

软件开发视频大讲堂  
图书+视频光盘+互动答疑

# Visual Basic

## 从入门到精通

刘彬彬 高春艳 孙秀梅 等编著



**25** DVD-ROM  
小时语音视频讲解

DVD语音视频教学光盘

**25**小时教学视频录像，全程语音讲解  
本书实例源程序、相关素材

本书特色

基础知识→核心技术→高级应用→项目实战

**235**个应用实例，**44**个典型应用，**1**个项目案例  
内容极为详尽，实例典型丰富

全程跟踪服务

服务热线：400-675-1066 QQ：100310063  
答疑网站：[www.mingribook.com](http://www.mingribook.com)



清华大学出版社

软件开发视频大讲堂

# Visual Basic 从入门到精通

## (第2版)

刘彬彬 安 剑 等编著

清华大学出版社  
北 京



## 内 容 简 介

本书从初学者的角度出发,以通俗易懂的语言、丰富多彩的实例,详细介绍了使用 Visual Basic 进行程序开发需要掌握的知识。全书共分 22 章,包括初识 Visual Basic 6.0,VB 语言基础,算法和程序控制结构,数组的声明和应用,过程的创建和使用,内置函数与 API 函数,窗体和系统对象,标准模块和类模块,常用标准控件,菜单、工具栏和状态栏,对话框,常用 ActiveX 控件,鼠标键盘处理,程序调试和错误处理,文件系统编程,图形图像技术,多媒体技术,SQL 应用,数据库开发技术,数据库控件,网络编程技术以及企业进销存管理系统。书中所有知识都结合具体实例进行介绍,涉及的程序代码给出了详细的注释,可以使读者轻松领会 Visual Basic 程序开发的精髓,快速提高开发技能。

本书列举了大量的中小型实例、综合实例和部分项目案例;所附 DVD 光盘内容有同步视频讲解、实例源程序、“实践与练习”答案等;本书的服务网站提供了模块库、案例库、题库、素材库、答疑服务。

本书内容详尽、实例丰富,非常适合作为编程初学者的学习用书,也适合作为开发人员的查阅、参考资料。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

### 图书在版编目(CIP)数据

Visual Basic 从入门到精通/刘彬彬,安剑等编著. —2 版. —北京:清华大学出版社,2010.7  
(软件开发视频大讲堂)

ISBN 978-7-302-22661-1

I. ①V… II. ①刘… ②安… III. ①BASIC 语言-程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字(2010)第 079719 号

责任编辑:刘利民

版式设计:杨 洋

责任校对:姜 彦

责任印制:何 芊

出版发行:清华大学出版社

地 址:北京清华大学学研大厦 A 座

<http://www.tup.com.cn>

邮 编:100084

社 总 机:010-62770175

邮 购:010-62786544

投稿与读者服务:010-62776969, [c-service@tup.tsinghua.edu.cn](mailto:c-service@tup.tsinghua.edu.cn)

质 量 反 馈:010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

印 刷 者:北京鑫丰华彩印有限公司

装 订 者:三河市新茂装订有限公司

经 销:全国新华书店

开 本:203×260 印 张:36.5 字 数:976 千字

(附 DVD 视频光盘 1 张)

版 次:2010 年 7 月第 2 版

印 次:2010 年 7 月第 1 次印刷

印 数:1~5000

定 价:69.80 元

产品编号:035721-01

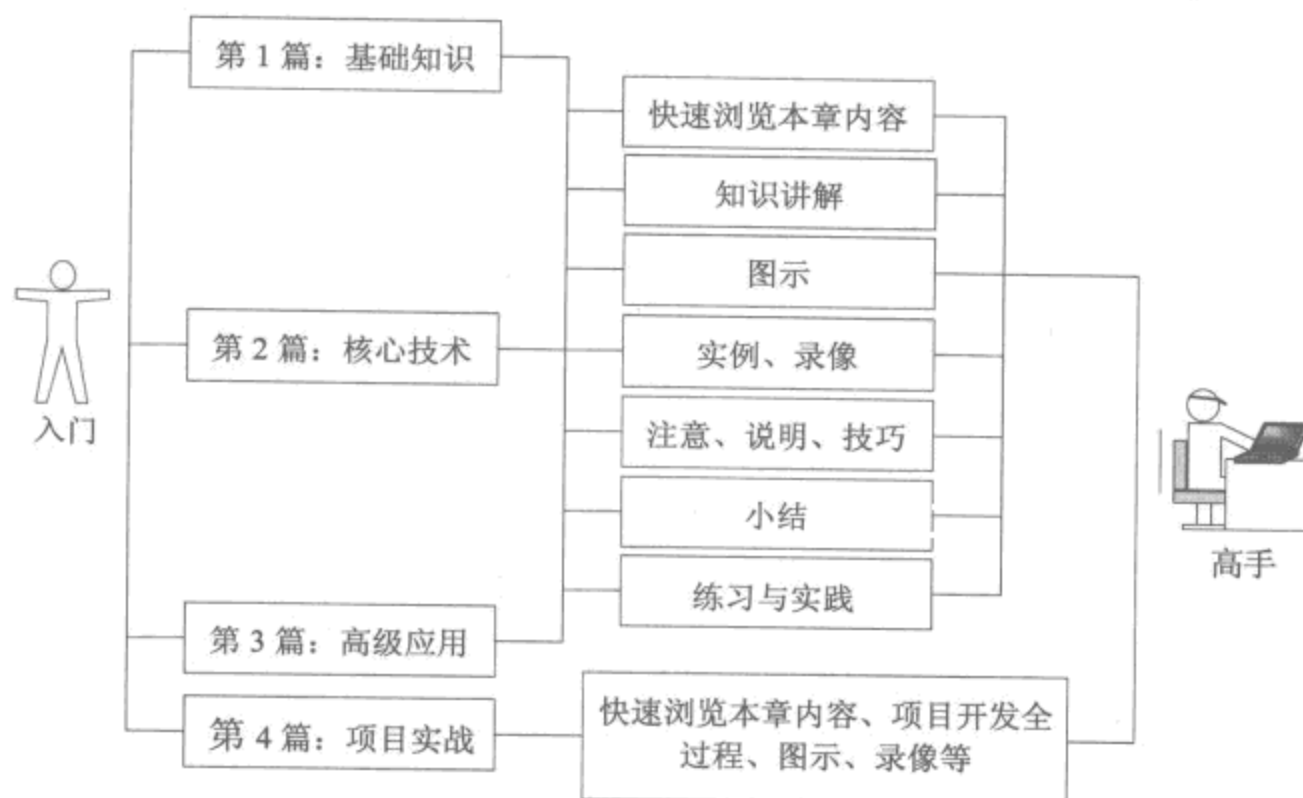
# 前言

## Preface

Visual Basic 6.0 是 Microsoft 公司推出的基于 Windows 环境的一种面向对象的可视化编程环境，自面世以来便凭借其易学易用、功能强大的特点备受用户的青睐。其强大的可视化用户界面设计，让程序员从复杂的界面设计中解脱出来，使编程成为一种享受。Visual Basic 不仅可以开发数据库管理系统，而且可以开发集声音、动画、视频为一体的多媒体应用程序和网络应用程序，这使得 Visual Basic 6.0 成为应用最广泛的编程语言之一。

## 本书内容

本书提供了从入门到编程高手所必备的各类知识，共分 4 篇，大体结构如下图所示。



**第1篇：基础知识。**本篇通过初识 Visual Basic 6.0、VB 语言基础、算法和程序控制结构、数组的声明和应用、过程的创建和使用以及内置函数与 API 函数并结合大量的图示、举例、录像等内容的介绍，使读者快速掌握 Visual Basic 语言基础知识，为以后的编程奠定坚实的基础。

**第2篇：核心技术。**本篇介绍了窗体和系统对象，标准模块和类模块，常用标准控件，菜单、工具栏和状态栏，对话框，常用 ActiveX 控件，鼠标键盘处理，程序调试和错误处理以及文件系统编程等。学习完这一部分，将能够开发一些小型应用程序。

**第3篇：高级应用。**本篇介绍了图形图像技术、多媒体技术、SQL 应用、数据库开发技术、数据



库控件以及网络编程技术等。学习完这一部分，将能够开发数据库应用程序、多媒体程序和网络程序等。

**第4篇：项目实战。**本篇通过一个大型、完整的企业进销存管理系统，运用软件工程的设计思想，演示了如何进行软件项目的实践开发。书中按照“编写项目计划书→系统设计→数据库设计→创建项目→实现项目→运行项目→项目打包部署→解决开发常见问题”的流程进行介绍，带领读者一步一步亲身体验开发项目的全过程。

## 本书特点

- **由浅入深，循序渐进：**本书以初中级程序员为对象，先从 VB 语言基础学起，再学习 VB 的核心技术，然后学习 VB 的高级应用，最后学习开发一个完整项目。讲解过程中步骤详尽，版式新颖，在操作的内容图片上以“①②③……”的“编号+内容”的方式进行标注，让读者在阅读时一目了然，从而快速把握书中内容。
- **语音视频，讲解详尽：**书中每一章节均提供声图并茂的语音视频教学录像，读者可以根据书中提供的录像位置，在光盘中找到相应的文件。这些录像能够引导初学者快速入门，感受编程的快乐和成就感，增强进一步学习的信心，从而快速成为编程高手。
- **实例典型，轻松易学：**通过例子学习是最好的学习方式，本书通过“一个知识点、一个例子、一个结果、一段评析、一个综合应用”的模式，透彻详尽地讲述了实际开发中所需的各类知识。另外，为了便于读者阅读程序代码、快速学习编程技能，书中几乎为每行代码都提供了注释。
- **精彩栏目，贴心提醒：**根据需要，本书在各章节中使用了很多“注意”、“说明”、“技巧”等小栏目，让读者可以在学习过程中更轻松地了解相关知识点及概念，并轻松地掌握个别技术的应用技巧。
- **应用实践，随时练习：**书中几乎每章都提供了“练习与实践”，读者能够通过对问题的解答重新回顾、熟悉所学的知识，举一反三，为进一步学习做好充分的准备。

## 读者对象

- |   |   |
|---|---|
| <input checked="" type="checkbox"/> 初学编程的自学者    | <input checked="" type="checkbox"/> 编程爱好者         |
| <input checked="" type="checkbox"/> 大中专院校的老师和学生 | <input checked="" type="checkbox"/> 相关培训机构的老师和学员  |
| <input checked="" type="checkbox"/> 毕业设计的学生     | <input checked="" type="checkbox"/> 初中级程序开发人员     |
| <input checked="" type="checkbox"/> 程序测试及维护人员   | <input checked="" type="checkbox"/> 参加实习的“菜鸟”级程序员 |

## 读者服务

为了方便读者，本书提供了学习答疑网站：[www.mingribook.com](http://www.mingribook.com)。有关本书的问题读者均可在网上留言，我们力求在 24 小时内回复（节假日除外）。

## 致读者

本书由 Visual Basic 程序开发团队组织编写，主要编写人员有刘彬彬、安剑、高春艳、孙秀梅、安剑、王茜、王永生、刘欣、刘玲玲、刘书娟、梁晓岚、顾彦玲、黄锐、杨丽、孙明娇、寇长梅、张鹏斌、董大永、张艳、郭佳博、乔敏、刘中华、陈紫宏、张领、苗春义、李严、李贺、张世辉、张金辉、王敬杰、高飞、郭铁、贯伟红、陈丹丹、房大伟、王小科、吕双、梁冰、苏宇、王殊宇等。

在编写本书的过程中，我们以科学、严谨的态度，力求精益求精，但错误和疏漏之处在所难免，敬请广大读者批评指正。我们的服务邮箱是 [tmoonbook@sina.com](mailto:tmoonbook@sina.com)、[th\\_press@263.net](mailto:th_press@263.net)，读者在阅读本书时，如果发现错误或遇到问题，可以发送电子邮件及时与我们联系，我们会尽快给予答复。

感谢您购买本书，希望本书能成为您编程路上的领航者。

“零门槛”编程，一切皆有可能。

祝读书快乐！

编 者






# 目 录

## Contents

### 第 1 篇 基础知识


#### 第 1 章 初识 Visual Basic 6.0..... 3

 视频讲解: 1 小时 26 分钟

1.1 Visual Basic 简介.....	4
1.1.1 Visual Basic 的发展.....	4
1.1.2 Visual Basic 6.0 的特点.....	4
1.2 如何学好 VB.....	5
1.2.1 VB 可以做什么.....	5
1.2.2 学习 VB 的几点建议.....	6
1.3 VB 6.0 的安装与管理.....	6
1.3.1 VB 6.0 的运行环境.....	6
1.3.2 VB 6.0+SP6 的安装.....	7
1.3.3 VB 6.0 的更改或删除.....	9
1.4 VB 6.0 的启动.....	10
1.4.1 通过“开始”菜单启动.....	10
1.4.2 通过快捷方式启动.....	10
1.5 VB 6.0 的集成开发环境.....	12
1.5.1 集成开发环境简介.....	12
1.5.2 菜单栏.....	13
1.5.3 工具栏.....	15
1.5.4 工具箱.....	16
1.5.5 工程资源管理器.....	18
1.5.6 属性窗口.....	19
1.5.7 窗体布局窗口.....	20
1.5.8 窗体设计器.....	21
1.5.9 代码编辑窗口.....	21
1.6 定制开发环境.....	22
1.6.1 设置在编辑器中要求变量声明.....	22
1.6.2 设置网格大小和不对齐到网格.....	23

1.6.3 设置启动时保存.....	23
1.6.4 定制工具栏.....	24
1.6.5 为代码编辑器设置鼠标滚动.....	25
1.7 VB 6.0 的帮助系统.....	26
1.7.1 MSDN Library 的安装与使用.....	26
1.7.2 利用附带的实例源程序学习编程.....	28
1.7.3 使用 VB 的帮助菜单.....	29
1.8 创建第一个 VB 程序.....	29
1.8.1 创建工程文件.....	29
1.8.2 设计界面.....	30
1.8.3 编写代码.....	30
1.8.4 调试运行.....	31
1.8.5 保存工程.....	31
1.8.6 编译程序.....	32
1.9 小结.....	32

#### 第 2 章 VB 语言基础..... 33

 视频讲解: 29 分钟

2.1 关键字和标识符.....	34
2.2 数据类型.....	34
2.2.1 基本数据类型.....	35
2.2.2 记录类型.....	38
2.2.3 枚举类型.....	40
2.3 变量.....	40
2.3.1 什么是变量.....	41
2.3.2 变量的命名.....	41
2.3.3 变量的声明.....	42
2.3.4 变量的作用域.....	44
2.3.5 静态变量.....	45

2.3.6 变量同名问题的处理.....	46	3.4.3 Do...Loop 循环语句.....	83
2.3.7 变量的生命周期.....	47	3.4.4 嵌套循环.....	87
2.4 常量.....	47	3.4.5 选择结构与循环结构的嵌套.....	88
2.4.1 常量的声明.....	47	3.5 其他辅助控制语句.....	89
2.4.2 局部常量和全局常量.....	48	3.5.1 跳转语句 GoTo.....	89
2.5 运算符和表达式.....	48	3.5.2 复用语句 With...End With.....	90
2.5.1 运算符.....	48	3.5.3 退出语句 Exit.....	90
2.5.2 表达式.....	50	3.5.4 结束语句 End.....	91
2.5.3 运算符的优先级.....	51	3.6 小结.....	92
2.6 代码编写规则.....	51	3.7 练习与实践.....	92
2.6.1 对象命名规则.....	51		
2.6.2 代码书写规则.....	53	<b>第4章 数组的声明和应用.....93</b>	
2.6.3 处理关键字冲突.....	54	<b>视频讲解: 26 分钟</b>	
2.6.4 代码注释规则.....	54	4.1 数组的概述.....	94
2.7 小结.....	56	4.1.1 数组的概念.....	94
2.8 练习与实践.....	56	4.1.2 数组与简单变量的区别.....	95
		4.2 数组的分类.....	95
<b>第3章 算法和程序控制结构.....57</b>		4.2.1 静态数组.....	95
<b>视频讲解: 59 分钟</b>		4.2.2 动态数组.....	97
3.1 算法.....	58	4.2.3 一维数组.....	99
3.1.1 什么是算法.....	58	4.2.4 数组中的数组.....	100
3.1.2 算法的特性.....	59	4.2.5 二维数组及多维数组.....	101
3.1.3 算法的描述方法.....	59	4.3 数组的基本操作.....	102
3.1.4 构成算法的基本控制结构.....	61	4.3.1 数组元素的输入.....	103
3.2 顺序结构.....	64	4.3.2 数组元素的输出.....	104
3.2.1 赋值语句.....	64	4.3.3 数组元素的插入.....	104
3.2.2 数据的输入.....	66	4.3.4 数组元素的删除.....	105
3.2.3 数据的输出.....	67	4.3.5 数组元素的查找.....	105
3.3 选择结构.....	69	4.3.6 数组元素的排序.....	106
3.3.1 单分支 If...Then 语句.....	69	4.4 记录数组.....	108
3.3.2 双分支 If...Then...Else 语句.....	71	4.4.1 记录数组的概念.....	108
3.3.3 If 语句的嵌套.....	72	4.4.2 记录数组的使用.....	109
3.3.4 多分支 If...Then...ElseIf 语句.....	75	4.5 控件数组.....	109
3.3.5 Select Case 语句.....	77	4.5.1 控件数组的概念.....	109
3.3.6 IIf 函数.....	78	4.5.2 创建控件数组.....	110
3.4 循环结构.....	79	4.5.3 使用控件数组.....	110
3.4.1 For...Next 循环语句.....	79	4.6 数组相关函数及语句.....	113
3.4.2 For Each...Next 循环语句.....	82	4.6.1 Array 函数.....	113



4.6.2 UBound 函数和 LBound 函数 .....	114	6.1.3 Sgn 函数 (返回符号) .....	139
4.6.3 Split 函数 .....	114	6.1.4 Sqr 函数 (平方根) .....	140
4.6.4 Option Base 语句 .....	115	6.2 字符串函数 .....	140
4.7 小结 .....	116	6.2.1 Len 函数 .....	140
4.8 练习与实践 .....	116	6.2.2 Left 和 Right 函数 .....	141
<b>第 5 章 过程的创建和使用 .....</b>	<b>117</b>	6.2.3 Mid 函数 .....	142
 <b>视频讲解: 53 分钟</b>		6.2.4 Trim、RTrim、LTrim 函数 (去空格) .....	142
5.1 认识过程 .....	118	6.3 类型转换函数 .....	143
5.2 事件过程 .....	118	6.3.1 Asc 函数 (转换为 ASCII) .....	143
5.2.1 建立事件过程 .....	119	6.3.2 Chr 函数 (转换为字符) .....	143
5.2.2 调用事件过程 .....	119	6.3.3 Val 函数 (转换为数值型) .....	144
5.3 子过程 (Sub 过程) .....	120	6.3.4 Str 函数 (转换为字符型) .....	144
5.3.1 建立子过程 .....	120	6.4 判断函数 .....	145
5.3.2 调用子过程 .....	122	6.4.1 IsNull 函数 .....	145
5.3.3 调用其他模块中的子过程 .....	123	6.4.2 IsNumeric 函数 .....	146
5.4 函数过程 (Function 过程) .....	124	6.4.3 IsArray 函数 .....	146
5.4.1 建立函数过程 .....	124	6.5 日期和时间函数 .....	147
5.4.2 调用函数过程 .....	124	6.5.1 Date 函数、Now 函数、Time 函数 .....	147
5.4.3 函数过程与子过程的区别 .....	125	6.5.2 Timer 函数 .....	147
5.5 参数的传递 .....	125	6.5.3 Weekday 函数 .....	148
5.5.1 认识参数 .....	125	6.5.4 Year、Month、Day 函数 (年、月、日) .....	150
5.5.2 参数按值和按地址传递 .....	127	6.5.5 Hour、Minute、Second 函数 (时、分、秒) .....	150
5.5.3 数组参数 .....	128	6.6 随机函数 .....	151
5.5.4 对象参数 .....	129	6.6.1 Randomize 函数 .....	151
5.6 嵌套过程 .....	130	6.6.2 Rnd 函数 .....	152
5.7 递归过程 .....	132	6.7 格式化函数 .....	153
5.8 属性过程 (Property 过程) .....	133	6.8 API 函数 .....	155
5.8.1 使用属性过程建立类的属性 .....	134	6.8.1 API 的概念 .....	155
5.8.2 使用类属性 .....	135	6.8.2 API 的相关概念 .....	157
5.8.3 只读属性和对象属性 .....	136	6.9 API 浏览器 .....	158
5.9 小结 .....	136	6.9.1 启动 API 浏览器 .....	158
5.10 练习与实践 .....	136	6.9.2 API 浏览器的加载 .....	159
<b>第 6 章 内置函数与 API 函数</b>		6.9.3 API 浏览器的使用 .....	160
 <b>视频讲解: 53 分钟</b>		6.10 API 的使用 .....	162
6.1 数学函数 .....	138	6.10.1 API 函数的声明 .....	162
6.1.1 Abs 函数 (求绝对值) .....	138		
6.1.2 Exp 函数 (e 的 n 次方) .....	138		

6.10.2 API 常数与类型 .....	163	6.12 小结 .....	164
6.11 API 函数的调用 .....	164	6.13 练习与实践 .....	165

## 第2篇 核心技术

### 第7章 窗体和系统对象 ..... 169

 视频讲解: 1 小时 9 分钟

7.1 窗体的概述 .....	170
7.1.1 窗体的结构 .....	170
7.1.2 模式窗体和无模式窗体 .....	170
7.1.3 SDI 窗体和 MDI 窗体 .....	171
7.1.4 添加和移除窗体 .....	173
7.1.5 加载 (Load) 与卸载 (Unload) 窗体 .....	174
7.2 窗体的属性 .....	175
7.2.1 名称 (Name 属性) .....	176
7.2.2 标题 (Caption 属性) .....	176
7.2.3 图标 (Icon 属性) .....	177
7.2.4 背景 (Picture 属性) .....	178
7.2.5 边框样式 (BorderStyle 属性) .....	179
7.2.6 显示状态 (WindowState 属性) .....	180
7.2.7 显示位置 (StartPosition 属性) .....	181
7.3 窗体的方法 .....	182
7.3.1 显示窗体 (Show 方法) .....	182
7.3.2 隐藏窗体 (Hide 方法) .....	183
7.3.3 移动窗体 (Move 方法) .....	183
7.4 窗体的事件 .....	184
7.4.1 单击和双击 (Click / DblClick 事件) .....	184
7.4.2 载入和卸载 (Load / QueryUnload / Unload 事件) .....	185
7.4.3 活动性 (Activate / Deactivate 事件) .....	187
7.4.4 初始化 (Initialize 事件) .....	188
7.4.5 调整大小 (Resize 事件) .....	189
7.4.6 重绘 (Paint 事件) .....	190
7.4.7 焦点事件 (GotFocus / LostFocus 事件) .....	190
7.5 窗体事件的生命周期 .....	191

7.5.1 窗体启动过程 .....	191
7.5.2 窗体运行过程 .....	192
7.5.3 窗体关闭过程 .....	192
7.6 MDI 窗体 .....	194
7.6.1 MDI 窗体概述 .....	194
7.6.2 MDI 窗体的添加和移除 .....	195
7.6.3 MDI 子窗体 (MDIChild 属性) .....	197
7.6.4 MDI 程序的特点 .....	198
7.6.5 MDI 主窗体的设计 .....	199
7.7 系统对象 .....	200
7.7.1 应用程序对象 (APP 对象) .....	200
7.7.2 屏幕对象 (Screen 对象) .....	202
7.7.3 剪贴板对象 (Clipboard 对象) .....	203
7.7.4 调试对象 (Debug 对象) .....	203
7.8 小结 .....	204
7.9 练习与实践 .....	204

### 第8章 标准模块和类模块 ..... 205

 视频讲解: 12 分钟

8.1 标准模块 .....	206
8.1.1 标准模块概述 .....	206
8.1.2 添加标准模块 .....	206
8.2 类模块 .....	207
8.2.1 类模块的概述 .....	207
8.2.2 添加类模块 .....	208
8.3 标准模块和类模块的区别 .....	209
8.4 小结 .....	209
8.5 练习与实践 .....	209

### 第9章 常用标准控件 ..... 211


 视频讲解: 1 小时 29 分钟

9.1 控件概述 .....	212
----------------	-----






9.1.1 控件的作用.....	212	10.2.6 设置菜单无效.....	248
9.1.2 控件的属性、方法和事件.....	212	10.2.7 为菜单事件添加代码.....	249
9.1.3 控件的分类.....	213	10.3 弹出式菜单.....	249
9.2 控件的相关操作.....	214	10.3.1 弹出式菜单概述.....	249
9.2.1 向窗体上添加控件.....	214	10.3.2 PopupMenu 方法.....	249
9.2.2 调整控件的大小.....	214	10.3.3 弹出式菜单的设计和调用.....	250
9.2.3 复制与删除控件.....	214	10.4 菜单数组.....	251
9.2.4 使用窗体编辑器调整控件布局.....	215	10.4.1 创建菜单数组.....	251
9.2.5 锁定控件.....	216	10.4.2 为菜单数组编写代码.....	252
9.3 标签和文本框.....	217	10.5 工具栏设计.....	253
9.3.1 标签 (Label 控件).....	217	10.5.1 工具栏概述.....	253
9.3.2 文本框 (TextBox 控件).....	218	10.5.2 利用 Toolbar 控件创建最简工具栏.....	253
9.4 命令按钮.....	222	10.5.3 为工具栏按钮添加图片.....	254
9.4.1 命令按钮的属性.....	222	10.5.4 为工具栏按钮设置分组.....	255
9.4.2 命令按钮的事件.....	223	10.5.5 为工具栏添加下拉菜单.....	256
9.5 单选按钮、复选框及框架.....	224	10.5.6 为工具栏按钮添加事件处理代码.....	257
9.5.1 单选按钮 (OptionButton 控件).....	224	10.6 状态栏设计.....	258
9.5.2 复选框 (CheckBox 控件).....	226	10.6.1 状态栏概述.....	258
9.5.3 框架 (Frame 控件).....	226	10.6.2 在状态栏中显示日期、时间.....	258
9.6 列表框与组合框.....	228	10.6.3 在状态栏中显示操作员信息.....	259
9.6.1 列表框 (ListBox 控件).....	228	10.6.4 在状态栏中显示鼠标位置.....	260
9.6.2 组合框 (ComboBox 控件).....	232	10.7 小结.....	260
9.7 滚动条.....	234	10.8 练习与实践.....	261
9.8 Timer 控件.....	237	第 11 章 对话框.....	263
9.9 小结.....	239	视频讲解: 40 分钟	
9.10 练习与实践.....	239	11.1 输入对话框 (InputBox).....	264
第 10 章 菜单、工具栏和状态栏.....	241	11.2 消息对话框 (MsgBox).....	265
视频讲解: 1 小时		11.3 公用对话框.....	267
10.1 菜单概述.....	242	11.3.1 公用对话框概述.....	267
10.1.1 菜单的组成.....	242	11.3.2 “打开”对话框.....	269
10.1.2 菜单编辑器.....	243	11.3.3 “另存为”对话框.....	270
10.2 标准菜单.....	245	11.3.4 “颜色”对话框.....	272
10.2.1 创建最简菜单.....	245	11.3.5 “字体”对话框.....	272
10.2.2 设置菜单的快捷键和访问键.....	246	11.3.6 “打印”对话框.....	274
10.2.3 创建级联菜单.....	247	11.3.7 “帮助”对话框.....	275
10.2.4 创建复选菜单.....	247	11.4 小结.....	275
10.2.5 设置菜单分隔条.....	248	11.5 练习与实践.....	275

第 12 章 常用 ActiveX 控件 .....	277	12.7 日期/时间控件 (DateTimePicker) ...	303
 视频讲解: 1 小时 44 分钟		12.7.1 认识 DateTimePicker 控件 .....	304
12.1 ActiveX 控件的使用 .....	278	12.7.2 设置和返回日期 .....	304
12.1.1 添加 ActiveX 控件 .....	278	12.7.3 实时读取 DTPicker 控件中的日期 .....	305
12.1.2 删除 ActiveX 控件 .....	279	12.7.4 使用 CheckBox 属性来选择无日期 .....	305
12.1.3 注册 ActiveX 控件 .....	279	12.7.5 使用日期和时间的格式 .....	305
12.2 图像列表控件 (ImageList) .....	281	12.7.6 使用 DTPicker 控件计算日期或天数 .....	307
12.2.1 认识 ImageList 控件 .....	281	12.8 小结 .....	308
12.2.2 添加图像 .....	281	12.9 练习与实践 .....	308
12.2.3 与其他控件关联 .....	283		
12.2.4 创建组合图像 .....	285	第 13 章 鼠标键盘处理 .....	309
12.3 视图控件 (ListView) .....	285	 视频讲解: 30 分钟	
12.3.1 认识 ListView 控件 .....	286	13.1 鼠标指针的设置 .....	310
12.3.2 添加数据 .....	286	13.1.1 设置鼠标指针形状 .....	310
12.3.3 用“ListView 控件+数据表”创建 报表视图 .....	287	13.1.2 设置鼠标指针为指定的图片 .....	311
12.3.4 用 ListView 控件创建大图标视图 .....	289	13.1.3 设置鼠标指针为指定的动画 .....	311
12.4 树状控件 (TreeView) .....	290	13.2 鼠标事件的响应 .....	312
12.4.1 认识 TreeView 控件 .....	290	13.2.1 鼠标单击和双击 (Click 事件和 DbClick 事件) .....	313
12.4.2 添加数据 .....	290	13.2.2 鼠标按下和抬起 (MouseDown 事件和 MouseUp 事件) .....	313
12.4.3 删除指定节点数据 .....	292	13.2.3 鼠标移动 (MouseMove 事件) .....	314
12.4.4 节点展开与折叠 .....	292	13.2.4 鼠标拖动 (OLE 拖动操作) .....	315
12.4.5 用“TreeView 控件+数据表”创建 多级树状视图 .....	293	13.3 键盘事件的响应 .....	319
12.5 选项卡控件 (SSTab) .....	297	13.3.1 ASCII 码 .....	319
12.5.1 认识 SSTab 控件 .....	297	13.3.2 KeyDown 事件和 KeyUp 事件的使用 .....	319
12.5.2 设置选项卡数目和行数 .....	298	13.3.3 KeyPress 事件的使用 .....	322
12.5.3 在选项卡中添加控件 .....	298	13.4 小结 .....	323
12.5.4 运行时启用和停用选项卡 .....	298	13.5 练习与实践 .....	323
12.5.5 定制不同样式的选项卡 .....	299		
12.5.6 图形化选项卡 .....	300	第 14 章 程序调试和错误处理 .....	325
12.6 进度条 (ProgressBar) .....	301	 视频讲解: 16 分钟	
12.6.1 认识 ProgressBar 控件 .....	301	14.1 错误类型 .....	326
12.6.2 显示进展情况 .....	302	14.1.1 编译错误 .....	326
12.6.3 将 Max 属性设置为已知的界限 .....	302	14.1.2 运行错误 .....	327
12.6.4 隐藏 ProgressBar 控件 .....	302	14.1.3 逻辑错误 .....	327
12.6.5 用 ProgressBar 控件显示清空数据 的进度 .....	302	14.2 工作模式 .....	327
		14.2.1 设计模式 .....	328
		14.2.2 运行模式 .....	328




14.2.3 中断模式.....	328	15.3.2 改变目录或文件夹 (ChDir 语句) .....	348
14.3 调试工具及使用 .....	328	15.3.3 删除文件 (Kill 语句) .....	348
14.3.1 调试工具栏的使用.....	329	15.3.4 创建目录或文件夹 (MkDir 语句) .....	350
14.3.2 本地窗口的使用.....	329	15.3.5 复制文件 (FileCopy 语句) .....	350
14.3.3 立即窗口的使用.....	330	15.3.6 重命名 (Name 语句) .....	351
14.3.4 监视窗口的使用.....	330	15.3.7 设置文件属性 (SetAttr 语句) .....	352
14.3.5 插入断点和逐语句跟踪.....	332	15.4 常用的文件操作函数 .....	352
14.4 错误处理语句和对象 .....	332	15.4.1 获取路径 (CurDir 函数) .....	352
14.4.1 Err 对象.....	332	15.4.2 获取文件属性 (GetAttr 函数) .....	353
14.4.2 捕获错误 (On Error 语句) .....	333	15.4.3 获取文件创建或修改时间 (FileDateTime 函数) .....	354
14.4.3 退出错误处理 (Resume 语句) .....	334	15.4.4 返回文件长度 (FileLen 函数) .....	354
14.4.4 编写错误处理函数.....	335	15.4.5 测试文件结束状态 (EOF 函数) .....	354
14.5 小结 .....	335	15.4.6 获取打开文件的大小 (LOF 函数) .....	355
第 15 章 文件系统编程 .....	337	15.5 顺序文件 .....	355
 视频讲解: 1 小时 38 分钟		15.5.1 顺序文件的打开与关闭.....	356
15.1 文件的基本概念 .....	338	15.5.2 顺序文件的读取操作.....	357
15.1.1 文件的结构.....	338	15.5.3 顺序文件的写入操作.....	360
15.1.2 文件的分类.....	338	15.6 随机文件 .....	362
15.1.3 文件处理的一般步骤.....	339	15.6.1 随机文件的打开与关闭.....	362
15.2 文件系统控件 .....	339	15.6.2 读取随机文件.....	362
15.2.1 驱动器列表框 (DriveListBox 控件) .....	340	15.6.3 写入随机文件.....	363
15.2.2 目录列表框 (DirListBox 控件) .....	341	15.7 二进制文件 .....	365
15.2.3 文件列表框 (FileListBox 控件) .....	343	15.7.1 二进制文件的打开与关闭.....	365
15.2.4 文件系统控件的联动.....	346	15.7.2 二进制文件的读取与写入操作.....	365
15.3 文件的操作语句 .....	347	15.8 小结 .....	367
15.3.1 改变当前驱动器 (ChDrive 语句) .....	347	15.9 练习与实践 .....	367

## 第 3 篇 高级应用

第 16 章 图形图像技术 .....	371	16.1.4 RGB 函数 .....	373
 视频讲解: 25 分钟		16.2 坐标系统 .....	374
16.1 图形图像处理基础 .....	372	16.2.1 默认的坐标系统.....	374
16.1.1 系统颜色.....	372	16.2.2 自定义的坐标系统.....	374
16.1.2 在对象浏览器中查看系统颜色常量.....	372	16.3 图形外观效果 .....	376
16.1.3 QBColor 函数.....	373	16.3.1 绘图坐标.....	376
		16.3.2 图形位置和大小.....	376

16.3.3 图形的边框效果.....	377	17.4.2 ShockwaveFlash 控件的属性.....	404
16.3.4 绘制效果.....	378	17.4.3 ShockwaveFlash 控件的方法.....	405
16.3.5 前景色和背景色.....	378	17.4.4 ShockwaveFlash 控件的事件.....	405
16.3.6 填充效果.....	379	17.5 DirectX.....	406
16.4 绘图方法.....	379	17.5.1 下载和安装 DirectX.....	406
16.4.1 画点.....	379	17.5.2 在 VB 中使用 DirectX.....	407
16.4.2 画线.....	380	17.5.3 利用 DirectSound 编程实现实时混音.....	407
16.4.3 画圆.....	381	17.6 多媒体综合应用.....	410
16.4.4 清屏.....	382	17.6.1 CD 播放器.....	410
16.4.5 获取颜色值.....	383	17.6.2 VCD 播放器.....	411
16.4.6 绘制图形.....	383	17.6.3 多媒体演示程序.....	413
16.5 图像处理函数.....	384	17.7 小结.....	414
16.5.1 加载图像 (LoadPicture 函数).....	384	17.8 练习与实践.....	414
16.5.2 保存图片 (SavePicture 函数).....	384	第 18 章 SQL 应用.....	415
16.6 图形、图像处理控件.....	385	 视频讲解: 1 小时 27 分钟	
16.6.1 Shape 控件.....	385	18.1 数据库的基本知识.....	416
16.6.2 Line 控件.....	386	18.1.1 什么是数据库.....	416
16.6.3 PictureBox 控件.....	386	18.1.2 数据库软件的安装和使用.....	416
16.6.4 Image 控件.....	387	18.2 SQL 基础.....	421
16.7 小结.....	388	18.2.1 什么是 SQL.....	421
16.8 练习与实践.....	388	18.2.2 执行 SQL 语句的工具.....	422
第 17 章 多媒体技术.....	391	18.3 检索数据 (SELECT 子句).....	423
 视频讲解: 50 分钟		18.3.1 SELECT 子句.....	424
17.1 MMControl 控件.....	392	18.3.2 检索单个列.....	425
17.1.1 认识 MMControl 控件.....	392	18.3.3 检索多个列.....	425
17.1.2 MMControl 控件的属性.....	392	18.3.4 检索所有列.....	426
17.1.3 MMControl 控件的事件.....	397	18.4 排序检索数据 (ORDER BY 子句).....	426
17.2 Animation 控件.....	398	18.4.1 排序数据.....	426
17.2.1 认识 Animation 控件.....	398	18.4.2 按多个列排序.....	426
17.2.2 Animation 控件的属性.....	398	18.4.3 按列位置排序.....	427
17.2.3 Animation 控件的方法.....	399	18.4.4 指定排序方向.....	427
17.3 MediaPlayer 控件.....	400	18.4.5 对新生成的列进行排序.....	428
17.3.1 认识 MediaPlayer 控件.....	401	18.5 过滤数据 (WHERE 子句).....	428
17.3.2 MediaPlayer 控件的属性.....	401	18.5.1 使用 WHERE 子句.....	429
17.3.3 MediaPlayer 控件的方法.....	402	18.5.2 WHERE 子句比较运算符.....	429
17.4 ShockwaveFlash 控件.....	403	18.5.3 检索指定范围的值.....	430
17.4.1 认识 ShockwaveFlash 控件.....	403	18.5.4 模式条件查询.....	430



18.5.5 组合条件查询 (AND、OR 和 NOT) ....	431	19.6.1 认识 ADO 控件 .....	461
18.6 高级查询 .....	432	19.6.2 用 ADO 控件连接各种数据源 .....	462
18.6.1 汇总数据 .....	432	19.6.3 用 ADO 控件连接记录源 .....	464
18.6.2 分组统计 .....	433	19.6.4 ADO 控件常用属性、方法和事件 .....	464
18.6.3 子查询 .....	433	19.6.5 ADO 控件的综合应用 .....	465
18.7 插入数据 .....	434	19.7 小结 .....	466
18.7.1 插入完整的行 .....	434	19.8 练习与实践 .....	467
18.7.2 插入部分行 .....	434		
18.7.3 插入检索出的数据 .....	435	第 20 章 数据库控件 .....	469
18.7.4 将一个表中的数据复制到另一个表 .....	435	 视频讲解: 51 分钟	
18.8 修改和删除数据 .....	436	20.1 DBCombo 和 DBList 控件 .....	470
18.8.1 修改数据 .....	436	20.2 DataCombo 和 DataList 控件 .....	471
18.8.2 删除数据 .....	437	20.2.1 认识 DataCombo 和 DataList 控件 .....	471
18.9 小结 .....	437	20.2.2 DataCombo 和 DataList 控件的属性 .....	471
18.10 练习与实践 .....	437	20.2.3 显示关系表中的数据 .....	472
第 19 章 数据库开发技术 .....	439	20.3 DataGrid 控件 .....	474
 视频讲解: 1 小时 3 分钟		20.3.1 认识 DataGrid 控件 .....	474
19.1 VB 访问数据库 .....	440	20.3.2 用 DataGrid 控件显示数据 .....	474
19.2 ODBC .....	440	20.3.3 格式化数据 .....	476
19.2.1 认识 ODBC .....	440	20.3.4 锁定数据 .....	477
19.2.2 配置 ODBC 数据源 .....	441	20.3.5 将 DataGrid 控件中的数据显示在 文本框中 .....	477
19.3 DAO 对象 .....	443	20.4 MSFlexGrid 和 MSHFlexGrid 控件 ....	478
19.3.1 引用 DAO 对象 .....	443	20.4.1 认识 MSHFlexGrid 控件 .....	478
19.3.2 DAO 对象的子对象 .....	444	20.4.2 用 MSHFlexGrid 控件显示数据 .....	479
19.3.3 DAO 对象的综合应用 .....	448	20.4.3 数据排序与合并 .....	481
19.4 Data 控件 .....	451	20.4.4 隐藏行或列 .....	482
19.4.1 认识 Data 控件 .....	451	20.4.5 冻结字段 .....	482
19.4.2 用 Data 控件连接数据库 .....	452	20.5 小结 .....	483
19.4.3 Data 控件的综合应用 .....	453	20.6 练习与实践 .....	483
19.5 ADO 对象 .....	454	第 21 章 网络编程技术 .....	485
19.5.1 引用 ADO 对象 .....	454	 视频讲解: 26 分钟	
19.5.2 ADO 对象的子对象 .....	455	21.1 网络基础知识 .....	486
19.5.3 连接多种数据库 (Connection 对象) ....	455	21.1.1 OSI 参考模型 .....	486
19.5.4 连接记录源 (Recordset 对象) .....	457	21.1.2 HTTP 协议 .....	486
19.5.5 执行 SQL 语句 (Command 对象) .....	458	21.1.3 FTP 协议 .....	486
19.5.6 ADO 对象的综合应用 .....	459	21.2 Winsock 控件编程 .....	487
19.6 ADO 控件 .....	461	21.2.1 TCP 与 UDP 基础 .....	487

21.2.2 Winsock 控件 .....	487	21.4 WebBrowser 控件编程 .....	498
21.2.3 开发客户端/服务器端聊天程序 .....	491	21.4.1 WebBrowser 控件 .....	498
21.3 Internet Transfer 控件编程 .....	493	21.4.2 制作自己的浏览器 .....	499
21.3.1 Internet Transfer 控件 .....	493	21.5 小结 .....	501
21.3.2 文件上传与下载 .....	495	21.6 练习与实践 .....	501

## 第4篇 项目实战

### 第22章 企业进销存管理系统 .....

 视频讲解: 2小时45分钟

22.1 系统分析 .....	506	22.8.1 设计窗体界面 .....	527
22.1.1 需求分析 .....	506	22.8.2 向 ListView 控件中添加用户名 .....	528
22.1.2 可行性分析 .....	506	22.8.3 添加用户名和编号 .....	529
22.1.3 编写项目计划书 .....	507	22.8.4 判断用户名和密码 .....	529
22.2 系统设计 .....	509	22.8.5 移动无标题栏窗体 .....	530
22.2.1 系统目标 .....	509	22.9 主窗体设计 .....	531
22.2.2 系统功能结构 .....	509	22.9.1 设计窗体界面 .....	531
22.2.3 系统业务流程图 .....	510	22.9.2 设计菜单栏 .....	532
22.2.4 系统编码规范 .....	511	22.9.3 利用 Flash 设计工具栏 .....	533
22.3 系统运行环境 .....	513	22.9.4 利用图片设计浮动工具栏 .....	534
22.4 数据库与数据表设计 .....	513	22.9.5 设计状态栏 .....	536
22.4.1 数据库分析 .....	513	22.10 商品进货模块设计 .....	537
22.4.2 创建数据库 .....	514	22.10.1 设计窗体界面 .....	537
22.4.3 创建数据表 .....	515	22.10.2 窗体初始化 .....	539
22.4.4 数据表逻辑关系 .....	518	22.10.3 商品信息录入 .....	540
22.5 创建项目 .....	520	22.11 库存状况模块设计 .....	541
22.6 公共模块设计 .....	520	22.11.1 设计窗体界面 .....	542
22.6.1 主函数 .....	521	22.11.2 窗体初始化 .....	544
22.6.2 数据库连接函数 .....	521	22.11.3 库存上下限设置 .....	544
22.6.3 拼音简码函数 .....	522	22.11.4 自定义过程向 MSFlexGrid 控件中 添加数据 .....	545
22.7 启动窗体的设计 .....	523	22.12 月销售状况模块设计 .....	546
22.7.1 设计窗体界面 .....	523	22.12.1 设计窗体界面 .....	546
22.7.2 添加资源文件 .....	524	22.12.2 统计全年商品销售状况 .....	547
22.7.3 代码注册 Flash 控件 .....	525	22.12.3 设计“每月销售比较”窗体界面 .....	548
22.7.4 调用 Flash 动画 .....	526	22.12.4 利用图表分析月销售状况 .....	549
22.8 系统登录窗体设计 .....	526	22.13 系统用户及权限设置模块设计 .....	553
		22.13.1 设计窗体界面 .....	553

22.13.2 窗体初始化.....	554	22.16.3 解决用户定义类型未定义的问题.....	560
22.13.3 工具栏按钮.....	554	22.16.4 数据批量录入.....	561
22.13.4 执行操作.....	555	22.16.5 使用数据回滚来恢复数据备份.....	563
22.14 运行项目 .....	556	22.16.6 字段大小问题导致数据添加失败.....	563
22.15 程序打包 .....	558	22.16.7 字段设置主键后不能插入重复值.....	564
22.16 开发常见问题与解决 .....	559	22.16.8 数据库中表存在关系, 如何进行 数据库清理.....	564
22.16.1 书写错误的函数名.....	559	22.17 小结 .....	565
22.16.2 提示文件未找到错误信息.....	559		



# 第 1 篇

## 基础知识

- » 第 1 章 初识 Visual Basic 6.0
- » 第 2 章 VB 语言基础
- » 第 3 章 算法和程序控制结构
- » 第 4 章 数组与集合
- » 第 5 章 过程
- » 第 6 章 内置函数

本篇通过初识 Visual Basic 6.0、VB 语言基础、算法和程序控制结构、数组和集合、过程和内置函数并结合大量的图示、举例、录像等，让读者快速掌握 Visual Basic 语言，并为以后编程奠定坚实的基础。





# 第 1 章

## 初识 Visual Basic 6.0

(教学录像: 1 小时 27 分钟)

Visual Basic 是 Microsoft 公司推出的可视化的开发环境, 是 Windows 下最优秀的程序开发工具之一。利用 Visual Basic 可以开发出具有良好交互功能、良好的兼容性和扩展性的应用程序。

本章致力于使读者了解 Visual Basic 的宏观世界, 知道如何安装和卸载 Visual Basic 程序, 掌握 Visual Basic 集成开发环境的各个要素, 并能编写简单的应用程序。

通过阅读本章, 您可以:

- » 了解 Visual Basic 的发展和特点
- » 掌握 Visual Basic 的安装与卸载
- » 掌握 Visual Basic 的启动方法
- » 掌握 Visual Basic 集成开发环境中各要素的使用
- » 掌握如何定制开发环境
- » 熟悉如何使用 Visual Basic 的帮助系统
- » 通过编写第一个 Visual Basic 程序来熟悉 Visual Basic 程序开发的基本流程

## 1.1 Visual Basic 简介

 教学录像：光盘\TM\lx\1\Visual Basic 简介.exe

使用一门语言，就要对这门语言有所了解，下面介绍 Visual Basic 的发展和 Visual Basic 6.0 的特点。

### 1.1.1 Visual Basic 的发展

微软公司在 1991 年推出了建立在 Windows 开发平台基础上的开发工具 Visual Basic 1.0。随着 Windows 操作平台的不断完善，微软公司也相继推出了 Visual Basic 2.0、Visual Basic 3.0 和 Visual Basic 4.0，这些版本主要用于在 Windows 3.X 环境中的 16 位计算机上开发应用程序。1997 年微软公司推出的 Visual Basic 5.0 可以在 Windows 9X 或者 Windows NT 环境中的 32 位的计算机上开发应用程序。1998 年，微软公司又推出了 Visual Basic 6.0，使得 Visual Basic 在功能上进一步完善和扩充，尤其在数据库管理、网络编程等方面得到了更加广泛的应用。图 1.1 中所示的是 Visual Basic 及其他语言近年来的发展趋势。

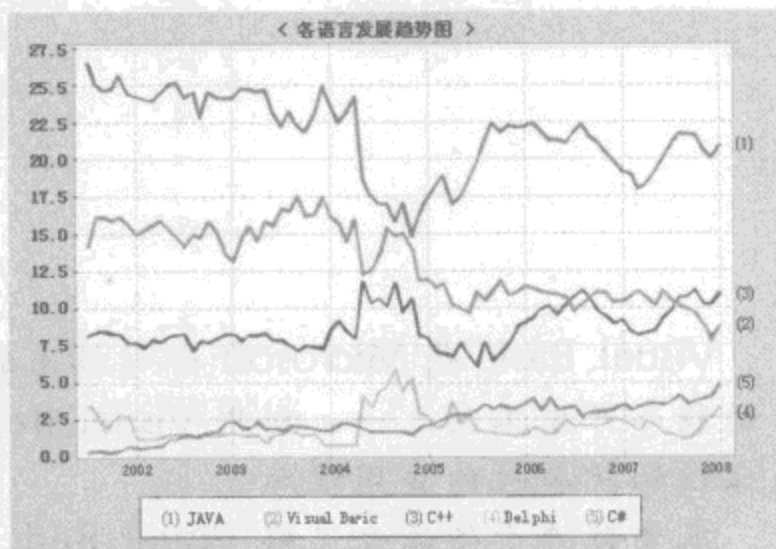



图 1.1 VB 语言近年来的发展趋势

 说明：图 1.1 中的数据摘自 <http://www.tiboe.com> 网站。

### 1.1.2 Visual Basic 6.0 的特点

Visual 的意思是“视觉的，可视的”，Visual Basic 就是可视化的编程语言。使用 Visual Basic 语言进行编程时会发现，在 Visual Basic 中无须编程就可以完成许多步骤。因为在 Visual Basic 中引入了控件的概念，在 Windows 中控件的身影无处不在，如按钮、文本框等，Visual Basic 把这些控件模式化，并且每个控件都有若干属性用来控制控件的外观、工作方法，并且能够响应用户操作（事件）。这样就可以像在画板上一样，随意点几下鼠标，一个按钮就完成了，这些在以前的编程语言环境下是要经过相当复杂的操作的。

在初步了解了 Visual Basic 语言之后，下面来介绍一下 Visual Basic 语言的特点：

#### 1. 可视化编程

Visual Basic 为用户提供大量的界面元素（Visual Basic 中称为控件），例如“窗体”、“菜单”、“命令按钮”等，用户只需要利用鼠标或键盘把这些控件拖动到适当的位置，设置它们的外观属性等，就可以设计出所需的应用程序界面。

Visual Basic 还提供了易学易用的集成开发环境,在该环境中集程序的设计、运行和调试为一体,在本章后面的小节中将对集成开发环境进行详细地介绍。

## 2. 事件驱动机制

Windows 操作系统出现以来,图形化的用户界面和多任务多进程的应用程序要求程序设计不能是单一性的,在使用 Visual Basic 设计应用程序时,必须首先确定应用程序如何同用户进行交互。例如,发生鼠标单击、键盘输入等事件时,用户必须编写代码控制这些事件的响应方法。这就是所谓的事件驱动编程。

## 3. 面向对象的程序设计语言

VB 6.0 是支持面向对象的程序设计语言。它不同于其他的面向对象的程序设计语言。用户不需要编写描述每个对象的功能特征的代码,这些都被封装到各个控件中了,用户只需调用即可。

## 4. 支持多种数据库访问机制

Visual Basic 6.0 具有强大的数据库管理功能。利用其提供的 ADO 访问机制和 ODBC 数据库连接机制,可以访问多种数据库,如 Access、SQL Server、Oracle、MySQL 等。数据库连接方面的知识,也将在后面的章节中进行介绍。

 注意:为了简单起见,以后都用 VB 表示 Visual Basic。

# 1.2 如何学好 VB

 教学录像:光盘\TM\lx\1\如何学好 VB.exe

了解 VB 的发展及特点后,下面介绍 VB 的作用以及如何学好 VB。

## 1.2.1 VB 可以做什么

学习 VB 可以用来做什么呢?或者说学习 VB 以后可以开发出什么样的程序来呢?很多读者都会问这样的一些问题。其实,只要是可想到的程序 90%都可以用 VB 来开发和实现。从设计新型的用户界面到利用其他应用程序的对象,从处理文字图像到使用数据库,从开发小工具到大型企业应用系统,甚至通过 Internet 的遍及全球分布式应用程序,都可利用 VB 来实现。

VB 是微软公司的一种通用程序设计语言,包含在 Microsoft Excel、Microsoft Access 等众多 Windows 应用软件中的 VBA 都是使用 VB 语言,以供用户二次开发;目前制作网页使用较多的 VBScript 脚本语言也是 VB 的子集。

## 1.2.2 学习 VB 的几点建议

VB 语言如此强大,对于一个对编程一无所知,而又迫切希望掌握一种快捷实用的编程语言的初学者来说能快速上手吗?没问题。它的快捷的开发速度,简单易学的语法,体贴便利的开发环境,是初

学者的首选。下面针对初学者的学习，笔者提出几点学习的建议，如图 1.2 所示。

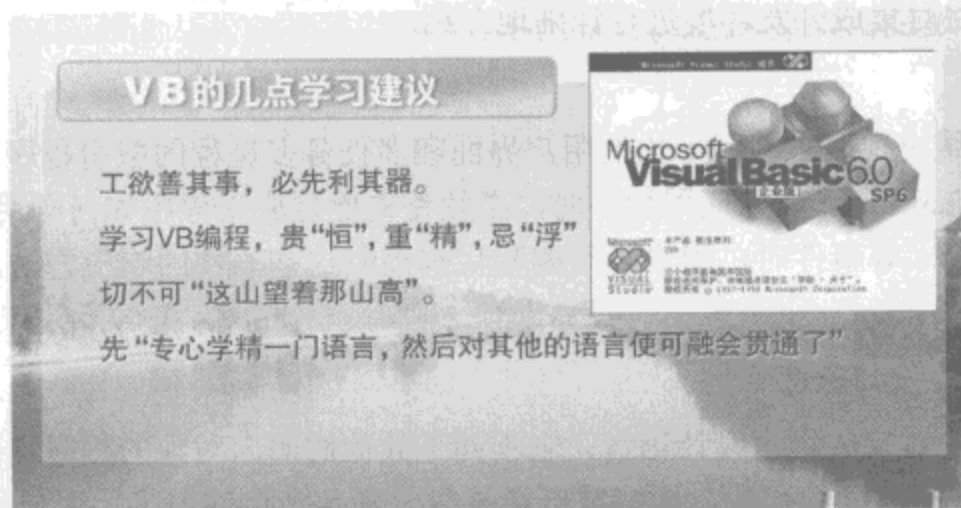


图 1.2 学习 VB 的几点建议

### 1. 工欲善其事，必先利其器

要想利用 VB 开发出高质量的程序，必须先打下良好的基础，然后再一步一步地不断提升技术水平。万事开头难，刚刚开始，遇到些困难没关系，慢慢来。编程是一个不断学习、不断积累的过程，编程的乐趣也正是在于学习的过程中。

### 2. 学习 VB 编程，贵“恒”，重“精”，忌“浮”

学习语言切不可半途而废，一定要持之以恒，只有坚持到最后才可以看见最美丽的彩虹。在学习的过程中一定要“精”，就是说在选择学习的图书、工具、网站时，一定要选择信誉和口碑都很好的，并且精益求精，切不可为了图便宜买盗版图书，或者去不正规的网站学习，这样会给学习带来不便。学习时要忌浮躁，切不可觉得自己学习了一点就沾沾自喜，以为自己的技术已经很好了。

### 3. 切不可“这山望着那山高”

学习中切不可“这山望着那山高”，今天觉得 VB 好就学习 VB，明天觉得 VC 好就学习 VC，这样到最后肯定是一事无成。

VB 和 VC++、VF 等都是 Microsoft Visual Studio 编程套装组合里的成员。因此，学习微软的开发工具，绝对不用担心过时的问題，因为作为世界软件龙头，微软早就考虑到这个问题了。

### 4. 先专心学精一本语言，然后对其他语言就可以融会贯通了

这是许多老程序员告诫新手的一句话，在此与大家共勉。其实每种语言只是在语法上有所不同，在设计和开发思路都是相通的，因此首先学习好 VB，以后即可轻松地学习其他语言。

## 1.3 VB 6.0 的安装与管理

 教学录像：光盘\TM\lx\1\VB 6.0 的安装与管理.exe

在了解了 VB 的发展、版本以后，下面介绍 VB 6.0 的安装、启动和退出的方法。下面以安装 VB 6.0 中文企业版为例，介绍 VB 6.0 的安装、启动和退出。



### 1.3.1 VB 6.0 的运行环境

#### 1. 硬件要求

在安装 VB 6.0 时要注意硬盘的剩余空间, 下面列出安装 VB 6.0 时所需的硬件要求。

- ☒ 90MHz 或更高的微处理器。
- ☒ VGA (640×480) 或者更高的监视器。
- ☒ 鼠标或其他定点设备 (如指令杆、滚动球等)。
- ☒ CD-ROM 或 DVD-ROM 驱动器。
- ☒ 32MB 以上内存。
- ☒ 磁盘空间要求如下。
  - 学习版: 典型安装 48MB, 完全安装 80MB。
  - 专业版: 典型安装 48MB, 完全安装 80MB。
  - 企业版: 典型安装 128MB, 完全安装 147MB。

#### 2. 软件要求

VB 6.0 可以在多个操作系统下运行, 如 Windows 98、Windows 2000、Windows 2003、Windows XP、Windows Vista 等。

VB 6.0 需要在 Windows 95 (或更高版本的操作系统)、Windows NT3.51 (或更高版本的操作系统) 上安装。

### 1.3.2 VB 6.0+SP6 的安装

#### 1. 安装 VB 6.0

(1) 将 VB 6.0 的安装光盘放入光驱, 系统会自动执行安装程序。如果不能自动安装, 可以双击安装光盘中的 Setup.exe 文件 (如图 1.3 所示)。执行安装程序, 将弹出如图 1.4 所示的安装程序向导。

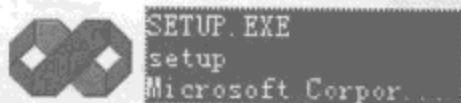


图 1.3 安装文件图标

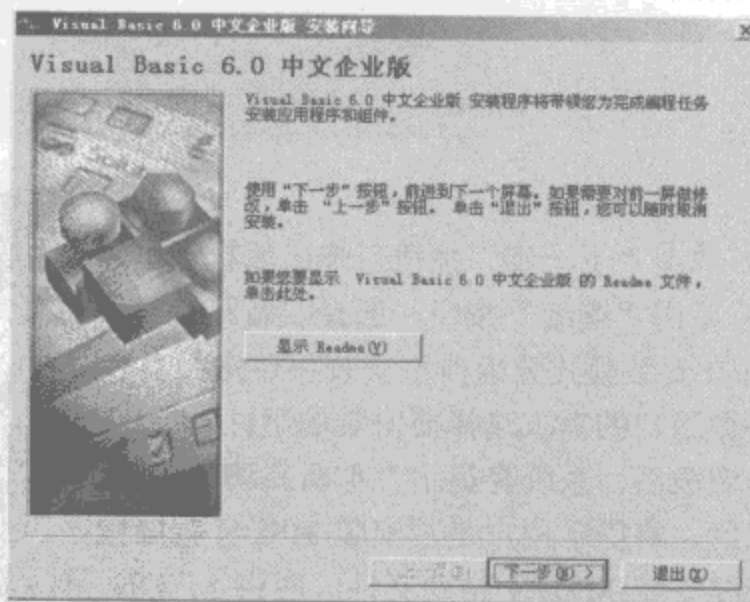


图 1.4 “VB 6.0 中文企业版安装向导”对话框

- (2) 单击“下一步”按钮，选择“接受协议”选项。
- (3) 单击“下一步”按钮，在“产品号和用户 ID”对话框中输入产品 ID、姓名与公司名称。
- (4) 单击“下一步”按钮，在“Visual Basic 6.0 中文企业版”对话框中选择“安装 Visual Basic 6.0 中文企业版”选项，如图 1.5 所示。
- (5) 单击“下一步”按钮，设置安装路径，然后打开“选择安装类型”对话框，如图 1.6 所示。

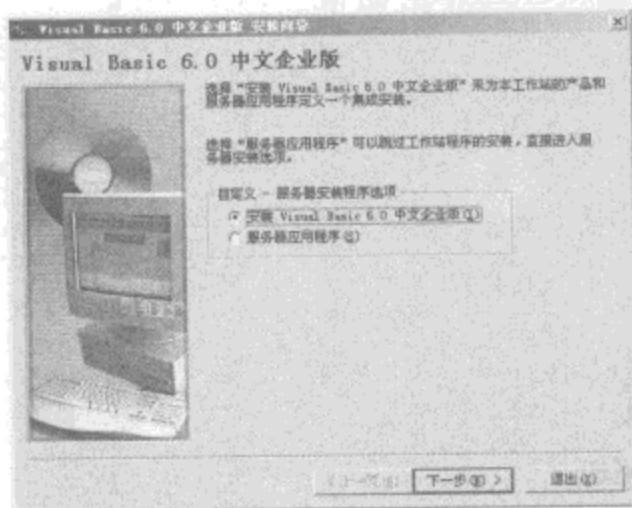


图 1.5 “选择安装程序”对话框

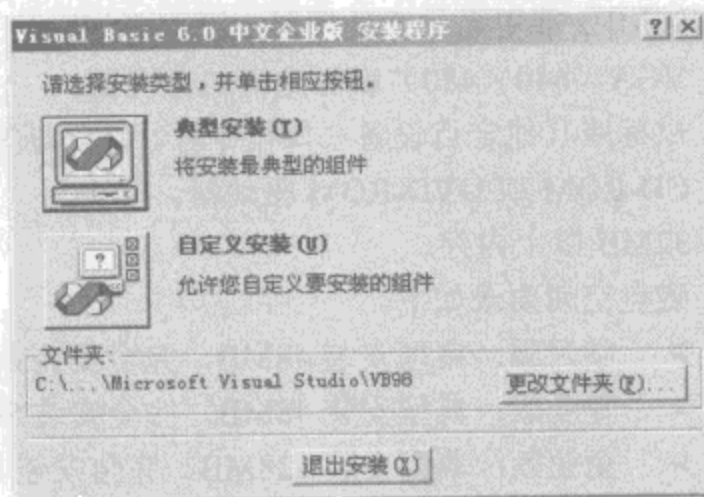


图 1.6 “选择安装类型”对话框

- (6) 在“选择安装类型”对话框中，如果选择“典型安装”，系统会自动安装一些最常用的组件；选择“自定义安装”，用户则可以根据自己的实际需要有选择地安装组件，如图 1.7 所示。

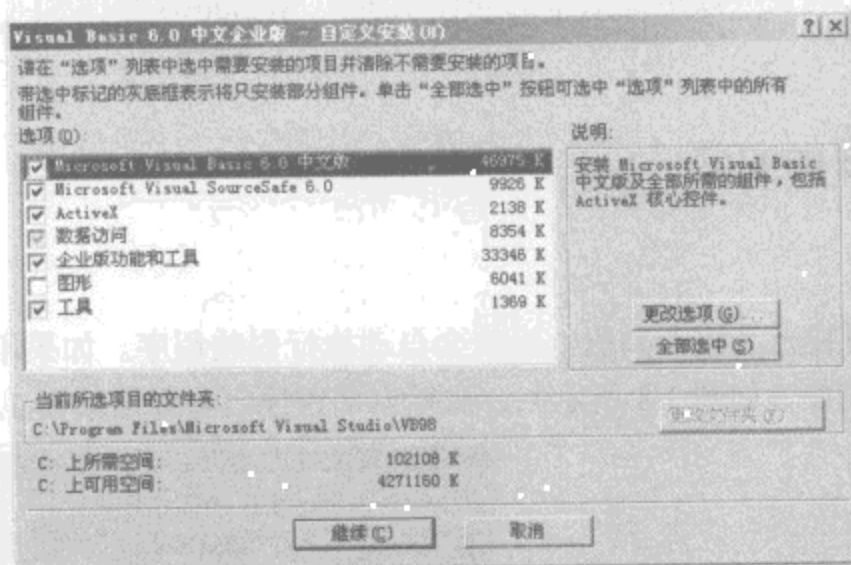


图 1.7 自定义安装

- (7) 单击“下一步”按钮，弹出版权警示与说明内容对话框。

- (8) 单击“继续”按钮，选择安装路径与安装模式后，将开始自动安装 VB 6.0 环境。

VB 6.0 安装模式分为典型安装和自定义安装两种。在一般的情况下采用典型安装模式；自定义安装可以根据用户的需求选择要安装的组件。

安装完成后，系统将提示“重新启动计算机”，以便进行一系列的更新及配置工作。当 VB 6.0 安装完成以后，将提示用户是否安装 MSDN 帮助程序。

如果要安装 MSDN 帮助文件，应将 MSDN 帮助文件光盘放入光驱，按提示进行安装。安装完成 MSDN 程序后，在 VB 6.0 开发环境中按〈F1〉键，即可打开 MSDN 帮助程序。如果用户不想安装 MSDN，

在安装界面中取消 MSDN 安装选项即可。

## 2. 安装 VB 6.0 的 SP6 补丁

为了使安装的 VB 6.0 更加完整和全面,在安装完 VB 6.0 以后还需要安装补丁程序 SP6。SP6 补丁程序可以到微软的网站上自行下载,其下载后是一个可执行文件,双击图标即可安装,这里不再赘述。

### 1.3.3 VB 6.0 的更改或删除

安装完 VB 6.0 后,在程序开发的过程中,有时还需要添加或删除某些组件。具体实现的步骤如下:

(1) 将 VB 6.0 光盘插入到光驱中。

(2) 双击“控制面板”中的“添加或删除程序”,打开“添加或删除程序”对话框。

(3) 在当前程序列表中选择“Microsoft Visual Basic 6.0 中文企业版(简体中文)”选项,如图 1.8 所示。

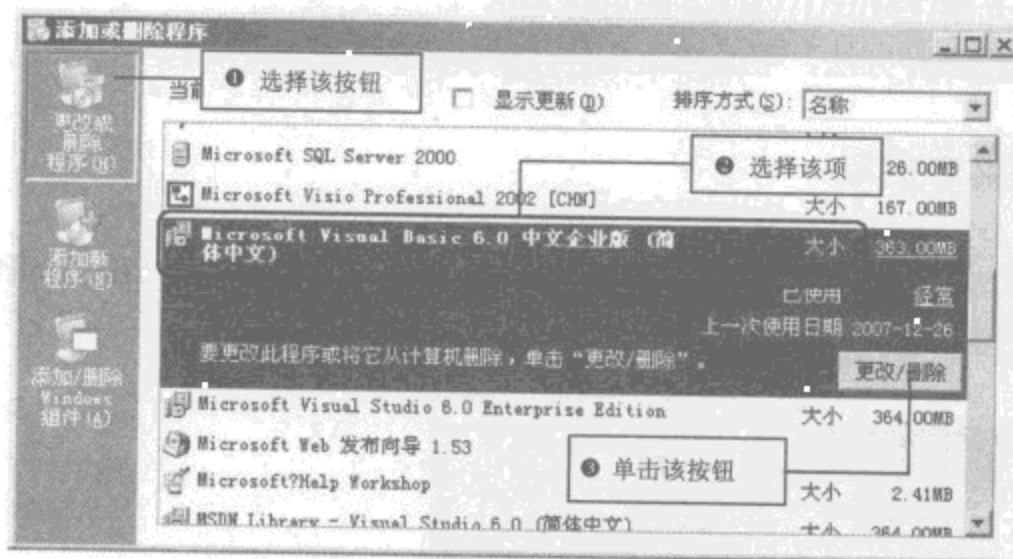


图 1.8 删除 VB 6.0 程序

(4) 单击“更改/删除”按钮。此时将弹出 VB 6.0 安装程序对话框,如图 1.9 所示。

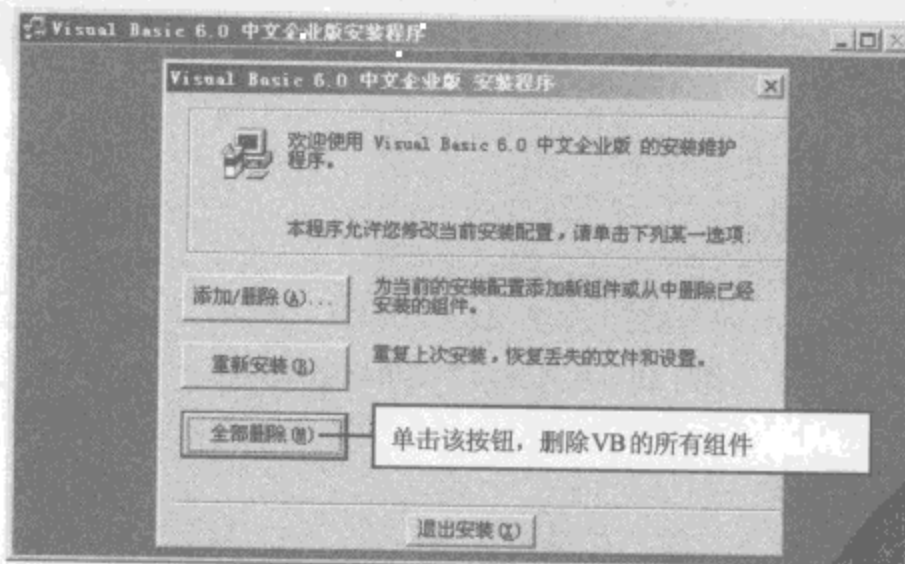


图 1.9 “VB 6.0 中文企业版安装程序”对话框

(5) 在如图 1.9 所示的对话框中包括 3 个选项按钮:

☒ “添加/删除”按钮

如果用户要添加新的组件或删除已安装的组件, 单击此按钮, 在弹出的“Maintenance Install”对话框中, 选中需要添加或清除组件前面的复选框。

☒ “重新安装”按钮

如果以前安装的 VB 6.0 有问题, 可单击此按钮重新安装。

☒ “全部删除”按钮

单击此按钮可将 VB 6.0 所有组件从系统中卸载。

## 1.4 VB 6.0 的启动

 教学录像: 光盘\TM\1\VB 6.0 的启动.exe。

VB 6.0 的启动有很多种方法, 下面介绍几种比较常用的方法。

### 1.4.1 通过开始菜单启动

选择“开始”/“所有程序”/“Microsoft Visual Basic 6.0 中文版”/“Microsoft Visual Basic 6.0 中文版”命令, 如图 1.10 所示。

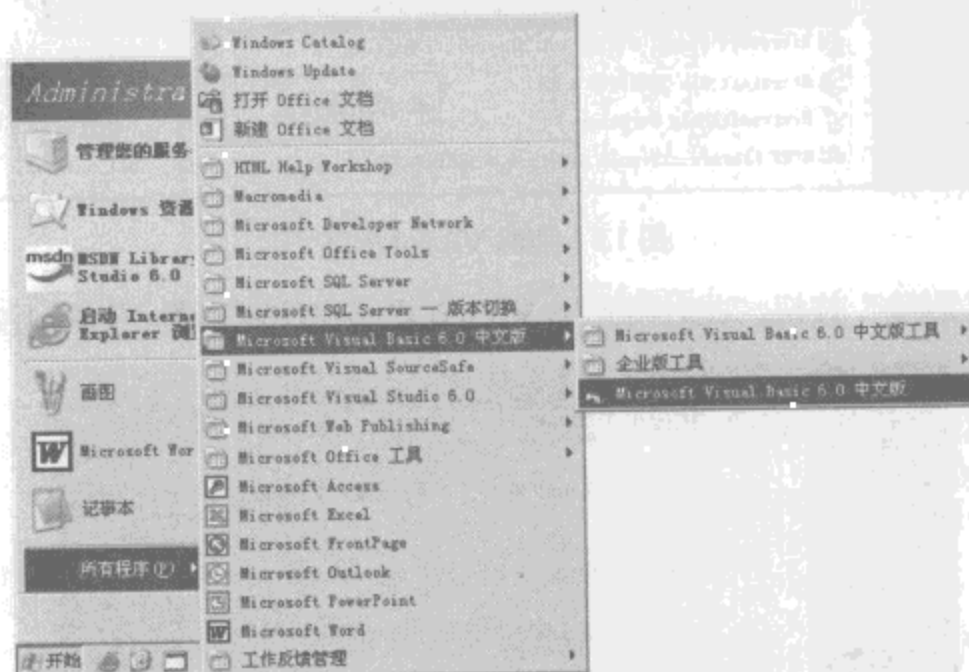


图 1.10 从“开始”菜单启动 VB 6.0

### 1.4.2 通过快捷方式启动

如果在桌面上创建了快捷方式, 可以通过在桌面上双击 VB 6.0 的快捷方式图标来启动 VB 6.0。

VB 6.0 启动时, 首先看到如图 1.11 所示的界面。在启动界面中, 可以看到相关的信息: 安装的

VB 6.0 的版本, 这里为企业版; 该版本所安装的补丁, 即升级服务包, 这里为 SP6 (Service Pack 6)。

在启动 VB 6.0 以后, 将打开一个“新建工程”对话框。在该对话框中包括 3 个选项卡, 分别是“新建”、“现存”、“最新”, 其具体的功能如下:

- ☑ “新建”选项卡, 显示可打开的工程类型。
- ☑ “现存”选项卡, 显示一个对话框, 可以在那里定位并选择要打开的工程。
- ☑ “最新”选项卡, 列出最近打开的工程及其位置。

选择“新建”选项卡, 选择“标准 EXE”图标, 单击“打开”按钮, 即可创建一个标准 EXE 工程, 如图 1.12 所示。



图 1.11 VB 6.0 启动界面

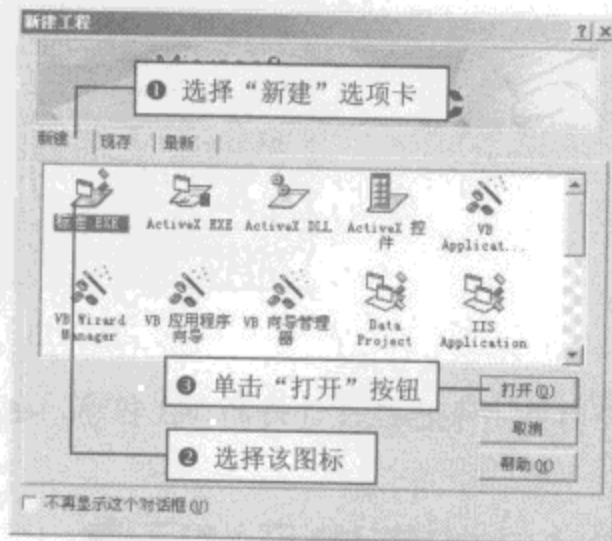


图 1.12 “新建工程”对话框

在“新建”选项卡中, 列出了用户可以创建的工程的类型, 根据需要用户可以创建不同类型的工程。表 1.1 中列出了其中常用的工程类型。

表 1.1 常用的工程类型

图 标	类 型	说 明
	标准EXE	创建一个标准的可执行文件
	ActiveX EXE	创建一个ActiveX可执行文件
	ActiveX DLL	创建一个ActiveX动态链接库文件
	ActiveX控件	创建一个ActiveX控件
	VB向导管理器	创建一个向导程序
	数据工程	创建一个数据工程
	DHTML应用程序	创建一个基于网络浏览器的应用程序
	IIS应用程序	创建一个用于开发网络应用程序的服务器端程序
	VB 企业版控件	创建一个具有企业版控件的应用程序

注意: “新建”选项卡仅在启动 VB 6.0 时出现, 在选择“文件”/“新建工程”命令时, 出现的“新建工程”对话框中, 将不出现该选项卡。

在启动 VB 时, 可以略过“新建工程”对话框, 直接创建一个标准的 EXE 工程。具体的方法如下: 选择“工具”/“选项”命令, 弹出“选项”对话框, 在该对话框中选择“环境”选项卡, 在“启



动 Visual Basic 时”区域中选择“创建缺省工程”单选按钮，单击“确定”按钮，即可在启动时创建一个标准的 EXE 工程，如图 1.13 所示。如果还想显示“新建工程”对话框，选择“启动 Visual Basic 时”区域中的“提示创建工程”单选按钮即可。

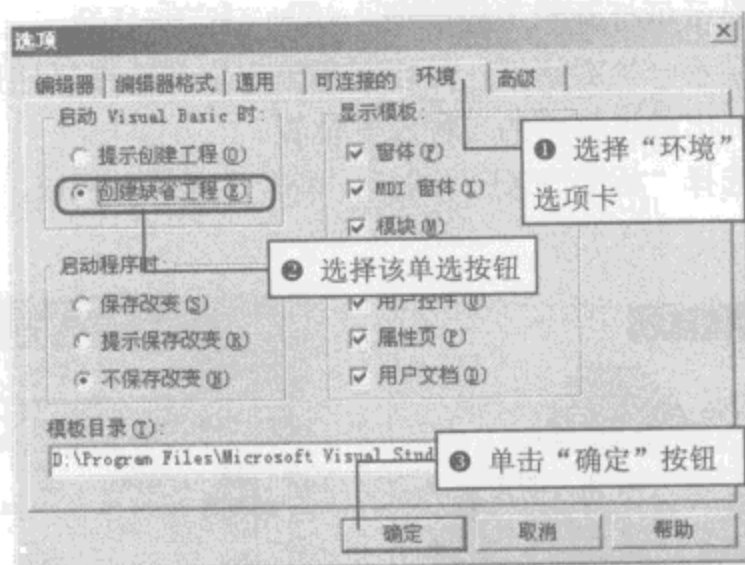


图 1.13 “选项”对话框

说明：打开一个已经设计好的 VB 程序，也可以启动 VB 6.0。

## 1.5 VB 6.0 的集成开发环境

教学录像：光盘\TM\1\VB 6.0 的集成开发环境.exe

本节主要介绍 VB 6.0 的集成开发环境，包括菜单栏、工具栏、工具箱、工程资源管理器、属性窗口、窗体布局窗口和窗体设计器等。

### 1.5.1 集成开发环境简介

集成开发环境，Integrated Development Environment (IDE)，是指一个集设计、运行和测试应用程序为一体的环境，VB 6.0 已经不只是一门单纯的语言，而是一个集成开发环境。在这个环境中可以进行程序的设计、运行和测试。

当用户在“新建工程”对话框中选择“标准 EXE”图标，单击“确定”按钮以后，即可以进入到 VB 6.0 的集成开发环境中。VB 6.0 集成开发环境包括工作区窗口、菜单栏、工具栏、工程资源管理器、属性窗口、窗体布局窗口和立即窗口等，如图 1.14 所示。

### 1.5.2 菜单栏

#### 1. 菜单的内容

菜单栏显示了所有可用的 VB 命令。其中不仅包括“文件”、“编辑”、“帮助”等常见标准命

令菜单, 还包括 VB 的专用编程菜单, 如“工程”、“调试”及“运行”等。用鼠标单击可以打开菜单项, 也可以通过〈Alt〉键加上菜单项上的字母打开菜单项。菜单栏的显示效果如图 1.15 所示。

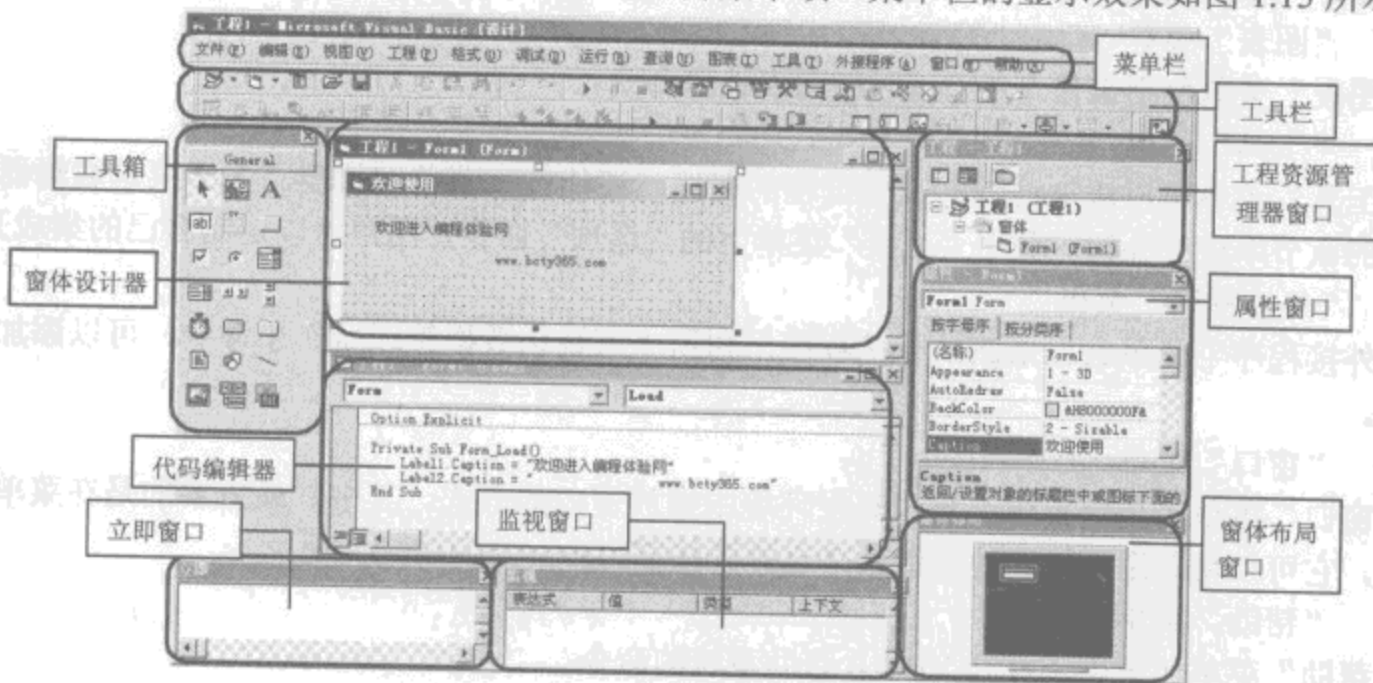


图 1.14 集成开发环境



图 1.15 菜单栏

#### ☑ “文件”菜单

“文件”菜单可以用于创建、打开、保存文件对象和编译应用程序。在这个菜单中还可以设置打印机信息、打印文件或退出 VB。

#### ☑ “编辑”菜单

“编辑”菜单包含在窗体设计时或代码编写时使用的各种编辑命令。实现了标准剪切板的操作, 如“剪切”、“复制”和“粘贴”等, 还有类似 Word 的“查找”、“替换”等操作。

#### ☑ “视图”菜单

“视图”菜单用于显示或隐藏集成开发环境中的各种窗口、工具栏以及其他组成部分的命令。

#### ☑ “工程”菜单

“工程”菜单是用户操作工程的核心, 利用该菜单可以设置工程属性、为工具箱添加部件、引用对象、为工程添加窗体等。

#### ☑ “格式”菜单

“格式”菜单主要用于处理控件在窗体中的位置, 包括在设计控件时需要使用的各种命令, 如“对齐”、“统一尺寸”、“调整间距”等。

#### ☑ “调试”菜单

“调试”菜单包括程序调试时所需要的各种命令。如“逐语句”、“逐过程”、“切换断点”等。

#### ☑ “运行”菜单

“运行”菜单包括了用于启动、终止程序执行的命令。如“启动”、“全编译执行”、“中断”、“结束”、“重新启动”命令。

### ☑ “查询”菜单

“查询”菜单包括涉及查询或 SQL 语句的命令。如“运行”、“清除结果”、“验证 SQL 语法”等。

### ☑ “图表”菜单

“图表”菜单包括操作 VB 工程时的图表处理命令。

### ☑ “工具”菜单

“工具”菜单可以添加过程并设置过程的属性，还能打开菜单编辑器。关于菜单编辑器的使用将在后面的章节中进行介绍。利用“工具”菜单下的“选项”命令，用户可以定制自己的集成开发环境。

### ☑ “外接程序”菜单

“外接程序”菜单可以增删的外接程序，利用“可视化数据管理器”子命令，可以添加、删除外接程序。

### ☑ “窗口”菜单

“窗口”菜单为用户提供在集成开发环境中摆放窗口的方式。其中，最重要的是在菜单底部的窗口清单，它可以帮助用户快速地激活某个已打开的窗口。

### ☑ “帮助”菜单

“帮助”菜单包含用于打开 VB 6.0 帮助系统的命令。

## 2. 菜单的选择

用户可以通过下面的方法打开菜单栏中的菜单项：

(1) 单击菜单项。

(2) 按〈Alt〉键+访问键。

(3) 按〈F10〉键或〈Alt〉键激活菜单栏，再按访问键打开菜单。或者在激活菜单以后按〈↑〉、〈↓〉键打开菜单项。

(4) 在菜单项激活或打开以后，可以利用〈←〉、〈→〉键选择相邻的菜单。

(5) 当菜单项被打开以后，利用〈↑〉、〈↓〉键选择菜单命令，按〈Enter〉键执行命令。

## 3. 集成开发环境中的快捷菜单

在对象上单击鼠标右键即可打开快捷菜单，它包括了经常使用的操作命令。由于鼠标所指向的对象的不同，弹出的快捷菜单也是不同的，即快捷菜单取决于鼠标右击的对象。

下面介绍一下这些快捷菜单中比较常用的几个。

### ☑ “工具箱”快捷菜单

在工具箱上单击鼠标右键，将弹出如图 1.16 所示的快捷菜单。在该快捷菜单中，最常用的是“部件”命令，执行该命令，将弹出一个“部件”对话框，用于添加控件、设计器或者可插入的对象。

### ☑ “窗体”快捷菜单

在窗体上单击鼠标右键，即可弹出如图 1.17 所示的快捷菜单，在该菜单中比较常用的命令有：“菜单编辑器”、“锁定控件”和“粘贴”命令。其中，“菜单编辑器”命令，用于调用菜单编辑器，为应用程序设计菜单；“锁定控件”命令，用于将窗体上的控件锁定，以防止用户随意移动；“粘贴”命令，用于执行在窗体上粘贴控件或其他对象的操作。

### ☑ “工程资源管理器”快捷菜单

在工程资源管理器上单击鼠标右键，即可弹出如图 1.18 所示快捷菜单，该快捷菜单中“添加”菜单下面的子菜单是比较常用的，用于添加窗体、模块或者设计器等。

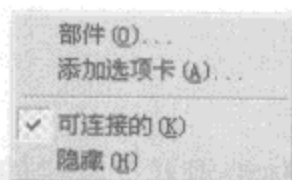


图 1.16 “工具箱”快捷菜单

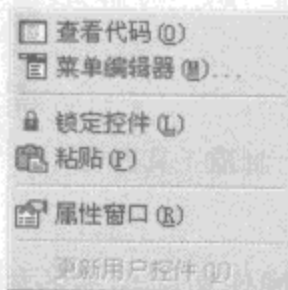


图 1.17 “窗体”快捷菜单

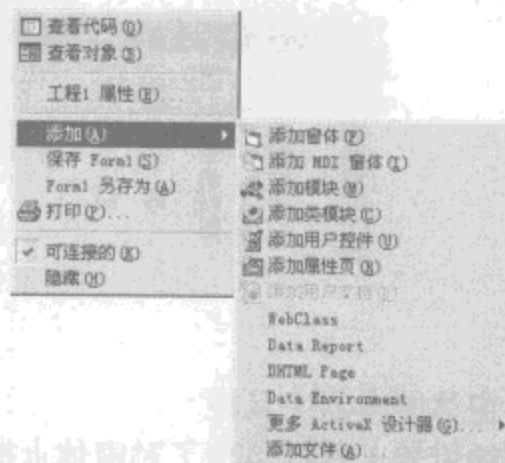


图 1.18 “工程资源管理器”快捷菜单

### 1.5.3 工具栏

和大多数的 Windows 应用程序一样，VB 6.0 也将菜单中的常用功能放置到工具栏中，通过这些工具栏可以快速访问菜单中的常用命令。

在工具栏上单击鼠标右键，可以弹出如图 1.19 所示的快捷菜单，用户可以根据需要自己添加或删除工具栏。也可以选择“自定义”命令，设置工具栏按钮。

从图 1.19 中可以看出，VB 6.0 所包含的工具栏有“编辑”工具栏、“标准”工具栏、“窗体编辑器”工具栏、“调试”工具栏 4 种，其添加到 VB 6.0 工程中的效果如图 1.20 所示。

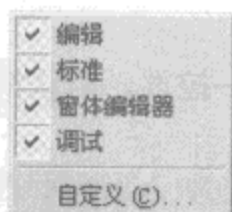


图 1.19 添加工具栏的快捷菜单

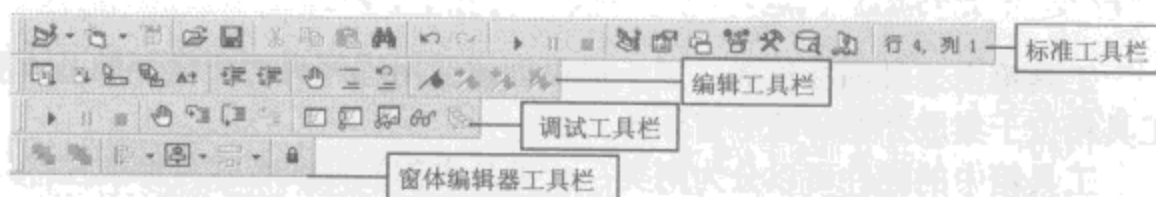


图 1.20 工具栏

#### ☑ 编辑工具栏

编辑工具栏包括了在进行编辑时所使用的命令按钮，如图 1.21 所示。

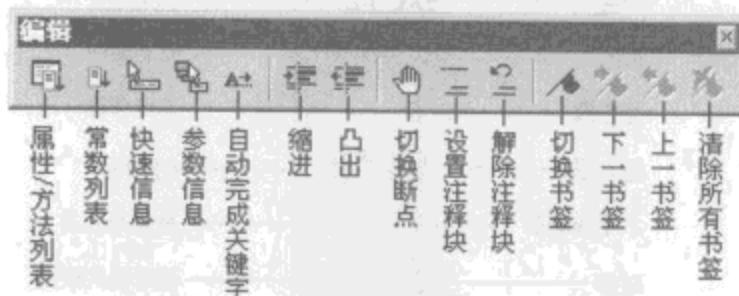


图 1.21 编辑工具栏

#### ☑ 标准工具栏

标准工具栏包括了在 VB 程序开发中可以用到的大部分的命令按钮，如“添加标准工程”、“添加窗体”、“添加菜单编辑器”等。标准工具栏如图 1.22 所示。



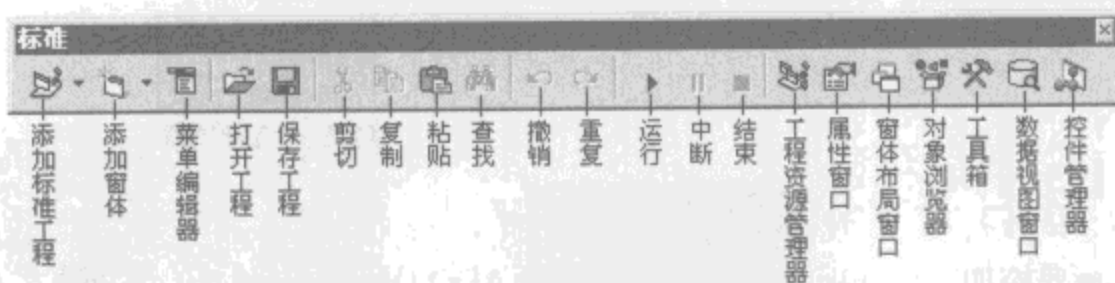


图 1.22 标准工具栏

#### ☑ 窗体编辑器工具栏

窗体编辑器工具栏包括了对窗体上控件进行操作所需要的各种命令，窗体编辑器工具栏如图 1.23 所示。

#### ☑ 调试工具栏

调试工具栏包括了在进行程序调试时所需要使用的命令，调试工具栏如图 1.24 所示。

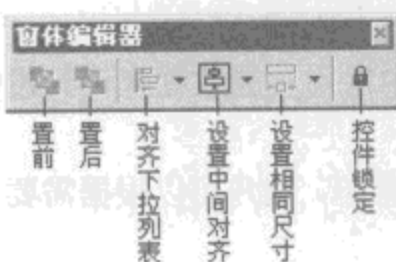


图 1.23 窗体编辑器工具栏

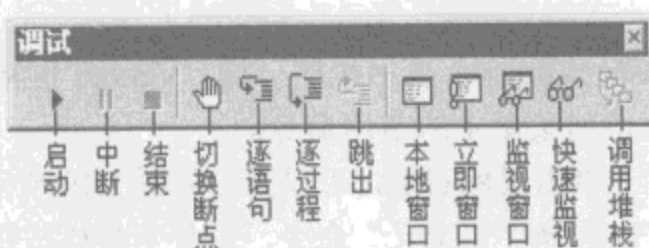


图 1.24 调试工具栏

## 1.5.4 工具箱

工具箱由工具图标组成，用于提供创建应用程序界面所需要的基本要素——控件。默认情况下，工具箱位于集成开发环境中窗体的左侧。

工具箱中的控件可以分为两类：一类是内部控件或者称为标准控件，另一类为 ActiveX 控件，需要手动添加到应用程序中。如果没有手动添加，则默认只显示内部控件。工具箱如图 1.25 所示。

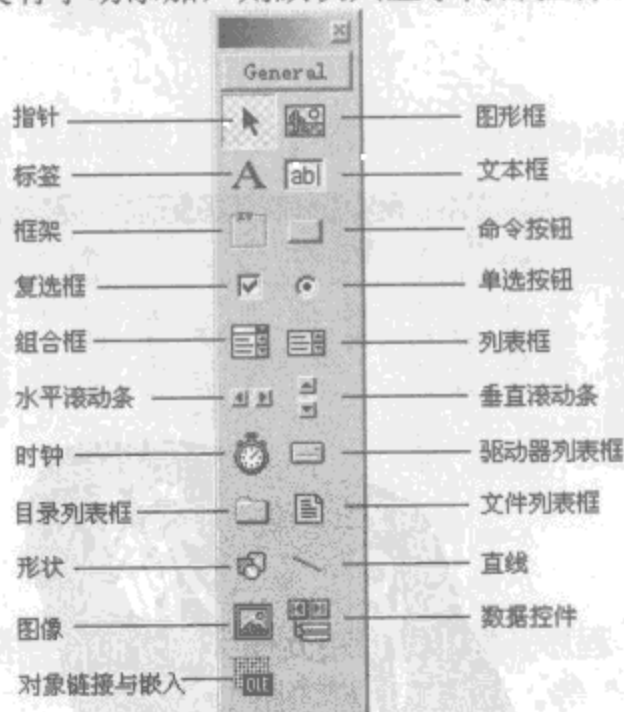


图 1.25 工具箱



用户可以自己手动设计工具箱，将所需要的控件或者选项卡添加到工具箱中。下面介绍如何向工具箱中添加 ActiveX 控件和选项卡。

### 1. 添加 ActiveX 控件

在工具箱上单击鼠标右键，在弹出的快捷菜单中选择“部件”命令，将弹出“部件”对话框。在“控件”选项卡中选中需要添加的控件项，如选中 Microsoft ADO Data Control 6.0 (SP6)。如果在控件列表中没有所需要的控件，则可以通过单击“浏览”按钮，将所需要的控件添加到控件列表中。选择完毕，单击“确定”按钮，即可将 ADO 控件添加到工具箱中，具体的执行过程如图 1.26 所示。

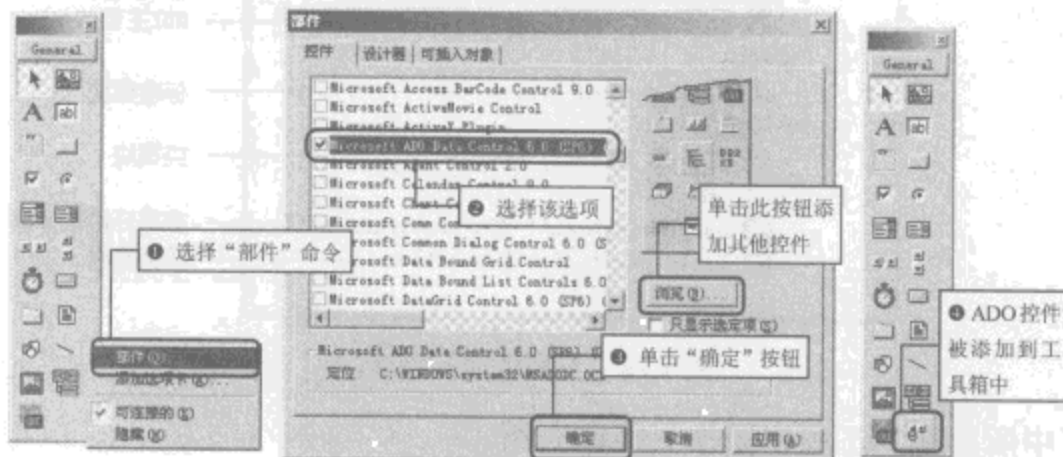


图 1.26 添加 ActiveX 控件

### 2. 添加选项卡

当添加的 ActiveX 控件过多时，都存放在一起不便于查找，这时可以在工具箱中添加一个选项卡，将控件分门别类，这样就便于查找和使用。具体的添加选项卡的方法如下：

在工具箱上单击鼠标右键，在弹出的快捷菜单中选择“添加选项卡”命令，在弹出的对话框中输入要创建的选项卡的名称，如“ActiveX 控件”，单击“确定”按钮，即可在工具箱中添加一个选项卡，效果如图 1.27 所示。

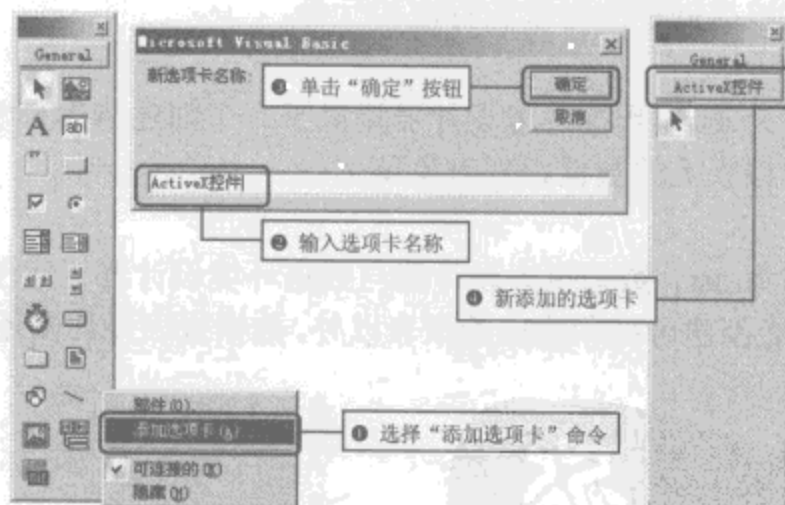


图 1.27 添加选项卡

## 1.5.5 工程资源管理器

工程资源管理器窗口列出了当前应用程序中所使用的窗体、模块、类模块、环境设计器以及报表

设计器等资源。

在工程中，用户可以通过单击标题栏上面的关闭按钮将其关闭，并通过选择“视图”/“工程资源管理器”命令将其显示，也可以通过使用快捷键〈Ctrl+R〉来实现。工程资源管理器窗口如图 1.28 所示。

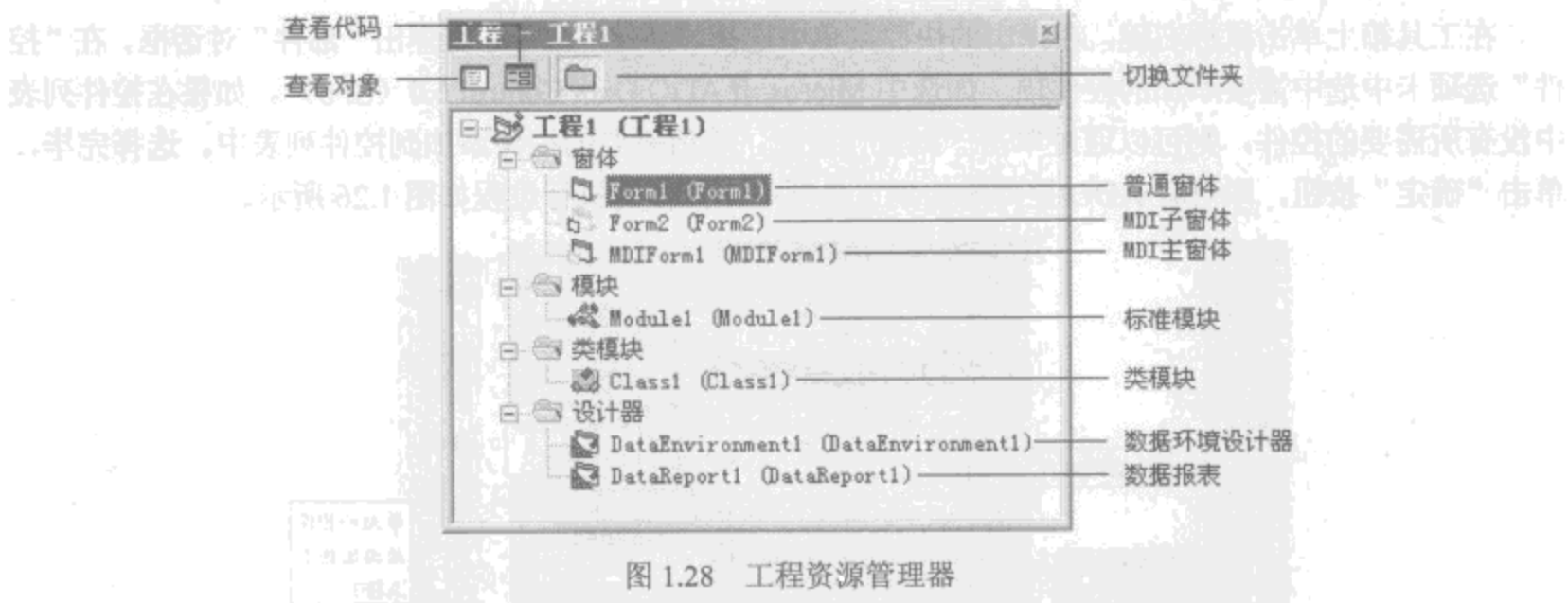


图 1.28 工程资源管理器

下面对图 1.28 中所出现的工程资源作一简单介绍。

#### ☑ 窗体模块

窗体模块的文件扩展名为.frm，是 VB 应用程序的基础。在窗体模块中可以设置窗体控件的属性、窗体级变量、常量的声明以及过程和函数的声明等。窗体模块包括普通窗体、MDI 主窗体、MDI 子窗体。

#### ☑ 标准模块

标准模块的文件扩展名为.bas，只包含过程、类型以及数据的声明和定义的模块。在标准模块中，模块级别声明和定义都被默认为 Public。

#### ☑ 类模块

类模块的文件扩展名为.cls，类模块是一个模板，用于创建工程中的对象，并为对象编写属性和方法。模块中的代码描述了从该类创建的对象的特性和行为。

#### ☑ 数据环境

数据环境的文件扩展名为.dsr，数据环境设计器提供了一个创建 ADO 对象的交互式的设计环境，可以作为数据源提供窗体或报表上的数据识别对象使用。

#### ☑ 数据报表

数据报表的文件扩展名为.dsr，数据报表设计器与数据环境设计器一起使用。可以通过几个不同的相关联的表创建报表。除了能创建可打印输出的报表以外，数据报表设计器还可以将报表导出到 HTML 或文本文件中。

## 1.5.6 属性窗口

属性窗口用于显示或设置已经选定的对象（如窗体、控件、类等）的各种属性名和属性值。用户可以通过设置“按字符序”或“按分类序”选项卡，来设置属性窗口中的属性的排序方式。可以在属性值文本框或下拉列表框中输入或选择属性的值，并进行修改或设置。在属性窗口的属性描述区域中显示了当前所选定属性的具体的意义，通过属性描述，用户可以快速地了解属性意义。

在工程中，用户可以通过单击标题栏上的关闭按钮，将属性窗口关闭，通过选择“视图”/“属性窗口”命令显示该窗口，也可以通过快捷键〈F4〉来实现。属性窗口的组成如图 1.29 所示。



图 1.29 属性窗口

### 1.5.7 窗体布局窗口

窗体布局窗口位于集成开发环境的右下角，主要用于指定程序运行时的初始位置，使所开发的程序能在各种不同分辨率的屏幕上正常运行，常用于多窗体的应用程序。

在窗体布局窗口中可以将所有可见的窗体都显示出来。当把光标放置到某个窗体上时，它改变为一个 $\oplus$ 。在运行时，通过鼠标可以将窗体定位在希望它出现的地方。

可以通过在窗体布局窗口上单击鼠标右键，在弹出的快捷菜单中选择“分辨率向导”命令，设置不同的分辨率。

选中要设置启动位置的窗体以后，单击鼠标右键，在弹出的快捷菜单中选择“启动位置”命令，设置该窗体的启动位置。

在工程中，用户可以通过选择“视图”/“窗体布局窗口”命令，来显示该窗口。窗体布局窗口的组成如图 1.30 所示。

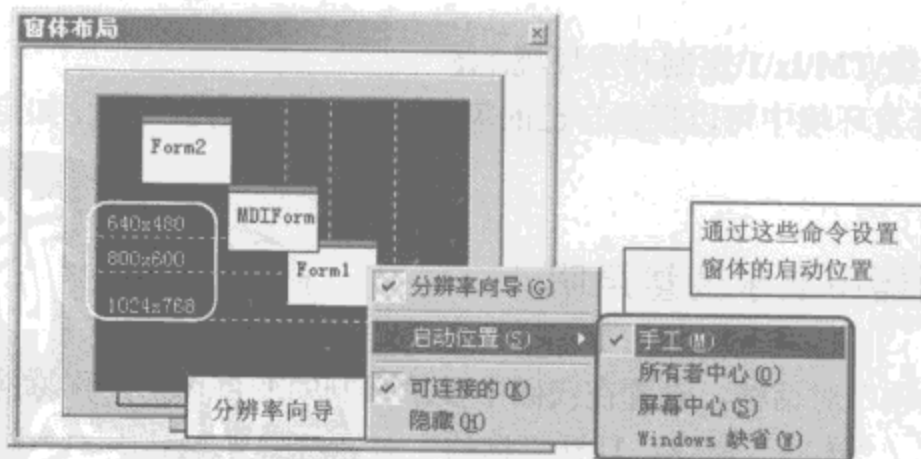


图 1.30 窗体布局窗口

## 1.5.8 窗体设计器

窗体是应用程序最主要的组成部分，每个窗体模块都包含事件过程，即代码部分，其中有为响应特定事件而执行的指令。窗体可包含控件。在窗体模块中，对窗体上的每个控件都有一个对应的事件过程集。除了事件过程，窗体模块还可包含通用过程，它对来自任何事件过程的调用都作出响应。

在工程中，选择“视图”/“对象窗口”命令，即可显示窗体设计器。如图 1.31 所示。

## 1.5.9 代码编辑窗口

代码编辑窗口也就是代码编辑器，用于输入应用程序的代码。工程中的每个窗体或代码模块都有一个代码编辑窗口，代码编辑窗口一般和窗体是一一对应的。标准模块或类模块只有代码编辑窗口，没有窗体部分。

在工程中，可以通过选择“工程”/“代码窗口”命令，显示代码编辑窗口。代码编辑窗口的各部分功能如图 1.32 所示。



图 1.31 窗体设计器

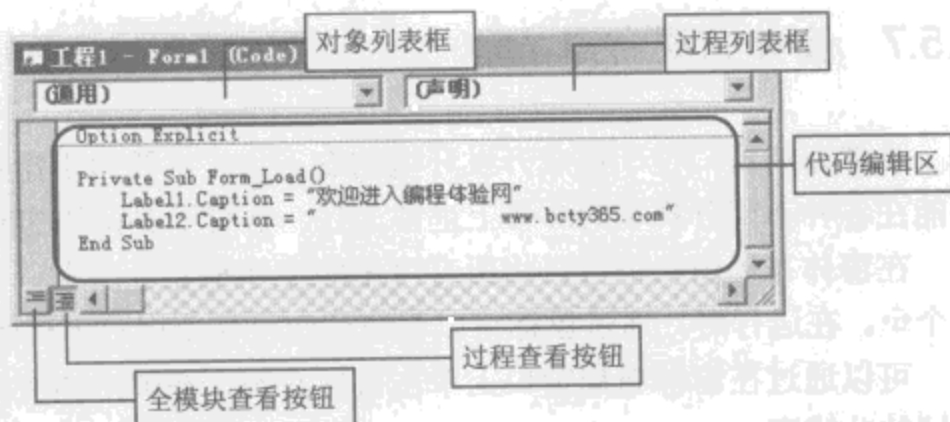


图 1.32 代码编辑窗口

## 1.6 定制开发环境

教学录像：光盘\TM\lx\1\定制开发环境.exe

在 VB 6.0 的集成开发环境中可以根据自己的需要定制自己的开发环境。如定制各子窗口、工具栏、通用选项等。

### 1.6.1 设置在编辑器中要求变量声明

用户可以通过“选项”对话框，设置在代码中要求强制的变量声明，具体的操作步骤如下：

- (1) 选择“工具”/“选项”命令，打开“选项”对话框。
- (2) 选择“编辑器”选项卡，在“代码设置”区域中选“要求变量声明”复选框。
- (3) 单击“确定”按钮，完成设置。



这样在代码的编辑区域中，将自动添加 Option Explicit 语句。操作过程如图 1.33 所示。

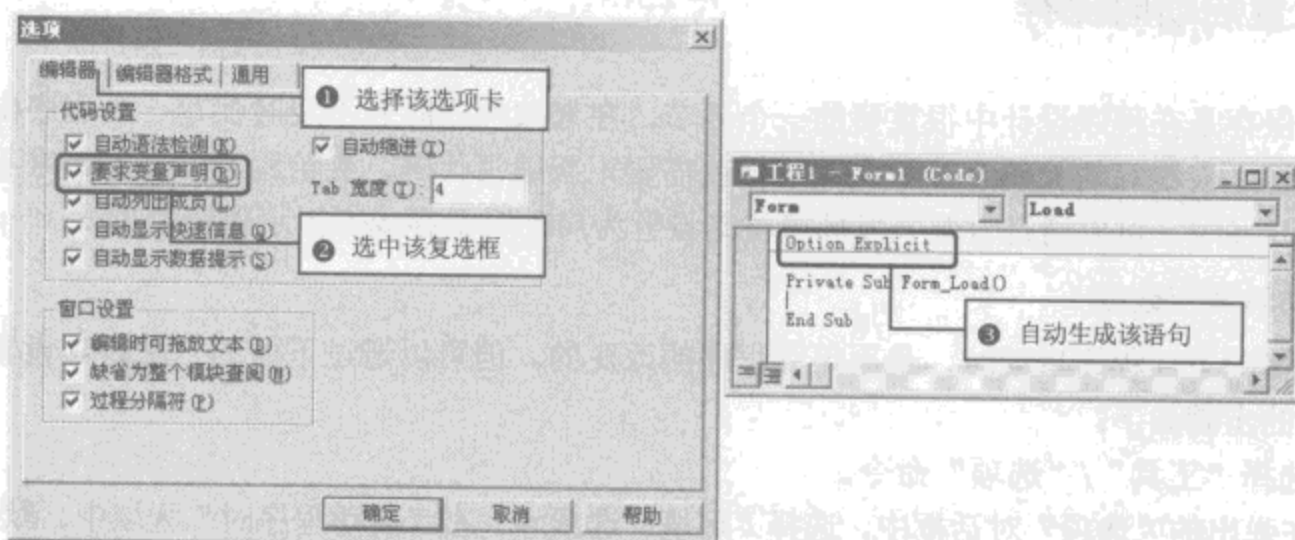


图 1.33 强制变量声明的设置过程

## 1.6.2 设置网格大小和不对齐到网格

在窗体上有一些排列整齐的点，这些点就是窗体上的网格，VB 利用这些网格精确确定控件的位置。这些网格的大小是可以调整的，也可以将控件设置为不对齐到网格上，这样在调整控件位置的时候就可以利用〈Ctrl〉键加上〈↑〉、〈↓〉、〈←〉和〈→〉键来微调控件的位置。具体的设置方法如下：

(1) 选择“工具”/“选项”命令，弹出“选项”对话框。

(2) 在“选项”对话框中，选择“通用”选项卡。

(3) 在“窗体网格设置”区域中，选中“显示网格”复选框。如果不选中该项，在窗体上将不显示网格。在“高度”和“宽度”文本框中设置网格的大小，默认的大小为 120×120。这里为了突出显示效果，将其设置为 500×500。

(4) 取消选中“对齐控件到网格”的复选框，单击“确定”按钮，完成设置。

具体的操作过程如图 1.34 所示。

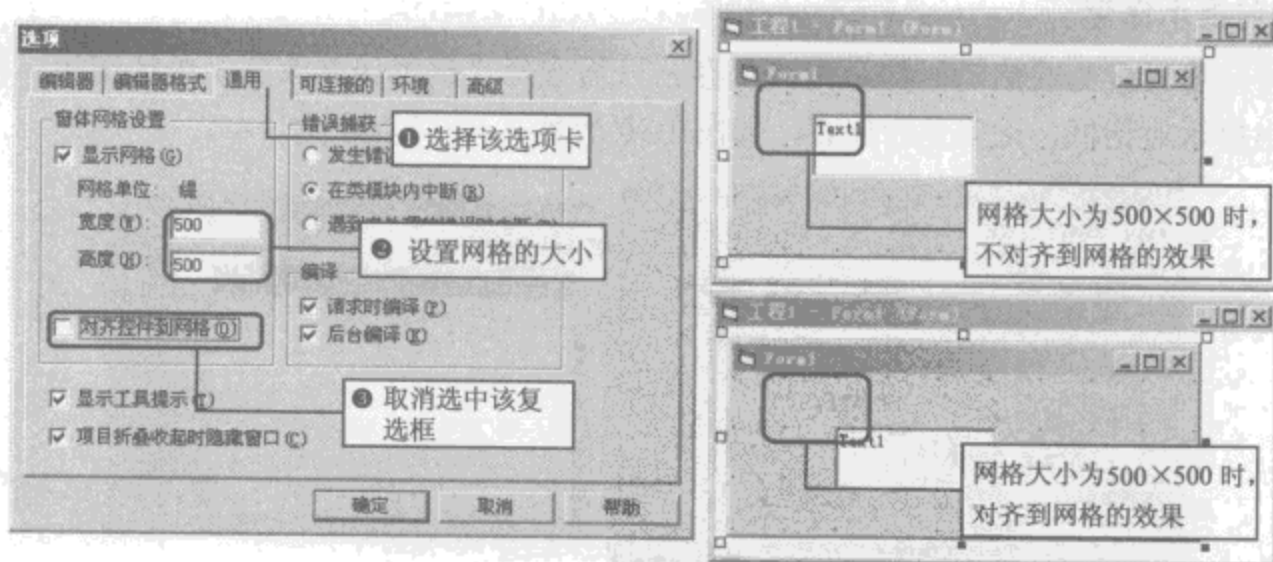


图 1.34 设置网格大小和是否对齐到网格



### 1.6.3 设置启动时保存

工程的保存是在程序设计中很重要的一个环节，在修改程序时若不及时保存，当程序出现错误自动关闭时会将之前编写的代码全部丢失，这样就需要重新编写代码，会给程序的开发带来不必要的麻烦。在程序开发时，可以通过设置开发环境将其设置为启动时保存或者提示保存的形式。

#### 1. 启动时保存改变

在默认情况下，程序启动时，是不保存程序的改变的。但可以通过下面的方法将其设置为启动时保存，具体的步骤如下：

(1) 选择“工具”/“选项”命令。

(2) 在弹出的“选项”对话框中，选择“环境”选项卡。在“启动程序时”区域中，默认情况下，设置为“不保存改变”，这里选择“保存改变”单选按钮。

(3) 单击“确定”按钮，关闭“选项”对话框，当程序没保存就启动时，系统将会自动保存。如果工程为新创建的，没有存储路径，将弹出“文件另存为”对话框，如图 1.35 所示。

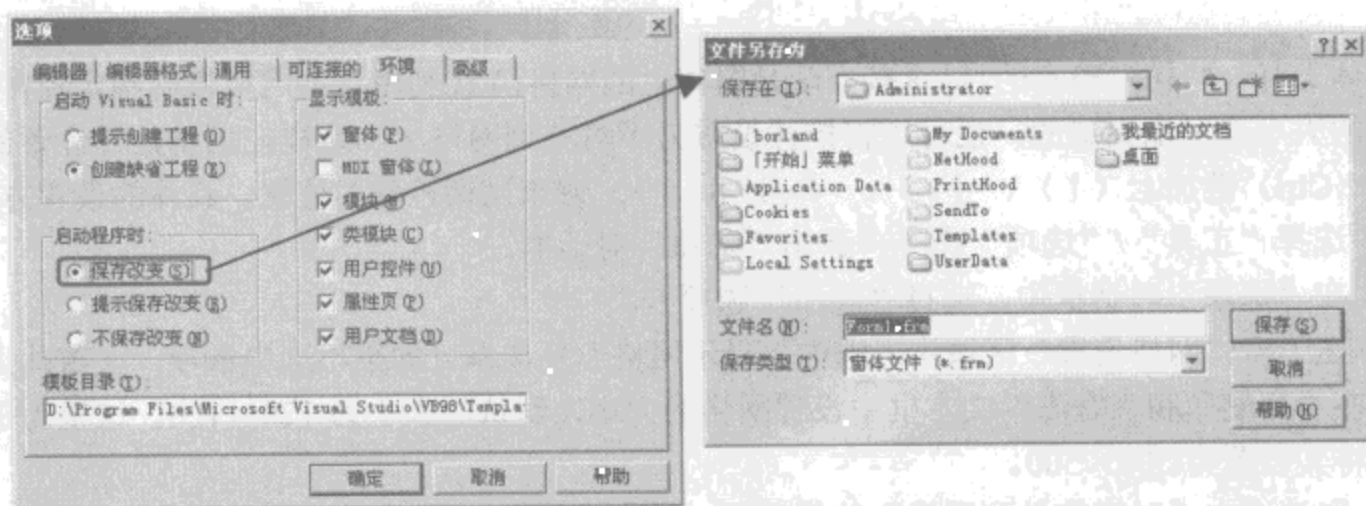


图 1.35 启动时保存

#### 2. 启动时提示保存

也可以将环境设置为在启动时提示是否保存。只需选择“提示保存改变”单选按钮，即可在程序启动时，弹出提示对话框，询问是否保存，如图 1.36 所示。

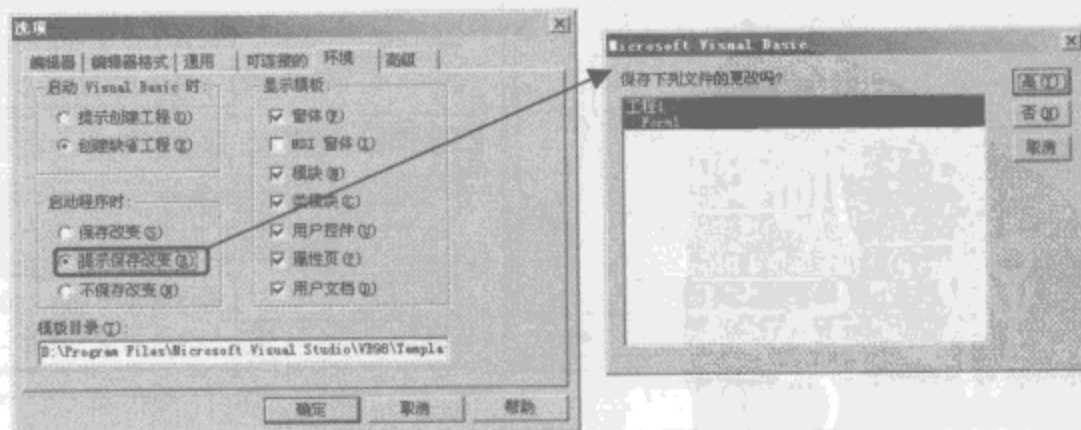


图 1.36 启动时提示保存

### 1.6.4 定制工具栏

工具栏是将一些在菜单中经常使用的命令组合在一起。如果有些菜单命令经常被使用到，可以将其添加到工具栏菜单中。用户可以通过选择“视图”/“工具栏”/“自定义”命令，将启动“自定义”对话框。

下面根据个人的需要创建一个自己的工具栏，用于存放自己经常使用的菜单命令，具体方法如下：

(1) 启动“自定义”对话框，选择“工具栏”选项卡，单击“新建”按钮，将弹出“新建工具栏”对话框。

(2) 在该对话框中输入要创建的工具栏的名称，这里为“我的工具栏”。单击“确定”按钮，完成工具栏的添加。

(3) 再次启动“自定义”对话框，选择“命令”选项卡，拖动想要添加的命令到“我的工具栏”，如：拖动“打开工程”命令到“我的工具栏”。当鼠标变成箭头带一个加号的形式时，释放鼠标。

(4) 重复步骤(3)将需要的命令都添加到“我的工具栏”上，单击“关闭”按钮，关闭“自定义”对话框。

整个创建自定义工具栏的操作过程如图 1.37 所示。

### 1.6.5 为代码编辑器设置鼠标滚动

默认安装的 VB 6.0，在编辑代码时，并不支持鼠标滚轮，这将给用户的程序开发带来很大的不便。微软提供了支持鼠标滚动的 DLL，用户可以到微软网站下载这个 DLL 文件，从而使自己的开发环境具有支持鼠标滚动的功能。具体的下载地址为：<http://download.microsoft.com/download/e/f/b/efb39198-7c59-4ace-a5c4-8f0f88e00d34/vb6mousewheel.exe>。

下载后将压缩文件解压，将 vb6idmousewheeladdin.dll 文件复制到“C:\Windows\system32\”目录下。这里 C:\Windows 是默认的系统路径，如果操作系统安装在其他的盘符下，则修改为其他的路径。然后，在开始菜单下的“运行”对话框中输入“regsvr32 vb6idmousewheeladdin.dll”，如图 1.38 所示。

**注意：**由于这里是系统路径所以不需要添加具体的路径，如果文件没有存放在系统路径下，则需要在 vb6idmousewheeladdin.dll 文件的前面添加具体的路径。

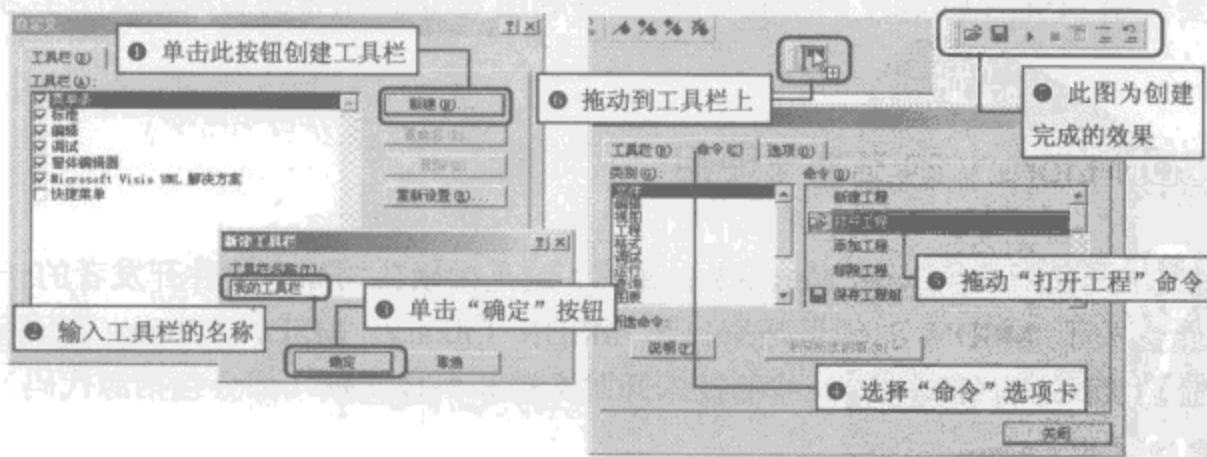


图 1.37 定制工具栏

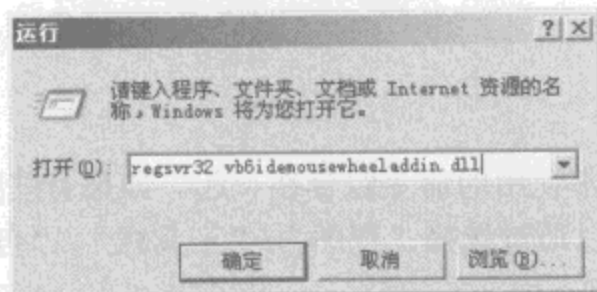


图 1.38 注册 DLL 文件

单击“确定”按钮，注册 DLL 文件，当弹出注册成功的对话框时，则说明注册成功。启动 VB 6.0，选择“外接程序”/“外接程序管理器”命令，在“外接程序管理器”对话框中选择 MouseWheel Fix 选项，在“加载行为”区域中选中“加载/卸载”和“在启动中加载”复选框，单击“确定”按钮，完成操作，如图 1.39 所示。

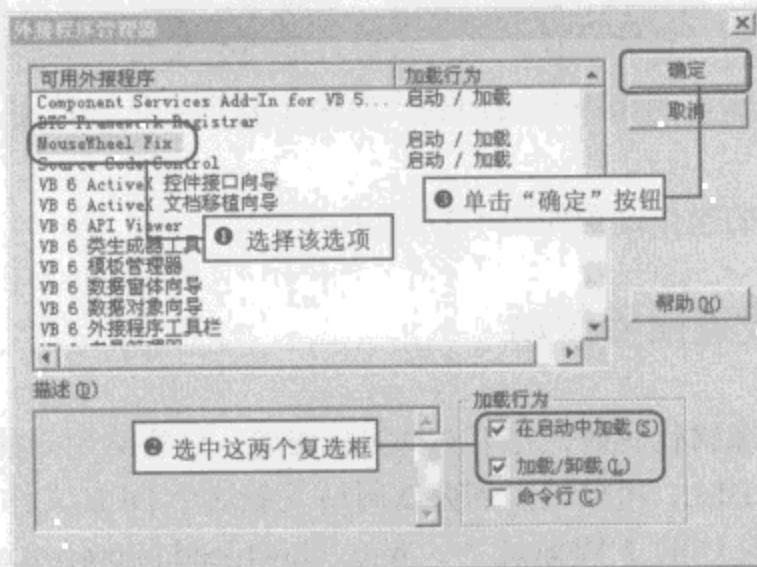


图 1.39 “外接程序管理器”对话框

## 1.7 VB 6.0 的帮助系统

教学录像：光盘\TM\1\1\VB 6.0 的帮助系统

本节主要介绍 MSDN Library 的安装与使用，并利用 MSDN 附带的实例源程序学习编程和使用 VB 的帮助菜单。

### 1.7.1 MSDN Library 的安装与使用

MSDN 是 Microsoft Developer Network 的缩写。这是微软公司面向软件开发者的一种信息服务。用户接触到的最多关于 MSDN 的信息是来自于 MSDN Library。MSDN Library 就是通常人们眼中的 MSDN，涵盖了微软全套可开发产品线的技术开发文档和科技文献（部分包括源代码）。

