWrap 属性使 Label 根据其内容进行垂直方向变化,而保持其宽度不变,如图 6.7 所示。



图 6.6 AutoSize 属性示例



图 6.7 WordWrap 属性示例

注意:要想使 Label 的 WordWrap 属性起作用,就必须设置 AutoSize 为 True。只有在一个单字的宽度超过控件的当前宽度时,Label 的宽度才会增加。

6.4.2 用标签创建访问键

如果要将 Caption 属性中的字符定义成访问键,应将 UseMnemonic 属性设置为 True。定义了 Label 控件的访问键后,用户按 Alt+ 指定的字符,就可将焦点按 Tab 键次序移动到下一个控件。

在作为访问键的字母之前添加一个连字符 (&),就可为其他具有 Caption 属性的控件创建访问键。标签不接受焦点,因此焦点会按照 Tab 键次序自动移动到下一控件处。可用这种技术为文本框、图片框、组合框、列表框、驱动器列表框、目录列表框、网格和图像指定访问键。

将标签指定为控件的访问键,可按以下步骤进行:

1. 首先绘制标签,然后再绘制控件,或者以任意顺序绘制控件,并将标签的 TabIndex 属性设置为控件的 TabIndex 属性减 1。

2. 在标签的 Caption 属性中用连字符为标签指定访问键。

注意:有时可能要在 Label 控件中显示连字符而不是用它们创建访问键。如果在一个记录集中,数据包含连字符,而且要将 Label 控件绑定到记录集的某个字段,就会出现上述情况。为在 Label 控件中显示连字符,应将 UseMnemonic 属性设置为 False。

6.5 选项按钮 (OptionButton 控件)

选项按钮表示给用户一组两个或更多的选择。但是,不同于 CheckBox,选项按钮总是作为组的组成部分工作的;因此,选择一个选项按钮就会立即清除该组中的其他按钮。

6.5.1 创建选项按钮组

要将选项按钮分组,可把它们绘制在不同的容器控件中,像 Frame 控件、PictureBox 控件,或窗体这样的容器控件中。运行时,用户在每个选项组中只能选定一个选项按钮。例如,如果把选项按钮分别添加到窗体和窗体上的一个 Frame 控件中,则相当于创建两组不同的选项按钮。图 3.8 显示了一个具有两个选项按钮组的窗体。



图 6.8 选项按钮组

所有直接添加到窗体的选项按钮成为一组选项按钮。要添加附加按钮组,应把按钮放置在 Frame 或 PictureBox 控件中。要将框架或图片框中的 OptionButton 控件分组,应首先绘制框架或图片框,然后在内部绘制 OptionButton 控件。设计时,可选择在 Frame 控件或 PictureBox 控件中的选项按钮,并把它们作为一个单元来移动。

要选定 Frame 控件、PictureBox 控件或窗体中所包含的多个控件时,可在按住 Ctrl 键的同时用鼠标在这些控件周围绘制一个方框。

在 Frame 中画了一组选项按钮后,用户只能在组内选择一个选项按钮。

在 Frame 中为控件分组:

1. 在【工具箱】中选择“Frame”控件,并在窗体上画出 Frame。
2. 在【工具箱】中选择“OptionButton”控件,并在 Frame 内画出该控件。
3. 如果还想在 Frame 中增加选项按钮,重复步骤 2。

注意:先画 Frame 再在 Frame 上画每个控件,这样移动 Frame 时控件会一起移动。如果将已经存在的控件移到 Frame 上,此控件不会和 Frame 一起移动。

6.5.2 选定或禁止选项按钮

一个选项按钮可以用以下方法选择:

- 在运行期间用鼠标单击选项按钮。
- 用 Tab 键定位到选项按钮组,然后在组内使用方向键(箭头键)定位选项按钮。
- 用代码将它的 Value 属性设置为 True;

```
optChoice.Value = True
```

- 使用在 Label 的标题中指定的快捷键。

要使某个按钮成为选项按钮组中的缺省按钮,只要在设计时将其 Value 属性设置成 True。它保持被选中状态,直到用户选择另一个不同的选项按钮或用代码改变它。

要禁用选项按钮,将其 Enabled 属性设置为 False。程序运行时,若此选项按钮显示模糊,表示无法选取此选项按钮。

6.5.3 选项按钮的事件和属性

选项按钮常用的事件是 Click 事件,常用的属性是 Value 和 Caption 属性。

1. Click 事件

选定选项按钮时将触发其 Click 事件。是否有必要响应此事件,这将取决于应用程序的功能。例如,当希望通过更新 Label 控件的标题向用户提供有关选定项目的信息时,对此事件作出响应是很有益的。

2. Value 属性

选项按钮的 Value 属性指出是否选定了此按钮。选定时,数值将变为 True。可在代码中设置选项按钮的 Value 属性来选定按钮。例如:

```
Option1.Value = True
```

要在选项按钮组中设置缺省选项按钮,可在设计时通过【属性】窗口设置 Value 属性,也可在运行时在代码中用上述语句来设置 Value 属性。

3. Caption 属性

可用 Caption 属性为选项按钮创建访问键快捷方式,这只要在作为访问键的字母前添加一个连字符 (&)。例如,要为选项按钮标题“Dos”创建访问键,应在字母“D”前添加连字符:“&Dos”。运行时,字母“D”将带下划线,同时按 Alt + D 组合键就可选定此选项按钮。

注意:要使标题包含连字符但不创建访问键,就应使标题包含两个连字符 (&&)。这样,标题中将显示一个连字符,而且没有字符带下划线。

6.5.4 选项按钮示例

示例使用选项按钮为一台计算机选择处理器类型和操作系统。当用户在组内选择了选项按钮后,Label 的标题就改变,反映当前的选择。

表 6.3 列出了应用程序中对象属性的设置。

表 6.3 示例程序中对象属性的设置

对象	属性	设置
Label	Name	lblDisplay
	Caption	(空)
CommandButton	Name	cmdClose
	Caption	&Close

续表

对 象	属 性	设 置
第一个 OptionButton	Name	Opt486
	Caption	&486
	Value	False
第二个 OptionButton	Name	Opt586
	Caption	&Pentium
	Value	False
第三个 OptionButton	Name	Opt686
	Caption	P&entium Pro
	Value	False
Frame	Name	fraSystem
	Caption	操作系统
第四个 OptionButton	Name	OptWin95
	Caption	Windows 95
	Value	True
第五个 OptionButton	Name	OptWinNT
	Caption	Windows NT
	Value	False

示例应用程序对事件的响应如下：

- 前三个选项按钮的 Click 事件为窗体级的 string 变量 strComputer 分配一个相应的描述。
- 后两个选项按钮的 Click 事件为第二个窗体级变量 strSystem 分配一个相应的描述。

这个方法的关键就在于使用这两个窗体级变量, strComputer 和 strSystem。这两个变量具有不同的 string 值, 选项按钮的最后选定就依赖于这些值。

每次选择新的选项按钮, 其 Click 事件的代码就更新相应变量的值。

```
Private Sub opt586_Click()
    strComputer = "Pentium"
    Call DisplayCaption
End Sub
```

然后它调用一个名为 DisplayCaption 的过程, 此过程把两个变量连在一起并改变 Label 的 Caption 属性。

```
Sub DisplayCaption()
    lblDisplay.Caption = "You selected a " & _
        strComputer & " running " & strSystem
End Sub
```

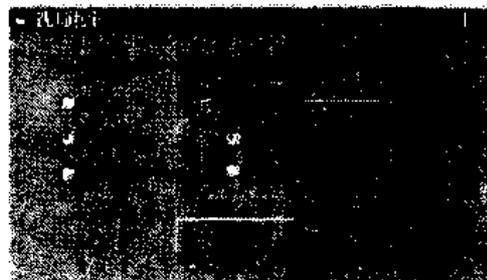


图 6.9 选项按钮用于选择项目

使用子过程是因为改变 Caption 属性的过程对所有五个选项按钮来说基本是一样的, 只不过变量的值因情况而异。这就可以避免在每个 Click 事件中重复同样的代码。

程序运行结果如图 6.9 所示,当用户选择【Pentium】单选框和【Windows 95】单选框时,标签显示“You selected a Pentium running Windows 95”。

6.6 复选框(CheckBox 控件)

CheckBox 表明一个特定的状态是选定 (on) 还是清除 (off)。在应用程序中使用 CheckBox 为用户提供了“True/False”或“yes/no”的选择。因为 CheckBox 彼此独立工作,所以用户可以同时选择任意多个 CheckBox。例如,在图 6.10 中,可同时选定粗体和斜体。

提示: CheckBox 控件与 OptionButton 控件的相同之处在于,每个都是用来指示用户所作的选择。不同之处在于,对于一组 OptionButton,一次只能选定其中的一个,而对 CheckBox 控件,则可选定任意数目的复选框。

6.6.1 复选框的属性和事件

复选框的常用属性和事件分别是 Value 属性和 Click 事件。

1. Value 属性

CheckBox 控件的 Value 属性指示复选框处于选定、未选定或禁止状态(暗淡的)中的哪一种。选定时, Value 设置值为 1。

表 6.4 列出用于设置 Value 属性的数值和相应的 VB 常数。



图 6.10 CheckBox 控件

表 6.4 Value 属性设置

设置值	值	常数
Unchecked	0	vbUnchecked
Checked	1	vbChecked
Unavailable	2	vbGrayed

用户单击 CheckBox 控件指定选定或未选定状态,然后可检测控件状态并根据此信息编写应用程序以执行某些操作。缺省时, CheckBox 控件设置为 vbUnchecked。若要预先在一列复选框中选定若干复选框,则应在 Form_Load 或 Form_Initialize 过程中将 Value 属性设置为 vbChecked。可将 Value 属性设置为 vbGrayed 以禁用复选框。

2. Click 事件

无论何时单击 CheckBox 控件都将触发 Click 事件,然后编写应用程序,根据复选框的状态执行某些操作。在下例中,每次单击 CheckBox 控件时都将改变其 Caption 属性以指示选定或未选定状态。

```
Private Sub Check1_Click()  
If Check1.Value = vbChecked Then  
Check1.Caption = "Checked"  
ElseIf Check1.Value = vbUnchecked Then  
Check1.Caption = "Unchecked"  
End If  
End Sub
```

注意: 如果试图双击 CheckBox 控件,则将双击当作两次单击,而且分别处理每次单击;这就是说,CheckBox 控件不支持双击事件。

6.6.2 CheckBox 示例程序

这个例子使用一个 CheckBox 来决定文本用普通字体还是用斜体字显示。此应用程序有一个 TextBox,一个 CommandButton 和两个 CheckBox。

表 6.5 列出了应用程序中的对象的属性设置。

表 6.5 示例程序中的对象的属性设置

对象	属性	设置
Form	Name	frmCheck
	Caption	复选框示例
Textbox	Name	txtDisplay
	Text	Some sample text
第 1 个 CheckBox	Name	chkBold
	Caption	粗体
第 2 个 CheckBox	Name	chkItalic
	Caption	斜体
CommandButton	Name	cmdClose
	Caption	关闭

选定粗体或斜体时,CheckBox 的 Value 属性值设置为 1,不选定时为 0。缺省 Value 值为 0,所以除非改变 Value 属性值,否则第一次显示时不会选定 CheckBox。可以用常量 vbChecked 和 vbUnchecked 表示数值 1 和 0。

当单击 CheckBox 时,会发生 CheckBox 的 Click 事件。此事件过程测试是否选定 CheckBox(即是否 Value = vbChecked)。如果选定,通过设置 Font 对象的 Bold 或 Italic 属性,将文本转换为粗体或斜体;该 Font 对象是由 TextBox 的 Font 属性返回的。

```
Private Sub chkBold_Click ()
    If ChkBold.Value = vbChecked Then '如果选定。
        txtDisplay.Font.Bold = True
    Else '如果没有选定。
        txtDisplay.Font.Bold = False
    End If
End Sub
```

```
Private Sub chkItalic_Click ()
    If ChkItalic.Value = vbChecked Then '如果选定。
        txtDisplay.Font.Italic = True
    Else '如果没有选定。
        txtDisplay.Font.Italic = False
    End If
End Sub
```

程序运行结果如图 6.11 所示,当用户选中【粗体】复选框时,文本框的文字以粗体显示。

提示: CheckBox 控件像 CommandButton 和 OptionButton 控件一样,可通过更改 Style 属性的设置值后使用 Picture、DownPicture 和 DisabledPicture 属性增强其视觉效果。例如,可以在复选框中添加图标或位图,或者在单击或禁止控件时显示不同的图像。

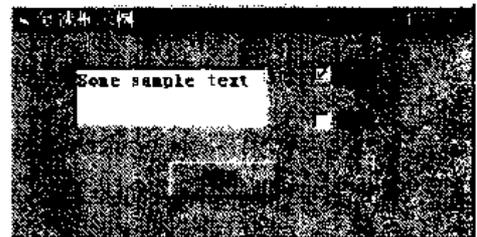


图 6.11 CheckBox 示例

6.7 列表框(ListBox 控件)

列表框为用户提供了选项的列表。虽然也可设置多列列表,但在缺省时将在单列列表中垂直显示选项。如果项目数目超过列表框可显示的数目,控件上将自动出现滚动条。这时用户可在列表中上、下、左、右滚动。

6.7.1 Click 和 Double-Click 事件

对于列表框事件,特别是当列表框作为对话框的一部分出现时,建议添加一个命令按钮,并把该按钮同列表框并用。按钮的 Click 事件过程应该使列表框的选项执行适于应用程序的操作。双击列表中的项目与先选定项目然后单击命令按钮,这两者应该具有相同的效果。为此,应在 ListBox 控件的 DblClick 过程中调用命令按钮的 Click 过程:

```
Private Sub List1_DblClick ()  
    Command1_Click  
End Sub
```

也可将命令按钮的 Value 属性值设置为 True,这就将自动调用事件过程:

```
Private Sub List1_DblClick ()  
    Command1.Value = True  
End Sub
```

6.7.2 向列表增减项目

为了向列表框中添加项目,应使用 AddItem 方法,其语法如下:

控件名.AddItem 项目[,索引值]

其中,【控件名】是列表框的名称。【项目】是添加到列表中的字符串表达式。若【项目】是文字常数,则用引号将它括起来。【索引值】指定在列表中插入新项目的位置。【索引值】为 0 表示第一个位置。若省略【索引值】,则将项目插入在末尾(或按排序次序插入在适当的位置)。

通常在 Form_Load 事件过程中添加列表项目,但也可在任何时候使用 AddItem 方法添加项目,于是可动态(响应用户的操作)添加项目。

下列代码将 Germany、India、France 和 USA 添加到名为 List1 的列表框中:

```
Private Sub Form_Load ()  
    List1.AddItem "Germany"  
    List1.AddItem "India"  
    List1.AddItem "France"  
    List1.AddItem "USA"  
End Sub
```

程序运行结果如图 6.12 所示。

提示:为了在指定位置添加项目,应对新项目指定索引值。例如,下行代码将 "Japan" 插入到第一个位置并把其他项目向下调整:

```
List1.AddItem "Japan", 0
```

注意：指定列表中的第一个位置是 0 而不是 1。

另外,用户可用 RemoveItem 方法从列表框中删除项目。RemoveItem 方法的语法如下:

```
控件名.RemoveItem 项目[,索引值]
```

其中,【控件名】和【索引值】参数与 AddItem 中的参数相同。例如要删除列表中的第一个项目,可添加下行代码:

```
List1.RemoveItem 0
```

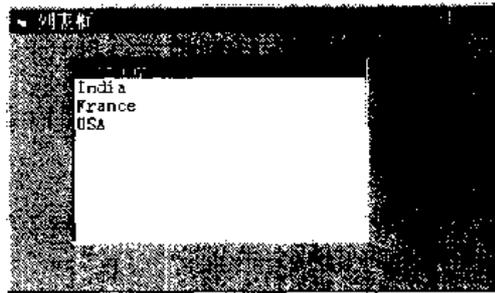


图 6.12 添加列表框项目

6.7.3 获取列表内容

如果用户希望在程序运行时,能够使用列表中的项目,常常要用到 Text、List、ListIndex 和 ListCount 属性。

1. Text 属性

通常,获取当前选定项目值的最简单方法是使用 Text 属性。Text 属性总是对应用户在运行时选定的列表项目。例如,下列代码在用户从列表框中选定“Canada”时显示有关加拿大人口的信息:

```
Private Sub List1_Click ()  
    If List1.Text = "Canada" Then  
        Text1.Text = "Canada has 24 million people."  
    End If  
End Sub
```

Text 属性包含当前在 List1 列表框中选定的项目。代码检查是否选定了 Canada,若已选定,则在 Text 框中显示信息。

2. List 属性

用户还可用 List 属性访问列表的全部项目。此属性包含一个数组,列表的每个项目都是数组的元素。每个项目以字符串形式表示。引用列表的项目时应使用如下语法:

```
控件名.List(索引值)
```

其中,【控件名】参数是列表框的引用,【索引值】是项目的位置。顶端项目的索引为 0,接下来的项目索引为 1,依此类推。例如,下列语句在一个文本框中显示列表的第三个项目(index = 2):

```
Text1.Text = List1.List(2)
```

3. ListIndex 属性

如果要了解列表中已选定项目的位置,则用 ListIndex 属性。此属性只在运行时可用,它设置或返回控件中当前选定项目的索引。设置列表框的 ListIndex 属性也将触发控件的 Click 事件。

如果选定第一个(顶端)项目,则属性的值为 0;如果选定下一个项目,则属性的值为 1,依

此类推。若未选定项目,则 ListIndex 值为 -1。

注意: NewIndex 属性可用于跟踪添加到列表的最后一个项目的索引。在向排序列表插入项目时,这一点十分有用。

4. ListCount 属性

为了返回列表框中项目的数目,应使用 ListCount 属性。例如,下列语句用 ListCount 属性判断列表框中的项目数:

```
Text1.Text = "You have " & List1.ListCount & "entries listed"
```

6.8 组合框(ComboBox 控件)

组合框控件将文本框和列表框的功能结合在一起。有了这个控件,用户可通过在组合框中输入文本来选定项目,也可从列表选定项目。组合框向用户提供了供选择的列表。如果项目数超过了组合框能够显示的项目数,控件上将自动出现滚动条。用户即可上下或左右滚动列表。

通常,组合框适用于建议性的选项列表,而当希望将输入限制在列表之内时,应使用列表框。组合框包含编辑区域,因此可将不在列表中的选项输入到区域中。

此外,组合框节省了窗体的空间。只有单击组合框的向下箭头时(样式 1 的组合框除外,它总是处于下拉状态)才显示全部列表,所以无法容纳列表框的地方可以很容易地容纳组合框。

6.8.1 组合框的样式

有三种组合框样式,每种样式都可在设计时或运行时来设置,而且都是使用数值或相应的 VB 常数来设置组合框的各种样式。

表 6.6 列出了三种样式的值和常数。

表 6.6 组合框三种样式的值和常数

样 式	值	常 数
下拉式组合框	0	vbComboDropDown
简单组合框	1	vbComboSimple
下拉式列表框	2	vbComboDropDownList

1. 下拉式组合框

在缺省设置 (Style = 0) 下,组合框为下拉式。用户可(像在文本框中一样)直接输入文本,也可单击组合框右侧的附带箭头打开选项列表。选定某个选项后,将此选项插入到组合框顶端的文本部分中。当控件获得焦点时,也可按 Alt + 向下方向键打开列表。图 6.13 为一下拉式组合框。

注意: 组合框的第一行中有光标闪烁,标明用户可在此输入新项。

2. 简单组合框

将组合框 Style 属性设置为 1 将指定一个简单的组合框,任何时候都在其内显示列表。为

显示列表中所有项,必须将列表框绘制得足够大。当选项数超过可显示的限度时将自动插入一个垂直滚动条。用户可直接输入文本,也可从列表中选择。像下拉式组合框一样,简单组合框也允许用户输入那些不在列表中的选项。图 6.14 为一简单组合框。

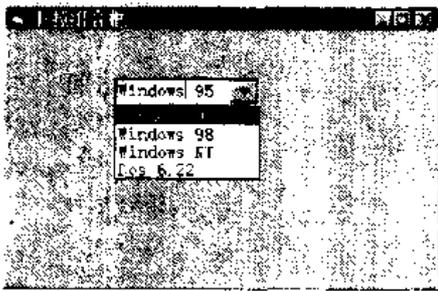


图 6.13 下拉式组合框

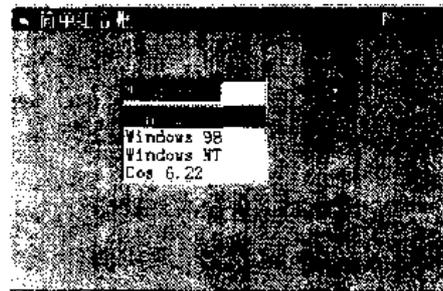


图 6.14 简单组合框

3. 下拉式列表框

下拉式列表框 (Style = 2) 与正规列表框相似——它显示项目的列表,用户必须从中选择。但下拉式列表框与列表框的不同之处在于,除非单击框右侧的箭头,否则不显示列表。这种列表框与下拉式组合框的主要差别在于,用户不能在列表框中输入选项,而只能在列表中选择。当窗体上的空间较少时,可使用这种类型的列表框。图 6.15 为一下拉式列表框。

6.8.2 增减列表中的项目

为在组合框中添加项目,应使用 AddItem 方法,其语法如下:

控件名.AddItem 项目[,索引值]

其中,【控件名】是列表框的名称。【项目】是添加到列表中的字符串表达式。若【项目】是文字常数,则用引号将它括起来。【索引值】指定在列表中插入新项目的位置。【索引值】为 0 表示第一个位置。若省略【索引值】,则将项目插入在末尾(或按排序次序插入在适当的位置)。

通常在 Form_Load 事件过程中添加列表项目,但也可在任何时候使用 AddItem 方法添加项目,于是可动态(响应用户的操作)添加项目。

以下代码将 China, Japan, England, 和 German 放置到名为 Combo1, Style 属性为 0 (vbComboDropDown) 的组合框中:

```
Private Sub Form_Load ()
    Combo1.AddItem "China"
    Combo1.AddItem "Japan"
    Combo1.AddItem "England"
    Combo1.AddItem "German"
End Sub
```

运行时,只要加载窗体,而且用户单击向下箭头,则将显示如图 6.16 所示的列表。

提示: 为了在指定位置添加项目,应对新项目指定索引值。例如,下行代码将“Canada”插入到第一个位置并把其他项目向下调整: Combo1.AddItem "Canada", 0

注意: 是 0 而不是 1 指定列表中的第一个位置。

另外,用户可用 RemoveItem 方法从组合框中删除项目。RemoveItem 方法的语法如下:

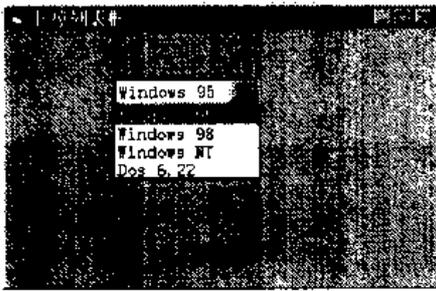


图 6.15 下拉式列表框

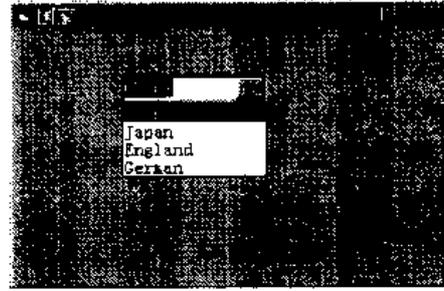


图 6.16 【国家】组合框

控件名.RemoveItem 项目[,索引值]

其中,【控件名】和【索引值】参数与 AddItem 中的参数相同。例如要删除列表中的第一个项目,可添加下行代码:

```
Combo1.RemoveItem 0
```

为了在组合框中删除所有列表项目,应使用 Clear 方法:

```
Combo1.Clear
```

6.8.3 获取列表内容

如果用户希望在程序运行时,能够使用列表中的项目,常常要用到 Text、List、ListIndex 和 ListCount 属性。

1. Text 属性

获取当前选定项目值的最简单的常用方法就是使用 Text 属性。在运行时无论向控件的文本框部分输入了什么文本,Text 属性都与这个文本相对应。这可以是选定的列表选项,或者是用户在文本框中输入的字符串。

例如,如果用户选定列表框中的 Chardonnay,则通过下列代码显示有关 Chardonnay 的信息:

```
Private Sub Combo1_Click ()  
    If Combo1.Text = "Chardonnay" Then  
        Text1.Text = "Chardonnay is a medium-bodied  
                    white wine."  
    End If  
End Sub
```

2. List 属性

有了 List 属性就可访问列表中所有项目。该属性包含一个数组,而且列表中的每个项目都是数组的元素。每一项都表示为字符串的形式。为了引用列表中的项目,应使用如下语法:

[控件名].List(索引值)

【控件名】参数引用组合框,而【索引值】是项目的位置。顶端项目的索引为 0,下一个项目的索引为 1,依次类推。例如,在文本框中,以下语句显示列表中的第三个项目(索引值 = 2):

```
Text1.Text = Combol.List(2)
```

3. ListIndex 属性

欲知组合框列表中选定项目位置,请使用 ListIndex 属性。该属性设置或返回控件中当前选定项目的索引值,而且只在运行时有效。对组合框的 ListIndex 属性进行设置也会触发控件的 Click 事件。若选定第一个(顶端)项目,则属性值为 0,选定的下一个项目属性值为 1,依次类推。如果未选定项目,或者用户在组合框中输入选项(样式 0 或 1)而不在列表中选择现有项目,则 ListIndex 为 -1。

注意: NewIndex 属性可用于跟踪列表中最后添加的项目的索引。向排序列表插入项目时,这一点很有用。

4. ListCount 属性

为了返回组合框中的项目数,应使用 ListCount 属性。例如,下列语句用 ListCount 属性判断组合框中的项目数:

```
Text1.Text = "You have " & Combol.ListCount & " entries listed"
```

6.9 滚 动 条

有了滚动条,就可在应用程序或控件中水平或垂直滚动,相当方便地巡视一长列项目或大量信息。滚动条是 Windows 95、Windows 98 和 Windows NT 界面上的共同元素。

在 VB 6 的工具箱里,有两个滚动条可供选择,它们是水平滚动条和垂直滚动条。水平、垂直滚动条控件不同于 Windows 中内部的滚动条或 VB 中那些附加在文本框、列表框、组合框或 MDI 窗体上的滚动条。无论何时,只要应用程序或控件所包含的信息超过当前窗口(或者在 ScrollBars 属性被设置成 True 时的文本框和 MDI 窗体)所能显示的信息,那些滚动条就会自动出现。

在较早的 VB 版本中,通常用滚动条作为输入设备。但目前的 Windows 界面指南则建议用滑块取代滚动条作为输入设备。VB 6.0 专业版和企业版都包括 Windows 95 / 98 的滑块控件。

但滚动条在 VB 中仍然有价值,因为它为那些不能自动支持滚动的应用程序和控件提供了滚动功能。

6.9.1 滚动条的主要事件和属性

使用滚动条控件时,首先应了解它的常用事件和属性。

1. 滚动条控件的主要事件

滚动条控件的主要事件有 Change 和 Scroll 事件。在释放滚动滑块或单击滚动条或滚动箭头时,Change 事件就会发生。Scroll 事件在移动滚动滑块时发生,在单击滚动箭头或滚动条时不发生。用户可用 Scroll 事件访问滚动条被拖动后的数值。

2. 滚动条控件的主要属性

滚动条控件的主要属性是 Value、LargeChange 和 SmallChange 属性。

Value 属性(缺省值为 0)是一个整数,它对应于滚动滑块在滚动条中的位置。当滚动滑块

位置在最小值时,它将移动到滚动条的最左端位置(水平滚动条)或顶端位置(垂直滚动条)。当滚动滑块在最大值时,它将移动到滚动条的最右端或底端位置。同样,滚动滑块取中间数值时将位于滚动条的中间位置。

除了可用鼠标单击改变滚动条数值外,也可将滚动滑块沿滚动条拖动到任意位置。结果取决于滚动滑块的位置,但总是在用户所设置的 Min 和 Max 属性之间。

注意: 如果希望滚动条显示的信息从较大数值向较小数值变化,可将 Min 设置成大于 Max 的值。

为了指定滚动条中的移动量,对于单击滚动条的情况可用 LargeChange 属性;对于单击滚动条两端箭头的情况可用 SmallChange 属性。滚动条的 Value 属性增加或减少的长度是由 LargeChange 和 SmallChange 属性设置的数值决定的。要设置滚动滑块在运行时的位置,可将 Value 属性设为 0 到 32,767 中的某个数值(包括 0 和 32,767)。

6.9.2 滚动条控件示例

下面举一个例子,说明如何使用滚动条控件,用户可以通过三个滚动条来调整一个文本框的底色。

在这个程序中,我们将窗口的 Caption 属性设置为“调整颜色”,然后在窗口上添加了三个水平滚动条,分别用来调整红绿蓝三种颜色,他们的各项属性设置如表 6.7 所示。使用一个文本框作为颜色区,它的名称是 txtColor。为了增强一些美感,还特意在文本框外又添加了一个清除了 Caption 属性 Frame 控件,作为颜色区外的边框。

表 6.7 HScrollBar 控件的属性设置值

名称	Min	Max	Smallchange	Largechange
Hsbred	0	255	8	32
Hsbgreen	0	255	8	32
HsbBlue	0	255	8	32

因为滚动条只能从视觉上给人一个相对数量的概念,为了标识滚动条的功能,同时让用户了解到红绿蓝 3 种颜色的具体值,我们还添加了 4 个 Label 控件,它们的名称分别是 lblred, lblGreen, lblBlue 和 lblColor。程序执行时的样子如图 6.17 所示。

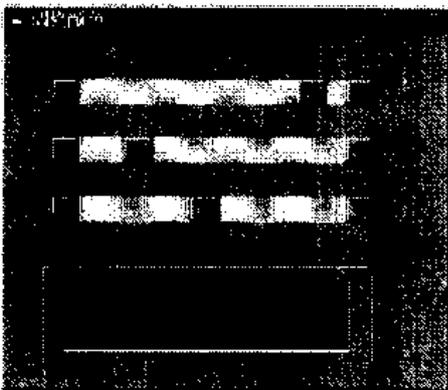


图 6.17 滚动条示例

以下是程序的源代码(根据 3 个滚动条的当前值,利用 RGB 函数设置文本框的背景颜色):

```
Private Sub ChangeColor()
    TxtColor.BackColor = RGB(hsbRed.Value, hsbGreen.Value,
    hsbBlue.Value)
End Sub

Private Sub Form_Load()
    ChangeColor
    LblRed.Caption = "红色值:" & hsbRed.Value
    LblGreen.Caption = "绿色值:" & hsbGreen.Value
```

```

    LblBlue.Caption = "蓝色值:" & hsbBlue.Value
End Sub

Private Sub hsbBlue_Change()
    LblBlue.Caption = "蓝色值:" & hsbBlue.Value
    ChangeColor
End Sub

Private Sub hsbGreen_Change()
    LblGreen.Caption = "绿色值:" & hsbGreen.Value
    ChangeColor
End Sub

Private Sub hsbRed_Change()
    LblRed.Caption = "红色值:" & hsbRed.Value
    ChangeColor
End Sub

```

6.10 ActiveX 控件简介

ActiveX 控件是 VB 工具箱的扩充部分,使用 ActiveX 控件的方法与使用其他标准内装的控件,如 CommandButton 控件,完全一样。在程序中加入 ActiveX 控件后,它将成为开发和运行环境的一部分,并为应用程序提供新的功能。

ActiveX 控件保留了一些熟悉的属性、事件和方法,如 Name 属性。而且 ActiveX 控件特有的方法和属性大大地增强了 VB 用户的能力和灵活性。例如,VB 专业版和企业版包括了 Windows 公共控件,用户可以使用它们创建具有 Windows 95/98 面貌和风格的工具栏、状态栏以及选项卡、对话框的应用程序。此外,还有一些 ActiveX 控件可以用来创建具有 Internet 功能的应用程序。

用户如果要使用 ActiveX 控件,首先应该将它们添加到工具箱中,在工程的工具箱中加入 ActiveX 控件的方法为:

1. 在【工程】菜单中,选择【部件】项,或者在工具箱中单击鼠标右键,然后从快捷菜单中选择【部件】项(如图 6.18 所示),以显示如图 6.19 所示的【部件】对话框。在该对话框中列出了所有已注册的可加入对象、设计器和 ActiveX 控件。

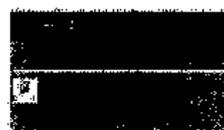


图 6.18 工具箱快捷菜单

2. 选中要加入工具箱的 ActiveX 控件名称前的复选框。
3. 单击【确定】按钮,以关闭【部件】对话框。所有选定的 ActiveX 控件将出现在工具箱中。

如果要将其他的 ActiveX 控件加入【部件】对话框,可以单击【部件】对话框的【浏览】按钮,并找到扩展名为 .ocx 的文件。在将 ActiveX 控件加入可用控件列表时,VB 自动在【部件】对话框中选中它的复选框。

如果要从工程中删除控件,则应如下操作:

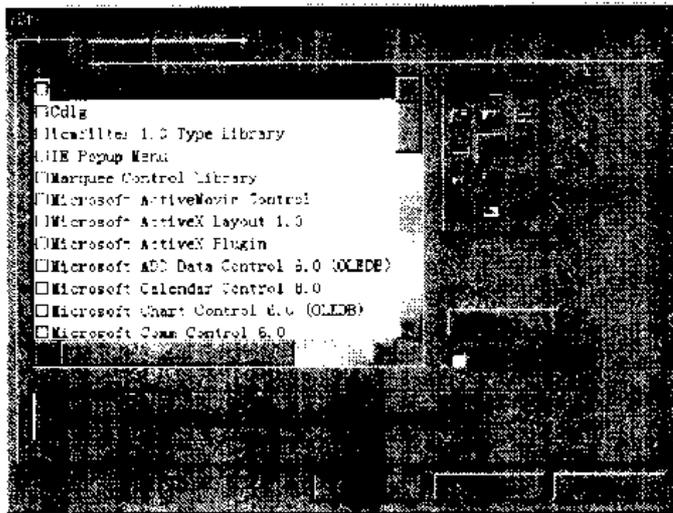


图 6.19 【部件】对话框

1. 在【工程】菜单中,选择【部件】命令,或者在工具箱中单击鼠标右键,然后从快捷菜单中选择【部件】命令,以显示【部件】对话框。在该对话框中列出了所有已注册的可加入对象、设计器和 ActiveX 控件。

2. 清除要在工具箱中删除的 ActiveX 控件名称前的复选框。

3. 单击【确定】按钮,以关闭【部件】对话框。所有选定的 ActiveX 控件将从工具箱中消失。

注意:若某个控件的一个实例被这个工程的任何窗体所用,则不能从工具箱中删掉这个控件。

用户可以看到,在【部件】对话框的控件列表中,有一项为“Microsoft Common Dialog Control 6.0”,它就是 Vb 6 的通用对话框控件。在本书的第七章将详细介绍该控件的使用。

6.11 本章小结

虽然 VB 中可用的控件很多,但是经常使用的却是其中的几个,本章主要介绍了最常用的一些控件,它们是:

- CommandButton 控件
- TextBox 控件
- Label 控件
- OptionButton 控件
- CheckBox 控件
- ListBox 控件
- ComboBox 控件
- HscrollBar 和 VscrollBar 控件

此外,还介绍了如何向自己的工具箱中添加 ActiveX 控件。同时,在本章开始,介绍了一个 Windows 中常用到概念——焦点。

第7章 对话框

Windows 程序的一个突出优点是实现了人机对话,而大多数的人机对话任务都在对话框中完成。例如当正在使用的程序出了问题,屏幕上会弹出一个对话框,告诉用户相关信息。VB 6 允许用户用多种方法创建不同的对话框,包含用函数生成消息框和输入框,用通用对话框控件生成通用对话框以及用户自己定制对话框。

7.1 使用函数生成对话框

使用 VB 函数,能够建立两种预制对话框:消息框和输入框。消息框和输入框的实现都只需利用系统提供的两个函数而不必为对话框另建窗体。

7.1.1 消息框

在应用程序中,可能会需要显示一些暂时性的简短的错误或者警告信息,以引起用户的注意。用户可以自己设计一个窗体来完成这个任务。但如果使用 MsgBox 函数来生成消息框,会显得更为直接和方便。

MsgBox 函数让用户在一个简单的对话框中显示消息,这个对话框可包括预定义的按钮和项目、用户要指定对话框中所使用的消息、标题、按钮和项目。图 7.21 就是由 MsgBox 函数产生的一个典型的对话框。



图 7.1 MsgBox 函数产生的消息框

该对话框由三部分组成,分别为:

- 标题:显示在对话框最顶端的信息即标题栏文本。
- 提示信息:中间显示的即为对话框的提示内容。
- 按钮:对话框下部为供用户选择的按钮。

MsgBox 函数可以用来在对话框中显示消息并等待用户单击按钮,然后返回一个整型的值,让程序了解用户单击的是哪一个按钮。

MsgBox 函数的语法是这样的:

```
MsgBox(Prompt[, buttons[, title], helpfile, context])
```

其中各个参数的含义如下:

- Prompt

这个参数是必要的,为一字符串表达式,作为显示在对话框中的消息。Prompt 的最大长度大约为 1024 个英文字符,所能显示的字符数由所用字符的宽度决定。如果要显示的 Prompt 内容超过一行,则可以在每一行之间用回车符或者换行符,但是注意不能在字符串之中直接敲 Enter 键,而是要使用 Chr 函数。Chr 函数可以返回与指定的字符代码相关的字符,回车符用 Chr(10)表示,换行符用 Chr(13)表示。

注意:回车符或者换行符与字符串之间要用“+”或者“&”来连接。

• buttons

这个参数是可选的。为数值表达式值的总和,指定显示按钮的数目及形式、使用的图标样式、默认按钮是什么以及消息框的强制回应等。如果省略该参数,则 VB 默认 Buttons 的值为 0, buttons 参数的设置值请见表 7.1。

• title

这个参数是可选的。为在对话框标题栏中显示的字符串表达式,如果省略 title 参数则 VB 将应用程序名放在标题栏中。

• helpfile

这个参数是可选的。为一字符串表达式,识别用来向对话框提供上下文相关帮助的帮助文件,如果提供了 helpfile 参数,则也必须提供 Context 参数。

• Context

这个参数也是可选的。为一数值表达式,由帮助文件的作者指定给适当的帮助主题的帮助上下文编号。同样,如果提供了 Context 参数,则也必须提供 helpfile 参数,二者要同时提供。

提示:当 helpfile 和 Context 两个参数都被设置时,用户可通过按 F1 键来查看同帮助目录相关的帮助主题。

表 7.1 列出了 buttons 参数的允许设置值。

表 7.1 buttons 参数的设置值

符号常量	值	描述
vbOkonly	0	只显示 OK 按钮
vbOkCancel	1	显示 OK 及 Cancel 按钮
vbAbortRetryIgnore	2	显示 Abort、Retry 及 Ignore 按钮
vbYesNoCancel	3	显示 Yes、No 及 Cancel 按钮
vbYesNo	4	显示 Yes 及 No 按钮
vbRetryCancel	5	显示 Retry 及 Cancel 按钮
vbCritical	16	显示 Critical Message 图标
vbQuestion	32	显示 Warning Query 图标
vbExclamation	48	显示 Warning Message 图标
vbInformation	64	显示 Information Message 图标
vbDefaultButton1	0	第一个按钮是默认值
vbDefaultButton2	256	第二个按钮是默认值
vbDefaultButton3	512	第三个按钮是默认值
vbDefaultButton4	768	第四个按钮是默认值
vbApplicationModal	0	应用模态对话框,用户必须在当前应用程序上工作之前对消息框作出反应。
vbSystemModal	4096	系统模态对话框,所有应用程序都被挂起,直到用户对消息框作出响应才继续工作

第一组值(0~5)描述了对话框中显示的按钮的数目及类型;第二组值(16,32,48,64)描述了对话框中显示的图标的样式;第三组值(0,256,512,768)说明了哪一个按钮是默认按钮;第

四组值(0,4096)则决定消息框的模式。用户可以将这些数值相加以生成一个组合的 Buttons 参数值,在相加的时候,每组值中只能取一个数字。

注意:编写程序代码时,用户最好使用符号常量名称,而不使用实际整数值,这样可以使程序易读,容易理解。

除了显示信息,用户还可以通过 MsgBox 函数的返回值来确定用户单击了对话框中的哪一个按钮,以作出不同的处理。表 7.2 列出了 MsgBox 函数的返回值及其含义。

表 7.2 MsgBox 函数的返回值

符号常量	值	用户单击的按钮
vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
vbYes	6	Yes
vbNo	7	No

如果对话框显示 Cancel 按钮,则按 Esc 键与单击 Cancel 按钮的效果相同,但是,若按 Esc 键,则直到其他按钮中有一个被单击之前,都不会返回任何值。

下面举一个使用 MsgBox 函数的例子:在具有【是】、【否】以及【取消】按钮的对话框中显示一首歌曲中的几句词,然后询问用户是否喜欢这首歌。示例中的默认按钮为【是】,程序还根据 MsgBox 函数不同的返回值作出不同的反应:

```
Option Explicit
Private Sub Form1_Click( )
Dim msg1 As String
Dim style As Integer
Dim title As String
Dim response As Integer
Dim msg2 As String
Msg1 = "谁能用爱烘干我这颗潮湿的心" + Chr(13) + Chr(13) + _
"给我一声问候,一点温情。" + Chr(13) + Chr(13) + _
"谁能用心感受我这份滴水的痴情," + Chr(13) + Chr(13) + _
"给我一片晴空,一声叮咛。" + Chr(13) + Chr(13) + Chr(13) + Chr(13) + _
"你喜欢这首歌吗?"
style = vbApplicationModal + vbDefaultButton1 + _
vbInformation + vbYesNoCancel
title = "歌一首"
response = MsgBox(msg1, style, title)
Select Case response
Case vbYes
msg2 = "你我想法一致。"
```

```

Case vbNo
msg2 = "非常遗憾。"
Case vbCancel
msg2 = "不想回答是吗?"
End Select
MsgBox msg2, "询问结果:"
End Sub

```

在程序运行后,用户只要在窗体上单击,就会显示一个消息框,如图 7.2 所示。

单击【是】、【否】、【取消】按钮后,显示另一消息框,比如单击【是】按钮,则显示如图 7.3 所示的消息框,消息框显示询问结果。

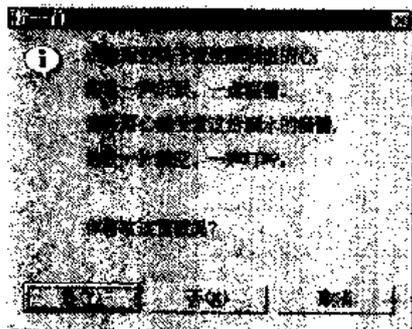


图 7.2 示例运行结果



图 7.3 单击【是】按钮后显示的消息框

7.1.2 输入框

输入框与消息框有些不同,它是在对话框中显示提示,等待用户输入正文或按下按钮,并返回包含文本框内容的字符串。VB 中提供 InputBox 函数为用户建立输入框。

InputBox 函数的语法为:

```
InputBox(prompt[, title][, default][, xpos][, ypos][, helpfile context])
```

其中各个命名参数的含义如下:

- **prompt**

这个参数是必要的,为对话框消息出现的字符串表达式。prompt 的最大长度大约是 1024 个字符,由所用字符的宽度决定。如果要显示的 prompt 内容超过一行,则可以在各行之间用回车符(Chr(13)、换行符(Chr(10))或回车换行符的组合(Chr(13)&Chr(10))来分隔。

- **title**

这个参数是可选的,显示对话框标题栏中的字符串表达式,如果省略参数 title,则把应用程序名放入标题栏中。

- **default**

这个参数是可选的,显示文本框中的字符串表达式,在没有其他输入时作为默认值。如果省略 default 参数,则文本框初始化为空。

- **xpos**

这个参数是可选的,为一数值表达式,与 ypos 参数成对出现,用于指定对话框的左边与屏幕左边的水平距离。如果省略参数 xpos,则对话框显示在水平方向居中。

- ypos

这个参数是可选的,为一数值表达式,与 xpos 参数成对出现,用于指定对话框的上边与屏幕上边的垂直距离。如果省略参数 ypos,则对话框显示在屏幕垂直方向距下边大约 1/3 的位置。

- helpfile

这个参数是可选的,为一字符串表达式,用来识别帮助文件,并用该文件为对话框提供上下文相关的帮助。如果已提供了 helpfile,则也必须提供 context。

- context

这个参数也是可选的,为一数值表达式,由帮助文件的作者指定给某个帮助主题的帮助下文编号。如果已提供 context,则也必须提供 helpfile。

如果用户单击【确定】按钮或按下 Enter 键,则 InputBox 函数返回文本框的内容。如果用户单击【取消】按钮,则 InputBox 函数返回一个长度为 0 的字符串(“”)。

下面举一个使用 InputBox 函数的例子,提示用户输入密码,输入框的标题为“输入密码”,显示的信息为“请输入程序密码:”。如果用户单击【确定】按钮或按下 Enter 键,则变量 strPassword 会保存用户输入的密码,如果用户单击【取消】按钮,则返回一零长度字符串。

```
Option Explicit
Private Sub Form1_Click()
Dim strPassword As String
strPassword = InputBox("请输入程序密码:", "输入密码")
End Sub
```

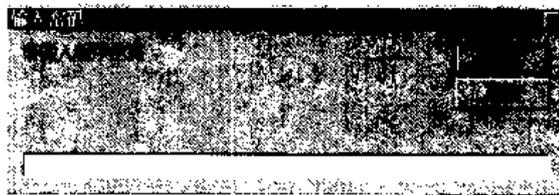


图 7.4 用 InputBox 函数生成的输入框

在程序运行后,用户只要在窗体上单击,就会显示一个输入框,如图 7.4 所示。

7.2 通用对话框

VB 6 提供了通用对话框控件(CommonDialog),通用对话框控件提供了一组标准的操作对话框,可以进行诸如打开和保存文件、选择颜色、选择字体和设置打印选项等操作。此外,通用对话框还能启动帮助系统。在默认的工具箱中,没有显示通用对话框控件(CommonDialog),要使工具箱中显示出通用对话框控件可按以下方法进行。

1. 在工具箱的任一位置右击,系统弹出快捷菜单,选择其中的【部件】选项;或者在【工程】菜单中选择【部件】选项;或者按 Ctrl + T 组合键,系统打开【部件】对话框。

2. 单击【部件】对话框中的【控件】选项卡,在列表中显示了所有已经注册的 ActiveX 控件,从中选择“Microsoft Common Dialog Control 6.0”复选框,如图 7.5 所示。

3. 单击【确定】按钮以关闭【部件】对话框,这时在工具箱中就出现了一个控件按钮 CommonDialog。这就是 VB 6 的通用对话框控件。

7.2.1 通用对话框概念

在程序中要使用 CommonDialog 控件,可将其添加到窗体中并设置其属性,控件所显示的对话框由控件的方法决定。在设计时,CommonDialog 控件是以图标的形式显示在窗体中;在运行时,当相应的方法被调用时,将显示一个对话框或启动帮助系统。

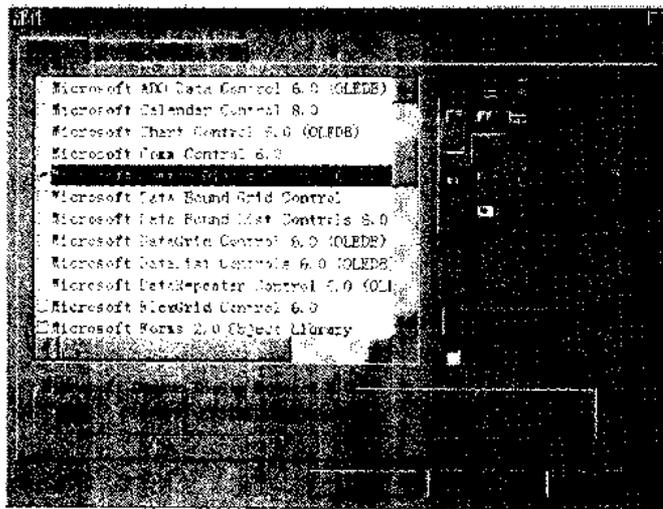


图 7.5 利用【部件】对话框添加 CommonDialog 控件

注意：CommonDialog 控件的大小不能改变，用户也无法指定对话框在屏幕上显示的位置。

在 VB 6.0 中，通用对话框的方法和属性见表 7.3 和表 7.4。

表 7.3 通用对话框的方法列表

方法	功能
ShowOpen	显示打开对话框
ShowSave	显示另存为对话框
ShowColor	显示颜色对话框
ShowFont	显示字体对话框
ShowPrinter	显示打印对话框
ShowHelp	显示 Windows 帮助对话框

表 7.4 通用对话框的属性

属性	描述
CancelError	确定一个错误是否由取消操作引起
Color	读取或设置选中对象的颜色
Copies	读取或设置打印份数
DefaultExt	设置对话框的缺省扩展名
DialogTitle	设置对话框中标题文本
FileName	设置打开或存取的文件的名字和路径
Filter	设置在打开文件或另存为对话框中的文件时使用的过滤器
FilterIndex	设置缺省的文件过滤器
Flags	设置每个对话框的选项
FontBold	设置对象字体为黑体
FontItalic	设置对象字体为斜体
FontStrikethru	设置对象字体有横线划过
FontUnderline	设置对象字体带下划线

属 性	描 述
FontName	读取或设置对象的字体名
FontSize	读取或设置对象字体的尺寸
FromPage, ToPage	设置打印页范围
HelpCommand	设置所要帮助的类型
HelpContext	设置上下文相关帮助的编号
HelpFile	设置要显示的帮助文件的名字
HelpKey	设置帮助文件中进行搜索的键值
Index	唯一标识控件数组中的一个元素
InitDir	设置启动文件路径
Left, Top	设置通用对话框控件在窗体上的位置
Max, Min	设置显示的最大和最小字体
MaxFileSize	设置由属性 FileName 给出的文件名的最大长度
Orientation	设置打印纸走纸方向
PrinterDefault	确定打印对话框的改变是否改变缺省打印设置
Tag	读取或设置与控件相关的其他字符串数据
Name	读取或设置在代码中引用对象的名字

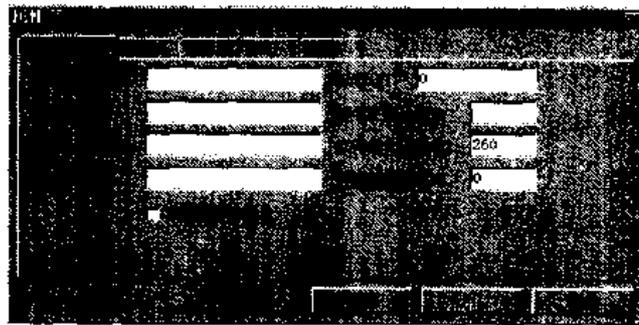


图 7.6 【属性页】对话框

使用通用对话框能够保证用户的应用程序和其他 Windows 程序有一致接口,而不需要用户增加任何程序代码就能够在用户程序中增加功能。每种对话框都有不同的属性设置,这可以在属性窗口中设置,也可以在【属性页】对话框中完成。打开【属性页】对话框的方法为:

1. 选中通用对话框控件。
2. 单击鼠标右键,在弹出的快捷菜单中选择【属性】命令,或者选择属性窗口中的【(自定义)]项,再单击右边的省略号“...”按钮。即可弹出如图 7.6 所示的通用对话框控件的【属性页】对话框。

对话框中各项属性的含义已在表 7.4 中给出,用户可以看到不同类型的对话框需要设置的属性并不相同。

7.2.2 打开和另存为对话框

使用【打开】对话框可以指定驱动器、文件夹、文件扩展名和文件名,如图 7.7 所示。【另存为】对话框在外观上与【打开】对话框基本相同,如图 7.8 所示。

要显示【打开】或者【另存为】对话框,只需要一行语句:

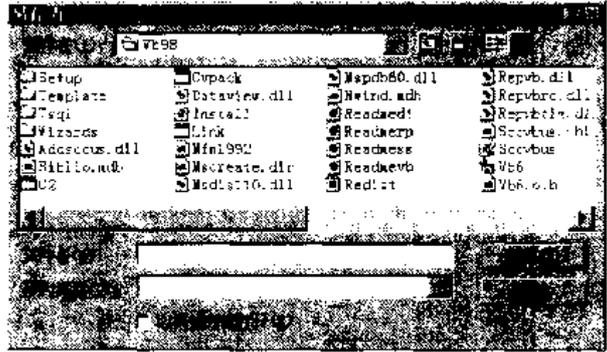
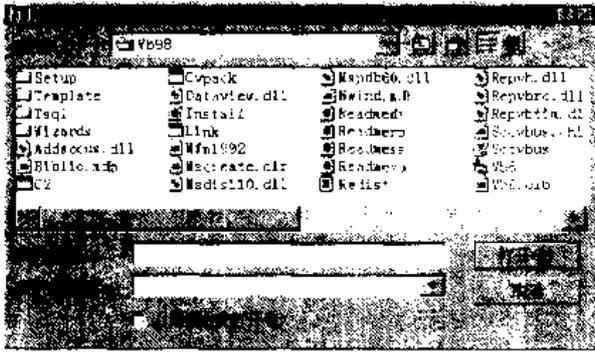


图 7.7 使用通用对话框控件生成的【打开】对话框 图 7.8 使用通用对话框控件,生成的【另存为】对话框

`CommonDialog1.ShowOpen` '显示【打开】对话框

或:

`CommonDialog1.ShowSave` '显示【另存为】对话框

其中;`CommonDialog1` 为通用对话框控件的 `Name` 属性。

当用户选定文件并关闭对话框后,可以从控件的 `FileName` 属性获取选定的文件名。用户还可以通过设置 `Filter` 属性来指定在【文件类型】列表框中显示的文件过滤器列表,其格式如下:

`description1|filter1|description2|filter2...`

其中,`Description` 是列表框中显示的字符串,例如“Executable Files(.exe)”;而 `Filter` 是实际的文件过滤器,例如,“.exe”。每个 `description|filter` 设置间必须用符号“|”隔开。至于要选择哪一种文件类型,则必须通过设置控件的 `FilterIndex` 属性来指定。

下面举一个使用 `Show` 方法显示【打开】对话框的例子,并以选定的文件名为打开文件过程的参数。示例在窗体上放一个 `CommandButton` 控件,其 `Name` 属性为 `OpenFile`,`Caption` 属性为“打开文件”,窗体 `Caption` 属性为“打开文件示例”,窗体上还有一通用对话框控件如图 7.9 所示。

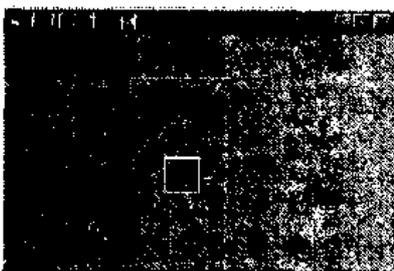


图 7.9 示例使用的窗体

程序代码为:

```
Option Explicit
Private Sub OpenFile_Click()
On Error GoTo ErrHandler 'CancelError 为 True
'设置文件过滤器
CommonDialog1.Filter = "All Files(*.*)|*.*| " _
&"Executable Files(*.exe)|*.exe| " _
&"Text Files(*.txt)|*.txt|"
CommonDialog1.FilterIndex = 2 '指定默认文件类型
```

`CommonDialog1.ShowOpen` '显示【打开】对话框

`MyOpenFileSub` '调用用户自己的打开文件过程

`ErrHandler:`

`Exit Sub` '用户按【取消】按钮

`End Sub`

注意：设置文件过滤器时，符号“|”前后不能包含空格，否则将得不到想要的文件类型列表。

提示：对所有公共对话框，当 CancelError 属性为 True，而且用户单击了对话框中的【取消】按钮时，将生成一个错误，在显示对话框时捕获错误，以此检测是否单击了【取消】按钮。

程序运行后，单击窗体上的【打开文件】按钮，显示如图 7.9 所示的对话框，用户可从文件类型列表中看到文件过滤器的结果。

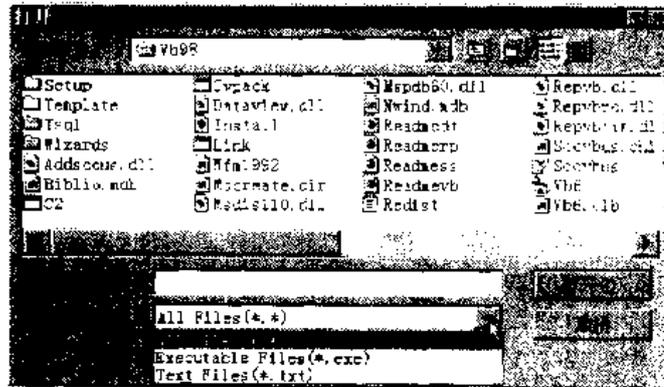


图 7.10 对【打开】对话框中显示的文件进行过滤

7.2.3 字体对话框

字体对话框可以让用户根据需要从中选择字体的大小、样式等等。用户一旦在【字体】对话框中选定字体后，有关用户选项的信息就包含在通用对话框控件的属性中，用户也可以在【属性页】对话框中的【字体】选项卡中设置字体属性。如图 7.11 所示。

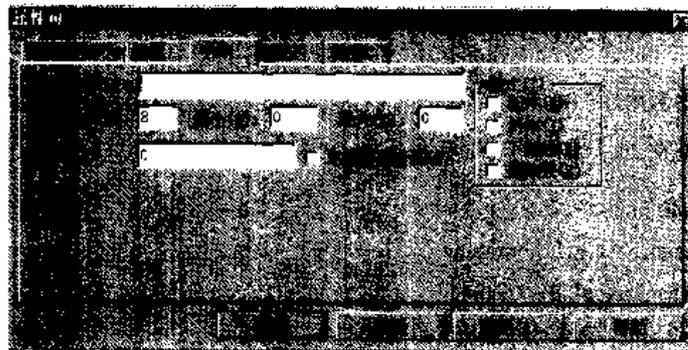


图 7.11 【属性页】对话框的【字体】选项卡

使用 CommonDialog 控件显示【字体】对话框的第一步，就是要将 Flags 属性设置为下表所列的三个 VB 常量为—。

表 7.5 Flags 属性值列表

字体集	常量	值
屏幕字体	cdlCFScreenFonts	1
打印字体	cdlCFPrinterFonts	2
以上两种字体	cdlCFBoth	3

注意：在显示【字体】对话框前如果没有为 Flags 属性设置合法的数值，在程序运行时将出现没有安装字体的错误信息，如图 7.12 所示。

使用属性窗口可以设置 Flags 属性值，也可以用赋值语句在程序代码中设置 Flags 属性值，在设置好 Flags 属性后，就可以使用 ShowFont 方法显示字体对话框，例如：



图 7.12 没有设置 Flags 时出现的错误消息

```
Option Explicit
Private Sub Form1_Click()
    CommonDialog1.Flags = cdlCFBoth Or cdlCFEffects
    CommonDialog1.ShowFont
End Sub
```

其中：常量 cdlCFEffects 使用户能在【字体】对话框中进行效果设置。程序运行后，用户只要在窗体上单击，就可显示

如图 7.13 所示的字体对话框。

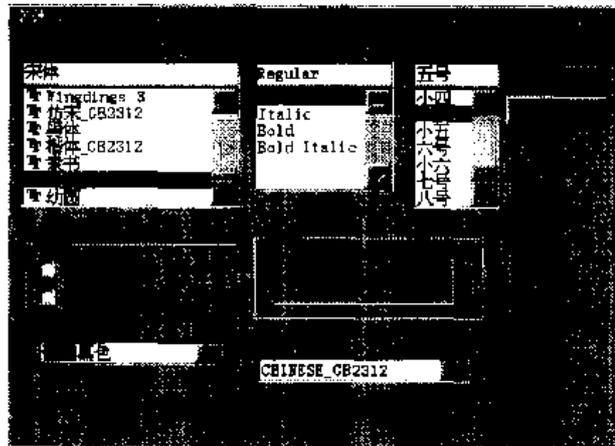


图 7.13 字体对话框

字体信息可用来设置程序中所有控件的字体，甚至可以为 Printer 对象设置字体。下面的代码说明如何检索字体信息，并用来改变文本框中的字体。

```
CommonDialog1.ShowFont
txtTemp.FontName = CommonDialog1.FontName
txtTemp.FontSize = CommonDialog1.FontSize
txtTemp.FontUnderline = CommonDialog1.FontUnderline
txtTemp.FontStrikethru = CommonDialog1.FontStrikethru
txtTemp.FontBold = CommonDialog1.FontBold
txtTemp.FontItalic = CommonDialog1.FontItalic
```

7.2.4 颜色对话框

使用通用对话框控件生成的【颜色】对话框的主要功能就是让用户在调色板中选择颜色，或者自行定制需要的颜色。所以，运行颜色对话框只需传回颜色值即可，该颜色记录在控件的 Color 属性中。

为了显示【颜色】对话框，应首先将通用对话框控件的 Flags 属性设置成 VB 常数 cdlCCRGBInit，这样可用 Color 属性获取选定颜色的 RGB 值，然后用 ShowColor 方法显示对话框，

例如：

```
Option Explicit
Private Sub Form1_Click()
    CommonDialog1.Flags = cdlCCRGBInit
    CommonDialog1.ShowColor
End Sub
```

程序运行后，用户只要在窗体上单击，就可显示如图 7.14 所示的【颜色】对话框。

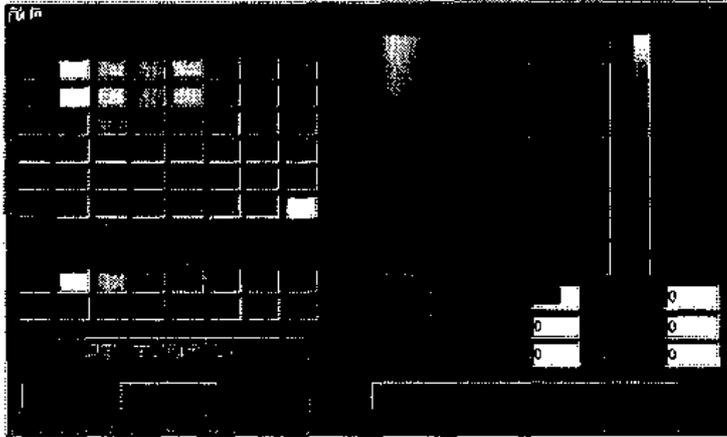


图 7.14 【颜色】对话框

下面的代码说明如何用 Color 对话框改变窗体的背景色。

```
CommonDialog1.Flags = cdlCCRGBInit
CommonDialog1.ShowColor
Form1.BackColor = CommonDialog1.Color
```

7.2.5 打印对话框

使用通用对话框控件生成的【打印】对话框允许用户指定打印输出的方法。在打印对话框中用户可指定打印页数范围、打印质量、复制数目等等，此外还有一个是否打印到文件的选项。为了显示【打印】对话框，应首先设置相应的【打印】对话框属性，为对话框设置默认值。例如，设置默认打印份数为 3 等等。然后使用 ShowPrinter 方法显示打印对话框，例如：

```
Option Explicit
Private Sub Form1_Click()
    CommonDialog1.Copies = 3
    CommonDialog1.ShowPrinter
End Sub
```

程序运行后，用户只要在窗体上单击，就可显示如图 7.15 所示的【打印】对话框。

一旦显示了【打印】对话框，用户就可以从对话框中的【名称】列表中选择打印机，这个列表包含了安装在 Windows 中的所有打印机。在【名称】列表下边是【状态】行，告诉用户选中的打印机的当前状态。

如果用户想改变打印机的任何参数（例如纸张大小和打印质量等），可以单击【打印】对话框中的【属性】按钮，这样就弹出选中的打印机的属性对话框，如图 7.16 所示。此对话框能够

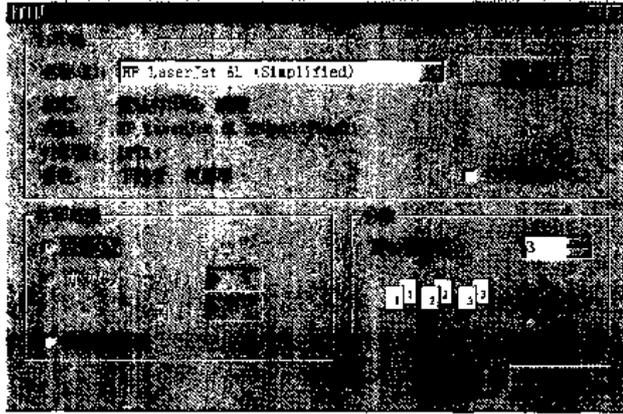


图 7.15 【打印】对话框

控制打印机的所有设置。

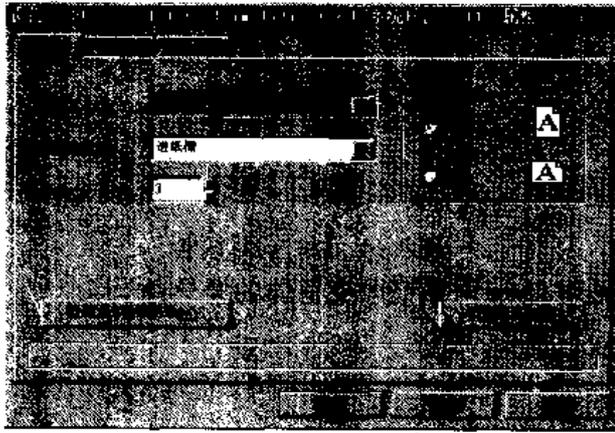


图 7.16 打印机的属性对话框

当用户在打印对话框中作出选择并退出后,相应的打印信息被保存到控件的某些属性中, `Frompage` 和 `Topage` 属性告诉用户选中打印输出的开始和结束页, `Copies` 属性告诉用户要打印的份数。

提示: 打印对话框并不自动建立打印输出,它允许用户指定如何打印数据,但是用户必须编写代码实现用选定格式打印数据。

7.2.6 帮助对话框

使用 `CommonDialog` 控件可以启动帮助系统,通用对话框控件的 `HelpFile`、`HelpKey`、`HelpCommand` 和 `HelpContext` 属性用于向帮助文件调用传递适当信息弹出通用帮助,上下文相关帮助。

要显示帮助对话框,首先必须指定帮助文件所在的路径和文件名,即先设置好控件的 `HelpFile`、`HelpCommand` 属性等,然后调用 `ShowHelp` 方法显示帮助对话框,例如:

```
Option Explicit
Private Sub Form_Click()
CommonDialog1.HelpFile = "C:\Windows\Help\Notepad.hlp"
CommonDialog1.HelpCommand = cdCHelpContents
```

CommonDialog1.ShowHelp

End Sub

程序运行后,用户只要在窗体上单击,即可启动 Windows 帮助系统,如图 7.17 所示。

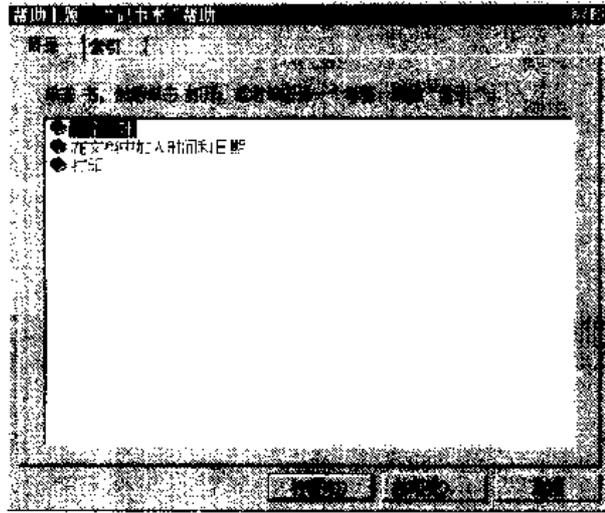


图 7.17 显示帮助对话框

7.3 定制对话框

用户在编写应用程序时,常常要用到各种各样的对话框,VB 提供了 MsgBox 和 InputBox 函数生成消息框和输入框,还提供了通用对话框控件快速生成【打开】、【另存为】、【颜色】、【字体】、【打印】等对话框。如果用户所要创建的对话框不能由 VB 现成的函数或控件生成,那么就只得自己定制对话框了。

定制对话框的过程也是编写一个小应用程序的过程,实际上就是创建一个窗体,窗体上包括一些控件用来实现对话框所需要的功能。

下面以建立一个运行程序的对话框为例,讲述用户自己定制对话框的基本步骤。

1. 单击【文件】菜单中的【新建工程】命令,创建一个新工程文件。一个缺省的窗体也被建立。
2. 调整窗体的位置和大小,在属性窗口中,设置窗体的属性如下表 7.6 所示。

表 7.6 对话框(窗体)属性设置

属性	属性值
Name	RunProgramForm
Caption	运行程序
BorderStyle	3-Fixed Dialog
ControlBox	True
MaxButton	False
MinButton	False



图 7.18 初始化界面

3. 往窗体上添加控件并设置其属性为下表所示,建立如图 7.18 所示的界面。

表 7.7 控件及其属性

控 件	Name 属性	Caption 属性
命令按钮	RunButton	确定
文本框	RunText	无
标签	RunLabel	文件名
框架	RunFrame	选项
单选框 1	NormalOption	常规
单选框 2	MaxOption	最大化
单选框 3	MinOption	最小化

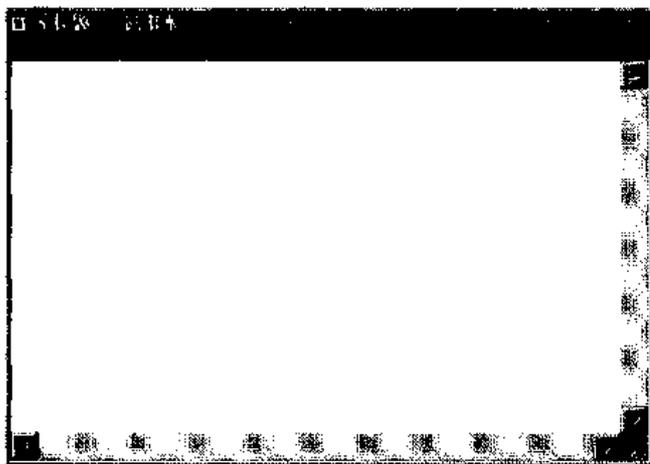


图 7.19 运行记事本程序

4. 编写程序代码,双击窗体的空白处打开代码编辑窗口,在对象列表框和过程列表框中选择相应的对象及事件过程,编下如下代码:

```
Option Explicit
Private Sub RunButton_Click()
On Error GoTo errorhandler
If NormalOption Then
Shell RunText.Text, 1
ElseIf MaxOption Then
Shell RunText.Text, 3
ElseIf MinOption Then
Shell RunText.Text, 2
Exit Sub
errorhandler:
MsgBox "不能运行该文件!"
Resume Next
End If
End Sub
```



图 7.20 指定文件不能运行

该程序运行后,在对话框中键入程序名,然后单击【确定】按钮即可运行该程序。例如键入 C:\Windows\notepad.exe 就可运行记事本程序,结果如图 7.19 所示。

如果键入的程序不能运行,则弹出如图 7.20 所示的消息框。

注意: 在文件名文本框要输入应用程序完整的路径和文件名,否则单击【确定】按钮都将出现不能运行的错误消息。

在程序中使用了 Shell 函数来运行一个可执行文件。Shell 函数的语法是:

Shell(文件名[窗口样式])

其中的“文件名”参数是必要参数,它是要执行程序的文件,以及任何必须的参数或命令行变量。若程序不在当前目录下,还应包括目录或文件夹,以及驱动器的信息。而“窗口样式”是可选参数,它用来表示在程序运行时窗口的风格,如果省略这个参数,则程序将以具有焦点的最小化窗口来执行。“窗口样式”参数取值及其描述参见表 7.8。

表 7.8 “窗口样式”参数取值

常 量	值	描 述
vbHide	0	窗口被隐藏,且焦点会移到隐式窗口
vbNormalFocus	1	窗口具有焦点,且还原到原来的大小和位置
vbMinimizedFocus	2	窗口会以一个具有焦点的图标来显示
vbMaximizedFocus	3	窗口将最大化并具有焦点
vbNormalNoFocus	4	窗口正常显示且没有焦点
vbMinimizedNoFocus	6	窗口最小化显示且没有焦点

注意: Shell 函数是以异步方式来执行其他程序的。也就是说,用 Shell 启动的程序可能还没有完成执行过程,而调用它的程序已经执行 Shell 函数之后的语句。

7.4 本章小结

对话框是应用程序中一种不可缺少的重要组成部分。学完本章后,用户要掌握以下内容:

1. 使用 MsgBox 函数生成消息框。
2. 使用 InputBox 函数生成输入框。
3. 使用通用对话框控件(CommonDialog)生成以下几种常用对话框:
 - 打开对话框
 - 字体对话框
 - 另存为对话框
 - 颜色对话框
 - 打印对话框
 - 帮助对话框

4. 由于应用程序需要各式各样的对话框为完成人机对话功能,当用户需要特定的对话框时,就要自己定制对话框,这也是一个窗体设计、编写代码的过程。

第 8 章 菜单设计

菜单是用户界面最重要的元素之一,它使用户界面更加友好、直观。如果一个比较大的应用程序的用户界面中没有菜单,用户就会感到无从下手,本章主要介绍如何使用 VB 6.0 创建应用程序的菜单。首先,我们应该了解一下菜单的基本概念。

8.1 菜单的基本概念

用户可利用菜单为应用程序提供一组命令,它提供一种给命令分组的方便的方法,使用户很容易访问这些命令。菜单一般在窗体内,图 8.1 所示的就是一个典型的有菜单的窗体。

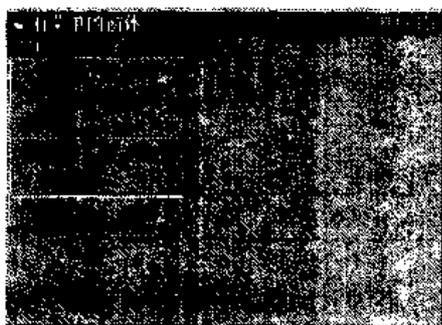


图 8.1 有菜单的窗体

关于菜单,主要有以下几个基本概念:

- 菜单栏

菜单栏一般出现在窗体的标题栏下面,并包含一个或多个菜单标题。图 7.1 中,【文件】、【编辑】、【视图】、【帮助】就是菜单标题。

- 菜单标题

菜单标题是菜单命令的一个子集,通常包含一个或多个菜单项。当用鼠标左键单击菜单标题时,包含菜单项目(以下简称菜单项)的列表就被拉下来,因此,有时称之为下拉式菜单。当菜单标题没有菜单项时,就称之为顶层菜单(TopLevel Menu),通常要在它的后面加上标记符号“!”。

- 菜单项

菜单项隶属于菜单标题,它可以包括菜单命令,分隔线和子菜单标题。图 8.1 所示菜单标题【文件】是由菜单命令【新建】、【打开】、【关闭】、【保存】、【另存为】、【退出】和两个分隔条共 8 个菜单项组成。

- 快捷菜单

快捷菜单又叫弹出式菜单,它是独立于菜单栏而显示在窗体上的浮动菜单。在快捷菜单中显示的项目取决于单击鼠标右键时指针所在的位置,因此,快捷菜单也被称为上下文菜单。在 Windows 95 / 98 中,一般是通过单击鼠标右键来激活快捷菜单。而应用快捷菜单能提供一种访问公共的上下文命令的高效方法。

- 菜单的状态

菜单有两种状态:一是显示状态,叫显式菜单;另一是隐藏状态,叫隐藏菜单。显式菜单即是窗口运行后出现在菜单栏上,可直接对其进行操作的菜单,其 Visible 属性为 True。而隐藏菜单是指窗口运行后不出现在菜单栏上,遇上一定的事件或方法后可能显示出来的菜单,其 Visible 属性为 False。用户可通过设置菜单的 Visible 属性控制菜单的显示或隐藏。

- 菜单项的访问键和快捷键

通过定义访问键和快捷键可以改进键盘对菜单命令的访问。如果某一字符是该菜单项的

访问键,在菜单中,这一字符下方会有一下划线,用户只要同时按下 Alt 键和该字符键,就可以激活该菜单。使用快捷键比使用访问键能够更快地选取命令,快捷键按下时会立即执行一个相应的菜单命令。

此外,关于菜单和菜单项还有有效与失效、菜单项的复选标记等概念,将在以下两节中作详细介绍。

8.2 菜单编辑器

在 VB 6 中,有一个很好的工具可以帮助用户创建自己的菜单系统,这个工具就是 VB 的菜单编辑器。使用菜单编辑器,可向现有的菜单中增加新命令,用自己的命令替代现有的菜单命令,产生新的菜单和菜单栏以及修改和删除现有菜单和菜单栏。菜单编辑器的主要优点是使用方法,用户可以在只用很少编程的交互方式中自定义菜单。

8.2.1 菜单编辑器的显示与隐藏

设计菜单主要是在菜单编辑器内进行,因此开始设计菜单时,必须打开菜单编辑器窗口,让菜单编辑器显示在集成开发环境中。用户可以用以下几种方法的任何一种打开菜单编辑器窗口:

1. 让窗体显示在开发环境中,选择【工具】菜单中的【菜单编辑器】命令。
2. 让窗体显示在开发环境中,按 Ctrl + E 键。
3. 单击标准工具栏中的【菜单编辑器】按钮。
4. 右击窗体上任一位置,在弹出的快捷菜单中(如图 8.2 所示)选择【菜单编辑器】命令。



图 8.2 从窗体快捷菜单中选择【菜单编辑器】

打开后的菜单编辑器窗口如图 8.3 所示。

如果用户要关闭菜单编辑器窗口,可以直接单击菜单编辑器窗口右上角的【关闭】按钮,或者在菜单编辑器窗口的标题栏上单击鼠标右键,在弹出的快捷菜单中选择【关闭】菜单项。

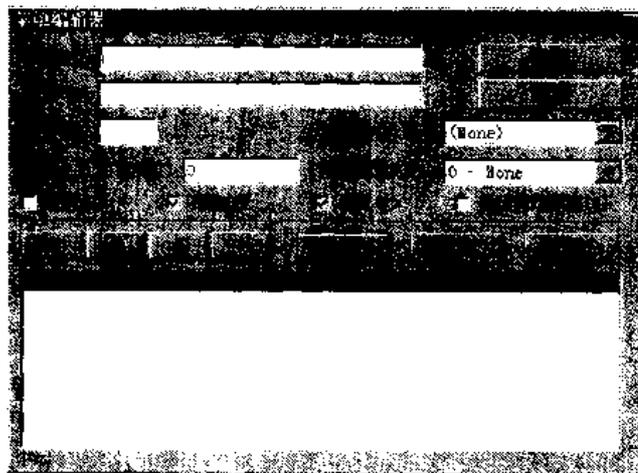


图 8.3 菜单编辑器窗口

8.2.2 菜单编辑器的组成

菜单编辑器分为上下两部分。上半部分用来设置属性,下半部分则用来显示用户设置的菜单和菜单项。各项内容和作用如下:

- 【标题】(Caption)输入框

【标题】输入框是一个文本框,用来输入用户建立菜单的标题名。在这个文本框中输入标题后,将显示在用户所建立的菜单中。菜单标题名将在下面的菜单控件列表框中显示出来。

- 【名称】(Name)输入框



【名称】输入框也是一个文本框,用来输入各菜单名及菜单项的控制名,控制名是代码编辑时标明身份的标志。菜单设置中的每一项,包括菜单名、菜单项,甚至不参加代码编辑的分隔符,只要在【标题】框中输入过标题,都要为其取一个控制名,否则,单击【确定】按钮后系统弹出如图 8.4 所示的消息框。

图 8.4 菜单控件必须有一个名称

- 【索引】(Index)输入框

【索引】框也是文本框,它为用户建立的控件数组设立下标,控件数组是一组享有同一控制名但各自拥有不同的属性的控件。

- 【快捷键】输入框

【快捷键】输入框为一下拉列表框,单击该框或者单击其右端的向下方向箭头,即会出现一列表框,列出用户可以选择的各种快捷键。

- 【帮助上下文标识符】(HelpcontextID)输入框

【帮助上下文标识符】是一文本框,用户可以通过键入数字来选择帮助文件中特定的页数。

- 【协调位置】框

【协调位置】框为一下拉列表框,单击该框或者其右侧的下拉箭头可以显示列表,通过这一列表框来确定顶层菜单在窗体中是否出现或怎样出现。

- 【复选】框

若选中【复选】框,可以把一个复选标记“V”放置在菜单项前面,一个复选标记可用来表示一个控件打开/关闭的状态,也可以用来指示几个模式中哪一个模式正在起作用。

- 【有效】属性

【有效】属性是一个用来设置该菜单项是否为可执行的复选框,即用户是否想让这一菜单项响应某一事件,如果不选中该复选框,则不能访问这一菜单项,也即这一菜单命令是无效的,呈灰色显示。

- 【可见】属性

【可见】属性决定菜单或菜单项是否可见,若不选中该复选框,则相应的菜单或菜单项将不可见。

- 【显示窗口列表】属性

用户可以用【显示窗口列表】属性来决定是否在使用多文档应用程序时,使菜单控件中有一包含打开的多文档子窗口的列表框。

- 菜单控件列表框

在该列表框中将列出用户为某一窗体所设计的所有菜单名和菜单项。它位于菜单编辑器的最底端。

在菜单编辑器窗口中,还有四个箭头按钮和【下一个】、【插入】及【删除】按钮,关于它们的用法和作用将在以下章节中详细介绍。

8.3 用菜单编辑器创建菜单

用菜单编辑器可以创建新的菜单和菜单栏,在已有的菜单上增加新命令,用自己的命令来替换已有的菜单命令,以及修改和删除已有的菜单和菜单栏。

8.3.1 在菜单编辑器中创建菜单控件

在 VB 中,菜单和菜单下的菜单项也都是控件,它们使用起来与命令按钮这样的控件没有什么区别。用户可以在设计或运行时设置菜单控件的 Caption 属性、Enabled 和 Visible 属性、Checked 属性以及其他属性。菜单控件只包含一个事件即 Click 事件。当程序运行后,用鼠标或键盘选择该菜单控件时,将调用该事件。

大多数菜单控件属性可用菜单编辑器设置,同样,所有的菜单属性也可以在属性窗口中得到。在菜单编辑器中创建菜单控件的具体方法如下:

1. 选取要添加菜单栏的窗体。
2. 从【工具】菜单中,选择【菜单编辑器】命令,或者单击标准工具栏上的【菜单编辑器】按钮,显示菜单编辑器。
3. 在【标题】文本框中,为第一个菜单标题键入在菜单栏上显示的文本,即菜单控件的 Caption 属性。菜单标题文本会显示在菜单控件列表框中。
4. 在【名称】文本框中,输入将用来在代码中引用该菜单控件的名字,即菜单控件的 Name 属性。
5. 单击向左或向右箭头按钮,可以改变该控件的缩进级。单击向右箭头按钮可以增加一级缩进,单击向左箭头按钮可以删除一级缩进。在菜单编辑器中,每一缩进级都使用 4 个点“....”表示。
6. 如果需要的话,还可以设置菜单控件的其他属性,这一工作可以在菜单编辑器中做,也可以以后在属性窗口中做。
7. 单击【下一个】按钮就可以再建一个菜单控件。或者单击【插入】按钮,可以在现有的控件之间增加一个菜单控件。也可以单击向上或向下的箭头按钮,在现有菜单控件之中移动控件,改变它们之间的相对位置。
8. 如果窗体所有的菜单控件都已创建,单击【确定】按钮可关闭菜单编辑器。

创建的菜单标题将显示在窗体上,在设计时,单击一个菜单标题可下拉其相应的菜单项。菜单编辑器下部的列表框列出了当前窗体的所有菜单控件,从列表框中选取一个已存在的菜单控件可以编辑该控件的属性。

例如,图 8.5 示出典型应用程序中【文件】菜单的各种菜单控件。菜单控件在菜单控件列表框中的位置决定了该控件是菜单标题、菜单项、子菜单标题还是子菜单项:

- 位于列表框中左侧平齐的菜单控件作为菜单标题显示在菜单栏中。
- 列表框中被缩进过的菜单控件,当单击其前导的菜单标题时,才会在该菜单上显示。
- 一个缩进过的菜单控件,如果后面紧跟着再次缩进的一些菜单控件,它就成为一个子菜单的标题,在子菜单标题以下缩进的各个菜单控件,就成为该子菜单的菜单项。

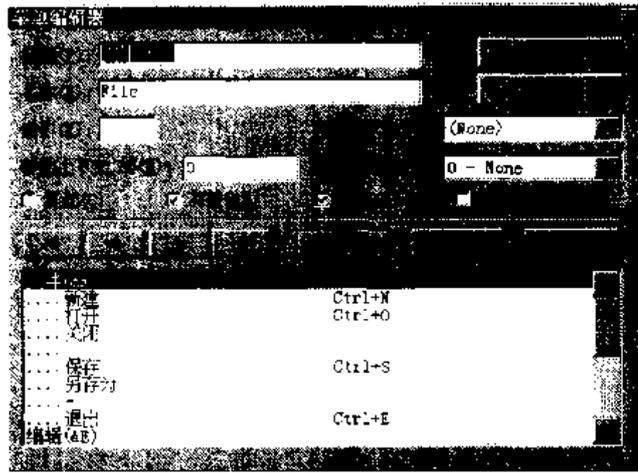


图 8.5 菜单控件列表框列出所有菜单控件

提示：如果某个菜单命令将激活一个对话框，则在其名称后面最好有一个省略号“...”，这种表示方法便于用户理解。

8.3.2 分隔菜单项

分隔线作为菜单项间的一个水平行显示在菜单上，在菜单项较多的菜单上，可以使用分隔线将各项划分成一些逻辑组。如图 8.6 所示的【文件】菜单就被分隔线分成三组。

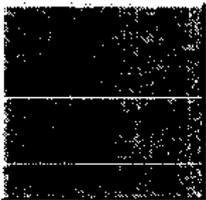


图 8.6 分隔线

用户可以在菜单编辑器中创建分隔线，具体方法为：

1. 若菜单中已有菜单项，应选择【插入】按钮，在想要分隔开来的菜单项之间插入一个菜单控件。
2. 若有必要，单击向右箭头按钮使新菜单控件与它要隔开的菜单项同级。
3. 在【标题】文本框中键入一个连字符“—”。
4. 在【名称】文本框中输入该菜单控件的名字，即菜单控件的【名称】属性。
5. 单击【确定】按钮，关闭菜单编辑器。

注意：虽然分隔线是当作菜单控件来创建的，它们却不能响应 Click 事件，而且也不能被选取。

8.3.3 赋值访问键和快捷键

通过定义访问键和快捷键可改进键盘对菜单命令的访问。如果某一字符是该菜单项的访问键，则该字符下方会有一条下划线(如图 8.7)所示，用户只要同时按下 Alt 键和该字符键，就可以激活该菜单。一旦菜单打开，通过按下所赋值的字符(访问键)可选择控件。例如，控下 Alt + E 键可以打开【编辑】菜单，再按下 X 键可执行【剪切】命令。

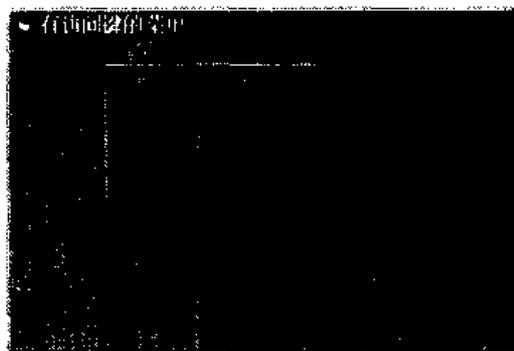


图 8.7 访问键

如果要在菜单编辑器中给菜单控件赋值访问键，只需在菜单编辑器的菜单控件列表框中选择要赋值访问键的菜单项，然后在其【标题】框中，在

要作为访问键字符的前面直接键入一个“&”字符。例如,如果图 8.7 所示的【编辑】菜单被打开,则各个菜单控件的一些属性值如下表所示。

表 8.1 【编辑】菜单下的菜单控件属性

菜单控件标题	标题属性	访问键
剪切	剪切(&T)	T
复制	复制(&C)	C
粘贴	粘贴(&P)	P
删除	删除(&L)	L
全选	全选(&A)	A

注意: 菜单中不能使用重复的访问键,如果多个菜单项使用同一个访问键,则该键将不起作用。例如,不能用 C 同时作为【剪切】(Cut)和【复制】(Copy)的访问键。

如果使用快捷键就可以大大提高选择菜单命令的速度,快捷键按下时会立即执行一个菜单项。用户可以为频繁使用的菜单项指定一个快捷键。快捷键的赋值包括功能键与控制键的组合,例如剪切、复制、粘贴的快捷键分别为 Ctrl + X, Ctrl + C 和 Ctrl + V,它们出现在菜单中相应菜单项的右边,如图 8.8 所示。

图 8.8 快捷键

如果要对菜单项指定快捷键,要首先打开(菜单编辑器),然后从菜单控件列表中先选择该菜单项,再从【快捷键】下拉列表框中选择功能键和键的组合,如图 8.9 所示。快捷键将自动出现在菜单上,不需要在菜单编辑器的【标题】框中键入诸如“Ctrl + X”这样的字符,如果要删除快捷键赋值,应选择列表顶部的“(None)”选项。

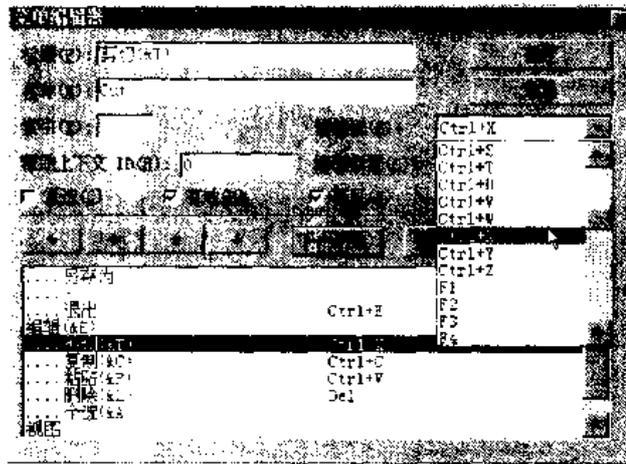


图 8.9 为菜单命令指定快捷键

注意: 快捷键提供一种键盘单步的访问方法,而访问键则是按住 Alt 键,再按菜单标题访问字符,然后再按菜单项访问字符的三步方法。

8.3.4 编写菜单控件的代码

当用户选择一个菜单控件时,一个 Click 事件出现。需要在代码中为每个菜单控件编写

个 Click 事件过程。除分隔线以及无效的或不可见的菜单控件外,其他所有菜单控件都能识别 Click 事件。

在菜单事件过程中编写的代码与在控件任何其他事件过程中编写的代码完全相同。例如:【文件】菜单中的【关闭】菜单项的 Click 事件代码如下:

```
Sub mnuFileClose_Click()  
    Unload Me  
End Sub
```

一旦菜单标题被选择,VB 将自动地显示出其下拉菜单;但是,没有必要为一个菜单标题的 Click 事件过程编写代码,除非想执行其他操作,如每次显示菜单时使某些菜单项有效或无效。

注意:在设计时,当关闭菜单编辑器时,所创建的菜单将显示在窗体上,在窗体上选择一个菜单项时,将显示那个菜单控件的 Click 事件过程。

8.3.5 创建子菜单

使用菜单编辑器创建的每个菜单最多可以包含 5 级子菜单,子菜单会分支出另一个菜单以显示它自己的菜单项。一般地,在以下几种情况下需要使用子菜单:

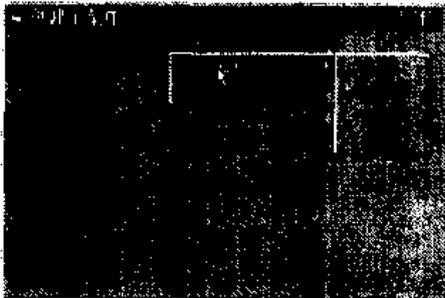


图 8.10 运行后的子菜单

- 菜单栏已满。
- 某一特定菜单很少被用到。
- 要突出某一菜单栏中还有空间,最好再创建一个菜单标题而不是子菜单。这样,当菜单拉下时,所有菜单控件都可见。

在菜单编辑器中,在不是菜单标题的菜单控件之下缩进的任何菜单控件,都是子菜单控件。一般说来,子菜单控件可以包括子菜单项、分隔线和子菜单标题。如图 8.10 所示的菜单项【工具栏】有子菜单【标准】、【格式】、【绘图】

和【表格】,它们在菜单编辑器列表框中显示如图 8.11 所示。

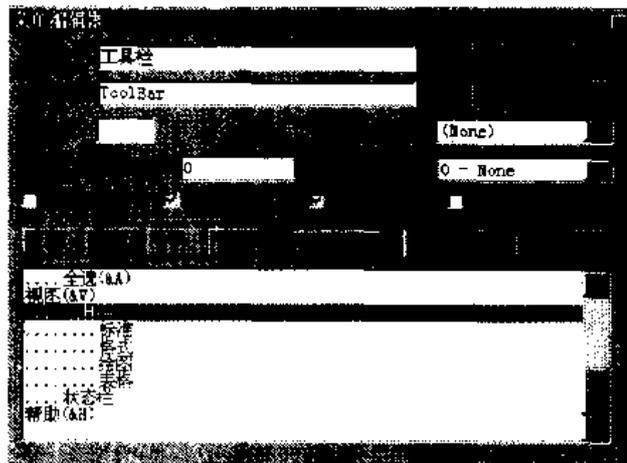


图 8.11 创建子菜单

用户如果要在菜单编辑器中创建了子菜单,应首先创建想作为子菜单标题的菜单项,再创建将出现在新子菜单中的各个项目,然后单击向右箭头按钮将它们缩进。要删除一级缩进,可单

击向左箭头按钮。

提示：如果想用多于一级的子菜单，可以考虑使用对话框来替代。对话框允许在一个地方指定好几个选择。

8.4 运行时创建和修改菜单

用户在设计时创建的菜单也能动态地响应运行时的条件。例如，如果菜单项的动作在某些点上成为不适当时，通过使其失效可防止对该菜单项的选取。也可以编写应用程序，使用复选标志来指示几个命令中的哪一个是最后选取的。如果有一个菜单控件数组，也可以动态地增加菜单项，这个内容将在下节“在菜单中增加文件列表”中介绍。

8.4.1 使菜单命令有效或无效

所有的菜单控件都具有 Enabled 属性，当这个属性为 False 时，菜单命令无效，使它不响应动作。当 Enabled 属性设为 False 时，快捷键的访问也无效。当一个菜单控件无效时，它将呈灰色显示，如图 8.12 所示的【粘贴】菜单项。

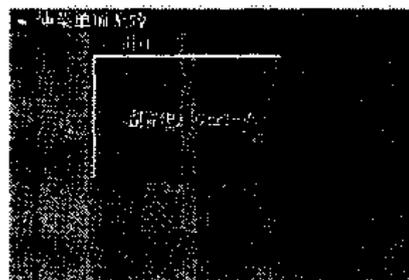


图 8.12 使菜单项无效

例如：设【粘贴】菜单控件的 Name 属性为 mnuEditPaste，下列语句可使【粘贴】菜单项无效：

```
mnuEditPaste.Enabled = False
```

菜单标题的无效使得整个菜单无效，因为不首先单击菜单标题，就不能访问任何菜单项。例如，下列代码将使某应用程序中的【编辑】菜单无效：

```
mnuEdit.Enabled = False
```

若用户某一动作要使该菜单有效，只需将其 Enabled 属性设为 True 即可，代码为：

```
mnuEdit.Enabled = True
```

8.4.2 使菜单控件不可见

在菜单编辑器中，通过选取被标记为【可见】的复选框，可以设置菜单控件的 Visible 属性的初值。在运行时，要使一个菜单控件可见或不可见，可以在代码中设置其 Visible 属性。例如：

```
mnuEditCopy.Visible = True '使菜单控件可见
```

```
mnuEditCopy.Visible = False '使菜单控件不可见
```

当一个菜单控件不可见时，菜单中的其余控件会上移以填补空出的空间。如果控件位于菜单栏上，则菜单栏上其余控件会左移以填补该空间。

注意：使菜单控件不可见也产生使之无效的作用。因为该控件通过菜单、访问键或者快捷键都再无法访问。如果菜单标题不可见，则该菜单上所有控件均无效。

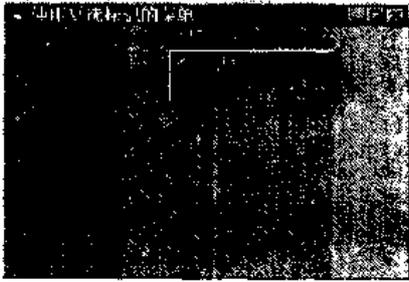


图 8.13 在菜单中使用复选标记

8.4.3 在菜单中使用复选标记

有时候,为了表示打开/关闭状态或指示几个模式中哪一个正在起作用,需要在菜单的选项前显示一个复选标记“√”,如图 8.13 所示。

在 VB 中可以通过设置控件的 Checked 属性来设置菜单选项的复选标记。在菜单编辑器中,选中【复选】复选框可以设置菜单选项的初始状态就是显示“√”,当 Checked 属性为 False 时,“√”标记消失。例如:

```
mnuViewToolbar.Checked = False'消除复选标记
```

8.5 在菜单中添加文件列表

现在许多 Windows 应用程序都具有在文件菜单下列出最近访问过的文件列表的功能,如图 8.14 所示。

要实现这样的功能,只靠菜单编辑器是完成不了的,还必须有代码来配合控制,创建菜单控件数组是一个重要步骤。

8.5.1 创建菜单控件数组

菜单控件数组就是在同一菜单上共享相同名称和事件过程的菜单项目的集合。在下列两种情况一般可使用菜单控件数组:

- 在运行时要创建一个新菜单项,它必须是控件数组中的成员。
- 简化代码,因为通用代码块可以被所有菜单项使用。

每个菜单控件数组元素都由唯一的索引值来标识,该值在菜单编辑器上【索引】框中指定。当一个控件数组成员识别一个事件时,VB 将其 Index 属性值作为一个附加的参数传递给事件过程。事件过程必须包含有核对 Index 属性值的代码,因此可以判断出正在使用的是哪一个控件。

用户如果要在菜单编辑器中创建菜单控件数组,可以按下列方法进行:

1. 选取窗体。
2. 在【工具】菜单中选择【菜单编辑器】选项或单击工具栏上的【菜单编辑器】按钮,显示菜单编辑器窗口。
3. 在【标题】文本框中,输入想出现在菜单栏中的第一个基本标题的文本,例如“文件”。菜单控件列表框中将显示该标题文本。
4. 在【名称】文本框中,输入将在代码中用来引用菜单控件的名称。保持【索引】框为空的。

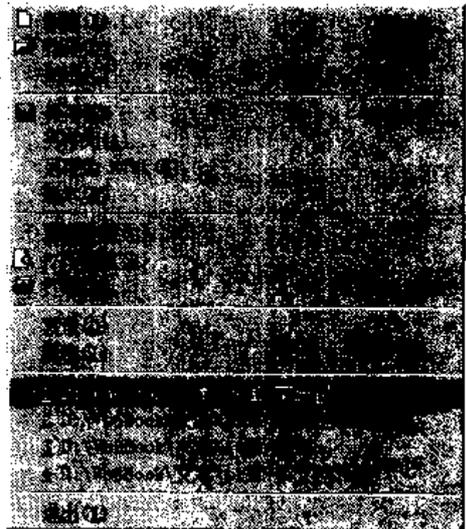


图 8.14 在文件菜单下列出文件列表

5. 在下一个缩进级,通过设定【标题】和【名称】来创建将成为数组中第一个元素的菜单项。
6. 将数组中第一个元素的【索引】设置为 0。
7. 在第一个的同一缩进级上创建第二个菜单项。
8. 将第二个元素的【名称】设置成与第一个元素相同,且把它的【索引】设置为 1。
9. 对于数组中的后续元素重复步骤 5~8。

提示: 菜单控件数组的各元素在菜单控件列表框中必须是连续的,而且必须在同一缩进级上。创建菜单控件数组时,要把在菜单中出现的分隔线也包括进去。

8.5.2 添加文件列表

创建了菜单控件数组之后,用户可以在打开文件的过程中添加一些代码,把文件名添加到文件菜单下。例如,定义一个 4 个元素的菜单控件数组,它们的 Name 属性为 mnuFileM,其中第一个元素(即 Index 为 0 的元素)被定义为分隔线,所有的数组元素在没有打开文件之前,Visible 属性都是 False,用户无法看到。当用户从【打开】对话框中选择了-一个文件名后,可以添加这样的代码,将文件名添加到文件列表中:

```
ItemCnt = ItemCnt + 1
If ItemCnt = 1 Then
mnuFileM(0).Visible = True '显示分隔线
mnuFileM(ItemCnt).Caption = SFile '用户选择的文件名保存在 SFile 变量中
mnuFileM(ItemCnt).Visible = True '显示文件名
```

其中,变量 ItemCnt 是一个窗体级变量,它用来保存最新文件的位置。当然用户还可以向这段代码中添加内容,完成排序等功能。

采用上述方法,只能放置固定数目的文件列表,如果要无限制地添加文件列表,必须使用动态的菜单控件数组。

要使用动态的菜单控件数组,用户只需在菜单编辑器中定义一个控件数组元素 mnuFileM(0)并将它设置为分隔线。遇到要向文件列表中添加的情况时,只需使用下列语句:

```
ItemCnt = ItemCnt + 1
If ItemCnt = 1 Then
mnuFileM(0).Visible = True
Load mnuFileM(ItemCnt) '添加一个数组元件
mnuFileM(ItemCnt).Caption = SFile
mnuFileM(ItemCnt).Visible = True
```

用户不仅可以向数组中添加元素。同样可以根据需要删除元素,如果要删除数组中的某个元素,可以使用下列语句:

```
Unload mnuFileM(Index)
```

其中,变量 Index 是要删除元素的索引值。

8.5.3 保存文件列表

使用 8.5.2 节添加文件列表的方法不能保存文件列表。当应用程序结束后,所有的文件

列表都将丢失,用户下一次启动应用程序后,无法通过文件列表直接打开最近使用的文件。

要想在关闭应用程序后仍然保留数据,就必须将数据保存到文件中去。所以用户可以专门创建一个文件用来记录文件列表,在 Windows 95、Windows 98 或者 NT 下,可以利用注册表来保存应用程序的有关设置。

单击 Windows 95、Windows 98 或者 Windows NT 的【开始】菜单,然后单击【运行】命令,在【运行】对话框中输入“C:\Windows\regedit”就可以运行注册表编辑器,查看系统注册表的内容,如图 8.15 所示。

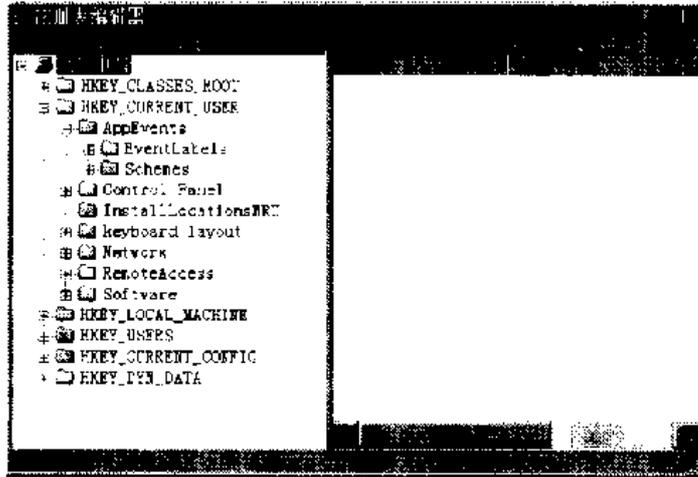


图 8.15 注册表编辑器

注意: 如果用户对注册表不了解,最好不要修改或删除注册表中的内容,否则可能会导致系统的混乱甚至崩溃。

另外,VB 还提供了 4 个语句或函数来处理存储在应用程序注册表中的程序设置值,这四个函数或语句及其描述见表 8.2。

表 8.2 VB 中与注册表有关的函数和语句

函数或语句	描述
GetSetting 函数	检索注册表设置值
SaveSetting 语句	保存或创建注册表设置值
GetAllSettings 函数	返回一个包含多项注册表设置值的数组
DeleteSetting 语句	删除注册表设置值

关于这四个函数的具体使用,请读者查阅帮助信息,本书不作详细介绍了。

8.6 快捷菜单

快捷菜单是独立于菜单栏而显示在窗体上的浮动菜单。在快捷菜单中显示的项目取决于按下鼠标右键时指针所处的位置;因而快捷菜单也被称为上下文菜单,通常用来提供一种访问与上下文有关的命令的高效方法,Windows 95/98 中,一般是通过单击鼠标右键来激活快捷菜单。

在运行时,只要菜单至少包含一个菜单项,就可作出可以被显示的快捷菜单。为了显示快捷菜单,可使用 PopupMenu 方法。PopupMenu 方法的语法是:

```
[object].PopupMenu menuname[, flags[, x[, y[, boldcommand]]]]
```

例如当用户单击鼠标右键时,以下代码将显示一个名为 mnuEdit 的菜单。用户可用 MouseUp 或者 MouseDown 事件来检测是否单击了鼠标右键,但标准用法是使用 MouseUp 事件:

```
Private Sub Form1_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As Single)
    If Button = 2 Then
        PopupMenu mnuEdit
    End If
End Sub
```

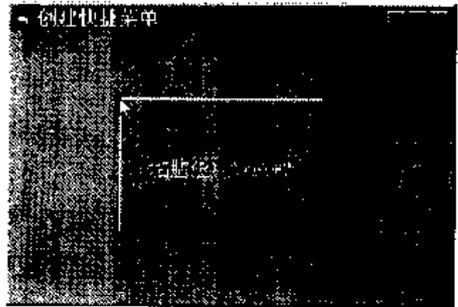


图 8.16 创建快捷菜单

程序运行后,用户只要在窗体上任意位置单击鼠标右键,就会弹出一个快捷菜单,菜单内容是【编辑】菜单下的菜单项,如图 8.16 所示。

运行这个程序时,程序直到菜单中被选取一项或者取消这个菜单时,调用 PopupMenu 方法后面的代码才会运行。

注意: 每次只能显示一个快捷菜单,在已显示一个快捷菜单的情况下,对后面的调用 PopupMenu 方法将不予理睬。

用户常常会想用快捷菜单来访问在菜单栏中不常用的选项,要创建一个不显示在菜单栏里的菜单,可在设计时使顶级菜单项目为不可见(在菜单编辑器中消除【可见】复选框)。当 VB 显示一个快捷菜单时,指定的顶级菜单的 Visible 属性会被忽略。

另外,在 PopupMenu 方法中使用 flags 参数可以进一步定义快捷菜单的位置与性能。表 8.3 列出了可用于描述快捷菜单位置的 flags 参数的取值。

表 8.3 用于描述快捷菜单位置的 flags 参数

位置常数	值	描述
vbPopupMenuLeftAlign	0	缺省。指定的 X 位置定义了快捷菜单的左边界
vbPopupMenuCenterAlign	4	快捷菜单以指定的 X 位置为中心
vbPopupMenuRightAlign	8	指定的 X 位置定义了快捷菜单的右边界

表 8.4 列出了可用于定义快捷菜单行为的参数取值。

表 8.4 描述快捷菜单行为的 flags 参数

行为常数	值	描述
vbPopupMenuLeftButton	0	缺省。快捷菜单仅识别鼠标左键对菜单项的选择
vbPopupMenuRightButton	2	快捷菜单识别鼠标左键和右键对菜单项的选择 PopupMenu 方法中的 boldcommand 参数用来指定在显示的快捷菜单中以粗体字体出现的菜单控件的名称,在快捷菜单中只能有一个菜单控件被加粗

如果要指定一个 `flags` 参数,可以以每组中选择一个常数,再用 `Or` 操作符将它们连起来。下面的代码是:当用户单击鼠标右键时,显示一个上边框以鼠标当前位置为中心的快捷菜单,而且这个快捷菜单还可以识别右键对菜单命令的选择,本例中还使用粗体显示菜单控件 `FileNew`:

```
Private Sub Form1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 2 Then
PopupMenu mnuFile, vbPopupMenuCeterAlign Or vbPopupMenuRightButton, X, Y, mnuFileNewEnd If
End Sub
```

8.7 本章小结

许多简单的应用程序只由一个窗体和几个控件组成,但是通过增加菜单可以增强 VB 应用程序的功能,本章介绍了如何在应用程序中创建菜单和使用菜单。学完本章读者应掌握以下内容:

1. 菜单的基本概念
2. 菜单编辑器的组成
3. 在菜单编辑器中创建菜单控件
4. 分隔菜单项以及给菜单项赋值访问键和快捷键
5. 创建子菜单
6. 使菜单命令有效或无效、可见或不可见以及在菜单中使用复选标记
7. 创建菜单控件数组并会在菜单中添加文件列表
8. 显示快捷菜单

第 9 章 工具栏和状态栏

工具栏已经成为 Windows 应用程序的一个标准配备,使用它可以进一步增强应用程序的菜单界面。工具栏含有工具栏按钮,对大多数 Windows 程序的使用者来说,许多工具栏按钮可以不加思索地加以利用,例如【打开文件】按钮、【保存文件】按钮以及【复制】按钮等等。

用户也可以在应用程序中添加状态栏。状态栏通常位于窗体底部,可以显示文本或预定义数据,例如自动更改的时间和日期等。

要在 VB 6.0 应用程序的窗体中添加工具栏和状态栏是很容易的。VB 提供了两个 ActiveX 控件——ToolBar 控件和 StatusBar 控件。使用它们可以非常容易,而且方便地创建工具栏和状态栏。

ToolBar 和 StatusBar 都是一组 ActiveX 控件的一部分,要想方便地应用这两个控件,首先要将它们添加到工具箱中;而且在创建脱离 VB 集成开发环境的 Windows 应用程序时,还必须把包含该控件的文件安装到用户的系统文件中。

9.1 关于 ActiveX 控件

在 VB 编程系统中,ActiveX 控件以前被称为 OLE 控件,它是 VB 工具箱的扩充部分。在程序中加入 ActiveX 控件后,它将成为开发和运行环境的一部分,并为应用程序提供新的功能。

ActiveX 控件文件的扩展名为 .ocx,将它加入到工具箱中,即可在工程中使用它们。在第六章已经介绍过如何在工具箱中添加 ActiveX 控件的工作,在这里简述添加 ToolBar、StatusBar 和 ImageList 控件的方法:

首先从【工程】菜单中选择【部件】项,然后从【部件】对话框中选中“Microsoft Windows Common Control 6.0”的复选框,如图 9.1 所示。载入文件后,工具箱中会增加几个控件按钮,其

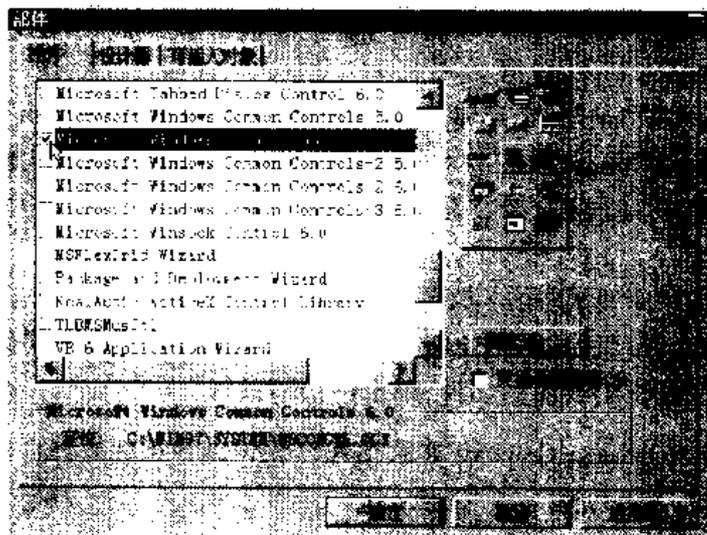


图 9.1 添加 ToolBar 和 StatusBar 控件

中有 ToolBar、StatusBar 和 ImageList 控件等。

如果脱离 VB 集成开发环境,直接在 Windows 中运行 VB 应用程序,就必须创建可执行文件。如果应用程序含有 ActiveX 控件,就必须在系统中注册与其相关的 .ocx 文件,否则,应用程序就找不到创建该控件的代码。

如果找不到某个控件,VB 运行时将产生“文件未找到”错误信息。为此,必须在该应用程序的安装过程中,将所需的 .ocx 文件复制到 \Windows\System 目录。该安装过程还应在系统中注册所需控件。

9.2 创建工具栏

有了 ToolBar 控件,就可以通过将按钮添加到按钮集合中创建工具栏。每个按钮对象可以用 Caption 属性显示文本,或者拥有相关联的 ImageList 控件提供的图像,或者二者兼而有之。这里提到了 ImageList 控件,为其他 Windows 公共控件保管图像,我们将在后面对它予以介绍。

在创建好工具栏之后,还要给其编程,即对 Button_Click 事件添加代码,以便对选定的按钮作出反应。概括一下,使用 ToolBar 控件创建工具栏的一般步骤如下:

1. 在与 ToolBar 控件相关联的 ImageList 控件中插入合适的图像。
2. 创建工具栏快捷按钮。
3. 对按钮的 ButtonClick 事件编程,进行相应的操作。

下面先介绍 ImageList 控件的有关内容。

9.2.1 使用 ImageList 控件

如前所述,ImageList 为其他 Windows 公共控件保管图像,所以,可以把它视为一种图像仓库。它提供了单一的、一致的图像目录,这样就节省了开发时间。用户可以不编写装载位图或图标(使用 LoadPicture 函数)的代码,而是一次性将用到的所有图像填充到 ImageList 中。在需要的时候设置 key 的值,然后在代码中使用 key 或 Index 属性引用所需的图像。

该控件使用 ListImage 对象集合中的位图(.bmp)或图标(.ico)文件,在设计时和运行时均可添加和删除图像。ListImage 对象具有集合对象的标准属性:Key、Index 和 count。它还有标准的方法,例如 Add、Remove 和 Clear 等。

双击工具箱中的 ToolBar 按钮,窗体上会出现一个 ToolBar 控件;双击工具箱中的 ImageList 按钮,窗体上会出现一个 ImageList 控件。按以下步骤可以在该控件中添加 ListImage 对象:

1. 用鼠标右键单击 ImageList 控件,然后在弹出菜单中单击【属性】项,显示出 ImageList 控件的【属性页】对话框

2. 单击【图像】选项卡,如下面的图 9.2 所示。
3. 单击【插入图片】按钮,显示出如图 9.3 所示的【选定图片】对话框。
4. 用该对话框找到位图或图标文件,单击【打开】按钮。

提示: 在 VB98 \ Common \ Graphics 目录下有大量位图和图标文件。

注意: 可以同时选中多个位图或图标文件。

5. 单击【关键字】框,并在其中键入一个字符串,为该图像赋予唯一的 Key 属性。
6. (可选的操作)单击【标记】编辑框,并在其中键入一个字符串,为该图像赋予一个 Tag

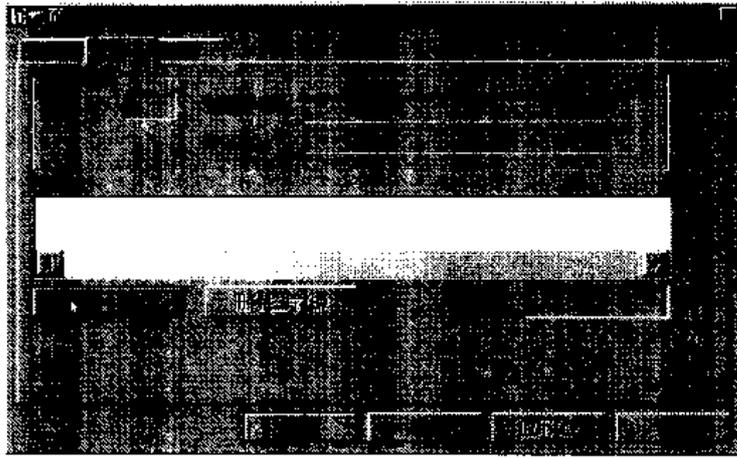


图 9.2 ImageList 控件【属性页】对话框的【图像】选项卡

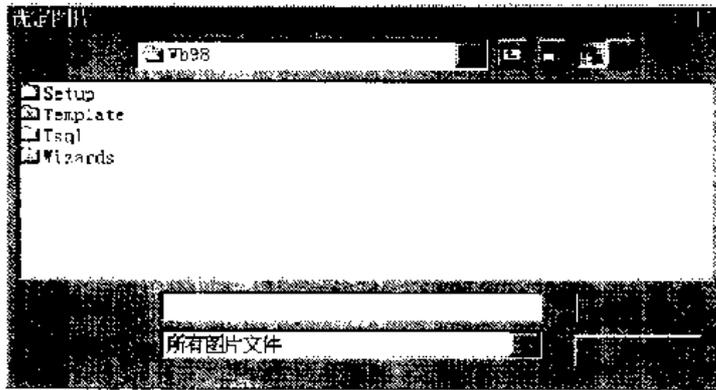


图 9.3 【选定图片】对话框

属性, Tag 属性不必唯一。

7. 重复 3 至 6 步, 直到将需要全部图像填充到该控件中。

该控件可包含任意大小的 .bmp 和 .ico 图像。通常, 加入该控件的第一幅图像决定了随后加入图像的显示大小。例如, 开始加入一个大小为 64×32 像素的图标, 那么其后的图像将显示相同的大小, 都为 64×32 像素的图标。

注意: 如 ImageList 控件中的图像用于 Image 控件时, 图像的大小将自动调整为该控件的大小。

在设计时, 可以在 ImageList 控件【属性页】对话框的【通用】选项卡中, 设置图像的高度和宽度, 以像素为单位。

将图像加入到 ImageList 控件之后, 要在 ToolBar 控件中使用这些图像, 还必须把它们和 ToolBar 控件相关联。

按以下步骤可完成 ImageList 和 ToolBar 控件相关联:

1. 用鼠标右键单击 ToolBar 控件, 并单击【属性】以显示该控件的【属性页】对话框。如图 9.4 所示。

2. 在【通用】选项卡的【图像列表】编辑框中选择 ImageList 控件的名称, 便完成了操作。

对上述内容熟悉之后, 我们就可以继续创建工具栏的第二步: 将图像赋给 Button 对象, 创建快捷按钮。

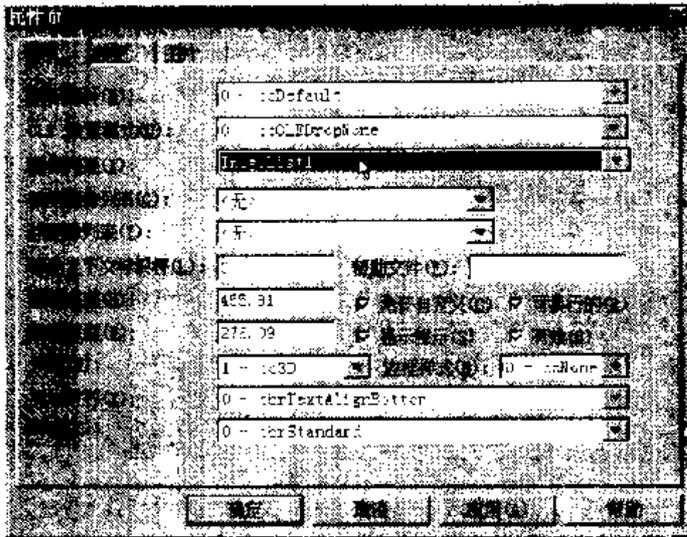


图 9.4 将 ImageList 与 ToolBar 控件相关联。

9.2.2 利用 ToolBar 控件创建工具栏快捷按钮

如前所述,利用 ToolBar 控件创建工具栏是很方便的,在完成了 9.2.1 节所述的工作之后,只需按照以下步骤即可在窗体上完成创建工具栏快捷按钮。

1. 单击工具框中 ToolBar 控件,将其放置在窗体中,即有一条工具栏出现在窗体标题栏下。
2. 用鼠标右键单击工具栏,在弹出菜单中选择【属性】菜单。在【属性页】对话框中,单击【按钮】选项卡,如图 9.5 所示。

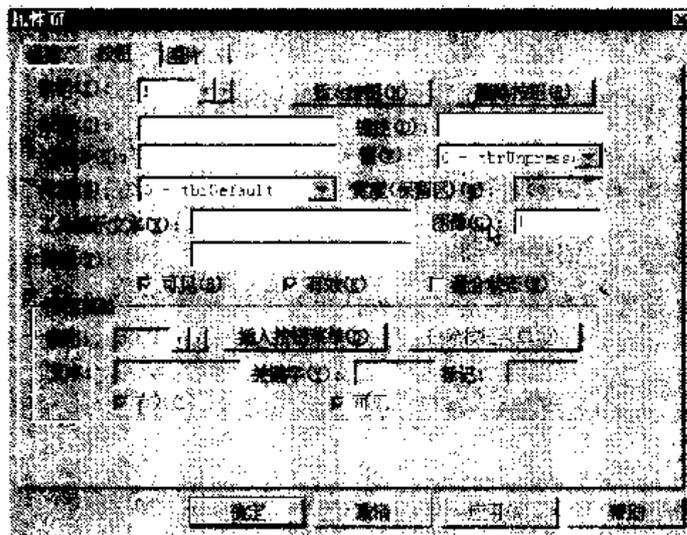


图 9.5 用【按钮】选项卡添加按钮对象并为其赋予图像

3. 单击【插入按钮】按钮以添加新的按钮对象。
4. 单击【图像】框,并键入 ImageList 对象的 Key 值。
5. 单击【应用】按钮。
6. 重复 2 至 4 步以添加更多的按钮,并将图像赋给新添加的 Button 对象。

提示：在【属性页】对话框的【通用】选项卡中，将工具栏的【外观】和【边框样式】属性都设置为 1，可以使工具栏更加突出、漂亮。

提示：在工具栏中的每个按钮还有【样式】属性，该属性决定了按钮的行为特点。在【属性页】对话框的【按钮】选项卡中可以设定按钮的【样式】属性。表 9.1 列出了五种按钮样式及其说明

表 9.1 五种按钮样式及其说明

样 式	值	说 明
tbrDefault	0	如果按钮所代表的功能不依赖于其他功能，使用 Default 按钮样式。如该按钮被按下，完成功能后它会自动弹回
tbrCheck	1	当按钮代表的功能是某种开关类型时，可使用 Check 样式。例如，在使用 RichTextBox 控件时，被选定的文本可被设置成粗体或非粗体。如果按下了该按钮，那么在再次按下该按钮之前，它将保持按下状态
tbrButtonGroup	2	当一组功能相互排斥时，可以使用 ButtonGroup 样式。相互排斥的意思是说一组功能同时只能有一个有效。例如，RichTextBox 控件中的文本只能是左对齐、右对齐或居中，在任何时刻都只有一种样式。注意：同一时刻只能按下一个按钮，但所有按钮可同时处于抬起状态
tbrSeparator	3	分隔符类型只是创建宽度为八个像素的按钮，此外没有任何功能。分隔符样式的按钮可以将其他按钮分隔开，或者用它将 ButtonGroup 样式的按钮封闭起来
tbrPlaceholder	4	占位符样式按钮的功能如同“哑”按钮：该按钮的作用是在 ToolBar 控件中占据一定位置，以便显示其他控件（如 Lombobox 控件或 ListBox 控件）

将按钮添加到工具栏并且为其赋予图像之后，剩下的工作就是为所有的按钮添加代码，以使用户单击该按钮时，执行相应的操作。

9.2.3 为工具栏编写代码

当用户单击按钮（占位符和分隔符样式的按钮除外）时，会发生 ButtonClick 事件。可以用按钮的 Index 属性或 Key 属性来识别单击的按钮。利用这些属性中的任意一个，可以用 Select Case 语句编写按钮的功能，下面用一个简单的例子加以说明。

图 9.6 所示为一个添加了三个按钮的窗体，三个按钮的关键词（Key）分别为“NewFile”、“OpenFile”和“SaveFile”，可用如下代码段为其编程：

```
Option Explicit
Private Sub ToolBar1.ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Key
        Case "NewFile"
            MsgBox "You clicked the NewFile Button."
```

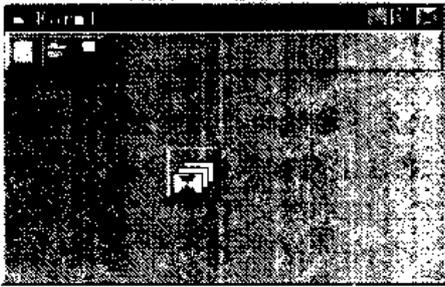


图 9.6 添加了工具按钮的窗体

```

Case "OpenFile"
    MsgBox "You clicked the OpenFile Button."
Case "SaveFile"
    MsgBox "You clicked the SaveFile Button."
End Select
End Sub

```

程序运行后,用户只要在任一个按钮上单击,就可打开一个消息框,提示用户按了哪一个按钮。

9.2.4 灵活使用 ToolBar 控件

ToolBar 控件除了创建工具栏很方便以外,还有许多其他的功能,可以大大减少程序员的工作量。

在程序运行时,双击工具栏就会显示【自定义工具栏】对话框,如图 9.7 所示。有了这个对话框,就可以重新安排工具栏按钮。



图 9.7 【自定义工具栏】对话框

通过对每个 Button 对象的 ToolTipText 描述进行编程,可以进一步增强程序的可用性。为此,必须将 ShowTips 属性设置为 True。当用户调用【自定义工具栏】对话框时,单击按钮就会导致在对话框中显示按钮的描述;这种描述可通过设置 Description 属性来编程实现。

如果 Wrappable 属性设置为 True,则当用户改变窗体的大小时 ToolBar 控件会自动折行。尽管 Button 对象能自动折行,但放置在它们上面的其他控件不能。

要使其他控件能够折行,则首先要创建 Placeholder 样式 Button 对象,然后在对应位置绘制其他控件,并在 Form 对象的 Resize 事件中用 Move 方法定位该控件,如下代码所示:

```

Private Sub Form_Resize()
'用 Move 方法将 ComboBox 覆盖在 Button 对象的位置上
with tbexap.Buttons("btnexap")
    cmbexap.Move .left, .Top, .Width
    cmbexap.Zorder 0
'Toolbar 控件名为"tbexap"
'Button 对象的 key 为"btnexap"
'ComboBox 名为"cmbexap"
End WithEnd Sub

```

在某些情况下,应用程序的函数可能返回不确定状态——返回的状态是两种或多种状态

的组合。例如,如果用户选中的 RichTextBox 控件中的文本,其中的一部分文本是粗体字的,另一部分非粗体,那么代表粗体的按钮既不能按下也不能抬起。为表示这种不确定状态,需要将该按钮的 MixedState 属性设置为 True,这样使按钮上的图像朦胧,以表示第三种状态。

9.2.5 手工创建工具栏

在不使用 ToolBar 控件的情况下,还可以手工创建工具栏。这听起来很困难,其实做起来是很简单的。

手工创建工具栏,一个常用的思路就是针对一个工具栏按钮,准备几个图像。起码有两个:一个是普通状态下的图案,一个是被按下时的图案。如果要做一个类似 Office 97 风格的工具栏,还需多准备一个鼠标指针停留时凸起的图案。这样,就可以针对鼠标不同的操作,显示不同的图案。

用户使用 Windows 95 / 98 自带的“画图”程序,就可以根据一个按钮的图案绘制出凸起或按下的状态,其实无非是添加深色边框和白色边框的位置不同罢了。

通常,大家都习惯使用 PictureBox 控件来作为工具栏按钮的面貌,而使用 CommandButton 或 Image 控件作为工具栏的按钮。下面介绍一下手工创建工具栏的操作步骤:

1. 在窗体上放置一个 PictureBox 控件,并将它的 Align 属性设置为 1,图片框的宽度会自动伸展,即可在窗体的标题栏下出现一个空白行。

2. 在这个图片框中,可以放置想在工具栏上显示的控件。

使用 Image 控件创建工具栏按钮,对于每一个要在工具栏中显示的按钮,至少要使用 3 个 Image 控件。一个显示在工具栏中,另外的负责保存按钮在不同状态下的图案。设计程序需要指定每个 Image 控件的 Picture 属性,以指定按钮在不同状态下的图像文件。

3. 指定好图像文件之后,还要将工具栏中按钮的 Visible 属性设置为 True,其余的 Image 控件的 Visible 属性设置为 False,使它们在运行时不可见。

4. 使用步骤 2 和 3,继续创建工具栏按钮。如果在图片框中没有位置再存放隐藏的 Image 控件,可以将它们放在窗体上。

5. 为工具栏按钮进行编码,因为是手工创建工具栏,所以按钮的动作必须依靠程序代码完成。假设我们将工具栏按钮命名为 imgBtn,2 个隐藏按钮命名为 imgBtn1, imgBtn2,它们的 Picture 属性分别表示按钮凹下和正常时的图像。下面给出基本的程序代码:

```
Private Sub imgBtn_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
'如果用户按下的是左键,显示按钮凹下的图像
If Button = 1 Then
    imgBtn.Picture = imgBtn1.Picture
End Sub
Private Sub imgBtn_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
'显示正常的按钮图像
imgBtn.Picture = imgBtn2.Picture
End Sub
```

以上只是个基本的程序框架,还有许多需要在实际应用中加以完善,并需添加代码,使按钮按下时执行一定的操作。

9.3 创建状态栏

使用 Microsoft Windows Common Control 6.0 中的 StatusBar 控件,可以为程序添加状态栏。StatusBar 控件可以创建一个窗口,通常位于窗体底部,也可以置于顶部或侧面。

StatusBar 控件是由 Panels 集合构成的。在该集合中至多可包含 16 个 Panel 对象,每个对象可以显示一个图像和文本。

在运行时,可以通过 Text、Picture 和 Width 属性动态地改变在向 Panel 对象的文本、图像或宽度。要在设计时添加 Panel 对象,可以用鼠标右键单击 StatusBar 控件,然后单击【属性】菜单项,即可打开【属性页】对话框,如图 9.8 所示。

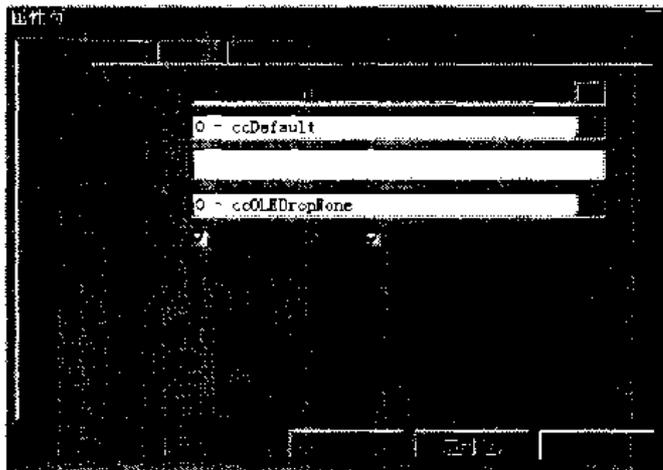


图 9.8 StatusBar【属性页】对话框

使用该对话框,既可以添加 Panel 对象,也可以设置每个对象的各种属性。用户也可以在运行时添加 Panel 对象,这需要使用带 Add 方法的 Set 语句。首先声明 Panel 类型的对象变量,然后将该对象变量设置为由 Add 方法创建的 Panel,如下代码所示:

```
'StatusBar 控件的名称为"sbTest"  
Dim pnTest As Panel  
Set pnTest = sbTest.panels.Add()
```

一旦创建了 Panel 对象之后,就可以用设置的对象变量引用该对象,并设置其各种属性:

```
pnTest.Text = Drive1.Drive  
pnTest.Picture = LoadPicture("mapnet.bmp")  
pnTest.Key = "drive"
```

其中,Key 属性必须是唯一的,它用来标明特定的对象。在单击了特定的 Panel 对象时,应用程序代码据此作出相应的响应。下面给出介绍。

要在 StatusBar 控件中用程序响应单击事件,可以在 PanelClick 事件中使用 SelectCase 语句。下面给出一段代码:

```
Private Sub sbTest_PanelClick(ByVal Panel As MSCornctlLib.Panel)  
    Select Case Panel.Key  
        Case "drive"
```

```

        Panel.Text = Drive1.Drive
    Case "openDB"
        Panel.Text = rsopenDB.Name
    Case Else
        '处理其他情况
    ...
End Select
End Sub

```

StatusBar 的每个 Panel 对象的外观,还可以使用 Bevel、AutoSize 和 Alignment 属性改变,下面分别予以介绍:

1. Bevel 属性指定 Panel 对象是否凹下、凸起或者水平。Bevel 属性的可用设置如表 9.2 所示。

表 9.2 Beval 属性值及其描述

常 数	值	指 定
SbrNoBevel	0	面板不显示斜面,这样文本就象显示在状态条上一样
SbrInSet	1	面板凹入状态条
SbrRaised	2	面板凸出状态条

2. AutoSize 属性决定当窗体或容器控件的大小改变时,Panel 对象本身的大小应该如何改变。AutoSize 属性的可用设置如表 9.3 所示。

表 9.3 AutoSize 属性值及其描述

常 数	值	描 述
SbrNoAutoSize	0	不会自动改变大小。该 Panel 的宽度始终精确地由 Width 属性指定
SbrSpring	1	可伸缩的,当窗体的大小改变,产生了多余的空间时,所有具有该设置的面板将均分空间并相应地变大,然而这些面板的宽度不会小于由 MinWidth 属性指定的宽度
SbrContents	2	面板的宽度与其内容匹配

提示: 如果想保证 Panel 中的内容始终可见,可将 Autosize 属性设置为 SbrContents。

3. Alignment 属性指定面板中的文本和图像在面板中如何对齐,类似于在文字处理器中,文本可以是左对齐、居中或右对齐的,Alignment 属性的可用设置如表 9.4 所示。

表 9.4 Alignment 属性值及其描述

常 数	值	描 述
SbrLeft	0	文本在位图的右侧,以左对齐方式显示
SbrCenter	1	文本在位图的右侧,以居中方式显示
SbrRight	2	文本在位图的左侧,以右对齐方式显示

以上介绍了如何用 StatusBar 创建状态栏,如同创建工具栏一样,用户也可以手工创建状态栏。为此,在窗体中添加一个 PictureBox 控件,将其 Align 属性设置为 2(Align Bottom),这会使得

图片框沿窗体底部边缘扩展。对于一个状态栏来说,这是较佳的位置。在这个图片框中,放入任何需要的控件,例如 Label 控件,然后添加代码改变其标题,当程序执行时会显示状态行信息。

9.4 本章小结

本章主要介绍了应用 VB 6.0 的 ActiveX 控件——ToolBar 控件和 StatusBar 控件创建应用程序的工具栏和状态栏。当然,也可以手工创建。读完本章读者应掌握以下内容:

1. 会向 ImageList 控件添加图像,并会设置图像属性。
2. 如何创建工具栏,并会 ImageList 控件的图像与工具按钮相关联。
3. 如何创建状态栏。

第 10 章 使用图形

图形要素是一个语言中最有趣的部分之一。VB 6.0 提供了许多操作图形的工具。利用它们,用户可以很方便地创建、添加及优化图形,这些工具将在本章予以介绍。

一般来说,图形可以分为两大类:矢量图形和位图图形。矢量图形是由图形命令产生的,而位图图形是显示在不同控件上的对像素进行处理的图形。两者的主要差别在于:矢量图形与具体的分辨率无关,它是由画图命令产生的,可以在不同分辨率中显示。而位图图形则与分辨率密切相关。分辨率的变化或图形的缩放都会对图形质量造成影响。本章介绍 VB 对这两种图形的操作工具,这两种图形各有所长,可以综合产生所要的效果。

下面首先介绍一下 VB 常用的图形控件。

10.1 使用图形控件

VB 6.0 提供了三种可以放置图形的控件:Form(窗体)、PictureBox(图片框)和 Image(图像)控件。对于窗体,大家在前面的章节中早已熟悉,PictureBox 和 Image 控件显示如图 10.1 所示。

这三个控件的重要差别在于:窗体和图片框控件提供了画图的方法,可以在运行时画图。而图像控件适合于显示图形,因为它比图片框控件占用的资源要少。

图片框控件提供了运行时画图的方法,因此更加灵活,但占用更多的资源。每个控件都提供了不同的属性,用于控制显示图形的外观。

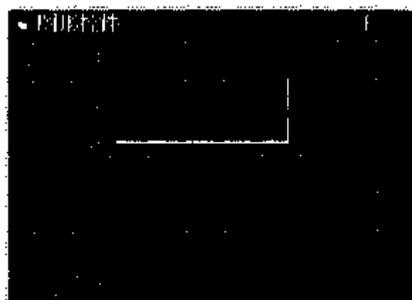


图 10.1 PictureBox 和 Image 控件

10.1.1 给应用程序添加图形

基本说来,可以有两种方法将图形放在控件上:设计时装入或运行时装入。下面分别予以介绍。

1. 设计时添加图片

在设计时将图片(位图或图标)放在控件上,要在【属性】窗口为图形文件命名,同时可指定控件的 Picture 属性,此时,应把图形和应用程序一起存放。VB 工程中,为每个窗体生成一个扩展名为 FRX 的文件,这是存放图形(实际位图)的地方,因此,应用程序的长度会增加。

在设计时添加图片有两种方法:

(1) 从图片文件中将图片加载到窗体上,图片框中或图片框控件里。

在【属性】窗口中,从【属性】列表中选择 Picture,并单击右边的显示按钮[...] (如图 10.2 所示),VB 将显示一个【加载图片】对话框,从中可选择要加载的图片文件。选定的图片就会显示在窗体或图片框、图像控件中。

(2) 把一个图片粘贴到窗体上,图片框或图像控件中。

无论是用位图图形还是用 VB 画图方法生成的图形,都要与 Windows 应用程序交互使用。最简单的方法是应用剪贴板。首先将一个图片从另一应用程序(如 Windows 的画图板)复制到

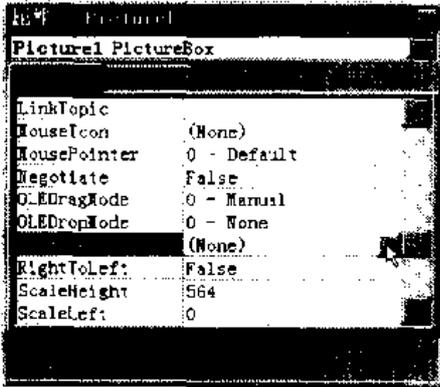


图 10.2 从【属性】列表中加载图片

方法的代码如下：

```
Form1.Picture = LoadPicture(fileName)
```

其中, fileName 变量是包含图形的文件名。这个文件的扩展名可以是 .BMP(位图)、.GIF(图形交换格式)、.JPG(联合图像组格式)、.DIB(设备无关位图)、.WMF(Windows 元文件)、.EMF(加强文件)或 ICO(图标)。目前, VB 还不支持其他类型的图形文件。

利用这种方法可以在任何需要的时候加载一个新图片到窗体、图片框或图像控件中, 新加载的图片将完全代替正在显示的图片。

注意: 如果用不带变元的 LoadPicture 方法, 则不会有图形装入, 并且当前控件中显示的内容将被清除, 因此, 要删除图片框或其他显示图形控件上的图形, 可以使用下列命令:

```
Picture1.Picture = LoadPicture()
```

它类似于 Cls 方法, 清除窗体或控件中的图形。

(2) 使用 LoadresPicture 函数, 可添加工程中的 .Res 文件的图片。

下列语句把资源文件里资源标号 Id 为 10 的位图, 加载到名为 Picresource 的图片框内:

```
Set Picresource.Picture = LoadresPicture(10, Vbresb, # map)
```

(3) 对象间图片的相互复制。

对象间图片可相互复制。例如, 下列语句将把名为 Picdisplay 图片框中的图片复制到名为 Imgdisplay 的图像控件内:

```
Set Imgdisplay.Picture = Picdisplay.picture
```

(4) 从剪贴板对象复制图片。

在设计时从文件中加载或粘贴图片, 则图片就和窗体一起被保存和加载, 应用程序可将图片从一个对象复制到其他对象。

3. 保存图形

如果用户希望保存图形, 可以用 SavePicture 语句。该语句的语法如下:

```
SavePicture Picture, filename
```

其中, Picture 变元是要保存其内容的图片框控件或图像控件的属性, filename 是存放图形

的文件名。

要将控件 Picture1 的内容保存,应用如下语句:

```
SavePicture Picture1.Picture,"C:\PictureTest.bmp"
```

即使装入的图形原来为 GIF 格式,SavePicture 方法仍会将图形存为 BMP 格式。此外,SavePicture 语句存放控件上的整个图形,包括不显示部分。

用 FileOpen 公共对话框提示用户输入或选择要打开的图形文件时,可用扩展名 BMP、GIF 或 JPG,但对应的 FileSave 公共对话框中,只能指定扩展名 BMP 或 DIB。

注意:如果用 SavePicture 语句存放图片框控件内容,而控件的 AutoRedraw 属性设置为 False,则结果可能是个空的 BMP 文件。如果要将图片框控件内容存到文件中,则一定要先将 AutoRedraw 属性设置为 True,然后再装入或生成图形。

4. 调整图形位置及大小

如果窗体、图片框或图像控件被移动(设计时或运行时),则其上的图片也将自动地随它一起移动。如果窗体、图片框或图像控件调整大小后太小,容纳不下图片,则该图片将从右边和底部被裁剪。如果图片被加载到过小的窗体、图片框或图像控件中而不能完整显示时,该图片也会被裁剪。

对图片框来说,与之相关的是 AutoSize 属性。如果图片框的 AutoSize 属性设置为 True,则图片框能自动扩展到可容纳新图片的大小。这样,在运行时当往图片框加载或复制图片时,VB 会自动扩展该控件到恰好能够显示整个图片。由于窗体不会自动改变大小,如果加载的图像超过窗体,图像将被裁剪。

对图像控件来说,与之相关的是 Stretch 属性。如果 Stretch 属性设置为 True,则图形会自动调整大小以适应图像控件的尺寸,但要注意的是,除非控件尺寸与图形尺寸的高宽比相同,图形才保持原貌;否则,在缩放图形时,图形会发生扭曲。如果 Stretch 属性设置为 False,则图像控件会自动调整大小以适应加载图片的要求。

10.1.2 窗体和控件的图形属性概述

窗体和各种控件都具有图形属性,表 10.1 列出了这些属性。

表 10.1 窗体和控件的图形属性

类别	属性
显示处理	AutoRedraw, ClipControls
当前绘图位置	CurrentX, CurrentY
绘图技术	DrawMode, DrawStyle, DrawWidth, BorderStyle, BorderWidth
填充技术	FillColor, FillStyle
颜色	BackColor, ForeColor, Bordercolor, FillColor

窗体和图片框还具有一些附加属性,主要的是 Scale 属性和 Font 属性。Scale 属性又可分为 4 个属性:ScaleHeight、ScaleWidth、ScaleLeft 和 ScaleTop。Font 属性返回一个 Font 对象,其语法为:

```
ObjectFont
```

Object 所在处代表一个对象表达式,其值是具有 Font 属性的一个对象。例如,下面的代码将改变一个 Font 对象的 Bold 属性设置,该 Font 对象被 PictureBox 对象的 Font 属性所标识:

```
PictureName.Font.Bold = True
```

下面具体介绍二个与显示处理有关的属性:AutoRedraw 属性和 ClipControls 属性。

1. 用 AutoRedraw 创建持久图形

每个窗体和图片框都具有 AutoRedraw 属性。AutoRedraw 是 Boolean 型属性,当它设置为 True 时,会把图形输出保存在控件中,因此可用 AutoRedraw 创建持久的图形。

Microsoft Windows 操作屏幕图像,造成重叠窗口的影像。当一个窗口移动到其他窗口上时,可暂时隐藏其他窗口;移动后,被覆盖的窗口及其内容需要重新显示。而用户的 VB 应用程序,必须控制窗体和图片框内图形的重新显示。

如果在窗体上用图形方法创建图形,通常希望它们重新显示以前的位置,这时可用 AutoRedraw 属性。当 AutoRedraw 属性设置为 False(这是它的缺省值)时,窗体上显示的任何一个用图形方法创建的图形如果被另一个窗口挡住,将会丢失。另外,如果扩大窗体,窗体边界外的图形也会丢失。

当窗体的 AutoRedraw 属性设置为 True 时,VB 会将图形存于内存中的一块“画板”上。应用程序复制此画板的内容,以便重新显示被其他窗口隐藏起来的图形。

图片框的 AutoRedraw 属性也有 True 和 False 两种设置。当设置为 True 时,VB 仅在内存中保存此图片框的可视内容。这是因为保存图片框内容的画板与图片框的大小是相同的,超出其外的图形被裁剪后,即使改变图片框的大小,也不能再显示。

也可将窗体及其所有图片框的 AutoRedraw 设置为 False 来节省内存。但那样的图形将不会自动成为持久性的,必须用代码管理所有图形的重画。

可将所有想要重画图形的代码,包括在窗体或图片框的 Paint 事件中,这种方式最适用于有限个易于重建的图形的情况。

无论何时需要重画窗体或图片框的一部分,都可调用 Paint 事件过程。当覆盖住对象的窗体移去后,可使原有图形恢复可视。

提示:在运行时也可以改变 AutoRedraw 的设置。如果 AutoRedraw 的设置为 False,图形只输出到屏幕,不到内存。当 AutoRedraw 设置为 True 时,如果用 Cls 方法清除对象,已有输出并不能被清除。这是因为输出保存在内存中,必须再次设置 AutoRedraw 为 False 时,才可用 Cls 方法清除。

2. 用 ClipControls 裁剪区域

每个窗体、图片框控件,都具有 ClipControls 属性。ClipControls 是 Boolean 属性,当它设置为 True 时导致该容器(窗体或框架控件)定义一个裁剪区域,用以绘制该容器中的许多控件。

将 ClipControls 属性设置为 False,则可提高窗体在屏幕上的绘制速度。对于具有许多不重叠控件的窗体,绘制速度的提高较为明显。Clipping 是当显示窗体或容器时,决定窗体或容器的哪部分被绘制的过程。而决定窗体或容器的哪部分被绘制所用的轮廓线(或称为“被剪裁”),就定义为 ClippingRegion。当基于 Windows 的应用程序需要保存部分显示内容,同时重画其余部分时,裁剪区域是十分有用的。当 ClipControls 属性为 True 时,Windows 在 Paint 事件之前,在窗体或容器的背景上定义一个裁剪区域,此裁剪区域围绕所有非图形控件。

在 Paint 事件期间,Windows 只重画裁剪区域的背景,而不重画非图形控件。

注意:该裁剪区域没有裁剪窗体上的标签或形状控件的边线。

当 ClipControls 设置为 False 时,Windows 不在事件之前给窗体或容器的背景定义裁剪区域。而且,Paint 事件中图形方法的输出,只显示窗体或控件必须重画的部分,因为计算和管理裁剪区域需要长时间。那么设置 ClipControls 为 False,就可以使含有许多非重叠控件的窗体(如复杂的对话框)显示得更快。

注意:应避免把 ClipControls 为 False 的控件嵌入到 ClipControls 为 False 的控件中。这样做会导致嵌入控件不能正确重画。把容器和控件的 ClipControls 都设置为 True,就可以避免这种情况发生。

10.2 坐标系统

描述一个图形操作,需要使用绘图区或容器的坐标系统,本章介绍坐标、坐标系统及其在绘图中的应用方法。

坐标描述一个像素在屏幕上的位置或打印纸上的点的位置。坐标系系统类似于市区地图,地图上的每个地段都有唯一的地址:引号和列号的组合,引号是水平坐标,也称 X 坐标,列号是垂直坐标,也称 Y 坐标,窗体上任何一点都可以用 X 和 Y 坐标表示,称它为坐标(x,y)处的点,或点(x,y)。

10.2.1 VB 的坐标系

坐标值从 0 开始,(0,0)点称为坐标系的原点。VB 中的原点为控件或窗体的左上角。X 坐标从左到右递增,Y 坐标从上到下递增。每个坐标是一个数字,但不一定对应有意义的单位。比如市区地图上的字母和数字就不对应有意义的单位,而是任意的。但拓扑图上的坐标就对应实际的距离。所有这些,都取决于具体的应用程序。

VB 支持多种坐标系,包括可以设置自己的单位的用户自定义坐标系。如果熟悉其他语言或操作系统中的计算机绘图方法,则用户就会知道,最常用的坐标系是基于像素的。但像素并不是所有应用程序中的最佳单位。如果用像素表示控件内容,则要受到具体分辨率的影响。如果计算机监视器的分辨率提高,则图形的显示效果会有很大的不同,甚至图形的比例也会不同。图形会出现毛刺甚至变形。

VB 缺省的坐标单位为 twip,它等于一点的 1/20。点是个印刷测量单位,1 英寸为 72 点,所以 1 英寸为 1440twip。twip 是个精确的测量单位,一般用不到这么高的精度,所以有时用 twip 不太方便,VB 还提供了 8 个坐标系,如表 10.2 所示。

表 10.2 VB 坐标系

数值	说明	常量名	大小
0	用户自定义坐标系	VbUser	无
1	Twips	vbTwips	每英寸 1440twip
3	Pixels(像素数)	vbPixel	无
4	字符数	vbCharacters	120twip 宽,240twip 高
5	英寸	vbInches	1440 twip

数值	说明	常量名	大小
6	毫米	vbMillimeters	
7	厘米	vbCentimeters	

要改变缺省的坐标系,将相应数值赋予 ScaleMode 属性即可。如果选择 ScaleMode 为 Inches,则控件上的距离必须指定为 1 英寸。这时相距 1 个单位的两个点之间相距 1 英寸;如为 0.1,则对应于 1/10 英寸。

注意: 改变 ScaleMode 并不影响控件的大小,只是改变控件上点的网格状分布密度。

如果预定坐标系都不适应具体需要,可以生成用户定义的坐标系。要建立适合应用程序需要的用户自定义坐标系,可以用相应的 Scale 方法和相关属性。

10.2.2 Scale 方法及其属性

Scale 方法及其属性是与控件位置和坐标系有关的属性和方法,基本说来,有两组属性:控制控件大小和位置的属性,以及与坐标系选择相关的属性。

1. Width 和 Height 属性

这两个属性确定控件的实际大小,它们总是表示为控件容器的单位。假如将一个图片框控件放在使用缺省坐标系 twip 的窗体上,则这个图片框控件的 Width 和 Height 属性表示为 twip。如果只是改变控件的坐标系,则这两个属性的值不变。如果改变窗体上的控件尺寸,则 Width 和 Height 的属性也会相应地发生变化。如果改变容器的坐标系,则 Width 和 Height 属性也会改变,从而反映控件在新坐标系中的尺寸。

2. Left 和 Top 属性

Left 和 Top 属性是控件左上角的坐标,用容器的坐标系表示。改变 Left 和 Top 的值时,控件位置改变。如果改变容器的坐标系则 Left 和 Top 属性也会随之改变,以反映控件在新坐标系中的位置。例如,下面这些语句给当前窗体的左上角和名为 Picture 的图片框左上角设定数值。

```
ScaleLeft = 100
ScaleTop = 100
Picture.ScaleLeft = 100
Picture.ScaleTop = 100
```

前二条语句虽不直接改变窗体的位置,但它使得图片框位于窗体的最顶端。

3. ScaleMode 属性

ScaleMode 属性用于设置或返回控件的当前坐标系。设置 ScaleMode 属性如表 10.2 中的数值之一时,可以建立新的坐标系。如果设置 ScaleMode 属性为 0,则还要设置 ScaleWidth 和 ScaleHeight 属性。反之,如果设置 ScaleWidth 和 ScaleHeight 属性,则 ScaleMode 属性值将复位为 0。

ScaleWidth 和 ScaleHeight 属性是当前坐标系单位的控件内部尺寸。改变坐标系时,并不改变控件大小,但会改变控件两个轴间的单位数。例如,窗体上宽为 1440twip、高为 1440twip 的图片框,大约为 1 英寸宽和 1 英寸高。其中 ScaleWidth 和 ScaleHeight 属性的值为 1440。如果将窗体的坐标系变为 Inches(英寸),则控件大小不变,但其 ScaleWidth 和 ScaleHeight 属性值将变为 1

(英寸)。

4. ScaleLeft 和 ScaleTop 属性

ScaleLeft 用于返回/设置对象左边界的水平坐标,ScaleTop 用于返回/设置对象上边界的垂直坐标,这两者共同表示用户定义坐标系中控件左上角的坐标。

5. Scale 方法

Scale 方法是建立用户坐标系的最方便的方法。其语法如下:

```
Form1.Scale(x1,y1)-(x2,y2)
```

该语句表示控件左上角坐标为(x1,y1),右下角坐标为(x2,y2),它通知 VB 控件的水平尺寸为(x2 - x1)个单位,垂直尺寸为(y2 - y1)单位。

例如,可以调用 Scale 方法如下:

```
Form1.Scale(0,0)-(10,10)
```

上述语句等效于下列赋值语句:

```
Form1.ScaleTop = 0
```

```
Form1.ScaleLeft = 0
```

```
Form1.ScaleWidth = 11
```

```
Form1.ScaleHeight = 7
```

提示:设置 Scale 属性即把 ScaleMode 复位为 0,所以不必显式设置 ScaleMode 属性,设置 ScaleWidth,ScaleHeight,ScaleTop,ScaleLeft 或调用 Scale 方法,均会把 ScaleMode 复位为 0。

所有以 Scale 开头的属性都使用用户定义坐标系。设置这些属性即切换到用户定义坐标系,但不改变控件和窗体的位置和大小。同样,每次发出 Scale 命令时,这四个属性值就变成了 Scale 命令的变元。

6. ScaleX 和 ScaleY 方法

使用 ScaleX 和 ScaleY 方法,将一种刻度模式转换成另一种刻度模式。例如,有时需要在给定坐标系中表示控件的新尺寸,而不改变容器的坐标系。

假设窗体的坐标系数为 t(twips),要在其上放一个 1.20 × 2.00 英寸的图片框控件。这时,就要先计算 1.20 和 2.00 英寸合多少 twips,然后再将这些值赋予控件的 Width 和 Height 属性。借助 VB 的 ScaleX 和 ScaleY 方法,可以在任何两个坐标系之间进行单位转换。

这两个方法的语法是一样的,如下:

```
Form1.ScaleX(Width,fromscale,toscale)
```

```
Form1.ScaleY(Height,fromscale,toscale)
```

其中 Width 和 Height 都是要转换的单位数,fromscale 是原单位所在的坐标系,toscale 是新单位所在的坐标系。假设 ImageBox 控件为 100 像素宽、100 像素高。此时,图形框所在的窗体采用缺省坐标系 twips,则 ImageBox 控件的 Width 和 Height 属性应表示为 twips。要将 100 像素数变成 twips,语句如下:

```
WidthTwips = Form1.ScaleX(100,vbPixels,vbTwips)
```

同样,要将 100 像素高变成 twips,语句如下:

```
HeightTwips = Form1.ScaleY(100,vbPixels,vbTwips)
```

然后可以利用上述结果设置控件尺寸：

```
Imagine.Width = WidthTwips  
Imagine.Height = HeightTwips
```

TwipsPerPixelX 和 TwipsPerPixelY 属性适用于 Screen 对象,它们分别返回对象中水平方向和垂直方向上每个像素的 twips 数。

7. CurrentX 和 CurrentY 属性

VB 绘图方法中的一个基本概念是当前点,在程序中,用户可以不指定起点而画一个图形(如直线)。如果没有指定起点,则当前点即变成直线的起点,线完成后,终点变为当前点。CurrentX 和 CurrentY 属性用当前坐标系单位设置和读取当前点的坐标。

10.3 绘图方法

除了图形控件之外,VB 还提供了一些绘图的方法,现总结在表 10.3 中,这些方法均适用于窗体和图形控件。

表 10.3 绘图方法

方 法	描 述
Cls	清除所有图形和 Print 输出
Point	返回指定点的颜色值
Print	显示一个字符串
Line	画线、矩形或填充域
Circle	画圆、椭圆或圆弧
PaintPicture	在任意位置画出图形

Print 方法与打印机毫无关系,它是指在窗体上或控件上绘制文本。Line 和 Circle 方法接受许多变元,扩展了 VB 的绘图功能,利用 Line 和 Circle 方法可以绘制大量几何图形。Point 和 Pset 方法对对象颜色进行操作,常用于图形处理程序和逐点绘制的曲线,PaintPicture 方法可以在任意位置绘出图形。

在许多情况下,使用 VB 的图形方法是很有效的。例如,在图表中要创建表格,需要用到直线控件数组,而使用 Line 方法时只需少量代码即可。当窗体改变大小时,跟踪数组中 Line 控件的位置,比用 Line 方法重新画线要麻烦得多。

当需要可视效果直接显示在窗体上时,如显示一些有关文本、色带时,可以为之编写几行代码,而不是使用另一控件。

另外,也是最关键的,图形方法提供了一些在图形控件无效的可视效果。例如,要创建位于其他控件之下的窗体层的图形时,这种方法就很好。

提示:用图形方法创建图形是在代码中进行的,这就意味着,必须运行应用程序才能看到图形方法的结果。因而,对于创建界面之类的设计来说,图形方法就不能代替图形控件的作用。因为设计时改变图形控件的外观,此修改比测试图形方法的代码要简单的多。

10.3.1 绘制文本

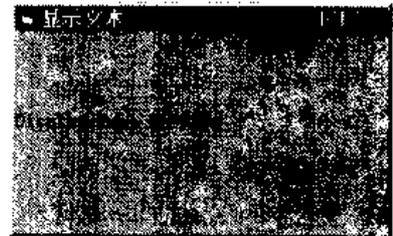
首先介绍一下 Print 方法。从当前点开始,在窗体或图片框控件上绘制文本。文本按控件的当前字体和字号绘制,然后当前点移动到文本末尾。

注意:到达控件右边时,长文本行并不自动换行。程序必须将文本行分成短行之后再绘制。

TextWidth 和 TextHeight 方法常用于将窗体对齐,或将图表框控件上的文本对齐。这两个方法均接受字符串变元,TextWidth 返回字符串宽度,TextHeight 返回字符串高度。但必须注意的是,它们只适用于窗体。在图片框控件上绘制文本,而且又要求与 TextWidth 和 TextHeight 属性值相一致时,必须使窗体的 Font 属性值与控件的 Font 属性值相同。

下面的一个简单应用程序,显示了如何用 Print 方法及 TextWidth 和 TextHeight 属性在窗体上绘制文本。要将字符串放在窗体中央,首先要计算窗体中央点 (Form1.Width/2, Form1.Height/2),将当前点设置为该坐标,以下是程序代码:

```
Private Sub Form_Load()
    Dim txtString As String
    txtString = "Display Text On Form"
    Form1.CurrentY = (Form1.Height)/2
    Form1.Print txtString
End Sub
```



程序运行后,结果如图 10.3 所示。

图 10.3 显示文本

10.3.2 画直线

VB 在两个端点之间画一条直线,使用 Line 方法,其语法如下:

```
Line[ Step ] ( x1, y1 ) - [ Step ] ( x2, y2 ),
[ color ], [ B ] [ F ]
```

直线的起点坐标为 (x1, y1), 终点坐标为 (x2, y2)。

其最简形式为:

```
Line(x1, y1) - (x2, y2)
```

直线的端点坐标表示为控件坐标系中的单位,还有两个属性决定了直线的特点:其宽度取决于 DrawWidth 属性,样式取决于 DrawStyle 属性。DrawStyle 属性显示于表 10.4。

表 10.4 DrawStyle 属性的取值

常量名	数值	说明
vbSolid	0	实线(缺省值)
vbDash	1	虚线
vbDot	2	点线
vbDashDot	3	点划线
vbDashDotDot	4	点线划
vbInvisible	5	透明线
vbInsideSolid	6	内实线

提示: 如果 DrawWidth 属性设置的线宽超过 1 个像素,则设置 DrawStyle 属性的 1 到 4 都是无效的,都被系统置为 0,因为这些线的线宽不能大于 1 个像素。当绘制线宽大于 1 个像素的直线时,VB 将线宽分布到指定坐标的两边。如果 DrawStyle 属性设置为 6,则整个形体(线、框、圆)都画在指定坐标内部。

下列简短的代码在窗体 Form1 上画出不同样式的线段,结果示于图 10.4 中。

```
Private Sub Form_Click()
    Dim y As Long
    Dim i As Integer
    For i = 0 to 6
        DrawStyle = i
        y = 400 + 200 * i
        Line(200, y) - (2400, y)
```

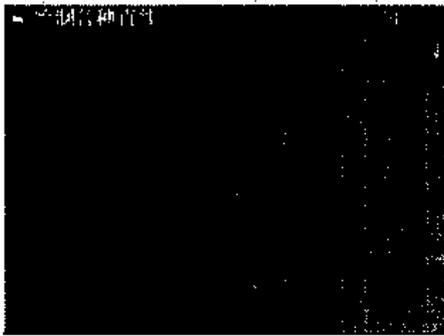


图 10.4 同 Line 方法画
各种样式的线段

```
Next
End Sub
```

图片框控件和窗体的 ForeColor 属性确定直线(或图)的颜色,也可以在 Line 或 Circle 方法中,用可选变元 Color 的不同颜色画线。下列语句显示了 Color 变元的用法:

```
Line(10,10)-(100,100),RGB(255,0,0)
Line(10,10)-(100,100),&H0000FF
Line(10,10)-(100,100),QBColor(3)
```

当以上几个语句都画一条从(10,10)到(100,100)的直线,而无论 ForeColor 属性设置为何值。关于 Color 变元的有效用法,将在本节后面“指定颜色”部分介绍。画直线时,如果不指定 Color 变元,则该直线仍沿用控件的 ForeColor 属性绘制。

利用 Line 方法的 Step 选项,可以相对于第一个端点定义第二个端点。Step 选项不是通过坐标定义端点,而是用离直线第一个端点的距离来定义点。换句话说,前面所用的坐标是绝对坐标,是指定屏幕上的唯一点。而 Step 选项后面的坐标是相对坐标,即选项的值是从当前点开始测量的。例如,下面的语句:

```
Line(10,10)-(100,100)
```

画一条直线,从(10,10)点开始,向下移动 90 个单位,向右移动 90 个单位。语句:

```
Line(10,10)-Step(100,100)
```

也是从(10,10)点开始,却是向下移动 100 个单位,向右移动 100 个单位。Step 语句后的数值不是一个屏幕上点的坐标,而是与当前点的距离。还可以在 Line 方法的第一个变元前面用 Step 关键词,这时直线的起点是当前点(CurrentX, CurrentY)。

Step 变元的一个重要作用是绘制闭合图形,因为相对于前一个端点定义后一个端点非常方便。假设要画一个矩形,其左上角坐标在点(100,100)处,用下面的简单代码即可实现。

```
Line(100,100)-Step(0,100)
Line-Step(100,0)
Line-Step(0,-100)
Line-Step(-100,0)
```

起点必须用绝对坐标定义,其他的点可以用相对坐标。

提示: VB 提供了更为方便的办法来画矩形。即在 Line 方法中用 B(BoxWindow)变元,矩形的左上角由第一个坐标定义,右下角由第二个坐标定义。上例中的四条语句可以用下面的一个语句代替。

```
Line(100,100)-(200,200),,B
```

注意: B 选项之前要有两个逗号,这表示色彩参数被忽略了,否则的话 VB 会把 B 看成指定方框颜色的变量名。

还有一种更为方便地画矩形的方法是同时应用 B 和 Step 选项。这时,只要知道左上角点的坐标和方框尺寸即可,例如:

Option Explicit

```
Private Sub Form_Click()
```

```
Line (400, 400) - Step(2000, 1500), , B
```

```
End Sub
```

程序运行后,用户只要在窗体上单击,结果就如图 10.5所示。

这一方法的好处就在于不需进行任何计算,只需输入方框的尺寸即可。

Line 方法还有另外一个选项,但只在选择了 B 选项之后才有用;即可以在 B 选项后面用 F(Fill)选项填充方框,注意 B 和 F 之间不要有逗号分隔。因为 F 选项无法单独使用,属性 FillColor 确定填充的颜色。BF 选项对一个方框改写 FillColor 属性,就像一个 Color 变元改写控件或窗体上直线的 ForeColor 属性一样。

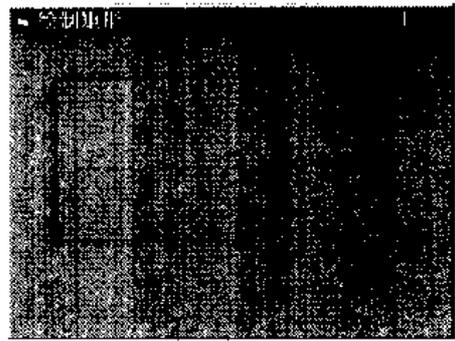


图 10.5 用 Line 方法绘制矩形

10.3.3 填充形体

根据 FillStyle 属性的不同设置,可以用不同的图案填充闭合形体,缺省情况下,FillStyle 设置为透明的。要用不同图案填充,需设置 FillStyle 的不同值,其取值如表 10.5 所示。

表 10.5 FillStyle 属性的值

常 量	数 值	说 明
VbFSSolid	0	实心,可用 FillColor 属性设置颜色
VbFSTransparent	1	透明(缺省)
VbHorizontalLine	2	水平线
VbVerticalLine	3	垂直线
VbUpwardDiagonal	4	向上对角线
VbDownwardDiagonal	5	向下对角线
VbCross	6	十字交叉线
VbDiagonalCross	7	对角交叉线

提示: 设置 FillStyle 属性值之后,闭合形体自动填充为指定图案,可以用 FillColor 属性确定图案的颜色。

提示: 如果 FillStyle 属性值为 1——透明,又要绘制实心颜色的方框,可以用 Line 方法的 BF 选项。图 10.6 显示了用不同的 FillStyle 属性值填充方框的情况。

程序代码如下:

Option Explicit

```
Private Sub Form_Click()
```

```
Dim I As Integer
```

```
For I = 0 To 3
```

```
FillStyle = I
```

```
Line (100 + I * 900, 100) - Step(700, 800), , B
```

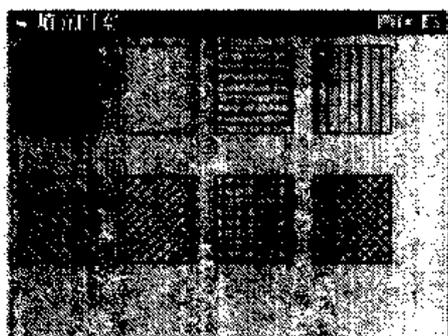


图 10.6 用不同的 FillStyle 属性填充方框

```

Next
For I = 4 To 7
    FillStyle = I
    Line (100 + (I - 4) * 900, 1300) - Step(700, 800), , B
Next
End Sub

```

程序运行后,用户只要在窗体上单击,结果就如图 10.6 所示。

10.3.4 使用 Circle 方法

Circle 方法可画出圆形和椭圆形,还可以画出圆弧和楔形饼块,使用变化的 Circle 方法,可画出多种曲线。

Circle 方法的完整语法如下:

```
Circle [Step](x,y), Radius, [Color], [Start], [End], [Aspect]
```

其中(x,y)为圆心坐标,radius 为半径,这些是必须的变元,方括号中为可选变元。

1. 画圆

用 Circle 方法的最简形式即可画出一个圆。下列语句在窗体 Form1 的中心画一个圆:

```
Circle(Form1.ScaleWidth/2,Form1.ScaleHeight/2),Form1.ScaleHeight/3
```

圆的圆心在窗体的中心,半径为窗体高度的 1/3,如果圆的范围超出窗体,则圆的某些部分将被裁剪。

Step 选项是用与当前点的距离来表示圆心坐标,值得注意的是,要用当前点为圆心画圆时,必须用下面的命令:

```
Circle Step (0,0),R
```

Step(0,0)是不能忽略的,这与 Line 方法不同。

注意:圆的半径是用水平轴的单位指定的。大多数坐标系中,水平轴和垂直轴的单位一致。但在用户自定义的坐标系中,两者可能会有不同。这时,圆是不会变形的,只是垂直轴所表示的长度就不对了。

下面一段简单的代码演示了用 Circle 方法画圆。程序运行后,用鼠标不断单击窗体,即会在窗体上出现许多不同颜色的同心圆,程序运行结果示于图 10.7。

```

Private Sub Form_Click()
    Dim Radius,R, G, B ,x, y
    '将红、绿、蓝设置为随机数
    R = 255 * Rnd
    G = 255 * Rnd
    B = 255 * Rnd
    '将圆心设在窗体中央
    X = ScaleWidth/2

```

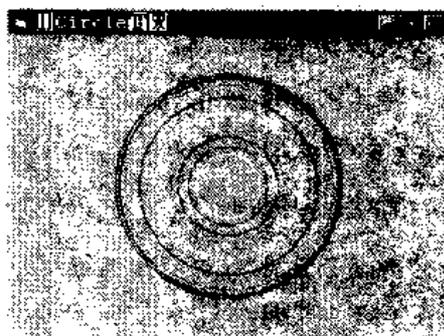


图 10.7 用 Circle 方法画圆

```

Y = ScaleHeight/2
'半径取随机数
Radius = ((Y * 0.9) + 1) * Rad
'用随机色、随机半径画圆
Circle(x, y), Radius, RGB(R, G, B)
End Sub

```

2. 画椭圆

Circle 方法中的 Aspect(高宽比)变元,可以用来画椭圆。高宽比是指椭圆的垂直半径与水平半径的比,可以为整数或浮点数。下面的程序,画了两个椭圆并显示了高宽比的定义。

```

Option Explicit
Private Sub Form1_Click()
Dim SideX, SideY, Side, XC, YC
'定义高宽比
SideX = 1
SideY = 0.75
Side = 3000
'定义显示大小
Form1.Line (300, 300) - Step(Side * SideX, Side * SideY), , B
'定义椭圆圆心坐标
XC = 300 + Side * SideX / 2
YC = 300 + Side * SideY / 2
'画椭圆
Form1.Circle (XC, YC), Side / 2, , , SideY / SideX
'显示高宽比定义
Form1.Line (XC, YC) - Step(Side * SideX / 2, 0)
Form1.Line (XC, YC) - Step(0, -Side * SideY / 2)
Form1.CurrentX = XC + Side * SideX / 10
Form1.CurrentY = YC + 20
Form1.Print "1"
Form1.CurrentX = XC + 20
Form1.CurrentY = YC - Side * SideY / 4
Form1.Print "0.75"
End Sub

```



图 10.8 用 Circle 方法画椭圆

程序的运行结果示于图 10.8。

注意: 如果 Aspect 参数值小于 1 的话,则 Radius 半径指的是水平方向的 X 半径。如果 Aspect 参数大于或等于 1 的话,Radius 半径指的是垂直方向的 Y 半径,即椭圆半径的含义是随着 Aspect 取值的不同而不同的。

3. 画弧

Circle 方法还可以用于画圆弧,Start 和 End 变元指定弧的始角和终角的弧度数。

弧的始角和终角是逆时针计算的,负角并不使弧的方法逆转,它如同正角一样画弧,然后再从圆心连接到对应的负角所在端点。

提示：由于圆和连接弧都是闭合形体，所以可用 FillStyle 指定图案和 FillColor 指定颜色填充。

10.3.5 画曲线

对于直线、圆等规则的几何形状，VB 可以轻易地完成。那么对于任意一条数字曲线呢？这时就需要用程序画出曲线上的每一个点，可以想象，画曲线将会困难一些。

以一个简单的数字函数 SinX 曲线为例加以说明，要逐点绘出曲线，可以使用 VB 中的 Pset 方法。

在画曲线之前，首先必须确定用户坐标系。假设我们画 0~20 之间的 SinX 的曲线，X 轴是自变量，从 0 到 20；Y 轴则必须先求出函数的最大值和最小值之后再建立（当然，对 SinX 来说，其取值范围是显而易见的，这里的做法是对一般函数来说的）。