#### 1. 安装 MSDN Library

在安装 VB 6.0 以后，将弹出“安装向导”对话框，在该对话框中选择 MSDN 单选按钮，单击“下

一步”按钮，即可安装 MSDN。MSDN 的安装非常简单。

## 2. 启动 MSDN Library

安装完成以后，用户可以通过下面两种方法打开 MSDN。

### ☒ 通过“开始”菜单启动

通过在“开始”菜单中，选择“程序”/Microsoft Developer Network/MSDN Library Visual Studio 6.0 (CHS) 命令，启动 MSDN。

### ☒ 在集成开发环境中启动

如果启动了 VB 6.0 的集成开发环境，可以通过“帮助”菜单启动 MSDN，启动后的 MSDN 如图 1.40 所示。

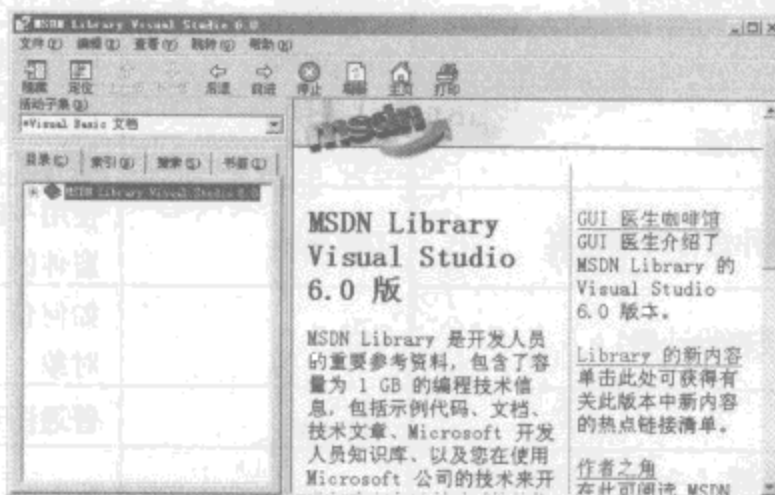


图 1.40 启动后的帮助菜单

## 3. 使用 MSDN Library

在程序开发过程中 MSDN 可以帮助用户解决程序开发中遇到的相关问题，用户只需选定需要帮助的相关对象，然后按〈F1〉键，即可获取相关的 MSDN 帮助信息。

## 1.7.2 利用附带的实例源程序学习编程

VB 附带的实例源程序位于 MSDN 光盘中，用户可以在安装 MSDN 时在 Custom 对话框中选定 Visual Basic 6.0 Product Samples 来安装这些实例源程序，从而帮助学习 VB。由于安装 MSDN 的版本不同，实例源程序所在的路径也不同。例如，笔者自身计算机提取的路径为 Program Files\Microsoft Visual Studio\MSDN98\98VS\2052\SAMPLES\VB98，该文件夹下的实例源程序内容介绍如表 1.2 所示。

表 1.2 实例文件夹的内容

工程名称	工程内容	工程名称	工程内容
ActXDoc	ActiveX 文档教程	HelloWorldRemoteAutomation	简单的远程自动化 (RemoteAutomation)
AXData	ActiveX 部件担当其他控件的数据源	Interface	使用 COM 单元模型资源分配算法
Coffee	创建和使用 ActiveX 部件	MessageQueue	企业消息
CtlPlus	创建 ActiveX 控件	PassthroughServer	简单的传递服务器



续表

工程名称	工程内容	工程名称	工程内容
DataAware	创建能够担当数据源或客户的类	PoolManager	客户向缓冲池管理器请求对象的指针
GeoFacts	演示在 VB 应用程序中 Excel 对象的使用	ATM	如何使用资源文件
Controls	演示了 TextBox、CommandButton 和 ImageShows 等控件的使用	CallDlls	调用动态链接库中的过程
CtlsAdd	在运行时向应用程序中添加控件	Errors	错误处理技术
Datatree	使用 TreeView、ListView 和 ProgressBar	MdiNote	构造简单的多文档界面应用程序及菜单的创建
ListCmbo	数据绑定到列表框和组合框	Optimize	优化技术
OleCont	OLE 容器控件	ProgWOb	用对象编程
RedTop	创建陀螺旋转的动画	SdiNote	构造简单的单文档界面应用程序。菜单和工具栏的创建
Visual Basic 6.0Term	用 MSComm 控件进行终端仿真	TabOrder	使用 VB 的扩展性模型重新设置指定窗体的 Tab 键次序
DataReport	演示新的数据报表设计器	VCR	如何使 VB 的类能够模拟真实世界的对象
Biblio	使用 Data 控件	Blanker	普通图形技术
BookSale	使用自动化服务器 (AutomationServer) 封装商务策略和规则的逻辑	Palettes	PaletteMode 设置; Picture 对象
FirstApp	使用 Data 控件和其他数据识别的控件	DhShowMe	DHTML 技术
MSFlexGd	使用 MSFlexGrid 控件	PropBag	保存 HTML 页之间的状态值
Visdata	DAO 技术	Wcdemo	WebClass 演示
Callback	由服务器初始化回调到客户端		

### 1.7.3 使用 VB 的帮助菜单

在程序开发的时候,用户会遇到很多难题或者疑问,利用 VB 的帮助系统就可以解决很多开发中的问题。下面首先介绍一下 VB 的帮助菜单。VB 的帮助菜单如图 1.41 所示。



图 1.41 帮助菜单的相关信息



## 1.8 创建第一个 VB 程序

国外文献 6.8.1

教学录像：光盘\TM\lx\1\创建第一个 VB 程序.exe

前面介绍了很多关于 VB 的知识，下面通过一个小例子来使读者了解一下 VB 应用程序的开发流程。

例 1.1 创建第一个 VB 程序。下面以创建一个 Hello VB 的程序为例进行介绍。程序的执行流程为：运行程序，单击“确定”按钮，在窗体的标签中显示出 Hello VB 的信息；单击“退出”按钮，退出程序。（实例位置：光盘\TM\sl\1\1）

### 1.8.1 创建工程文件

选择“文件”/“新建工程”命令，弹出“新建工程”对话框。选择“标准 EXE”图标，单击“确定”按钮，如图 1.42 所示，即可创建一个标准的 EXE 工程。



图 1.42 新建工程

### 1.8.2 设计界面

在工程创建以后，会自动创建一个新窗体，命令为 Form1。在该窗体上添加一个 Label 控件，两个 CommandButton 控件，具体的摆放位置如图 1.43 所示。

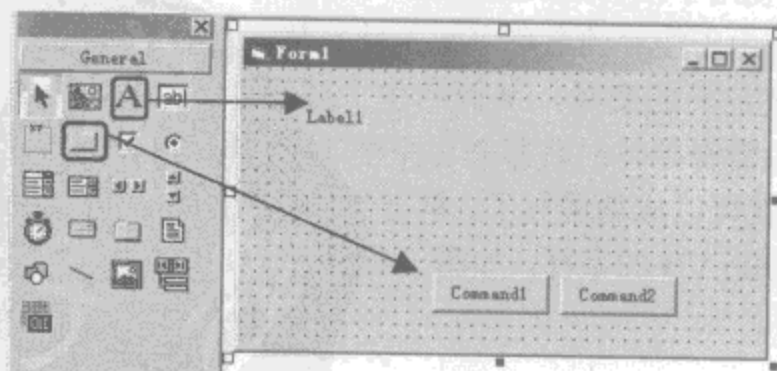


图 1.43 设置窗体界面

### 1.8.3 编写代码

进入到代码编辑器中，编写代码。本程序中需要在窗体的加载事件，在 Command1 和 Command2 的单击事件下面编写代码，具体的代码形式如下：

```
Private Sub Command1_Click()
    Label1.Caption = "Hello VB"
End Sub
Private Sub Command2_Click()
    End
End Sub
Private Sub Form_Load()
    Me.Caption = "第一个 VB 应用程序"
    Label1.Font = "宋体"
    Label1.FontSize = 32
    Label1.FontBold = True
    Command1.Caption = "确定"
    Command2.Caption = "退出"
End Sub
```

'设置标签内容  
'退出程序  
'设置窗体的标题栏  
'设置标签的字体  
'设置标签字体的大小  
'设置标签文字为粗体  
'设置 Command1 按钮文字  
'设置 Command2 按钮文字

### 1.8.4 调试运行

程序编写完成以后，需要对程序进行调试和运行。在进行调试的时候，可能会出现如图 1.44 所示的变量未定义错误。

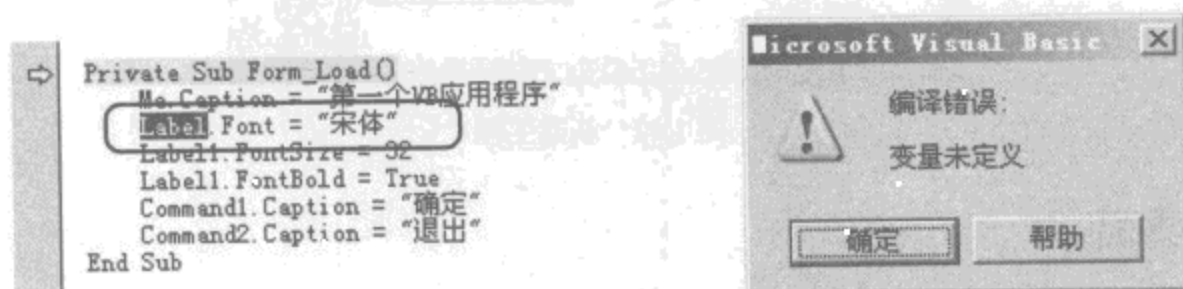


图 1.44 程序调试

产生该错误一般是由于使用了没有定义的变量。而在此处，光标停留在 Label 处，是由于控件的名称书写不够完整，使得系统以为是一个没有被定义的变量，从而产生上述错误。解决的方法非常简单，只需将控件的名称书写完整即可。

当程序没有错误以后，就可以成功运行了。单击“确定”按钮，在标签中即可显示 Hello VB 的字样，如图 1.45 所示。单击“退出”按钮退出程序。

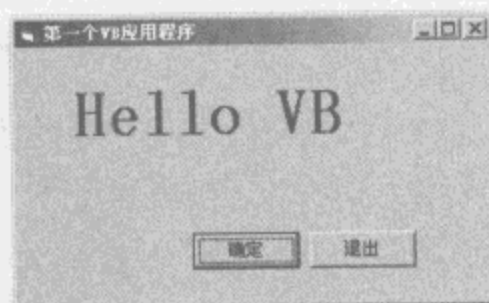


图 1.45 运行效果

### 1.8.5 保存工程

当程序调试运行成功以后,就可以将其保存起来。选择“文件”/“保存工程”命令,在打开的“另存为”对话框中,选择工程的保存路径,然后单击“确定”按钮,首先将扩展名为.frm 的窗体文件保存,再保存扩展名为.vbp 的工程文件。如图 1.46 所示。

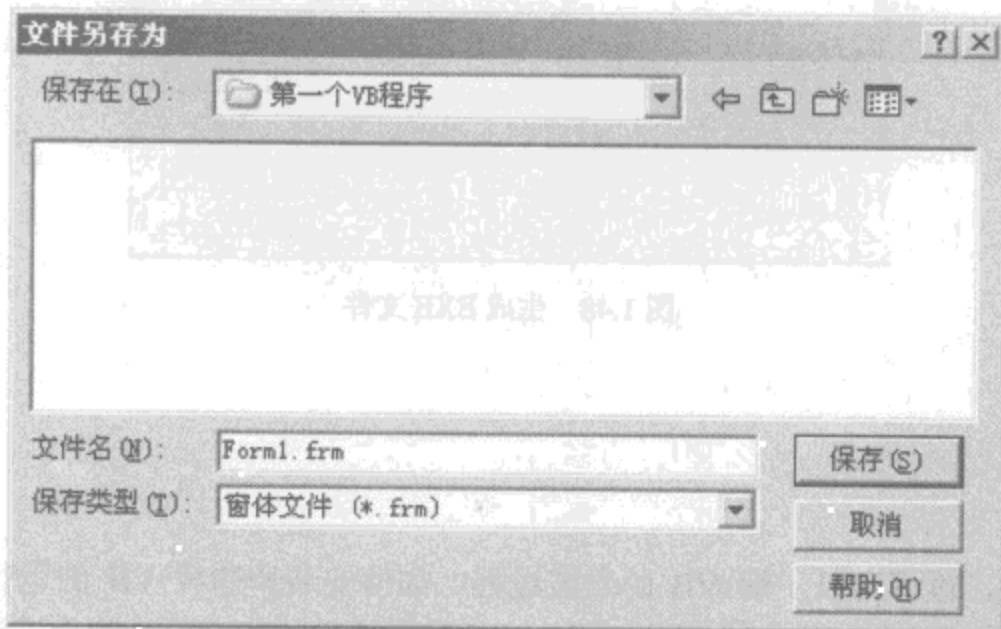


图 1.46 保存工程

当工程保存完成以后,在安装了 VSS 的系统中还会弹出如图 1.47 所示的对话框。由于本程序比较简单,就不需要进行版本控制了,只需单击 No 按钮即可完成工程的保存。

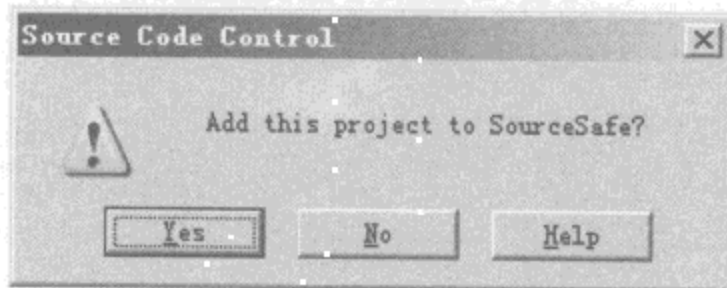



图 1.47 VSS 提示对话框

### 1.8.6 编译程序

程序保存完成以后,需要将已经编写好的程序编译成 EXE 可执行文件,以方便在其他的计算机上运行。具体的方法如下:

选择“文件”/“生成工程 1.exe”命令,在弹出的“生成工程”对话框中输入要生成的 EXE 文件名称,如这里为“第一个 VB 程序.exe”,如图 1.48 所示。

 **说明:** 在生成可执行文件之前,选择“工程”/“工程 1 属性”命令,将打开工程属性对话框。在该对话框中,选择“通用”选项卡,可以设置工程的启动对象和工程名称。在“生成”选项

卡中,可以修改程序的版本号、应用程序标题、版本信息等。

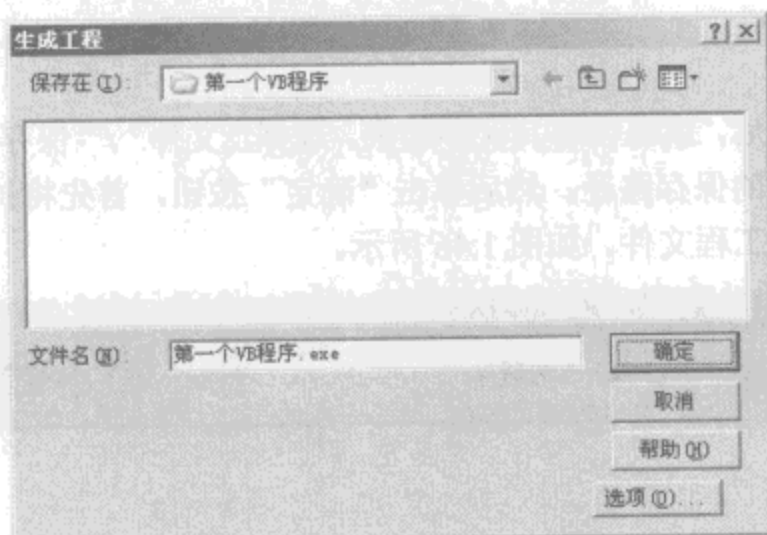


图 1.48 生成 EXE 文件

## 1.9 小结

通过本章的学习,读者可以了解 VB 的发展过程、如何安装和卸载 VB 的运行环境以及如何安装 SP6 补丁。本章重点介绍了 VB 的集成开发环境,对开发环境中的各个要素都作了详细的介绍,力求读者达到熟练掌握 VB 集成开发环境的目的。最后通过一个简单的 VB 应用程序,使读者了解 VB 程序开发的整个过程。

# 第 2 章

## VB 语言基础

( 教学录像: 31 分钟 )

VB 是一个功能强大、使用方便、容易上手的工具。使用它编写应用程序，熟悉 VB 开发环境是第一步。掌握 VB 语言是基础，只有打好基础，循序渐进地学习，实际运用时才会得心应手。本章主要介绍组成 VB 语言的基本元素，包括关键字和标识符、数据类型、变量和常量、运算符和表达式以及代码编写规则。

通过阅读本章，您可以：

- » 了解 VB 的关键字和标识符
- » 掌握基本数据类型，了解记录类型和枚举类型
- » 掌握变量和常量
- » 掌握运算符和表达式
- » 掌握代码编写规则



## 2.1 关键字和标识符

 教学录像：光盘\TM\lx\2\关键字和标识符.exe

关键字和标识符是 VB 代码中的一部分。关键字是指系统使用的具有特定含义的字符（如定义变量时使用的 Dim 语句），不能用作其他用途。常用的关键字有 Dim、Private、Sub、Public、End、If、Else、Form、Me、Single、As、Integer、Unload、Do、While、MessageBox 等。

在 VB 中所有的常量、变量、模块、函数、类、对象及其属性等都有各自的名称，这些名称就是标识符。

在一个 VB 工程中有如下标识符：

- ☒ 工程 1：表示一个工程的标识符。
- ☒ Form1：表示一个窗体的标识符。
- ☒ Class1：表示一个类模块的标识符。
- ☒ Module1：表示一个模块的标识符。

## 2.2 数据类型

 教学录像：光盘\TM\lx\2\数据类型.exe

“数据”是信息在计算机内的表现形式，也是程序的处理对象。不同类型的数据有不同的操作方式和不同的取值范围。VB6 具有系统定义的基本数据类型，这种基本数据类型是 VB6 中数据结构的基本单元。VB6 还有两种完全不同的数据类型：记录类型和枚举类型，它们的名称及数据项由用户任意定义。这两种数据类型使得 VB6 中的数据类型得以扩展。数据类型的分类如图 2.1 所示。

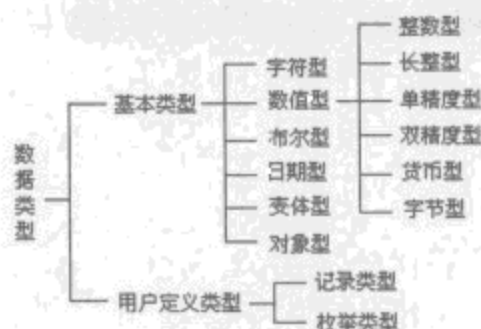


图 2.1 数据类型分类图

### 2.2.1 基本数据类型

VB 提供的常用基本数据类型有字符型、数值型、布尔型和日期型。对于数值型数据，考虑到运算效率、所占空间及精度要求，又分为整数型、长整型、单精度型、双精度型、货币型和字节型。另外，还有变体型和对象型，这两种数据类型是实际编程过程中不常使用的。

有关基本数据类型的大体介绍如表 2.1 所示，其中加粗内容是常用的数据类型。

表 2.1 基本数据类型

数据类型		类型名称	类型声明符	存储空间	前缀	值的有效范围
字符型	变长字符型	String	\$	10 字节加字符串长度	str	0 个至大约 20 亿个字符
	定长字符型		\$	字符串长度	str	1 个至大约 65400 个字符
数值型	整型	Integer	%	2 个字节	int	-32768~32767
	长整型	Long	&	4 个字节	lng	-2147 483648~2147483647
单精度型		Single	!	4 个字节	sng	-3.402823E38~-1.4011298E-45; 1.401298E-45~3.402823E38
双精度型		Double	#	8 个字节	dbl	±4.94D-324~±1.79D308
货币型		Currency	@	8 个字节	cur	-922337203685477.5808~ 922337203685 4775807
字节型		Byte	无	1 个字节	bty	1~255
布尔型		Boolean	无	2 个字节	bln	True 或 False
日期型		Date	无	8 个字节	dtm	100 年 1 月 1 日~9999 年 12 月 31 日
对象型		Object	无	4 个字节	obj	任何对象引用
变体型		Variant	无	按需分配	vnt	又称通用类型, 是上述有效范围之一

下面详细介绍常用的 4 种数据类型。

### 1. 字符型

如果一个变量或常量包含字符串, 就可以将其声明为字符型, 即 String 类型。字符串是用双引号括起来的若干个字符。字符串中的字符可以是计算机系统允许使用的任意字符。例如, 以下都是合法的 VB 字符串:

```
"VB"
"WelcometoChangchun"
"吉林省长春市"
"1+1=? "
"8888"
"*****"
"" (空字符串)
```

了解了字符串, 下面再来了解一下在 VB 中如何声明字符型变量。

例 2.1 声明一个字符型变量 A, 代码如下。(实例位置: 光盘\TM\sl\2\1)

```
Private A As String
```

然后将字符串“吉林省长春市”赋予这个变量, 并用字符串函数 Right 取右边 3 个, 最后输出, 代码如下。

```
Private Sub Form_Load()
```

```

A = "吉林省长春市"      '给字符型变量 A 赋值
A = Right(A, 3)           '用 Right 函数取右边 3 个
MsgBox A                  '用 MsgBox 函数输出字符型变量 A
End Sub

```

按〈F5〉键，运行程序，结果为：长春市

按照默认规定，String 变量或参数是一个可变长度的字符串，随着对字符串赋予新数据，它的长度可增可减。但也可以声明固定长度的字符串，语法如下。

**String \* size**


例 2.2 将例 2.2 中的变量 A 改为固定长度为 4 的字符型变量，代码如下。(实例位置：光盘\TM\sl\2\2)

```

Private A As String*4      '声明一个固定长度为 4 的字符型变量

```

此时将字符串“吉林省长春市”赋予这个变量，用 MsgBox 函数输出，结果为“吉林省长”。这说明如果赋予字符串的长度大于 4，就不是定长字符串了，VB 会直接截去超出部分的字符。反之，如果赋予字符串的长度小于 4，则 VB 会用空格将变量 A 不足部分填满。

 **说明：**标准模块中的定长字符串用 Public 或 Private 语句声明。在窗体和类模块中，必须用 Private 语句声明定长字符串。


## 2. 数值型

VB 支持 6 种数值型数据类型，分别是整型 (Integer)、长整型 (Long)、单精度浮点型 (Single)、双精度浮点型 (Double)、货币型 (Currency) 和字节型 (Byte)。

如果知道变量总是存放整数 (如 88) 而不是带小数点的数字 (如 88.88)，就应当将它声明为 Integer 类型或 Long 类型。因为整数的运算速度较快，而且比其他数据类型占据的内存少。在 For...Next 循环语句 (将在第 3 章介绍) 中作为计数器变量使用时，整数类型尤其重要。

浮点数值可表示为 mmmEeee 或 mmmDeee 形式，其中 mmm 是底数，而 eee 是指数 (以 10 为底的幂)。用 E 将数值文字中的底数部分或指数部分隔开，表示该值是 Single 类型；同样，用 D 则表示该值是 Double 类型。

货币类型 (Currency) 的数值保留小数点后面 4 位和小数点左面 15 位，适用于金额计算。

 **说明：**所有数值型变量都可以相互赋值。但浮点型或货币型数值赋予整型变量时，VB 会自动将该数值的小数部分四舍五入之后去除，而不是直接去除。

例如：

```

Dim i As Integer
i = 2.6873453453
MsgBox i

```

输出结果为 3。

## 3. 布尔型

若变量的值只是 True/False、Yes/No、On/Off 等信息，则可将其声明为布尔型，其默认值为 False。

例如，定义一个布尔型变量，输出该变量，代码如下。

```
Dim mybln As Boolean
MsgBox mybln
```

输出结果为 False。

#### 4. 日期型

日期型变量用来存储日期或时间。可以表示的日期范围为 100 年 1 月 1 日到 9999 年 12 月 31 日，时间则是从 0:00:00 到 23:59:59。日期常数必须用“#”符号括起来。如果变量 mydate 是一个日期型变量，可以使用下面的几种格式为该变量赋值。

```
mydate=#2/4/1977#
mydate=#1977-02-04#
mydate=#77,2,4#
mydate=#February 4,1977#
```

以上表示的都是 1977 年 2 月 4 日，并且无论在代码窗口中输入哪条语句，VB 都将其自动转换为第一种形式，即 mydate=#2/4/1977#。

例 2.3 另外，赋予日期/时间变量的值与输出的日期/时间格式不一定一致，这与系统区域和语言选项中的设置有关。例如：

```
Private Sub Form_Load()
    Dim mydate As Date
    mydate = #2/4/1977#
    MsgBox mydate
End Sub
```

'定义日期型变量  
'给变量赋值  
'输出变量

上述代码输出结果为 77-02-04，原因是系统区域和语言选项中的日期格式为 yy-mm-dd。

### 2.2.2 记录类型

2.2.1 节介绍的各种数据类型是由系统设定的，下面介绍的数据类型将由用户自定义。用户自定义类型，也称记录类型，主要通过 Type 语句来实现，其语法格式如下。

```
[Private | Public] Type 数据类型名
    数据类型元素名 As 类型名
    数据类型元素名 As 类型名
    ...
End Type
```

数据类型名是要定义的数据类型的名字；数据类型元素名不能是数组名；类型名可以是任何基本数据类型，也可以是用户定义的类型。

#### 说明：

- (1) Type 语句只能在模块级使用。使用 Type 语句声明了一个记录类型后，就可以在该声明范围内的任何位置声明该类型的变量。可以使用 Dim、Private、Public、ReDim 或 Static 语句来声明记录类型中的变量。



- (2) 在标准模块中, 记录类型按默认设置是公用的, 可以使用 Private 关键字来改变其可见性。而在类模块中, 记录类型只能是私有的, 且使用 Public 关键字也不能改变其可见性。
- (3) 在 Type...End Type 语句块中不允许使用行号和行标签。
- (4) 用户自定义类型经常用来表示数据记录, 该数据记录一般由多个不同数据类型的元素组成。

**例 2.4** 下面将使用 Type 语句声明一个新的数据类型 Sell, 然后为该类型中的各个元素赋值, 最后输出, 具体实现过程如下。(实例位置: 光盘\TM\sl\2\3)

- (1) 创建一个 VB 工程, 在该工程中添加一个模块, 在该模块的声明部分编写如下代码。

```
Type Sell
    name As String * 20
    standard As String * 10
    price As Currency
End Type
```

- (2) 在窗体的 Form\_Load 事件过程中声明一个 Sell 类型 mySell, 然后为其各个元素赋初值, 最后输出, 代码如下。

```
Private Sub Form_Load()
    Dim mySell As Sell
    mySell.name = "Epson 打印机"
    mySell.standard = "Epson Style C65"
    mySell.price = 450
    MsgBox "产品名称: " & mySell.name & Chr(10) & "产品型号: " & mySell.standard & Chr(10) & "单价: " & mySell.price
End Sub
```

按〈F5〉键, 运行程序, 结果如图 2.2 所示。



图 2.2 输出打印机相关信息

### 2.2.3 枚举类型

枚举是为了一组整数值提供便于记忆的标识符, 它的作用是管理和使用常量。枚举类型主要使用 Enum 语句来定义, 语法如下。

```
[Private | Public] Enum 数据类型名
    数据类型元素名 = 整型常数表达式
    数据类型元素名 = 整型常数表达式
    ...
End Enum
```

其中的整型常数表达式可以是默认的, 默认情况下, 第一个数据类型元素取值从 0 开始, 其余数



据类型元素名依次为 1, 2, 3, 4, 5, ..., 枚举类型的实质就是定义一个符号常量集, 并用一个名称表示该集合。

**例 2.5** 下面用 Enum 语句定义一个颜色类型, 其中包括一些颜色常数, 可以用于设计数据库的数据输入窗体, 代码如下。(实例位置: 光盘\TM\sl\2\4)

```
Public Enum myColors
    myRose = &HE1E4FF
    myGray = &H908070
    myBlue = &HFF901E
    mySkyBlue = &HFFBF00
    mySpringGreen = &H7FFF00
    myForestGreen = &H228B22
End Enum
```

## 2.3 变量

 教学录像: 光盘\TM\lx\2\变量.exe

前面介绍数据类型的同时, 已经简单地涉及了一些变量, 这一节我们就详细介绍一下变量的概念及声明、变量的命名规则、变量的分类以及使用变量时的注意事项。

### 2.3.1 什么是变量

一个变量相当于一个容器, 这个容器对应着计算机内存中的一块存储单元, 因此, 它可以保存数据。下面通过举例进一步说明变量。

**例 2.6** 有两个存储单元, 分别为 strUser 和 strPassword, 存放的值分别为“管理员”、“111”, 如图 2.3 所示。

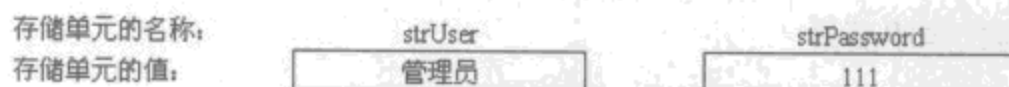


图 2.3 存储管理员

也可以将这两个存储单元的值改为“普通用户”、“222”, 如图 2.4 所示。

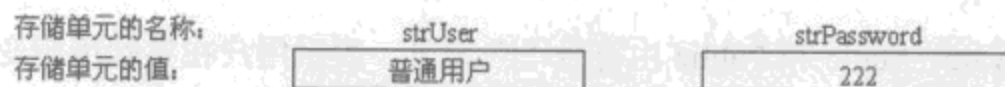


图 2.4 存储普通用户

综上所述, 变量也就是在程序运行时, 其值在不断发生改变的量, 它在程序设计中是一个非常重要、非常关键的内容。

### 2.3.2 变量的命名


为了便于在程序中区分和使用变量，必须给每一个变量命名。在 Visual Basic 中，变量的命名要遵循以下规则。

(1) 变量名只能由西文字母、汉字、数字及下划线组成。

(2) 变量名必须以西文字母或汉字开头，最后一个符号可以是数据类型声明符，如 Dim a%，更多的数据类型声明符可以参考表 2.1。

(3) 变量名长度可达 255 个字符，有效字符为 40 个。

(4) VB 中的关键字不能作为变量名。例如，Print、Dim 和 For 等都是非法变量名。

 说明：虽然 VB 中的关键字不能作为变量名，但可将关键字嵌入变量名中。例如，print 是非法变量名，但 print\_3 或 print3 都是合法的变量名。

(5) 不能在变量名中出现标点符号、空格或者嵌入！、@、#、\$、%、&等字符。

(6) 变量名在变量有效的范围内必须是唯一的，否则会出现“当前范围内的声明重复”的错误。

(7) 变量名中不区分大小写。

以上是变量的基本命名规则，在实际编程过程中，笔者建议变量名应能对变量的含义具有一定的提示作用，且能反映变量类型及变量作用域，这样可以增强程序代码的可读性。例如，可以将用来保存产品名称的变量命名为 strName，保存产品价格的变量命名为 curPrice，保存用户名的全局变量命名为 gstrUserName。

### 2.3.3 变量的声明

在 VB 程序中，使用变量前，一般要先声明变量的名称和变量的数据类型，以决定系统为变量分配的存储单元。下面介绍几种方式来声明变量及其数据类型。

#### 1. 用声明语句显式声明变量

使用声明语句声明变量，也称显式声明，语句格式如下：

```
Dim|Private|Static|Public 变量名 As 数据类型
```

其中，变量名必须符合变量的命名规则，数据类型可以是 VB 的基本数据类型，也可以是记录类型或枚举类型。

关键字 Dim、Private、Static 和 Public 由符号“|”隔开，表示用户在实际声明变量中，可以从中任选其一。但选用不同的关键字，在程序的不同位置所定义的变量的种类和使用范围是不同的，这就是 2.3.4 节将要介绍的内容。

**例 2.7** 下面通过几个例子，介绍如何显式声明变量。

声明一个字符串变量、一个整型变量：

```
Dim Str As String
Dim Int As Integer
```

使用数据类型的类型符号来替代 As 子句:

```
Dim Str$
Dim Int%
```

 注意: 变量名与数据类型符之间不能有空格。

一条 Dim 语句可声明多个变量, 各变量之间以逗号隔开。

```
Dim Str As String, Int As Integer, Sng As Single
```

声明指定字符串长度的字符型变量。

```
Dim Str1 As String*128
```

如果赋给字符串变量 Str1 的字符少于 128 个, 则用空格填充变量 Str1; 如果赋给字符串变量 Str1 的字符大于 128, 则 Visual Basic 会自动截去超出部分的字符。

## 2. 隐式声明变量

在 VB 中, 也可以不事先声明而直接使用变量, 这种方式称为隐式声明。所有隐式声明的变量都是变体型 (Variant) 数据类型, 这一类型在上一节已有所介绍。

例 2.8 声明一个变量 a, 并为 a 赋值, 代码如下。

```
Dim a
a=111
```

或直接使用:

```
a = 111
```

## 3. 强制声明变量 (Option Explicit 语句)

前面介绍了变量的两种声明方式, 其中隐式声明显然用起来很方便, 但如果变量名拼错了, 系统就会认为它是另一个新的变量, 从而引起潜在的错误。这时如果设置了强制声明变量, 就不会出现这种情况了。因为强制声明变量会在声明段手动或自动地加入 Option Explicit 语句, 如果程序中存在直接使用的变量, 运行程序时, 系统就会提示“变量未定义”。

下面介绍如何强制声明变量。强制声明变量可以在声明段手动添加 Option Explicit 语句, 但这种方法很费时。下面这种方法可以自动在声明段添加 Option Explicit 语句, 方法是: 选择“工具”/“选项”菜单命令, 在“选项”对话框中, 单击“编辑器”选项卡, 选择“要求变量声明”复选框, 如图 2.5 所示, 此时 VB 会在以后的窗体模块、标准模块及类模块中的声明段自动地插入 Option Explicit 语句, 如图 2.6 所示。但不会将它加入现有的模块中, 要想在现有的模块中加入 Option Explicit 语句, 还需使用第一种方法, 也就是在声明段手动添加 Option Explicit 语句。

 说明: 如果要强制声明变量, 建议在程序设计的开始就在“选项”中设置“要求变量声明”。

## 4. 用 DefType 语句声明变量

用 DefType 语句可以在标准模块或窗体模块的声明部分定义变量, 语句的格式为:

```
DefType 字母范围
```

其中, Def 是保留字; Type 是数据类型标志, 它可以是 Int (整型)、Lng (长整型)、Sng (单精度型)、Dbl (双精度型)、Cur (货币型)、Str (字符型)、Byte (字节型)、Bool (布尔型)、Date (日期型)、Obj (对象型)、Var (变体型)。把 Def 和 Type 写在一起就构成了定义的类型关键字。字母范围用“字母-字母”的形式给出, 例如:

DefLng i-l

'凡是变量名以字母 i 到 l 开头的变量均定义为长整型

#### 注意:

- (1) DefType 语句只对它所在的模块起作用。
- (2) 当使用 DefType 语句和使用类型说明符方式定义变量发生矛盾时, 类型说明符定义变量的方式总是比 DefType 语句优先起作用。

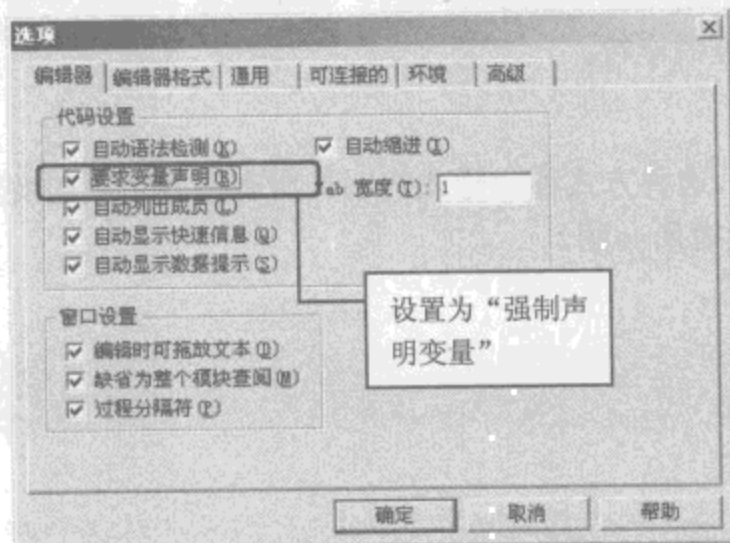
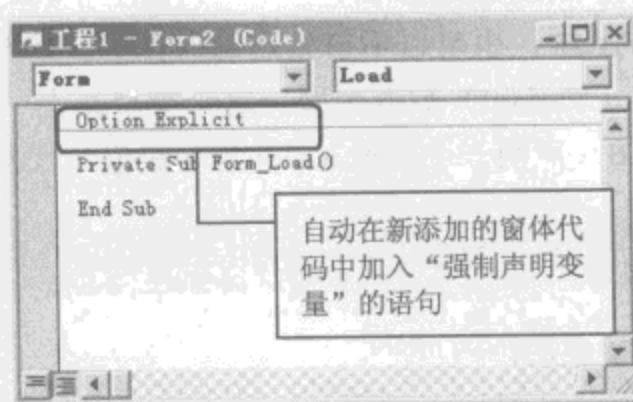


图 2.5 设置强制声明变量



### 2.3.4 变量的作用域

一个变量被声明后, 并不是在任何地方都能使用。每个变量都有它的作用范围, 也就是作用域。例如在一个过程内部声明的变量, 只在该过程内部有效, 一个模块的通用声明部分声明的变量, 只在该模块内的所有过程有效, 而对于使用 Public 语句声明的变量, 不仅对于同一模块内的所有过程有效, 甚至对于整个应用程序的所有过程也都是有效的。

在 VB 中允许在声明变量时指定它的范围, 主要包括局部变量、模块级变量和全局变量, 详细介绍如表 2.2 所示。

表 2.2 变量的作用域

变量作用域	声明语句	有效位置	有效范围	举 例
局部变量	Dim 或 Static	在过程内部	过程内部	Private Sub Form_Load() Dim intNumber As Integer End Sub
模块级变量	Dim 或 Private	模块的通用声明段	模块内的所有过程	Dim intNumber As Integer
全局变量	Public 或 Global	在标准模块 (.bas) 的声明段	整个工程的任何模块中都有效	Public intNumber As Integer



**例 2.9** 下面通过实例比较局部变量和模块级变量的作用范围。创建一个 VB 工程，在窗体的 Form\_Load 的事件过程中定义一个整型变量，并为其赋值，然后在文本框控件中的值发生改变时，显示该变量的值，代码如下。（实例位置：光盘\TM\sl\2\5）

```
Private Sub Form_Load()
    Dim intNumber As Integer
    intNumber = 2
End Sub
Private Sub Text1_Change()
    Dim intNumber = intNumber + 1
    MsgBox intNumber
End Sub
```

按〈F5〉键，运行程序，结果为 1。由于变量 intNumber 是在 Form\_Load 事件过程中定义的，因此决定它是一个局部变量，只在过程内部有效，而 Text1\_Change 事件过程中的 intNumber 变量被当作一个新的变量，其初值为 0，加 1 后结果为 1。

将上述代码中的 Dim intNumber As Integer 语句移到窗体的通用声明部分，运行程序，其结果为 3，由于变量 intNumber 已被放在窗体的通用声明部分，因此决定它是一个模块级变量，即在模块内的所有过程有效。Form\_Load 事件过程中变量 intNumber 的值为 2，在 Text1\_Change 事件过程变量 intNumber 再加上 1，那么结果就是 3。

### 2.3.5 静态变量

在过程中，既可以使用 Dim 语句声明局部变量，也可以使用 Static 语句声明局部变量，并且 Static 语句的一般形式与 Dim 语句相同：

**Static 变量名 As 数据类型**

使用 Static 语句声明的变量称为静态变量，它与用 Dim 语句声明的变量的不同之处在于：当一个过程结束时，过程中所用到的静态变量的值会保留，下次再调用该过程时，变量的初值是上次调用结束时保留的值。

对于使用 Dim 语句声明的局部变量，随过程的调用而分配存储单元，并进行变量的初始化。一旦过程结束，变量的内容自动消失，占用的存储单元也被释放。因此，每次调用过程时，变量都将重新初始化。

### 2.3.6 变量同名问题的处理

如果不同模块中的公用变量使用同一名字，则通过同时引用模块名和变量名在代码中进行区分。例如，如果在 Form1 和 Module1 中都声明了一个整型公用变量 intNumber，则可以用 Form1.intNumber 和 Module1.intNumber 来区分这两个同名的变量。

**例 2.10** 下面通过实例进一步说明同名变量在程序中是如何区分的。（实例位置：光盘\TM\sl\2\6）

(1) 新建一个工程，在工程中添加 3 个窗体，默认名称为 Form1、Form2 和 Form3。



(2) 在 Form2 中声明第一个变量 intX, 然后在 Form\_Load 事件过程中设置它的值。

```
Public intNumber As Integer           '声明 Form2 的 intX
Private Sub Form_Load()
    intNumber = 1                     '设置 Form2 的 intX 变量的值
End Sub
```

(3) 在 Form3 中声明第二个变量 intX, 它与 Form2 中的变量名字相同, 同样在 Form\_Load 事件过程中设置它的值。

```
Public intNumber As Integer           '声明 Form3 的 intX
Private Sub Form_Load()
    intNumber = 2                     '设置 Form3 的 intX 变量的值
End Sub
```

(4) 在 Form1 窗体中添加两个按钮, 在它们的 Click 事件过程中编写代码时分别调用 Form2 和 Form3 窗体, 并将这两个窗体模块中定义的变量用 MsgBox 显示。

```
Private Sub Command1_Click()
    Form2.Show                        '调用 Form2 窗体
    MsgBox Form2.intNumber            '显示 Form2 窗体的 intX
End Sub
Private Sub Command2_Click()
    Form3.Show                        '调用 Form3 窗体
    MsgBox Form3.intNumber            '显示 Form3 窗体的 intX
End Sub
```

按〈F5〉键, 运行程序, 单击两个命令按钮中的每一个按钮, 将看到两个公用变量被分别引用。

## 2.4 常量

 教学录像: 光盘\TM\lx\2\常量.exe

熟悉了变量, 接下来了解一下常量。常量与变量正好相反, 它是在程序设计时, 值始终不发生改变的量。本节就常量的声明及使用进行介绍。

### 2.4.1 常量的声明

当程序中有需要重复使用的常量时, 可以使用 Const 语句声明, 语法如下。

```
Const <常量名> [As <数据类型>] = <常量表达式>
```

**Public:** 可选的参数, 用于声明可在工程的所有模块的任何过程中使用这个常量。

**Private:** 可选的参数, 用于声明只能在包含该声明的模块中使用常量。

**常量名:** 必选的参数, 用于指定该常量名称, 必须是合法的 Visual Basic 标识符。

**数据类型:** 可选的参数, 也可以通过数据类型符号规定常量的类型。

常量表达式：必选的参数，包括常量和操作符，但不包含变量，而且计算结果总是常值。

例如：

```
Const PI As Single=3.14159265357
Print 3 * PI
```

'声明符号常量 PI 代替 3.14159265357  
'结果为：9.42477796071

⚠ 注意：在程序中如果改变已定义常量的值，则会出现错误提示。

## 2.4.2 局部常量和全局常量

在模块级的声明中 Public 和 Private 省略的情况下，系统默认是 Private。在模块中使用 Public 语句声明后的符号常量，就是一个全局常量，该常量可以在程序中所有模块的过程中使用。同样，用 Private 语句声明过的常量就是局部常量。

例如：

```
Const MyVar = 123
Public Const MyString = "mr"
Private Const MyInt As Integer = 5
Const MyStr = "mr", MyDouble As Double = 3.1415
```

'默认情况下常量是局部的  
'全局常量  
'声明局部整型常量  
'在一行中声明多个常量

⚠ 注意：全局常量必须在标准模块中声明。

## 2.5 运算符和表达式

📺 教学录像：光盘\TM\lx\2\运算符和表达式.exe

在进行程序设计时，经常会进行各种运算，那么就会涉及一些运算符，而表达式是运算符和数据连接而成的式子。本章将详细介绍运算符和表达式在程序中的应用。

### 2.5.1 运算符

在 VB 中有 4 种运算符，分别是算术运算符、字符串运算符、关系运算符和逻辑运算符。

#### 1. 算术运算符

算术运算符按照优先级从高到低，依次为指数运算 (^)，乘法 (\*) 运算、除法 (/) 运算、求余数运算 (Mod)、整除运算 (\)、加法 (+) 和减法 (-) 运算。其中整除运算只求运算结果的整数部分，例如在 VB 工程中的“立即”窗口中输出“5 除以 2”，代码如下。

```
Private Sub Form_Load()
    Debug.Print 5 \ 2
End Sub
```

结果为：2

算术运算符的基本用法相信您已经学会了，下面介绍使用算术运算符时需要注意的事项。

(1) 当指数运算 (^) 与负号 (-) 相邻时, 负号 (-) 优先。

(2) 运算符左右两边的操作数应是数值型数据, 如果是数字字符或逻辑型数据, 需要将它们先转换成数值数据后, 再进行算术运算。

(3) 在进行算术运算时, 不要超出数据取值范围。对于除法运算, 应保证除数不为零。

## 2. 关系运算符

关系运算符用于比较运算符左右两侧表达式之间的大小关系, 因此又称为比较运算符, 它的运算结果为布尔型数据, 即结果为 True 或者 False, 如果其中的任何一个表达式结果为 NULL, 则关系运算的结果还可以是 NULL。关系运算符没有优先级的不同, 因此在计算时, 按照它们的出现次序, 从左到右进行计算。VB 中的关系运算符有: 等于 (=)、大于 (>)、小于 (<)、大于等于 (>=)、小于等于 (<=) 和不等于 (<>)。

另外, 还要说明两点:

(1) 对于字符型数据的比较。

如果直接比较单个字符, 则比较两个字符的 ASCII 码值的大小; 而对于两个汉字字符, 则比较两个汉字字符的区位码。

如果比较两个字符串, 则从关系运算符的左边字符串的第 1 个字符开始, 逐一对右边字符串的对应位置上的字符进行比较 (即比较对应位置上的字符的 ASCII 码值), 其中 ASCII 码值较大的字符所在的字符串大。

常见的字符值的大小比较关系如下:

“空格” < “0” < ..... < “9” < “A” < ..... < “Z” < “a” < ..... < “z” < “所有汉字”

(2) 赋值号 “=” 与关系运算符 “=” 的区别。

在书写上它们没有区别, 只是含义与作用不同。赋值号 “=” 专用于给变量、对象属性、数组等赋值, 赋值号左边必须是变量名、对象属性、数组等, 不能为常量或表达式。

而关系运算符 “=” 用于比较两个表达式的值是否相等。关系运算符 “=” 的左右两边都可以是常量、变量或表达式。用关系运算符 “=” 连接形成的关系表达式不能单独作为一条语句出现在程序中, 它只能出现在其他语句或表达式中间。

例如:

```
x=10
```

```
y=10
```

```
z=(x+10=y-100)
```

其中, 前 3 个 “=” 都是赋值号, 第 3 行语句中括号内的 “=” 是关系运算符。

## 3. 连接运算符

连接运算符有两个, 它们是 “+” 和 “&”。其中, “&” 连接运算符用于强制将两个表达式作为字符串连接。而 “+” 连接运算符则与它不同: 当两个表达式都为字符串时, 将两个字符串连接; 如果一个字符串而另一个是数字, 则进行相加, 结果为两个数字相加的和。

下面举例说明连接运算符的用法。

```
a=2+3
```

```
a="2"+"3"
```

'a 值为 5

'a 值为 23

```
a="吉林省" & "长春市"
a="a1"+3
a="a1" & 3
```

'a 值为“吉林省长春市”  
'出现“类型不匹配”的错误提示信息  
'a 值为 a13

⚠ 注意：变量名与“&”之间一定要加一个空格。因为“&”本身还是长整型数据类型的类型符，不加空格容易出现视觉和理解上的误差。

#### 4. 逻辑运算符

逻辑运算符包含下列运算符，将它们按照运算优先级由高到低排列为：

逻辑非 (Not)、逻辑与 (And)、逻辑或 (Or)、逻辑异或 (Xor)、逻辑等于 (Eqv) 及逻辑蕴含 (Imp)。逻辑运算得出的结果是布尔型值，也就是 True 或 False。

### 2.5.2 表达式

前面我们简单地介绍了表达式是运算符和数据连接而成的式子，那么具体地说表达式就是由常量、变量、运算符、圆括号、函数等连接形成的一个有意义的运算式子。它包括算术表达式、字符串表达式、日期表达式、关系表达式和逻辑表达式。

在书写表达式时，应注意：

- (1) 表达式中所有符号都必须一个一个并排写在同一横线上，不能写成上标或下标的形式。例如，数学上的  $2^2$  在 VB 中要写成  $2^2$  的形式。
- (2) 不能省略乘号，乘号“\*”必须写。
- (3) 表达式中所有的括号一律使用圆括号，并且括号左右必须配对。
- (4) 数学表达式中表示特定含义的符号要写成具体的数值，如  $\pi$  要写成 3.1415926（根据精度取小数点后的位数）。

### 2.5.3 运算符的优先级

一个表达式中，通常包含一种或多种运算符，这时系统会按预先确定的顺序进行计算，这个顺序称为运算符的优先级。

算术运算符 → 连接运算符 → 关系运算符 → 逻辑运算符

📖 说明：各种运算符的优先级，已在讲解各运算符时介绍了，这里不再赘述。

## 2.6 代码编写规则

📺 教学录像：光盘\TM\lx\2\代码编写规则.exe

代码编写规则是养成良好编程习惯的基础。本节主要介绍代码编写规则，包括对象命名规则、代码书写规则、处理关键字冲突和代码注释规则。



## 2.6.1 对象命名规则

当为对象、属性、方法及事件命名时，应选择易于被用户理解的名字。名字含义越清晰，则代码的可用性越强。

这里的对象命名规则适用于：

- ☒ 对象。
- ☒ 组成对象接口的属性、方法及事件。
- ☒ 属性、方法及事件的命名的参数。
- ☒ 具体命名规则如下。

(1) 尽可能使用完整的单词或音节。

对用户来说，记住整个的单词比记住缩略词更容易，例如 Window 被缩略为 Wind、Wn 或 Wnd，不如 Window 本身好记。下面通过两个例子说明推荐使用的对象名称，如表 2.3 所示。

表 2.3 推荐使用的对象名称 (1)

用	不要 用	用	不要 用
Application	App	SpellCheck	SpChk

如果标识符太长而需要缩略时，则尽量用完整的首音节。例如，用 AltExpEval，而不用 Alternate ExpressionEvaluation 或 AltExpnEvln。

(2) 大小写混用。

所有标识符都应混用大小写，而不是用下划线来分割其中的单词。下面通过两个例子说明推荐使用的对象命名，如表 2.4 所示。

表 2.4 推荐使用的对象名称 (2)

用	不要 用
ShortcutMenus	Shortcut_Menus, Shortcutmenus, SHORTCUTMENUS, SHORTCUT_MENU
BasedOn	BasedOn

(3) 使用一致的术语。

使用与接口相同的单词，不要用诸如 HWND 之类的基于匈牙利命名法的标识符命名。记住，这些代码是要被其他用户访问的，因此尽量使用用户描述一个概念时可能会采用的单词。

(4) 集合类名使用正确的复数。


对集合采用复数而不用新的名称可以减少用户必须记忆项的数目。这样也简化了对集合的命名。下表列出了集合类名称的一些例子。

表 2.5 推荐使用的对象名称 (3)

用	不要 用	用	不要 用
Axes	Axiss	SeriesCollection	CollectionSeries
Windows	ColWindow		



例如,如果有一名为 Axis 的类,则 Axis 对象的集合存储在 Axes 类中。同样,Vertex 对象的集合存储在 Vertices 类中。极少情况下当单数和复数的拼写一样时,则在其后面添加一个 Collection,例如 SeriesCollection。

 **注意:** 此命名约定可能不适用于某些集合,尤其在一组对象存在于多个集合时。例如,Mail 程序可能有一个 Name 对象存在于多个集合中: ToList、CcList 等。在这种情况下,可以将这些独立的 name 集合命名为 ToNames 和 CcNames。

#### (5) 常数使用前缀。

选择三、四个小写字母组成标识部件的前缀,把它用在部件类型库中部件提供的常数名上,以及定义那些常数的 Enums 名上。

例如,提供贷款评估的代码可以使用 levs 作为前缀。下面贷款的枚举类型 Enum 使用了该前缀。(此外,这些常数包含大写字母 LT,以标识它们所属的枚举。)

```
Public Enum LoanType
    levsLTMortgage = 1
    levsLTCommercial
    levsLTConsumer
End Enum
```

#### (6) “动词/对象”和“对象/动词”。

如果创建的方法名是一个动词及其作用的对象名的组合,则次序必须保持一致。或者在所有情况下都将动词放在对象前面,如 InsertWidget 和 InsertSprocket; 或者总是将对象放在前面,如 WidgetInsert 和 SprocketInsert。

两种方法各有所长。“动词/对象”次序创建的名称更像日常说话,因而能更好地表示此方法的意图。而“对象/动词”的次序则便于将影响某一特定对象的所有方法集合到一起。

## 2.6.2 代码书写规则

代码书写规则如下:

☒ 可将单行语句分成多行。

可以在“代码”窗口中用续行符“\_”(一个空格后面跟一个下划线)将长语句分成多行。由于使用续行符,无论在计算机上还是打印出来的代码都变得易读。例如声明一个 API 函数,代码如下。

```
'声明 API 函数用于异步打开一个文档
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" _
    (ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, _
    ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
```

 **注意:** 在同一行内,续行符后面不能加注释。

☒ 可将多个语句合并写到同一行。

通常,一行之中有一个 Visual Basic 语句,而且不用语句终结符。但是也可以将两个或多个语句放在同一行,只是要用冒号“:”将它们分开。例如给数组连续赋值,其代码如下。

```
a(0) = 11: a(1) = 12: a(3) = 13: a(4) = 14: a(5) = 15: a(6) = 16
```

☒ 可在代码中添加注释。


以 Rem 或 “'”（半个引号）开头，VB 就会忽略该符号后面的内容。这些内容就是代码段中的注释，既方便开发者也为以后可能检查源代码的其他程序员提供方便。例如为下面的代码添加注释：

```
Dim a As String
```

'定义一个字符型变量

```
Dim a As String:
```

Rem 定义一个字符型变量

 注意：如果在语句行后使用 Rem 关键字，则必须在语句后使用冒号 “:” 与 Rem 关键字隔开，而且 Rem 关键字与注释文字间要有一个空格。

☒ 在输入代码时不区分大小写。

☒ 一行最多允许输入 255 个字符。

### 2.6.3 处理关键字冲突

在代码的编写中为避免 Visual Basic 中元素（Sub 和 Function 过程、变量、常数等）的名字与关键字发生冲突，它们不能与受到限制的关键字同名。

受到限制的关键字是在 Visual Basic 中使用的词，是编程语言的一部分。其中包括预定义语句（比如 If 和 Loop）、函数（比如 Len 和 Abs）和操作符（比如 Or 和 Mod）。

窗体或控件可以与受到限制的关键字同名。例如，可以将某个控件命名为 If。但在代码中不能用通常的方法引用该控件，因为在 Visual Basic 中 If 意味着关键字。例如，下面这样的代码就会出错。

```
If.Caption = "同意"
```

'出错

为了引用那些与受到限制的关键字同名的窗体或控件，就必须限定它们，或者将其用方括号 “[ ]” 括起来。例如，下面的代码就不会出错。

```
MyForm.If.Caption = "同意"
```

'用窗体名将其限定

```
[If].Caption = "同意"
```

'方括号起了作用

### 2.6.4 代码注释规则

注释是一种非执行语句，它不仅是对程序的解释说明，同时还对程序的调用起着非常重要的作用。如利用注释来屏蔽一条语句，当程序再次运行时，可以发现问题或错误。这样大大提高了编程速度，减少了不必要的代码重复。代码注释规则如下：

- (1) 程序功能模块部分要有代码注释，简洁明了地阐述该模块的实现功能。
- (2) 程序或模块开头部分要有以下注释：模块名、创建人、日期、功能描述等。
- (3) 在给代码添加注释时，尽量使用中文。
- (4) 用注释来提示错误信息以及出错原因。

下面介绍几种注释的方法。

## 1. 利用代码或语句添加注释

在 Visual Basic 中使用 “'” 符号或 “Rem” 关键字, 可以为代码添加注释信息。“'” 符号可以忽略掉后面的一行内容, 这些内容是代码段中的注释。这些注释主要是为了以后查看代码时, 帮助用户快速理解该代码的内容。注释可以和语句在同一行出现, 并写在语句的后面, 也可独自占据一整行。

(1) 注释占据一行, 在需要解释的代码前。

```
'为窗体标题栏设置文字
Me.caption="明日科技"
Rem 在文本框中放欢迎词
Text1.Text = "欢迎您使用本软件!!!"
```



(2) 注释和语句在同一行并写在语句的后面。

```
Me.caption="明日科技"           '为窗体标题栏设置文字
Text1.Text = "欢迎您使用本软件!!!"; Rem 在文本框中放欢迎词
```

(3) 注释占据多行, 通常用来说明函数、过程等的功能信息。通常在说明前后使用注释和 “=”、“\*” 符号强调。例如下面的代码:

```
'=====
'名称: CalculateSquareRoot
'功能: 求平方根
'日期: 2008-03-02
'单位: mingrisoft
'=====
Function CalculateSquareRoot(NumberArg As Double) As Double
    If NumberArg < 0 Then           '评估参数
        Exit Function              '退出调用过程
    Else
        CalculateSquareRoot = Sqr(NumberArg) '返回平方根。
    End If
End Function
```

## 2. 利用工具栏按钮为代码添加注释

为了方便对大段程序进行注释, 可以通过选中两行或多行代码, 并在 “编辑” 工具栏上单击 “设置注释块” 按钮  或 “解除注释块” 按钮  来对大段代码块添加或解除注释 “'” 符号。设置或取消连续多行的代码注释块的步骤如下:

(1) 在工具栏上单击鼠标右键, 在弹出的菜单中选择 “编辑” 命令, 将 “编辑” 工具栏添加到窗体工具栏中。

(2) 选中要设置注释的代码, 然后单击 “编辑” 工具栏中的 “设置注释块” 按钮, 如图 2.7 所示。也可以将光标放置在需要注释的代码所在行, 单击 “设置注释块” 按钮即可。



图 2.7 编辑工具栏

例 2.11 下面使用注释块注释代码。选中需要注释的代码，单击“设置注释块”按钮，即可将选中的代码全部注释，注释后的效果如下。

```
'Private Sub Command1_Click()  
' Command2.Enabled = True  
' Command1.Enabled = False  
'End Sub
```

“解除注释块”按钮与“设置注释块”按钮功能正好相反，主要用于清除选中代码前的“'”符号，从而解除该代码块的注释。

注意：在使用注释符号“'”时，不能将注释符号“'”放在“\_”续行符之后。

## 2.7 小结

本章主要介绍了关键字、标识符、数据类型、变量、常量、运算符、表达式和代码编写规则。读者应重点掌握变量，它是程序设计的关键内容，另外，良好的编程习惯也是必不可少的。

## 2.8 练习与实践

1. 在 VB 中，对于没有赋值的变量，系统默认值是什么？（答案位置：光盘\TM\sl\2\7）
2. 在程序中声明一个常量，然后试着更改这个常量的值，看会引发什么错误。（答案位置：光盘\TM\sl\2\8）
3. 设  $A=5$ ,  $B=4$ ,  $C=3$ ，求下列表达式的值，并将其输出到“立即”窗口。（答案位置：光盘\TM\sl\2\9）
  - (1)  $A+3*C$
  - (2)  $A^2/6$
  - (3)  $A/2*3/2$
  - (4)  $A \text{ Mod } 3+B^3/C^5$
4. 定义一个变量 myval，将文本框中的值赋给变量 myval，然后使用 MsgBox 函数显示该变量。（答案位置：光盘\TM\sl\2\10）



# 第 3 章

## 算法和程序控制结构

(教学录像: 1 小时 12 分钟)

算法是问题求解过程的精确描述, 因此掌握算法是学习程序设计的核心, 它可以帮助用户更好、更快地掌握编程思想及编程方法。本章将简要地介绍算法, 使读者初步了解算法, 为日后编程打下良好的基础。学好程序的基本控制结构是结构化程序设计的基础, 本章详细地介绍了程序常用的三种控制结构, 讲解过程中为了便于读者理解, 结合了大量的举例。

通过阅读本章, 您可以:

- ▶▶ 了解算法基本概念、特性及算法的几种描述方法
- ▶▶ 掌握顺序结构程序设计方法中的基本语句、输入/输出语句
- ▶▶ 掌握赋值语句中“=”的使用方法
- ▶▶ 掌握 If 语句及其嵌套使用方法
- ▶▶ 掌握 Select Case 语句的使用方法
- ▶▶ 掌握 If 函数的使用方法
- ▶▶ 掌握 For...Next 循环语句的使用方法
- ▶▶ 掌握 Do...Loop 循环语句的使用方法
- ▶▶ 学会如何使用多重循环
- ▶▶ 学会选择结构与循环结构的嵌套



## 3.1 算法

 教学录像：光盘\TM\lx\3\算法.exe

算法是学习程序设计的基础，也可以说是程序设计的入门知识，掌握算法可以帮助读者快速理清程序设计的思路，找出多种解决问题的方法，从而选择最合适的解决方案。

本节将介绍什么是算法、算法的特点、算法的描述方法以及构成算法的基本结构。

### 3.1.1 什么是算法

“算法”这个术语听起来可能很陌生，其实大多数人每天都会用到许多算法。

**例 3.1** 我们早晨坐车上上班，一般情况下会坐公交车上班，但如果时间来不及或遇到其他特殊情况，可能会打车上上班，这就是一个“算法”。因此，广义地讲，“算法”就是解决某个问题或处理某件事的方法和步骤。下面再来了解在计算机程序设计中，算法是如何体现的。

**例 3.2** 商家给客户打折，规定一种商品一次消费金额超过 200 元的客户可以获得折扣(10%)。(实例位置：光盘\TM\sl\3\1)

首先，把单价和数量相乘，然后判断相乘后所得的结果：消费金额是否超过(>)200 元？显然，问题有两种答案：是或不是。如果消费金额不大于 200 元，则将消费金额赋值给应收金额，这种情况下没有折扣；如果消费金额大于 200 元，则首先计算折扣金额(本例中，存在 10%的折扣)，然后将消费金额减去折扣金额，所得结果就是应收金额。

具体算法描述如下。

(1) 计算消费金额(txtSum)，消费金额(txtSum) = 单价(txtPrice) × 数量(txtQYT)。

(2) 判断消费金额(txtSum)是否大于 200，如果不大于 200，则执行步骤(3)，否则执行步骤(4)、(5)。

(3) 将消费金额(txtSum)赋值给应收金额(txtRsum)。

(4) 计算折扣金额(txtDisCount)，折扣金额(txtDisCount) = 消费金额(txtSum) × 0.1

(5) 计算应收金额(txtRsum)，应收金额(txtRsum) = 消费金额(txtSum) - 折扣金额(txtDisCount)

有了上述描述，就可以在 VB 中编写程序了，程序代码如下。

```
Private Sub Command1_Click()  
    txtSum = Val(txtPrice) * Val(txtQYT)           '计算消费金额  
    If txtSum > 200 Then                             '消费金额大于 200  
        txtDisCount = Val(txtSum) * 0.1            '计算折扣金额  
        txtRSum = Val(txtSum) - Val(txtDisCount)   '应收金额为消费金额减去折扣金额  
    Else  
        txtRSum = txtSum                           '将消费金额赋值给应收金额  
    End If  
End Sub
```

按〈F5〉键，运行程序：消费金额不大于 200 没有折扣，结果如图 3.1 所示；消费金额大于 200，

有折扣，结果如图 3.2 所示。

图 3.1 无折扣效果

图 3.2 有折扣效果

上述例子中，为了解决商家给客户打折的问题，我们事先做了很多分析，并确定了采用的方法和步骤，因此狭义地讲，“算法”就是指计算机解决某个问题的方法和步骤。

### 3.1.2 算法的特性

一个算法应该具有以下 5 个主要特性。

- ☑ 有穷性：一个算法（对任何合法的输入）在执行有穷步后能够结束，并且在有限的时间内完成。
- ☑ 确定性：算法中的每一步都有确切的含义。
- ☑ 可行性：算法中的操作能够用已经实现的基本运算执行有限次来实现。
- ☑ 输入：一个算法有零个或者多个输入，零个输入就是算法本身确定了初始条件。
- ☑ 输出：一个算法有一个或多个输出，以反映出数据加工的结果，没有输出的算法是没有意义的。

### 3.1.3 算法的描述方法

为了让算法清晰易懂，需要选择一种好的描述方法。算法的描述方法有很多，有自然语言、伪代码、传统流程图、N-S 结构化流程图等。

#### 1. 自然语言

自然语言就是用人们日常使用的语言描述解决问题的方法和步骤，例如 3.1.1 节的例 3.2 中商家给客户打折算法的描述。这种描述方法通俗易懂，即使是不熟悉计算机语言的人也很容易理解程序。但是，自然语言在语法和语义上往往具有多义性，并且比较繁琐，对程序流向等描述不明了、不直观。

#### 2. 伪代码

伪代码是介于自然语言和计算机语言之间的文字和符号，它与一些高级编程语言（如 Visual Basic 和 Visual C++）类似，但是不需要真正编写程序时所遵循的严格规则。伪代码用一种从顶到底，易于阅读的方式表示算法。在程序开发期间，伪代码经常用于“规划”一个程序，然后再转换成 VB 程序。


例 3.3 用伪代码描述例 3.2 中商家给客户打折的算法。

```
SUM=QYT*PRICE
IF SUM>200 THEN
```

```

DISCOUNT=SUM*0.1
RSUM=SUM-DISCOUNT
ELSE
    RSUM=SUM
ENDIF

```


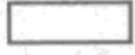




 说明：为了强调和清晰起见，关键字 IF、THEN、ELSE 和 ENDIF 通常用大写字母，THEN 和 ELSE 子句运用缩进格式（一般为几个字符的位置），关键字 ELSE 和 ENDIF 保证与关键字 IF 左边对齐，以便清晰地看出它们属于相同的判断步骤。

### 3. 传统流程图

传统流程图，使用不同的几何图形来表示不同性质的操作，使用流程线来表示算法的执行方向。比起前两种描述方式，它具有直观形象、逻辑清楚、易于理解等特点，但它占用篇幅较大，流程随意转向，较大的流程图不易读懂。

传统流程图的基本流程图符号及说明如表 3.1 所示。

表 3.1 流程图符号及说明

流程图符号	名 称	说 明
	起止框	表示算法的开始和结束
	处理框	表示完成某种操作，如初始化或运算赋值等
	判断框	表示根据一个条件成立与否，决定执行两种不同操作的其中一个
	输入输出框	表示数据的输入输出操作
	流程线	用箭头表示程序执行的流向
	连接点	用于流程分支的连接

例 3.4 用传统流程图描述例 3.2 中商家给客户打折的算法，如图 3.3 所示。

### 4. N-S 结构化流程图

N-S 结构化流程图是 1973 年美国学者 I·Nassi 和 B·Shneiderman 提出的一种符合结构化程序设计原则的描述算法的图形方法，又叫做盒图。

N-S 结构化流程图有以下几个特点：

- ☒ 图中每个矩形框（除 Case 语句中表示条件取值的矩形框外）都是明确定义了的功能域（即一个特定控制结构的作用域），以图形表示，清晰可见。
- ☒ 它的控制转移不能任意规定，必须遵守结构化程序设计的要求。
- ☒ 很容易确定局部数据和（或）全局数据的作用域。
- ☒ 很容易表现嵌套关系，也可以表示模块的层次结构。

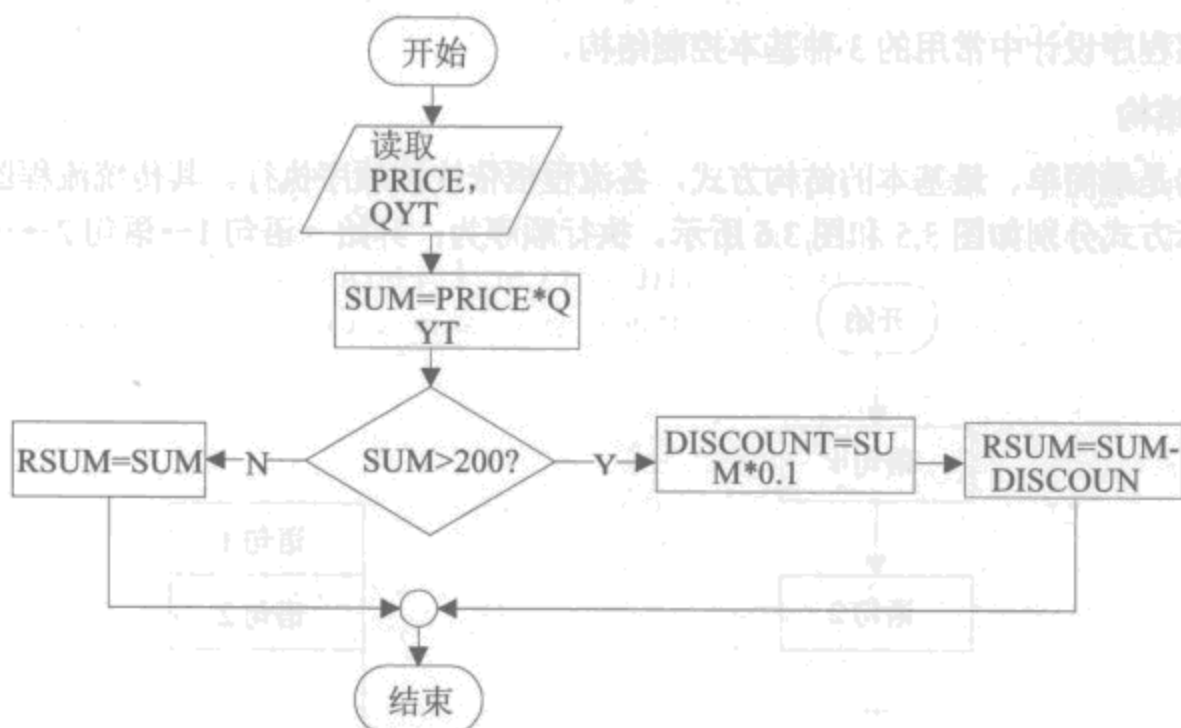


图 3.3 用传统流程图描述商家给客户打折的算法

例 3.5 用 N-S 结构化流程图描述例 3.2 中商家给客户打折的算法，如图 3.4 所示。



图 3.4 用 N-S 结构化流程图描述商家给客户打折的算法

以上描述算法的方法各有特点，在实际工作中如何选择使用呢？主要参考以下几点：

- ☑ 行业惯例和软件人员使用的普遍性，易于学习掌握和交流。
- ☑ 易于表达逻辑条件及其相应的处理，能有效地表达各种数据类型和数据结构。
- ☑ 便于转换成计算机能接受的代码，易于进行逻辑验证和便于修改。

### 3.1.4 构成算法的基本控制结构

在程序设计中，构成算法的基本控制结构有 3 种：顺序结构、选择结构和循环结构。合理使用这些控制结构可以使程序结构清晰，易读性强，并且易于查错和排错，这也是正确算法的体现；反之则会造成程序质量下降，运行速度慢等。

下面介绍程序设计中常用的 3 种基本控制结构。

### 1. 顺序结构

顺序结构是最简单、最基本的结构方式，各流程框依次按顺序执行。其传统流程图与 N-S 结构化流程图的表示方式分别如图 3.5 和图 3.6 所示。执行顺序为：开始→语句 1→语句 2→…→结束。

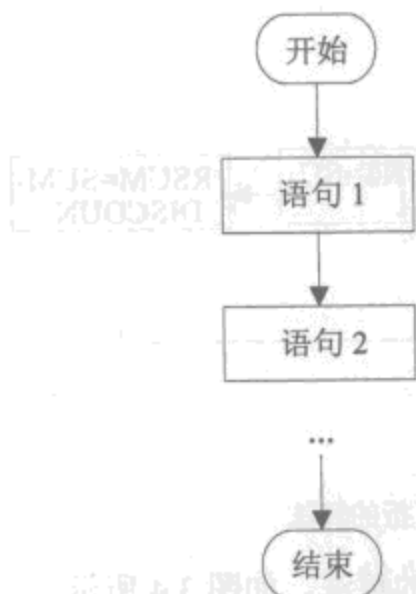


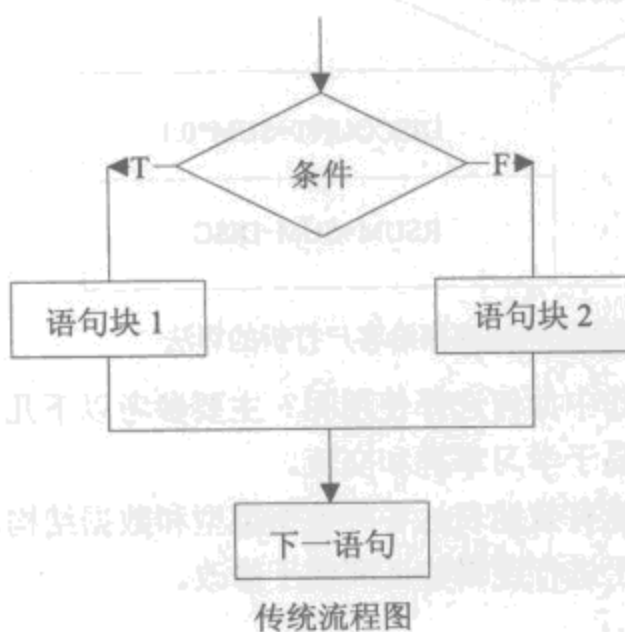
图 3.5 顺序结构传统流程图



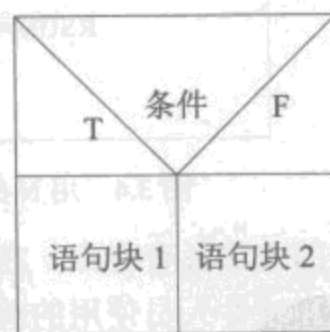
图 3.6 顺序结构 N-S 结构化流程图

### 2. 选择（分支）结构

选择结构就是对给定条件进行判断，条件为 True 时执行一个分支，条件为 False 时执行另一个分支。下面是双分支和单分支选择结构的传统流程图表示方式与 N-S 结构化流程图表示方式，如图 3.7 和图 3.8 所示。



传统流程图



N-S 结构化流程图

图 3.7 双分支选择结构的两种流程图



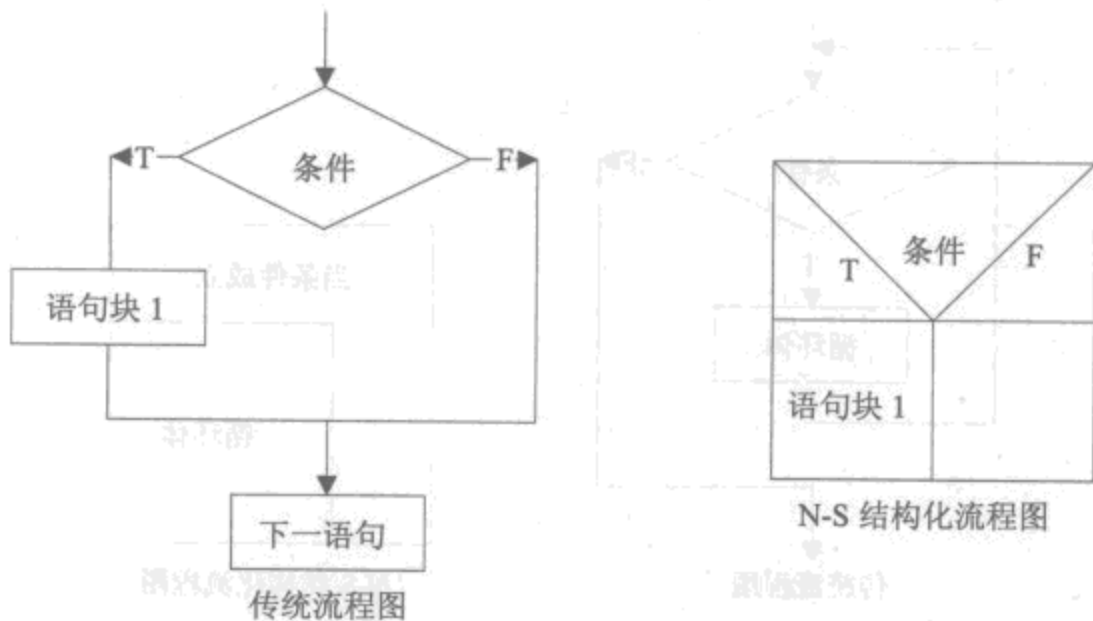


图 3.8 单分支选择结构的两种流程图

当选择的情况较多时，使用前面介绍的选择结构，就会很麻烦而且不直观。这时可以使用多情况选择结构（即 Case 结构），其传统流程图表示方式与 N-S 结构化流程图表示方式，分别如图 3.9 和 3.10 所示。

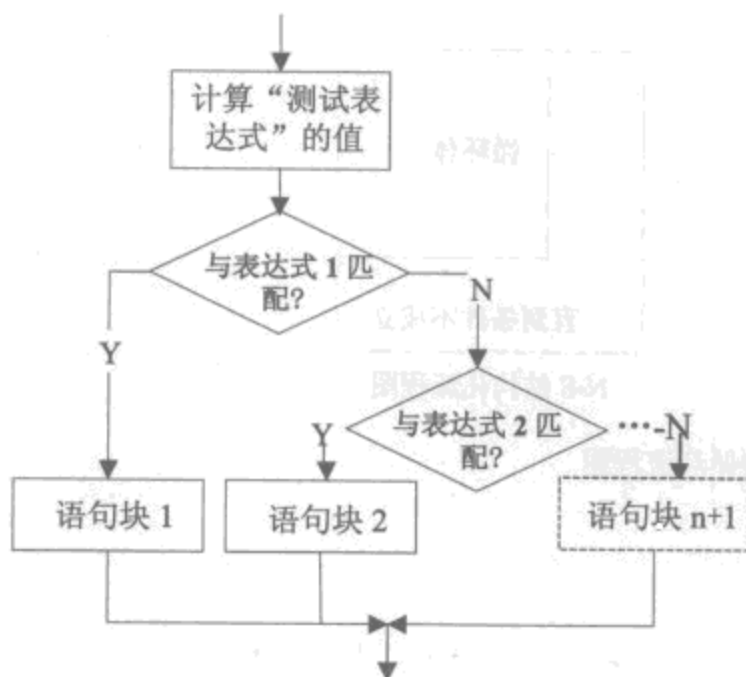


图 3.9 Case 结构的传统流程图

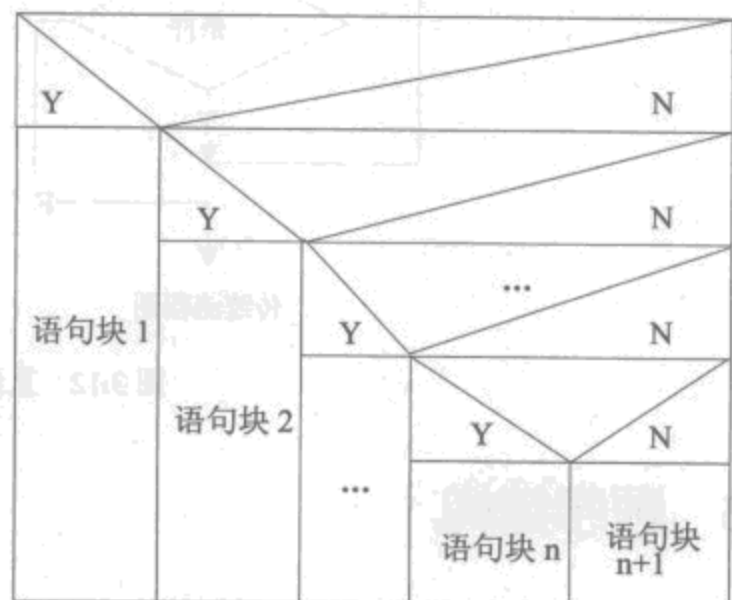


图 3.10 Case 结构的 N-S 结构化流程图

### 3. 循环结构

循环结构可以根据需要多次重复执行一行或多行代码。循环结构分为两种：当型循环和直到型循环。当型循环，先判断后执行。当条件为 True 时反复执行语句或语句块；条件为 False 时，跳出循环，继续执行循环后面的语句。流程图如图 3.11 所示。

直到型循环，先执行后判断。先执行语句或语句块，再进行条件判断，直到条件为 False 时，跳出循环，继续执行循环后面的语句，否则一直执行语句或语句块。流程图如图 3.12 所示。

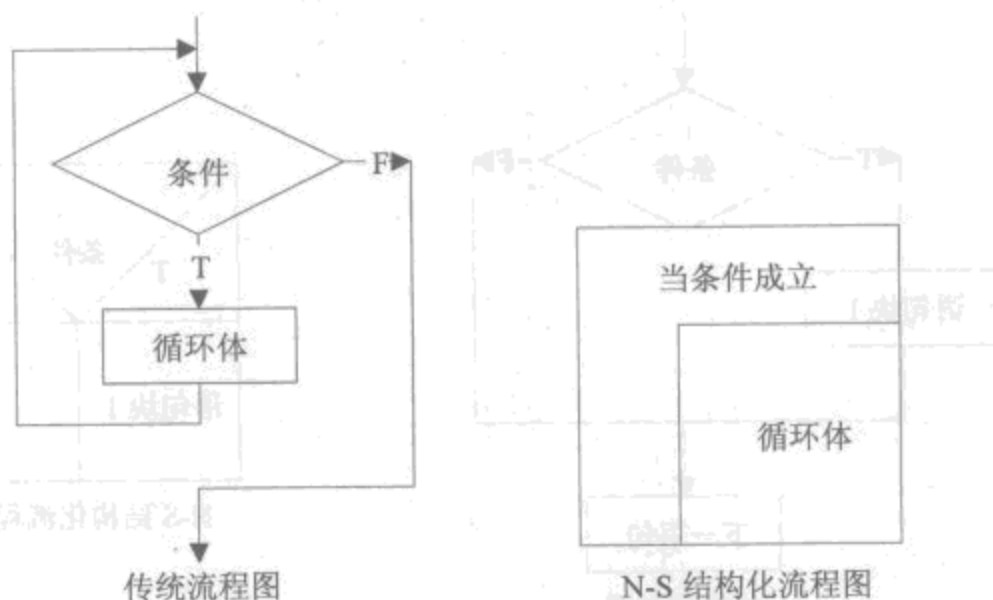


图 3.11 当型循环流程图

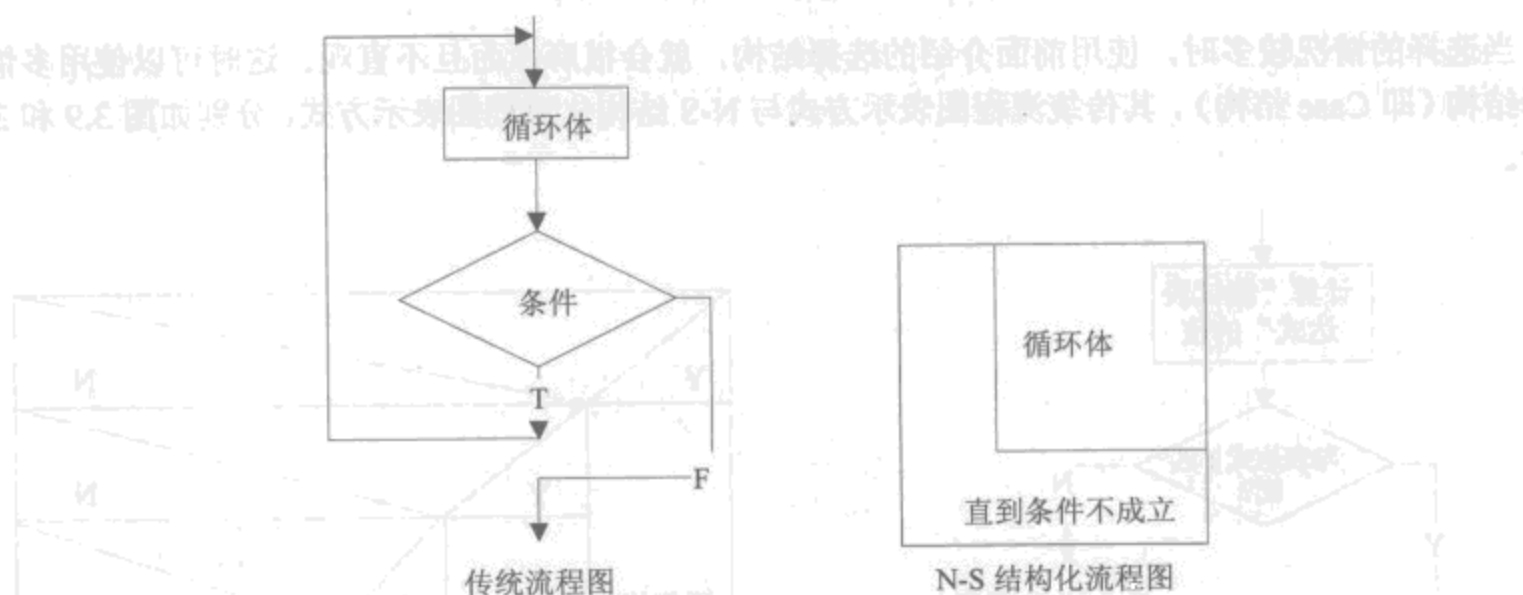


图 3.12 直到型循环流程图

## 3.2 顺序结构

教学录像：光盘\TM\lx\3\顺序结构.exe

顺序结构的语句主要包括赋值语句、输入/输出语句等，其中输入/输出一般可以通过文本框控件、标签控件、InputBox 函数、MsgBox 函数以及 Print 方法来实现。

### 3.2.1 赋值语句

赋值语句是将表达式的值赋给变量或属性，通过 Let 关键字使用赋值运算符“=”给变量或属性赋值，其语法格式如下。

[Let] <变量名> = <表达式>

Let: 可选的参数。显式使用的 Let 关键字是一种格式, 通常省略该关键字。

变量名: 必需的参数。变量或属性的名称, 变量命名遵循标准的变量命名约定。

表达式: 必需的参数。赋给变量或属性的值。

例如定义一个长整型变量, 给这个变量赋值 2205, 代码如下。

```
Dim a As Long
Let a = 2205
```

上述代码中可以省略关键字 Let。例如在文本框中显示文字, 代码如下。

```
Text1.Text="mingrisoft"
```

赋值语句看起来简单, 但使用时也要注意以下几点。

(1) 赋值号与表示等于的关系运算符都用“=”表示, VB 系统会自动区分, 即在条件表达式中出现的是等号, 否则是赋值号。

(2) 赋值号左边只能是变量, 不能是常量、常数符号和表达式。下面均是错误的赋值语句。

X+Y=1	'左边是表达式
vbBlack =myColor	'左边是常量, 代表黑色
10 = abs(s)+x+y	'左边是常量

(3) 当表达式为数值型并与变量精度不同时, 需要强制转换左边变量的精度。

例如:

n%=4.6	'n 为整型变量, 转换时四舍五入, 值为 5
--------	-------------------------

(4) 当表达式是数字字符串, 左边变量是数值型时, 右边值将自动转换成数值型再赋值。如果表达式中有非数字字符或空字符串, 则出错。

n%="123"	'将字符串 123 转换为数值数据 123
----------	-----------------------

下列情况会出现运行时错误。

```
n%="123mr"
n%=""
```

(5) 当逻辑值赋值给数值型变量时, True 转换为-1, False 转换为 0; 反之当数值赋给逻辑型变量时, 非 0 转换为 True, 0 转换为 False。

例 3.6 例如在立即窗口中将单选按钮被选择的状态赋值给整型变量。(实例位置: 光盘\TM\sl3\2)

① 新建一个工程, 在 Form1 窗体中添加一个 CommandButton 控件和两个 OptionButton 控件。

② 在代码窗口中编写如下代码。

Private Sub Command1_Click() Dim a As Integer, b As Integer a = Option1.Value b = Option2.Value Debug.Print "Opt1 的值: " & a Debug.Print "Opt2 的值: " & b End Sub	'定义整型变量 '将逻辑值赋给整型变量 a '将逻辑值赋给整型变量 b '输出结果
---	--

③ 按〈F5〉键运行程序，结果如图 3.13 所示。

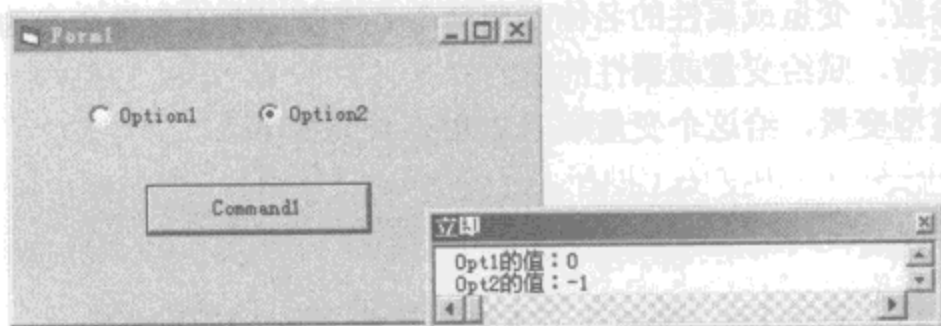


图 3.13 在立即窗口中显示 OptionButton 控件的返回值

(6) 任何非字符型的值赋值给字符型变量，自动转换为字符型。

为了保证程序的正常运行，一般利用类型转换函数将表达式的类型转换成与左边变量匹配的类型。

### 3.2.2 数据的输入

在程序设计时，通常使用文本框 (TextBox 控件) 或 InputBox 函数来输入数据。当然，也可以使用其他对象或函数来输入数据。

#### 1. 文本框

利用文本框控件的 Text 属性可以获得用户从键盘输入的数据，或将计算的结果输出。

例 3.7 在两个文本框中分别输入“单价”和“数量”，然后通过 Label 控件显示金额，代码如下。

(实例位置：光盘\TM\sl\3\3)

```
Private Sub Command1_Click()  
    Dim mySum As Single           '定义单精度浮点型变量  
    mySum = Val(Text1.Text) * Val(Text2.Text) '计算"单价"和"数量"相乘  
    Label1.Caption = "金额为：" & mySum      '显示计算结果  
End Sub
```

按〈F5〉键，运行程序，结果如图 3.14 所示。

#### 2. 输入对话框 InputBox 函数

InputBox 函数提供了一个简单的对话框供用户输入信息，如图 3.15 所示。在该对话框中有一个输入框和两个命令按钮。显示对话框后，将等待用户输入。当用户单击“确定”按钮后返回输入的内容。

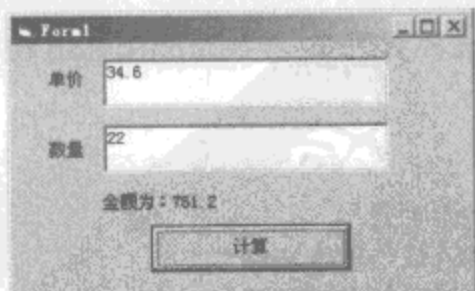


图 3.14 输入“单价”和“数量”后计算金额

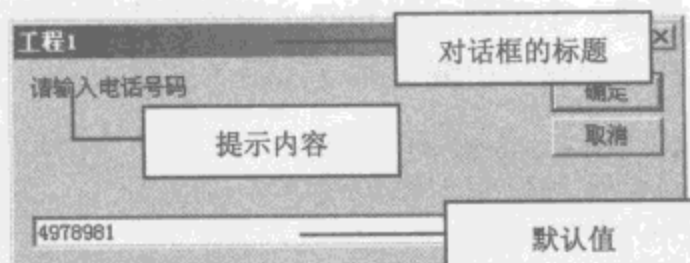


图 3.15 InputBox 输入框

InputBox 输入函数有两种表达方式，一种为带返回值的，另一种为不带返回值的。


带返回值的输入函数的使用方法举例如下：

```
MyValue = InputBox("请输入电话号码", , 4978981)
```

上述语句中 InputBox 函数后面的一对圆括号不能省略，其中各参数之间用逗号隔开。

不带返回值的输入函数的使用方法举例如下：

```
InputBox "请输入电话号码", , 4978981
```

 说明：有关 InputBox 函数更详细的介绍，可参见第 11 章。

### 3.2.3 数据的输出

输出数据可以通过 Label 控件、输出对话框函数 MsgBox 函数和 Print 方法等。由于通过 Label 控件输出数据较简单，这里就不介绍了，下面仅介绍 MsgBox 函数和 Print 方法。

#### 1. MsgBox 函数

MsgBox 函数的功能是在对话框中显示消息，如图 3.16 所示，等待用户单击按钮，并返回一个整数告诉系统用户单击的是哪一个按钮。

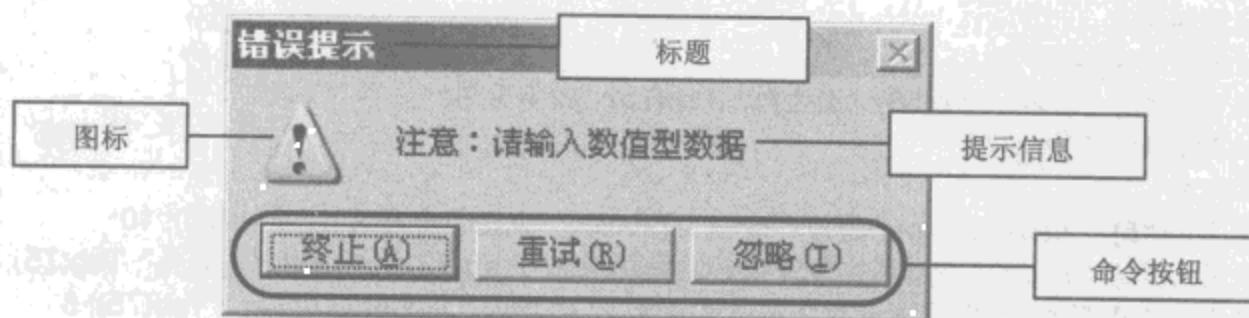


图 3.16 MsgBox 对话框

MsgBox 函数有两种表达方式，一种为带返回值的，另一种为不带返回值的。


带返回值的函数的使用方法举例如下：

```
myvalue = MsgBox("注意：请输入数值型数据", 2 + vbExclamation, "错误提示")
If myvalue = 3 Then End
```

上述语句中 MsgBox 函数后的一对圆括号不能省略，其中各参数之间用逗号隔开。

不带返回值的函数的使用方法举例如下：

```
MsgBox "请输入数值型数据!", , "提示"
```

 说明：有关 MsgBox 函数更详细的介绍，可参见第 11 章。

#### 2. Print 方法

Print 是输出数据、文本的一个重要方法，其语法格式如下：

```
窗体名称.Print[<表达式>[,|:]<表达式>...]
```

<表达式>：可以是数值或字符串表达式。对于数值表达式，先计算表达式的值，然后输出；而字



符串则原样输出。如果表达式为空，则输出一个空行。

当输出多个表达式时，各表达式用分隔符（逗号、分号或空格）隔开。若用逗号分隔将以 14 个字符位置为单位把输出行分成若干个区段，每区段输出一个表达式的值。而表达式之间用分号或空格作为分隔符，则按紧凑格式输出。

一般情况下，每执行一次 Print 方法将自动换行，可以通过末尾加上逗号或分号的方法使输出结果在同一行显示。

**注意：**Print 方法除了可以作用于窗体外，还可以作用于其他多个对象，如立即窗口（Debug）、图片框（PictureBox）、打印机（Printer）等。如果省略“对象名”，则在当前窗体上输出。

**例 3.8** 下面使用 Print 方法在窗体中输出图书排行数据，代码如下。（实例位置：光盘\TM\sl\3\4）

```
Private Sub Form_Click()  
    Print                                     '输出空行  
    Font.Size = 14                           '设置字体  
    Font.Name = "华文行楷"  
    Print Tab(45); Year(Date) & "年" & Month(Date) & "月份图书销售排行"  
    '打印标题  
    CurrentY = 700  
    Font.Size = 9  
    Font.Name = "宋体"  
    Print Tab(15); "书名"; Tab(55); "出版社"; Tab(75); "销售数量"  
    Print Tab(14); String(75, "-")           '输出线  
    '打印内容  
    Print Tab(15); "Visual Basic 经验技巧宝典"; Tab(55); "人民邮电出版社"; Tab(75); 10  
    Print Tab(15); "Visual Basic 数据库系统开发案例精选"; Tab(55); "人民邮电出版社"; Tab(75); 8  
    Print Tab(15); "Delphi 数据库系统开发案例精选"; Tab(55); "人民邮电出版社"; Tab(75); 6  
End Sub
```

代码说明：

☒ Tab(n)：内部函数，用于将指定表达式从窗体第 n 列开始输出。

☒ Print：如果 Print 后面没有内容，则输出空行。

按〈F5〉键，运行工程，单击窗体，结果如图 3.17 所示。

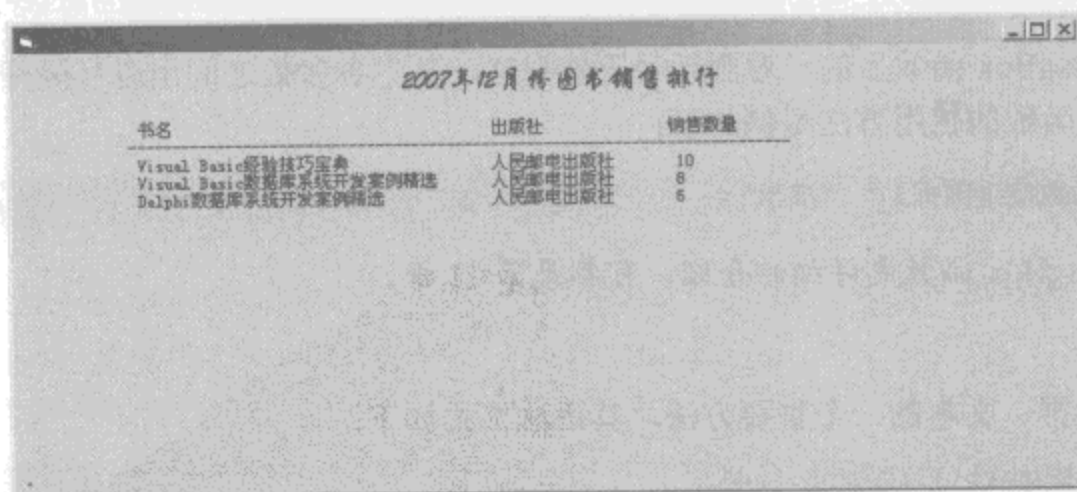


图 3.17 使用 Print 语句在窗体中输出数据

### 3.3 选择结构

 教学录像：光盘\TM\lx\3\选择结构.exe

选择结构属于分支结构的一种，也可以称为判定结构。程序通过判断所给的条件和判断条件的结果执行不同的程序段。

#### 3.3.1 单分支 If...Then 语句

If...Then 语句用于判断表达式的值，满足条件时执行其包含的一组语句，执行流程如图 3.18 所示。

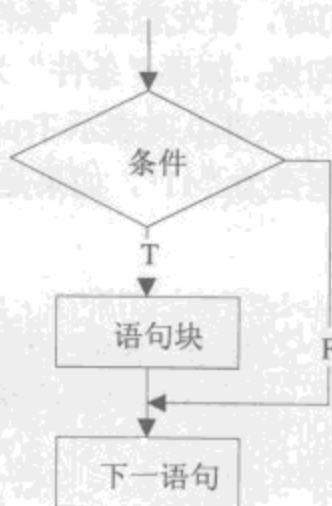


图 3.18 If...Then 语句执行流程图

If...Then 语句有两种形式，即单行和块形式。

##### ☒ 单行形式

顾名思义，单行形式的 If...Then 语句只能在一行内书写完毕，即不能一行超过 255 个字符的限度。语法格式如下：

If 条件表达式 Then 语句

If 和 Then 都是关键字。“条件表达式”应该是一个逻辑表达式，或者其值是可以转换为逻辑值的其他类型表达式。

当程序执行到单行形式的 If...Then 语句时，首先检查“条件表达式”，以确定下一步的流向。如果“条件”为 True，则执行 Then 后面的语句；如果“条件”为 False，则不执行“语句”中的任何语句，直接跳到下一条语句执行。

下面是一条单行形式的 If...Then 语句：

If Text1.Text = "11" Then MsgBox "登录成功！"

条件表达式

语句

##### ☒ 块形式

块形式的 If...Then 语句是以连续数条语句的形式给出的。

语法格式如下：

```
If 条件表达式 Then
    语句块
End If
```

其中“语句块”可以是单个语句，也可以是多个语句。多个语句可以写在多行中，也可以写在一行中，并用冒号“:”隔开。

例如，如果变量 a 等于 1，那么变量 b 等于 100，c 等于 100 且 d 等于 100，代码如下。

```
If a=1 Then
    b=100:c=100:d=100
End If
```

给多个变量赋值，用冒号“:”隔开

当程序执行到块形式的 If...Then 语句时，首先检查“条件表达式”，以确定下一步的流向。如果“条件”为 True，则执行 Then 后面的语句块；如果“条件”为 False，则跳过 Then 后面的语句或语句块。如果逻辑表达式为数值表达式，计算结果非 0 时表示 True，计算结果为 0 时表示 False。

例 3.9 判断“密码”文本框中的值是否为“11”，如果是则提示用户登录成功，代码如下。（实例位置：光盘\TM\sl3\5）

```
Private Sub Command1_Click()
    If Text1.Text = "11" Then
        MsgBox "登录成功！"
    End If
End Sub
```

'判断“密码”文本框中的值是否为“11”  
'提示用户输入正确

块形式

注意：块形式的 If...Then...End If 语句必须使用 End If 关键字作为语句的结束标志，否则会出现语法错误或逻辑错误。

### 3.3.2 双分支 If...Then...Else 语句

在 If...Then...Else 语句中，可以有若干组语句块，根据实际条件只执行其中的一组，其执行流程如图 3.19 所示。

If...Then...Else 语句也分为单行形式和块形式。

#### ☑ 单行形式

语法格式如下：

```
If 条件表达式 Then 语句块 1 Else 语句块 2
```

当条件满足时（即“条件表达式”的值为 True），执行“语句块 1”，否则执行“语句块 2”，然后继续执行 If 语句下面的语句。

例如下面就是一个单行形式的 If...Then...Else 语句：

```
If Text1.Text = "11" Then MsgBox "登录成功！" Else MsgBox "密码错误，重新输入！"
```

条件表达式
语句块 1
语句块 2

### ☑ 块形式

如果单行形式中的两个语句块中的语句较多，则写在单行不易读且容易出错，这时就应该使用块形式的 If...Then...Else 语句。

语法格式如下：

```
If 条件表达式 Then
    语句块 1
Else
    语句块 2
End If
```

块形式的 If...Then...Else...End If 语句与单行形式的 If...Then...Else 语句功能相同，只是块形式更便于阅读和理解。

另外，块形式中的最后一个 End If 关键字不能省略，它是块形式的结束标志，如果省略会出现编译错误，如图 3.20 所示。

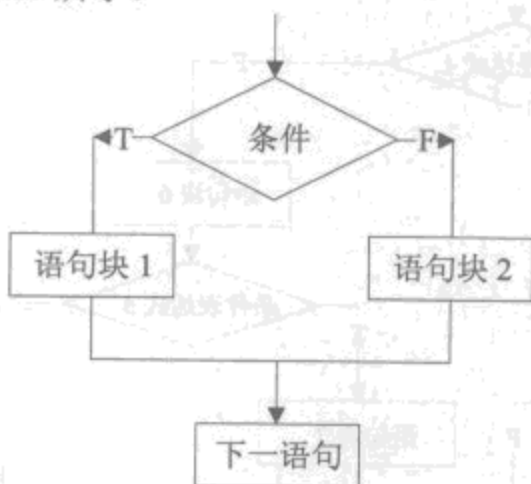


图 3.19 If...Then...Else 语句执行流程图

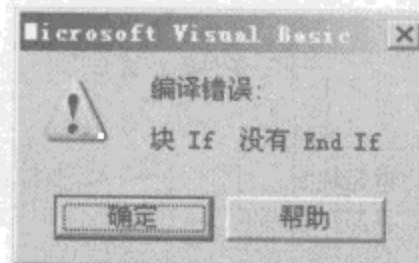


图 3.20 省略最后一个 End If 时出现错误

**例 3.10** 下面用块形式判断用户输入的密码，如果“密码”文本框中的值为“11”，则提示用户登录成功，否则提示用户“密码错误，请重新输入！”，代码如下。（实例位置：光盘\TM\sl3\6）

```
Private Sub Command1_Click()
    If Text1.Text = "11" Then
        MsgBox "登录成功！", , "提示"
    Else
        MsgBox "密码错误，请重新输入！", , "提示"
    End If
End Sub
```

'判断“密码”文本框中的值是否为“11”  
'提示登录成功  
'否则提示密码错误

### 3.3.3 If 语句的嵌套

一个 If 语句的“语句块”中可以包括另一个 If 语句，这种就是“嵌套”。在 VB 中允许 If 语句嵌套。下面语句就是 If 语句的嵌套形式。

```
If 条件表达式 1 Then
    语句块 1
    '最外层 If 语句
```

If 条件表达式 2 Then	'内层 If 语句
语句块 2	
Else	
If 条件表达式 4 Then ...语句块 3 Else ...语句块 4	'最内层 If 语句
End If	'内层 If 结束语句
语句块 5	
Else	'最外层 If 语句
语句块 6	
If 条件表达式 3 Then	'内层 If 语句
语句块 7	
End If	'内层 If 结束语句
语句块 8	
End If	'最外层 If 结束语句

上面的语句看起来不太直观，下面用流程图来表示，如图 3.21 所示。

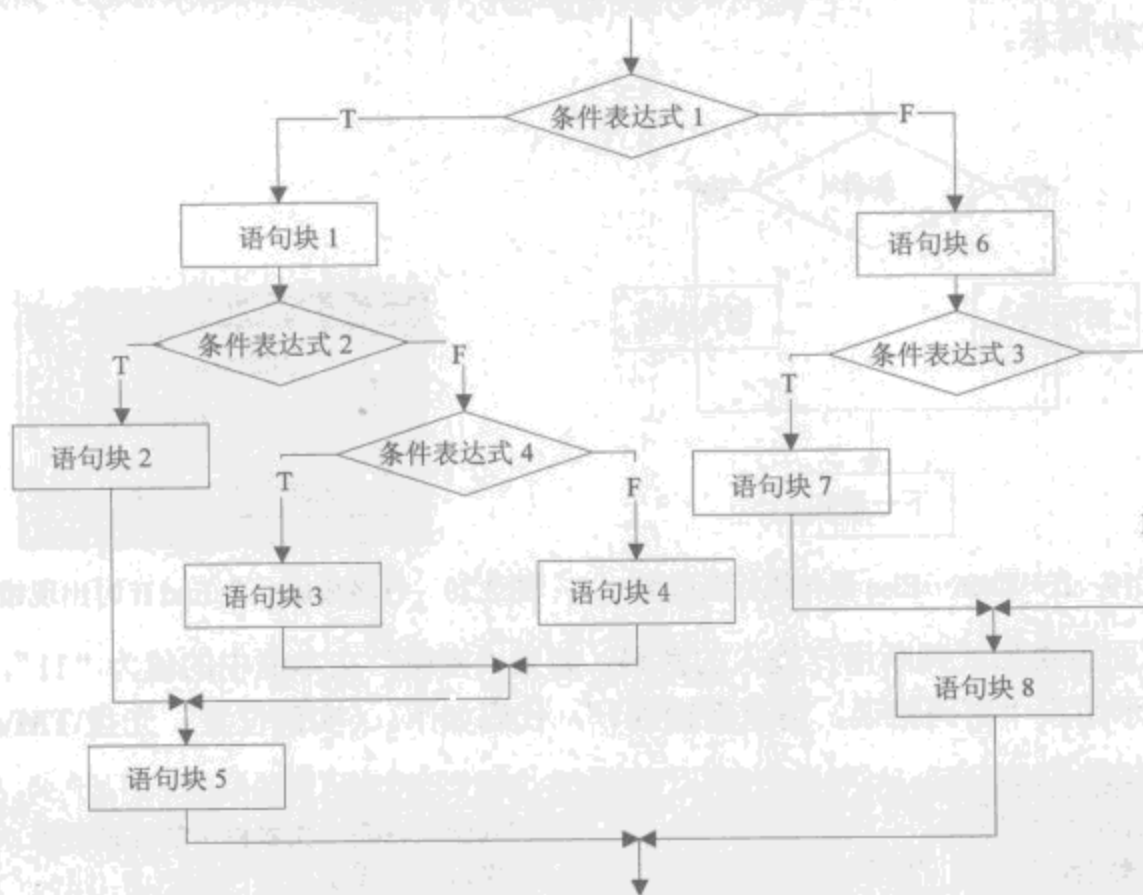


图 3.21 If 语句嵌套执行流程图

对于这种结构，书写时应该采用缩进形式，这样可以使程序代码看上去结构清晰，增强代码可读性，便于日后修改调试。另外，Else 或 End If 必须与它相关的 If 语句相匹配，构成一个完整的 If 结构语句。

**例 3.11** 下面通过一个典型的“用户登录”实例，介绍 If 语句的嵌套在实际项目开发中的应用，设计步骤如下。（实例位置：光盘\TM\sl\3\7）

（1）新建一个工程，在 Form1 窗体中添加 Label 控件、ComboBox 控件和 TextBox 等控件，如图 3.22 所示。



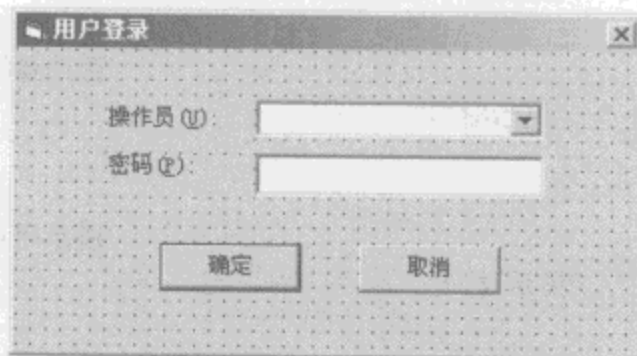


图 3.22 用户登录界面设计效果

设置窗体和控件的相关属性，设置结果如表 3.2 所示。

表 3.2 各窗体和控件的主要设置

窗体/控件	Name 属性	Caption 属性	Default 属性	Text 属性
Form	FrmLogin	用户登录		
TextBox	FrmPwd			为空
ComboBox	cboUserName			为空
CommandButton	CmdOk	确定	True	
CommandButton	CmdCancel	取消		
Label	LabPwd	操作员(&U):		
Label	LabKind	密码(&P):		

(2) 在代码窗口中编写如下代码。

```

Option Explicit
Public intMyTimes As Integer
Const MaxTimes As Integer = 3
Private Sub Form_Load()
    intMyTimes = 1 '给变量赋初值
    cboUserName.AddItem "管理员"
    cboUserName.AddItem "操作员 1"
    cboUserName.AddItem "操作员 2"
End Sub
Private Sub cmdOK_Click()
    If cboUserName.Text <> "" Then '如果操作员不为空
        If txtPassword.Text = "" Then '判断密码是否为空
            MsgBox "请输入密码!", , "提示窗口"
            txtPassword.SetFocus
        Else
            If txtPassword.Text <> "11" Then '如果密码不是“11”
                If intMyTimes > MaxTimes Then '密码输入次数大于 3 次，则退出程序
                    MsgBox "您无权使用该软件!", , "提示窗口"
                End If
            Else '否则提示密码输入不正确
                intMyTimes = intMyTimes + 1 '每输入一次错误的密码，变量 intMyTimes 就加 1
                MsgBox "密码不正确，请重新输入!", , "提示窗口"
                txtPassword.SetFocus
            End If
        End If
    End If
End Sub

```

```

End If
Else                                     '否则登录成功
    MsgBox "登录成功!", , "提示窗口"
End If
End If
Else                                     '提示用户操作员不能为空
    MsgBox "操作员不能为空!", , "提示窗口"
Exit Sub
End If
End Sub
Private Sub cmdCancel_Click()
End                                     '退出程序
End Sub

```

(3) 按〈F5〉键，运行工程。选择操作员，输入密码，单击“确定”按钮，如图 3.23 所示。

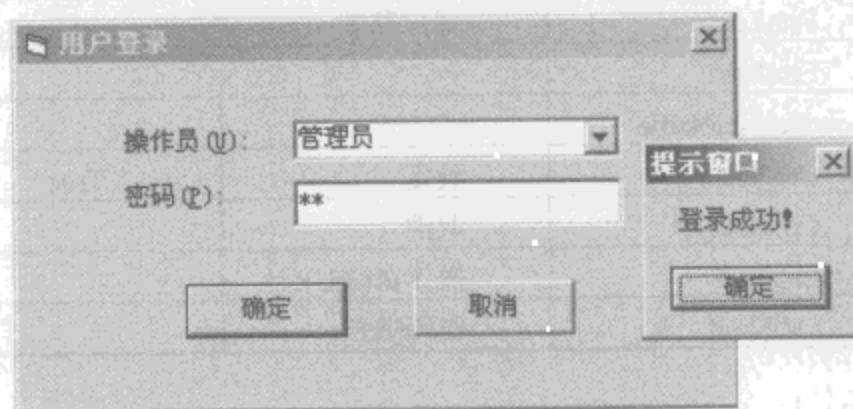


图 3.23 登录成功

具体执行过程如下：

① 判断操作员是否为空，如果操作员为空则提示用户，如图 3.24 所示；否则执行②。

② 判断密码是否为空，如果密码为空则提示用户，如图 3.25 所示；否则执行③。

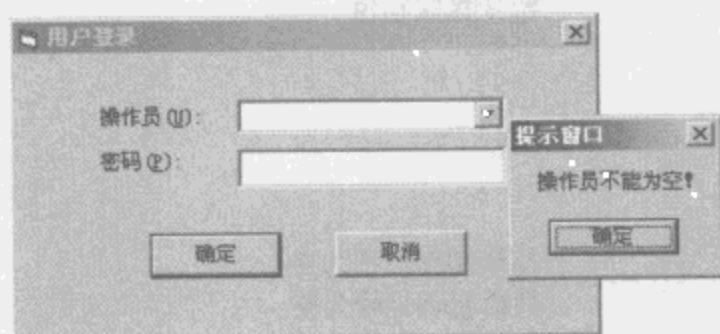


图 3.24 操作员为空

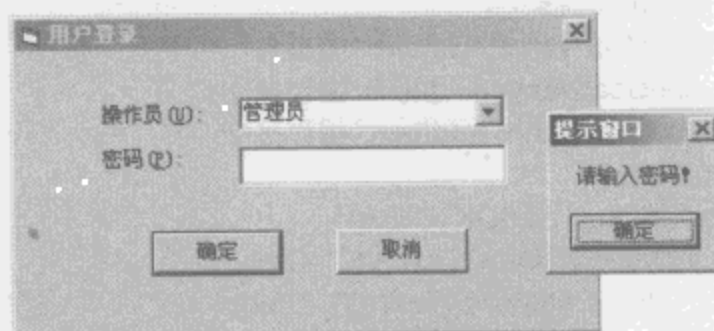


图 3.25 密码为空

③ 判断密码输入是否正确，如果正确则提示“登录成功”；否则执行④。

④ 判断密码输错的次数是否大于 3 次，如果大于 3 次，则提示用户无权使用，如图 3.26 所示，然后退出程序；否则执行⑤。

⑤ 每输入一次错误的密码，变量 intMyTimes 就加 1，并提示用户密码输入有误，如图 3.27 所示。

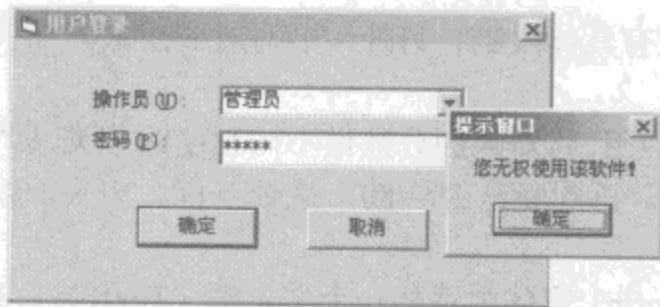


图 3.26 密码输错次数大于 3 次

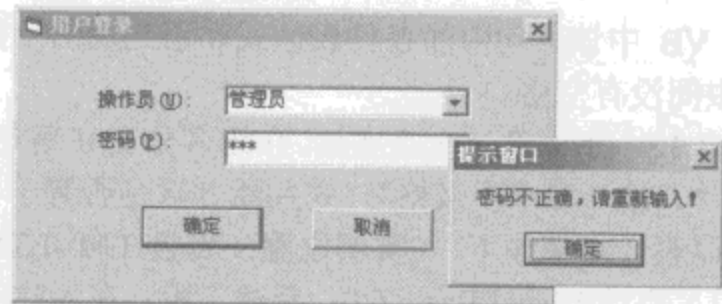


图 3.27 提示用户密码有误

### 3.3.4 多分支 If...Then...ElseIf 语句

只有块形式的写法，语句格式为：

```

If 条件表达式 1 Then
    语句块 1
Elseif 条件表达式 2 Then
    语句块 2
Elseif 条件表达式 3 Then
    语句块 3
...
Elseif 条件表达式 n Then
    语句块 n
...
[Else
    语句块 n+1]
End If
  
```

该语句的作用是根据不同的条件确定执行哪个语句块，其执行顺序为条件表达式 1、条件表达式 2……，一旦条件表达式的值为 True，则执行该条件下的语句块。

多分支 If...Then...ElseIf 语句的执行流程如图 3.28 所示。

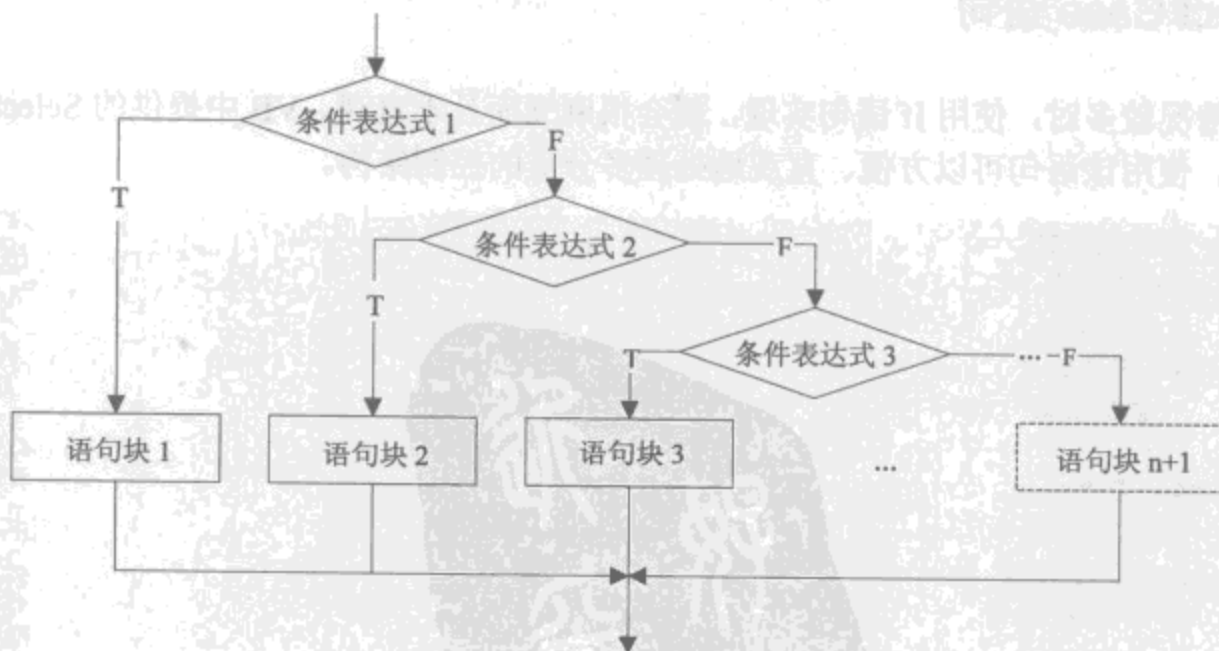


图 3.28 多分支 If...Then...ElseIf 语句执行流程图

在 VB 中该语句中的条件表达式和语句块的个数没有具体限制。另外，书写时应注意，在关键字 ElseIf 中间没有空格。

**例 3.12** 下面通过一个实例，介绍多分支 If 语句的应用。将输入的分数做不同程度的分类，即“优”、“良”、“及格”和“不及格”，先判断分数是否等于 100，再判断是否  $\geq 80$ ，是否  $\geq 60$ ，……，以此类推。程序设计步骤如下。（实例位置：光盘\TM\sl\3\8）

(1) 启动 Visual Basic 6.0，新建工程，在新建的 Form1 窗体中添加一个文本框（Text1）、3 个标签（Label1、Label2 和 Label3）和一个命令按钮（Command1）。

(2) 在代码窗口中编写如下代码。

```
Private Sub Command1_Click()
    '定义一个整型变量
    Dim a As Integer
    '给变量 a 赋值
    a = Val(Text1.Text)
    If a > 100 Or a < 0 Then
        MsgBox "只能输入 0-100 以内的数!"
        Exit Sub
    End If
    If a = 100 Then
        lblResult.Caption = "优"
    ElseIf a >= 80 Then
        lblResult.Caption = "良"
    ElseIf a >= 60 Then
        lblResult.Caption = "及格"
    Else
        lblResult.Caption = "不及格"
    End If
End Sub
```

(3) 按〈F5〉键，运行程序，在“成绩”文本框中输入 88，单击“判断”按钮，结果如图 3.29 所示。

### 3.3.5 Select Case 语句

当选择的情况较多时，使用 If 语句实现，就会很麻烦而且不直观。VB 中提供的 Select Case 语句，语法格式如下。使用该语句可以方便、直观地处理多分支的控制结构。

```
Select Case 测试表达式
    Case 表达式 1
        语句块 1
    Case 表达式 2
        语句块 2
    ...
    Case 表达式 n
        语句块 n
    [Case Else
        语句块 n+1]
End Select
```

Select Case 语句的执行流程如图 3.30 所示。

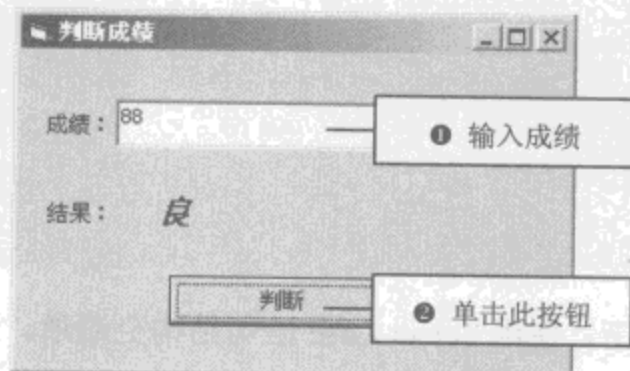


图 3.29 多分支 If 语句实例运行效果图

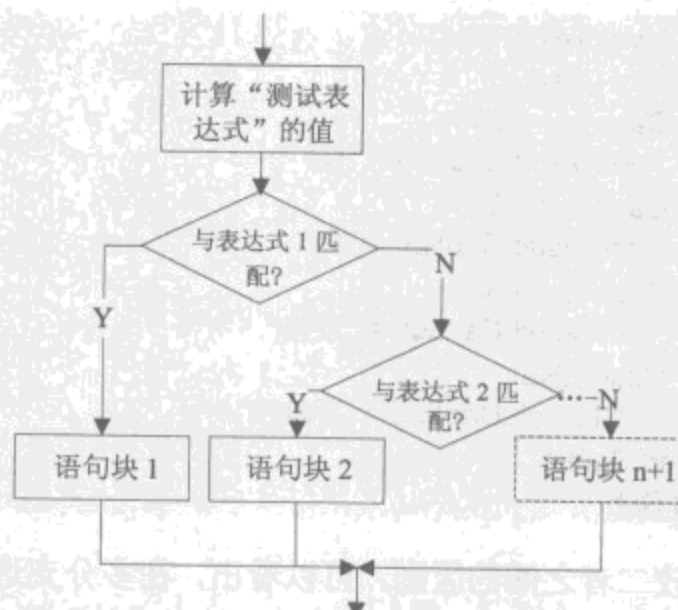


图 3.30 Select Case 语句的执行流程图

执行过程说明:

- (1) 首先计算“测试表达式”的值。
- (2) 然后用这个值与 Case 后面表达式 1, 表达式 2, …… , 表达式 n 中的值比较。
- (3) 若有相匹配的, 则执行 Case 表达式后面的语句块, 执行完该语句块则结束 Select Case 语句, 不再与后面的表达式比较。

(4) 当“测试表达式”的值与后面所有表达式的值都不相匹配时, 若有 Case Else 语句, 则执行 Case Else 后面的语句块 n+1; 若没有 Case Else 语句, 则直接结束 Select Case 语句。

在 Select Case 语句中“表达式”通常是一个具体的值 (如 Case 1), 每一个值确定一个分支。“表达式”的值称为域值, 通过以下几种方法可以设定该值:

- (1) 表达式列表为表达式, 例如: X+100。

Case X+100                      '表达式列表为表达式

- (2) 一组值 (用逗号隔开), 例如:

Case 1,4,7                      '表示条件在 1、4、7 范围内取值

- (3) 表达式 1 To 表达式 2, 例如:

Case 50 TO 60                      '表示条件取值范围为 50~60

- (4) Is 关系表达式, 例如:

Case Is<4                      '表示条件在小于 4 的范围内取值

**例 3.13** 下面将例 3.12 (多分支 If 语句的应用实例) 改写为 Select Case 语句形式, 代码如下: (实例位置: 光盘\TM\sl3\9)

```
Private Sub Command1_Click()
    '定义一个整型变量
    Dim a As Integer
    '给变量 a 赋值
```



```

a = Val(Text1.Text)
Select Case a
    Case Is = 100
        lblResult.Caption = "优"
    Case Is >= 80
        lblResult.Caption = "良"
    Case Is >= 60
        lblResult.Caption = "及格"
    Case Else
        lblResult.Caption = "不及格"
End Select
End Sub

```

比较二者之间的区别，可以看出，在多分支选择情况下，使用 Select Case 语句结构更清晰。当然，若只有两个分支或分支数很少的情况下，直接使用 If...Then 语句更好一些。

### 3.3.6 IIf 函数

IIf 函数的作用是根据表达式的值，返回两部分中的其中一个的值或表达式，其语法格式如下。

IIf(<表达式>, <值或表达式 1>, <值或表达式 2>)

表达式是必要参数，用来判断值的表达式；值或表达式 1 是必要参数，如果表达式为 True，则返回这个值或表达式；值或表达式 2 是必要参数，如果表达式为 False，则返回这个值或表达式。

⚠ 注意：如果表达式 1 或表达式 2 中任何一个在计算时发生错误，那么程序就会发生错误。

例 3.14 下面使用 IIf 函数实现例 3.10 中的实例，即如果“密码”文本框中的值为 11，则提示用户输入正确，否则提示用户“密码不正确，请重新输入！”，代码如下。（实例位置：光盘\TM\3\10）

```

Private Sub Command1_Click()
    Dim str As String '定义字符型变量
    str = IIf(Text1.Text = "11", "输入正确！", "密码不正确，请重新输入！")
    MsgBox str, "提示"
End Sub

```

从例 3.10 和例 3.14 两个示例来看，虽然使用 IIf 函数比使用 If...Then...Else 语句简化了代码，但代码并不直观。

## 3.4 循环结构

📺 教学录像：光盘\TM\3\循环结构.exe

当程序中有重复的工作要做时，就需要用到循环结构。循环结构是指程序重复执行循环语句中一行或多行代码。例如在窗体上输出 10 次 1，每个 1 单独一行。如果使用顺序结构实现，就需要书写 10 次“Print 1”这样的代码，而使用循环语句则简单多了，使用 For...Next 语句实现的代码如下。

```

For i = 1 To 11
  Print 1
Next i

```

在上述代码中  $i$  是一个变量，用来控制循环次数。

VB 提供了 3 种循环语句来实现循环结构：For...Next、Each...For Next 和 Do...Loop，下面分别进行介绍。

### 3.4.1 For...Next 循环语句

当循环次数确定时，可以使用 For...Next 语句，语法格式如下。

```

For 循环变量 = 初值 To 终值 [Step 步长]
  循环体
[Exit For]
  循环体
Next 循环变量

```

For...Next 语句执行过程如图 3.31 所示。

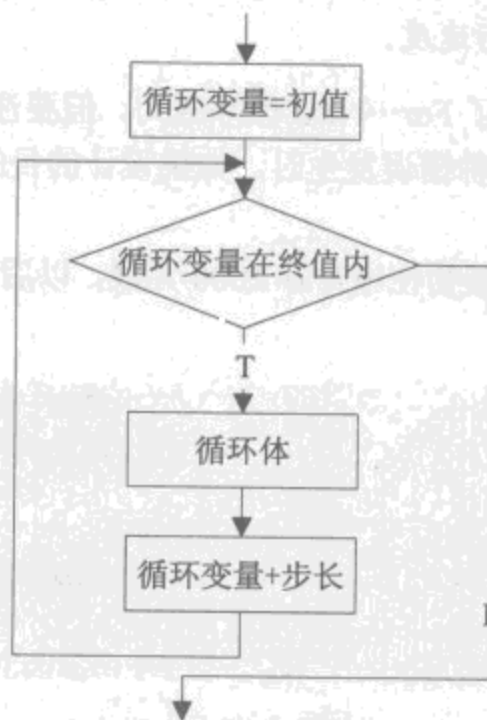


图 3.31 For 语句的执行流程图

(1) 如果不指定“步长”，则系统默认步长为 1；当“初值 < 终值”时，“步长”为 0；当“初值 > 终值”时，“步长”应小于 0。

(2) Exit For 用来退出循环，执行 Next 后面的语句。

(3) 如果出现循环变量的值总是不超出终值的情况，则会产生死循环。此时，可按 <Ctrl+Break> 组合键，强制终止程序的运行。

(4) 循环次数  $N = \text{Int}((\text{终值} - \text{初值}) / \text{步长} + 1)$ 。

(5) Next 后面的循环变量名必须与 For 语句中的循环变量名相同，并且可以省略。

例 3.15 在 ListBox 列表控件中添加 1~12 个月，代码如下。（实例位置：光盘\TM\sl3\11）

```
Private Sub Form_Load()
    Dim i%           '定义一个整型变量
    For i = 1 To 12
        List1.AddItem i & "月"    '在列表中添加月份
    Next i
End Sub
```

按〈F5〉键，运行工程，结果如图 3.32 所示。

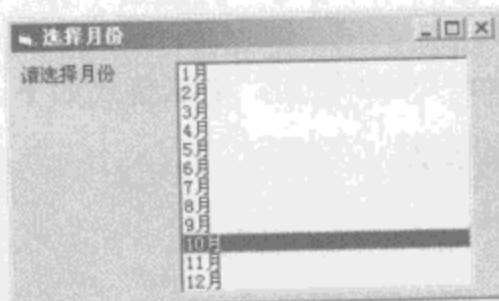


图 3.32 For 语句的简单应用

提示：For...Next 循环的计数器变量应定义为整型或长整型，这样可使 VB 在进行算术运算时节省时间，从而加快循环的执行速度。

通过上述例子，相信您已经学会了 For...Next 循环语句，但要注意一点：For...Next 循环中有个最常见的错误，即“差 1 错误”。当这种错误发生时，如果设计的目的是进行 100 次循环，则可能执行的循环次数是 99 或 101 次。

下面就是一个错误的例子。例如最初在银行存 1000 元钱，以后每年存 1000 元钱，计算 10 年后存款的总金额，代码如下。

```
Dim i As Integer           '定义一个整型变量
Dim mysum As Single        '定义一个单精度浮点型变量
Private Sub Command1_Click()
    For i = 1 To 10
        mysum = mysum + 1000    '累加金额
    Next i
End Sub
```

上述代码已经产生了差 1 错误，因为计数器变量初始值应是 0。下面才是正确的代码：

```
Dim i As Integer           '定义一个整型变量
Dim mysum As Single        '定义一个单精度浮点型变量
Private Sub Command1_Click()
    For i = 0 To 10
        mysum = mysum + 1000    '每循环一次，变量 mysum 就加 1000
    Next i
End Sub
```

For...Next 循环并不总是按 1 进行计数，有时需要按 2、按小数、按负数进行计数，这可以通过在 For...Next 循环中加入 Step 关键字来实现。Step 关键字用于通知 VB 不按 1 进行计数，而按指定的量

进行计数。

例 3.16 如果只显示 2、4、6 等偶数月份，则应将例 3.15 代码改为：

```
Private Sub Form_Load()
    Dim i%
    For i = 1 To 12 Step 2
        List1.AddItem i + 1 & "月"           '在列表中添加月份
    Next i
End Sub
```

如果只显示 1、3、5 等奇数月份，则只须将上述代码中的“List1.AddItem i + 1 & '月'”改为“List1.AddItem i & '月’”。

另外，For 循环中的计数还可以是倒数，只要把间隔值设为负值（即间隔值小于 0），而令初始值大于终止值就可以了。这时，循环的停止条件将会变成是计数值小于终止值时停止。

例 3.17 在窗体上输出 10~1 的整数，代码如下。

```
Private Sub Form_Click()
    Dim i%
    For i = 10 To 1 Step -1
        Print i           '在窗体上输出变量 i
    Next i
End Sub
```

⚠ 注意：进行小数步循环要比进行整数步循环慢得多，即使步值是整数，若计数器是变体类型则循环也会慢得多。

### 3.4.2 For Each...Next 循环语句

For Each...Next 语句用于依照一个数组或集合中的每个元素，循环执行一组语句，语法格式如下。

```
For Each 数组或集合中元素 In 数组或集合
    循环体
    [Exit For]
    循环体
Next 数组或集合中元素
```

📖 说明：（1）数组或集合中元素：必要参数，是用来遍历集合或数组中所有元素的变量。对于集合，可能是一个 Variant 类型变量、一个通用对象变量或任何特殊对象变量；对于数组，这个变量只能是一个 Variant 类型变量。

（2）数组或集合：必要参数，对象集合或数组的名称（不包括用户定义类型的数组）。

（3）循环体：可选参数，循环执行的一条或多条语句。

例 3.18 单击窗体时使用 For Each...Next 语句列出窗体上所有控件名称，代码如下。（实例位置：光盘\TM\sl\3\12）



```
Private Sub Form_Click()
    Dim Myctl As Control
    For Each Myctl In Me.Controls '遍历窗体中的控件
        Print Myctl.Name '在窗体上显示控件名称
    Next Myctl
End Sub
```

按〈F5〉键，运行工程，效果如图 3.33 所示。

### 3.4.3 Do...Loop 循环语句

对于那些循环次数难以确定，但控制循环的条件或循环结束的条件已知的情况下，常常使用 Do...Loop 语句。Do...Loop 语句是最常用、最有效、最灵活的一种循环结构，它有以下 4 种不同的形式。

#### 1. Do While...Loop

使用 While 关键字的 Do...Loop 循环称为“当型循环”，是指当循环条件的值为 True 时执行循环。语法格式如下：

```
Do While <循环条件>
    循环体 1
    <Exit Do>
    循环体 2
Loop
```

该语句的执行流程如图 3.34 所示。

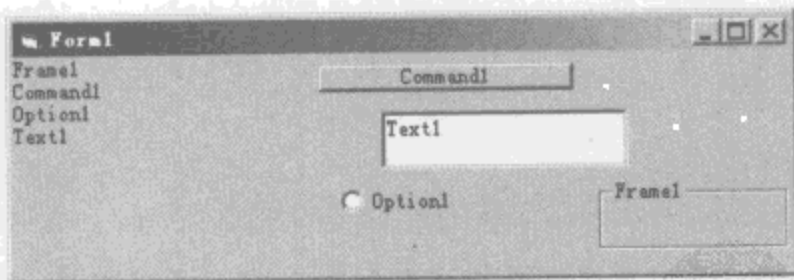


图 3.33 在窗体中显示所有控件名称

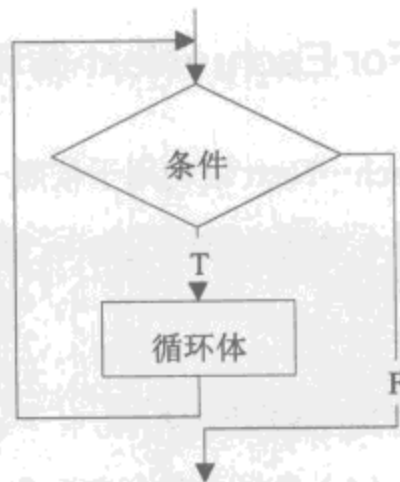


图 3.34 Do While...Loop 语句执行流程图

从上述流程图可以看出，Do While...Loop 语句的执行过程为：

<循环条件>定义了循环的条件，是逻辑表达式，或者能转换成逻辑值的表达式。当程序执行到 Do While...Loop 语句时，首先判断 While 后面的<循环条件>，如果其值为 True，则由上到下执行循环体中的语句。当执行到 Loop 关键字时，返回到循环开始处再次判断 While 后面的<循环条件>是否为 True。如果为 True，则继续执行循环体中的语句；否则跳出循环，执行 Loop 后面的语句。

**例 3.19** 下面使用 Do While...Loop 语句计算  $1+2+3+\dots+50$  的值，代码如下。（实例位置：光盘\TM\sl\3\13）



```

Private Sub Form_Click()
    Dim i%, mySum%           '定义整型变量
    Do While i < 50
        mySum = mySum + i    '每循环一次, 变量 mySum 就加变量 i
        i = i + 1            '每循环一次, 变量 i 就加 1
    Loop
    Print mySum               '输出计算结果
End Sub

```

结果为: 1225

## 2. Do...Loop While

这是“当型循环”的第二种形式, 它与第一种形式的区别在于 While 关键字与<循环条件>在 Loop 关键字后面。

语法格式如下:

```

Do
    循环体 1
    <Exit Do>
    循环体 2
Loop While <循环条件>

```

该语句的执行流程如图 3.35 所示。

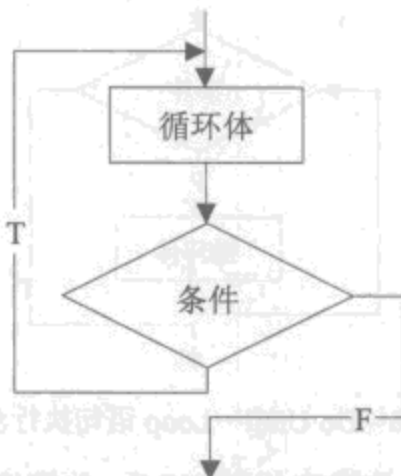


图 3.35 Do...Loop While 语句执行流程图

从上述流程图可以看出, Do...Loop While 语句的执行过程为:

当程序执行 Do...Loop While 语句时, 首先执行一次循环体, 然后判断 While 后面的<循环条件>。如果其值为 True, 则返回到循环开始处再次执行循环体, 否则跳出循环, 执行 Loop 后面的语句。

**例 3.20** 下面使用 Do...Loop While 语句计算  $1+2+3+\dots+\text{myVal}$  的值, myVal 值通过 InputBox 输入对话框输入, 代码如下。(实例位置: 光盘\TM\sl\3\14)

```

Private Sub Form_Click()
    Dim i%, mySum%, myVal%           '定义整型变量
    myVal = Val(InputBox("请输入一个数: ")) '得到输入的值
    Do
        i = i + 1                     '每循环一次, 变量 i 就加 1
    Loop While i <= myVal

```

```

    mySum = mySum + i          '每循环一次，变量 mySum 就加变量 i
    Loop While i < myVal
    Print mySum                '输出计算结果
End Sub

```

上述代码中，如果 myVal 的值大于或等于 256 时，程序会出现“溢出错误”，因为代码中变量 myVal 定义的是整型，整型的有效范围是 -32768~32768，因此出现错误。

解决办法有两种：一种是将变量 myVal 定义为长整型，这样输入值的有效范围会大些；另一种就是在代码  $i = i + 1$  后面加上代码 `If myVal >= 256 Then Exit Do`，判断如果变量 myVal 的值大于或等于 256，则使用 Exit Do 语句退出循环。

### 3. Do Until...Loop

使用 Until 关键字的 Do...Loop 循环被称为“直到型循环”。

语法格式如下：

```

Do Until <循环条件>
    循环体 1
    <Exit Do>
    循环体 2
Loop

```

该语句的执行流程如图 3.36 所示。

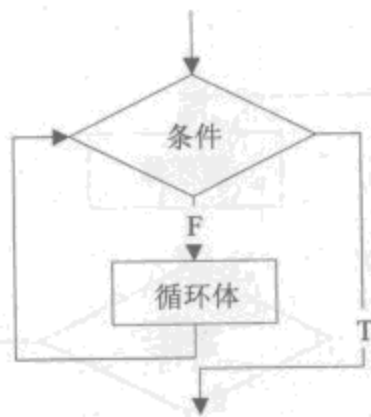


图 3.36 Do Until...Loop 语句执行流程图

从上述流程图可以看出，用 Until 关键字代替 While 关键字的区别在于，当循环条件的值为 False 时才进行循环，否则退出循环。

**例 3.21** 下面用 Do Until...Loop 语句计算阶乘  $n!$ ， $n$  值通过 InputBox 输入对话框输入，代码如下。（实例位置：光盘\TM\sl3\15）

```

Private Sub Form_Click()
    Dim i%, n%, mySum%          '定义整型和长整型变量
    n = Val(InputBox("请输入一个数: ")) '得到输入的值
    mySum = 1                   '给变量 mySum 赋初值
    Do Until i = n
        i = i + 1               '每循环一次，变量 i 就加 1
        mySum = mySum * i       '每循环一次，变量 mySum 就乘以变量 i
        If n > 12 Then Exit Do   '如果输入数大于 12，就退出循环
    Loop

```

```
Print mySum          '输出计算结果
End Sub
```

#### 4. Do...Loop Until

Do...Loop Until 语句是“直到型循环”的第二种形式。

语法格式如下：

```
Do
    循环体 1
    <Exit Do>
    循环体 2
Loop Until <循环条件>
```

该语句的执行流程如图 3.37 所示。

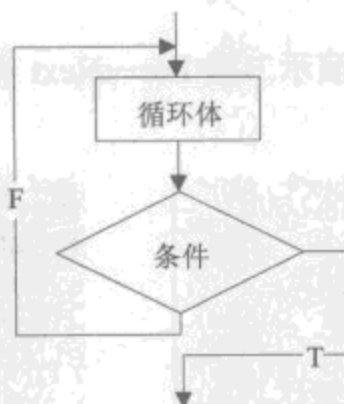


图 3.37 Do...Loop Until 流程图

从上述流程图可以看出，Do...Loop Until 语句的执行过程为：

当程序执行 Do...Loop Until 语句时，首先执行一次循环体，然后判断 Until 后面的<循环条件>，如果其值为 False，则返回到循环开始处再次执行循环体，否则跳出循环，执行 Loop 后面的语句。

**注意：**因为浮点数和精度问题，两个看似相等的值实际上可能不精确相等。所以，在构造 Do...Loop 循环条件时要注意，如果测试的是浮点类型的值，要避免使用相等运算符“=”，应尽量使用运算符“>”或“<”进行比较。

#### 3.4.4 多重循环

在一个循环体内又包含了循环结构称为多重循环或循环嵌套。循环嵌套对 For...Next 语句、Do...Loop 语句均适用。在 VB 中，对嵌套的层数没有限制，可以嵌套任意多层。嵌套一层称为二重循环，嵌套两层称为三重循环。

**注意：**(1) 外循环必须完全包含内循环，不可以出现交叉现象。

(2) 内循环与外循环的循环变量名称不能相同。

下面介绍几种合法且常用的二重循环形式，如表 3.3 所示。

表 3.3 合法的循环嵌套形式

(1) For i= 初值 To 终值 For j=初值 To 终值 循环体 Next j Next i	(2) For i= 初值 To 终值 Do While/Until 循环体 Loop Next i	(3) Do While/Until For i=初值 To 终值 循环体 Next i Loop
(4) Do While/Until Do While/Until 循环体 Loop Loop	(5) Do For i=初值 To 终值 循环体 Next i Loop While/Until	(6) Do Do While/Until 循环体 Loop Loop While/Until

例 3.22 下面通过一个简单的例子演示二重 For...Next 循环，代码如下。多重循环的道理相同。（实例位置：光盘\TM\sl\3\16）

第一种形式：

```
Private Sub Form_Click()
    Dim i%, j%      '定义整型变量
    For i = 1 To 3   '外层循环
        Print "i="; i '输出变量 i
        For j = 1 To 3 '内层循环
            Print Tab; "j="; j '输出变量 j
        Next j
    Next i
End Sub
```

第二种形式：

```
Private Sub Form_Click()
    Dim i%, j%      '定义整型变量
    For i = 1 To 3   '外层循环
        For j = 1 To 3 '内层循环
            Print "i="; i; "j="; j '输出变量 i 和 j
        Next j
        Print         '输出空行
    Next i
End Sub
```

上述两段程序只是输出形式不同（即输出语句上有些区别），运行结果分别如图 3.38 和图 3.39 所示。从这两段程序的执行情况可以看出，外层循环执行一次（如 i=1），内层循环要从头循环一遍（如 j=1、j=2 和 j=3）。

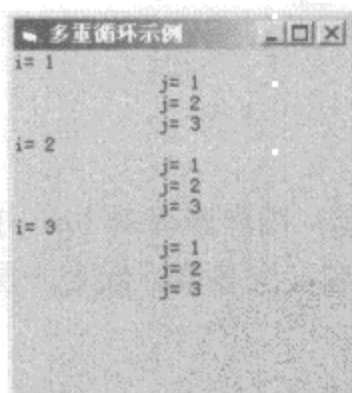


图 3.38 多重循环示例 (1)

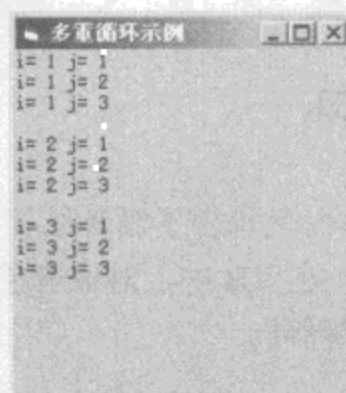


图 3.39 多重循环示例 (2)

### 3.4.5 选择结构与循环结构的嵌套

在 VB 中，所有的控制结构（包括 If 语句、Select Case 语句、Do...Loop 语句、For...Next 语句等）都可以嵌套使用。

**例 3.23** 将 100 元钱换成零钱（5 元、10 元、20 元中的任意多个面值）有很多种换法。组成 100 元的零钱中，最多有 20 个 5 元、10 个 10 元和 5 个 20 元。判断所有的组合中，总和正好是 100 元的，代码如下。这类方法称为“穷举法”，也称为“列举法”。（实例位置：光盘\TM\sl\3\17）

```
Private Sub Form_Click()
    Dim x%, y%, z%, n%
    Print "5 元个数", "10 元个数", "20 元个数"
    For x = 0 To 20
        For y = 0 To 10
            For z = 0 To 5
                If 5 * x + y * 10 + z * 20 = 100 Then
                    n = n + 1
                    Print x, y, z
                End If
            Next z
        Next y
    Next x
    Print "共有" & n & "种换法"
End Sub
```

'定义整型变量  
'输出标题  
'5 元的个数  
'10 元的个数  
'20 元的个数  
'满足条件  
'满足条件的组合数  
'输出结果  
'输出满足条件的组合数

上述程序使用了三重 For...Next 循环，循环计数器变量分别为 x、y、z，代表 5 元、10 元和 20 元的个数，20 个 5 元、10 个 10 元和 5 个 20 元之内共有  $21 \times 11 \times 6 = 1386$  种组合，内嵌 If...Then 判断总和正好等于 100 元的只有 36 种，如图 3.40 所示。

5元个数	10元个数	20元个数
0	0	5
0	2	4
0	4	3
0	6	2
0	8	1
0	10	0
2	0	4
2	2	3
2	4	2
2	6	1
2	8	0
4	0	3
4	2	2
4	4	1
4	6	0
6	0	2
6	2	1
6	4	0
8	0	1
8	2	0
10	0	0
12	0	0
14	0	0
16	0	0
18	0	0
20	0	0

共有36种换法

图 3.40 兑换零钱的换法

### 3.5 其他辅助控制语句

教学录像：光盘\TM\lx\3\其他辅助控制语句.exe

本节主要介绍其他辅助控制语句，包括跳转语句 GoTo、复用语句 With...End With、退出语句 Exit 和结束语句 End。



### 3.5.1 跳转语句 GoTo

GoTo 语句使程序无条件跳转到过程中指定的语句行执行，语法格式如下。

GoTo <行号|行标签>

 说明：(1) GoTo 语句只能跳转到它所在过程中的行。

(2) 行标签是任何字符的组合，不区分大小写，必须以字母开头，以冒号“:”结尾，且必须放在行的开始位置。

(3) 行号是一个数字序列，且在使用行号的过程内该序列是唯一的。行号必须放在行的开始位置。

(4) 太多的 GoTo 语句，会使程序代码不容易阅读及调试。应尽可能少用或不用 GoTo 语句。

例 3.24 在程序中使用 GoTo 语句，代码如下。

```
Private Sub Command1_Click()  
    GoTo I1          '程序跳转到 I1 标签下的语句  
End  
Exit Sub           'Exit Sub 的作用是立即退出 Command1_Click 的 Sub 过程  
I1:  
    Print "没有退出"  
End Sub
```

当程序执行到 GoTo 语句时，程序跳转到 I1 标签下的语句去执行，而不执行 End 语句结束程序。

### 3.5.2 复用语句 With...End With

With 语句是在一个定制的对象或一个用户定义的类型上执行的一系列语句，其语法格式如下。

```
With <对象>  
    [<语句组>]  
End With
```

对象是必要参数，表示一个对象或用户自定义类型的名称；语句组是可选参数，是要在对象上执行的一条或多条语句。

With 语句可以嵌套使用，但是外层 With 语句的对象或用户自定义类型会在内层的 With 语句中被屏蔽住，所以必须在内层的 With 语句中使用完整的对象或用户自定义类型名称来引用在外层的 With 语句中的对象或用户自定义类型。

例 3.25 嵌套使用 With 语句，在窗体 Load 事件中设置按钮与窗体的部分属性，代码如下。（实例位置：光盘\TM\sl\3\18）

```
Private Sub Form_Load()  
    With Form1          '外层 With 语句  
        .Height = 10000: .Width = 10000
```

```

With Command1                                '内层 With 语句
    .Height = 2000: .Width = 2000
    .Caption = "按钮高度与宽度都是 2000"
    Form1.Caption = "窗体高度与宽度都是 10000"
End With
End With
End Sub

```

在外层嵌套 With 语句中直接设置 Form1 窗体的高度和宽度，在内层嵌套 With 语句中直接设置 Command1 按钮的高度、宽度和显示的标题。在设置 Form1 的显示标题时需要写入窗体的名称。

### 3.5.3 退出语句 Exit

Exit 语句用来退出 Do...Loop、For...Next、Function、Sub 或 Property 代码块，其类型及作用如表 3.4 所示。

表 3.4 Exit 语句类型及作用

语 句 类 型	作 用
Exit Do	退出 Do...Loop 循环的一种方法，只能在 Do...Loop 循环语句中使用。Exit Do 语句会将控制权转移到 Loop 语句之后的语句。当 Exit Do 语句用在嵌套的 Do...Loop 循环语句中时，Exit Do 语句会将控制权转移到 Exit Do 语句所在位置的外层循环
Exit For	退出 For...Next 循环的一种方法，只能在 For...Next 或 For Each...Next 循环中使用。Exit For 语句会将控制权转移到 Next 语句之后的语句。当 Exit For 语句用在嵌套的 For...Next 或 For Each...Next 循环中时，Exit For 语句将控制权转移到 Exit For 语句所在位置的外层循环
Exit Function	立即从包含该语句的 Function 过程中退出。程序会从调用 Function 过程的语句之后的语句继续执行
Exit Property	立即从包含该语句的 Property 过程中退出。程序会从调用 Property 过程的语句之后的语句继续执行
Exit Sub	立即从包含该语句的 Sub 过程中退出。程序会从调用 Sub 过程语句之后的语句继续执行

例 3.26 在 For...Next 循环语句中，当满足某种条件时，可以使用 Exit For 语句退出循环，如下面的代码。

```

For i = 1 To 100
    If i = 50 Then Exit For      '当 i=50 时退出循环
Next i

```

### 3.5.4 结束语句 End

End 语句用来结束一个过程或块。End 语句与 Exit 语句容易混淆，Exit 语句是用来退出 Do...Loop、For...Next、Function、Sub 或 Property 的代码块，并不说明一个结构的终止；而 End 语句是终止一个结构。End 语句类型及作用如表 3.5 所示。

表 3.5 End 语句类型及作用

语 句 类 型	作 用
End	停止执行。不是必要的，可以放在过程中的任何位置关闭程序
End Function	必要的语句，用于结束一个 Function 语句
End If	必要的语句，用于结束一个 If 语句块
End Property	必要的语句，用于结束一个 Property Let、Property Get 或 Property Set 过程
End Select	必要的语句，用于结束一个 Select Case 语句
End Sub	必要的语句，用于结束一个 Sub 语句
End Type	必要的语句，用于结束一个用户定义类型的语句（Type 语句）
End With	必要的语句，用于结束一个 With 语句

⚠ 注意：在使用 End 语句关闭程序时，VB 不调用 Unload、QueryUnload、Terminate 事件或任何其他代码，而是直接终止程序（代码）执行。

## 3.6 小结

通过本章的学习，读者会发现看似简捷的几个控制语句能够构筑出灵活多变的程序流程，因此学好控制语句是掌握好一门语言的基础。另外，在学习控制语句的过程中，应注意以下几点：

- （1）结构必须有入口和出口，即结构的起始语句和终止语句。
  - （2）结构的控制功能由结构本身决定，控制的依据是条件，控制机理是当前变量的值与预先设定的临界值进行比较，得到一个逻辑值“真”或“假”，然后根据这一逻辑值选择程序的执行流程。
  - （3）在程序中仅有结构语句没有任何意义，结构中必须“装”进内容。结构中可“装入”任何合法的语句及结构。
  - （4）在书写结构时尽量采用缩排格式，以增强代码的可读性。
- 最后，为了提高编程效率，建议读者牢记常用的编程定式，如累加定式、连乘定式、穷举法等。

## 3.7 练习与实践

1. 判断如果  $a > 60$ ，则  $I=1$ ；如果  $a > 70$ ，则  $I=2$ ；如果  $a > 80$ ，则  $I=3$ ；如果  $a > 90$ ，则  $I=4$ 。（答案位置：光盘\TM\sl3\19）
2. 编写一个程序，计算增加后的工资。要求基本工资大于或等于 1000 元，增加 20% 工资；若小于 1000 元，且大于或等于 800 元，则增加 15%；若小于 800，则增加 10% 工资。（答案位置：光盘\TM\sl3\20）
3. 将 a、b 两个值中较小的一个显示在文本框中。（答案位置：光盘\TM\sl3\21）
4. 使用 For...Next 语句计算库存表（kc）中所有药品的库存数量和库存金额，光盘中提供数据库 db\_medicine.mdb。（答案位置：光盘\TM\sl3\22）

# 第 4 章

## 数组与集合

(教学录像: 32 分钟)

数组与集合都可以用相同名称引用一系列变量，并用数字（索引）来识别它们。合理地使用数组可以简化程序，因为可以利用索引值设计一个循环，高效处理多种情况。集合提供了一种把相关对象分组的方法，在需要同时对多个对象进行处理时，非常有用。

通过阅读本章，您可以：

- » 熟悉数组的概念
- » 了解数组的分类
- » 掌握数组的声明方法
- » 掌握数组的基本操作方法
- » 掌握创建和使用控件数组的方法
- » 了解集合的概念
- » 掌握集合创建与使用的方法

## 4.1 数组的概述

 教学录像：光盘\TM\lx\4\数组的概述.exe

编程时，如果涉及数据不多，可以使用变量存取和处理数据，但对于成批的数据处理，就要用到数组。利用数组，可以简化程序、提高编程效率。本节主要介绍数组的概念和数组与简单变量的区别。

### 4.1.1 数组的概念

数组是一组相同数据类型变量的集合，而并不是一种数据类型。通常把数组中的变量称为数组元素，数组中每一个数组元素都有一个唯一的下标来标识自己，并且同一个数组中各个元素在内存中是连续存放的。在程序中使用数组名代表逻辑上相关的一些数据，用下标表示该数组中的各个元素，这使得程序书写简洁，操作方便，编写出来的程序出错率低，可读性强。

### 4.1.2 数组与简单变量的区别

数组与简单变量的声明方法类似，但它们之间仍有区别：

- ☒ 数组是以基本数据类型为基础，数组中每一个元素都属于同一数据类型。
- ☒ 数组的定义类似于简单变量的定义，所不同的是数组需要指定数组中的元素个数。

## 4.2 数组的分类

 教学录像：光盘\TM\lx\4\数组的分类.exe

按照是否可以重新定义数组上下标为标志，可将数组分为“静态数组”和“动态数组”。按照数组的维数可将数组分为“一维数组”、“二维数组及多维数组”。下面将对数组的各个类别进行讲解。

### 4.2.1 静态数组

#### 1. 静态数组的声明

静态数组使用 Dim 语句来声明。

语法格式如下：


```
Public|Private|Dim 数组名(下标)[As 数据类型]
```

声明静态数组语法中各部分说明如表 4.1 所示。




表 4.1 声明静态数组语法中参数及说明

参 数	说 明
Public Private Dim	只能选取一个而且必选其一。Public 用于声明可在工程中所有模块的任何过程中使用的数组；Private 用于声明只能在包含该声明的模块中使用的数组；Dim 用于模块或过程级别的数组。如果声明的是模块级别的数组，数组在该模块中的所有过程都是可用的；如果声明的是过程级别的数组，数组只能在该过程内可用
数组名	必要参数。数组的名称；遵循标准的变量命名约定
下标	必要参数。数组变量的维数，必须为常数；最多可以声明 60 维的多维数组，下标下界最小可为-32768，最大上界为 32767。可省略下界，默认值为 0。一维数组的大小是上界与下界之差加 1
数据类型	可选参数。变量的数据类型；可以是 Byte、布尔、Integer、Long、Currency、Single、Double、Date、String（对变长的字符串）、String * length（对定长的字符串）、Object、Variant、用户定义类型或对象类型

 说明：数组的下标由下界与上界组成，下界即数组中最小的数组元素，上界是数组中最大的数组元素。

在 VB 6.0 中可使用 Dim 语句声明几种不同数据类型、不同大小的数组。代码如下：

```
Dim a(3) As String      '声明 String 型数组 a, 包含 4 个数组元素即 a(0)、a(1)、a(2)、a(3)
Dim b(6)                '声明 Variant 型数组 b, 包含 7 个数组元素即 b(0)~b(6)
Dim c(2 To 7) As Integer '声明 Integer 型数组 c, 包含 6 个数组元素即 c(2)~c(7)
```

 注意：程序运行时访问静态数组，使用的数组元素下标不能超出定义的范围，否则程序将产生“下标越界”的错误。

## 2. 静态数组的使用

例 4.1 下例使用冒泡排序法，实现当单击“排序”按钮时，对包含十个数组元素的数组进行排序，并将结果输出在 TextBox 控件中。程序代码如下：（实例位置：光盘\TM\sl\4\1）

```
Dim a(9) As Long      '声明模块级数组 a
Private Sub Command1_Click()
    Dim i As Long, j As Long, b As Long
    For i = 1 To 9
        For j = 0 To 9 - i
            If a(j) < a(j + 1) Then
                b = a(j)
                a(j) = a(j + 1)
                a(j + 1) = b
            End If
        Next j
    Next i
    For i = 0 To 9
        Text1.Text = Text1.Text + CStr(a(i)) + "    "
        If i = 4 Then Text1.Text = Text1.Text + Chr(13) + Chr(10)
    Next i
End Sub
```

```

Next i
End Sub

Private Sub Command2_Click()
    Dim i As Long, l
    Text1.text = ""
    For i = 0 To 9
N:
        l = InputBox("请输入排序的 10 个数字, 这是第" & CStr(i + 1) & "个", "提示", "")
        If IsNumeric(l) Then
            a(i) = l
        Else
            MsgBox "请输入数字", vbOKOnly, "错误"
            GoTo N
        End If
    Next i
End Sub

```

## 4.2.2 动态数组

### 1. 动态数组的声明

动态数组使用 ReDim 语句声明。

注意：ReDim 语句是在过程级别中使用的语句。

语法格式如下：

ReDim [Preserve] 数组名(下标) [As 数据类型]

声明动态数组语法中各部分的说明如表 4.2 所示。

表 4.2 声明动态数组语法中参数及说明

参 数	说 明
Preserve	可选参数。关键字，当改变原有数组最末维的大小时，使用此关键字可以保持数组中原来的数据
数组名	必要参数。数组的名称；遵循标准的变量命名约定
下标	必要参数。数组变量的维数；最多可以声明 60 维的多维数组
数据类型	可选参数。变量的数据类型；可以是 Byte、Boolean、Integer、Long、Currency、Single、Double、Date、String（对变长的字符串）、String * length（对定长的字符串）、Object、Variant、用户定义类型或对象类型。所声明的每个变量都要有一个单独的 As 数据类型子句。对于包含数组的 Variant 而言，数据类型描述的是该数组的每个元素的类型，不能将此 Variant 改为其他类型

例如在程序中声明动态数组 a(10)，程序代码如下：

```
ReDim a(10) As Long
```

注意：动态数组只能改变其数组元素的多少，从而改变所占内存大小，不能改变其已经定义的数据

据类型。动态数组还可以使用 Dim 语句声明。在使用 Dim 语句声明动态数组时,将数组下标定义为空(给数组附以一个空维数表),并在需要改变这个数组大小时,使用 ReDim 语句重新声明这个数组的下标。

## 2. 动态数组的使用

例 4.2 实现:单击“输入”按钮,使用 InPutBox 函数弹出“输入”对话框,输入一些数据储存在动态数组 A 中,并将动态数组 A 中数据在 TextBox 控件中显示出来。程序运行效果如图 4.1 所示。(实例位置:光盘\TM\sl\4\2)

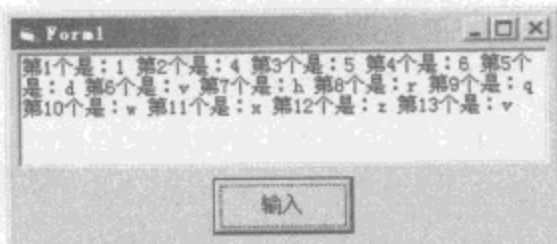


图 4.1 单击“输入”按钮输入 12 个字符串后的效果

程序代码如下:

```
Private Sub Command1_Click()  
    Text1.Text = ""           '清空文本框内容  
    Dim S As Long, i As Long '声明两个长整型变量  
    Dim A()  
    Do                        '循环体  
        ReDim Preserve A(S) '重新定义数组上下标,并保留原元素  
        A(S) = InputBox("请输入字符串,输入空串时结束","输入")  
        S = S + 1             '累加  
    Loop Until A(S - 1) = "" '当元素 A(S - 1)=空字符串时  
    For i = 0 To S - 2        '创建 For 循环体  
        Text1.Text = Text1.Text & "第" & CStr(i + 1) & "个是: " & CStr(A(i)) & "  
    Next i  
    Erase A  
End Sub
```

### 4.2.3 一维数组

#### 1. 一维数组概念

一维数组,是指在定义数组时,不论该数组是静态还是动态的数组,只要这个数组只有一个下标,那么该数组即为一维数组。数组在内存中是连续存放的。

例如,声明一个含有 4 个数组元素的数组 A, A 中各元素在内存中存放顺序如图 4.2 所示。

A(0)
A(1)
A(2)
A(3)

图 4.2 数组 A 中每个元素内存中存放的顺序代码

#### 2. 一维数组的声明

使用 Dim 语句或 ReDim 语句声明一维数组,代码如下:

Dim a() As Long	'声明动态一维数组
ReDim a(0 To 3) As Long	'重新为动态一维数组设置下标不上标
ReDim Preserve a(0 To 3) As Long	'重新为动态一维数组设置下标不上标并保留原元素中的数据
Dim b(3) As String	'声明静态一维数组
Dim c(5)	'声明默认 Variant 数据类型静态一维数组

### 3. 一维数组的使用

例 4.3 下例使用选择排序法, 实现当单击“排序”按钮时, 将一维数组 a 中各元素按从小到大的顺序输出在立即窗口中。程序代码如下: (实例位置: 光盘\TM\sl\4\3)

```
Option Explicit
Dim a(9) As Long
Private Sub Command1_Click()
    Dim i As Long, l As Long, n As Long
    For i = 0 To 9
        For l = i To 9
            If a(i) > a(l) Then
                n = a(i)
                a(i) = a(l)
                a(l) = n
            End If
        Next l
        Debug.Print a(i)
    Next i
End Sub
Private Sub Form_Load()
    a(0) = 564
    a(1) = 78 : a(2) = 45 : a(3) = 456412 : a(4) = 456 : a(5) = 1 : a(6) = 45 + 79 : a(7) = 12 : a(8) = 1 * 966 : a(9) = 65 / 5
    Dim i As Long
    For i = 0 To 9
        Label1.Caption = Label1.Caption & "第" & CStr(i + 1) & "是: " & CStr(a(i)) & " "
    Next i
End Sub
```

### 4.2.4 数组中的数组

数组的元素可以是任意的数据类型, 因此可以建立 Variant 数据类型数组。Variant 数据类型元素可以是其他数组。

例 4.4 建立两个数组, 一个包含整数, 而另一个包含字符串。然后声明第三个 Variant 数据类型数组, 并将整数和字符串数组放置其中: (实例位置: 光盘\TM\sl\4\4)

```
Private Sub Command1_Click()
    Dim i As Integer
    Dim intarray(5) As Integer
    For i = 0 To 4
        intarray(i) = 2008
    Next i
    '声明并放置字符串数组。
    Dim strarray(5) As String
```



For i = 0 To 4	'循环体
strarray(i) = "奥运"	'设置元素值
Next i	
Dim arr(1 to 2) As Variant	'声明拥有两个成员的新数组
arr(1) = intarray()	'将其他数组移居到数组
arr(2) = strarray()	
MsgBox arr(1)(2)	'显示结果 "2008"
MsgBox arr(2)(3)	'显示结果 "奥运"
End Sub	

## 4.2.5 二维数组及多维数组

### 1. 二维数组的概念

二维数组是指拥有两个下标的数组。可以把二维数组看作一个  $xy$  坐标系中的点。

例如，二维数组元素  $A(1, 3)$  可以看作是在  $xy$  坐标系中的点，如图 4.3 所示。

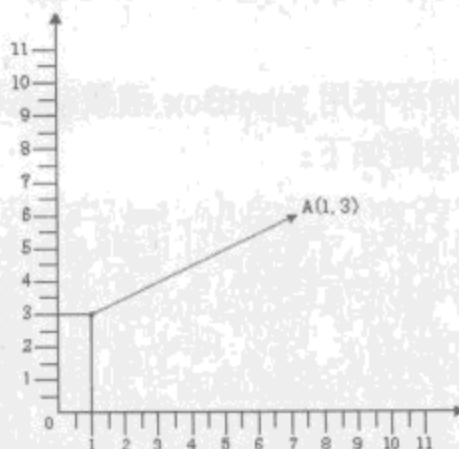


图 4.3  $xy$  坐标系表示二维数组

在定义数组时，将数组定义 3 个下标即三维数组，4 个下标即四维数组，依此类推，这些数组都可以称为多维数组。VB 中数组的维数最大限定为 60 个。

多维数组在使用时占用的内存空间较大，特别是 Variant 型多维数组，所以要谨慎使用。

### 2. 二维数组的声明

使用 Dim 语句或 ReDim 语句声明二维数组。代码如下：

Dim a(3, 4) As String	'声明静态二维数组
Dim b(5, 9)	'声明默认 Variant 数据类型静态二维数组
Dim c(,)	'声明二维动态数组
ReDim c(1 To 3, 0 To 2) As Long	'更改二维数组的上下标
ReDim Preserve d(0 To 3, 0 To 2) As Long	'更改二维数组的上下标并保留元素中的数据

### 3. 二维数组的使用

下例实现：单击“赋值输出”按钮，将二维数组 A 中所有元素赋值，并将 A 中每个数组元素的值输出在立即窗口中。程序代码如下：

Dim a(1 To 9, 1 To 9)	'声明二维数组
Private Sub Command1_Click()	
Dim i As Long, j As Long	'声明两个长整型变量



```

For i = 1 To 9                                '循环体
    For l = 1 To 9                            '循环体
        a(i, l) = l                          '设置元素值
        Debug.Print "a(" & CStr(i) & ", " & CStr(l) & ")=" & CStr(a(i, l))
    Next l
Next i
End Sub

```

#### 4. 多维数组的声明

使用 Dim 语句或 ReDim 语句声明多维数组，程序代码如下：

```

Dim a(,) As Long                                '声明多维动态数组
ReDim a(0 To 3, 0 To 2, 1 To 4) As Long        '更改多维数组的上下标
ReDim Preserve a(0 To 3, 0 To 2, 1 To 4) As Long
'更改多维数组的上下标并保留元元素中的数据
Dim b(3, 4, 6, 9) As Double                    '声明静态多维数组
Dim c(5, 9, 8, 1, 3)                          '声明默认 Variant 数据类型静态多维数组代码

```

#### 5. 多维数组的使用

下例实现的是，通过 For...Next 循环使用 InputBox 函数动态创建一个三维数组，并且将创建的三维数组显示在 TextBox 控件内。程序代码如下：

```

Dim a() As Long
Private Sub Command1_Click()
    Dim n As Long, i As Long                    '声明长整型变量
    Dim m As String                            '声明字符串类型变量
    Dim s(1 To 3) As Long                      '声明长整型数组
    For i = 1 To 3                              '循环体
        m = InputBox("请输入数组的第" & CStr(i) & "个下标，数值不要过大。", "多维数组")
        If IsNumeric(m) Then                    '判断对话框中输入的是否为数值
            s(i) = CLng(m)                     '设置元素值
        Else
            MsgBox "错误：输入不是数字。", vbOKOnly, "错误"
            Exit For                            '退出循环
        End If
    Next i
    On Error Resume Next
    ReDim a(s(1), s(2), s(3))                  '重新定义数组 a
    '将三维数组 a 显示在 TextBox 控件文本内
    Text1.Text = "a(" & CStr(s(1)) & ", " & CStr(s(2)) & ", " & CStr(s(3)) & ")"
End Sub

```

### 4.3 数组的基本操作

 教学录像：光盘\TM\4\数组的基本操作.exe

数组的基本操作包括：元素的输入输出、插入删除、查询排序等。下面将结合大量的举例，对几种数组的基本操作进行详细地讲解。

### 4.3.1 数组元素的输入

数组的输入是指给数组元素赋值，可以使用给变量赋值的方法为数组元素赋值。

例如，在数组元素较少的情况下可以像变量一样为数组赋值。代码如下：

```
Dim A(1 To 3) As String
A(1) = "奥运"
A(2) = "2008"
A(3) = "北京"
```

也可以使用 Visual Basic 所提供的函数为数组元素赋值，如 Array 函数。

Array 函数可以创建一个数组，并返回一个 Variant 数据类型的变量。

语法格式如下：

Array(arglist)

arglist: 一个数值表，各数值之间用“,” 分开。这些数值是用来给数组元素赋值的。当 arglist 中没有任何参数时，则创建一个长度为 0 的数组。

例如，将一个 Variant 型变量，使用 Array 函数赋值成 Variant 型数组，代码如下：


```
Dim A As Variant
A = Array(45, 2, 6, 7) 'A 中包含 4 个数组元素，各元素的值为：45, 2, 6, 7
```

 注意：数组 A 中第一个元素是 A(0)。使用 Array 函数创建的数组只能是 Variant 数据类型，返回的变量也只能是 Variant 型，如果这个变量不是 Variant 型，VB 将产生类型不匹配的错误。

在数组元素较多的情况下，可以使用循环结构语句为数组中每个元素赋值。

利用嵌套循环结构为二维数组中每个元素赋值。代码如下：

Dim A(1 To 9, 1 To 9)	'声明二维循环
Dim i As Long, l As Long	'声明两个长整型变量
For i = 1 To 9	'外层循环
For l = 1 To 9	'内层循环
A(i, l) = 0	'遍历二维数组中每个元素，并将其赋值为 0
Next l	
Next i	

 说明：可以在循环中使用 InputBox 函数与用户交互为数组赋值。

### 4.3.2 数组元素的输出

数组的输出是指将数组元素输出。数组的输出与访问变量类似，其方法也大致相同。

例 4.5 将 String 型数据类型数组 A 中的元素输出在立即窗口中，程序代码如下：（实例位置：光盘\TM\sl\4\5）

Dim A(1 To 3) As String	'声明字符串类型数组
A(1) = "奥运" : A(2) = "2008" : A(3) = "北京"	'设置元素值
Debug.Print A(1), A(2), A(3)	'显示元素值

数组的输出与输入类似,对于数组元素较多的数组,也可以使用循环语句结构将数组中的元素输出。多维数组也可采用嵌套循环结构将数组元素输出。

### 4.3.3 数组元素的插入

数组的插入是指将相同数据类型的元素插入到数组的指定位置。图 4.4、图 4.5 演示的是向数组插入元素前与插入元素后的效果。

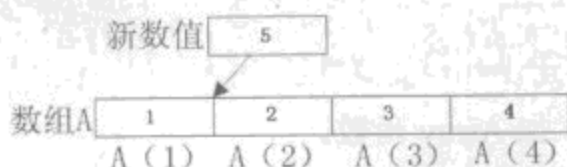


图 4.4 插入数组元素前



图 4.5 插入数组元素后

对数组进行插入操作时,数组的大小会被改变,所以插入操作只能针对动态数组进行。

**例 4.6** 向数组 A 中的指定位置插入一个新数值 5,并将插入后的数组 A 中各数组元素输出在立即窗口中。(实例位置:光盘\TM\sl\4\6)

程序代码如下:

```
Dim A() As Long
Private Sub Form_Load()
    ReDim Preserve A(1 To 4)
    A(1) = 1: A(2) = 2: A(3) = 3: A(4) = 4
    Dim n As Long: n = 5
    ReDim Preserve A(1 To 5)
    For i = 2 To 5
        Dim m As Long, m As long
        m = A(i)
        A(i) = n
        n = m
    Next i
    For i = 1 To 5
        Debug.Print "a(" & CStr(i) & ")=" & CStr(A(i)),
        ' "输出结果为 a(1)=1      a(2)=5      a(3)=2      a(4)=3      a(5)=4 "
    Next i
End Sub
```

'为动态数组 A 中元素赋值  
'声明长整型变量并赋值  
'调整数组上下标并保存原元素值  
'插入新数值  
'声明尺整型变量  
'将元素 A(i)值赋予变量 m  
'将 n 值赋予元素 A(i)元素  
'将变量 m 值赋予变量

'循环体  
'输出插入 5 后数组 A 中的元素

### 4.3.4 数组元素的删除

数组的删除是指删除数组中一个或多个元素。图 4.6、图 4.7 演示的是从数组中删除一个数组元素的前后情况。

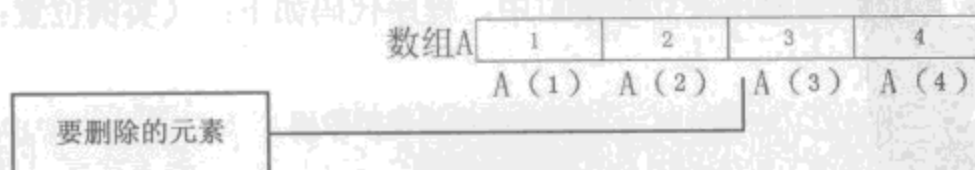


图 4.6 删除数组元素之前

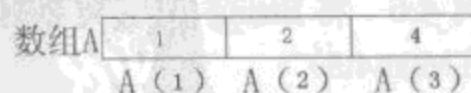


图 4.7 删除数组元素之后

例 4.7 从数组 A 中删除一个数组元素，并将删除后数组 A 中的各数组元素输出到立即窗口当中。  
(实例位置：光盘\TM\sl\4\7)

程序代码如下：

```
Dim A() As Long                                '声明变体类型动态数组
Private Sub Form_Load()
    ReDim A(1 To 4) As Long                    '设置数组上下标
    Dim i As Long                              '声明长整型变量
    A(1) = 1: A(2) = 2: A(3) = 3: A(4) = 4      '设置元素值
    Debug.Print vbNewLine & "删除元素前"
    For i = 1 To 4                              '循环体
        Debug.Print "a(" & CStr(i) & ")=" & CStr(A(i)),
    Next i
    A(3) = A(4)                                '将元素 A(4) 值赋予元素 A(3)
    ReDim Preserve A(1 To 3)                   '重新定义数组上下标并保留原元素
    Debug.Print vbNewLine & "删除元素后"
    For i = 1 To 3                              '循环体
        Debug.Print "a(" & CStr(i) & ")=" & CStr(A(i)),
    Next i
End Sub
```

### 4.3.5 数组元素的查找

数组的查找是指查找数组中指定的一个数组元素。可以使用循环语句结构对数组元素进行顺序查找，即遍历数组中每一个元素，查看数组中每一个数组元素是否与所要查找的数据相符，将符合的数组元素输出。

例 4.8 使用 For...Next 语句在包含 1~10 数组元素的 Long 型数据类型数组 A 中，查找一个值等于 19 的数组元素。本实例程序运行效果如图 4.8 所示。(实例位置：光盘\TM\sl\4\8)

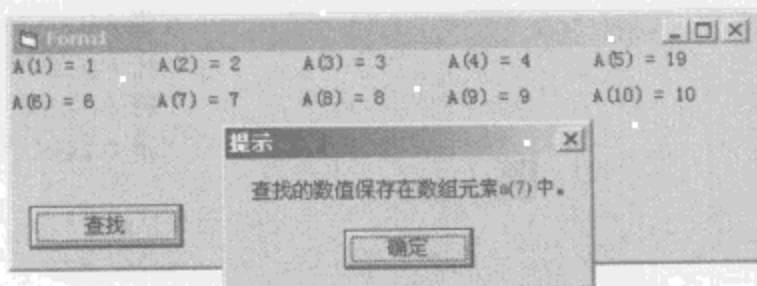


图 4.8 查询元素值为 19 的元素位置

代码如下：

```
For i = 1 To 10
    If a(i) = 19 Then
        MsgBox "查找的数值保存在数组元素 a(" & CStr(i) & ")中。", vbOKOnly, "提示"
        Exit For
    End If
Next i
```



### 4.3.6 数组元素的排序

数组的排序是指将数组中的数组元素按一定顺序进行排序，如由大到小、由上到下排序等。通常为数组排序可以使用选择排序法与冒泡排序法。

#### 1. 选择排序法

选择排序法指每次选择所要排序的数组中最大值（由小到大排序则选择最小值）的数组元素，将这个数组元素的值与前面的数组元素的值互换。

例如，表 4.3 演示的是使用选择排序的过程。

表 4.3 使用选择排序法为数组 A 排序

数组元素 排序过程	A (1)	A (2)	A (3)	A (4)	A (5)
起始值	3	2	7	9	5
第 1 次	9	2	3	7	5
第 2 次	9	7	2	3	5
第 3 次	9	7	5	2	3
第 4 次	9	7	5	3	2
排序结果	9	7	5	3	2

程序代码如下：

```
Dim i As Long, l As Long, n As Long
For i = 1 To 4
    For l = n To 5
        If A(i) < A(l) Then
            n = A(i)
            A(i) = A(l)
            A(l) = n
        End If
    Next l
Next i
```

'当 A(i)小于 A(l)时  
'将 A(i)值赋予变量 n  
'将 A(l)值赋予 A(i)  
'将 n 值赋予 A(l)

#### 2. 冒泡排序法

冒泡排序法指的是在排序时，每次比较数组中相邻的两个数组元素的值，将较大的排在较小的前面。例如，表 4.4 演示的是使用冒泡排序法进行数据排序的过程。

表 4.4 使用冒泡排序法为数组 A 排序

数组元素 排序过程	A (1)	A (2)	A (3)	A (4)	A (5)
起始值	3	2	7	9	5
第 1 次	3	7	9	5	2



续表

数组元素 排序过程	A (1)	A (2)	A (3)	A (4)	A (5)
第2次	7	9	5	3	2
第3次	9	7	5	3	2
第4次	9	7	5	3	2
排序结果	9	7	5	3	2

程序代码如下：

```

For i = 1 To 4
    For l = 0 To 3
        If A(l) < A(l + 1) Then
            n = A(l)
            A(l) = A(l + 1)
            A(l + 1) = n
        End If
    Next l
Next i

```

'当 A(l) 小于 A(l+1) 时  
'将 A(l) 值赋予变量 n  
'将 A(l+1) 值赋予 A(l)  
'将 n 值赋予 A(l+1)

### 3. 二分法排序

如果数组很小，使用上面的方法可行；但数组较大时，一个一个比较将浪费大量时间。对于数组元素的取值有顺序（由小到大或由大到小等）的数组，可以采用二分法查找数组元素。二分法是将所要查询的数值，先与位于数组中间的数组元素进行比较，根据比较结果在对前一半或后一半进行查找，然后继续取前一半或后一半中间的数组元素与查询的数值循环进行比较，直到查询到符合条件的结果。

```

Dim myarray(100) As Integer
Private Sub Command3_Click()
    Dim low, high, mid As Integer
    Dim found As Boolean
    low = 0
    high = UBound(myarray)
    found = False
    mid = (high + low) / 2
    Do While Not found And (high >= low)
        If CInt(Text1.Text) = myarray(mid) Then
            found = True
            MsgBox (mid)
            MsgBox (myarray(mid))
            Exit Do
        ElseIf CInt(Text1.Text) < myarray(mid) Then
            high = mid - 1
        Else
            low = mid + 1
        End If
        mid = (high + low) / 2
    Loop

```

'声明数组  
'声明整型变量  
'声明布尔类型变量  
'设置初始值  
'设置初始值  
'设置初始值  
'计算中间值  
'当没找到并且 high 值大于等于 low 值时  
'当输入的查询值与 myarray(mid) 相等时  
'设置为真  
'显示数组元素的下标  
'显示数组元素的值  
'跳出循环  
'当输入的查询值小于 myarray(mid) 时

```

End Sub
Private Sub Form_Load()
    Dim i As Integer           '声明整型变量
    For i = 0 To UBound(myarray) '遍历数组元素
        myarray(i) = i + 5     '设置数组元素
        Print myarray(i)       '显示元素值
    Next
End Sub

```

 注意：二分法只适用于有序数组。

## 4.4 记录数组

 教学录像：光盘\TM\lx\4\记录数组.exe

本节主要介绍记录数组的概念和记录数组的使用。

### 4.4.1 记录数组的概念

记录数组是指数据类型为自定义（记录型）数据类型的数组。与其他数组一样，记录数组也使用 Dim 语句或 ReDim 语句声明。记录数组声明后，这个数组中每个元素都拥有这个记录数据类型中的每个记录元素。

### 4.4.2 记录数组的使用

程序中使用记录数组与其他数组一样，但是需要先定义一个自定义（记录型）数据类型，然后在需要使用记录数组时进行声明。

例 4.9 定义一个 Peo 自定义数据类型，并在程序中使用。代码如下：（实例位置：光盘\TM\sl\4\9）

```

Private Type Peo           '创建自定义类型
    Nam As String          '声明字符串类型变量
    Age As Integer          '声明整型类型变量
End Type
Private Sub Command1_Click()
    Dim A(1 To 2) As Peo   '声明自定义类型数组
    A(1).Age = 20           '设置元素 A(1)的 Age 值
    A(1).Nam = "吴一"       '设置设置元素 A(1)的 Nam 值
    A(2).Age = 23           '设置设置元素 A(2)的 Age 值
    A(2).Nam = "方多"       '设置设置元素 A(2)的 Nam 值
    For i = 1 To 2          '循环体
        Print A(i).Nam ; A(i).Age & "岁"
    Next i
End Sub

```

## 4.5 控件数组

 教学录像：光盘\TM\lx\4\控件数组.exe

在程序中应用控件数组不仅使代码看起来统一，而且可以提高编码效率。这一节就介绍控件数组的概念、创建控件数组和使用控件数组。

### 4.5.1 控件数组的概念

控件数组是一组相同类型的控件，是使用相同名称并共享同一过程的集合。这个控件集合中的每一个控件，都可以称为该控件数组中的数组元素。

在创建控件数组时，系统会给这个控件数组中每一个控件唯一的索引（Index），即下标。这个索引的作用是用来区分控件数组中不同的控件。

### 4.5.2 创建控件数组

创建控件数组常使用如下两种方法。

#### 1. 复制粘贴法

通过复制粘贴控件，创建控件数组。具体步骤如下：

- (1) 在窗体上添加一个要创建控件数组的控件。
- (2) 选中该控件，单击鼠标右键，在弹出的菜单中选择“复制”命令。
- (3) 使用鼠标选中窗体，单击鼠标右键，在弹出的菜单中选择“粘贴”命令。此时会弹出一个如图 4.9 所示的提示对话框。单击“是”按钮后，即在窗体上添加了一个新的控件数组元素。
- (4) 重复执行步骤（3），直到添加完所需要的控件数组元素为止。

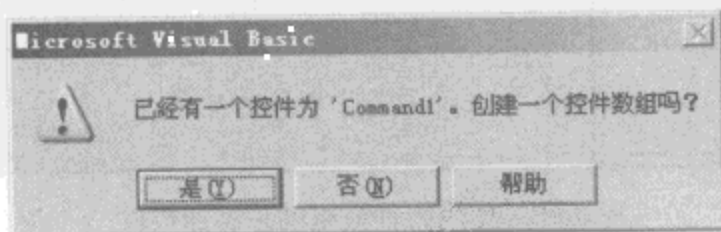



图 4.9 创建控件数组时弹出的对话框

 **注意：**要在容器类型控件内创建控件数组，需要选中容器控件（如 Frame（框架）控件等）执行“粘贴”命令。

#### 2. 设置控件 Name 属性

控件的 Name 属性在代码中用来标识控件的名字。通过将同类型控件 Name 属性设置为相同名称，也可以创建控件数组。创建的步骤如下：

- (1) 向窗体或容器控件中添加两个或多个同类型控件。

(2) 逐一选中添加上的每个控件，在属性窗口中设置这些控件的 Name 属性名称一致，即可完成创建控件数组的过程。在第一次出现 Name 属性同名时，也会出现如图 4.9 所示的提示对话框。单击“是”按钮即可创建控件数组。

### 4.5.3 使用控件数组

**例 4.10** 在单击 CommandButton 控件数组中的按钮时，通过 Index（索引）属性判断单击的是哪个按钮。程序代码如下：（实例位置：光盘\TM\sl\4\10）

```
Private Sub Command1_Click(Index As Integer)
    Select Case Index
        Case 0                                     '当控件索引值为 0 时
            MsgBox "你单击的是“确定”按钮", vbOKOnly, "提示"
        Case 1                                     '当控件索引值为 1 时
            MsgBox "你单击的是“取消”按钮", vbOKOnly, "提示"
    End Select
End Sub
```

控件数组在创建后，可以在程序执行时使用代码对控件数组中的元素进行添加。这样可以增加程序的灵活性。

**例 4.11** 选择所要添加的控件，在所要添加控件的位置单击鼠标左键，即可使用控件数组添加控件。运行程序，添加控件后的效果如图 4.10 所示。（实例位置：光盘\TM\sl\4\11）

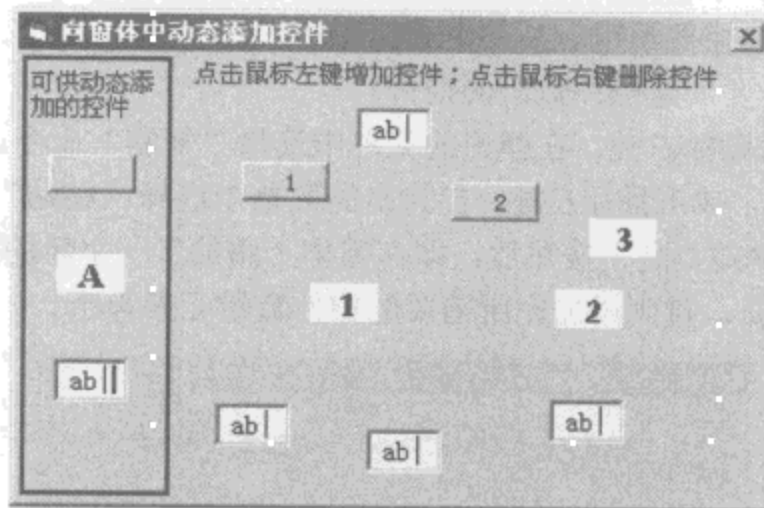


图 4.10 使用控件数组动态添加控件

程序代码如下：

```
Private Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long
Private Type POINTAPI                                     '声明自定义类型用于保存鼠标坐标
    X As Long
    Y As Long
End Type
Private Sub Command1_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    Lbl_edg(0).BorderStyle = 0
End Sub
```



```

Private Sub Form_Load()
    Text1(0).Locked = True
End Sub
Private Sub Label1_Click(Index As Integer)
    Text4.Text = 3
End Sub
Private Sub Label1_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 2 And Index > 0 Then
        Unload Label1(Index)           '删除 label 控件
    End If
    If Index = 0 Then
        Lbl_edg(1).BorderStyle = 1
    End If
End Sub
Private Sub Label1_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Index = 0 Then
        Lbl_edg(1).BorderStyle = 0
    End If
End Sub
Private Sub Text1_Click(Index As Integer)
    Text4.Text = 2
End Sub
Private Sub Text1_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 2 And Index > 0 Then
        Unload Text1(Index)           '删除 text 控件
    End If
    If Index = 0 Then
        Lbl_edg(2).BorderStyle = 1
    End If
End Sub
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim mouse As POINTAPI
    GetCursorPos mouse
    Text2.Text = Val(mouse.X) * 15 - Me.Left - 50
    Text3.Text = Val(mouse.Y) * 15 - Me.Top - 300
    If Text2.Text > 1000 Then
        If Text4.Text = 1 Then
            If Button = 1 Then           '添加 command 控件
                i = Command1.ubound + 1
                Load Command1(i)
                Command1(i).Left = Text2.Text
                Command1(i).Top = Text3.Text
                Command1(i).Caption = Str(i)
                Command1(i).Visible = True
            End If
        End If
        If Text4.Text = 2 Then
            If Button = 1 Then           '添加 text 控件
                i = Text1.ubound + 1

```




```

        Load Text1(i)
        Text1(i).Left = Text2.Text
        Text1(i).Top = Text3.Text
        Text1(i).Visible = True
    End If
End If
If Text4.Text = 3 Then
    If Button = 1 Then
        i = Label1.Ubound + 1
        Load Label1(i)
        Label1(i).Left = Text2.Text
        Label1(i).Top = Text3.Text
        Label1(i).Caption = Str(i)
        Label1(i).Visible = True
        '添加 label 控件
    End If
End If
End If
End Sub
Private Sub Command1_Click(Index As Integer)
    Text4.Text = 1
End Sub
Private Sub Command1_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 2 And Index > 0 Then
        Unload Command1(Index)
        '删除 command 控件
    End If
    If Index = 0 Then
        Lbl_edg(0).BorderStyle = 1
    End If
End Sub
Private Sub Command2_Click()
    End
End Sub
Private Sub Text1_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Index = 0 Then
        Lbl_edg(2).BorderStyle = 0
    End If
End Sub
End Sub

```

## 4.6 数组相关函数及语句

 教学录像：光盘\TM\4\数组相关函数及语句.exe

在 Microsoft Visual Basic 中有几个与数组相关的函数或语句，它们分别是：Array 函数、UBound 函数和 LBound 函数、Split 函数、Option Base 语句。下面将对上述的函数和语句进行介绍。

### 4.6.1 Array 函数

Array 函数可以创建一个数组，并返回一个 Variant 数据类型的变量。

语法格式如下：

Array(arglist)


arglist: 一个数值表，各数值之间用“,” 分开。这些数值是用来给数组元素赋值的。当 arglist 中没有任何参数时，则创建一个长度为 0 的数组。

例如，将一个 Variant 型变量，使用 Array 函数赋值成 Variant 型数组，代码如下：

```
Dim A As Variant
```

```
A = Array(45, 2, 6, 7)
```

'A 中包含 4 个数组元素，各元素的值为：45, 2, 6, 7

 注意：数组 A 中第一个元素是 A(0)。使用 Array 函数创建的数组只能是 Variant 数据类型，返回的变量也只能是 Variant 型，如果这个变量不是 Variant 型，Visual Basic 将产生类型不匹配的错误。

### 4.6.2 UBound 函数和 LBound 函数

UBound 函数可以返回指定数组中的指定维数可用的最大下标，其返回值为 Long 型。而 LBound 函数与 UBound 函数相反，该函数可以返回指定数组中的指定维数可用的最小下标，其值为 Long 型。

语法格式如下：

UBound(<数组>[, <维数>])

LBound(<数组>[, <维数>])

数组：必要参数。数组的名称，遵循标准的变量命名约定。

维数：可选参数。用来指定返回哪一维，默认值是 1（第一维）。UBound 函数返回指定维的上界；LBound 函数返回指定维的下界。

例如获取数组 A(1 To 100) 的上标和下标。

```
Dim A(1 To 100)
```

```
MsgBox "上标为: " & UBound(A) & " 下标为: " & LBound(A)
```

'返回结果为“上标为: 100 下标为: 1”

### 4.6.3 Split 函数

Split 函数返回一个下标从零开始的一维数组，此一维数组中包含了指定数目的子字符串。

语法格式如下：

Split(<表达式>[, <字符>[, count[, compare]]])

Split 函数语法中各部分的说明如表 4.5 所示。

表 4.5 Split 函数语法中参数及说明

参 数	说 明
表达式	必要参数。包含子字符串和分隔符的字符串表达式。如果表达式是一个长度为零的字符串(""), Split 则返回一个空数组, 即没有元素和数据的数组
字符	可选参数。用于标识子字符串边界的字符串字符。如果忽略, 则使用空格字符(" ")作为分隔符。如果字符是一个长度为零的字符串, 则返回的数组仅包含一个元素, 即完整的表达式字符串
count	可选参数。要返回的子字符串数, -1 表示返回所有的子字符串
compare	可选参数。数字值, 表示判别子字符串时使用的比较方式。关于其值, 请参阅表 4.6 中的设置值部分

compare 参数的设置值如表 4.6 所示。

表 4.6 compare 参数的设置

常 数	值	描 述
vbUseCompareOption	-1	用 Option Compare 语句中的设置值执行比较
vbBinaryCompare	0	执行二进制比较
vbTextCompare	1	执行文字比较
vbDatabaseCompare	2	仅用于 Microsoft Access

例 4.12 使用 Split 函数以 “.” 号作为分隔符将字符串拆分为字符串数组。程序代码如下: (实例位置: 光盘\TM\sl\4\12)

```

Dim A
Private Sub Form_Load()
A = Split("abc.def.ghi", ".", -1, 1)           '字符串拆分为数组
' A(0) 包含"abc"
' A(1) 包含"def"
' A(2) 包含 "ghi"
End Sub
Private Sub Command1_Click()
Dim i As Long
For i = 0 To 2                                '循环显示元素值
    Debug.Print A(i)
Next i
End Sub

```

#### 4.6.4 Option Base 语句

Option Base 语句用来指定声明数组时下标下界省略时的默认值。该语句是在模块中使用的语句。一个模块中只能出现一次, 该语句必须写在模块的所有过程之前, 而且必须位于带维数的数组声明之前。只对该语句所在模块中的数组下界有影响。

语法:

Option Base [0 | 1]

[0 | 1]: 设置数组下标中下界省略时的默认值。一般情况下数组的下标下界省略时的默认值为 0。

例如, 在声明数组之前使用该语句将下标中默认值设置为 1 后, 声明数组 A。代码如下:

```
Option Base 1
Dim A(4) As Long
```

数组 A 中的元素分别为: A(1), A(2), A(3), A(4)。

## 4.7 集合

 教学录像: 光盘\TM\lx\4\集合.exe

集合在数学中是一个很重要的概念,也是一种原始的概念。在 VB 中,集合仍然起着很重要的作用。集合提供了一种把相关对象分组的方法。它是将一系列相关的项构成组的一种方法,其优点在于可以对集合中的多个对象进行处理。

集合最常用来处理对象,但它也可以被用来处理任何类型的数据。集合中的每一个元素都是通过 Variant 来存储和操作的,这样就可以利用 Variant 的各种操作来简化集合的各项操作。同时,集合的大小也是不固定的,可以随时添加或者删除元素。

在集合操作中,索引(Index)和键值(Key)是很重要的两个概念,不仅可以通过索引访问集合中的元素,还可以通过键值来完成访问。集合根据其索引的不同,可以分为基于 0 的集合和基于 1 的集合。

### 4.7.1 集合的创建

Visual Basic 6.0 不仅提供了大量的预期定义的集合(如窗体集合、控件集合等),而且还提供了一个自定义的集合类型 Collection。Collection 对象是元素所组成的有序集合,可以把这个集合作为单元来引用。用户可以使用 Collection 类型来声明一个自定义的集合,并向集合中处理任何类型的数据或者对象。

集合声明如下:

```
Dim Col as New Collection
```

一旦声明了一个集合,就可以使用 Add 方法来增加集合中的元素,使用 Remove 方法来删除元素,使用 Item 方法来访问集合中的元素。

Collection 对象将每一项存储于 Variants 对象中。于是,能够添加到 Collection 对象里的内容列表就和能够存储到 Variants 中的内容列表是相同的。内容列表中包括标准数据类型、对象和数组,但不包括用户定义类型。通过建立自己的集合类型能够提供更强健的功能,以及额外的属性、方法和事件。

在 Visual Basic 中,各个不同的集合根据各自的特点提供了不同的属性和操作方法,但是每个集合都有相同的一组属性和方法,如表 4.7 所示。

表 4.7 Collection 对象方法说明

方 法	说 明
Count 方法	该方法包含了计划总的对象数目。该方法返回值为长整型,且在设计和运行时均为只读
Add 方法	向集合添加一个元素
Remove 方法	从集合中删除某一个元素
Item 方法	从集合中获取指定的元素



## 4.7.2 控件集合 (Controls 集合)

Controls 集合是在窗体上的所有对象的统称。Controls 集合是在打开新窗体或为窗体添加对象时自动创建的。

Controls 集合具有与 Collection 集合对象一样的 4 种方法, 它们分别是 Add、Count、Item 和 Remove。它们分别用于添加集合对象、获取对象总数、选择指定对象元素和移除集合对象。

通过使用 Controls 集合, 可精简控件管理部分的代码。下面将介绍几个关于 Controls 集合的常见的实例。

例 4.13 动态创建 TextBox 控件。(实例位置: 光盘\TM\sl\4\13)

```
Private Sub Form_Load()
    Dim text As TextBox           '声明对象变量
    Set text = Controls.Add("VB.TextBox", "Text") '动态创建 TextBox
    text.Visible = True          '设置为可见
End Sub
```

例 4.14 将窗体上所有的 TextBox 控件中显示“VB”, 代码如下: (实例位置: 光盘\TM\sl\4\14)

```
Private Sub Form_Load()
    Dim i As Object
    For Each i In Me.Controls '遍历窗体中所有控件
        '判断是否为 TextBox 对象
        If TypeName(i) = "TextBox" Then
            i.Text = "VB" '设置 Text 属性
        End If
    Next i
End Sub
```

窗体中有 3 个 TextBox 控件, 3 个 Label 控件, 将所有 TextBox 控件的 Text 属性设置为“VB”, 显示效果如图 4.11 所示。

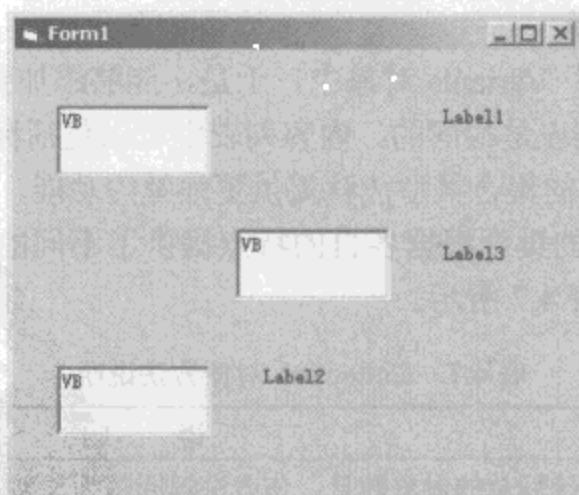


图 4.11 设置所有 TextBox 控件 Text 的属性



## 4.8 小结

本章主要对数组的声明与基本操作进行了讲解，并对集合的概念与集合的创建进行了介绍。学习本章，可以为以后的程序开发中，正确地使用数组或集合打下良好的基础，从而提高程序开发效率。

## 4.9 练习与实践

1. 查找下标为 0、上标为 99 的数组中值为“奥运”的元素索引值。（答案位置：光盘\TM\sl\4\15）
2. 窗体中有一 TextBox 控件，名称为 Text1 并且索引值为 0。动态添加两个 TextBox 控件。（答案位置：光盘\TM\sl\4\16）
3. 使用控件集合（Controls 集合）动态添加一个 TextBox 控件。（答案位置：光盘\TM\sl\4\17）



# 第 5 章

## 过程

(教学录像: 55 分钟)

过程能够使计算机完成特定的任务, 因此熟练地使用过程是编写高质量应用程序的基础。VB 中的过程包括事件过程、子过程(即通用过程)和函数过程等。

通过阅读本章, 您可以:

- » 了解什么是过程以及过程的分类
- » 掌握事件过程的建立和调用方法
- » 掌握建立子过程、调用子过程和其他模块中子过程的方法
- » 掌握如何建立函数过程, 调用函数过程, 以及函数过程与子过程的区别
- » 了解参数的传递过程
- » 学会编写嵌套过程和递归过程
- » 了解属性过程

## 5.1 认识过程

 教学录像：光盘\TM\lx\5\认识过程.exe

“过程”就是一个功能相对独立的程序逻辑单元，即一段独立的程序代码，Visual Basic 应用程序一般都是由过程组成的，如图 5.1 所示。

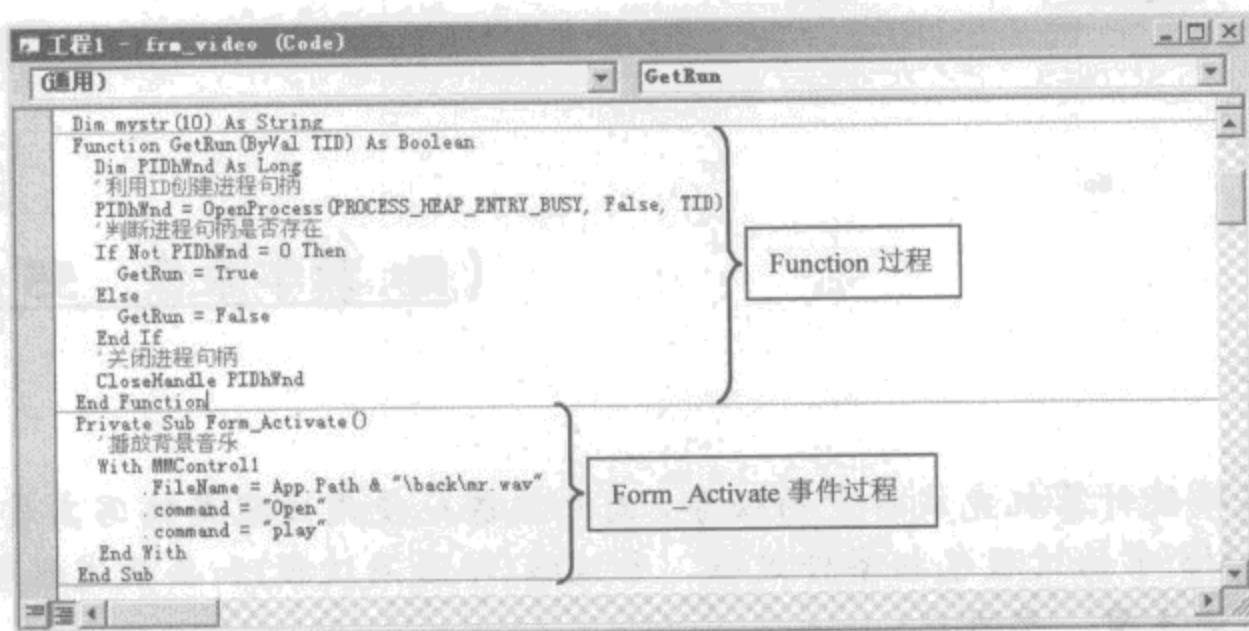


图 5.1 认识过程

Visual Basic 中的过程分为事件过程和通用过程。其中事件过程是当发生了某个事件（如单击鼠标的 Click 事件，窗体载入的 Load 事件、控件发生改变的 Change 事件）时，对该事件作出响应的程序段。例如图 5.1 中的代码，即当窗体被激活时，播放音乐。

通用过程是多个事件过程需要使用的一段相同的程序代码，它可以单独建立、供事件过程或其他过程调用。在 Visual Basic 中通用过程又分为子过程（Sub 过程）、函数过程（Function 过程）和属性过程（Property 过程）。

- ☒ Sub 过程不返回值。
- ☒ Function 过程返回一个值。
- ☒ Property 过程可以返回和设置窗体、标准模块以及类模块，也可以设置对象的属性。

## 5.2 事件过程

 教学录像：光盘\TM\lx\5\事件过程.exe

事件过程是附加在窗体和控件上的过程。当 Visual Basic 中的对象对一个事件的发生作出认定时，便自动用该事件的名字调用该事件的过程。例如单击一个按钮，便引发按钮的单击事件过程，如图 5.2 所示。

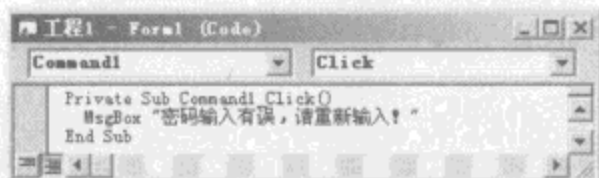


图 5.2 按钮单击事件过程

了解了事件过程，接下来介绍建立事件过程和调用事件过程。

### 5.2.1 建立事件过程

一个控件的事件过程将控件的（在 Name 属性中规定的）实际名字、下划线(\_)和事件名组合起来。例如，如果希望单击一个名为 cmdPlay 的命令按钮之后调用事件过程，则要使用 cmdPlay\_Click 过程。

一个窗体事件过程将词汇 Form、下划线和事件名组合起来。如果希望在单击窗体之后调用事件过程，则要使用 Form\_Click 过程（和控件一样，窗体也有唯一的名字，但不能在事件过程的名字中使用这些名字）。如果正在使用 MDI 窗体，则事件过程将词汇 MDIForm、下划线和事件名组合起来，如 MDIForm\_Load。

虽然可以自己编写事件过程，但使用 VB 提供的代码过程更方便，这个过程自动将正确的过程名包括进来。在“代码窗口”中，从“对象列表框”中选择一个对象，从“事件列表框”中选择一个事件，如图 5.3 所示，便可创建一个事件过程模板。



图 5.3 建立事件过程

**说明：**建议在开始为控件编写事件过程之前就设置好控件的 Name 属性。如果对控件附加一个过程之后又更改控件的名字，那么也必须更改过程的名字，以符合控件的新名字。否则，VB 无法使控件和过程相符。过程名与控件名不符时，过程就成为通用过程。

### 5.2.2 调用事件过程

事件过程可以使用 Call 语句进行调用，也可以直接使用过程名称。

#### 1. 使用 Call 语句

使用 Call 语句调用事件过程，语法格式如下。

Call <事件过程名>[(<参数列表>)]



例 5.1 窗体载入时，使用 Call 语句调用命令按钮（Command1）的 Click 事件过程。

```
Private Sub Form_Load()  
    Call Command1_Click  
End Sub
```

⚠ 注意：使用 Call 语句时，参数列表必须放在括号内。

## 2. 直接使用过程名称

直接使用过程名称调用事件过程，语法格式如下。

<事件过程名>[<参数列表>]

这里要强调的是，参数列表与使用 Call 语句中的参数列表正好相反，即参数列表不能用括号括起来。另外，调用事件过程语句中的实际参数列表必须在数目、类型、排列顺序上与事件过程语句的形式参数列表一致。

## 5.3 子过程（Sub 过程）

📺 教学录像：光盘\TM\lx\5\子过程（Sub 过程）.exe

子过程也可叫做 Sub 过程或通用过程，它用来完成特定的任务。使用 Sub 过程首先要建立它，然后直接使用过程名或使用 Call 语句调用。下面详细介绍 Sub 过程的建立和如何调用 Sub 过程。

### 5.3.1 建立子过程

要使用子过程，首先就要建立它。建立子过程有两种方法。

#### 1. 直接在代码窗口中输入

打开窗体或标准模块的代码窗口，将插入点定位在所有现有过程的外面，然后输入子过程即可，语法格式如下。

```
[Private|Public][Static]Sub 子过程名(参数列表)  
    <语句>  
    [Exit Sub]  
    <语句>  
End Sub
```

具体说明如下：

- ☑ Sub 是子过程的开始标记，End Sub 是子过程的结束标记，<语句>是具有特定功能的程序段，Exit Sub 语句用于退出子程序。
- ☑ 如果在子过程的前面加上 Private 语句，则表示它是私有过程，也就是该过程只在本模块中有效。如果在子过程的前面加上 Public 语句，则表示它是公用过程，可在整个应用程序范围内调用。

- ☑ 如果在子过程的前面加上 Static 语句,则表示该过程中的所有局部变量都是静态变量。
- ☑ 参数是调用子过程时给它传送的信息。过程可以有参数,也可以没有参数,没有参数的过程称为无参过程。如果带有多个参数,则各参数之间使用逗号隔开。参数可以是变量,也可以是数组。

## 2. 使用“添加过程”对话框

如果认为手工输入子过程比较麻烦,那么也可以通过“添加过程”对话框在代码窗口自动添加,其操作步骤如下:

- (1) 打开或新建一个 Visual Basic 工程。
- (2) 打开想要添加子过程的代码窗口。
- (3) 选择“工具”/“添加过程”菜单命令,打开“添加过程”对话框,如图 5.4 所示。
- (4) 在“名称”文本框中输入子过程名称,在“类型”选项组中选择过程类型,这里选择“子程序”单选按钮,在“范围”选项组中选择子过程的作用范围。如果选择“所有本地变量为静态变量”复选框,那么在子过程名称的前面将加上 Static 关键字。

(5) 设置完成后,单击“确定”按钮,则代码窗口中就会出现相应的子过程的框架。

例 5.2 在“名称”文本框中输入 SubComputeArea,选择范围是“私有的”,如图 5.5 所示。(实例位置:光盘\TM\sl\5\1)

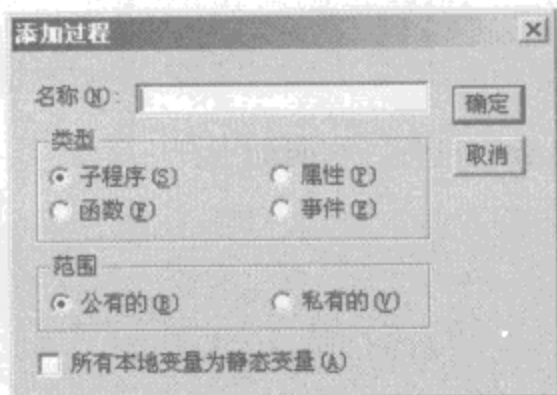


图 5.4 “添加过程”对话框(设置前)

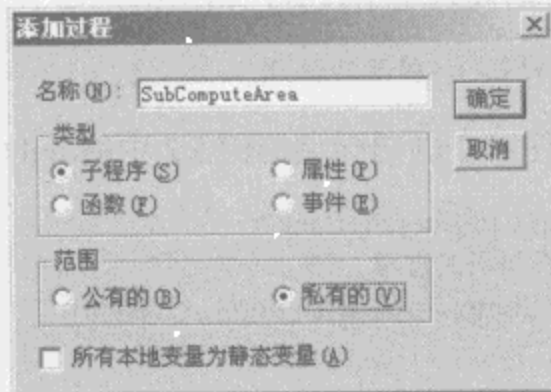


图 5.5 “添加过程”对话框(设置后)

⚠ 注意: 使用“添加过程”对话框创建过程,必须切换到代码窗口,否则“工具”下的“添加过程”菜单命令不可用。

单击“确定”按钮,代码窗口就会出现一个名为 SubComputeArea 的过程,如图 5.6 所示。

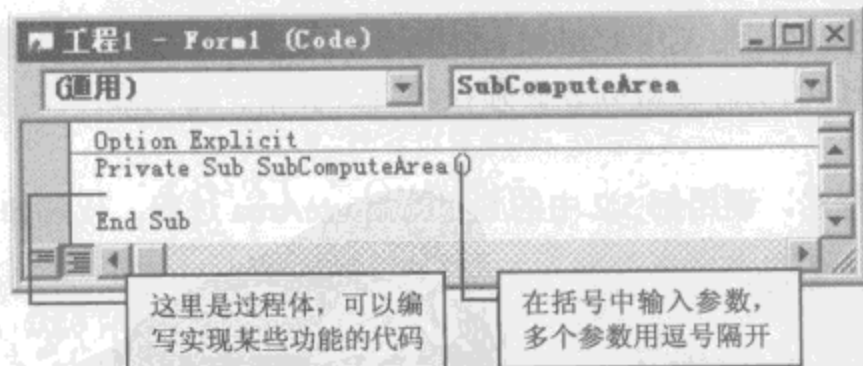


图 5.6 SubComputeArea 过程

从图 5.4 可以看出该过程没有参数，没有过程体，那么参数用户可以根据需要添加，过程体则必须自行编写。接下来就编写计算面积的过程，代码如下。

```
Option Explicit
'子过程的定义
'子过程带有两个参数
Private Sub SubComputeArea(Length, TheWidth)
    lblArea.Caption = Val(Length) * Val(TheWidth)           '计算矩形的面积
End Sub
```

上述过程使用了两个参数 Length 和 TheWidth，一个代表长度，一个代表宽度，过程体实现是将这两个参数相乘，然后将结果（也就是面积）赋值给 Label 控件，从而显示出来。

### 5.3.2 调用子过程

前面介绍了定义子过程的方法，那么定义完成后，就要考虑如何在程序中使用它。Sub 过程可以使用 Call 语句进行调用，也可以直接使用过程名称。


#### 1. 使用 Call 语句

使用 Call 语句调用子过程，语法格式如下。

```
Call <子过程名>[(<参数列表>)]
```

例 5.3 使用 Call 语句调用例 5.2 中的 SubComputeArea 过程，代码如下：

```
Private Sub CmdResult_Click()
    '调用计算面积的过程
    Call SubComputeArea(txtLength, txtWidth)
End Sub
```

 注意：使用 Call 语句时，参数列表必须放在括号内。另外，这里的参数是两个 TextBox 控件。

#### 2. 直接使用过程名称

直接使用过程名称调用子过程，语法格式如下。

```
<子过程名>[(<参数列表>)]
```

这里要强调的是，参数列表与使用 Call 语句中的参数列表正好相反，即参数列表不能用括号括起来。另外，子过程调用语句的实际参数列表必须在数目、类型、排列顺序上与子过程定义语句的形式参数列表一致。

例 5.4 直接使用过程名，调用例 5.2 中的 SubComputeArea 过程，代码如下：

```
Private Sub CmdResult_Click()
    '调用计算面积的过程
    SubComputeArea txtLength, txtWidth
End Sub
```

运行程序，在文本框中输入长和宽，单击“计算”按钮，结果将显示出来，如图 5.7 所示。

图 5.7 计算面积

### 5.3.3 调用其他模块中的子过程

#### 1. 调用窗体中的子过程

所有窗体模块的外部调用必须指向包含此过程的窗体模块。如果在窗体模块 Form1 中包含 MySub 子过程，则可使用下面的语句调用 Form1 窗体中的子过程。

```
Call Form1.MySub (参数列表)
```

#### 2. 调用类模块中的子过程

与调用窗体中的子过程类似，在类模块中调用子过程要调用与过程一致并且指向类实例的变量。例如，DemoClass 是类 Class1 的实例：

```
Dim DemoClass as New Class1
DemoClass.SomeSub
```

但是不同于窗体的是，在引用一个类的实例时，不能用类名作限定符。必须首先声明类的实例为对象变量（在这个例子中是 DemoClass）并用变量名引用它。

#### 3. 调用标准模块中的子过程

如果子过程名是唯一的，则不必在调用时加模块名。无论是在模块内，还是在模块外调用。

如果两个以上的模块都包含同名的子过程，那就有必要用模块名来限定了。在同一模块内调用一个公共过程就会运行该模块内的过程。例如，对于 Module1 和 Module2 中名为 CommonName 的子过程，从 Module2 中调用 CommonName 子过程则运行 Module2 中的 CommonName 子过程，而不是 Module1 中的 CommonName 子过程。此时就必须指定模块名，语句如下：

```
Module1.CommonName (参数列表)
Module2.CommonName (参数列表)
```

## 5.4 函数过程（Function 过程）

 教学录像：光盘\TM\lx\5\函数过程（Function 过程）.exe

Function 过程又称函数过程。函数过程与前面介绍的子过程基本一样，通过接下来的介绍您就会



看到，它也是用来完成特定功能的且独立的程序代码。与子过程不同的是，函数过程可以返回一个值给程序调用。下面就详细介绍函数过程的建立、如何调用函数过程及子过程与函数过程的区别。

### 5.4.1 建立函数过程

同样，使用函数过程也要先建立，方法也是两种。第一种是通过“添加过程”对话框，初步建立函数过程的框架，这与前面介绍子过程的方法基本一样，只是在“类型”选项组中选择“函数”单选按钮，其他用法都一样。

建立函数过程还可以使用 Function 语句，语法格式如下。

```
[Private|Public][Static] Function 函数名[(参数列表)][As 类型]
    <语句>
    [Exit Function]
    <语句>
End Function
```

从上述语句可以看出函数过程的形式与子过程的形式类似。Function 是函数过程的开始标记，End Function 是函数过程的结束标记。<语句>是具有特定功能的程序段，Exit Function 语句表示退出函数过程。As 子句决定函数过程返回值的数据类型，如果忽略 As 子句，函数过程返回值的数据类型为变体型。这里建议在实际编程中，使用 As 子句，以养成良好的编程习惯。

### 5.4.2 调用函数过程

函数过程也可称为用户自定义的函数，因此它与调用 VB 中的内部函数没有区别，也就是将一个函数的返回值赋给一个变量，语法格式如下。

```
变量名=函数名(参数列表)
```

这里需要说明的是：如果没有函数名，则函数过程将返回一个默认值：数值函数返回 0；字符串函数返回一个零长度字符串，也就是空字符串；变体函数则返回 Empty。如果在返回对象引用的函数过程中没有将对象引用通过 Set 赋给函数名，则函数过程返回 Nothing。

### 5.4.3 函数过程与子过程的区别

在对比函数过程和子过程之前，先来看一个实例。

**例 5.5** 将例 5.2 计算面积的子过程改为用函数过程实现。（实例位置：光盘\TM\sl\5\2）  
首先定义一个函数过程，代码如下。

```
'定义一个计算面积的函数，有两个参数
Private Function SubComputeArea(Length As Long, TheWidth As Long)
    SubComputeArea = Length * TheWidth
End Function
```



然后在“计算”按钮的 Click 事件过程中，调用函数过程，代码如下。

```
Private Sub CmdResult_Click()  
    '调用计算面积的函数过程  
    lblArea = SubComputeArea(txtLength, txtWidth)  
End Sub
```

将例 5.5 与例 5.2 进行对比，可以看出函数过程与子过程的区别，具体如下。

函数过程与子过程不同之处是：用函数过程可以通过过程名返回值，但只能返回一个值；子过程不能通过过程名返回值，但可以通过参数返回值，并可以返回多个值。它们的相同点是：子过程与函数过程都可以修改传递给它们的任何变量的值。

另外，还需要注意一点的是：无论是子过程还是函数过程，如果建立过程中，括号中没有参数，那么 VB 不会传递任何参数，但是，如果调用过程时使用了参数，则会出现错误。

## 5.5 参数的传递

 教学录像：光盘\TM\lx\5\参数的传递.exe

前面讲解建立过程和调用过程时，经常提到“参数”这个名词。什么是参数，参数是如何传递数据的？通过本节的讲解您便会了解。

### 5.5.1 认识参数

在调用一个有参数的过程时，参数就是在本过程中有效的局部变量，通过“形参和实参结合”达到传递数据的目的。例如下面的代码。

```
'定义一个用于计算面积的 Function 函数过程  
Private Function SubComputeArea(Length As Long, TheWidth As Long)  
    SubComputeArea = Length * TheWidth  
End Function
```

形式参数

```
Private Sub CmdResult_Click()  
    '调用计算面积的函数过程 SubComputeArea  
    lblArea.caption = SubComputeArea(txtLength, txtWidth)  
End Sub
```

实际参数

#### 1. 形参

从上述代码可以看出被调用过程中的形式参数就是形参，出现在 Sub 过程和 Function 过程中。形参列表中的各参数之间用逗号隔开，可以是变量名和数组名，但是定长字符串不可以。

#### 2. 实参

从上述代码可以看出在调用 Function 过程时，调用了两个参数将数据传递给了前面定义的形参，那么这两个参数就是实际参数，也就是实参。

实参列表与形参列表的对应变量名可以不同,但实参和形参的个数、顺序以及数据类型必须相同。因为“形实结合”是按照位置结合的,例如上述代码第一个实参 txtLength 与第一个形参 Length 结合,第二个实参 txtWidth 与第二个形参 TheWidth 结合。

如果实参和形参的个数不匹配,就会出现错误。例如在例 5.5 中调用函数过程时,把实参改为一个,代码如下。

```
Private Sub CmdResult_Click()  
    '调用计算面积的函数过程 SubComputeArea  
    lblArea.caption = SubComputeArea(txtLength)  
End Sub
```

运行程序,单击“计算”按钮,出现错误提示信息,如图 5.8 所示。

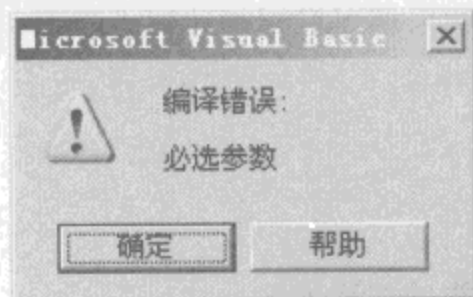


图 5.8 参数出错

出现上述错误,是由于前面定义的 Function 过程 SubComputeArea 有两个参数,而调用语句中只使用了一个。在实际编程过程中,一定要注意这个问题。

### 3. 参数的数据类型

前面介绍了实参和形参的个数、顺序以及数据类型必须相同,个数不同会出现错误,那么数据类型不同会如何呢?

(1) 创建过程时,如果没有声明形参的数据类型,那么数据类型默认为变体 (Variant) 型。

(2) 如果实参数据类型与形参数据类型不一致,则 VB 会按要求对实参进行数据类型转换,然后将转换后的值传递给形参。

### 4. 使用可选的参数

在前面的讲解过程中,讲到某个语句的语法时,经常会提到“可选的参数”、“必要的参数”。那么在定义过程时,参数也是可选的,只要参数列表中含有 Optional 关键字就可以。

语法:

```
Sub|Function 过程名(Optional 变量名)
```

例 5.6 将例 5.5 定义的函数过程 SubComputeArea 中的两个参数改为可选参数。

```
Private Function SubComputeArea(Length As Long, TheWidth As Long)
```

那么在“计算”按钮的 Click 事件过程中,下面的程序代码都是合法的。

```
lblArea.caption = SubComputeArea(txtLength)          '未提供第二个参数  
lblArea.caption = SubComputeArea(txtWidth)           '未提供第一个参数
```

如果未提供可选参数, 该参数将作为变体 (Variant) 型的 Empty 值, 不会出现如图 5.8 所示的编译错误。

注意: (1) 定义带可选参数的过程, 必须在参数表中使用 Optional 关键字。

(2) 可选参数必须放在参数表的最后, 而且必须是 Variant 类型。

## 5.5.2 参数按值和按地址传递

在 VB 中传递参数有两种方式, 即按值传递和按地址传递。其中按地址传递, 又称为“引用”。

### 1. 按值传递参数

按值传递使用 ByVal 关键字定义参数。使用时, 程序为形参在内存中临时分配一个内存单元, 并将实参的值传递到这个内存单元中。当过程中改变形参的值时, 则只是改变形参内存单元中的值, 实参的值不会改变。

例 5.7 下面用一个子过程 test, 来测试按值传递参数。(实例位置: 光盘\TM\sl\5\3)

```
Private Sub test(ByVal a As Integer, ByVal b As Integer)
    A = a+20:b = b+0
    Print "a=" & a, "b=" & b
End Sub
Private Sub cmdTest_Click()
    Dim num1 As Integer, num2 As Integer
    num1 = 10: num2 = 10
    Print "num1=" & num1, "num2=" & num2
    Call test(num1, num2)
    Print "num1=" & num1, "num2=" & num2
End Sub
```

上述代码中, test 过程中修改了形参 a 和 b 的值, a 和 b 是按值传递参数的, 单击“测试”按钮后, 从图 5.9 所示的窗体上显示的运行结果可以看出, 形参 a 和 b 的改变没有影响实参 num1 和 num2 的取值。

### 2. 按地址传递参数

按地址传递使用 ByRef 关键字定义参数。在定义过程时, 如果没有 ByVal 关键字, 默认的是按地址传递参数。

按地址传递参数, 是指把形参变量的内存地址传递给被调用的过程。形参和实参具有相同的地址, 即形参和实参共享同一段存储单元。

例 5.8 将例 5.7 按值传递改为按地址传递, 程序代码如下, 被调用的子过程 test 的代码不变。(实例位置: 光盘\TM\sl\5\4)

```
Private Sub test(a As Integer, b As Integer)
    ...
End Sub
Private Sub cmdTest_Click()
    Dim num1 As Integer, num2 As Integer
    num1 = 10: num2 = 10
```

‘此处省略了子过程代码’

```
Print "num1=" & num1, "num2=" & num2
Call test(num1, num2)
Print "num1=" & num1, "num2=" & num2
End Sub
```

上述代码中，test 过程中修改了形参 a 和 b 的值，a 和 b 是按地址传递定义的，所以，单击“测试”按钮后，从图 5.10 窗体上显示的运行结果可以看出，形参 a 和 b 的改变影响了实参 num1 和 num2 的取值，这是由参数传递方式所决定的。

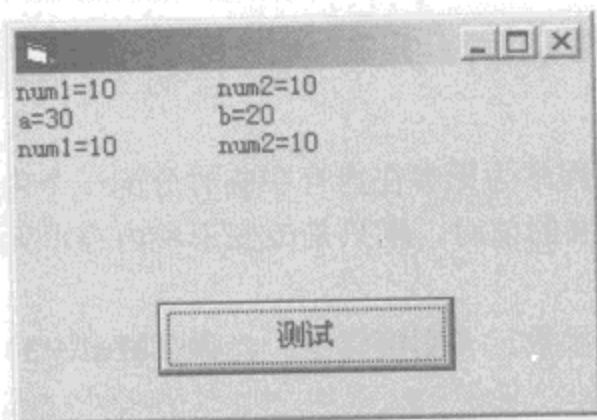


图 5.9 按值传递参数测试

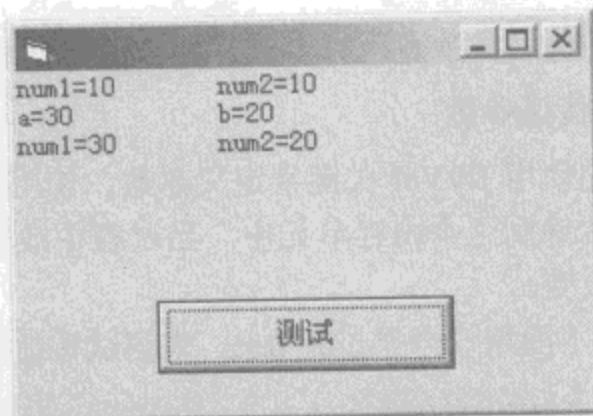


图 5.10 按地址传递参数测试

前面介绍了按值传递参数和按地址传递参数，那么究竟什么时候用传值方式，什么时候用传地址方式呢？没有硬性规定。下面几条规则可供参考。

- (1) 对于整型、长整型或单精度参数，如果不希望过程修改实参的值，则采用传值方式。而为了提高效率，字符串和数组应采用传地址方式。此外，用户定义的类型和控件只能通过地址传送。
- (2) 对于其他数据类型，可以采用两种方式传送。但是，建议此类参数最好用传值方式传送，这样可以避免错用参数。
- (3) 用函数过程可以通过过程名返回值，但只能返回一个值；子过程不能通过过程名返回值，但可以通过参数返回值，并可以返回多个值。但需要子过程返回值时，其相应的参数要用传地址方式。

### 5.5.3 数组参数

数组参数，就是在定义过程时，用数组作为形参出现在过程的形参列表。语法格式如下。

形参数组名() [As 数据类型]

形参数组对应的实参也必须是数组，数据类型与形参一致，实参列表中的数组不需要使用括号“()”。过程传递数组只能按地址传递，即形参与实参共有同一段内存单元。

例 5.9 下面使用函数过程 Average 计算员工平均年龄，代码如下。（实例位置：光盘\TM\sl\5\5）

```
Private Function Average(age() As Integer, n As Integer) As Integer
    '定义三个整型变量
    Dim i As Integer, aver As Integer, sum As Integer
    '使用循环语句求和
    For i = 0 To n - 1
        sum = sum + age(i)
```



```

Next i
'求平均数
aver = sum / n
Average = aver
End Function
Private Sub Command1_Click()
'定义一个用于存储员工年龄的数组
Dim Employees() As Integer
ReDim Employees(6)
'给数组赋值
Employees(0) = 20: Employees(1) = 28: Employees(2) = 30
Employees(3) = 24: Employees(4) = 25: Employees(5) = 35
'调用求平均数的函数
Text1 = Average(Employees, 6)
End Sub

```

上述代码中数组 Employees 作为实参传递给形参 age，形参 age 需要改变数组的维界，因此实参 Employees 必须用“Dim Employees() As Integer”语句声明为动态数组。按〈F5〉键，运行程序，结果如图 5.11 所示。

#### 5.5.4 对象参数

除了变量和数组作为实参传递给过程中的形参，Visual Basic 还允许对象（如窗体、控件等）作为实参传递给过程中的形参。

对象参数可以用引用方式，也可以用传递的方式，即在定义过程时，在对象参数的前面加 ByVal。

例 5.10 下面通过子过程 objectEna 设置 TextBox 和 CommandButton 控件不可用，代码如下。（实例位置：光盘\TM\sl\5\6）

```

Private Sub objectEna(obj1 As Object, obj2 As Object)
    obj1.Enabled = False: obj2.Enabled = False
End Sub
Private Sub Form_Load()
    objectEna Text1, Command1
End Sub

```

按〈F5〉键，运行程序，结果如图 5.12 所示。

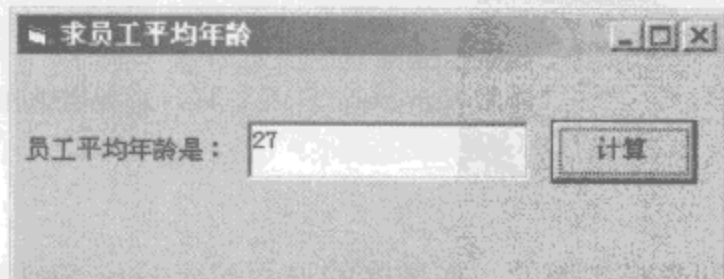


图 5.11 求员工平均年龄

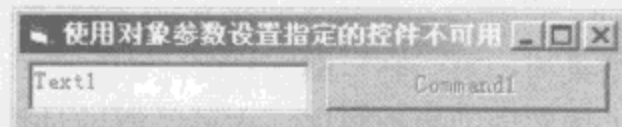


图 5.12 对象参数传递



## 5.6 嵌套过程

 教学录像：光盘\TM\lx\5\嵌套过程.exe

嵌套过程是指一个被调用的过程又调用了一个或若干个过程。例如：

```
Sub mySub1()
    ...
End Sub
Sub mySub2()
    Call mySub1
End Sub
Private Sub Form_Load()
    Call mySub2
End Sub
```

上面的代码中，mySub2 过程调用了 mySub1 过程，而 Form\_Load 事件过程又调用了 mySub2 过程。

例 5.11 下面通过嵌套过程实现数据排序，代码如下。（实例位置：光盘\TM\sl\5\7）

```
Private Sub Numbers_Change(a, b)
    Dim num1 As Integer           '定义一个整型变量
    num1 = a: a = b: b = num1     '交换变量
End Sub
Private Sub Numbers_Sort(arr As Variant)
    Dim i As Long, j As Long      '定义三个长整型变量
    For i = 0 To UBound(arr)
        For j = i + 1 To UBound(arr)
            If arr(j) < arr(i) Then '如果前一个数小于后一个数
                Call Numbers_Change(arr(j), arr(i)) '调用 Numbers_Change 过程，交换两个数
            End If
        Next j
    Next i
    For i = 0 To UBound(arr)
        Debug.Print arr(i)        '在立即窗口中输出数据
    Next i
End Sub
Private Sub Form_Load()
    Dim myarr                     '定义一个变量
    myarr = Array(45, 68, 120, 31) '给数组赋值
    Call Numbers_Sort(myarr)      '调用 Numbers_Sort 过程，对数据排序
End Sub
```

结果为：31 45 68 120

上述实现的数据排序，主要是通过对数据的比较和交换实现的。在排序的过程 Numbers\_Sort 中，使用循环语句多次嵌套调用过程 Numbers\_Change 实现数据的交换。

 说明：建议读者按〈F8〉键，通过单步调试，弄清楚整个嵌套过程的执行过程。

## 5.7 递归过程

教学录像：光盘\TM\lx\5\递归过程.exe

递归过程是指在过程中直接或间接地调用过程本身，也就是自己调用自己的过程。例如：

```
Private Function MyFunction(a As Integer)
    Dim b As Integer
    ...
    MyFunction = MyFunction(b)
    ...
End Function
```

在该过程中，MyFunction 函数过程中调用了 MyFunction 函数本身。使用递归过程时，要确保递归能终止，否则将出现“堆栈空间溢出”错误。

例 5.12 用递归的方法计算 1~30 中任意一个整数的阶乘，代码如下。（实例位置：光盘\TM\lx\5\8）

```
Function F(n As Integer) As Single
    If n > 1 And n <= 30 Then
        F = n * F(n - 1)
    Else
        F = 1
    End If
End Function
Private Sub Command1_Click()
    Text2.Text = F(Val(Text1.Text))
End Sub
```

'如果 n 大于 1 并小于等于 30  
'函数 F 调用自身  
'否则  
'函数 F 等于 1  
  
'调用函数过程，输出结果

程序执行流程如图 5.13 所示。按〈F5〉键，运行程序，输入 4，单击“计算”按钮，结果为“24”，如图 5.14 所示。

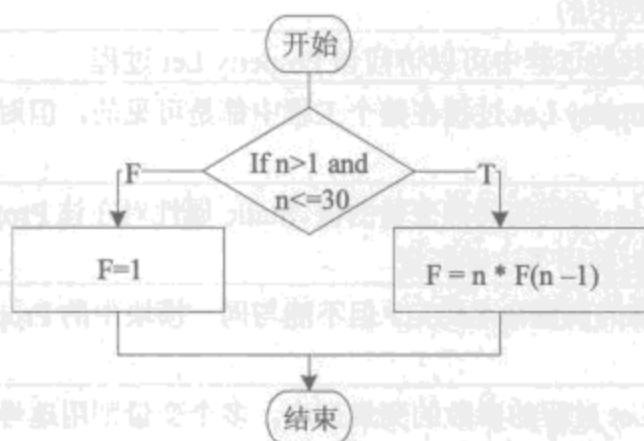


图 5.13 递归执行流程

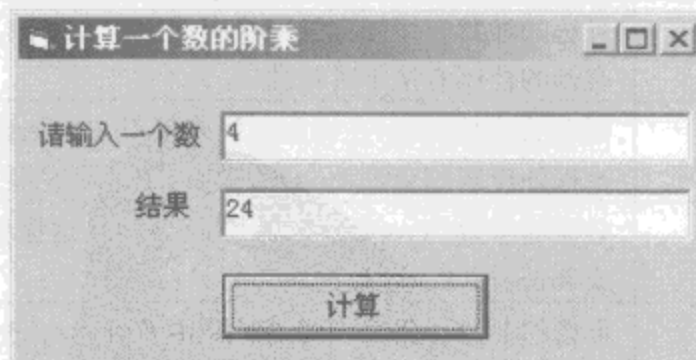


图 5.14 4 的阶乘

说明：递归过程可以转化为循环结构，但是，通常递归过程更快一些。另外，建议读者按〈F8〉键，通过单步调试，弄清楚整个递归过程的执行流程。

## 5.8 属性过程 (Property 过程)

 教学录像: 光盘\TM\lx\5\属性过程 (Property 过程) .exe

Property 过程也称属性过程, 该过程用于创建和操作类模块的属性。它包括以一个 Property Let、Property Get 或 Property Set 语句开头, 以一个 End Property 语句结束。下面介绍使用属性过程建立类的属性、使用类属性和创建只读属性和对象属性。

### 5.8.1 使用属性过程建立类的属性

建立类的属性主要使用 Property Let 语句和 Property Get 语句。下面分别介绍这两个语句。

#### 1. Property Let 语句

Property Let 语句用于声明 Property Let 过程的名称, 参数以及构成其主体的代码, 该过程用于给一个属性赋值。

语法格式如下:

```
[Public | Private | Friend] [Static] Property Set name ([arglist,] reference)
    <语句>
[Exit Property]
    <语句>
End Property
```

Property Let 语句中各参数的说明如表 5.1 所示。

表 5.1 Property Let 语句参数说明

参 数	说 明
Public	可选的参数。表示所有模块的所有其他过程都可访问该 Property Let 过程。如果在包含 Option Private 的模块中使用, 则这个过程在该工程外是不可使用的
Private	可选的参数。表示只有在包含其声明的模块的其他过程中可以访问该 Property Let 过程
Friend	可选的参数。只能在类模块中使用。表示该 Property Let 过程在整个工程中都是可见的, 但对于对象实例的控制者是不可见的
Static	可选的参数。表示在调用之间将保留 Property Let 过程的局部变量的值。Static 属性对在该 Property Let 过程外声明的变量不会产生影响, 即使过程中也使用了这些变量
name	必需的参数。Property Let 过程的名称; 遵循标准的变量命名约定, 但不能与同一模块中的 Property Get 或 Property Set 过程同名
arglist	可选的参数。代表在调用时要传递给 Property Let 过程的参数的变量列表。多个变量则用逗号隔开。Property Let 过程中的每个参数的名称和数据类型必须与 Property Get 过程中的相应参数一致
value	必需的参数。指用于给属性赋值的变量。当调用该过程时, 这个参数出现在调用表达式的右边。value 的数据类型必须和相应的 Property Get 过程的返回值类型一致
statements	可选的参数。Property Let 过程中执行的任何语句组

## 2. Property Get 语句

Property Get 语句用于声明 Property Get 过程的名称、参数以及构成其主体的代码，该过程获取一个属性的值。

语法格式如下：

```
[Public | Private | Friend] [Static] Property Get name [(arglist)] [As type]
    <语句>
[Exit Property]
    <语句>
End Property
```

Property Get 语句中各参数的说明与 Property Let 语句相似，这里就不再介绍了，下面重点介绍参数 type。该参数是可选的参数，用于表示 Property Get 过程的返回值的数据类型；可以是 Byte、Boolean、Integer、Long、Currency、Single、Double、Date、String（除定长）、Object、Variant 或任何用户定义类型。任何类型的数组都不能作为返回值，但包含数组的 Variant 可以作为返回值。

Property Get 过程的返回值类型必须与相应的 Property Let 过程（如果有）的最后一个（有时是仅有的）参数的数据类型相同，该 Property Let 过程将其右边表达式的值赋给属性。

**例 5.13** 下面使用 Property Let 过程和 Property Get 过程建立 Class1 类的标记属性，具体步骤如下。（实例位置：光盘\TM\sl\5\9）

- （1）创建一个工程，选择“工程”/“添加类模块”，在该工程中添加一个名为 Class1 的类模块。
- （2）在该类模块中编写如下代码。

```
Private i As Integer           '定义整型变量作为标记属性的值
Public Property Let mark(ByVal NewValue As Integer) '定义标记属性
    i = NewValue
End Property
Public Property Get mark() As Integer '获取标记属性的值
    mark = i
End Property
```

## 5.8.2 使用类属性

前面介绍了通过属性过程创建类的属性。接下来介绍给类的属性赋值和读取类的属性值。给类的属性赋值和读取的类属性值与窗体和控件基本相同，但有一点区别：给类的属性赋值和读取类的属性值，要先声明类，然后使用 Set 语句和 New 关键字，创建该类的一个新实例。

**例 5.14** 使用例 5.13 中建立的类的属性。当窗体载入时，为 Class1 类的标记属性赋值，然后获取该值，并显示出来，代码如下。（实例位置：光盘\TM\sl\5\10）

```
Private Sub Form_Load()
    Dim c1 As Class1
    Set c1 = New Class1
    c1.mark = 1 '赋给新的属性值
    MsgBox "类 class 1 的 mark 属性值为：" & c1.mark '显示属性值
End Sub
```



按〈F5〉键，运行程序，结果如图 5.15 所示。

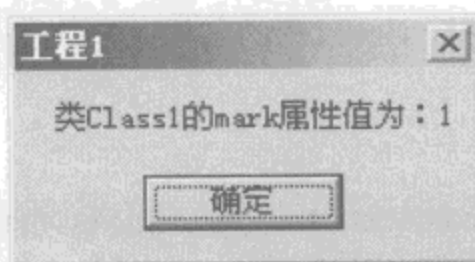


图 5.15 使用类属性

### 5.8.3 只读属性和对象属性

要创建只读属性，很简单，只要省略 Property Let 或（对于对象属性）Property Set 即可。

若要创建一个读写对象属性，应使用 Property Get 和 Property Set 语句，例如下面的代码。

```
Private mwdgWidget As Widget
Public Property Get Widget() As Widget
    Set Widget = mwdgWidget           'Set 语句被用来返回一个对象引用
End Property
Public Property Set Widget(ByVal NewWidget As Widget)
    Set mwdgWidget = NewWidget
End Property
```

## 5.9 小结

本章主要介绍了事件过程、子过程、函数过程、参数的传递、嵌套过程、递归过程和属性过程。读者应重点掌握子过程和函数过程；嵌套过程、递归过程和属性过程，对于初学者建议了解即可。

## 5.10 练习与实践

1. 编写一个用于获取字符串长度的函数。（答案位置：光盘\TM\5\11）
2. 利用子过程或函数过程，求 1~5 的阶乘之和。（答案位置：光盘\TM\5\12）
3. 第一年在银行存款 230 元，以后每年存 230 元，用递归计算 50 年后存款的总金额。（答案位置：光盘\TM\5\13）



# 第 6 章

## 内置函数

(教学录像: 36 分钟)

VB 6.0 中有很多内置的函数, 程序员利用这些函数可以更轻松地实现许多功能, 以减少代码的编写量, 使程序设计水平更上一个台阶。在本章中将针对在实际的开发中经常使用的数学函数、字符串函数、类型转换函数、判断函数、日期和时间函数、随机函数和格式化函数等函数进行详细地介绍。

通过阅读本章, 您可以:

- » 掌握常用数学函数的功能和使用
- » 掌握常用字符串函数的功能和使用
- » 掌握常用类型转换函数的功能和使用
- » 掌握常用判断函数的功能和使用
- » 掌握常用日期和时间函数的功能和使用
- » 掌握随机函数的功能和使用
- » 掌握格式化函数的功能和使用

## 6.1 数学函数

 教学录像：光盘\TM\lx\6\数学函数.exe

在程序设计中，用于数学计算的函数为数学函数，如 Abs 函数（求绝对值）、Exp 函数（e 的 n 次方）、Sgn 函数（返回符号）和 Sqr 函数（平方根）等。本节就介绍这几个比较常用的数学函数，VB 中还有很多的数学函数，有兴趣的读者可以查阅专门介绍函数的参考书。


### 6.1.1 Abs 函数（求绝对值）

Abs 函数用于返回参数的绝对值，其类型和参数相同。

语法：

Abs(number)

number：必要的参数，是任何有效的数值表达式。如果 number 包含 Null，则返回 Null；如果 number 是未初始化的变量，则返回 0。

 说明：一个数的绝对值是将正负号去掉以后的值。例如，ABS(-1) 和 ABS(1) 都返回 1。

例如，使用 Abs 函数计算数的绝对值。执行效果如图 6.1 所示。

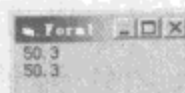


图 6.1 Abs 演示效果

```
Private Sub Form_Activate()  
    Print Abs(50.3)           '返回值为 50.3  
    Print Abs(-50.3)         '返回值为 50.3  
End Sub
```

### 6.1.2 Exp 函数（e 的 n 次方）

Exp 函数用于返回 Double 类型值，指定 e（自然对数的底）的某次方。

语法：

Exp(number)

number：必要的参数，number 是 Double 类型或任何有效的数值表达式。

 说明：

- (1) 如果 number 的值超过 709.782712893，则会导致错误发生。常数 e 的值大约是 2.718282。
- (2) Exp 函数的作用和 Log 的作用互补，所以有时也称作反对数。

例如，使用 Exp 函数计算 e（e ~ 2.71828）的某次方。执行效果如图 6.2 所示。

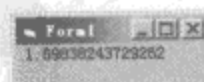


图 6.2 Exp 演示效果

```
Private Sub Form_Activate()
```

```

Dim MyAngle, MyHSin           '定义变量
MyAngle = 1.3                 '定义角度（单位为弧度）
MyHSin = (Exp(MyAngle) - Exp(-1 * MyAngle)) / 2 '计算双曲正弦函数值
Print MyHSin                  '输出
End Sub

```

### 6.1.3 Sgn 函数（返回符号）

Sgn 函数用于返回一个 Variant(Integer)类型的值，指出参数的正负号。

语法：


Sgn(number)

number: 必要的参数，number 是任何有效的数值表达式。

Sgn 函数的返回值如表 6.1 所示。

表 6.1 Sgn 函数的返回值

如果 number 为	Sgn 返回值
大于 0	1
等于 0	0
小于 0	-1

 说明：number 参数的符号决定了 Sgn 函数的返回值。

例如，使用 Sgn 函数来判断某数的正负号。执行效果如图 6.3 所示。

```

Private Sub Form_Activate()
    Dim MyVar1, MyVar2, MyVar3           '定义变量
    MyVar1 = 28: MyVar2 = -24: MyVar3 = 0 '给变量赋值
    Print Sgn(MyVar1)                    '返回值为 1
    Print Sgn(MyVar2)                    '返回值为-1
    Print Sgn(MyVar3)                    '返回值为 0
End Sub

```

### 6.1.4 Sqr 函数（平方根）

Sqr 函数用于返回一个 Double 类型值，指定参数的平方根。

语法：

Sqr(number)

number: 必要的参数，number 是一个 Double 类型的值或任何有效的大于或等于 0 的数值表达式。

例如，使用 Sqr 函数来计算某数的平方根。执行效果如图 6.4 所示。

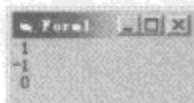


图 6.3 Sgn 执行效果



图 6.4 Sqr 执行效果

```
Private Sub Form_Activate()
    Dim MySqr
    MySqr = Sqr(4)
    Print MySqr
End Sub
```

'定义变量  
'返回值是 2  
'输入返回值

## 6.2 字符串函数

 教学录像：光盘\TM\lx\6\字符串函数.exe

在程序设计中，用于处理字符串的函数为字符串函数。本节主要介绍常用的字符串函数，包括 Len 函数、Left 函数和 Right 函数、Mid 函数和 Trim、RTrim、LTrim 函数（去空格）。

### 6.2.1 Len 函数

Len 函数用于返回一个 Long 类型的值，其中包含字符串内字符的数目，或是存储一个变量所需的字节数。

语法：

Len(string | varname)

string：任何有效的字符串表达式。如果 string 包含 Null，会返回 Null。

varname：任何有效的变量名称。如果 varname 包含 Null，会返回 Null。

如果 varname 是 Variant，Len 会视其为 String 并且总是返回其包含的字符数。

例如，使用 Len 函数可以得知某字符串的长度（字符数）或某变量的大小（位数）。执行效果如图 6.5 所示。



图 6.5 Len 执行效果

```
Private Sub Form_Activate()
    Print Len("MyString")
End Sub
```

'变量 Str 的值为 8

### 6.2.2 Left 函数和 Right 函数

#### 1. Left 函数

Left 函数用于返回一个 Variant (String) 类型的值，其中包含字符串中从左边算起指定数量的字符。

语法：

Left(string, length)

string：必要参数。字符串表达式中最左边的那些字符将被返回。如果 string 包含 Null，将返回 Null。

length：必要参数；为 Variant (Long)。数值表达式，指出将返回多少个字符。如果为 0，返回零长度字符串("")；如果大于或等于 string 的字符数，则返回整个字符串。

例如，使用 Left 函数来得到某字符串最左边的几个字符。执行效果如图 6.6 所示。

```
Private Sub Form_Activate()
    Print Left("changchun university", 2)
    Print Left("changchun university", 12)
End Sub
```

'返回值为"ch"  
'返回值为"changchun un"

## 2. Right 函数

Right 函数用于返回一个 Variant (String)类型的值, 其中包含从字符串右边取出的指定数量的字符。  
语法:

Right(string, length)

string: 必要参数。字符串表达式, 从中最右边的字符将被返回。如果 string 包含 Null, 将返回 Null。

length: 必要参数; 为 Variant (Long)。为数值表达式, 指出将返回多少字符。如果为 0, 返回零长度字符串(""); 如果大于或等于 string 的字符数, 则返回整个字符串。

例如, 使用 Right 函数来得到某字符串最右边的几个字符。执行效果如图 6.7 所示。

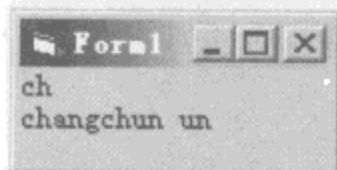


图 6.6 Left 执行效果

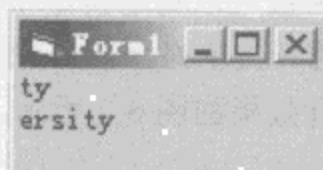


图 6.7 Right 执行效果

```
Private Sub Form_Activate()
    Print Right("changchun university", 2)
    Print Right("changchun university", 6)
End Sub
```

'输出"ty"  
'输出"ersity"

## 6.2.3 Mid 函数

Mid 函数用于返回一个 Variant (String)类型的值, 其中包含字符串中指定数量的字符。

语法:

Mid(string, start[, length])

string: 必要参数。字符串表达式, 从中返回字符。如果 string 包含 Null, 将返回 Null。

start: 必要参数; 为 Long。string 中被取出部分的字符位置。如果 start 超过 string 的字符数, Mid 返回零长度字符串("")。

length: 可选参数; 为 Variant (Long)。要返回的字符数。如果省略或 length 超过文本的字符数 (包括 start 处的字符), 将返回字符串中从 start 到尾端的所有字符。

例如, 使用 Mid 函数来得到某个字符串中的几个字符。执行效果如图 6.8 所示。

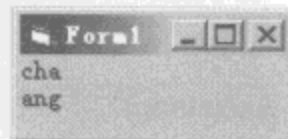


图 6.8 Mid 执行效果

```
Private Sub Form_Activate()
```



```
Print Mid("changchun university", 1, 3)
Print Mid("changchun university", 3, 3)
End Sub
```

'变量 MyStr 中的值为"cha"  
'变量 MyStr 中的值为"ang"

## 6.2.4 Trim、RTrim、LTrim 函数（去空格）

这几个函数用于返回 Variant (String)，其中包含指定字符串的拷贝，没有前导空白 (Ltrim)、尾随空白 (Rtrim) 或前导和尾随空白 (Trim)。