要画一条曲线，需要对 X 值计算函数值，并画出对应的点。下面是画出该曲线的代码段：

```
Option Explicit
'计算函数值
Function FuncSin(ByVal X As Double) As Double
FuncSin = Sin(X)
End Function
'将曲线画在一个图片框中
Private Sub Picture1_Click()
Dim XPixels, XMin, XMax, i, t, YMax, YMin
Dim funcval As Double
'将图片框控件的 Scale Mode 属性切换到(像素)
Picture1.ScaleMode = 3
'计算图片框控件的像素点
XPixels = Picture1.ScaleWidth - 1
'定义 X 的取值范围
XMin = 0
XMax = 20
'计算 SinX 的最大最小值
For i = 1 To XPixels
t = XMin + (XMax - XMin) * i / XPixels
funcval = FuncSin(t)
If funcval > YMax Then YMax = funcval
If funcval < YMin Then YMin = funcval
Next
'设置自定义的坐标系
Picture1.Scale (XMin, YMin) - (XMax, YMax)
'画出曲线
For i = 0 To XPixels
'将像素点映射到函数自变量值
t = XMin + (XMax - XMin) * i / XPixels
Picture1.PSet (t, FuncSin(t))
```

```
Next  
End Sub
```

该代码段的运行结果如图 10.9 所示。

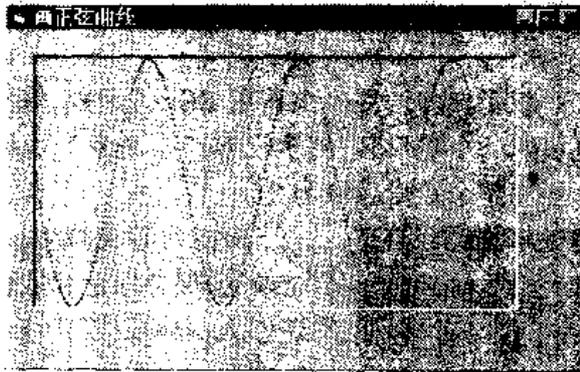


图 10.9 用 Pset 方法画曲线

10.4 在图形中使用颜色

前几节中,我们讲到了 VB 中两个操作像素的方法 Pset 和 Point。如前所述,Pset 打开一个像素,Point 读取像素取值,操作像素的方法是很灵活的,可以完成大量的工作。这一节中主要介绍一下 VB 是如何操作颜色值的。

10.4.1 定义颜色

对于一台计算机而言,指定颜色是很复杂的。如果通过 Color 公共对话框指定颜色值,就会看到红、绿、蓝三个对话框,这是计算机用于指定颜色的三种基本色彩。计算机监视器可以显示的任何颜色,都可以表示为红、绿、蓝的组合值。通过这三原色的配合,可以显示几乎所有的颜色。

可以根据红、绿、蓝三色的饱和度来指定颜色,这称为 RGB 模型,任何颜色都是这三种基本色的适当百分比的组合。因此,任何颜色都可用表示红、绿、蓝的三元组表示。最小值 0 表示没有用该色,最大值 255 表示最高的饱和度。例如,(0,0,0)表示黑色,什么颜色也没有,(255,255,255)表示白色,(255,0,0)表示纯红色等等。

注意:尽管可以指定的颜色有 $256 \times 256 \times 256$ 约为 1600 万种,但其灰度等级只有 256 级,因为灰度色调是由等量的基本色构件组成的,即三原色的值始终是相同的。所以,灰度图形不需用真色彩文件格式存放,只要用 256 种颜色的文件格式就可以了。

VB 定义了 RGB()函数以定义颜色。它的语法为:

```
RGB(Red, Green, Blue)
```

利用 RGB()函数,可以定义几乎所有的颜色。例如下面的语句将变量 newcolor 定义为纯黄色:

```
newcolor = RGB(255,255,0)
```

10.4.2 定义渐变

如上所说,计算机上的任何颜色都是由红、绿、蓝三种基本色构件组成。如果我们固定其中两个构件的值不变,而把另一个构件的饱和度从 0 到 255 变化,所得到的这 256 种类似的颜色称为颜色的渐变。

利用 RGB 函数可以产生渐变,这时,需多次调用 RGB()函数,每次应用稍有变化的变元。例如,如果要用渐变的颜色填满一个矩形区,由于没有产生渐变的函数,所有要用稍有不同的颜色值画水平线或垂直线,画满整个矩形区域为止。

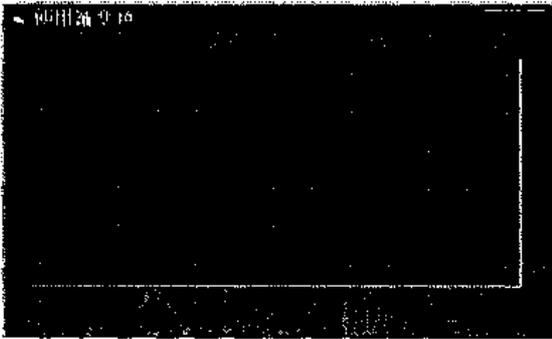


图 10.10 用渐变色填充图片框

```
Startgreen = 100; Endred = 255
PicWidth = Picture1.ScaleWidth
redAdd = (Endred - Startred) / PicWidth
For XPixel = 0 To PicWidth - 1
    newColor = RGB(Startred + redAdd * XPixel, Startgreen, Startblue)
    Picture1.Line (XPixel, 0) - (XPixel, Picture1.Height - 1), newColor
Next
End Sub
```

上述代码运行后,在图片框内单击,结果就如图 10.10 所示。

10.4.3 取得颜色值

在有些图形处理的程序中,应用程序需要读取某一点的像素值,分离红、绿、蓝三色的构成,然后分别进行处理。我们知道,RGB()函数组合三种基本颜色,在内存中这个函数值是以一个长整型的颜色值存放的,VB 没有提供一个可以返回三种颜色构成的函数。因此,必须自己编写函数,返回红、绿、蓝三原色的构成值。

要做到这一点,必须了解三种颜色构成是如何存放在长整型值中的。颜色值的长整型值是由四个字节构成的,其最高位为 0,次高位存放蓝色的值,次低位存放绿色值,最低位存放红色值。例如对于三元组(18,24,56),其左内存中的长整型值为 H00121838(H 代表 16 进制数)。下面的程序段可从一个像素的颜色值中取出三原色的构成值,分别存放在 red,green,blue 三个变量中:

```
Pixel = Form1.Picture1.Point(i,j)
red = Pixel Mod 256
```

下面的代码完成以上功能,在一个图片框中填充渐变,保持绿色和蓝色两个基本色构件不变,使红色从 Startred 变到 Endred。每次变化都在图片框中画一条垂直线,用 redAdd 控制增加量,代码如下:

```
Option Explicit
Private Sub Picture1_Click()
    Dim PicWidth, redAdd, newColor, XPixel
    Dim Startred, Startblue, Startgreen, Endred
    Startred = 0; Startblue = 100
```

```
green = ((Pixel & HFF00FF00) / 256)
```

```
blue = (Pixel & HFF0000) / 65536
```

由于红色位于最低字节, 所以其值就是像素值除以 256 的余数; 绿色值位于次低字节, 将像素值与 HFF00FF00 相与, 即可屏蔽其他字节(注意像素值的最高位始终为零), 再除以 256 即得绿色值; 蓝色值的取得也是同样的道理。

10.5 本章小结

本章前三节介绍了绘制图形的方法和技术, 尽管图形方法有限, 但加上图形控件和窗体的各个属性, 就使其变得相当强大和灵活。第四节介绍了 VB 操作像素的方法, 主要是如何生成像素值和取得像素值。利用它们, 可对窗体和控件上的每一个像素点灵活操作, 也可以方便地完成许多任务。具体说来, 学完本章后, 读者应掌握以下内容:

1. 使用图形控件给应用程序添加图片;
2. 会用窗体和图形控件的几个主要属性;
3. 了解坐标系统的基本概念;
4. 使用图形方法绘制各种图形, 包括显示文本, 画直线、矩形、圆、椭圆等;
5. 在图形上使用颜色。

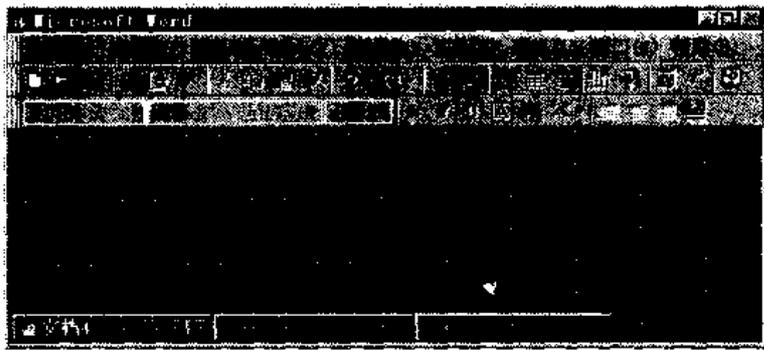


图 11.2 最小化了的子窗体显示在 MDI 窗体工作空间

1. 创建 MDI 窗体。从【工程】菜单中选择【添加 MDI 窗体】菜单命令,系统打开【添加 MDI 窗体】对话框,如图 11.3 所示。

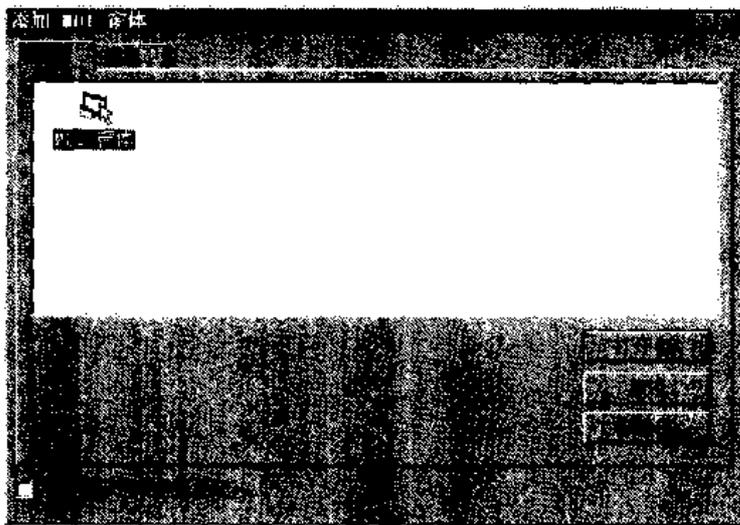


图 11.3 【添加 MDI 窗体】对话框

注意: 一个应用程序中只能有一个 MDI 窗体。如果向工程中添加了一个 MDI 窗体,则【工程】菜单上的【添加 MDI 窗体】项将呈灰色显示,不可用。

2. 选中【MDI 窗体】图标,单击【打开】按钮,将 MDI 窗体的 Caption 属性设为“MDI 窗体”。

3. 创建应用程序的子窗体。要创建一个 MDI 子窗体,先创建一个新窗体(或者打开一个已存在的窗体),将其 Caption 属性设为“子窗体”,并将窗体的 MDIChild 属性设置为 True,就使它变为一个子窗体。

VB 自动将子窗体和父窗体相联系。子窗体只能在父窗体中打开。按上述步骤生成的 MDI 程序还没有任何功能,如果这时运行程序,VB 会提示要指定程序的启动窗体。

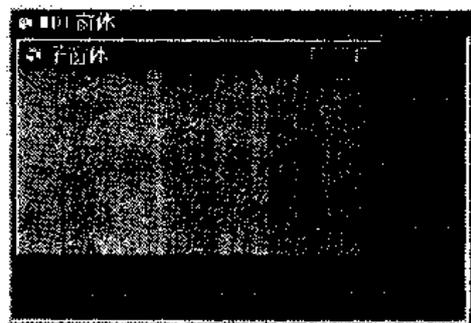


图 11.4 MDI 窗体及其子窗体

4. 从【工程】菜单中选择【属性】项,打开【工程属性】对话框,设置子窗体为启动窗体。如果此时设置 MDI 窗体为启动窗体,则子窗体将不会显示,必须在程序中装入子窗体。此时自运行程序,将得到图 11.4 的显示结果。

如果单击子窗体的最大化按钮,则两个窗体将重合,窗口的标题变成了父窗口的标题加上子窗口标题。还可以将子窗体移出父窗体,此时父窗体会自动加上相应的滚动条,并且子窗体的移出部分不予显示。

11.2 多文档界面的设计

上面一节我们介绍了 MDI 应用程序的基本操作。这些功能是 VB 中自身包含的,可以通过属性的设置提供给应用程序。

下面再举一个简单的例子,以便更好地了解 MDI 应用程序。我们使用【工程】菜单上的【添加窗体】菜单项向上一节完成的 MDI 窗体上再加入一个子窗体,将其 MDIChild 属性设置为 True, Caption 属性设置为:“子窗体 2”,并将第一个子窗体的 Caption 属性设置为:“子窗体 1”。

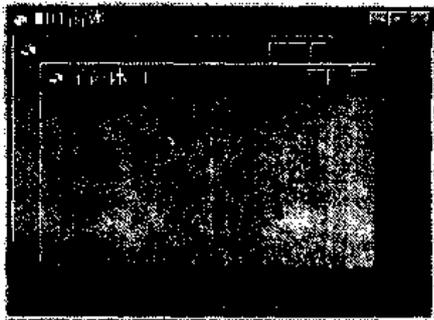


图 11.5 多子窗体的 MDI 窗体

如果这时运行程序,第二个子窗体是看不到的,需要在程序控制之下打开或关闭子窗体。要显示第二个子窗体,需要在 MDI 窗体的 Load 事件中加入下列代码:

```
Form2.Show
```

这时,MDI 窗体装入时两子窗体就都可以得到如图 11.5 的显示运行结果,还可以用以上方法加入多个子窗体,并且全都打开它们。

下面我们具体介绍 MDI 应用程序菜单的设计及窗体的使用。

11.2.1 菜单设计

MDI 窗体上不能放置控件,只能包含子窗体,但 MDI 有自己的菜单。每一个子窗体也有自己的菜单,并且彼此可能不同。在装入子窗体时,子窗体上的菜单将代替 MDI 窗体上原有的菜单。下面通过设计一个具有简单菜单的 MDI 窗体,来进一步说明这个问题。

开始一个新工程,加入一个 MDI 窗体,并设计如下结构的菜单(设计菜单的详细步骤参见第 8 章“菜单设计”):

- MDIMenu 菜单标题
- ……MDIOpen 打开新的子窗体
- ……MDIExit 退出应用程序

然后对子窗体设计下列菜单:

- ChildMenu 菜单标题
- ……ChildOpen 打开
- ……ChildClose 关闭

如果这时运行应用程序,则会看到子窗体的菜单【打开】和【关闭】出现在 MDI 窗体的菜单条上,如图 11.6 所示。关闭子窗体,则重新出现 MDI 窗体的菜单,如图 11.7 所示。

注意: MDI 菜单和各个子窗体上的不同菜单会给用户带来许多麻烦,在设计 MDI 应用程序菜单时应当注意到这一点。因为应用程序的大多数操作只有在至少打开了一个子窗体时才能实现,所以通常的作法是只在 MDI 窗体中加入有限的公共菜单项,并且这些菜单项在子窗

体的菜单中。

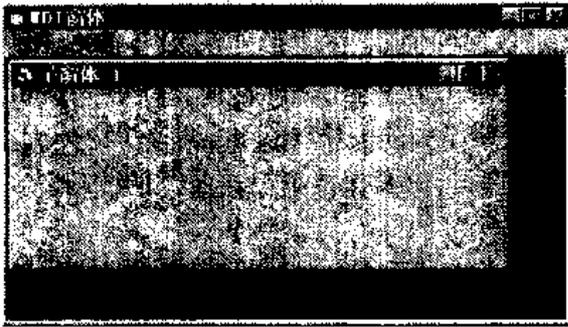


图 11.6 MDI窗体显示子窗体的菜单



图 11.7 子窗体关闭时,显示 MDI窗体的菜单

还可以在 MDI 窗体中创建一个【窗口】菜单(Window)。大多数 MDI 应用程序(例如 Microsoft Word 和 Microsoft Excel)都有【窗口】菜单,这是一个显示所有打开的子窗体标题的特殊菜单,如图 11.8 是 VB 6.0 的【窗口】菜单。在这个菜单中,还可以放置一些操纵子窗体的命令。

在 MDI 窗体或者 MDI 子窗体上的任何菜单控件,只要将其 WindowList 属性设置为 True,都可以用于显示打开子窗体的清单。在运行时,VB 自动管理与显示标题清单,并在当前激活的标题旁边显示一个复选标志。

说明: WindowList 属性应用于菜单控件,它的值决定菜单对象是否维护 MDI 对象中当前子窗体的列表。WindowList 属性的设置值及其描述,如下表所示:

设置值	描述
True	表示菜单对象维护一个已打开窗口的列表并显示一个活动窗口的复选标志。单击窗口名可以激活该窗口。
False	(缺省值)菜单对象不维护已打开窗口的列表。

按下面的步骤,可以设置菜单的 WindowList 属性:

1. 选取所需的窗体,如子窗体 1,再从【工具】菜单中,选择【菜单编辑器】菜单项。

注意: WindowList 属性只应用于 MDI 窗体或 MDI 子窗体。

2. 在【菜单编辑器】列表框中,选取或建立所需的菜单,如【窗口】菜单。

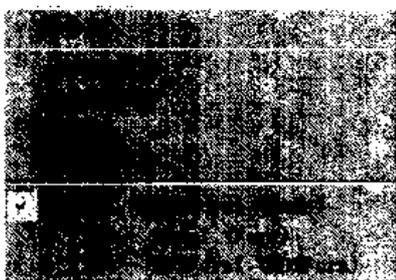


图 11.8 MDI中的【窗口】菜单

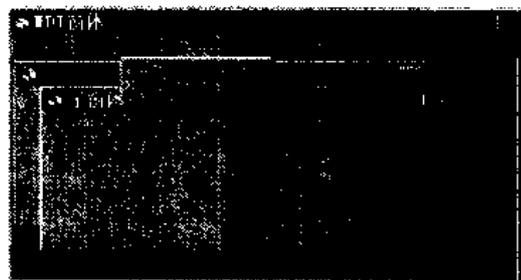


图 11.9 【窗口】菜单显示窗口列表

3. 选择该菜单的【显示窗口列表】复选框。

在运行时,这个菜单显示打开的子窗体的清单,如图 11.9 所示。另外,这个菜单控件的

WindowList 属性返回 True。

11.2.2 使用 MDI 窗体及其子窗体

当 MDI 应用程序在一次运行中要打开、关闭和保存几个子窗体时,它应当能够完成引用活动窗体和保持关于子窗体的信息的功能。这一节介绍了如何安排子窗体、调整子窗体大小、指定活动窗体或者控件、加载和卸载 MDI 窗体及其子窗体,以及保持子窗体信息的技巧。

1. Arrange 方法

安排 MDI 窗体上的窗口,需要进行编程。Windows 提供了三种在 MDI 窗体排列子窗体的方法——可以层叠、垂直平铺或者水平平铺。子窗口在窗体上的排列可用 Arrange 方法实现,其语法为:

```
Object.Arrange arrangement
```

各个部分的含义如下:

部分 描述

object 必需的,是一个对象表达式

arrangement 必需的,是一个数值或常数,它指定如何重排 MDI 中的窗口或图标

其中, arrangement 的设置如表 11.1 所示。

表 11.1 arrangement 的设置值及其说明

常 量	数 值	说 明
VbCascade	0	层叠所有非最小化 MDI 子窗体
VbTileHorizontal	1	水平平铺非最小化 MDI 子窗体
VbTileVertical	2	垂直平铺非最小化 MDI 子窗体
VbArrangeIcons	3	重排最小化 MDI 子窗体的图标

下面的例子显示了 Arrange 方法的用法。首先建立一个 MDI 窗体(MDIForm1),并添加一个 MDI 子窗体(Form1)和一个图片框(Picture)。输入以下代码:

```
Const FORMCOUNT = 5
Dim F(1 TO FORMCOUNT) As NewForm1
Private Sub MDIForm1_Load( )
Dim K 'K 为变量
Load Form1 '加载 Form1 窗体
For K = 1 To FORMCOUNT
    F(K).Caption = "Form"& K + 1 '改变标题
Next
End Sub
Private Sub Picture1_Click( )
DIM K,Pwidth, Start
Static ClickCount
ClickCount = ClickCount + 1 '控制排列方法
Select Case ClickCount
Case 1
```

```

MDIForm1.Arrange 1 '水平平铺
Case 2
MDIForm1.Arrange 2 '垂直平铺
ClickCount = 0
End Select
End Sub

```

运行上述应用程序,并单击 Picture1 的任何地方,即可查看 Arrange 方法的效果。

2. 设置子窗体的大小和位置

如果 MDI 子窗体具有大小可变的边框(即 `BorderStyle = 2`),其初始大小与位置取决于 MDI 窗体的大小,而不是设计时子窗体的大小。当 MDI 子窗体的边框大小不可变(即 `BorderStyle = 0, 1` 或 `3`)时,则它将用设计时的 `Height` 和 `Width` 属性被载入。

如果设置 `AutoShowChildren` 为 `False`,则在 MDI 子窗体载入以后,可以改变其位置。有关 `AutoShowChildren` 属性的信息,将在下面介绍。

3. 指定活动子窗体或控件

有时我们想要对当前活动子窗体上被激活的控件进行操作。例如,假设有子窗体的文本框中把所选文本复制到剪贴板上,【编辑】菜单上的【复制】项的 `Click` 事件将会调用 `EditCopyProc`,它把选定的文本复制到剪贴板上。由于应用程序可以有同一子窗体的许多实例,`EditCopyProc` 需要知道使用的是哪一个窗体,这就需要程序指定活动子窗体。为了描写这一点,可使用 MDI 窗体的 `ActiveForm` 属性,该属性可以返回具有焦点或最后被激活的子窗体。

注意: 当访问 `ActiveForm` 属性时,至少应有一个 MDI 子窗体被加载或可见,否则会返回一个错误。

当一个窗体中有几个控件时,也需要指定哪个控件是活动的。像 `ActiveForm` 属性一样,`ActiveControl` 属性能返回活动子窗体上具有焦点的控件。下面是一段代码,从子窗体菜单,MDI 窗体菜单或工具栏按钮上可对它进行调用,它将当前活动窗体上具有焦点的控件内容复制到剪贴板上。

```

Private Sub EditCopyProc( )
'将选定的文本复制到剪贴板上
Clipboard.SetText( = frmMDI.ActiveForm.ActiveControl.Text
End sub

```

假如正在编写被多个窗体实例调用的代码,不用窗体标识符访问窗体的控件或属性是一个好办法。例如,用 `Text1.Height` 引用 `Form1` 上的文本框的高度,而不是使用 `Form1.Text1.Height`,该代码总是影响当前窗体。使用 MDI 和子窗体时,另一个有用的关键字是 `Me`,它表示活动子窗体。如果把 MDI 窗体看成是子窗体的桌面环境,则 `Me` 关键字等于 `Screen` 对象的 `ActiveForm` 属性(表示活动窗体)。`Me` 是 MDI 窗体中的活动子窗体,可以通过 `Me` 属性得出其属性值。

但是,大多数的情况下,程序中要知道活动子窗体的索引号。例如,不能用 `Me` 关键字确定特定的子窗体是否打开。VB 并不报告活动窗口,所以必须设计跟踪活动窗口的方法。具体的办法是用窗体的 `Tag` 属性。每次装入新的窗体时,可以将其索引值放在窗体的 `Tag` 属性中:

```
Form(i).Tag = i
```

卸载窗体时,VB会自动复位其标志。窗体装入时,可以用窗体的 Tag 属性找出活动子窗体的索引号,并访问其属性和方法。

4. 加载 MDI 窗体及其子窗体

加载子窗体时,其父窗体(MDI 窗体)会自动加载并显示。而加载 MDI 窗体时,其子窗体并不会自动加载。

当将子窗体设置为缺省的启动窗体时,程序一旦运行,子窗体和 MDI 窗体两者都会被加载。如果在程序中设置 MDI 窗体为启动窗体,则程序运行时只有 MDI 窗体被加载。要加载子窗体,必须通过命令加载才行。

AutoShowChildren 属性用来加载隐藏状态的 MDI 子窗口,使它们处于隐藏状态,直至用 Show 方法把它们显示出来。Show 方法前面已经提到过,这里不再赘述,只介绍一下 AutoShowChildren 属性及其用法。

AutoShowChildren 应用于 MDIForm 对象,返回或设置一个值,确定在加载 MDI 子窗体时是否显示它。其语法为:

```
Object.AutoShowChildren[ = boolean]
```

各部分的描述如下:

组成部分	描述
object	对象表达式,指一个 MDI 窗体
boolean	布尔表达式,指定 MDI 子窗体是否自动可见

boolean 的设置值如下:

设置值	描述
True	(缺省值)MDI 子窗体在加载时自动显示
False	MDI 子窗体在加载时不自动显示

我们可用该属性加载隐藏状态的 MDI 子窗口,这样就允许在子窗体变成可见之前更新标题、位置和菜单等各种细节。

提示: 不能把 MDI 子窗体或者 MDI 窗体显示为横式窗体(用带 VbModal 参数的 Show 方法)。如果想在 MDI 应用程序中使用模态对话框,可使用 MDIChild 属性为 False 的窗体。

5. 维护子窗体的状态信息

在用户决定退出 MDI 应用程序之前,必须保护用户更改的信息。要做到这一点,应用程序必须随时都能确定自上次保存以来子窗体中的数据是否有所改变。

很容易想到,通过在每个子窗体中声明一个公用变量即可实现此功能。例如,可以在子窗体的声明部分声明一个变量:

```
Public TextChange As Boolean
```

这样,窗体中的文本每一次改变时,子窗体文本框的 Change 事件就会将 TextChange 设置为 True。

可添加如下代码,以指示自上次保存以来窗体 Form1 的内容有所更改:

```
Public Sub Form1_Change()  
TextChange = True  
End Sub
```

反之,用户每次保存过子窗体的内容后,文本框的 Change 事件就将 TextChange 设置为 False,以指示 Form1 中的内容不再需要保存。在下列代码中,假设有一个【保存】(MnuFileSave)的菜单命令和一个用来保存文本框内容的名为 FileSave 的过程。单击【保存】菜单项,调用如下代码:

```
Sub mnuFileSave_Click()  
    '保存 Form1 中的内容  
    FileSave  
    '设置状态变量  
    TextChange = False  
End Sub
```

当用户决定退出 MDI 程序时,就可以通过 TextChange 作用来决定是否要保存窗体信息。

6. QueryUnload 事件

QueryUnload 事件是 MDI 提供的在卸载子窗体时触发的事件。如果卸载整个 MDI 窗体,则触发所有子窗体的 QueryUnload 事件。利用该事件和以上所述的 TextChange 变量,可以容易地实现在卸载子窗体时保存更改信息。

QueryUnload 的语法如下:

```
QueryUnload(Cancel As Integer, Unloadmode As Integer)
```

应用程序可以设置 Cancel 变量,以终止处理进程。如果程序中设置 Cancel 变量为 True,则不卸载子窗体,从而 MDI 窗体也不卸载。

Unloadmode 变量说明哪个事件导致 QueryUnload 事件触发,即使关闭了 Windows 本身,也会触发 QueryUnload 事件。通过子窗体的 QueryUnload 事件代码,保证除了计算机崩溃或用户复位计算机之外,不会丢失应用程序的数据。

可以从程序中判断这个变量的值,检查窗体卸载的原因,然后采取相应的操作。有些情况下,根据导致窗体卸载的外部事件的不同,可能要采取不同的措施。但实际过程的本质是相同的,即向用户提供保存数据、放弃数据或取消程序操作的机会。

要使用 QueryUnload 事件卸载子窗体,可以在子窗体的 File 菜单的 Exit 菜单项中,加入以下代码:

```
Private Sub FileExit_Click()  
    Unload MDIForm1  
End  
End Sub
```

注意:这是子窗体中 File 菜单的 Exit 命令。MDI 窗体的 File 菜单只在没有打开子窗口时才会显示。这时没有要保存的编辑结果,一条 End 语句就足够了。要终止 MDI 应用程序,则首先卸载 MDI 窗体,然后再结束程序。

卸载 MDI 窗体时,即触发了所有子窗体的 QueryUnload 事件。在每个子窗体的 QueryUnload 事件中,可以提示用户,使其有机会保存数据、放弃数据或取消操作。具体的事件处理代码如下:

```
Private Sub Form_QueryUnload(Cancel As Integer, Unloadmode As Integer)
```

```

Dim replay As Integer
reply = MsgBox("Are you sure you want to close the document?" & Me.Tag, vbYesNoCancel + vbInformation)
If reply = vbCancel Then
Cancel = True
ElseIf reply = vbYes Then
Exit Sub
Else
FileSave_Click()
End If
End
Sub

```

这几行代码允许用户放弃未保存数据、取消操作或调用子窗体的 FileSave_Click 子程序来保存未存储的数据。

11.3 MDI NotePad 应用程序

这一节中,我们通过创建一个简单的 MDI 应用程序,使读者进一步加深对 MDI 应用程序的了解和认识。

MDI NotePad 应用程序是一个简单的文本编辑器,它与 Microsoft Windows 中的 NotePad 程序有相似之处。不过 NotePad 是单文档模式(SDI),MDI NotePad 应用程序使用的是一个多文档界面。在运行时,如果用户需要一个新文档(用应用程序的【文件】菜单中的【新建】菜单项执行),应用程序就会创建子窗体的一个新实例。这样就可以按照需求,创建多个子窗体或文档。

在 VB 中,为了创建以文档为中心的应用程序,至少需要两个窗体:一个 MDI 窗体和一个子窗体。在设计时,应创建一个 MDI 窗体以容纳该应用程序,再创建一个子窗体作为这个应用程序文档的模板。

按照以下步骤,创建自己的 MDI NotePad 应用程序:

1. 单击工具条上的【新建工程】按钮,创建一个标准窗体(Form1),设 MDIChild 属性为 True。
2. 从【工程】菜单中,选择【添加 MDI 窗体】菜单项来创建窗体容器。
现在,这个工程中应当包含一个 MDI 窗体(MDIForm1)和一个标准窗体(Form1)。
3. 在 Form1 上创建一个文本框(Text1)。
4. 按表 11.2 方式为两个窗体和文本框设置属性。

表 11.2

对象	属性	设置值
MDIForm1	Caption	MDI NotePad
Form1	Caption	无标题
	MDIChild	True

续表

对象	属性	设置值
Text1	Multiline	True
	Text	空值
	Left	0
	Top	0

5. 单击【菜单编辑器】按钮,使用【菜单编辑器】为 MDIForm1 创建一个【文件】菜单。

标 题	名 称	缩 进
&File	mnuFile	否
&New	mnuFileNew	是

6. 在 mnuFileNew_Click 过程中增加以下代码:

```
Private Sub mnuFileNew_Click()
    '创建名为 NewDoc 的窗体 Form1 的一个新实例
    Dim NewDoc As New Form1
    '显示此新窗体
    NewDoc.Show
End Sub
```

这个过程创建并显示 Form1 的名为 NewDoc 的一个新实例(或称副本)。每当从【文件】菜单中选取【新建】命令时,将会创建一个与 Form1 完全相同的副本(实例),它包含 Form1 所包含的所有控件和代码。

7. 给 Form1 窗体的 Form_Resize 过程添加以下代码:

```
Private Sub Form_Resize()
    '扩展文本框以放置当前子窗体。
    Text1.Height = ScaleHeight
    Text1.Width = ScaleWidth
End Sub
```

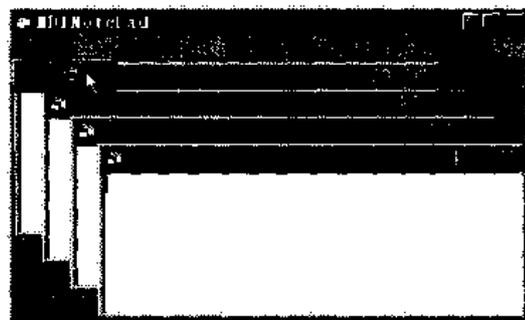


图 11.10 MDI 窗体示例

Form_Resize 事件过程的代码,能为 Form1 的每一个新实例所共享。当显示窗体的几个副本时,每个窗体都能识别各自的事件。

程序运行后,显示 MDI 窗体,开始只显示一个子窗体,每次从【文件】菜单中选择【新建】项,便建立一个新的子窗体显示在 MDI 窗体内。如图 11.10 所示。

11.4 本章小结

与普通的 VB 窗体相比,MDI 窗体是很独特的。当设计和编写 MDI 应用程序时,应记住 MDI 窗体的下列特征:

- 子窗体总是出现在 MDI 窗体上,从不出现在别处,甚至当最小化时,子窗体图标也仅在

MDI 窗体上出现。

- 如果子窗体有菜单,则当该子窗体具有焦点时,菜单会出现在 MDI 窗体上。运行时,在子窗体中看不到菜单,它们迁移到了 MDI 窗体中。
- 在设计阶段可以给 MDI 窗体添加菜单,但只能添加带有 `Align` 属性的控件。
- 可以运用 `Dim ... As New Form1` 语句在运行时创建一个子窗体的许多实例。

同时,读者还应知道一下基本技术:

1. 创建 MDI 窗体。
2. MDI 窗体的菜单设计。
3. 如何安排子窗体。
4. 调整子窗体大小。
5. 指定活动窗体或者控件。
6. 加载和卸载 MDI 窗体及其子窗体。
7. 保持子窗体的信息。

第 12 章 调试程序

无论用户如何仔细的编写代码,都可能会出现这样那样的错误,而且随着应用程序代码量的增加,出现错误的概率也将成倍的增长。因此,用户在编写较大一些程序的时候,应该了解如何查找程序的出错。

用户在编程时出现的错误,有些是用户在编写代码时由于考虑不周导致的逻辑错误;另一些是代码运行时产生的错误,这类错误是由于试图完成非法操作而造成的。

在应用程序中查找并修改错误的进程称之为调试,即 Debug。为了分析应用程序的操作方式,VB 提供了几种工具。这些调试工具可以帮助用户处理逻辑错误和运行时错误,并观察出错误代码的状况。调试主要是有助于了解在应用程序运行时正在发生的事情。

VB 提供的调试工具提供了应用程序当前状态的信息,包括:

- 用户界面(UI)的外观;
- 变量、表达式和属性的值;
- 活动的过程调用。

但并不能自动诊断或更正错误,不过能帮助用户分析运行是如何从过程的一部分流动到另一部分,以及变量和属性如何随着语句的执行而改变。有了调试工具,就能深入到应用程序内部去观察,从而确定到底发生了什么以及为什么会发生。

本章要介绍的就是处理应用程序中可能出现的各种错误,同时介绍如何使用 VB 中的调试工具查找逻辑错误,使程序可以适应各种复杂的情况。

12.1 错误类型和程序模式

12.1.1 VB 程序中的错误类型

程序中出现的错误是多种多样的,一般地,我们把可能遇到的错误分成三类:

- 编译错误
- 逻辑错误
- 运行错误

编译错误是由于不正确构造代码而产生的。例如错误的键入了关键字、遗漏了某些必须的标点符号,或在设计时使用了一个 Next 语句而没有 For 语句与之对应,那么 VB 在编译应用程序时就会检测到这些错误。例如:当用户在 Form_Load 过程中输入

```
For I = 30 To 60
```

此时,系统会弹出如图 12.1 所示的消息框提示错误信息。如果仅仅通过消息框上简单的提示信息不能够了解产生错误的原因,用户还可以单击对话框的【帮助】按钮取得更加详细的错误分析信息。

如果已在【选项】对话框的【编辑器】选项卡中选定【自动语法检测】复选框,那么只要在代码窗口中输入一个有语法错误的语句后,在将光标移到其他行时,VB 就会立即显示错误信息。

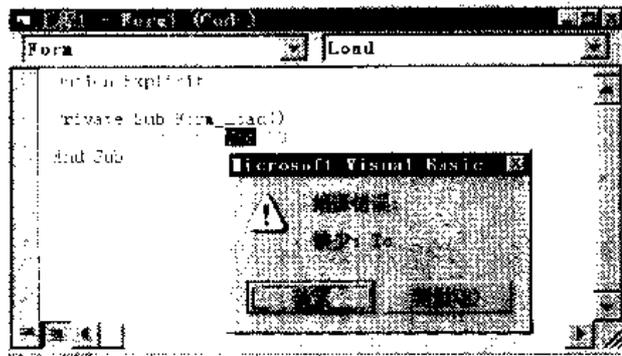


图 12.1 编译错误

当应用程序未按预期方式执行时,就会产生逻辑错误。从语法角度来看,应用程序的代码可以是有效的,在运行时也未执行无效操作,但还是产生了不正确的结果。应用程序运行的正确与否,只有通过测试应用程序和分析产生的结果才能检验出来。

应用程序正在运行(而且被 VB 检测)期间,当一个语句试图执行一个不能执行的操作时,就会发生运行时错误。假定有这样一个语句:

```
Dim Array ( 1 To 10 ) As Double
Dim datMy As Double
datMy = Array(11)
```

尽管语句本身的语法是正确的,必须运行应用程序才能检测到这个错误。因为数组大小为 10,不能存取第 11 个数。应用程序中可以包含处理运行时错误的代码,在这些错误发生时捕获并处理它们。

在以上列出的三类错误中,以编译错误最为简单,也最容易发现解决,只要针对编译时提供的错误信息进行修改就可以了。如果存在编译错误,应用程序也就不可能运行起来。而要解决其他两种错误,就要花一番功夫。本章后面的内容将介绍如何处理逻辑错误和运行时错误。

当然,最好的情况是编写代码时尽量减少错误。用户可以用几个方法,避免在应用程序中产生错误:

1. 写出相关事件以及代码响应每个事件的方法,细心设计应用程序。为每个事件过程和每个普通过程都指定一个特定、明确的目标。
2. 多加注释。如果用注释说明每个过程的目的,那么在回过头来分析代码时,就能更深入理解这些代码。
3. 尽可能显示引用对象。要像对象浏览器中所列举对象那样声明对象,而不用 Variant 或一般的 Object 数据类型。
4. 造成错误的一个最普遍的原因就是键入了不正确的变量名,或把一个控件和另一个控件搞混了。要解决这个问题,可以用 Option Explicit 语句避免变量名的拼写错误。

12.1.2 三种程序模式

在任何时候 VB 总是处于下列三种模式之一:设计时、运行时和中断模式。设计时用 VB 创建应用程序,而在运行时运行这个程序,而中断模式能够中断程序的执行,可以在这个时候检查和改变数据。为测试和调试应用程序,在任何时候都要知道应用程序正处在哪种模式之

下。VB 集成环境的标题栏总是显示当前模式。例如在运行时会显示“[运行]”的字样,如图 12.2 所示。表 12.1 列出了 3 种模式的特性。

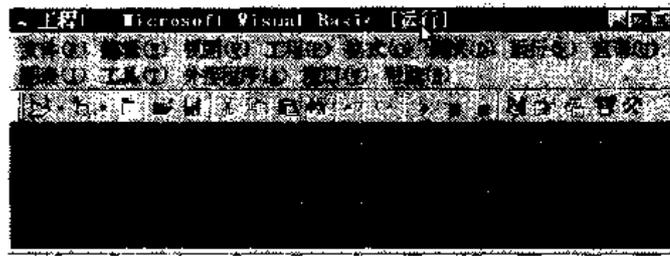


图 12.2 运行模式下的标题栏

表 12.1 VB 应用程序的 3 种模式

模 式	描 述
设计模式	创建应用程序的大多数工作都是在设计时完成的。在设计时,可以设计窗体、绘制控件、编写代码并使用【属性】窗口来设置或查看属性设置值。除了可以设置断点和创建监视表达式外,不能运行代码或使用调试工具。在【运行】菜单中选择【启动】选项或单击【启动】按钮,可以切换到运行模式。
运行模式	当应用程序运行时,可以与应用程序进行交互,还可以查看代码,但不能改动它。选择【运行】菜单中的【结束】选项或单击【结束】按钮,切换回设计模式。
中断模式	在运行时,选择【运行】菜单中的【中断】选项,或者单击【中断】按钮,或按下 Ctrl + Break 可切换到中断模式。在中断模式下,可以查看并编辑代码(选择【视图】菜单中的【代码窗口】,或按下 F7 键),检查或修改数据、重新启动应用程序,结束执行或从中断处继续运行。大多数调试工具只能在中断模式下使用。

12.2 进入中断模式

为了使用调试工具,应首先进入中断模式。在中断模式可随时终止应用程序的执行,并提供有关应用程序的情况对照。因为变量和属性设置值被保留下来,所以用户可以分析应用程序的当前状态并输入修改内容,这些修改将影响应用程序的运行。而且除了可以在设计时设置断点和监视表达式外,其他的调试工具只能在中断模式中运行。

进入中断模式的方法有很多,这里介绍几种途径。

12.2.1 在程序中设置断点

当 VB 正在运行一个过程并遇到一行具有断点的代码时,会在该行代码之前终止运行,并切换到中断模式。

用户可在中断模式下或设计时设置或删除断点。通常情况下,断点被用来在程序运行到被怀疑出问题的代码区域时,进入中断模式。要设置断点,可以使用下面列出的任何一种方式:

1. 在代码编辑窗口中,把插入光标移到要设置或删除断点的代码行,然后:
 - 单击“调试”菜单的“切换断点”选项。

- 单击“调试”工具栏的“切换断点”按钮(为显示“调试”工具栏,在 VB 工具栏上单击鼠标右键并选定“调试”选项)。
- 按下 F9 键。

2. 在代码编辑窗口要设置或删除断点的那一行代码的左边空白区单击鼠标。

设置了断点后,VB 将以粗体形式突出显示选定行,并在该行前面显示一个黑色圆点符号,以指出这是一个断点,如图 12.3 所示。

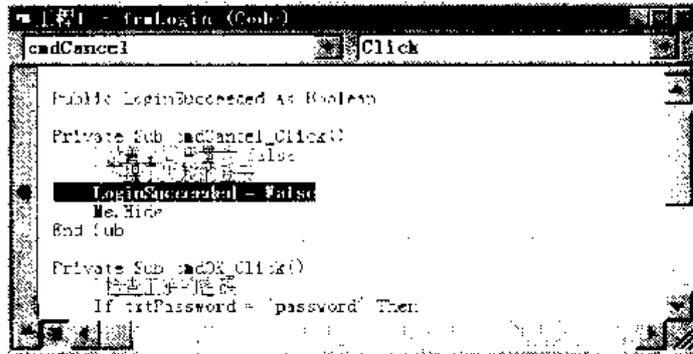


图 12.3 在程序中设置断点

要清除断点,只需将插入光标移到要删除断点所在的行,然后再执行一遍设置断点的操作即可。

在程序运行时,当执行到设置了断点的代码行时,会自动停下来进入中断模式,并在代码行之前显示一个→符号,指示下一条要执行的代码行,如图 12.4 所示。

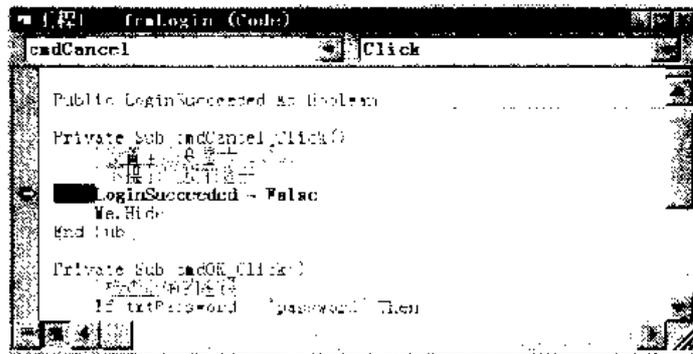


图 12.4 一个被断点中断的过程

应用程序一旦运行到断点处并被中断执行时,就可检查应用程序的当前状态。检查应用程序的结果是很容易的,因为可以在应用程序的窗体和模块、代码窗口以及调试窗口之间移动焦点。

在执行含有断点的行之前,这个断点中断执行应用程序。如果要观察设有断点的行在运行时发生了什么,就必须至少再运行一个语句,为此要使用逐语句或逐过程运行,这些内容将在后面陆续介绍。

提示: 断点所在行的颜色可以通过选择【工具】菜单下的【选项】命令来重新设置,在【选项】对话框的【编辑器格式】选项卡中,用户可以在【代码颜色】列表框中选定【断点文本】,然后进行各种设置,如图 12.5 所示。

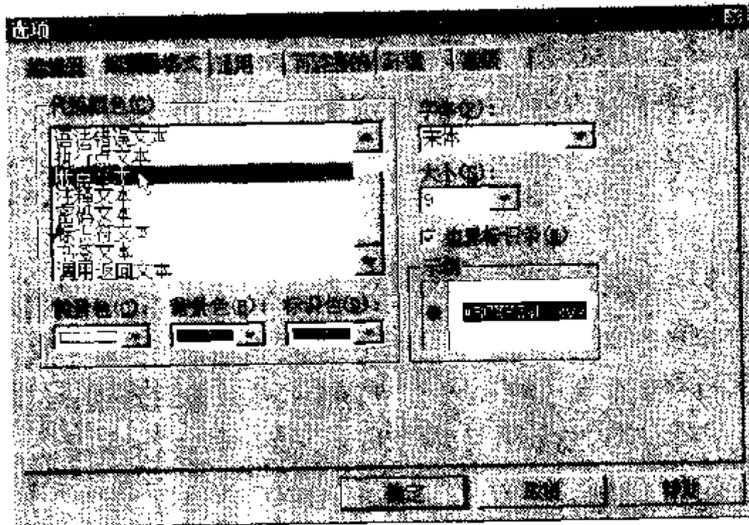


图 12.5 设置断点文本颜色

12.2.2 使用 Stop 语句进入中断模式

在过程中放置一条 Stop 语句也可以设置断点。每当 VB 遇到 Stop 语句时,这条语句就终止执行并切换到中断模式。虽然 Stop 语句与断点的功能相似,但它们的设置和清除方法却是不同的。在程序中设置的断点在程序被重新载入时就会被全部清除,而 Stop 语句却会一直存在。

Stop 语句除了暂时中断执行外,不做任何事情,而 End 语句则终止执行、重置变量并返回运行模式。但可选择【运行】菜单中的【继续】项来继续运行应用程序。

12.2.3 其他方式

在运行时,用户设置的断点和 Stop 语句都会使应用程序在用户认为可能有问题的代码处终止执行,进入中断模式,而且在应用程序运行时,还可以用下面的方法进入中断模式:

- 按下 Ctrl + Break 键。
- 选择【运行】菜单中的【中断】菜单项。
- 单击工具栏上的【中断】按钮。

此外,在运行时产生的一些错误也可以让应用程序自动进入中断模式。某些运行时错误是由于输入代码时的疏忽造成的,这些错误很容易更正。例如,把变量名拼错,属性或方法与对象不匹配(诸如对文本框使用了 Circle 方法,对滚动条使用 Text 属性),这些都是常见错误。图 12.6 显示一个运行时的错误消息,用户只要单击【调试】按钮就可以进入中断模式进行错误更正,也可以单击【结束】按钮结束程序的运行。

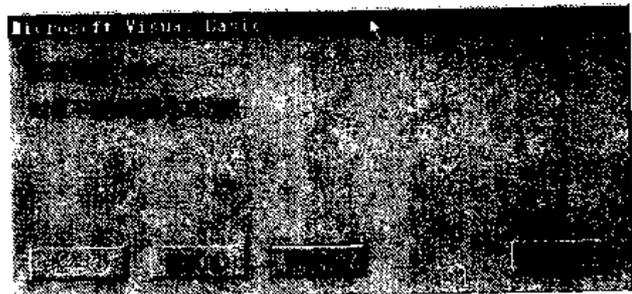


图 12.6 运行出错时,系统弹出消息框

通常可在终止应用程序的同一行处输入更正并继续执行程序,即使已改变了某些代码,情

况亦如此。直接选择【运行】菜单的【继续】选项,就可以继续执行程序。在继续运行应用程序时,可以验证问题是否得到更正。

如果对应用程序的代码进行了某些变更(最常见的是改变变量声明、添加新的变量或过程)之后重新启动应用程序,VB 会弹出一条消息,询问是否想要重新启动应用程序。

12.3 跟踪应用程序的执行

常见的情况是只知道产生错误的大致区域,需要使用断点将问题区域进行隔离,然后用逐语句和逐过程执行来观察每个语句的效果,这种方式往往被称为“跟踪”。如果有必要,还可在一条新行上开始执行,从而跳过几条语句或倒退回去。

12.3.1 逐语句执行代码

逐语句执行就是一条语句一条语句的执行代码,然后通过查看应用程序的窗体或调试窗口来判断这条语句是否正确。

在中断模式下,选择【调试】菜单的【逐语句】选项或者单击【调试】工具栏【逐语句】按钮,都可以进行逐语句跟踪,但是最常见的手段还是按 F8 键。

当逐语句执行代码时,VB 先暂时切换到运行模式,当执行完当前语句后再切换回中断模式。如果一行上有多条语句(以“:”隔开),也仍然可以使用逐语句执行的功能一条语句一条语句的执行。

12.3.2 跳过代码中的过程

【调试】菜单中的【逐过程】选项其实与【逐语句】选项没有什么不同,只不过逐过程执行的时候,将整个用户定义的函数或者过程视为一条语句,一次执行过去。而逐语句执行的方式则会进入到被调用的过程和函数体中。

提示:用户可在逐语句和逐过程之间随意选择。使用什么命令,这取决于在任何已知时刻要分析代码的哪个部分。如果确认某个过程不会有问题,就没有必要再进入过程中一步一步地浪费时间了。

除了选择【调试】菜单的【逐过程】选项,还可以单击【调试】工具栏的【逐过程】按钮或按下 Shift + F8 键来进行逐过程跟踪。

12.3.3 从过程中跳出

当用逐语句跟踪进入到过程中后,如果发现过程中的语句没有问题,可以单击【调试】工具栏的【跳出】按钮,从当前的过程中跳出,去执行过程调用者的下一条语句。

此外,用户还可以选择【调试】菜单的【跳出】选项,或者按下 Ctrl + Shift + F8 键来使用【跳出】功能。

12.3.4 运行到光标处

在对应用程序进行跟踪时,可以略过不感兴趣的部分代码(比如巨大的循环),方法是把插入光标设置在需要停止运行的代码上,然后按下 Ctrl + F8 键,或者选择【调试】菜单的【运行到

光标处】选项。

12.3.5 设置下一条要执行的语句

在调试或检验应用程序时,有时需要在修改了某项变量或者属性后,回过头再执行一次某条语句以验证修改是否正确,这时可以设置下一条要执行的语句,返回到前面的语句处,只要这行代码也在同一过程中。

要设置下一条执行语句,只需在中断模式下,把插入光标移到下一次要执行的代码行处,然后选择【调试】菜单的【设置下一条语句】选项或者按 Ctrl + F9 键。

在设置完后,可以选择【运行】菜单中的【继续】项来恢复执行,也可以选择【调试】菜单的【运行到光标处】或【逐过程】等其他选项。

注意: 如果已经执行错误处理程序中的代码,但不能肯定要在哪儿恢复运行,则可用【调试】菜单中的【显示下一条语句】选项把光标设置到下一个要执行的行。

12.4 使用调试窗口

VB 提供了多个调试窗口供用户调试程序使用。

12.4.1 使用监视窗口

在很多情况下,调试的问题不是由单个语句产生的,所以需要在整个过程中观察变量或表达式的情况,VB 可以自动对用户定义的表达式进行监视。当应用程序进入中断模式后,这些监视表达式会出现在【监视】窗口中,可在此处观察它们的值。

在设计或中断模式下都可添加监视表达式,方法是选择【调试】菜单中的【添加监视】选项,然后用【添加监视】对话框(如图 12.7 所示)来添加监视表达式。

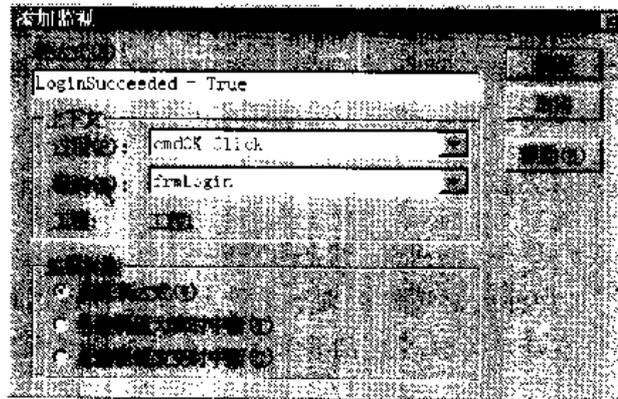


图 12.7 【添加监视】对话框

在【添加监视】对话框中,用户可以设置【表达式】框、【上下文】和【监视类型】选项,来达到对应用程序中变量进行有效监视的目的。

1. 【表达式】框

输入表达式的地方,监视表达式将计算表达式的值。表达式可以包含一个变量、属性、函数调用或其他任何有效的表达式。

2. 【上下文】选项组

用来设置表达式中要监视的变量的范围。当有名字相同而范围不同的变量时,使用该选项组。还可把监视表达式中的变量的范围限制为由特定过程、特定窗体或模块组成,也可通过从【模块】下拉列表选定【所有过程及所有模块】使范围可用于整个应用程序。需要注意的是,VB 在小范围内可更快的计算变量。

3. 【监视类型】选项组

用来设置 VB 对监视表达式作响应的方式。VB 可在应用程序进入中断模式后对表达式进行监视并显示其值。

提示: 在表达式的值为真(非零)语句时,或表达式的值每次发生改变时,就可使应用程序自动进入中断模式。

无论何时,只要表达式的值改变或等于一个特定的值,就可把应用程序设置为中断模式来直接观察表达式。例如,在循环计数器 count 达到一个特定值时,可用一个监视表达式

```
count = 10
```

并且把【监视类型】设置为【当监视值为真时中断】选项,这样当循环执行了 10 次后, count = 10 的表达式值为真,则应用程序自动进入中断模式,而不是一次一条语句逐步执行数十次乃至成百上千次循环。同样,在过程中一个变量的值改变时也可以使用应用程序进入中断模式。

提示: 也可直接把表达式从代码编辑器中拖放到监视窗口来添加表达式。

从【视图】菜单中选择【监视窗口】项可打开【监视】窗口,在【监视】窗口中,【上下文】列指出过程、模块,每个监视表达式都在这些过程或模块中进行计算。只有当前语句在指定的上下文中时,【监视】窗口才能显示监视表达式的值,否则【值】列只显示一条消息,指出语句不在上下文中。图 12.8 显示了监视窗口。

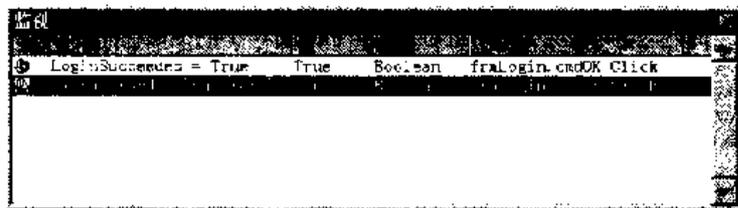


图 12.8 VB 的监视窗口

提示: 在中断模式下,只要将鼠标停留在某个变量上,就可以通过提示信息得到它当前的值(如图 12.9 所示),完全没有必要再浪费时间将简单变量添加到监视窗口中。

如果要对监视表达式进行编辑,可以在【监视】窗口中双击想要编辑的表达式,然后直接对表达式进行修改。也可以在【监视】窗口中选定想要编辑的监视表达式,然后选择【调试】菜单中的【编辑监视】选项,显示【编辑监视】对话框。除了标题栏不同以及增加了一个【删除】按钮外,这个对话框与【添加监视】对话框完全相同。用户可以在这个对话框中改变表达式、变量的求值范围或监视类型,或者单击【删除】按钮删除表达式。

提示: 在【监视】窗口中选定要删除的监视表达式,按下 Delete 键,也可以删除一个表达式。

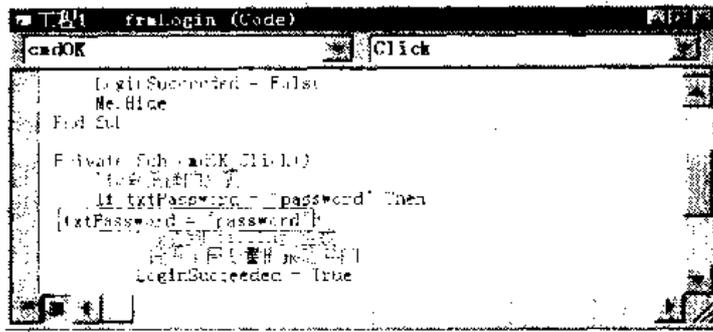


图 12.9 直接查看变量的值

在中断模式下,还可以使用【快速监视】检查那些没有在【监视】窗口中定义的属性、变量或表达式的值,如图 12.10 所示。方法是在【代码】窗口中选中要进行快速监视的表达式,然后单击【调试】工具栏的【快速监视】按钮,或按下 Shift + F9 键。