语法：

```
LTrim(string)
RTrim(string)
Trim(string)
```

string: 必要的参数，可以是任何有效的字符串表达式。如果 string 包含 Null，将返回 Null。

例如，使用 Trim 函数可以将字符串中开头和结尾的空格全部去除：利用 LTrim 函数将某字符串的开头空格全部去除；利用 RTrim 函数将某字符串的结尾的空格全部去除。执行效果如图 6.9 所示。

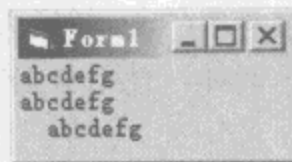


图 6.9 执行效果

```
Private Sub Form_Activate()
    Print Trim(" _abcdefg _")
    Print LTrim(" _abcdefg _")
    Print RTrim(" _abcdefg _")
End Sub
```

'输出值为"abcdefg"  
'输出值为"abcdefg \_"  
'输出值为" \_abcdefg"

说明：去空格函数也经常被应用到查询中，用于去除需要查询的关键字两端或一端的空格。

## 6.3 类型转换函数

教学录像：光盘\TM\lx\6\类型转换函数.exe

在编写程序过程中，经常需要数据类型之间进行转换，此时可使用类型转换函数。本节主要介绍 Asc 函数（转换为 ASCII）、Chr 函数（转换为字符）和 Val 函数（转换为数值型）等。

### 6.3.1 Asc 函数（转换为 ASCII）

Asc 函数用于返回一个 Integer 类型值，代表字符串中首字母的字符代码。

语法：

```
Asc(string)
```

string: 必要的参数，可以是任何有效的字符串表达式。如果 string 中没有包含任何字符，则会产生运行错误。

例如，本示例使用 Asc 函数返回字符串首字母的字符值的 ASCII 码值。执行效果如图 6.10 所示。

```
Private Sub Form_Activate()
    Print Asc("A")           '返回值为 65
    Print Asc("b")          '返回值为 98
    Print Asc("Apple")      '返回值为 65
End Sub
```

### 6.3.2 Chr 函数（转换为字符）

Chr 函数用于返回 String 类型值，其中包含与指定的字符代码相关的字符。

语法：

Chr(charcode)

charcode: 必要的参数。charcode 是一个用来识别某字符的 Long 型值。

例如，本示例使用 Chr 函数来返回一个具有一个字符的符号，该字符的编码与给定的数值相同。执行效果如图 6.11 所示。

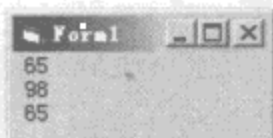


图 6.10 Asc 执行效果

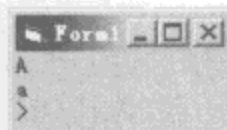


图 6.11 Chr 执行效果

```
Private Sub Form_Activate()
    Print Chr(65)           '返回值是 A
    Print Chr(97)           '返回值是 a
    Print Chr(62)           '返回值是 >
End Sub
```

### 6.3.3 Val 函数（转换为数值型）

返回包含于字符串内的数字，字符串中是一个适当类型的数值。

语法：

Val(string)

string: 必要的参数，可以是任何有效的字符串表达式。

例如，使用 Val 函数返回字符串中所含的数值。执行效果如图 6.12 所示。

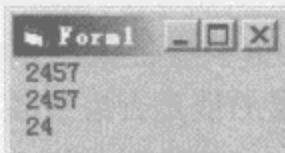


图 6.12 Val 执行效果

```
Private Sub Form_Activate()
    Print Val("2457")       '返回值是 2457
    Print Val("_2_45_7")    '返回值是 2457
    Print Val("24 and 57")  '返回值是 24
End Sub
```

End Sub

### 6.3.4 Str 函数（转换为字符型）

Str 函数用于返回一个 Variant (String) 类型的数值。

语法：

Str(number)

Number: 必要的参数, Long 类型值, 其中可包含任何有效的数值表达式。

例如, 使用 Str 函数将一个数字转成字符串。当数字转成字符串时, 字符串的首位一定是空格或是正负号。执行效果如图 6.13 所示。

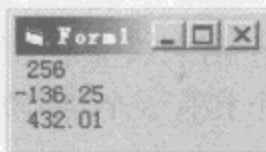


图 6.13 Str 执行效果

```
Private Sub Form_Activate()  
    Print Str(256)           '返回值是" 256"  
    Print Str(-136.25)      '返回值是"-136.25"  
    Print Str(432.01)       '返回值是" 432.01"  
End Sub
```

## 6.4 判断函数

教学录像: 光盘\TM\lx\6\判断函数.exe

在进行数据验证的过程中, 判断函数为编码提供了很大的方便。本节主要介绍 IsNull 函数、IsNumeric 函数和 IsArray 函数。

### 6.4.1 IsNull 函数

IsNull 函数用于返回一个 Boolean 类型值, 指出表达式是否不包含任何有效数据(Null)。

语法:

IsNull(expression)

expression: 必要的参数, 是一个 Variant 类型的值, 其中包含数值表达式或字符串表达式。

例如, 使用 IsNull 函数来检测某一变量的值是否为 Null。执行效果如图 6.14 所示。

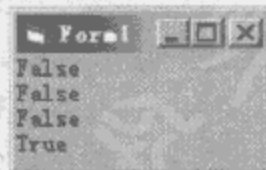


图 6.14 IsNull 执行效果

```

Private Sub Form_Activate()
    Dim Num                                '定义变量
    Print IsNull(Num)                      '输出值是 False
    Num = ""                              '给变量 Num 赋值
    Print IsNull(Num)                      '输出值是 False
    Num = "abcd"                          '给变量 Num 赋值
    Print IsNull(Num)                      '输出值是 False
    Num = Null                            '给变量 Num 赋值
    Print IsNull(Num)                      '输出值是 True
End Sub

```

### 6.4.2 IsNumeric 函数

IsNumeric 函数用于返回一个 Boolean 类型的值，指出表达式的运算结果是否为数值。

语法：

IsNumeric(expression)

expression: 必要的参数，是一个 Variant 类型的值，包含数值表达式或字符串表达式。

例如，使用 IsNumeric 函数检测某一变量或表达式是否为数值。执行效果如图 6.15 所示。

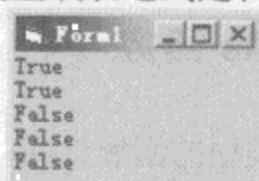


图 6.15 IsNumeric 执行效果

```

Private Sub Form_Activate()
    Print IsNumeric(62)                    '输出值是 True
    Print IsNumeric(62.5)                  '输出值是 True
    Print IsNumeric("changchun")          '输出值是 False
    Print IsNumeric(#4/1/2006#)           '输出值是 False
    Print IsNumeric(Null)                  '输出值是 False
End Sub

```

### 6.4.3 IsArray 函数

IsArray 函数用于返回一个 Boolean 类型的值，指出变量是否为一个数组。

语法：

IsArray(varname)

varname: 必要的参数，是一个指定变量的标识符。

例如，使用 IsArray 函数来检验某变量是否为数组。执行效果如图 6.16 所示。

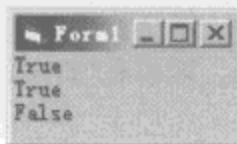


图 6.16 IsArray 执行效果

```

Private Sub Form_Activate()
    Dim aa(1 To 5) As Integer    '声明数组变量
    Dim bb                      '定义一个变体类型的变量
    Dim cc As String           '定义一个字符型变量
    bb = Array(1, 2, 3)        '使用数组函数
    Print IsArray(aa)           '输出值是 True
    Print IsArray(bb)           '输出值是 True
    Print IsArray(cc)           '输出值是 False
End Sub

```

## 6.5 日期和时间函数

 教学录像：光盘\TM\lx\6\日期和时间函数.exe

在涉及日期和时间的程序中，经常会用到各种格式的日期和时间，这可以通过日期和时间函数获得。本节主要介绍的日期和时间函数包括 Date 函数、Now 函数、Time 函数，Timer 函数，Weekday 函数，Year、Month、Day 函数，Hour、Minute、Second 函数。

### 6.5.1 Date 函数、Now 函数、Time 函数

Date 函数用于返回一个 Variant (Date) 类型的系统日期。

Now 函数用于返回一个 Variant (Date)，根据计算机系统设置的日期和时间来指定日期和时间。

Time 函数用于设置系统时间。

语法：

```

Date
Now
Time = time

```

time：必要的参数，可以是任何能够表示时刻的数值表达式、字符串表达式或它们的组合。

例如，使用 Date 函数返回系统当前的日期。使用 Now 函数返回系统当前的日期与时间。使用 Time 函数返回系统当前的时间。执行效果如图 6.17 所示。

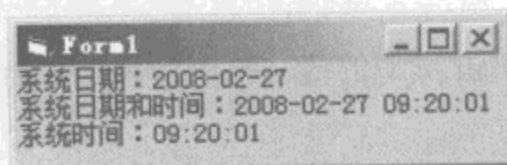


图 6.17 执行效果

```

Private Sub Form_Activate()
    Print "系统日期：" & Date    '输出系统日期
    Print "系统日期和时间：" & Now '系统日期和时间
    Print "系统时间：" & Time    '输出系统时间
End Sub

```



## 6.5.2 Timer 函数

Timer 函数用于返回一个 Single 类型的值，代表从午夜开始到现在经过的秒数。

语法：

Timer

例如，下面的代码实现的是从午夜开始到现在经过的所有秒数。

DateStr = Timer                      '变量 DateStr 的返回值为从午夜开始到现在经过的所有秒数

例 6.1 计时器。在编写应用软件的过程当中，如果需要返回从午夜开始到现在所有经过的秒数，也可以将该函数应用在计时器类的软件当中，如图 6.18 所示。（实例位置：光盘\TM\sl\6\1）

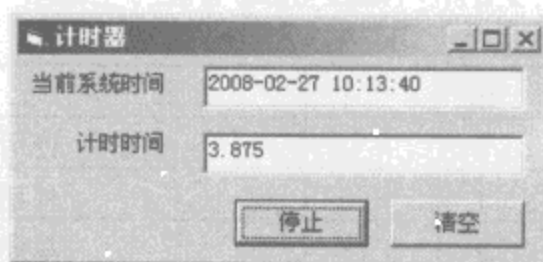


图 6.18 计时器

程序代码如下：

```
Option Explicit
Dim Time_Start
Private Sub Command1_Click()
    If Command1.Caption = "开始计时" Then
        Time_Start = Timer     '将当前的秒数值保存在变量 Time_Start 中
        Command1.Caption = "停止"
        Timer1.Enabled = True
    ElseIf Command1.Caption = "停止" Then
        Timer1.Enabled = False
        Command1.Caption = "开始计时"
    End If
End Sub
Private Sub Command2_Click()
    Text1.Text = ""
    Text2.Text = ""
End Sub
'控制计时显示的时钟控件中的代码如下:
Private Sub Timer1_Timer()
    Text2.Text = Timer - Time_Start     '计算显示的时间
End Sub
Private Sub Timer2_Timer()
    Text1.Text = Now
End Sub
```

### 6.5.3 Weekday 函数

Weekday 函数用于返回一个 Variant (Integer)类型的值, 包含一个整数, 代表某个日期是星期几。语法格式如下:

**Weekday(date, [firstdayofweek])**

**date:** 必要参数。能够表示日期的变体表达式、数值表达式、字符串表达式或它们的组合。如果 date 包含 Null, 则返回 Null。

**firstdayofweek:** 可选参数。指定一星期第一天的常数。如果未予指定, 则以 vbSunday 为默认值。firstdayofweek 参数的设置值如表 6.2 所示。

表 6.2 firstdayofweek 参数的设置

常 数	值	说 明
VbUseSystem	0	使用 NLS API 设置
VbSunday	1	星期日 (默认值)
VbMonday	2	星期一
VbTuesday	3	星期二
vbWednesday	4	星期三
vbThursday	5	星期四
VbFriday	6	星期五
VbSaturday	7	星期六

Weekday 函数的返回值如表 6.3 所示。

表 6.3 Weekday 函数的返回值

常 数	值	描 述
vbSunday	1	星期日
vbMonday	2	星期一
vbTuesday	3	星期二
vbWednesday	4	星期三
vbThursday	5	星期四
vbFriday	6	星期五
vbSaturday	7	星期六

**例 6.2** 判断星期几。利用 Weekday 函数和 Date 判断今天是星期几, 并将其输出, 如图 6.19 所示。  
(实例位置: 光盘\TM\sl\6\2)

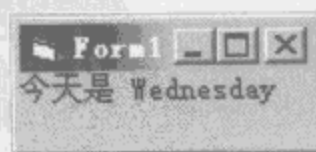


图 6.19 Weekday 执行效果

```

Private Sub Form_Activate()
    Dim day As String           '定义字符型变量
    Dim n As Integer           '定义整型变量
    n = Weekday(Date)          '利用 Weekday 函数判断星期几
    If n = 1 Then day = "Sunday" '如果值为 1, 给 day 赋值 Sunday
    If n = 2 Then day = "Monday" '如果值为 2, 给 day 赋值 Monday
    If n = 3 Then day = "Tuesday" '如果值为 3, 给 day 赋值 Tuesday
    If n = 4 Then day = "Wednesday" '如果值为 4, 给 day 赋值 Wednesday
    If n = 5 Then day = "Thursday" '如果值为 5, 给 day 赋值 Thursday
    If n = 6 Then day = "Friday" '如果值为 6, 给 day 赋值 Friday
    If n = 7 Then day = "Saturday" '如果值为 7, 给 day 赋值 Saturday
    Print "今天是 " & day       '输出今天的星期
End Sub

```

#### 6.5.4 Year、Month、Day 函数（年、月、日）

Year 函数返回一个 Variant (Integer)类型的值, 包含表示年份的整数。

Month 函数返回一个 Variant (Integer)类型的值, 其值为 1~12 之间的整数, 表示一年中的某月。

Day 函数返回一个 Variant (Integer)类型的值, 其值为 1~31 之间的整数, 表示一个月中的某一日。

语法:

```

Year(date)
Month(date)
Day(date)

```

date: 必要的参数, 可以是任何能够表示日期的变体表达式、数值表达式、字符串表达式或它们的组合。如果 date 包含 Null, 则返回 Null。


例如, 利用 Year 函数返回当前系统时间的年, 利用 Month 函数返回系统时间的月, 利用 Day 函数返回系统时间的日, 程序代码如下:

```

Private Sub Form_Activate()
    Print Year(Now) & "年" & Month(Now) & "月" & Day(Now) & "日" '输出当前的日期
End Sub

```

输出结果为“2008 年 2 月 27 日”的形式。

 说明: 输出“2008 年 2 月 27 日”的形式, 除了可以利用上面介绍的几个函数以外, 还可以利用 Format 函数将当前时间格式化为“××××年×月×日”的形式。

#### 6.5.5 Hour、Minute、Second 函数（时、分、秒）

Hour 函数返回一个 Variant (Integer)类型值, 其值为 0~23 之间的整数, 表示一天之中的某一钟点。

Minute 函数返回一个 Variant (Integer)类型值, 其值为 0~59 之间的整数, 表示一小时中的某分钟。

Second 函数返回一个 Variant (Integer)类型值, 其值为 0~59 之间的整数, 表示一分钟之中的某一秒。

语法:


```
Hour(time)
Minute(time)
Second(time)
```

time: 必要的参数, 可以是任何能够表示时刻的 Variant、数值表达式、字符串表达式或它们的组合。如果 time 包含 Null, 则返回 Null。

例如, 利用 Year 函数返回当前系统时间的年, 利用 Month 函数返回系统时间的月, 利用 Day 函数返回系统时间的日, 程序代码如下:

```
Private Sub Form_Activate()
    Print & Hour(Now) & "点" & Minute(Now) & "分" & Second(Now) & "秒" '输出当前的时间
End Sub
```

输出结果为“13 点 20 分 25 秒”的形式。

 说明: 输出“13 点 20 分 25 秒”的形式, 除了可以利用上面介绍的几个函数以外, 还可以利用 Format 函数将当前时间格式化为“××点××分××秒”的形式。

## 6.6 随机函数

 教学录像: 光盘\TM\lx\6\随机函数.exe

本节主要介绍随机函数, 包括 Randomize 函数和 Rnd 函数。

### 6.6.1 Randomize 函数

Randomize 函数是初始化随机数生成器。

语法:

```
Randomize [number]
```

number: 可选的参数, 是 Variant 类型的值或任何有效的数值表达式。

 注意: 若想得到重复的随机数序列, 要在使用具有数值参数的 Randomize 之前直接调用具有负参数值的 Rnd。使用具有同样数值的 Randomize 是不会得到重复的随机数序列的。

### 6.6.2 Rnd 函数

Rnd 函数用于返回一个 Single 类型的随机数值。

语法:

```
Rnd[(number)]
```

number: 可选的参数, number 是一个 Single 类型的值或任何有效的数值表达式。

Rnd 函数的返回值如表 6.4 所示。

表 6.4 Rnd 函数的返回值

如果 number 的值	Rnd 生成
小于 0	每次都使用 number 作为随机数种子得到的相同结果
大于 0	序列中的下一个随机数
等于 0	最近生成的数
默认	序列中的下一个随机数

例 6.3 下面的例子实现的是掷骰子的功能。其中应用了 Randomize 函数和 Rnd 函数。在程序运行时，单击“开始”按钮，左边的骰子就显示一个随机的数。实现界面如图 6.20 所示。程序代码如下：（实例位置：光盘\TM\sl\6\3）



图 6.20 掷骰子

```
Private Sub Command1_Click()  
    num = Int(Rnd * 6) + 1  
    Label1.Caption = num & "点"  
    Display (num)  
End Sub
```

'单击“开始”按钮  
'获取一个随机数赋给 num  
'显示该随机数  
'调用自定义函数（参见光盘）

```
Private Sub Form_Activate()  
    For i = 0 To 6  
        Shape2(i).FillColor = &H000000  
        Shape2(i).FillStyle = 0  
        Shape2(i).Shape = 3  
        Shape2(i).Visible = False  
    Next i  
    Shape1.FillColor = &HFFFFFF  
    Shape1.FillStyle = 0  
    Shape1.Shape = 5  
    Randomize Timer  
    Label1.Caption = "1 点"  
    Display (1)  
End Sub
```

'窗体加载  
'从 0~6 做循环，设置“点”的样式  
'设置填充颜色为黑色  
'设置填充样式为实心  
'设置形状为圆形  
'设置为不可见

'设置 Shape1 的填充颜色为白色  
'设置填充样式为实心  
'设置形状为圆角正方形  
'初始化  
'在标签中显示“1 点”  
'调用自定义函数，显示“1 点”效果

## 6.7 格式化函数

教学录像：光盘\TM\lx\6\格式化函数.exe

Format 函数用于返回 Variant(String)，其中含有一个表达式，它是根据格式表达式中的指令来格式化数据的。

语法：

```
Format(expression[, format[, firstdayofweek[, firstweekofyear]])
```

Format 函数的语法的参数说明如表 6.5 所示。



表 6.5 Format 函数的参数说明

参 数	说 明
Expression	必要参数。任何有效的表达式
Format	可选参数。有效的命名表达式或用户自定义格式表达式
firstdayofweek	可选参数。常数，表示一星期的第一天，其设置值如表 6.6 所示
firstweekofyear	可选参数。常数，表示一年的第一周，其设置值如表 6.7 所示

表 6.6 firstdayofweek 参数的设置

常 数	值	说 明
VbUseSystem	0	使用 NLS API 设置
VbSunday	1	星期日（默认值）
VbMonday	2	星期一
VbTuesday	3	星期二
vbWednesday	4	星期三
VbThursday	5	星期四
VbFriday	6	星期五
VbSaturday	7	星期六

表 6.7 firstweekofyear 参数的设置

常 数	值	说 明
VbUseSystem	0	使用 NLS API 设置
vbFirstJan1	1	从包含一月一日的那一周开始（默认值）
vbFirstFourDays	2	从本年第一周开始，而此周至少有四天在本年中
vbFirstFullWeek	3	从本年第一周开始，而此周完全在本年中

下面分别在日期时间、数组和字符串这三个方面介绍 Format 函数的使用。

#### （1）日期时间

在程序中显示日期时间时，经常需要将其格式化为某些特定的形式，这需要使用一些格式符，利用这些格式符可以格式化出需要的形式。在格式化日期时间时需要使用的格式符及其应用如表 6.8 所示。

表 6.8 日期和时间类型的例子

格 式 符	说 明	举 例	结 果
d	显示日期（1~31）	Format(Now, "d")	27
ddd	用英文缩写显示星期（Sun~Sat）	Format(Now, "ddd")	Wed
dddd	显示完整日期	Format(Now, "dddd")	2008-02-27
w	显示星期代号（1~7，1 是星期日）	Format(Now, "w")	4（星期三）
m	显示月份（1~12）	Format(Now, "m")	2
mmm	用英文缩写显示月份（Jan~Dec）	Format(Now, "mmm")	Feb
y	显示一年中第几天（1~366）	Format(Now, "y")	58

续表

格 式 符	说 明	举 例	结 果
yyyy	四位数显示年份 (0100~9999)	Format(Now, "yyyy")	2008
h	显示小时 (0~23)	Format(Now, "h")	16
m	放在 h 后显示分 (0~59)	Format(Now, "hm")	1616
s	显示秒 (0~59)	Format(Now, "s")	37
A/P 或 a/p	每日 12 时前显示 A 或 a, 12 时后显示 P 或 p	Format(Now, "A/P")	P
dd	显示日期 (01~31), 个位数用 0 补位	Format(Now, "dd")	27
dddd	用英文显示星期全名 (Sunday~Saturday)	Format(Now, "dddd")	Wednesday
dddddd	用汉字显示完整日期	Format(Now, "dddddd")	2008 年 2 月 27 日
ww	显示一年中第几个星期 (1~53)	Format(Now, "ww")	9
mm	显示月份 (01~12), 个位数用 0 补位	Format(Now, "mm")	02
mmmm	用英文月份全名 (January~December)	Format(Now, "mmmm")	February
yy	两位数显示年份 (00~99)	Format(Now, "yy")	08
q	显示季度数 (1~4)	Format(Now, "q")	1
hh	显示小时 (00~23), 个位数用 0 补位	Format(Now, "hh")	16
mm	放在 h 后显示分 (00~59), 个位数用 0 补位	Format(Now, "hhmm")	1620
ss	显示秒 (00~59), 个位数用 0 补位	Format(Now, "ss")	32
AM/PM 或 m/pm	每日 12 时前显示 AM 或 am, 12 时后显示 PM 或 pm	Format(Now, "AM/PM")	PM

## (2) 数值

同格式化日期时间一样, 在利用 Format 函数格式化数值类型数据时也需要使用到格式符, 具体的应用如表 6.9 所示。

表 6.9 数值类型的例子

格 式 符	说 明	举 例	结 果
0	实际数字小于符号位数, 数字前后加 0	Format(2, "00")	02
#	实际数字小于符号位数, 数字前后不加 0	Format(2, "##")	2
.	加小数点	Format(2, "00.00")	02.00
,	千分位	Format(1024, "0,000.00")	1,024.00
%	数值乘以 100, 在结尾加% (百分号)	Format(0.31415, "##.##%")	31.415%
\$	在数字前强加\$	Format(35.26, "\$##.##")	\$35.26
+	在数字前强加+	Format(-3.1415, "+##.####")	+3.1415
-	在数字前强加-	Format(3.1415, "-##.####")	-3.1415
E+	用指数表示	Format(34145, "0.0000e+00")	3.4145e+04
E-	与 E+相似	Format(34145, "0.0000e-00")	3.4145e04

## (3) 字符串

利用 Format 函数格式化字符串类型数据使用的格式符如表 6.10 所示。

表 6.10 字符串类型的例子

格 式 符	说 明	举 例	结 果
>	以大写显示	Format("tsoft", ">")	TSOFT
<	以小写显示	Format("TSOFT", "<")	tsoft
@	当字符位数小于符号位数时, 字符前加空格	Format("TSoft", "@@@@@@")	TSoft
&	当字符位数小于符号位数时, 字符前不加空格	Format("TSoft", "&&&&&&")	TSoft

## 6.8 小结

本章主要介绍了 VB 6.0 中的常用的内部函数, 每个函数都配有简明的实例, 读者可以通过实例更深入地理解函数的应用, 通过应用实践中的实例可以使读者达到融会贯通的目的。由于篇幅的限制不能将 VB 6.0 中所有的内置函数都进行详细地介绍。

## 6.9 练习与实践

1. 设计一个计算器程序。使其除了具有加、减、乘、除的运算功能以外还具有乘方和开方的功能。设计界面如图 6.21 所示。(答案位置: 光盘\TM\sl\6\4)
2. 设计一个程序。当在文本框中输入一个字符串的时候, 利用字符串函数将其中的空格去掉, 如输入 I Like VB, 将输出 IlikeVB。(答案位置: 光盘\TM\sl\6\5)
3. 设计一个随机抽奖的程序。主要利用 Randomize 函数和 Rnd 函数。其操作界面如图 6.22 所示。当用户单击“开始”按钮时, 数据开始滚动, 当用户单击“抽奖”按钮时, 在每个文本框中产生一个 0~10 的随机数, 同时显示在下面的文本框中。(答案位置: 光盘\TM\sl\6\6)

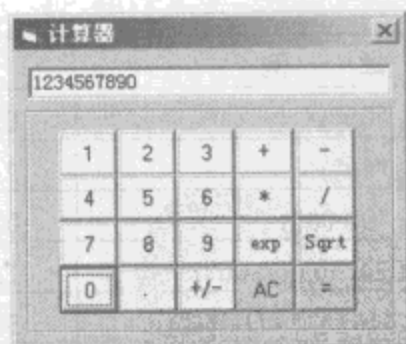


图 6.21 计算器



图 6.22 随机抽奖

4. 设计一个程序。实现判断输入的年份是否为闰年。(提示, 闰年应满足如下条件之一: ①可以被 400 整除; ②可以被 4 整除, 但是不能被 100 整除。)(答案位置: 光盘\TM\sl\6\7)

# 第2篇

## 核心技术

- » 第7章 窗体
- » 第8章 标准模块和类模块
- » 第9章 常用标准控件
- » 第10章 菜单、工具栏和状态栏
- » 第11章 对话框
- » 第12章 OOP 及系统对象
- » 第13章 常用 ActiveX 控件
- » 第14章 鼠标键盘处理
- » 第15章 程序调试和错误处理
- » 第16章 文件系统编程

本篇介绍了窗体，标准模块和类模块，常用标准控件，菜单、工具栏和状态栏，对话框，OOP 和系统对象，ActiveX 控件，鼠标键盘处理，程序调试和错误处理，文件系统编程等。学习完这一部分，将能够开发一些小型应用程序。







# 第 7 章

## 窗体

(教学录像: 1 小时)

用户界面是应用程序的重要组成部分,任何软件都有非常友好的人机界面,既方便用户使用,又可以将绝大部分的程序功能体现出来。一般用户界面都是将菜单、工具栏、状态栏等控件放置在窗体上,窗体作为一个容器来容纳这些控件,使其更好地发挥作用。本章将对窗体对象进行详细地介绍。

通过阅读本章,您可以:

- » 了解窗体的结构
- » 了解窗体的分类
- » 掌握窗体的添加和移除
- » 掌握窗体的加载和卸载
- » 掌握窗体的属性
- » 掌握窗体的方法
- » 掌握窗体的事件
- » 了解窗体事件的生命周期
- » 掌握 MDI 窗体的特点
- » 掌握 MDI 窗体的设计

## 7.1 窗体的概述

 教学录像：光盘\TM\lx\7\窗体的概述.exe

本节主要介绍窗体的结构、模式窗体和无模式窗体、SDI 窗体和 MDI 窗体、添加和移除窗体、加载 (Load) 与卸载 (Unload) 窗体。

### 7.1.1 窗体的结构

#### 1. 窗体的概念

窗体是应用程序的一个重要组成部分。在程序设计阶段，窗体是程序员的“工作台”，程序员在窗体上创建应用程序。在程序运行的时候，每一个窗体对应一个窗口。





窗体是用户和应用程序交互的接口。它由属性定义外观，由方法定义行为，由事件定义与用户的交互。它是 VB 中一个重要的对象，可以作为其他控件的“父对象”。也就是说，窗体除了具有自己的属性、方法外，还可以作为其他控件的容器，可以在它上面放置除窗体之外的其他控件，如：文本框、图片框、各种按钮等。当窗体显示时，在它上面的控件是可见的；当窗体移动时它们随之移动；当窗体隐藏时，在它上面的控件也跟着隐藏。

窗体文件的扩展名是 .frm，可以作为文件存储到磁盘中。

#### 2. 窗体的组成

一般的 VB 窗体都由标题栏、控制按钮和窗体区域组成，具体效果如图 7.1 所示。

##### (1) 标题栏

标题栏是指在窗体顶部的长条区域，包括（从左到右）：窗体图标（）、窗体标题 (Form1)、最小化按钮（）、最大化按钮（）、关闭按钮（）。

##### (2) 控制按钮

窗体的控制按钮在窗体标题栏的最右端，包括最大化、最小化和关闭按钮，其作用是对窗体进行控制。

##### (3) 窗体区域

窗体的主体部分。程序员可以在上面放置各种控件，操作用户可以通过该部分的控件与应用程序进行交互。

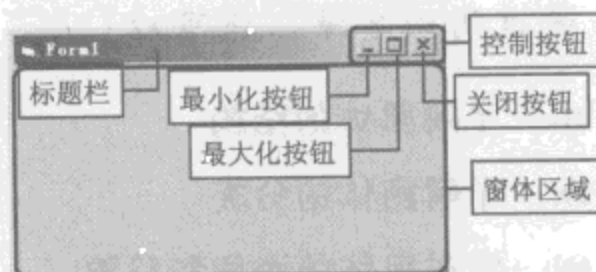


图 7.1 窗体的组成

### 7.1.2 模式窗体和无模式窗体

在窗体的分类中，可以根据窗体的显示状态分为模式窗体和无模式窗体两种类型，这两种类型的窗体在设计时基本相同，不同的是调用的代码和显示状态。

#### 1. 模式窗体

模式窗体是描述窗口的类型，在焦点可以切换到其他窗体之前要求用户采取动作。即，当新显示的窗体为模式窗体时，则该窗体为当前窗体，此时，其他窗体都不可选，只有将模式窗体关闭以后，

才可以操作其他窗体。

⚠ 注意：在显示模式窗体时，应用程序中的其他窗体失效，并不等于相应的应用程序失效。

在利用 Show 方法显示窗体时，当 Style 参数被设置为 1 或者 vbModal，这样显示的窗体即为模式窗体。

📖 说明：Show 方法的使用在显示窗体一节中将有详细的介绍。

## 2. 无模式窗体

无模式窗体是描述窗体类型，在焦点可以切换到其他窗体之前不要求用户采取动作。即，当新显示的窗体为无模式窗体时，用户单击任何一个窗体都可以将其设置为当前的窗体，并显示在屏幕的最前面。

在利用 Show 方法显示窗体时，当 Style 参数被设置为 0、vbModeless 或者省略，这样显示的窗体即为无模式窗体。

⚠ 注意：模式窗体指窗体完全占有控制权，只有关闭窗口之后才能使应用程序继续执行；而无模式窗体允许用户交流，并可以直接切换到应用程序的其他窗体，如果省略，则窗体以无模式显示。

### 7.1.3 SDI 窗体和 MDI 窗体

窗体根据其功能的不同可以分为 SDI 窗体（单文档窗体）和 MDI 窗体（多文档窗体），下面简单介绍一下单文档窗体和多文档窗体。

#### 1. SDI 窗体

SDI 窗体（Single Document Interface）是单文档窗体，指在应用程序中每次只能打开一个文档，想要打开另一个文档时，必须先关上已打开的文档。如，在 Windows 系统中经常使用的“记事本”工具。

SDI 窗体程序不能将一个窗体包含在另一个窗体中，所有的窗体都可以在屏幕上自由移动。在默认情况下创建的 VB 程序都是 SDI 窗体程序。

#### 2. MDI 窗体

MDI 窗体（Multiple Document Interface）是多文档窗体，在应用程序中可以同时打开多个文档。每个文档都有自己的窗口，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。当最小化父窗口时，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。例如：Microsoft Word 和 Microsoft Excel 应用程序就是 MDI 界面，VB 默认的开发环境也是 MDI 的形式。

#### 3. 设置 SDI 开发环境

这里介绍的 SDI 窗体和 MDI 窗体都是指利用 VB 的开发环境所创建的应用程序的窗体类型。对于 VB 这个环境，在默认情况下是 MDI 的形式，也可以通过环境设置，将其设置为 SDI 的形式，具体步骤如下：

(1) 选择“工具”/“选项”命令。

(2) 在弹出的“选项”对话框中，选择“高级”选项卡，在该选项卡中选中“SDI 开发环境”复选框，单击“确定”按钮，将弹出提示对话框，如图 7.2 所示。单击“确定”按钮，将开发环境设置为 SDI 的形式。

(3) 重新启动 VB 开发环境，此时的集成开发环境被设置为 SDI 的形式，如图 7.3 所示。此时开发环境中的各个组成部分（如工具箱、窗体、属性窗口等）都分开了，没有连接在一起。所有 IDE 窗口可在屏幕上任何地方自由移动；只要 VB 是当前应用程序，它们将位于其他应用程序之上。

如果不设置为 SDI 模式，即采用默认的 MDI 模式，所有 IDE 窗口包含在一个大小可调的父窗口内，并且各个 IDE 之间是可以连接的。

**注意：**在设置完开发环境以后，需要重新启动 VB 开发环境才生效，而当前没有重新启动的环境还保持着上次启动前设置的状态。

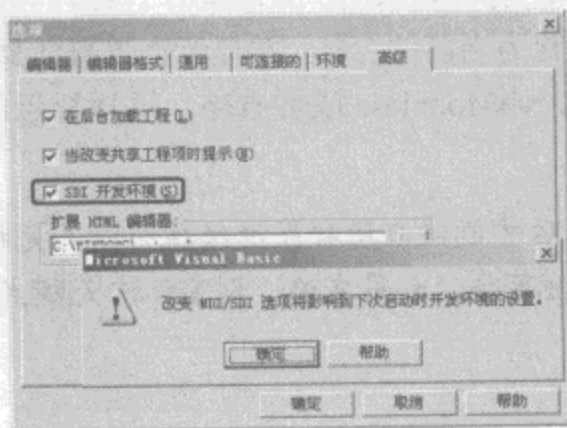


图 7.2 设置 SDI 开发环境

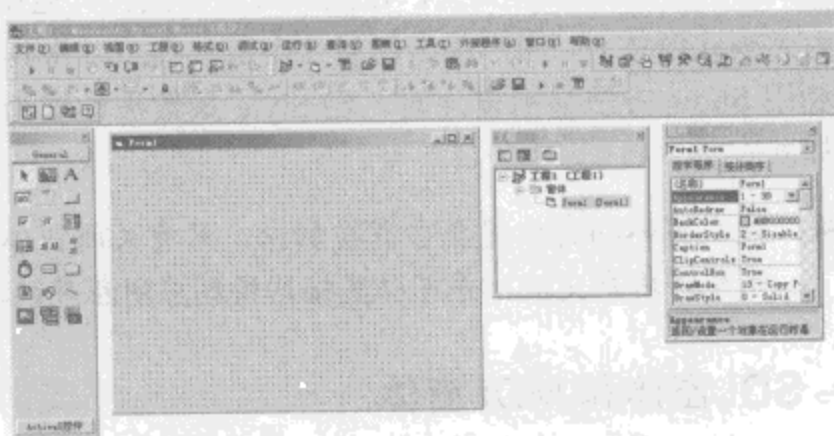


图 7.3 SDI 开发环境

**说明：**在使用中一般不采用 SDI 的开发环境，根据 Windows 用户的使用习惯，一般都采用默认的 MDI 的开发环境。

## 7.1.4 添加和移除窗体

### 1. 添加新窗体

在创建 VB 的工程时，默认会创建一个新窗体，如果工程中需要多个窗体，就需要再添加窗体，可以通过下面的步骤来添加窗体。

(1) 选择“工程”/“添加窗体”命令。

(2) 在弹出的“添加窗体”对话框中，选择“新建”选项卡。在该选项卡中选择“窗体”图标，单击“打开”按钮，或者双击“窗体”图标，即可创建一个新窗体。如图 7.4 所示。

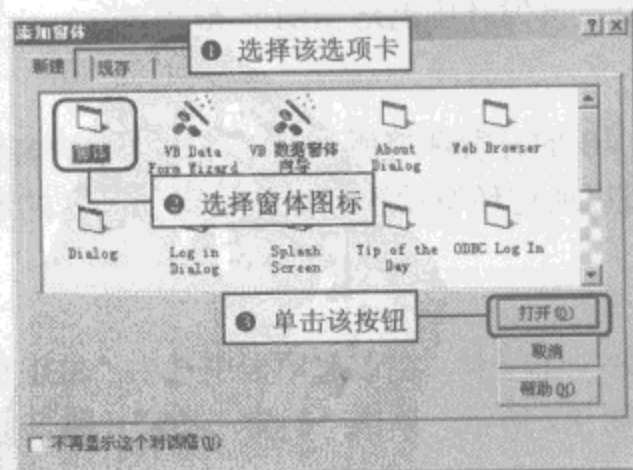


图 7.4 添加窗体

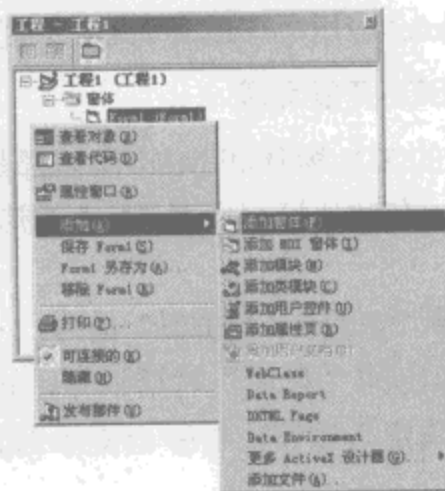



图 7.5 利用资源管理器添加窗体



 说明：也可以在工程资源管理器中单击鼠标右键，在弹出的快捷菜单中选择“添加”/“添加窗体”命令项，如图 7.5 所示，同样可以弹出如图 7.4 所示的对话框。

## 2. 添加现存窗体

当开发的程序数量积累到一定的程度，会发现在程序中有很多可以重复使用的窗体，如：将一个权限设置的窗体设置为通用的模块，每次使用时只需稍作改动就可以应用到新的程序中。对于这样的窗体将如何添加到工程中呢？下面就介绍一下如何添加现存的窗体。

选择“工程”/“添加窗体”命令，在弹出的“添加窗体”对话框中，选择“现存”选项卡，选择要添加的窗体，如这里的操作权限窗体，单击“打开”按钮，添加现存窗体，如图 7.6 所示。

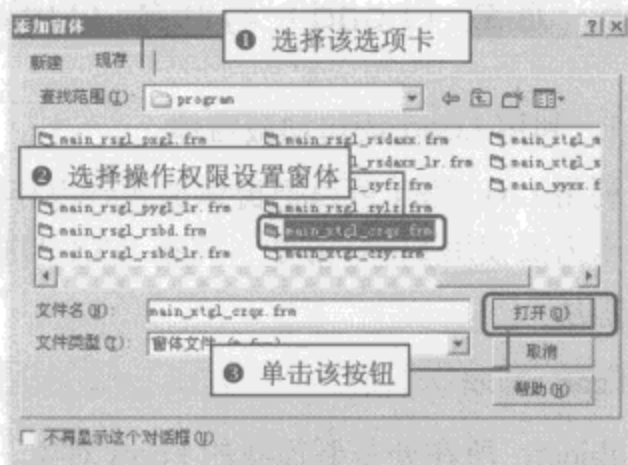



图 7.6 添加现存窗体

 说明：在使用此方法添加窗体时要注意添加的窗体名称和工程中已经存在的窗体名称不能相同，如果出现名称相同的情况，则弹出如图 7.7 所示的错误提示信息。

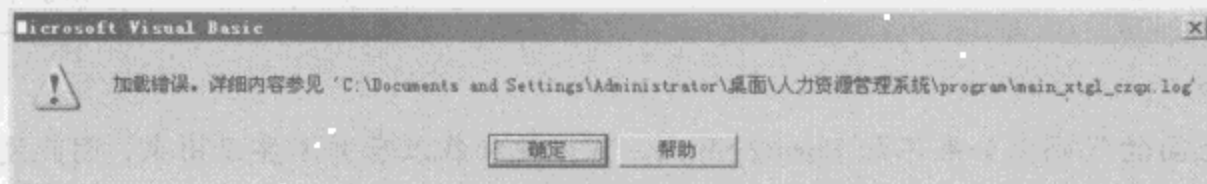


图 7.7 添加的窗体与已经存在的窗体同名

## 3. 移除窗体


当工程中的窗体不再需要的时候，就可以将其从工程中移除。具体的方法为：选中要删除的窗体，如 Form2，选择“工程”/“移除 Form2”命令，即可将该窗体移除。如图 7.8 所示。

 说明：这里的“移除 窗体名”命令，根据所选择的窗体的名称的不同而不同。



图 7.8 移除窗体



 **注意：**利用上面的方法只是将窗体从工程中移除，并没有将其从硬盘上删除。如果需要将其彻底删除，需要在工程存储的具体路径下将其删除。

### 7.1.5 加载（Load）与卸载（Unload）窗体

#### 1. 利用 Load 语句加载窗体

利用 Load 语句可以把窗体加载到内存中。这里仅仅是加载到内存中，并没有显示出来，如果想显示出来需要使用 Show 方法（将在 7.3.1 节中进行介绍）。


语法：

Load object

object: 所在处是要加载的 Form 对象、MDIForm 对象的名称。

**例 7.1** 加载窗体。下面的代码，在 Form1 中单击按钮，调用 Form2 窗体。（实例位置：光盘\TM\sl\7\1）

```
Private Sub Command1_Click()  
    Load Form2                                '加载 Form2 窗体  
    Form2.Show                                '显示 Form2 窗体  
End Sub
```

 **说明：**在上面的代码中如果不加 Form2.Show，窗体被加载以后并不显示出来，因此没有什么效果。这里添加 Form2.Show 语句，是用于突出程序效果。

#### 2. 利用 Unload 语句卸载窗体

在利用 Load 语句加载窗体以后，已经加载的窗体会占用一部分的内存，如果不将其卸载会使计算机的运行速度变慢，影响程序的执行。利用 Unload 语句可以将窗体从内存中卸载。


语法：

Unload object

object: 所在处是要卸载的 Form 对象的名称。

**例 7.2** 卸载窗体。当 Form2 窗体被加载以后，即可利用 Unload 语句将其卸载；如果需要卸载本窗体，则直接使用 Unload Me 即可。关键代码如下：（实例位置：光盘\TM\sl\7\2）

```
Private Sub Command2_Click()  
    Unload Form2                                '卸载 Form2 窗体  
End Sub  
Private Sub Command3_Click()  
    Unload Me                                    '卸载本窗体  
End Sub
```

 **注意：**如果利用 Unload 语句卸载的窗体是工程中最后一个被卸载的窗体，此卸载将结束程序的执行。

## 7.2 窗体的属性

 教学录像：光盘\TM\lx\7\窗体的属性.exe

本节主要介绍影响窗体外观的几个重要属性，包括名称（Name 属性）、标题（Caption 属性）、图标（Icon 属性）、背景（Picture 属性）和边框样式（BorderStyle 属性）等。

### 7.2.1 名称（Name 属性）

窗体的名称是工程中用于窗体的唯一标识，因此在一个工程中，不能有两个名称相同的窗体。在窗体创建时，默认会创建一个窗体名，一般形式为 Form\*，其中的\*为从 1 开始的自然数。

在使用时可以通过 Name 属性来设置窗体的名称，该属性返回在代码中用于标识窗体对象的名字，在运行时是只读的。

语法：

```
object.Name
```

object：所在处代表一个对象表达式。这里为活动窗体模块相联系的窗体。

Name 属性的设置只能通过属性窗口中进行设置，如在属性窗口中设置窗体的 Name 属性为 Frm\_Main，如图 7.9 所示。设置完成以后在资源管理器中的显示如图 7.10 所示。



图 7.9 通过属性窗口设置 Name 属性

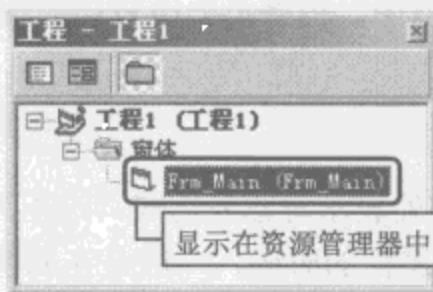


图 7.10 窗体在资源管理器中显示

### 7.2.2 标题（Caption 属性）

窗体的 Caption 属性用于显示在 Form 或 MDIForm 对象的标题栏中的文本。当窗体为最小化时，该文本被显示在窗体图标下面。

语法：

```
object.Caption [= string]
```

object：对象表达式。这里为窗体对象。

string：字符串表达式，其值是被显示为标题的文本。

设计时，可以在属性窗口中进行设置。在属性窗口中选中 Caption 属性，在后面输入要显示的窗体

标题, 如图 7.11 所示。设置完成的效果如图 7.12 所示。



图 7.11 通过属性窗口设置

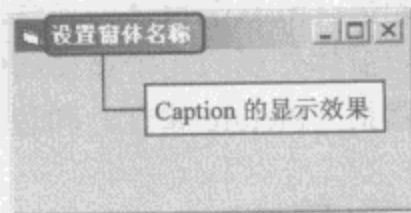


图 7.12 设置完成后的效果

在运行时, Caption 属性也可以通过程序代码设置。

例如, 在运行时, 设置窗体的标题为“设置窗体名称”, 其运行后的显示效果如图 7.12 所示。程序代码如下:

```
Private Sub Form_Load()  
    Me.Caption = "设置窗体名称" '设置窗体的 Caption 属性  
End Sub
```

### 7.2.3 图标 (Icon 属性)

窗体的 Icon 属性适用于设置窗体在运行时窗体处于最小化时显示的图标。一般情况下, 如果不对 Icon 属性进行设置, VB 会给窗体设置一个默认的图标, 在将工程生成 Exe 文件的时候, 将显示这个图标。在实际的开发中, 程序员会给自己的程序设置一个美观大方又具有实际意义的图标。

语法:

object.Icon

object: 所在处表示对象表达式, 这里为窗体对象。

在设置 Icon 属性的时候, 一般是在属性窗口中进行设置。具体的设置过程如下:

(1) 选择要设置图标的窗体。


(2) 在属性窗口中找到 Icon 属性, 单击该属性后的  按钮, 将弹出“加载图标”对话框。选择需要添加的图标, 单击“打开”按钮, 将选中的图标添加到窗体的标题栏中。如图 7.13 所示。



图 7.13 图标添加过程

设置 Icon 属性时,除了可以通过属性窗口设置以外,还可以通过程序代码进行设置。如设置上面的形式,可以通过下面的代码实现。

```
Private Sub Form_Load()  
    Me.Icon = LoadPicture(App.Path & "\85.ico")           '加载窗体图标  
End Sub
```

#### 7.2.4 背景 (Picture 属性)

VB 灰色的窗体背景并不美观,在设计应用程序的时候,为了窗体的美观可以给窗体设置一个符合程序主题的背景图片。这里可以通过在窗体上添加一个 Image 控件或者 PictureBox 控件,然后在控件中添加图片来实现,当然也可以通过设置窗体的 Picture 属性来实现。

Picture 属性用于返回或设置窗体中要显示的图片。

语法:

object.Picture [= picture]

☑ object: 对象表达式。

☑ picture: 字符串表达式,指定一个包含图片的文件,其设置值如表 7.1 所示。

表 7.1 picture 的设置值

设置值	描述
(None)	(默认值) 无图片
(Bitmap, icon, metafile, GIF, JPEG)	指定一个图片。设计时可以从属性窗口中加载图片。在运行时,也可以在位图、图标或元文件上使用 LoadPicture 函数来设置该属性

在使用 Picture 属性时,可以通过属性窗口来设计实现,也可以通过程序代码来实现。通过属性窗口的设计实现过程与设置 Icon 属性的过程是类似的。具体如下:

(1) 选择要添加图片的窗体。

(2) 在属性窗口中找到 Picture 属性,单击属性后面的...按钮,将弹出“加载图片”对话框。在该对话框中选择需要添加到窗体上的图片,单击“打开”按钮,将选中的图片添加到窗体上。其执行过程如图 7.14 所示。



图 7.14 Picture 属性的设置



Picture 属性除了可以通过属性窗口设置实现以外，还可以通过程序代码实现。下面的代码即可实现上面介绍的效果。

```
Private Sub Form_Load()  
    Me.Picture = LoadPicture(App.Path & "\VB 餐饮管理系统启动界面.jpg")  
End Sub
```

'给窗体添加图片

### 7.2.5 边框样式 (BorderStyle 属性)

不同的窗体有不同的用处，根据窗体不同的用处，可以将其设置成不同的样式。利用窗体的 BorderStyle 属性可以设置窗体的样式，该属性用于返回或设置对象的边框样式。当窗体对象在运行时，此设置是不可用的。

语法：

object.BorderStyle = [value]

object: 对象表达式，这里为窗体对象。

value: 值或常数，用于决定边框样式，其设置值如表 7.2 所示。

表 7.2 Value 参数的设置

常 数	设 置 值	描 述
vbBSNone	0	无（没有边框或与边框相关的元素）
vbFixedSingle	1	固定单边框。可以包含控制菜单框、标题栏、“最大化”按钮和“最小化”按钮。只有使用“最大化”和“最小化”按钮才能改变大小
vbSizable	2	（默认值）可调整的边框。可以使用设置值 1 列出的任何可选边框元素重新改变尺寸
vbFixedDouble	3	固定对话框。可以包含控制菜单框和标题栏，不能包含“最大化”和“最小化”按钮，不能改变尺寸
vbFixedToolWindow	4	固定工具窗口。不能改变尺寸。显示“关闭”按钮并用缩小的字体显示标题栏。窗体在 Windows 95 的任务条中不显示
vbSizableToolWindow	5	可变尺寸工具窗口。可变大小。显示“关闭”按钮并用缩小的字体显示标题栏。窗体在 Windows 95 的任务条中不显示

图 7.15 中列出了这些窗体的不同样式，读者在使用中可以根据不同的需要自己选择。例如，在设计启动窗体时，可以将 BorderStyle 属性设置为 0，即无边框的形式，同时利用 Picture 属性设置窗体的背景图片，这样显示比较美观；在设计类似对话框的窗体时，可以将 BorderStyle 属性设置为 3，此时窗体只包括控制菜单框和标题栏，不能包含“最大化”和“最小化”按钮，不能改变尺寸。

⚠ 注意：BorderStyle 属性只能在设计时，通过属性窗口设置，不能通过程序代码设计实现。

### 7.2.6 显示状态 (WindowState 属性)

在进行窗体显示的时候，根据程序的需要可以将窗体显示为全屏、最小化或者正常显示的模式。



利用窗体的 `WindowState` 属性可以设计窗体的显示状态。`WindowState` 属性用于返回或设置一个值, 该值用来指定在运行时窗体窗口的可视状态。



图 7.15 BorderStyle 属性设置

语法:

`object.WindowState [= value]`

`object`: 对象表达式。

`value`: 一个用来指定对象状态的整数, 其设置值如表 7.3 所示。

表 7.3 value 的设置值

常 数	值	描 述
<code>VbNormal</code>	0	(默认值) 正常
<code>vbMinimized</code>	1	最小化 (最小化为一个图标)
<code>vbMaximized</code>	2	最大化 (扩大到最大尺寸)

该属性可以在属性窗口中进行设置, 也可以通过程序代码进行设置。例如, 下面的代码用于设置窗体最大化显示。

```
Private Sub Form_Load()
    Me.WindowState = vbMaximized
End Sub
```

'设置窗体最大化显示

当窗体被设置为不同的效果时, 在状态栏中的效果也不相同, 如图 7.16 所示。

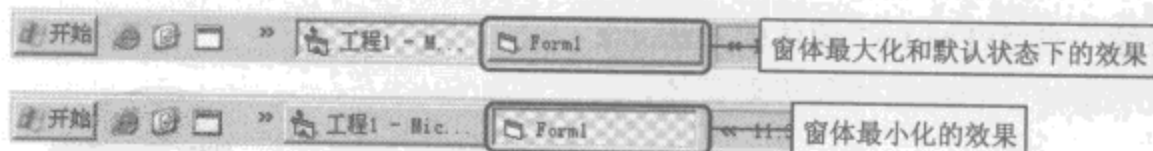


图 7.16 窗体不同状态的状态栏效果

### 7.2.7 显示位置 (`StartPosition` 属性)

利用窗体的 `StartPosition` 属性可以设置窗体初次显示的位置, 如, 显示在屏幕的中央, 或者屏幕的左上角等。该属性只能在设计时通过属性窗口进行设置, 在运行时不能使用。

StartPosition 属性返回或设置一个值，指定对象首次出现时的位置。

语法：

```
object.StartPosition = position
```

object: 对象表达式。

StartPosition: 整数，规定当对象显示时的位置。其设置值如表 7.4 所示。

表 7.4 StartUpPosition 参数的设置值

常 数	值	描 述
VbStartUpManual	0	没有指定初始设置值
VbStartUpOwner	1	UserForm 所属的项目中央
vbStartUpScreen	2	屏幕中央
vbStartUpWindowsDefault	3	屏幕的左上角

利用 StartUpPosition 属性可以将窗体设置为 4 种不同的显示效果，下面利用窗体布局窗口来说明这 4 种设置的不同显示效果，如图 7.17 所示。



图 7.17 窗体的显示位置

## 7.3 窗体的方法

教学录像：光盘\TM\lx\7\窗体的方法.exe

本节介绍显示窗体、隐藏窗体和移动窗体。

### 7.3.1 显示窗体 (Show 方法)

利用 Show 方法可以显示一个 MDIForm 或 Form 对象。不支持命名参数。

语法：

```
object.Show style, ownerform
```

object: 可选的参数。一个对象表达式。这里为窗体对象。

**style:** 可选的参数。一个整数，它用以决定窗体是模式还是无模式。如果 style 为 0，则窗体是无模式的；如果 style 为 1，则窗体是模式的。

**ownerform:** 可选的参数。字符串表达式，指出部件所属的窗体被显示。对于标准的 VB 窗体，使用关键字 me。

**例 7.3 窗体显示。**在前面已经介绍了模式窗体和无模式窗体，这里利用 Show 方法显示这两种形式的窗体。下面通过一个例子介绍 Show 方法的使用，并演示模式窗体和无模式窗体的区别。（实例位置：光盘\TM\sl\7\3）

在工程中添加 3 个窗体，在其中一个窗体上添加两个按钮，一个用于以无模式的形式调用窗体，另一个用于以有模式的形式调用窗体。当无模式调用窗体 Form2 以后，单击 Form1 窗体，Form1 窗体将获得焦点，成为当前的窗体，并可以再切换回 Form2 窗体。如图 7.18 所示。无模式调用的程序代码如下：

```
Private Sub Command1_Click()
    Form2.Show
End Sub
```

'调用无模式窗体  
'无模式调用 Form2 窗体

 **说明：**在 Show 的后面没有添加参数，这里默认为无模式显示。

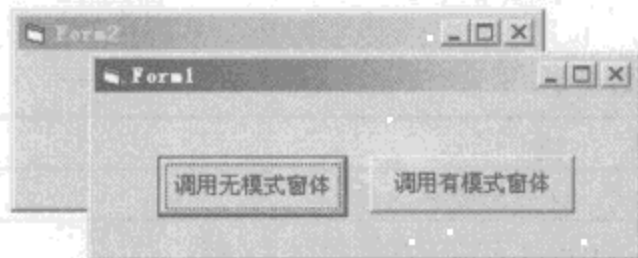


图 7.18 显示无模式窗体

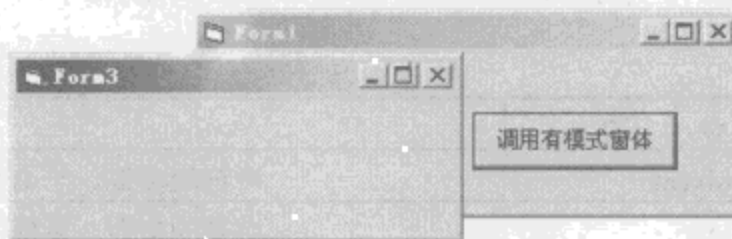


图 7.19 显示有模式窗体

单击按钮，调用有模式窗体 Form3，Form3 窗体获得焦点成为当前窗体，此时不能切换到其他窗体。除非 Form3 窗体被关闭，否则，只能对 Form3 窗体进行操作，如图 7.19 所示。调用有模式窗体的代码如下：

```
Private Sub Command2_Click()
    Form3.Show 1
End Sub
```

'调用有模式窗体  
'有模式调用 Form3 窗体


### 7.3.2 隐藏窗体（Hide 方法）

利用窗体的 Hide 方法可以将 MDIForm 或 Form 对象隐藏，但不能使其卸载。窗体被隐藏时，用户只有等到被隐藏窗体的事件过程的全部代码执行完后，才能够与该应用程序交互。如果调用 Hide 方法时窗体还没有加载，那么 Hide 方法将加载该窗体但不显示它。

**语法：**

```
object.Hide
```

**object:** 所在处代表一个对象表达式，这里为窗体对象。

 说明：隐藏窗体时，它就从屏幕上被删除，并将其 Visible 属性设置为 False。用户将无法访问隐藏窗体上的控件。如果调用 Hide 方法时窗体还没有加载，那么 Hide 方法加载该窗体但并不显示。

例 7.4 隐藏窗体。下面的代码用于隐藏 Form2 和隐藏自己。（实例位置：光盘\TM\sl\7\4）

```
Private Sub Command2_Click()  
    Form2.Hide                                '隐藏 Form2  
End Sub  
Private Sub Command3_Click()  
    Me.Hide                                  '隐藏自己  
End Sub
```

### 7.3.3 移动窗体（Move 方法）

利用窗体的 Move 方法可以移动 MDIForm、Form 窗体对象。此方法不支持命名参数。


语法：

```
object.Move left, top, width, height
```

Move 方法的语法参数说明如表 7.5 所示。

表 7.5 Move 方法的参数说明

参 数	描 述
object	可选的参数。一个对象表达式，这里为窗体对象
left	必需的参数。单精度值，指示 object 左边的水平坐标（x 轴）
top	可选的参数。单精度值，指示 object 顶边的垂直坐标（y 轴）
width	可选的参数。单精度值，指示 object 新的宽度
height	可选的参数。单精度值，指示 object 新的高度

 注意：只有 left 参数是必需的，但是，要指定任何其他的参数，必须先指定出现在语法中该参数前面的全部参数。例如，如果不先指定 left 和 top 参数，则无法指定 width 参数。任何没有指定的尾部的参数则保持不变。

下面利用“窗体布局”窗口，详细地说明 Move 方法中各个参数所表示的意义。具体的示意图如图 7.20 所示。



图 7.20 “窗体布局”窗口



## 7.4 窗体的事件

 教学录像：光盘\TM\lx\7\窗体的事件.exe

当在窗体进行各种操作时，便会引发窗体的相关事件。本节主要介绍常用窗体事件，包括单击和双击（Click / DblClick 事件），载入和卸载（Load / QueryUnload / Unload 事件）等。

### 7.4.1 单击和双击（Click / DblClick 事件）

#### 1. Click 事件

Click 事件是在窗体上按下然后释放一个鼠标按钮时发生。对一个 Form 对象来说，该事件是在单击一个空白区域或一个无效控件时发生。

语法：

```
Private Sub Form_Click()
```

例 7.5 Click 事件。下面的代码实现的是在程序运行的时候单击窗体弹出提示信息。（实例位置：光盘\TM\sl\7\5）

```
Private Sub Form_Click()                                '窗体单击事件
    MsgBox "您单击了窗体!", vbInformation, "信息提示"    '弹出提示对话框
End Sub
```

#### 2. DblClick 事件


DblClick 事件是当在窗体上按下和释放鼠标按钮并再次按下和释放鼠标按钮时发生。对于窗体而言，当双击被禁用的控件或窗体的空白区域时，DblClick 事件发生。

语法：

```
Private Sub Form_DblClick()
```

例 7.6 DblClick 事件。下面的代码实现的是在程序运行的时候双击窗体弹出提示信息。（实例位置：光盘\TM\sl\7\6）

```
Private Sub Form_DblClick()                            '窗体双击事件
    MsgBox "您双击了窗体", vbInformation, "信息提示"    '弹出提示对话框
End Sub
```

 注意：在 Click 事件中使用 MsgBox 将阻止 DblClick 事件的发生。因此，在程序开发中，应避免在同时存在 Click 事件和 DblClick 事件的 Click 事件中使用 MsgBox 函数。

### 7.4.2 载入和卸载（Load / QueryUnload / Unload 事件）

#### 1. Load 事件

Load 事件是在一个窗体被装载时发生。当使用 Load 语句启动应用程序，或引用未装载的窗体属



性或控件时，此事件发生。

语法：

```
Private Sub Form_Load()  
Private Sub MDIForm_Load()
```

例 7.7 Load 事件。在窗体加载时，向窗体上的 ComboBox 控件中添加项目，如图 7.21 所示。程序代码如下：（实例位置：光盘\TM\sl\7\7）

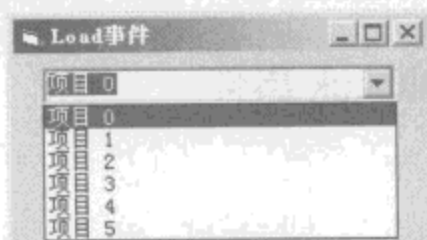


图 7.21 Load 事件

```
Private Sub Form_Load()  
    Dim i As Integer  
    For i = 0 To 5  
        Combo1.AddItem "项目 " & i  
    Next i  
    Combo1.ListIndex = 0  
End Sub
```

'窗体加载事件  
'定义整型变量  
'循环添加项目  
'添加项目  
'显示第一个项目

## 2. QueryUnload 事件

QueryUnload 事件在一个窗体关闭之前发生。当一个 MDIForm 对象关闭时，QueryUnload 事件先在 MDI 窗体中发生，然后在所有 MDI 子窗体中发生。如果没有窗体取消 QueryUnload 事件，该 Unload 事件首先发生在所有其他窗体中，然后再发生在 MDI 窗体中。当一个子窗体或一个 Form 对象关闭时，在那个窗体中的 QueryUnload 事件先于该窗体的 Unload 事件发生。

语法：

```
Private Sub Form_QueryUnload(cancel As Integer, unloadmode As Integer)  
Private Sub MDIForm_QueryUnload(cancel As Integer, unloadmode As Integer)
```

cancel: 一个整数。将此参数设定为除 0 以外的任何值，可在所有已装载的窗体中停止 QueryUnload 事件，并阻止该窗体和应用程序的关闭。

unloadmode: 一个值或一个常数，如表 7.6 所示，它指示引起 QueryUnload 事件的原因。

表 7.6 unloadmode 参数返回值

常 数	值	描 述
vbFormControlMenu	0	用户从窗体上的“控件”菜单中选择“关闭”指令
vbFormCode	1	Unload 语句被代码调用
vbAppWindows	2	当前 Microsoft Windows 操作环境会话结束
vbAppTaskManager	3	Microsoft Windows 任务管理器正在关闭应用程序
vbFormMDIForm	4	MDI 子窗体正在关闭，因为 MDI 窗体正在关闭
vbFormOwner	5	因为窗体的所有者正在关闭，所以窗体也在关闭

例 7.8 QueryUnload 事件。下面的例子演示的是，当用户单击窗体上的“退出”按钮时，将退出本程序。此时触发 QueryUnload 事件，弹出提示对话框，确认是否退出，如图 7.22 所示。如果用户单击“是”按钮，则退出程序；如果单击“否”按钮，则结束 QueryUnload 事件的执行，不退出程序。关键代码如下：（实例位置：光盘\TM\sl\7\8）

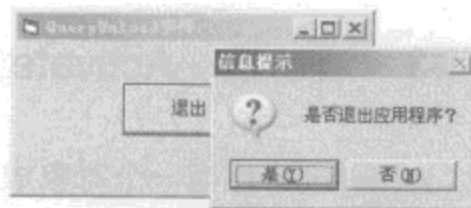


图 7.22 QueryUnload 事件

```
Private Sub Command1_Click()
    Unload Me                '卸载
End Sub
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    Dim Msg                  '声明变量
    If UnloadMode > 0 Then   '如果正在退出应用程序
        Msg = "是否退出应用程序?" '给变量赋值
    Else                    '如果正在关闭程序
        Msg = "是否关闭窗体?" '给变量赋值
    End If
    '如果用户单击“否”按钮，则停止 QueryUnload 事件
    If MsgBox(Msg, vbQuestion + vbYesNo, "信息提示") = vbNo Then Cancel = True
End Sub
```

### 3. Unload 事件

Unload 事件当窗体从屏幕上被删除时发生。当那个窗体被重新加载时，它的所有控件的内容均被重新初始化。当使用在 Control 菜单中的 Close 命令或 Unload 语句关闭该窗体时，此事件被触发。  
语法：

```
Private Sub object_Unload(cancel As Integer)
```

object: 一个对象表达式，这里为窗体对象。

cancel: 一个整数，用来确定窗体是否从屏幕删除。如果 cancel 为 0，则窗体被删除；将 cancel 设置为任何一个非零的值可防止窗体被删除。

例 7.9 Unload 事件。利用 Unload 事件提示用户是否关闭窗体，在程序运行时，当用户单击“关闭”按钮时，将关闭窗体，此时触发 Unload 事件，在该事件中弹出提示对话框提示用户是否关闭，如图 7.23 所示。如果用户单击“是”按钮，则关闭窗体；单击“否”按钮，则退出 Unload 事件，不关闭窗体。程序代码如下：（实例位置：光盘\TM\sl\7\9）

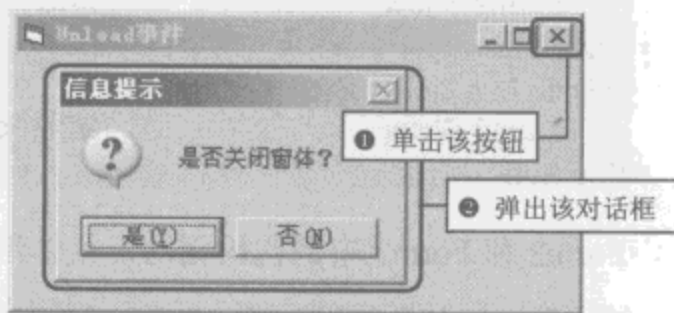


图 7.23 Unload 事件

```
Private Sub Form_Unload(Cancel As Integer)
    '如果用户单击 No 按钮，则停止 Unload 事件。
    If MsgBox("是否关闭窗体?", vbQuestion + vbYesNo, "信息提示") = vbNo Then
        Cancel = True
    End If
End Sub
```

### 7.4.3 活动性 (Activate / Deactivate 事件)

Activate 事件当一个对象成为活动窗口时发生。

Deactivate 事件当一个对象不再是活动窗口时发生。

语法:

```
Private Sub object_Activate()  
Private Sub object_Deactivate()
```

object: 所在处代表一个对象表达式, 这里为窗体对象。

**例 7.10** Activate / Deactivate 事件。利用 Activate 事件和 Deactivate 事件来判断窗体的活动性。当窗体被调用获得焦点, 处于活动状态, 触发 Activate 事件, 其窗体的标题为“当前活动窗体”; 当窗体失去焦点, 焦点被其他窗体占用, 将触发 Deactivate 事件, 窗体的标题被设置为“当前非活动窗体”。如图 7.24 所示。(实例位置: 光盘\TM\sl\7\10)

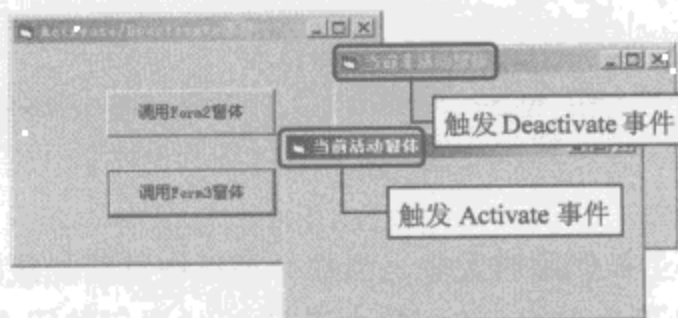


图 7.24 Activate / Deactivate 事件

**注意:** 演示本程序时, 注意要无模式调用窗体, 否则窗体不能失去焦点。

窗体 Form1 主要用于调用 Form2 窗体和 Form3 窗体, 其中的代码如下:

```
Private Sub Command1_Click()  
    Form2.Show          '调用 Form2  
End Sub  
Private Sub Command2_Click()  
    Form3.Show          '调用 Form3  
End Sub
```

Form2 和 Form3 中的代码相同, 具体形式如下:

```
Private Sub Form_Activate()      'Activate 事件  
    Me.Caption = "当前活动窗体"    '设置窗体标题  
End Sub  
Private Sub Form_Deactivate()   'Deactivate 事件  
    Me.Caption = "当前非活动窗体"  '设置窗体标题  
End Sub
```

### 7.4.4 初始化 (Initialize 事件)

Initialize 事件当应用程序创建 Form、MDIForm 时发生。

语法:

```
Private Sub object_Initialize()
```

object: 所在处代表对象表达式, 这里为窗体对象。

例 7.11 Initialize 事件。下面的代码可以在窗体初始化的时候, 设置窗体的标题为“Initialize 事件”。(实例位置: 光盘\TM\sl\7\11)

```
Private Sub Form_Initialize()  
    Me.Caption = "Initialize 事件"  
End Sub
```

'触发 Initialize 事件  
'设置窗体的标题

注意: 在使用 Initialize 事件时, 要特别注意 SetFocus 方法的使用。不能在 Initialize 事件中使用 SetFocus 方法, 如果使用将弹出“无效的过程调用或参数”, 如图 7.25 所示。这是因为在触发 Initialize 事件时, TextBox 控件还没有被加载到内存中, 因此不能对其进行焦点设置。将 Text1.SetFocus 语句写在 Load 事件中即可。

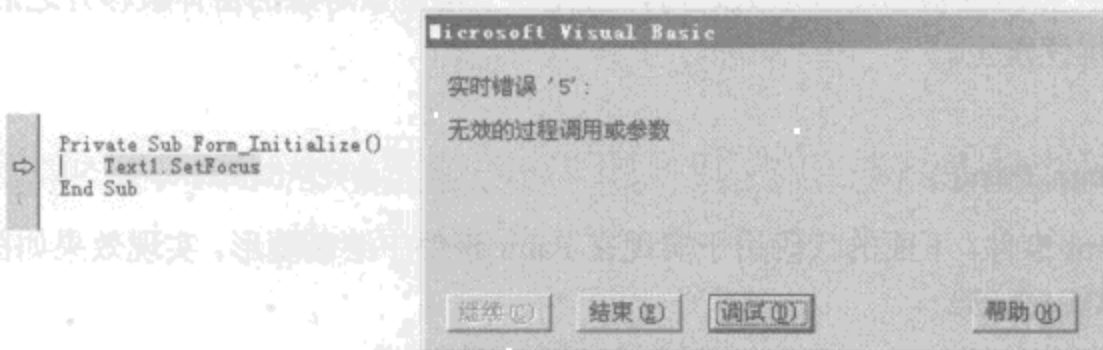


图 7.25 无效的过程调用参数

### 7.4.5 调整大小 (Resize 事件)

Resize 事件当一个窗体对象第一次显示或当该窗口对象状态改变时发生。例如, 一个窗体被最大化、最小化或被还原。

语法:

```
Private Sub object_Resize()
```

object: 一个对象表达式, 这里为窗体对象。

例 7.12 Resize 事件。本实例演示的是当窗体的大小被改变时, 即触发了 Resize 事件, 在该事件中调整窗体上的 TextBox 控件的大小和位置。如图 7.26 所示, 为窗体启动时的默认状态; 改变窗体的大小, 触发 Resize 事件, 窗体上的 TextBox 控件也随之改变, 如图 7.27 所示。(实例位置: 光盘\TM\sl\7\12)



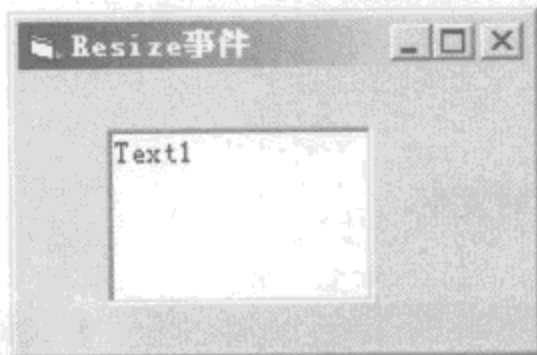


图 7.26 窗体启动的效果



图 7.27 触发 Resize 事件后的效果

程序代码如下：

```
Private Sub Form_Resize()  
    Text1.Top = (Me.Height - Text1.Height) / 2  
    Text1.Left = (Me.Width - Text1.Width) / 2  
    Text1.Width = Me.Width / 2: Text1.Height = Me.Height / 2  
End Sub
```

'设置 TextBox 控件的 Top 属性  
'设置 TextBox 控件的 Left 属性  
'设置控件的 Width 和 Height 属性

#### 7.4.6 重绘 (Paint 事件)

Paint 事件在一个对象被移动或放大之后，或在一个覆盖该对象的窗体被移开之后，该对象部分或全部暴露时，此事件发生。

语法：

```
Private Sub Form_Paint()
```

例 7.13 Paint 事件。下面的代码用于实现在 Paint 事件中绘制菱形。实现效果如图 7.28 所示。(实例位置：光盘\TM\sl\7\13)

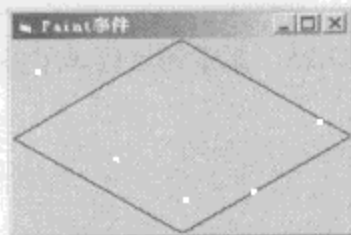


图 7.28 Paint 事件

```
Private Sub Form_Paint()  
    Dim X, Y  
    X = ScaleLeft + ScaleWidth / 2  
    Y = ScaleTop + ScaleHeight / 2  
    Line (ScaleLeft, Y)-(X, ScaleTop)  
    Line -(ScaleWidth + ScaleLeft, Y)  
    Line -(X, ScaleHeight + ScaleTop)  
    Line -(ScaleLeft, Y)  
End Sub
```

'触发 Paint 事件  
'定义变量  
'设置横坐标  
'设置纵坐标  
'画左上方直线  
'画右上方直线  
'画右下方直线  
'画左下方直线



### 7.4.7 焦点事件 (GotFocus / LostFocus 事件)

#### 1. GotFocus 事件

当对象获得焦点时产生该事件；获得焦点可以通过诸如〈Tab〉切换，或单击对象之类的用户动作，或在代码中用 SetFocus 方法改变焦点来实现。

语法：

```
Private Sub Form_GotFocus()
```

#### 2. LostFocus 事件

此事件是在一个对象失去焦点时发生，焦点的丢失或者是由于制表键移动或单击另一个对象操作的结果，或者是代码中使用 SetFocus 方法改变焦点的结果。

语法：

```
Private Sub Form_LostFocus()
```

**例 7.14** GotFocus/LostFocus 事件。下面的例子用于显示窗体的焦点事件。运行程序，单击“调用 Form2 窗体”，调用 Form2 窗体。Form2 窗体上没有控件，因此窗体获得焦点，触发 GotFocus 事件，输出“Form2 获得焦点”文字。单击 Form1 窗体，使 Form2 窗体失去焦点，触发 LostFocus 事件，输出“Form2 失去焦点”文字。如图 7.29 所示。程序代码如下：（实例位置：光盘\TM\sl\7\14）

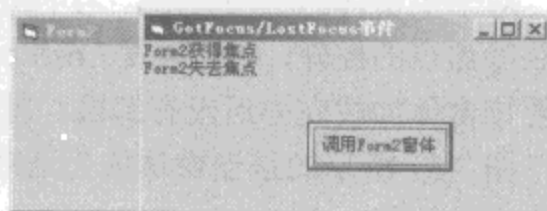


图 7.29 GotFocus/LostFocus 事件

```
Private Sub Form_GotFocus()      '获得焦点事件
    Form1.Print "Form2 获得焦点" '输出文字
End Sub
Private Sub Form_LostFocus()    '失去焦点事件
    Form1.Print "Form2 失去焦点" '输出文字
End Sub
```

## 7.5 窗体事件的生命周期

**教学录像：**光盘\TM\lx\7\窗体事件的生命周期.exe

窗体启动、运行和关闭过程会触发不同的事件，每个事件都有一定的生存期。下面就详细介绍每个过程中事件的生存期。

### 7.5.1 窗体启动过程

在程序运行时，如果窗体被调用，首先发生启动窗体的 Initialize 事件，紧接着是 Load 事件，将窗

体装入内存后,窗体被激活时,Activate 事件发生。这三个事件是在一瞬间发生的。如图 7.30 所示。

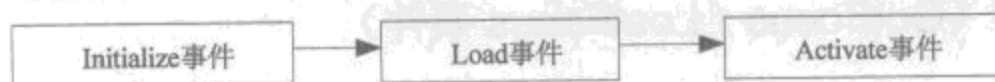


图 7.30 窗体启动过程的执行次序

窗体的 Initialize 事件和 Load 事件都是发生在窗体被显示前,所以经常在事件过程中放置一些命令语句来初始化应用程序,但所能使用的命令语句是有限的,如 SetFocus 一类的语句就不能使用。而 Print 语句仅当窗体的 AutoRedraw 属性值为真(True)时,在 Load 事件中的 Print 语句才有效。

VB 程序在执行时会自动装载启动窗体,在使用 Show 方法显示窗体时,如果窗体尚未载入内存,则首先将其载入内存,并引发窗体的 Load 事件。若想将窗体载入内存但不显示,可利用 Load 语句实现。

## 7.5.2 窗体运行过程

对于 GotFocus 事件,则有两种不同的情况:若窗体上没有可以获得焦点的控件,则窗体在 Activate 事件后立即触发 GotFocus 事件;当窗体上有可以获得焦点的控件时,则控件获得焦点,而不是窗体获得焦点。

对于多窗体的应用程序,当 Form1 由当前窗体变成非当前窗体时,若窗体是焦点或窗体上没有可以获得焦点的控件,则先触发 LostFocus 事件,后触发 Deactivate 事件。当该窗体再次成为活动窗体时,只要该窗体加载完毕后没有卸载,就不会触发 Load 事件,但是会触发 Activate 事件。

图 7.31 所示的是当窗体成为活动窗体以后,到失去焦点时事件的执行次序,且窗体上没有可以获得焦点的控件。

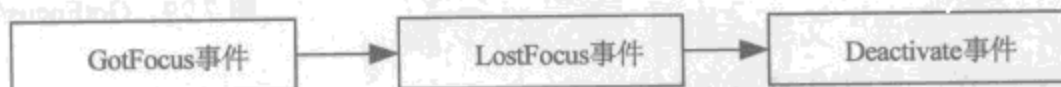


图 7.31 窗体运行过程中的事件执行次序

## 7.5.3 窗体关闭过程

在调用 Hide 方法时,仅仅是将窗体暂时隐藏,这不同于卸载。卸载是将窗体上的所有属性重新恢复为初始值;卸载还将引发窗体的卸载事件。如果卸载的窗体是工程的唯一窗体,将终止程序。

在 Windows 下,用户可通过使用菜单中的“关闭”按钮或单击窗体上的“关闭”按钮来关闭窗口,并结束程序的运行。当需要用程序来控制时可通过 End 语句来实现。执行该语句后将终止应用程序的执行,并从内存卸载所有窗体。

在窗体卸载时,首先触发 QueryUnload 事件,该事件发生在窗体卸载或关闭之前,即 Unload 事件之前。当 Unload 事件发生以后,将触发 Terminate 事件,当 Terminate 事件发生以后,窗体所有的调用或引用都将从内存中删除,即窗体的一个生命周期完成。卸载窗体的一般次序如图 7.32 所示。



图 7.32 窗体关闭时的事件执行次序

**例 7.15** 窗体事件的生命周期。本实例主要用于演示窗体事件的触发次序。在工程中添加两个窗体，主要用于演示 Form1 窗体的事件触发次序。具体执行的操作如下：  
(实例位置：光盘\TM\sl\7\15)

首先启动窗体，将依次触发 Initialize 事件、Load 事件、Activate 事件。窗体变成活动窗体，触发 Activate 事件，由于窗体上没有任何控件，则窗体获得焦点，触发 GotFocus 事件。

单击窗体调用 Form2 窗体，Form1 窗体失去焦点，依次触发 LostFocus 事件、Deactivate 事件。关闭 Form2，Form1 再次成为活动窗体，并获得焦点，依次触发 Activate 事件、GotFocus 事件。

关闭窗体依次触发的事件为 QueryUnload 事件、Unload 事件、Terminate 事件。

图 7.33 中显示了上面所述操作中窗体事件的执行次序。程序代码如下：



图 7.33 窗体事件的执行次序

Private Sub Form_Activate()	'窗体活动事件
Print Spc(3); "触发 Activate 事件"	'输出文字
End Sub	
Private Sub Form_Click()	'窗体单击事件
Form2.Show	'调用 Form2
End Sub	
Private Sub Form_Deactivate()	'窗体非活动事件
Print Spc(3); "触发 Deactivate 事件"	'输出文字
End Sub	
Private Sub Form_GotFocus()	'窗体焦点事件
Print Spc(3); "触发 GotFocus 事件"	'输出文字
End Sub	
Private Sub Form_Initialize()	'窗体初始化事件
MsgBox "触发 Initialize 事件", vbInformation, "信息提示"	'提示对话框
End Sub	
Private Sub Form_Load()	'窗体加载事件
Print Spc(3); "触发 Load 事件"	'输出文字
End Sub	
Private Sub Form_LostFocus()	'窗体失去焦点事件
Print Spc(3); "触发 LostFocus 事件"	'输出文字
End Sub	
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)	'窗体询问关闭事件
MsgBox "触发 QueryUnload 事件", vbInformation, "信息提示"	'提示对话框
End Sub	
Private Sub Form_Terminate()	'窗体销毁事件
MsgBox "触发 Terminate 事件", vbInformation, "信息提示"	'提示对话框
End Sub	
Private Sub Form_Unload(Cancel As Integer)	'窗体卸载事件

```
MsgBox "触发 Unload 事件", vbInformation, "信息提示"
End Sub
```

提示对话框

## 7.6 MDI 窗体

 教学录像：光盘\TM\lx\7\MDI 窗体.exe

MDI 窗体是类似于 Word 应用程序的多文档窗体，在同时操作多个窗口时，MDI 窗体非常重要。本节介绍什么是 MDI 窗体、MDI 窗体的添加和移除、MDI 子窗体和 MDI 主窗体的设计等。

### 7.6.1 MDI 窗体概述

#### 1. 多文档界面

MDI 窗体即多文档界面，Multiple-Document Interface。在 MDI 模式窗体中，一个应用程序在运行时，除了一个主窗口外，还包含一系列的子窗口。MDI 应用程序由一个父窗体和若干个子窗体组成，可以同时显示多个窗体，每个窗体都在自己的窗口中显示。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的空间。如常用的 Word、Excel 等都是 MDI 的应用程序。在 MDI 主窗体中，所有的子窗体都显示在工作区中，在工作区可以打开多个窗体，并且窗体之间的切换比较灵活。

#### 2. 主窗体和子窗体

MDI 应用程序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。文档或子窗体被包含在主窗体中，当主窗体最小化时所有的文档窗口也被最小化，只有主窗体的图标显示在任务栏中。

子窗体就是将 MDIChild 属性设置为 True 的普通窗体。一个应用程序可以包含许多相似或者不同样式的 MDI 子窗体。

#### 3. SDI 和 MDI

SDI 窗体和 MDI 窗体都有其各自的优点，在实际的开发中要根据应用程序的开发目的来确定选择哪种设计方式。比如在工资管理系统中，用户在使用中经常会出现多个窗体中的多个数据表进行对账的情况，这时采用 MDI 窗体的形式就比较合适。它可以非常方便地实现工资管理中各个数据表间的对账。如图 7.34 所示，可以将员工发放工资数和员工工资汇总部分进行比较对账。



图 7.34 工资管理系统



## 7.6.2 MDI 窗体的添加和移除

### 1. MDI 窗体的添加

MDI 窗体的添加和普通窗体的添加操作非常相似，具体的步骤如下：

(1) 在工程资源管理器中单击鼠标右键，在弹出的快捷菜单中选择“添加”/“添加 MDI 窗体”命令。

(2) 在弹出的“添加 MDI 窗体”对话框中，选择“新建”选项卡，选择“MDI 窗体”图标，单击“打开”按钮，即可向工程中添加一个 MDI 窗体。其执行过程如图 7.35 所示。

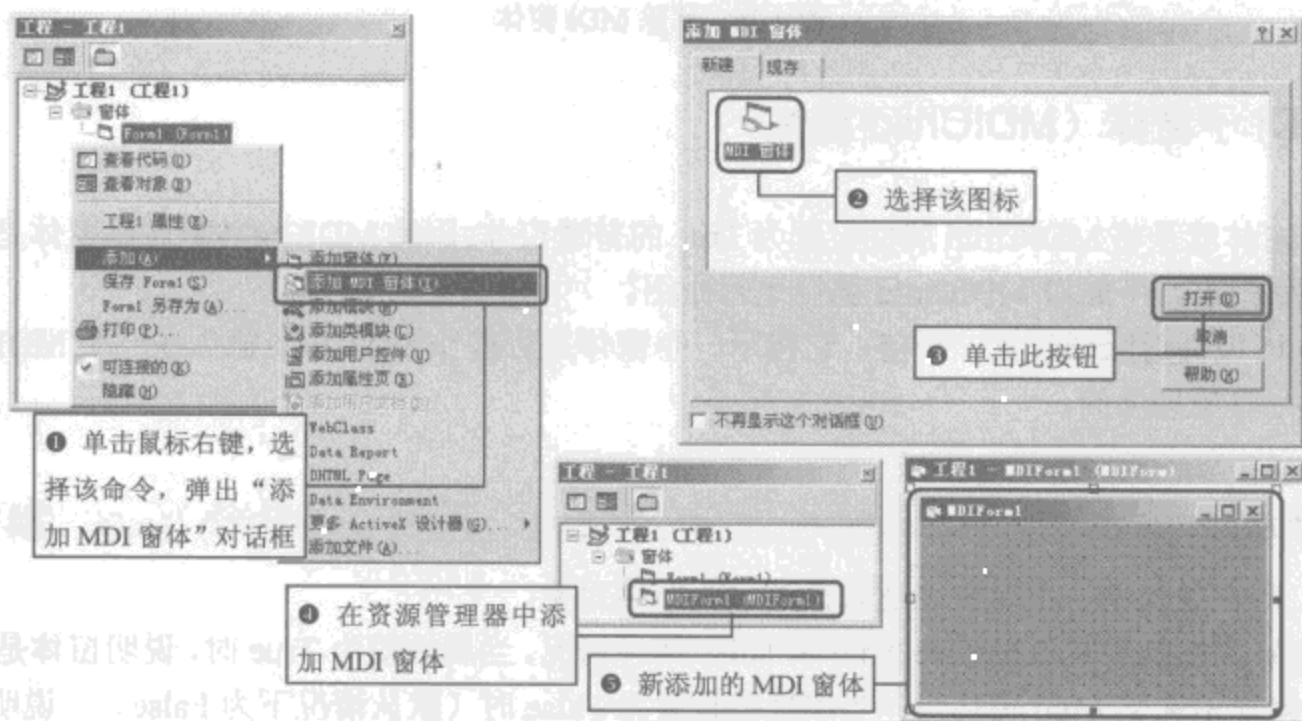


图 7.35 添加 MDI 窗体

**说明：**选择菜单中的“工程”/“添加 MDI 窗体”命令，同样也可以弹出“添加 MDI 窗体”对话框。

### 2. MDI 窗体的移除

MDI 窗体的移除和普通窗体的移除是一样的，在工程资源管理器中选中要删除的 MDI 窗体，然后选择“工程”/“移除 MDIForm1”命令，即可将 MDI 窗体从工程中移除，如图 7.36 所示。这里的命令会随着窗体名称的不同而不同。

**说明：**也可以通过右键菜单的形式移除 MDI 窗体。在工程资源管理器中选中该 MDI 窗体，单击鼠标右键，在弹出的快捷菜单中选择“移除 MDIForm1”命令，即可移除该 MDI 窗体。

**注意：**利用上面介绍的方法仅仅是将窗体从工程中移除，并没有将该窗体删除，如果需要将该窗体彻底删除，需要打开窗体的存储路径然后将该窗体删除。





图 7.36 移除 MDI 窗体

### 7.6.3 MDI 子窗体 (MDIChild 属性)

MDI 子窗体就是将 MDIChild 属性设置为 True 的普通窗体。因此 MDIChild 属性是窗体是否为 MDI 子窗体的重要标志。下面对 MDIChild 属性进行介绍。

MDIChild 属性返回或设置一个值，它指示一个窗体是否被作为 MDI 子窗体在一个 MDI 窗体内部显示，在运行时是只读的。

语法：

object.MDIChild

object: 所在处代表一个对象表达式，这里为窗体对象。

MDIChild 属性的设置值可以为 True 和 False 两种情况。当属性值为 True 时，说明窗体是一个 MDI 子窗体并且被显示在父 MDI 窗体内；当属性值设置为 False 时（默认情况下为 False），说明窗体不是一个 MDI 子窗体。

MDIChild 属性只能通过属性窗口进行设置，具体的设置方法如下：

- (1) 在资源管理器中，选择需要设置为 MDI 子窗体的窗体，如 Form1。
- (2) 在属性窗口中选择 MDIChild 属性，将其设置为 True。
- (3) 此时，在资源管理器中该窗体的图标被设置为 MDI 子窗体的效果如图 7.37 所示。

**注意：**在设置 MDI 子窗体的工程中一定要包括 MDI 主窗体。如果在包含 MDI 子窗体的工程中没有 MDI 主窗体，当该子窗体被调用时将弹出如图 7.38 所示的错误。



图 7.37 MDIChild 属性设置

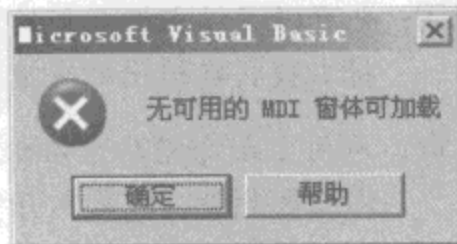



图 7.38 没有 MDI 主窗体

如果 MDI 子窗体在其父窗体装入之前被引用, 则其父 MDI 窗体将被自动装入。然而, 如果父 MDI 窗体在 MDI 子窗体装入前被引用, 则子窗体并不被装入。

 **说明:** 当建立一个多文档接口 (MDI) 应用程序时要使用该属性。在运行时, 该属性被设置为 True 的窗体被显示在 MDI 窗体内。一个 MDI 子窗体能够被最大化、最小化和移动, 都在父 MDI 窗体内部进行。

#### 7.6.4 MDI 程序的特点

MDI 主窗体除了不能添加没有 Align 属性的控件以外, 还具有以下特点:

- (1) 一个应用程序最多只能有一个 MDI 窗体。
- (2) MDI 子窗体不能是模式的。
- (3) 所有 MDI 子窗体都有可调整大小的边框、控制菜单框以及“最小化”和“最大化”按钮, 而不管 BorderStyle、ControlBox、MinButton 和 MaxButton 属性的设置值如何。
- (4) 所有的子窗体都显示在 MDI 窗体工作区内。用户可移动、改变子窗体的大小, 但对子窗体的所有操作都被限制在 MDI 窗体工作区之内。
- (5) MDI 窗体和子窗体可拥有各自的菜单栏、工具栏和状态栏。如果子窗体有自己的菜单栏, 则子窗体被显示时, MDI 窗体的菜单栏将被子窗体的菜单栏取代。

MDI 窗体和子窗体也可拥有各自的标题栏。当子窗体被最大化时, 它的标题显示在 MDI 窗体的标题栏中, 它的“最小化”、“最大化”、“关闭”按钮则显示在 MDI 窗体菜单栏的右端。如图 7.39、图 7.40 所示。



图 7.39 不显示子窗体的效果

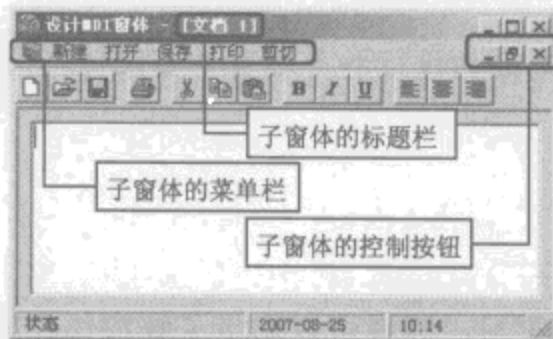


图 7.40 显示子窗体的效果

- (6) 当子窗体被最小化时, 它的图标显示在 MDI 窗体底部, 而不是显示在任务栏中。当 MDI 窗体被最小化时, 所有的子窗体也被最小化, 任务栏上只显示 MDI 窗体的图标。如图 7.41 所示。

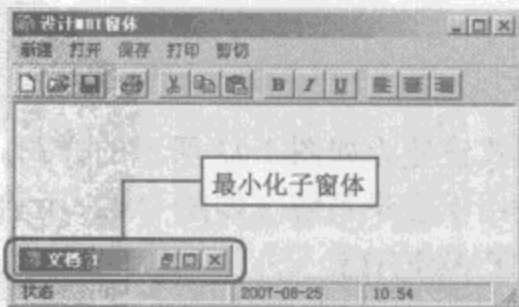



图 7.41 最小化子窗体

### 7.6.5 MDI 主窗体的设计

在设计 MDI 窗体时，和设计普通的窗体有一些不同：在 MDI 窗体上不能添加没有 Align 属性的控件。例如，文本框、列表框、标签等都不能放在 MDI 窗体上，而 ImageList、ToolBar、StatusBar、Timer、Data 等控件则可以。

**例 7.16** 设计 MDI 主窗体。下面介绍如何设计出如图 7.34 所示的 MDI 窗体。具体的方法如下：

(1) 首先创建一个 VB 工程，选择“工程”/“添加 MDI 窗体”菜单，添加 MDI 窗体，将弹出“添加 MDI 窗体”对话框，单击“打开”按钮，将添加一个 MDI 窗体。（实例位置：光盘\TM\sl\7\16）

 **注意：**一个工程中只能有一个 MDI 窗体，所以，当已经添加一个 MDI 窗体时，“添加 MDI 窗体”菜单项将变为灰度的。

(2) 添加 TreeView 控件。在图 7.34 的右侧是一个 TreeView 控件，TreeView 控件不能被直接放置到 MDI 窗体上，它需要通过一个辅助的控件来实现，即 PictureBox 控件。PictureBox 控件具有 Align 属性，而且该控件是一个具有容器性质的控件，可以容纳其他控件。在使用时，首先将 PictureBox 控件添加到窗体上，通过设置其 Align 属性设置其位置，并设置 PictureBox 无边框，然后将 TreeView 控件添加到 PictureBox 控件上，即可实现将 TreeView 控件放置在 MDI 窗体上。该方法对于其他的控件同样适用。

(3) 添加子窗体。添加完 MDI 窗体以后将添加 MDI 子窗体，MDI 子窗体的添加和普通子窗体的添加的方法是一样的，可以选择“工程”/“添加窗体”，弹出“添加窗体”对话框，单击“打开”按钮，将窗体添加到工程中。

此时添加的窗体只是普通的窗体，要想让它成为 MDI 子窗体，需要设置其 MDIChild 属性为 True，这样该窗体才能显示在 MDI 窗体的内部。

 **说明：**其他部分的设计和普通窗体的设计相同，这里就不再赘述，关键代码可参见本书光盘。

## 7.7 小结

通过本章的学习，读者可以初步了解窗体在应用程序中的构建和使用，包括窗体的添加、移除、属性、方法、事件等。本章还介绍了 MDI 窗体的相关知识，并结合实例使读者可以更加全面地了解窗体的相关知识。

## 7.8 练习与实践

1. 设计一个窗体，将窗体的样式设置为如图 7.42 所示的样式。（答案位置：光盘\TM\sl\7\17）
2. 设计两个窗体，一个窗体是用于登录系统（如图 7.42 所示），另一个窗体为系统主窗体（如图 7.43 所示）。（答案位置：光盘\TM\sl\7\18）

**要求：**程序启动时，显示登录窗体，系统主窗体不可见；当输入用户名和密码，单击“登录”按钮后，显示系统主窗体，关闭登录窗体。在主窗体中单击“关闭系统”按钮，退出程序。

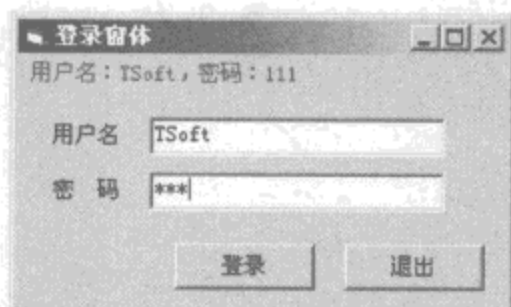


图 7.42 登录窗体

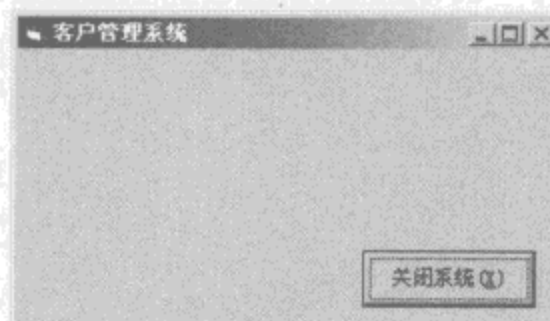


图 7.43 系统主窗体

3. 在上面的例子中, 当用户单击“关闭系统”按钮时, 提示对话框询问是否退出系统。(答案位置: 光盘\TM\sl\7\19)





图 19-2 船体主剖面图



图 19-3 船体主剖面图

图 19-2 船体主剖面图

图 19-3 船体主剖面图





# 第 8 章

## 标准模块和类模块

(教学录像: 20 分钟)



VB 中的代码都存储在模块中，模块有 3 种类型：窗体模块、标准模块和类模块。窗体模块可以看作是具有界面的类模块，在第 7 章中已经介绍过了。本章将对标准模块和类模块进行介绍。

通过阅读本章，您可以：

- » 了解什么是标准模块
- » 掌握标准模块的添加
- » 了解什么是类模块
- » 掌握类模块的添加
- » 了解什么是类生成器
- » 掌握类生成器的添加
- » 掌握如何使用类生成器添加属性、方法和事件
- » 了解标准模块和类模块的区别



## 8.1 标准模块

 教学录像：光盘\TM\lx\8\标准模块.exe

标准模块用于放置工程中共用的变量、常量、数据类型、函数过程和子过程等。下面介绍什么是标准模块及在工程中如何添加标准模块。

### 8.1.1 标准模块概述

标准模块的扩展名是.bas，在应用程序中标准模块是应用程序内供其他模块访问的公共过程和声明的容器。它可以包括变量、常量、类型、外部过程和全局过程的声明。在标准模块中的代码，不仅仅可以应用于一个工程，还可以应用到其他的工程中。

在工程资源管理器中，一般情况下会存在以下几种资源：窗体对象（在第7章已经介绍过）、标准模块（Module）、类模块（Class），如图8.1所示。在本节中将对标准模块进行简单介绍。

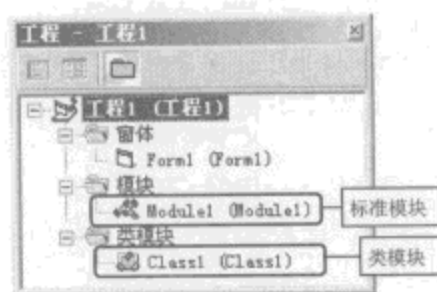


图 8.1 工程资源管理器

### 8.1.2 添加标准模块

#### 1. 添加新标准模块

选择“工程”/“添加模块”命令，即可弹出如图8.2所示的“添加模块”对话框，选择“新建”选项卡，选择“模块”图标，单击“开始”按钮，即可添加一个标准模块到工程中。如图8.3所示。

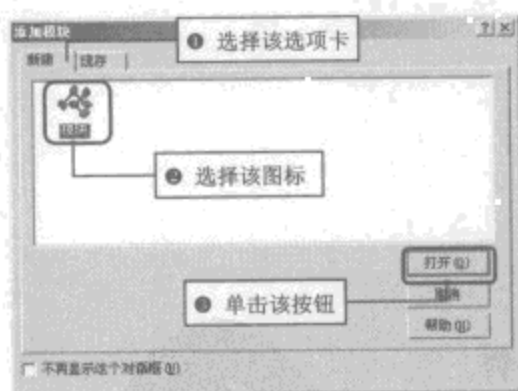



图 8.2 添加模块



图 8.3 添加了标准模块的工程

 说明：在工程资源管理器中单击鼠标右键，在弹出的快捷菜单中选择“添加”/“模块”命令，同样可以弹出如图8.2所示的对话框。

#### 2. 添加现存标准模块

对于一些比较通用的模块，如数据连接等，只需对其中的少部分内容进行修改，就可以将其应用到其他的程序中，这样减少了程序代码的编写量，加快了程序的开发速度。具体方法为：

选择“工程”/“添加模块”命令，在弹出的“添加模块”对话框中选择“现存”选项卡，选中需要添加的模块，单击“打开”按钮，如图 8.4 所示，即可将现存的标准模块添加到工程中。

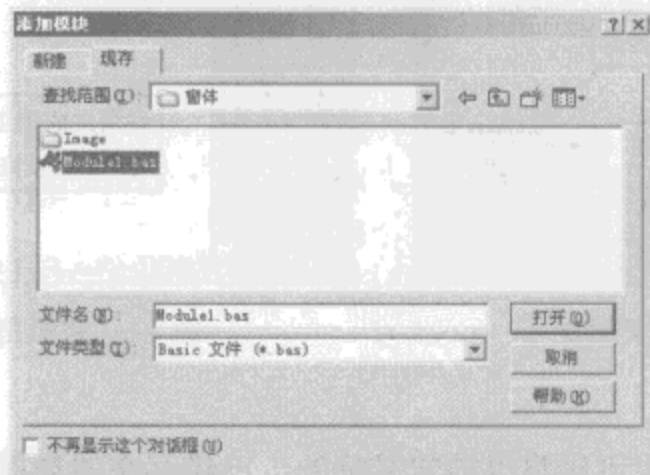


图 8.4 添加现存模块

## 8.2 类模块

教学录像：光盘\TM\lx\8\类模块.exe

类模块和标准模块在功能上比较类似。下面介绍什么是类模块以及在工程中如何添加类模块。

### 8.2.1 类模块的概述

类模块的扩展名是.cls，它是 VB 面向对象编程的基础。在类模块中可以编写代码创建新对象，该对象可以包括自己的属性、方法和事件。而自定义类模块的使用和使用 VB 中已经定义好的类是完全相同的。

在类模块中一般包括如下内容：

- (1) 常数、类型、变量和动态链接库的声明。
- (2) 子过程 (Sub)、函数过程 (Function) 和属性过程 (Property)。

### 8.2.2 添加类模块

#### 1. 添加新类模块

添加类模块和添加标准模块的方法类似，选择“工程”/“添加类模块”命令，在弹出的“添加类模块”对话框中选择“新建”选项卡，选择“类模块”图标，单击“打开”按钮，如图 8.5 所示，即可将新的类模块添加到工程中，如图 8.6 所示。

说明：在工程资源管理器中单击鼠标右键，在弹出的快捷菜单中选择“添加”/“添加类模块”命令，同样可以弹出如图 8.5 所示的对话框。

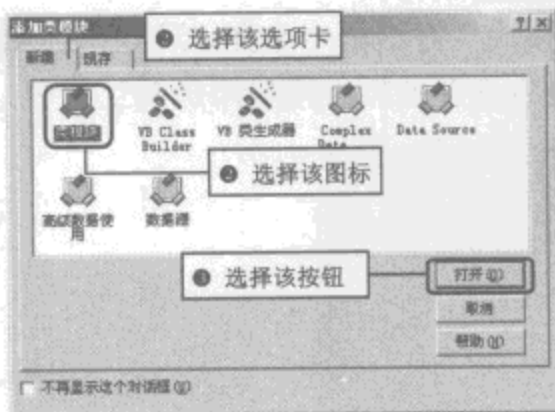


图 8.5 添加新类模块



图 8.6 添加完类模块的工程

## 2. 添加现存的类模块

和标准模块相同，类模块也可以将以前定义好的、已经存在的类模块添加到工程中，具体的方法为：选择“工程”/“添加类模块”命令，在弹出的“添加类模块”对话框中选择“现存”选项卡，选择要添加的类模块，单击“打开”按钮，如图 8.7 所示，即可将该模块添加到工程中。

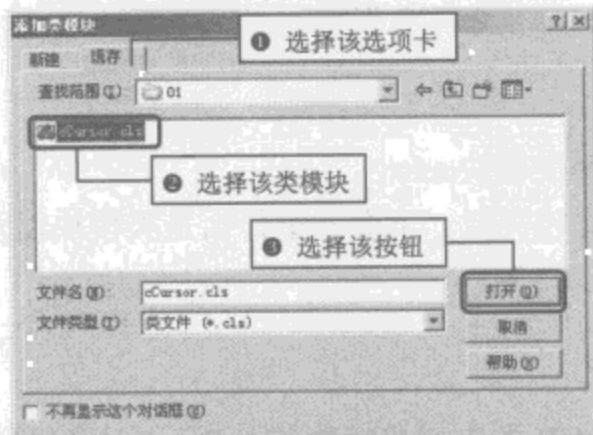


图 8.7 添加现存的类模块

## 8.3 标准模块和类模块的区别

**教学录像：**光盘\TM\lx\8\标准模块和类模块的区别.exe

大多数标准模块都可以转换为类模块，但是这种转换会使代码的可用性降低。在 VB 中标准模块和类模块各有其不同的用处：一般情况下，可以将那些与特定窗体或控件无关的代码放在标准模块中，这样一个过程可以响应不同对象的调用，避免了代码的重复。类模块既包含代码又包含数据，可以被理解是没有物理表现的控件。标准模块和类模块在使用时，可以从以下几点进行考虑：

- (1) 数据的存储方法。标准模块的数据存储只是一个备份，当其中定义的公共变量在程序运行时改变，后面的操作再次调用该变量时，得到的值还是原来的值。而类模块中的数据相对于类模块是独立存在的。
- (2) 变量的可见性。当变量在标准模块中声明为 Public 时，它在工程中任何地方都是可见的；而类模块中的 Public 变量，只有当对象变量含有对某一类实例的引用时才能访问。
- (3) 变量的引用。类模块必须要先在程序里进行引用，然后才可以使用。而公共模块则不需要进行引用就可以直接使用。

(4) 存活期。标准模块中的数据在程序作用域内存在，也就是说，它存在于程序的存活期中；而类模块中的数据只存在于对象的存活期内，它随对象的创建而创建，随对象的撤销而消失。

(5) 类模块是集结了公共模块中具有相同的方法或属性的模块内容。

例如，要写一个坦克大战的游戏。做成标准的模块，就需要将坦克的生命、攻击力、攻击范围等都写在标准模块中，相应的该模块就会变得很大，同时模块中的内容也比较繁杂。如果是做成类模块，因为每个坦克都有生命、攻击力、攻击范围等属性，所以可以将这些动作写成一个类模块，在使用时，直接调用这些属性即可。

综上所述，类模块可以把标准模块中的内容进行分类，使模块中的内容和作用更加清晰。

## 8.4 小结

程序的模块化是程序员程序设计水平不断提升的一个重要标志，在程序的开发过程中应力求达到程序的模块化。在 VB 6.0 中由于类的引入可以使程序代码更加简化，更加规范化。希望通过本章的学习，读者可以对模块和类模块有所了解，并达到融会贯通的效果。

## 8.5 练习与实践

1. 尝试在工程中添加一个名为 MyModule 的标准模块，并在该模块中声明一个公用整型变量 a。  
(答案位置：光盘\TM\sl\8\1)

2. 尝试在工程中添加一个名为 MyClass 的类模块，并在该模块中创建一个名为 Myval 的属性，属性值为字符型。(答案位置：光盘\TM\sl\8\2)



到美国，中国将开始向美国干戈，美国将开始向中国干戈，美国将开始向中国干戈（A）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（B）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（C）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（D）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（E）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（F）

美国将开始向中国干戈（G）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（H）

美国将开始向中国干戈（I）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（J）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（K）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（L）

美国将开始向中国干戈（M）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（N）

美国将开始向中国干戈（O）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（P）

美国将开始向中国干戈，美国将开始向中国干戈，美国将开始向中国干戈（Q）



# 第 9 章

## 常用标准控件

(教学录像: 1 小时 30 分钟)

在 VB 中, 控件是构成应用程序的最基本的组成部分。VB 的学习过程, 很大一部分是在学习各种控件的属性和设置。因此学习和掌握各种控件的使用方法尤为重要。

本章主要介绍控件的基本知识, 并详细介绍几种常用控件的使用方法, 使读者能够掌握并使用基本控件。

通过阅读本章, 您可以:

- » 了解控件的作用
- » 了解控件的分类
- » 掌握控件的相关操作
- » 了解开发过程中的几种常用控件
- » 掌握相应控件的属性、事件和方法
- » 学会相应控件的使用方法
- » 能够在程序中使用控件

## 9.1 控件概述

 教学录像：光盘\TM\lx\9\控件概述.exe

控件是 VB 开发环境中最重要的组成部分，VB 程序其实就是由许多控件组成的。学习 VB 编程就要学会各种控件的使用，下面介绍控件的基本知识。

### 9.1.1 控件的作用

控件用来实现用户与计算机的交互，还可以通过控件访问其他应用程序并处理数据。控件通过将固有的功能封装起来，只留出一些属性、方法和事件作为应用程序编写的接口，程序员在了解这些属性、事件和方法之后，就可以使用控件编写程序了。在通常情况下，基本的控件在工具箱中可以直接找到，如按钮、标签和列表框等控件，高级或特殊控件要将其添加到工具箱中才能使用。

控件对于面向对象的编程来说，具有非常重要的意义。VB 属于事件驱动程序，其程序代码大多是写进一个控件的事件中的。可以说 VB 程序功能的实现，就是窗体中每个控件的属性、方法和事件的实现。

VB 开发环境中的控件实际上是一个控件类，当某个控件被放置到窗体上时，就创建了该控件类的一个对象。当进入到运行模式时，一旦窗体被加载，就生成了控件运行时的对象。直到窗体被卸载时，该对象将被销毁。然而，当窗体再次出现在设计模式下时，会重新生成一个设计时的对象。

### 9.1.2 控件的属性、方法和事件

#### 1. 属性

控件属性是指控件的性质。如果把一个控件比喻成一个物品的话，则控件的属性就可以体现为该物品的颜色、大小、重量等。控件的属性可分为公共属性和专有属性。公共属性是每个控件都具有的属性，专有属性是针对某个控件的特有属性。

控件属性可在属性窗口中进行设置，或在程序代码中设置。在属性窗口中设置控件的属性比较方便直观，但是如果需要控件的属性在程序运行时改变，就必须在程序代码中设置其属性。

#### 2. 方法

方法只能在程序代码中使用，指的是某些规定好的、用于完成某种特定功能的特殊过程。例如 Show 方法用于显示窗体、Hide 方法用来隐藏窗体。

语法：

对象名.方法名[参数]

#### 3. 事件

事件是指能够被对象识别的一系列特定的动作。如 Click 鼠标单击事件、Load 窗体加载事件、KeyDown 键盘按下事件。

事件可以由用户激活（如键盘鼠标操作），也能被系统激活（如定时器事件），但在绝大多数情况下，事件都是被用户激活的。

### 9.1.3 控件的分类

在 VB 中控件主要分为标准内部控件和 ActiveX 控件两种。下面进行详细介绍。

#### 1. 标准内部控件

标准内部控件又称为常用控件，标准内部控件在 VB 开发环境中默认显示在工具箱中，如图 9.1 所示。这些控件是基础控件，使用的频率非常高，几乎所有的应用程序都会用到标准内部控件。

#### 2. ActiveX 控件

ActiveX 是扩展名为 .OCX 的独立文件，通常存放在 Windows 系统盘的 System 或 System32 目录下。在 VB 初始状态下的工具箱中，不包括 ActiveX 控件。ActiveX 控件拓展了 VB 的能力。如果使用 ActiveX 控件，应先将其添加到工具箱中。添加方法如下：

在 VB 开发环境的菜单中选择“工程”/“部件”命令，在弹出的对话框的列表中选择相应的 ActiveX 控件，如图 9.2 所示。单击“确定”按钮，ActiveX 控件就被添加到工具箱中了，如图 9.3 所示。

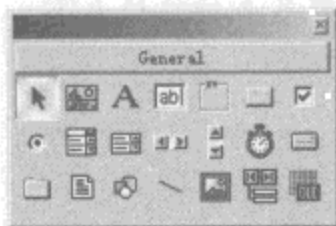


图 9.1 标准内部控件

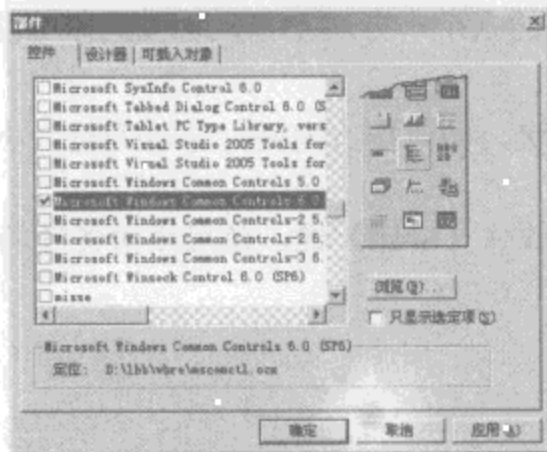


图 9.2 “部件”对话框

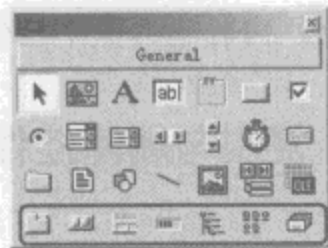


图 9.3 添加 ActiveX 控件到工具箱中

#### 3. 可插入对象

可插入对象又称为 OLE 控件，在 VB 的窗体中可以插入大量的第三方对象，也可以插入 Word、Excel 等对象。由于这些对象能够被添加到工具箱中，因此也称这些对象为控件，并且这些对象也可以像控件一样使用。

## 9.2 控件的相关操作

教学录像：光盘\TM\lx\9\控件的相关操作.exe

学习使用各种控件首先要学会控件的相关操作，学会在 VB 开发环境中对控件进行操作能够方便编程，有效地进行窗体布局，提高编程效率。下面介绍在 VB 开发环境中常用的控件操作。

## 9.2.1 向窗体上添加控件

向窗体上添加控件的方法很简单，主要有两种：

- ☑ 在工具箱中单击要添加到窗体中的控件，将鼠标放到窗体的适当位置后，按下鼠标左键，拖动到合适大小，再松开鼠标。
- ☑ 双击工具箱中要添加的控件，直接将控件添加到窗体上。

工具箱中的任何控件都可以采用这两种方法向窗体上添加。只是采用第二种方法添加控件后，需要在窗体上重新调整控件的大小和位置。

如果想在窗体上添加多个同一类型的控件，可以按住〈Ctrl〉键，单击工具箱中的控件，然后将鼠标放置在窗体上，当鼠标指针为十字形时，按住鼠标左键拖曳鼠标添加控件，重复此操作直到添加完所需要的控件为止。

## 9.2.2 调整控件的大小

控件添加到窗体上后，可以对其大小进行调整，以达到美观的效果。这里有两种调整方法：

- ☑ 选择控件，在该控件周边的八个小方块上当鼠标变为双箭头时，按住左键不放，然后拖曳到合适大小，松开鼠标。
- ☑ 选择控件，按住〈Shift〉键，同时按方向键，即可调整大小。

 技巧：同时选择多个控件然后按住〈Shift〉键，使用方向键可同时调整多个控件的大小。

## 9.2.3 复制与删除控件

### 1. 复制控件

控件就像文字一样可以剪切和复制。可以将一个窗体上的控件复制到另外一个窗体上，操作方法为：

- (1) 选中控件。
- (2) 在控件上单击鼠标右键，在弹出的快捷菜单中选择“复制”命令，或用工具栏上的“复制”按钮复制。
- (3) 在需要粘贴的窗体上，单击鼠标右键，在弹出的快捷菜单中选择“粘贴”命令。也可使用工具栏上的“粘贴”按钮粘贴。

 注意：将控件复制到同一窗体上实际上是创建了控件数组。

### 2. 删除控件

删除窗体上控件的方法很简单，用鼠标选择要删除的控件，直接按下键盘上的〈Delete〉键，或者直接单击鼠标右键，在弹出的快捷菜单中选择“删除”命令，即可删除所选择的控件。



### 3. 恢复被删除的控件

如果误删了某个控件，还可以将其恢复回来。单击工具栏中的“撤销删除”按钮，或者使用〈Ctrl+Z〉组合键来恢复所删除的控件。

**注意：**当连续删除多个控件时，撤销删除只能恢复最后一个被删除的控件，也就是撤销操作只能撤销最近一步的删除操作。

## 9.2.4 使用窗体编辑器调整控件布局

当窗体上控件有多个时，为了使窗体看起来整齐、美观，需要对窗体进行合理布局。例如对齐控件、统一控件的尺寸或者调整控件间的距离等。手动调整不但速度慢而且调整的效果也不会很好。为此 VB 给用户提供了窗体编辑器，用来进行窗体布局。窗体编辑器在工具栏中，如图 9.4 所示。下面介绍使用窗体编辑器进行窗体布局的方法。

(1) 如果窗体编辑器没有显示在工具栏中，首先在工具栏上单击鼠标右键，在弹出的快捷菜单中选中“窗体编辑器”命令，将窗体编辑器添加到工具栏中。

(2) 添加完窗体编辑器后，在窗体上选择要调整的控件，然后在“窗体编辑器”上选择相应的按钮，或在相应的下拉列表中选择相应的命令。

如图 9.5 所示，可以对选择的多个控件进行对齐。例如，如果控件是纵向排列的可以选择“左对齐”，如果控件是横向排列的可以选择“顶端对齐”或是“底端对齐”。

如图 9.6 所示，可以统一多个控件的尺寸。例如，要使选择的控件大小都相等，可以选择“两者都相同”。

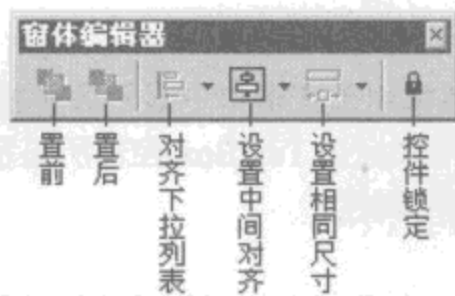


图 9.4 窗体编辑器

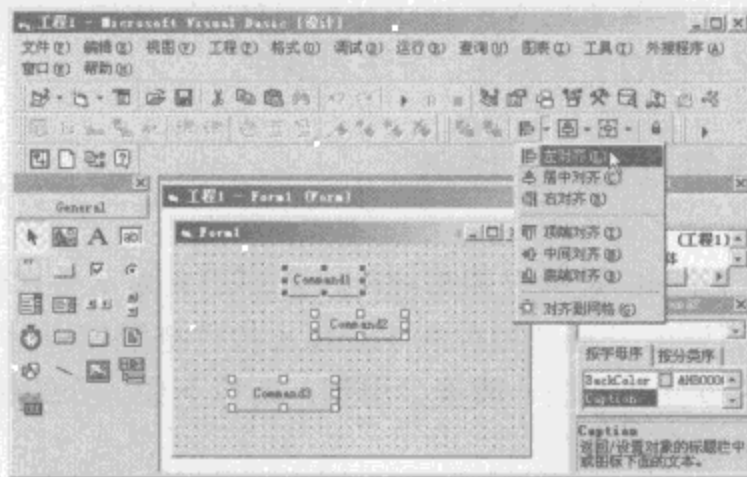


图 9.5 利用“窗体编辑器”对齐控件图

**说明：**在菜单栏的“格式”菜单下也可以实现上述功能。如图 9.7 所示。

## 9.2.5 锁定控件

在设计窗体时，有时会不小心将窗体中已经设计好的控件误调到其他位置，这样就必须重新调整。为了避免这种情况的发生，可以将设计好的控件锁定，使其不能移动或改变大小，防止设计时随意移动控件或改变控件大小。锁定控件的设置方法有多种：

- ☑ 选择菜单栏中的“格式”/“锁定控件”命令，即可锁定当前窗体中的控件。
- ☑ 在工具栏的窗体编辑器中直接单击“锁定控件”按钮，即可将窗体上控件锁定。
- ☑ 在窗体上单击鼠标右键，在弹出的快捷菜单中选择“锁定控件”命令，即可锁定窗体中的控件。如图 9.8 所示。

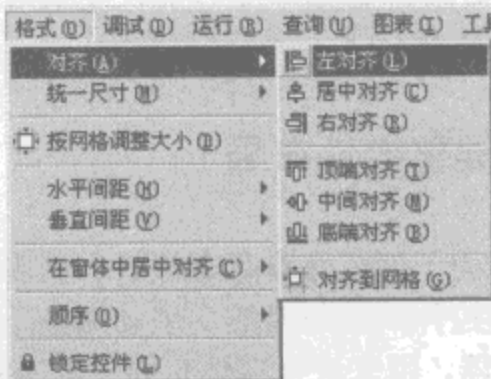
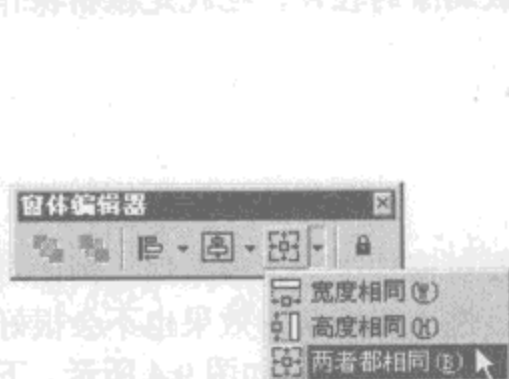


图 9.6 利用“窗体编辑器”设置控件尺寸相同 图 9.7 选择“格式”菜单中的命令 图 9.8 选择“锁定控件”命令

**说明：**解除对控件的锁定，只需再次选择“锁定控件”命令，或是单击窗体编辑器中的“锁定控件”按钮，窗体上的控件即可进行移动或更改大小。

## 9.3 标签和文本框

**教学录像：**光盘\TM\lx\9\标签和文本框.exe

标签和文本框都是 VB 程序开发中最经常使用的控件，主要应用于文本信息的操作，本节主要介绍标签和文本框的主要属性、方法和事件。

### 9.3.1 标签 (Label 控件)

标签 (Label) 控件是图形控件，主要用来显示文本信息，通常用于在窗口中显示各种操作提示和文字说明。例如，Label 控件和 TextBox 控件搭配使用时，标签用来标识 TextBox 控件所显示的内容。因为标签的事件和方法一般很少用到，所以下面只介绍标签的常用属性。

#### (1) Caption 属性

Caption 属性是标签最重要的属性，用于确定标签控件的显示文本内容。例如设置标签的 Caption 属性为“学生姓名”，其效果如图 9.9 所示。

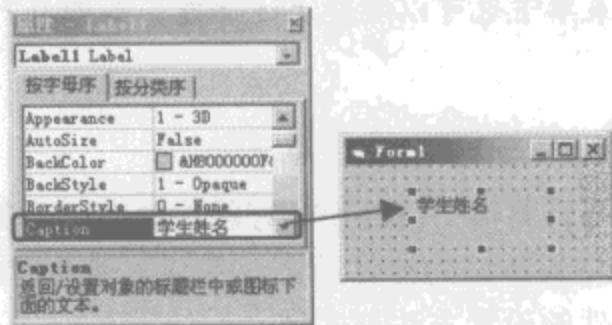


图 9.9 设置 Caption 属性

可以通过代码设置 Caption 属性,代码如下:

```
Label1.Caption = "学生姓名"
```

### (2) AutoSize 属性

AutoSize 属性用于决定标签是否自动改变大小来显示其全部的内容。当属性值为 True 时,标签会根据标题内容自动调整大小;当属性值为 False 时,控件将保持设计时定义的大小,超出控件区域的内容将被覆盖。该属性设置显示效果如图 9.10 所示。

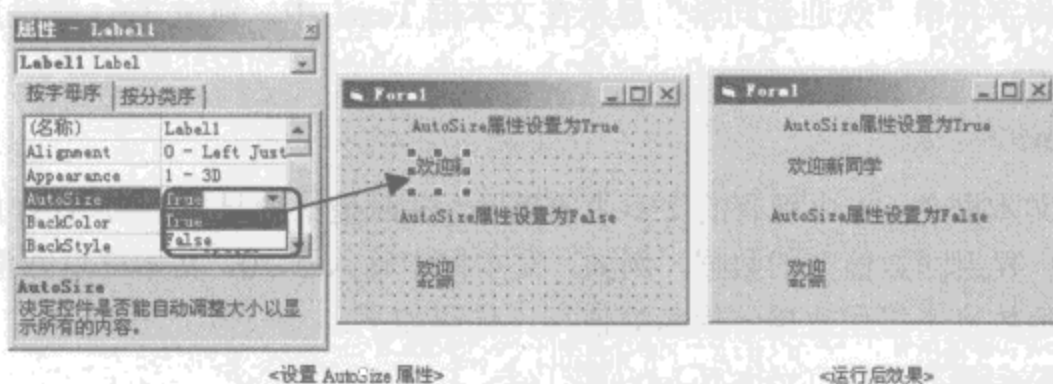


图 9.10 AutoSize 属性设置显示效果

另外,也可以通过代码实现设置 AutoSize 属性,代码如下:

```
Label1.AutoSize = True  
Label2.AutoSize = False
```

### (3) BackStyle 属性

BackStyle 属性返回或设置一个值,该属性决定标签控件的背景是否透明。在有背景图片的窗体上放置标签控件时,可以通过 BackStyle 属性来创建一个透明的控件。

当 BackStyle 属性设置为 0 时,该控件的背景是透明的;当 BackStyle 属性值为 1 时,其背景不透明是可见的。其效果如图 9.11 所示。

另外,也可以通过代码实现设置 BackStyle 属性,代码如下:

```
Label1.BackStyle = 1: Label2.BackStyle = 0
```