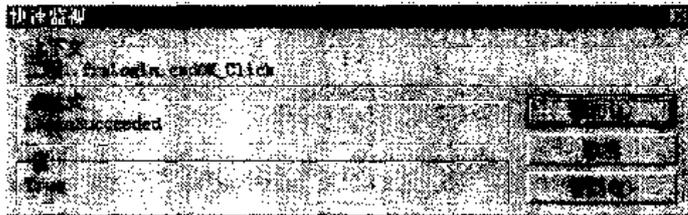


图 12.10 【快速监视】对话框

【快速监视】对话框显示从代码窗口中选定的表达式的值。如果要继续监视这个表达式,应单击【添加】按钮,该表达式就会出现在监视窗口中。

12.4.2 使用【调用堆栈】对话框

在应用程序执行一系列嵌套过程时,【调用堆栈】对话框有助于跟踪程序的操作。例如,一个事件过程可以调用第二个过程,而第二个过程又可以调用第三个过程。这样的嵌套过程调用很难跟踪,并且还很可能把调试过程弄复杂。而使用【调用堆栈】对话框可以显示所有活动过程调用的一个列表,所谓活动过程调用,就是应用程序中已启动但尚未完成的过程。

要显示【调用堆栈】对话框,应选择【视图】菜单的【调用堆栈】选项或者单击【调试】工具栏【调用堆栈】按钮,图 12.11 显示了一个【调用堆栈】对话框。

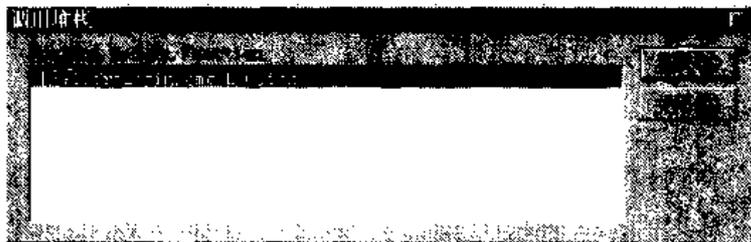


图 12.11 【调用堆栈】对话框

【调用堆栈】对话框列出了一系列嵌套调用中的所有活动过程调用,最早被调用的活动过程调用位于列表的底部,而在列表顶部添加最新的过程调用,这也是“堆栈”这个词所表示的含义——先入后出,即最后被调用的过程将最先退出,弹出堆栈。

注意：仅在应用程序处在中断模式下才可显示【调用堆栈】对话框。而且如果在空闲循环期间把应用程序置为中断模式，那么在【调用堆栈】对话框中将不出现任何项。

利用【调用堆栈】对话框还可以跟踪嵌套过程，显示调用下一个过程的语句的位置。方法是在【调用堆栈】对话框中选择一个过程调用，然后选择【显示】按钮，【调用堆栈】对话框将消失，而在代码窗口中将使用一个向右箭头符号指出在该过程中调用下一个过程的语句。如果在【调用堆栈】对话框中选择当前过程，则光标将出现在当前语句处。

12.4.3 使用本地窗口

用户在调试程序时还可以利用【本地】窗口显示当前过程中所有变量的值，如图 12.12 所示。当程序的执行从一个过程切换到另一个过程时，【本地】窗口的内容会发生改变，它只反映当前过程中可用的变量。如果要查看访问【本地】窗口，应选择【视图】菜单中的【本地窗口】选项或者单击【调试】工具栏的【本地窗口】按钮。

提示：如果单击【本地】窗口中【过程】框右旁的“...”按钮，可以显示【调用堆栈】对话框。

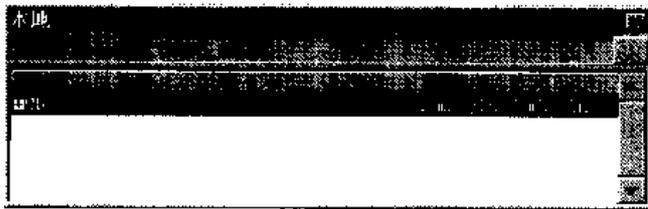


图 12.12 VB 的本地窗口

12.4.4 使用立即窗口

立即窗口可以说是 VB 功能最强大的调试工具，使用立即窗口可以检查某个属性或者变量，还可以执行单个的过程、对表达式求值，或为变量或属性赋予新的值。

一般来讲，在进入中断模式后就会显示立即窗口。如果在其他时候要显示立即窗口，可以选择【视图】菜单中【立即窗口】选项，或单击【调试】工具栏的【立即窗口】按钮，或者按 Ctrl + G 键。

1. 从应用程序中输出信息到立即窗口

在立即窗口中也可以对变量和表达式的变化进行监视，方法很简单，只要在 Print 方法前加上 Debug，就能把输出传送到立即窗口，它的语法是：

```
Debug Print [输出内容] [;]
```

例如，下面的语句在每次执行时都把变量 count 的值和字体名打印到立即窗口中，如图 12.13 所示。

```
For count = 0 To Screen.FontCount - 1
    Debug.Print "count = " & count , Screen.Fonts(count)
Next
```

注意：当把应用程序编译到 .exe 执行文件时，将删掉 Debug.Print 语句。于是，如果在使用 Debug.Print 语句时，应用程序仅仅把字符串或简单变量类型当作参数，那么它将没有任何

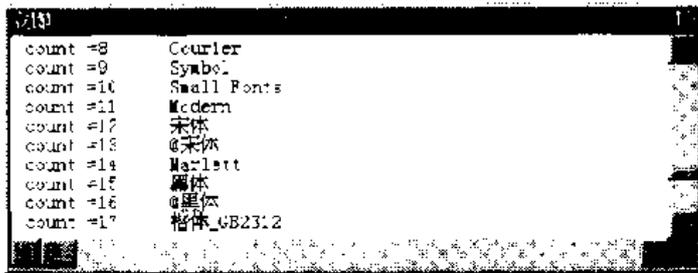


图 12.13 从应用程序输出信息到立即窗口

Debug.Print 语句。但是,VB 不能去掉在 Debug.Print 中作为参数的函数调用。所以那些函数的副作用将继续在编译后的 .exe 执行文件中存在,即使不打印函数结果,情况也是如此。

2. 改变变量和属性值

在程序处于中断模式下,可以在立即窗口中用下面的语句设置变量或者属性的值:

```
Command1.Caption = "学习使用立即窗口"  
Count = 20
```

第一个语句改变按钮的 Caption 属性,第二个语句则是给变量 Count 赋值。在设置了一个或多个变量和属性的值后,可以继续执行程序并观察结果。

注意:立即窗口对任何有效的 VB 的可执行语句求值,但不接受数据声明。

12.5 本章小节

软件开发的过程中错误在所难免,只有掌握了一定的调试工具和调试方法后,才能很好的处理应用程序中的各种错误,本章向读者简述了调试 VB 程序的基本概念和方法,实际应用中,会出现各种各样的错误,读者只有在实践中慢慢锻炼,才能掌握更好更多的调试技巧。

学完本章后,读者应能知道以下内容:

1. 了解 VB 程序设计中出现的三种错误类型:编译错误、逻辑错误和运行错误。
2. 了解 VB 程序的三种工作模式:设计模式、运行模式、中断模式,调试应用程序常常在中断模式进行。
3. 掌握几种进入中断模式的方法。
4. 会使用集成环境中的【运行】和【调试】菜单跟踪程序的运行。
5. 掌握如何使用调试窗口来发现和处理应用程序中的错误,包括:监视窗口的使用、本地窗口的使用、立即窗口的使用以及调用堆栈对话框的使用。

在以后的程序设计中,读者应该尝试使用各种调试方法调试自己的应用程序。

第 13 章 文件操作

应用程序必须在内存中运行,运行结果也保存在内存中,但计算机的内存是会时刻刷新的,退一步说,即使我们的应用程序及结果总在内存中占有-定的区域,当计算机关闭时,内存中的数据也将全部丢失。所以,必须将应用程序以文件的形式保存在磁盘或磁带等长期存储设备中。因此,应用程序需要具备储存数据到磁盘或从磁盘读入数据的能力,VB 为用户提供了强大的对文件系统的支持能力,使用户可以很方便地访问文件系统,而且极其简单。本章将介绍与文件系统有关的内容,包括:文件系统的基本概念,VB 操作文件系统的语句;3 个有关文件系统的标准控件的使用;以及如何在应用程序中进行不同类型文件的读写。

13.1 文件系统的概念

VB 根据计算机访问文件的方式将文件分成三类:顺序文件、随机文件和二进制文件。

1. 顺序文件

顺序文件是普通的正文文件,是最简单的文件结构,其中的记录按顺序一个接一个地排列。顺序文件中提供第一个记录的存储位置,其他记录的位置无法获悉。所以要在顺序文件中找一个记录,必须从第一个记录开始读取,直到找到该记录为止。顺序文件的缺点是:如果要修改数据,必须将所有数据读入 RAM 中进行修改数据,然后再将修改好的数据重新写入磁盘。由于无法灵活地随意存取,它只适用于有规律的、不经常修改的数据,如文本文件。优点是所占空间较少,而且容易使用。

2. 随机文件

随机文件是可以按任意次序读写的文件,每个记录的长度必须相同。在这种文件结构中,每个记录都有其唯一的一个记录号,所以在存入数据时,只要知道记录号,便可以直接读取记录。随机文件的优点是存取数据快,更新容易;缺点是所占空间较大,设计程序较繁。

3. 二进制文件

二进制文件是字节的集合,它允许程序按所需的任何方式组织和访问数据,这类文件的灵活性最大,但程序的工作量也最大。

13.2 用于处理文件系统的语句和函数

在 VB 中提供了一些处理文件系统的语句和函数,已基本可以满足用户要求。使用这些语句和函数,用户就可以在应用程序中进行一些改变目录、删除目录和删除文件等操作。

1. 取得当前目录信息

要取得当前目录的有关信息,可以使用 CurDir 函数,它的语法是:

```
CurDir[(drive)]
```

方括号表示(drive)是可选参数,drive 是一个字符串表达式,指定一个存在的驱动器。如果

没有指定驱动器,或 drive 是零长度字符串(""),则 CurDir 会返回当前驱动器的当前路径。

假如 C 驱动器的当前路径为"C:\Windows\vb98",D 驱动器的当前路径为"D:\Word97",并且 C 为当前驱动器,下面的语句可以取到关于 C、D 驱动器当前目录的信息:

```
Dim ReturnPath
ReturnPath = CurDir '返回当前驱动器的当前目录"C:\Windows\vb98"。
ReturnPath = CurDir("C") '返回值同上
ReturnPath = CurDir("D") '返回"D:\Word 97"。
```

2. 改变当前目录

要改变当前的目录或文件夹,应使用 ChDir 语句,它的语法是:

```
ChDir Path
```

Path 参数是必不可少的,它是一个字符串表达式,指明哪个目录或文件夹将成为新的默认目录或文件夹。Path 可能会包含驱动器名称,如果没有指定驱动器,则 ChDir 不改变驱动器,只在当前驱动器上改变默认目录或文件夹。例如下面的语句可以将当前目录或文件夹改为"VbHelp"。

```
ChDir "VbHelp"
```

注意: ChDir 语句改变默认目录位置,但不会改变默认驱动器位置。例如,如果默认的驱动器是 C,则下面的语句将会改变驱动器 D 上的默认目录,但是 C 仍然是默认的驱动器:

```
ChDir "D:\ProgrammFiles"
```

3. 改变当前驱动器

使用 ChDrive 语句可以改变当前的驱动器,它的语法是:

```
ChDrive drive
```

drive 参数是必不可少的,它是一个字符串表达式,指定一个存在的驱动器。如果使用零长度的字符串(""),则当前的驱动器将不会改变。如果 drive 参数中有多个字符,则 ChDrive 只会使用第一个字母。例如,下面的语句将使 D 驱动器变成当前驱动器:

```
ChDrive "D"
```

4. 建立和删除目录

使用 MKDir 语句可以创建一个新的目录。如果没有指定驱动器,新目录或文件将会建在当前驱动器中。例如,使用下面的语句可以建立一个名为"Mytemp"的目录:

```
MKDir "Mytemp"
```

如果要删除一个已经存在的目录,可以使用 RMDir 语句。例如,使用下面的语句将会删除 C 驱动器中名为 MyTemp 的目录

```
RMDir "C:\MyTemp"
```

如果没有指定驱动器,则 RMDir 会在当前驱动器上删除目录或文件夹。

注意: 如果想要使用 RMDir 来删除一个含有文件的目录或文件夹,则会发生错误,在用 RMDir 删除目录或文件夹之前,必须先使用 Kill 语句删除其中所有的文件。

5. 删除文件

使用 Kill 语句可以从磁盘中删除文件,它的语法是这样的:

```
Kill Pathname
```

其中 Pathname 参数是必要的,它是用来指定一个文件名的字符串表达式。Pathname 可以包含目录或文件夹,以及驱动器。例如下面的语句可以删除 D 驱动器上 TEMP 目录下的 MUSIC.DAT 文件:

```
Kill "D: \ TEMP \ MUSIC.DAT"
```

Kill 语句还可以支持通配符“*”和“?”来指定多重文件。例如:语句

```
Kill "D: \ TEMP \ * . *"
```

可以删除 D 驱动器 TEMP 目录下的所有文件。

注意: Kill 语句不能用来删除一个已打开的文件,否则会产生错误。

6. 设置文件的属性

可以使用 SetAttr 语句设置一个文件的属性信息。它的语法是:

```
SetAttr Pathname, attributes
```

Pathname 和 attributes 这两个参数都是必不可少的。Pathname 是用来指定一个文件名的字符串表达式,其中可以包含目录以及驱动器。attributes 参数是一个常数或数值表达,其总和用来表示文件的属性。attributes 参数的取值如表 13.1 所示。

表 13.1 attributes 参数的取值及其含义

常数	值	含义
VbNormal	0	常规(默认值)
VbReadOnly	1	只读
VbHidden	2	隐藏
VbSystem	4	系统文件
VbArchive	32	上次备份以后,文件已经改变

例如下面的语句可以将文件“C : \ System \ osput.sys”的属性设置为系统的、隐藏的、只读的:

```
SetAttr "C: \ System \ osput.sys", vbSystem + vbHidden + vbReadOnly
```

注意: SetAttr 不能用于给一个已打开的文件设置属性,否则,会产生运行错误。

7. 得到当前可执行文件的路径

在使用有关路径的信息时,经常会用当前可执行文件的路径,因为应用程序的大多数文件往往都在这个目录下。要取得当前可执行文件的路径,可以使用这样的语句:

```
Dim CurrentPath  
CurrentPath = App.Path
```

APP 对象是通过关键字 APP 访问的全局对象。它不需要事先声明,用户可以从其中取得如

下信息:应用程序的标题、版本信息、可执行文件和帮助文件的路径及名称,以及是否运行前一个应用程序的示例。Path 只是 App 对象的一个属性,它在运行时是只读的,而且只在运行时可用,在设计时不可用。

除了上面介绍的语句或函数外,VB 还提供了一些函数或语句可以用来对文件进行操作。表 13.2 列出了这些函数或语句以及关于其功能的简要说明。

表 13.2 VB 中用于文件系统操作的函数或语句

函数或语句	说 明
Dir	返回一个字符串,用以表示一个文件名、目录名或文件夹名称,它必须与指定的模式、文件属性相匹配
EOF	说明数据是否已经读完,它只带有一个参数(文件标识符),并返回一个布尔值,返回值为 True,说明读完了所有数据
FileCopy	复制一个文件
FileDateTime	返回一个 Date(Variant)值,表明一个文件被创建或最后修改的日期和时间
FileLen	返回一个 Long 值,代表一个文件的长度,单位是字节
FreeFile	返回一个 Integer 值,代表下一个可供打开的文件号
GetAttr	返回一个 Integer 值,表明一个文件、目录或文件夹的属性
Loc	返回一个 Long 值,在已打开的文件中指定当前读/写位置
LOF	返回一个 Long 值,表示用 Open 语句打开的文件的大小,该大小以字节为单位
Seek(语句)	在 Open 语句打开的文件中,设置下一个读/写操作的位置
Seek(函数)	返回一个 Long,在 Open 语句打开的文件中指定当前的读/写位置

13.3 文件系统控件

由于应用程序必须将文件从内存存入磁盘,或从磁盘中读入文件,所以要显示关于磁盘驱动器、目录和文件的信息。为使用户方便地利用文件系统,VB 提供了两种方法:既可以使用 CommonDialog 控件提供的标准对话框,也可以使用 VB 提供的文件系统控件创建自定义对话框。

VB 提供了 3 个文件系统控件,它们是:

- 目录列表框(DirListBox)
- 驱动器列表框(DriveListBox)
- 文件列表框(FileListBox)

利用这些控件能够自动地从操作系统获取所需的信息,用户可以访问这些信息或通过其属性判断每个控件的信息。例如,在默认方式下显示当前工作目录的内容。

如果用户只需要标准的【打开】和【保存】对话框,则可以使用 CommonDialog 控件,这些对话框也为许多其他基于 Microsoft Windows 的应用程序所使用,因此具有标准化的外观。

13.3.1 驱动器列表框

驱动器列表框缺省时在用户系统上显示当前驱动器。运行时,用户可在该控件上输入任

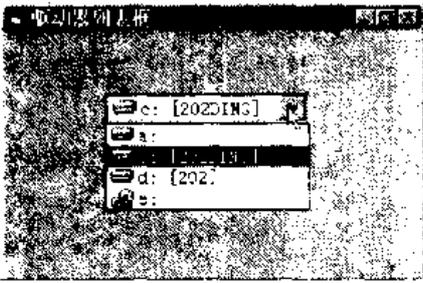


图 13.1 驱动器列表框

何有效的驱动器标识符,或者单击列表框右侧的箭头,从下拉列表选定新的驱动器,选定的驱动器将出现在列表框的顶端,如图 13.1 所示。

用户可用代码检查 Drive 属性来判断当前选择的是哪一个驱动器,也可以通过以下简单的赋值语句指定出现在列表框顶端的驱动器:

```
Drive1.Drive = "C: \\"
```

驱动器列表框显示有效的驱动器,但在列表框中选择驱动器并不能自动更改当前的工作驱动器;然而在用户选择新的驱动器后,将触发一个 Change 事件,在这个事件中,可用 ChDrive 语句在操作系统级别变更驱动器,语句如下:

```
ChDrive Drive1.Drive
```

13.3.2 目录列表框

目录列表框可以显示当前驱动器上的目录结构。它以根目录开头,显示的目录按照子目录的层次依次缩进,运行中选定的目录将用色条标识出来,双击某一目录,将显示该目录的所有子目录。

注意: 目录列表框只列出系统的目录结构,不能显示目录中的文件。

用户可以利用目录列表框的 Path 属性设置或返回列表框中显示的当前目录,例如:若将目录列表框 Dir1 和 Path 属性值设为"D: \ GAMES",则目录"D: \ GAMES"将成为显示的当前目录。

如果把驱动器列表框的 Drive 属性赋予目录列表框的 Path 属性,就可以显示驱动器列表框中选定的驱动器的目录:

```
Dir1.Path = Drive1.Drive
```

目录列表框中的每个目录都有一个整型标识符 ListIndex,可用它来标识每个目录。而 CommonDialog 控件没有提供这种功能。目录列表框显示的当前目录的 ListIndex 值 - 1,紧邻其上的目录的 ListIndex 值 - 2,再上一个的 ListIndex 值为 - 3,依此类推... 当前目录的第一子目录具有的 ListIndex 值为 0,如有多个子目录,每个子目录的值按 1、2、3... 的顺序依次排列。图 13.2 显示目录列表框显示的目录结构。

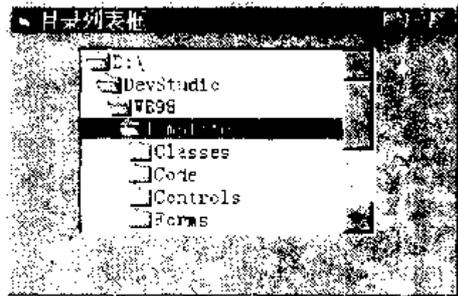


图 13.2 目录列表框显示的目录结构

利用 ListIndex 属性可以轻易地实现在目录中的上下移动,例如下面的语句就可以将目录列表框中显示的目录变为当前目录的上一级目录:

```
If Dir1.List(-2) <> "" Then
Dir1.Path = Dir1.List(-2)
```

提示: 如果要了解目录列表框中当前显示的目录下有多少直接的子目录,可以查看目录列表框的 ListCount 属性

目录列表框并不在操作系统级别设置当前目录,如果要设置当前工作目录应使用 ChDir 语句。例如,下列语句将当前目录变成目录列表框中显示的一个目录:

```
ChDir Dir1.Path
```

13.3.3 文件列表框

文件列表框在运行时显示由 Path 属性指定的目录中包含的文件。例如,可用如下语句显示当前目录中的所有文件:

```
File1.Path = Dir1.Path
```

文件列表框还有一个 Pattern 属性用来显示文件的子集,例如:设置 Pattern 为 .Lib,将只显示这种扩展名的文件。Pattern 属性还接受由分号分隔的扩展名列表,例如,下列代码将显示所有扩展名为 .Lib 和 .Tag 的文件:

```
File1.Pattern = ".Lib;.Tag"
```

Pattern 属性还支持 ? 通配符。例如:

```
File1.Pattern = "???.Lib"
```

将显示所有文件名包含三个字符且扩展名为 .Lib 的文件。

用户还可以根据文件的属性来决定该文件是否在文件列表框中显示。对于 Archive、Normal、System、Hidden 和 ReadOnly 几种属性的文件,可以在文件列表框中指定要显示的文件类型。方法是某个属性设置为 True,则具有相应属性的文件就可以在文件列表框中显示出来。

下面一个例子说明了驱动器列表框、目录列表框及文件列表框配合进行文件系统操作的过程,并且用户可以使用复选框来决定是否显示系统文件、只读文件和隐藏文件。

首先,在窗体中加入控件,并设置控件的 Caption 属性,然后加入如下代码:

```
Private Sub Drive1_Change( )  
'显示当前驱动器下的文件目录  
Dir1.Path = Drive1.Drive  
End Sub  
Private Sub Dir1_Change( )  
'显示当前目录下的文件  
File1.Path = Dir1.Path  
End Sub  
Private Sub ChkSystem_Click( )  
'检查用户是否选择了【系统文件】复选框  
If ChkSystem.Value = 0 Then  
File1.System = False  
ElseIf ChkSystem.Value = 1 Then  
File1.System = True  
End If  
End Sub  
Private Sub ChkReadOnly_Click( )  
'检查用户是否选择了【只读文件】复选框
```

```

If ChkReadOnly.Value = 0 Then
File1.Read Only = False
ElseIf ChkReadOnly.Value = 1 Then
File1.ReadOnly = True
End If
End Sub
Privat Sub ChkHidden.Click( )
'检查用户是否选择了【隐藏文件】复选框
If ChkHidden.Value = 0 Then
File1.Hidden = False
Else If ChkHidden.Value = 1 Then
File1.Hidden = True
End If
End Sub

```

程序运行结果如图 13.3 所示。

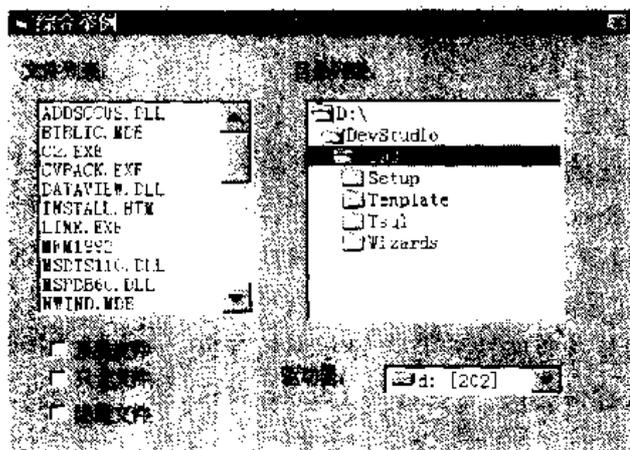


图 13.3 利用文件系统控件操作文件系统

提示：缺省状态下，在文件列表框中只突出显示一个选定的文件。要选定多个文件，应该用文件列表框的 MultiSelect 属性。

13.4 文件的读写

计算机中的文件，其实就是一系列磁盘中的相关字节。应用程序要想正确地访问一个文件，必须知道这些字节的正确含义，也就是所说的文件类型。

如第一节中提到的，VB 中有三种文件访问的类型：顺序型、随机型和二进制型，顺序型文件适用于读写在连结块中的文本文件。随机型文件适用于读写有固定长度记录结构的文本文件或者二进制文件。二进制型文件适用于读写任意有结构的文件。

虽然每种类型的文件存取方式不同，但操作这些文件的步骤大致相同，归纳如下：

1. 使用 Open 语句打开文件，并为文件指定一个文件号，程序根据文件的存取方式使用不同的方式打开文件。

2. 从文件中将全部或者部分数据读取到变量中。
3. 使用、处理或者改变变量中的数据。
4. 将变量中的数据保存到文件中。
5. 文件操作结束,使用 Close 语句关闭文件。

13.4.1 顺序文件

如果用户只需要处理纯文本文件,采用顺序存取方式最好,因为这样可以编写很简单的过程读写文本的内容。然而顺序存取也有其缺点,下列情况不适宜于采用顺序存取方式:

- 文件存储的大部分是较长的记录数据;
- 不是从文件的开头而是从文件的任意部分存取文件;
- 文件内容要经常修改。

1. 打开和关闭顺序文件

用户想对顺序文件进行操作,必须首先找顺序文件。VB 提供了 Open 语句,使用该语句时用户可以按顺序、随机或二进制方式打开文件。在对文件进行访问操作之前,必须打开此文件。

Open 为 I/O 分配一个文件缓冲区,并确定缓冲区的访问模式,要顺序访问一个文件,Open 语句的语法为:

```
Open File For[Input|Output|Append]As[#]filenumber
```

其中 File 代表文件名或路径名;可选参数 Input 表示从文件中读取字符;Output 表示向文件输出字符,输出的内容将重写整个文件,文件中原来的内容将丢失;Append 把字符追加到文件的最后,文件中原来的内容不会丢失;filenumber 参数是必不可少的,它为文件指明了一个有效的文件号,范围是 1~511 之间的整数,用作标识符,在文件打开时标识该文件。

提示:当以 Input 模式打开顺序文件时,该文件必须已经存在,否则会产生一个错误。以 Output 和 Append 模式打开文件时,可以打开一个不存在的文件。这时,Open 语句会首先创建该文件,然后再打开它。

例如,使用下面的语句打开一个顺序文件:

```
Open"Project.Vbp" For input.As #1
```

本例打开名为 Project.Vbp 的文件并指定文件号为 1,在打开文件之后,便可以从文件中读取字符或一行文本。

用户在打开一个文件并做了某些处理之后应及时关闭文件,VB 提供了 Close 语句用来关闭文件。Close 语句的语法格式为:

```
Close[#]filenumber[,#file number...]
```

其中,filenumber 参数是 Open 语句中用来打开文件所用的标识符,也可以使用任何等于打开文件号的数字表达式。当 Close 语句没有参数时,将关闭所有已被打开的文件。例如,执行以下语句:

```
Close #1
```

则文件号为 1 的文件将关闭。

2. 读写顺序文件

VB 提供了 3 种方法从文件中读取其内容,分别是 LineInput # 语句、Input()函数或者 Input # 语句。

LineInput # 语句可以从已打开的顺序文件中读出一行,并将它分配给 String 类型的变量。例如,以下代码段逐行读取一个文件到文本框 Text1:

```
Dim TextLine As String
Do Until EOF(FileNum)
LineInput # FileNum, TextLine
Text1.Text = Text1.Text + TextLine + Chr(13) + Chr(10)
Loop
```

对以上代码段需要做一点说明。FileNum 表示文件号,即用来打开文件的标识符,而且尽管 LineInput # 语句到达回车换行时会识别行尾,但当它把该行读入变量时,是不能读入回车换行符的。所以如果要保留回车换行,必须使用代码添加。

Input 函数可以从文件向变量拷贝任意数量的字符,所以所声明的变量应具有足够的空间,例如,以下代码使用 Input 把整个文件一次拷贝到文本框 Text1 中:

```
Text1.Text = Input(LOF(FileNum), FileNum)
```

从上面的例子也可以看出,Input 函数只包含两个必要的参数,前一个参数指明要读取字符的数目,后一个指明要读取文件的文件号。例如,以下语句从文件号为 1 的文件中读取 512 个字符。

```
Input(512, #1)
```

Input # 也用于从已被打开的文件中读取数据并赋予变量,它的语法格式为:

```
Input # FileNum, Varlist
```

其中 FileNum 是一个有效的文件标识符;Varlist 是一个变量,而且文件中数据的类型与变量的类型必须匹配。

注意: Input # 语句只能读出第一个逗号或空格之前的信息。

要将变量的内容写回顺序文件中,必须用 Output 或 Append 变元打开文件,然后使用 Print # 语句。例如,可以使用以下代码把文件框 Text1 中修改后的内容写回到文件中(文件必须以 OutPut 或 Append 打开):

```
Print # FileNum, Text1.Text
```

VB 还支持 Write # 语句,它把一系列数字或者字符串表达式写入文件中,自动地用逗号分开每个表达式,并且在字符串表达式两端放置引号。下面的代码说明这个问题。

```
Dim StuName As String * 8
Dim StuScore As Integer
Dim StuNum As String * 6
StuName = "丁琴"
StuScore = 97
StuNum = "SY7021025"
```

```
Open "C:\Students\exam.txt" For Output As #1
Write #1, StuName, StuScore, StuNumber
Close #1
```

这段代码把3个信息写入文件标识符为1的文件中,因而文件中包含以下内容:

```
"丁琴",97,"SY7021025"
```

由 Write # 语句写入文件中的内容,需要使用 Input # 语句读出。例如,要读出以上文件中的内容,可使用以下语句:

```
Input # FileNum, StuName, StuScore, StuNumber
```

使用 Append 模式可以向顺序文件中追加信息。当用户用 Append 模式向顺序文件追加信息时,VB 打开文件(如果不存在,就建立一个)并设置适当的缓冲区,然后把文件指针设置到文件末端,最后用一个 Print # 语句或 Write # 语句追加信息到文件尾部。例如:

```
Open "TextFile" For Append As #1
Msg = "Add information to a file"
Print #1, Msg $
Close #1
```

13.4.2 随机文件

随机文件是一个由长度固定的记录组成的集合。也就是说,随机文件是由一系列记录构成的,每个记录都由定长的字段构成,所以每个记录的长度都是相同的。例如,我们在日常生活中经常要用到的进行各种信息管理的情况,如在一个工厂中,对职工工资进行管理的文件通常包括以下三个方面的信息:职工姓名,职工编号,工资。这三方面的信息可以用一个自定义类型 SalaryMag 来定义:

```
Type SalaryMag
Name As String * 8 '职工姓名,8个字节
Number As String * 6 '职工编号,6个字节
Salary As Integer '职工工资,2个字节
End Type
```

这个类型的特点就是每个字段都具有固定的长度,针对这一个特点,就可以使用随机存取的文件来保存这些信息。

在打开一个文件进行随机访问之前,应定义一个类型,该类型应与文件中包含的类型一致,如前面定义的 SalaryMag 类型。

因为随机文件中所有记录必须有相同的长度,所以用户定义类型中的 String 类型的字段要使用固定的长度。如果实际字符串所包含的字符数比它写入的字符串元素的固定长度少,则 VB 会用空白符来填充记录中后面的空间。如果字符串的长度超过了字段,则它就会被截断。

1. 打开随机文件

在定义了与典型记录对应的类型之后,为方便地处理随机文件,应接着声明程序需要的任何其他变量,用来处理作为随机访问而打开的文件。例如,在定义了以上的 SalaryMag 记录以

后,可以定义以下变量:

```
'记录变量  
Public Salaryrec As SalaryMag  
'跟踪当前记录  
Public Position As Long  
'文件中最后一条记录的位置  
Public LastRecord As Long
```

在定义了以上变量之后,可以使用 Open 语句打开随机访问的文件:

```
Open pathname [For Random] As filename Len = reclength
```

由于 Random 是默认的文件访问类型,所以关键字 For Random 是可选项。表达式 Len = reclength 指定了每个记录的长度。如果 reclength 比写文件记录的实际长度要短,则会产生一个错误。如果 reclength 比记录的实际长度长,则记录可以写入,只是会浪费磁盘空间,所以在程序中,可以使用以下代码段来打开文件:

```
Dim Filenum As Integer, Salaryrec As SalaryMag, Reclength As Long  
Dim Name As String  
'计算每条记录的长度  
Reclength = Len(Salary)  
'取出下一个可用的文件编号  
Filenum = FreeFile  
'设置文件的路径  
Name = App.path & "\ "&"Manage.dat"  
'用 Open 语句打开新文件  
Open Name For Random As Filenum Len = Reclength  
'查看文件中共有多少条记录  
LastRecord = LOF(File num)/Len(Salaryrec)
```

在打开随机文件之后,如果要编辑随机访问的文件,请先把记录读到程序变量,然后改变变量中的值,最后把变量写回该文件。

2. 读写随机文件中的指定记录

使用 Get 语句可以把指定的记录复制到变量中。例如可以使用以下代码从打开的文件中读取 Position 号的记录:

```
Get Filenum, Postition, Salaryrec
```

在把记录读到变量中以后,就可以进一步对记录进行处理。例如可以将记录的各个字段显示到文本框中:

```
textName.Text = Salaryrec.Name  
textNumber.Text = Salaryrec.Number  
textSalary.Text = Salaryrec.Salary
```

然后用户就可以通过文本框来修改记录中的内容。

在修改完毕后,应该把修改后的值重新写回记录中:

```
Salaryrec.Name = textName.Text
Salaryrec.Number = textNumber.Text
Salaryrec.Salary = textSalary.Text
```

然后,使用 Put 语句把记录添加或者替换到随机型访问打开的文件中。随机文件的另一个好处就是,不必为了切换文件的 Input 和 Output 模式而不断地打开或关闭文件。只要打开了随机文件,就可以同时进行读或写的操作。

如要替换记录,就要指出想要替换的记录位置,例如以下代码用 Salaryrec 变量中的数据替换由 Position 所指定编号的记录:

```
Put # FileNum, Position, Salaryrec
```

用户还能利用 Put 语句向随机文件中添加记录,方法是把 Position 变量的值设置为比文件中的记录数多 1。例如,要在一个包含 10 个记录的文件中添加一个记录,把 Position 的值设置为 11。下面的语句可以把一个记录添加到文件的末尾,并且使 LastRecord 变量加 1:

```
LastRecord = LastRecord + 1
Put # FileNum, LastRecord, Salaryrec
```

3. 删除随机文件中的记录

通过清除字段可以删除一个记录,但该记录所占的空间仍在文件中存在。通常文件中是不能包含这类空字段的,因为它们会浪费空间而且会干扰顺序操作。最好把余下的记录拷贝到一个新文件,然后删除老文件。例如,下面的代码将名为“Temp.dat”文件中的第 Position 号记录删除:

```
Dim I As Integer
Dim filetemp as Integer
filetemp = FreeFile
Open "Temp.tmp" For Random As filetemp Len = Len(Salaryrec) '打开一个临时文件
For I = 1 To LastRecord
If I <> Position Then
'将记录拷贝到临时文件中并删除第 Position 号的记录。
Get FileNum, I, Salaryrec
Put Filetemp, I, Salaryrec
End If
Next
LastRecord = LastRecord - 1
Close FileNum
Close Filetemp
Kill("Temp.dat") '删除原来的文件
'将新文件命名为"text.dat"
Name "Temp.tmp" As "Temp.dat"
```

注意: 在程序中要首先关闭文件,然后才能对文件进行删除和重命名等操作。

4. 利用滚动条来访问随机文件中的记录

在随机文件中可以直接读写某一条特定的记录,所以可以使用一个滚动条控件来直接对

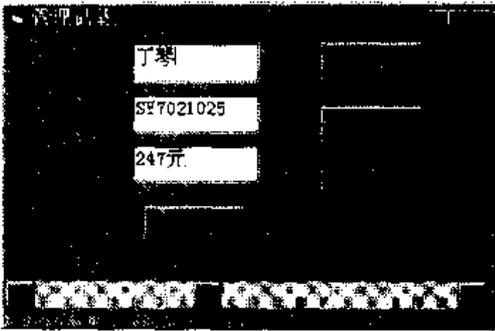


图 13.4 利用滚动条快速定位记录

记录进行定位。如图 13.4 所示。

要使用滚动条来控制记录的位置,只要注意以下几个方面即可。首先,在使用 Open 语句打开文件后,使用下面的语句设置滚动条的初始状态:

```
'滚动条的名称为 SlidBar
'取该记录总数
LastRecord = LOF(FileNum)/ Len(Salaryrec)
'设置滚动条的最大值
SlidBar.Max = LastRecord - 1
SlidBar.Value = 0 '设置滚动条的当前值
```

然后,在滚动条的 Change 文件中,添加下列语句:

```
Private Sub SlidBar_Change( )
Position = SlidBar.Value + 1
Get FileNum, Position, Salaryrec
'重新设置窗体,显示新的数据
...
End Sub
```

注意:在对文件进行增删之后,要对滚动条的 Max 属性重新设置。

13.4.3 使用二进制文件

二进制访问能提供对文件的完全控制,因为文件中的字节可以代表任何东西,例如:通过创建长度可变的记录可保存磁盘空间。当要使文件的尺寸尽量小时,应使用二进制型访问。

提示:当把二进制数据写入文件中时,使用变量是 Byte 数据类型的数据,而不是 String 变量。String 被认为包含的是字符,而二进制型数据无法正确地保存在 String 变量中。

要打开二进制的文件,应使用以下 Open 语句的语法:

```
Open Pathname For Binary As filenumber
```

通过使用二进制型访问可使磁盘空间的使用降到最小。因为二进制文件不需要固定长度的字段,类型声明语句可以省略字符串长度参数。例如可以为 SalaryMag 类型添加一个字段 Description 来记录一些附加信息:

```
Type SalaryMag
Name As String * * 8
Number As String * * 6
Salary As Integer
Description As String
End Type
```

因为 Description 字段中的内容长度是不固定的,而且往往相差很多,有的可能没有,而有的可能会很长。如果使用固定长度的 String 类型势必会造成设置字符串长度时的矛盾——太长会浪费磁盘空间,而太短又无法容纳某些很长的文字。而在二进制文件中每个记录只占用

所需的字节,所以不必为字段指定长度。当记录中的字段要包含大段的说明文件时,使用二进制文件可以节省大量的磁盘空间。

二进制文件的读写使用与读写随机文件相同的语句:

```
[Get|Put][ # ]filename, Position, Varname
```

其中 Varname 参数可以是任何类型的变量,包括可变长度的字符串以及用户自定义的类型,而 Position 参数则指明 Get 或 Put 语句要处理的位置(文件中的第一个字节位置是 1)。

二进制存取方式由于可以使用长度可变字段,所以不能随机地访问记录——必须顺序地访问记录以了解每一个记录的长度,这是进行二进制输入/输出的主要缺点。但是在这种文件模式下,可以直接查看文件中指定的字节,所以二进制模式也是唯一支持用户到文件的任何位置读写任意长度数据的方法。

为了可以同时使用随机文件和二进制文件的优点,可以使用这样的组合——当字段的长度固定或长度变化不大时,可以将这些字段存储在随机文件中,而对于长度变化很大的字段,可以保存在二进制文件中,而在随机文件的记录中设置一个字段指定它在二进制文件中的位置即可。这样,既可以利用随机文件的方便快捷,又可以大大节省磁盘空间。

13.5 文件管理

前面我们介绍了如何利用文件列表框、目录列表框、驱动器列表协同进行文件操作,以及如何操作顺序文件、随机文件和二进制文件类型,这一节我们再介绍一些其他的文件操作。

13.5.1 拷贝和移动文件

VB 提供了一个 FileCopy 语句来拷贝一个文件,FileCopy 语句的语法格式为:

```
FileCopy SrcFile, DestFile
```

其中,SrcFile 是指定被拷贝文件的文件名的确定的字符串表达式,不支持含通配符的字符串,字符串中可以包含驱动器和路径信息。用户要注意的是不能拷贝已由 VB 打开的文件。例如,以下代码可以将文件名为 filename 的文件拷贝到用户输入的文件名中去。

```
DestFile = InputBox("复制到哪里?", "复制文件")  
SrcFile = File1.filename  
FileCopy SrcFile, DestFile
```

文件的移动过程实际上也就是文件的拷贝和删除的过程,要实现文件的移动,只需将原文件拷贝到新文件中,然后再删除原文件即可。

13.5.2 文件的更名

在 VB 中文件的更名是很容易的,只需要使用 Name 语句就可以了,Name 语句的语法格式如下:

```
Name OldFileName As NewFileName
```

其中 OldFileName 是一个字符串表达式,指定已存在的文件及路径,可能包括目录或文件夹的驱动器。NewFileName 也是一个字符串表达式,它确定新文件的路径——也可能包括目

录、文件夹和驱动器,如果 NewFileName 指定的路径存在并且不同于 OldFileName 指定的路径,则 Name 语句移动文件到新的目录或文件夹中并更改文件名。如果 NewFileName 和 OldFileName 指定不同的路径和同一文件名,则 Name 语句移动文件到新的地点且保持文件名不改变。从这里可以看出,Name 语句不仅有更名能力,而且还有移动文件的能力。利用 Name 语句,用户可以从一个目录或文件夹中移动文件到另一个目录或文件夹中。但要注意的是不能移动目录和文件夹。例如,下面的语句将路径为 FileSource 的文件更名为 RetVal:

```
FileSource = File1.path & "\"& File1.FileName  
RetVal = InputBox("请输入新的文件名","更新文件名")  
Name FileSource As RetVal
```

13.6 本章小结

本章介绍了 VB 中有关文件操作的基本知识。读者学完本章后,应掌握以下内容:

1. 文件系统的基本概念和类型。
2. 文件管理的常用语句和函数。
3. 驱动器列表框、目录列表框、文件列表框三种文件操作控件。
4. 如何使用三种文件类型。
5. 文件的移动、复制和更名等基本操作技术。

有了这些知识,用户在应用程序中就可以方便、灵活地进行文件管理,同时也可以有效地与本机系统相互交换信息,方便地利用系统资源。

第 14 章 多媒体(MCI)编程

所谓多媒体,是指在计算机中同时处理声、图、文信息。VB 专业版和企业版中提供了多媒体 MCI 控件,通过 MCI 控件可以在 VB 中操纵控制各种多媒体外部设备并读取各种多媒体系统所需的文件格式。

在第 10 章我们已对如何在 VB 中使用图形作了比较详细的叙述,主要介绍了一些图形控件和画图方法。图形作为多媒体的一个重要组成部分,本章首先对图形作进一步的讨论,然后介绍如何使用音频和视频。

14.1 多媒体中的图形

图形对于 Windows 程序非常重要,不仅能使程序赏心悦目,而且能完成许多重要的功能,如提示用户,显示结果等。本节介绍几个使用图形的基本函数和控件,进一步说明如何使用图形。

14.1.1 LoadPicture 函数

用户可以用 LoadPicture 函数将图形载入到窗体的 Picture 属性、PictureBox 控件或 Image 控件。

1. 语法

LoadPicture 函数的语法形式如下:

```
LoadPicture([stringexpression])
```

其中 stringexpression 参数为被载入的图形文件名。

2. 说明

VB 可以识别的图形格式有:位图 (.bmp) 文件、图标 (.ico) 文件、行程编码 (.rle) 文件、元 (.wmf) 文件、增强的元文件 (.emf), GIF 文件以及 JPEG (.jpg) 文件。

注意: 赋值不带参数的 LoadPicture 将清除窗体、图片框及图像控件中的图形。

提示: 为了加载在 PictureBox 控件和 Image 控件中显示的图形或加载作为窗体背景的图形,必须将 LoadPicture 的返回值赋给要显示该图片的对象的 Picture 属性。例如:

```
Set Picture = LoadPicture("PARTY.BMP")
```

```
Set Picture1.Picture = LoadPicture("PARTY.BMP")
```

如果要将图标赋予窗体,则要把 LoadPicture 函数的返回值赋给 Form 对象的 Icon 属性:

```
Set Form1.Icon = LoadPicture("MYICON.ICO")
```

图标也可以被赋予除 Timer 控件和 Menu 控件外的其他控件的 DragIcon 属性,例如:

```
Set Command1.DragIcon = LoadPicture("MYICON.ICO")
```

另外,使用 LoadPicture 函数可将图形文件载入到系统剪贴板,其方法如下所示:

```
Clipboard.SetData LoadPicture("PARTY.BMP")
```

3. LoadPicture 函数示例

本例使用 LoadPicture 函数将图片加载到窗体的 Picture 属性(单击 Form 对象可装入图形到窗体),然后从 Form 对象上清除掉该图片。

程序代码如下:

```
Private Sub Form_Click ()
    Dim Msg as String ' 声明变量。
    On Error Resume Next ' 设置错误句柄。
    Height = 3990
    Width = 4890 ' 设置高度和宽度。
    Set Picture = LoadPicture("桌面.BMP")
    ' 载入位图。
    If Err Then
        Msg = "找不到.bmp 文件!"
        MsgBox Msg ' 显示错误消息。
        Exit Sub ' 如果发生错误则退出。
    End If
    Msg = "请单击'确定'以清除窗体。"
    MsgBox Msg
    Set Picture = LoadPicture() ' 清除窗体。
End Sub
```

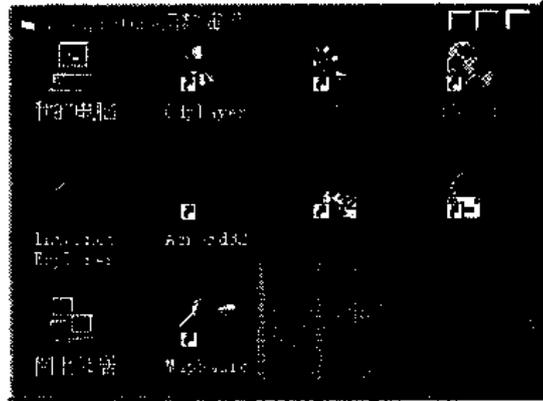


图 14.1 LoadPicture 函数示例结果

程序运行的结果如图 14.1 所示。

14.1.2 Paintpicture 方法

用户可以用 Paintpicture 方法在 Form、PictureBox 或 Printer 上绘制图形文件(图形文件的扩展名可以是 .bmp, .wmf, .emf, .ico 或 .dib)的内容。

1. 语法

Paintpicture 方法的语法如下所示:

```
object.PaintPicture picture, x1, y1, width1, height1, x2, y2, width2, height2, opcode
```

PaintPicture 方法的语法包含的各个部分的说明如下:

参 数	描 述
object	可选的。一个对象表达式,该对象一定能在“应用于”列表中找到。如果省略 object,带有焦点的 Form 对象缺省为 object。
Picture	必需的。要绘制到 object 上的图形源。Form 或 PictureBox 必须是 Picture 属性。
x1, y1	必需的。均为单精度数值,指定在 object 上绘制 picture 的目标坐标(x-轴和 y-轴)。object 的 ScaleMode 属性决定使用的度量单位。

参 数	描 述
Width1	可选的。单精度数值,指示 picture 的目标宽度。object 的 ScaleMode 属性决定使用的度量单位。如果目标宽度比源宽度 (width2) 大或小,将适当地拉伸或压缩 picture。如果该参数省略,则使用源宽度。
Height1	可选的。单精度数值,指示 picture 的目标高度。object 的 ScaleMode 属性决定使用的度量单位。如果目标高度比源高度 (height2) 大或小,将适当地拉伸或压缩 picture。如果该参数省略,则使用源高度。
x2, y2	可选的。均为单精度数值,指示 picture 内剪贴区的坐标(x - 轴和 y - 轴)。object 的 ScaleMode 属性决定使用的度量单位。如果该参数省略,则缺省为 0。
Width2	可选的。单精度数值,指示 picture 内剪贴区的源宽度。object 的 ScaleMode 属性决定使用的度量单位。如果该参数省略,则使用整个源宽度。
Height2	可选的。单精度数值,指示 picture 内剪贴区的源高度。object 的 ScaleMode 属性决定使用的度量单位。如果该参数省略,则使用整个源高度。
Opcode	可选的。是长型数值或仅由位图使用的代码。它用来定义在将 picture 绘制到 object 上时对 picture 执行的位操作(例如, vbMergeCopy 或 vbSrcAnd 操作符)。

2. 说明

通过使用负的目标高度值 (height1) 或目标宽度值 (width1), 可以水平或垂直翻转位图。

提示: 可以省略任何多个可选的尾部的参数。如果省略了一个或多个可选尾部参数, 不能在指定的最后一个参数后面使用逗号。如果想指定某个可选参数, 则必须先指定语法中出现在该参数前面的全部参数。

14.1.3 SavePicture 语句

用户可以用 SavePicture 语句从对象或控件(如果有一个与其相关)的 Picture 或 Image 属性中将图形保存到文件中。

1. 语法

SavePicture 的语法如下:

```
SavePicture picture, stringexpression
```

SavePicture 语句的语法包含下面部分:

参数	描述
picture	产生图形文件的 PictureBox 控件或 Image 控件
stringexpression	欲保存的图形文件名

2. 说明

无论在设计时还是运行时, 都可以把图形从文件加载到对象的 Picture 属性。但是如果它是位图、图标、元文件或增强元文件, 则图形将以原始文件同样的格式保存, 如果它是 GIF 或 JPEG 文件, 则将保存为位图文件。而 Image 属性中的图形总是以位图的格式保存而不管其原始格式。

3. SavePicture 语句示例

本例使用 SavePicture 语句保存画在 Form 对象的 Picture 属性中的图形。
程序代码如下：

```
Private Sub Form_Click ()  
    ' 声明变量。  
    Dim CX, CY, Limit, Radius as Integer, Msg as String  
    ScaleMode = vbPixels ' 设置比例模型为像素。  
    AutoRedraw = True ' 打开 AutoRedraw。  
    Width = Height ' 改变宽度以便和高度匹配。  
    CX = ScaleWidth / 2 ' 设置 X 位置。  
    CY = ScaleHeight / 2 ' 设置 Y 位置。  
    Limit = CX ' 圆的尺寸限制。  
    For Radius = 0 To Limit ' 设置半径。  
        Circle (CX, CY), Radius, RGB(Rnd * 255, Rnd * 255, Rnd * 255)  
        DoEvents ' 转移到其他操作。  
    Next Radius  
    Msg = "单击'确定'以保存文件到 T.BMP。"  
    MsgBox Msg  
    SavePicture Image, "T.BMP" ' 将图片保存到文件。  
End Sub
```

程序运行的结果如图 14.2 所示。



图 14.2 SavePicture 语句示例结果

14.1.4 MSChart 控件

每个 MSChart 控件都与数据网格(DataGrid 对象)相关联。这个数据网格是保存图表中数据的表。数据网格中还可包括在图中标识系列和分类的标签。设计图表应用程序时可将数据表单或数组中的数据填入或导入数据网格中。MSChart 控件支持如下特性：

- 真正的三维表示。
- 支持所有主要的图表类型。
- 通过随机数据和数据数组充填数据网格。

MSChart 控件的用途为：

- 以二维的条形图/直方图、折线图、面积图、阶梯图、组合图、饼图或散点图显示数据。
- 以三维的条形图、折线图、面积图、阶梯图或组合图显示数据。
- 通过将系列叠置在一起,显示在时间上的数据对比。
- 从数组中将数据装载到网格中。

下面介绍 MSChart 控件的主要内容。

1. 改变图表类型

图表提供了数据的图形表示。值和数据点以条形图、折线图、标记图、填充区域图或饼图形式显示。这些数据点组成系列,用唯一的颜色或填充样式进行标识。在许多图表类型中,每个系列的数据点与沿坐标轴的分类组合在一起。图表还具有标题、背景、图例、图形和脚注。

MSChart 控件刚放到窗体中时,显示的是缺省类型的图表。MSChart 控件缺省类型的图表如图 14.3 所示。

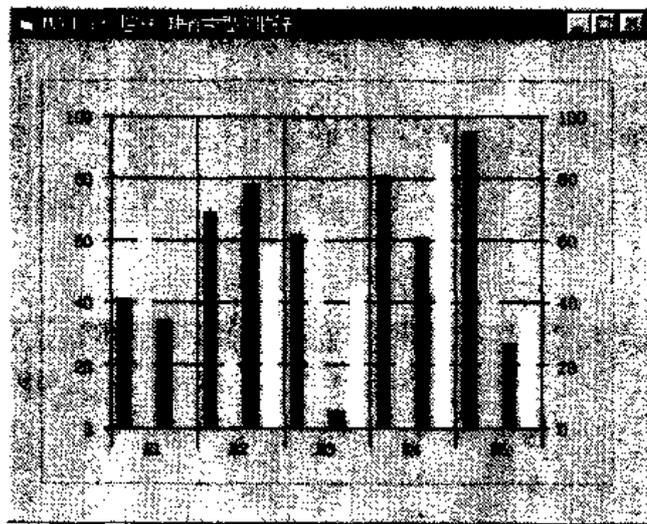


图 14.3 MSChart 控件显示的缺省类型的图表

为更形象地显示数据,可能需要改变图表的类型。改变图表类型可按如下步骤进行:

(1) 用鼠标右键单击 MSChart 控件,从弹出的快捷菜单中选择【属性】项,以显示【MSChart 属性页】对话框。如图 14.4 所示。

(2) 在【图表】选项卡的【图表类型】页中,选择【3D】单选框以显示出可用的图表类型。

(3) 在【图表类型】列表框中,选择【行】。

(4) 在【图表选项】栏中,可以选择一个或多个选项以隐藏或显示图例,隐藏或显示数据点标记,叠置系列,或者显示数据网格中行的系列数据而不是列的。

作出以上步骤后的【MSChart 属性页】对话框如图 14.5 所示。

(5) 单击【确定】按钮或【应用】按钮以新类型重新显示该图表。新的图表如图 14.6 所示。

2. 添加图表元素

除了图形以外,主要的图表元素是标题和脚注。

在图表中添加标题和脚注的步骤如下:

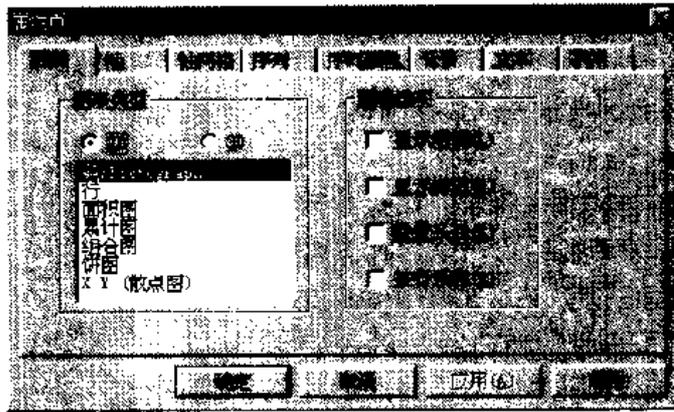


图 14.4 【MSChart 属性页】对话框

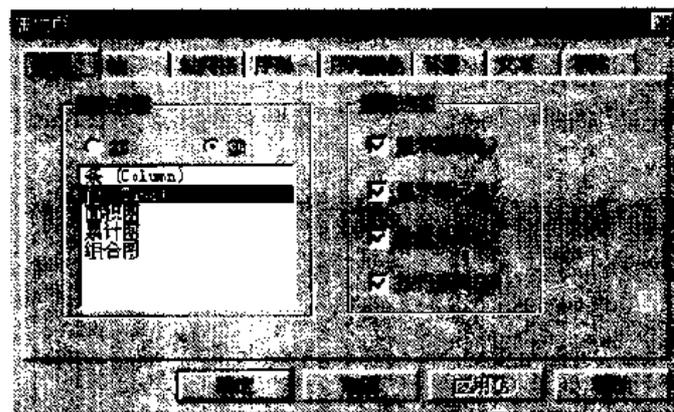


图 14.5 【MSChart 属性页】对话框

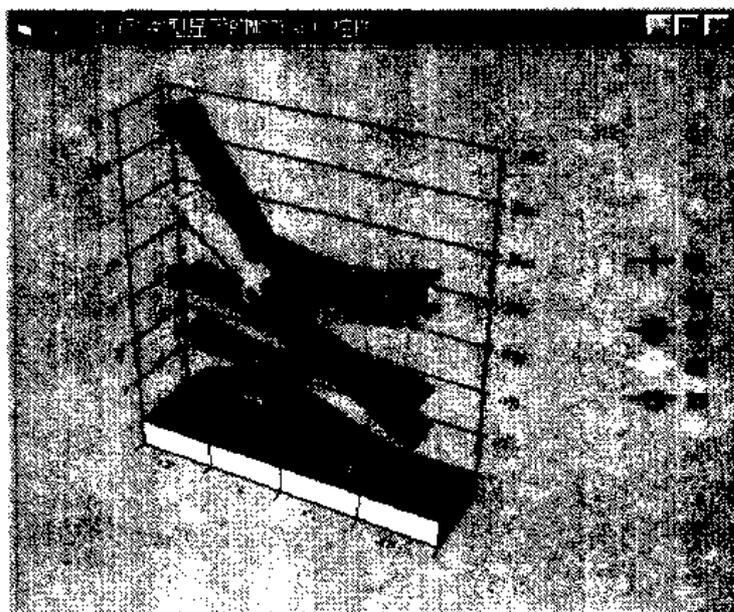


图 14.6 以【3D】、【行】类型显示的 MSChart 控件

(1) 用鼠标右键单击 MSChart 控件,从弹出的快捷菜单中选择【属性】项,以显示【MSChart 属性页】对话框。

- (2) 在【文本】选项卡中的属性名称下拉式列表框中选择名称【脚注】或某个标题类型。
- (3) 在【文本】框中键入标题或脚注。
- (4) 选择希望的文本对齐方式和文本方向。
- (5) 单击【确定】按钮或【应用】按钮重画该图表,以显示出新的图表元素。
- (6) 在【字体】选项卡中,选择图表元素中使用的字体名称、大小、样式、效果和颜色。
- (7) 单击【确定】按钮或【应用】按钮重画该图表,以显示出新设定的字体。

3. 使用组合图表

组合图表可以在一个图表中以不同的形式显示每个系列。通过以唯一方法绘制每个系列,可以使图表更引人注目,并提高图表的可读性。

注意: 使用组合图表时,必须指定在图表中显示每个系列的类型。

改变系列的类型的步骤如下:

- (1) 用鼠标右键单击 MSChart 控件,从弹出的快捷菜单中选择【属性】项,以显示【MSChart 属性页】对话框。
- (2) 在【系列】选项卡的【系列】下拉列表框中选择一个选项。
- (3) 在【系列类型】下拉列表框中,为选定的系列选择一有效的类型。
- (4) 必要时,继续指定每个系列的类型。

4. 叠置系列

在【图表】选项卡的【图表选项】栏中的选项组控制该图表是否叠置,以及特定的图表特征是否可见。可在条形图、折线图、面积图、阶梯图和组合图表中叠置系列。叠置就是将同分类的数据点一个一个地互相叠置在一起。每一线条仍旧表示相同的值。

在一个图表中叠置系列,为绘制多个系列相似的数据提供了更好的方法。通过将数据叠起来,可以在图表的一个位置上,表现出多个系列数据的改变和变化趋势。叠置数据可以消除在三维图表中常见的遮挡现象。

叠置系列的步骤如下:

- (1) 用鼠标右键单击 MSChart 控件,从弹出的快捷菜单中选择【属性】项,以显示【MSChart 属性页】对话框。
- (2) 在【图表】选项卡中,选择【图表选项】中的【叠置系列】复选框。
- (3) 单击【应用】按钮以重画该图表。
- (4) 为了取消叠置系列,再次选定【叠置系列】复选框。

5. 赋予和编辑背景

可以通过为图表本身,或图表的某个元素,加上背景来改进图表的外观。背景可以包括包围图表或图表元素的边框、图表元素后面的阴影,或者图表元素后面的填充样式。

决定了为图表元素使用背景后,需要编辑该背景以获得理想的效果。

编辑背景的方法如下:

- (1) 用鼠标右键单击 MSChart 控件,从弹出的快捷菜单中选择【属性】项,以显示【MSChart 属性页】对话框。
- (2) 在【背景】选项卡中,选择要编辑背景的特定图表对象。
- (3) 选定图表对象后,选择背景内部的颜色、填充样式和填充颜色,以及背景边框的样式、宽度和颜色。

(4) 单击【确定】按钮或【应用】按钮重画图表,以反映所做的改变。

6. 格式化底和墙

格式化底和墙可以改变用于二维图表的墙的颜色和填充样式,以及三维图表的底和墙,即用于绘制底和墙的线条的颜色,底的高度和墙的宽度。其中高度和宽度是以磅为单位的。

格式化墙和底可按以下步骤进行:

(1) 用鼠标右键单击 MSChart 控件,从弹出的快捷菜单中选择【属性】项,以显示【MSChart 属性页】对话框。

(2) 在【背景】选项卡的【属性名】列表框中选择【绘图】选项。

(3) 对颜色、填充样式和线条的宽度做必要的改变,并单击【确定】按钮或【应用】按钮重画图表,以反映所做的改变。

7. 操纵 MSChart 的数据网格

在窗体中绘制 MSChart 控件时,会创建数据网格(DataGrid 对象),并以随机数据填充。可以使用该网格的一些属性和方法来操纵它。ColumnCount 和 RowCount 属性分别决定了数据的列数和行数。ColumnLabelCount 和 RowLabelCount 属性分别设置列和行标签的层数。Column 和 Row 属性用于标识数据网格中的特定点。ColumnLabelIndex 和 RowLabelIndex 属性分别标识标签的特定行或列。ColumnLabel 和 RowLabel 属性分别改变标识行或列的标签。

下面的示例通过图表参数,将其设置为三维条形图,用随机数据填充它,并为数据网格的列设置标签。

程序代码如下:

```
Private Sub Command1_Click()  
    Dim rowLabelCount As Integer  
    Dim columnLabelCount As Integer  
    Dim rowCount As Integer  
    Dim columnCount As Integer  
    Dim DataGrid As DataGrid  
  
    Set DataGrid = MSChart1.DataGrid  
    MSChart1.chartType = VtChChartType3dBar  
  
    With MSChart1.DataGrid  
        '用方法设置 MSChart 参数。  
        rowLabelCount = 2  
        columnLabelCount = 2  
        rowCount = 6  
        columnCount = 6  
        .SetSize rowLabelCount, columnLabelCount, rowCount, columnCount  
  
        '填充随机数据  
        .RandomDataFill  
  
        '然后为第二层标签赋值。  
        Dim labelIndex as Integer
```

```

Dim column as Integer
Dim row as Integer
labelIndex = 2
column = 1
.ColumnLabel(column, labelIndex) = "Product 1"
column = 4
.ColumnLabel(column, labelIndex) = "Product 2"

row = 1
.RowLabel(row, labelIndex) = "1994"
row = 4
.RowLabel(row, labelIndex) = "1995"
End With
End Sub

```

程序运行的结果如图 14.7 所示。

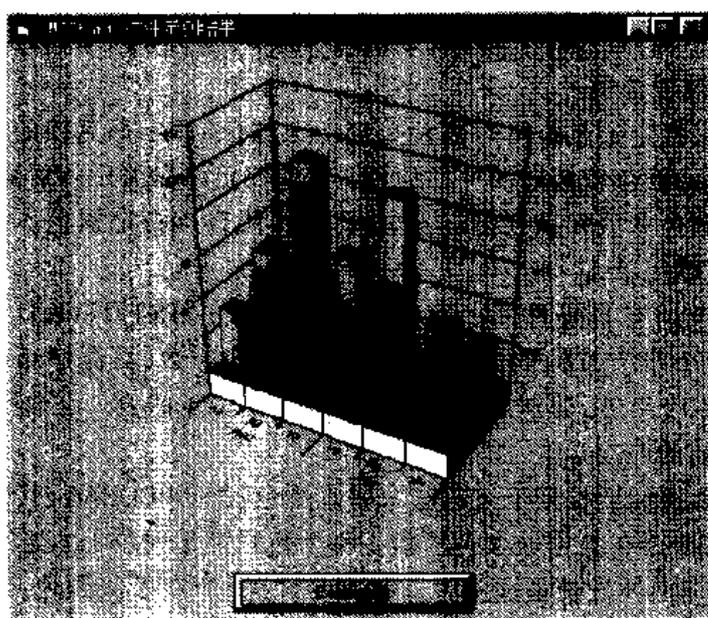


图 14.7 MSChart 控件示例结果

14.1.5 关于颜色

下面介绍影响多媒体功能的一个重要方面：颜色。

1. 使用颜色属性

VB 中的许多控件，有决定控件显示颜色的属性。请记住，这些属性中有些也适用于不是图形的控件。下表描述了这些颜色属性。

属 性	描 述
BackColor	对用于绘画的窗体或控件设置背景颜色。如果在绘图方法进行绘图之后改变属性，则已有的图形将会被新的背景颜色所覆盖。

续表

属 性	描 述
ForeColor	设置绘图方法在窗体或控件中创建文本或图形的颜色。改变 ForeColor 属性不影响已创建的文本或图形。
BorderColor	给形状控件边框设置颜色。
FillColor	为用 Circle 方法创建的圆和用 Line 方法创建的方框设置填充色。

2. 定义颜色

颜色属性可使用几种方法定义颜色值。在第 10 章的“在图形中使用颜色”一节所述的 RGB 函数,就是定义颜色的一种方法。这部分介绍的是另外两种定义颜色的方法。

(1) 使用定义的常数

当使用“对象浏览器”中列出的内部常数时,没有必要去了解这些常数是如何产生的。另外,这些内部常数也无须声明。例如:无论什么时候想指定红色,作为颜色参数或颜色属性的设置值,都可以使用常数 vbRed:

```
BackColor = vbRed
```

(2) 直接使用颜色设置值

使用 RGB 函数来指定颜色和用内部常数来指定颜色,都不是直接的,因为 VB 只是将它们解释为与它所代表的颜色较接近的一种颜色。如果自己清楚知道 VB 是如何用数值来指定颜色的话,就可以给颜色参数和属性指定一个值,这样能直接指定颜色。多数情况下,用十六进制数输入这些数值更简单。

正常的 RGB 颜色的有效范围,是从 0 到 16 777 215 (&HFFFFFF&)。每种颜色的设置值(属性或参数)都是一个四字节的整数。对于这个范围内的数,其高字节都是 0,而低三个字节,从最低字节到第三个字节,分别定义了红、绿、蓝三种颜色的值。红、绿、蓝三种成分都是用 0 到 255 (&HFF) 之间的数表示。

因此,可以用十六进制数按照下述语法来指定颜色:&HBBGGRR&,其中 BB 指定蓝颜色的值,GG 指定绿颜色的值,RR 指定红颜色的值。每个数段都是两位十六进制数,即从 00 到 FF。中间值是 80。因此,下面的数值是这三种颜色的中间值,指定了灰颜色:&H808080&。将最高位设置为 1,就改变了颜色值的含义:颜色值不再代表一种 RGB 颜色,而是一种从 Windows“控制面板”指定的环境范围颜色。这些数值对应的系统颜色范围是从 &H80000000 到 &H80000015。

注意: 尽管可以指定 1,600 万种以上的不同颜色,但并不是所有的系统都能精确地显示出来。关于 Windows 如何指定颜色的详细内容,请参阅本章后面的“使用 256 种颜色”。

3. 使用系统颜色

应用程序中设置控件或窗体的颜色时,可以不指定颜色值,而用操作系统指定的颜色。如果指定了操作系统的颜色,当应用程序的用户改变计算机上的系统颜色值时,应用程序将自动地反映用户所指定的颜色值。

每一种系统颜色,既有所定义的常数也有直接的颜色设置值。对系统颜色来说,其直接颜色设置值的高位字节与普通 RGB 颜色的高位字节是不同的。对于 RGB 颜色来说,其高位字节为 0,而对于系统颜色来说,其高位字节为 80,剩下的数字则指的是某一特定的系统颜色。

例如: &H8000002& 这个十六进制数, 是用来指定一个活动窗口的标题颜色。

设计时, 通过属性窗口选择颜色属性时, 选择“系统”选项卡, 就能够选择系统设置值, 可自动转换成十六进制值。也可在“对象浏览器”中寻找系统颜色的定义常数。

4. 使用 256 种颜色

基于具有能处理 256 色或 256 色以上的视频适配器和显示驱动程序的系统, VB 可支持 256 种颜色。对于多媒体应用程序, 或对于那些需要显示接近相片质量图像的应用程序来说, 能同时显示 256 色的功能, 是特别有价值的。

在以下场合, 可显示 256 色的图像, 也可为图形方法定义高达 256 色:

- 窗体
- PictureBox 控件
- Image 控件(仅显示图像)

注意: Windows 的元文件, 不支持 256 种色。VB 用缺省的 VGA 16 色的调色板来显示元文件。

5. 调色板

在 VB 应用程序中, 调色板提供了支持 256 种颜色的基础。在关于调色板的讨论中, 了解不同类型调色板之间的关系很重要。硬件调色板包含了 256 个记录项, 它们定义了将在屏幕上显示的实际 RGB 值。系统中间色调的调色板是一套预定义的 256 种 RGB 值, 对 Windows 本身有效。逻辑调色板是一套可到达 256 种 RGB 的值, 它们包含在位图或其他图像中。

Windows 可使用硬件调色板中的 256 种颜色绘图。其中有 20 种颜色被称为静态颜色, 是由系统保护的, 应用程序不能将其改变。静态颜色包括缺省的 VGA 调色板中的 16 种颜色(与 VB QBColor 函数定义的颜色一样), 外加四种不同深度的灰色。系统中间色调的调色板, 通常包含这些静态颜色。

前景窗口(有焦点的窗口), 将决定硬件调色板中剩下的 236 种非静态颜色。每当硬件调色板改变时, 使用这些颜色的所有背景窗口都要重画。如果背景窗口的逻辑调色板的颜色与当前硬件调色板的颜色不完全匹配, Windows 会赋予一个最接近的匹配值。

6. 256 色图像的显示

如果显示硬件和软件能支持 256 种颜色的话, 窗体、图片框和图像控件能自动地用 256 色显示图像。如果用户系统支持的颜色少于图像所需的颜色, VB 尽可能地将所需颜色映射到最相近的颜色上。

对于真彩色(1,600 万种颜色)的显示, VB 总是使用正确的颜色。对于单色或 16 色的显示, VB 会使背景色和用 FillColor 属性设置的颜色抖动。抖动是用来模拟从视频适配器和显示驱动设备得不到的颜色的过程。

7. 用调色板绘图

对于 256 色的视频驱动程序, 通过图形方法可使用高达 256 种的颜色。按照缺省规定, VB 中有效的 256 色是系统中间色调的调色板中的颜色。

尽管缺省调色板对于 VB 来说是系统中间色调的调色板, 但使用窗体、用户控件和用户文档的 PaletteMode 属性和 Palette 属性, 也可控制颜色的显示。这种情况下, 除了颜色要在硬件调色板中与最接近色匹配之外, 颜色的匹配关系是一样的。

14.2 使用音频和视频

Windows 程序的魅力之一是它的多媒体效果,VB 作为 Windows 编程的最佳工具之一,其多媒体功能是非常强大的,本节将详细介绍在 VB 中如何使用音频和视频以实现 VB 的多媒体编程。

14.2.1 使用 Animation 控件

Animation 控件如图 14.8 所示。

Animation 控件显示无声的音频视频动画。AVI 动画类似于电影,由若干帧位图组成。

注意:虽然 AVI 动画可以有声音,但这样的动画不能在 Animation 控件中使用,如果试图装载这样的文件将会产生错误。在该控件中只能使用无声的 AVI 动画。要播放有声的 .avi 文件,可使用 Multimedia (MCI) 控件。

提示:在运行时,Animation 控件是不可见的。

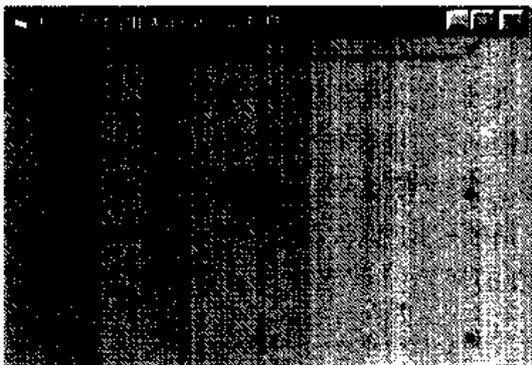


图 14.8 放在窗体上的 Animation 控件

下面详细介绍 Animation 控件。

1. Animation 控件的用途

- 在对话框中显示出操作的长短和特征。
- 播放有关应用程序的无声动画,提供使用指导。
- 使用户能够播放放入该控件的文件。

2. 基本操作:Open、Play、Stop 和 Close 方法

在使用该控件时,可用 Open 方法打开扩展名为 avi 的文件,用 Play 方法进行播放,用 Stop 方法停止播放。在动画播放完毕以后,可用 Close 方法

关闭该文件。在打开新文件之前不必关闭旧文件。

3. 程序示例

下面的代码使用了两个 CommandButton 控件 cmdPlay 和 cmdStop,以及名为 dlgOpen 的 CommonDialog 控件。将 cmdPlay 的标题设置为“打开并播放”。CommandButton 控件 cmdStop 的标题设置为“停止”。

程序代码为:

‘配置一个 CommonDialog 控件,以便用户找到要播放的 .avi 文件。

‘CommonDialog 控件的名称是“dlgOpen”。

‘Animation 控件的名称是“anmAVI”。

```
Private Sub Form_Click()  
    dlgOpen.Filter = "avi 文件 (*.avi)|*.avi"  
    dlgOpen.ShowOpen  
    anmAvi.Open dlgOpen.FileName  
    anmAVI.Play
```

End Sub

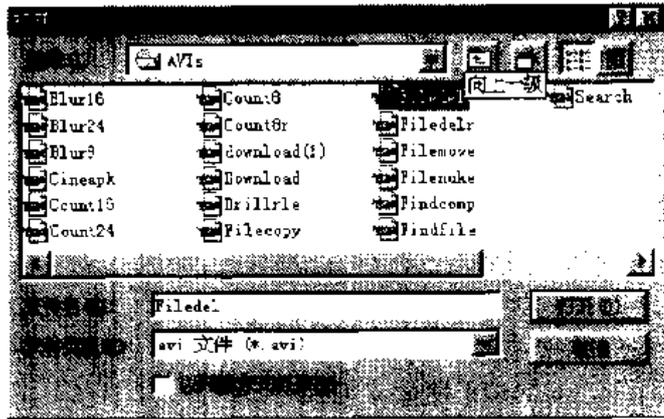


图 14.9 Animation 控件示例,打开 avi 文件

程序运行过程和结果如图 14.9 和 14.10 所示。

下面的代码停止播放视频动画:

```
Private Sub cmdStop_Click()
    annAVI.Stop
End Sub
```

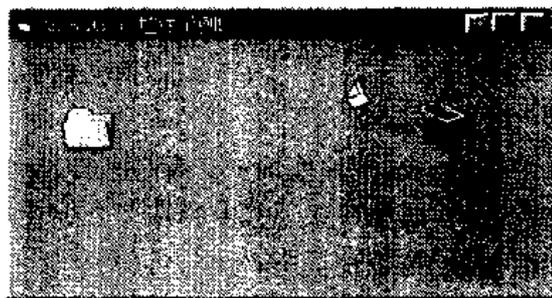


图 14.10 Animation 控件示例:播放 avi 文件

14.2.2 Multimedia MCI 控件

Multimedia 控件是通过多媒体控制接口 (MCI)对多媒体设备进行控制。Multimedia 控件可用于管理多媒体控制接口(MCI)设备。这些设备有:声卡、MIDI 发生器、CD-ROM 驱动器、音频播放器、视盘播放器和视频磁带录放器。

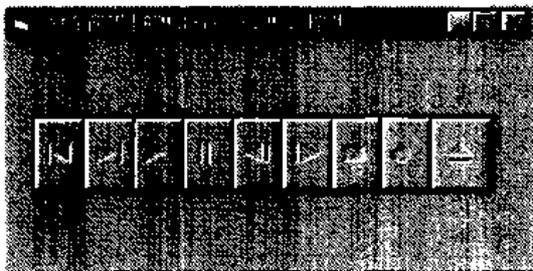


图 14.11 放在窗体上的 Multimedia MCI 控件

下面详细介绍 Multimedia MCI 控件。

1. 用途

管理 MCI 设备的录制和播放。

2. 多媒体的要求和支持的设备类型

哪些按钮可用, Multimedia 控件提供哪些功能,取决于特定计算机的硬件和软件配置。例如,应用程序中使用了特定的多媒体设备和驱动程序,则它们必须已安装在该机器中。

在 Windows 95 和 Windows NT 操作系统中提供了支持多种多媒体设备(比如音频和视频文件)的驱动程序。其他设备,如数字音频磁带机或图像扫描仪需要独立的驱动程序,这些驱动程序通常由制造商提供。

设备分为两种:简单的和复合的。简单的多媒体设备不需要数据文件即可播放。例如,打

Multimedia 控件具有一组执行 MCI 命令的下压式按钮。这些命令与通常的 CD 机或录像机上的命令(功能)很相象。

Multimedia 控件如图 14.11 所示。

从左到右,这些按钮依次定义为前一个、下一个、播放、暂停、向后步进、向前步进、停止、录制和弹出。

开视频或音频 CD 播放器后,可通过“曲目”进行播放、回绕和快进。而复合设备则必须通过数据文件才能播放。

下表列出了 Multimedia 控件支持的部分设备。那些同时列出了对应文件类型的是复合设备。

下面是设备类型和文件类型的对应描述:

animation	动画设备
cdaudio	音频 CD 播放器
dat	数字音频磁带播放器
sequencer	.mid MIDI 发生器
ver	视频磁带录放器
video	.avi 视频文件
videodisc	视盘播放器
waveaudio	.wav 播放数字波形文件的音频设备,比如 PC 的声卡

3. MCI 命令

Multimedia 控件使用一套高层次的、与设备无关的命令,被称为媒体控制接口命令,它们可控制多种多媒体设备。其中的许多命令直接与 Multimedia 控件的按钮对应。例如,Play 命令就与【播放】按钮相对应。

Multimedia 控件本质上是该命令集的 VB 接口。如 Play 或 Close 等命令在 Win32 API 的 MCI 命令结构中都有等价命令。例如,Play 对应 MCI_PLAY。下表列出了 Multimedia 控件使用的 MCI 命令。

命 令	MCI 命令	描 述
Open	MCI_OPEN	打开 MCI 设备
Close	MCI_CLOSE	关闭 MCI 设备
Play	MCI_PLAY	用 MCI 设备进行播放
Pause	MCI_PLAY 或 MCI_RESUME	暂停播放或录制
Stop	MCI_STOP	停止 MCI 设备
Back	MCI_STEP	向后步进可用的曲目
Step	MCI_STEP	向前步进可用的曲目
Prev	MCI_SEEK	使用 Seek 命令跳到当前曲目的起始位置
Seek	MCI_SEEK	向前或向后查找曲目
Record	MCI_RECORD	录制 MCI 设备的输入
Eject	MCI_SET	从 CD 驱动器中弹出音频 CD
Save	MCI_SAVE	保存打开的文件

提示: 如果在前一 Prev 命令执行后三秒内再次执行,则跳到前一曲目的起始位置;或者如果已在第一个曲目,则跳到第一个曲目的起始位置。

在 VB 中,这些命令用 Multimedia 控件的 Command 属性启动。例如:

```
MMControl1.Command = "Open"
```

虽然 Multimedia 控件实现的 MCI 命令集,在大多数情况下是足够用的;然而直接使用 Win32 API,可以提供高级的编程函数和技术。

4. 对 Multimedia 控件进行编程

通过设置 Enabled 和 Visible 属性,可使 Multimedia 控件在运行时可见或不可见。按照缺省规定,Enabled 和 Visible 属性被设置为 True,这样该控件在运行时就是可见的。

提示:如果不希望通过 Multimedia 控件上的按钮直接与用户交互,而希望使用该控件以求实现它的多媒体功能,那么可将 Visible 属性设置为 False。应用程序使用或不使用用户交互,都可以对 MCI 设备进行控制。

要使其中的单个按钮可见或不可见、有效或无效,可以设置该按钮对应的 Visible 和 Enabled 属性。例如,Back 按钮中的 BackEnabled 和 BackVisible 属性。九个按钮中的每一个都有对应的这些属性。

注意:在绝大多数情况下,这些按钮的缺省功能已足以管理 MCI 设备。然而, Multimedia 控件还包含一些运行时属性,允许添加或重定义按钮命令。

Notify、NotifyMessage 和 NotifyValue 属性提供了有价值的反馈信息,表明某个命令出错或完成。

将 Multimedia 控件放置到窗体中后,不管它被设置为可见的还是不可见的,第一步都是要访问 MCI 设备。为做到这一点,需要设置一些运行时才可用的属性。例如,如下设置媒体设备的初始化属性值:

```
MMControl1.Notify = False
MMControl1.Wait = True
MMControl1.Shareable = False
MMControl1.DeviceType = "CDAudio"
```

提示:如果将 Notify 属性设置为 True,则在下一命令完成时,将产生 Done 事件。Done 事件提供了很有用的反馈信息,以指出该命令成功还是失败。Wait 属性指定 Multimedia 控件是否等到 play 命令执行完毕,才将控制权还给应用程序。Shareable 属性限制或允许其他应用程序或进程使用该媒体设备。DeviceType 属性被用来指定 MCI 设备的类型。

最后,Open 命令用来打开 MCI 设备。例如:

```
MMControl1.Command = "Open"
```

当该控件是可见的时候,设置这些属性,并使用 Open 命令,将激活由该 MCI 设备支持的 Multimedia 控件的下压式按钮。例如,打开 cdaudio 设备将激活【前一个】、【下一个】、【播放】和【弹出】按钮。按下【播放】后,【停止】和【暂停】按钮被激活。

在一个窗体中可以加入多个 Multimedia 控件的实例,以提供对多个 MCI 设备的并行控制。对每个设备只能使用一个 Multimedia 控件。

5. 管理多媒体资源

要正确地管理多媒体和系统资源,就必须在退出应用程序前将打开的 MCI 设备关闭。在包含 Multimedia 控件的窗体卸载时,可在该窗体的 Form_Unload 过程中使用下面的语句,以关闭正打开着的 MCI 设备。

```
Private Sub Form_Unload (Cancel as Integer)
```

```
Form1.MMControl1.Command = "Close"
```

```
End Sub
```

下表列出了控件的一些属性及其说明：

属 性	说 明
Mode	表示当前多媒体设备的状态
Filename	表示当前设备打开的多媒体文件名
Command	存放对设备的命令
DeviceType	当前设备的设备类型, MCI 控件可根据文件扩展名自动设置设备类型

下面的示例演示了打开一台使用兼容数据文件的 MCI 设备的过程。将这些代码放到 Form_Load 过程, 应用程序就可以使用 Multimedia MCI 控件来对文件 Gong.wav 进行播放、记录和倒带。在试运行这个示例之前, 首先应创建一个包含 Multimedia MCI 控件的窗体。

程序代码如下：

```
Private Sub Form_Load ()  
    设置打开 MCI 的有关属性  
    MMControl1.Notify = FALSE  
    MMControl1.Wait = TRUE  
    MMControl1.Shareable = FALSE  
    MMControl1.DeviceType = "WaveAudio"  
    MMControl1.FileName = "C:\WINDOWS\MMDATA\GONG.WAV"  
  
    '打开 MCI WaveAudio 设备。  
    MMControl1.Command = "Open"  
End Sub
```

为了正确管理多媒体资源, 在退出应用程序之前, 应该关闭那些已经打开的 MCI 设备。将下面的语句放到 Form_Unload 过程, 那么在退出包含 Multimedia MCI 控件的窗体之前, 就可以关闭那些已经打开的 MCI 设备。例如：

```
Private Sub Form_Unload (Cancel As Integer)  
    MMControl1.Command = "Close"  
End Sub
```

在窗体中加上通用对话框、按钮和 MCI 控件。

```
Private Sub Command1_Click()  
    CommonDLG.filename = ""  
    CommonDLG.ShowOpen  
    If CommonDLG.filename <> "" Then  
        MCI.filename = CommonDLG.filename  
        MCI.Command = "open"  
    End If  
End Sub
```

程序运行过程和结果如图 14.12 和 14.13 所示。

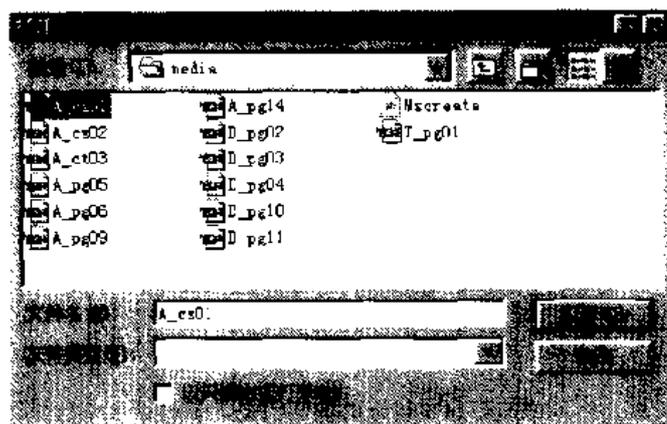


图 14.12 Multimedia MCI 控件示例:打开 .avi 文件

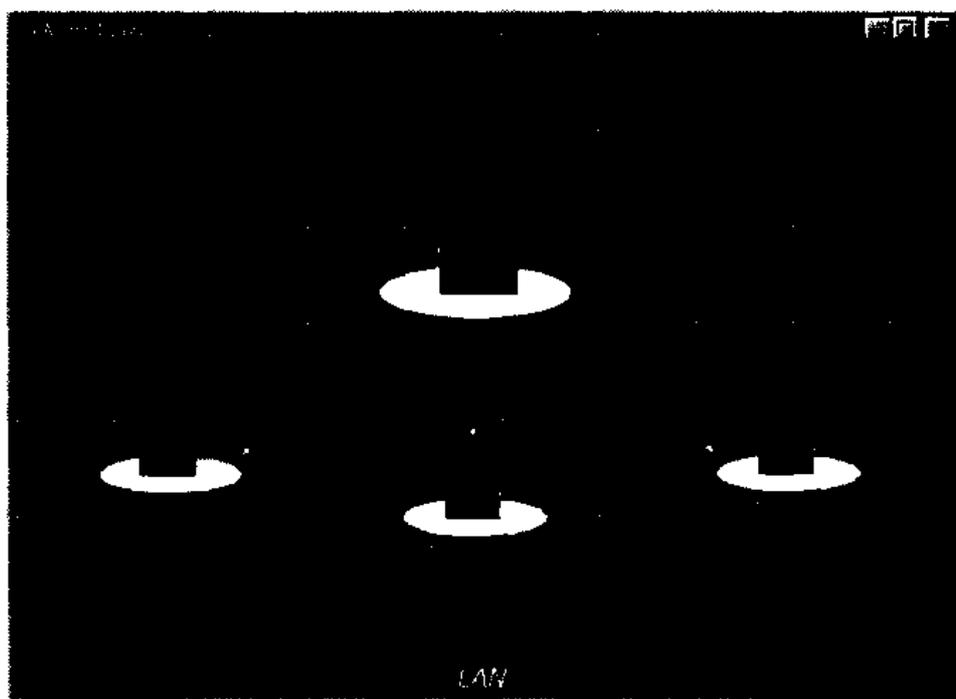


图 14.13 Multimedia MCI 控件示例结果

14.2.3 MCIWnd 控件

MCIWnd 控件在 MCI 控件的基础上更好地提供了对视频的支持,使用户在指定的窗体内可以任意地放大或缩小播放窗口。

MCIWnd 控件运行时如图 14.14 所示。

提示:除了 Mode, Filename, Command, DeviceType 等属性外,还有如下几个重要属性:

Volume:音量大小,缺省 = 1000

Speed:播放速度

Repeat:是否循环播放, = TRUE 循环播放



图 14.14 运行时的 MCIWnd 控件

14.3 本章小结

读者学完本章后,应对多媒体有初步的了解。下面是本章内容及应掌握的程度:

1. 熟练使用图形的基本函数和控件。
2. 熟练使用多媒体控件,包括 Animation 控件,MCI 控件和 MCIWnd 控件。
3. 理解 MCI 命令。
4. 了解几种关于颜色的定义。

VB 的多媒体编程可以使程序变得丰富多彩,读者会觉得很有意思。

第 15 章 数 据 库

本章将主要介绍几个重要的数据库控件的编程以及 Microsoft Jet 数据库引擎及其编程模式:数据访问对象 (DAO)。

15.1 数据库编程概述

本节介绍了关系数据库的一些基本概念,以及用于数据库编程的数据定义语言 (DDL)、数据操作语言 (DML),这些基本概念是数据库编程的基础。

15.1.1 关系数据库概述

关系数据库模型将数据表示为表的集合。通过建立简单表之间的关系来定义结构,而不是根据数据的物理存储方式建立在数据中的关系。

1. 表、字段和记录

不管它在数据库文件中的物理存储方式如何,表都可以看作一组行和列,与电子表格的行和列类似。在关系数据库中,行被称为记录,而列则被称为字段。

2. 键

键就是表中的字段(或多个字段),它(们)为快速检索而被索引。键可以是唯一的,也可以是非唯一的,取决于它(们)是否允许重复。唯一键可以指定为主键,用来唯一标识表的每行。

3. 关系

数据库可以由多个表组成,表与表之间可以用不同的方式相互关联。

4. 规范化

数据库设计者的任务就是组织数据,而组织数据的方法,应能消除不必要的重复,并为所有必要信息提供快速查找路径。为了达到这种目标而把信息分离到各种独立的表中去的过程,叫做规范化。

提示:规范化可以用许多指定的规则 and 不同级别的范式来进行的复杂过程。大多数简单数据库的规范化可以用下面简单的经验规则来完成:包含重复信息的表必须分成独立的几个表来消除重复。

15.1.2 VB 数据库体系结构

VB 数据库体系是由用户界面、数据库引擎和数据仓库三层组成的层次结构。

1. Microsoft Jet 数据库引擎

VB 提供了基于 Microsoft Jet 数据库引擎(即驱动 Microsoft Access 的数据库引擎)的数据访问能力。该 Jet 引擎处理存储、检索、更新数据的结构,并提供了功能强大的、面向对象的 DAO 编程接口。

VB 数据库应用程序有三个部分,如图 15.1 所示。

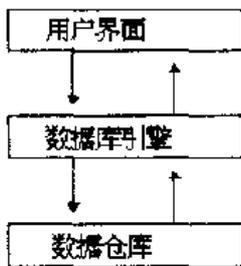


图 15.1 VB 数据库应用程序的三个部分

提示：数据库引擎存在于程序和物理数据库文件之间。这使用户与正在访问的特定数据库无关。不管这个数据库是本地的 VB 数据库，还是所支持的其他任何数据库格式，所使用的数据库访问对象和编程技术都是相同的。

2. 用户界面和应用程序代码

用户界面是用户所看见的并用于交互的界面。它包括显示数据并允许用户查看或更新数据的窗体。驱动这些窗体的是应用程序的 VB 代码，包括用来请求数据库服务的数据库访问对象和方法，比如添加或删除记录、或执行查询等。

如上所述，这些服务请求不是直接对物理数据库文件的，而是向 Jet 数据库引擎提出的，该引擎执行对数据仓库请求的操作，并向应用程序返回所需要的结果。

3. 数据仓库

数据仓库是包含数据库表的一个或多个文件。对于本地的 VB 或 Microsoft Access 数据库来说，就是 .mdb 文件。对于不同的数据库，它可能是包含 .dbf 文件或其他扩展名文件的目录。或者，应用程序可能会访问保存在几个不同的数据库文件或格式中的数据。在任何情况下，数据仓库本质上都是被动的。它包含数据，但它不对数据或用数据做任何事情。那是数据库引擎的任务。

4. 分类

所有数据库编程任务可分为如下两类：

- 数据定义语言 (DDL)
- 数据操作语言 (DML)

这并不意味着包含两种独立的语言。它是将语言元素分组的简单方法——在这种情况下，语言元素指的是数据库访问对象、属性和方法——它们被用来定义数据库的结构以及用来操作数据库的各个部分。

(1) 数据定义

DDL 由那些用来定义和创建数据库本身的属性和方法组成，包括它的表、字段、关系等等。在传统的数据库术语中，这组定义构成了数据库的结构描述。

定义数据库通常是一次性操作，因为 Jet 引擎把定义数据库结构描述的数据访问对象存储在数据库本身之内。一旦创建好数据库，就没有必要为访问它而指定其结构。简单地打开数据库就可带来在 DAO 接口的管理之下它的所有对象，包括那些定义其结构的对象。

(2) 数据操作

DML 由用来编写访问和操作现存数据库的应用程序的属性和方法组成。这些属性和方法包括查询数据库、在表中定位、执行更新和添加或删除记录等工具。

如果正在以独占方式使用另一个应用程序已创建好的数据库，那么应用程序的编程就可能完全在 DML 范围内。弄清 DML 方法，将增加对数据库结构的了解，并能成为更加灵活的程序员。

15.2 数据库相关控件及其编程

用数据库访问对象(DAO)编程有些难度，所以先介绍 Data 控件和一些数据库绑定控件的编

程,稍后再介绍 DAO 编程。

15.2.1 DATA 控件

为使数据库的编程更简便,VB 提供了一些功能很强大的与数据库相关的控件。下面将详细介绍这些控件及其用法。

1. 使用 DATA 控件

用户可以用 DATA 控件的 Recordset 对象来提供对存储在数据库中数据的访问。Data 控件允许从一个记录移动到另一个记录,并显示和操纵来自被连结控件的记录的数据。

可以使用 Data 控件来执行大部分数据访问操作,而根本不用编写代码。与 Data 控件相连结的数据绑定控件自动显示来自当前记录的一个或多个字段的数据,或者,在某些情况下,显示来自当前记录旁边的一个记录集合中的一个或者多个字段中的数据。Data 控件在当前记录上执行所有操作。

如果 Data 控件被指示移动到一个不同的记录,则所有被连结的控件自动把当前记录的任何改变传递给 Data 控件以保存在数据库中。Data 控件移动到被指定的记录,同时把当前记录中的数据传回被连结的控件,并在那里显示。

注意: Data 控件自动处理一些意外事件包括空记录集,添加新记录,编辑和更新现有记录,处理某些类型的错误。然而,在更复杂的应用程序里,则需要捕获 Data 控件不能处理的某些错误类型。

(1) 被连结的控件

当与 Data 控件相连结时,DBList、DBCombo、DBGrid 和 MSFlexGrid 控件都能管理记录集合。所有这些控件都允许一次显示或操作几个记录。

内部的 Picture、Label、TextBox、CheckBox、Image、OLE、ListBox 和 ComboBox 控件也是数据绑定的,能和由 Data 控件管理的 Recordset 的一个字段相连结。

(2) 操作

一旦应用程序开始,VB 就用 Data 控件属性打开选定的数据库,创建 Database 对象和创建 Recordset 对象。Data 控件的 Database 和 Recordset 属性引用新创建的,可独立于 Data 控件操作的 Database 和 Recordset 对象——有或没有被连结的控件。Data 控件在其所在窗体的初始化 Form_Load 事件之前被初始化。

提示: 可以用鼠标操纵 Data 控件,由一个记录移动到另一个记录或移动到 Recordset 的开始或结尾。

EOFAction 和 BOFAction 属性决定了当用户使用鼠标移动到 Recordset 的开始或结尾时将发生的事情。不能将焦点置于 Data 控件上。

(3) 数据访问对象

在过程中可以使用由 Data 控件创建的 Database 和 Recordset 数据访问对象。每个 Database 和 Recordset 对象都有自己的属性和方法,可以编写使用这些属性和方法的过程来操纵数据。

例如,Recordset 对象的 MoveNext 方法把当前记录移动到 Recordset 中的下一个记录。要调用这个方法,可以使用此代码:

```
Data1.Recordset.MoveNext
```

(4) 应用

在应用 DATA 控件时,首先要设置好两个最重要的属性:

DatabaseName:返回或设置 Data 控件的数据源的名称及位置。

RecordSource:返回或设置 Data 控件的基本表、SQL 语句或 QueryDef 对象。

下面举一个示例程序说明如何使用 DATA 控件:

```
Sub Form_Click ()
    Dim Td As TableDef
    ' 设置数据库文件。
    Data1.DatabaseName = "BIBLIO.MDB"
    Data1.Refresh ' 打开数据库。
    ' 读入并输出数据库中每个表的名称。
    For Each Td in Data1.Database.TableDefs
        Print Td.Name
    Next
End Sub
```

2. 数据库绑定控件 DBGrid

数据库绑定控件 DBGrid 显示,并允许对代表 Recordset 对象中记录和字段的一系列行和列的数据进行操作。它的最重要的属性是 DataSource,用来设置一个指定 DBGrid 控件的值,通过这个控件将当前控件连接到数据库上。

下面用 VB 的数据窗体向导来生成一示例程序,以解释如何运用 Data 控件和 DBGrid 控件。其步骤如下:

(1) 打开【工程】菜单,单击【添加窗体】,出现【添加窗体】对话框,如图 15.2 所示作出选择,单击【打开】按钮。

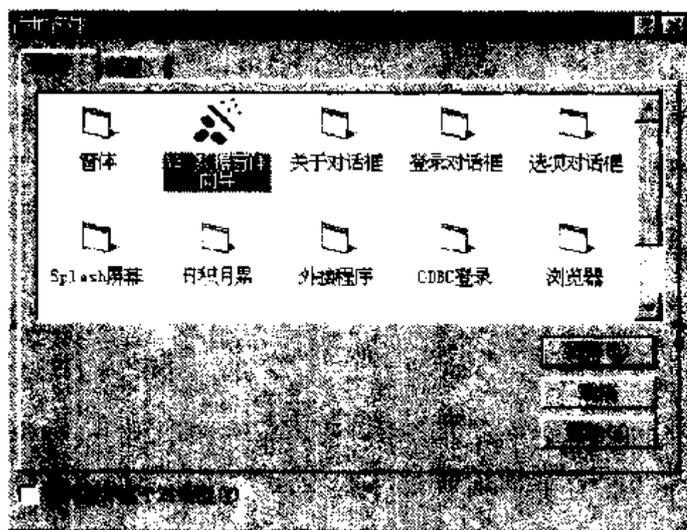


图 15.2 【添加窗体】对话框

(2) 此时出现【数据窗体向导—介绍】对话框,如图 15.3 所示。单击【下一步】按钮。

(3) 此时出现【数据窗体向导—数据库类型】对话框,如图 15.4,选择 ACCESS 数据库,单击下一步。

(4) 此时出现【数据窗体向导—数据库】对话框如图 15.5 所示,单击【浏览】按钮,从【打开

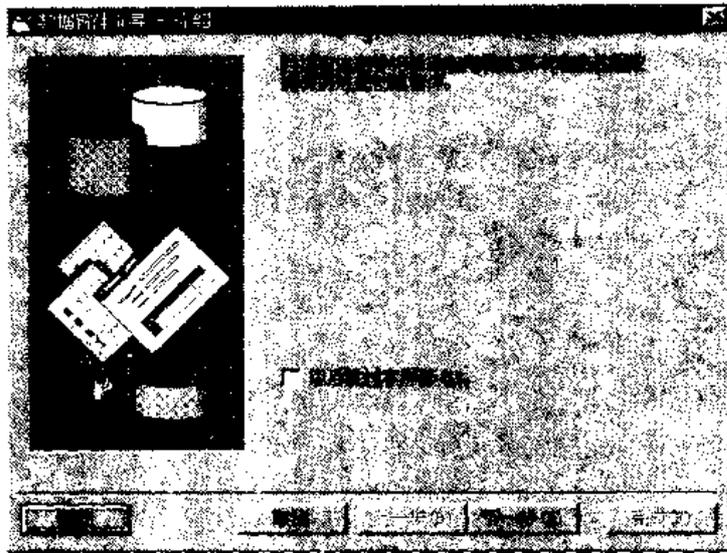


图 15.3 【数据窗体向导—介绍】对话框

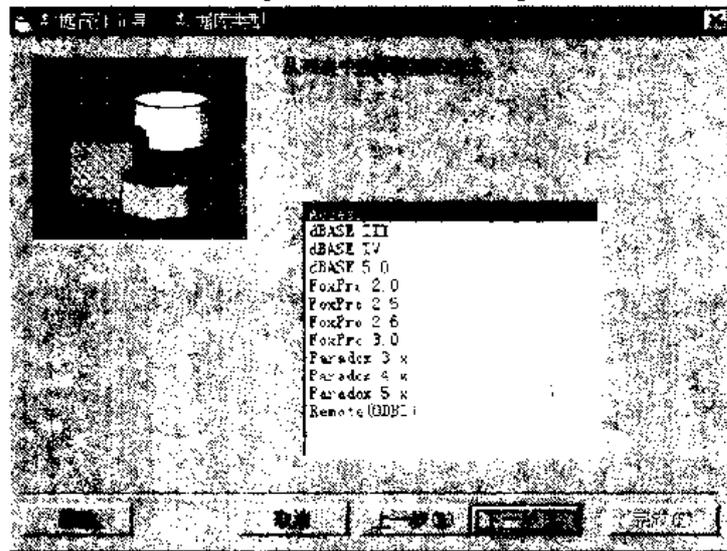


图 15.4 【数据窗体向导—数据库类型】对话框

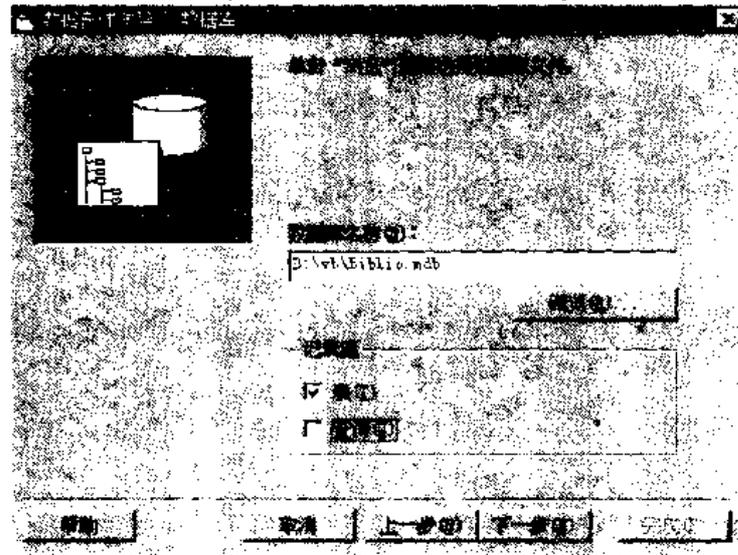


图 15.5 【数据窗体向导—数据库】对话框

文件]对话框中选择 VB \ Biblio.mdb,在记录源中选择【表】复选框,单击【下一步】按钮。

(5) 此时出现【数据窗体向导—窗体】对话框,如图 15.6 所示,在【窗体布局】中选择【网格】单选框,单击【下一步】按钮。

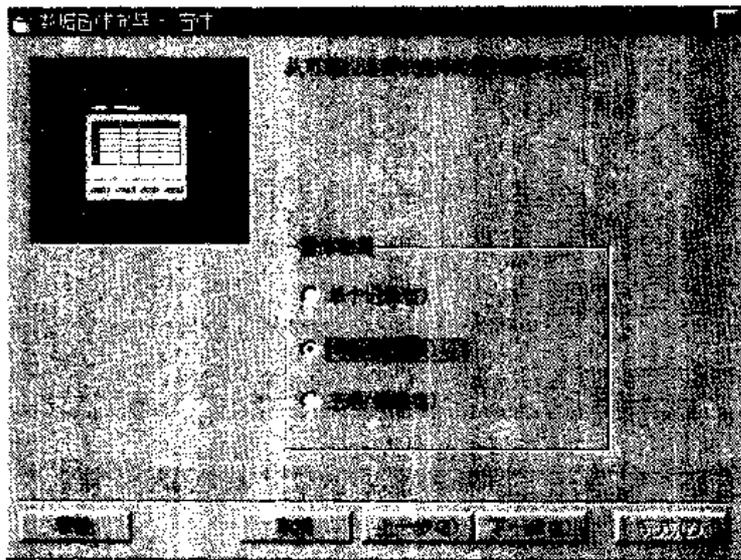


图 15.6 【数据窗体向导—窗体】对话框

(6) 此时出现【数据窗体向导—记录源】对话框,如图 15.7 所示在记录源中选择 Authors,单击“>>”选择所有的字段,单击【下一步】按钮。

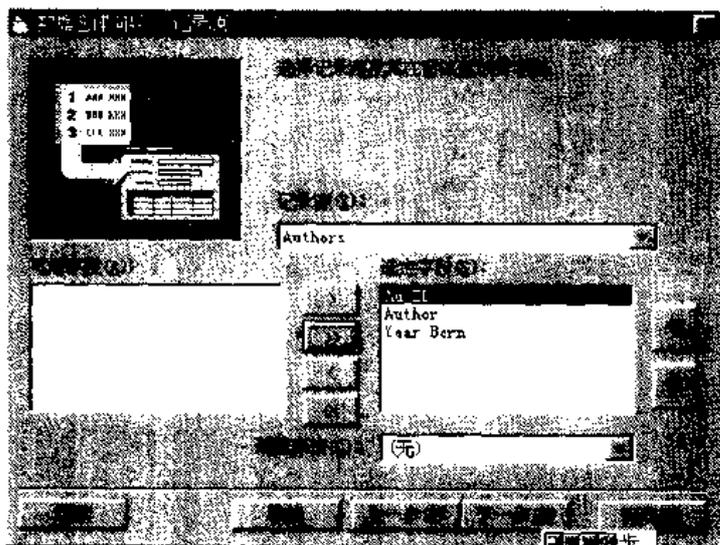


图 15.7 【据窗体向导—记录源】对话框

(7) 此时出现【数据窗体向导—控件选择】对话框,如图 15.8 所示选择所有控件,单击【下一步】按钮。

(8) 此时出现【数据窗体向导—完成】对话框,如图 15.9 所示窗体取名为“我的第一个数据库程序”,单击【完成】按钮。

(9) 在【工程】菜单的选择【属性】项,打开【工程属性】对话框,设置“我的第一个数据库程序”作为启动窗体。

运行程序后,结果如图 15.10 所示。

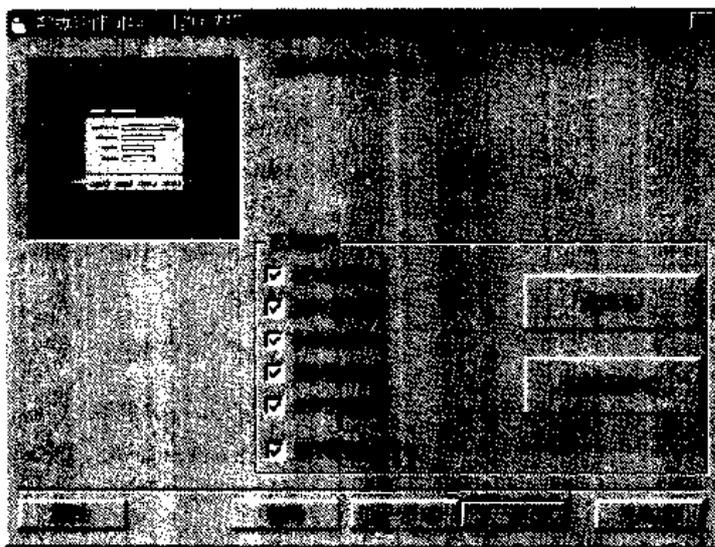


图 15.8 【数据窗体向导—控件选择】对话框

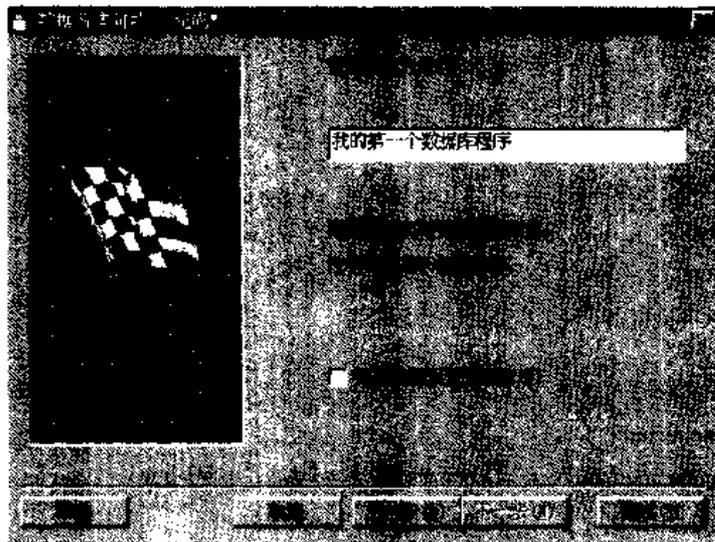


图 15.9 【数据窗体向导—窗体】对话框

下面让我们分析一下代码,首先看主要属性设置。

(1) Data 控件的属性设置

```
Name = "DatPrimaryRS"
Connect = "Access"
DatabaseName = " D: \ vb \ Biblio.mdb"
RecordSource = "select [ Au_ ID], [ Author ], [ Year Born] from [ Authors]"
```

注意: 对于属性设置,在属性设置窗口中设置时不用引号,但在编程时必须用引号,因为上述属性都是字符串类型。

(2) DBGrid 控件的属性设置

```
Name = "GridDataGrid"
DataSource = "DatPrimaryRS"
```

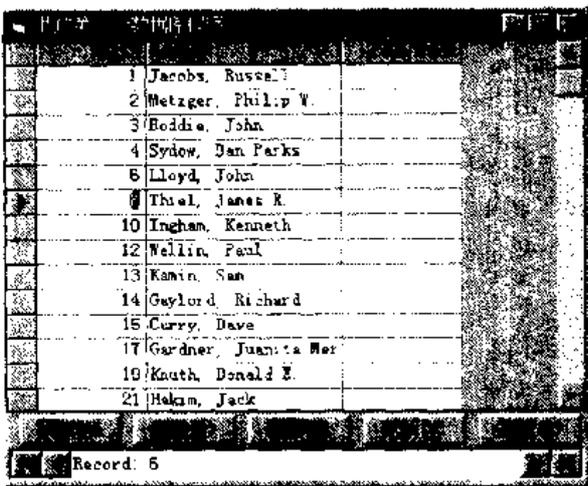


图 15.10 数据窗体向导生成程序的运行结果

```
End With
End Sub
```

(3) 刷新按钮

```
Private Sub cmd 刷新_Click()
    '只有多用户应用程序需要
    datPrimaryRS.Refresh
End Sub
```

(4) 更新按钮

```
Private Sub cmd 更新_Click()
    datPrimaryRS.UpdateRecord
    datPrimaryRS.Recordset.Bookmark = datPrimaryRS.Recordset.LastModified
End Sub
```

(5) 关闭按钮

```
Private Sub cmd 关闭_Click()
    Screen.MousePointer = vbDefault
    Unload Me
End Sub
```

其余的代码是作验证和错误处理用的。

可以用 DATA 控件访问数据库,在设置了 DATA 控件的 DatabaseName 和 RecordSource 属性后,可以修改记录。例如,修改记录可以用如下语句:

```
Data1.Recordset.Edit
Data1.Recordset("字段名") = xxxx
Data1.Recordset.Update
```

添加记录的语句如下:

```
Data1.Recordset.AddNew
```

下面是代码简析,主要是五个按钮的代码。

(1) 添加按钮

```
Private Sub cmd 添加_Click()
    datPrimaryRS.Recordset.MoveLast
    grdDataGrid.SetFocus
    SendKeys "{down}"
End Sub
```

(2) 删除按钮

```
Private Sub cmd 删除_Click()
    With datPrimaryRS.Recordset
        .Delete
        .MoveNext
        If .EOF Then .MoveLast
    End With
End Sub
```

```
Data1.Recordset("字段名") = xxxx
```

```
Data1.Recordset.Update
```

而特定记录的定位可用 Find 或 Seek 方法,这与本章稍后介绍的 DAO 编程非常类似,这里就不详细介绍了。

提示:在一般情况下,可用 DATA 控件和 DBGrid 控件配合,实现对数据库的访问和修改,这需要对 DATA 控件和 DBGrid 控件非常熟悉才行,如用 SQL 语言来填写 DATA 控件的 RecordSource 属性以寻找特定记录,用 DBGrid 控件的 Row 和 Col 来定位记录,用 DBGrid1.columns(i)来访问记录,DBGrid1.columns(i) = xxxx 来修改记录。

3. 其他数据库绑定控件

(1) DBCombo 控件

DBCombo 控件是带有下拉列表框的与数据相连的组合框,它能自动从与它相连的 Data 控件的字段中移居,也可以有选择地更新其他 Data 控件中相关表的字段。DBCombo 的文本框部分能用来编辑选定的字段。

DBCombo 控件和标准 ComboBox 控件不同。ComboBox 控件的列表用 AddItem 方法填充数据项,而 DBCombo 控件由和它相连的 Data 控件的 Recordset 对象中的字段中的数据自动填充数据项。标准 ComboBox 控件必须用 AddItem 方法手工移动。另外,DBCombo 控件有能力更新驻留在不同的 Data 控件中的相关的 Recordset 对象的字段。

(2) DBList 控件

DBList 控件是和数据相连的列表框,它会被与之相连的 Data 控件的字段所填充,并有选择地更新其他 Data 控件中相关表的字段。下面举一个 DBList 控件的示例,向读者讲述如何使用 DBList 控件。

在窗体上放上两个控件:DBList 和 Data,它们的属性设置如下:

Data 控件的 DatabaseName 属性设为: Biblio.mdb(应在 VB 目录下),RecordSource 属性设为 Authors。

DBList 控件的 Rowsource 属性设为 Data1,ListField 属性设为 Author。

程序运行结果如图 15.11 所示。

(3) MSFlexGrid 控件

MSFlexGrid 控件对表格数据进行显示和操作。在对包含字符串和图片的表格进行分类、合并以及格式化时,具有完全的灵活性。当绑定到 Data 控件上时,MSFlexGrid 所显示的是只读数据。

提示:可以将文本、图片,或者文本和图片,放在 MSFlexGrid 的任意单元中。Row 和 Col 属性指定了 MSFlexGrid 中的当前单元。程序员可以在代码中指定当前单元,也可以在运行时,使用鼠标或者方向键来对其进行修改。Text 属性引用当前单元的内容。