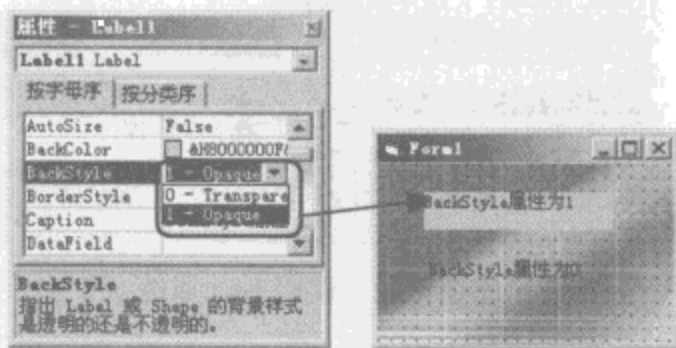


图 9.11 设置 BackStyle 属性

## 9.3.2 文本框 (TextBox 控件)

文本框 (TextBox) 在窗体中为用户提供了一个既能显示又能编辑文本的对象。在文本框内,可进行文字编辑,如进行选择、删除、复制、粘贴和替换等操作。文本框常用于显示运行时代码中赋予控件的信息或显示用户输入的信息。

下面介绍文本框常用的属性、事件和方法。

## 1. 文本框的属性

### (1) Text 属性

Text 属性是文本框最重要的属性,用于返回或设置编辑域中的文本。该属性设置显示效果如图 9.12 所示。

另外,也可以通过代码进行设置。例如将设置的内容显示在文本框中,代码如下:

```
Text1.Text = "欢迎新同学"
```

该语句实现了将字符串“欢迎新同学”显示在文本框 Text1 中。通过 Text 属性还可以实现返回文本框内容,代码如下:

```
Mystr = Text1.Text
```

上面代码是将文本框 Text1 内现有的文本内容返回,并赋值给变量 Mystr。Text 属性返回值的类型是字符型,如果用户要返回数值型的数据,例如,文本框中输入了 20,文本框返回值为字符串“20”,这时需要使用 Val 函数将字符型数据转换为数值型。代码如下:

```
Mystr = Val(Text1.Text)
```

### (2) PasswordChar 属性

PasswordChar 属性用来设置文本框内键入的字符如何显示,多用于密码文本框,以此来隐藏密码。

例如,将要显示密码的文本框的 PasswordChar 属性设置为“\*”号,则文本框内字符全部显示为“\*”号。效果如图 9.13 所示。

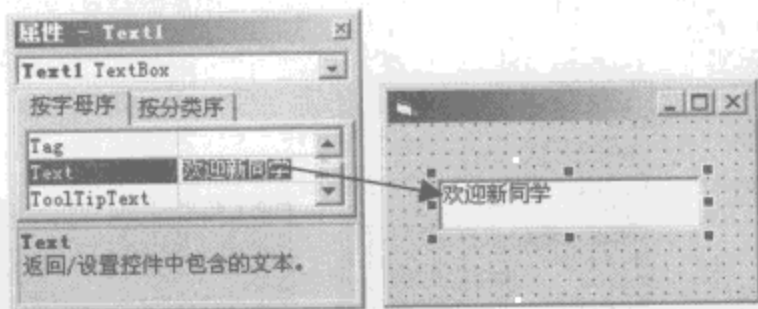


图 9.12 设置 Text 属性

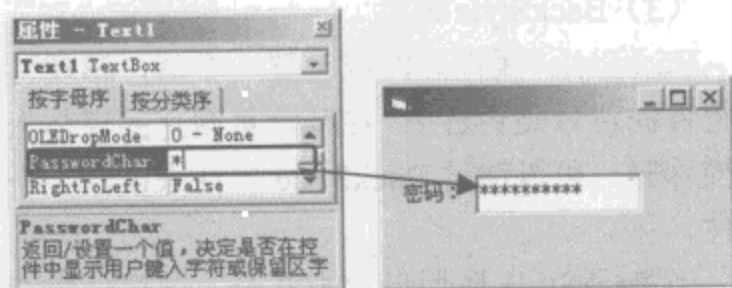


图 9.13 设置 PasswordChar 属性效果

说明: 当然也可以使用其他字符或是符号代替“\*”号,但是覆盖密码的符号习惯上使用“\*”号。

默认情况下 PasswordChar 属性为空,文本框内显示输入的字符。另外,也可以通过代码设置该属性,代码如下:

```
Text1.PasswordChar = "*"'
```

说明: 一般手动设置该属性比较方便,除非有特殊需要。

### (3) Font 属性

Font 属性用来设置文本框显示文本的字体、字号、是否为粗体等。用属性窗口设置 Font 属性大家一定都很熟悉,下面介绍使用代码设置该属性。

Font 属性包含 FontName (字体名称)、FontSize (字体大小)、FontBold (粗体)、FontItalic (斜体)、FontUnderline (下划线) 等。例如,通过下面代码设置文本框 Text1 中文本的字体效果,代码如下:



```
Text1.FontName = "黑体"
```

```
Text1.FontSize = 10: Text1.FontBold = True: Text1.FontItalic = True: Text1.FontUnderline = True
```

上面的代码设置文本框 Text1 中的文本内容的字体名称为黑体, 字号为 10, 并设置字符为粗体、斜体并加了下划线。运行效果如图 9.14 所示。

#### (4) MultiLine 属性

MultiLine 属性用来指定文本框内文本是否能够换行, 也就是能否显示多行文本。当该属性值为 True 时, 文本框运行多行显示文本; 当该属性值为 False 时, 文本框忽略回车符并将文本内容限制在一行内。默认值为 False。在程序运行时, 该属性为只读。其设置效果如图 9.15 所示。



图 9.14 Font 属性设置效果

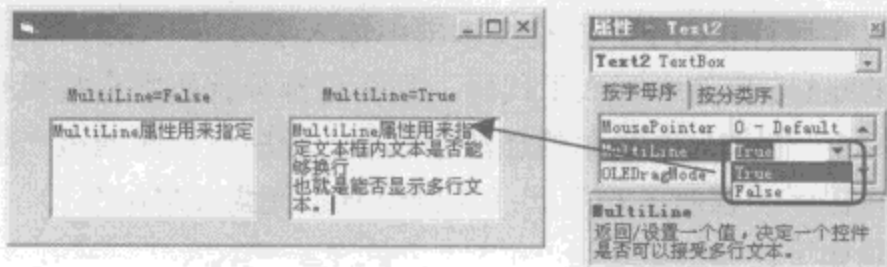


图 9.15 MultiLine 属性设置效果

**说明:** 当输入多行文本时, 如果 Multiline 属性设置为 True, 在 Text 属性处会显示下拉按钮, 如图 9.16 所示。此时必须使用〈Ctrl+Enter〉组合键才能把焦点移到下一行。

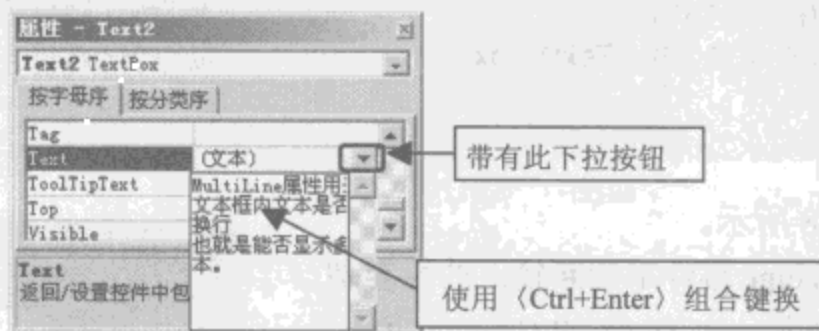


图 9.16 文本设置时换行

#### (5) ScrollBars 属性

ScrollBars 属性用于设置文本框的滚动条显示方式。该属性值默认值为 0, 表示控件中没有滚动条; 当该属性值设置为 1 时, 表示该控件中只有水平滚动条; 当该属性设置为 2 时, 表示控件中只有垂直滚动条; 当该属性值设置为 3 时, 表示该控件中既有水平滚动条又有垂直滚动条。其效果如图 9.17 所示。

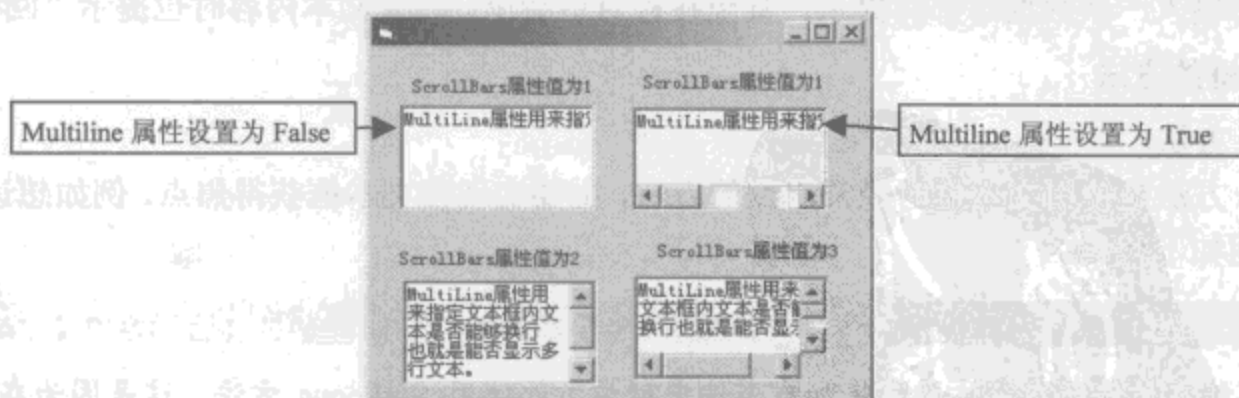


图 9.17 ScrollBars 属性设置效果



只有当 Multiline 属性设置为 True 时,文本框才能添加滚动条。此属性是只读属性,不能用代码进行设置。

#### (6) Locked 属性

Locked 属性用于指定文本框在运行时能否进行编辑。当 Locked 属性值为 True 时,文本框中可以滚动和加亮控件中的文本,但不能对内容进行编辑;当 Locked 属性值为 False 时,文本框中可以编辑文本。

### 2. 文本框的事件

#### Change 事件

程序运行后当文本框内容进行了改变或是文本框的 Text 属性有所改变时将触发 Change 事件。例如,在程序运行状态下,在空文本框中输入字符串“1234”,则会触发 4 次 Change 事件。如果在该文本框的 Change 事件中写入了代码,那么这部分代码将会被执行 4 次。

**例 9.1** 制作一个简单的加法测试器,程序运行时随机产生整数,在后面的文本框内输入得数,如果正确将提示“答对了!”。(实例位置:光盘\TM\9\1)

代码如下:

```
Private Sub Text3_Change()
```

```
    If Val(Text3.Text) = Val(Text1.Text) + Val(Text2.Text) Then
```

```
        MsgBox "答对了!"
```

```
        Text3.Text = ""
```

```
        Text1.Text = Int(Rnd() * 10)
```

```
        Text2.Text = Int(Rnd() * 10) + Text1.Text
```

```
    End If
```

```
End Sub
```

'Text3 的 Change 事件

'判断如果第三个文本框为前两个文本框之和

'提示答对了信息

'将文本框设置为空

'为 Text1 赋随机数

'为 Text2 赋值

程序运行效果如图 9.18 所示。

实例 9.1 在 Text3 的 Change 事件下判断输入的内容是否正确,每当输入一次就触发一次 Text3 的 Change 事件,执行一遍 Change 事件下的代码。输入正确时显示提示信息,不正确时没有反应。读者可以尝试在 End If 语句前加上如下代码:

```
Else
```

```
    MsgBox "回答错误!"
```

这样在每输入一次数据时都有提示,甚至在答对后清空 Text3 文本内容时也提示“回答错误!”。

### 3. 文本框的方法

#### SetFocus 方法

SetFocus 方法是使用文本框时经常用到的方法,主要实现使文本框获得焦点。例如想让 Text1 获得焦点,可使用如下代码:

```
Text1.SetFocus
```

**注意:** 不能在控件所在窗体的窗体加载事件中对该控件使用 SetFocus 方法,这是因为在窗体的 Load 事件完成前窗体或控件都是不可视的,所以不能使用 SetFocus 方法将焦点移动到正在加载的控

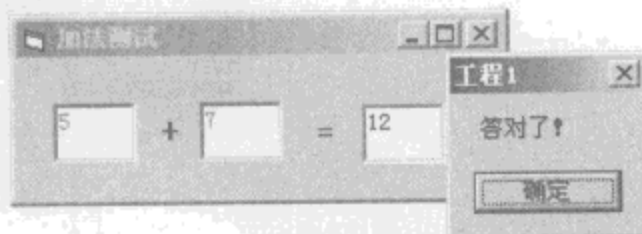


图 9.18 加法测试程序运行效果

件或窗体上。但是可以通过在 Form\_Activate 事件中添加 Text1.SetFocus 代码来实现焦点的设置。

## 9.4 命令按钮

 教学录像：光盘\TM\lx\9\命令按钮.exe

命令按钮 (CommandButton) 也是编程中最常应用的控件，其使用方法简单，用户可以通过单击按钮来执行操作。命令按钮常被用来启动、中断或结束一个进程。通过编写命令按钮的 Click 事件，可以指定按钮的功能。下面介绍命令按钮的属性和事件。

### 9.4.1 命令按钮的属性

#### (1) Caption 属性

命令按钮的 Caption 属性用于设置按钮的显示标题文字，通常被设置为显示按钮的功能。例如，“确定”、“退出”、“上一步”等。命令按钮的 Caption 属性设置后显示效果如图 9.19 所示。

另外，命令按钮的 Caption 属性也可以通过代码进行设置，实现上面操作的代码如下：

```
Command1.Caption = "确定"
```

#### (2) Picture 属性

Picture 属性用于在按钮上显示图片。为按钮设置图片后效果如图 9.20 所示。



图 9.19 设置按钮 Caption 属性



图 9.20 为按钮加载图片

 注意：只有当 Style 属性设置为 1 时，命令按钮上才能显示加载的图片。

还可以使用代码实现为按钮加载图片，代码如下：

```
Command1.Picture = LoadPicture("D:\登录.bmp")
```

其中括号内内容为图片所在路径。

#### (3) Default 属性和 Cancel 属性

这两个属性分别用于设置使用键盘上的 <Enter> 键和 <Esc> 键触发窗体上相应的按钮单击事件。当命令按钮的 Default 属性设置为 True 时，在默认情况下，在运行程序时按回车键就等于用鼠标单击了该按钮。当命令按钮的 Cancel 属性设置为 True 时，程序运行时当按下键盘上的 <Esc> 键时就相当于单击了此按钮。

在程序设计时，将“确定”、“是”等按钮的 Default 属性设置为 True，将“取消”、“否”等按钮的 Cancel 属性设置为 True。这样既符合 Windows 操作系统的使用风格，用户使用起来也会很方便。

## 9.4.2 命令按钮的事件

命令按钮最常用的事件就是 Click 事件。程序运行时用户单击按钮触发该事件，将执行该事件下的代码，实现相应的功能。

**例 9.2** 在窗体上添加两个命令按钮，并进行属性设置。当单击“确定”按钮时，在窗体上输出“您单击了确定按钮！”；当单击“取消”按钮时，窗体上的文本消失。代码如下：（实例位置：光盘\TM\9\2）

```
Private Sub Command1_Click()           'Command1 的 Click 事件
    Print "您单击了确定按钮"         '在窗体上打印字符串
End Sub
Private Sub Command2_Click()         'Command2 的 Click 事件
    Form1.Refresh                    '刷新窗体
End Sub
```

程序运行效果如图 9.21 所示。

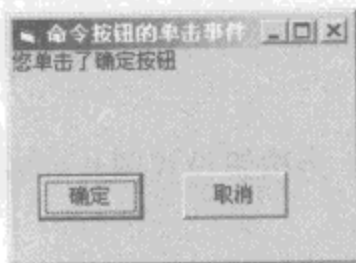


图 9.21 命令按钮的单击事件

## 9.5 单选按钮、复选框及框架

 教学录像：光盘\TM\9\9\单选按钮、复选框及框架.exe

即使不在 VB 程序中，单选按钮和复选框也经常能够看到。例如类似如图 9.22 所示的管理员信息设置界面，其中选择“性别”的控件叫做单选按钮，选择“管理权限”的控件叫做复选框。单选按钮只能在一组选项中选择一项，而复选框可以在一组选项中选择多项。本节主要介绍单选按钮、复选框及框架的主要属性、方法和事件。

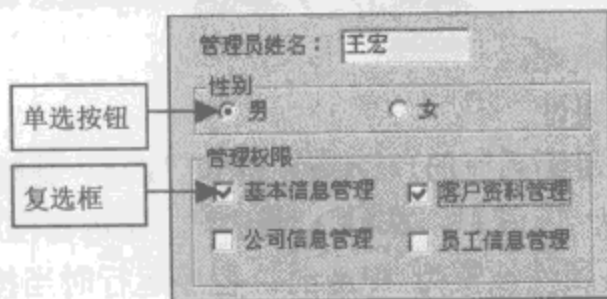


图 9.22 单选按钮与复选框显示界面

### 9.5.1 单选按钮 (OptionButton 控件)

单选按钮 (OptionButton) 表示给用户一组选项, 用户只能在这组选项中选择一项。单选按钮总是作为一个组来使用, 当选中某一选项时, 该单选按钮的圆圈内显示一个黑点, 表示选中, 同时其他单选按钮中的黑点消失, 表示未选中。

#### 1. 单选按钮的属性

##### (1) Caption 属性

Caption 属性用于设置显示在单选按钮中的文本信息及单选按钮的标题。设置单选按钮的 Caption 属性效果如图 9.23 所示。

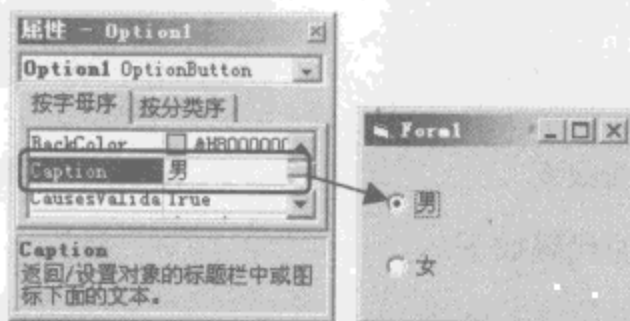


图 9.23 设置单选按钮的 Caption 属性

单选按钮的 Caption 属性也可以通过代码进行设置, 代码如下:

```
Option1.Caption = "男"; Option2.Caption = "女"
```

##### (2) Value 属性

Value 属性是单选按钮比较重要的属性, 用于返回或设置控件的状态。当单选按钮被选中时, Value 属性值为 True, 当单选按钮没被选中时其 Value 属性值为 False。例如图 9.23 中, 第一个单选按钮 (Caption 为“男”) 的 Value 值为 True, 第二个单选按钮 (Caption 为“女”) 的 Value 值为 False。

一般情况下, 单选按钮的 Value 属性不需要在属性窗口进行设置。在程序运行时, 系统会自动为单选按钮组中的第一个按钮的 Value 值赋值为 True, 也就是在程序运行时, 会默认选中一个单选按钮。Value 属性常用于返回一个单选按钮的状态, 然后根据按钮状态进行判断或其他操作。在代码设计时其语句如下:

```
object.Value[=value]
```

object: 对象表达式。

value: 该值指定控件状态、内容或位置。当值设置为 True 时, 表示已经选择了该按钮; 当值为 False 时, 表示没有选择该按钮。

##### (3) Style 属性

Style 属性用来指示单选按钮的显示类型和行为, 在程序运行时是只读的。当值为 0 时 (默认样式), 为标准单选按钮显示方式, 即一个同心圆和一个标题的显示方式; 当值为 1 时, 以图形方式显示, 显示为命令按钮样式。如图 9.24 所示。

除特殊需要, 单选按钮一般都使用默认样式, 即空心圆加上标题的显示样式, 这样符合用户的使



用习惯。

## 2. 单选按钮的事件

单击单选按钮会触发 Click 事件，也可在代码中通过将 Value 属性设置为 True 触发 Click 事件。下面通过实例介绍单选按钮的 Click 事件的使用方法。

**例 9.3** 在程序运行时单击单选按钮，标签中字体大小改变。实现效果如图 9.25 所示。（实例位置：光盘\TM\sl\9\3）

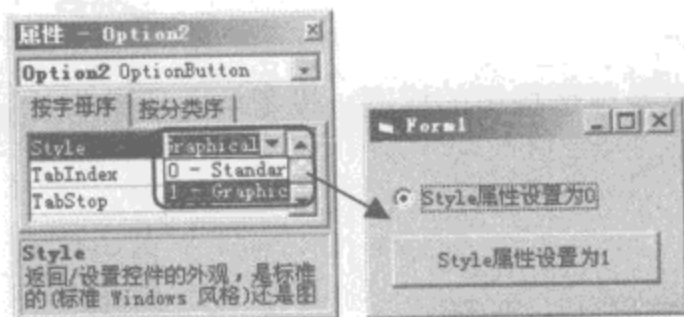


图 9.24 Style 属性设置效果

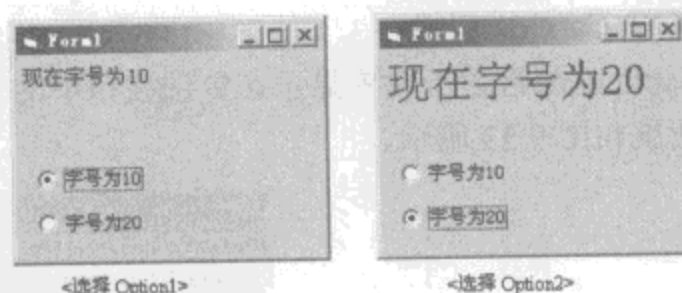


图 9.25 单选按钮单击事件实例演示

两个单选按钮的 Click 事件的代码如下：

```
Private Sub Option1_Click()  
    Label1.FontSize = 10: Label1.Caption = "现在字号为 10"  
End Sub  
Private Sub Option2_Click()  
    Label1.FontSize = 20: Label1.Caption = "现在字号为 20"  
End Sub
```

运行程序后，默认选择“字号为 10”单选按钮，这时此单选按钮的 Value 值为 True，所以同样响应该按钮的单击事件，并执行其中的代码，将标签中文本显示为 10 号字体。

## 9.5.2 复选框（CheckBox 控件）

复选框（CheckBox）和单选按钮一样有两种状态即“选中”和“未选中”。当复选框被选中时，在前面的方框内显示一个“√”号。在一组复选框中可以选择多个选项，也可以一个都不选。下面介绍复选框的属性和事件。

复选框的属性和单选按钮的属性基本相同，只是 Value 属性有很大的差别，下面介绍复选框的 Value 属性。

复选框的 Value 属性同样是用来返回或设置控件的状态。只是复选框的 Value 值包含 3 个，即 0、1 和 2。当 Value 属性值为 0 时，表示控件没有被选中；当值为 1 时，表示已选中；当值为 2 时，表示控件变灰，此时控件禁止使用。其效果如图 9.26 所示。

## 9.5.3 框架（Frame 控件）

框架（Frame 控件）用于为控件提供可标识的分组。单独使用框架控件没有什么实际意义，框架



和窗体一样可以看成是容器类控件，将窗体上相同性质的控件放在框架中进行分组。

例如，窗体上的单选按钮组在程序运行时只能选择其中的一个。如果使用框架将单选按钮分成几组，那么每组中就都可以选择一个单选按钮。如图 9.27 所示，将单选按钮分成了两组，这样每一组中都可以选择一项。

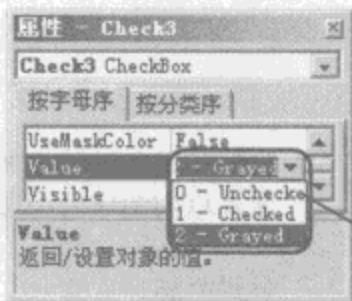


图 9.26 设置 Value 属性

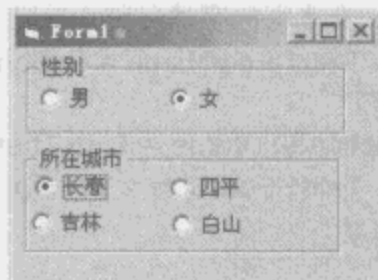
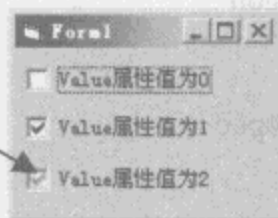


图 9.27 框架作用演示

**注意：**窗体上的控件是不能被拖曳到框架中的，必须将控件直接从工具箱添加到框架中或是粘贴到框架中。框架中的控件会随着框架的移动而移动。当框架不够大时，框架边框以外的部分将被覆盖。

下面介绍框架控件的常用属性。

#### (1) Caption 属性

框架的 Caption 属性用于显示在框架中的文本信息，即显示框架内容的标题。设置框架的 Caption 属性效果如图 9.28 所示。

#### (2) BorderStyle 属性

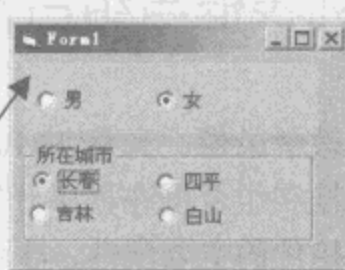
BorderStyle 属性用于设置框架是否有边线。当 BorderStyle 属性值为 0 时，框架无边线；当属性值为 1 时（默认值），框架有凹陷边线。该属性设置效果如图 9.29 所示。



图 9.28 框架 Caption 属性设置效果



图 9.29 BorderStyle 属性设置效果



**说明：**当 BorderStyle 属性值为 0 时，框架的标题也不显示。

**例 9.4** 本实例实现设置字体的功能，在程序运行时，选择字体、字号、效果和颜色后单击“确定”按钮，文本框内的文字就会显示设置的效果。其实现效果如图 9.30 所示。（实例位置：光盘\TM\sl\9\4）



图 9.30 字体设置程序界面

程序关键代码如下：

```
Private Sub Command1_Click() '按钮的 Click 事件
    For i = 0 To 2 '循环语句
        '判断如果单选按钮 Value 值为 True
        If Option1(i).Value = True Then
            '文本框的字体名称为单选按钮的标题
            Text1.FontName = Option1(i).Caption
        End If
        '判断如果单选按钮的 Value 值为 True
        If Option3(i).Value = True Then '文本框内的字号为单选按钮的标题
            Text1.FontSize = Val(Option3(i).Caption)
        End If
        '继续循环
    Next i
    '是否为粗体
    Text1.FontBold = Check1.Value
    '是否为斜体
    Text1.FontItalic = Check2.Value
    '是否有下划线
    Text1.FontUnderline = Check3.Value
    '是否有删除线
    Text1.FontStrikethru = Check4.Value
    '如果选择红，则文本字体颜色为红色
    If Option2(0).Value = True Then Text1.ForeColor = vbRed
    '如果选择黄，则文本字体颜色为黄色
    If Option2(1).Value = True Then Text1.ForeColor = vbYellow
    '如果选择蓝，则文本字体颜色为蓝色
    If Option2(2).Value = True Then Text1.ForeColor = vbBlue
End Sub
```

在上述代码中使用了控件数组，利用循环语句查找被选中的单选按钮，单选按钮的 Caption 值恰好可以作为文本的字体名称（或字号大小），所以直接使用单选按钮的 Caption 属性值为文本框字体和字号大小赋值。对于字体效果的设置使用了复选框，当复选框被选中时也就是选择了该字体效果，所以使用复选框当前 Value 值（True 或 False）为文本框字体效果属性值赋值。

## 9.6 列表框与组合框

 教学录像：光盘\TM\lx\9\列表框与组合框.exe

列表框和组合框在程序开发中也经常使用，主要用于显示和选择数据，本节主要介绍列表框控件与组合框控件的主要属性、事件和方法。

### 9.6.1 列表框（ListBox 控件）

列表框（ListBox 控件）用于显示项目列表，从列表中可以选择一项或多项。如果有多种项目让用户选择，使用列表框将会很方便。在项目总数超过可显示的项目数时，则自动在列表框上添加滚动条。下面介绍列表框的主要属性、方法和事件。

#### 1. 列表框属性

##### （1）Columns 属性

Columns 属性用来确定列表框项目显示的列数。当属性值为 0 时（默认值），以单列显示，在项目条数较多不能全部显示出来时，自动添加竖直滚动条；当属性值设置为大于 0 的数值时，列表框中显示指定的列数，列表框水平滚动。Columns 属性设置效果如图 9.31 所示。

## (2) List 属性

List 属性用于返回或设置控件列表部分的项目。可以在属性窗口通过 List 属性设置项目内容，也可以使用代码进行提取或者设置。列表是一个字符串数组，数组的每一项都是一个列表项目。

设置列表框的 List 属性为“长春、四平、吉林”等信息，其效果如图 9.32 所示。



图 9.31 Columns 属性设置显示效果

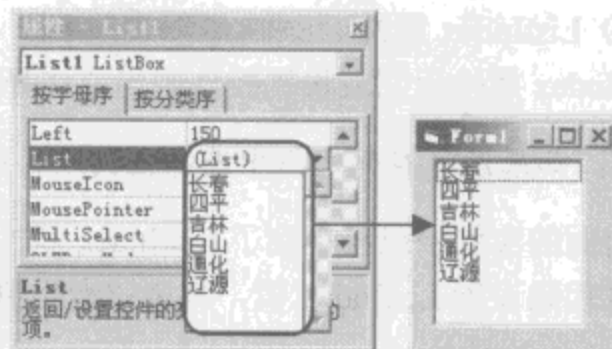


图 9.32 List 属性设置效果

**技巧：**每输完一项后，按〈Ctrl+Enter〉组合键换行。

也可以通过代码实现上述操作，代码如下：

```
List1.List(0) = "长春"
List1.List(1) = "四平"
List1.List(2) = "吉林"
.....
```

**例 9.5** 本实例实现在程序运行时，选择列表框中的项目时，在下面的文本框内显示选择的项目。效果如图 9.33 所示。（实例位置：光盘\TM\9\5）

程序代码如下：

```
Private Sub List1_Click() '列表框单击事件
    '将选择的项目的内容赋给标签标题
    Label1.Caption = "选择的城市为：" & List1.List(List1.ListIndex)
End Sub
```

## (3) ListCount 属性

ListCount 属性用于返回列表框中项目的个数。

**例 9.6** 本实例实现利用 ListCount 属性来获取列表中元素个数，并将其显示在窗体的标签控件中。程序运行效果如图 9.34 所示。（实例位置：光盘\TM\9\6）

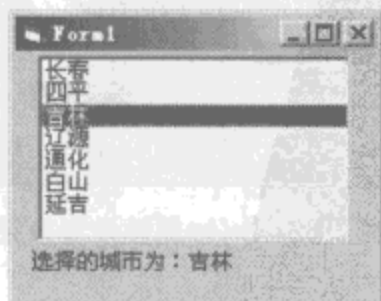


图 9.33 List 属性的应用

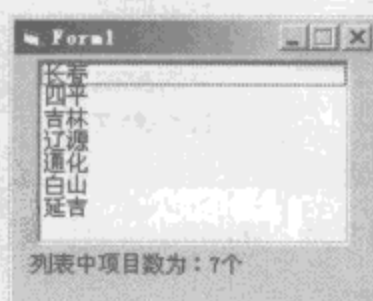



图 9.34 ListCount 属性应用

程序代码如下：

```
Private Sub Form_Load()
    Label1.Caption = "列表中项目数为：" & List1.ListCount & "个"
End Sub
```

#### (4) ListIndex 属性

ListIndex 属性返回或设置控件中当前选择项目的索引值，在程序设计时不可用。

 说明：如果没有在列表框中选择任何项，ListIndex 属性值为-1。

在前面的实例中已经用到了 ListIndex 属性，使用 ListIndex 属性获取当前选择的选项的索引值。也可以通过代码设置 ListIndex 值，将为该索引值的项目选中。例如执行下面的代码：

```
List1.ListIndex = 2
```

则在图 9.33 所示的 List1 中第 3 项被选中。

#### (5) MultiSelect 属性

MultiSelect 属性决定了能否在列表框中选择多项，以及选择多项时的选择方式。其属性值有 0、1 和 2。具体描述如下：

- ☒ 当属性值为 0 时，表示一次只能选择一项，不能选择多项。
- ☒ 当属性值为 1 时，表示允许选择多项，单击或按下〈Spacebar〉键（空格键），在列表框中选中或取消选中项（方向键移动焦点）。
- ☒ 当属性值为 2 时，表示可以选择列表框中某个连续范围内的项，按下〈Shift〉键并单击或按下〈Shift〉键以及一个方向键，将在以前选中项的基础上扩展选择到当前选中项。按下〈Ctrl〉键并单击，可在列表框中选中或取消选中项。

MultiSelect 属性设置效果如图 9.35 所示。

#### (6) Style 属性

Style 属性用于设置列表框的显示样式和行为。在运行时该属性只读。

当 Style 属性值为 0 时，显示标准样式，即显示样式如文本项的列表；当 Style 属性值为 1 时，每一个文本项的边上都有一个复选框，此时在列表框中可以选择多项。设置效果如图 9.36 所示。

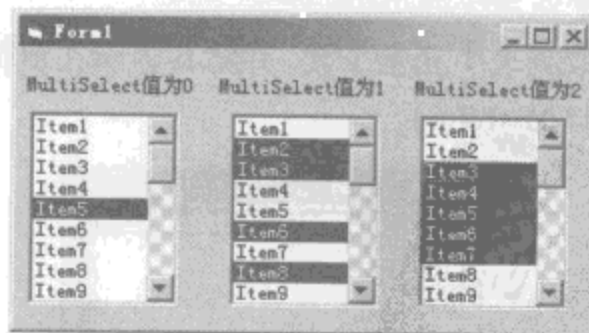


图 9.35 MultiSelect 属性设置效果

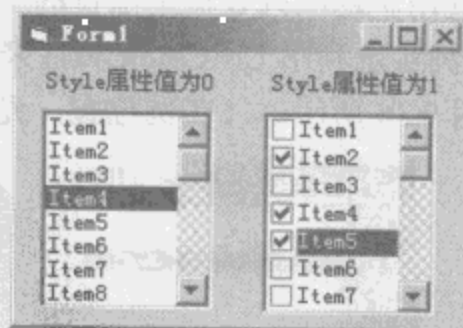


图 9.36 Style 属性设置效果

## 2. 列表框的方法

### (1) AddItem 方法

AddItem 方法用于将项目添加到列表框中。前面已经介绍，使用列表框属性窗口设置 List 属性同



样可以将项目添加到列表框中。

AddItem 方法的语法:

```
object.AddItem item, index
```

object: 必需的。一个对象表达式。

Item: 必需的。字符串表达式, 它用来指定添加到该对象的项目。

Index: 可选的。是整数, 它用来指定 ListBox 控件的首项。

例 9.7 本实例实现运行程序后, 单击“添加”按钮, 就会在列表框中添加一项, 并将新添加的内容显示在下面的标签中。程序运行效果如图 9.37 所示。(实例位置: 光盘\TM\sl\9\7)

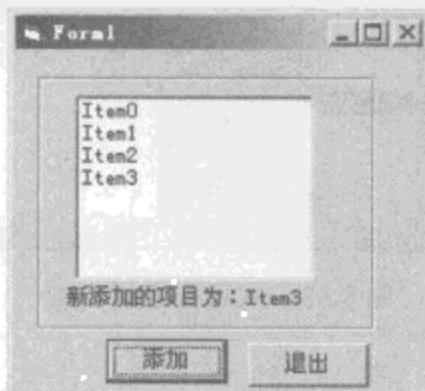



图 9.37 AddItem 方法应用

程序代码如下:

```
Dim i As Integer           '定义全局变量
Private Sub Command1_Click()
    List1.AddItem "Item" & i    '向列表框中添加一项
    '标签显示添加的新项
    Label1.Caption = "新添加的项目为: " & List1.List(i)
    i = i + 1                 '变量值加 1
End Sub
```

 说明: 上面的实例中须将变量 i 定义为全局型, 这样在每次单击后变量值都会加 1。或者定义为静态的局部变量, 将 Dim 替换为 Static 即可。

## (2) Clear 方法

Clear 方法用于清除列表框中的项目。

语法:

```
object.Clear
```

object: 一个对象表达式。

例如, 清除 List1 中的所有内容, 代码如下:

```
List1.Clear
```

## (3) RemoveItem 方法

RemoveItem 方法用于从列表框中删除指定项。



语法:

object.RemoveItem index

object: 必需的参数。一个对象表达式。

index: 必需的参数。一个整数, 表示要删除的列表框中的首项。

**例 9.8** 本实例实现当程序运行时, 选择左边列表框中的项目, 单击“添加”按钮, 将选中的项目添加到右边的列表框中; 单击“移除”按钮, 将选中的项目从右边的列表框中移除。单击“全部清除”按钮, 将右边的列表框清空。程序运行效果如图 9.38 所示。(实例位置: 光盘\TM\sl\9\8)

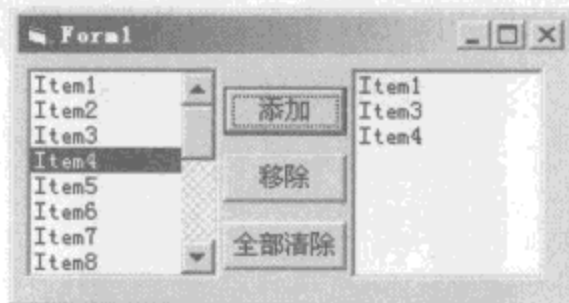


图 9.38 列表框应用实例

程序代码如下:

```
Private Sub Command1_Click()
    '添加按钮单击事件
    If List1.ListIndex = -1 Then
        '如果在 List1 中没有选择项目
        MsgBox "请选择项目"
        '提示信息
        Exit Sub
        '退出过程
    End If
    For i = 1 To 20
        '变量循环
        If List1.List(List1.ListIndex) = List2.List(i) Then
            '如果在 List1 中选中的项目在 List2 中已存在
            MsgBox "该项已添加"
            '提示信息
            Exit Sub
            '退出过程
        End If
        '继续循环
    Next i
    '将在 List1 中选择的项目添加到 List2 中
    List2.AddItem (List1.List(List1.ListIndex))
End Sub
Private Sub Command2_Click()
    '移除按钮的单击事件
    If List2.ListIndex = -1 Then
        '如果在 List2 中没有选择项目
        MsgBox "请选择要移除的项目!"
        '提示信息
        Exit Sub
        '退出过程
    End If
    '将选中项目移除
    List2.RemoveItem (List2.ListIndex)
End Sub
Private Sub Command3_Click()
    '全部清除按钮的单击事件
    If MsgBox("确定清空?", 4) = vbYes Then List2.Clear
    '如果选择确定, 将 List2 清空
End Sub
```

### 9.6.2 组合框 (ComboBox 控件)

组合框 (ComboBox) 是文本框 (TextBox) 和列表框 (ListBox) 的组合。用户可以在文本框中输

入文本，也可以从列表框中选择列表项。

### 1. 组合框的属性

#### (1) List 属性

组合框的 List 属性同列表框的 List 属性一样，也是用于返回或设置控件列表部分的项目。设置方法也基本相同。

例如，设置 List 属性为“长春、四平、吉林……”信息，其效果如图 9.39 所示。

**注意：**同列表框的 List 属性设置一样，在每输入完一项后，按〈Ctrl+Enter〉组合键换行。

使用代码设置 List 属性的语句如下：

```
Combo1.List(0) = "长春"
Combo1.List(1) = "四平"
Combo1.List(2) = "吉林"
.....
```

#### (2) Style 属性

Style 属性用于指定组合框的显示类型和行为，在程序运行时是只读的。下面对其属性值进行介绍。

- ☒ Style 属性值为 0 时，表示显示类型为下拉组合框。此时组合框包括一个下拉式列表和一个文本框，可以在文本框内输入或在下拉列表中选择。
- ☒ Style 属性值为 1 时，表示为简单组合框。此时包括一个文本框和一个不能下拉的列表，可以在文本框中输入或从列表中选择。简单组合框包括编辑和列表部分。按默认规定，简单组合框的大小调整是在没有任何列表显示的状态。增加 Height 属性值可显示列表的更多部分。
- ☒ Style 属性值为 2 时，表示为下拉式列表。这种样式允许从下拉式列表中选择，而不能在文本框内输入。

设置 Style 属性的显示效果如图 9.40 所示。



图 9.39 设置 List 属性效果

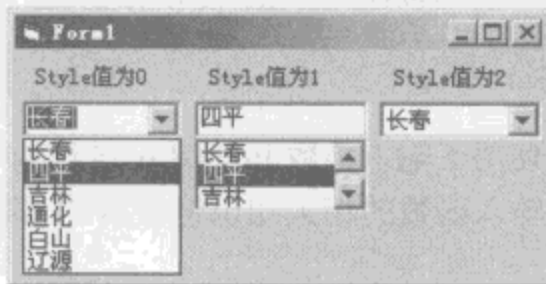


图 9.40 设置 Style 属性效果

#### (3) ListIndex 属性

组合框的 ListIndex 属性同列表框的 ListIndex 属性一样，也是用于返回或设置控件中当前选择项目的索引。

语法：

```
object.ListIndex [= index]
```

object: 对象表达式。

index: 数值表达式，指定当前项目的索引。

**例 9.9** 本实例实现当程序运行时,单击“添加项目”按钮,弹出输入对话框,输入要添加到组合框中的内容单击“确定”按钮,即可添加到 Combo1 中;选择要移除的项目,单击“移除项目”按钮,即可移除所选项目。程序运行效果如图 9.41 所示。(实例位置:光盘\TM\9\9)

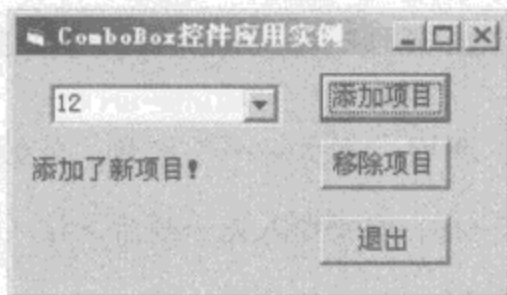


图 9.41 ComboBox 控件应用实例

程序代码如下:

```
Dim s As String                                '定义变量
Private Sub Command1_Click()
    s = InputBox("输入添加项, 单击确定按钮", "添加项目")    '将输入内容赋给 s
    If s <> "" Then                                           '如果输入不为空
        Combo1.AddItem s                                     '将输入内容添加到组合框中
        Label1.Caption = "添加了新项目!"                    '设置标签显示内容
        Combo1.ListIndex = 0                                '将组合框中第一项显示在组合框的文本框中
    End If
End Sub
Private Sub Command2_Click()
    If Combo1.ListIndex = -1 Then                            '如果选项为空
        MsgBox "请先选择要删除的内容!"                     '提示信息
        Exit Sub                                             '退出此过程
    End If
    Combo1.RemoveItem Combo1.ListIndex                       '移除选中项目
    Label1.Caption = "移除了项目!"                           '设置标签显示内容
    Combo1.Text = ""                                         '组合框文本框为空
End Sub
```

这里使用了输入函数 InputBox 进行输入数据,输入函数用来弹出一个输入对话框,并将输入的内容返回。输入函数的相关知识将在第 11 章详细介绍。

## 9.7 滚动条

教学录像: 光盘\TM\9\9\滚动条.exe

滚动条分为水平滚动条 (HScrollBar 控件) 和竖直滚动条 (VScrollBar 控件) 两种。这两种滚动条为在应用程序或控件中的水平或垂直滚动提供了便利,在信息量很大而控件又没有自动添加滚动条功能时,可以利用滚动条来提供便利的定位。

水平滚动条和竖直滚动条只是方向不同,它们的结构和操作是一样的,也就是具有相同的属性、事件和方法。

## 1. 滚动条的属性

## (1) Max 和 Min 属性

滚动条的值以整数形式表示,对于每个滚动条可指定在-32768~32767之间的一个整数。垂直滚动条的最上端代表其最小值(Min),最下端代表最大值(Max);水平滚动条的最左端代表最小值,最右端代表最大值。默认设置为:Max属性值为32767,Min属性值为0。

**注意:** 设置时应该尽量将Max属性值设置得比Min属性值大,并且Min属性值必须总是大于或等于0。如果Max设置得比Min的值小,那么最大值将被分别设置为水平或垂直滚动条的最左或最上位置处。

## (2) Value 属性

Value属性指定当前滑块在滚动条上的位置,其值在Min和Max属性值之间,语法:

```
object.Value[=value]
```

object: 对象表达式。

value: 该值指定控件的状态,设置介于-32768与32767之间的值以定位滚动框。

## (3) LargeChange 属性

LargeChange属性表示当用户单击滚动条的空白处时,滑块移动的增量值。

## (4) SmallChange 属性

SmallChange属性表示当用户单击滚动条两端箭头时,滑块移动的增量值。

**例 9.10** 本实例实现单击滚动条的空白处或两端箭头,使滑块移动,当滑块移动时,图片框(PictureBox 控件)中的背景颜色深浅程度改变。程序运行效果如图 9.42 所示。(实例位置:光盘\TM\sl\9\10)



图 9.42 滚动条属性实例

程序代码如下:

```
Private Sub Form_Load()  
    '设置最大和最小值属性  
    HScroll1.Max = 200: HScroll1.Min = 0  
    HScroll1.LargeChange = 40: HScroll1.SmallChange = 40  
    HScroll1.Value = 0  
    Picture1.BackColor = RGB(255, 255, 255)  
End Sub  
Private Sub HScroll1_Change()  
    '滚动条的 Change 事件  
    Picture1.BackColor = RGB(255 - HScroll1.Value, 255, 255 - HScroll1.Value)  
End Sub
```



这里使用了 RGB 函数来通过设置一个颜色值，返回颜色。

语法：

RGB (red, green, blue)

red: 数值范围从 0~255，表示颜色的红色成分。

green: 数值范围从 0~255，表示颜色的绿色成分。

blue: 数值范围从 0~255，表示颜色的蓝色成分。

RGB(255, 255, 255)表示白色；RGB(0, 0, 0)表示黑色。

## 2. 滚动条的事件

### (1) Change 事件

当滚动条的 Value 值改变时触发 Change 事件，即移动滚动条的滑块或是通过代码改变 Value 值时触发 Change 事件。

### (2) Scroll 事件

当拖动滚动条的滑块时触发 Scroll 事件。

**注意：** Scroll 事件只有在滚动条的滑块上按下鼠标进行拖动时才触发，单击两侧箭头或是滚动条空白处改变滑块位置并不能触发此事件。但是 Scroll 事件能够触发 Change 事件，因为拖动滑块时改变了 Value 值，从而触发 Change 事件。

**例 9.11** 本实例实现在程序运行时，使用鼠标拖动滚动条中的滑块，控制图片框中图片的移动，其实现效果如图 9.43 所示。（实例位置：光盘\TM\9\11）



图 9.43 滚动条事件实例

程序代码如下：

```
Private Sub Form_Resize()  
    On Error Resume Next          '错误处理  
    '除去竖直滚动条后窗体所剩宽度  
    w = Form1.ScaleWidth - VScroll1.Width  
    '除去水平滚动条后窗体所剩高度  
    h = Form1.ScaleHeight - HScroll1.Height  
    VScroll1.Move w, 0, VScroll1.Width, h: HScroll1.Move 0, h, w  
    Picture1.Move 0, 0, w, h     '移动滚动条  
                                '移动图片框并设置大小
```



```

VScroll1.Min = 0: VScroll1.Max = Image1.Height - Picture1.Height '设置竖直滚动条最小值和最大值
HScroll1.Min = 0: HScroll1.Max = Image1.Width - Picture1.Width '设置水平滚动条的最小值和最大值
HScroll1.LargeChange = (Image1.Width - Picture1.Width) / 10 '设置水平滚动条最大增量值
HScroll1.SmallChange = (Image1.Width - Picture1.Width) / 10 '设置水平滚动条最小增量值
VScroll1.LargeChange = (Image1.Height - Picture1.Height) / 10 '设置竖直滚动条最大增量值
VScroll1.SmallChange = (Image1.Height - Picture1.Height) / 10 '设置竖直滚动条最小增量值
End Sub
Private Sub HScroll1_Scroll() '水平滚动条的 Scroll 事件
    Image1.Left = -HScroll1.Value '设置图片位置
End Sub
Private Sub VScroll1_Scroll() '竖直滚动条 Scroll 事件
    Image1.Top = -VScroll1.Value '设置图片位置
End Sub

```

使用滚动条的 Scroll 事件只能在使用鼠标拖动滚动条的滑块时才能移动图片。如果把上个实例中的 Scroll 事件全部换为 Change 事件, 无论是拖动滑块还是单击箭头或是滚动条空白处, 都能实现图片的移动。

## 9.8 Timer 控件

 教学录像: 光盘\TM\lx\9\Timer 控件.exe

一个时钟 (Timer) 控件能够有规律地以一定的时间间隔触发 Timer 事件, 从而每隔一段事件执行一次 Timer 事件下的代码。

### 1. Timer 控件的属性

Interval 属性是时钟控件最重要的属性。它表示执行两次 Timer 事件的时间间隔, 以 ms (0.001s) 为单位, 取值范围为 0~65535ms, 所以最大的时间间隔大约为 1 分 5 秒。

当 Interval 属性值为 0 时表示 Timer 控件无效。如果希望每半秒触发一次 Timer 事件, 将 Interval 属性值设置为 500; 如果希望每一秒执行一次 Timer 事件, 将 Interval 属性值设置为 1000。

程序运行期间, Timer 控件隐藏不显示在窗体上, 通常将时间显示在一个标签中。

### 2. Timer 控件的事件

Timer 控件只有一个 Timer 事件, 在一个 Timer 控件的预定的时间间隔过去之后发生, 该间隔的频率储存于该控件的 Interval 属性中。

**例 9.12** 本实例是一个打砖块的小游戏。在程序运行时, 选择相应的等级, 单击“开始”按钮, 开始游戏, 图片框中的砖块不停移动, 使用鼠标点击砖块得分, 时间限制为 30 秒。程序运行效果如图 9.44 所示。(实例位置: 光盘\TM\sl\9\12)

本程序使用了 3 个 Timer 控件, Timer1 用于控制砖块的显示位置和颜色, Interval 属性设置为 1000; Timer2 用于控制结束时的时间, Interval 属性设置为 30000; Timer3 用于计算并显示剩余时间, Interval 属性设置为 1000。

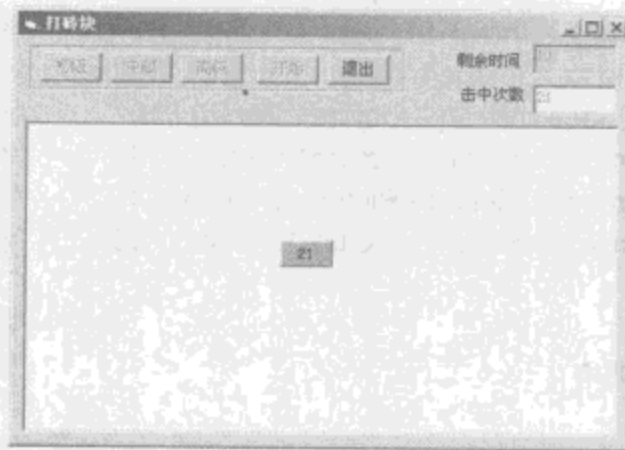


图 9.44 打砖块游戏运行效果

程序主要代码如下：

```

Dim n, s As Integer                                '定义全局变量
Private Sub Command1_Click()
    Timer1.Interval = 1000                          '设置 Timer1 的 Interval 属性值
    Command4.Enabled = True                          '开始按钮有效
End Sub
Private Sub Command2_Click()
    Timer1.Interval = 800                          '设置 Timer1 的 Interval 属性值
    Command4.Enabled = True                          '开始按钮有效
End Sub
Private Sub Command3_Click()
    Timer1.Interval = 700                          '设置 Timer1 的 Interval 属性值
    Command4.Enabled = True                          '开始按钮有效
End Sub
Private Sub Command5_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single) '鼠标按下按钮事件
    n = n + 1                                        '击中次数加 1
    Text1.Text = n                                  '显示在击中次数文本框中
    Command5.Caption = n                            '击中次数显示在砖块上
End Sub
Private Sub Timer1_Timer()
    Randomize                                        '初始化随机变量
    R = Int(Rnd * 256 + 0)                          '把 Rnd 函数生成的随机数赋给变量 R
    G = Int(Rnd * 256 + 0)                          '把 Rnd 函数生成的随机数赋给变量 G
    B = Int(Rnd * 256 + 0)                          '把 Rnd 函数生成的随机数赋给变量 B
    L = Int(Rnd * 5000 + 0)                         '把 Rnd 函数生成的随机数赋给变量 L
    T = Int(Rnd * 2000 + 0)                         '把 Rnd 函数生成的随机数赋给变量 T
    Command5.BackColor = RGB(R, G, B)               '用生成的变量 R、G、B 的值设置砖块的颜色
    Command5.Left = Picture1.Left + L               '用生成的变量 L 的值设置砖块的 left 位置
    Command5.Top = Picture1.Top + T                 '用生成的变量 T 的值设置砖块的 top 位置
End Sub
Private Sub Timer2_Timer()
    Timer1.Enabled = False: Timer3.Enabled = False '设置时间控件无效
    Command5.Visible = False: Command1.Enabled = True '设置按钮有效或无效
    Command2.Enabled = True: Command3.Enabled = True

```

```
End Sub
Private Sub Timer3_Timer()
    s = s + 1                                '使用时间累加
    Label1.Caption = 30 - s                 '将剩余时间显示在标签中
End Sub
```

## 9.9 小结

本章主要介绍了 VB 常用标准控件的基本属性、事件和方法。通过本章的学习，读者可以掌握常用标准控件的使用方法，并能够在应用程序中使用这些控件进行编程。介绍的这些控件是编写应用程序最经常使用的控件。

## 9.10 练习与实践

1. 设计一个简单的调色板。使用 3 个水平滚动条，分别控制红、黄、蓝 3 种颜色，当拖动滚动条中的滑块时，改变窗体背景颜色的对应滚动条的颜色深度。例如，当向左滑动绿色滚动条的滑块时，窗体背景颜色的绿色成分减少。（答案位置：光盘\TM\sl\9\13）

2. 参照实例 9.11，做一个使用滚动条控制图片移动的程序，实现无论是拖动滑块还是单击箭头或是滚动条空白处，都能使图片移动的效果。（答案位置：光盘\TM\sl\9\14）

图1.9

图1.9展示了世界森林资源的分布情况。图中显示了不同地区的森林覆盖率，以及森林资源的利用情况。图中还标注了森林资源的分布密度和森林资源的利用密度。

图1.9

图1.9展示了世界森林资源的分布情况。图中显示了不同地区的森林覆盖率，以及森林资源的利用情况。图中还标注了森林资源的分布密度和森林资源的利用密度。



# 第10章

## 菜单、工具栏和状态栏

(教学录像: 1小时12分钟)

菜单、工具栏和状态栏在 Windows 应用软件中比较常见, 也是在设计应用程序时必不可少的重要元素。如果在自己设计的应用程序中使用菜单、工具栏和状态栏, 一定会使程序看起来更加专业, 在使用上也更加方便和快捷。本章将对菜单、工具栏和状态栏的设计和使用进行详细介绍。

通过阅读本章, 您可以:

- 熟悉菜单的组成
- 掌握菜单编辑器的使用
- 掌握标准菜单的设计
- 掌握弹出式菜单的设计和使用
- 掌握菜单数组的设计和使用
- 掌握工具栏的按钮设计
- 掌握工具栏的代码设计
- 掌握状态栏的设计和使用



## 10.1 菜单概述

 教学录像：光盘\TM\lx\10\菜单概述.exe

在可视化程序设计中菜单是最重要的元素之一。菜单可以方便的显示程序的各项功能，以方便用户的选择，并让用户快速进入到需要的界面中。

### 10.1.1 菜单的组成

在开发程序时，经常利用菜单将程序的各项功能归类，集中存放在程序的菜单中，用户只需利用鼠标单击或者利用键盘上的几个快捷键就可以访问需要的功能。

下面以车辆管理系统中的菜单为例，介绍一下菜单的组成。菜单中包含的界面元素主要有菜单栏、访问键、快捷键、分隔条、选中提示、子菜单提示等，具体的组成如图 10.1 所示。

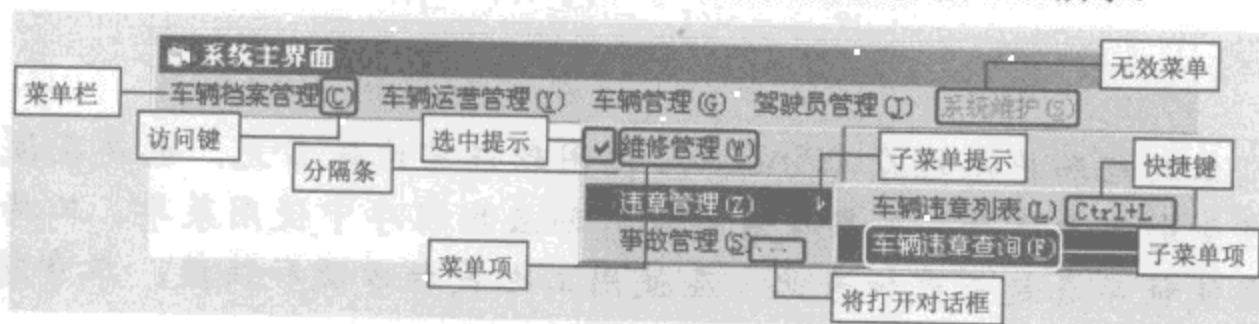


图 10.1 菜单的组成

- ☑ 菜单栏：菜单栏紧跟在标题栏下面，由多个菜单标题组成。
- ☑ 访问键：访问键是为某个菜单项指定的字母键，在显示出有关菜单项以后，按该字母键就可以选中该菜单项。
- ☑ 分隔条：分隔条用于将属于同一类的菜单项分组显示。
- ☑ 选中提示：当某个菜单被选中时，可在菜单项的左边打一个“√”，表示该菜单项被选中；再次选中该菜单项时，选中提示消失。
- ☑ 菜单项：菜单项是菜单或子菜单的组成部分，每个菜单项代表一条命令或一个子菜单项。
- ☑ 子菜单提示：如果某菜单项下面有子菜单，则在该菜单的右侧就会出现一个指向右侧的三角箭头，该箭头即为子菜单提示。
- ☑ 快捷键：为了更快捷地执行命令，可以为每个最底层的菜单项设置一个快捷键。在有快捷键的菜单项中，用户可以在不单击菜单项的情况下，直接利用键盘上的快捷键执行相应的功能。
- ☑ 对话框标识：在菜单项文字的末尾，添加 3 个点，用于标识当用户单击该菜单项时，将打开一个对话框。


### 10.1.2 菜单编辑器

在 VB 中设置菜单非常容易，可以通过 VB 提供的菜单编辑器来设计实现。利用菜单编辑器可以

创建菜单和菜单栏，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。

### 1. 菜单编辑器的调用

在使用菜单编辑器以前首先需要启动它，它的启动方式有下面 4 种形式。

- (1) 选择“工具”/“菜单编辑器”命令。
- (2) 在“标准”工具栏上选择“菜单编辑器”图标.
- (3) 用鼠标右键单击要添加菜单窗体，在弹出的快捷菜单中选择“菜单编辑器”命令。
- (4) 利用快捷键〈Ctrl+E〉来调用“菜单编辑器”。

### 2. 菜单编辑器的组成

利用上面介绍的 4 中方法都可以打开“菜单编辑器”。打开的菜单编辑器如图 10.2 所示。其中包括 3 个区域：菜单属性设置区、菜单编辑区、菜单列表区。

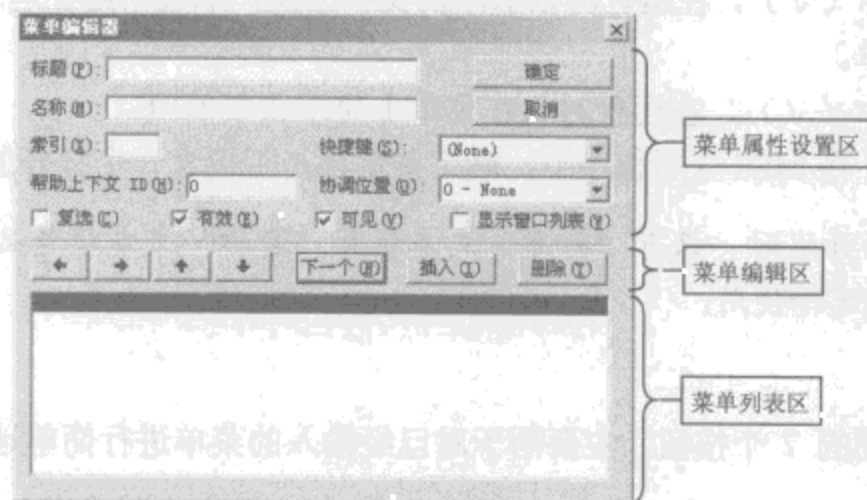


图 10.2 菜单编辑器的组成

### 3. 菜单属性设置区

菜单属性设置区是指在菜单编辑器中分隔条上面的部分，它主要用于设置菜单的相关属性。其主要属性如下：


- ☒ 标题。“标题”文本框用于设置在菜单栏上显示的文本。
  - 调用对话框。如果菜单项想调用一个对话框，在标题文本框的后面应加“...”。
  - 设置访问键。如果想通过键盘来访问菜单，使某一字符成为该菜单项的访问键，可以用“(&+访问字符)”的格式。访问字符应当是菜单标题的第一个字母，除非别的字符更容易记。两个同级菜单项不能用同一个访问字符。在运行时访问字符会自动加上一条下划线，&字符则不见了。
  - 设置分隔条。菜单的分隔条可以将菜单分割成具有独立功能的几个菜单组。在设置时，在“标题”文本框中输入连字符(-)，在显示时，即可显示为分割条的形式。
- ☒ 名称。在“名称”文本框中，设置用来在代码中引用该菜单项的名字。不同菜单中的子菜单可以重名，但是菜单项名称应当唯一。
- ☒ 索引。索引在设置菜单数组的时候使用，用于指定该菜单项在菜单数组中的下标。一般为整

型数值。在设置时，其索引值可以不连续，但是一定要按照递增的顺序填写下标，否则将不被菜单编辑器接受。

- ☒ 快捷键。可以在快捷键组合框中输入快捷键，也可以选取功能键或键的组合来设置快捷键。快捷键将自动出现在菜单上，要删除快捷键应选取列表顶部的（none）。

 注意：在菜单条上的第一级菜单不能设置快捷键。

- ☒ 帮助上下文 ID。指定一个唯一的数值作为帮助文本的标识符，可根据该数值在帮助文件中查找适当的帮助主题。
- ☒ 协调位置。允许选择菜单的 NegotiatePosition 属性。该属性决定当窗体的链接对象或内嵌对象活动而且显示菜单时，是否在菜单栏显示最上层 Menu 控件。
- ☒ 复选。如果选中（√），在初次打开菜单项时，该菜单项的左边显示“√”。在菜单条上的第一级菜单不能使用该属性。
- ☒ 有效。如果选中（√），在运行时以清晰的文字出现；未选中则在运行时以灰色的文字出现，不能使用该菜单。
- ☒ 可见。如果选中（√），在运行时将在菜单上显示该菜单项。
- ☒ 显示窗口列表。在 MDI 应用程序中，确定菜单项是否包含一个打开的 MDI 子窗体列表。

 说明：在实际的程序开发时，只有“标题”文本框和“名称”文本框是必须填写的，其他属性可根据需要自己选择使用。

#### 4. 菜单编辑区

菜单编辑区是指中间的 7 个按钮，主要用于对已经输入的菜单进行简单地编辑操作。下面介绍一下这几个按钮的功能。

- ☒ “右箭头”按钮：每次单击都把选定的菜单向右移一个等级。一共可以创建四个子菜单等级。
- ☒ “左箭头”按钮：每次单击都把选定的菜单向上移一个等级。一共可以创建四个子菜单等级。
- ☒ “上箭头”按钮：每次单击都把选定的菜单项在同级菜单内向上移动一个位置。
- ☒ “下箭头”按钮：每次单击都把选定的菜单项在同级菜单内向下移动一个位置。
- ☒ “下一个”按钮：将选定移动到下一行。
- ☒ “插入”按钮：在列表框的当前选定行上方插入一行。
- ☒ “删除”按钮：删除当前选定行。

#### 5. 菜单列表区

该列表框显示菜单项的分级列表。将子菜单项缩进以指出它们的分级位置或等级。

## 10.2 标准菜单

 教学录像：光盘\TM\lx\10\标准菜单.exe

在程序中，最常用的就是标准菜单，一般应用到程序的主界面中，并放置在窗体的顶部，用于调用程序的所有功能。下面通过几个例子介绍利用菜单编辑器设计菜单的方法。

### 10.2.1 创建最简菜单

在前面已经介绍了菜单编辑器的基本组成。在菜单的属性设置区域中有诸多的属性需要设置，其中，“标题”和“名称”属性是必须要设置的，其他的属性可以采用默认值，或者不进行设置。仅设置了“标题”和“名称”属性的菜单就是最简菜单，

**例 10.1** 下面以客户管理系统中的部分菜单为例，介绍最简菜单的设计过程。创建最简菜单的操作步骤如下：（实例位置：光盘\TM\sl\10\1）

（1）选中需要创建菜单的窗体，启动菜单编辑器。需要注意如果不选中窗体，菜单编辑器将不可用。

（2）在“标题”文本框中输入要显示在菜单上的标题，在“名称”文本框中输入菜单的名称。这里菜单的名称是菜单的标识，用于在编写代码时使用，而标题则用于显示在菜单上。

如这里输入菜单的标题为“客户信息管理”，在顶层菜单将显示“客户信息管理”字样。在“名称”文本框中输入“khxxgl”，用于在代码中使用。

（3）单击“下一个”按钮，设计下一个菜单，下一个菜单为“客户信息管理”的子菜单。需要单击“右箭头”按钮，将该菜单向右移一个等级。

如设计“客户信息添加”菜单，在显示时将显示为“客户信息管理”的子菜单。

（4）重复步骤（2）和步骤（3），直至完成菜单的设计。其设计和显示的效果如图 10.3 所示。

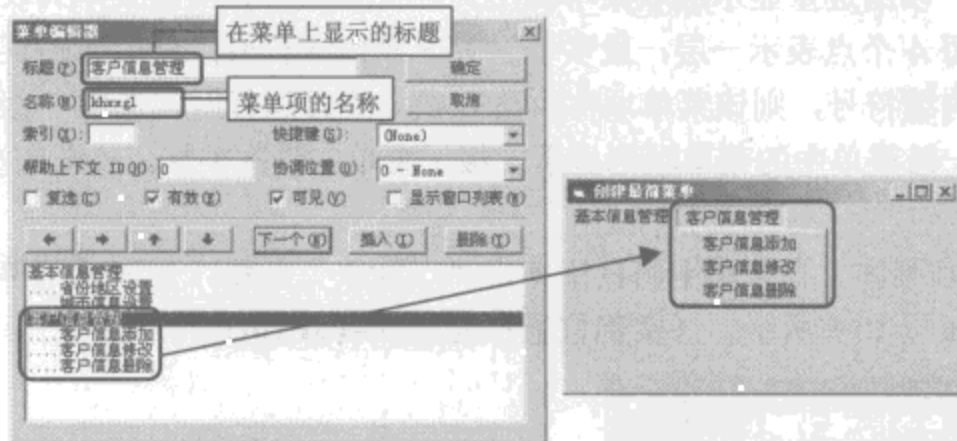


图 10.3 创建最简菜单

**注意：**“标题”属性和“名称”属性必须都设置，缺一不可，否则将不被菜单编辑器接受。

### 10.2.2 设置菜单的快捷键和访问键

快捷键就是用于执行一个命令的功能键或者组合键，如，〈Ctrl+C〉为复制操作，〈Ctrl+V〉为粘贴操作。为菜单设置快捷键，用户就可以直接利用键盘执行菜单的命令。

访问键是指用户按下〈Alt〉键同时又按下的键。如，在一般的 Windows 环境中，〈Alt+F〉用于打开“文件”菜单，这里的〈F〉键即为访问键。

**例 10.2** 创建带快捷键和访问键的菜单。如设置“客户信息管理”菜单的访问键为〈C〉，只需在编辑“客户信息管理”菜单时，在“标题”文本框中输入“客户信息管理(&C)”。这里的“&C”，



即用于设置访问键，在显示时即可显示为 **C** 的形式。为了符合 Windows 操作系统的风格，这里使用 **()** 将访问键括起来。（实例位置：光盘\TM\sl\10\2）

利用菜单编辑器设置快捷键也非常简单，只需选中要设置快捷键的菜单，在“快捷键”下拉列表框中选择需要的快捷键即可。如设置“客户信息删除”菜单的快捷键为 **<Ctrl+D>**，只需在“快捷键”下拉列表框中选择 **Ctrl+D** 项即可。如果不需要，则选择 **None** 项即可。其操作过程和演示效果如图 10.4 所示。

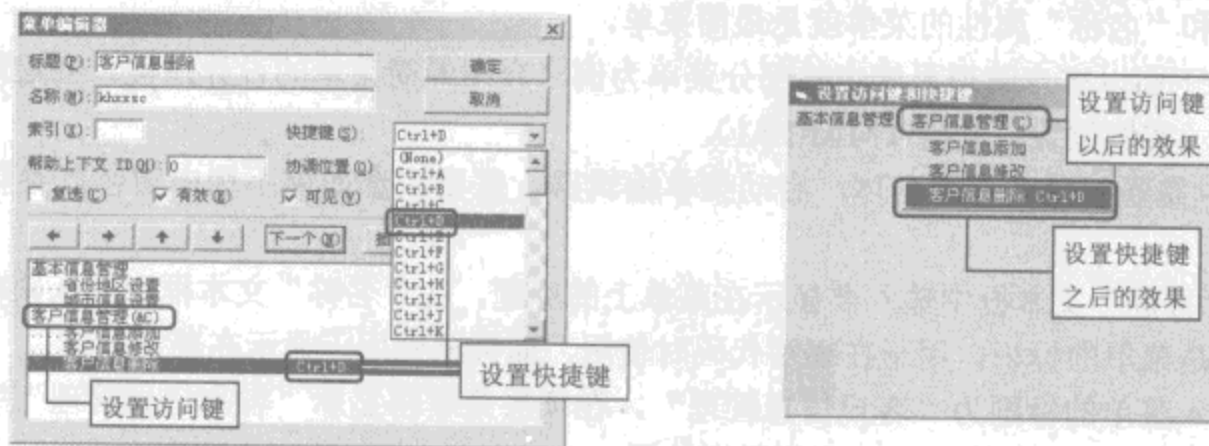


图 10.4 设置快捷键和访问键

### 10.2.3 创建级联菜单

在菜单编辑器中，以缩进量显示级联菜单的形式。在菜单编辑器的菜单列表区中由内缩符号表明菜单项所在的层次，每 4 个点表示一层，最多可以有 5 个内缩符号，最后面的菜单项为第 5 层。如果一个菜单项前面没有内缩符号，则该菜单项称为第 0 级。程序运行时，选取 0 级菜单中的菜单项则显示一级子菜单，选取一级菜单中的菜单项则显示二级子菜单，依次类推。当选到没有子菜单的项目时，将执行菜单事件过程。

**例 10.3 创建级联菜单。**在菜单编辑器中单击“右箭头”按钮，创建子菜单。在设置菜单时最多可以设置 5 级菜单，如图 10.5 所示。（实例位置：光盘\TM\sl\10\3）

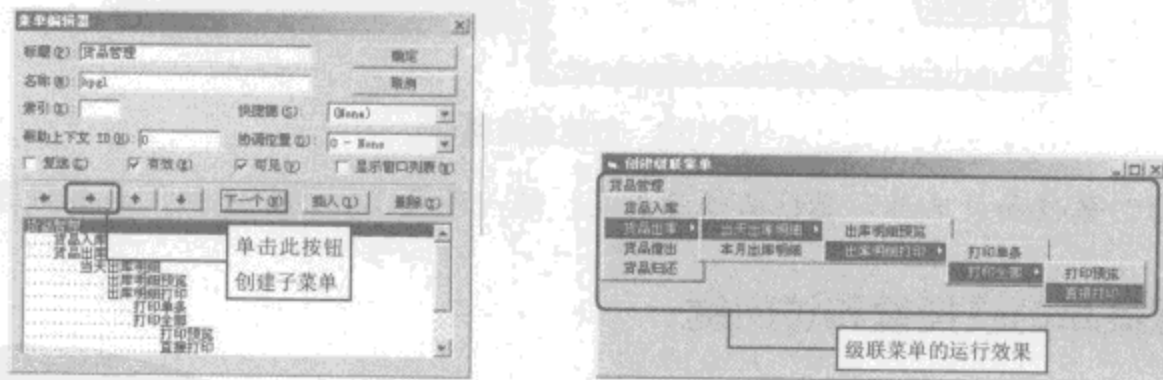


图 10.5 创建级联菜单

### 10.2.4 创建复选菜单

通过复选菜单可以实现在菜单中执行或取消执行某项操作。菜单的复选标记有两个作用：一是表示打开或关闭的条件状态，选取菜单命令可以交替地添加或删除复选标记；二是指示几个模式中哪个



或哪几个在起作用。

**例 10.4** 通过菜单编辑器创建复选菜单。在菜单编辑器中选中需要设置为复选的菜单，例如选中“客户信息删除”，然后勾选“复选”复选框，这样在菜单显示时即为复选的效果，其设置和实现的效果如图 10.6 所示。（实例位置：光盘\TM\sl\10\4）

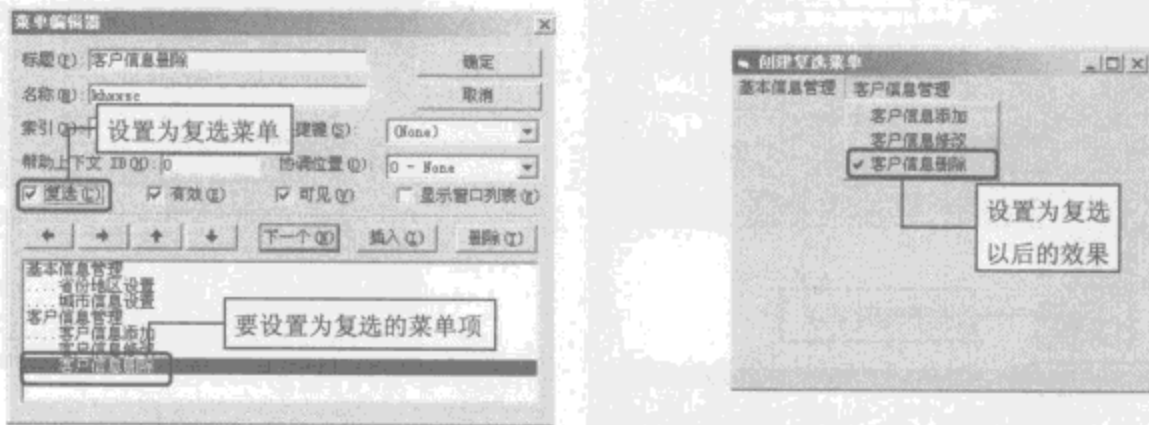


图 10.6 创建复选菜单

### 10.2.5 设置菜单分隔条

在 Windows 的菜单中经常将一些功能相近的菜单放在一组，利用菜单分隔条分开，这样可以使子菜单看起来更加清晰、明了。

**例 10.5** 设置菜单分隔条。如果想利用菜单分隔条将菜单分成几个逻辑的组，则只需在“标题”文本框中输入一个连字符，并在“名称”文本框中输入该菜单的名称，即可设置出菜单分隔条的效果，如图 10.7 所示。（实例位置：光盘\TM\sl\10\5）

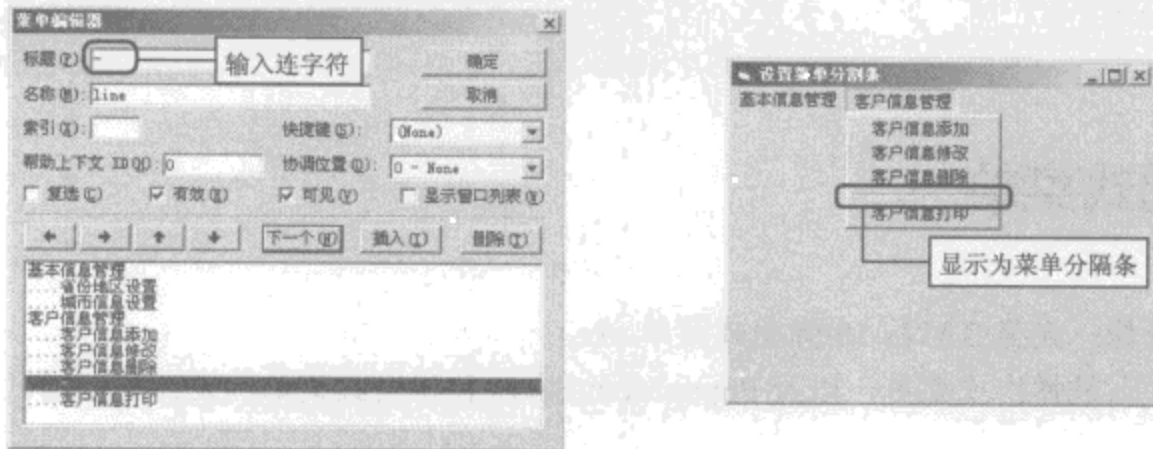


图 10.7 设置菜单分隔条

**注意：**在运行时，菜单的分隔条不能被选中，也不能执行代码。

### 10.2.6 设置菜单无效

有些菜单对于不同权限的操作用户的使用权限是不同的，如系统设置方面的菜单，只有系统管理员才能使用，当普通用户进入到系统中时，这些菜单将被设置为无效。

**例 10.6** 设置菜单无效。在菜单使用时，还有一种状态，即设置菜单无效。利用菜单编辑器设置菜单无效也比较简单：选中需要设置的菜单项，然后，取消“有效”复选框之前的勾即可。例如，设置“客户信息打印”菜单项为无效，如图 10.8 所示。（实例位置：光盘\TM\sl\10\6）

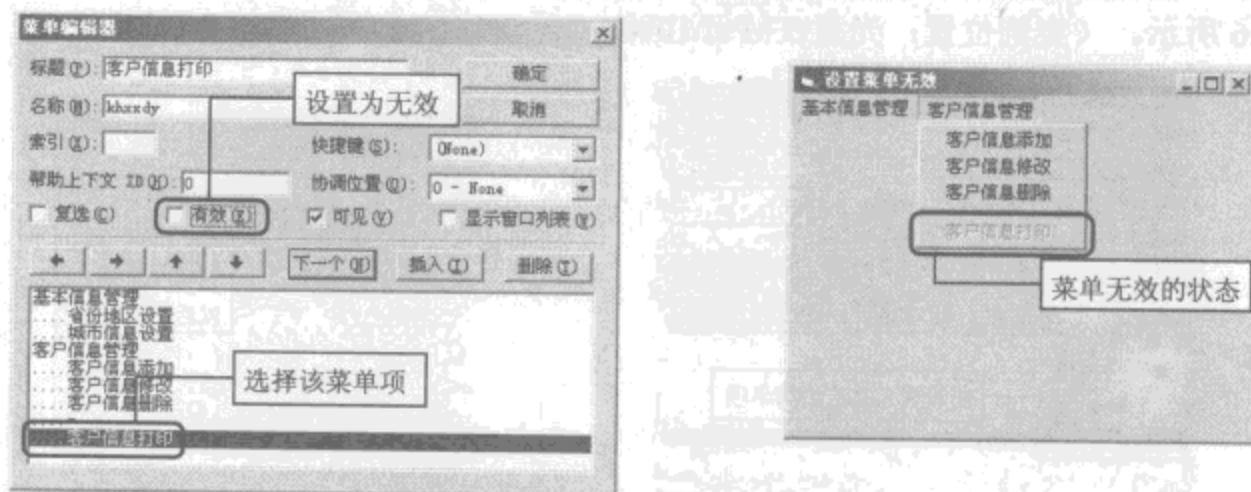


图 10.8 设置菜单无效

### 10.2.7 为菜单事件添加代码

单击菜单所实现的功能是通过执行菜单事件中的程序代码来实现的。程序员在菜单编辑器中定义一个菜单之后，在该菜单的 Click 事件中就可以添加所需要的程序代码，完成相应的功能。例如：在单击“显示好友列表”菜单项之后，调用“好友列表”窗体，同时隐藏本窗体。其相关的程序代码如下：

```
Private Sub showF_Click()  
    frm_HYLB.Show  
    Unload me  
End Sub
```

'显示好友列表菜单项  
'显示好友列表窗体  
'卸载自己

## 10.3 弹出式菜单

教学录像：光盘\TM\lx\10\弹出式菜单.exe

本节介绍什么是弹出式菜单，以及 PopupMenu 方法和弹出式菜单的设计与调用。

### 10.3.1 弹出式菜单概述

弹出式菜单是指在窗体上单击鼠标右键之后弹出的菜单，弹出式菜单也称为浮动菜单。除了不显示 0 级菜单项的标题以外，弹出式菜单的每个菜单项都可以有自己的子菜单。一般来说，弹出式菜单所显示菜单项的位置取决于单击鼠标右键时指针所处的位置。

可使用 PopupMenu 方法显示弹出式菜单。在 Windows 操作系统中激活上下文菜单，关键在于是在何种事件中调用 PopupMenu 方法。

### 10.3.2 PopupMenu 方法

可以使用 PopupMenu 方法调用弹出式菜单。其实在大部分响应事件中都可以激活弹出式菜单，但在通常情况下都是使用鼠标事件来调用 PopupMenu 方法的。


语法：

```
object.PopupMenu menuname, flags, x, y, boldcommand
```

PopupMenu 方法的语法中各参数的说明如表 10.1 所示。

表 10.1 参数说明

参 数	说 明
object	可选的参数。对象表达式，其值为 Form 或者 MDIForm
menuname	必须的参数。指出要显示的弹出式菜单名，指定的菜单项必须至少含有一个子菜单
flags	可选的参数。为一个数值或常数，用以指定弹出式菜单的位置和行为
x	可选的参数。指定显示弹出式菜单的 x 坐标
y	可选的参数。指定显示弹出式菜单的 y 坐标
boldcommand	可选的参数。指定弹出式菜单中的菜单控件的名称，用以显示其黑体正文标题

 说明：x 和 y 坐标定义了弹出式菜单相对于指定窗体显示的位置，可使用 ScaleMode 属性指定 x 和 y 坐标的度量单位。如果没有包括 x 和 y 坐标，则弹出式菜单就显示在鼠标指针的当前位置。

### 10.3.3 弹出式菜单的设计和调用

定义弹出式菜单的方法和定义下拉式菜单的方法一样，任何含有一个或一个以上的子菜单的菜单项都可作为弹出式菜单。弹出式菜单的最高一级菜单项称为顶级菜单项，该顶级菜单的菜单项不会显示出来，这一点与下拉菜单不同。如果弹出式菜单的顶级项是 0 级菜单项，则弹出时仅显示一级以下的菜单项和它们的子菜单项。这个 0 级菜单项必须被定义，因为 0 级菜单项的名字用于激活弹出式菜单。同样道理，可以使用任何一个级别已定义、具有下一级子菜单的菜单项作为弹出式菜单。

如果这个菜单仅在某个位置单击鼠标右键时才弹出，而不需要以下拉菜单的形式显示在屏幕上，则应在设计时使顶级菜单不可见，即取消选中菜单编辑器里的“可见”复选框或在属性窗口设定 Visible 属性为 False。当一个菜单既作为下拉菜单使用，又作为弹出式菜单使用时，激活的弹出式菜单将自动不显示顶级菜单项。

**例 10.7** 利用弹出式菜单设置窗体的背景色。本例中利用菜单编辑器设计菜单，并利用 PopupMenu 方法调用该菜单。（实例位置：光盘\TM\sl\10\7）

利用菜单编辑器设计用于设置窗体背景色的菜单项。设置顶层菜单的标题为“背景色”；名称为 MyMenu，该名称用于在 PopupMenu 方法中使用；设置顶层菜单不可见，即不显示在窗体的顶部。当利用 PopupMenu 方法调用该菜单时，顶层的菜单不可见，仅显示调用菜单的子菜单，如图 10.9 所示。

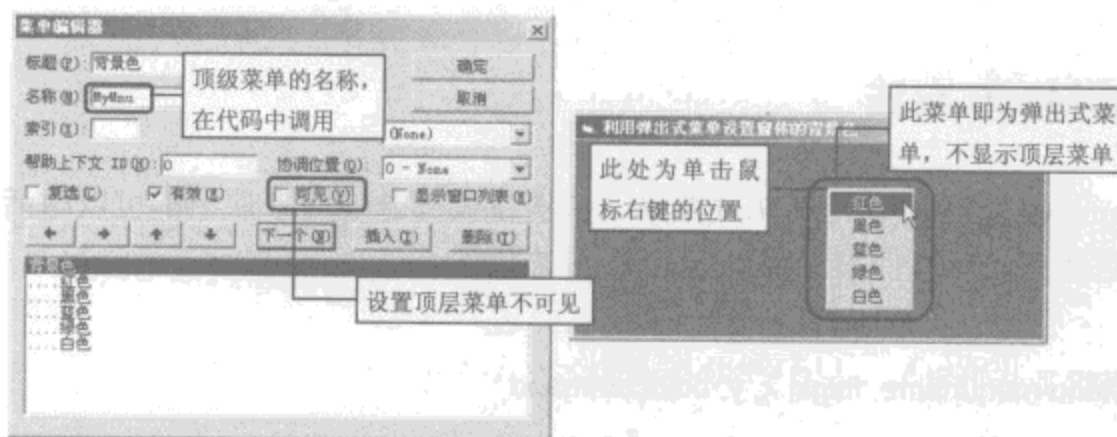


图 10.9 利用弹出式菜单设置窗体背景色

程序代码如下:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 2 Then
        PopupMenu MyMnu
    End If
End Sub
Private Sub MnuRed_Click()
    Form1.BackColor = &HFF&
End Sub
```

'当用户在窗体上单击鼠标右键  
'利用 PopupMenu 方法弹出菜单  
  
'设置窗体背景色为红色的菜单命令  
'设置窗体背景色为红色

## 10.4 菜单数组

教学录像: 光盘\TM\lx\10\菜单数组.exe

在设计应用程序菜单的时候, 可以将同一组内的菜单设置成菜单数组的形式, 这样不仅方便管理, 而且可以简化大量的程序代码。下面介绍创建菜单数组和为菜单数组编写代码。

### 10.4.1 创建菜单数组

每个菜单数组元素都用唯一索引值来标记, 该值通过菜单编辑器上的“索引”文本框中设置。当一个数组元素识别一个事件时, VB 将 Index 属性值作为一个附加参数传递给事件过程。事件过程必须包含判断 Index 属性值的代码, 从而确定正在使用哪个菜单项, 进而执行相应的命令操作。

**例 10.8 创建菜单数组。**下面以客户管理系统中的部分菜单为例, 介绍如何创建菜单数组。在“菜单编辑器”中创建菜单数组的步骤如下: (实例位置: 光盘\TM\lx\10\8)

(1) 打开“菜单编辑器”, 创建一个菜单项, 设置“标题”和“名称”后, 在“索引”文本框中将数组的第一个元素的索引设置为 0。

如, 设置“区域信息设置”菜单项的“名称”为 Menu1, 索引值为 0。

(2) 在与上一步中创建的菜单的同一级上, 创建第 2 个菜单项。将第 2 个元素的“名称”设置为与第 1 个元素相同的名称, 即 Menu1。并把其“索引”设置为 1, 并设置菜单的标题。

(3) 重复步骤 (2), 依次创建第 3 个、第 4 个菜单项, 依次类推。但要保证所创建菜单项的索



引值不相同，且为递增的形式。设计完成后的形式如图 10.10 所示。

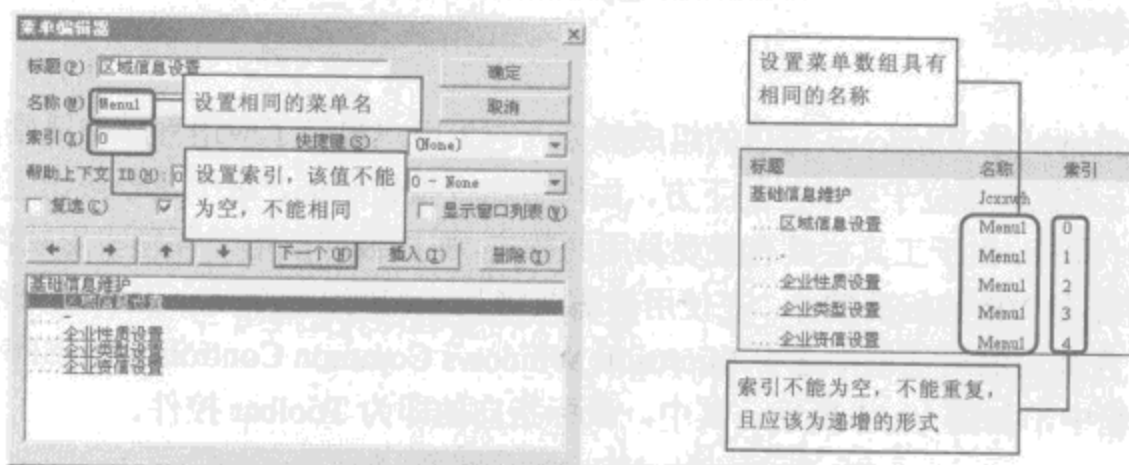


图 10.10 创建菜单数组

**注意：**菜单数组中的各元素必须存在于同一级别中，同时在菜单控件列表框中必须是连续的。而且，如果菜单数组中使用了分隔线，要把它也作为菜单数组中的一个元素。

#### 10.4.2 为菜单数组编写代码

因为菜单数组的名称都是相同的，和一般的控件数组一样，菜单数组的事件也是写在一个事件中，利用 Index 属性值进行区别。在实际的应用中利用 Select Case 语句块来判断触发的是哪个菜单项，并执行对应的 Case 语句后面的代码。上面介绍的“基础信息维护”的子菜单的单击事件代码如下：

```
Private Sub Menu1_Click(Index As Integer)
    Select Case Index
    Case 0
        Load Frm_Jcxxwh_Qysz
        Frm_Jcxxwh_Qysz.Show 1
    Case 2
        Load Frm_Jcxxwh_Qyxz
        Frm_Jcxxwh_Qyxz.Show 1
    Case 3
        Load Frm_Jcxxwh_Qylx
        Frm_Jcxxwh_Qylx.Show 1
    Case 4
        Load Frm_Jcxxwh_Qyzx
        Frm_Jcxxwh_Qyzx.Show 1
    End Select
End Sub
```

'基础信息维护  
'利用 Index 值确定菜单项  
'区域信息设置  
'加载区域设置窗体  
'显示区域设置窗体  
'企业性质设置  
'加载企业性质设置窗体  
'显示企业性质设置窗体  
'企业类型设置  
'加载企业类型设置窗体  
'显示企业类型设置窗体  
'企业资信级别设置  
'加载企业资信级别设置窗体  
'显示企业资信级别设置窗体

### 10.5 工具栏设计

**教学录像：**光盘\TMlx\10\工具栏设计.exe

本节介绍工具栏的概念、最简单工具栏的创建，以及为工具栏添加图片、分组、添加下拉菜单等。



### 10.5.1 工具栏概述

工具栏 (Toolbar) 是 Windows 窗口的组成部分, 它为用户提供了应用程序中最常用的菜单命令的快速访问方式。工具栏通常位于菜单栏的下方, 由许多命令按钮组成, 每个命令按钮上都有一个代表某一项操作功能的小图标。由于工具栏具有直观易用的特点, 所以被广泛用于各种实用软件的主界面当中。

Toolbar 控件不是 VB 的标准控件, 在使用前需要将其添加到工具箱中。具体的方法: 选择“工程”/“部件”命令, 在弹出的对话框中选中 Microsoft Windows Common Controls 6.0 (SP6) 项, 即可添加一组控件到工具箱中, 如图 10.11 所示。其中, 鼠标指向的即为 Toolbar 控件。

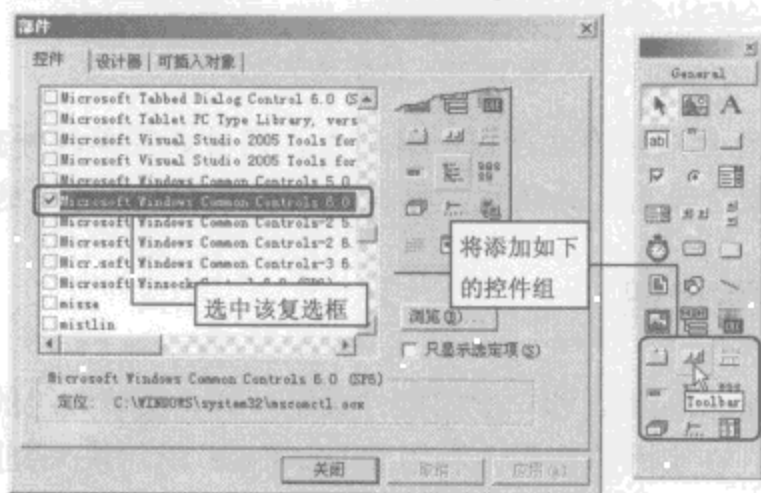


图 10.11 添加 Toolbar 控件

### 10.5.2 利用 Toolbar 控件创建最简工具栏

在工具栏中一般包括文字和图片, 或者仅是图片。仅仅显示文字形式的工具栏, 称为最简工具栏, 因为其设计最简单, 只需设置工具栏控件的按钮文字即可。下面介绍如何设计最简工具栏。

**例 10.9** 创建最简工具栏。创建最简工具栏的步骤如下: (实例位置: 光盘\TM\sl\10\9)

- (1) 添加 Toolbar 控件到工具箱, 添加一个 Toolbar 控件到窗体上。
- (2) 用鼠标右键单击 Toolbar 控件, 在弹出的快捷菜单中选择“属性”命令, 即可弹出“属性页”对话框。在该对话框中选择“按钮”选项卡。
- (3) 单击“插入按钮”按钮, 插入一个按钮, 自动生成“索引”值。在“标题”文本框中输入“新建”, 该标题将显示在工具栏的第一个按钮上。
- (4) 重复步骤 (3), 直到创建完成所有的按钮。在创建工程中, 可以调整所选择的按钮, 当发现有不需要的按钮时, 可以通过单击“删除按钮”按钮将其删除。

创建最简工具栏的过程和实现效果如图 10.12 所示。

### 10.5.3 为工具栏按钮添加图片

在一般的工具栏按钮中, 都是在按钮中添加一个图片, 利用这个图片表达该按钮所执行的功能。

下面介绍如何为工具栏按钮添加图片。

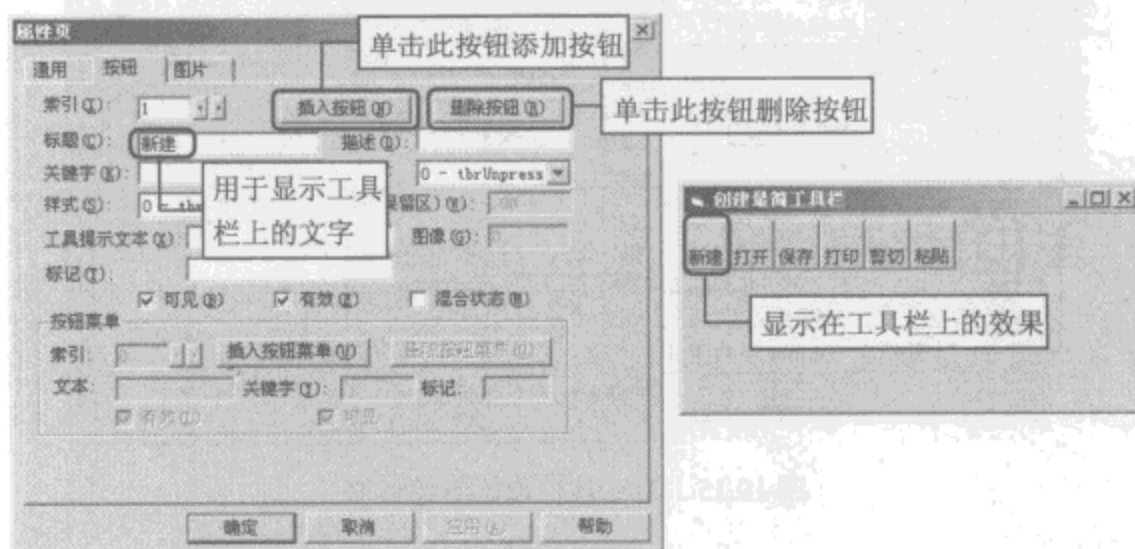


图 10.12 创建最简工具栏

**例 10.10** 为工具栏按钮添加图片。下面以设计效果为如图 10.13 所示的工具栏为例，介绍为工具栏添加图片的步骤。（实例位置：光盘\TM\10\10）

(1) 添加一个 Toolbar 控件和一个 ImageList 控件到窗体上，ImageList 控件与 Toolbar 控件属于一个控件组，在第 13 章中将其进行详细介绍。

(2) 向 ImageList 控件中添加图片，并设置图片的关键字。

(3) 用鼠标右键单击 Toolbar 控件，在弹出的快捷菜单中选择“属性”命令，将弹出“属性页”对话框，选择“通用”选项卡。

(4) 在“图像列表”下拉列表框中选择需要连接的 ImageList 控件，这里为 imlToolbarIcons，如图 10.14 所示。

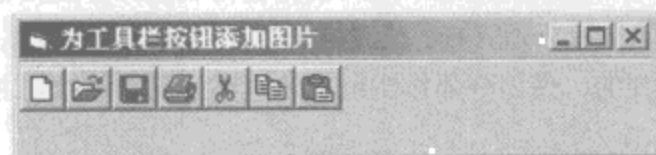


图 10.13 工具栏添加图片的效果

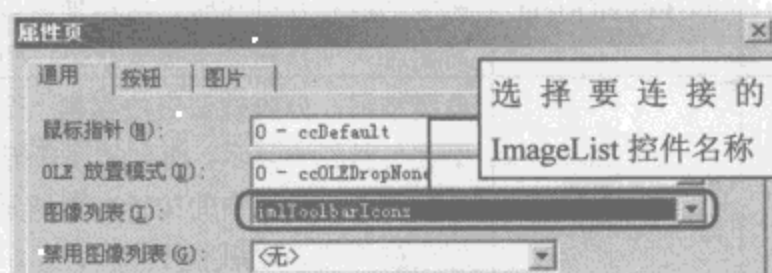


图 10.14 连接 ImageList 控件

(5) 选择“按钮”选项卡，向 Toolbar 控件中添加按钮，因为在工具栏按钮上不显示文字，因此在“标题”文本框中不输入文字。

由于要显示图片，因此需要在“图像”文本框中输入要显示的图片的关键字。如，在 Toolbar 控件中显示 ImageList 控件中的第一个图片，该图片在 ImageList 控件中的关键字为 New，因此需要在 Toolbar 控件“属性页”对话框的“图像”文本框中输入关键字 New，单击“应用”按钮，即可在该按钮上显示出对应的图片。如图 10.15 所示。

(6) 重复步骤 (5) 直至图片全部添加完成。

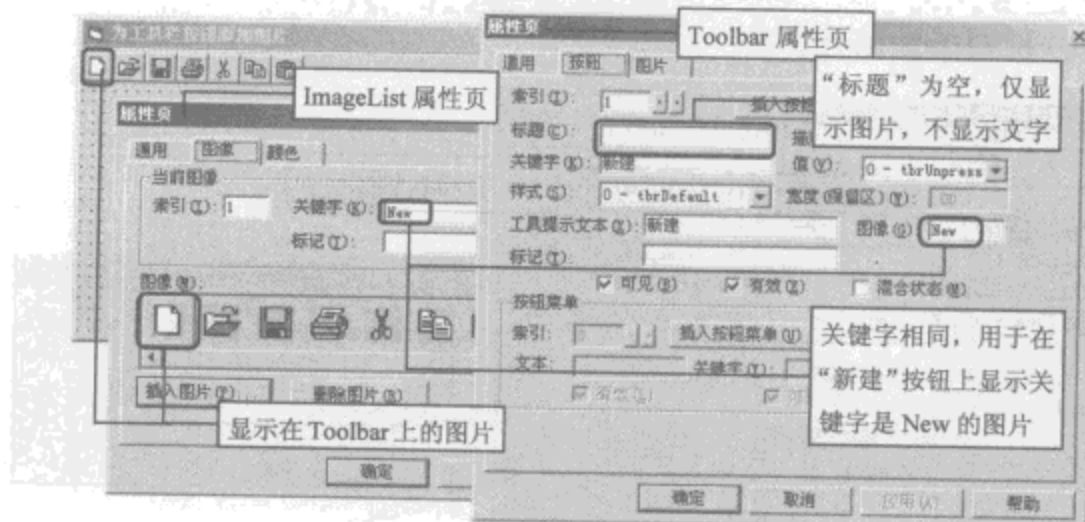


图 10.15 为工具栏按钮添加图片

### 10.5.4 为工具栏按钮设置分组

将一类功能的按钮划分为一组，可以方便用户的操作。其设置方法也比较简单，只需通过设置 Toolbar 控件的按钮样式即可。在设置工具栏按钮样式时，应用到了 Toolbar 控件的 Button 对象的 Style 属性，该属性的设置值如表 10.2 所示。

表 10.2 Toolbar 控件的 Button 对象的 Style 属性设置值

值	常数	描述
0	tbrDefault	一般按钮。如果按钮代表的功能不依赖于其他功能，可以选择它
1	tbrCheck	开关按钮。当按钮具有开关类型时，可以使用该样式
2	tbrButtonGroup	编组按钮。该按钮的功能是将按钮进行分组，属于同一组的编组按钮相邻排列。当一组按钮的功能相互排斥时，可以使用该样式。编组按钮同时也是开关按钮，即同一组的按钮中只允许一个按钮处于按下状态，但所有按钮可能同时处于抬起状态
3	tbrSeparator	分隔按钮。分隔按钮只是创建一个宽度为 8 个像素的按钮，此外没有任何功能。分隔按钮不在工具栏中显示，而只是用来把它左右的按钮分隔开来，或者用来封闭 ButtonGroup 样式的按钮。工具栏中的按钮本来是无间隔排列的，使用分隔按钮可以让同类或同组的按钮并列排放而与邻近组分开
4	tbrPlaceholder	占位按钮。占位按钮在工具栏中占据一定的位置，也不在工具栏中显示。占位按钮是唯一支持宽度 (Width) 属性的按钮
5	tbrdropdown	下拉按钮。单击它可以下拉一个菜单

该属性的设置也可以通过在“属性页”对话框中选择“按钮”选项卡，在样式列表框中选择相应的属性值来实现。

**例 10.11** 为工具栏按钮设置分组。通过设置 Toolbar 控件的按钮样式来为工具栏按钮设置分组。在“属性页”对话框中，选择“按钮”选项卡，设置“样式”下拉列表框中的样式为 3，即可实现分隔按钮的效果。其中，由于工具栏控件的样式不同（有标准工具栏 tbrStandard 和扁平工具栏 tbrFlat 两种），其分隔按钮的样式也不同，效果如图 10.16 所示。（实例位置：光盘\TM\sl\10\11）

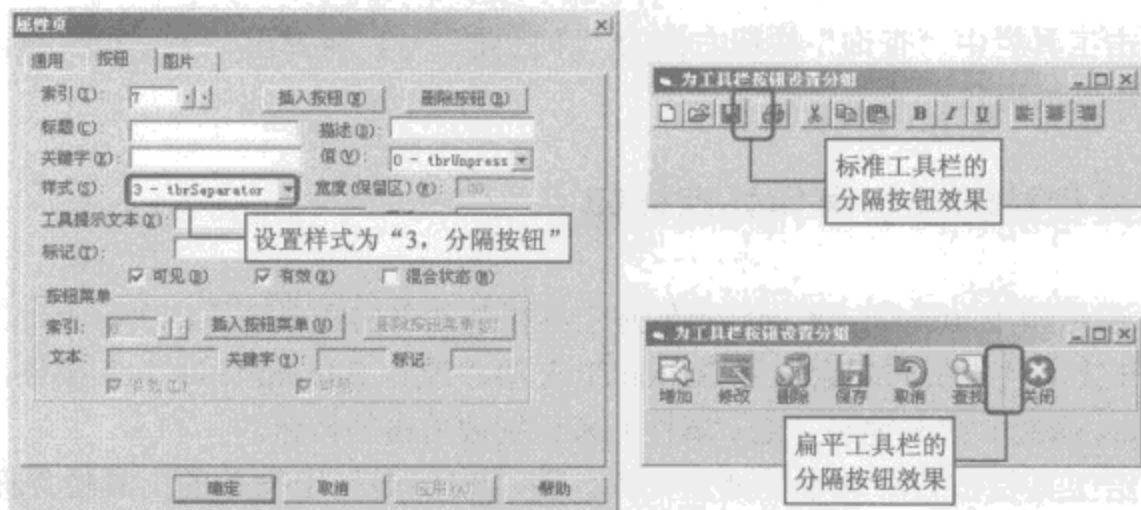


图 10.16 为工具栏按钮设置分组

### 10.5.5 为工具栏添加下拉菜单

在使用工具栏的时候，还会遇到另一种形式的工具栏按钮，即下拉按钮。下拉按钮可以将一类按钮都归为下拉菜单的形式，以改善将多个按钮都放置在工具栏上而导致工具栏杂乱无章的后果。

**例 10.12** 为工具栏添加下拉菜单。在工具栏中添加下拉菜单的方法很简单，不用编写任何代码，只需在 Toolbar 控件的“属性页”对话框中进行设置即可，实现的具体方法如下。（实例位置：光盘 \TM\sl\10\12）

(1) 鼠标右击 Toolbar 控件，在弹出的快捷菜单中选择“属性”命令，弹出“属性页”对话框。选择“按钮”选项卡。

(2) 单击选项卡中“索引”旁边的箭头，将索引移动到要添加下拉菜单的工具栏按钮上。设置“样式”下拉列表框中的样式为“5、下拉按钮”。

(3) 单击“按钮菜单”区域中的“插入按钮菜单”按钮，在“按钮菜单”区域将自动生成索引，输入按钮菜单的“文本”和“关键字”。

(4) 重复步骤 (3)，直到添加完所需要的菜单为止，单击“确定”按钮完成下拉菜单的创建。创建及演示的效果如图 10.17 所示。

**说明：**可以通过单击“删除按钮菜单”按钮删除已经创建的下拉菜单中的菜单项。也可以在下拉菜单中设置分隔条，设置的方法与在菜单编辑器中设置的方法类似。

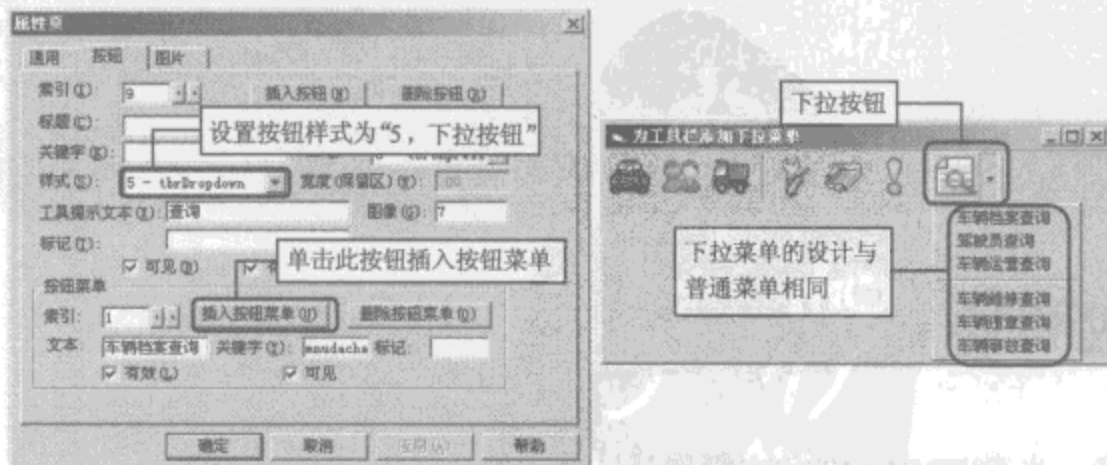


图 10.17 为工具栏添加下拉菜单



运行时，单击工具栏中“查询”按钮右侧的“下三角”号，将弹出一个下拉菜单，单击其中的菜单项，可以执行相关的操作。

### 10.5.6 给工具栏按钮添加事件处理代码

ButtonClick 事件和 ButtonMenuClick 事件是工具栏最常用的两个事件。实际上，工具栏上的按钮是控件数组，单击工具栏上的按钮会发生 ButtonClick 事件或者 ButtonMenuClick 事件，其中主要利用数组的索引（Index 属性）或关键字（Key 属性）来识别被单击的按钮。

例 10.13 给工具栏按钮添加事件处理代码。图 10.18 是一个简单的工具栏界面。其中应用到 Toolbar 控件的 ButtonClick 事件。在程序运行时，单击 Toolbar 上的按钮，利用 Select Case 语句块判断按钮的关键字（Key），进而判断单击的是那个按钮，来实现相应的功能。（实例位置：光盘\TM\10\13）

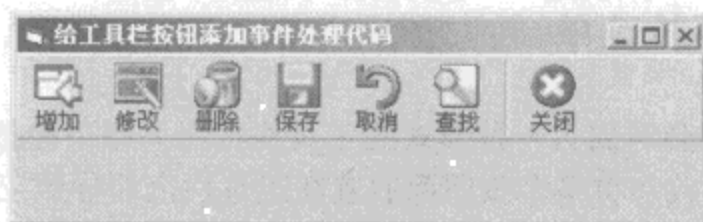


图 10.18 给工具栏按钮添加事件处理代码

程序代码如下：

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MS ComctlLib.Button)
    Select Case Button.Key
        Case "add"
            '执行添加操作
        Case "modify"
            '执行修改操作
        Case "delete"
            '执行删除操作
        Case "save"
            '执行保存操作
        Case "cancel"
            '执行取消操作
        Case "find"
            '执行查找操作
        Case "close"
            '执行关闭操作
    End Select
End Sub
```

## 10.6 状态栏设计

 教学录像：光盘\TM\10\状态栏设计.exe

本节介绍什么是状态栏，以及在状态栏中显示日期、时间、操作员信息和鼠标位置。



### 10.6.1 状态栏概述

StatusBar 控件提供窗体。该窗体通常位于父窗体的底部，通过这一窗体，应用程序能显示各种状态数据。StatusBar 最多能被分成 16 个 Panel 对象，这些对象包含在 Panels 集合中。

该控件是 ActiveX 控件，在使用该控件前需要先将其添加到工具箱中。选择“工程”/“部件”命令，在弹出的对话框中选中 Microsoft Windows Common Controls 6.0 (SP6) 项，即可将一组控件添加到工具箱中，其中，图 10.19 中鼠标所指的即为 StatusBar 控件。状态栏一般用来提示系统信息和用户的提示，如系统日期、软件版本、键盘的状态等。

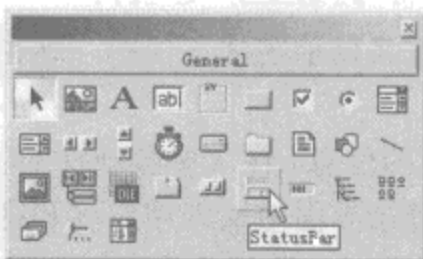


图 10.19 工具箱中的 StatusBar 控件

### 10.6.2 在状态栏中显示日期、时间

在状态栏中显示系统当前的日期、时间是状态栏控件比较常见的使用方式。下面通过例子来介绍其实现步骤。

**例 10.14** 在状态栏中显示日期、时间。一般有两种方法可以实现，一种是通过在“属性页”中设置，另一种则是通过代码进行设置。如，要实现在第 1 个窗格中显示日期，在第 2 个窗格中显示时间。（实例位置：光盘\TM\sl\10\14）

#### 1. 通过“属性页”对话框设置

将 StatusBar 控件添加到窗体上，用鼠标右键单击该控件，在弹出的快捷菜单中选择“属性”命令，即可弹出“属性页”对话框。选择“窗格”选项卡，默认会自动创建一个窗格，设置第 1 个窗格的“样式”为 6-sbrDate，显示当前系统的日期，其设置和显示效果如图 10.20 所示。

单击“插入窗格”按钮，插入一个窗格，设置第 2 个窗格的“样式”为 5-sbrTime，用于显示时间。

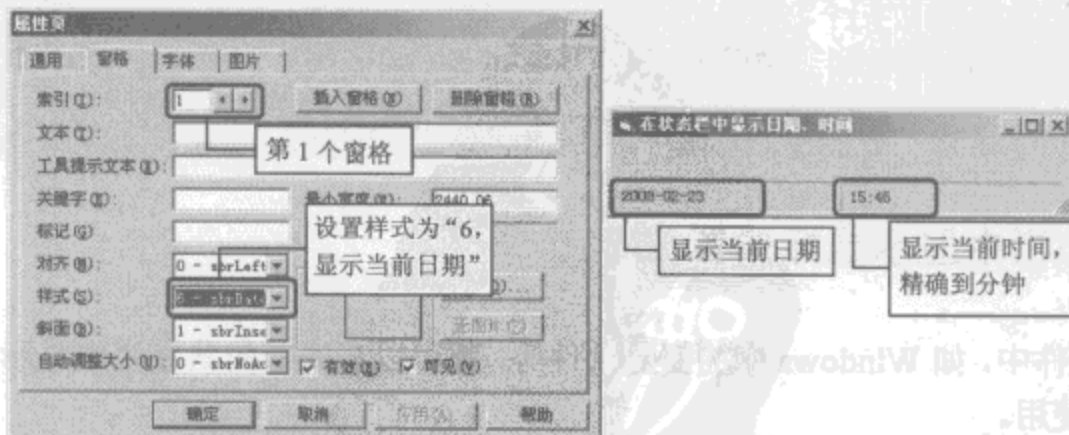


图 10.20 在状态栏中显示日期、时间

## 2. 通过程序代码设置

另一种方法是通程序代码来设置。在窗体中加入一个 Timer 控件，设置 Interval 属性为 60，然后添加如下代码：

```
Private Sub Timer1_Timer()  
    StatusBar1.Panels(1).Text = Format(Date, "YYYY-MM-DD")  
    StatusBar1.Panels(2).Text = Format(Now, "hh:mm")  
End Sub
```

'显示系统时间、日期  
'在第 1 个窗格中显示日期  
'在第 2 个窗格中显示时间

### 10.6.3 在状态栏中显示操作员信息

在很多应用软件中都将操作员的姓名显示在状态栏中，这也是在状态栏使用时应用比较广泛的一种方法。

**例 10.15** 在状态栏中显示操作员的信息。在大多数软件的状态栏中，都具有显示系统登录操作员信息的功能。其实现原理为：用户在登录界面中输入用户名和密码，系统将用户名记录，将其赋值给主窗体状态栏的对应窗格，当主窗体显示时，即可在状态栏中显示出当前操作员的信息。（实例位置：光盘\TM\10\15）

在本实例中，在“用户名”文本框中输入用户名，单击“登录”按钮，进入到“在状态栏中显示操作员信息”窗体中，在状态栏中即可显示出当前操作员的信息，如图 10.21 所示。

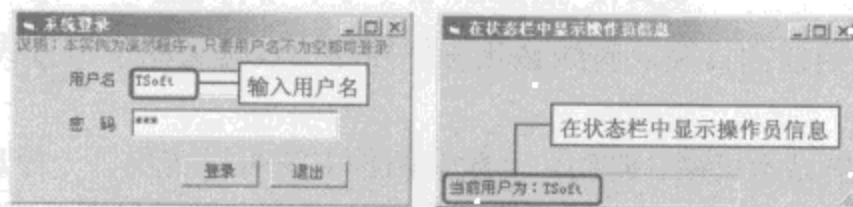


图 10.21 在状态栏中显示操作员信息

程序代码如下：

```
Private Sub Command1_Click()  
    If Text1.Text <> "" Then  
        Form2.StatusBar1.Panels(1).Text = "当前用户为: " & Text1.Text  
        Form2.Show  
        Unload Me  
    Else  
        MsgBox "请输入用户名!", vbCritical, "信息提示"  
    End If  
End Sub
```

'如果用户名不为空  
'将用户名赋值到状态栏中  
'显示窗体 2  
'关闭登录窗体  
'如果用户名为空  
'输出提示信息

### 10.6.4 在状态栏中显示鼠标位置

在一些绘图软件中，如 Windows 中的画图软件等，都可以在底部的状态栏中显示当前鼠标的位置，以方便用户的使用。

**例 10.16** 在状态栏中显示鼠标位置。利用窗体的 Mouse Move 事件，可以获取鼠标在当前位置的

坐标, 将其赋值给状态栏的窗格, 就可以实现在状态栏中显示鼠标位置的效果, 如图 10.22 所示。(实例位置: 光盘\TM\sl\10\16)



图 10.22 在状态栏中显示鼠标的位置

程序代码如下:

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    StatusBar1.Panels(1).Text = " 当前鼠标的位置: " & X & ", " & Y
End Sub
```

'显示鼠标位置

## 10.7 小结

本章介绍了菜单、工具栏和状态栏的设计和使用, 读者可以使用它们来强化界面的设计。将本章的内容和标准控件以及 ActiveX 控件结合起来就可以设计出满足大多数用户需要的应用程序界面。读者在学习过程中结合本章中实例, 融会贯通, 就能轻松设计实现自己的菜单、工具栏和状态栏。

## 10.8 练习与实践

1. 设计一个生产管理系统的菜单, 其主要设置参数如图 10.23 所示。(答案位置: 光盘\TM\sl\10\17)

标题	名称	索引	标题	名称	索引
基础信息管理	Jcxx		产品库存管理	Kcgl	
... 产品基础信息	menu1	0	... 产品完工入库	menu4	0
... 物料基本信息	menu1	1	... 产品入库查询	menu4	1
...	menu1	2	...	menu4	2
... 设备状态设置	menu1	3	... 物料入库	menu4	3
... 设备类型设置	menu1	4	... 物料入库查询	menu4	4
计划信息管理	Jhgl		...	menu4	5
... 生产计划单管理	menu2	0	... 生产领料	menu4	6
... 生产计划单查询	menu2	1	... 生产领料查询	menu4	7
...	menu2	2	...	menu4	8
... 物料需求计划	menu2	3	... 物料库存信息查询	menu4	9
生产设备管理	Sbgl		系统维护	Xtwh	
... 生产设备档案	menu3	0	... 密码修改	menu5	0
... 生产设备查询	menu3	1	... 操作权限设置	menu5	1
...	menu3	2	...	menu5	2
... 生产设备报废	menu3	3	... 数据备份与恢复	menu5	3
... 生产设备报废查询	menu3	4	帮助信息	Bzxx	
...	menu3	5	... 关于	menu6	0
... 生产设备维修	menu3	6	...	menu6	1
... 生产设备维修查询	menu3	7	... 帮助	menu6	2

图 10.23 生产管理系统菜单

2. 设计一个工具栏。设计完成的效果如图 10.24 所示, 当用户单击某个按钮时, 弹出单击了某个按钮的提示对话框。其中, 需要的图片到光盘对应路径下查找。(答案位置: 光盘\TM\sl\10\18)



图 10.24 设计工具栏

3. 设计一个状态栏。显示效果如图 10.25 所示, 显示网址、操作员和日期时间信息。(答案位置: 光盘\TM\sl\10\19)

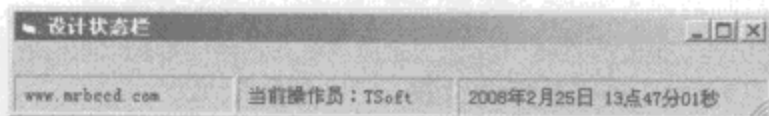


图 10.25 设计状态栏

在本例中, 我们设计了一个简单的工具栏, 用于实现一些基本的操作, 如增加、修改、删除、保存、取消、查找等。在设计过程中, 我们使用了 Windows 资源库中的图标, 并为其添加了相应的提示信息。此外, 我们还设计了一个状态栏, 用于显示当前的网址、操作员和日期时间信息。通过本例, 我们可以了解到如何设计自定义的工具栏和状态栏, 以及如何为它们添加功能。

答案已给出 8.01

(答案位置: 光盘\TM\sl\10\19) 显示效果如图 10.25 所示, 显示网址、操作员和日期时间信息。



图 10.25 设计状态栏

# 第11章

## 对话框

(教学录像: 41 分钟)

对话框是程序与用户进行交互的主要途径，在应用程序中会经常用到，如输入对话框、打开和保存对话框、消息对话框等。这些对话框既可以输入信息又可以显示信息，在应用程序中扮演着非常重要的角色。本章将介绍对话框的相关内容，结合具体实例进行讲解，使读者能够更容易理解和学习。

通过阅读本章，您可以：

- » 学会如何打开输入对话框
- » 掌握输入对话框的使用方法
- » 学会如何调用消息对话框
- » 掌握如何对消息对话框进行设置
- » 掌握消息对话框的返回值
- » 学会如何调用“打开”对话框
- » 学会如何调用“另存为”对话框
- » 学会如何调用“颜色”对话框
- » 学会如何调用“字体”对话框
- » 学会如何调用“打印”对话框
- » 学会如何调用“帮助”对话框



## 11.1 输入对话框 (InputBox)

 教学录像：光盘\TM\lx\11\设计输入对话框.exe

输入对话框返回一个输入值，用于输入数据或查找数据。在 VB 编程中可以使用 InputBox 函数弹出一个输入对话框，InputBox 函数的语法格式如下：

InputBox([\$](提示 [, 标题], 默认值[, x 坐标, y 坐标])

InputBox 函数语法中参数及其说明如表 11.1 所示。

表 11.1 InputBox 函数参数说明

参 数	说 明
[\$]	当该参数存在时，返回的是字符型数据；当该参数不存在时，返回的是变体型数据
提示	一个字符串表达式，用于提示用户输入的信息内容。该参数可以显示单行文字，也可以显示多行文字，但必须在行文字的末尾加上回车符 Chr(13) 和换行符 Chr(10) 或使用 vbCrLf 语句换行
[标题]	一个字符串表达式，该参数用于设置输入对话框标题栏中的标题。该参数是可选项，省略时，将使用工程名的标题
[默认值]	可选项，用来在输入对话框的输入文本框中显示一个默认值
[x 坐标, y 坐标]	表示对话框（左上角）在屏幕上出现的位置。如果省略此参数，则对话框出现在屏幕的中央

 注意：使用 InputBox 函数时应注意以下几点：

(1) 在默认情况下，InputBox 函数返回字符串型的值。如果要返回数值型数据，须将返回值使用 Val 函数转换为数值型（其他字段类型与此相同）。如果声明了返回值的变量类型则可不进行类型转换。

(2) 在使用输入对话框输入数据后，单击“确定”按钮（或按〈Enter〉键），返回输入值；单击“取消”按钮（或按〈Esc〉键），返回一个空字符串。

例 11.1 本实例实现通过输入对话框输入信息，将输入的信息显示在窗体上。程序运行效果如图 11.1 所示。（实例位置：光盘\TM\sl\11\1）

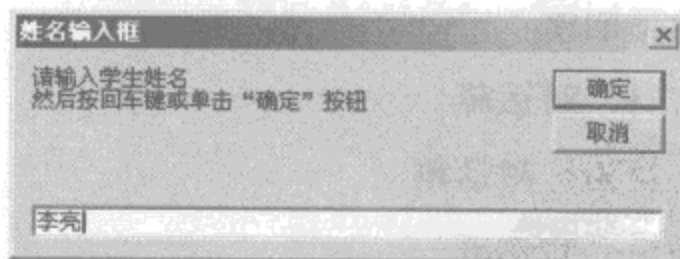


图 11.1 使用输入对话框

程序代码如下：

```
Dim str As String           '定义字符串变量
Dim stu As String          '定义字符串变量
Private Sub Command1_Click()
```