下面举两个示例程序,说明如何使用 MSFlexGrid 控件。

(1) 程序一

在该示例中,将 VB 图标库中的图标加载到 MSFlexGrid 控件的两个单元中。可以使用任意两个图标。MSFlexGrid 控件的 COLS 属性设为 3(列)。

```
Private Sub Form1_Click ()
```

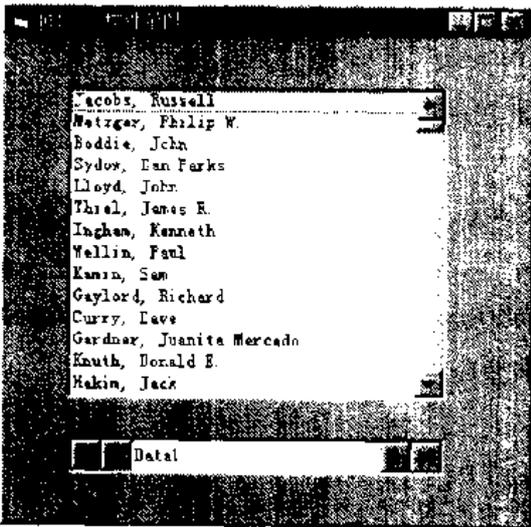


图 15.11 DBList 控件示例运行结果

```
Private Sub Form1_Load ()
    MSFlexGrid1.Rows = 8
    MSFlexGrid1.Cols = 5
End Sub
```

```
Private Sub MSFlexGrid1_Click ()
    '将文本放到当前单元中。
    MSFlexGrid1.Text = "Flex"
End Sub
```

```
Private Sub MSFlexGrid1_DblClick ()
    MSFlexGrid1.Clear
End Sub
```

程序运行结果如图 15.13 所示。

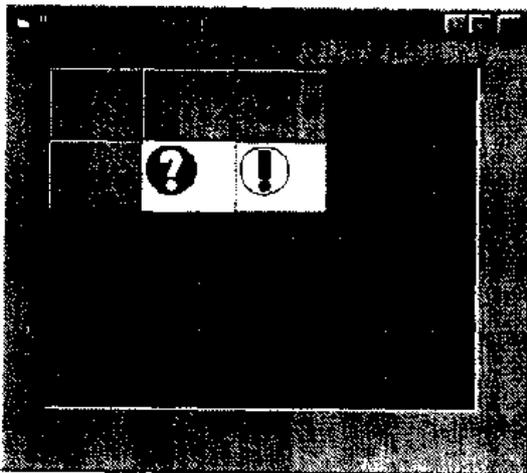


图 15.12 MSFlexGrid 控件示例程序 1 结果

```
'加载图标。
MSFlexGrid1.Row = 1
MSFlexGrid1.Col = 1
Set MSFlexGrid1.CellPicture = LoadPicture("Icons \
Computer \ Msgbox02.ico")
MSFlexGrid1.Row = 1
MSFlexGrid1.Col = 2
Set MSFlexGrid1.CellPicture = LoadPicture("Icons \
Computer \ Msgbox03.ico")
End Sub
```

程序运行结果如图 15.13 所示。

(2) 示例程序二

在该示例中,无论何时,当用户在单元上单击时,都将“Flex”放到当前单元中。无论何时,当用户双击时,都清除 MSFlexGrid 控件。

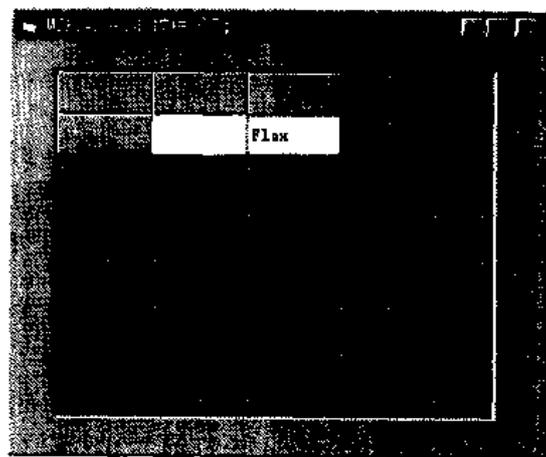


图 15.13 MSFlexGrid 控件示例程序 2 结果

15.3 DAO 编程

VB 提供了两种与 Jet 数据库引擎接口的方法:Data 控件和数据访问对象。上一节介绍了 Data 控件。Data 控件只给出有限的不需编程而能访问现存数据库的功能,而 DAO 模型则是全面控制数据库的完整编程接口。这两种方法并不是互斥的,实际上,它们常同时使用。

DAO 模型是设计关系数据库系统结构的对象类的集合。它们提供了完成管理这样一个系统所需的全部操作的属性和方法,包括创建数据库,定义表、字段和索引,建立表间的关系,定位和查询数据库等工具。

15.3.1 创建数据库

DAO 不是创建数据库的唯一方法。其他的途径可能有:

- VB 中的可视化数据管理器。使用可视化数据管理器,不需要编程就可创建 Jet 数据库。
- Microsoft Access。因为 Microsoft Access 使用了与 VB 相同的数据库引擎和格式,所以用 Microsoft Access 创建的数据库和直接在 VB 里创建的数据库是一样的。
- 外部数据库应用程序。像 FoxPro、dBASE 或 ODBC 客户/服务器应用程序这样的产品,可以创建新的外部数据库,VB 可通过 ISAM 或 ODBC 驱动程序来访问此数据库。

尽管有许多方法可供选择,但本节只把讨论焦点放在用 VB 中的自备工具来创建数据库上。DAO 提供了最大程度的数据库编程控制。如果要求所创建的应用程序能够在运行时自己创建数据库,DAO 也是可供选择的方法之一。

创建新的 Jet 数据库的过程是一个简单的过程,只需创建和定义与数据库设计的表、字段、索引和关系相一致的数据访问对象。从这个概述出发,将该过程按部就班地表现出来,创建新数据库的第一步是创建 Database 对象本身,然后添加与设计相一致的 TableDef 和 Field 对象来定义它的结构。

注意:下面的示例中常常使用一个名叫 Biblio.mdb 的数据库,这个数据库应和 VB 在同一目录中,是 VB 的示例数据库。

下面详细介绍与创建数据库相关的一些问题。

1. 创建数据库的步骤

(1) 使用 Dim 语句给数据库中的每一个对象创建对象变量。除了构成工作环境的 DBEngine 和缺省的 Workspace 对象外,还需要:

- 一个 Database 对象
- 每个表要一个 TableDef 对象
- 每个表中的每个字段要一个 Field 对象
- 每个表中的每个索引要一个 Index 对象

例如,可以用如下代码创建 Biblio.mdb 数据库的对象变量:

```
Dim MyDB As Database, MyWs As Workspace
Dim AuTfD As TableDef, TitfD As TableDef, PubTfD As TableDef
Dim AuFlds(2) As Field, TitFlds(5) As Field, PubFlds(10) As Field
```

```
Dim Auldx As Index, TitIdx(3) As Index, PubIdx As Index
```

(2) 使用 Workspace 对象的 CreateDatabase 方法创建新的数据库。在本例中,该方法使用了两个参量:一个用以指定数据库的名称,一个用以指定区域:

```
Set MyWs = DBEngine.Workspaces(0)
Set MyDb = MyWs.CreateDatabase("C:\VB\Biblio.mdb", dbLangGeneral, dbVersion03)
```

(3) 使用 Database 对象的 CreateTableDef 方法,为数据库中的所有表创建新的 TableDef 对象,如下所示:

```
Set TitTd = MyDB.CreateTableDef("Titles")
Set AuTd = MyDB.CreateTableDef("Authors")
Set PubTd = MyDB.CreateTableDef("Publishers")
```

(4) 使用 TableDef 对象的 CreateField 方法,为表中每个字段创建 Field 对象,然后设置每个字段的属性来定义字段长度、数据类型和其他所需的属性。例如,下面的代码用以创建 Biblio.mdb 数据库中的 Authors 表:

```
Set AuFlds(0) = AuTd.CreateField("Au_ID", dbLong)
'使其成为计数字段:
AuFlds(0).Attributes = dbAutoIncrField
Set AuFlds(1) = MyTd.CreateField("Author", dbText)
AuFlds(1).Size = 50
```

(5) 用 Append 方法把每个字段添加到表中,并把每个表添加到数据库中,如下所示:

```
AuTd.Fields.Append AuFlds(0)
AuTd.Fields.Append AuFlds(1)
MyDB.TableDefs.Append AuTd
```

2. 添加关系和索引

用 CreateIndex 方法可以给 TableDef 添加索引。为了指定被索引的字段,需要用 Index 对象的 CreateField 方法创建新的 Field 对象。

给数据库表添加索引的步骤

(1) 用 TableDef 对象中的 CreateIndex 方法给每个表创建索引,并设置其属性:

```
Set Auldx = MyTd.CreateIndex("AuthorID")
Auldx.Primary = True
Auldx.Unique = True
```

(2) 用 Index 对象的 CreateField 方法给每个索引对象创建字段:

```
Set NewFld = Auldx.CreateField("Au_ID")
```

(3) 把字段追加到 Index 中,然后把 Index 追加到 TableDef 对象中:

```
Auldx.Fields.Append NewFld
MyTd.Indexes.Append Auldx
```

注意:用 Index 对象中的 CreateField 方法创建的字段不添加到 TableDef 中。相反,它们被添加到 Index 对象中,并赋予 Name 属性,该属性与被索引的 TableDef 字段中的 Name 属性相

同。不指定 Index 对象中字段的 Type 和 Size 属性。

3. 关系和引用完整性

像上面描述过的那样,创建实现主键和外部键的 TableDefs,允许按照公共主键/外部键值使一个表中的记录同另一个表中的相应记录建立关系。然而,要使关系成为有用的关系,在添加或删除记录时,保持引用完整性是至关重要的。引用完整性意味着在任一引用表中的外部键必须时刻指向被引用表中的有效记录。

例如,在 Biblio.mdb 数据库中,Titles 表中的 PubID 字段是外部键,它引用 Publishers 表的 PubID 键。例如,如果 Titles 表中有一个记录,其 PubID 项为 25,则 Publishers 表中必须有一个 PubID 项为 25 的记录存在。如果不存在这样的记录,则没有办法确定哪一个出版商出版过这本书,这样该数据库实际上已经受损了。这类损坏被称为引用完整性侵犯。Jet 数据库引擎提供 Relation 对象来强制引用完整性,并防止这样的侵犯。

下面是给数据库添加关系的步骤

(1) 用 Database 对象的 CreateRelation 方法创建 Relation 对象,然后设置其 Table 和 ForeignTable 属性:

```
Dim Au_Tit As Relation
Set Au_Tit = MyDb.CreateRelation("Authors_Titles")
Au_Tit.Table = "Authors"
Au_Tit.ForeignTable = "Titles"
```

(2) 用 Relation 对象中的 CreateField 方法,创建一个字段,用来定义关系中的公共主键/外部键字段。

```
Dim TempField As Field
Set TempField = Au_Tit.CreateField("Au_ID")
TempField.ForeignName = "Au_ID"
```

(3) 用 Append 方法把 Field 对象添加到 Relation 中,然后把 Relation 添加到 Database 中。

```
Au_Tit.Fields.Append TempField
MyDb.Relations.Append Au_Tit
```

4. 设置附加属性

使用某个 Create 方法时,如果省略了一个或多个选项的参数,在把新对象追加到集合中前,可以用赋值语句设置或重设相应的属性。例如:

```
Set MyFld = MyTableDef.CreateField("ClientName", dbText, 30)
```

等价于

```
Set MyFld = MyTableDef.CreateField()
MyFld.Name = "ClientName"
MyFld.Type = dbText
MyFld.Size = 30
```

15.3.2 修改数据库

用户可以用 VB 和数据访问对象在程序中改变数据库结构。修改数据库和创建数据库的

过程十分相似。大多数情况下,使用同样的 Create 和 Append 方法来添加对象。可以向数据库中添加新的 TableDef 对象,或向已经存在的表中添加新的 Field 和 Index 对象。还可以从数据库中删除 TableDef,或从 TableDef 中删除 Index。删除 Field 对象时,要遵循某些约定,详细信息,请参阅后面的“更改或删除字段”。还可以使用 SQL 语句来修改数据库,关于 SQL 语句的详细信息,请参阅本章稍后的 SQL 简介。

下面详细介绍与修改数据库有关的一些问题。

1. 向数据库中添加表

只要简单地向已经存在的 TableDefs 集合中添加新的 TableDef 对象,就可以向数据库中添加表。下面是向数据库中添加表的代码:

```
Dim db As Database
Dim NewTd As TableDef      '创建新的 TableDef 对象。
Dim NewFld As Field        '创建新的 Field 对象。

Set db = DBEngine.Workspaces(0).OpenDatabase("Test.mdb")

Set NewTd = db.CreateTableDef("New Table Name")
Set NewFld = NewTd.CreateField("New Field Name", dbInteger)

NewTd.Fields.Append NewFld      '向 TableDef 中添加字段。
db.TableDefs.Append NewTd      '把 TableDef 添加到数据库中。
db.Close
```

2. 删除表

使用 TableDefs 集合的 Delete 方法来删除表。表删除之后,在该表中的所有字段、索引和数据都要被删除,所以应该小心地使用这个方法。下面是删除 Authors 表的代码:

```
Db.TableDefs.Delete "Authors"
```

3. 向表中添加字段

在已经存在的 Fields 集合中追加字段就等于在已经存在的表中添加字段。下面的代码在 Authors 表中添加了两个新的字段:Address 和 Phone。

```
Dim Db As Database
Dim Td As TableDef
Dim Fld As Field

Set Db = DBEngine.Workspaces(0).OpenDatabase("C:\Test.mdb")
Set Td = Db.TableDefs("Authors")
'创建第一个新字段。
Set Fld = Td.CreateField("Address", dbText, 30)
'把新的 Field 对象追加到 Fields 集合中。
Td.Fields.Append Fld
'复用 Fld 变量来创建第二个 Field 对象。
Set Fld = Td.CreateField("Phone", dbText, 25)
'追加到 Fields 集合中。
```

```
Td.Fields.Append Fld
```

```
Db.Close
```

4. 更改或删除字段

一旦一个字段被追加到 TableDef 中就不能再更改它。只有在字段不是任何 Index 或 Relation 对象的一部分时才能删除它。使用 TableDef 对象的 Delete 方法来删除单独的字段。如果要更改单独的字段,必须添加一个新的 TableDef,来反映在字段上所希望的更改,然后把旧表中的数据移到新表中并删除旧表。

注意:必须首先删除索引才能删除索引字段。在能够删除为某个关系的一部分的 Field 或 TableDef 对象前,必须把任何受到影响的 Relation 对象也删除掉。

5. 向表中添加索引

使用 Append 方法来向已经存在的 Indexes 集合中添加新的索引。下面的例子向 Authors 表中添加了一个新的索引:

```
Dim Db As Database, Td As TableDef, NewIdx As Index, NewFld As Field
```

```
Set Db = DBEngine.Workspaces(0).OpenDatabase("C:\Biblio.mdb")
```

```
Set Td = Db.TableDefs("Authors")
```

```
Set NewIdx = Td.CreateIndex("Address_Index")
```

```
NewIdx.Unique = False
```

```
Set NewFld = NewIdx.CreateField("Address")
```

```
NewIdx.Fields.Append NewFld '向索引中添加字段
```

```
Td.Indexes.Append NewIdx '向 TableDef 中添加索引
```

```
Db.Close
```

6. 删除索引

删除索引的语法和删除表的语法相似。下面示例中的代码从 Authors 表的索引集合中删除了 "Address_Index" 索引:

```
Db.TableDefs("Authors").Indexes.Delete "Address_Index"
```

15.3.3 使用记录和字段

Microsoft Jet 数据库引擎支持丰富的数据访问对象(DAO),可用它对数据进行组织、排序、查找、更新、添加和删除操作。Recordset 对象提供了 24 种方法和 26 种属性,利用它们可以对数据库中的记录进行各种处理。用 Recordset 对象的 Fields 集合以及 Field 对象的各种属性与方法,可在字段一级上对数据进行操作。本节将讲述如何使用 DAO 的 Recordset 和 Field 对象对记录和字段进行处理。

1. 创建 Recordset 对象

Recordset 对象可以表示基本表中的记录或者作为查询结果的记录。使用 Recordset 对象可以在记录一级上对数据库中的数据进行处理。

Recordset 对象有五种类型:表、动态集、快照、动态和仅向前。它们之间存在明显的区别。

(1) 表类型的 Recordset 对象是指当前数据库中的本地表,或者 Microsoft Jet 创建的外部数

数据库。在创建表类型的记录集时,数据库引擎打开实际表,后续的数据操作都是直接对基本表进行的。只能对单个的表打开表类型的记录集,而不能对联接或者联合查询打开。

如果使用基本表创建的索引,就可以对表类型的 Recordset 对象进行索引。与其他类型的 Recordset 对象相比,表类型的搜索与排序速度最快。

提示: 定位特定的记录时,Seek 方法要快于 Find 方法。

(2) 动态集类型的 Recordset 对象可以是本地的或者链接的表,也可以是返回的行查询结果。它实际上是对一个或者几个表中的记录的一系列引用。可用动态集从多个表中提取和更新数据,其中包括链接其他数据库中的表。动态集类型具有一种与众不同的特点:不同数据库的可更新联接。利用这种特性,可以对不同类型的数据库中的表进行可更新的联接查询。

动态集和它的基本表可以互相更新。如果动态集中的记录发生改变,同样的变化也将在基本表中反映出来。在打开动态集的时候,如果其他的用户修改了基本表,那么动态集中也将反映出被修改过的记录。动态集类型是最灵活的 Recordset 类型,也是功能最强的。不过,它的搜索速度与其他操作的速度不及表类型的 Recordset。

(3) 快照类型的 Recordset 对象包含的数据是固定的,它反映了在产生快照的一瞬间数据库的状态。从 Microsoft Jet 数据源得到的快照是不可更新的,从开放数据库互连 (ODBC) 数据源得到的某些快照是可以更新的,这取决于后端数据库的能力。

与动态集类型和表类型的 Recordset 对象相比,快照的处理开销较少。因此,它执行查询和返回数据的速度更快,特别是在使用 ODBC 数据源时。需要注意的是,对于 .mdb 文件,在快照中用指针表示 Memo 和 Long Binary 字段中的数据。

(4) 仅向前类型的 Recordset 对象,有时被称为“向前滚动快照”或者“仅向前快照”,提供了快照的一部分功能。它提供了最基本的 Recordset 对象功能,但是通常可以达到最快的速度。与快照类似,从 Microsoft Jet 得到的仅向前类型的 Recordset 对象是不可更新的。另外,仅向前的快照只允许在记录中向前移动,而不能向相反的方向移动。这种类型的 Recordset 对象不能被复制,而且只支持 Move 和 MoveNext 方法。

(5) 动态类型的 Recordset 对象是从一个或几个基本表中查询到的结果集,对于返回行的查询,可以在其中添加、修改或删除记录。另外,其他用户对基本表的添加、删除和修改操作也将出现在您的记录集中。这种类型对应于 ODBC 的动态游标。

提示: 快照类型保存了表中所有记录的完整复本,因此,如果记录的个数很多,快照的性能将比动态集慢得多。为了确定快照与动态集哪一个更快,可以先以动态集方式打开记录集,然后再以快照方式打开它。

具体使用什么记录集,取决于需要完成的任务:是要更改数据呢,还是简单地查看数据。例如,如果必须对数据进行排序或者使用索引,可以使用表。因为表类型的 Recordset 对象是做了索引的,它定位数据的速度是最快的。如果希望能够对查询选定的一系列记录进行更新,可以使用动态集。如果在特殊的情况下不能使用表类型的记录集,而且只需对记录进行扫描,那么使用仅向前的快照类型可能会快一些。

可以使用 OpenRecordset 方法创建 Recordset 对象变量。首先,必须声明一个 Recordset 类型的变量,然后将变量设置为 OpenRecordset 方法返回的对象。

OpenRecordset 方法可以在 Database、Connection、TableDef、QueryDef 以及已经存在的 Recordset 对象中使用。Connection 和 Database 对象的 OpenRecordset 方法的语法如下:

```
Set variable = database.OpenRecordset (source [, type [, options, [lockedits ]]])
```

其他所有类型的对象的 OpenRecordset 方法的语法如下:

```
Set variable = object.OpenRecordset ([type [, options [, lockedits ]]])
```

其中:variable 参数是新的 Recordset 对象的名称。

database 参数是一个打开的 Database 或者 Connection 对象的名称,可用该对象创建新的 Recordset 对象。

object 参数是一个 TableDef、QueryDef 或者现有的 Recordset 对象,我们用该对象创建新的 Recordset 对象。

source 参数指定新的 Recordset 对象的记录来源。source 的值将成为新产生的 Recordset 对象的 Name 属性的值。用 Connection 或者 Database 对象创建新的 Recordset 对象时,source 参数可以是数据库中现有的 TableDef 或 QueryDef 对象,或者为一个有效的、返回行的 SQL 查询或语句。如果用 TableDef、QueryDef 或者现有的 Recordset 对象创建新的 Recordset 对象,那么对象自身将为新的记录集提供数据源。

type 参数是一个内部常数,它指定要创建的记录集的类型

2. 在记录中移动

在引用记录集中的字段时,获取到的值来自当前位置的记录,即:当前记录。然而,在记录集中,当前位置也可能位于第一个记录之前,或紧接在最后一个记录之后。在特定环境中,当前位置是未定义的(即没有当前记录)。

可以使用 Move 方法遍历整个记录集中的记录:

- 使用 MoveFirst 方法移至第一个记录。
- 使用 MoveLast 方法移至最后一个记录。
- 使用 MoveNext 方法移至下一个记录。
- 使用 MovePrevious 方法移至上一个记录。
- 使用 Move [n] 方法向前或向后移 n 个记录,n 为指定的数值。

(1) 对记录集边界的检测

在记录集中,如果向一个方向移动得太远,将产生运行时错误。例如,在移出记录集末端之后,如果还试图使用 MoveNext 方法,将产生一个可捕获错误。因此,必须知道 Recordset 对象的边界。

BOF 属性指示当前位置是否在记录集的开头。若 BOF 为 True,则当前位置位于记录集的第一个记录之前。在打开记录集时,若其中没有记录,则 BOF 属性为 True。与此类似,若当前位置位于记录集最后一个记录之后,或记录集中无记录,EOF 属性将为 True。

下列程序段用 BOF 和 EOF 属性检测记录集的首尾。该代码段为 Orders 表创建一个表类型的记录集,并用两种顺序遍历其中的记录:先从头至尾,再从尾至头。

```
Dim dbs As Database, rstOrders As Recordset
Set dbs = OpenDatabase("Northwind.mdb")
Set rstOrders = dbs.OpenRecordset ("Orders", dbOpenTable)
Do Until rstOrders.EOF
    ' ... 处理数据 ...
    rstOrders.MoveNext ' 移到下一个记录。
```

```

Loop
rstOrders.MoveLast ' 移到最后一个记录。
' 重复操作,一直移动到文件的开头。
Do Until rstOrders.BOF
    ' ...处理数据...
    ' 移到上一个记录。
    rstOrders.MovePrevious
Loop
rstOrders.Close ' 关闭记录集。
dbs.Close

```

(2) 计算记录集的个数

在有些情况下,需要知道 Recordset 对象中包含的记录数目。例如,有时需要显示数据库的各个表分别包含的记录个数;有时需要根据记录的数目修改报表的外观。

对于表类型的记录集,RecordCount 属性为它的记录总数;对于快照或动态集型的 Recordset 对象,该属性为访问过的所有记录的个数。如果记录集中没有记录,那么它的 RecordCount 值为 0。

注意: RecordCount 的值等于实际被访问的记录个数。例如,在首次创建动态集或快照类型的记录集时,只有一个记录被访问。如果在创建了动态集或快照(假设其中至少有一个记录)后立即查看 RecordCount 属性,你将发现它的值为 1。为了访问所有的记录,可在打开记录集后立即使用 MoveLast 方法。如果需要回到第一个记录,可以使用 MoveFirst 方法。动态集或快照不会自动访问所有的记录,因为这种操作的速度较慢,对于较大的记录集尤其如此。

在打开表类型的 Recordset 对象时,可以高效地访问基本表中的所有记录,而且在打开记录集之后 RecordCount 就能够列出表中的记录数。在某些多用户条件下,取消的事务可能会使 RecordCount 的值成为无效的。对数据库进行压缩即可使表的记录计数重新恢复正常。

下列代码段用 SQL SELECT 语句创建了一个快照类型的记录集,并返回记录集中记录的个数。

```

Function RecCount(strSQL As String, strDB As String) As Long
Dim rstCount As Recordset
Dim dbs As Database
On Error GoTo ErrorHandler
Set dbs = OpenDatabase(strDB)
Set rstCount = dbs.OpenRecordset(strSQL, dbOpenSnapshot)
' 若无记录,返回 0 并退出。
If rstCount.EOF Then
    rstCount.Close
    RecCount = 0
    Exit Function
Else
    rstCount.MoveLast
    RecCount = rstCount.RecordCount
    rstCount.Close

```

```

        Exit Function
    End If
ErrorHandler:
Select Case Err
    Case 0
        Exit Function
    Case Else
        MsgBox "Error " & Err & ": " & Error, vbOKOnly, "ERROR"
        Exit Function
End Select
End Function

```

(3) 当前记录在记录集中的位置

有的时候需要确定当前记录在记录集中的位置,并将当前记录位置指示给用户。例如,可以用拨号盘或仪表盘等类型的控件显示当前的记录位置。有两种属性可以指示当前的记录位置:AbsolutePosition 属性和 PercentPosition 属性。

AbsolutePosition 属性的值为当前记录相对于 0 的位置。然而,不要误以为它就是记录号;在当前记录处于不定状态时,AbsolutePosition 的值为 -1。另外,在访问记录集时,不能保证记录每次都以同样的顺序出现。

PercentPosition 属性表示当前位置占 RecordCount 的百分比。但是,在记录集完全完成移居之前,RecordCount 属性并不反映记录集中的记录总数,所以 PercentPosition 属性只反映了当前记录位置与打开记录集后被访问的记录的记录的比例。要使 PercentPosition 属性反映当前记录位置与整个记录集的比例,应在打开记录集后立即使用 MoveLast 和 MoveFirst 方法,使记录集完全完成移居。如果结果集很大,对非表类型的 Recordset 对象使用 MoveLast 方法可能要花很长时间。

注意: PercentPosition 属性只是近似值,不要把它当作精确的参数使用。可以用它控制某些进度指示器,例如,在遍历记录集的时候,可以将它用于报告工作进度“已完成百分之几”的控件。

下列代码段遍历了 Employees 表,如果职员的雇佣时间早于 1993 年 1 月 1 日,则在 Notes 字段中添加文本“Senior Staff”。SysCmd 函数的作用是:显示进度栏,并在其中显示表中已被处理的记录占的百分比。

```

Function PercentPos()
    Dim dbs As Database, strMsg As String
    Dim rstEmployees As Recordset, intRet As Integer
    Dim strQuery As String
    Dim varReturn As Variant
    On Error GoTo ErrorHandler
    strQuery = "SELECT * FROM Employees;"
    Set dbs = OpenDatabase("Northwind.mdb")
    Set rstEmployees = dbs.OpenRecordset (strQuery, dbOpenDynaset)
    '若无职员记录,退出。

```

```

If rstEmployees.EOF Then Exit Function
With rstEmployees
    strMsg = "Processing Employees table..."
    intRet = SysCmd(acSysCmdInitMeter, strMsg, 100)
Do Until .EOF
    If ! HireDate < #1/1/93# Then
        .Edit
        ! Notes = ! Notes & ";" & "Senior Staff"
        .Update
    End If
    If .PercentPosition < > 0 Then
        intRet = SysCmd(acSysCmdUpdateMeter, .PercentPosition)
    End If
    .MoveNext
Loop
.Close
End With
intRet = SysCmd(acSysCmdRemoveMeter)
dbz.Close
ErrorHandler:
Select Case Err
    Case 0
        Exit Function
    Case Else
        MsgBox "Error " & Err & "; " & Error, vbOKOnly, "ERROR"
        ' 将进度显示清 0。
        varReturn = SysCmd(acSysCmdSetStatus, "")
    Case Else
        dbz.Close
End Select
Exit Function
End Select
End Function

```

提示：若应用程序删除了动态集类型的记录集中的数据，那么 RecordCount 属性的值将减小。然而，在多用户环境中，如果其他用户删除了某些记录，RecordCount 的值并不立即发生改变，直到当前记录移动到一条被删除的记录时，RecordCount 属性才会减一。要将 RecordCount 属性设为记录集中当前所有记录的总数，可以先对记录集使用 Requery 方法，然后再用 MoveLast 方法。

3. 查找特定的记录

在表类型的记录集中，可以使用 Seek 方法定位记录。在动态集或快照类型的记录集中，可用 Find 方法。

在使用 Seek 方法定位记录时，Microsoft Jet 将使用 Index 属性定义的当前索引。

Seek 方法的语法为：

```
table.Seek comparison, key1, key2 ...
```

table 参数是一个表类型的 Recordset 对象。Seek 允许接受多个参数,第一个是 comparison,该字符串确定比较的类型。下表列出了 Seek 方法中可用的比较字符串。

比较字符串	描述
" = "	等于指定的键值
" > = "	大于或等于指定的键值
" > "	大于指定的键值
" < = "	小于或等于指定的键值
" < "	小于指定的键值

keyn 参数可以是一个或者多个值,分别对应于记录集的当前索引中的字段。Microsoft Jet 用这些值与 Recordset 对象的记录进行比较。

下列代码段打开了 Suppliers 的表类型记录集,并使用 Seek 方法查找某个供应商的 ID。如果找不到,返回 Null。

```
Public Function GetSupplierID(strCompanyName As String) As Variant
    Dim dbs As Database, rstSuppliers As Recordset
    On Error GoTo ErrorHandler
    Set dbs = OpenDatabase("Northwind.mdb")
    Set rstSuppliers = dbs.OpenRecordset("Suppliers", dbOpenTable)
    rstSuppliers.Index = "CompanyName"
    rstSuppliers.Seek "=", strCompanyName
    If rstSuppliers.NoMatch Then
        GetSupplierID = Null
    Else
        GetSupplierID = rstSuppliers! SupplierID
    End If
    rstSuppliers.Close
    dbs.Close
ErrorHandler:
Select Case Err
    Case 0
        Exit Function
    Case Else
        MsgBox "Error " & Err & ": " & Error, vbOKOnly, "ERROR"
        Exit Function
End Select
End Function
```

Seek 方法总是从记录集的头部开始查找记录。若在同一记录集中多次使用同样的 Seek 方法(参数相同),那么找到的将是同一个记录。

提示:要判断是否找到了满足条件的记录,可以检查 Recordset 对象的 NoMatch 属性。如果找到了满足条件的记录, NoMatch 将设置为 False;否则将设置为 True。

在动态集类型和快照类型的记录集中,可以使用下列方法定位记录。DAO 总共提供了四种 Find 方法。

- FindFirst 方法找到满足条件的第一个记录。
- FindLast 方法找到满足条件的最后一个记录。
- FindNext 方法找到满足条件的下一个记录。
- FindPrevious 方法找到满足条件的上一个记录。

提示：在使用 Find 方法时，需要指定搜索条件，这通常是使字段名等于特定值的表达式。

4. 修改现有记录

使用 Edit 和 Update 方法，可以修改表类型或者动态集类型的 Recordset 对象。

下面是修改表类型或者动态集类型的记录集对象的步骤：

- (1) 移动到需要修改的记录。
- (2) 使用 Edit 方法，准备修改当前记录。
- (3) 对记录进行修改。
- (4) 使用 Update 方法，保存对当前记录的修改。

例如：下列代码改变了名叫“Employees”表中的所有销售代表的职务头衔。其中，职务头衔是表“Employees”中的一个字段：

```
Sub ChangeTitle()
Dim dbs As Database, rstEmployees As Recordset
Set dbs = OpenDatabase("Northwind.mdb")
Set rstEmployees = dbs.OpenRecordset("Employees")
rstEmployees.MoveFirst
Do Until rstEmployees.EOF
    If rstEmployees! Title = "Sales Representative" Then
        rstEmployees.Edit
        rstEmployees.Update
    End If
    rstEmployees.MoveNext
Loop
rstEmployees.Close
dbs.Close
End Sub
```

注意：在对当前记录进行修改之前，如果未使用 Edit 方法，将会发生运行时错误。在编辑了当前记录之后，如果立即移动到另一个记录，或者关闭了记录集，但没有使用 Update 方法，那么所做的修改将丢失，预先没有任何的警告。例如，在前例中如果没有使用 Update 方法，那么对 Employees 表将不会进行任何实际的修改。

提示：可以使用 CancelUpdate 方法中止 Edit 方法和其他挂起的事务，从而取消修改。尽管也可以用离开当前记录的方法中止 Edit 方法，但是如果当前记录是记录集中的第一个或者最后一个，或者是新加的记录，那么这种方法就不可行了。使用 CancelUpdate 方法是最简单的了。

5. 删除现有记录

使用 Delete 方法，可以删除表类型或者动态集类型的记录集中的记录。快照类型的记录

集中的记录是不可删除的。

下列函数删除了 Shippers 表中的重复记录。重复值是以发货人的名字为根据判断出来的。

注意：程序中的 SQL 语言将在稍后介绍

```
Function DeleteDuplicateShippers() As Integer
    Dim rstShippers As Recordset
    Dim strQuery As String
    Dim dbs As Database
    Dim strName As String
    On Error GoTo ErrorHandler
    strQuery = "SELECT * FROM Shippers" & "ORDER BY CompanyName;"
    Set dbs = OpenDatabase("Northwind.mdb")
    Set rstShippers = dbs.OpenRecordset (strQuery, dbOpenDynaset)
    ' 如果 Shippers 表中没有记录,退出。
    If rstShippers.EOF Then Exit Function
    strName = rstShippers! [CompanyName]
    rstShippers.MoveNext
    Do Until rstShippers.EOF
        If rstShippers! [CompanyName] = strName Then
            rstShippers.Delete
        Else
            strName = rstShippers! [CompanyName]
        End If
        rstShippers.MoveNext
    Loop
ErrorHandler:
Select Case Err
    Case 0
        DeleteDuplicateShippers = SUCCESS
        Exit Function
    Case Else
        MsgBox "Error " & Err & ": " & Error, vbOKOnly, "ERROR"
        DeleteDuplicateShippers = FAILED
        Exit Function
End Select
End Function
```

注意：在使用 Delete 方法时,Microsoft Jet 将立即删除当前记录,不加任何的警告或者提示。删除记录操作并不自动将下一条记录设置为当前记录;要移动到下一条记录,必须使用 MoveNext 方法。然而,必须牢记这一点:在离开被删除的记录之后,不能再移动回来。

6. 添加记录

使用 AddNew 和 Update 方法,可以在表类型或者动态集类型的记录集中添加记录。

下面是在表类型或者动态集类型的记录集中添加记录的步骤：

- (1) 使用 AddNew 方法,准备编辑记录。
- (2) 为记录的每个字段赋值。
- (3) 使用 Update 方法保存新的记录。

下列代码在 Shippers 中添加了一个新的记录：

```
Dim dbs As Database, rstShippers As Recordset
Set dbs = OpenDatabase("Northwind.mdb")
Set rstShippers = dbs.OpenRecordset("Shippers")
rstShippers.AddNew
rstShippers! CompanyName = "Global Parcel Service"
' ...设置其他字段...
rstShippers.Update
rstShippers.Close
dbs.Close
```

在使用 AddNew 方法的时候,Microsoft Jet 准备了一个新的、空白的记录,并使之成为当前记录。当使用 Update 方法保存新的记录后,使用 AddNew 方法之前的当前记录又重新成为当前记录。

15.4 SQL 简介

本节简单介绍了 SQL,作为一种广泛使用的标准数据库编程语言,它在 VB 的数据库编程中也是必不可少的。

15.4.1 SQL 简介

SQL 是一种数据库编程语言,它的起源与关系数据库紧密相联。关系数据库是 E. F. Codd 在 20 世纪 70 年代早期发明的一种数据库。现在的 SQL 的前身是 Sequel 语言,因此,SQL 的发音通常是“sequel”,而不是“ess cue ell”,虽然这两种发音都是可接受的。现代的 SQL 已经发展成为关系数据库所广泛使用的标准,并且被 ANSI 标准所定义。包括 Jet 数据库引擎所支持的版本在内的大多数 SQL 的实现,和定义的标准之间只有很小的变动。这些差异在本章的后面进行了论述,但是,这种语言的全部结构和功能不会因供应商的不同而有任何差异。如果已用过 SQL 的任何一种应用,那么再来学习 Microsoft Jet 版本将不会有任何困难。

15.4.2 SQL 和 定位的比较

Microsoft Jet 数据库引擎提供了两种独立的方法来实现大多数数据库任务：

- 以在数据库记录中到处移动为基础的定位模型。
- 以结构化查询语言 (SQL) 为基础的关系模型。

定位模型由“创建和修改数据库”,以及“使用记录和字段”中所描述的属性和方法组成。在大多数情况下,与之等价的 SQL 方法的效率更高。所以,在需要强调性能时,通常应当使用 SQL 方法。此外,SQL 还有这样一个优点:它是行业标准的数据库接口,所以,一旦了解了 SQL 命令,就可以访问和操作不同厂商的大量数据库产品。

实际上常常发现,这两种模式同时使用。例如,可能先使用一条 SQL SELECT 语句,从一个很大的表中选定某些项,从而创建一个较小的记录集,然后使用 Move 定位方法来逐个检查记录集中的每个记录。

SQL 属性包含了结构化的查询语言语句,它们决定了在执行查询时如何对行进行选择、分组和排序。使用查询可以选择数据行,也可以定义动作查询修改数据而不返回数据行。

查询的 SQL 语法必须遵循数据源查询处理程序定义的 SQL 方言。当需要从一个查询返回数据行的时候,通常要在 SQL 属性中提供一个 SELECT 语句。SELECT 语句中指定:

- 要返回的每一列的名称,或者用“*”表示返回的表的所有列。可能引起混淆的列名必须包括对应表的名称。
- 查询需要用到的表的名称。如果要指定多个表,则必须提供一个 WHERE 子句,表明哪些列用于交叉引用表中的信息。通常这些数据列具有相同的名称和意义。
- 任选地,WHERE 子句说明如何连接两个指定的表,如何限制和筛选返回列的数目和类型。可以在 WHERE 子句中使用用户提供的参数,从而在各次查询中检索不同的信息。如果需要在运行时提供 WHERE 子句的条件,那么必须创建一个参数查询。
- 其他的任选子句,例如 ORDER BY 子句用于将数据列按照指定方式排序, GROUP BY 子句用于将数据列分组到相关的集合中。

每种 SQL 方言支持不同的语法和辅助子句。

15.4.3 SQL 部件

下面详细讲述 SQL 部件。

1. SQL 命令

和 DAO 定位方法一样,SQL 不但提供了数据定义语言 (DDL) 命令,还提供了数据操作语言 (DML) 命令。DDL 命令允许创建和定义新的数据库、字段和索引,而 DML 命令则允许创建查询来从数据库中排序、筛选和抽取数据,但这两者之间有重复的地方。

下面介绍 DDL 和 DML

(1) DDL

SQL 中的 DDL 语句是由下列命令组成的表达式。

下面是命令和对应的描述

CREATE	该命令用来创建新的表、字段和索引
DROP	该命令用来删除数据库中的表和索引
ALTER	该命令通过添加字段或改变字段定义来修改表

(2) DML

DML 语句是由下列命令组成的表达式。

下面是命令和对应的描述

SELECT	该命令用来在数据库中查找满足特定条件的记录
INSERT	该命令用来在数据库中用单一的操作加载一批数据
UPDATE	该命令用来改变特定记录和字段的值
DELETE	该命令用来从数据库表中删除记录

2. SQL 子句

子句是用来修改条件的,这些条件被用来定义要选定或要操作的数据。下表列出了可用

的子句。

下面是子句和对应的描述

FROM	用来为从其中选定记录的表命名
WHERE	用来指定所选记录必须满足的条件
GROUP BY	用来把选定的记录分成特定的组
HAVING	用来说明每个组需要满足的条件
ORDER BY	用来按特定的次序将记录排序

3. SQL 运算符

在 SQL 中有两类运算符:逻辑运算符和比较运算符。

(1) 逻辑运算符

逻辑运算符用来连接两个表达式,它通常位于 WHERE 子句中。例如:

```
SELECT * from Mytable WHERE condition1 AND condition2
```

在这里,AND 运算符把表达式 condition1 和 condition2 连接起来,表示这两个条件必须同时符合才满足选择条件。逻辑运算符包括:

■ AND

■ OR

■ NOT

(2) 比较运算符

比较运算符用来比较两个表达式的关系值,从而决定应当采取哪个行动。例如:

```
SELECT * from Publishers WHERE PubId = 5
```

在这里,'=' 运算符指出:只有 PubID 字段的值为 5 的记录才会被选中。

下表列出了比较运算符和它们的含义。

运算符	含义/用法
<	小于
< =	小于等于
>	大于
> =	大于等于
=	等于
< >	不等于
BETWEEN	用来指定值的范围
LIKE	在模式匹配中使用
IN	用来指定数据库中的记录

4. SQL 合计函数

在 SELECT 子句内使用合计函数对记录组进行操作,它返回应用于一组记录的单一值。例如,AVG 合计函数可以返回记录集的特定字段中所有值的平均数。下表列出了合计函数和它们对应的描述

AVG	用来获得特定字段中的值的平均数
COUNT	用来返回选定记录的个数
SUM	用来返回特定字段中所有值的总和

MAX 用来返回指定字段中的最大值
MIN 用来返回指定字段中的最小值

5. SQL 数据定义语言

SQL 数据定义语言(DDL)包含了一些命令,可以用它们来创建表和索引,以及在表中添加或删除列或索引。这些数据定义语句只能用于 Jet 数据库;而不被任何其他外部数据库格式所支持。

注意: 要使用 DDL 命令,或任何不返回记录行的查询,则必须用双引号括住整个语句,并且将其用作 Database 或 QueryDef 对象的 Execute 方法的一个参数。例如:

```
MyDB.Execute "CREATE TABLE Employees ([First Name] TEXT, [Last Name] TEXT)"
```

要使用任何返回记录行的命令(如 SELECT),必须将该语句作为 OpenRecordset 方法的源参数,例如:

```
MyDB.OpenRecordset("SELECT * FROM Titles WHERE Au_ID = 5",dbOpenDynaset)
```

6. SQL 数据操作语言

SQL 数据操作语言(DML)语句用于在表中检索记录、更新记录以及添加或删除记录。有一些其他的语句也可以用来完成这些任务,但是它们大部分可以纳入通用结构的 SELECT 查询。

(1) 选择查询

使用 SELECT 语句从数据库中检索记录形成记录集合,并将它们存入新的 Recordset 对象中。当然,应用程序可以操作该记录集——根据需要显示、添加、更改或者删除记录。应用程序也可以用这些数据 display 和生成报表。

SELECT 通常位于一个 SQL 语句的开始。大部分的 SQL 语句或者是 SELECT 语句或者是 SELECT...INTO 语句。可以在 QueryDef 对象的 SQL 属性中使用 SELECT 语句,也可以在数据控件的 RecordSource 属性中使用 SELECT 语句,或者将 SELECT 语句用作 OpenRecordset 方法的一个参数。SELECT 语句并不会改变数据库中的数据,它只是检索数据。

下面是 SELECT 查询的完整形式:

```
SELECT fieldlist  
FROM tablename IN databasename  
WHERE searchconditions  
GROUP BY fieldlist  
HAVING group criteria  
ORDER BY fieldlist  
WITH OWNERACCESS OPTION
```

提示: 最简单的 SELECT 查询可能是:

```
SELECT * FROM tablename
```

例如,下面的 SELECT 查询将返回 Employees 表中的所有记录的所有列:

```
SELECT * FROM Employees
```

其中的“*”表示要检索该表中的所有列。也可以指定只检索部分列。显示时,每一列中

的数据将按照它们排列的顺序出现。为了提高可读性可以重新排序：

```
SELECT [First Name], [Last Name] FROM Employees
```

(2) 指定数据源

SELECT 语句总有 FROM 子句,用来指出记录取自哪些表。

如果一个字段名被包含在 FROM 子句的多个表中,在它前面加上表的名和一个 . (点)运算符。在下面的例子中, Department 字段同时包含在 Employees 表和 Supervisor 表中。下面的 SQL 语句将从 Employees 表中选择 Department 字段,从 Supervisor 表中选择 SupvName 字段：

```
SELECT Employees.Department, SupvName _  
FROM Supervisors, Employees _  
WHERE Employees.Department = Supervisors.Department;
```

提示：当 FROM 子句列出多个表时,它们出现的顺序并不重要。

15.5 数据库的维护

维护数据库是数据库编程的重要方面之一。维护数据库的任务包括：

- 映射数据库以查明其当前结构。
- 压缩数据库以节省磁盘空间并提高性能。
- 修复受损数据库。

15.5.1 映射当前数据库结构的方法

维护数据库时,常常需要查看其基本结构或模型以确定数据访问对象的名称和属性。这样的操作就被称作映射数据库。一旦对数据库做了全面的映射,修改和写操作这些数据的应用就变得更加容易。

由于数据访问对象是以嵌套容器的层次结构而存在的,因此可用一系列嵌套的 For Each 循环遍历这个层次结构,来映射数据库。集合中的每个元素都可以用这种格式的语句块来访问。

```
For Each object in collection  
    '映射对象的属性。  
Next object
```

下面的示例代码映射了 Biblio.mdb 数据库。在研究注释的代码时请注意:为简便起见,只有 TableDef 对象的特征属性被完全映射了。如果希望映射其他对象的属性,只要简单地在标出的地方参考为 TableDef 属性写的代码作样本,插入恰当的语句就可以了。

```
Sub MapDatabase ()  
    Dim Db As Database, Td As TableDef, Fld As Field  
    Dim Idx As Index, Rel As Relation  
    Dim i As Integer, n As Integer  
    Set Db = DBEngine.Workspaces(0).OpenDatabase("Biblio.mdb")  
    '映射数据库属性。
```

```

Print "DATABASE"
Print "Name:", Db.Name
Print "Connect string:", Db.Connect
Print "Transactions supported?:", Db.Transactions
Print "Updatable?:", Db.Updatable
Print "Sort order:", Db.CollatingOrder
Print "Query time-out:", Db.QueryTimeout
'映射 TableDefs。
Print "TABLEDEFS"
For Each Td in Db.TableDefs
Print "Name:", Td.Name
Print "Created:", Td.DateCreated,
Print "Updated:", Td.LastUpdated,
If Td.Updatable = True Then
    Print "Updatable",
Else
    Print "Not Updatable",
End If
'显示 TableDef 的属性。
Print Hex $ (Td.Attributes)
If (Td.Attributes And dbSystemObject) < > 0 Then
    Print "System Object"
End If
If (Td.Attributes And dbAttachedTable) < > 0 Then
    Print "Attached table"
End If
If (Td.Attributes And dbAttachedODBC) < > 0 Then
    Print "Attached ODBC Table"
End If
If (Td.Attributes And dbAttachExclusive) < > 0 Then
    Print "Attached Table opened in exclusive mode"
End If
'映射每个 TableDef 中的字段。
Print "FIELDS"
For Each Fld in Td.Fields
    Print "Name:", Fld.Name
    Print "Type:", Fld.Type
    Print "Size:", Fld.Size
    Print "Attribute Bits:", Hex $ (Fld.Attributes)
    Print "Collating Order:", Fld.CollatingOrder
    Print "Ordinal Position:", Fld.OrdinalPosition
Print "Source Field:", Fld.SourceField
Print "Source Table:", Fld.SourceTable

```

如果需要可以在这里显示字段属性。如下所示：

```
Print Hex $ (Fld.Attributes)
If (Fld.Attributes And dbSystemObject) < > 0 Then
Print "System Object"
End If
Next Fld '得到 TableDef 中下一个字段。
'映射每个 TableDef 的索引。
Print "INDEXES"
For Each Idx in Td.Indexes
'设置索引变量。
Set Idx = Td.Indexes(n)
Print "Name:", Idx.Name
Print "Clustered:", Idx.Clustered
Print "Foreign:", Idx.Foreign
Print "IgnoreNulls:", Idx.IgnoreNulls
Print "Primary:", Idx.Primary
Print "Unique:", Idx.Unique
Print "Required:", Idx.Required
'映射索引的字段。
For Each Fld in Idx.Fields
Print "Name:", Fld.Name
Print "ForeignName:", Fld.ForeignName
Next Fld '得到索引中的下一个字段。
Next Idx '得到 TableDef 中的下一个索引。
Next Td '得到数据库中的下一个 TableDef。
'映射关系。
Print "RELATIONS"
For Each Rel in Db.Relations
Print "Name:", Rel.Name
Print "Attributes:", Rel.Attributes
Print "Table:", Rel.Table
Print "ForeignTable:", Rel.ForeignTable
'映射此关系的字段
For Each Fld in Rel.Fields.Count
Print "Name:", Fld.Name
Print "ForeignName:", Fld.ForeignName
Next Fld '得到关系中的下一个字段。
Next Rel '得到数据库中的下一个关系。
End Sub
```

15.5.2 压缩数据库

VB 把移去废弃页的操作推迟到关闭数据库并压缩废弃页之时,来保持高的性能状态。这种设计以可回收磁盘空间为代价来保持较高的数据库交互性能。不久,将会发现由于保存了

被添加和删除的数据,.mdb 数据库文件显著变大了。使用 DBEngine 对象的 CompactDatabase 方法,可以把一个数据库中的所有数据复制到另一个数据库中,而且在复制过程中在结果库里可以连续地组织数据,所以磁盘空间可以回收。但只能压缩 Jet 数据库。如果试图将此语句用在任何打开的数据库上,或不是 Jet 数据库格式的数据库上,会产生一个可以捕获的错误。

CompactDatabase 还提供了数据库被压缩时更改加密状态、版本和数据库区域的选项。换句话说,提供了将加密的数据库转换为不加密的数据库的机会,或者将不加密数据库加密的机会。还可以从一个地点转换到另一个,或从一个版本转到另一个。

在压缩数据库之前必须关闭数据库。另外,被压缩的数据库的源名和目标名决不能相同,因为如果 CompactDatabase 方法没有完成,被压缩的数据库将被删除。也不能在事务中执行压缩操作。

CompactDatabase 方法的语法是:

```
DBEngine.CompactDatabase sourcename, destinationname [, locale [, options]]
```

参 数	描 述
sourcename	要压缩的数据库的完整路径和数据库文件名
destinationname	新产生的压缩数据库的完整路径和数据库文件名。不要给源文件和目标文件指定相同的名字
locale	与 CreateDatabase 的 locale 参数含义相同
options	与 CreateDatabase 的 options 参数含义相同

例如,下面的代码压缩了一个名为 Old.mdb 的数据库,并产生了一个区域为西班牙语的名为 New.mdb 的数据库:

```
DBEngine.CompactDatabase "C:\Old.mdb", C:\VB\New.mdb, dbLangSpanish
```

15.5.3 修复数据库

如果 Jet 数据库受到损伤,可以使用 DBEngine 对象的 RepairDatabase 方法加以修复。该方法能使因不完整的数据页读/写操作而失效的数据库重新有效。在数据库非正常关闭(如电源故障)时,就会产生此类毁损形式。RepairDatabase 方法并不能修复所有可能的数据库毁损,因此应该牢牢记住要定期地备份数据库以避免不可恢复的数据库丢失。

RepairDatabase 方法检查数据库中的所有数据页来纠正链接、核实所有系统表并核实所有索引。在成功地完成之后,数据库又变成“安全”的了,所有不能修正的(如非法地引用其他页)页会被放弃。

RepairDatabase 方法只需要一个参数,即希望修复的数据库文件名。例如可以使用以下的代码来修复名为 Biblio.mdb 的数据库:

```
DBEngine.RepairDatabase "Biblio.mdb"
```

提示:数据库在被修复后大小会增加,因为在创建索引的过程中会留下一些被删除的页。在修复后接着运行 CompactDatabase 方法来消除不必要的页是个好方法。

15.6 本章小结

本章介绍系统地介绍了 VB 数据库编程。读者学完本章之后,应掌握如下几个方面:

1. 数据库的基本概念包括表、字段、记录和键等。
2. 几个重要的数据库控件及其编程包括 DATA 控件,DBCGrid 控件及其他数据库绑定控件。
3. SQL 语言,特别是 SELECT 语句。
4. 了解 DAO 编程,在用控件编程之后,试用 DAO 编程。
5. 了解数据库的映射及其维护。

实际上数据库这部分的内容很多,要想在程序中熟练应用数据库,前面讲的知识远远不够详细,还请读者自己在实际中慢慢锻炼。

第 16 章 WEB 浏览

本章将介绍如何用 VB 来编写 WEB 浏览器,以及在 Internet 应用程序中使用 ActiveX 文档,控件和代码部件。

16.1 编写 WEB 浏览器

我们可以用微软公司的 Internet Explorer 或网景公司的 Netscape 的浏览器在网上浏览。通过 VB 的强大功能我们也可以用 VB 编写浏览器。

下面介绍两种用 VB 编写浏览器的方法:

1. 使用 Browser 窗体。
2. 使用 WebBrowser 控件。

16.1.1 使用 Browser 窗体编写 WEB 浏览器

用 Browser 窗体编写 WEB 浏览器的步骤如下:

1. 单击[工程]菜单,选择[添加窗体],在[新建]中选择“浏览器窗体”。
2. 修改浏览器窗体的属性设置,MDIChild = false。
3. 单击[工程]菜单,选择[属性]选项,设置其启动窗体为 Web。

运行程序即可。

程序运行结果如图 16.1 所示。

下面是代码简析。

处理 Toolbar 的代码:

```
Private Sub tbToolBar_ButtonClick(ByVal Button As Button)
    On Error Resume Next
    tmrTimer.Enabled = True
    Select Case Button.Key
        Case "Back"
            brwWebBrowser.GoBack
        Case "Forward"
            brwWebBrowser.GoForward
        Case "Refresh"
            brwWebBrowser.Refresh
        Case "Home"
            brwWebBrowser.GoHome
        Case "Search"
            brwWebBrowser.GoSearch
        Case "Stop"
            tmrTimer.Enabled = False
    End Select
End Sub
```

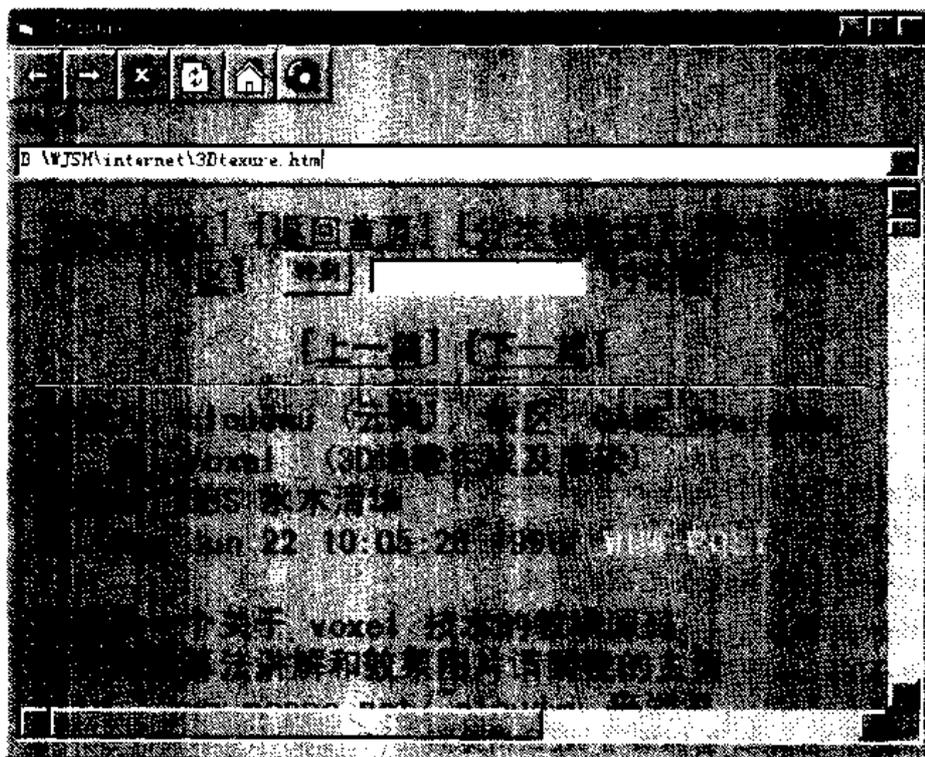


图 16.1 Browser 窗体编写的 WEB 浏览器

```

brwWebBrowser.Stop
Me.Caption = brwWebBrowser.LocationName
End Select
End Sub

```

以上代码包括了向前,向后,刷新,回到主页,搜索。
键入 WWW 网址后的处理:

```

Private Sub cboAddress_Click()
    If mbDontNavigateNow Then Exit Sub
    timTimer.Enabled = True
    brwWebBrowser.Navigate cboAddress.Text
End Sub

```

其中 cboAddress.Text 中记录了网址。
其他为辅助代码:

```

Private Sub brwWebBrowser_DownloadComplete()
    On Error Resume Next
    Me.Caption = brwWebBrowser.LocationName
End Sub
Private Sub brwWebBrowser_NavigateComplete(ByVal URL As String)
    Dim i As Integer
    Dim bFound As Boolean
    Me.Caption = brwWebBrowser.LocationName
    For i = 0 To cboAddress.ListCount - 1

```

```

        If cboAddress.List(i) = brwWebBrowser.LocationURL Then
            bFound = True
            Exit For
        End If
    Next i
    mbDontNavigateNow = True
    If bFound Then
        cboAddress.RemoveItem i
    End If
    cboAddress.AddItem brwWebBrowser.LocationURL, 0
    cboAddress.ListIndex = 0
    mbDontNavigateNow = False
End Sub

Private Sub cboAddress_KeyPress(KeyAscii As Integer)
    On Error Resume Next
    If KeyAscii = vbKeyReturn Then
        cboAddress_Click
    End If
End Sub

Private Sub timTimer_Timer()
    If brwWebBrowser.Busy = False Then
        timTimer.Enabled = False
        Me.Caption = brwWebBrowser.LocationName
    Else
        Me.Caption = "运行中..."
    End If
End Sub
End Sub

```

16.1.2 使用 WebBrowser 控件编写 WEB 浏览器

在窗体中放置如图 16.2 所示控件：包括六个按钮，一个标签，一个文本框，一个 WebBrowser 控件。

设置 WebBrowser 控件的 Name = "Web"加入如下代码：

向后：

```

Private Sub Command1_Click()
    Web.GoBack
End Sub

```

向前：

```

Private Sub Command2_Click()
    Web.GoForward
End Sub

```

回到微软主页：

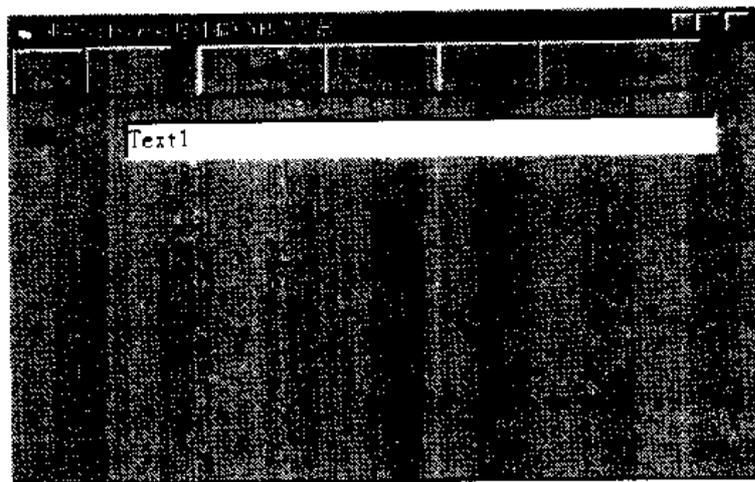


图 16.2 WebBrowser 控件

```
Private Sub Command3_Click()
    Web.GoHome
End Sub
```

停止：

```
Private Sub Command4_Click()
    Web.Stop
End Sub
```

刷新：

```
Private Sub Command5_Click()
    Web.Refresh
End Sub
```

搜索：

```
Private Sub Command6_Click()
    Web.Navigate Text1.Text
End Sub
```

```
Private Sub Form_Resize()
    With Web
        .Left = Form1.ScaleLeft
        .Width = Form1.ScaleWidth
        .Height = Form1.ScaleHeight - Web.Top
    End With
End Sub
```

令 Text1 的 text 等于统一资源定位器。

```
Private Sub Web_BeforeNavigate(ByVal URL As String, ByVal Flags As Long, ByVal TargetFrameName As String,
PostData As Variant, ByVal Headers As String, Cancel As Boolean)
    Text1.Text = URL
End Sub
```

16.2 VB 的 Internet 编程

Internet 技术为程序员发挥自己的才能不断地提供了越来越多复杂和多样的方法。在过去,“Internet 应用程序”可能一度被当作只不过是用文件传送协议来下载文件,或者浏览 HTML 页面的实用工具,而在今天,那些应用程序可能与所创建的其他与 Internet 无关的应用程序一样,可以多种多样。简而言之,除了为 Internet 开发应用程序之外,现在,也可以利用 Microsoft VB 将 Internet 技术结合进去,作为范围更广的解决方案的一部分。

VB 的这个版本,使得创建 ActiveX 部件成为可能,它在面向 Internet 解决方案中能起重要的作用。比如可以创建在 HTML 页面上使用的 ActiveX 控件。所开发的 ActiveX 文档,也可以在 Internet 浏览器中使用,并和其他部件一起与 HTML 页面集成到一起。而 ActiveX 代码部件(先前称为 OLE 服务器)则可以部署在客户或者服务器计算机上,以便于后台处理之用。

下面简单介绍 Internet 开发环境。

严格说来,Internet 是操作在以 TCP/IP 为基本协议集的全球性的、非集中化的计算机网络。然而,撇开全球性的概念,Internet 技术也可以应用到本地网络。大公司的内部网之所以使用 Internet 技术,就在于它们可以基于 TCP/IP 成为非集中的网络。计算机之间可以用与 Internet 上所使用的核心同样的协议,在公司的内部网上交换信息。

一方面,Internet 技术只是为程序员的开发成果提供了另一个领域。例如,当连接 Web 来部署 ActiveX 控件时,可以用另一种方式进行——把 HTML 和 VBScript 代码结合起来,并加上安全性的特点,等等。但是仍然是在调用方法、设置属性,以及处理事件。这样,作为 VB 开发者,就可以将自己所掌握的知识运用到 Internet 的赛场。

另一方面,应用 Internet 技术,可以通过令人振奋的新途径,扩展自己的开发计划。例如,可以将 ActiveX 部件与 Internet 技术相结合,其优点包括:

- 易于客户维护——作为某个 Web 的部分而部署的部件,可以在需要时下载下来,这就消除了要求用户通过运行安装程序来更新的必要性。
- 扩展 Internet 功能的能力——通过公共协议,在某个内部网和 Internet 之间切换,对用户来说实际上可能是不可见的。

16.3 VB 的 Internet 应用程序简述

本节将介绍在 VB 的 Internet 应用程序中如何使用 ActiveX 文档、控件和代码部件。

注意: 在阅读本章之前读者应具备一些 VBScript 的简单基本知识,而这部分知识不在本书讲的范围之内。

16.3.1 在 Internet 应用程序中使用 ActiveX 文档

使用 ActiveX 文档,可以开发与 Internet 或者内部网站点的其他元素紧密集成在一起的 VB 应用程序。以这种方式来部署部件,用户可以在 HTML 页面和 ActiveX 文档之间透明地定位。

下面简单介绍 Internet 应用程序。

可以把 ActiveX 文档想象为 VB 的 Internet 应用程序。当前用标准应用程序所作的任何事

情,几乎都可以用 ActiveX 文档来完成。这样的示例包括:

- 作为三级客户/服务器应用程序的用户接口部件,调用其他部件所提供的业务服务,或者将查询提交给远程数据库,并在网格中显示返回结果。
- 比带有控件的 HTML 页面更为复杂的客户端处理。ActiveX 文档同时还提供了更丰富、更复杂的环境,设计和调试窗体。

像 HTML 页面元素的情况一样,如果某些 ActiveX 文档还没有安装到用户计算机上时,那么当用户定位到它们时,对它们就可以自动进行下载,如果服务器上可用版本更新,还可以自动升级。

1. ActiveX 文档的特点

下面的特点综合在一起,ActiveX 文档提供了安装、定位以及持久性的一种使用方式:

- Internet 部件下载——可以创建对自己 ActiveX 文档的链接,它能使浏览器自动找到并下载运行部件所需要的所有部件。
- 超链接对象——在超链接意识的容器中,可以用 VB 的超链接对象的属性和方法,跳转到给定的 URL,或者在整个历史记录中定位。
- 菜单协商——就像通过 Microsoft Word 或者 Microsoft Excel 可以使用的文档一样,VB 的 ActiveX 文档提供了将其菜单与浏览器中的那些菜单融合到一起的能力。比如说,当在 Internet Explorer 中加载文档时,浏览器的菜单项,将与程序员所创建出来并附加到 ActiveX 文档中的那些菜单项融合到一起。
- PropertyBag——在 Internet 资源管理器中部署 ActiveX 文档时,通过 PropertyBag 对象可以固定数据。

2. 部署 ActiveX 文档

下面介绍如何在 Internet 资源管理器中部署 ActiveX 文档。

从由 HTML 和 VBScript 结合的 HTML 页面可链接到 ActiveX 文档。在下面的例子中,从 HTML 页面上的超链接到达 ActiveX 文档。

- 用 HTML 提供对引用 ActiveX 文档的 HTML 页面的链接。

因为 ActiveX 文档将取代 Internet Explorer 中的 HTML 页面,所以有必要将部件的 OBJECT 标签放到单独的 HTML 页面上,然后跳转到该页面,当 ActiveX 文档加载后,它将有提示地消失。

下面的 HTML 代码创建了一个超链接,从第 1 个页面到第 2 个页面(下载 ActiveX 文档的那一个页面):

```
< a href="Accounts.htm"> View or update accounts </a>
```

- 使用 HTML 为浏览器提供了对 ActiveX 文档的下载、注册以及定位的方法。

下面的 HTML 片段,指示 Internet 资源管理器下载 ActiveX 部件,并在 Windows 注册表中对 ActiveX 文档进行注册:

```
< OBJECT  
  classid = "clsid:2F390484-1C7D-11D0-8908-00A0C90395F4"  
  codebase = "Accounts.cab # version = 1,0,0,0" >  
</OBJECT>
```

本例中的 OBJECT 标签包括:

- ActiveX 文档的类 ID,这样就可以把它包括在 Windows 注册表中,或者在那里找到它。
- CODEBASE 属性,它告诉浏览器,如果部件尚未在客户机器上,应到何处才能找到该部件(同时还有一个版本号,进行检查比较,决定是否需要更新)。
- 在同一页上,放置 VBScript,它指示 Internet 资源管理器,通过 ActiveX 文档的 .vbd 文件,直接定位到该文档(这个 .vbd 文件,是在对 ActiveX 文件进行编译时创建的,它包含了一个指针,该指针指向提供 ActiveX 文档对象的部件)。

```
< SCRIPT LANGUAGE = "VBScript" >
Sub Window_OnLoad
Navigate "Accounts.vbd"
End Sub
</SCRIPT >
```

这个代码片段仅包含给定的 .vbd 文件名,而非一条完整限定的路径。使用该文件相关路径,Internet Explorer 将在包含 VBScript 的 .htm 文件的同一目录中寻找它。

16.3.2 在 Internet 应用程序中使用 ActiveX 控件

ActiveX 控件可以使 HTML 页面活跃起来,并为其增添力量。除了使用自己所创建的控件以外,也可以用 VB 的专业版和企业版所提供的三个控件,扩展使用 Internet 技术的标准应用程序。

下面介绍 HTML 页面上的 ActiveX 控件。

使用 HTML 和 VBScript,在 HTML 页面上可以包括 ActiveX 控件、设置它们的属性、调用它们的方法,以及处理它们的事件。于是,包含控件的 HTML 页面就像是 VB 设计器一样。

一般来说,ActiveX 控件可以借助带有交互或者动画的用户接口特点,使得 HTML 页面更有活力。

将 HTML 和 VBScript 结合起来,可引用和描述 ActiveX 控件的行为。在下面的示例中,嵌入了两个标签类型的控件,从而提供了可单击的目录登录项表。

- 使用 HTML,为浏览器提供了对 ActiveX 控件进行下载、注册以及引用的方法。

```
< OBJECT
  classid = "clsid:2F390484-1C7D-11D0-8908-00A0C90395F4"
  codebase = "http://www.mysite.com/controls/label.ocx # version = 1,0,0,0"
  id = "Welcome"
  width = "150"
  height = "20"
  align = "center"
  vspace = "0"
  >
  < PARAM name = "Caption" value = "Welcome Page" >
  < PARAM name = "FontName" value = "News Gothic MT" >
  < PARAM name = "FontSize" value = "11" >
  < PARAM name = "FontBold" value = "1" >
  < PARAM name = "ForeColor" value = "000000" >
</OBJECT >
```

< OBJECT

```
classid = "clsid:2F390484-1C7D-11D0-8908-00A0C90395F4"  
codebase = "http://www.mysite.com/controls/label.ocx # version = 1,0,0,0"  
id = "Catalog"  
width = "150"  
height = "20"  
align = "center"  
vspace = "0"
```

>

```
< PARAM NAME = "Caption" value = "Our Catalog" >  
< PARAM NAME = "FontName" value = "News Gothic MI" >  
< PARAM NAME = "FontSize" value = "11" >  
< PARAM NAME = "FontBold" value = "1" >  
< PARAM NAME = "ForeColor" value = "000000" >
```

</OBJECT >

该代码片段中的 OBJECT 标签包括:

- 控件的类 IDs,这样就可以把它们包括在 Windows 注册表中,并在其中找到它们。
- ID 属性,当在 VBScript 中引用控件时将要用到它(与 VB 中的 Name 属性类似)。
- CODEBASE 属性,它告诉浏览器,如果部件尚未在用户机器上,应到何处才可以找到该部件(同时还有一个版本号,检查比较决定是否需要更新)。
- PARAM NAME 标签,用它来设置控件的属性值。
- 使用 VBScript 调用句柄单击事件,这样当单击每个控件时,都将把特定的 HTML 框架中的某个特定 .htm 文件加载进去:

```
< SCRIPT LANGUAGE = "VBScript" >
```

```
Sub Welcome_Click
```

```
Parent.Parent.Frames(1).Location.Href = "Welcome.htm"  
Welcome.ForeColor = 000000
```

```
End Sub
```

```
Sub Catalog_Click
```

```
Parent.Parent.Frames(1).Location.Href = "Catalog.htm"  
Catalog.ForeColor = 000000
```

```
End Sub
```

```
</SCRIPT >
```

VB 专业版还包括有三种专门为了对 Internet 相关技术进行封装而设计的控件。这些控件包括:

- Internet 传输控件——它包装了三种公用的 Internet 协议:HyperText Transfer Protocol (HTTP)、File Transfer Protocol (FTP) 以及 Gopher。
- WebBrowser 控件——通过 Internet 资源管理器把许多可用的功能合并进去。
- WinSock 控件——允许连接到远程机器,并用 User Datagram Protocol (UDP) 或者 Transmission Control Protocol (TCP) 协议来交换数据。

这里是几个用 Internet 技术扩展的应用程序示例:

- 添加从 FTP 站点自动下载文件的功能。
- Web 浏览器的 VB IDE 外接程序。可以从内部部署为 HTML 页面的某个公用代码仓库对示例进行访问,或者使用 Internet,对第三程序站点进行访问。

16.3.3 在 Internet 应用程序中使用 ActiveX 代码部件

正如使用 ActiveX 控件一样,可以使用 ActiveX 代码部件(.dll 或者 .exe 文件),将功能添加到客户端或者服务器端上的 HTML 页面中。

1. 下载给客户的代码部件

将代码部件部署在客户端,可以提供速度上的优越性,因为用户命令不需要传递给服务器。若是将部件部署在服务器端的话,也有好处,那就是能够显示出用户接口元素。通过下载 DLL 可能得到好处的例子包括:

- 将用户接口元素(比如登录对话框)添加到 HTML 页面中。
- 提供了后台客户端处理,比如与 VBScript 在一起的函数库。

若是部署在客户端的话,可使用这样的 HTML 和 VBScript 代码,引用和描述代码部件——这些 HTML 和 VBScript 代码与用于 ActiveX 控件的那些代码是同一类型的。下面的 HTML 和 VBScript 代码部件,提供怎样用 DLL 显示登录对话框的示例。

(1) 用 HTML 来创建一个“窗体”,该窗体包含用来调用该登录对话框的按钮。包括有:在脚本中要用到的窗体名和按钮名,在该页上显示的类型(在这里是一个命令类型的按钮),以及一个“值”,它与 VB 中的某个 CommandButton 的 Caption 属性相类似:

```
< FORM NAME = "LoginButton" >
Click here to log in:
< INPUT NAME = "cmdLogin" TYPE = "Button" VALUE = "Log in..." >
</FORM >
```

(2) 用 HTML 为浏览器对 ActiveX 部件进行下载、注册以及引用,提供一种方法。

```
< OBJECT
classid = "clsid:2F390484-1C7D-11D0-8908-00A0C90395F4"
id = "Login"
codebase = "Login.cab # version = 1,0,0,0"
>
</OBJECT >
```

在该例中,OBJECT 标签包括:

- 部件的类 ID,这样该部件就可以被包括在 Windows 注册表中,并在那里找到它。
- ID 属性,当引用 VBScript 中的类时要用到它(正如 VB 中的 Name 属性一样)。
- CODEBASE 属性,它告诉浏览器,当部件尚未在客户机器上,应到何处才可以找到该部件(同时还有一个版本号,检查比较决定是否需要更新)。
- 用 VBScript 调用显示登录对话框部件中的方法:

```
< SCRIPT LANGUAGE = "VBScript" >
```

(3) 为包含按钮的 HTML 窗体,以及为显露这种显示对话框方法的对象,创建一些变量。

```
Dim dlgLogin
```

```
Dim TheForm
Set TheForm = Document.LoginButton
```

将过程包括进来,该过程在单击按钮时将显示对话框。

```
Sub cmdLogin_OnClick
Set dlgLogin = Login
dlgLogin.ShowDialog
End Sub
</SCRIPT>
```

部件本身将包括这些代码,它们组成登录字符串,并将其发送到服务器进行验证。

2. 访问在服务器端部署的 DLLs

将自己的 DLL 与 Internet 技术相结合的另一种方法,是把它部署在服务器端。如果有 Internet Information Server (IIS)(它包括在 Windows NT Server 4.0 中)的话,就可以用 Oleisapi2.dll 来调用 VB 内置的 DLLs 函数,这些函数包含在 VB CD-ROM 的 Tools 目录下。

Oleisapi2.dll 包装了 Internet Server Application Programming Interface (ISAPI) 的功能,这是 IIS 的一个特点。可以将自己的 DLL 部署在 HTTP 服务器上,并利用从 HTML 页面发送来的信息调用其方法。简言之,Oleisapi2.dll 所起的作用,就是充当对 DLLs 请求的媒体。就像在 HTML 页面上所编制的代码一样,这些请求都只不过是一些 URLs。

服务器端 DLLs 的使用,可能包括:

- 定制后的 HTML 页面的创建和返回——或许是根据客户提交的参数。
- 管理数据库连接,对收到的查询以及返回的结果进行调度。

16.4 本章小结

本章主要介绍了 VB6 的网络编程。读者学完本章后应能:

1. 用 VB 编制自己的浏览器
2. 理解 VB 的 INTERNET 编程环境。
3. 了解 VB 的 ActiveX 文档,ActiveX 代码部件,编一些简单的 Internet 应用程序。

附录 A 安装和卸载 VB 6.0 中文版

1. 准备工作

在安装 VB 之前,必须确认计算机满足最低安装要求,并阅读安装盘根目录下的文件 Readme。

(1) 检查硬件和系统需求

为运行 VB,必须在计算机上安装相应的硬件和软件系统。这些系统要求包括:

- Microsoft Windows NT 3.51 或更新版本,或 Microsoft Windows 95/98。
- 80486 或更高微处理器。
- 如果是完全安装,则至少需要 140 MB 的硬盘空间。
- 一个 CD-ROM 驱动器。
- Microsoft Windows 支持的 VGA 或分辨率更高的监视器。
- 16 MB 或 16MB 以上 RAM。
- 鼠标或其他定点设备。

(2) 阅读文件 Readme

文件 Readme 列举了自 VB 文档出版发行以来的更改。可在最初的安装屏幕上单击【显示 Readme】按钮来读取该文件,也可在 CD-ROM 的根目录下查找到这个文件。

2. 第一次安装 VB 6.0 中文版

从 CD 盘上安装 VB 6.0,可执行如下操作:

(1) 开启计算机并进入 Windows 95 / 98 或 Windows NT 界面后,在 CD-ROM 驱动器中插入 VB 6.0 中文版的 CD 盘。

(2) 安装程序在 CD 盘的根目录下、可运行安装盘根目录下的 Setup.exe 来加载安装程序。如果用户的计算机能够在系统中运行 AutoRun (自动运行),则在插入 CD 盘时,安装程序将被自动加载。这时出现如图 A.1 所示的安装向导画面。

(3) 单击【下一步】按钮,显示最终用户许可协议,选中【接受协议】单选框在单击【下一步】按钮。

(4) 安装程序打开【产品号和用户 ID】对话框,再此对话框输入产品 ID 号以及用户姓名和公司名称。如图 A.2 所示。

(5) 单击【下一步】按钮,如果用户的计算机上未安装 IE4.01 或以上版本,则安装程序打开【安装 Internet Explorer 4.01】对话框,选中其中的复选框并单击【下一步】按钮。

注意: 如果用户的计算机上已经安装了 IE4.0 以上版本,或者操作系统是 Windows 98,那么安装程序跳过这一步,直接开始安装 VB 6.0 中文版。

(6) 按照屏幕提示操作——在安装选项中选择“标准安装”,在国家地区列表中选择“中国”,并选择安装路径,单击【下一步】按钮。

(7) 安装程序将 Internet Explorer 4.01 的文件安装到指定路径,完成 IE 安装后,要重新启动计算机才能继续安装。



图 A.1 开始安装 VB 6.0 的安装向导

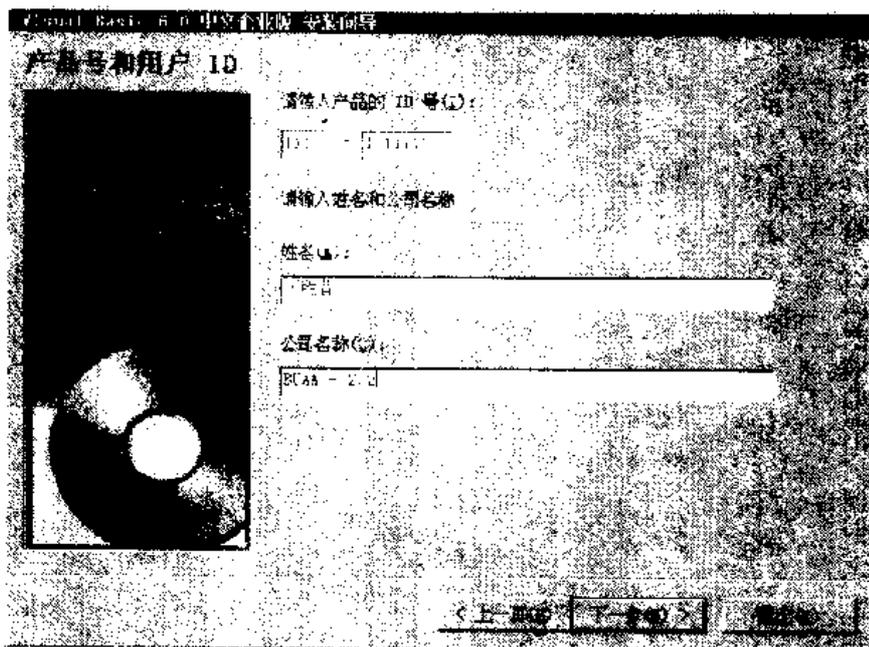


图 A.2 输入产品 ID 和用户、公司名

(8) 启动计算机后, 安装程序显示对话框提示用户 IE4.01 安装成功, 单击【下一步】按钮, 这时出现如图 A.3 所示【VB 6.0 中文企业版】对话框, 选择【安装 VB 6.0 中文企业版】单选框, 然后单击【下一步】按钮。

(9) 安装程序显示【选择公用文件的文件夹】对话框, 单击【浏览】按钮为公用文件选择一个文件夹, 如图 A.4 所示。

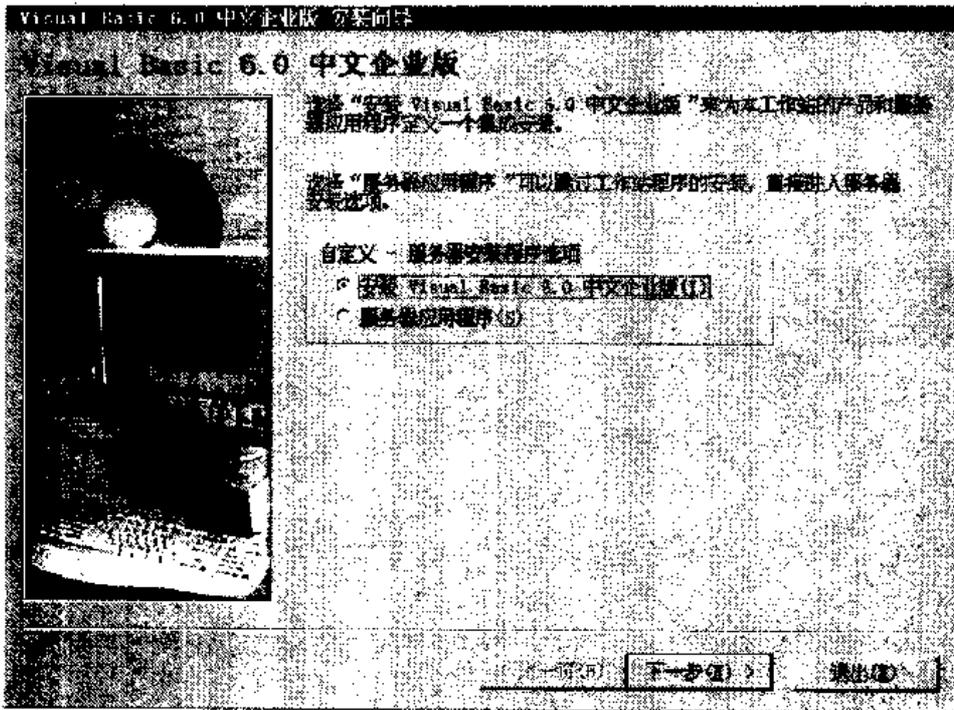


图 A.3 开始安装 VB 6.0 中文企业版

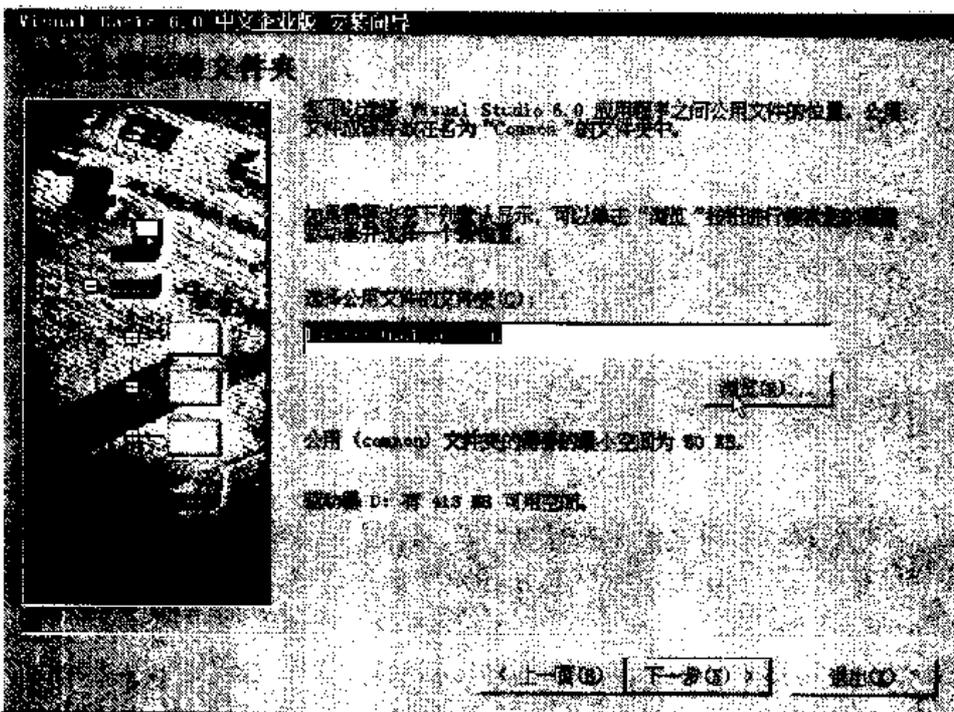


图 A.4 选择公用安装文件夹

(10) 单击【下一步】按钮,出现一个消息框,单击其中的【继续】按钮,安装程序开始搜索已安装的组件,然后打开如图 A.5 所示的对话框,让用户选择安装类型,单击【自定义安装】按钮,允许用户自定义安装组件;单击【典型安装】按钮,将安装最典型的组件;单击【更改文件夹】按钮,可选择安装路径。

(11) 在这里,我们单击【自定义安装】按钮,安装程序显示【自定义安装】对话框,在【选项】

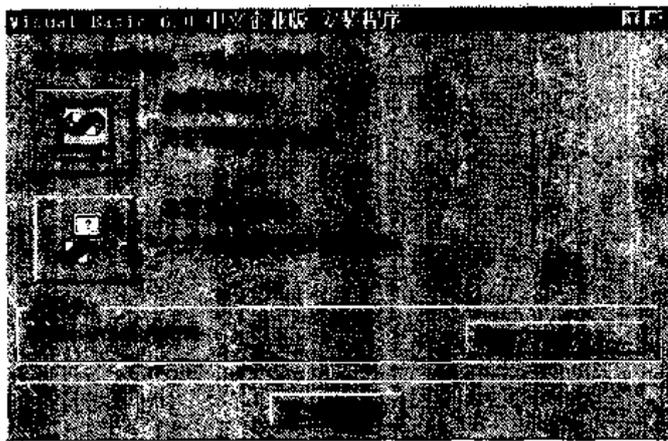


图 A.5 选择安装类型

列表中选中要安装的组件,清除不安装的组件,如果要完全安装,可单击【全部选中】按钮。如图 A.6 所示。

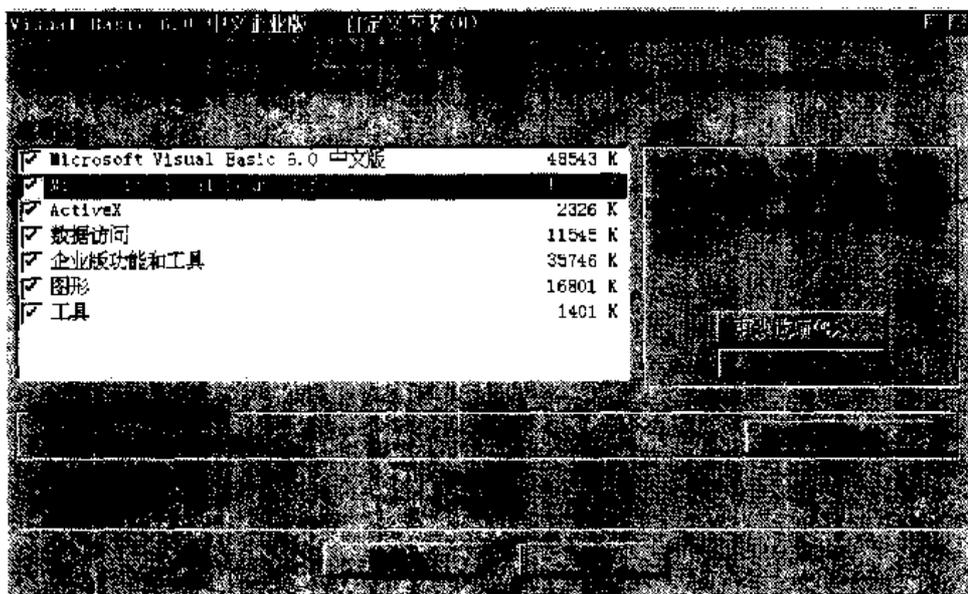


图 A.6 【自定义安装】对话框

(12) 单击【继续】按钮,接下来安装程序检查可用磁盘空间,如果磁盘空间足够,便开始复制程序文件到硬盘中,复制文件完毕后,提示用户须重新启动计算机来完成安装过程,单击【重新启动 Windows】按钮。

(13) 系统启动后,安装程序显示如图 A.7 所示的【安装 MSDN】对话框,如果要安装 MSDN Library,将 MSDN 光盘插入 CD ROM 驱动器,并选中该对话框中的【安装 MSDN】复选框,然后单击【下一步】按钮,如果不想安装 MSDN,清除该复选框即可。

(14) 安装程序显示【服务器安装】对话框,从【服务器组件】列表中选择要安装的服务器工具并单击【安装】按钮,安装完毕,再启动计算机后,安装程序返回到该对话框,然后单击【下一步】按钮。如果不想安装服务器工具,可直接单击【下一步】按钮。

(15) 接着安装程序显示注册对话框,如果要现在注册,选中【注册】复选框,然后单击其中的【完成】按钮开始注册;如果不想注册,清除【注册】复选框,直接单击【完成】按钮来完成安装

过程。

至此,整个安装过程结束,用户可以开始启动 VB 6.0 来编写 VB 应用程序。

提示:不能直接将 CD-ROM 上的文件复制到硬盘,然后从硬盘运行 VB,必须使用安装程序将文件解压缩并安装到合适的目录中。

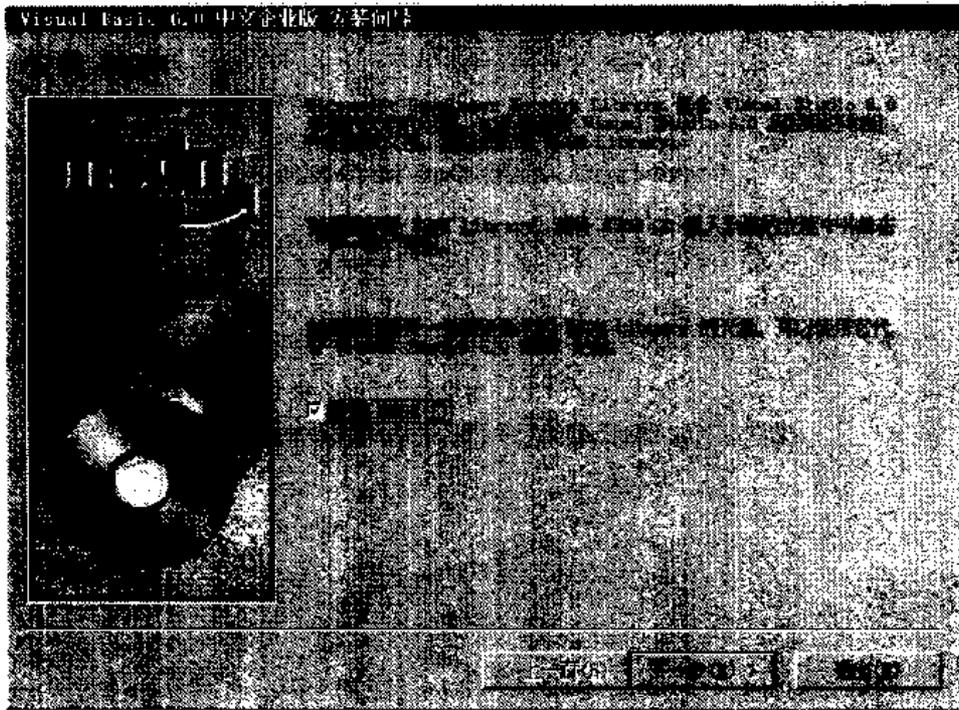


图 A.7 【安装 MSDN】对话框

3. 添加或删除 VB 部件

第一次安装 VB 6.0 后,可在必要时随意多次运行安装程序。例如,可在另一个目录下重新安装 VB,也可安装 VB 的其他部分。

添加或删除 VB 部件可按以下步骤进行:

(1) 启动计算机后,在 CD-ROM 驱动器插入 CD 盘。

(2) 打开 Windows 的【控制面板】窗口,双击【添加/删除程序】图标,系统打开【添加/删除程序】对话框,如图 A.8 所示。

(3) 单击【安装/卸载】选项卡,从列表中选择“Microsoft VB 6.0 中文企业版”,再单击【添加/删除】按钮。这时打开如图 A.9 所示的【Microsoft VB 6.0 中文企业版 安装程序】对话框,单击对话框中的【添加/删除】按钮。

提示:如果要重新安装 VB 6.0,可单击【重新安装】按钮,如果要删除 VB6 的全部组件,可单击【全部删除】按钮,如果要退出安装,可单击【退出安装】按钮。

(4) 系统打开如图 A.10 所示的【Microsoft VB 6.0 中文企业版 - 维护】对话框,在【选项】列表中选中需要安装的项目并清除需要删除的项目,然后单击【继续】按钮。

(5) 接下来按照屏幕提示操作即可。

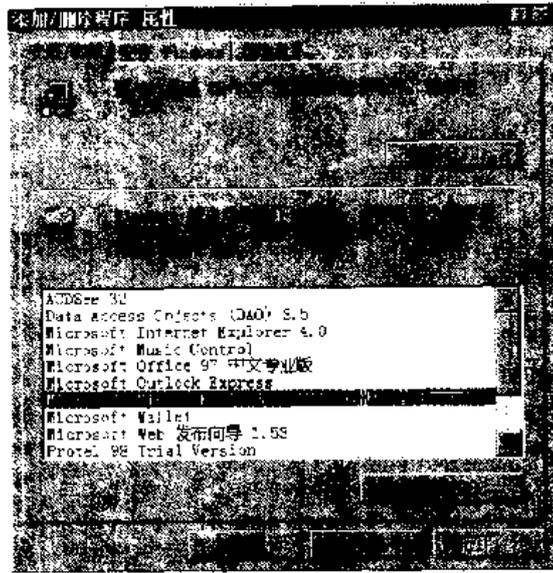


图 A.8 【添加/删除程序】对话框

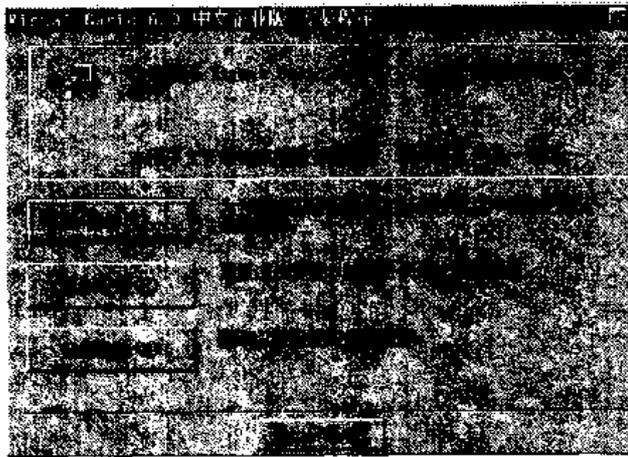


图 A.9 【Microsoft VB 6.0 中文版安装程序】对话框

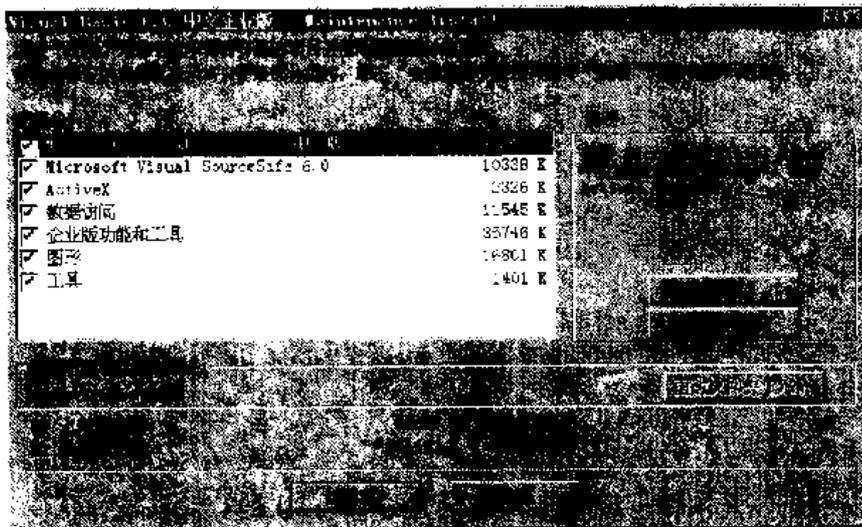


图 A.10 【Microsoft VB 6.0 - 维护】对话框

附录 B 菜单总汇

一、【文件】菜单

【文件】菜单包括 16 个菜单项：

- **新建工程**
打开【新建工程】对话框,创建新的工程文件。
- **打开工程**
打开一个现有的工程文件。
- **添加工程**
显示【添加工程】对话框,往已打开的工程组中加入一个新工程或现有工程,并建立一个包含原工程和新打开工程的工程组。
- **移除工程**
在工程组中移去选定的工程。
- **保存工程**
保存当前工程。
- **工程另存为**
把工程存为其他名字。
- **保存 Form1**
保存当前窗体文件。
- **Form1 另存为**
把当前文件以其他名字保存。
- **保存选择**
保存工程中选择的文件。
- **保存更改的脚本**
将当前修改过的脚本保存。
- **打印**
设置部分打印选项,打印窗体和代码。
- **打印设置**
设置打印信息。
- **生成工程 1.EXE1**
生成当前工程或工程组的 EXE、或 DLL、OCX 文件。
- **生成工程组**
对工程组中的每个工程生成一个 EXE、ELL、OCX 文件。
- **文件 1、2、3、4**
按次序列出最近打开的四个工程(.Vbp)或工程组(.Vbg)的名字。
- **退出**
退出 VB 6.0。

二、【编辑】菜单

【编辑】菜单包括 25 个菜单项；

- **撤消**

取消前一次操作。

- **重复**

重复前一次操作。

- **剪切**

把选择的内容剪切到 Windows 的剪切板中。

- **复制**

把已选择的内容拷贝到剪切板中。

- **粘贴**

把剪切板中的内容粘贴到当前光标所在位置。

- **粘贴链接**

把链接粘贴到一个有效的 DDE 源上链接是一个 DDE 术语,表示两个对象建立一个动态数据交换关系。

- **移除**

移除当前选定的对象。

- **删除**

删除当前选择的控件、文本或观察表达式。

- **从数据库删除表**

从数据库中删除选择的表。

- **全选**

在代码窗口中选择全部文本,或选择窗体上的全部控件。

- **选择全部列**

选择当前表中的全部列。

- **表**

对选择的表进行操作,如:设置主关键字、插入列、删除行。

- **查找**

在代码窗口中查找指定的文本。

- **查找下一个**

查找指定文本下一次出现位置。

- **替换**

替换指定的文本。

- **缩进**

使选择的文本行向右移动一 Tab 位置。

- **凸出**

使选择的文本行向左移动一 Tab 位置。

- **插入文件**

打开一个文件,把其中的代码插入当前位置。

- **属性/方法列表**

在代码窗口中用下拉式列表显示对象的属性和方法,如果光标处于空白位置,则在列表中显示所有可用的方法;在【选项】对话框【编辑器】选项卡中选中自动列表成员功能,则可自动打开列表。

- **常数列表**

列出属性的有效值,以供用户在编程时使用。

- **快速信息**

显示代码窗口中选择的变量、方法、语句、函数和过程的语法。

- **参数信息**

在代码窗口弹出一列表窗口,显示函数或语句的参数信息。

- **自动完成关键字**

输入一个单词的前几个字母后,若 VB 能推测出后面的字母,则自动插入后面的字母。

- **到行**

在表中定位光标所在的行。

- **书签**

在代码中显示一个菜单,该菜单包含切换书签、移动到下一个或前一个书签、清除所有书签命令。

三、【视图】菜单

【视图】菜单包括 21 个菜单项:

- **代码窗口**

显示或激活代码窗口。

- **对象窗口**

显示或激活的对象窗口,如窗体等。

- **定义**

显示光标所在处的变量或过程的定义。

- **最后位置**

在代码窗口中使光标回到上一次的位置。

- **对象浏览器**

显示对象浏览器,列出对象库、类库、类、方法、属性、事件和代码中使用的常量、模块和过程。

- **立即窗口**

在【立即】窗口中显示调试语句和输入命令的结果。

- **本地窗口**

在【本地】窗口中显示当前栈中所有变量及其值。

- **监视窗口**

在监视窗口中显示当前观察表达式的值。

- **调用堆栈**

显示调用堆栈对话框,在中断模式下列出一个过程正在调用的其他过程。

- **工程资源管理器**

显示工程资源管理器窗口。

- **属性窗口**
显示当前所选对象的属性窗口。
- **窗体布局窗口**
显示窗体布局窗口。
- **属性页**
显示用户控件的属性页。
- **表**
显示表的一些属性,如:名称、关键字等。
- **缩放**
对当前选中的对象进行缩放显示。
- **显示窗格**
显示图表、网格、SQL 或结果。
- **工具箱**
显示标准 VB 控件。
- **数据视图窗口**
显示或激活数据视图窗口。
- **调色板**
显示调色板。
- **工具栏**
设置或自定义工具栏显示。
- **Visual Component Manager**
显示部件管理器对话框。

四、【工程】菜单

【工程菜单】中有 17 个菜单项:

- **添加窗体**
打开[添加窗体]对话框,可在工程中加入一个窗体。
- **添加 MDI 窗体**
打开[添加 MDI 窗体]对话框,可在工程中加入一个 MDI 窗体。
- **添加模块**
在工程中加入一个模块。
- **添加类模块**
在工程中加入一个类模块。
- **添加用户控件**
在工程中加入一个用户控件。
- **添加属性页**
显示[添加属性页]对话框,在工程中加入一个新的或现有的属性页,可以用属性页向导来建立新的属性页。
- **添加用户文档**
显示添加用户文档对话框,在工程中加入一个新的或现有的用户文档。

- **添加 DHTML Page**
在工程中添加 HTML 页。
- **添加 Data Report**
向当前工程中添加数据报表。
- **添加 WebClass**
在工程中添加 WebClass。
- **添加 Microsoft UserConnection**
在工程中添加 Microsoft UserConnection。
- **添加 ActiveX 设计器**
显示可用的 ActiveX 设计器,从中选取一种,可加入到当前的工程中。
- **添加文件**
在工程中加入一个文件。
- **移除 Form1**
在工程中移除选择的窗体对象。
- **引用**
显示[引用]对话框,用于在工程中加入一个对象库或类库。
- **部件**
显示[部件]对话框,用于往工具箱中加入控件,设计器和对象。
- **工程属性**
用于查看和设置选定工程的属性。

五、【格式】菜单

【格式菜单】含有 8 个菜单项:

- **对齐**
在一组选中的对象中,根据最后选择的对象排列它们的位置。
- **统一尺寸**
使选择的多个对象具有与核对象同样的大小。
- **按网格调整大小**
按照窗体上最近的网格线调整对象的高度和宽度,使用时要选择好被操作对象。
- **水平间距**
改变对象之间的水平距离。
- **垂直间距**
改变对象之间的垂直距离。
- **在窗体中居中对齐**
调整所选择的对象到窗体中央并对齐排列。
- **顺序**
调整窗体中对象的顺序。
- **锁定控件**
把窗体上所有控件锁定到当前位置,以免在设计中不小心移动它们。

六、【调试】菜单

【调试】菜单包含 11 个菜单项:

- **逐语句**

用于调试程序,每次只执行一条语句,若调用一个过程,也是逐条执行过程中的语句。

- **逐过程**

类似于逐语句菜单项,区别在于它把调用过程当作一条语句执行,不进入被调用过程内部。

- **跳出**

执行本过程中所有剩下的语句,然后从该过程中跳出。

- **运行到光标处**

程序运行到光标所在位置。

- **添加监视**

在设计时或中断模式下,显示[添加监视]对话框,输入监视表达式。监视表达式可以是任何合法的 Basic 表达式。

- **编辑监视**

显示[编辑监视]对话框,用于编辑或删除监视表达式。

- **快速监视**

显示[快速监视]对话框和监视表达式的当前值,可以用它查看变量、属性或其他表达式的值,在代码窗口或立即窗口中选择表达式,然后才能执行此命令。

- **切换断点**

在当前设置或取消断点。注意:在不可执行语句(如注释、空行等)上不能设置断点。

- **清除所有断点**

清除工程中所有已设置的断点。

- **设置下一条语句**

设置下一条要执行的语句先选择要执行的语句,再选择该命令,或把当前行的执行指示器拖动到要执行的代码行,可以让程序在执行完本行后执行指定的语句。

- **显示下一条语句**

加亮下一条要执行的语句,该命令把光标停留在下一条要执行的语句中。

七、【运行】菜单

【运行】菜单包含有 5 个菜单项:

- **启动**

运行工程管理器启动工程,选择此命令后,系统先自动关闭设计时显示的所有窗体,变量初始化,并装入启动窗体。注意:在设计时不能运行控件工程,对于 DLL 工程,可以用另一个可执行的工程创建一个实例来运行它。

- **全编译执行**

对工程进行完全编译,然后再运行它。

- **中断**

当程度正在运行时中断它,进入中断模式。

- **结束**

结束运行程序,返回设计状态。运行时和中断模式下都有效。

- **重新启动**

在中断模式下使用,重新开始执行应用程序。

八、【工具】菜单

【工具】菜单包含 5 个菜单项：

- **添加过程**

在活动的模块中插入一个新的子程序(Sub) 属性(Property)、函数(Function)或事件(Event)过程。

- **过程属性**

打开过程属性对话框,设置与对象的属性和方法有关的性质。

- **菜单编辑器**

显示菜单编辑器对话框。

- **选项**

显示选项对话框,设置 VB 编程环境。

- **发布**

用于发布资源文件和建立输出。

九、【外接程序】菜单

【外接程序】菜单包括 2 个菜单项：

- **可视化数据管理器**

打开可视化数据管理器,可以访问或操作数据。

- **外接程序管理器**

显示[加载外接程序管理器]对话框,用于装入或卸载 VB 的外接程序。

十、【窗口】菜单

【窗口】菜单包含 6 个菜单项：

- **拆分**

用水平线分隔代码窗口或取消这种分隔。

- **水平平铺**

在 MDI 模式下使对象、代码窗口和对象浏览器上下排列。

- **垂直平铺**

在 MDI 模式下使对象、代码窗口和对象浏览器左右排列。

- **层叠**

在 MDI 模式下使对象、代码窗口和对象浏览器层叠排列。

- **排列图标**

在窗口左下角排列最小化窗口的图标。

- **窗口列表**

显示所有打开的窗口。

十一、【帮助】菜单

【帮助】菜单包含 6 个菜单项：

- **内容**

运行帮助 MSDN,显示帮助的目录。

- **索引**

运行帮助 MSDN,显示帮助的索引。

- **搜索**

运行帮助 MSDN,打开帮助搜索。

- **技术支持**

运行帮助系统 MSDN,显示 Microsoft 产品的技术支持信息。

- **Web 上的 Microsoft**

显示一个包含 Internet 网址的菜单。在 VB 中可以直接访问这些网点,前提是计算机已接入 Internet。

- **关于 Microsoft VB**

显示关于对话框,列出有关 VB6.0 产品信息。

附录 C 编码约定

使用统一编码约定集可使应用程序的结构和编码风格标准化,以便于阅读和理解这段编码。好的编码约定可使源代码严谨、可读性强且意义清楚,与其他语言约定相一致,并且尽可能的直观。

附录中提出的约定是简洁和建议性的,这些约定中没列出每一个可能的对象或控件,也没列出每种有用的信息注释。根据工程及机构的特殊要求,可以扩充这些准则,以包含附加的元素。

一、对象命名约定

在代码中应该用一致的前缀来命名对象,使人们容易识别对象的类型。下面列出了 VB 支持的一些推荐使用的对象约定。

1. 推荐使用的控件前缀

控件类型	前 缀	示 例
3D Panel	pnl	pnlGroup
Animated button	pni	aniMailBox
Checkbox	chk	chkReadOnly
Combobox	cbo	cboEnglish
Command button	cmd	cmdExit
Common dialog	dlg	dlgFileOpen
Communications	com	comFax
Control	ctr	ctrCurrent
Data control	dat	datBiblio
Data-bound combo box	dbcbo	dbcboLanguage
Data-bound grid	dbgrd	dbgrdQueryResult
Data-bound list box	dblst	dblstJobType
Directory list box	dir	dirSource
Drive list box	drv	drvTarget
File list box	fil	filSource
Form	frm	frmEntry
Frame	fra	fraLanguage
Gauge	gau	gauStatus
Graph	gra	graRevenue
Grid	grd	grdPrices
Horizontal scroll bar	hsb	hsbVolume
Image	img	imgIcon

续表

控件类型	前 缀	示 例
Key status	key	keyCaps
Label	lbl	lblHelpMessage
Line	lin	linVertical
List box	lst	lstPolicyCodes
MAPI message	mpm	mpmSentMessage
MAPI session	mps	mpsSession
MCI	mci	mciVideo
MDI child form	mdi	mdiNote
Menu	mnu	mnuFileOpen
MS Flex grid	msg	msgClients
MS Tab	mst	mstFirst
OLE	ole	oleWorksheet
Outline	out	outOrgChart
Pen BEdit	bed	bedFirstName
Pen HEdit	hed	hedSignature
Pen ink	ink	inkMap
Picture	pic	picVGA
Picture clip	clp	clpToolbar
Report	rpt	rptQtr1 Earnings
Shape	shp	shpCircle
Spin	spn	spnPages
Text box	txt	txtLastName
Timer	tmr	tmrAlarm
UpDown	upd	updDirection
Vertical scroll bar	vsb	vsbRate
Slider	sld	sldScale
ImageList	ils	ilsAllIcons
TreeView	tre	treOrganization
Toolbar	tlb	tlbActions
TabStrip	tab	tabOptions
StatusBar	sta	staDateTime
List View	lvw	lvwHeadings
ProgressBar	prg	prgLoadFile
RichTextBox	rtf	rtfReport

2. 推荐使用的数据访问对象 (DAO) 的前缀

数据库对象	前缀	例子
Container	con	conReports
Database	db	dbAccounts
DBEngine	dbe	dbeJet
Document	doc	docSalesReport
Field	fld	fldAddress
Group	grp	grpFinance
Index	idx	idxAge
Parameter	prm	prmJobCode
QueryDef	qry	qrySalesByRegion
Recordset	rec	recForecast
Relation	rel	relEmployeeDept
TableDef	tbd	tbdCustomers
User	usr	usrNew
Workspace	wsp	wspMine

3. 推荐使用的菜单前缀

应用程序频繁使用许多菜单控件,对于这些控件具备一组唯一的命名约定很实用。除了最前面“mnu”标记以外,菜单控件的前缀应该被扩展;对每一级嵌套增加一个附加前缀,将最终的菜单的标题放在名称字符串的最后。下表列出了一些例子:

菜单标题	菜单名称
File Open	mnuFileOpen
File Send Email	mnuFileSendEmail
File Send Fax	mnuFileSendFax
Format Character	mnuFormatCharacter
Help Contents	mnuHelpContents

当使用这种命名约定时,一个特定的菜单组的所有成员一个接一个地列在 VB 的【属性】窗口中,而且,菜单控件的名字清楚地表示出它们所属的菜单项。

提示:对于上面没有列出的控件,应该用唯一的由两个或三个字符组成的前缀使它们标准化,以保持一致性。例如:对于派生的或修改的控件,像上述那样扩展其前缀,使得在真正使用了哪一个控件的问题上避免混淆。对于第三方控件,应该把制造商的小写缩写名附加到前缀中。例如,从 VB Professional 3D 框架中创建的一个控件实例可以用 fra3d 这样的前缀,以避免混淆所使用的控件。

二、常量和变量的命名约定

除了对象之外,常量和变量也需要良好格式的命名约定。本节列出了 VB 支持的常量和变量的推荐约定。

变量应该总是被定义在尽可能小的范围内。全局 (Public) 变量可以导致极其复杂的状态机构,并且使一个应用程序的逻辑非常难于理解。全局变量也使代码的重用和维护更加困难。

1. 变量范围编码约定

范围	前缀	例子
全局	g	gstrUserName
模块级	m	mblnCalcInProgress
本地到过程	无	dblVelocity

2. 常量

常量名的主体是大小写混合的,每个单词的首字母大写。尽管标准 VB 常量不包含数据类型和范围信息,但是象 i,s,g 和 m 这样的前缀对于理解一个常量的值和范围还是很有用的。对于常量名,应遵循与变量相同的规则。例如:

mintUserListMax '对用户列表的最大限制 (整数值,本地到模块)
 gstrNewLine '新行字符 (字符串,应用程序全局使用)

3. 变量

应该给变量加前缀来指明它们的数据类型,而且前缀可以被扩展,用来指明变量范围,特别是对大型程序。

数据类型	前缀	例子
Boolean	bln	blnFound
Byte	byt	bytRasterData
Currency	cur	curRevenue
Double	dbl	dblTolerance
Integer	int	intQuantity
Long	lng	lngDistance
Object	obj	objCurrent
Single	sng	sngAverage
String	str	strFName
Variant	vnt	vntCheckSum
Date	dat	datMyBirthday

4. 描述变量和过程名

变量或过程名的主体应该使用大小写混合形式,并且应该足够长以描述它的作用。而且,函数名应该以一个动词起首,如 InitNameArray 或 CloseDialog。

对于频繁使用的或长的项,推荐使用标准缩略语以使名称的长度合理化。一般来说,超过 32 个字符的变量名在 VGA 显示器上读起来就困难了。

当使用缩略语时,要确保它们在整个应用程序中的一致性。在一个工程中,如果一会儿使用 Cnt,一会儿使用 Count,将导致不必要的混淆。

无忧书库书籍版权申明

无忧书库提供的所有书籍、资料版权归原作者和出版商所有。

本站提供下载的书籍仅供个人学习、参考之用，任何集体、个人不得用于商业用途。

请您在下载书籍 24 小时之后删除书籍，购买正版书籍！

本站书籍如有侵犯您的版权，请与我们联系，我们将尽快删除。联系信箱：
pcbook@51soft.com。

无忧书库
<http://pcbook.51soft.com>
2000 年 4 月 13 日