```

str = "请输入学生姓名" + vbCrLf + "然后按回车键或单击“确定”按钮"
stu = InputBox(str, "姓名输入框", , 2000, 3000)
Print stu
End Sub

```

'设置提示内容  
'返回输入值

'打印输入值

## 11.2 消息对话框 (MsgBox)

 教学录像：光盘\TM\lx\11\设计消息对话框.exe

消息对话框主要用于显示提示信息，等待用户单击按钮，并返回一个值，告诉应用程序用户单击的是哪个按钮并执行了什么操作。例如，当用户关闭应用程序时会弹出一个“是否确定退出程序”的提示对话框，包括“是”和“否”两个按钮供用户选择，然后根据用户的选择确定后面的操作。

消息框是使用 MsgBox 函数进行调用的，该函数的格式如下：

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


MsgBox 函数语句的参数说明如表 11.2 所示。

表 11.2 MsgBox 函数参数说明

参 数	说 明
prompt	必需的参数。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 个字符，由所用字符的宽度决定。如果 prompt 的内容超过一行，则可以在每一行之间用回车符 (Chr(13))、换行符 (Chr(10)) 或是回车与换行符的组合 (Chr(13)&Chr(10)) 将各行分隔开来
buttons	可选的参数。数值表达式是值的总和，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，则 buttons 的默认值为 0
title	可选的参数。在对话框标题栏中显示的字符串表达式。如果省略 title，则将应用程序名放在标题栏中
helpfile	可选的参数。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，则必须提供 context
context	可选的参数。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。如果提供了 context，则必须提供 helpfile


其中 buttons 参数的设置值如表 11.3 所示。

表 11.3 buttons 参数设置值

常 数	值	说 明
vbOKOnly	0	在对话框中只显示“确定”按钮
vbOKCancel	1	在对话框中显示“确定”和“取消”两个按钮
vbAbortRetryIgnore	2	在对话框中显示“终止 (A)”、“重试 (R)”和“忽略 (I)”3 个按钮
vbYesNoCancel	3	在对话框中显示“是 (Y)”、“否 (N)”和“取消”按钮
VbYesNo	4	在对话框中显示“是 (Y)”和“否 (N)”两个按钮
vbRetryCancel	5	在对话框中显示“重试 (R)”和“取消”两个按钮
vbCritical	16	在对话框中显示严重错误图标  并伴有声音

续表

常 数	值	说 明
vbQuestion	32	在对话框中显示询问图标 ? 并伴有声音
vbExclamation	48	在对话框中显示警告图标 ! 并伴有声音
vbInformation	64	在对话框中显示消息图标 i 并伴有声音
vbDefaultButton1	0	第 1 个按钮是默认值
vbDefaultButton2	256	第 2 个按钮是默认值
vbDefaultButton3	512	第 3 个按钮是默认值
vbDefaultButton4	768	第 4 个按钮是默认值
vbApplicationModal	0	应用程序强制返回; 应用程序一直被挂起, 直到用户对消息框作出响应才继续工作
VbSystemModal	4096	系统强制返回; 全部应用程序都被挂起, 直到用户对消息框作出响应才继续工作
vbMsgBoxHelpButton	16384	将 Help 按钮添加到消息框
vbMsgBoxSetForeground	65536	指定消息框窗口作为前景窗口
VbMsgBoxRight	524288	文本为右对齐
vbMsgBoxRtlReading	1048576	指定文本应在希伯来和阿拉伯语系统中的从右到左显示

 说明: 第 1 组值 (0~5) 描述了对话框中显示的按钮的类型与数目; 第 2 组值 (16、32、48 和 64) 描述了图标的样式; 第 3 组值 (0、256 和 512) 说明哪一个按钮是默认值; 而第 4 组值 (0 和 4096) 则决定消息框的强制返回性。将这些数字相加以生成 buttons 参数值的时候, 每组值只能取用一个数值。例如  $1+48+0=49$ , 表示在消息框中显示“确定”和“取消”两个按钮, 显示“!”图标, 默认按钮为第一个按钮, 即“确定”按钮。也可以使用常数值相加的样式表示 buttons 参数值, 例如, vbOKCancel+vbQuestion 表示在消息框中显示“确定”和“取消”两个按钮并显示“?”图标。

在弹出的消息框中选择相应的按钮后, 系统将根据选择的按钮返回一个值给程序, 然后根据这个值选择下面的操作, 函数返回值如表 11.4 所示。

表 11.4 MsgBox 函数返回值

操 作	返 回 值	常 数
选择“确定”按钮	1	vbOK
选择“取消”按钮	2	vbCancel
选择“终止”按钮	3	vbAbort
选择“重试”按钮	4	vbRetry
选择“忽略”按钮	5	vbIgnore
选择“是”按钮	6	vbYes
选择“否”按钮	7	vbNo

**例 11.2** 本实例通过 MsgBox 函数调用消息对话框。当程序运行时, 单击窗体上的“退出程序”按钮, 提示消息框, 选择“是”按钮退出程序; 选择“否”按钮继续执行程序, 并将返回值显示在窗体上; 选择“取消”按钮, 取消操作, 并将返回值显示在窗体上。提示的消息框如图 11.2 所示。(实例位置: 光盘\TM\sl\11\2)

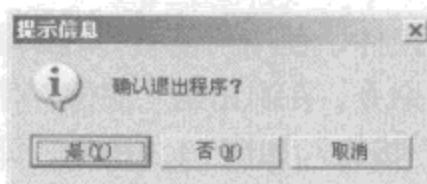


图 11.2 消息对话框

程序代码如下：

```
Dim N1 As Integer                                '定义整型变量存放返回值
Private Sub Command1_Click()
    N1 = MsgBox("确认退出程序?", 67, "提示信息")    '提示消息对话框
    If N1 = vbNo Then                               '如果选择“否”
        Print "选择“否”的返回值为：" & N1         '在窗体上输出返回值
    ElseIf N1 = vbYes Then                          '如果选择“是”
        End                                           '退出程序
    ElseIf N1 = vbCancel Then                       '如果选择“取消”
        MsgBox "操作已经被取消!", 64, "提示信息"    '提示信息
        Print "选择“取消”的返回值为：" & N1        '在窗体上输出返回值
    End If
End Sub
```

## 11.3 公用对话框

教学录像：光盘\TM\11\公用对话框应用.exe

在应用程序中经常会用到一些公用对话框，例如对于打开文件和保存文件、打印和设置字体等操作，使用这些标准对话框可以减轻编程工作量。下面介绍几种公用对话框的使用方法。

### 11.3.1 公用对话框概述

VB 的 CommonDialog 控件提供了一组基于 Windows 的标准对话框界面。用户可以通过此控件在窗体上创建 6 种标准对话框，分别为：“打开”对话框（Open）、“另存为”对话框、“颜色”对话框（Color）、“字体”对话框（Font）、“打印”对话框（Printer）和“帮助”对话框（Help）。

Windows 所提供的几种常见的对话框及其说明如表 11.5 所示。

表 11.5 公用对话框

对 话 框	描 述
“打开”对话框	选取要打开文件的文件名和路径
“另存为”对话框	指定保存信息的文件名和路径，通常用于保存文件
“颜色”对话框	在程序中从标准色中选取或创建要使用的颜色
“字体”对话框	选取基本字体及设置想要的字体属性
“打印”对话框	选取打印机同时设置一些打印参数
“帮助”对话框	与自制或原有的帮助文件取得连接



CommonDialog 控件属于 ActiveX 控件，使用前需要先将其添加到工具箱中，添加方法为：

选择菜单栏中的“工程”/“部件”命令，在弹出的“部件”对话框中选择 Microsoft Common Dialog Control 6.0 (SP3) 选项，单击“确定”按钮，如图 11.3 所示，即可将 CommonDialog 控件添加到工具箱中，添加到工具箱中的 CommonDialog 控件如图 11.4 所示。



图 11.3 部件对话框

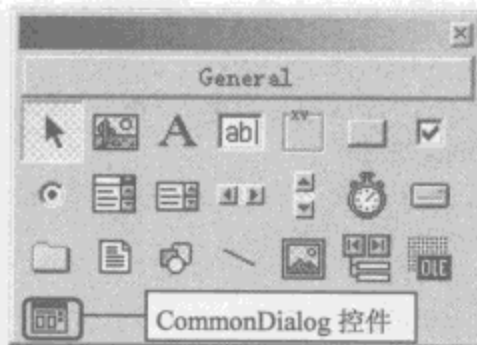


图 11.4 添加到工具箱中的 CommonDialog 控件

**说明：**CommonDialog 控件添加到工具箱中后，就可以像使用标准控件一样将其添加到窗体中进行使用了。在程序运行时，该控件隐藏不显示。使用其 Action 属性或 Show 方法调出所需的对话框，然后通过编程实现相应的对话框功能。

通过设置 CommonDialog 控件的 Action 属性或使用 Show 方法都可调出所需的对话框。下面介绍 Action 属性和 Show 方法。

#### (1) Action 属性

该属性指定打开何种类型的对话框。各属性值对应打开的对话框如下。

- ☒ 0: 无对话框打开。
- ☒ 1: “打开”对话框。
- ☒ 2: “另存为”对话框。
- ☒ 3: “颜色”对话框。
- ☒ 4: “字体”对话框。
- ☒ 5: “打印”对话框。
- ☒ 6: “帮助”对话框。

该属性不能通过属性窗口进行设置，只能在程序中赋值。

#### (2) Show 方法

使用 Show 方法同样可以调用公用对话框。这些方法如下。

- ☒ ShowOpen 方法: “打开”对话框。
- ☒ ShowSave 方法: “另存为”对话框。
- ☒ ShowColor 方法: “颜色”对话框。
- ☒ ShowFont 方法: “字体”对话框。
- ☒ ShowPrinter 方法: “打印”对话框。
- ☒ ShowHelp 方法: “帮助”对话框。



### 11.3.2 “打开”对话框

“打开”对话框是应用程序中经常用到的对话框，用户使用“打开”对话框选择要打开的文件。“打开”对话框如图 11.5 所示。



图 11.5 “打开”对话框

将 CommonDialog 控件的 Action 属性设置为 1 或利用该控件的 ShowOpen 方法，都可调用“打开”对话框。此时的打开对话框不能真正打开一个文件，它仅提供一个打开文件的用户界面，供用户选择要打开的文件，真正的打开文件的工作要在后面通过编程实现。

要用“打开”对话框打开一个文件还要对下面的属性进行设置。

#### (1) FileName 属性

该属性用于设置“文件名”文本框中所显示的文件名，在程序执行时用户用鼠标选中某个文件，选择文件的文件名被显示在“文件名”文本框中，用此文件名为 FileName 属性赋值，FileName 属性将得到一个包含路径名和文件名的字符串。

#### (2) FileTitle 属性

该属性用于返回或设置用户所要打开文件的文件名，它不包含路径。当用户在对话框中选中要打开的文件时，系统自动将该文件名赋值给该属性。FileTitle 是不包含路径的文件名，FileName 是包含路径的文件名。

#### (3) Filter 属性

Filter 也称为过滤器，用于确定“打开”对话框文件列表框中所显示文件的类型。该属性值是由一组元素或用“|”符号隔开的表示文件类型的字符串组成。该属性显示在“打开”对话框的“文件类型”下拉列表框中。例如想要在“文件类型”列表框中显示 3 种文件类型（扩展名为 DOC 的 Word 文件，扩展名为 TXT 的文本文件，所有文件）以供用户选择，Filter 属性应设置为：

```
"文档(*.doc)|*.doc|TextFiles (*.txt) |*.txt|所有文件(*.*)|*.*"
```

下面通过实例介绍“打开”对话框的调用和使用方法。

**例 11.3** 本实例实现通过调用“打开”对话框，获取文件的名称和所在路径。如图 11.6 所示。（实例位置：光盘\TM\sl\11\3）



```

Open CommonDialog1.FileName For Output As #1
Print #1, Text1.Text
Close #1

End If
End Sub

```

'打开文件  
'输入文本  
'关闭文件

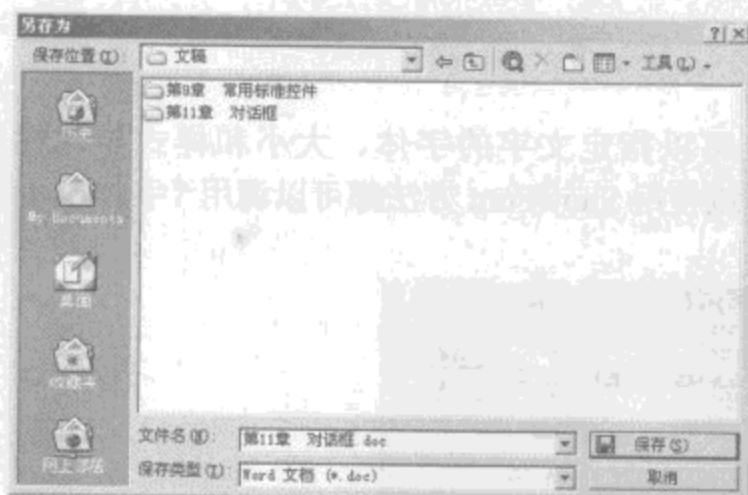


图 11.7 “另存为”对话框

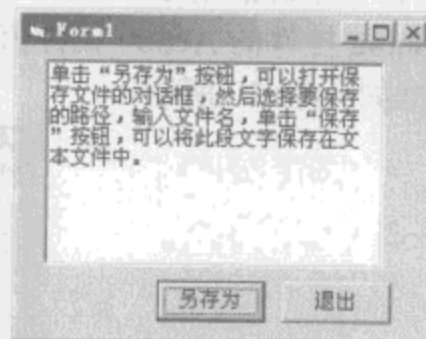


图 11.8 使用“另存为”对话框

另外使用 ShowSave 方法也可以打开“另存为”对话框，代码为：

```
CommonDialog1.ShowSave
```

### 11.3.4 “颜色”对话框

“颜色”对话框供用户选择颜色，在应用软件中经常用到。通过将 CommonDialog 控件的 Action 属性值设置为 3 或使用 ShowColor 方法都可以调用“颜色”对话框。

在“颜色”对话框的调色板中提供了基本颜色，也可以自定义颜色，当用户在调色板中选中某颜色时，该颜色值赋给 Color 属性。“颜色”对话框如图 11.9 所示。

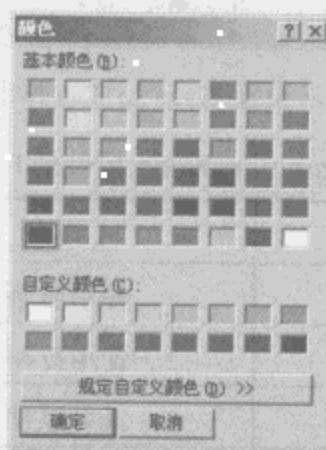


图 11.9 “颜色”对话框

**例 11.5** 本实例实现通过调用“颜色”对话框，设置在文本框中的字体颜色。程序代码如下：（实例位置：光盘\TM\sl\11\5）

```

Private Sub Command1_Click()
    CommonDialog1.Action = 3          '打开颜色对话框
    Text1.ForeColor = CommonDialog1.Color '设置文本框前景颜色

```

End Sub

同样可以使用 ShowColor 方法调用“颜色”对话框，代码可写为：

```
CommonDialog1.ShowSave
```

### 11.3.5 “字体”对话框

“字体”对话框用于供用户选择字体，可以指定文字的字体、大小和样式等属性。通过将 CommonDialog 控件的 Action 属性值设置为 4，或使用 ShowFont 方法都可以调用“字体”对话框。“字体”对话框如图 11.10 所示。

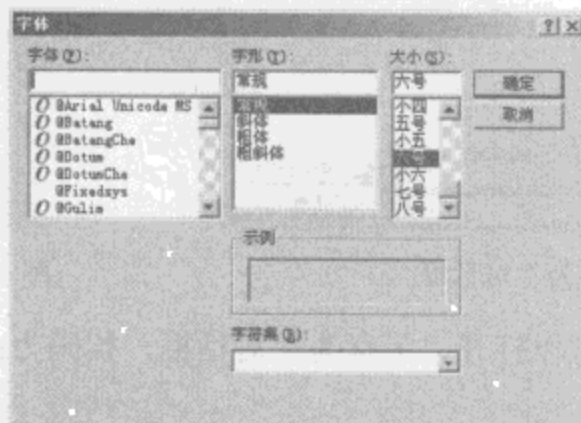


图 11.10 “字体”对话框

**注意：**在调用“字体”对话框前应先设置 Flags 属性，否则会产生不存在字体的错误，提示信息如图 11.11 所示。Flags 属性的设置值如表 11.6 所示。

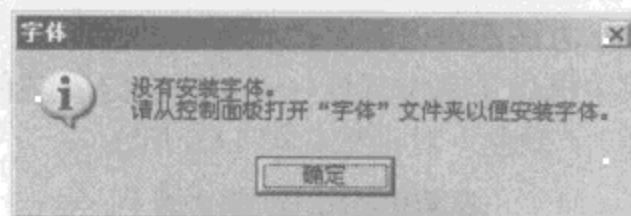


图 11.11 未设置 Flags 属性时弹出的消息框

表 11.6 CommonDialog 控件的 Flags 属性值

常 数	值	说 明
cdlCFScreenFonts	1	使用屏幕字体
cdlCFPrinterFonts	2	使用打印机字体
cdlCFBoth	3	既可以使用屏幕字体，也可以使用打印机字体

**例 11.6** 本实例实现调用“字体”对话框，对文本框中的文字进行字体设置。程序代码如下：（实例位置：光盘\TM\sl\11\6）

```
Private Sub Command1_Click()  
    CommonDialog1.Flags = 3           '设置 Flags 属性值  
    CommonDialog1.Action = 4         '调用“字体”对话框  
    If CommonDialog1.FontName <> "" Then Text1.FontName = CommonDialog1.FontName '为字体赋值
```



```
Text1.FontSize = CommonDialog1.FontSize      '为文本框字号赋值
Text1.FontBold = CommonDialog1.FontBold      '设置是否为粗体
Text1.FontItalic = CommonDialog1.FontItalic   '设置是否为斜体
End Sub
```

同样可以使用 ShowFont 方法调用“字体”对话框，代码为：

```
CommonDialog1.ShowFont
```

另外通过设置 Flags 属性值可以调用带有“下划线”、“删除线”和“颜色”下拉列表框的“字体”对话框。代码可写为：

```
CommonDialog1.Flags = cdICFBoth Or cdICFEffects
```

这时在对文本框字体进行赋值时就要加上这几种属性的赋值，代码如下：

```
Text1.FontStrikethru = CommonDialog1.FontStrikethru  '设置删除线属性
Text1.FontUnderline = CommonDialog1.FontUnderline    '设置下划线属性
Text1.ForeColor = CommonDialog1.Color                '设置字体颜色
```

### 11.3.6 “打印”对话框

在打印文件时要用到“打印”对话框，在“打印”对话框中可以设置打印方式。通过将 CommonDialog 控件的 Action 属性设置为 5 或使用 ShowPrinter 方法都可以调用“打印”对话框。调用的这个打印对话框并不能真正处理打印工作，仅是一个供用户选择打印参数的界面，选择的参数存储在 CommonDialog 控件的各属性中，再通过编程实现打印操作。打印对话框如图 11.12 所示。

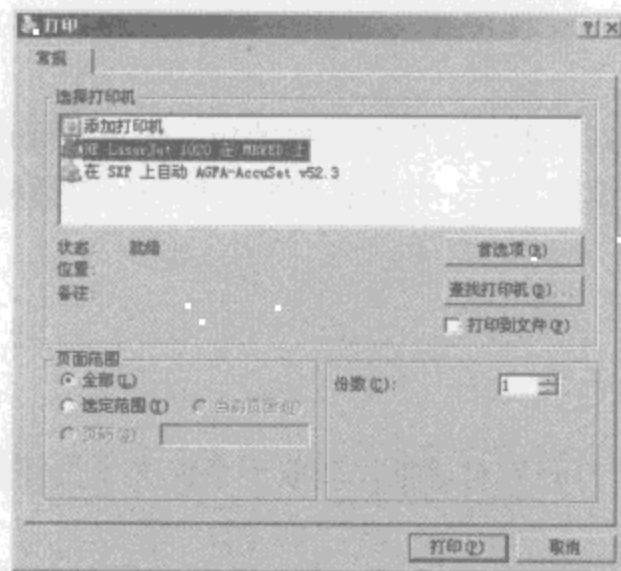


图 11.12 “打印”对话框

注意：“打印”对话框中显示了当前安装的打印机信息，允许配置或重新安装默认打印机。

例 11.7 本实例实现调用“打印”对话框。当程序运行时单击窗体上的“打印”按钮，调出“打印”对话框，程序代码如下：（实例位置：光盘\TM\sl\11\7）

```
Private Sub Command1_Click()
    CommonDialog1.Action = 5      '调用“打印”对话框
End Sub
```




End Sub

使用 ShowPrinter 方法调用“打印”对话框的代码如下：

```
CommonDialog1.ShowPrinter
```

### 11.3.7 “帮助”对话框

通过将 CommonDialog 控件的 Action 属性值设置为 6 或使用 ShowHelp 方法都可以调用“帮助”对话框。“帮助”对话框不能制作应用程序的帮助文件，只能用于提取指定的帮助文件。

 说明：使用 CommonDialog 控件的 ShowHelp 方法调用“帮助”对话框前，应该先通过控件的 HelpFile 属性设置帮助文件 (\*.hlp) 的名称和位置，并将 HelpCommand 属性设置为一个常数，否则将无法调用帮助文件。

例 11.8 本实例实现运行程序后单击窗体上的“帮助”按钮，打开一个指定的帮助文件。程序代码如下：（实例位置：光盘\TM\sl\11\8）

```
Private Sub Command1_Click()  
    CommonDialog1.HelpCommand = cdlHelpContents      '设置帮助类型属性  
    CommonDialog1.HelpFile = "C:\windows\help\notepad.hlp"  '指定要打开的帮助文件  
    CommonDialog1.ShowHelp                             '打开帮助对话框  
End Sub
```

## 11.4 小结

本章主要介绍了输入、输出对话框和公用对话框的相关知识。通过本章的学习，读者可以轻松地了解掌握输入、输出对话框的调用和设置方法，并能够掌握 5 种公用对话框的相关知识以及调用方法。

## 11.5 练习与实践

1. 设计一个小程序，调用“打开”对话框，获取一个纯文本文件的文件名和所在路径。（答案位置：光盘\TM\sl\11\9）
2. 设计一个小程序，用于对窗体上文本框内的文字进行字体和颜色的设置。（答案位置：光盘\TM\sl\11\10）

# 第12章

## OOP 及系统对象

(教学录像: 32 分钟)

程序开发语句从结构化程序设计到面向对象的程序设计是软件设计方法上的一大进步。它解决了程序维护的复杂性和程序代码重用性的难题,大大提高了开发效率。VB 6.0 基于面向对象,可以实现面向对象的几个基本特征:类、对象等。本章将面向对象的编程方法进行介绍。

通过阅读本章,您可以:

- » 了解面向对象编程
- » 了解类的概念
- » 掌握类的创建方法
- » 掌握类的使用方法
- » 了解对象的概念
- » 掌握对象的创建方法
- » 掌握对象的使用方法
- » 了解对象数组
- » 了解对象浏览器的使用方法
- » 了解常用系统对象

## 12.1 面向对象编程

 教学录像：光盘\TM\lx\12\面向对象编程.exe

OOP (Object Oriented Programming) 意为面向对象编程。面向对象编程不同于其他的编程方式，其一方面借鉴了哲学、心理学、生物学的思考方式，另一方面建立在其他编程方式的基础上。

面向对象编程将希望解决的特定问题在经过“面向对象分析”和“面向对象设计”后转换为程序中的类，将问题中的客观物体抽象化或概念化，将程序视为各个类的对象之间的对话。

## 12.2 类

 教学录像：光盘\TM\lx\12\类的创建与使用.exe

本节首先介绍类的概念，然后介绍类的创建，最后通过一个典型的实例介绍类的使用。

### 12.2.1 类的概念

类是面向对象中最重要的概念，面向对象程序设计的所有操作都可以归结为对类的操作。简单地说，“类”是对一组客观对象的抽象，它将该组对象所具有的共同特征（包括结构特征和行为特征）集中起来，以说明该组对象的能力和性质。

例如，我们说的“人类”就是一个类，这个类是对所有人的抽象描述。

### 12.2.2 类的创建

类的创建有两种方法，一种是使用代码的方式创建，另一种是使用类生成器创建。下面将对两种创建方法进行介绍。

类是对象所属类型，对象所拥有的特征是由所属类决定的。类中的字段、属性、方法、事件是该类对象特征的决定性因素。下面将对类中的字段、属性、方法、事件的声明或创建方法进行介绍。

#### 1. 字段与属性

(1) 字段与属性类似，它们的目都是用来保存对象的数据；它们之间的差别在于属性是以过程的方式实现。

字段只是声明的共有变量而已，就如同类中的其他私有变量一样。字段能提供比较好的执行性能，但使用时机制则比较有限。

例如声明类模块中的私有字段 num，代码如下：

```
Private num as Integer
```

(2) 属性则是表现类对象特征的过程。在属性过程中可以执行复杂的操作设置或获取类对象的特征。属性赋值语法如下：

```
[Public | Private | Friend] [Static] Property Let name ([arglist,] value)
[statements]
```

```
[Exit Property]
[statements]
End Property
```

参数如表 12.1 所示。

表 12.1 Property Let 语句参数及说明

参 数	说 明
Public	可选参数。表示所有模块的所有其他过程都可访问该 Property Let 过程。如果在包含 Option Private 的模块中使用，则这个过程在该工程外是不可使用的
Private	可选参数。表示只有在包含其声明的模块的其他过程中可以访问该 Property Let 过程
Friend	可选参数。只能在类模块中使用。表示该 Property Let 过程在整个工程中都是可见的，但对于对象实例的控制者是不可见的
Static	可选参数。表示在调用之间将保留 Property Let 过程的局部变量的值。Static 属性对在该 Property Let 过程外声明的变量不会产生影响，即使过程中也使用了这些变量
name	必要参数。Property Let 过程的名称。遵循标准的变量命名约定，但不能与同一模块中的 Property Get 或 Property Set 过程同名
arglist	可选参数。代表在调用时要传递给 Property Let 过程的参数的变量列表，多个变量则用逗号隔开。Property Let 过程中的每个参数的名称和数据类型必须与 Property Get 过程中的相应参数一致。
value	必要参数。指用于给属性赋值的变量。当调用该过程时，这个参数出现在调用表达式的右边。value 的数据类型必须和相应的 Property Get 过程的返回值类型一致
statements	可选参数。Property Let 过程中执行的任何语句组

其中的 arglist 参数的语法如下：

```
[Optional] [ByVal | ByRef] [ParamArray] varname([ ]) [As type] [= defaultvalue]
```

参数说明如表 12.2 所示。

表 12.2 arglist 参数声明语法的参数及说明如下

参 数	说 明
Optional	可选参数。表示参数不是必需的。如果使用了该选项，则 arglist 中的后续参数都必须是可选的，而且必须都使用 Optional 关键字声明。注意，一个 Property Let 表达式的右边是不可能为 Optional 的
ByVal	可选参数。表示该参数按值传递
ByRef	可选参数。表示该参数按地址传递。ByRef 是 Visual Basic 的默认选项
ParamArray	可选参数。只用于 arglist 的最后一个参数，指明最后这个参数是一个 Variant 元素的 Optional 数组。使用 ParamArray 关键字可以提供任意数目的参数。ParamArray 关键字不能与 ByVal、ByRef 或 Optional 一起使用
varname	必要参数。代表参数变量的名称；遵循标准的变量命名约定
type	可选参数。传递给该过程的参数的数据类型；可以是 Byte、Boolean、Integer、Long、Currency、Single、Double、Decimal（目前尚不支持）、Date、String（只支持变长）、Object 或 Variant。如果参数不是 Optional，则也可以是用户定义类型或对象类型
defaultvalue	可选参数。任何常数或常数表达式。只在 Optional 参数时是合法的。如果类型为 Object，则显式的默认值只能是 Nothing



属性获取语法如下：

```
[Public | Private | Friend] [Static] Property Get name [(arglist)] [As type]
[statements]
[name = expression]
[Exit Property]
[statements]
[name = expression]
End Property
Property Get
```

语法参数如表 12.3 所示。

表 12.3 Property Get 语句参数及说明

参 数	说 明
Public	可选参数。表示所有模块的所有其他过程都可访问 Property Get 过程。如果在包含 Option Private 的模块中使用，则该过程在该工程外是不可使用的
Private	可选参数。表示只有包含其声明的模块的其他过程可以访问该 Property Get 过程
Friend	可选参数。只能在类模块中使用。表示该 Property Get 过程在整个工程中都是可见的，但对于对象实例的控制者是不可见的
Static	可选参数。表示在调用之间保留该 Property Get 过程的局部变量的值。Static 属性对在该 Property Get 过程外声明变量不会产生影响，即使过程中也使用了这些变量
name	必要参数。Property Get 过程的名称。遵循标准的变量命名约定，但不能与同一模块中的 Property Let 或 Property Set 过程同名
arglist	可选参数。代表在调用时要传递给 Property Get 过程的参数的变量列表，多个变量则用逗号隔开。Property Get 过程中的每个参数的名称和数据类型必须与相应 Property Let 过程（如果存在）中的参数一致
type	可选参数。该 Property Get 过程的返回值的数据类型；可以是 Byte、Boolean、Integer、Long、Currency、Single、Double、Decimal（目前尚不支持）、Date、String（除定长）、Object、Variant 或任何用户定义类型。任何类型的数组都不能作为返回值，但包含数组的 Variant 可以作为返回值。PropertyGet 过程的返回值类型必须与相应的 Property Let 过程（如果有）的最后一个（有时是仅有的）参数的数据类型相同，该 Property Let 过程将其右边表达式的值赋给属性
statements	可选参数。Property Get 过程体中所执行的任何语句组
expression	可选参数。Property Get 语句所定义的过程返回的属性值

将对象引用赋给变量或属性的语法如下：

```
Set objectvar = {[New] objectexpression | Nothing}
```

Set 语句的语法参数及说明如表 12.4 所示。

表 12.4 Set 语句语法参数及说明

参 数	说 明
objectvar	必要参数。变量或属性的名称，遵循标准变量命名约定
New	可选参数。通常在声明时使用 New，以便可以隐式创建对象。如果 New 与 Set 一起使用，则将创建该类的一个新实例。如果 objectvar 包含了一个对象引用，则在赋新值时释放该引用。不能使用 New 关键字来创建任何内部数据类型的新实例，也不能创建从属对象



续表

参 数	说 明
objectexpression	必要参数。由对象名, 所声明的相同对象类型的其他变量, 或者返回相同对象类型的函数或方法所组成的表达式
Nothing	可选参数。断绝 objectvar 与任何指定对象的关联。若没有其他变量指向 objectvar 原来所引用的对象, 将其赋为 Nothing 会释放该对象所关联的所有系统及内存资源

例 12.1 创建一个 People 类, 包含姓名 Name、性别 Sex、年龄 Age、配偶 Darling 4 个属性。其中 Name 和 Sex 为字符串类型, Age 为整型, Darling 为对象类型。(实例位置: 光盘\TM\sl\12\1)

运行实例程序, 显示 People 类对象信息, 如图 12.1 所示。程序代码可参见光盘中的源程序。

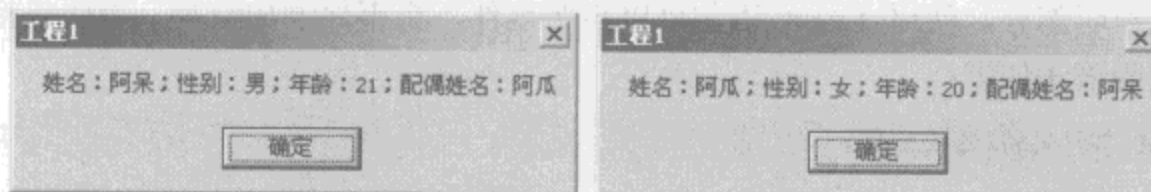


图 12.1 对象属性设置与获取

## 2. 方法

方法是可执行的代码, 用于描述对象的行为。通过创建类的对象后调用对象所属类的方法执行相应的操作。

方法声明与过程的声明相同, 其语法如下:

```
[Public | Private] Sub name [arglist]
```

其参数及说明如表 12.5 所示。

表 12.5 方法声明语句的参数及说明

参 数	说 明
Public	可选参数。用于声明对所有模块中的所有其他过程都可以使用的过程
Private	可选参数。用于声明只能在包含该声明的模块中使用的过程
name	必要参数。任何合法的过程名。注意动态链接库的入口处 (entry points) 区分大小写
arglist	可选参数。代表调用该过程时需要传递的参数的变量表

例 12.2 为 People 类添加方法 Eat(吃饭), 添加方法的主要代码如下:(实例位置: 光盘\TM\sl\12\2)

```
Public Sub Eat(ByVal food As String)
    MsgBox Me.Name & "吃" & food
End Sub
```

创建 People 类对象并调用 Eat 方法的关键代码如下:

```
Dim i As People, j As People
Set i = New People
Set j = New People
...
```

'声明对象变量  
'创建实例  
'创建实例

i.Eat "面条"  
j.Eat "馒头"

'执行方法  
'执行方法

程序运行效果如图 12.2 所示。

### 3. 事件

事件是一个信号,它告知应用程序有重要的事情发生。例如,用户单击窗体时,窗体引发 Click 事件并调用一个处理该事件的过程。在创建类时也可以创建该类对象的事件,下面将对创建事件的方法进行介绍。

#### (1) 声明事件

在类模块的声明部分,使用 Event 语句可以声明事件,将事件添加给类,声明包括事件的名称及使用的参数。声明语法如下:

**[Public] Event 事件名称(形式参数)**

**注意:** 事件不能有返回值、可选参数 (Optional 声明) 和静态数组。传递动态数组参数需要使用 ByRef 声明该形式参数数据类型为 Variant 类型。

#### (2) 事件处理

声明事件仅仅是说明了对象可以引发特定的事件,但要使某事件真正发生,必须使用 RaiseEvent 语句引发,并在窗体中声明一个可以响应事件的窗体级对象变量。在声明对象变量时需要在变量名称前添加 WithEvents 关键字。

RaiseEvent 语句的语法如下:

**RaiseEvent eventname [(argumentlist)]**

其参数如下:

**Eventname:** 必要参数。所引发的事件的名称。

**Argumentlist:** 可选参数。用逗号分隔的变量、数组或者表达式的列表。argumentlist 必须用圆括号括起来。如果没有参数,则圆括号必须被忽略。

#### (3) 事件处理

声明可以响应事件的窗体级对象变量后,在代码窗口中单击窗体左侧的“对象”下拉列表框,选择对象变量名称;单击窗体右侧的“过程”下拉列表框选择对象所响应的事件,自动创建对象事件过程。例如,创建对象变量 myobject 的 MyEvent 的事件过程,如图 12.3 所示。

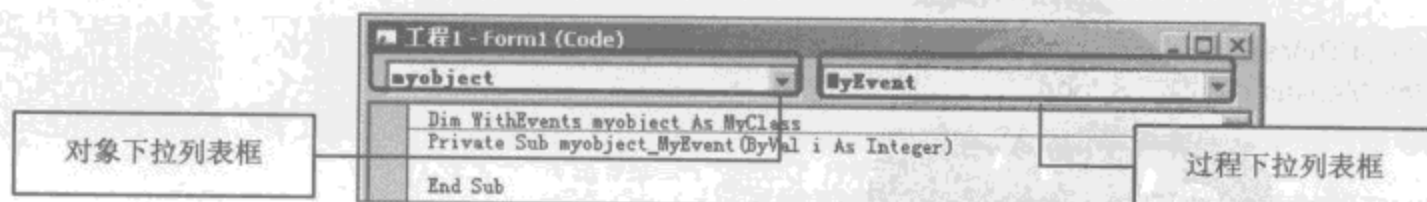


图 12.3 创建对象事件过程

**例 12.3** 创建一事件 MyEvent, 当执行 Method 方法时响应, 并且接收方法中的整型变量 i 的值。类模块 MyClass 内代码: (实例位置: 光盘\TM\sl\12\3)

```
Public Event MyEvent(ByVal i As Integer)      '声明事件
Public Sub Method()                          '创建方法
    Dim i As Integer
    i = 123
    RaiseEvent MyEvent(i)                    '引发事件
End Sub
```

窗体 Form1 内代码:

```
Dim WithEvents obj As MyClass                '声明对象变量
Private Sub Form_Load()
    Set obj = New MyClass                    '创建实例
    obj.Method                               '调用方法
End Sub
Private Sub obj_MyEvent(ByVal i As Integer) '事件过程
    MsgBox i
End Sub
```

上面代码当执行 MyClass 类型对象 obj 中的方法 Method 后响应对象 obj 的 MyEvent 事件, 弹出对话框, 显示事件返回的变量值“123”。

### 12.2.3 类的使用

例 12.4 通过类获得鼠标位置。下面以客户管理系统中使用的获得鼠标在屏幕上的坐标位置的类为例, 介绍在 VB 中如何利用类实现程序的模块化, 如图 12.4 所示的即为客户管理系统中主界面。(实例位置: 光盘\TM\sl\12\4)



图 12.4 利用类模块获得鼠标坐标位置

具体的实现方法如下:

#### (1) 创建类模块

创建一个名为 cCursor 的类模块。

## (2) 编写代码

## ① 声明成员

类模块和标准模块一样，都没有界面而只有代码编辑区域。在类模块中可以将成员声明为 Private，即私有成员，只能在类内部调用；或 Public，即公有成员，可以在类内外部调用。

被声明为 Private（私有成员）的将只在类模块内部是可见的。被声明为 Public（共有成员）不仅在类模块的内部是可见的，对于类模块以外的其他代码也是可见的。在本方案中变量和类型成员的声明如下：

 注意：在类模块中没有使用 Private 或 Public 关键字声明的被默认为 Public。

Private Type POINTAPI	'声明类型
X As Long	'声明 X 为 Long 型
Y As Long	'声明 Y 为 Long 型
End Type	
Private Type RECT	
Left As Long	'声明 Left 为 Long 型
Top As Long	'声明 Top 为 Long 型
Right As Long	'声明 Right 为 Long 型
Bottom As Long	'声明 Bottom 为 Long 型
End Type	
Private CurVisible As Boolean	'声明 Boolean 类型变量

## ② 类模块的属性

在声明属性时，利用 Public 类型的变量将成为类的属性，同 Property Get、Property Let 和 Property Set 语句显式声明的属性一样使用。如，在本程序中 X 坐标的属性设置代码如下：

Public Property Get X() As Long	'获取 X 的值
Dim tmpPoint As POINTAPI	'定义 POINTAPI 类型变量
Call GetCursorPos(tmpPoint)	'调用过程
X = tmpPoint.X	'给 X 赋值
End Property	
Public Property Let X(ByVal vNewValue As Long)	'设置 X 的值
Call SetCursorPos(vNewValue, Y)	'调用过程
End Property	

## ③ 类模块的方法

在类模块中声明为 Public 的过程，（包括 Sub 或 Function）将成为类的方法。如下面的代码，即将 ClipTo 声明为方法。

Public Sub ClipTo(ToCtl As Object)	'声明公共方法
On Error Resume Next	'错误处理语句
Dim tmpRect As RECT	'定义变量
Dim pt As POINTAPI	'定义变量
With ToCtl	'With 语句
If TypeOf ToCtl Is Form Then	'控件类型为窗体
tmpRect.Left = (.Left \ Screen.TwipsPerPixelX)	'给临时 Left 赋值
tmpRect.Top = (.Top \ Screen.TwipsPerPixelY)	'给临时 Top 赋值
tmpRect.Right = (.Left + .Width) \ Screen.TwipsPerPixelX	'给临时 Right 赋值



tmpRect.Bottom = (.Top + .Height) \ Screen.TwipsPerPixelY	'给临时 Bottom 赋值
Elseif TypeOf ToCtl Is Screen Then	'如果控件类型为屏幕
tmpRect.Left = 0	'给临时 Left 赋值
tmpRect.Top = 0	'给临时 Top 赋值
tmpRect.Right = (.Width \ Screen.TwipsPerPixelX)	'给临时 Right 赋值
tmpRect.Bottom = (.Height \ Screen.TwipsPerPixelY)	'给临时 Bottom 赋值
Else	'否则
pt.X = 0	'X 坐标为 0
pt.Y = 0	'Y 坐标为 0
Call ClientToScreen(.hWnd, pt)	'调用过程
tmpRect.Left = pt.X	'给临时 Left 赋值
tmpRect.Top = pt.Y	'给临时 Top 赋值
pt.X = .Width	'设置 X 坐标
pt.Y = .Height	'设置 Y 坐标
Call ClientToScreen(.hWnd, pt)	'调用过程
tmpRect.Bottom = pt.Y	'给临时 Bottom 赋值
tmpRect.Right = pt.X	'给临时 Right 赋值
End If	
Call ClipCursor(tmpRect)	'调用过程
End With	
End Sub	

其他代码请参见本书光盘中的源程序。

## 12.3 对象

 教学录像：光盘\TM\lx\12\对象的创建与使用.exe

本节介绍对象的概念、创建、使用，以及对象数组的创建。

### 12.3.1 对象的概念

“对象”是一个很广义的概念，在现实世界中的一切都可以看作对象，比如一个人、一本书等。对象的特点：

- ☒ 有一个名字用以区别于其他对象。
- ☒ 有一个状态描述该对象的特征。
- ☒ 有一组操作决定该对象的功能。

例如，有个人名字叫张三，性别男，会打球。张三就是一个对象，状态为性别（男），功能是“打球”。

在计算机中“对象”是可视为一个单元的代码和数据的组合。对象可以是一段应用程序，如控件或窗体。这个应用程序也可以是一个对象。

Visual Basic 对象具有属性、方法和事件。属性是描述对象的数据。方法告诉对象应该做的事情。事件是对象所产生的事情，事件发生时编写代码进行处理。



在 Visual Basic 中，对象是由类创建的，因此对象被说成是类的一个实例。为了使用对象，必须在对象变量中保存其引用。绑定的类型决定使用对象变量访问对象方法。对象变量可以是后期绑定（慢），或者是事前绑定（快）。

所谓事前绑定就是直接声明对象类名称的方式。也就是说 VB 6.0 在编译过程中，可以确切知晓对象变量的类型，因而能立即配置适当的记忆控件与中间语言代码。

后期绑定就是先将对象变量声明为变体类型 Variant 或对象类型 Object。就是说在编译阶段，VB 6.0 完全不知道变量的类型，只有在执行到指定语句时，才会将变量类型执行为特定对象。

### 12.3.2 对象的创建

在现实世界中的一切都可以看作对象，在 Microsoft Visual Basic 中的窗体、控件都可以视作对象。下面将对对象的创建、对象的比较等进行介绍。

#### ☑ 创建静态对象

当创建窗体或将 Visual Basic 工具栏中的控件拖放到窗体上时，实际就是创建了一个对象。

#### ☑ 创建动态对象

动态创建对象是指利用 Visual Basic 的可编程性能创立对象。

动态创建对象的步骤如下。

##### (1) 声明对象变量。

如下例：

```
Dim conn As ADODB.Connection
```

如果要响应该对象的事件需要在声明的时候加入关键字 “WithEvents”。如下例：

```
Dim WithEvents T As TextBox
```

##### (2) 使用 Set 和 New 关键字创建这个类的新实例。

```
Set conn = New ADODB.Connection
```

### 12.3.3 对象的使用

#### 1. 对象的比较

在编程过程中程序员可能需要判断多个对象变量是否指向同一对象实例。对象变量的比较并不使用等号 “=” 进行判断，而是通过使用 Is 运算符判断。

例 12.5 例如判断两个同为 MyClass 类型对象变量 i 和 j 是否指向同一对象实例，代码如下：（实例位置：光盘\TM\sl12\5）

```
Private Sub Form_Load()  
    Dim i As Class1  
    Set i = New Class1  
    Dim j As Class1  
    Set j = i
```

'声明实例变量  
'创建实例  
'声明实例变量  
'创建实例

```
MsgBox i Is j           '判断同一性
End Sub
```

上面代码中对象变量 j 指向 Class1 类对象 i，所以返回值为 True。

**例 12.6** 判断两个同为 MyClass 类型对象变量 i 和 j。它们并没有被实例化，判断它们是否指向同一对象。代码如下：（实例位置：光盘\TM\sl\12\6）

```
Private Sub Form_Load()
    Dim i As Class1           '声明实例变量
    Dim j As Class1           '声明实例变量
    MsgBox i Is j             '判断同一性
End Sub
```

上面代码比较没被实例化的对象变量，返回结果也为 True。因为对象变量 i 和 j 没被实例化，它们的对象都被视为 Nothing，所以返回值为 True。

**例 12.7** 判断两个同为 MyClass 类型对象变量 i 和 j。它们都被实例化为 MyClass 类型对象，判断它们是否指向同一对象。代码如下：（实例位置：光盘\TM\sl\12\7）

```
Private Sub Form_Load()
    Dim i As Class1           '声明实例变量
    Dim j As Class1           '声明实例变量
    Set i = New Class1        '声明实例变量
    Set j = New Class1        '声明实例变量
    MsgBox i Is j             '判断同一性
End Sub
```

上面代码比较已被实例化的对象变量，返回结果为 False。因为对象变量 i 和 j 都被实例化，系统为它们分配了不同的内存地址，虽然它们所属类型相同，但它们是两个独立的实例，所以返回值为 False。

## 2. 对象所属类的判断

在使用对象变量时，可能要根据对象的不同类而采取不同的操作。例如，某些对象可能不具有某些个别的属性和方法，为了防止调用对象不存在的属性或方法需要对对象的类型进行判断。

Visual Basic 提供了两种方法用以判别对象所属类：TypeOf 关键字和 TypeName 函数。

☒ TypeOf 关键字只能在 If...Then...Else 语句中使用。必须在代码中直接包含类名。例如：

```
If TypeOf MyControl Is TextBox Then
```

☒ TypeName 函数则比较灵活，因为返回的是它的类名的字符串返回值，所以可以将它与字符串进行比较，例如：

```
If TypeName(MyControl) = "TextBox" Then
```

## 3. 对象的清除

每个对象都消耗内存和系统资源，当不再使用对象时需要及时释放这些资源，否则会影响程序运行效率甚至会造成操作系统死机。

☒ 清除静态对象（窗体或控件）

使用 Unload 语句从内存中卸载窗体或控件。

### ☑ 清除动态对象

用 Nothing 释放对象变量占用的资源，释放时使用 Set 语句将 Nothing 赋予对象变量。

## 12.3.4 对象数组

像声明和使用任何数据类型的数组那样，也能够声明和使用对象类型的数组。这些数组既可以是静态数组，也可以是动态数组。

与声明其它任何类型数组的方法相同，可以使用 Private、Dim、ReDim、Static 或 Public 声明窗体数组。如果数组声明使用了 New 关键字，则当使用该数组的元素时，Visual Basic 将自动为数组中每个元素创建一个新窗体实例。如下面代码：

```
Private Sub Command1_Click ()
    Dim intX As Integer
    Dim frmNew(1 To 5) As New Form1           '声明窗体类型数组
    For intX = 1 To 5
        frmNew(intX).Show                     '显示窗体实例
    Next
End Sub
```

按命令按钮执行上述代码，将建立 5 个 Form1 实例。

## 12.4 系统对象

### 📺 教学录像：光盘\TM\lx\12\常见系统对象的使用.exe

在 VB 6.0 中提供了几种非常重要的系统对象。它们分别是：应用程序对象（APP 对象）、屏幕对象（Screen 对象）、剪切板对象（Clipboard 对象）、调试对象（Debug 对象）等。下面将对常用的系统对象进行详细介绍。

### 12.4.1 应用程序对象（APP 对象）

App 对象是通过关键字 App 访问的全局对象。它指定如下信息：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，以及判断是否运行前一个应用程序的示例。

下面将介绍该对象的几个常用属性：

#### 1. CompanyName 属性

返回或设置一个字符串，该字符串包括运行中的应用程序的公司或创建者名称。该属性在运行时是只读的。

**例 12.8** 在程序标题栏中添加程序的创建者或公司的名字，代码如下：（实例位置：光盘\TM\lx\12\8）

```
Me.Caption = Me.Caption & " (" & App.CompanyName & "创建) "
```

程序运行效果如图 12.5 所示。

## 2. EXENAME 属性与 Path 属性

**EXENAME 属性:**

返回当前正运行的可执行文件的根名(不带扩展名)。

如果是在开发环境下运行,则返回该工程名。

**Path 属性:**

返回或设置当前路径。在设计时是不可用的。对于 App 对象,在运行时是只读的。

**例 12.9** 将当前运行的程序创建一个备份,代码如下。

(实例位置:光盘\TM\s\12\9)

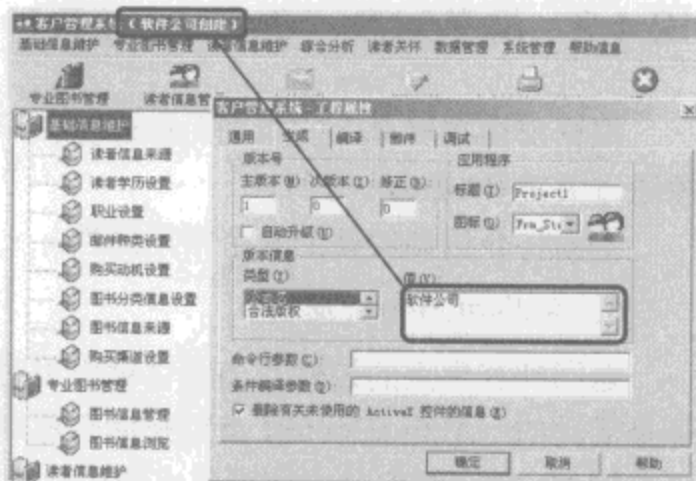


图 12.5 获取程序创建者或公司名称

```
FileCopy App.Path & "\" & App.EXENAME & ".EXE", App.Path & "\" & App.EXENAME & "备份.EXE"
```

注意: 本实例无法在工程中直接调试,需要将其生成并运行可执行文件后才能正常运行。

## 3. PrevInstance 属性

返回是否运行前一个应用程序的示例。返回值为布尔类型。

**例 12.10** 判断应用程序是否运行过,如果运行过则关闭当前运行的程序。代码如下:(实例位置:光盘\TM\s\12\10)

```
Private Sub Form_Load()  
    If App.PrevInstance = False Then  
        '当程序没运行过  
    Else  
        '当程序已运行过  
        MsgBox "程序已运行"  
        '弹出提示对话框  
    End  
        '关闭工程  
    End If  
End Sub
```

注意: 本实例无法在工程中直接调试,需要将其生成并运行可执行文件后才能正常运行。

## 4. Title 属性

返回或设置应用程序的标题,该标题要显示在 Microsoft Windows 的任务列表中。如果在运行时发生改变,那么发生的改变不会与应用程序一起被保存。

**例 12.11** 设置应用程序标题,代码如下:(实例位置:光盘\TM\s\12\11)

```
Private Sub Form_Load()  
    App.Title = "读者管理系统"  
    MsgBox "请按正确格式输入"  
End Sub
```

运行本实例程序,弹出对话框,显示效果如图 12.6 所示。



## 12.4.2 屏幕对象 (Screen 对象)

根据窗体在屏幕上的布局而操作窗体，并在运行时控制应用程序窗体之外的鼠标指针。Screen 对象通过关键字 Screen 访问。

下面将介绍该对象的几个常用属性：

### 1. ActiveControl 属性

返回拥有焦点的控件。当窗体被引用时，例如，在 ChildForm.ActiveControl 中，如果被引用的窗体是活动的，ActiveControl 指定将拥有焦点的控件。在设计时是不可用的；在运行时是只读的。

例 12.12 将窗体中获得焦点的控件隐藏，代码如下：（实例位置：光盘\TM\sl\12\12）

```
Private Sub Form_Click()  
    Screen.ActiveControl.Visible = False  
End Sub
```

当 TextBox 控件 (Text1) 获得焦点时，显示效果如图 12.7 所示。单击程序窗体后显示效果如图 12.8 所示。

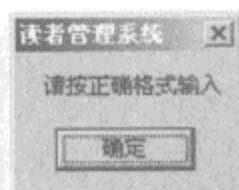


图 12.6 程序对话框

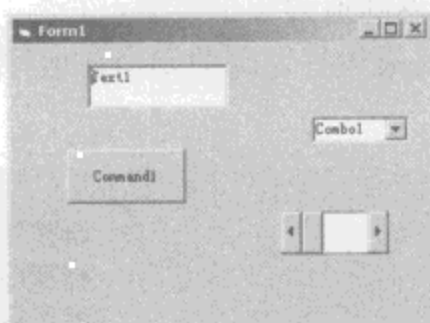


图 12.7 单击窗体前

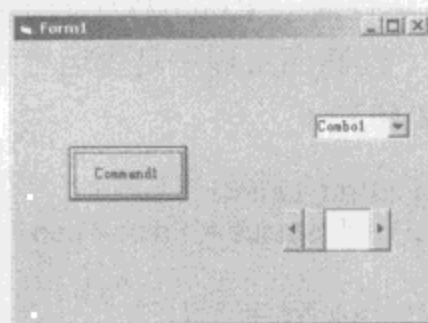


图 12.8 单击窗体后

### 2. ActiveForm 属性

返回活动窗口的窗体。如果 MDIForm 对象是活动的或者是被引用的，则所指定的是活动的 MDI 子窗体。

例 12.13 通过使用 ActiveForm 属性实现“打老鼠”游戏的效果。（实例位置：光盘\TM\sl\12\13）老鼠所在窗体的变化主要是由 Timer 控件实现的，主要代码如下：

```
Private Sub Timer1_Timer()  
    Randomize  
    clt(Int(Rnd * 3) + 1).SetFocus  
    Screen.ActiveForm.Image1.Visible = True  
End Sub
```

'随机初始化  
'随机设置窗口焦点，其中 clt 为自定义的窗体对象集合  
'显示获得焦点窗体中的老鼠图片

运行本实例程序后运行效果如图 12.9 所示，小老鼠在各个窗体间来回穿梭。



图 12.9 打老鼠效果图



### 3. Height、Width 属性

返回或设置对象高度与宽度，对于 Printer 和 Screen 对象在设计时不可用。

例 12.14 将程序窗口全屏显示。（实例位置：光盘\TM\sl\12\14）

代码如下：

```
Private Sub Form_Load()  
    With Me  
        .Top = 0  
        .Left = 0  
        .Width = Screen.Width  
        .Height = Screen.Height  
    End With  
End Sub
```

### 12.4.3 剪贴板对象（Clipboard 对象）

Clipboard 对象用于操作剪贴板上的文本和图形。它使用户能够复制、剪切和粘贴应用程序中的文本和图形。该对象的几个常用方法如表 12.6 所示。

表 12.6 Clipboard 对象常用方法

方 法	描 述	示 例 代 码
GetData	用于从 Clipboard 对象返回一个图形	Me.Picture = Clipboard.GetData
GetText	用于返回 Clipboard 对象中的文本字符串	Me.Print Clipboard.GetText
SetData	用指定的图形格式将图片放到 Clipboard 对象上	Clipboard.SetData Me.Picture
SetText	用指定的 Clipboard 图像格式将文本字符串放到 Clipboard 对象中	Clipboard.SetText Text1.Text

### 12.4.4 调试对象（Debug 对象）

Debug 对象在运行时将输出发送到立即窗口。下面将介绍该对象的方法。

#### 1. Assert 方法

有条件地在该方法出现的行上挂起执行。

语法：

```
object.Assert booleanexpression
```

Assert 方法的语法有如下的对象限定符和参数：

object：必要参数。总是 Debug 对象。

booleanexpression：必要参数。一个值为 True 或者 False 的表达式。

#### 2. Print 方法

在立即窗口中显示文本。

语法：

```
object.Print [outputlist]
```

Print 方法的语法具有下列对象限定符和参数:

Object: 必要参数。对象表达式, 其值为应用于列表中的对象。

Outputlist: 可选参数。要打印的表达式或表达式的列表。如果省略, 则打印一空白行。

## 12.5 小结

本章首先介绍了面向对象编程的概念, 然后对类和对象的概念、创建以及使用进行介绍, 最后详细地介绍了 VB 系统对象。作为初学者, 面向对象的相关知识了解即可, 重点应掌握实际应用较为广泛的 VB 系统对象——App 对象。

## 12.6 练习与实践

1. 创建一个类模块, 其中具有弹出对话框的方法, 对话框样式与内容任意。(答案位置: 光盘\TM\12\15)
2. 获取当前程序所在目录路径。(答案位置: 光盘\TM\12\16)
3. 使用对象数组创建 2 个窗体对象。(答案位置: 光盘\TM\12\17)

# 第13章

## 常用 ActiveX 控件

(  教学视频: 1 小时 46 分钟 )

ActiveX 控件, 是扩展名为 .ocx 的独立文件, 其中包括各种版本 VB 提供的控件 (例如 DataCombo、DataList 控件等) 和仅在专业版和企业版中提供的控件 (例如 ListView、Toolbar、Animation 和 Tabbed Dialog 控件), 另外还有许多第三方提供的 ActiveX 控件。

本章从添加 ActiveX 控件、注册 ActiveX 控件开始, 到常用 ActiveX 控件的介绍及应用, 让您逐步体会 ActiveX 控件在实际编程中的应用。

通过阅读本章, 您可以:

- ▶▶ 掌握添加 ActiveX 控件、删除 ActiveX 控件的方法
- ▶▶ 掌握手动注册 ActiveX 控件和编程注册 ActiveX 控件的方法
- ▶▶ 学会 ImageList 控件与其他控件的联合应用
- ▶▶ 掌握 ListView 控件几种显示数据的方法
- ▶▶ 掌握树状列表控件的重要属性及其在程序中的应用
- ▶▶ 灵活地运用选项卡、进度条和日期/时间选择控件

## 13.1 ActiveX 控件的使用


 教学录像：光盘\TM\lx\13\ActiveX 控件的使用.exe

了解了 ActiveX 控件，接下来介绍在程序中如何加载和删除已有的 ActiveX 控件，以及对于从外界复制过来的 ActiveX 控件如何进行手动注册、编程注册和使用 REGSVR32.exe 工具注册。

### 13.1.1 添加 ActiveX 控件


ActiveX 控件文件的扩展名为.ocx，可以是 Visual Basic 提供的 ActiveX 控件，也可以是从第三方开发商获得的附加控件。

将 ActiveX 控件和其他可加入的对象添加到工具箱当中，即可在工程中使用它们。

 注意：Visual Basic 的 ActiveX 控件是 32 位控件。一些第三方开发商提供的 ActiveX 控件是 16 位控件，这样的控件不能在 VB 中使用。

下面介绍在工程中的工具箱中加入 ActiveX 控件，具体步骤如下。

(1) 在“工程”菜单中，选择“部件”命令，打开“部件”对话框，如图 13.1 所示。

 技巧：也可以在工具箱中单击鼠标右键，在快捷菜单中选择“部件”，然后打开“部件”对话框。

(2) 该对话框中将列出所有已经注册的可加入的对象、设计者和 ActiveX 控件。

(3) 要在工具箱中加入 ActiveX 控件，需要选中该控件名称左边的复选框。

(4) 单击“确定”按钮，关闭“部件”对话框。所有选择的 ActiveX 控件将出现在工具箱中。

如果要将 ActiveX 控件加入“部件”对话框，应单击“浏览”按钮，并找到扩展名为.ocx 的文件。这些文件通常安装在\Windows\System 或 System32 目录中，如图 13.2 所示。在将 ActiveX 控件加入可用控件列表中时，Visual Basic 自动在“部件”对话框中选中它的复选框。

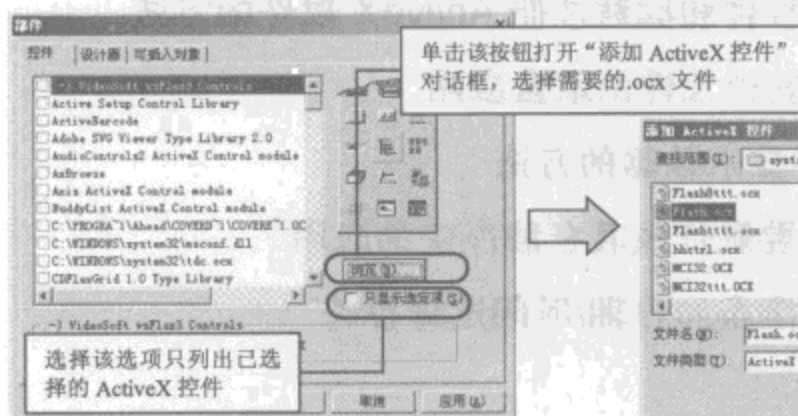


图 13.1 “部件”对话框

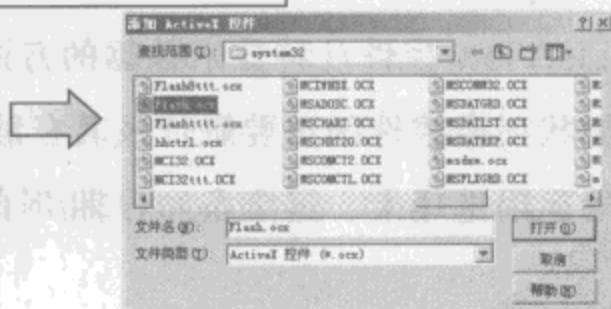



图 13.2 “添加 ActiveX 控件”对话框

 说明：在众多控件名称中浏览已选择的 ActiveX 控件很困难，此时可以选择“只显示选定项”复选框，将已选择的 ActiveX 控件显示出来，以方便查看。

### 13.1.2 删除 ActiveX 控件

删除 ActiveX 控件就是从工具箱中将选定的 ActiveX 控件删除，方法是：打开“部件”对话框，将已选择的 ActiveX 控件名称前中复选框中的“√”符号去掉，但这里要确保该 ActiveX 控件没被使用，否则将无法从工具箱中将其删除。

### 13.1.3 注册 ActiveX 控件

作为 ActiveX 控件，在能够被任何 Windows 程序（包括 Visual Basic 程序）识别或者使用之前，外接程序必须被正确地注册到系统注册表中。在外接程序的编译过程中 VB 自动将其注册到系统中，但是，如果从外界复制 ActiveX 控件到本系统中使用，则用户必须在使用之前将其注册到该系统中。

注册 ActiveX 控件主要使用 Regsvr32.exe 实用程序。下面分别介绍手动注册和编写程序自动注册。

#### 1. 手动注册

单击“开始”按钮选择“运行”命令，如图 13.3 所示，打开“运行”对话框。在“打开”文本框中输入如下内容：

```
regsvr32 C:\WINDOWS\system32\*.ocx
```

例如注册 Flash.ocx 控件，在“打开”文本框中输入 regsvr32 C:\WINDOWS\system32\flash.ocx，如图 13.4 所示。

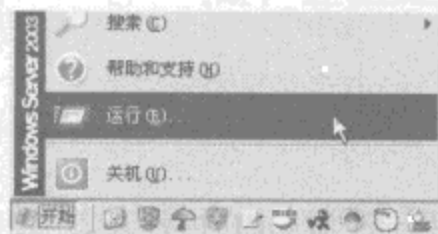


图 13.3 选择“运行”菜单命令

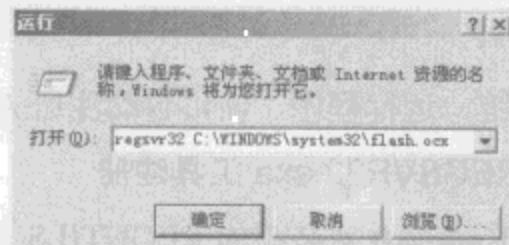


图 13.4 在“运行”对话框中输入注册命令

单击“确定”按钮，弹出注册成功的对话框，表示 Flash 控件注册成功。

#### 2. 编写程序自动注册

**例 13.1** 手动注册有时操作起来比较麻烦，下面编写程序，实现自动注册所需的 ActiveX 控件。新建一个工程，在该工程中会自动创建一个名为 Form1 的窗体，切换到代码窗口，编写如下代码。（实例位置：光盘\TM\sl\13\1）

```
Option Explicit
Private Declare Function GetSystemDirectory Lib "kernel32" Alias "GetSystemDirectoryA" (ByVal lpBuffer As String, ByVal nSize As Long) As Long
Private Const max_path = 260
Private Const max_path1 = 261
Dim sysdir As String
Private Sub Form_Activate()
    On Error GoTo orroelink
```



```

Dim retval, retval1, retval2
Dim chrLen As Long
Dim windir As String, MYPATH As String, a1 As String, a2 As String
'获取系统路径
sysdir = Space(max_path)
chrLen = GetSystemDirectory(sysdir, max_path)
If chrLen > max_path Then chrLen = GetSystemDirectory(sysdir, chrLen)
sysdir = Left(sysdir, chrLen)
'开启 Sccrun.dll
Shell ("regsvr32 /s " & sysdir & "\Sccrun.dll 开启")
'判断系统中是否存在 ActiveX 控件
a1 = Dir(sysdir & "\Flash.ocx")
If a1 = "" Then
    '复制并注册 ActiveX 控件
    FileCopy App.Path & "\link\Flash.ocx", sysdir & "\Flash.ocx"
    Shell ("regsvr32 /s " & sysdir & "\flash.ocx")
End If
a2 = Dir(sysdir & "\MCI32.OCX")
If a2 = "" Then
    '复制并注册 ActiveX 控件
    FileCopy App.Path & "\link\MCI32.OCX", sysdir & "\MCI32.OCX"
    Shell ("regsvr32 /s " & sysdir & "\MCI32.OCX")
End If
Exit Sub
orroelink:
MsgBox Err.Description, vbOKOnly, "提示信息"
End Sub

```

按〈F5〉键，运行程序，便可自动注册 ActiveX 控件。

### 3. 使用 REGSVR32.exe 工具注册

在 Visual Basic 6.0 安装盘的 REGUTILS 目录下有 3 个可用于注册 OLE 控件及其 DLL 的工具，即 REGIT.EXE、REGOCX32.EXE 和 REGSVR32.EXE。

- ☒ REGSVR32.EXE 用于注册 OLE Server，包括 OLE 控件的 DLL。
- ☒ REGOCX32.EXE 专用于注册 OCX 控件。
- ☒ REGIT.EXE 用于一次注册多个 OLE Server。

## 13.2 图像列表控件 (ImageList)

图像列表控件 ImageList 用于提供一些图像，类似于图像库，与其他控件关联应用。下面就认识一下 ImageList 控件，向 ImageList 控件中添加图像和使用屏蔽颜色。

### 13.2.1 认识 ImageList 控件

图像列表控件 ImageList 位于“部件”对话框的 Microsoft Windows Common Controls 6.0 (SP6) 选

项中，其添加到工具箱中的图标为.

它的作用有些像图像的储藏室，需要第二个控件显示它所储存的图像。第二个控件可以是任何能显示图像 Picture 对象的控件，也可以是特别设计的、用于关联 ImageList 控件的 Windows 通用控件之一。这些控件包括 ListView、ToolBar、TabStrip、Header、ImageCombo 和 TreeView。为了与这些控件一同使用 ImageList，必须通过一个适当的属性将特定的 ImageList 控件关联到第二个控件。对于 ListView 控件，必须设置其 Icons 和 SmallIcons 属性为 ImageList 控件。对于 TreeView、TabStrip、ImageCombo 和 Toolbar 控件，必须设置 ImageList 属性为 ImageList 控件。

### 13.2.2 添加图像


向 ImageList 控件中添加图像有两种方法，一是设计时通过“属性页”对话框添加图像，二是编写代码添加图像。

#### 1. 设计时添加图像

要在设计时添加图像，可以使用 ImageList 控件的“属性页”对话框，具体步骤如下。

(1) 在窗体上添加一个 ImageList 控件，右击它，在弹出的快捷菜单中选择“属性”命令，打开“属性页”对话框。

(2) 单击“通用”选项卡，在此设置图像的大小和是否使用屏蔽颜色。如果使用屏蔽颜色，则图像可以透明。图像的大小可以选择，也可以自定义，如使用 32×32，选择 32×32 复选框即可。

 **注意：**一定要先设置图像的尺寸，然后再添加图像到 ImageList 控件中，否则如果图像尺寸不合适，就得将添加进来的图像全部删除，然后重新设置图像的尺寸。这样很麻烦，尤其图像较多的情况，笔者就遇到过这样的问题。

(3) 单击“图像”选项卡，显示 ImageList 控件的“属性页”，如图 13.5 所示。

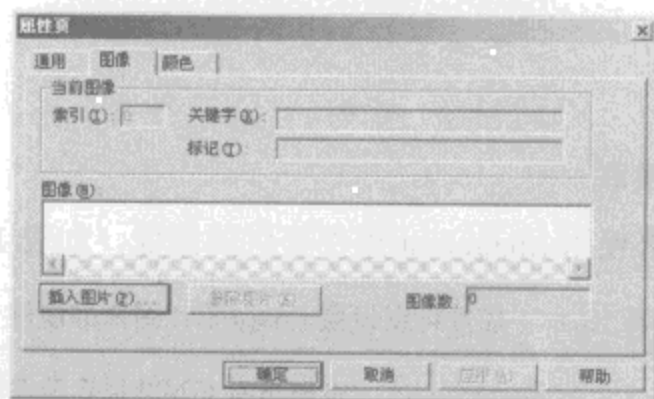


图 13.5 ImageList 控件的“属性页”对话框

 **技巧：**可以选择多个位图或图标文件。

① 单击“插入图片”按钮，打开“选择图片”对话框，使用该对话框查找位图或图标文件，并单击“打开”按钮。

② 在“关键字”文本框中输入一个字符串，为 Key 属性设置一个唯一的属性值。

③ 在“标记”文本框中输入一个字符串，为 Tag 属性设置属性值，该项是可选的。另外，

Tag 属性不必唯一。

④ 重复第①步至第③步，直到在 ImageList 控件中充填了全部想要的图像。

## 2. 编写代码添加图像

要在程序运行时，编写代码添加图像到 ImageList 控件中，可以使用 LoadPicture 函数，并结合 ListImages 集合的 Add 方法。

例 13.2 窗体载入时，将名为 ImageList1 的 ImageList 控件和一个图标一起加载，代码如下。（实例位置：光盘\TM\sl\13\2）

```
Private Sub Form_Load()  
    '如果路径是正确的，那么 5.ico  
    '图标将被添加到 ListImages  
    '集合中。为 Key 属性分配的属性  
    '值是工资 gz  
    ImageList1.ListImages.Add, "gz", LoadPicture(App.Path & "\工具栏图标\5.ico")  
End Sub
```

为了验证该图标确实被添加到了 ImageList 控件中，编写如下代码，将该图标文件作为窗体的图标显示出来。

```
Set Form1.Icon = ImageList1.ListImages(1).Picture
```

下面给出运行前后的效果，如图 13.6 和图 13.7 所示。

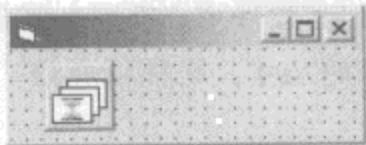


图 13.6 设计时效果



图 13.7 运行时效果


## 13.2.3 与其他控件关联

### 1. 与 Windows 公用控件关联


可以利用下列控件的属性来使用 ImageList 控件提供的图像，如表 13.1 所示。

表 13.1 控件可以设置 ImageList 图像的属性

控 件	可设置为 ImageList 图像的属性
ImageCombo 控件	ComboItemImage、OverlayImage 和 SellImage 属性
ListView 控件	ListItemImage 和 Icon 属性
TreeView 控件	NodeImage 和 SelectedImage 属性
Toolbar 控件	ButtonImage、ButtonHotImageList 和 ButtonDisabledImageList 属性
TabStrip 控件	TabImage 属性

 说明: 有关和 TreeView、ListView、Toolbar 控件一起使用的 ImageList 的实例, 可参阅这些控件的应用方案。

要和上述控件一起使用 ImageList, 必须首先将 ImageList 和这些控件关联起来, 然后为表 13.1 所列的某一种属性指定 Key 或 Index 属性值。可以在设计或运行时进行这些工作。所有 Windows 公用控件, 除了 ListView 控件 (在本主题中讨论) 外, 都具有一个 ImageList 属性, 这个属性可以设置为正在使用的 ImageList 控件名。

 注意: 在将 ImageList 控件和其他控件关联之前将需要的图像添加到 ImageList 控件中。一旦将 ImageList 和某个控件关联起来并将某个图像分配给控件的属性后, ImageList 控件就不允许添加其他的图像了。

**例 13.3** 下面将 ImageList 控件和 TreeView、TabStrip 或 Toolbar 控件关联, 具体步骤如下。(实例位置: 光盘\TM\sl\13\3)

(1) 在使用 ImageList 控件中图像的控件上右击, 然后选择“属性”命令打开“属性页”对话框。

(2) 在“通用”选项卡上, 在 ImageList 下拉列表框中选择 ImageList 控件的名称。

如果要在运行时关联 ImageList 控件, 例如将 TreeView 控件与 ImageList 控件关联, 代码如下。

```
Set TreeView1.ImageList = ImageList1
```

将 TreeView 控件与 ImageList 控件关联

将 ImageList 控件和其他控件关联以后, 可以使用 ImageList 控件中图像的 Key 或 Index 属性来设置各种对象的属性。例如, 下面的代码将 TreeView 控件中 Node 对象的 Image 属性设置成 Key 属性为 yg 的 ImageList 图像。

```
Private Sub Form_Load()  
    Set TreeView1.ImageList = ImageList1  
    '添加一个节点并设置其 Image 属性。  
    TreeView1.Nodes.Add , , "a1", "员工 1", "yg"  
End Sub
```

按〈F5〉键, 运行程序, 结果如图 13.8 所示。

## 2. 与 ListView 控件关联

ListView 控件可以同时使用两个 ImageList 控件, 原因是 ListView 控件具有 Icons 和 SmallIcons 两个属性, 每个属性都可以和一个 ImageList 控件关联。可以在设计时或运行时设置这些关联。

在设计时将 ListView 控件和两个 ImageList 控件关联, 具体步骤如下。

(1) 在 ListView 控件上右击, 在弹出的菜单中选择“属性”命令, 打开“属性页”对话框。

(2) 单击“图像列表”选项卡, 在“普通”下拉列表框中选择一个 ImageList 控件的名称。

(3) 在“小图标”下拉列表框中, 选择另一个 ImageList 控件的名称。

**例 13.4** 在程序运行时, 也可以将 ImageList 控件与 ListView 控件关联, 例如下面的代码。(实例

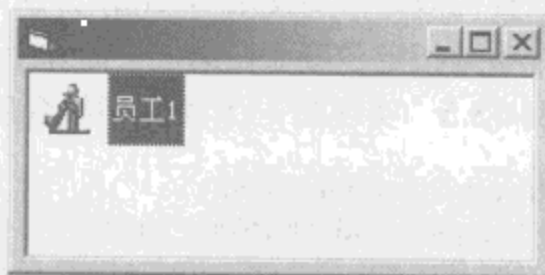



图 13.8 ImageList 控件与 TreeView 控件关联



位置: 光盘\TM\sl\13\4)

```
'第一个 ImageList 的名称是"ImageList1"
'第二个的名称是"ImageList2"
Set ListView1.SmallIcons = ImageList1
Set ListView1.LargeIcons = ImageList2
```

所使用的 ListView 控件取决于 ListView 控件的 View 属性中所决定的显示模式。如果 ListView 控件是在“图标”视图下,那么它将使用 Icons 属性中命名的 ImageList 控件所提供的图像。在其他视图(“列表”、“报告”或“小图标”)中,ListView 控件使用 SmallIcons 属性中命名的 ImageList 控件所提供的图像。

 说明: 有关详细的 ListView 控件的应用,可参见本章第 13.3 节。

### 3. 与不是 Windows 公用控件的控件关联

还可以把 ImageList 控件用作具有 Picture 属性的对象的图像图像库。这些对象包括: CommandButton 控件、OptionButton 控件、Image 控件、PictureBox 控件、CheckBox 控件、Form 对象和 Panel 对象(StatusBar 控件)。

ListImage 对象的 Picture 属性返回一个 Picture 对象,该对象可以用来分配给其他控件的 Picture 属性。例如,下面的代码将在名为“Picture1”的 PictureBox 控件中显示第二个 ListImage 对象。

```
Set Picture1.Picture = ImageList1.ListImages(2).Picture
```

## 13.2.4 创建组合图像

可以使用 ImageList 控件来创建组合图像(图片对象),这个组合图像是由两个图像通过使用 Overlay 方法结合 MaskColor 属性合成的。例如,如果有一个“不允许操作”的图像(一个圆中间有一个对角斜杠),那么可以把这个图像放在其他的图像上面,如图 13.9 所示,这样便形成了组合效果。

这样的效果需要使用 Overlay 方法,该方法的语法需要两个参数。第一个参数指定了下面的图像;第二个参数指定了覆盖在第一个图像上的图像。两个参数都可以是 ListImage 对象的 Index 或 Key 属性。

例 13.5 实现图 13.9 所示的组合后的效果,需要编写如下代码。(实例位置: 光盘\TM\sl\13\5)

```
Private Sub Form_Load()
    ImageList1.MaskColor = vbGreen           '设置屏蔽颜色为绿色
    Set Picture1.Picture = ImageList1.Overlay(2, 1) '将组合的图像通过 Picture 控件显示出来
End Sub
```

也可以使用图像的 Key 属性,代码如下。

```
'第一个图像的 Key 值是"no", 第二个图像的 Key 值是"box"
Set Picture1.Picture = ImageList1.Overlay("box","no")
```

按〈F5〉键,运行程序,结果如图 13.10 所示。





图 13.9 图像组合过程



图 13.10 组合后的图像在 Picture 控件中的显示效果

上面的实例同时也说明了 MaskColor 属性是如何工作的。简单地说, MaskColor 属性指定了当一个图像覆盖在另一个图像上面时哪一种颜色将变成透明的。no 图像有一个绿色的背景, 这样, 在代码指定 MaskColor 属性为 vbGreen (内部常量) 时, 图像中的绿色会在组合图像中变成透明的。

### 13.3 视图控件 (ListView)

 教学录像: 光盘\TM\lx\13\视图控件 (ListView) .exe

视图控件 ListView 控件可使用 4 种不同视图显示项目。通过此控件, 可将项目组成带有或不带有列标头的列, 并显示伴随的图标和文本。下面介绍在工程中如何添加 ListView 控件以及 ListView 控件在程序中的应用。

#### 13.3.1 认识 ListView 控件

视图控件 ListView 位于“部件”对话框的 Microsoft Windows Common Controls 6.0 (SP6) 选项中, 其添加到工具箱中的图标为 .

ListView 控件可以用来显示不同类型的视图, 如列表、图标、报表等, 如图 13.11 所示。另外, 与 Treeview 控件联合使用, 可以给出 TreeView 控件节点的扩展视图。



图 13.11 用 ListView 控件显示不同类型的视图

#### 13.3.2 添加数据

Add 方法是 ListItems 集合的方法, 该方法用于向 ListView 控件中添加 ListItem 对象, 其语法格式如下, 相关参数说明如表 13.2 所示。

```
object.ListItems.Add([Index],[key],[Text],[Icon],[SmallIcon])
```

表 13.2 Add 方法中各参数的说明

参 数	说 明
object	必需的参数。对象表达式，其值是 ListItems 集合
index	可选的参数。指定在何处插入 ListItem 对象的整数。若未指定索引，则将 ListItem 对象添加到 ListItems 集合的末尾
key	可选的参数。唯一的字符串表达式，用来访问集合成员
text	可选的参数。与 ListItem 对象控件关联的字符串
icon	可选的参数。当 ListView 控件设为图标视图时，此整数设置从 ImageList 控件中选定的要显示的图标
smallIcon	可选的参数。当 ListView 控件设为小图标时，此整数设置从 ImageList 控件中选定的要显示的图标

例 13.6 下面使用 Add 方法将数据添加到 ListView 控件中，代码如下。（实例位置：光盘 \TM\sl\13\6）

```
Private Sub Form_Load()
    ListView1.ListItems.Add 1, "employee1", "小李"
    ListView1.ListItems.Add 2, "employee2", "小张"
    ListView1.ListItems.Add 3, "employee3", "小王"
End Sub
```

运行程序，结果如图 13.12 所示。

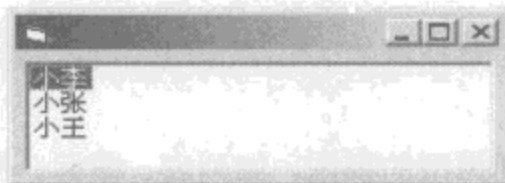


图 13.12 用 ListView 控件显示数据

### 13.3.3 用“ListView 控件+数据表”创建报表视图

用 ListView 控件创建报表视图，首先要使用 ColumnHeader 对象的 Add 方法给报表添加一个表头，然后使用 ListSubItem 对象的 Add 方法添加内容。这里需要说明的是：ListSubItem 对象是否存在，以及其数目都取决于 ColumnHeader 对象是否存在及其数目。也就是说，如果没有 ColumnHeader 对象，就不能创建任何 ListSubItem 对象。进一步说，ColumnHeader 对象的数目，决定了可为 ListItem 对象设置的 ListSubItem 对象的数目。

例 13.7 下面使用 ListView 控件以报表视图形式显示“客房信息表”中的所有记录，代码如下。（实例位置：光盘 \TM\sl\13\7）

```
Private Sub Form_Load()
    '建立一个 ADO 数据连接
    Dim cnn As New ADODB.Connection
    Dim rs As New ADODB.Recordset
    '建立一个连接字符串
```

```

'这个连接串可能根据数据库配置的不同而不同
cnn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & "\db_kfgl.mdb;
Persist Security Info=False"
'建立数据库连接
cnn.Open
rs.Open "select * from kf", cnn
If rs.EOF Then Exit Sub
'网格行
ListView1.GridLines = True
'采用报表显示模式
ListView1.View = lvwReport
Dim ListX As ListItem
Dim ListSubX As ListSubItem
Dim ColumnX As ColumnHeader
Dim i As Integer
'填充表头
For i = 0 To rs.Fields.Count - 1
    Set ColumnX = ListView1.ColumnHeaders.Add
    ColumnX.Text = rs.Fields(i).Name
    ColumnX.Width = ListView1.Width / rs.Fields.Count
Next i
'填充数据
Do Until rs.EOF
    '添加一行
    Set ListX = ListView1.ListItems.Add
    ListX.Text = rs.Fields(0).Value
    For i = 1 To rs.Fields.Count - 1
        Set ListSubX = ListX.ListSubItems.Add
        ListSubX.Text = rs.Fields(i).Value
    Next i
    rs.MoveNext
Loop
rs.Close
Set rs = Nothing
cnn.Close
Set cnn = Nothing
End Sub


```

'关闭记录集对象  
'从内存中删除记录集对象  
'关闭连接  
'内存中删除连接对象

按〈F5〉键，运行程序，结果如图 13.13 所示。

房间号	房间类型	楼层	房态	单价	容纳	已住	备注
2302	标房	二楼	空闲房	130	1	0	21
2303	标房	二楼	入住房	130	1	1	21
2304	标房	二楼	打扫中	130	1	0	21
2305	标房	二楼	维修中	130	1	0	21
2306	标房	二楼	维修中	130	1	0	21
2307	标房	二楼	入住房	130	2	1	21
2308	套房	二楼	空闲房	260	2	0	21
3301	标房	三楼	空闲房	130	2	0	21
3302	标房	三楼	入住房	130	2	2	21
3303	标房	三楼	空闲房	130	2	0	21
3304	标房	三楼	空闲房	130	2	0	21
3305	标房	三楼	维修中	130	2	0	21

图 13.13 以报表视图显示客房记录

 **技巧：**用 ListView 显示数据的时候，有时可能出现：“索引超出边界”。出现该问题原因有以下两点：

(1) 主要是行数或是列数超出了范围，如索引应从 1 开始，设置为 0 则报错。例如：

```
ListView1.ListItems(i).SubItems(0)
```

(2) 在加入一行之前没有创建一个新行，要避免出现该错误，必须在加入前添加一行。例如：

```
Dim itmX As ListItem
While Not Rs.EOF
    Set itmX = ListView1.ListItems.Add(, CStr(Rs!Name))
    If Not IsNull(Rs!dj) Then itmX.SubItems(1) = CStr(Rs!dj)
    If Not IsNull(Rs!jc) Then itmX.SubItems(2) = Rs!jc
    Rs.MoveNext
Wend
```

### 13.3.4 用 ListView 控件创建大图标视图

用 ListView 控件创建大图标视图需要将 ListView 控件与前面介绍的 ImageList 控件关联，并设置 ListView 控件的 View 属性值为 0-lvwIcon。

**例 13.8** 将例 13.7 中的房态信息用 ListView 控件以大图标形式显示出来，代码如下。（实例位置：光盘\TM\sl\13\8）

```
Dim itmX As ListItem           '声明一个 ListItem 对象
Dim text As String            '声明字符串变量
Dim cnn As New ADODB.Connection '声明一个数据连接对象
Dim rs1 As New ADODB.Recordset '声明一个记录集对象
Private Sub Form_Load()
    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & "\db_kfgl.mdb;Persist Security Info=False" '连接数据库
    rs1.Open "select * from kf order by 房态", cnn, adOpenKeyset, adLockOptimistic '连接数据表
    If rs1.RecordCount > 0 Then '如果记录大于零
        rs1.MoveFirst '将记录移到第一条
        '填充房间信息，不同的房态显示不同的图标
        Do While rs1.EOF = False
            text = rs1.Fields("房间号")
            Select Case rs1.Fields("房态")
                Case Is = "入住"
                    Set itmX = ListView1.ListItems.Add(, text, 1)
                Case Is = "空闲"
                    Set itmX = ListView1.ListItems.Add(, text, 2)
                Case Is = "维修"
                    Set itmX = ListView1.ListItems.Add(, text, 3)
            End Select
            rs1.MoveNext
        Loop
    End If
End Sub
```



按〈F5〉键，运行程序，结果如图 13.14 所示。



图 13.14 以大图标视图显示房态信息

## 13.4 树状控件 (TreeView)

 教学录像：光盘\TM\13\树状控件 (TreeView).exe

本节介绍认识 TreeView 控件、向 TreeView 控件添加数据、删除数据和展开/收缩数据，以及结合数据表创建多级树状视图。

### 13.4.1 认识 TreeView 控件

树状列表控件 TreeView 位于“部件”对话框的 Microsoft Windows Common Controls 6.0 (SP6) 选项中，其添加到工具箱中的图标为 。

TreeView 控件可以用来显示具有层次结构的数据，例如组织树、索引项、磁盘中的文件和目录等，如图 13.15 就是 TreeView 控件两个典型的用法。



图 13.15 用 TreeView 控件显示多级数据



### 13.4.2 添加数据

Add 方法是 TreeView 控件的 Node 对象的一个方法，该方法用于在 Treeview 控件的 Nodes 集合中添加一个 Node 对象，其语法格式如下，参数及其相关说明如表 13.3 所示。

```
object.Add(relative, relationship, key, text, image, selectedimage)
```

表 13.3 Add 方法的参数说明

参 数	说 明
object	必需的参数。对象表达式
relative	可选的参数。已存在的 Node 对象的索引号或键值。新节点与已存在的节点间的关系，可在下一个参数 relationship 中找到
relationship	可选的参数。指定的 Node 对象的相对位置，其设置值如表 13.4 所示
key	可选的参数。唯一的字符串，可用于用 Item 方法检索 Node 对象
text	必需的参数。在 Node 中出现的字符串
image	可选的参数。在关联的 ImageList 控件中的图像的索引
selectedimage	可选的参数。在关联的 ImageList 控件中的图像的索引，在 Node 对象被选中时显示

表 13.4 relationship 参数的设置值

常 数	值	描 述
tvwFirst	0	第一个节点，该节点（Node 对象）和在 relative 中被命名的节点位于同一层，并位于所有同层节点之前
tvwLast	1	最后的节点。该 Node 和在 relative 中被命名的节点位于同一层，并位于所有同层节点之后。任何连续添加的节点可能位于最后添加的节点之后
tvwNext	2	（默认）下一个节点。该 Node 位于在 relative 中被命名的节点之后
tvwPrevious	3	前一个节点。该 Node 位于在 relative 中被命名的节点之前
tvwChild	4	（默认）子节点。该 Node 为在 relative 中被命名的节点的子节点

❗ 注意：如果在 relative 参数中没有被命名的 Node 对象，则新节点被放在节点顶层的最后位置。

例 13.9 下面将图书信息以树状显示，代码如下。（实例位置：光盘\TM\sl\13\9）

```
Private Sub Form_Load()
    Dim nodex As Node
    Dim i As Integer
    TreeView1.Style = tvwTreelinesPlusMinusPictureText
    TreeView1.BorderStyle = ccFixedSingle
    Dim a
    a = Array("(01)工程部", "(02)销售部", "(03)财务部", "(04)企划部")
    '填充 TreeView 控件
    With TreeView1.Nodes
        Set nodex = .Add(, "R", "吉林省长春市公司", 2)
        For i = 0 To 3
            Set nodex = .Add("R", tvwChild, "C" & i, a(i), 1)
        Next i
    End With
End Sub
```

'定义一个 Node 对象  
'定义一个整型变量  
'设置 TreeView 控件的样式  
'设置 TreeView 控件边框的样式  
'定义一个变量 a  
'给数组赋值

```

nodex.EnsureVisible
Next
End With
End Sub

```

按〈F5〉键，运行程序，结果如图 13.16 所示。

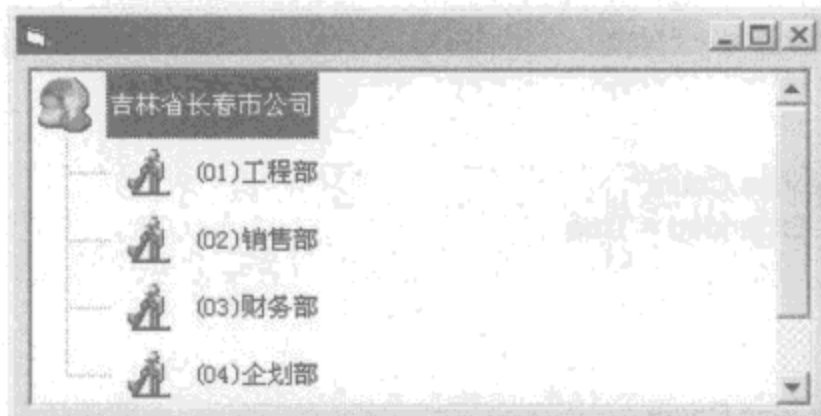


图 13.16 使用 Add 方法向 TreeView 控件添加数据

**技巧：**当 TreeView 失去焦点时，原来选定的内容也同样会失去焦点，这样用户使用起来很不方便，不知道之前选定了哪项。如果设置 Node 对象的 HideSelection 属性值为 False，即使 TreeView 失去焦点，选定内容还是会以选定状态出现。

### 13.4.3 删除指定节点数据

删除指定节点数据应使用 Node 对象的 Remove 方法。例如删除选定节点的内容，代码如下。

```
TreeView1.Nodes.Remove (TreeView1.SelectedItem.Index)
```

执行该语句便可删除选定节点的内容。但是如果选定了根节点，则该根节点和其下的子节点的内容也将全部被删除。此时，应判断如果选定的根节点存在子节点，则不允许删除该根节点，方法是：首先使用 SelectedItem 属性确定选定的节点，然后使用 Children 属性确定选定根节点的子节点的数量，代码如下。

```

If TreeView1.SelectedItem.Children > 0 Then
    MsgBox "此节点存在子节点不允许删除！"
End If

```

'如果选定节点的子节点大于零  
'提示用户

**技巧：**一次性清除所有数据可以使用 Clear 方法。

### 13.4.4 节点展开与折叠

节点展开与收缩应使用 Expanded 属性。该属性返回或设置一个布尔型值，其值指定节点是展开的还是折叠的，值为 True 表示展开节点，值为 False 表示折叠节点。

例如：

展开第一个节点：

```
TreeView1.Nodes(1).Expanded = True
```

折叠第一个节点：

```
TreeView1.Nodes(1).Expanded = False
```

展开所有节点：

```
For i = 1 To TreeView1.Nodes.Count
    TreeView1.Nodes(i).Expanded = True
Next i
```

折叠所有节点：

```
For i = 1 To TreeView1.Nodes.Count
    TreeView1.Nodes(i).Expanded = False
Next i
```

### 13.4.5 用“TreeView 控件+数据表”创建多级树状视图

当某些数据存在上下级关系时，例如部门、地区等，此时使用 TreeView 控件显示既节省空间，又方便查看。

例 13.10 下面将使用“TreeView 控件+部门数据表”实现添加和浏览多级部门信息，如图 13.17 所示。（实例位置：光盘\TM\13\10）

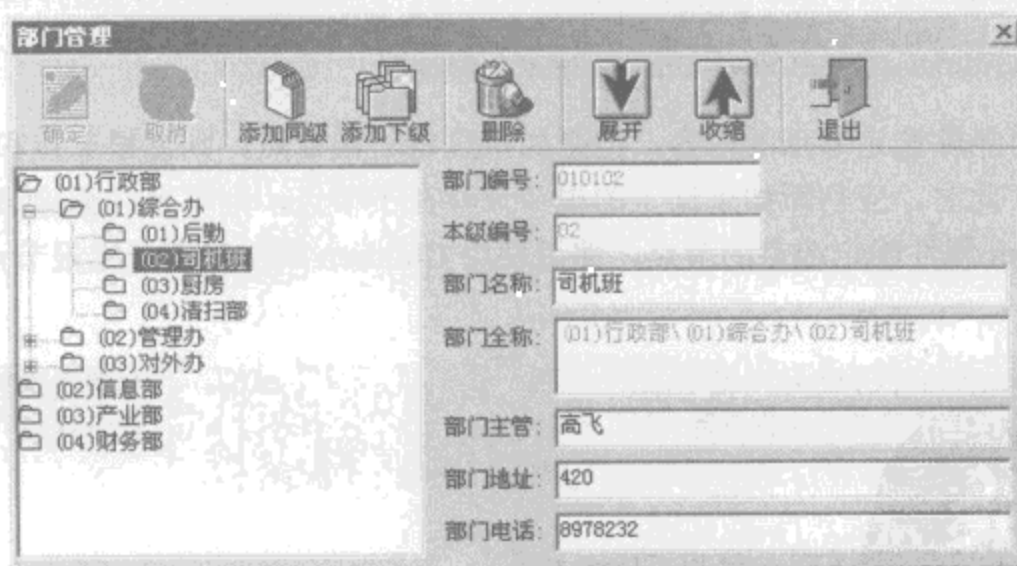


图 13.17 添加和浏览多级部门信息

具体实现步骤如下：

- (1) 新建一个工程，在该工程中会自动创建一个名为 Form1 的窗体，将该窗体命名为 main\_jbzl\_bmgj。
- (2) 使用 ToolBar 控件和 ImageList 控件制作工具栏，工具栏按钮属性设置如表 13.5 所示。

表 13.5 工具栏按钮属性设置

索引	标题	关键字	样式
1	确定	ok	0-tbrDefault
2	取消	cancel	0-tbrDefault
3			3-tbrSeparator
4	添加同级	addnew	0-tbrDefault
5	添加下级	addChild	0-tbrDefault
6			3-tbrSeparator
7	删除	del	0-tbrDefault
8			3-tbrSeparator
9	展开	expand	0-tbrDefault
10	收缩	n expand	0-tbrDefault
11			3-tbrSeparator
12	退出	exit	0-tbrDefault

(3) 在窗体上添加一个 TreeView 控件和另外一个 ImageList 控件, 名称为 imlSmallIcons (其中包含两个文件夹图片, 一个打开状态, 一个关闭状态, 关键字分别为 open 和 close), 在 TreeView 控件的属性页对话框中设置“图像列表”为 imlSmallIcons。

(4) 按照如图 13.17 所示, 添加一些 Label 控件和 TextBox 控件。

(5) 引用 ADO 对象, 用来操纵数据库。选择“工程”/“引用”命令, 在打开的“引用”对话框中选择 Microsoft ActiveX Data Objects 2.7 Library, 单击“确定”按钮。

(6) 程序关键代码如下。

定义公用变量, 代码如下。

Dim cnn As New ADODB.Connection	'定义 Connection 对象, 用于连接数据库
Dim rs1 As New ADODB.Recordset	'定义 Recordset 对象, 用于连接数据表
Dim i As Integer, bmjc As Integer	'定义两个整型变量
Dim blnTJ As Boolean	'定义一个布尔型变量, 用于判断是同级还是下级
Dim bmbh As String	'定义一个字符型变量, 用于存储部门编号

定义向 TreeView 控件中添加数据的子过程 tree\_add, 详细代码可参见光盘中的源程序。

窗体载入时, 初始化数据, 连接数据库, 代码如下。

```

Private Sub Form_Load()
    Me.Caption = "部门管理"           '设置窗体标题
    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & "\db_manpowerinfo.mdb;Persist
Security Info=False"               '连接数据库
    tree_add                          '调用向 TreeView 控件添加数据的过程
    tlbState True                     '调用 tlbState 子过程设置工具栏按钮状态
    If TreeView1.Nodes.Count > 0 Then TreeView1.Nodes(1).Selected = True '设置第一个节点内容为选中状态
End Sub

```



单击工具栏按钮，实现向数据表和 TreeView 控件添加同级、下级数据，展开和收缩 TreeView 控件节点，代码如下。

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Key
        Case "ok"
            '单击"确定"按钮
            tlbState True
            '调用 tlbState 子过程设置工具栏按钮状态
            If Len(Text1(0)) > 10 Then
                '如果"部门编号"长度大于 10
                MsgBox "部门编号超长!"
                '提示用户
            End If
            Exit Sub
            '退出子过程
        Case "addnew"
            '向数据表中添加数据
            rs1.Open "select * from 部门表", cnn, adOpenKeyset, adLockOptimistic '连接部门表
            rs1.AddNew
            '添加新记录
            For i = 0 To 6
                rs1.Fields(i) = Text1(i)
                '使用循环语句为各字段赋值
            Next i
            rs1.Fields("编码级次") = Len(Text1(0)) / 2
            '给"编码级次"字段赋值
            rs1.Update
            '更新数据表
            rs1.Close
            '关闭记录集对象
            '向 TreeView 控件中添加数据
            If TreeView1.Nodes.Count > 0 Then
                If blnTJ = True Then
                    '如果为"添加同级"状态
                    TreeView1.Nodes.Add TreeView1.SelectedItem.Key, twwLast, Text1(3), "(" & Text1(1) & ")" & Text1(2), "close"
                    '为当前选中节点的下一个节点添加数据
                    TreeView1.SelectedItem.LastSibling.Selected = True
                Else
                    '否则
                    TreeView1.Nodes.Add TreeView1.SelectedItem.Key, twwChild, Text1(3), "(" & Text1(1) & ")" & Text1(2), "close"
                    '为当前选中节点的子节点添加数据
                    '对所有节点排序
                    For i = 1 To TreeView1.Nodes.Count
                        TreeView1.Nodes(i).Sorted = True
                    Next i
                    TreeView1.SelectedItem.Child.LastSibling.Selected = True
                End If
            Else
                Set node1 = TreeView1.Nodes.Add(, twwFirst, Text1(3), "(" & Text1(1) & ")" & Text1(2))
                node1.Selected = True
            End If
        Case "cancel"
            '单击"取消"按钮
            tlbState True
            '调用 tlbState 子过程设置工具栏按钮状态
        Case "addnew"
            '单击"添加同级"按钮
            tlbState False: blnTJ = True
            '调用 tlbState 子过程同时设置标记为"添加同级"
            '清空所有文本框中的数据
            For i = 0 To Text1.UBound
                Text1(i).Text = ""
            Next i
            If TreeView1.Nodes.Count > 0 Then
                rs1.Open "select * from 部门表 where 部门全称=" & TreeView1.SelectedItem.LastSibling.Key +
```



```

"order by 编码级次", cnn, adOpenKeyset, adLockOptimistic '按部门全称查询并排序表中记录
If rs1.RecordCount > 0 Then '如果表中有记录
    Text1(1) = Format(Val(rs1.Fields("本级编号")) + 1, "00") '给"本级编号"文本框赋值
    '给"部门编号"文本框赋值
    Text1(0) = Left(rs1.Fields("部门编号"), Len(rs1.Fields("部门编号")) - 2) & Text1(1)
End If
rs1.Close '关闭数据集对象
Else
    Text1(0) = "01"
    Text1(1) = "01"
End If
Text1(2).SetFocus '部门名称"文本框获得焦点
Case "addchild" '单击"添加下级"按钮
If TreeView1.Nodes.Count = 0 Then
    MsgBox "没有上级部门，无法添加下级"
Exit Sub
End If
tlbState False: blnTJ = False '调用 tlbState 子过程同时设置标记为"添加下级"
rs1.Open "select * from 部门表 where 部门编号 like '" + Text1(0) + "' or 部门编号=" + Text1(0) +
"order by 部门编号", cnn, adOpenKeyset, adLockOptimistic
If rs1.RecordCount > 1 Then '如果表中记录大于 1
    rs1.MoveLast '将记录指针移到最后一条记录
    Text1(1) = Format(Val(rs1.Fields("本级编号")) + 1, "00") '给"本级编号"文本框赋值
    '给"部门编号"文本框赋值
    Text1(0) = Left(rs1.Fields("部门编号"), Len(rs1.Fields("部门编号")) - 2) & Text1(1)
Else
    Text1(1) = "01" '本级编号"文本框值为 01
    Text1(0) = rs1.Fields("部门编号") & Text1(1) '给"部门编号"文本框赋值
End If
rs1.Close '关闭数据集对象
'清空所有文本框中的数据
For i = 2 To Text1.UBound
    Text1(i).Text = ""
Next i
Text1(2).SetFocus '部门名称"文本框获得焦点
Case "del"
If TreeView1.SelectedItem.Children > 0 Then
    MsgBox "此部门存在下级部门，不允许删除！"
Exit Sub
End If
cnn.Execute ("delete from 部门表 where 部门全称='" + TreeView1.SelectedItem.Key + "'")
tree_add
Case "expand" '单击"展开"按钮
'使用循环语句展开所有节点
For i = 1 To TreeView1.Nodes.Count
    TreeView1.Nodes(i).Expanded = True
Next i
Case "nexpand" '单击"折叠"按钮
'使用循环语句将所有节点折叠

```

```

For i = 1 To TreeView1.Nodes.Count
    TreeView1.Nodes(i).Expanded = False
Next i
Case "exit"
    '单击"退出"按钮
    '结束程序
End
End Select
End Sub

```

## 13.5 选项卡控件 (SSTab)

 教学录像：光盘\TM\lx\13\选项卡控件 (SSTab) .exe

由于大量数据需要输入，就要不停地切换界面和使用一些控件，导致了数据输入速度过慢。这时，我们必须要在一个界面上完成这些工作，以提高工作效率，保证数据的完整性，并为用户操作提供方便。使用选项卡控件 SSTab 便可做到这一点，接下来就详细介绍选项卡控件 SSTab。

### 13.5.1 认识 SSTab 控件

选项卡控件 SSTab 位于“部件”对话框的 Microsoft Tabbed Dialog Control 6.0 (SP5) 选项中，其添加到工具箱中的图标为。

SSTab 控件提供了一组选项卡，每个选项卡都可作为其他控件的容器。在该控件中，同一时刻只有一个选项卡是活动的，这个选项卡向用户显示它本身所包含的控件而隐藏其他选项卡中的控件。

例如，在日常考勤管理界面中，将考勤记录、加班记录和出差记录放在了一个包含 3 个页面的 SSTab 控件中，如图 13.18 所示。



图 13.18 SSTab 控件在日常考勤管理中的应用

### 13.5.2 设置选项卡数目和行数

在设置 SSTab 控件中的选项卡数目之前，需要确定在 SSTab 控件中包含什么内容并如何摆放。

在设计时和运行时均可设置选项卡的数目，但在设计时设置选项卡的数目更加快捷简便。在设计时可以使用 SSTab 控件提供的“属性页”对话框来设置其相关属性，用鼠标右键单击该控件，在弹出的菜单中选择“属性”命令，即可显示该对话框，如图 13.19 所示。

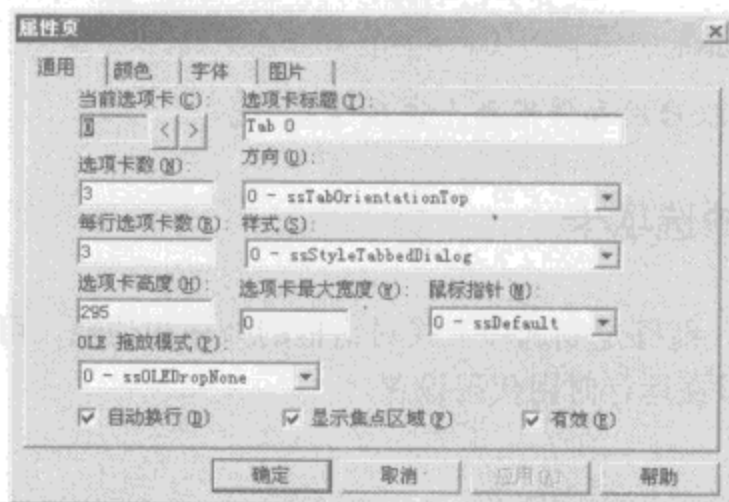


图 13.19 SSTab 控件的“属性页”

通过设置 Tab 和 TabsPerRow 属性，可以定义选项卡数和行数。例如，如果创建包含 9 个选项卡的选项卡式对话框，那么可以将“选项卡数”选项设置为“9”，将“每行选项卡数”选项设置为“3”，这样就创建了一个包含 3 行选项卡的选项卡式对话框，每行 3 个，共有 9 个选项卡。

在设置了选项卡的数目和行数后，每个选项卡就得到一个编号，该编号从零 (0) 开始，并可以被单独选定。例如，可以在“当前选项卡”选项中设置被选定的选项卡，然后改变该选项卡的标题，也就是 TabCaption 属性。

在运行时，用户可以通过单击选项卡、按〈Ctrl+Tab〉键或每个选项卡的标题中定义的热键在选项卡页之间切换。例如，如果希望创建名为“打印”的选项卡，并希望通过〈ALT+P〉键访问该选项卡，则可以将该选项卡的标题设置为&Print。

### 13.5.3 在选项卡中添加控件

SSTab 控件中的每个选项卡本质上都是其他控件的容器。使用 SSTab 控件时，应将完成相近功能的控件组合在一起，例如在“日常考勤管理”中，可以将“考勤记录”、“加班记录”和“出差记录”通过一个包含 3 个选项卡页的 SSTab 控件显示出来。确定了这些，就可以添加完成这些功能所需的控件了。

在设计时，要在某一选项卡页中添加控件，首先要单击该选项卡选中它，然后在该选项卡页中添加所需控件。

**注意：**不能用双击的方法在选项卡页中添加控件。在工具栏中双击某个控件，该控件将被放到 SSTab 控件的每一页中。

### 13.5.4 运行时启用和停用选项卡

根据应用程序的功能，以及创建的选项卡式对话框的特殊情况，可能需要在某些情况下停用某些

选项卡。可以用 `TabEnabled` 属性启用或者停用某个选项卡。当选项卡被停用时，选项卡上的文本变灰，成为不可使用的。

例如让第二个选项卡页不可用，代码如下。

```
SSTab1.TabEnabled(2) = False
```

注意：用 `Enabled` 属性可以启用和停用整个选项卡控件。

### 13.5.5 定制不同样式的选项卡

利用 `SSTab` 控件的属性，能够定制选项卡式对话框的外观和功能。可以在设计时用该控件的“属性页”设置这些属性，也可以在运行时用代码设置。

#### 1. Style 属性

用 `Style` 属性能够设置两种不同样式的选项卡式对话框。在默认情况下，`Style` 属性被设置为显示 Microsoft Office 样式的选项卡式对话框，效果如图 13.20 所示。默认情况下，被选中的选项卡的标题文本用粗体字显示。

另一种可用的样式是“Windows 95 属性页”样式的选项卡式对话框，效果如图 13.21 所示，与 Microsoft Office 样式不同，被选中的选项卡的标题不显示为粗体。

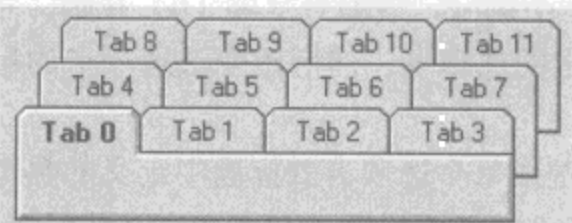


图 13.20 Microsoft Office 样式的选项卡式对话框

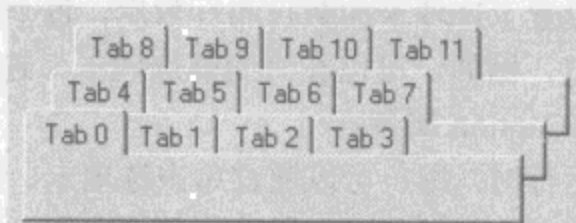


图 13.21 “Windows 95 属性页”样式的选项卡式对话框

要使用以上所显示的样式，可分别设置 `Style` 属性（样式）为 `ssStyleTabbedDialog` 或 `ssStylePropertyPage`。

#### 2. TabOrientation 属性

选项卡式对话框的选项卡可以放在它的任何一边（上、下、左、右），这是由 `TabOrientation` 属性决定的，该属性值为 `ssTabOrientationTop`、`ssTabOrientationBottom`、`ssTabOrientationLeft` 和 `ssTabOrientationRight`，分别代表将选项卡位于上边、下边、左边和右边。例如将选项卡位于左边，效果如图 13.22 所示。

如果将选项卡的位置设置为上、下以外的值时，就必须改变选项卡的字形。将选项卡放在左边或右边，都需要将文本旋转为竖直方向，在 `SSTab` 控件中，只有 TrueType 字体（也就是带 @ 符号的字体）才能够竖直显示。可以用 `Font` 属性或控件“属性页”中的“字体”选项卡，如图 13.23 所示来改变字形，改变后的效果如图 13.24 所示。

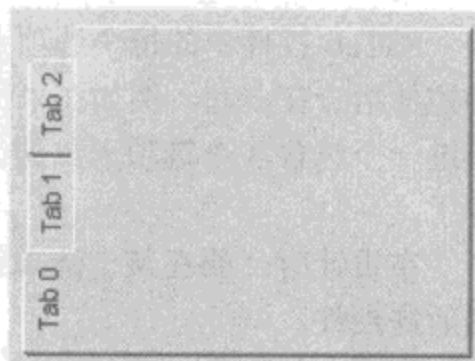


图 13.22 选项卡位于左边



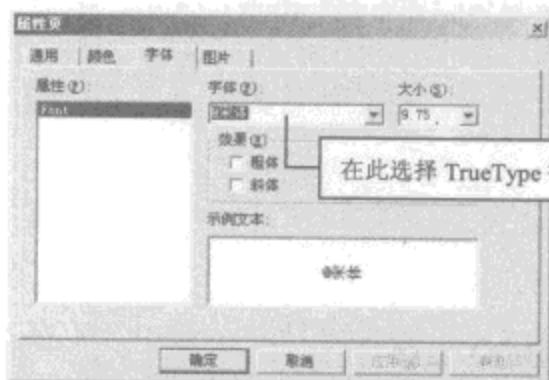


图 13.23 在“字体”选项卡中选择 TrueType 字体



图 13.24 在选项卡中显示竖直方向的文字

### 13.5.6 图形化选项卡

可以在 SSTab 控件的任何一个选项卡上添加图片（位图、图标或元文件），如图 13.25 所示的效果。

实现类似图 13.25 所示的图形化选项卡，在设计时，要为每个选项卡设置 Picture 属性，方法是：单击选项卡，然后在“属性”窗口中设置该选项卡的 Picture 属性。在运行时，可以使用 LoadPicture 函数设置 SSTab 控件的 TabPicture 属性。例如给第一个选项卡添加一个图形，代码如下。

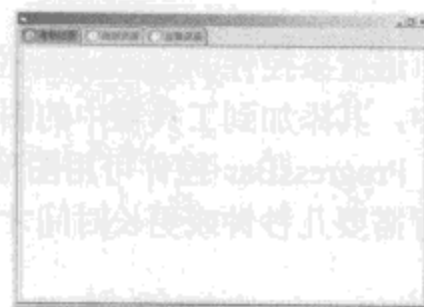


图 13.25 图形化选项卡

```
Set SSTab1.TabPicture(0) = LoadPicture(App.Path & "\image\11.bmp")
```

**例 13.11** 下面介绍实现图形化选项卡的过程。（实例位置：光盘\TM\sl\13\11）

- （1）新建一个工程，在该工程中会自动创建一个名为 Form1 的窗体。
- （2）按照前面介绍的方法，在 Form1 窗体上添加一个 SSTab 控件。
- （3）在第一个选项卡上添加一个 PictureBox 控件，设置 Picture 属性，然后复制该 PictureBox 控件分别到另外两个选项卡中。
- （4）切换到“代码窗口”，编写如下代码。

```
Private Sub Form_Load()
    SSTab1_Click (PreviousTab) '调用事件过程 SSTab1_Click
End Sub
Private Sub SSTab1_Click(PreviousTab As Integer)
    '为 3 个选项卡添加图形
    For i = 1 To 3
        Set SSTab1.TabPicture(i - 1) = LoadPicture(App.Path & "\image\" & i & ".bmp")
    Next i
    '为选定的选项卡添加另外一种图形
    Select Case SSTab1.Tab
        Case 0
            Set SSTab1.TabPicture(0) = LoadPicture(App.Path & "\image\11.bmp")
        Case 1
            Set SSTab1.TabPicture(1) = LoadPicture(App.Path & "\image\22.bmp")
        Case 2
```




```
Set SSTab1.TabPicture(2) = LoadPicture(App.Path & "image\33.bmp")
End Select
End Sub
```

## 13.6 进度条 (ProgressBar)

 教学录像：光盘\TM\lx\13\进度条 (ProgressBar) .exe

当要进行需要几秒钟或者更长时间才能完成的操作时，为用户提供可视的反馈信息，表明这个耗时的操作还要进行多长时间才能完成，这是非常有必要的。这就需要使用进度条控件 ProgressBar。

### 13.6.1 认识 ProgressBar 控件

进度条控件 ProgressBar 位于“部件”对话框的 Microsoft Windows Common Controls 6.0 (SP6) 选项中，其添加到工具箱中的图标为 。

ProgressBar 控件可用图形显示事务的进程，该控件的边框在事务进行过程中逐渐被充满。如果需要进行需要几秒钟或更长时间才能完成的操作时，就要使用 ProgressBar 控件，如图 13.26 所示。

### 13.6.2 显示进展情况

要显示某个操作的进展情况，Value 属性将持续增长，直到达到了由 Max 属性定义的最大值。这样该控件显示的填充块的数目总是 Value 属性与 Min 和 Max 属性之间的比值。例如，如果 Min 属性被设置为 1，Max 属性被设置为 100，Value 属性为 50，那么该控件将显示百分之五十的填充块，如图 13.27 所示。

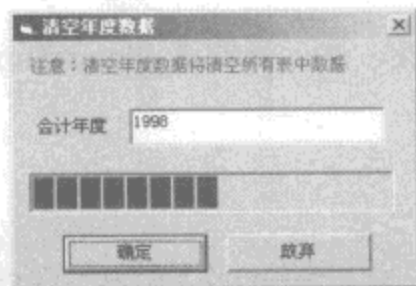


图 13.26 进度条示例



图 13.27 使用 Value、Min 和 Max 属性显示进展情况

### 13.6.3 将 Max 属性设置为已知的界限

要对 ProgressBar 控件进行编程，必须首先确定 Value 属性的界限。例如，如果正在下载文件，并且应用程序能够确定该文件有多少千字节，那么可将 Max 属性设置为这个数值。在该文件下载过程中，应用程序还必须能够确定该文件已经下载了多少千字节，并将 Value 属性设置为这个数值。

 技巧：在不能确定 Max 属性的情况下，则需要用 Animation 控件不停地显示动画，直到在结束事件中调用 Stop 方法为止。

### 13.6.4 隐藏 ProgressBar 控件

在操作开始之前通常不显示进度栏，并且在操作结束之后它应再次消失。在操作开始时，将 Visible 属性设置为 True 以显示该控件；在操作结束时，将该属性重新设置为 False 以隐藏该控件。

### 13.6.5 用 ProgressBar 控件显示清空数据的进度

通过前面的讲解，相信读者对 ProgressBar 控件已有了初步的了解，下面再通过一个实例，介绍 ProgressBar 控件在实际编程中的应用。

**例 13.12** 使用 ProgressBar 控件显示清空年度数据的进程，如图 13.28 所示，具体实现步骤如下。（实例位置：光盘\TM\sl\13\12）

(1) 新建一个工程，该工程中会自动创建一个名为 Form1 的窗体。

(2) 按照前面介绍的方法，在 Form1 窗体中添加一个 ProgressBar 控件。

(3) 在 Form1 窗体中添加 Label 控件、Text 控件和两个 CommandButton 控件。

(4) 切换到“代码窗口”，编写如下代码。

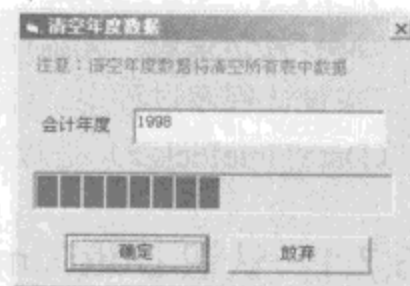


图 13.28 清空年度数据的进程

```
Dim sql As String
Dim workarea(200) As String
Dim counter As Integer '定义一个整型变量
Dim cnn As New ADODB.Connection
Private Sub Form_Load()
    Text1.Text = "1998"
    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & "\sj\" & Text1.Text &
    "\gzgl.mdb;Persist Security Info=False"
End Sub
Private Sub Command1_Click()
    ProgressBar1.Visible = True
    ProgressBar1.Max = UBound(workarea)
    ProgressBar1.Value = ProgressBar1.Min
    For counter = LBound(workarea) To UBound(workarea)
        workarea(counter) = "initial value " & counter
        ProgressBar1.Value = counter
        sql = "delete * from 部门表"
        cnn.Execute sql
        sql = "delete * from 人员表"
        cnn.Execute sql
        sql = "delete * from 人员类别表"
        cnn.Execute sql
        sql = "delete * from 工资数据表"
        cnn.Execute sql
        sql = "delete * from 工资汇总表"
        cnn.Execute sql
        sql = "delete * from 参数表"
        cnn.Execute sql
        sql = "update 数据处理状态表 set 计算标志=false,汇总标志=false"
        cnn.Execute sql
    Next counter
End Sub
```

```

sql = "update 结帐状态表 set 结帐标志=false"
cnn.Execute sql
Next counter
ProgressBar1.Visible = False
ProgressBar1.Value = ProgressBar1.Min
End Sub


```

## 13.7 日期/时间控件 (DateTimePicker)

 教学录像：光盘\TM\13\日期/时间控件 (DateTimePicker).exe

在设计程序过程中，经常需要对日期/时间进行显示、操纵、计算和读取等，此时使用日期/时间控件 (DateTimePicker) 是最为方便的。下面介绍 DateTimePicker 控件及其在程序中的应用。

### 13.7.1 认识 DateTimePicker 控件

日期/时间控件 DateTimePicker，以下称 DTPicker 控件，位于“部件”对话框的 Microsoft Windows Common Controls-26.0 (SP4) 选项中，其添加到工具箱中的图标为 。

DTPicker 控件显示日期和/或时间信息，并且可以作为一个用户用以修改日期和时间信息的界面。控件的显示包含由控件格式字符串定义的字段。当下拉 DTPicker 控件时，将会显示一个日历。

DTPicker 控件基本用途如下：

- ☒ 在需要显示使用受限制的或特殊格式字段的日期信息时，例如在某些工资表、计算住宿时间等涉及日期或时间的应用程序中。如图 13.29 所示。
- ☒ 使用户通过单击鼠标就可以选择日期而不用输入日期值。



姓名	地址	事由	日期	时间	天数	费用	押金	提醒	退房
张三	北京路100号	住宿	2008-03-21	09:00	1	100.00	100.00	2008-03-21	18:00
李四	上海路200号	住宿	2008-03-22	10:00	2	200.00	200.00	2008-03-22	19:00
王五	广州路300号	住宿	2008-03-23	11:00	3	300.00	300.00	2008-03-23	20:00

图 13.29 DTPicker 控件的典型用法

### 13.7.2 设置和返回日期

在 DTPicker 控件中当前选中的日期是由 Value 属性决定的。可以在显示该控件前（如在设计时或在 Form\_Load 事件中）设置它的 Value 属性，以便决定在控件中开始时选中哪个日期，例如下面的代码。

```
Private Sub Form_Load()
```

```
DTPicker1.Value = "2008-03-20"      '设置日期
End Sub
```

按〈F5〉键，运行程序，效果如图 13.30 所示。

如果要设置 Value 值为当前系统日期，则将前面代码中的“2008-03-20”改为“Date”。

Value 属性返回一个原始的日期值，或者空值。DTPicker 控件具有几个属性，可以返回有关显示日期的特定信息。

☒ Month 属性：返回包含当前选定日期的月份整数（1~12）。

☒ Day 属性：返回当前选定的日（1~31）。

☒ DayOfWeek 属性：返回一个值，指出所选日期是星期几，其值由 vbDayOfWeek 常量定义的值决定。

☒ Year 属性：返回包含当前选定日期的年份整数。

☒ Week 属性：返回包含当前选定日期的星期序号。

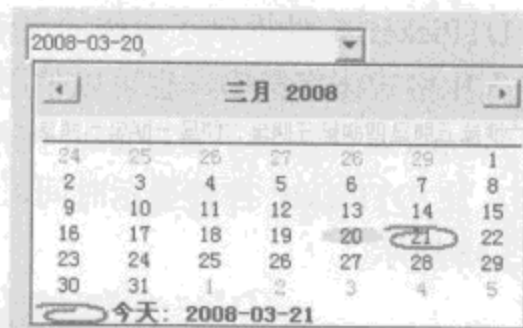


图 13.30 当前选中的日期

### 13.7.3 实时读取 DTPicker 控件中的日期

使用 DTPicker 控件的 Change 事件可以确定用户何时更改了该控件中的日期值。在该事件中使用 Value 属性，便可实时读取 DTPicker 控件中的日期。例如将实时读取的日期显示在立即窗口中，代码如下。

```
Private Sub DTPicker1_Change()
    Debug.Print DTPicker1.Value      '在立即窗口中显示日期
End Sub
```

### 13.7.4 使用 CheckBox 属性来选择无日期

使用 CheckBox 属性能够指定 DTPicker 控件是否返回日期。默认情况下，CheckBox 属性值为 False，说明 DTPicker 控件总是返回一个日期。

要让用户能够指定无日期，可以将 CheckBox 属性值设置为 True。如果 CheckBox 属性值设置为 True，那么在 DTPicker 控件日期和时间左边的编辑部分中将出现一个小的复选框。如果这个复选框没有被选中，那么 Value 属性返回一个空值；如果选中了这个复选框，那么 Value 属性返回当前显示日期。

例如，如果使用 DTPicker 控件输入软件完成日期，将 CheckBox 属性值设置为 True，效果如图 13.31 所示，此时软件完成日期为“2008-03-21”；如果软件没有完成，可以将复选框中的“√”符号去掉，效果如图 13.32 所示，此时软件完成日期为 Null。

软件完成日期: ☒ 2008-03-21

图 13.31 当前选中的日期

软件完成日期: ☐ 2008-03-21

图 13.32 返回空日期




### 13.7.5 使用日期和时间的格式

DTPicker 控件提供了很强的灵活性,以便在控件中将日期和时间的显示格式化。可以使用所有标准的 VB 格式化字符串,也可以使用回调字段来创建自定义格式。


Format 属性决定了 DTPicker 控件如何格式化原始日期值,可以从预定义的格式化选项中选择一个。例如,让 DTPicker 控件显示时间,应在“属性”窗口中设置 Format 属性为 2-dtpTime,运行效果如图 13.33 所示。



图 13.33 当前选中的时间

 **技巧:** 如果让 DTPicker 控件显示当前系统时间,可以编写代码设置 Value 属性为 Time。

CustomFormat 属性定义了用于显示 DTPicker 控件内容的格式表达式,可以通过指定格式字符串来告诉控件如何将日期输出格式化。

 **注意:** 在 CustomFormat 属性定义的日期或时间格式前,要先将 Format 属性值设置为 3-dtpCustom。

DTPicker 控件支持的格式字符串,如表 13.6 所示。

表 13.6 DTPicker 控件支持的格式字符串

格式字符串	意 义	效 果
d	1 或 2 位的日	1 日
dd	2 位日,1 位值时前加 0 (即 1 显示为“01”)	01 日
ddd	3 个字符,表示星期缩写	星期六
dddd	星期全名	星期六
h	12 小时格式 1 或 2 位小时	2 小时
hh	12 小时格式 2 位小时,有 1 位值前加 0 (即 1 显示为“01”)	02 小时
H	24 小时格式 1 或 2 位小时	1 小时
HH	24 小时格式 2 位小时,1 位值前加 0 (即 1 显示为“01”)	23 小时
m	1 或 2 位分钟	3 分钟
mm	2 位分钟,1 位值前加 0 (即 1 显示为“01”)	03 分钟
M	1 或 2 位月份	3 月
MM	2 位月份,1 位值前加 0 (即 1 显示为“01”)	03 月
MMM	3 个字符,表示月份缩写	三月
MMMM	月份全名	三月

续表

格式字符串	意 义	效 果
s	1 或 2 位的秒	5 秒
ss	2 位的秒, 1 位值前加 0 (即 1 显示为“01”)	05 秒
t	1 个字母或汉字 AM/PM (上午/下午) 的缩写 (“AM”简写为“A”, “上午”简写为“上”)	上
tt	2 个字母或汉字 AM/PM (上午/下午) 的缩写 (“AM”写为“AM”, “上午”写为“上午”)	上午
y	1 位年份 (即 2008 显示为“8”)	8 年
yy	年份的最后 2 位 (即 2008 显示为“08”)	08 年
yyy	完整的年份 (即 2008 显示为“2008”)	2008 年
X	回调字段, 使程序员可以控制显示字段, 可以使用一系列多个“X”来表示唯一的回调字段	

可以在格式字符串中添加主体文本。例如, 如果希望 DTPicker 控件按照“今天是: 2008 年 03 月 01 日星期六, 18 点 18 分 18 秒”的格式显示当前日期, 那么需要设置 CustomFormat 属性的格式字符串为: '今天是: 'yyy'年'MM'月'dd'日'dddd', 18 点 18 分 18 秒。设置完成后, 运行程序, 效果如图 13.34 所示。



图 13.34 按自定义的格式显示日期、星期和时间

注意: 主体文本必须用单引号括起来。

### 13.7.6 使用 DTPicker 控件计算日期或天数

DTPicker 控件还可以用于日期的计算和天数的计算。例如, 在宾馆客房住宿登记或退宿登记模块中自动计算客人退宿日期和住宿天数, 如图 13.35 所示。

图 13.35 客房住宿登记

计算退宿日期和住宿天数使用 DTPicker 控件非常方便。通过加减一个整数得到需要的日期; 通过两个日期控件相减得到相差的天数。公式如下。

计算天数: `d=DTPicker1.Value- DTPicker2.Value` 'd 为天数  
 计算日期: `DTPicker1.Value= DTPicker2.Value+d`

**例 13.13** 在客房住宿登记窗体中实现自动计算退宿日期和预住天数, 步骤如下: (实例位置: 光盘\TM\sl\13\13)

- (1) 新建一个工程, 在该工程中会自动创建一个名为 Form1 的窗体。
- (2) 按照前面介绍的方法, 在 Form1 窗体中添加两个 DTPicker 控件, DTPicker1 用于用户选择住宿日期, DTPicker2 用于显示退宿日期。退宿日期由系统计算得到, 因此设置该控件不可用。
- (3) 在 Form1 窗体上, 添加如图 13.35 所示的 Label、TextBox 和 CommandButton 控件。
- (4) 切换到代码窗口, 编写如下代码。

```
Private Sub DTP1_Change()  
    DTP2.Value = DTP1.Value + Val(TxtDays.Text)      '计算退宿日期  
End Sub  
Private Sub TxtDays_Change()  
    DTP2.Value = DTP1.Value + Val(TxtDays.Text)      '计算退宿日期  
End Sub
```

## 13.8 小结

本章介绍了添加 ActiveX 控件、删除 ActiveX 控件和注册 ActiveX 控件的多种方法, 这些是应用 ActiveX 控件的基础, 读者应熟练掌握。另外, 介绍常用 ActiveX 控件时, 结合了大量实例, 使读者可以更好地了解 ActiveX 控件在实际程序开发中的应用。

## 13.9 练习与实践

1. 用 TreeView 和 ListView 控件制作程序主界面。(答案位置: 光盘\TM\sl\13\14)
2. 为启动界面添加进度条。(答案位置: 光盘\TM\sl\13\15)
3. 在窗体上添加一个包含 3 个选项卡的 SSTab 控件, 当用户单击第二个选项卡时, 提示用户。(答案位置: 光盘\TM\sl\13\16)

# 第14章

## 鼠标键盘处理

(  教学录像: 32 分钟 )

鼠标键盘事件是应用程序中经常会用到的事件，窗体和大多数控件都能响应鼠标和键盘事件。例如，窗体和图像控件等能够检测鼠标指针位置，判断按下的是哪个鼠标键；利用键盘事件，可以响应多种键盘操作，也可以解释和处理 ASCII 字符。本章将分别介绍鼠标、键盘处理技术的相关知识。

通过阅读本章，您可以：

- » 学会如何设置鼠标指针形状
- » 学会如何将鼠标指针设置为指定的图片
- » 学会将鼠标指针设置为指定的动画
- » 掌握鼠标的单击、双击事件
- » 掌握鼠标的按下和抬起事件
- » 掌握鼠标的移动事件
- » 掌握鼠标拖放技术
- » 掌握键盘按下和抬起操作
- » 学会 KeyPress 事件



## 14.1 鼠标指针的设置

 教学录像：光盘\TM\14\鼠标指针的设置.exe

在 VB 中可以通过设置改变鼠标指针的样式，可以设置鼠标指针的形状、设置鼠标为指定的图片和动画。下面介绍具体的设置方法。

### 14.1.1 设置鼠标指针形状

通过设置控件的 `MousePointer` 属性可以定义当鼠标指针指向该控件时显示的形状。`MousePointer` 属性返回一个值，该值用于显示鼠标指针的类型。

语法：

`object.MousePointer [= value]`

`object`：对象表达式。

`value`：整数。`value` 设置值如表 14.1 所示。

表 14.1 value 的值

常 数	值	说 明
<code>vbDefault</code>	0	(默认值) 形状由对象决定
<code>vbArrow</code>	1	箭头
<code>vbCrosshair</code>	2	十字线
<code>vbIbeam</code>	3	I 型
<code>vbIconPointer</code>	4	图标 (矩形内的小矩形)
<code>vbSizePointer</code>	5	尺寸线 (指向东、南、西、北 4 个方向的箭头)
<code>vbSizeNESW</code>	6	右上、左下尺寸线 (指向东北和西南方向的双箭头)
<code>vbSizeNS</code>	7	垂直尺寸线 (指向南和北的双箭头)
<code>vbSizeNWSE</code>	8	左上、右下尺寸线 (指向东南和西北方向的双箭头)
<code>vbSizeWE</code>	9	水平尺寸线 (指向东和西两个方向的双箭头)
<code>vbUpArrow</code>	10	向上的箭头
<code>vbHourglass</code>	11	沙漏 (表示等待状态)
<code>vbNoDrop</code>	12	不允许放下
<code>vbArrowHourglass</code>	13	箭头和沙漏
<code>vbArrowQuestion</code>	14	箭头和问号
<code>vbSizeAll</code>	15	四向尺寸线
<code>vbCustom</code>	99	通过 <code>MouseIcon</code> 属性所指定的自定义图标

例如，当鼠标指向窗体上的按钮时，显示沙漏图标，只须将该按钮的 `MousePointer` 属性值设置为 11。当程序运行时鼠标指针放置在该按钮控件上时，鼠标指针变为沙漏样式，如图 14.1 所示。

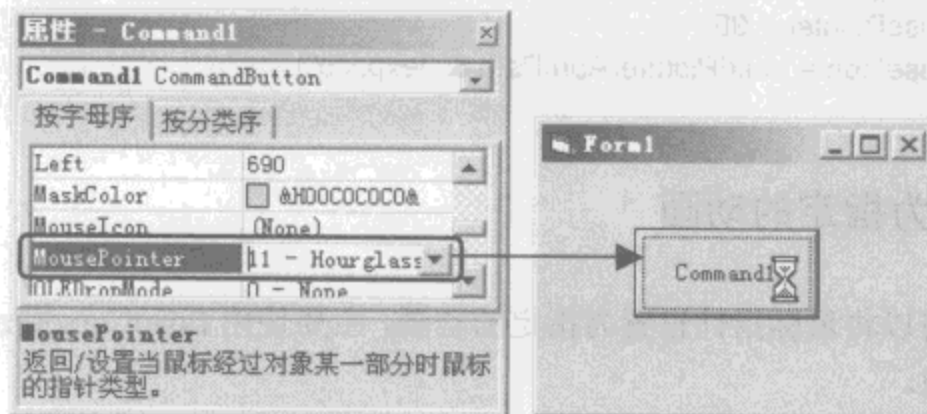


图 14.1 设置按钮控件的 `MousePointer` 属性

也可以使用代码实现上面的显示效果，代码如下：

```
Private Sub Form_Load()  
    Command1.MousePointer = 11  
End Sub
```

### 14.1.2 设置鼠标为指定的图片

通过将控件的 `MousePointer` 属性值设置为 99-Custom，然后通过控件的 `MouseIcon` 属性选择指定的图片，可以将鼠标指针定义为指定的图片。

`MousePointer` 属性用于返回或设置控件中自定义的鼠标指针图标。

语法：

```
object.MouseIcon = LoadPicture(pathname)
```

object: 对象表达式。

pathname: 字符串表达式，指定包含自定义图标文件的路径和文件名。

例如，将窗体上按钮的 `MousePointer` 属性值设置为 99，然后通过 `MouseIcon` 属性选择所需的图片。这样运行程序后，当鼠标放置在窗体中的按钮上时，鼠标指针显示为所选择图片的样式，效果如图 14.2 所示。

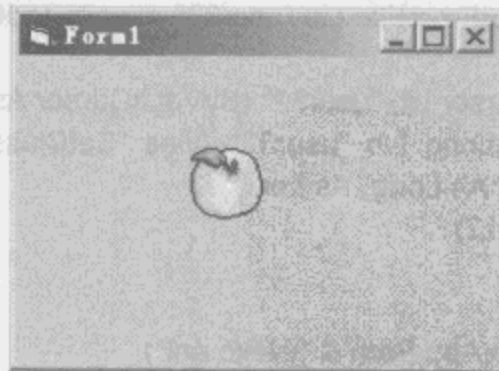


图 14.2 设置鼠标为指定的图片

也可以通过代码设置控件属性。将鼠标指针设置为指定图片，代码如下：

```
Private Sub Form_Load()
    Command1.MousePointer = 99
    Command1.MouseIcon = LoadPicture(App.Path & "\exp.ico")
End Sub
```

### 14.1.3 设置鼠标为指定的动画

通过 API 函数还可以将鼠标指针设置为指定的动画，主要使用 LoadCursorFromFile、DestroyCursor 和 SetClassLong 来实现。

在使用 API 函数前要先进行声明，这几个函数的声明语句如下：

```
Private Declare Function LoadCursorFromFile Lib "user32" Alias "LoadCursorFromFileA" (ByVal lpFileName As String) As Long
Private Declare Function DestroyCursor Lib "user32" (ByVal hCursor As Long) As Long
Private Declare Function SetClassLong Lib "user32" Alias "SetClassLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
```

**例 14.1** 本实例实现将鼠标指针设置为动画的样式。程序运行时，当鼠标放置在窗体上时，鼠标指针显示为设置的动画。程序运行效果如图 14.3 所示。（实例位置：光盘\TM\sl\14\1）

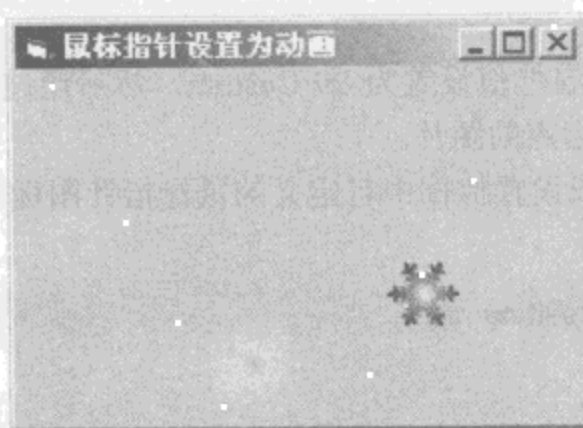


图 14.3 将鼠标指针设置为动画

程序代码如下：

```
'声明 API 函数
Private Declare Function LoadCursorFromFile Lib "user32" Alias "LoadCursorFromFileA" (ByVal lpFileName As String) As Long
Private Declare Function DestroyCursor Lib "user32" (ByVal hCursor As Long) As Long
Private Declare Function SetClassLong Lib "user32" Alias "SetClassLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
Private Const GCL_HCURSOR = (-12)
Dim AniCur As Long
Private Sub Form_Load()
    AniCur = LoadCursorFromFile(App.Path & "\Neko.ani")
    SetClassLong Me.hwnd, GCL_HCURSOR, AniCur
End Sub
Private Sub Form_Unload(Cancel As Integer)
```

'声明变量  
'将加载的文件赋给变量  
'显示动画文件

```
DestroyCursor AniCur  
End Sub
```

'卸载动画文件

## 14.2 鼠标事件的响应

 教学录像：光盘\TM\lx\14\鼠标事件的响应.exe

用户操作鼠标引发鼠标事件，对鼠标的动作作出反应就是对鼠标事件的响应。鼠标事件有 Click、DblClick、MouseDown、MouseUp 和 MouseMove 事件。下面分别进行介绍。

### 14.2.1 鼠标单击和双击（Click 事件和 DblClick 事件）

#### （1）Click 事件

鼠标的 Click 事件是在用鼠标单击某个对象时发生的。前面的章节中在使用实例进行讲解时已经多次用到了各个对象的 Click 事件。下面将详细介绍 Click 事件的相关知识。

语法格式如下：

```
Private Sub object_Click([index As Integer])
```

object：一个对象表达式。

index：一个整数，用来唯一标识一个在控件数组中的控件。

 注意：对于在控件上发生的 Click 事件，要注意以下几点：

- ① 对于 CheckBox 控件、CommandButton 控件、ListBox 控件或 OptionButton 控件来说，Click 事件仅当单击鼠标左键时发生。
- ② 当窗体带有 Default 属性设置为 True 的 CommandButton 控件时，按下〈Enter〉键触发 Click 事件。
- ③ 当窗体带有 Cancel 属性设置为 True 的 CommandButton 控件时，按下〈Esc〉键触发 Click 事件。

例如，当程序运行时，单击 CommandButton 控件，触发该控件的 Click 事件，就会执行 Click 事件中的代码，这样就是响应了鼠标的单击事件。

#### （2）DblClick 事件

当在某个对象上双击鼠标时触发 DblClick 事件，即对鼠标按下和释放并再按下和释放鼠标按键。

语法：

```
Private Sub object_DblClick (index As Integer)
```

object：对象表达式。

index：如果控件在控件数组内，则这个 index 值用来标识该控件。



## 14.2.2 鼠标按下和抬起 (MouseDown 事件和 MouseUp 事件)

### (1) MouseDown 事件

当按下鼠标按钮时触发 MouseDown 事件, MouseDown 事件是鼠标事件中非常常用的事件。语法格式如下:

```
private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

Button: 返回一个整数, 该整数用于标识按下的鼠标键是哪一个键 (左键、右键或中间键)。使用表 14.2 中所列的常数对该参数进行测试, 从而判断按下的鼠标键。

表 14.2 Button 常数按钮值

常 数	值	说 明
vbLeftButton	1	左键被按下
vbRightButton	2	右键被按下
vbMiddleButton	4	中键被按下

Shift: 返回一个整数, 在 Button 参数指定的键被按下或者被释放的情况下, 该整数对应于〈Shift〉、〈Ctrl〉和〈Alt〉键的状态。

X, Y: 返回一个鼠标指针的当前位置。

### (2) MouseUp 事件

当用户在对象上释放鼠标按钮时发生 MouseUp 事件。

语法:

```
private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

其参数说明与 MouseDown 语句参数说明相同。

**例 14.2** 本实例实现当程序运行时, 单击窗体上的按钮, 当在按钮上按下鼠标按钮时按钮标题显示“按钮被按下”; 当释放鼠标按钮时, 按钮标题显示“按钮被释放”。效果如图 14.4 和图 14.5 所示。  
(实例位置: 光盘\TM\sl\14\2)



图 14.4 当按下鼠标按钮时

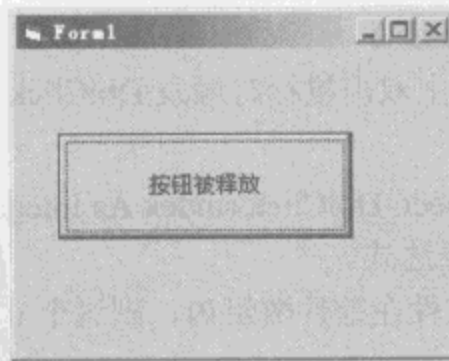


图 14.5 当释放鼠标按钮时

程序代码如下:

```

'鼠标按键被按下时的事件响应
Private Sub Command1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Command1.Caption = "按钮被按下"
End Sub
'鼠标按键被释放时的事件响应
Private Sub Command1_Mouseup(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Command1.Caption = "按钮被释放"
End Sub

```

### 14.2.3 鼠标移动 (MouseMove 事件)

当用户在窗体或控件上移动鼠标时触发 MouseMove 事件。只要鼠标位置在对象的边界范围内, 该对象就能接收鼠标的 MouseMove 事件。

语法:

```
private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

**例 14.3** 本实例实现当程序运行时, 当鼠标移动到窗体上标签位置上时, 该标签突出显示。主要是通过鼠标移动事件实现鼠标移动到标签上时, 标签的 BorderStyle 属性值赋值为 1, 显示在窗体上就是凹下去的效果。程序运行效果如图 14.6 所示。(实例位置: 光盘\TM\sl\14\3)

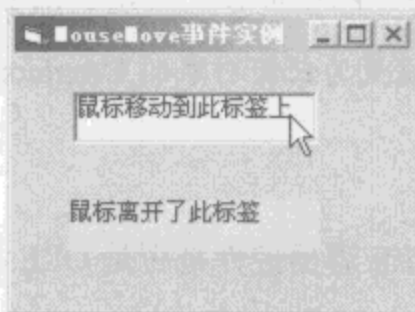


图 14.6 鼠标移动事件实例界面

程序代码如下:

```

Private Sub Label1_MouseMove(index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    Label1(index).BorderStyle = 1          '为当前鼠标所在标签的 BorderStyle 属性值赋值为 1
    Label1(index).Caption = "鼠标移动到此标签上" '设置当前标签标题
End Sub
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    For i = 0 To Label1.UBound
        Label1(i).BorderStyle = 0          '当鼠标不在标签上移动时, 将标签的 BorderStyle 属性值赋值为 0
        Label1(i).Caption = "鼠标离开了此标签" '设置当前标签标题
    Next i
End Sub

```

### 14.2.4 鼠标拖放 (OLE 拖放操作)


鼠标的拖放操作是在 Windows 操作系统下经常使用的操作, 例如将一个文件拖动放到另一个文件夹中。在 VB 应用程序中, 可以通过 OLE 拖放技术实现在控件和控件之间、控件和其他 Windows 应用程

序之间拖动文本和图形的功能。在使用 OLE 拖放技术时,并不是把一个控件拖动到另一个控件并调用代码,而是将数据从一个控件或应用程序移动到另一个控件或应用程序当中。例如,可以选择并拖动 Excel 中的一个单元范围,然后将它们放到应用程序的 DataGrid 控件上。下面介绍与拖放技术有关的属性、事件和方法。

#### (1) DragMode 属性


DragMode 属性返回或设置一个值,确定在拖放操作中所用的是手动还是自动拖动方式。当 DragMode 属性值设置为 1 时,则启用自动拖动模式。

当用户在源对象上按下鼠标左键同时拖动鼠标,对象的图标便随鼠标指针移动到目标对象上,当释放鼠标时在目标对象上产生 DragDrop 事件。

 **注意:** 如果仅将控件的 DragMode 属性值设置为 1,当程序运行时拖动控件,控件的图标随着鼠标移动,但当释放鼠标后对象本身并不会移动到新的位置上或被加到目标对象中。在目标对象的 DragDrop 事件中进行程序设计才能实现真正的拖放。

#### (2) DragIcon 属性

DragIcon 属性用于指定在拖动过程中显示的对象的图标。在拖动对象时,并不是对象在移动,而是移动对象的图标。当对对象的 DragIcon 属性进行设置后,拖动对象时显示的拖动图标为设置的图片,释放对象后恢复为原来的对象样式。可以在属性窗口进行设置,也可以在程序中进行赋值。如果 DragIcon 属性值为空,则在拖动控件时,随鼠标移动的是被拖动控件的边框。

 **注意:** 运行时,DragIcon 属性可以设置为任何对象的 DragIcon 或 Icon 属性,或者可以用 LoadPicture 函数返回的图标给它赋值。

#### (3) OLEDropAllowed 属性


OLEDropAllowed 属性用于决定 OLE 容器控件是否是 OLE 拖放操作的放目标。

语法:

```
object.OLEDropAllowed [= boolean]
```

object: 对象表达式。

Boolean: 布尔表达式,确定 OLE 容器控件是否为放目标。如果将设置值设置为 True,则在拖动链接对象或内嵌对象时,当鼠标指针在 OLE 容器控件上移动时会出现放图标;如果将设置值设置为 False,则拖动链接或嵌入对象时,在 OLE 容器控件上将不出现放图标。

 **说明:** 如果 OLEDropAllowed 属性设为 True,则拖动对象时,OLE 容器控件不接收 DragDrop 或 DragOver 事件。同样,当 OLEDropAllowed 属性设为 True 时,DragMode 属性的设置对 OLE 容器控件的拖放动作没有影响。

#### (4) Drag 方法

Drag 方法用于控件的开始、结束或取消拖动操作,但是除了 Line、Menu、Shape、Timer 和 CommonDialog 控件。仅当控件的 DragMode 属性值为 0 且采用手工拖放时,用该方法来实现控件的拖放操作。

语法:

object.Drag action

object: 一个对象表达式。

action: 一个可选的常数或数值。其设置值如表 14.3 所示。

表 14.3 Drag 方法 Action 参数设置值

常 数	值	说 明
VbCancel	0	取消拖动操作
vbBeginDrag	1	开始拖动 object
VbEndDrag	2	结束拖动 object

#### (5) DragDrop 事件

DragDrop 事件在一个完整的拖放动作完成或使用 Drag 方法, 并将其 action 参数设置为 2 (Drop) 时, 该事件发生。前面介绍的拖放属性和拖放方法都是作用在源对象上的, 而 DragDrop 事件是发生在目标对象上的。

语法:

```
Private Sub Form_DragDrop(source As Control, x As Single, y As Single)
Private Sub MDIForm_DragDrop(source As Control, x As Single, y As Single)
Private Sub object_DragDrop([index As Integer,]source As Control, x As Single, y As Single)
```

参数说明如表 14.4 所示。

表 14.4 DragDrop 事件语法参数说明

参 数	说 明
object	一个对象表达式
index	一个整数, 用来唯一地标识一个在控件数组中的控件
source	正在被拖动的控件。可用此参数将属性和方法包括在事件过程中
x, y	指定当前鼠标指针在目标窗体或控件中水平 (x) 和垂直 (y) 位置的坐标值。该坐标系是通过子 ScaleHeight、ScaleWidth、ScaleLeft 和 ScaleTop 属性进行设置的

例 14.4 本实例实现通过控件的 DragDrop 事件实现控件在窗体上动态移动的功能, 程序运行效果如图 14.7 和图 14.8 所示。(实例位置: 光盘\TM\sl\14\4)

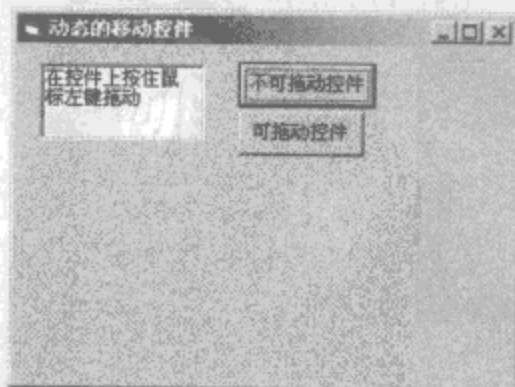


图 14.7 拖动前各控件位置

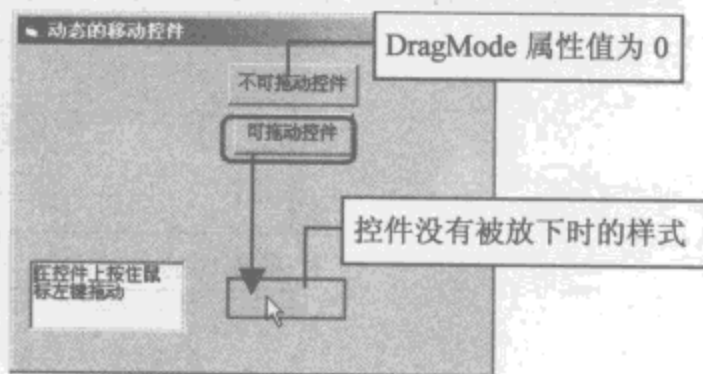


图 14.8 拖动的效果



程序代码如下：

```
Private Sub Form_DragDrop(Source As Control, X As Single, Y As Single)
    Source.Move (X - Source.Width / 2), (Y - Source.Height / 2)
End Sub
```

'窗体的 DragDrop 事件  
'控件放下位置

#### (6) DragOver 事件

在当源对象被拖动到某个对象上时，在该对象上便引发 DragOver 事件。可用此事件对鼠标指针在一个有效目标上的进入、离开或停顿等进行监控。鼠标指针的位置决定接收此事件的目标对象。

语法：

```
Private Sub Form_DragOver(source As Control, x As Single, y As Single, state As Integer)
Private Sub MDIForm_DragOver(source As Control, x As Single, y As Single, state As Integer)
Private Sub object_DragOver([index As Integer,]source As Control, x As Single, y As Single, state As Integer)
```

DragOver 事件语法中各参数的说明如表 14.5 所示。

表 14.5 DragOver 事件语法参数说明

参 数	说 明
object	一个对象表达式
index	一个整数，用来唯一标识控件数组中的控件
source	正在被拖动的控件。可用此参数在事件过程中引用各属性和方法
x, y	是一个指定当前鼠标指针在目标窗体或控件中水平 (x) 和垂直 (y) 位置的坐标值。该坐标系是通过 ScaleHeight、ScaleWidth、ScaleLeft 和 ScaleTop 属性而设置的
state	是一个整数，它对应于一个控件的转变状态 0: 进入 (源控件正被向一个目标范围内拖动) 1: 离去 (源控件正被向一个目标范围外拖动) 2: 跨越 (源控件在目标范围内从一个位置移到了另一位置)

注意：应使用 DragMode 属性和 Drag 方法指定开始拖动的方式。

例 14.5 本实例实现将一个文本框内选中的内容拖动到另一个文本框中。在程序运行时，在“源文件”文本框内选择内容，然后按住鼠标左键拖动到“目标文件”文本框中，即可将选择的内容拖动到“目标”文本框中。运行效果如图 14.9 所示。（实例位置：光盘\TM\sl\14\5）

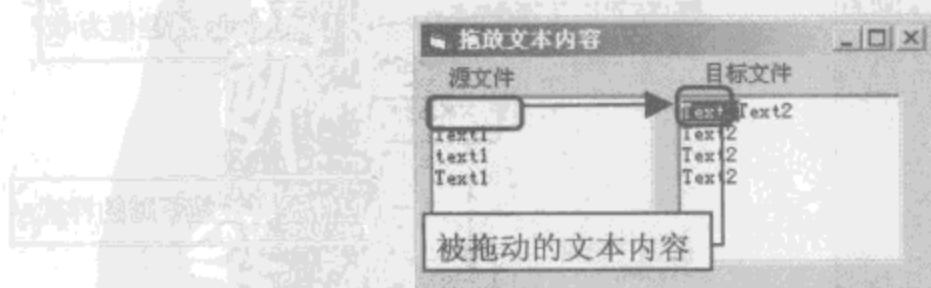


图 14.9 文本内容拖放


程序代码如下：

```
Private Sub Form_Load()
```

```
Text1.OLEDragMode = 1: Text2.OLEDropMode = 2
```

```
End Sub
```

'设置文本框的 OLEDragMode 属性

 说明: OLEDragMode 属性用于返回或设置目标部件如何处理放操作。当该属性值为 0 时, 目标部件不接受 OLE 放操作, 并且显示 No Drop 图标; 当属性值为 1 时, 为人工方式, 目标部件触发 OLE 放事件, 允许程序员用代码处理 OLE 放操作; 当属性值为 2 时, 为自动方式, 如果拖动对象包含目标部件能识别的格式的数据, 则自动接受 OLE 放操作; 当 OLEDropMode 设置为 2 时, 在目标上鼠标事件和 OLE 拖放事件都不会发生。

## 14.3 键盘事件的响应

 教学录像: 光盘\TM\lx\14\键盘事件的响应.exe

键盘事件也是编程中经常能够用到的事件, 在键盘上每个按键执行的每个动作都是一个键盘事件。对键盘按键动作作出反应就是键盘事件的响应。键盘事件有 3 种, 分别为 KeyDown 事件、KeyUp 事件和 KeyPress 事件。下面分别进行介绍。

### 14.3.1 ASCII 码

ASCII 码是美国标准信息交换码 (American Standard Code for Information Interchange) 的缩写。键盘上的按键与 ASCII 码值相对应, 通过读取或设置键盘的 ASCII 码值可以实现控制键盘的相关操作。常用 ASCII 字母键码如表 14.6 所示。读者可以对 ASCII 键码进行查阅。

### 14.3.2 KeyDown 事件和 KeyUp 事件的使用

当焦点置于某对象上时, 按下键盘上的键, 便会对相应的对象引发 KeyDown 事件, 释放按键便引发相应对象的 KeyUp 事件。VB 中的大部分控件都能接收这两个事件。

#### 1. KeyDown 事件

KeyDown 事件在窗体具有焦点且在键盘上按下一个键时被触发。其语句格式如下:

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
```

KeyCode: 该参数用来返回一个键码。键码将键盘上的物理按键与一个数值相对应, 并定义了对应的键码常数。详细的键码常数表参见 VB 帮助文件。

Shift: 该参数是用来响应〈Shift〉、〈Ctrl〉和〈Alt〉键状态的一个整数。如果需要测试 Shift 参数, 可使用该参数中定义的各位 Shift 常数。该常数值及其说明如表 14.7 所示。

表 14.6 常用 ASCII 字母键码表

十进制	字符	代码	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符
0	空	NUL	32	(space)	64	@	96	.	128	C	160	á	192	À	224	À
1	文头	SOH	33	!	65	A	97	a	129	ü	161	í	193	Á	225	Á
2	正文开始	STX	34	"	66	B	98	b	130	é	162	ó	194	Â	226	Â
3	正文结束	ETX	35	#	67	C	99	c	131	à	163	ù	195	Ã	227	Ã
4	文尾	EOT	36	\$	68	D	100	d	132	ä	164	ñ	196	Ä	228	Ä
5	查询	ENQ	37	%	69	E	101	e	133	å	165	ñ	197	Å	229	Å
6	确认	ACK	38	&	70	F	102	f	134	ä	166	ñ	198	Æ	230	Æ
7	振铃	BEL	39	'	71	G	103	g	135	ç	167	ñ	199	Ç	231	Ç
8	backspace	BS	40	(	72	H	104	h	136	è	168	ñ	200	È	232	È
9	水平制表符	HT	41	)	73	I	105	i	137	é	169	ñ	201	É	233	É
10	换行	LF	42	*	74	J	106	j	138	ê	170	ñ	202	Ê	234	Ê
11	起始	VT	43	+	75	K	107	k	139	ë	171	ñ	203	Ë	235	Ë
12	换页	FF	44	,	76	L	108	l	140	ï	172	ñ	204	Ï	236	Ï
13	回车	CR	45	-	77	M	109	m	141	ï	173	ñ	205	Ï	237	Ï
14	移出	SO	46	.	78	N	110	n	142	Ë	174	ñ	206	Ï	238	Ï
15	移入	SI	47	/	79	O	111	o	143	Ä	175	ñ	207	Ï	239	Ï
16	数据链路转意	DLE	48	0	80	P	112	p	144	É	176	ñ	208	Ï	240	Ï
17	设备控制 1	DC1	49	1	81	Q	113	q	145	æ	177	ñ	209	Ï	241	Ï
18	设备控制 2	DC2	50	2	82	R	114	r	146	æ	178	ñ	210	Ï	242	Ï
19	设备控制 3	DC3	51	3	83	S	115	s	147	ø	179	ñ	211	Ï	243	Ï
20	设备控制 4	DC4	52	4	84	T	116	t	148	ø	180	ñ	212	Ï	244	Ï
21	反确认	NAK	53	5	85	U	117	u	149	ø	181	ñ	213	Ï	245	Ï
22	同步空闲	SYN	54	6	86	V	118	v	150	ù	182	ñ	214	Ï	246	Ï
23	传输块结束	ETB	55	7	87	W	119	w	151	ù	183	ñ	215	Ï	247	Ï
24	取消	CAN	56	8	88	X	120	x	152	ý	184	ñ	216	Ï	248	Ï
25	媒体结束	EM	57	9	89	Y	121	y	153	Û	185	ñ	217	Ï	249	Ï
26	替换	SUB	58	:	90	Z	122	z	154	Ü	186	ñ	218	Ï	250	Ï
27	转意	ESC	59	;	91	[	123	[	155	Ü	187	ñ	219	Ï	251	Ï
28	文件分隔符	FS	60	<	92	\	124	\	156	£	188	ñ	220	Ï	252	Ï
29	组分隔符	GS	61	=	93	]	125	]	157	¥	189	ñ	221	Ï	253	Ï
30	记录分隔符	RS	62	>	94	^	126	^	158	ß	190	ñ	222	Ï	254	Ï
31	单元分隔符	US	63	?	95	_	127	_	159	ü	191	ñ	223	Ï	255	Ï

表 14.7 Shift 属性的常数值

常 数	值	说 明
vbShiftMask	1	〈Shift〉键的位屏蔽
vbCtrlMask	2	〈Ctrl〉键的位屏蔽
vbAltMask	4	〈Alt〉键的位屏蔽

注意：为了在每个控件识别其所有键盘事件之前使窗体接收这些键盘事件，需要将窗体上的 KeyPreview 属性设置为 True。

例 14.6 本实例实现当程序运行时，光标在文本框中，按〈Enter〉键，光标从文本框上移动到按钮上。程序代码如下：（实例位置：光盘\TM\sl\14\6）

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = 13 Then
        Command1.SetFocus
    End If
End Sub
```

'文本框的 KeyDown 事件  
'如果按下的按键为回车键  
'将焦点移动到按钮上

## 2. KeyUp 事件

KeyUp 事件在当窗体上具有焦点时释放一个键或者将窗体的 KeyPreview 属性设置为 True 释放一个键时触发。其语句格式如下：

```
Private Sub Form_Keyup(KeyCode As Integer, Shift As Integer)
```

KeyCode：一个键代码，诸如 vbKeyF1（〈F1〉键）或 vbKeyhome（〈Home〉键）。须指定键代码。

Shift：是在该事件发生时响应表示〈Shift〉、〈Ctrl〉和〈Alt〉键状态的一个整数。Shift 参数是一个位域，它用最少的位响应 Shift 键（位 0）、〈Ctrl〉键（位 1）和 Alt 键（位 2）。这些位分别对应于值 1、2 和 4。可通过对一些、所有或无位的设置来指明有一些、所有或零个键被按下。

例 14.7 本实例实现程序运行时按下键盘上的〈Enter〉键，窗体上显示“〈Enter〉键被按下”，当释放〈Enter〉键时，窗体上显示“〈Enter〉键被释放”。程序运行效果如图 14.10 所示。（实例位置：光盘\TM\sl\14\7）

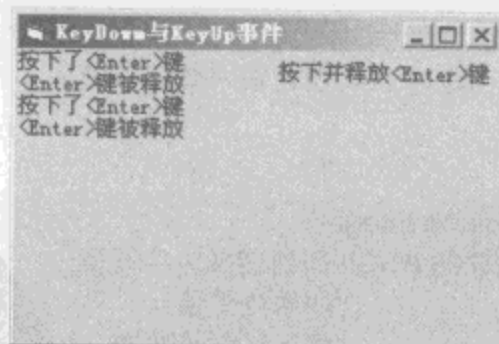


图 14.10 KeyDown 与 KeyUp 事件实例

程序代码如下：

```
Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
    If KeyCode = 13 Then
```

'如果键码值为 13



```

Print "<Enter>键被释放"      '在窗体上输出信息
End If
End Sub

```

### 14.3.3 KeyPress 事件的使用

当窗体中没有可视和有效的控件或 KeyPreview 属性被设置为 True 时,用户在按下和释放一个有 ASCII 码相对应的键时触发 KeyPress 事件。

KeyPress 事件并不一定按下键盘上任意一个键都会被触发,它只会对产生 ASCII 码值的按键进行响应。所以,对于像方向键和功能键这样没有 ASCII 码值的按键,KeyPress 事件不会被触发。

语法:

```
Private Sub Form_KeyPress(KeyAscii As Integer)
```

KeyAscii: 是返回一个标准数字 ANSI 键代码的整数。KeyAscii 通过引用传递,对它进行改变可给对象发送一个不同的字符。将 KeyAscii 改变为 0 时可取消按键,这样一来对象便接收不到字符。

⚠ 注意: KeyPress 与 KeyDown、KeyUp 事件有所区别,KeyPress 不显示键盘的物理状态,而只是传递一个字符。

KeyPress 事件中的参数 KeyAscii 用来接收被敲击键的 ASCII 码。可以通过该参数的返回值判断哪个键进行了敲击。KeyPress 事件将每个字符的大、小写形式作为不同的键代码解释,即作为两种不同的字符。

例 14.8 本实例实现当程序运行时在窗体上的文本框内输入内容,不能输入数值型信息。程序运行效果如图 14.11 所示。(实例位置:光盘\TM\sl\14\8)

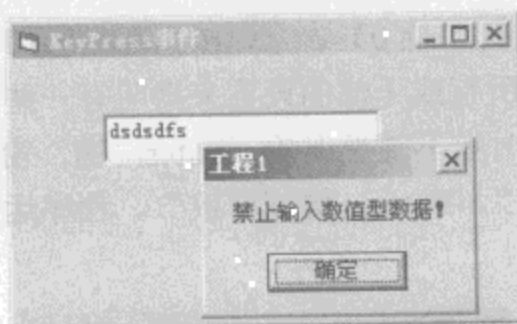


图 14.11 禁止输入数值型信息

程序代码如下:

```

Private Sub Text1_KeyPress(KeyAscii As Integer)
    char = Chr(KeyAscii)      '将按键的 ASCII 码值转换为字符型
    If IsNumeric(char) Then   '如果是数值
        KeyAscii = char      '将字符赋给参数
        MsgBox "禁止输入数值型数据!" '提示信息
    End If
End Sub

```

## 14.4 小结

本章主要介绍了鼠标键盘的基本属性、事件和方法。通过本章的学习，读者可以掌握处理鼠标和键盘事件的方法，在开发应用程序过程中熟练应用鼠标与键盘事件，可以增强应用程序的实用性。

## 14.5 练习与实践

1. 在窗体上添加一个标签控件，编写代码，使程序运行时：按下鼠标左键，标签变为绿色；释放鼠标时，标签变为蓝色。（答案位置：光盘\TM\s\14\9）
2. 设计一个程序实现当程序运行时在文本框中输入字母，输入的小写字母转换为大写字母；输入的大写或非字母字符，保持不变。（使用 Ucase 函数将小写字母转换为大写字母。）（答案位置：光盘\TM\s\14\10）

## 练习 4.4

本练习旨在帮助学生理解本章内容，并掌握本章所介绍的翻译技巧。学生应仔细阅读本章内容，并尝试将所学技巧应用于实际翻译中。练习内容如下：

## 练习 4.4

1. 将下列句子翻译成中文，并说明所使用的翻译技巧。  
(e.g. 词性转换、增补、删减、重组等)
2. 将下列句子翻译成英文，并说明所使用的翻译技巧。  
(e.g. 词性转换、增补、删减、重组等)
3. 将下列句子翻译成中文，并说明所使用的翻译技巧。  
(e.g. 词性转换、增补、删减、重组等)



# 第15章

## 程序调试和错误处理

(教学录像: 17 分钟)

在程序设计时, 出现错误是在所难免的, 越复杂的程序越容易产生错误, 有些错误还会导致程序无法正常运行。因此程序的调试工作就尤为重要, 在设计程序时要通过不断的测试来捕捉错误。VB 提供了强大的调试工具, 可以方便地进行程序调试。另外, 还可以通过 VB 有关错误处理语句对产生的错误进行处理。

通过阅读本章, 您可以:

- » 了解 3 种错误类型
- » 掌握错误产生的主要原因
- » 了解 3 种工作模式
- » 掌握 3 种工作模式的切换方法
- » 掌握调试工具的使用方法
- » 掌握插入断点和逐语句跟踪的排错方法
- » 掌握 Err 对象的使用方法
- » 掌握 On Error 语句的使用方法
- » 学会如何退出错误处理



## 15.1 错误类型

 教学录像：光盘\TM\lx\15\错误类型.exe

VB 捕获的错误有很多，在编写代码或程序调试运行中，如果捕获到错误信息，屏幕上将显示错误信息提示对话框，VB 中的这些错误信息按照类型大致可以分为编译错误、运行错误与逻辑错误 3 种类型。

### 15.1.1 编译错误

编译错误发生在代码编辑时，也可称为语法错误，是由于代码的结构违反了语句的语法规定而产生的错误。例如，遗漏了标点符号或关键字就属于编译错误。当出现编译错误时系统将出错行的代码变成红色高亮度显示，并提示错误原因。

通常情况下可以通过设置“自动语法检测”来使系统自动检测语法错误。当一行代码输入完后，光标转移到其他行的时候，如果存在错误，将提示错误信息。其设置方法为：

选择菜单栏中的“工具”/“选项”命令，在弹出的“选项”对话框中选择“编辑器”选项卡，选中“代码设置”栏中的“自动语法检测”复选框，这样程序在编辑时会自动检测语法错误。“编辑器”选项卡如图 15.1 所示。

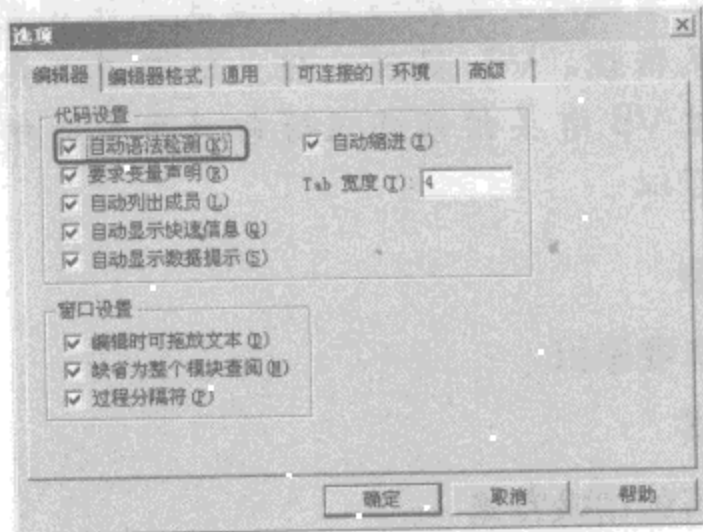


图 15.1 “编辑器”选项卡

在设置“自动语法检测”后，当用户在输入代码时，系统就在用户输入完一行代码并移动光标时自动执行语法检测来发现错误。例如，当输入 If 语句时，没有输入 Then 关键字，当光标离开该行以后，将弹出编译错误对话框，提示错误原因如图 15.2 所示。



图 15.2 “编译错误”实例

有些编译错误在代码编写时可能无法发现，需要当程序运行到相应的位置时，才提示错误。例如，当缺少配对结构的错误，在使用 If 语句时，如果没有 End If 语句，运行时将会提示错误信息，如图 15.3 所示。

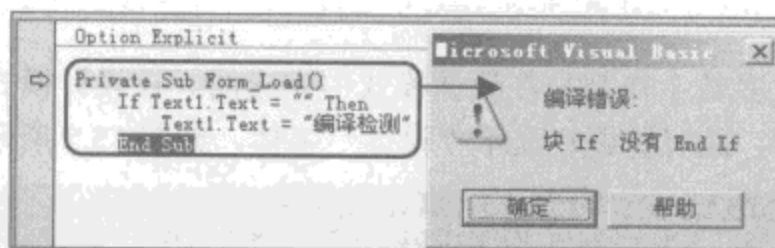


图 15.3 程序运行时发现的编译错误

### 15.1.2 运行错误

运行错误是在程序编译通过后运行代码时发生的，一般是由于程序执行过程中出现了非法操作引起的。例如，在进行除法运算时除数为 0、类型不匹配、访问的文件不存在等。

运行错误相对于语法错误比较难发现，需要通过不同的方法进行多次测试才会发现。因此在编写代码时，要考虑程序执行的每一步会发生什么变化，然后再将代码应用到程序中。

例如，定义了一个数值型变量 i，当程序运行时，通过文本框控件输入数据。将输入的数据赋值给变量 i，如果输入了非数值型信息，系统将出现错误提示，单击“调试”按钮，将会转到代码窗口中出错的代码行上，程序进入中断模式。如图 15.4 所示。

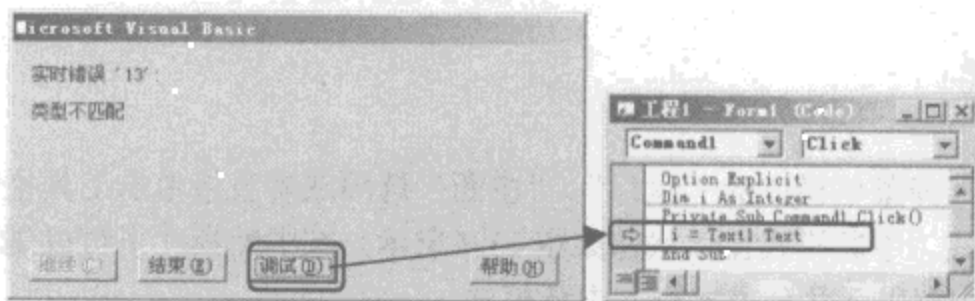


图 15.4 运行时错误

这时就要加判断，限制在文本框内输入的数据必须为数值型。

### 15.1.3 逻辑错误

逻辑错误是指程序没有按预期的方式执行，没有得到预期的效果，而代码又不存在语法错误，程序可以正常执行。这类错误不会产生错误提示信息，错误很难发现，需要程序员仔细分析程序，使用调试工具反复进行调试才可以发现。例如，函数定义错误、循环条件不正确、语句顺序不对等。

## 15.2 工作模式

教学录像：光盘\TM\lx\15\三种工作模式的切换.exe

VB 的工作状态可以分为 3 种模式，用户要时刻知道应用程序正处在何种模式下，这样才能更好地

测试和调试应用程序。工作模式分为设计模式、运行模式和中断模式（Break）。

### 15.2.1 设计模式

应用程序的创建过程大多数工作都是在设计模式中完成的。在设计模式下可以进行程序的界面设计、属性设置、代码编辑等，此时标题栏显示“设计”字体，如图 15.5 所示。



图 15.5 设计模式

### 15.2.2 运行模式

执行“启动”命令后，程序即由设计模式进入运行模式，标题栏显示“运行”字体。在运行模式下，可以查阅代码，但不能进行修改，如图 15.6 所示。



图 15.6 运行模式

### 15.2.3 中断模式

当程序运行时，选择菜单栏中的“运行”/“中断”选项或者直接单击工具栏中的“中断”按钮，程序将切换到中断模式。中断模式窗体状态如图 15.7 所示。在中断模式下可以浏览和修改代码、查阅变量取值等。在 VB 中有以下几种方式进入中断模式。



图 15.7 中断模式

- ☒ 在运行模式下选择“中断”命令。
- ☒ 在程序代码中设置了断点或 Stop 语句，程序执行到了断点或 Stop 语句处。
- ☒ 在程序运行时的错误而发生的中断。

## 15.3 调试工具及使用

 教学录像：光盘\TM\lx\15\调试工具的使用.exe

为了有效排除程序中发生的各种错误，VB 提供了强大的调试工具与良好的调试环境。通过使用这些工具，有助于查找程序中的错误。

### 15.3.1 调试工具栏的使用

VB 提供了专门的调试工具栏供程序员对代码进行调试。调试工具栏中包含了在调试代码时经常用到的一些常用菜单命令按钮。如果在 VB 集成开发环境的工具栏中没有找到调试工具栏，可以通过菜单命令将其添加到工具栏中。添加方法为：单击菜单栏中的“视图”/“工具栏”/“调试”命令，或在工具栏的空白处单击鼠标右键，在弹出的快捷菜单中选择“调试”命令，即可将调试工具栏添加到集成开发环境的工具栏中，如图 15.8 所示。

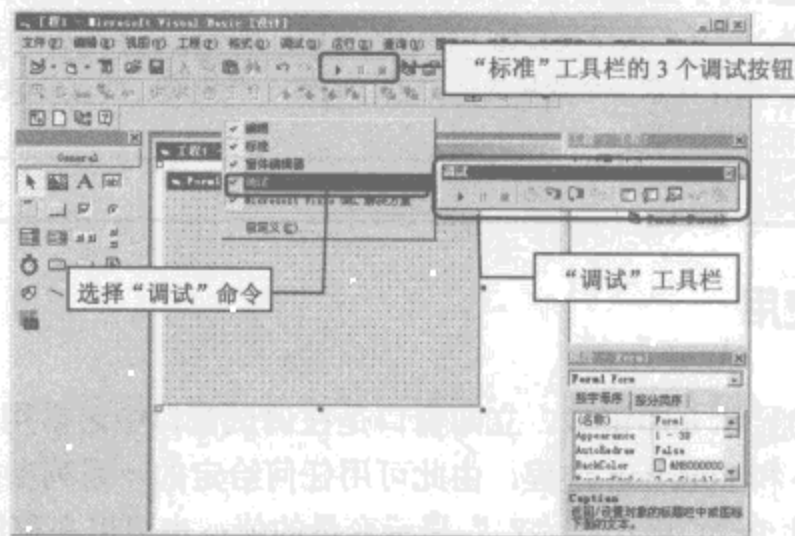


图 15.8 添加“调试”工具栏

“调试”工具栏如图 15.9 所示。“调试”工具栏中共有 3 组 12 个按钮，其中前 3 个按钮在标准工具栏中也可以看到。

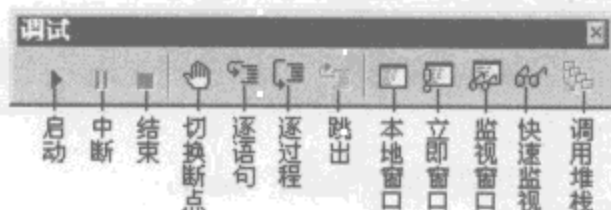


图 15.9 “调试”工具栏

### 15.3.2 本地窗口的使用

本地窗口显示所有在当前过程中声明的变量及变量的值。当程序从执行方式切换到中断模式或操纵堆栈中的变量时，就会自动刷新并显示当前过程中声明的变量及变量的值。本地窗口如图 15.10 所示。

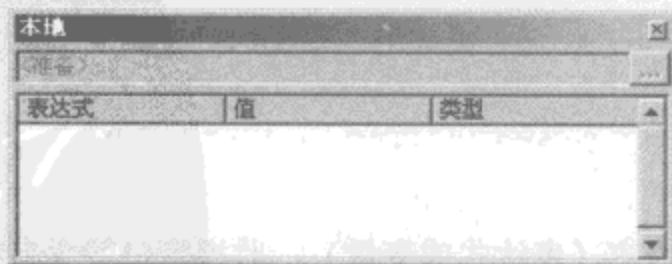


图 15.10 本地窗口



本地窗口各组成部分的功能如表 15.1 所示。

表 15.1 本地窗口各组成部分说明

项 目	说 明
<准备>	显示当前调用堆栈中的过程
“调用堆栈”	打开“调用堆栈”对话框，列出调用堆栈中的过程。选择相应的过程，本地窗口会显示该过程中声明的变量及变量的值
表达式	列出变量的名称。列表中的第一个变量是一个特殊的模块变量，可用来扩充显示出当前模块中的所有模块层次变量。对于类模块，会定义一个系统变量<Me>。对于常规模块，第一个变量是<name of the current module>。全局变量以及其他工程中的变量，都不能从本地窗口中访问
值	列出变量的值。选择任意一个变量的值，可以编辑一个新值。所有的数值变量都应该有一个值，而字符串变量则可以有空值
类型	列出变量的类型

### 15.3.3 立即窗口的使用

程序在中断模式时自动显示立即窗口。立即窗口是在调试窗口中最常用的窗口。立即窗口不接收数据声明，但可以使用 Sub 和 Function 过程，由此可用任何给定的一系列参数来测试过程的结果。

可以在立即窗口中使用 Print 语句或“?”显示变量的值；也可以在程序代码中利用 Debug.Print 方法，把输出送到立即窗口。

例如，查询网格控件的行数和列数，可使用立即窗口得到结果。在立即窗口中输入代码，如图 15.11 所示。在中断模式下，按〈Enter〉键，在立即窗口中就会显示结果，如图 15.11 所示。在程序调试过程中使用立即窗口调试既快捷又方便。

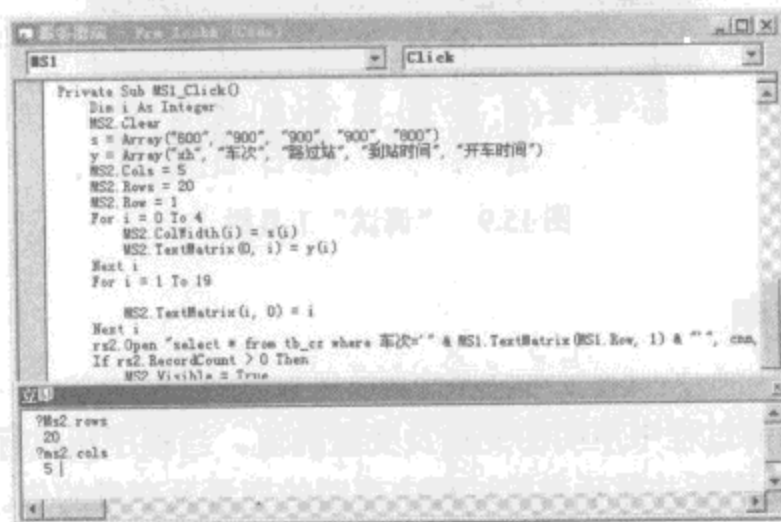


图 15.11 立即窗口

### 15.3.4 监视窗口的使用

在工程中如果设置了监视内容（表达式或变量），监视窗口就会自动显示。监视窗口可显示当前的监视表达式，并负责监视这个表达式或变量在程序中的变化。

在使用监视窗口监视某个表达式或变量前，需要先将其设置为监视内容。监视内容的设置方法如下：

- (1) 使用鼠标选中要监视的代码内容。
- (2) 在菜单栏中选择“调试”/“添加监视”命令，打开“添加监视”对话框，如图 15.12 所示。添加监视表达式以及设置的监视类型。

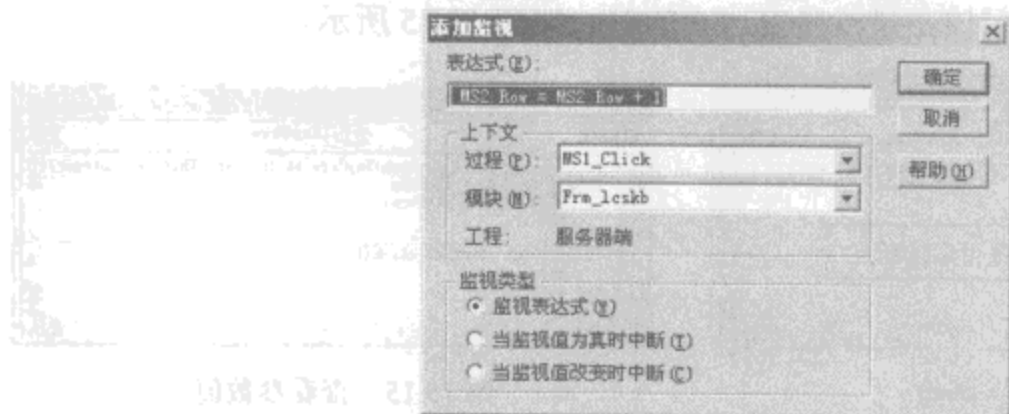


图 15.12 “添加监视”对话框

- (3) 设置完成后单击“确定”按钮，即可创建监视内容。这时监视窗口会自动显示并出现刚刚添加的监视内容，如图 15.13 所示。

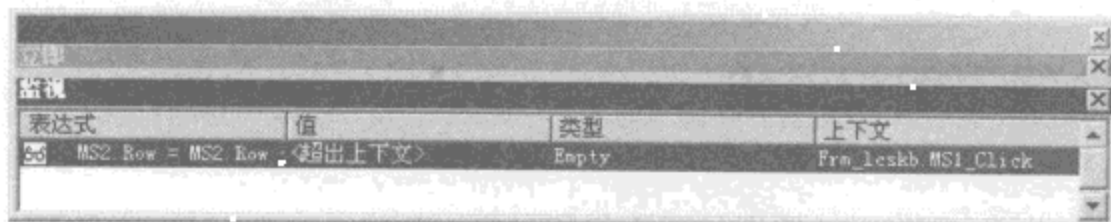


图 15.13 添加监视内容后的监视窗口

**说明：**添加监视内容时也可用鼠标选择要监视的内容，然后将其直接拖曳到监视窗口中。

监视窗口中各组成部分的说明如表 15.2 所示。

表 15.2 监视窗口中各组成部分的说明

组成部分	说 明
表达式	列出所监视的表达式，并在最左边列出监视图标
值	列出在切换为中断模式时表达式的值。选择任意一个监视内容的值，可以编辑一个新值。如果这个新值是无效的，则编辑字段会保持在作用中，并且值会以突出显示，还会出现一个消息框来描述这个错误，此时通过〈Esc〉键中止之前的更改
类型	列出表达式的类型
上下文	列出监视表达式的内容

**说明：**中断模式时，监视表达式的内容不在所选择的范围内时，当前的值不会显示出来。

### 15.3.5 插入断点和逐语句跟踪

设置断点可以在程序运行到断点处时中断程序的运行，然后逐语句跟踪检查相关变量、表达式的

值是否在预期的范围内。断点可在 3 种模式的任何模式下设置或删除。在代码窗口选择可能存在问题的语句作为断点,按下〈F9〉键或是在代码所在行的左侧灰色边框处单击鼠标左键,即可添加断点。如图 15.14 所示。在断点处再次单击即可取消断点。

程序运行到断点处时中断,切换到中断模式,此时可以直接查看变量或表达式的值,只要把鼠标悬停在要查看的语句上,就会显示该变量或是表达式的值,如图 15.15 所示。

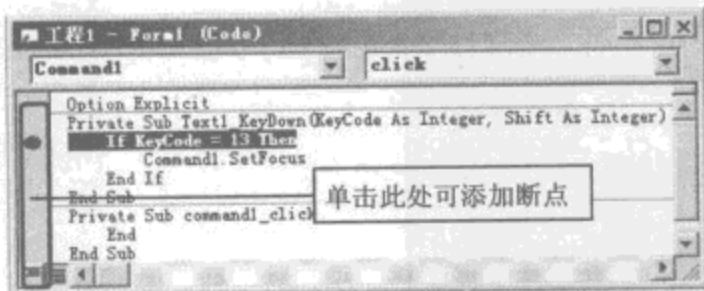


图 15.14 插入断点

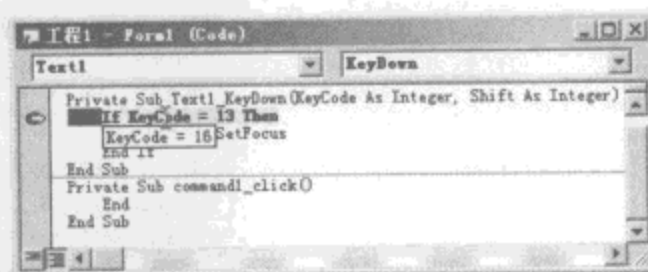


图 15.15 查看参数值

若要继续跟踪断点后面的语句执行情况,只要按〈F8〉键或者选择“调试”工具栏中的“逐语句”按钮,带有黄色箭头的突出显示将按照程序执行顺序逐行移动。

设置断点和逐语句跟踪是最简单、最常用的程序调试方式。

## 15.4 错误处理语句和对象

教学录像: 光盘\TM\15\错误处理语句和对象.exe

VB 预定义了一些专门包含运行时错误信息的对象,以及当程序产生错误后进行错误处理的语句。使用这些对象和语句可以有效地对错误进行处理,下面将介绍这些对象和语句。

### 15.4.1 Err 对象

Err 对象中含有有关当前程序运行时的错误信息,当程序运行中出现问题时,错误信息就会在 Err 对象中反应出来。Err 对象是程序中固有的公用对象,即在过程中不需要创建就可直接使用的对象。

Err 对象的属性和方法说明如表 15.3 所示。

表 15.3 Err 对象的属性和方法说明

属性/方法	说 明
Description 属性	返回或设置一个包含与对象相关联的描述性字符串
HelpContext 属性	返回或设置一个包含 Windows 操作系统帮助文件中主题的上下文 ID
HelpFile 属性	返回或设置一个表示帮助文件的完整限定路径
LastDLLError 属性	返回因调用动态链接库(DLL)而产生的系统错误号
Number 属性	返回或设置表示错误的数值,Err 对象的默认属性
Source 属性	返回或设置一个指明最初生成错误的对象或应用程序的名称

续表

属性/方法	说 明
Clear 方法	用来清除 Err 对象的所有属性设置。在错误处理之后使用，用来清除之前产生的 Err 对象的属性
Raise 方法	生成运行时错误

下面介绍 Err 对象的主要属性和方法。

#### (1) Description 属性

Description 属性返回值是 String 类型的值，返回程序的错误描述信息。当无法处理或不想处理错误时，可以使用这个属性提示用户。

#### (2) Number 属性

Number 属性表示错误的数值编号。可以根据这个编号进行错误处理。每个错误产生时都有一个错误号，错误号在 1~65535 之间，其中 1~1000 之间的错误号是系统已经使用的或保留为以后使用的，当产生自定义的错误时应该使用其他的错误号。

#### (3) Clear 方法

Clear 方法用于清除该对象的属性设置。它把 Err 的数值型属性设置为 0，字符串型属性设置为空字符串。

### 15.4.2 捕获错误 (On Error 语句)

利用 On Error 语句可以捕获错误信息，该语句可启动一个错误处理程序，并指定该错误处理程序在过程中的位置，也可用来禁止一个错误处理程序。

如果程序中不使用 On Error 语句，则运行时的错误信息可能导致终止程序运行。On Error 语句的语法包含了 3 种形式。

语法：

```
On Error GoTo <位置>
On Error GoTo 0
On Error Resume Next
```

#### 1. On Error GoTo 语句

当程序出现错误时，使用 On Error GoTo 语句将程序的执行流程转移到指定的代码行，由错误处理程序针对具体的错误进行处理。其中“位置”可以是任何标签或行号。指定的位置必须在发生错误的过程中，如果过程中不存在这个位置，VB 将会产生一个编译错误。

On Error GoTo 0 语句用来禁止当前过程中任何已启动的错误处理程序。计时过程中包含编号为 0 的行，也不把 0 指定为处理错误代码的起点。

#### 2. On Error Resume Next 语句

On Error Resume Next 语句是在程序运行错误发生时继续执行发生错误语句之后的语句，而不中断程序的运行。程序中使用该语句可以不顾运行时的错误，继续执行程序。该语句可以将错误处理程序



代码直接放在发生错误处，也就是将错误处理程序直接嵌入在发生错误的过程中，而不用像 On Error Resume Next 语句那样到指定的位置上去执行。

另外，如果不做错误处理程序，会忽略错误继续执行代码，这是危险的方法。此时，错误没有被提示也没有被纠正，这个错误可能会给后面的操作留下隐患。

### 15.4.3 退出错误处理 (Resume 语句)

错误处理完成后应及时退出，并控制程序返回到合适的位置，使程序继续执行。使用 Resume 语句来实现这一功能。

Resume 语句有如下 3 种用法。

(1) Resume [0]: 程序返回到出错语句处继续执行。

(2) Resume Next: 程序返回到出错语句的下一句继续执行。

(3) Resume Line: 程序返回到标签或行号处继续执行。

其中，Resume [0] 语句的作用是，重新执行与包含该语句的错误处理程序同一过程中的出错语句。如果错误来自从该过程中调用的被调过程，则重新执行这个调用过程的语句。

Resume line 语句的作用是，执行以参数 line 为行号或行标签的语句。参数 line 必须是和错误处理程序在同一个过程中的标签或行号。

例如，当用户删除了所有的类别信息后，又一次单击了“删除”按钮，这时便出现图 15.16 所示的错误信息。

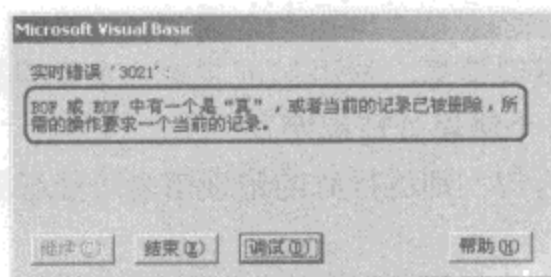


图 15.16 删除记录时出现的运行错误

出现这种错误时，可以使用 Resume Next 语句。只要在删除语句前加上“On Error Resume Next”语句即可解决这一问题。

### 15.4.4 编写错误处理函数

例 15.1 本实例编写了一个错误处理函数，在按钮单击事件中调用这个函数进行错误处理。程序运行效果如图 15.17 所示。(实例位置：光盘\TM\sl\15\1)

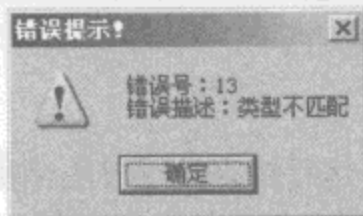


图 15.17 错误提示框

程序代码如下：

```
Public Function StrErr() As String
    Dim strinfo As String           '声明变量
    strinfo = "错误号：" & Err.Number & vbCrLf & _
        "错误描述：" & Err.Description '将错误信息赋给变量
    MsgBox strinfo, 48, "错误提示！" '显示提示信息
End Function
Private Sub Command1_Click()
    On Error GoTo handle1          '错误处理
    Dim i As Integer               '声明变量
    i = Text1.Text                  '错误的赋值
ExitSub
handle1:                           '指定标签
    Call StrErr                    '调用错误处理函数
End Sub
```

## 15.5 小结

本章主要介绍了在 VB 中的错误类型、调试工具的使用以及排错方法。通过本章的学习，读者可以掌握调试工具的使用，了解 VB 中的常见错误，掌握对 VB 应用程序设计过程中出现的常见问题的处理方法。



# 第16章

## 文件系统编程

(教学录像: 1 小时 55 分钟)

对文件的操作是程序开发语句可以实现的最基本的功能之一。Microsoft VB 6.0 具有较强的文件处理能力, 它为用户提供了多种处理文件的方法及大量与文件系统有关的语句、函数及控件。用户使用这些技术可以编写出功能强大的文件处理程序。本章将对各种文件的操作方法进行介绍。

通过阅读本章, 您可以:

- ▶▶ 了解文件的基本概念
- ▶▶ 认识文件系统控件
- ▶▶ 掌握文件系统控件的使用方法
- ▶▶ 掌握文件操作语句的使用方法
- ▶▶ 掌握文件操作函数的使用方法
- ▶▶ 掌握顺序文件的读写方法
- ▶▶ 掌握随机文件的读写方法
- ▶▶ 掌握二进制文件的读写方法
- ▶▶ 了解 FSO 文件系统对象

## 16.1 文件的基本概念

 教学录像：光盘\TMlx\16\文件的基本概念.exe

文件是存储在外部介质上的数据或信息的集合，用来永久保存大量的数据。计算机中的程序和数据都是以文件的形式进行存储的。大部分的文件都存储在诸如硬盘驱动器、磁盘和磁带等辅助设备上，并由程序读取和保存。在程序运行过程中所产生的大量数据，往往也都要输出到磁盘介质上进行保存。

### 16.1.1 文件的结构

为了有效地对数据进行读写，数据必须以某种特定的格式存储，这种特定的格式称为文件结构。VB 文件由记录组成，记录由字段组成，字段由字符组成。

文件：是由具有同一记录结构的所有记录组成的集合。每个文件都应该有一个文件名，它是对文件进行访问的唯一手段。文件名一般由主文件名（简称文件名）和扩展名组成。

记录：是由一组域组成的一个逻辑单位，一个记录中的各个域之间应该相互有关系，每个记录有一个记录名，用来表示一个唯一的记录结构，记录是计算机进行信息处理的基本单位。

字段：也称域。域是文件中的一个重要概念，一般是由几个字符所组成的一项独立的数据。每个域都有一个域名，每个域中具体的数据值称为该域的域值。

字符：是构成文件的最基本单位，凡是单一字节、数字、标点符号或其他特殊符号都是一个字符。

### 16.1.2 文件的分类

#### 1. 根据数据的使用分类

根据数据的使用情况，文件可以分为数据文件和程序文件两类。

##### (1) 数据文件

数据文件中存放普通的数据。如药品信息、学生信息等，这些数据可以通过特定的程序存取。

##### (2) 程序文件

程序文件中存放计算机可以执行的程序代码，包括源文件和可执行文件等。如 VB 中的工程文件（.vbp）、窗体文件（.frm）、窗体的二进制数据文件（.frx）、标准模块文件（.bas）、类模块文件（.cls）、可执行文件（.exe）等都是程序文件。

VB 中的程序文件还包括 ActiveX 控件的文件（.ocx）、单个资源文件（.res）以及动态链接库文件（.dll）等。

#### 2. 根据数据编码方式分类

##### (1) ASCII 文件

ASCII 文件又称为文本文件，字符以 ASCII 码方式存放，Windows 中的字处理软件建立的文件就是 ASCII 文件。



### (2) 二进制文件

二进制文件中的数据是以字节为单位存取的，不能用普通的字处理软件创建和修改。

### 3. 根据数据访问方式分类

VB 中提供了 3 种数据的访问方式：顺序访问、随机访问和二进制访问，相应的文件可分为顺序文件、随机文件和二进制文件。

## 16.1.3 文件处理的一般步骤

不同类型的文件的访问方式是有所区别的，但是在 VB 中无论是什么类型的文件，其处理步骤基本上是相同的，一般按照下列 3 个步骤进行。

#### (1) 打开文件

对文件进行操作前必须先打开文件。

#### (2) 对文件进行读写操作

在打开的文件上执行所要求的读写操作。对文件的读操作，就是把文件中的数据传送到内存程序中；对文件的写操作，就是把内存中的数据存放到外部设备并作为文件存放。

#### (3) 关闭文件

对打开的文件在完成了读写操作后，要关闭文件以释放相关文件缓冲，缓冲区中的剩余数据被读入内存或写入文件。

## 16.2 文件系统控件

 教学录像：光盘\TM\lx\16\文件系统控件.exe

VB 提供了两种文件系统控件：一种是由 CommonDialog 控件提供的通用对话框，就是经常会用到的“打开”、“保存”等对文件进行操作的通用对话框。另一种是使用 VB 提供的文件系统控件自行创建对话框。使用后一种方法可以自己设计一些具有独特风格的文件操作界面，可使访问文件系统的对话框更加直观。

VB 提供了 3 个文件系统控件，分别是驱动器列表框 (DriveListBox)、目录列表框 (DirListBox) 和文件列表框 (FileListBox)。由于这 3 个控件都用来进行文件的操作，所以又被统称为文件系统控件。使用文件系统控件可以方便地进行文件操作，这 3 个控件是 VB 的内部控件，在 VB 的工具箱中即可看到，如图 16.1 所示。

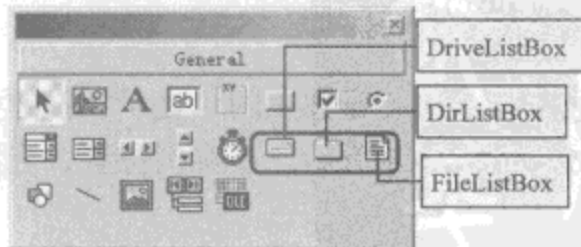


图 16.1 文件系统控件

## 16.2.1 驱动器列表框 (DriveListBox 控件)

驱动器列表框 (DriveListBox) 控件能够提供本地计算机上有效磁盘驱动器的名称, 是一个下拉式列表框, 与 ComboBox 控件相似, 只是列表内容是事先建立好的。程序运行时可以在其下拉列表中选择一磁盘驱动器, 如软驱、硬盘分区和光驱等。

### 1. 主要属性

#### (1) Drive 属性

Drive 属性用于返回或设置所选择的驱动器, 包括在运行中控件创建或刷新时系统已有的或连接到系统上的所有驱动器。在程序运行时, 可以通过键盘输入有效的驱动器名, 也可以在控件的下拉列表框中选择驱动器。默认显示用户系统上的当前驱动器, 程序设计时不可用。

语法:

```
object.Drive [= drive]
```

object: 对象表达式。

drive: 字符串表达式, 指定所选择的驱动器。

例 16.1 本实例实现在程序运行时, 在窗体上的 DriveListBox 控件中选择驱动器名称, 所选择的驱动器名称将显示在下面的标签中。程序运行效果如图 16.2 所示。(实例位置: 光盘\TM\sl\16\1)

程序代码如下:

```
Private Sub Drive1_Change()  
    Label1.Caption = Drive1.Drive  
End Sub
```

#### (2) List 属性

List 属性用于返回或设置控件的列表部分的项目。列表是一个字符串数组, 数组的每一项都是一个列表项目, 在程序运行时该属性只读。

语法:

```
object.List(index) [= string]
```

object: 对象表达式。

index: 列表中具体某一项目的号码。

string: 字符串表达式, 指定列表项目。

⚠ 注意: 列表中第一个项目的索引为 0, 而最后一个项目的索引为 ListCount-1。对于 DriveListBox 控件, 列表内容是包含有效的驱动连接列表。

例 16.2 本实例实现当程序运行后, 单击窗体上的“显示”按钮, 将当前驱动器列表框中的项目显示在下面的标签中, 程序运行效果如图 16.3 所示。(实例位置: 光盘\TM\sl\16\2)

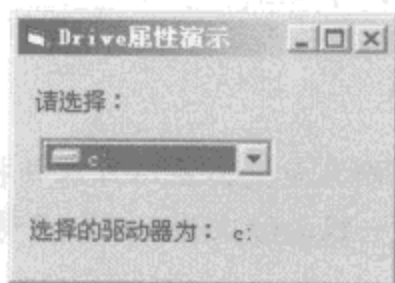


图 16.2 Drive 属性演示界面

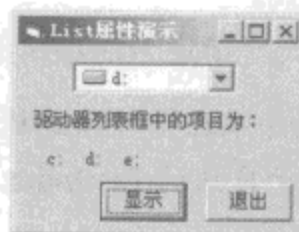


图 16.3 DriveListBox 控件的 List 属性演示界面

程序代码如下:

```
Private Sub Command1_Click()
    Dim i As Integer
    Dim s As String
    For i = 0 To Drive1.ListCount - 1
        s = s + " " & Drive1.List(i)
        Label1.Caption = s
    Next i
End Sub
```

'循环次数为驱动器中列表项数目  
'将列表项赋给变量  
'将变量赋给标签

## 2. 主要事件

Change 事件在驱动器列表框中当前所选择的驱动器名称发生改变时被触发, 如程序运行时在下拉列表框中选择一个新的驱动器或通过代码改变 Drive 属性的设置都会触发该事件。

### 16.2.2 目录列表框 (DirListBox 控件)

目录列表框 (DirListBox) 控件用于显示当前驱动器上的目录列表。其操作与 ListBox 控件相似, 但它具有显示当前所选驱动器目录清单的功能。目录显示方式与 Windows 显示风格相同, 根目录突出显示, 其他各级子目录依次缩进。

DirListBox 控件的常用属性有 List 属性、ListIndex 属性和 Path 属性。下面分别进行介绍。

#### (1) List 属性

DirListBox 控件的 List 属性用于返回或设置控件的列表部分的项目。列表是一个字符串数组, 数组的每一项都是一个列表项目, 此属性在运行时只读。

语法:

```
object.List(index) [= string]
```

object: 对象表达式。

index: 列表中具体某一项目的号码。

string: 字符串表达式, 指定列表项目。

#### (2) ListIndex 属性

ListIndex 属性用于返回或设置控件中当前选择项目的索引值, 该属性值为整型, 在设计时不可用。

语法:

```
object.ListIndex [= index]
```

object: 对象表达式。

index: 数值表达式, 指定当前项目的索引。

注意: DirListBox 并不在操作系统级设置当前目录, 而只是突出显示目录并将其 ListIndex 设置为 -1。第一个子目录的 ListIndex 属性值为 0, 下一级的依次为 1、2、3 等, 如图 16.4 所示。

可以利用该属性访问任何一级目录, 也可以访问当前目录的上一级或是下一级目录。

### (3) Path 属性

Path 属性用于返回或设置当前路径。例如在目录列表框中选择了 C 盘的根目录, 则 Path 属性为 “C:\”; 如果选中了 C 盘下的 Windows 文件夹, 则 Path 属性为 “C:\Windows”。也可以通过代码进行设置, 例如:

```
Dir1.Path = "C:\Windows"
```

上面的代码设置 “C:\Windows” 为当前目录, 并突出显示, 如图 16.5 所示。程序运行时, 在目录列表框中选择了某个目录, 系统就会把这个目录的路径赋给 Path 属性。Path 属性值的改变会触发该控件的 Change 事件。该属性在程序设计时是不可用的。

例 16.3 本实例演示 DirListBox 控件的 Path 属性的使用。运行程序, 在 DirListBox 控件中选择相应的文件夹, 在下面的标签中显示该文件夹的路径, 如图 16.6 所示。(实例位置: 光盘\TM\sl\16\3)

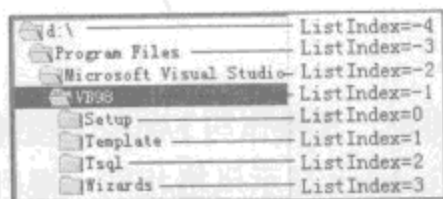


图 16.4 DirListBox 控件的 ListIndex 属性值 图 16.5 为 Path 属性赋值 图 16.6 Path 属性实例演示界面

程序代码如下:

```
Private Sub Drive1_Change()  
    Dir1.Path = Drive1.Drive  
End Sub  
Private Sub Dir1_Change()  
    Label1.Caption = Dir1.Path  
End Sub
```

'将选择的驱动器赋给 Path  
'将 Path 值赋给标签

## 16.2.3 文件列表框 (FileListBox 控件)

文件列表框 (FileListBox 控件) 用于将 Path 属性指定的目录下的文件列表显示出来。FileListBox 控件与 ListBox 控件相似, 只是其中的列表内容显示的是所选目录的文件名清单。

下面介绍 FileListBox 控件的主要属性和事件。

### 1. 主要属性

#### (1) Path 属性



Path 属性用于返回或设置当前路径。在设计时不可用。

语法:

`object.Path [= pathname]`

object: 对象表达式。

pathname: 一个用来计算路径名的字符串表达式。

例 16.4 本实例演示文件列表框的 Path 属性的使用。程序运行时, 在驱动器列表框中选择驱动器, 在目录列表框中选择文件所在文件夹, 选择的文件夹下的所有文件显示在文件列表框中, 用鼠标单击文件列表框中的文件, 则在下面的标签中显示选择的文件所在路径。程序运行效果如图 16.7 所示。(实例位置: 光盘\TM\sl\16\4)

程序代码如下:

```
Private Sub Dir1_Change()  
    File1.Path = Dir1.Path  
End Sub  
Private Sub Drive1_Change()  
    Dir1.Path = Drive1.Drive  
End Sub  
Private Sub File1_Click()  
    Label1.Caption = "选择文件的路径是: " & File1.Path  
End Sub
```

'为文件列表框的 Path 属性赋值  
'为目录列表框的 Path 属性赋值  
'将文件路径显示在标签中

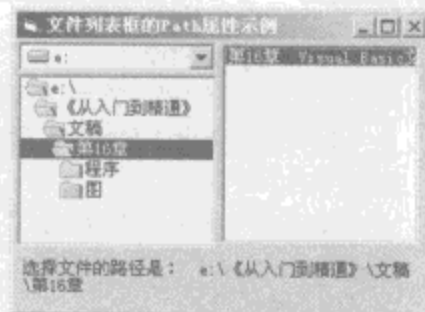


图 16.7 文件列表框的 Path 属性示例界面

## (2) Pattern 属性

Pattern 属性用于返回或设置一个值, 指示在运行时显示在 FileListBox 控件中的文件的扩展名。

语法:

`object.Pattern [= value]`

object: 对象表达式。

value: 一个用来指定文件规格的字符串表达式, 例如 "\*.txt" 或 "\*.frm"。默认值是 "\*.txt", 可返回所有文件的列表。除使用通配符外, 还能够使用以分号 (;) 分隔的多种模式。

例如可以使用下面的代码指定在运行时显示在 FileListBox 控件中的文件扩展名。

```
File1.Pattern = "*.txt"           '显示所有的文本文件  
File1.Pattern = "*.txt;*.doc"    '显示所有的文本文件和 Word 文档文件  
File1.Pattern = "???.txt"        '显示文件名包含 3 个字符的文本文件
```

## (3) FileName 属性

FileName 属性用于返回或设置所选文件的文件名, 其值为字符串。在设计时不可用。

语法:

`object.FileName [= pathname]`

object: 对象表达式。

pathname: 字符串表达式, 指定路径和文件名。

FileName 属性不包括路径名。这和 CommonDialog 控件的 FileName 属性不同。在程序设计时, 使



用文件系统控件浏览文件时，如果要进行进一步操作如打开、保存等，就必须获得具有全部路径的文件名，例如：“C:\MyFolder\MyFile.txt”。通常采用将 FileListBox 控件的 Path 属性和 File 属性值中的字符串连接起来的方法来获得带路径的文件名。在使用中要注意判断 Path 属性的最后一个字符是否是目录分隔号“\”，如果不是应添加一个“\”号，以保证目录的正确。可利用如下代码实现：

```
Dim MyStr As String
If Right(File1.Path, 1) = "\" Then           '如果路径最后一个字符是“\”
    MyStr = File1.Path & File1.FileName      '将路径和文件名连接起来
Else
    MyStr = File1.Path & "\" & File1.FileName '将加上“\”符号后的路径和文件名连接
End If
Print MyStr
```

## 2. 主要事件

### (1) PathChange 事件

当 FileListBox 控件中的路径改变时 PathChange 事件被触发。FileName 或 Path 属性值的改变都能引起路径的改变，也就是都能触发 PathChange 事件。

语法：

```
Private Sub object_PathChange([index As Integer])
```

object: 一个对象表达式。

index: 一个整数，用来唯一地标识一个在控件数组中的控件。

⚠ 注意：可使用 PathChange 事件过程来响应 FileListBox 控件中路径的改变。当将包含新路径的字符串给 FileName 属性赋值时，FileListBox 控件就调用 PathChange 事件。

例 16.5 本实例实现当程序运行时选择不同的文件夹，改变 FileListBox 的路径，触发 PathChange 事件，在标签中显示当前 FileListBox 的路径，如图 16.8 所示。（实例位置：光盘\TM\sl16\5）

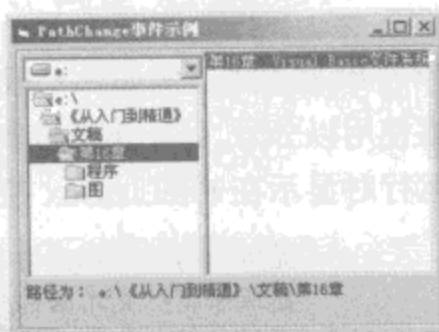


图 16.8 PathChange 事件示例界面

程序代码如下：

```
Private Sub Dir1_Change()
    File1.Path = Dir1.Path           '为文件路径赋值
End Sub
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive         '为目录列表框路径赋值
End Sub
Private Sub File1_PathChange()
```

```

Label1.Caption = "路径为: " & File1.Path           '将文件路径赋值给标签
End Sub
Private Sub Form_Load()
    Label1.Caption = "路径为: " & Dir1.Path         '将路径赋值给标签
End Sub

```

### (2) PatternChange 事件

当文件的列表样式, 如 "\*. \*", 被代码中对 FileName 或 Path 属性的设置所改变时, 此事件发生。  
语法:

```
Private Sub object_PatternChange([index As Integer])
```

object: 一个对象表达式。

index: 一个整数, 用来唯一地标识一个在控件数组中控件。

 说明: 可使用 PatternChange 事件过程来响应在 FileListBox 控件中样式的改变。当将包含新样式的字符串给 FileName 属性赋值时, FileListBox 控件将调用 PathChange 事件。

**例 16.6** 本例演示了 FileListBox 控件的 PatternChange 事件。运行程序, 在下拉列表框中选择文件类型, 在 FileListBox 控件中将显示选择的文件夹下的相应类型的文件, 在下面的标签中显示文件路径。如图 16.9 所示。(实例位置: 光盘\TM\sl\16\6)

主要代码如下:

```

Private Sub File1_PatternChange()
    File1.Pattern = Combo1.Text           '将选择的文件类型赋给 Pattern 属性
    Label1.Caption = "文件的路径为: " & File1.Path   '在标签中显示文件路径
End Sub

```

## 16.2.4 文件系统控件的联动

文件系统的 3 个控件的联合使用称为文件系统控件的联动。在应用程序开发过程中, 一般将文件系统的 3 个控件联合使用。下面使用一个典型范例介绍文件系统控件的联合使用。

**例 16.7** 将 DriveListBox 控件、DirListBox 控件、FileListBox 控件联动, 实现选择文件的功能。实现效果如图 16.10 所示。(实例位置: 光盘\TM\sl\16\7)



图 16.9 FileListBox 控件的 PatternChange 事件应用



图 16.10 文件系统控件联动程序

主要代码如下：

```
Private Sub Combo1_Click()
    File1.Pattern = Mid(Combo1.Text, 21)           '设置文件类型
End Sub
Private Sub Dir1_Change()
    File1.Path = Dir1.Path                         '为文件列表框赋值
End Sub
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive                       '为目录列表框赋值
End Sub
Private Sub File1_Click()
    Dim st As String, fpath As String              '定义变量
    If Right(Dir1.Path, 1) = "\" Then              '如果路径以“\”结尾
        fpath = Dir1.Path & File1.FileName         '将路径赋给变量
    Else
        fpath = Dir1.Path & "\" & File1.FileName   '将路径赋给变量
    End If
    Text1.Text = ""                                '文本框设置为空
    Open fpath For Input As #1                     '打开文件
    Do While Not EOF(1)                             '直到文件末尾
        Line Input #1, st                           '读取文件内容
        Text1.Text = Text1.Text + st + vbCrLf       '将文件内容赋给文本框
    Loop
    Close #1                                         '关闭文件
End Sub
```

## 16.3 文件操作的语句

 教学录像：光盘\TM\16\文件的操作语句.exe

VB 对文件的操作主要由文件操作语句和函数完成。本节将介绍几个常用的文件操作语句。

### 16.3.1 改变当前驱动器（ChDrive 语句）

ChDrive 语句用来改变当前的驱动器。


语法：

**ChDrive drive**

**drive**：必要的参数，是一个字符串表达式，它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 只会使用首字母。

例如，使用 ChDrive 语句设置“D”为当前驱动器，代码如下：

**ChDrive "D"** '使“D”成为当前驱动器

 **注意**：ChDrive 语句改变的是默认的驱动器，当再次加载窗体时显示的是更改后的驱动器。

例 16.8 本实例实现通过 ChDrive 语句改变当前驱动器，运行程序后单击窗体上的“更改盘符”按钮，打开 Form2 窗体，单击窗体上的“返回”按钮，返回 Form1。这时窗体上驱动器列表框中显示的是更改后的盘符，如图 16.11 所示。（实例位置：光盘\TM\sl\16\8）

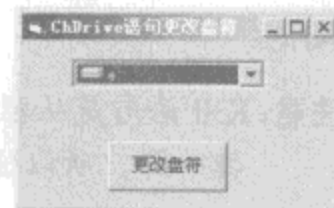


图 16.11 更改盘符

程序代码如下：

```
Private Sub Command1_Click()  
    ChDrive Drive1.List(Drive1.ListIndex)    '将驱动器设置为当前驱动器  
    Unload Me                                '卸载此窗体  
    Load Form2                              '加载 Form2  
    Form2.Show                              '显示 Form2 窗体  
    Drive1.Refresh                          '刷新驱动器列表框  
End Sub
```

### 16.3.2 改变目录或文件夹（ChDir 语句）

ChDir 语句用来改变当前的目录或文件夹。

语法：

ChDir path

path: 必要参数，是一个字符串表达式，它指明哪个目录或文件夹将成为新的默认目录或文件夹。path 可能包含驱动器。如果没有指定驱动器，则 ChDir 在当前的驱动器上改变默认目录或文件夹。

⚠ 注意：ChDir 语句可以改变默认目录位置，但不会改变默认驱动器位置。例如，如果默认的驱动器是 C，则可以改变驱动器 D 上的默认目录，但是 C 仍然是默认的驱动器。

例如，可以应用下面的语句改变目录或文件夹。

将当前目录或文件夹改为“MYDIR”：

```
ChDir "MYDIR"
```

将工作目录设到应用程序所在目录：

```
ChDir App.Path
```

将目录设到操作系统路径下：

```
ChDir "D:\WINDOWS\SYSTEM"
```

### 16.3.3 删除文件（Kill 语句）

Kill 语句用于从磁盘中删除文件。

语法：

Kill pathname

pathname: 必要参数，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。

注意: Kill 语句是从驱动器中删除一个或多个文件,作用就像 Windows 操作系统中的〈Shift+Delete〉键一样,所以使用时要谨慎,而且 Kill 语句允许使用“?”与“\*”通配符。

例 16.9 本实例实现使用 Kill 语句删除文件,建议删除文件时要谨慎,因为如果删除的是系统文件,将导致其他程序或者操作系统运行时出错,所以本例创建了“删除.txt”的文本文件,在使用程序时可以删除此文件。程序运行效果如图 16.12 所示。(实例位置:光盘\TM\sl\16\9)

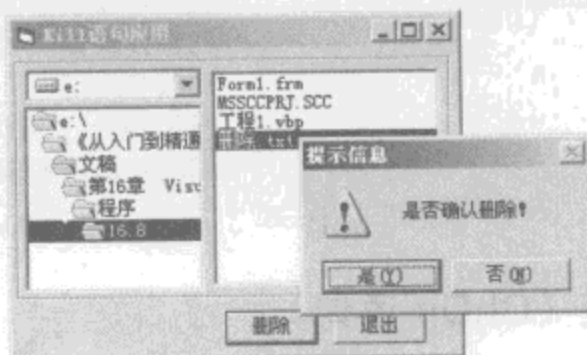


图 16.12 使用 Kill 语句删除文件

程序代码如下:

```
Private Sub Command1_Click()
    If MsgBox("是否确认删除!", 52, "提示信息") = vbYes Then
        Kill File1.Path & "\" & File1.FileName
        File1.Refresh
    End If
End Sub

Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub

Private Sub Form_Load()
    Drive1.Drive = App.Path
    Dir1.Path = Drive1.Drive
    File1.FileName = Dir1.Path
    Open App.Path & "\删除.txt" For Binary As #1
    Close #1
End Sub
```

'提示信息  
'删除指定路径下的文件  
'文件列表框刷新  
  
'为文件列表框路径赋值  
  
'为目录列表框赋值  
  
'将当前路径赋给驱动器  
'将驱动器路径赋给目录列表框  
'为文件名赋值  
'打开文本文件  
'关闭文本文件

### 16.3.4 创建目录或文件夹 (MkDir 语句)

MkDir 语句用于创建一个新的目录或文件夹。

语法:

MkDir path



path: 必要参数, 是用来指定所要创建的目录或文件夹的字符串表达式。path 可以包含驱动器。如果没有指定驱动器, 则 Mkdir 语句会在当前驱动器上创建新的目录或文件夹。

例如, 在 D 盘下创建一个新的文件夹, 代码如下:

```
Mkdir "d:\myfolder"
```

'在"D"盘符下创建一个 myfolder 文件夹

⚠ 注意: 如果创建的文件已经存在, 则会产生错误。

### 16.3.5 复制文件 (FileCopy 语句)

FileCopy 语句用于复制一个文件。

语法:

```
FileCopy source, destination
```

source: 必要参数。字符串表达式, 用来表示要被复制的文件名。source 可以包含目录或文件夹以及驱动器。

destination: 必要参数。字符串表达式, 用来指定要复制的目的文件名。destination 可以包含目录或文件夹以及驱动器。

📖 说明: 如果想要对一个已打开的文件使用 FileCopy 语句, 则会产生错误。

例 16.10 本例使用 FileCopy 语句实现对文件的复制。运行程序选择要复制的文件以及要将文件复制到的地址, 然后单击“确定”按钮, 即可将源文件中的文件复制到目标地址中, 在复制完成后会出现复制成功的提示信息。程序运行效果如图 16.13 所示。(实例位置: 光盘\TM\sl\16\10)

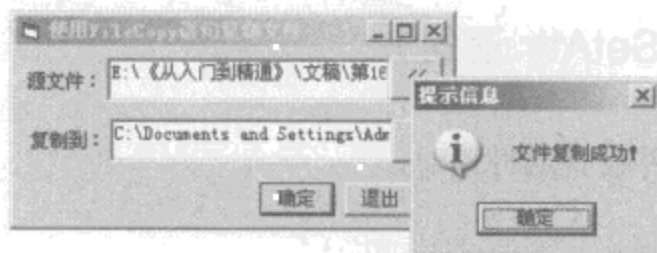


图 16.13 文件复制程序界面

程序代码如下:

```
Private Sub Command1_Click()
    CommonDialog1.ShowOpen
    Text1.Text = CommonDialog1.FileName
End Sub
Private Sub Command2_Click()
    CommonDialog1.ShowSave
    Text2.Text = CommonDialog1.FileName
End Sub
Private Sub Command3_Click()
    If Text1.Text = "" Or Text2.Text = "" Then
        MsgBox "输入不能为空值"
    Else
        '选择要复制文件的路径
        '打开“打开对话框”
        '将文件名赋给 text1
        '选择文件存放的路径
        '打开“保存”对话框
        '将文件名赋给 Text2
        '复制文件
        '如果文本框为空
        '提示信息
```

```

FileCopy Text1.Text, Text2.Text      '复制文件
MsgBox "文件复制成功!", vbInformation, "提示信息"      '提示信息
End If
End Sub
Private Sub Command4_Click()          '关闭退出
End
End Sub

```

### 16.3.6 重命名 (Name 语句)

Name 语句用于重新命名一个文件、目录或文件夹。

语法:


```
Name oldpathname As newpathname
```

**oldpathname:** 必要参数。字符串表达式, 指定已存在的文件名和位置, 可以包含目录或文件夹以及驱动器。

**newpathname:** 必要参数。字符串表达式, 指定新的文件名和位置, 可以包含目录或文件夹以及驱动器。由 newpathname 所指定的文件名不能存在。

例如使用下面的代码可以重命名一个文件。

```
Name OldName As NewName      '将名为 OldName 的文件命名为 NewName
```

 **注意:** 在一个已打开的文件上使用 Name 语句, 将会产生错误。因此, 必须在改变名称之前, 先关闭打开的文件。

### 16.3.7 设置文件属性 (SetAttr 语句)

SetAttr 语句用于为一个文件设置属性信息。

语法:

```
SetAttr pathname, attributes
```

**pathname:** 必要参数。用来指定一个文件名的字符串表达式, 可能包含目录或文件夹以及驱动器。

**attributes:** 必要参数。常数或数值表达式, 其总和用来表示文件的属性。attributes 参数设置如有表 16.1 所示。

表 16.1 attributes 参数的值及描述说明

常 数	值	描 述
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

例如使用下面的语句设置文件属性。

```
SetAttr "TEXT", vbHidden           '设置隐含属性
SetAttr "TEXT", vbHidden + vbReadOnly '设置隐含并只读
```

注意：如果想要给一个已打开的文件设置属性，则会产生运行时错误。

## 16.4 常用的文件函数

教学录像：光盘\TM\lx\16\文件操作函数.exe

本节介绍几个常用的文件操作函数的使用方法，使读者掌握对文件的基本操作。

### 16.4.1 获取路径（CurDir 函数）

CurDir 函数用于返回一个 Variant(String)（字符串）值，用来代表当前的路径。

语法：

```
CurDir[(drive)]
```

drive：可选的参数，是一个字符串表达式，它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 函数会返回当前驱动器的路径。

例如，下面对 CurDir 函数的应用：

使用 CurDir 函数来返回当前的路径。

```
' 假设 C 驱动器的当前路径为"C:\WINDOWS\SYSTEM"
' 假设 D 驱动器的当前路径为"D:\Program Files"
' 假设 D 为当前的驱动器
Dim MyPath
MyPath = CurDir           '返回"D:\Program Files"
MyPath = CurDir("C")     '返回"C:\WINDOWS\SYSTEM"
MyPath = CurDir("D")     '"D:\Program Files"
```

获取应用程序目录下的 Access 数据库所在的路径。

```
Dim Paths
Paths=CurDir & "\db_Data.mdb"
```

### 16.4.2 获取文件属性（GetAttr 函数）

GetAttr 函数用于返回一个 Integer（整数）值，此为一个文件、目录或文件夹的属性。

语法：


```
GetAttr(pathname)
```

pathname：必要参数，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。

由 GetAttr 返回的值是表 16.2 所示的这些属性值的总和。

表 16.2 GetAttr 函数返回值的说明

常 数	值	描 述
vbNormal	0	常规
vbReadOnly	1	只读
VbHidden	2	隐藏
VbSystem	4	系统文件
vbDirectory	16	目录或文件夹
vbArchive	32	上次备份以后，文件已经改变
VbAlias	64	指定的文件名是别名

 **注意：**若要判断是否设置了某个属性，在 GetAttr 函数与想要得知的属性值之间使用 And 运算符与逐位比较。如果所得的结果不为零，则表示设置了这个属性值。例如，在下面的 And 表达式中，如果档案（Archive）属性没有设置，则返回值为零；如果文件的档案属性已设置，则返回非零的数值。

### 16.4.3 获取文件创建或修改时间（FileDateTime 函数）

FileDateTime 函数返回一个 Variant(Date)（日期型）值，此值为一个文件被创建或最后修改后的日期和时间。

语法：

FileDateTime(pathname)

pathname：必要参数，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。

使用 FileDateTime 函数可以获得文件创建或最近修改的日期与时间。日期与时间的显示格式根据操作系统的地区设置而定。例如，文件路径为“D:\我的文件\系统文档”，上次被修改的时间为 2005 年 2 月 16 日下午 4 时 35 分 47 秒，获取其最后修改时间的代码如下：

```
StrTime = FileDateTime("D:\我的文件\系统文档") '变量 StrTime 中的值为 2005-2-16 4:35:47
```

### 16.4.4 返回文件长度（FileLen 函数）

FileLen 函数用于返回一个长整数型数值，代表一个文件的长度，单位是字节。

语法：

FileLen(pathname)

pathname：必要参数，是用来指定一个文件名的字符串表达式。pathname 可以包含目录、文件夹以及驱动器。

例如, 使用 FileLen 函数来返回指定文件夹中文件的字节长度。假如 “\Myfile.txt” 的长度为 20, 则变量 StrLen 的值也为 20。代码如下:

```
StrLen = FileLen("D:\我的文件\Myfile.txt") ' 变量 StrLen 的返回值为 20
```

### 16.4.5 测试文件结束状态 (EOF 函数)

EOF 函数返回一个 Integer (整数) 值, 它包含布尔型值 True, 表明已经到达为随机或顺序 Input 打开的文件的结尾。

语法:

EOF(filename)

filename: 必要参数, 是一个 Integer (整数) 值, 包含任何有效的文件号。

例如, 使用 EOF 函数可以检测文件是否已经读到末尾。假设 “D:\我的文件夹\Myfile.txt” 中的文件为有数个文本行的文本文件。

```
Open "D:\我的文件夹\Myfile.txt" For Input As #1 '为输入打开文件
Do While Not EOF(1) '检查文件尾
    Line Input #1, Inputstr '读入数据, 并将其存入变量 Input 中
    Debug.Print Inputstr '在立即窗口中显示
Loop
Close #1 '关闭文件
```

注意: (1) 只有到达文件结尾的时候, EOF 函数才返回 False。对于访问 Random 或 Binary 而打开的文件, 直到最后一次执行的 Get 语句无法读出完整的记录时, EOF 都返回 False。

(2) 对于为访问 Binary 而打开的文件, 在 EOF 函数返回 True 之前, 试图使用 Input 函数读出整个文件的任何尝试都会导致错误发生。在用 Input 函数读出二进制文件时, 要用 LOF 和 Loc 函数来替换 EOF 函数, 或者将 Get 函数与 EOF 函数配合使用。对于为 Output 打开的文件, EOF 总是返回 True。

### 16.4.6 获取打开文件的大小 (LOF 函数)

LOF 函数返回一个 Long (长整数) 型值, 表示用 Open 语句打开的文件的大小, 该大小以字节为单位。

语法:

LOF(filename)

filename: 必要参数, 是一个 Integer (整数) 值, 包含一个有效的文件号。

例如, 使用 LOF 函数来得知已打开文件的大小, 代码如下:

```
Dim FileLength
Open "D:\我的文件夹\Myfile.txt" For Input As #1 '打开文件
FileLength = LOF(1) '取得文件长度
```



Close #1

'关闭文件'

注意: 对于尚未打开的文件, 使用 FileLen 函数将得到文件的长度。

## 16.5 顺序文件

教学录像: 光盘\TM\16\顺序文件.exe

顺序文件就是普通的文本文件, 以字符的形式按照先后顺序存储数据。其文件结构简单, 占用的磁盘控件较少, 访问时要按照顺序逐个查找。下面介绍顺序文件的打开与关闭以及读取与写入操作。

### 16.5.1 顺序文件的打开与关闭

#### 1. 打开顺序文件

在对文件进行操作之前首先必须打开文件, 同时指定读写操作和数据存储位置。打开文件使用 Open 语句, Open 语句分配一个缓冲区供文件进行输入/输出之用, 并决定缓冲区所使用的访问模式。

语法:

```
Open FileName For [Input | Output | Append] [ Lock ] As filenumber [ Len = Buffersize ]
```

顺序文件可以有以下 3 种打开方式, 不同的方式可以对文件进行不同的操作。

#### ☒ Input 方式

以此方式打开的文件, 是用来读入数据的, 可从文件中把数据读入内存, 即读操作。FileName 指定的文件必须是已存在的文件, 否则会出错。不能对此文件进行写操作。例如, 用下面的语句打开一个文件。

```
Open "text" For input As #1 '打开输入文件
```

#### ☒ Output 方式

以此方式打开的文件, 是用来输出数据的, 可将数据写入文件, 即写操作。如果 FileName 指定的文件不存在, 则创建新文件; 如果是已存在的文件, 系统不覆盖原文件。不能对此文件进行读操作。例如, 下面的代码用来打开一个输出文件 Path。

```
Open "Path" For Output As #1 '打开输出文件
```

#### ☒ Append 方式

以此方法打开的文件, 也是用来输出数据的。与 Output 方式打开不同的是, 如果 FileName 指定的文件已存在, 不覆盖文件原内容, 文件原有内容被保留, 写入的数据追加到文件末尾; 如果指定文件不存在, 则创建新文件。

下面的代码用于打开 C 盘根目录下的 MyFile.txt 文件, 如果源文件存在, 则写入的数据追加在文件的末尾。

```
Open "c:\MyFile.txt" ForAppend As #1
```

当以 Input 方式打开顺序文件时, 该文件必须已经存在, 否则, 会产生一个错误。然而, 当以 Output 或 Append 方式打开一个不存在的文件, Open 语句首先创建该文件, 然后再打开它。当在文件与程序之间拷贝数据时, 可利用参数 Len 指定缓冲区的字符数。

## 2. 关闭顺序文件

当对顺序文件打开并对其进行读写操作后, 应将文件关闭, 避免占用资源, 使用 Close 语句将其关闭。

语法:

Close [filenumberlist]

filenumberlist: 可选的参数, 表示为文件号的列表, 如: #1, #2。如果省略, 将关闭 Open 语句打开的所有活动文件。

例如, 使用下面的语句关闭一个已打开的文件。

Close #1

 说明: (1) Close 语句用来关闭使用 Open 语句打开的文件。Close 语句具有下面两个作用:

① 将 Open 语句创建的文件缓冲区中的数据写入文件中。

② 释放表示该文件的文件号, 以方便被其他 Open 语句使用。

(2) 若 Close 语句的后面没有跟随文件号, 则关闭使用 Open 语句打开的所有文件。

(3) 若不使用 Close 语句关闭打开的文件, 当程序执行完毕时, 系统也会自动关闭所有打开的文件, 并将缓冲区中的数据写入文件中。但是, 这样执行有可能会使缓冲区中的数据最后不能写入到文件中, 造成程序执行失败。

## 16.5.2 顺序文件的读取操作

要读取文本文件的内容, 首先应使用 Input 方式打开文件, 然后再从文件中读取数据。VB 提供了一些能够一次读写顺序文件中的一个字符或一行数据的语句和函数。下面分别进行介绍。

### (1) Input#语句


Input#语句用于从文件中依次读出数据, 并放在变量列表中对应的变量中。变量的类型与文件中数据的类型要求对应一致。

语法:

Input #filenumber, varlist

filenumber: 必要的参数。任何有效的文件号。

varlist: 必要的参数。用逗号分界的变量列表, 将文件中读出的值分配给这些变量。这些变量不可能是一个数组或对象变量。但是, 可以使用变量描述数组元素或用户定义类型的元素。

 说明: 文件中的字符串数据项若用双引号括起来, 双引号内的任何字符 (包括逗号) 都视为字符串的一部分, 所以若有些字符串数据项内需要有逗号, 最好用 Write 语句写入文件, 再用 Input

语句读出来，这样在文件中存放数据时就不会出现问题。

**例 16.11** 读取文本文件中的内容，并显示在文本框中。运行本实例程序后单击“打开文件”按钮选择某文本文件，显示效果如图 16.14 所示。（实例位置：光盘\TM\sl\16\11）

关键代码如下：

```
Text1.Text = ""
With CommonDialog1
    .InitDir = App.Path
    .Filter = "文本文件|*.txt"
    .ShowOpen                                '打开“打开对话框”
End With
Open CommonDialog1.FileName For Input As 1    '打开文件
Dim str As String
Do While Not EOF(1)                          '循环至文件尾
    Input #1, str                             '读取文件
    Text1.Text = Text1.Text & vbNewLine & str '将文件赋值给文本框
Loop
Close 1                                       '关闭文件
```

## (2) Line Input#语句

Line Input#语句用于从已打开的顺序文件中读出一行，并将它分配给字符串变量。Line Input#语句一次只从文件中读出一个字符，直到遇到回车符（Chr（13））或回车/换行符（Chr（13）+Chr（10））为止。回车/换行符将被跳过，而不会被附加到字符变量中。

语法：

Line Input #filename, varname

filename: 必要的参数。任何有效的文件号。

varname: 必要的参数。有效的 Variant 或 String 变量名。

**例 16.12** 本实例使用 Line Input#语句读取顺序文件中的数据，运行程序后选择要读取的文件，单击“读取文本”按钮，即可将文本内容显示在文本框中。程序运行效果如图 16.15 所示。（实例位置：光盘\TM\sl\16\12）

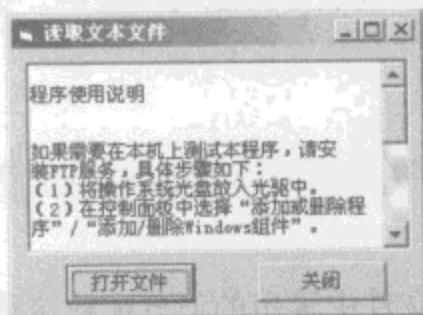


图 16.14 读取文本文件

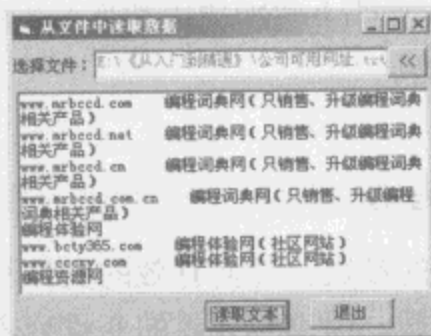


图 16.15 从文件中读取数据

程序代码如下：

```
Private Sub Command1_Click()
    Dim MyLine
```

```

If Text2.Text <> "" Then
    Open Text2.Text For Input As #1          ' 打开文件
    Do While Not EOF(1)                     ' 循环至文件尾
        Line Input #1, MyLine               ' 读入一行数据并将其赋予某变量
        Text1.Text = Text1.Text + MyLine + vbCrLf ' 在立即窗口中显示数据
    Loop
    Close #1                                ' 关闭文件
End If
End Sub
Private Sub Command2_Click()
    CommonDialog1.InitDir = App.Path
    CommonDialog1.Filter = "文本文件|*.txt"
    CommonDialog1.ShowOpen
    Text2.Text = CommonDialog1.FileName
End Sub
Private Sub Command3_Click()
    End
End Sub

```

### (3) Input 函数

Input 函数用于返回字符串类型的值。Input 函数只用于以 Input 或 Binary 方式打开的文件，它包含以 Input 或 Binary 方式打开的文件中的字符。通常用 Print#或 Put 语句将 Input 函数打开的数据写入文件。

语法：

Input(number, [#]filename)

number: 必要的参数。任何有效的数值表达式，指定要返回的字符个数。

filename: 必要的参数。任何有效的文件号。

例 16.13 读取文本文件中的内容，并显示在文本框中。运行本实例程序后单击“打开文件”按钮选择某文本文件，显示效果如图 16.16 所示。（实例位置：光盘\TM\16\13）

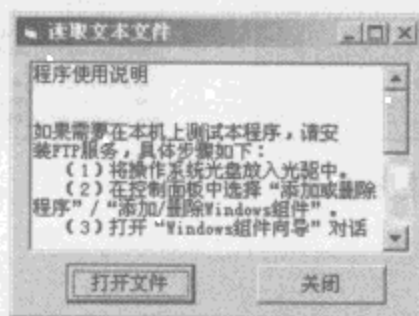


图 16.16 读取文本文件

代码如下：

```

Private Sub Command1_Click()
    With CommonDialog1
        .InitDir = App.Path
        .Filter = "文本文件|*.txt"          ' 指定文件类型
        .ShowOpen                          ' 打开“打开对话框”
    End With
    Open CommonDialog1.FileName For Binary As 1 ' 打开文件
    Text1.Text = Input(LOF(1), 1)           ' 将文件内容写入文本框

```



Close 1  
End Sub

'关闭文件

### 16.5.3 顺序文件的写入操作

在 VB 中对顺序文件进行写操作主要使用 Print #语句和 Write #语句。

#### (1) Print #语句

将格式化显示的数据写入顺序文件中。

语法:

Print #filenumber, [outputlist]

filenumber: 必要的参数。任何有效的文件号。

outputlist: 可选的参数。表达式或是要打印的表达式列表。

⚠ 注意: Print 方法所“写”的对象是窗体、打印机或控件, 而 Print #语句所“写”的对象是文件。  
如下面的语句实现了将 Text1 控件中的内容写入到 #1 文件中:

Print #1, Text1.Text

例 16.14 下面使用 Print #语句将 Excel 中的工资数据导出为网上银行数据, 程序运行效果如图 16.17 所示, 输入相关信息, 单击“导出”按钮, 导出效果如图 16.18 所示。(实例位置: 光盘\TM\sl\16\14)

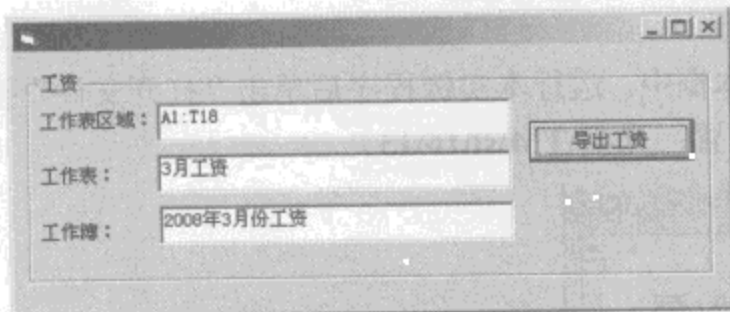


图 16.17 将 Excel 中数据导出为网上银行数据

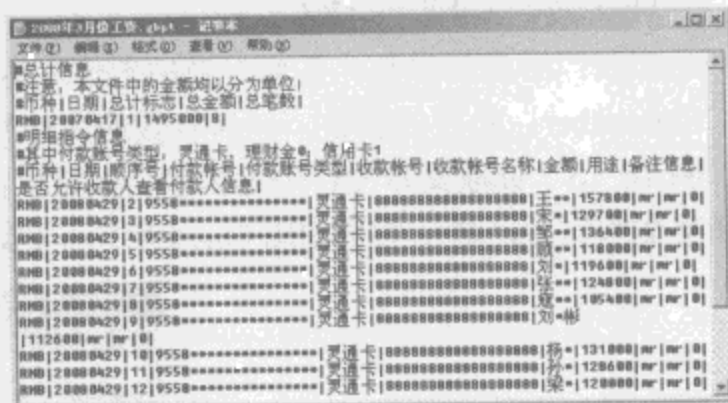


图 16.18 导出后效果

程序代码如下:

```
Private Sub Command1_Click()  
    Dim i As Integer, r As Integer  
    Dim newxls As Excel.Application  
    Dim newbook As Excel.Workbook  
    Dim newsheet As Excel.Worksheet  
    Set newxls = CreateObject("Excel.Application")  
    Set newbook = newxls.Workbooks.Open(App.Path & "\工资数据\" & Text3) '打开文件, 并赋给变量  
    Set newsheet = newbook.Worksheets(Text2.Text) '创建工作表  
    r = newsheet.Range(Text1.Text).Rows.Count '将数量赋给变量  
    Open App.Path & "MyFile.txt" For Output As #1 '打开文本文件  
    '写入信息  
    Print #1, "#总计信息"
```



```

Print #1, "#注意: 本文件中的金额均以分为单位!"
Print #1, "#币种|日期|总计标志|总金额|总笔数|"
Print #1, "RMB|20070417|1|1495000|8|"
Print #1, "#明细指令信息"
Print #1, "#其中付款账号类型: 灵通卡、理财金 0; 信用卡 1"
Print #1, "#币种|日期|顺序号|付款帐号|付款账号类型|收款帐号|收款帐号名称|金额|用途|备注信息|是否允许收款人查看付款人信息|"
For i = 6 To r
    Print #1, "RMB|" & Year(Date) & Format(Month(Date), "00") & Format(Day(Date), "00") & "|" & i - 4 &
        "|9558*****|灵通卡|" & _
        newsheet.Cells(i, 20) & "|" & newsheet.Cells(i, 2) & "|" & newsheet.Cells(i, 18) & "00|mr|mr|0|"
Next i
Print #1, ""
Close
newxls.Quit
End Sub

```

'关闭文件  
'退出 Excel

## (2) Write #语句

将数据写入顺序文件。

语法:

Write #filenumber, [outputlist]

filenumber: 必要的参数。任何有效的文件号。

outputlist: 可选的参数。要写入文件的数值表达式或字符串表达式, 用一个或多个逗号将这些表达式分界。

例 16.15 下例利用 Write #语句向文件中写入数据。(实例位置: 光盘\TM\sl\16\15)

程序代码如下:

```

Private Sub Command1_Click()
    Open App.Path & "MyFile.txt" For Output As #1
    Write #1, 123456789
End Sub

```

'打开输出文件  
'写入以逗号隔开的数字

## (3) Write #语句与 Print #语句的区别

Write #语句通常采用紧凑格式输出, 即各数据项之间用逗号分隔, 在写入文件时, 数据项之间会自动用逗号分界符分隔开。而 Print #语句中的表达式之间因所用分隔符是逗号或分号的不同, 其数据项间的位置也不同, 也不会自动加入定界符。

Write #语句输出字符串时带双引号, 而 Print #语句不带。

Write #语句通常与 Input #读语句配合使用, Print #语句通常与 Line Input #读语句配合使用。

Write#语句通常用于数据写入文件后还要用 VB 程序读出时; 而 Print #语句通常用在向文件中写入数据以后, 要显示或打印出来时作为格式输出语句。

## 16.6 随机文件

 教学录像：光盘\TM\lx\16\随机文件.exe

16.5 节对顺序文件进行了介绍，下面将对随机文件进行介绍。

### 16.6.1 随机文件的打开与关闭

#### (1) 随机文件的打开

随机文件的打开同样使用 `Open` 语句，但是打开模式必须是 `Random` 方式，同时要指明记录长度。文件打开后可同时进行读写操作。

语法：

```
Open FileName For Random [Access access] [lock] As [#]filenumber [Len=reclength]
```

表达式“`Len=reclength`”指定了每个记录的字节长度。如果 `reclength` 比写文件记录的实际长度短，则会产生一个错误；如果 `reclength` 比记录的实际长度长，则记录可写入，只是会浪费一些磁盘空间。例如，可利用下面的语句打开一个随机文件 `MyFile.txt`。

```
Open "C:\MyFile.txt" For Random Access Read As #1 Len = 100
```

#### (2) 随机文件的关闭

随机文件的关闭与关闭顺序文件相同。例如，下面的代码可以将所有打开的随机文件都关闭。

```
Close
```

### 16.6.2 读取随机文件

使用 `Get` 语句可以从随机文件中读取记录。

语法：

```
Get [#]filenumber, [recnumber], varname
```

`Get` 语句中各参数的说明如表 16.3 所示。

表 16.3 参数说明

参 数	描 述
filenumber	必要的参数。任何有效的文件号
recnumber	可选的参数。指出了所要读的记录号
varname	必要的参数。一个有效的变量名，将读出的数据放入其中

**例 16.16** 本实例利用 `Get` 语句将文件中的记录读取到变量中，单击窗体，并将其输出到窗体上。如图 16.19 所示。（实例位置：光盘\TM\sl\16\16）

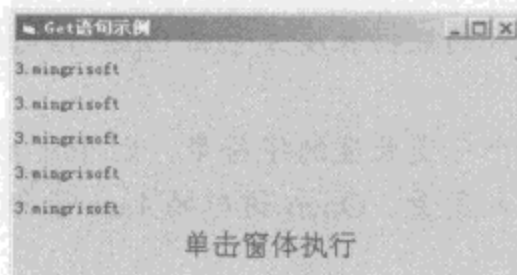


图 16.19 Get 语句示例

程序代码如下：

```
Private Type Record      '定义用户自定义的数据类型
    ID As Integer
    Name As String * 30
End Type
Private Sub Form_Click()
    Dim MyRecord As Record, Position      '声明变量
    ' 为随机访问打开样本文件
    Open App.Path & "MyFile.txt" For Random As #1 Len = Len(MyRecord)
    ' 使用 Get 语句来读样本文件
    Position = 3 ' 定义记录号
    Get #1, Position, MyRecord            '读第 3 个记录
    Print MyRecord.Name
    Close #1                              '关闭文件
End Sub
```

### 16.6.3 写入随机文件

Put #语句可以实现将一个变量的数据写入磁盘文件中。

语法：

```
Put [#]filenumber, [recnumber], varname
```

Put #语句中各参数的说明如表 16.4 所示。

表 16.4 参数说明

参 数	描 述
filenumber	必要的参数。任何有效的文件号
recnumber	可选的参数。Variant (Long)。记录号 (Random 方式的文件) 或字节数 (Binary 方式的文件)，指明在此处开始写入
varname	必要的参数。包含要写入磁盘的数据的变量名

**注意：**对于以 Random 形式打开的文件，在使用 Put #语句时，应注意以下几点：

- (1) 如果已写入的数据的长度小于用 Open 语句的 Len 子句所指定的长度，则 Put #语句以记录长度为边界写入随后的记录。记录终点与下一个记录起点之间的空白将用现有文件缓冲区内的内容填充。因为填入的数据量无法确定，所以一般说来，最好设法使记录的长度与写入

的数据长度一致。如果写入的数据长度大于由 Open 语句的 Len 子句所指定的长度,就会导致错误发生。

(2) 如果写入的变量是一个可变长度的字符串,则 Put #语句先写入一个含有字符串长度的双字节描述符,然后再写入变量。Open 语句的 Len 子句所指定的记录长度至少要比实际字符串的长度多两个字节。

(3) 如果写入的变量是数值类型的 Variant,则 Put #语句先写入两个字节来辨认 Variant 的 VarType,然后才写入变量。例如,当写入 VarType 3 的 Variant 时,Put #语句会写入 6 个字节,其中,前两个字节辨认出 Variant 为 VarType 3 (Long),后 4 个字节则包含 Long 类型的数据。Open 语句的 Len 子句所指定的记录长度至少需要比储存变量所需的实际字节多两个字节。

**例 16.17** 使用 Put #语句将记录信息写入到 MyFile.txt 文件中。运行程序,如图 16.20 所示。(实例位置:光盘\TM\sl\16\17)

单击“写入”按钮,即可将记录信息写入到 MyFile.txt 文件中;单击“输出”按钮,即可将 MyFile.txt 文件中的内容输入到窗体上。程序代码如下:



图 16.20 Put 语句示例

```
Private Sub Command1_Click()                '将记录写入 MyFile.txt 文件中
Dim MyRecord As Record, recordnumber      '声明变量
' 以随机访问方式打开文件
Open App.Path & "MyFile.txt" For Random As #1 Len = Len(MyRecord)
For recordnumber = 1 To 5                  '循环 5 次
    MyRecord.ID = recordnumber              '定义 ID
    MyRecord.Name = "奥运北京 " & recordnumber '建立字符串
    Put #1, recordnumber, MyRecord          '将记录写入文件中
Next recordnumber
Close #1                                   '关闭文件
MsgBox "已经将记录写入到 MyFile.txt 文件中", vbInformation, "信息提示"
End Sub

Private Sub Command2_Click()                '将写入到 MyFile.txt 文件中的记录输入到窗体上
Dim MyRecord As Record, recordnumber
Open App.Path & "MyFile.txt" For Random As #1 Len = Len(MyRecord)
For recordnumber = 1 To 5
    Get #1, recordnumber, MyRecord
    Print MyRecord.Name
Next recordnumber
Close #1
End Sub
```

## 16.7 二进制文件

教学录像: 光盘\TM\lx\16\二进制文件.exe

二进制文件是二进制数据的集合。二进制文件的访问与随机文件的访问十分类似,不同的是随机



文件是以记录为单位进行读写操作的，而二进制文件则是以字节为单位进行读写操作的。文件中的字节可以代表任何东西。二进制存储密集、控件利用率高，但操作起来不太方便，工作量也很大。

### 16.7.1 二进制文件的打开与关闭

#### (1) 二进制文件的打开

二进制文件一经打开，就可以同时进行读写操作，但是一次读写的不是一个数据项，而是以字节为单位对数据进行访问。任何类型的文件都可以以二进制的形式打开，因此二进制访问能提供对文件的完全控制。

语法：

```
Open pathname For Binary As filenumber
```

可以看出，二进制访问中的 Open 语句与随机存储中的 Open 语句不同，它没有 Len=reclength。如果在二进制访问的语句中包括了记录长度，则被忽略。

下面的语句用于以二进制的形式打开 C 盘根目录下的 MyFile.txt 文件。

```
Open "C:\MyFile.txt" For Binary As #1
```

#### (2) 二进制文件的关闭

二进制文件的关闭和其他文件的关闭相同，利用 Close #filenumber 即可实现。例如，下面的代码用于关闭#1 文件。

```
Close #1
```

### 16.7.2 二进制文件的读取与写入操作

二进制文件的读取和随机文件一样，使用 Get 语句从指定的文件中读取数据，使用 Put 语句将数据写入到指定的文件中。

#### (1) 二进制文件的读操作

二进制文件的读操作可以采用 Get 语句来实现。利用 Get 语句读取二进制文件和读取随机文件是十分相似的。这里不再赘述。

#### (2) 二进制文件的写操作

在二进制文件打开后，可以使用 Put 语句对其进行写操作。

语法：

```
Put [#]filenumber, [recnumber], varname
```

Put 语句将变量的内容写入到所打开的文件的指定位置，它一次写入的长度等于变量的长度。例如，变量为整型，则写入两个字节的数。如果忽略位置参数，则表示从文件指针所指的位置开始写入数据，数据写入后，文件指针会自动向后移动。文件刚打开时，指向第一个字节。如下例利用二进制文件备份数据。

如图 16.21 所示的是数据备份的过程，单击“备份”按钮，利用二进制形式打开系统的数据库



(db\_wygl.mdb)，并将这些数据以二进制的形式读出，再以二进制的形式写入备份的数据库文件。

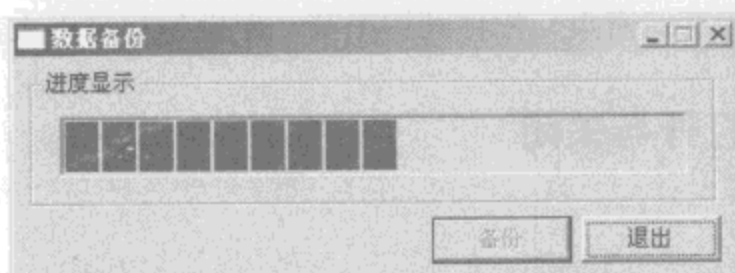


图 16.21 数据备份效果

程序代码如下：

```
Dim str1 As String: Dim str2 As String: Dim char(1 To 5) As Byte '声明两个字符串变量；一个字节类型数组
Private Sub Form_Load()
    str1 = App.Path & "\db_wygl.mdb" '设置源文件
    str2 = App.Path & "\数据备份\db_wygl.mdb" '设置备份目标
End Sub
Private Sub Command1_Click() '开始备份
    a = MsgBox("确定备份数据?", 4, "信息提示") '提示对话框
    If a = vbYes Then '当单击 Yes 按钮时
        Command1.Enabled = False '禁用 Command1
        ProgressBar1.Visible = True '将进度条控件设置为可见
        ProgressBar1.Max = 10 '设置进度条最大值
        ProgressBar1.Min = 0 '设置进度条最小值
        ProgressBar1.Value = 0 '设置进度条初始值
        Open str1 For Binary As #1 '打开源文件
        Open str2 For Binary As #2 '打开备份目标文件
        Dim i As Integer '声明整型变量
        For i = 1 To FileLen(str1) Step 5 '创建步长为 5 的 For 循环
            Get #1, i, char '获取字节
            Put #2, i, char '写入字节
            ProgressBar1.Value = CInt(i * 10 / FileLen(str1)) '设置进度条控件进度
        Next i
        Close
        ProgressBar1.Value = ProgressBar1.Min '设置进度为最小值
        MsgBox "数据库备份成功!", , "信息提示"
        Command1.Enabled = True '激活控件
    End If
End Sub
```

## 16.8 文件系统对象

教学录像：光盘\TM\16\文件系统对象.exe

FSO 对象具有文件管理、文件夹管理、驱动器管理等强大的功能。下面将对引用 FSO 类库的方法，以及创建 FSO 对象并实现某管理功能的方法进行介绍。

### 16.8.1 FSO 对象模型

FSO (FileSystemObject) 对象模型提供了一个基于对象的工具来处理文件夹和文件。这使得处理文件夹和文件除了使用VB提供的函数或使用Windows API函数以外,又多了一种方法。

FSO对象模型包含在Scripting类型库 (Sccrun.dll) 中。该对象并不是VB的内置对象,在使用之前首先应将其引用到工程中。方法为:选择“工程”/“引用”菜单项,打开“引用”对话框选中Microsoft Scripting Runtime复选框,如图16.22所示。

单击“确定”按钮,将FSO对象引用到工程中。引用了FSO对象以后,可以使用“对象浏览器”窗口(选择“视图”/“对象浏览器”菜单命令来调用对象浏览器),浏览新增的对象,如图16.23所示。可以在Scripting模块中看到Drive对象、File对象、FileSystemObject对象、Folder对象和TextStream对象。其中,FileSystemObject对象是这些对象中的关键,想要使用其他对象,必须先创建FileSystemObject对象。

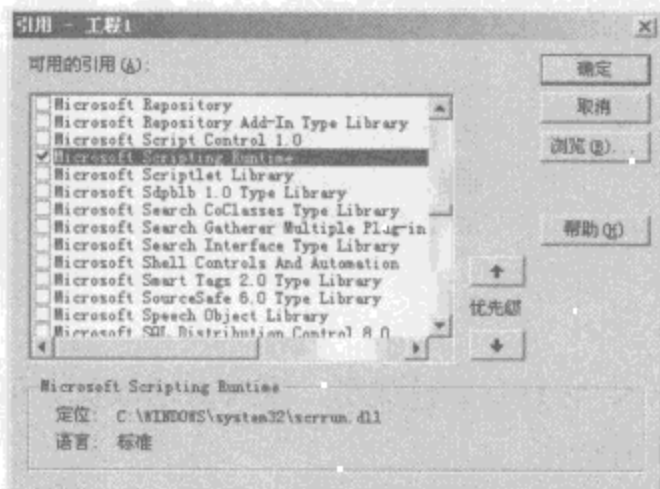


图 16.22 引用 Scripting 类型库

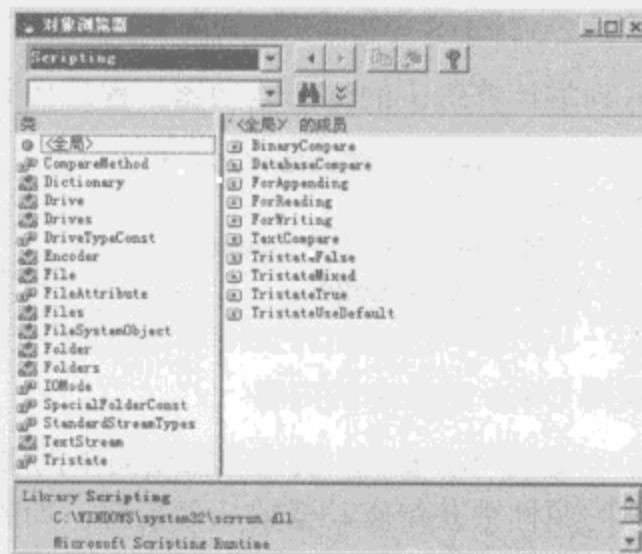


图 16.23 对象浏览器

### 16.8.2 FileSystemObject 对象

FileSystemObject 对象是 FSO 模型中的核心对象,在应用程序中使用 FSO 编程的主要步骤如下。

(1) 创建一个 FileSystemObject 对象。

创建 FileSystemObject 对象有如下两种方法。

☒ 使用 New 关键字声明一个变量为 FileSystemObject 对象类型,如以下代码:

```
Dim MyFSO As New FileSystemObject
```

☒ 使用 CreateObject 方法创建一个 FileSystemObject 对象,如以下代码:

```
Dim MyFSO As Object
Set MyFSO = CreateObject("Scripting.FileSystemObject")
```

(2) 根据程序的需要,通过调用 FileSystemObject 对象的方法来创建一个新的对象。

(3) 通过读取新对象的属性值来获取用户所需的信息。

### 16.8.3 Drive 对象及磁盘驱动器的操作

在 FSO 对象模型中, 驱动器 (Drive) 对象可以对特定磁盘驱动器或网络共享的属性提供访问。通过 Drive 对象, 可以方便地与系统中的驱动器进行信息交互。其中, 驱动器不一定是一个硬盘, 可以是一个 CD-ROM 驱动器、一个 RAM 等, 而且驱动器不一定和系统物理地连接, 也可以通过 LAN 进行逻辑连接。

#### 1. 创建 Drive 对象

可以通过如下两种方法创建并获得驱动器对象。

(1) 通过 FileSystemObject 的 GetDrive 方法。程序代码如下:

```
Dim MyFSO As New FileSystemObject      '创建 FSO 对象
Dim MyDrive As Drive                  '声明 Drive 类型变量
Set MyDrive = MyFSO.GetDrive("c:")    '获得 C 盘驱动器的 Drive 对象
```

(2) 列举计算机中的每个驱动器。程序代码如下:

```
Dim MyFSO As New FileSystemObject      '创建 FSO 对象
Dim MyDrive As Drive                  '声明 Drive 类型变量
For Each MyDrive In MyFSO.Drives      '遍历 Drives 集合
    Print MyDrive.DriveLetter          '显示驱动器名称
Next
```

#### 2. 获得 Drive 对象中的内容

Drive 对象是 Scripting 模块中比较简单的对象之一。它只有属性成员, 没有方法成员。可以通过 Drive 对象的属性来获得驱动器的相关内容, 如表 16.5 所示。

表 16.5 Drive 对象的属性

属 性	描 述
AvailableSpace	返回在指定驱动器或网络共享上的可用空间。以 Byte 为单位
DriveLetter	返回物理本地驱动器或网路共享的驱动器字母代号。只读
DriveType	返回一个值用来表示驱动器的类型, 可用以判断光盘、硬盘等
FileSystem	返回指定驱动器的文件系统类型
FreeSpace	返回指定驱动器上或网络共享的用户可用磁盘剩余空间容量。以 Byte 为单位。只读
IsReady	返回指定的驱动器设备是否准备好。对于外插式驱动器, 必须插入已经格式化的磁盘, 此属性才返回 True; 对于 CD-ROM, 仅当插入了适当的媒体, 并已准备好供访问时, 此属性才返回 True
Path	返回驱动器的路径
RootFolder	返回一个 Folder 对象。该对象表示指定驱动器的根文件夹。只读
SerialNumber	返回标识磁盘的十进制序列号
ShareName	返回网络共享驱动器的名称
TotalSize	返回驱动器或网络共享的总空间大小。以 Byte 为单位
VolumeName	返回或设置制定驱动器的卷标名

可以通过使用 DriveType 属性返回指定驱动器的类型。

语法:

object.DriveType

object: 表示一个 Drive 对象。

DriveType 属性的设置值如表 16.6 所示。

表 16.6 DriveType 属性的设置值

设 置 值	描 述
Unkown (0)	无法判断
Removeable (1)	外插式驱动器, 如软盘
Fixed (2)	硬盘
Remote (3)	远程存储设备
CD-ROM (4)	光驱
RamDisk (5)	RAM Disk

**例 16.18** 本实例利用 Drive 对象的 DriveType 属性来判断计算机中的驱动器类型。运行程序, 在窗体加载时, 遍历计算机中的所有驱动器, 并对驱动器类型进行判断, 同时将其添加到 ListBox 控件中, 如图 16.24 所示。(实例位置: 光盘\TM\sl\16\18)

程序代码如下:

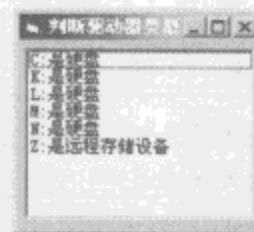


图 16.24 判断驱动器的类型

```

Private Sub Form_Load()
    Dim MyFSO As New FileSystemObject           '创建 FSO 对象
    Dim MyDrive As Drive                        '声明 Drive 类型变量
    For Each MyDrive In MyFSO.Drives            '遍历 Drives 集合
        List1.AddItem MyDrive.Path & "是" & ShowDriveType(MyDrive.DriveType) '将驱动器属性添加到列表
    Next
End Sub

Function ShowDriveType(ByVal NumType As Long) '判断驱动器类型
    Select Case NumType
        Case 0: ShowDriveType = "无法判断"
        Case 1: ShowDriveType = "外插式驱动器"
        Case 2: ShowDriveType = "硬盘"
        Case 3: ShowDriveType = "远程存储设备"
        Case 4: ShowDriveType = "光驱"
        Case 5: ShowDriveType = "RAM Disk"
    End Select
End Function

```

#### 16.8.4 Folder 对象与文件夹的浏览 (获取某路径的文件夹名)

要想从某个文件夹中查找一个文件, 首先得知道该文件夹中含有哪些子文件夹和文件。通过 VB




代码来直接获得是不容易的, 现在可以通过 Folder 对象来实现。

### (1) 创建 Folder 对象

如果希望获得 C:\ 下的文件夹的信息, 可以利用下面的代码创建代表文件夹的 Folder 对象。

```
Dim MyFSO As New FileSystemObject
Dim MyFolder As Folder
Set MyFolder = MyFSO.GetFolder("C:\")
```

 说明: 通过上述代码可以看出, FileSystemObject 对象是 Scripting 模块中的关键, 因此需首先创建 FileSystemObject 对象。在创建时在对象前加上关键字 New, 表示只是定义一个对象变量, 并没有创建该对象。然后, 利用 “Dim MyFolder As Folder” 语句定义一个 Folder 对象, 直到执行 “Set MyFolder = MyFSO.GetFolder("C:\")” 语句时, 才由 FileSystemObject 对象调用其 GetFolder 方法来创建 Folder 对象。

### (2) 获取 Folder 对象中的内容

在创建 Folder 对象之后, 可以通过下面的属性来获得文件夹的相关内容, 如表 16.7 所示。

表 16.7 Folder 对象的属性

属 性	描 述
Attributes	返回或设置文件夹的属性, 如存档、只读、隐藏、系统等
DateCreated	返回指定文件夹的创建日期和时间
DateLastAccessed	返回指定文件夹的最后访问日期和时间
DateLastModified	返回最后一次修改指定文件夹的日期和时间
Drive	返回指定文件所在的驱动器符号
Files	返回该文件夹包含的文件
IsRootFolder	返回指定的文件夹是否为根文件夹
ParentFolder	返回指定的文件夹是否为父文件夹
Path	返回指定文件夹的完整的路径名
ShortName	返回文件夹名称的短名字, 如文件夹的名称是 “longfilenamefolder”, 缩写形式为 “longfi~1”
ShortPath	返回文件夹短路径
Size	返回以字节为单位的包含在文件夹中的所有的文件和子文件夹中的大小
SubFolders	返回包含在该文件夹中的所有的子文件夹
Type	返回关于某个文件夹类型的信息, 如对于 .txt 文件, 返回 “Text Document”

SubFolders 属性用于返回包含所有文件夹的一个 Folders 集合, 这些文件夹包含在某个特定的文件夹中的包含隐藏和系统文件属性的那些文件。

语法:

```
object.SubFolders
```

object: 表示一个 Folder 对象。

下例中利用 Folder 对象的 SubFolders 属性显示 C 盘下的子文件夹。运行程序, 单击窗体, 即可在窗体上显示出 C 盘下的子文件夹的名称。程序代码如下:



```

Private Sub Form_Click()
    Dim MyFSO As New FileSystemObject      '创建 FSO 对象
    Dim MyFolder As Folder                 '声明文件夹类型变量
    Dim sFolder As Folder                 '声明文件夹类型变量
    Set MyFolder = MyFSO.GetFolder("C:\")  '获取指定路径下的所有文件夹
    Print MyFolder.Path & "下的子文件夹有: "
    For Each sFolder In MyFolder.SubFolders
        Print sFolder.Name                '遍历文件夹集合
        '显示文件夹名称
    Next
End Sub

```

### 16.8.5 File 对象与文件的操作

File 对象和 Folder 对象一样都含有 Attributes、DateCreated、DateLastAccessed、DateLastModified、Drive、Name、ParentFolder、Path、ShortName、ShortPath、Size、Type 等属性。其属性的意义与 Folder 对象的同名称的属性意义相同。

**例 16.19** 下面利用 File 对象显示文件中选择目录下的所有文件。本程序显示多媒体文件夹中的录像文件，运行程序，选择要查看的章名，在列表框中将显示所在章节文件夹中的录像文件，效果如图 16.25 所示。（实例位置：光盘\TM\sl\16\19）



图 16.25 File 对象应用实例图

程序代码如下：

```

Dim myflashstr1 As String
Dim fs As FileSystemObject                '定义 FileSystemObject 对象类型的变量
Dim f As Folder, faa As Folder            '定义 Folder 对象类型的变量
Dim i As Integer
Dim mystr(10) As String
Private Sub Label1_Click(Index As Integer)
    List2.Clear
    Set faa = fs.GetFolder(App.Path & "\video\" & mystr(Index))  '将文件名赋给变量
    myfoldercount = faa.Files.Count  '将数量赋给变量
    For Each f2 In faa.Files

```

```
List2.AddItem "□" & Left(f2.Name, Len(f2.Name) - 4) '将文件名添加到列表框中
Next
myflashstr1 = App.Path & "\video\" & mystr(Index) & "\" '将路径赋给变量
End Sub
```

### 16.8.6 TextStream 对象与文件的读写

创建 TextStream 对象以加快顺序文件的读写。

#### 1. 创建 TextStream 对象

创建 TextStream 对象的方法如下：

```
Dim MyST As TextStream
```

要想利用 TextStream 对象对文件进行读写操作，必须首先将文件打开。FileSystemObject 对象提供了 OpenTextFile 方法用于打开文件，还可以通过 File 对象的 OpenAsTextStream 方法来实现。

##### (1) FileSystemObject 对象的 OpenTextFile 方法


下例中利用 FileSystemObject 对象的 OpenTextFile 方法打开顺序文件。运行程序，单击窗体，即可将应用程序下的 MyFile1.txt 文件打开，并创建 MyFile2.txt 文件。程序代码如下：

```
Private Sub Form_Click()
    Dim MyFSO As New FileSystemObject '创建 FSO 对象
    Dim MyTS As TextStream '声明 TextStream 类型变量
    Set MyTS = MyFSO.OpenTextFile(App.Path & "MyFile1.txt") '以只读方式打开文件
    '以追加方式打开文件，若文件不存在，则创建文件
    Set MyTS = MyFSO.OpenTextFile(App.Path & "MyFile2.txt", ForAppending, True)
    Print "已经打开 MyFile1，并创建 MyFile2！"
    MyTS.Close
End Sub
```

##### (2) File 对象的 OpenAsTextStream 方法

下例中利用 File 对象的 OpenAsTextStream 方法打开指定的文件。运行程序，单击窗体，即可将应用程序中的 MyFile.txt 文件打开。程序代码如下：

```
Private Sub Form_Click()
    Dim MyFSO As New FileSystemObject '创建 FSO 对象
    Dim MyFile As File '声明 File 类型变量
    Dim MyTS As TextStream '声明 TextStream 类型变量
    Set MyFile = MyFSO.GetFile(App.Path & "MyFile.txt") '创建 File 对象实例
    Set MyTS = MyFile.OpenAsTextStream(ForReading) '创建 TextStream 对象实例
    Print "已经将 MyFile.txt 文件打开！"
    MyTS.Close
End Sub
```

 说明：上面介绍的两种方法都是以只读方式打开文件。其中 ForReading 表示从文件中读出信息，ForAppending 表示向文件中添加信息。

## 2. 获取 TextStream 对象中的内容

创建了 TextStream 对象以后, 可以通过 TextStream 对象的属性 (如表 16.8 所示)、方法 (如表 16.9 所示) 来获得其相关的内容。

表 16.8 TextStream 对象的属性

属 性	描 述
AtEndOfLine	对于以 ForReading 方式打开的 TextStream 文件, 如果文件指针正好在行尾标记的前面, 则该属性返回 True, 否则返回 False
AtEndOfStream	对于以 ForReading 方式打开的 TextStream 文件, 如果文件指针在 TextStream 文件末尾, 则该属性返回 True, 否则返回 False
Column	返回当前字符所在列号。只读
Line	返回当前字符所在的行号。只读

表 16.9 TextStream 对象的方法

方 法	描 述
Close	关闭一个打开的 TextStream 文件
Read(n)	从一个 TextStream 文件中读取 n 个字符并返回得到的字符串
ReadAll	读取整个 TextStream 文件并返回得到的字符串
ReadLine	从一个 TextStream 文件中读取一整行 (截止到换行符, 但是不包括换行符), 并返回得到的字符串
Skip(n)	读取 TextStream 文件时, 跳过 n 个字符
SkipLine	读取 TextStream 文件时, 跳过一行
Write(string)	将字符串写入到 TextStream 文件中
WriteBlankLines(n)	写入 n 个换行符到 TextStream 文件中
WriteLine(string)	写入一个指定的字符串和换行符到一个 TextStream 文件中

**例 16.20** 本实例将使用 TextStream 对象的相关属性和方法, 将指定文件 (MyFile.txt) 中的信息显示在 TextBox 中。运行程序, 如图 16.26 所示。(实例位置: 光盘\TM\sl\16\20)



图 16.26 利用 TextStream 对象显示文件内容

单击“打开”按钮, 选中要打开的文本文件, 即可在 TextBox 控件中显示该文件中的内容。程序代码如下:

```

Private Sub Command1_Click()
    Dim MyFSO As New FileSystemObject
    Dim MyFile As File
    Dim MyTS As TextStream
    Dim MyStr As String
    Dim FileName As String
    CommonDialog1.Filter = "文本文件(*.txt)|*.txt"
    CommonDialog1.Action = 1
    FileName = CommonDialog1.FileName
    Text1.Text = ""
    Set MyTS = MyFSO.OpenTextFile(FileName, ForReading, False)
    Set MyFile = MyFSO.GetFile(FileName)
    Do While Not MyTS.AtEndOfStream
        MyStr = MyTS.ReadLine
        Text1.Text = Text1.Text + MyStr + vbCrLf
        Text1.Refresh
    Loop
End Sub

```

'打开  
 '创建 FSO 对象  
 '声明 File 类型变量  
 '声明 TextStream 类型变量  
 '声明字符串变量  
 '声明字符串变量  
 '指定浏览文件类型  
 '设置文件名  
 '创建 TextStream 对象实例  
 '创建 File 对象实例  
 '遍历行  
 '行读取  
 '逐行显示

## 16.9 小结

本章介绍了文件系统的相关知识以及相关操作，重点介绍了对各种文件的读取和写入操作。通过本章的学习，读者可以了解文件的基本知识，掌握常用的文件操作语句和函数的使用方法，学会对不同文件的基本读写操作。

## 16.10 练习与实践

1. 更改某文件的文件名。（答案位置：光盘\TM\s\16\21）
2. 打开文本文件并显示在文本框中。（答案位置：光盘\TM\s\16\22）
3. 将文本框中内容输出到某文本文件中。（答案位置：光盘\TM\s\16\23）



# 第 3 篇

## 高级应用

- » 第 17 章 API 函数
- » 第 18 章 图形图像技术
- » 第 19 章 多媒体技术
- » 第 20 章 SQL 应用
- » 第 21 章 数据库开发技术
- » 第 22 章 数据库控件
- » 第 23 章 网络编程技术
- » 第 24 章 自动化控制
- » 第 25 章 创建和使用帮助文件
- » 第 26 章 应用程序打包

本篇介绍了 API 函数、图形图像技术、多媒体技术、SQL 应用、数据库开发技术、数据库控件、网络编程技术、自动化控制、创建和使用帮助文件、应用程序打包等。学习完这一部分，能够开发数据库应用程序，多媒体程序和网络程序等。





# 第17章

## API 函数

(教学视频: 18 分钟)

Windows 应用程序接口 (即 API) 是 Windows 提供的重要功能之一。它和 Windows 一起被安装到计算机中, 在 Visual Basic 的应用程序中, 可以像调用普通的过程一样调用 API 函数, 进而实现需要的功能。本章详细地介绍了如何声明调用 API 函数, 在讲解过程中为了便于读者理解结合了大量的实例。

通过阅读本章, 您可以:

- 了解 API 基本概念和其相关的概念
- 掌握 API 浏览器的启动
- 掌握 API 浏览器的加载
- 掌握 API 浏览器的使用
- 掌握 API 函数的声明
- 掌握 API 函数的使用及调用

## 17.1 API 概述

 教学录像：光盘\TM\lx\17\API 概述.exe

API 函数涵盖了计算机操作的各个方面，涉及面广，数量庞大。程序设计人员如果能掌握比较常用的 API 函数，即可以解决很多问题并可以使程序的功能更加完善。

本节将介绍 API 函数的相关概念以及基本使用方法。

### 17.1.1 API 的概念

API (Application Programming Interface) 函数又称“应用程序编程接口”。它是 Windows 提供给应用程序与操作系统之间的接口，犹如建筑工地所使用的“砖瓦”一般，可以搭建出各种丰富多彩的应用界面和功能灵巧的应用程序。所以可以认为 API 函数是构筑整个 Windows 框架的基石。程序设计人员可以在不同的程序设计语言中使用 API 函数，编写运行于 Windows 操作系统的应用程序。

程序员在编写应用程序的时候需要使用一些函数库，所谓函数库就是一些目标代码模块经过组合形成的代码群。应用程序从函数库中调用函数实际上就是通过链接，从而使应用程序能够从函数库中存取和使用目标代码，并把外部函数结合到一个应用程序中的进程。

与函数库链接有两种方法：静态链接和动态链接。

#### 1. 静态链接

静态链接是指在编写应用程序时，如果需要调用运行函数库中已有的函数，程序员无须在自己的源代码中重写函数库中的函数，而只是给出函数名和所需要的参数，就可以执行相应的操作。生成可执行文件时，首先对源程序进行编译，生成目标文件 (.obj)，此时目标文件中只有函数的调用语句，没有函数本身的代码。当用链接程序 (Link) 把目标文件 (.obj) 与库文件 (.lib) 相链接，生成可执行文件 (.exe) 时，链接程序从库函数文件中取出源程序中所需要的库函数，并加入到链接生成的可执行文件中。此时，可执行文件实际上包含了源程序及其所调用的库函数的代码。

#### 2. 动态链接

API 函数由许多能完成不同操作的动态链接库所组成。动态链接库文件的扩展名为 DLL，这是 Windows 系统中的一种特殊的可执行文件。在动态链接库中包含将某些函数预先编译成目标文件的程序模块，当程序访问所需的动态链接库时，Windows 才能确定被调函数的地址并将其连接到应用程序之中。使用动态链接库可以极大地加速 Windows 应用程序开发的过程。有了这些控件和类库，程序员便可以把主要精力放在程序整体功能的设计上，而不必过于关注技术细节。

Windows 提供了数以千计的 API 函数，这些函数存放在不同的动态链接库中。动态链接库按照功能可以分为三大类 (KERNEL、GDI、USER) 和一些较小的动态链接库。下面介绍 Windows 中常用的动态链接库，其功能如表 17.1 所示。

表 17.1 Windows 中常用的动态链接库

动态链接库	说 明
KERNEL32.DLL	低级内核函数。用于内存管理、任务管理、文件管理、资源控制及相关操作
GDI32.DLL	图形设备接口库。在该动态链接库中含有与设备输出有关的函数，包含大多数绘图、显示环境、图元文件、坐标及字体函数
USER32.DLL	与 Windows 管理有关的函数。包含消息、菜单、光标、插入标志、计时器、通信以及其他大多数非显示函数
ADVAPI32.DLL	高级 API 服务，支持大量的 API（其中包含许多安全与注册方面的调用）
COMCTL32.DLL	实现了一个新的 Windows 集，称为 Windows 公共控件（例如 VB 中的 Toolbar、TreeView、ListView、ImageList 等控件都属于这个控件集）
COMDLG32.DLL	通用对话框 API 库
LZ32.DLL	32 位压缩例程
MAPI32.DLL	为应用程序提供一系列电子邮件功能的 API 函数
MPR.DLL	多接口路由库
NETAPI32.DLL	32 位网络 API 库
SHELL32.DLL	32 位 Shell API 库
VERSION.DLL	版本库
WINMM.DLL	Windows 多媒体库
WINSPOOL.DRV	后台打印接口，包含后台打印相关的 API 函数

### 3. 静态链接与动态链接的区别

当使用同一个静态链接的多个应用程序在同时运行时，每个实例都拥有相同的代码，因此重复占用宝贵的内存空间；并且这些应用程序在存储时，也会由于代码相同而浪费磁盘空间。此外，如果修改了函数库中的函数代码，则调用该函数的相应的应用程序就需要重新进行链接。因此，在 Windows 环境下，通常不使用静态的链接方式，而是使用动态链接库（DLL，Dynamic Link Library）。

动态链接与静态链接的不同之处在于链接的过程不同，虽然包含库函数的源程序仍然是被预先编译成目标文件，但是它不再复制到可执行文件中，而是被链接成一种特殊的 Windows 可执行文件，即动态链接库。应用程序运行时，Windows 操作系统检查执行文件，如果需要不包含在可执行文件自身中的函数，Windows 就自动装入指定的 DLL 文件，使得 DLL 文件中的所有函数都能被应用程序访问。到这个时候，Windows 才能确定每个函数的地址并且将其动态地链接到应用程序中。DLL 文件是在运行时被装入的，并且所有使用这个 DLL 文件的应用程序都可以在运行时共享它。

## 17.1.2 API 的相关概念

### 1. Win32 API

Win32 API 是 Microsoft 32 位平台的应用程序编程接口，所有运行在 Win32 平台上的应用程序都可以调用它。所有的 Microsoft 32 位平台都支持统一的 API（包括函数、结构、消息、宏和接口）。使用



Win32 API 不仅可以开发适合各种开发平台的应用程序，还可以充分挖掘各种开发平台的潜力，以及利用各种开发平台的功能和属性。

标准的 Win32 API 可以分为窗口管理、图形设备接口 (GDI)、系统服务、窗口通用控制、Shell 特性、国际特性和网络服务几类。

## 2. 句柄

Windows 用一个 32 位的整数对每一个对象进行标识，这个整数就是“句柄”(Handle)。句柄是操作系统定义的用来唯一标识对象的整数。每个句柄都是一个类型标识符，以小写字母“h”开头。通过句柄，应用程序才能访问信息，才能借助系统完成实际工作，这是 Windows 系统在多任务环境下保护信息的一种途径。例如窗口句柄 hWnd (Windows Handle)、设备环境句柄 hDC (Device Context Handle)、图形接口对象句柄 GDI Object Handle 等。表 17.2 列出了一些常用的句柄。

表 17.2 常用句柄及其说明

对 象	句 柄	说 明
Bitmap (位图)	hBitmap	内存中存放图像信息的一个区域
Brush (刷子)	HBrush	绘图时用于填充区域
Cursor (光标)	HCursor	鼠标的光标。最大可为 32×32 的单色位图
Device Context (设备环境)	hDC	设备环境
File (文件)	hFile	磁盘文件对象
Font (字体)	hFont	文本字体
Icon (图标)	hIcon	图标对象
Instance (实例)	hInstance	正在运行的应用程序实例
Memory (内存)	hMem	内存块
Menu (菜单)	hMenu	菜单栏或弹出菜单
Module (模块)	Hmodule / hLibModule	程序模块
Pen (画笔)	hPen	绘图函数中划线的类型
Window (窗口)	hWnd	显示器上面的一个窗口

下面以窗口句柄 (hWnd) 为例，具体介绍句柄的使用方法。窗口句柄 (hWnd) 主要用来标识一个窗口，VB 中的窗体和控件都可以看作是一个窗口。几乎在所有情况下，都需要根据窗口句柄 hWnd 来确定 API 函数应用于哪个窗口。也就是说通过 API 函数在某个窗口中输出信息时，必须提供该窗口的句柄，并把它传送给要调用的 API 函数。

例如，在应用程序中将窗体的标题名称改变，可以使用 API 函数的 SetWindowText 来设置，其代码如下：

```
rs = SetWindowText (Form1.hWnd, "欢迎您的光临！")
```

其中 Form1.hWnd 代表 Form1 窗体的句柄。

⚠ 注意：在程序的运行过程中，窗口句柄 (hWnd) 有可能会改变，因此不要把它存放在一个变量中，而应该将其直接传送给 API 函数。

## 17.2 API 浏览器

📺 教学录像：光盘\TM\17\API 浏览器.exe

API 浏览器可以浏览含有 API 函数的声明、常量和类型，它们存放在文本文件或 Jet 数据库中，可以将其复制到剪贴板中，然后粘贴到 VB 代码中进行使用。下面介绍如何使用 API 浏览器。

### 17.2.1 启动 API 浏览器

启动 API 浏览器一般有两种方法。

☑ 方法一：

单击“开始”/“所有程序”/Microsoft Visual Basic 6.0 中文版/Microsoft Visual Basic 6.0 中文版工具/“API 文本浏览器”命令，即可打开 API 浏览器，如图 17.1 所示。

☑ 方法二：

(1) 启动 VB6，单击“外接程序”菜单中的“外接程序管理器”命令，打开“外接程序管理器”对话框。

(2) 在“外接程序管理器”对话框的“可用外接程序”列表中选择 VB 6 API Viewer 项，然后在“加载行为”选项组中选中“在启动中加载”和“加载/卸载”两个复选框，如图 17.2 所示。

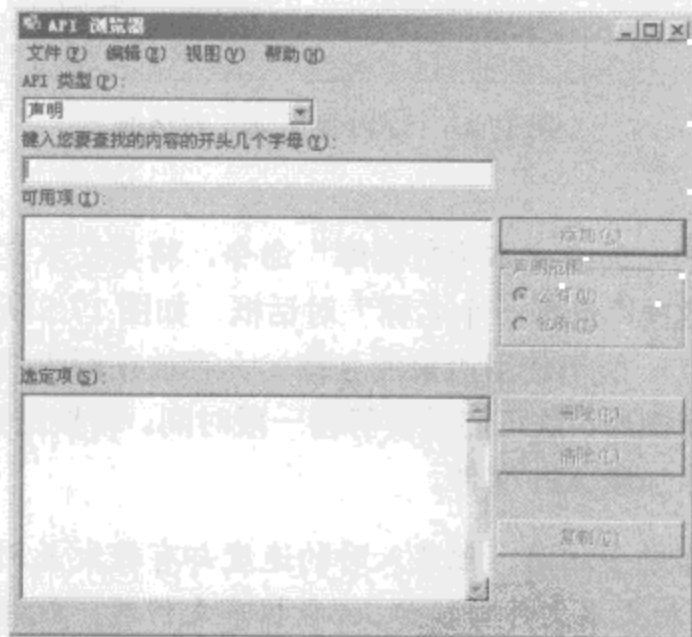


图 17.1 API 浏览器

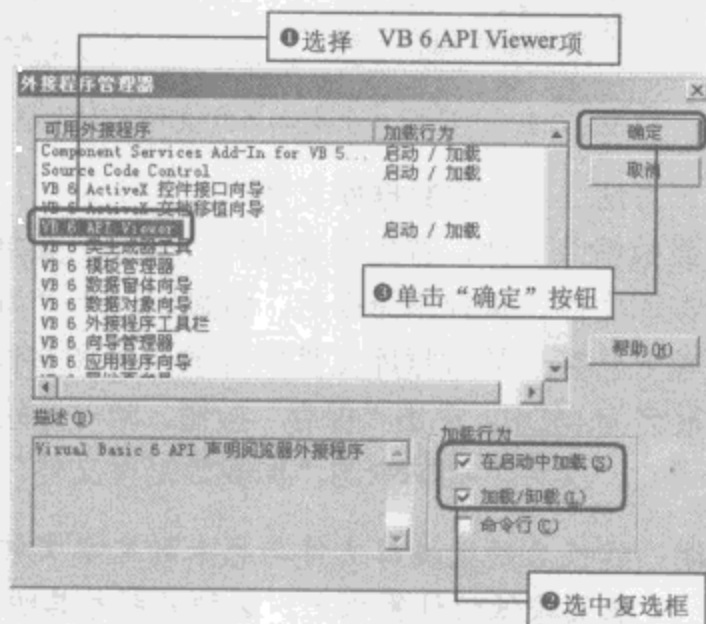


图 17.2 “外接程序管理器”对话框

(3) 单击“确定”按钮，即可把“API 浏览器”命令添加到“外接程序”菜单中。

(4) 单击“外接程序”菜单中的“API 浏览器”命令，即可打开 API 浏览器。

## 17.2.2 API 浏览器的加载

API 函数的相关信息（如类型、声明和常量）存放在两个文本文件中，即 WIN32API.TXT 和 MAPI32.TXT。WIN32API.TXT 中含有 Windows API 函数的 32 位版本的常量、声明和类型，而 MAPI32.TXT 中含有 Windows 多媒体 API 函数的常量、声明和类型。在 API 浏览器中可以读取文本文件或 Jet 数据库文件。下面对其两种加载文件的方法进行介绍。

### ☑ 方法一：加载文本文件

(1) 单击“文件”菜单中的“加载文本文件”命令，打开“选择一个文本 API 文件”对话框，如图 17.3 所示。

(2) 在该对话框中选择 WIN32API.TXT 项，然后单击“打开”按钮，即可装入文本 API 文件，这时 API 浏览器如图 17.4 所示。



图 17.3 “选择一个文本 API 文件”对话框

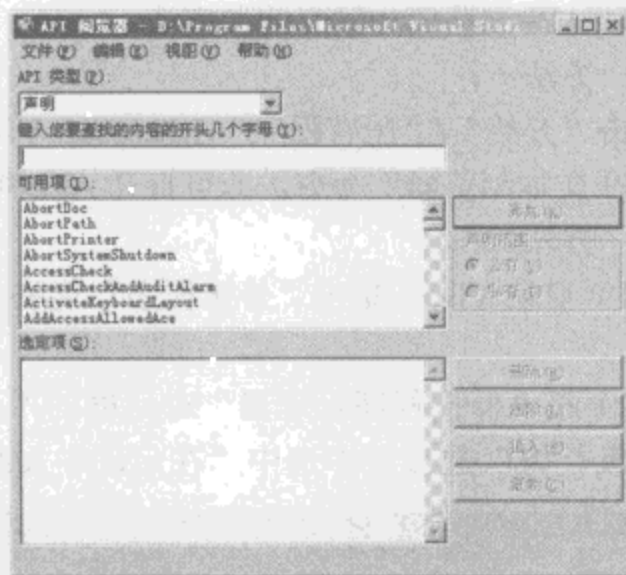



图 17.4 “API 浏览器”对话框

### ☑ 方法二：加载数据库文件

加载数据库文件，首先需要选择“文件”菜单中的“转换文本为数据库”命令，将其文本文件转换为数据库文件。执行该命令后，将显示一个“为新数据库选择一个名称”对话框，如图 17.5 所示，让用户输入转换后的数据库文件的存放位置和文件名（一般仍使用原来的名称）。存放在原来的目录下，扩展名为.MDB，单击“保存”按钮，即可开始转换操作，这可能需要等待一段时间。转换结束后，就可以通过“文件”菜单中的“加载数据库文件”命令加载 API 文件。

 **说明：**转换后的数据库文件与原来的文本文件内容完全相同，但装入时的速度却有较大差别，在配置较低的计算机上，这种差别尤其明显。把文本文件转换为 Jet 数据库文件后，就能以较快的速度装入。

## 17.2.3 API 浏览器的使用

在使用 API 浏览器时，首先需要装入文本文件或数据库文件，才可以对其进行操作（如查看文件

中的声明、常量或类型），然后把它们复制到 VB 代码中即可。下面对其具体操作进行介绍。

### 1. 查看声明、常量或类型

(1) 单击 API 浏览器中“API 类型”下拉列表框，如图 17.6 所示。从中选择“常数”（Constants）、“声明”（Declares）或“类型”（Types），即可在“可用项”列表框中列出相应的项目。

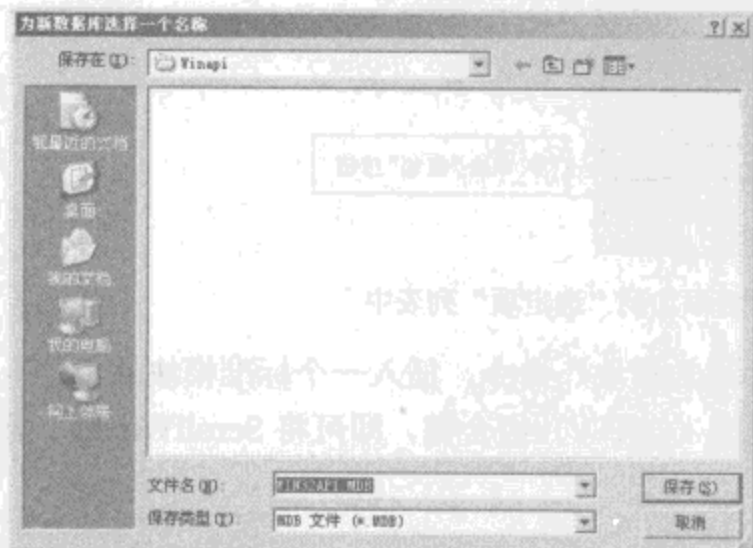


图 17.5 将文本文件转换为数据库文件

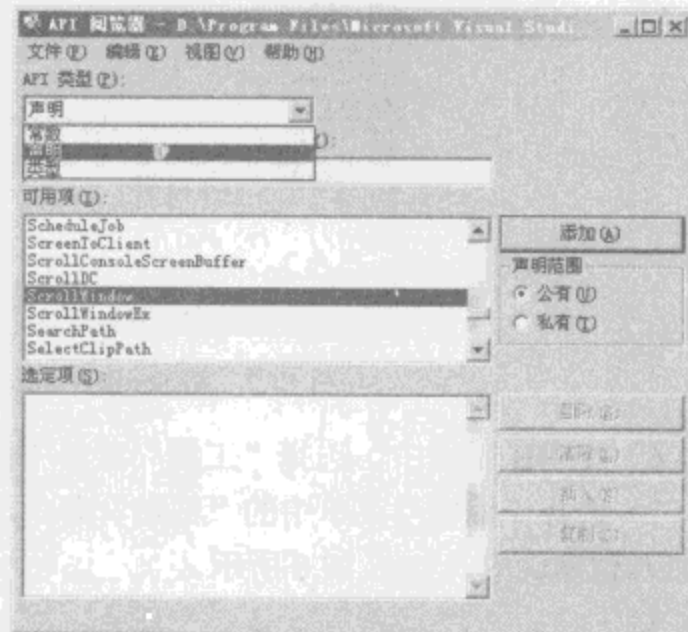


图 17.6 选择需要的 API 类型

(2) “可用项”（Available Item）列表框中的项目是按字母顺序排列的，通过垂直滚动条可以找到所需要的项目，但这样效率太低。为了能较快地找到指定的项目，还可以在“API 类型”下拉列表框下方的“键入您要查找的内容的开头几个字母”文本框中输入要查找的项目的前几个字母，“可用项”列表框中的显示内容将随着变化，当出现所需要的项目后，单击该项目，即可选择该项目。

### 2. 复制声明

为了把指定的项目复制到 VB 代码中，必须先在“可用项”列表框中选择将要复制的项目，然后单击“添加”按钮，把该项目添加到“选定项”（Selected Item）列表框内。此时单击“复制”按钮，即可把“选定项”列表框中的项目复制到剪贴板上，再将该函数的声明粘贴在 VB 代码中即可。

**说明：**在单击“添加”按钮前，可以在“声明范围”选项组中选择“公有”或“私有”。如果要把项目复制到标准模块中，则选择“公有”；如果要把项目复制到窗体模块或类模块中，则选择“私有”。

例如，将 API 函数 ScrollWindow 的声明复制到 VB 的标准模块中，其操作步骤如下。

(1) 在 API 浏览器中加载 WIN32API.TXT 或 WIN32API.MDB 文件，然后在“API 类型”下拉列表框中选择“声明”选项。

(2) 在“键入您要查找的内容的开头几个字母”文本框中输入“ScrollWindow”。

(3) 在“声明范围”选项组中选择“公有”单选按钮。

(4) 单击“添加”按钮，将 ScrollWindow 项目添加到“选定项”列表中，如图 17.7 所示。

(5) 单击“复制”按钮。



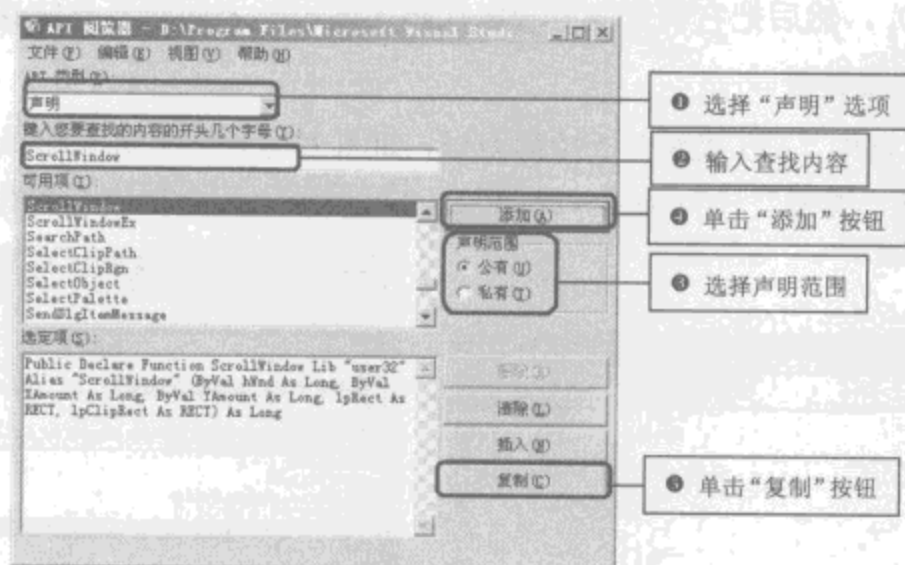


图 17.7 将 ScrollWindow 项目添加到“选定项”列表中

(6) 在 VB 工程中，选择“工程”菜单中的“添加模块”命令，插入一个标准模块。

(7) 选择“编辑”菜单中的“粘贴”命令，或按〈Ctrl+V〉组合键，即可将 ScrollWindow 函数的声明加入到模块中，如图 17.8 所示。

说明：如果在步骤（4）中单击“添加”按钮后，再单击“插入”按钮，可直接将函数的声明插入到 VB 代码窗口中的声明部分。

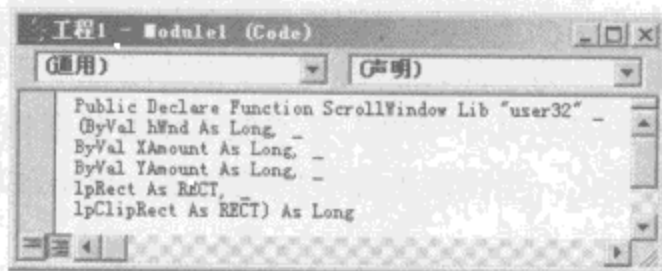


图 17.8 模块中的 ScrollWindow 函数的声明

## 17.3 API 的使用

教学录像：光盘\TM\lx\17\API 的使用.exe

API 函数在使用上很方便，在程序中只要事先对 API 进行了声明，就可以像使用 VB 自身的函数一样，在程序中自由进行使用，而且执行效率要比 VB 自带的函数高。

### 17.3.1 API 函数的声明

声明 API 函数的作用是确定将要使用的 API 函数的名称（有时还需要写出所用的 API 函数的别名）、该 API 函数所在的文件、函数中使用的参数及参数类型、数据传输方式及所用函数本身的函数类型。

在 VB 浏览器中，虽然已经提供了大量的 Windows API 的预定声明，但还是需要知道应该如何亲自编写声明的。具体声明 API 函数的语法格式如下：

[Public | Private]Declare Sub API 函数名 Lib"该函数所在的文件名"[Alias"该函数的别名"][(变量名及变量类型)] As API 函数类型

或

[Public | Private]Declare Function API 函数名 Lib"该函数所在的文件名"[Alias"该函数的别名"][(变量名及变量类型)] As API 函数类型

如果在声明的过程中，不需要返回值，则选择第一种语法格式；如果在声明的过程中，需要返回值，则选择第二种语法格式。声明 API 函数的具体参数说明如表 17.3 所示。

表 17.3 声明 API 函数的参数说明

参 数	说 明
Public	可选项。用于声明在所有模块中都可以使用的过程
Private	可选项。用于声明只能在包含该声明的模块中使用的过程
Sub	可选项。用于表示该过程没有返回值
Function	可选项。用于表示该过程会返回一个可用于表达式的值
Lib	必选项。用于指明包含所声明过程的动态链接库或代码资源。一般情况下 WIN32API 函数总是包含在 Windows 系统自带的或是其他公司提供的动态链接库 DLL 中，而 Declare 语句中的 Lib 关键字就用来指定 DLL（动态链接库）文件的路径，这样 VB 才能找到这个 DLL 文件，然后才能使用其中的 API 函数。如果只是列出 DLL 文件名而不指出其完整的路径，VB 会自动到 EXE 文件所在目录、当前工作目录、Windows\System 目录、Windows 目录或 Path 环境变量中的目录下搜寻这个 DLL 文件。所以如果所要使用的 DLL 文件不在上述几个目录下，应该指明其完整路径
Alias	可选项。用于指明该函数的别名。当外部过程名与某个关键字重名时，可以使用该参数。当动态链接库的过程与同一范围内的公用变量、常数或任何其他过程的名称相同时，也可以使用该参数。如果该动态链接库过程中的某个字符不符合动态链接库的命名约定时，也可以使用该参数。一般来说在 WIN9X 平台下把 API 函数名后加一个大写 A 作为别名即可

注意：（1）Public 和 Private 用于确定该 API 函数的使用范围。如果将 API 函数放在模块里作为公有函数，则在对 API 函数进行声明时，需要使用 Public。如果不进行函数使用范围的说明，系统默认该函数为公有函数。

（2）Alias 关键字用于指明 API 函数的别名后，在函数的实际调用上，是以别名为首要选择的。

### 17.3.2 API 常数与类型

API 常数与类型实际上和 VB 中的常数和数据类型的用法一样。在 API 领域里，各种常数和类型都是预先定义好的。其中定义 API 常数和类型的语法格式与定义自定义常数和自定义数据类型的语法格式基本相同。下面对其定义语法格式进行介绍：

定义 API 常数的语法格式：

Public Const constname [As type] = expression

定义 API 类型的语法格式：

```
Private Type TypeName
    elementname([[subscripts]]) As type
    ...
End Type
```

## 17.4 API 函数的调用

 教学录像：光盘\TM\lx\17\API 函数的调用.exe

在 VB 中，API 函数的调用方式有两种。

方式一：直接调用，注意调用时需要先给变量定义。

方式二：Call 调用。

以 API 函数 ScrollWindow 函数为例，介绍在 VB 中的调用方式。

(1) 在 VB 中声明 ScrollWindow 函数：

```
Private Declare Function SetWindowRgn Lib "user32" (ByVal hwnd As Long, ByVal hRgn As Long, ByVal bRedraw As Boolean) As Long
```

(2) 调用方式：

```
a = SetWindowRgn(Me.hwnd, MyRegion, True)
```

或者

```
Call SetWindowRgn(Me.hwnd, MyRegion, True)
```

## 17.5 小结

在开发灵活、实用且更有效率的应用程序时，往往离不开 API 函数的加盟。本章介绍了 API 相关概念以及在 VB 中的使用方法。通过本章的学习，程序设计人员如果能掌握比较常用的 API 函数，就可以将类库与控件难以解决的问题轻而易举地实现。

## 17.6 练习与实践

1. 尝试利用 API 函数设计一个复制文件内容的小程序，其中主要应用到 CopyFile 函数。将 C 盘根目录下的“1.txt”中的内容复制到 C 盘根目录下的“12.txt”文件中。（答案位置：光盘\TM\sl\17\1）
2. 尝试利用 API 函数设计一个调用画图程序的小程序，其中主要应用到 WinExec 函数。（答案位置：光盘\TM\sl\17\2）
3. 尝试利用 API 函数来控制窗体能否使用，即该窗体禁止所有鼠标及键盘相关操作。其中主要应用到 EnableWindow 函数。（答案位置：光盘\TM\sl\17\3）

# 第18章

## 图形图像技术

(  教学录像: 26 分钟 )

随着计算机软件技术的发展, 程序界面设计已经成为计算机学科分支之一, 简洁、美观、友好的程序界面是软件成功的主要因素之一。要开发一个成功的程序界面, 首先需要掌握的就是 Visual Basic 图形、图像技术。

图形、图像和文本是信息表示的重要形式, 几乎所有的应用程序都要涉及图形、图像和文本的处理。Visual Basic 为应用程序提供复杂的图形、图像和文本功能, 可以使开发出来的应用程序更生动、更具有吸引力, 让用户更方便地进行相关的操作。所以掌握图形、图像技术并将他们运用到软件开发中, 对于软件的发展具有重要的意义。

通过阅读本章, 您可以:

- » 了解图形图像的处理基础
- » 了解 Visual Basic 的坐标系统
- » 掌握如何使用 Visual Basic 的坐标系统
- » 常用的图形方法
- » 掌握常用的图形的属性
- » 掌握图形绘制控件
- » 掌握图像处理控件
- » 掌握常用的图像处理函数



## 18.1 图形图像处理基础

 教学录像：光盘\TM\18\图形图像处理基础.exe

图形界面是 Windows 操作系统的一个显著的特点，可视化的操作为人机交互提供了友好、方便和快捷的环境，深受广大用户的欢迎。要制作美观、友好的图形界面，首先要了解如何查看图形的颜色以及怎样设置图形的颜色等图形的基本操作知识。


### 18.1.1 系统颜色

在应用程序中设置控件或窗体的颜色时，可以不指定颜色而直接使用系统自带的颜色。当用户改变计算机中系统的颜色值时，应用程序将根据用户的定义自动改变颜色值。

对于系统颜色来说，直接设置颜色值的高字节和使用 RGB 函数设置颜色值的高字节不同。对于系统的颜色设置来说高位字节是 80，而对于 RGB 函数的颜色设置值来说高位字节是 0，剩下的数字则表示系统的颜色值。

### 18.1.2 在对象浏览器中查看系统颜色常数

在设计时，如果需要设置窗体或控件的系统颜色，可以在“属性”窗口中，选择“系统”选项卡，即设置其颜色为系统颜色如图 18.1 所示，设置完成后，可以自动转换为十六进制的值。

也可以在“对象浏览器”中的“所有库”中，选择 SystemColorConstants 来显示所有的系统颜色常量。使用对象浏览器可以显示工程和库中的有效的类，包括自己定义的类，用这些类所创建的对象都拥有同样的成员（属性、方法和事件），在对象浏览器中可以查看这些类的对象以及对象的成员。选择“视图”/“对象浏览器”命令、按快捷键〈F2〉或在工具栏上单击“对象浏览器”按钮都可以打开“对象浏览器”对话框。

在打开的“对象浏览器”对话框中的“所有库”中，选择 SystemColorConstants 选项，在其成员列表中即可显示所有的系统颜色常量，如图 18.2 所示。



图 18.1 设置系统颜色

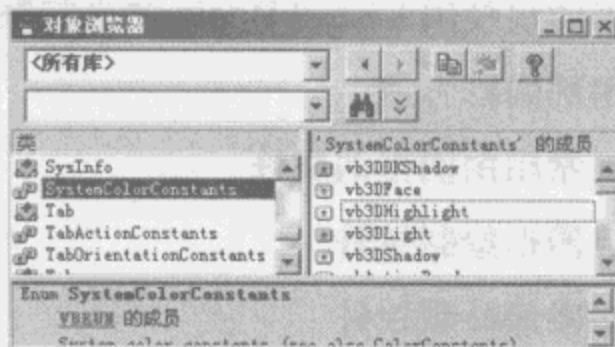


图 18.2 对象浏览器

### 18.1.3 QBColor 函数

在 VB 中提供了两种颜色函数: QBColor 函数和 RGB 函数。其中, QBColor 函数能够返回一个 Long 型值, 用来表示所对应颜色值的 RGB 颜色码, 其语法如下:

QBColor(color)

其中, color 为必要的参数, 是一个介于 0~15 的整型数值, color 的取值含义如表 18.1 所示。

表 18.1 QBColor 函数可选择的颜色

值	颜 色	值	颜 色	值	颜 色	值	颜 色
0	黑色	4	红色	8	灰色	12	亮红色
1	兰色	5	洋红色	9	亮兰色	13	亮洋红色
2	绿色	6	黄色	10	亮绿色	14	亮黄色
3	青色	7	白色	11	亮青色	15	亮白色

例如, 使用 QBColor 函数将 Text1 控件的背景色设置为亮绿色, 代码如下:

Text1.BackColor = QBColor(10)

'Text1 控件的背景颜色为亮绿色

### 18.1.4 RGB 函数

RGB 函数用来表示一个 RGB 颜色值。此函数通常用于和色彩有关的方法或属性上。在编写应用程序时, 如需要使某些记录或标识等显示不同的颜色以表示区分或警示, 可以使用 RGB 函数。RGB 函数和 QBColor 函数在使用范围上大致相同, 但是在颜色的显示上, RGB 函数要比 QBColor 函数更加丰富多彩。

语法:

RGB(red, green, blue)

RGB 函数的参数说明如表 18.2 所示。

表 18.2 RGB 函数的参数说明

参 数	说 明
red	必要的参数, Variant (Integer), 数值范围从 0~255, 表示颜色的红色成分
green	必要的参数, Variant (Integer), 数值范围从 0~255, 表示颜色的绿色成分
Blue	必要的参数, Variant (Integer), 数值范围从 0~255, 表示颜色的蓝色成分

⚠ 注意: 使用 RGB 函数设置颜色受系统限制, 如果系统只能显示 16 色, 那么 RGB 函数就不能设置出更多的颜色。

例如, 利用 RGB 函数将 Text1 控件的背景色设置成红色, 代码如下:

Text1.BackColor = RGB(255, 0, 0)

'设置 Text1 控件的背景颜色为红色

## 18.2 坐标系统

 教学录像：光盘\TM\lx\18\坐标系统.exe

在 VB 中，包括系统标准坐标系统和用户自定义坐标系统两种坐标系。坐标系的坐标单位可以分为 Twip、Point、Pixel、Character、Inch、mm、cm 和用户自定义 8 种形式。不同规格的坐标系统只是度量单位和精度改变，坐标轴的长度或者做图区域的大小并不因此而改变。

### 18.2.1 默认的坐标系统

每个容器都有一个二维的坐标系统，一个坐标系统由坐标原点、坐标度量单位、坐标轴的方向 3 个要素构成。坐标度量单位由容器对象的 ScaleMode 属性决定，但是无论 ScaleMode 属性取何值，默认坐标原点 (0, 0) 都为对象的左上角，横向向右为 X 轴正方向，纵向向下为 Y 轴正方向。

当新建一个窗体时，新窗体采用默认坐标系统，原点在窗体的左上角，Height=3600，Width=4800，ScaleHeight=3195，ScaleWidth=4680，单位为缇，即 Twip。

### 18.2.2 自定义的坐标系统

对象的坐标系统允许用户自行定义，自定义坐标系统的方法有两种。

#### 1. 采用 Scale 方法自定义坐标系统

Scale 方法是自定义坐标系统最常用的方法，用来定义 Form、PictureBox 或 Printer 的坐标系统。其语法如下：

[对象].Scale (xLeft, yTop) - (xRight, yBottom)

其中，对象可以是窗体、图片框或打印机，默认为焦点所在的窗体对象。(xLeft, yTop)表示对象的左上角的坐标值，(xRight, yBottom)表示对象右下角的坐标值。

例 18.1 使用 Scale 方法建立如图 18.3 所示的坐标系。(实例位置：光盘\TM\lx\18\1)

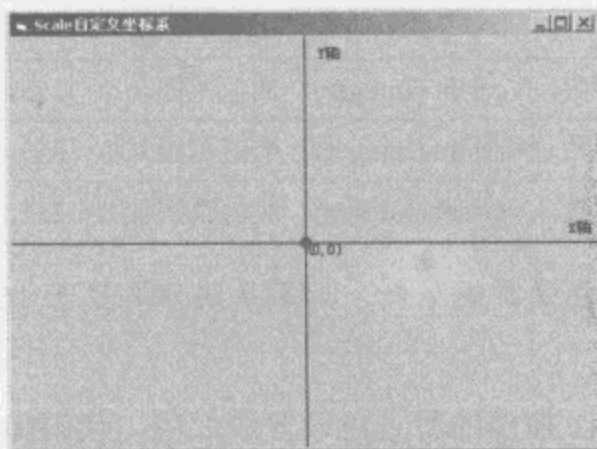


图 18.3 Scale 自定义坐标系

实现的代码如下：

```
Private Sub Form_Activate()
    Form1.Scale (-100, 100)-(100, -100)
    '自定义坐标系
    Line (-100, 0)-(300, 0)      '画 X 轴
    Line (0, 100)-(0, -150)      '画 Y 轴
    DrawWidth = 10                '设置图形的宽度
    Form1.Circle (0, 0), 0, vbRed '绘制红色圆心
    CurrentX = 0: CurrentY = 0: Print "(0,0)"
    '圆心处打印圆心标记 (0, 0)
    CurrentX = 90: CurrentY = 10: Print "X 轴"
    '在 X 轴下方打印 X 轴标记
    CurrentX = 5: CurrentY = 95: Print "Y 轴"
    '在 Y 轴右侧打印 Y 轴标记
End Sub
```

## 2. 使用 Scale 方法的属性自定义坐标系

当对象的 ScaleMode 属性设为数值 0 或者是常量 vbUser 时，用户可以使用 Scale 方法的属性自定义坐标系。Scale 方法的属性如表 18.3 所示。

表 18.3 Scale 方法的属性

属 性	含 义
ScaleHeight 属性	表示新坐标系统绘图区域的高度
ScaleWidth 属性	表示新坐标系统绘图区域的宽度
ScaleLeft 属性	表示新坐标系统绘图区域左上角的水平坐标
ScaleTop 属性	表示新坐标系统绘图区域左上角的垂直坐标

例如，要实现图 18.3 的坐标的关键代码也可以修改如下：

```
Form1.ScaleMode = 0      '指示对象坐标的度量单位为自定义
Form1.ScaleLeft = -100   '设置左上角的水平坐标
Form1.ScaleTop = 100     '设置左上角的垂直坐标
Form1.ScaleWidth = 200   '设置新坐标系统绘图区域的宽度
Form1.ScaleHeight = -200 '设置新坐标系统绘图区域的高度
```

## 18.3 图形外观效果

 教学录像：光盘\TM\18\图形属性.exe

本节主要介绍决定图形外观效果的相关知识，包括绘图坐标，图形位置和大小，图形的边框效果，绘制效果，前景色和背景色，填充效果。



### 18.3.1 绘图坐标

CurrentX 属性和 CurrentY 属性用于返回或者指定下次绘图时的指针位置, 其中, CurrentX 属性表示画笔水平坐标, CurrentY 属性表示画笔垂直坐标。建立窗体或者图片框以后这两个属性的默认值都是 0, 若要改变画笔的位置, 给这两个属性重新赋值即可。在调用相应的绘图方法以后, CurrentX 属性和 CurrentY 属性的值也随之变化。

用表 18.4 所列的图形方法时, CurrentX 和 CurrentY 的设置值按表中说明改变:

表 18.4 CurrentX 和 CurrentY 属性在不同方法中的设置值

方 法	设置 CurrentX,CurrentY 为:	方 法	设置 CurrentX,CurrentY 为:
Circle	对象的中心	NewPage	0, 0
Cls	0, 0	Print	下一个打印位置
EndDoc	0, 0	Pset	画出的点
Line	线终点		

例如, 把当前窗体的绘图坐标恢复到左上角的语句如下:

```
CurrentX = 0
```

'设置当前水平坐标为 0

```
CurrentY = 0
```

'设置当前垂直坐标为 0

### 18.3.2 图形位置和大小

#### 1. 图形位置 (Left 属性和 Top 属性)

Left 属性用于返回或设置图形左边与它的容器的左边之间的距离。

Top 属性用于返回或设置图形内顶部和它的容器的顶边之间的距离。

语法格式如下:

```
object.Left [= value]
```

```
object.Top [= value]
```

object: 对象表达式。

value: 数值表达式, 用于指定距离。

Left 属性和 Top 属性的度量单位取决于所在容器的坐标系统。当用户或使用代码移动对象时, 这些属性值将改变。

#### 2. 图形的大小 (Height 属性和 Width 属性)

图形的 Height 属性和 Width 属性用来设置图形的高度和宽度。

语法格式如下:

```
object.Height [= number]
```

```
object.Width [= number]
```

☑ object: 对象表达式。

☑ value: 数值表达式, 用于指定距离。

例 18.2 本例实现的是将 Shape 控件的大小设置为窗口大小的百分之五十, 并使控件位于窗体中间, 如图 18.4 所示。(实例位置: 光盘\TM\sl\18\2)

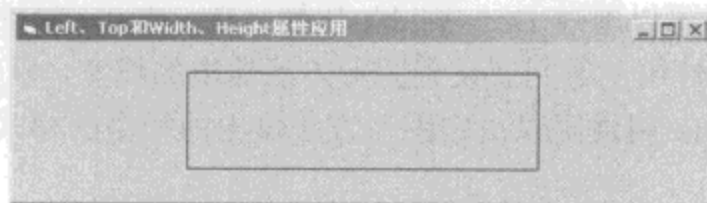


图 18.4 Left、Top 和 Width、Height 属性应用

程序实现代码如下:

```
Private Sub Form_Resize()  
    Shape1.Width = Form1.Width * 0.5           '设置控件的宽度  
    Shape1.Height = Form1.Height * 0.5         '设置控件的高度  
    Shape1.Left = (Form1.ScaleWidth - Shape1.Width) / 2 '在水平方向上居中  
    Shape1.Top = (Form1.ScaleHeight - Shape1.Height) / 2 '在垂直方向上居中  
End Sub
```

### 18.3.3 图形的边框效果

图形的边框效果是通过 BorderStyle 属性、BorderWidth 属性和 BorderColor 属性 3 个属性设置的。其中, BorderStyle 属性用于设置图形的边框的样式, 有 7 种不同的取值。当 BorderStyle 属性取不同值时的边框效果如图 18.5 所示。

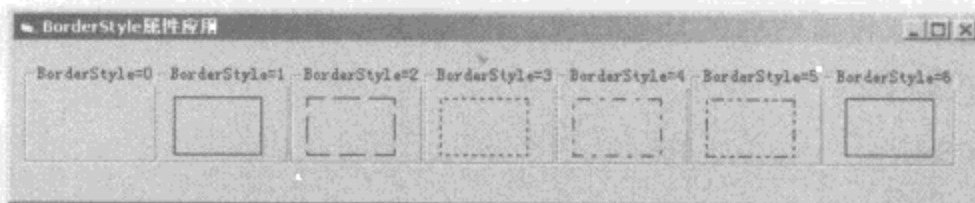


图 18.5 BorderStyle 属性应用

BorderWidth 属性用于设置边框的宽度, 其设置值受 BordStyle 属性影响, 表 18.5 给出了 BorderStyle 设置值对 BorderWidth 属性的影响。

表 18.5 BorderStyle 设置值对 BorderWidth 属性的影响

边框样式	对 BorderWidth 的影响
0	忽略 BorderWidth 设置
1-5	边框宽度从边框中心扩大, 控件的宽度和高度从边框的中心度量
6	边框的宽度在控件上从边框的外边向内扩大, 控件的宽度和高度从边框的外面度量

如果 BorderWidth 属性设置大于 1, 则有效的 BorderStyle 设置值为 1 (实线) 和 6 (内收实线)。

BorderColor 属性用于设置边框的颜色, 使用 Microsoft Windows 运行环境的红-绿-蓝 (RGB) 颜色方案, 有效取值范围是 0~16777215(&HFFFFFF)。

### 18.3.4 绘制效果

使用图形方法绘制图形时可以通过 DrawWidth 属性、DrawStyle 属性和 DrawMode 属性设置图形的效果。其中, DrawWidth 属性用于返回或设置图形方法输出的线宽, 以像素为单位, 取值范围是 1~32767, 默认值为 1。DrawStyle 属性用以决定图形方法输出的线型的样式, 有 7 中不同的取值, 取不同值时的效果如图 18.6 所示。

属性 DrawStyle = 0	线条类型: (默认值) 实线
属性 DrawStyle = 1	线条类型: 虚线
属性 DrawStyle = 2	线条类型: 点线
属性 DrawStyle = 3	线条类型: 点划线
属性 DrawStyle = 4	线条类型: 双点划线
属性 DrawStyle = 5	线条类型: 无线
属性 DrawStyle = 6	线条类型: 内收实线

图 18.6 DrawStyle 属性应用

DrawMode 属性用于返回或设置一个值, 以决定图形方法的输出外观或者 Shape 及 Line 控件的外观。当用 Shape、Line 控件或者图形方法画图时, 使用这个属性产生可视效果。VB 将绘图模式的每一个像素与现存背景色中相应的像素做比较, 然后进行逐位比较操作。

### 18.3.5 前景色和背景色

BackColor 属性返回或设置对象的背景颜色, ForeColor 属性返回或设置在对象里显示图片和文本的前景颜色。语法格式如下:

```
object.BackColor [= color]
object.ForeColor [= color]
```

- ☒ object: 对象表达式。
- ☒ color: 值或常数, 确定对象前景或背景的颜色。

对所有的窗体和控件, 在设计时的默认设置值为: BackColor 设置为由常数 vbWindowBackground 定义的系统默认颜色; ForeColor 设置为由常数 vbWindowText 定义的系统默认颜色。

在 Label 和 Shape 控件中, 如果 BackStyle 属性的设置值为 0 (透明), 则忽略 BackColor 属性。如果在 Form 对象或 PictureBox 控件中设置 BackColor 属性, 则所有的文本和图片, 包括指定的图片, 都被擦除。设置 ForeColor 属性值不会影响已经绘出的图片或打印输出。在其他的所有控件中, 屏幕的颜色会立即改变。

### 18.3.6 填充效果

FillColor 属性返回或设置用于填充形状的颜色, 也可以用来填充由 Circle 和 Line 图形方法生成的圆和方框。默认情况下, FillColor 设置为 0 (黑色)。除 Form 对象之外, 如果 FillStyle 属性设置为默

认值 1（透明），则忽略 FillColor 设置值。

FillStyle 属性返回或设置用来填充 Shape 控件、以及由 Circle 和 Line 图形方法生成的圆和方框的模式。使用语法是：

```
object.FillStyle [= number]
```

其中，number 是一个整数，指定填充样式，有 8 种不同的取值，取不同值时的填充效果说明如表 18.6 所示。

表 18.6 不同 FillStyle 属性值时的填充效果说明

常 数	设 置 值	功 能 描 述	常 数	设 置 值	功 能 描 述
vbFSSolid	0	实线	vbUpwardDiagonal	4	上斜对角线
vbFSTransparent	1	（默认值）透明	vbDownwardDiagonal	5	下斜对角线
vbHorizontalLine	2	水平直线	vbCross	6	十字线
vbVerticalLine	3	垂直直线	vbDiagonalCross	7	交叉对角线

## 18.4 绘图方法

 教学录像：光盘\TM\lx\18\绘图方法.exe

本节介绍绘制图形的方法，包括画点方法 Pset、画线方法 Line、画圆方法 Circle、清屏方法 Cls 等。

### 18.4.1 画点

Pset 方法可以在窗体和图片框的指定位置上，用指定的颜色画点。Pest 方法的格式为：

```
object.PSet [Step] (x, y), [color]
```

其中，object 表示窗体或者图片框的对象名；color 是颜色参数，指定所画点的颜色，可以是颜色函数、长整数或者是颜色常量；[step](x,y)是位置参数，指定画点位置的坐标。如果 Step 关键字省略，则(x,y)指的是绝对坐标，原点在窗体或者图片框的左上角；如果 step 关键字没有省略，则(x,y)指的是相对坐标，是相对于(CurrentX,CurrentY)点的坐标。

例如，在坐标（500，900）处画一个红点，代码如下：

```
CurrentX = 100           '指定当前横坐标
CurrentY = 100          '指定当前纵坐标
PSet (400, 800), RGB(255, 0, 0) '在坐标处绘制红色的点
```

### 18.4.2 画线

Line 方法可以在窗体和图片框的指定位置上，按指定的颜色画直线或者矩形。Line 方法的格式如下：



object.Line [Step] (x1, 1) [Step] (x2, y2), [color], [B][F]

Line 方法的语法的参数说明如表 18.7 所示。

表 18.7 Line 方法的参数说明

参 数	描 述
object	可选的参数。对象表达式。如果 object 省略，则为具有焦点的窗体
Step	可选的参数。关键字，指定起点坐标，它们相对于由 CurrentX 和 CurrentY 属性提供的当前图形位置
(x1, y1)	可选的参数。Single（单精度浮点数），直线或矩形的起点坐标。ScaleMode 属性决定了使用的度量单位。如果省略，线起始于由 CurrentX 和 CurrentY 指示的位置
Step	可选的参数。关键字，指定相对于线的起点的终点坐标
(x2, y2)	必需的参数。Single（单精度浮点数），直线或矩形的终点坐标
color	可选的参数。Long（长整型数），画线时用的 RGB 颜色。如果它被省略，则使用 ForeColor 属性值。可用 RGB 函数或 QBColor 函数指定颜色
B	可选的参数。如果包括，则利用对角坐标画出矩形
F	可选的参数。如果使用了 B 选项，则 F 选项规定矩形以矩形边框的颜色填充。不能不用 B 而用 F。如果不用 F 而使用了 B，则矩形用当前的 FillColor 和 FillStyle 属性值填充。FillStyle 属性的默认值为 transparent

例 18.3 使用 Line 方法在窗体上绘制三个矩形。（实例位置：光盘\TM\sl18\3）

程序运行以后如图 18.7 所示，代码如下：

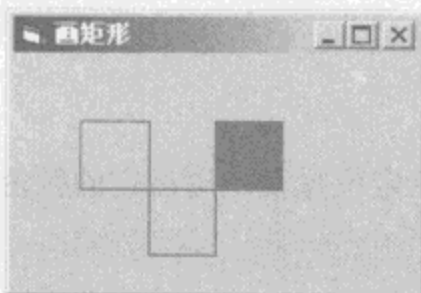


图 18.7 画矩形

```

Private Sub Form_Resize()
    Form1.ForeColor = QBColor(12)           '设置窗体的颜色为红色
    '绘制第一个矩形
    Line (500, 500)-Step(500, 0)           '绘制矩形上面的边
    Line -Step(0, 500)                     '绘制矩形右面的边
    Line -Step(-500, 0)                    '绘制矩形下面的边
    Line -Step(0, -500)                    '绘制矩形左面的边
    Line (1000, 1000)-Step(500, 500), , B   '绘制第二个矩形
    Line (1500, 500)-(2000, 1000), , BF    '绘制第三个矩形
End Sub

```

说明：程序第一行用于指定窗体的前景颜色。第一个矩形是通过绘制矩形的 4 条边完成的。

### 18.4.3 画圆

Circle 方法可以在指定的窗体或者图片框上,用指定的颜色画圆、椭圆或者一段弧。Circle 方法的语法格式为:

```
object.Circle [Step] (x, y), radius, [color, start, end, aspect]
```

其中, (x,y)是必需的参数, Single (单精度浮点数), 代表圆、椭圆或弧的中心坐标; radius 是必需的参数, Single (单精度浮点数), 代表圆、椭圆或弧的半径; start, end 是可选的参数, Single (单精度浮点数), 当弧、部分圆或椭圆画完以后, start 和 end 指定 (以弧度为单位) 是弧的起点和终点位置, 其范围从  $-2\pi$  到  $2\pi$ , 起点的默认值是 0, 终点的默认值是  $2\pi$ ; aspect 是可选的参数, Single (单精度浮点数), 表示圆的纵横尺寸比, 默认值为 1.0, 即它在屏幕上都产生一个标准圆 (非椭圆)。其他参数说明参见 Line 方法中的参数说明。

**例 18.4** 在窗体上绘制一个彩色的圆饼, 圆饼的圆心位于窗体的正中心, 圆饼的颜色由系统随机产生, 并使此圆饼最大。运行效果如图 18.8 所示。(实例位置: 光盘\TM\sl\18\4)

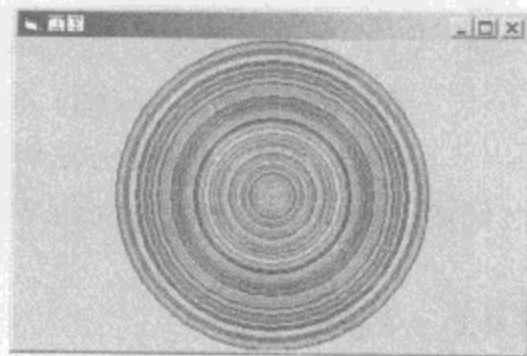


图 18.8 画同心圆

新建一个工程, 在打开的窗体的代码编辑器中输入代码:

```
Private Sub Form_Resize()  
    Dim Xpos, Ypos As Integer           '定义整型变量标记圆心  
    Dim Lim, Rad As Integer             '定义整型变量标记半径  
    Dim R, G, B As Integer              '定义整型变量记录颜色  
    ScaleMode = 3                       '指示对象坐标的度量单位为像素  
    Xpos = Me.ScaleWidth / 2            '确定圆心横坐标  
    Ypos = Me.ScaleHeight / 2           '确定圆心纵坐标  
    If Xpos > Ypos Then                  '确定最大半径  
        Lim = Ypos '如果窗口宽大于窗口高,则 Lim 为高度的一半  
    Else  
        Lim = Xpos '如果窗口宽小于窗口高,则 Lim 为宽度的一半  
    End If  
    For Rad = 0 To Lim '循环画出多个半径逐渐增大的同心圆  
        Randomize                      '初始化随机函数  
        R = 255 * Rnd: G = 255 * Rnd: B = 255 * Rnd '随机产生颜色值  
        Circle (Xpos, Ypos), Rad, RGB(R, G, B) '画圆  
    Next  
End Sub
```

### 18.4.4 清屏

Cls 方法可以清除运行时窗体或者图片框中由 Pset、Line、Circle 等方法所生成的图形和文本, 并使窗体返回到窗体或者图片框的左上角 (即 CurrentX 和 CurrentY 属性复位为 0)。Cls 方法不能清除窗体或图片框的背景色和窗体内部的控件对象。语法格式为:

`object.Cls`

`object` 所在处代表一个对象表达式。

例如, `Form1.Cls` 将清除在窗体 `Form1` 上绘制或者输出的图形文本等。

### 18.4.5 获取颜色值

`Point` 方法可以返回窗体或者图片控件上指定点的颜色, 其语法格式如下。

`object.Point(x, y)`

- ☑ `object`: 可选的参数。一个对象表达式。如果省略 `object`, 则为带有焦点的 `Form` 窗体。
- ☑ `x,y`: 必要的参数。均为单精度值, 指示 `Form` 或 `PictureBox` 的 `ScaleMode` 属性中该点的水平 (`x` 轴) 和垂直 (`y` 轴) 坐标。必须用括号包括这些值。如果由 `x` 和 `y` 坐标所引用的点位于 `object` 之外, `Point` 方法将返回 -1。

例如, `Form1.Point(Me.ScaleWidth / 2, Me.ScaleHeight / 2)` 将返回窗体中心的颜色值。

### 18.4.6 绘制图形

`PaintPicture` 方法用以在 `Form`、`PictureBox` 或 `Printer` 上绘制图形文件 (`.bmp`、`.wmf`、`.emf`、`.cur`、`.ico` 或 `.dib`) 的内容。`PaintPicture` 方法可以取代 Windows 的 API 函数 `BitBlt`, 其语法格式如下。

`object.PaintPicture picture, x1, y1, width1, height1, x2, y2, width2, height2, opcode`

`PaintPicture` 方法的语法的参数说明如表 18.8 所示。

表 18.8 `PaintPicture` 方法的参数说明

参 数	描 述
<code>object</code>	可选的参数。一个对象表达式。如果省略 <code>object</code> , 则为带有焦点的 <code>Form</code> 对象
<code>picture</code>	必需的参数。要绘制到 <code>object</code> 上的图形源。 <code>Form</code> 或 <code>PictureBox</code> 必须是 <code>Picture</code> 属性
<code>x1, y1</code>	必需的参数。均为单精度值, 指定在 <code>object</code> 上绘制 <code>picture</code> 的目标坐标 ( <code>X</code> 轴和 <code>Y</code> 轴)。 <code>object</code> 的 <code>ScaleMode</code> 属性决定使用的度量单位
<code>width1</code>	可选的参数。单精度值, 指示 <code>picture</code> 的目标宽度。 <code>object</code> 的 <code>ScaleMode</code> 属性决定使用的度量单位。如果目标宽度比源宽度 ( <code>width2</code> ) 大或小, 将适当地拉伸或压缩 <code>picture</code> 。如果该参数省略, 则使用源宽度
<code>height1</code>	可选的参数。单精度值, 指示 <code>picture</code> 的目标高度。 <code>object</code> 的 <code>ScaleMode</code> 属性决定使用的度量单位。如果目标高度比源高度 ( <code>height2</code> ) 大或小, 将适当地拉伸或压缩 <code>picture</code> 。如果该参数省略, 则使用源高度
<code>x2, y2</code>	可选的参数。均为单精度值, 指示 <code>picture</code> 内剪贴区的坐标 ( <code>x</code> 轴和 <code>y</code> 轴)。 <code>object</code> 的 <code>ScaleMode</code> 属性决定使用的度量单位。如果该参数省略, 则默认为 0
<code>width2</code>	可选的参数。单精度值, 指示 <code>picture</code> 内剪贴区的源宽度。 <code>object</code> 的 <code>ScaleMode</code> 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
<code>height2</code>	可选的参数。单精度值, 指示 <code>picture</code> 内剪贴区的源高度。 <code>object</code> 的 <code>ScaleMode</code> 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
<code>opcode</code>	可选的参数。是长型值或仅由位图使用的代码。它用来定义在将 <code>picture</code> 绘制到 <code>object</code> 上时对 <code>picture</code> 执行的位操作 (例如, <code>vbMergeCopy</code> 或 <code>vbSrcAnd</code> 操作符)

例如, 语句 `Form1.PaintPicture Picture1.Picture,0,200,Picture1.width,Picture1.height` 将复制窗体中的图片框并将其绘制到窗体中。

## 18.5 图像处理函数

 教学录像: 光盘\TM\lx\18\图像处理函数.exe

本节介绍加载图像函数 `LoadPicture` 和保存图像函数 `SavePicture`。

### 18.5.1 加载图像 (LoadPicture 函数)

`LoadPicture` 函数将图形载入到窗体的 `Picture` 属性、`PictureBox` 控件或 `Image` 控件。使用语法是:

```
LoadPicture([filename], [size], [colordepth],[x,y])
```

参数说明参见表 18.9 所示。

表 18.9 LoadPicture 函数的参数说明

参 数	说 明
filename	可选的参数。字符串表达式指定一个文件名。可以包括文件夹和驱动器。如果未指定文件名, <code>LoadPicture</code> 清除图像或 <code>PictureBox</code> 控件
size	可选的参数。如果 filename 是光标或图标文件, 指定想要的图像大小
colordepth	可选的参数。如果 filename 是光标或图标文件, 指定想要的颜色深度
x	可选的参数, 如果使用参数 y, 则必须使用参数。如果参数 filename 是一个光标或图标文件, 则参数为 x 指定想要图片的宽度
y	可选的参数, 如果使用参数 x, 则必须使用参数。如果参数 filename 是一个光标或图标文件, 则参数为 y 指定想要图片的高度

例如, `Set Form1.Icon = LoadPicture("MYICON.ICO")` 语句把 `LoadPicture` 函数的返回值赋给 `Form` 对象的 `Icon` 属性。

### 18.5.2 保存图片 (SavePicture 函数)

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。使用语法是:

```
SavePicture picture, stringexpression
```

其中, `picture` 是产生图形文件的 `PictureBox` 控件或 `Image` 控件; `stringexpression` 参数是欲保存的图形文件名。

例如, 语句 `SavePicture Image1.Picture, "c:\MyPicture.bmp"` 把 `Image` 对象中的图片 "MyPicture.bmp" 保存到 C 盘根目录下。



## 18.6 图形、图像处理控件

 教学录像：光盘\TM\18\图形、图像处理控件.exe

为了处理图形、图像方便，VB 提供了图形、图像处理控件。本节就详细介绍这些控件，包括 Shape 控件、Line 控件、PictureBox 控件和 Image 控件。

### 18.6.1 Shape 控件

形状控件提供了显示一些规则图形的简易方法，可以显示方框、圆或者椭圆，只要设置形状的相应属性，即可得到所需要的效果。形状控件能够响应事件，也可以在运行时动态改变。

下面介绍一下形状控件的常用属性。

#### ☒ Shape 属性

Shape 属性用来设置其显示的形状，有 6 种取值，如表 18.10 所示。

表 18.10 Shape 属性设置值及描述

Shape 属性值	常 数	描 述
0	vbShapeRectangle	矩形（默认值）
1	vbShapeSquare	正方形
2	vbShapeOval	椭圆形
3	vbShapeOval	圆形
4	vbShapeRoundedRectangle	圆角矩形
5	vbShapeRoundedSquare	圆角正方形

例 18.5 当 Shape 属性取不同值时的形状控件的外观如图 18.9 所示。（实例位置：光盘\TM\sl\18\5）  
实现代码如下：

```
Private Sub Form_Load()
    Dim i As Integer
    For i = 0 To 5
        Frame1(i).Caption = "Shape=" & i
        '设置 Frame 控件的标题栏
        Shape1(i).Shape = i
        '设置 Shape 控件的外观
    Next
End Sub
```

'循环设置控件数组

#### ☒ FillStyle 属性

FillStyle 属性可以构成不同的填充效果，可以取得值 0~7 之间的值，不同的取值对应不同的填充效果。

例 18.6 Shape 控件的不同的填充效果。执行后的效果如图 18.10 所示。（实例位置：光盘

\\TMsl\18\6)

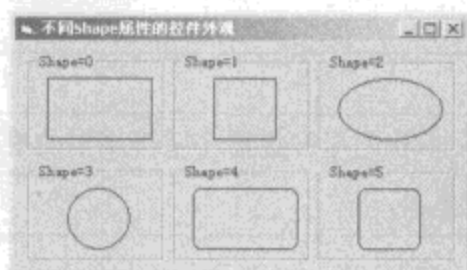


图 18.9 不同 Shape 属性的控件外观



图 18.10 不同 FillStyle 属性取值的控件外观

实现代码如下：

```
Private Sub Form_Load()
    Dim i As Integer
    For i = 0 To 7
        Frame1(i).Caption = "FillStyle=" & i
        Shape1(i).FillStyle = i
    Next
End Sub
```

'循环设置控件数组  
'设置 Frame 控件的标题栏  
'设置形状控件的填充效果

### 18.6.2 Line 控件

Line 控件是图形控件，可以显示水平线、垂直线或者对角线。它用于在界面上绘制线条，主要用于修饰。通过设置线条控件的属性，可以产生不同风格、不同颜色的线条。线条控件的属性主要有 BorderStyle、BorderWidth 和 BorderColor 等。

Line 控件和 Line 方法都可以用来在窗体上绘制直线，但是如果窗体的 AutoRedraw 属性设置为 False，则 Line 方法必须通过 Refresh 方法才能显示出来。而 Line 控件总是能够显示在窗体上，除非将 Visible 属性设置为 False。Line 控件不具有 Move 方法，但是可以通过改变 X1、X2 和 Y1、Y2 属性来移动直线或者调整直线的大小。

### 18.6.3 PictureBox 控件

PictureBox 控件的主要作用是为用户显示图片。实际显示图片由 Picture 属性决定，Picture 属性包括被显示的文件的文件名（及可选的路径名）。

PictureBox 控件常用的属性如表 18.11 所示。

表 18.11 PictureBox 控件的常用属性

属 性	功 能 描 述
AutoSize	返回或设置一个值，以决定控件是否自动改变大小以显示其全部内容
CurrentX	返回或设置下一次打印或绘图方法的水平（CurrentX）坐标。设计时不可用
CurrentY	返回或设置下一次打印或绘图方法的垂直（CurrentY）坐标。设计时不可用
FillColor	返回或设置用于填充形状的颜色

续表

属 性	功 能 描 述
FillStyle	返回或设置用来填充 PictureBox 控件的模式
FontTransparent	返回或设置一个值, 该值用来决定 PictureBox 控件上的背景文本和图形被显示在字符周围的空区
Image	返回持久图形的句柄, 该句柄由 Microsoft Windows 运行环境提供
Picture	返回或设置控件中要显示的图片

其中, Picture 属性是 PictureBox 控件最重要的属性之一。在 PictureBox 控件中显示图片是由 Picture 属性决定的, 有两种方法可以实现图片的添加。

(1) 在设计时加载。在“属性”窗口中找到 Picture 属性, 单击其右边的  按钮, 打开文件对话框, 选择要添加的图片。

(2) 在运行时加载。在运行时可以通过 LoadPicture 函数来设置 Picture 属性, 也可以将其他控件的 Picture 值赋给 PictureBox 控件的 Picture 属性。使用语法如下:


```
object.Picture [= picture]
```

☒ object: 对象表达式。

☒ picture: 字符串表达式, 指定一个包含图片的文件, 可以设置为 Bitmap、icon、metafile、GIF、JPEG 型的图片。

例如, 运行时向 PictureBox 控件中添加图片的代码如下:

```
Picture1.Picture = LoadPicture("D:\图片素材\明日企标.jpg") '动态加载图片
```

 注意: PictureBox 控件具有 AutoSize 属性, 当该属性设置为 True 时, PictureBox 能自动调整大小与显示的图片匹配, 但是此时图片框将不考虑窗体的大小以及窗体上的其他控件。

#### 18.6.4 Image 控件

Image 控件用来显示图形, 它可显示位图、图标、图元文件、增强型图元文件、JPEG 文件或 GIF 文件等几种格式的图形文件。Image 控件使用较少的系统资源, 所以重画起来比 PictureBox 控件要快, 但是它只支持 PictureBox 控件的一部分属性、事件和方法。虽然可以把 Image 控件放在容器里, 但是 Image 控件不能作为容器。常用的属性参见表 18.12 所示。

表 18.12 Image 控件的常用属性

属 性	描 述
BorderStyle	返回或设置对象的边框样式
Stretch	返回或设置一个值, 用来指定一个图形是否要调整大小, 以适应 Image 控件的大小
Picture	返回或设置控件中要显示的图片

其中, Stretch 属性在设计时调整 Image 控件大小, 这时, Stretch 属性决定是否使图片伸缩。若将

属性设置为 True, 则将伸缩 Picture 属性加载的图片。如图 18.11 所示。



图 18.11 不同 Stretch 属性的效果

## 18.7 小结

本章主要介绍了 VB 图形、图像的属性、方法、处理函数以及图形控件等知识, 并通过一些简单实用的例子和图片说明了它们的使用方法。实际上设计图形、图像程序是一个复杂的过程, 本章只是对图形、图像必须掌握的知识进行介绍, 读者可以根据这些知识进行扩展, 从而编写出丰富多彩的图形、图像处理程序。

## 18.8 练习与实践

1. 在绘图或者设计网页时经常需要使用网格, 以方便使用者进行设计。设计一个程序, 在窗体上绘制单元格, 其中单元格的颜色为非活动窗体标题栏的颜色即 &H80000003 或者 vbInactiveTitleBar。运行结果如图 18.12 所示。(答案位置: 光盘\TM\sl\18\7)

2. 复制文件或安装程序时经常要显示一个进度条, 以显示当前的进度, 用 PictureBox 和 Timer 控件设计制作一个类似 ProgressBar 控件的进度条, 运行时如图 18.13 所示。(答案位置: 光盘\TM\sl\18\8)



图 18.12 绘制网格

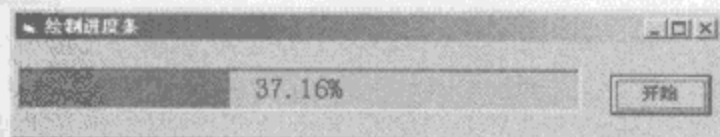


图 18.13 绘制进度条



3. 浏览图片时经常需要翻转图片，设计一个可以翻转图片的程序。运行效果如图 18.14 和图 18.15 所示。（答案位置：光盘\TM\sl\18\9）




图 18.14 翻转前的图片



图 18.15 水平翻转后的图片

# 第19章

## 多媒体技术

(  教学录像: 52 分钟 )

很多程序都具有多媒体的功能,如声音、图像、动画、视频甚至游戏等,这就要求我们掌握一些多媒体技术。在 VB 中,可以通过自身提供的 MMControl 控件和 Animation 控件播放音频、视频和动画,也可以通过 ActiveX 控件中的 MediaPlayer 控件播放 CD、VCD, ShockwaveFlash 控件播放 Flash 动画,并与 VB 程序进行交互。另外,VB 结合 DirectX 还可以开发游戏。

通过阅读本章,您可以:

- » 认识 MMControl 控件,掌握 MMControl 控件的常用属性和事件
- » 掌握 MMControl 控件播放各种音频和视频的方法
- » 认识 Animation 控件,掌握 Animation 控件的常用属性、方法
- » 学会用 Animation 控件播放 AVI 动画
- » 认识 MediaPlayer 控件,了解它的常用属性和方法,以及如何播放视频
- » 学会 ShockwaveFlash 控件加载到程序中的方法
- » 掌握如何使用 ShockwaveFlash 控件播放 Flash 动画,并与 VB 程序进行交互
- » 了解 DirectX,在 VB 程序中使用 DirectX 的 DirectSound 对象编程实现实时混音
- » 通过几个典型的综合应用,进一步掌握多媒体技术在 VB 程序中的应用

## 19.1 MMControl 控件

 教学录像：光盘\TM\lx\19\MM Control 控件

MMControl 控件包含一组高层次的独立于设备的命令，通过这些命令可以控制音频和视频等外围设备，包括 CD、VCD、WAV、MIDI、AVI 等。下面介绍 MMControl 控件的主要属性和事件。

### 19.1.1 认识 MMControl 控件


MMControl 控件属于 ActiveX 控件，使用前应首先将其添加到工具箱中。选择“工程”/“部件”命令，打开“部件”对话框，选择 Microsoft Multimedia Control 6.0 (SP3)，单击“确定”按钮将其添加到工具箱中，在工具箱中双击图标，即可将其添加到窗体上。添加过程如图 19.1 所示。



图 19.1 MMControl 控件添加过程

从图 19.1 可以看出，窗体上的 MMControl 控件由多个按钮组成，这些按钮从左到右依次是：起始点、终止点、播放、暂停、后退、前进、停止、录制和弹出。它们的功能是管理 MCI 设备和播放音频或视频文件。

### 19.1.2 MMControl 控件的属性

#### 1. Command 属性

用于指定将要执行的 MCI 命令，以控制播放、存储多媒体文件，这些命令及功能如表 19.1 所示。

表 19.1 MCI 命令

命 令	功 能
Open	打开 MCI 设备
Close	关闭 MCI 设备
Play	用 MCI 设备进行播放
Pause	暂停播放或录制
Stop	停止 MCI 设备

续表

命 令	功 能
Back	向后步进可用的曲目
Step	向前步进可用的曲目
Prev	使用 Seek 命令跳到当前曲目的起始位置。若在前一个 Prev 命令执行后 3s 内再次执行, 则跳到前一曲目的起始位置; 若已在第一个曲目, 则跳到第一个曲目的起始位置
Next	使用 Seek 命令跳到下一个曲目的起始位置。若已在最后一个曲目, 则跳到最后一个曲目的起始位置
Seek	向前或向后查找曲目
Record	录制 MCI 设备的输入
Eject	从 CD 驱动器中弹出音频 CD
Save	保存打开的文件

实际编程中, 常用命令为 open、play 和 close。例如:

打开一个多媒体文件:

```
MMControl1.FileName = "filename"
MMControl1.Command = "open"
```

上述代码中的 filename 是指定要打开的文件多媒体文件名及路径, 如果需要自动识别该路径, 可将多媒体文件放在工程所在的文件夹, 然后使用 App.Path。

播放多媒体文件:

```
MMControl1.Command = "Open"
```

关闭多媒体文件:

```
MMControl1.Command = "Close"
```

例 19.1 窗体加载时, 播放背景音乐; 窗体卸载时, 关闭背景音乐。(实例位置: 光盘\TM\sl\19\1)  
代码如下:

```
Private Sub Form_Load()
    '播放背景音乐
    With MMControl1
        .Visible=False                '设置 MMControl1 控件不可见
        .FileName = App.Path & "\back\mr.wav"    '指定声音文件
        .Command = "Open"              '打开多媒体文件
        .Command = "play"              '播放多媒体文件
    End With
End Sub
Private Sub Form_Unload (Cancel as Integer)
    Form1.MMControl1.Command = "Close"    '关闭多媒体文件
End Sub
```

## 2. DeviceType 属性

指定要打开的 MCI 设备的类型, 这些类型及说明如表 19.2 所示。



表 19.2 DeviceType 属性设置值

设备类型	设置值	文件类型	说明
CD audio	cdaudio	-	音频 CD 播放器
Digital Audio Tape	dat	-	数字音频磁带播放器
Digital video(not GDI-based)	DigitalVideo	-	窗口中的数字视频
Other	Other	-	未定义 MCI 设备
Overlay	Overlay	-	覆盖设备
Scanner	Scanner	-	图像扫描仪
Sequencer	Sequencer	.mid	音响设备数字接口 (MIDI) 序列发生器
Vcr	VCR	-	视频磁带录放器
AVI	AVIVideo	.avi	视频文件
VCD	mpegVideo	.dat	视频文件
videodisc	Videodisc	-	视频播放器
waveaudio	Waveaudio	.wav	播放数字波形文件的音频设备

DeviceType 属性一般可以不设置,但是以下两种情况必须设置。

(1) 播放 CD、VCD 时,必须指定设备类型。

(2) 如果文件的扩展名没有指定将要使用的设备类型,那么打开复杂 MCI 设备时也必须指定设备类型。

### 3. TimeFormat 属性

用来指定所有位置信息所使用的时间格式,其设置值为 0~10,如表 19.3 所示。

表 19.3 TimeFormat 属性的设置值

值	常量	说明
0	mciFormatMilliseconds	毫秒数用四字节整数变量保存
1	mciFormatHms	小时数、分钟数和秒数被压缩到一个四字节整数中。从最低有效字节到最高有效字节,这四个数分别是:小时数(最低有效字节)/分钟数/秒数/未使用(最高有效字节)
2	mciFormatMsf	分钟数、秒数和帧被压缩到一个四位的整数中。从最低有效字节到最高有效字节,这四个数分别是:分钟数(最低有效字节)/秒数/帧/未使用(最高有效字节)
3	mciFormatFrames	帧用四字节的整数变量保存
4	mciFormatSmppte24	24-帧 SMPTE 将以下数值压缩到一个四字节的整数中。从最低有效字节到最高有效字节,这四个数分别是:小时数(最低有效字节)/分钟数/秒数/帧(最高有效字节)。SMPTE(动画和电视工程师协会)时间是一种绝对的时间格式,它按小时数、分钟数、秒数和帧的格式显示。标准的 SMPTE 的分度类型有 24、25 和 30 帧每秒
5	mciFormatSmppte25	25-帧 SMPTE 按照与 24-帧 SMPTE 相同的顺序将数据压缩到一个四字节变量中
6	mciFormatSmppte30	30-帧 SMPTE 按照与 24-帧 SMPTE 相同的顺序将数据压缩到一个四字节变量中
7	mciFormatSmppte30Drop	30-放下-帧 SMPTE 按照与 24-帧 SMPTE 相同的顺序将数据压缩到一个四字节变量中

续表

值	常 量	说 明
8	mciFormatBytes	字节数用四字节整数变量保存
9	mciFormatSamples	示例用四字节整数变量保存
10	mciFormatTmsf	曲目、分钟数、秒数和帧被压缩到一个四字节整数中。从最低有效字节到最高有效字节，它们分别是：曲目（最低有效字节）/分钟数/秒数/帧（最高有效字节）

### 3. From 属性

指定开始播放文件或录制文件的开始时间。

### 4. To 属性

与 From 属性对应，指定播放文件或录制文件的结束时间。

### 5. Position 属性

该属性用于返回正在播放的多媒体文件的位置，时间单位由 TimeFormat 属性决定。

### 6. Length 属性

用于规定打开的 MCI 设备上多媒体文件的总体播放长度，时间单位由 TimeFormat 属性决定。

### 7. Start 属性

该属性指定当前正在播放的多媒体文件的起始位置，时间单位由 TimeFormat 属性决定。

### 8. Mode 属性

返回打开的 MCI 设备的当前模式，其设置值如表 19.4 所示。

表 19.4 Mode 属性的设置值

值	常数/设备模式	说 明
524	mciModeNotOpen	设备没有打开
525	mciModeStop	停止
526	mciModePlay	正在播放
527	mciModeRecord	正在录制
528	mciModeSeek	正在搜索
529	mciModePause	暂停
530	mciModeReady	设备准备好

**例 19.2** 播放背景音乐，并显示当前状态。（实例位置：光盘\TM\sl\19\2）

（1）启动 VB，新建一个工程，将 MMControl 控件添加到工具箱中。

（2）在窗体上添加一个 MMControl 控件和一个 Label 控件，均使用默认名称。

（3）切换到代码窗口，编写如下代码。

```
Private Sub Form_Load()  
    With MMControl1
```

```

.FileName = App.Path & "\back\mr.wav"           '指定多媒体文件
.Command = "Open"                               '打开多媒体文件
.Command = "play"                               '播放多媒体文件
End With
End Sub
'显示播放状态
Private Sub MMControl1_StatusUpdate()
    Select Case MMControl1.Mode
        Case 524
            Label1.Caption = "设备没有打开"
        Case 525
            Label1.Caption = "停止"
        Case 526
            Label1.Caption = "正在播放"
        Case 527
            Label1.Caption = "正在录制"
        Case 528
            Label1.Caption = "正在搜索"
        Case 529
            Label1.Caption = "暂停"
        Case 530
            Label1.Caption = "设备准备好"
    End Select
End Sub
Private Sub Form_Unload(Cancel As Integer)
    Form1.MMControl1.Command = "Close"           '关闭正在播放的多媒体文件
End Sub

```

按〈F5〉键，运行程序，结果如图 19.2 所示。

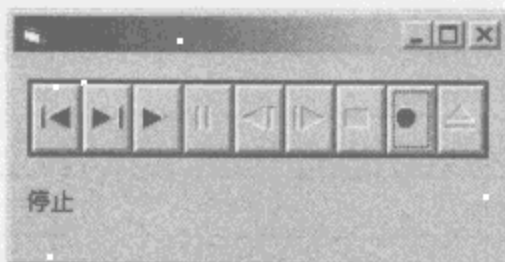


图 19.2 播放 WAV

## 9. Track 属性

表示当前 MCI 设备上可用的曲目个数。例如播放 CD 时，显示当前曲目编号。

```

Private Sub MMControl1_StatusUpdate()
    Label2.Caption = "当前曲目: " & Str$(MMControl1.Track) '显示当前曲目
End Sub

```

说明：如果要获得总曲目数，可以使用 Tracks 属性。

## 10. Error 和 ErrorMessage 属性

使用 Error 和 ErrorMessage 属性，可以处理 MMControl 控件产生的错误。在每个命令后可以检查

错误情况。例如，在 Open 命令之后，可用下面的代码检查 Error 属性的值，以判断是否存在 CD 驱动器。如果没有可用的 CD 驱动器，则返回错误信息。

```
If Form1.MMControl1.Error Then
    MsgBox Form1.MMControl1.ErrorMessage,vbCritical, "未安装 CD 播放器或 CD 播放器不能正常工作"
End If
```

### 19.1.3 MMControl 控件的事件

#### 1. ButtonClick 事件

当用户单击 MMControl 控件的各个命令按钮时，发生该事件。下面给出命令按钮所对应的事件，如表 19.5 所示。

表 19.5 各命令按钮所对应的事件

命令按钮	说明	事件
	倒带	MMControl1_PrevClick
	快进	MMControl1_NextClick
	步进	MMControl1_StepClick
	回倒	MMControl1_BackClick
	暂停	MMControl1_PauseClick
	播放	MMControl1_PlayClick
	录音	MMControl1_RecordClick
	停止	MMControl1_StopClick
	弹出	MMControl1_EjectClick

例 19.3 单击“弹出”按钮，提示光盘弹出，代码如下。

```
Private Sub MMControl1_EjectClick(Cancel As Integer)
    MsgBox "光盘弹出!"
End Sub
```

#### 2. StatusUpdate 事件

按照 UpdateInterval 属性所给定的时间间隔自动地发生。这一事件允许应用程序更新显示，以通知用户当前 MCI 设备的状态，如例 19.2。

#### 3. Done 事件

当 Notify 属性为 True，MCI 命令结束时发生 Done 事件。该事件有一个参数 NotifyCode，该参数表示 MCI 命令是否成功，其设置值如表 19.6 所示。



表 19.6 NotifyCode 参数的设置值

值	常 量	说 明
1	mciSuccessful	命令成功执行
2	mciSuperseded	命令被其他命令所替代
4	mciAborted	命令被用户中断
8	mciFailure	命令失败

例 19.4 当播放完多媒体文件时,将触发 MMControl 控件的 Done 事件,在该事件下将 MMControl 控件的“暂停”和“停止”按钮设置为不可用,代码如下。


```
Private Sub MMControl1_Done(NotifyCode As Integer)
    MMControl.StopEnabled = False: MMControl.PauseEnabled = False    '“暂停”和“停止”按钮不可用
End Sub
```

## 19.2 Animation 控件

 教学录像: 光盘\TM\lx\19\Animation 控件

Animation 控件可以用来播放无声的 AVI 文件。由于它使用独立的线程,因此应用程序可以在播放 AVI 文件的同时做其他的事情,如播放一些小巧的、用于提醒用户注意的动画。例如 Windows 操作系统中的文件复制和查找文件等动画就是用 Animation 控件实现的。

### 19.2.1 认识 Animation 控件

Animation 控件属于 ActiveX 控件,使用前应首先将其添加到工具箱中,选择“工程”/“部件”命令,打开“部件”对话框,选择 Microsoft Windows Common Controls-2 6.0 (SP4),单击“确定”按钮将其添加到工具箱中,在工具箱中双击图标,即可将其添加到窗体上。添加过程如图 19.3 所示。

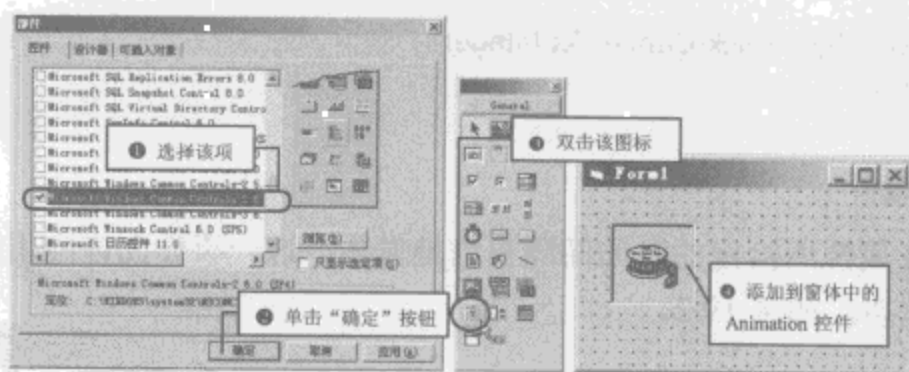


图 19.3 Animation 控件的添加过程

### 19.2.2 Animation 控件的属性

Animation 控件只有一个主要属性,即 AutoPlay 属性。该属性用于在将 avi 文件加载到 Animation

控件时, 返回或设置一个值, 该值确定 Animation 控件是否开始播放该 avi 文件。如果值为 True, 表示播放 avi 文件; 如果值为 False, 表示不播放 avi 文件。

### 19.2.3 Animation 控件的方法

#### 1. Open 方法

Open 方法用于打开一个将要播放的 avi 文件。如果 AutoPlay 属性设置为 True, 则只要加载了该 avi 文件, 就开始播放它。在关闭 avi 文件或设置 Autoplay 属性为 False 之前, 它都将不断重复播放。

例 19.5 窗体启动时, 自动循环播放指定的 avi 文件。(实例位置: 光盘\TM\sl\19\3)

- (1) 启动 VB, 新建一个工程, 将 Animation 控件添加到工具箱中。
- (2) 在窗体上添加一个 Animation 控件, 使用默认名称。
- (3) 切换到代码窗口, 编写如下代码。

```
Private Sub Form_Load()  
    Animation1.AutoPlay = True  
    Animation1.Open App.Path & "\aa.avi"  
End Sub
```

#### 2. Play 方法

在 Animation 控件中播放 avi 文件, 包括 3 个主要参数。其中 Repeat 参数用于指定重复播放的次数, 默认值-1, 它使重复播放次数不受限定; Start 参数用于指定开始的帧, 默认值是 0, 表示在第一帧上开始播放, 最大值是 65535; End 参数用于指定结束的帧, 默认值是-1, 表示上一次播放的帧, 最大值是 65535。

例 19.6 窗体启动时, 播放指定的 avi 文件。(实例位置: 光盘\TM\sl\19\4)

- (1) 启动 VB, 新建一个工程, 将 Animation 控件添加到工具箱中。
- (2) 在窗体上添加一个 Animation 控件, 使用默认名称。
- (3) 切换到代码窗口, 编写如下代码。

```
Private Sub Form_Load()  
    Animation1.Open App.Path & "\aa.avi"  
    Animation1.Play  
End Sub
```

按〈F5〉键, 运行程序, 结果如图 19.4 所示。




图 19.4 播放 AVI

### 3. Stop 方法

Stop 方法用于在 Animation 控件中终止播放 avi 文件。例如终止正在播放的 avi 文件，代码如下。

```
Animation1.Stop
```

 注意：Stop 方法仅终止那些用 Play 方法启动的 avi 动画。当设置 Autoplay 属性为 True 时，使用 Stop 方法终止 avi 动画，会导致返回错误。

### 4. Close 方法

Close 方法使 Animation 控件关闭当前打开的 avi 文件。如果没有加载任何文件，则 Close 方法不执行任何操作，也不会产生任何错误。

例如关闭正在播放的 avi 文件，代码如下。

```
Animation1.Close
```

## 19.3 MediaPlayer 控件

 教学录像：光盘\TM\19\MediaPlayer 控件

本节介绍的 MediaPlayer 控件是 Windows XP 操作系统提供的多媒体播放器 Windows Media Player。它支持音频文件 (\*.wav、\*.mp3 和 \*.mid 等)、影片文件 (\*.avi、\*.mov、\*.mpg 等)、VCD (\*.dat) 文件。另外，MediaPlayer 控件还支持媒体播放列表 (\*.m3u、\*.asx 和 \*.wpl 等)，常用的是 \*.m3u，可以将自己喜欢的音乐保存到一个播放列表中进行播放。

### 19.3.1 认识 MediaPlayer 控件



MediaPlayer 控件是 ActiveX 控件（文件名为 msdxm.ocx），使用时应首先将其添加到工具箱中。选择“工程”/“部件”命令，打开“部件”对话框，选择 Windows Media Player，单击“确定”按钮将其添加到工具箱中，在工具箱中双击  图标，即可将其添加到窗体上。添加过程如图 19.5 所示。



图 19.5 MediaPlayer 控件添加过程

 技巧：如果操作系统中没有该控件，可以通过其他方式获取该控件，然后将其复制到

Windows\System 或 Windows\System32 中, 最后进行注册, 方法可参见 13.1.3 节。

### 19.3.2 MediaPlayer 控件的属性

#### 1. FileName 属性

FileName 属性返回或设置要播放的多媒体文件的名称, 是一个字符串。

指定程序路径下的多媒体文件:

```
MediaPlayer1.FileName = "d:\avi\aa.avi"
```

指定对话框控件提供的多媒体文件:

```
CommonDialog1.ShowOpen  
MediaPlayer1.FileName = CommonDialog1.FileName
```

#### 2. ShowAudioControls 属性

该属性用于设置是否显示声音控制面板。

#### 3. ShowControls 属性

该属性用于设置是否显示控制面板, 下面是显示和不显示的效果, 如图 19.6 和 19.7 所示。



图 19.6 带控制面板的 MediaPlayer



图 19.7 不带控制面板的 MediaPlayer

### 19.3.3 MediaPlayer 控件的方法

#### 1. Open 方法

用于打开媒体播放器设备。

打开指定程序路径下的多媒体文件:

```
MediaPlayer1.Open ("d:\avi\aa.avi")
```

打开对话框控件提供的多媒体文件:

```
CommonDialog1.ShowOpen  
MediaPlayer1.Open (CommonDialog1.FileName)
```



## 2. Play 方法

播放多媒体文件。

例如: MediaPlayer1.Play

## 3. Pause 方法

用于暂停多媒体文件的播放或录制。

例如: MediaPlayer1.Pause

## 4. Stop 方法

用于终止媒体播放器的播放或录制。

例如: MediaPlayer1.Stop

## 19.4 ShockwaveFlash 控件

### 教学录像: 光盘\TM\lx\19\ShockwaveFlash 控件

Flash 是一款功能强大的多媒体工具, 它不仅仅可以制作出丰富多彩的网络动画, 而且还能打造出精彩的 MTV。在 VB 程序中, 可以通过使用 ShockwaveFlash 控件播放 Flash 动画。下面介绍在程序中添加 ShockwaveFlash 控件以及它的属性、方法和事件, 讲解过程中结合了大量的实例。

## 19.4.1 认识 ShockwaveFlash 控件

在 VB 程序中, 可以使用 ShockwaveFlash 控件播放 Flash 动画, 并可以实现暂停、播放、下一帧、上一帧等功能。通过其 FSCommand 命令与 VB 应用程序进行交互, 即通过 Flash 动画中提供的按钮来调用 VB 程序中相应的功能, 如图 19.8 所示。



图 19.8 Flash 与 VB 应用程序交互

ShockwaveFlash 控件是 ActiveX 控件, 主要通过安装 Flash 或者注册 Flash.ocx 文件来获得。使用


ShockwaveFlash 控件前应首先将其添加到工具箱中。选择“工程”/“部件”命令，打开“部件”对话框，选择 Shockwave Flash，单击“确定”按钮将其添加到工具箱中，在工具箱中双击图标，将其添加到窗体上。添加过程如图 19.9 所示。



图 19.9 Flash 控件添加过程

## 19.4.2 ShockwaveFlash 控件的属性

### 1. Movie 属性

Movie 属性是 ShockwaveFlash 控件最常用的属性之一，该属性用来设置一个路径，确定 ShockwaveFlash 控件播放的 Flash 动画文件的所在位置。

例 19.7 窗体启动时，播放 Flash 动画，代码如下。

```
Private Sub Form_Load()  
    ShockwaveFlash1.Movie = App.Path & "\main.swf"  
End Sub
```

'窗体载入时，播放 Flash

### 2. WMode 属性

WMode 属性用于设置 Flash 窗口的模式，包括 3 种模式：Window、Opaque 和 Transparent。在与 VB 结合时，一般将 WMode 属性设置为 Transparent（透明）。方法有两种：一种是在“属性”窗口中找到 WMode 属性，在其旁边的文本框中输入 Transparent，如图 19.10 所示；一种是打开“属性页”，在“窗口模式”下拉列表框中选择 Transparent，如图 19.11 所示。

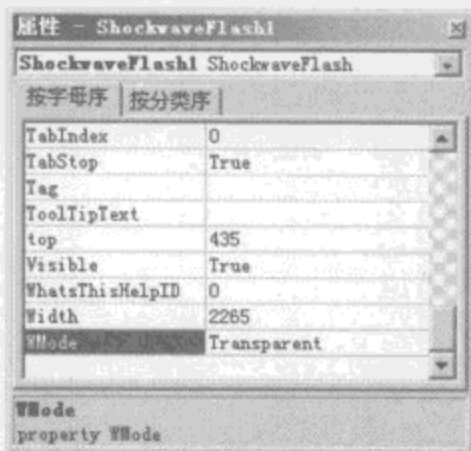


图 19.10 在“属性”对话框中设置 Wmode 属性

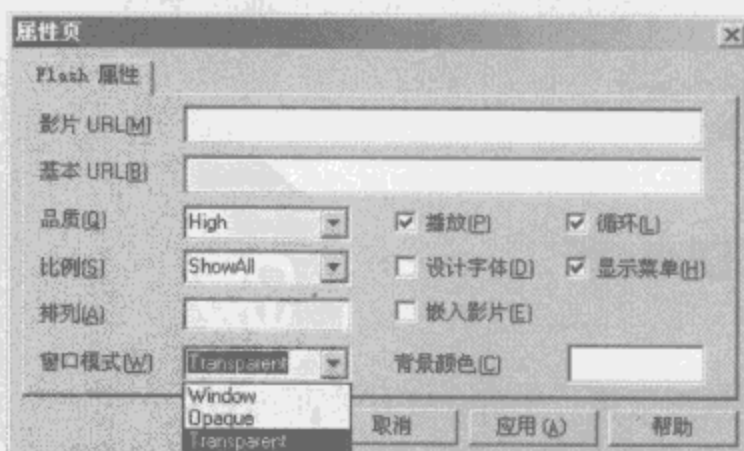


图 19.11 在“属性页”对话框中设置 Wmode 属性

 注意: WMode 属性不能通过代码设置。

### 19.4.3 ShockwaveFlash 控件的方法

Play 方法: 用来播放 ShockwaveFlash 控件加载的 Flash 动画。

Stop 方法: 用来终止 ShockwaveFlash 控件加载的正在播放的 Flash 动画。

Back 方法: 跳到 ShockwaveFlash 控件中的 Flash 动画的上一帧。

Forward 方法: 跳到 ShockwaveFlash 控件中的 Flash 动画的下一帧。


### 19.4.4 ShockwaveFlash 控件的事件

下面介绍 ShockwaveFlash 控件的一个重要的事件——FSCommand()事件。

首先了解一下 Flash 控制 VB 程序的基本原理: 在 Flash 的 ActionScript 里面有个 FSCommand()函数, 该函数可以发送 FScommand 命令, 使动画全屏播放、隐藏动画菜单, 更重要的就是, 它可以与外部文件和程序进行通信。而在 VB 程序中, 我们就是利用 ShockwaveFlash 控件的 FSCommand()事件过程来接收这些命令的, 从而根据不同的命令及参数实现对 VB 程序的控制。

例 19.8 首先用 Flash 制作一个界面和一些交互按钮, 并在每个按钮上面加入如下代码, 并将 Flash 导出成为 swf 文件。(实例位置: 光盘\TM\19\5)

```
on (release) {
    fscommand ("command1");
    //发送 command1 命令
}
```

 说明: command1 是命令的名称, 在实际应用中可以根据该按钮实现的功能进行命名。

然后打开 VB 工程, 加载 ShockwaveFlash 控件, 并使用它的 Movie 属性播放 Flash。

最后在窗体上双击 Flash 控件, 在其 FSCommand 事件过程中编写如下代码。

```
Private Sub ShockwaveFlash1_FSCommand(ByVal command As String, ByVal args As String)
    Select Case command
        Case "command1"
            MsgBox "明日提示", vbInformation, "信息"
        ...
    End Select
End Sub
```

## 19.5 DirectX

 教学录像: 光盘\TM\19\DirectX

DirectX 是微软为了开发游戏程序而研发出来的套件, 它不仅在游戏开发中很实用, 就是在平常的多媒体程序开发上也有其独特的功效。下面介绍 DirectX 的下载和安装, 在 VB 中使用 DirectX, 以

及利用 DirectSound 编程实现实时混音。

### 19.5.1 下载和安装 DirectX

要在 VB 中使用 DirectX, 首先就要考虑 DirectX 的下载和安装。下面介绍如何下载和安装 DirectX。

DirectX 这个开发工具目前是免费供应的, 可在微软的官方网站上免费下载, 下载后即可安装 DirectX。安装 DirectX 与安装一般的应用软件一样, 这里就不多介绍了。

安装完成 DirectX, 在控制面板中就可以看到它了, 如图 19.12 所示, 其中包括 DirectSound、Direct3D、DirectDraw、DirectInput、DirectMusic、DirectPlay 及 Direct Show 7 项, 如图 19.13 所示。

这 7 项是 DirectX 提供的组件, 其功能如表 19.7 所示。



图 19.12 控制面板中的 DirectX

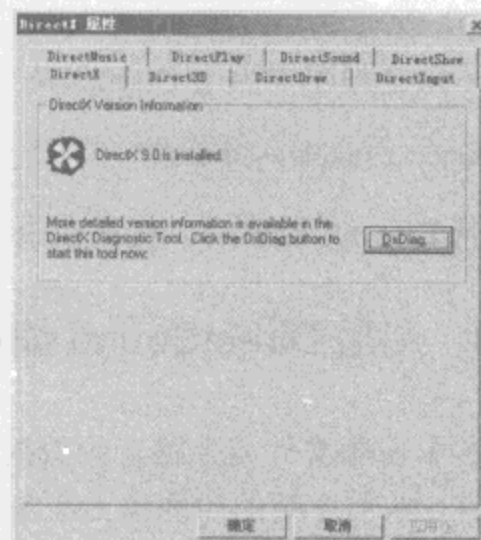


图 19.13 DirectX 属性对话框

表 19.7 DirectX 组件及其功能

组 件	用 途
Direct3D	高速 3D 图象
DirectDraw	高速 2D 图象
DirectInput	面向游戏的对游戏杆和其他输入设备的访问
DirectSound	播放 WAV, 实现实时混音
DirectMusic	播放 MIDI, 实现音效合成
DirectPlay	面向游戏的通信和网络支持
DirectShow	视频流支持

### 19.5.2 在 VB 中使用 DirectX

在 VB 中使用 DirectX, 与使用其他对象一样, 首先要将其引用到工程中。下面以 DirectX 7 为例介绍如何在 VB 中使用 DirectX。选择“工程”/“引用”, 打开“引用”对话框, 选择 DirectX 7 for Visual Basic Type Library, 如图 19.14 所示。



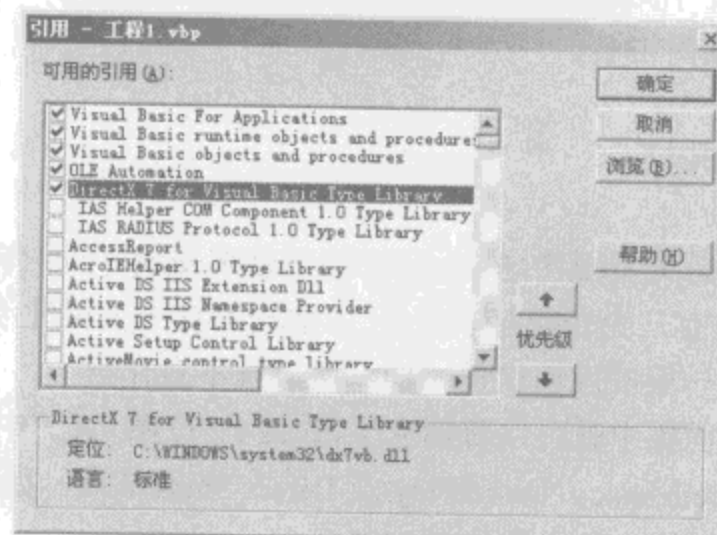


图 19.14 “引用”对话框

接下来在程序中声明 DirectX 7，代码如下。

```
Dim objdx As New DirectX7
```

声明 DirectX 对象后，在程序中就可以使用它了。

### 19.5.3 利用 DirectSound 编程实现实时混音

将多个音频文件或多路音频数据同时输出到音频输出设备上，就可同时听到多个不同的声音，这就是混音。前面介绍的控件都无法实现将播放的音频进行混音，只有 DirectSound 可以。它提供了低延迟混音和 3D 立体音效，支持各种声音格式（只要是 Windows 上可以播放的标准声音格式，DirectSound 都支持），它甚至可以使用两种以上不同格式来进行混音。

在 VB 程序中使用 DirectSound 组件的基本步骤为：新增 DirectX 对象→声明 DirectSound 对象→创建 DirectSound 对象→音效文件载入与设定→音效播放与控制设定。

#### 1. 新增 DirectX 对象

无论使用 DirectX 的哪个组件，都应首先声明一个 DirectX 对象，代码如下。

```
Dim objdx As New DirectX 7
```

#### 2. 声明 DirectSound 对象

一个 DirectSound 对象相当于一个声卡，一般计算机上只会有一个声卡，因此这里声明一个 DirectSound 对象就可以了，声明代码如下。

```
Dim objds As DirectSound
```

声明 DirectSound 对象后，还要声明声音文件的缓冲区对象（DirectSoundBuffer），代码如下。

```
Dim DSBuffer As DirectSoundBuffer
```

#### 3. 创建 DirectSound 对象

创建 DirectSound 对象主要使用 DirectX 对象的 DirectSoundCreate 方法，代码如下。

```
Set objds = objdx.DirectSoundCreate("")
```

#### 4. 设置音效协调层级

设置音效协调层级需要使用 DirectSound 对象的 SetCooperativeLevel 方法。该方法有 4 个参数，分别为 DDSCL\_NORMAL、DDSCL\_PRIORITY、EXCLUSIVE 和 WRITEPRIMARY，一般情况下使用 DDSCL\_NORMAL。

#### 5. 设置缓冲区结构

设置缓冲区结构需要使用 DSBUFFERDESC 对象，在使用该对象前应首先进行声明，代码如下。

```
Dim bufferdss As DSBUFFERDESC
```

DSBUFFERDESC 对象提供了两个属性，即 IFlags 属性和 IBufferBytes 属性，下面介绍这两个属性。

IFlags 属性用于设置缓冲区可用的功能，其设置值如表 19.8 所示；IBufferBytes 属性用于设置缓冲区的大小，单位为 bytes。

表 19.8 IFlags 属性值

属 性 值	说 明
DSBCAPS_CTRL3D	缓冲区有 3D 的控制能力
DSBCAPS_CTRLFREQUENCY	缓冲区有频率的控制能力
DSBCAPS_CTRLPAN	缓冲区有左右声道的控制能力
DSBCAPS_CTRLVOLUME	缓冲区有音量的控制能力
DSBCAPS_LOCHARDWARE	使用声卡上的内存作为缓冲区
DSBCAPS_LOC SOFTWARE	强迫使用软件内存，利用软件来仿真混音
DSBCAPS_PRIMARYBUFFER	指定为主要缓冲区
DSBCAPS_STATIC	指定缓冲区数据为静态数据，并在可能的情况下将缓冲区建立在声卡的内存上

#### 6. 设置 Wave 声音文件结构

设置 Wave 声音文件结构需要使用 WAVEFORMATEX 对象，使用该对象前应首先进行声明，代码如下。

```
Dim waveformat As WAVEFORMATEX
```

下面是设置 Wave 声音文件结构的代码。

```
With waveformat
    .nFormatTag = 1      'PCM 脉冲编码调制，一般为 PCM，非 PCM
    .nChannels = 1      '单声道 1，双声道 2
    .lSamplesPerSec = 22050 '采样频率
    .lAvgBytesPerSec = 44100 '平均字节率（即每秒数据量）
    .nBlockAlign = 2     '字节对齐
End With
```

#### 7. 建立缓冲区加载声音文件并播放

```
Set DSBuffer = objds.CreateSoundBufferFromFile(App.Path & "backwav.wav", bufferdss, waveformat)
DSBuffer.Play DSBPLAY_LOOPING
```

上面代码首先建立缓冲区，然后将声音文件、缓冲区结构 `bufferdss` 和 `Wave` 声音文件结构加载到缓冲区中，最后使用 `Play` 方法播放声音。播放声音时，将参数设置为 `DSBPLAY_LOOPING` 表示循环播放，将参数设置为 `DSBPLAY_DEFAULT` 则表示播放一遍。

**例 19.9** 下面利用 `DirectSound` 编程实现实时混音，即将背景音乐与人声混合播放。（实例位置：光盘\TM\sl\19\6）

前面介绍了使用 `DirectSound` 播放一个 `WAV` 文件的方法，而将多个文件进行混合播放只须将使用的对象声明为数组，代码如下。

```
Dim objds As DirectSound
Dim DSBuffer(1) As DirectSoundBuffer
Dim bufferdss(1) As DSBUFFERDESC
Dim waveformat(1) As WAVEFORMATEX
Private Sub Form_Load()
    Set objds = objdx.DirectSoundCreate("")
    objds.SetCooperativeLevel Me.hWnd, DSSCL_NORMAL
    bufferdss(0).lFlags = DSBCAPS_STATIC
    With waveformat(0)
        .nFormatTag = 1
        .nChannels = 1
        .lSamplesPerSec = 22050
        .lAvgBytesPerSec = 44100
        .nBlockAlign = 2
    End With
    Set DSBuffer(0) = objds.CreateSoundBufferFromFile(App.Path & "\backwav.wav", bufferdss(0), waveformat(0))
    DSBuffer(0).Play DSBPLAY_LOOPING
    bufferdss(1).lFlags = DSBCAPS_STATIC
    With waveformat(1)
        .nFormatTag = 1
        .nChannels = 1
        .lSamplesPerSec = 22050
        .lAvgBytesPerSec = 44100
        .nBlockAlign = 2
    End With
    '将声音文件、缓冲区结构和文件结构加载到第二个缓冲区中
    Set DSBuffer(1) = objds.CreateSoundBufferFromFile(App.Path & "\录像 1.wav", bufferdss(1), waveformat(1))
    DSBuffer(1).Play DSBPLAY_LOOPING
End Sub
```

'定义一个 DirectSound 对象  
'以数组方式声明两个缓冲区对象  
'以数组方式声明两个缓冲区结构对象  
'以数组方式声明两个 Wave 声音文件结构  
  
'创建 DirectSound 对象  
'设置音效协调层级  
'设置第一个缓冲区功能  
'设置第一个文件结构  
'设置为 PCM  
'单声道  
'采样频率  
'平均字节率  
'字节对齐  
  
'将声音文件、缓冲区结构和文件结构加载到第一个缓冲区中  
'播放第一个声音文件  
'设置第二个缓冲区功能  
'设置第二个文件结构  
'设置为 PCM  
'单声道  
'采样频率  
'平均字节率  
'字节对齐  
  
'播放第二个声音文件

 说明：由于篇幅有限，`DirectX` 的其他对象就不介绍了，请读者自行查阅相关资料。

## 19.6 多媒体综合应用

 教学录像：光盘\TM\lx\19\多媒体综合应用

以上介绍了用于操纵多媒体文件的各种控件，下面再通过 `CD` 播放器、`VCD` 播放器和多媒体演示程序，使读者更进一步掌握多媒体控件在程序中的应用。

### 19.6.1 CD 播放器

下面使用 MMControl 控件制作 CD 播放器。运行程序，效果如图 19.15 所示。（实例位置：光盘 \TM\sl\19\7）



图 19.15 CD 播放器

程序实现步骤如下。

- (1) 新建一个工程，在该工程中将 MMControl 控件添加到工具箱中。
- (2) 在窗体上添加一个 MMControl 控件，设置 Visible 属性为 False，用来播放 CD。
- (3) 将 Flash 控件添加到工具箱中，选择“工程”/“部件”命令，打开“部件”对话框，选择 Shock waveFlash，单击“确定”按钮。
- (4) 在窗体上添加一个 ShockwaveFlash 控件，用来显示 CD 播放器界面。
- (5) 设置窗体的 Height 属性值为 9000，Width 属性值为 12000，BorderStyle 属性值为“0-None”，StartPosition 属性值为“2-屏幕中心”。
- (6) 以上设置完成后，切换到“代码窗口”，编写如下代码。

```
Private Sub Form_Load()
    '加载 Flash 主界面
    ShockwaveFlash1.Movie = App.Path & "\main.swf"
    '初始化 MMControl 设备
    MMControl1.Visible = False
    MMControl1.Notify = True
    MMControl1.Shareable = False
    MMControl1.TimeFormat = 0
    MMControl1.DeviceType = "cdaudio"
    MMControl1.command = "Open"
    MMControl1.UpdateInterval = 1000
    Me.Height = 2265: Me.Width = 7050
    txtTracks.Text = MMControl1.Tracks
    txtTrack.Text = "0"
    '在文本框中显示总曲目数
    '设置当前曲目为 0
End Sub

Private Sub MMControl1_StatusUpdate()
    txtTrack.Text = Str$(MMControl1.Track)
    '在文本框中显示当前曲目
End Sub

Private Sub ShockwaveFlash1_FSCommand(ByVal command As String, ByVal args As String)
    Select Case command
        Case 1
            '播放
            MMControl1.command = "Play"
        Case 2
            '暂停播放
```



```

MMControl1.command = "Pause"
Case 3                                '播放上一首
    MMControl1.command = "Prev"
Case 4                                '播放下一首
    MMControl1.command = "next"
Case 5                                '停止播放，回到第一个曲目
    MMControl1.command = "Stop"
    MMControl1.To = MMControl1.Start
    MMControl1.command = "seek"
    MMControl1.Track = 1
Case 6                                '弹出光盘
    MMControl1.command = "stop"
    MMControl1.command = "eject"
Case 7                                '关闭 MMControl 设备并退出
    MMControl1.command = "Stop"
    MMControl1.command = "Close"
End
End Select
End Sub

```

### 19.6.2 VCD 播放器

下面使用 MediaPlayer 控件制作 VCD 播放器。运行程序，效果如图 19.16 所示。（实例位置：光盘\TM\sl\19\8）

程序实现步骤如下。

（1）新建一个标准工程，在该工程中将 MediaPlayer 和 CommonDialog 控件添加到工具箱中。

（2）在窗体上添加一个 MediaPlayer 控件、一个 CommonDialog 控件和 5 个 CommandButton 控件。

（3）主要程序代码如下：



图 19.16 VCD 播放器

```

Private Sub Form_Load()
    Caption = "MediaPlayer 控件示例"                                '设置窗体标题栏内容
    '设置按钮显示的内容
    Command1.Caption = "打开"
    Command2.Caption = "播放"
    Command3.Caption = "暂停"
    Command4.Caption = "停止"
    Command5.Caption = "退出"
    MediaPlayer1.PlayCount = 3                                        '设置播放数量
    MediaPlayer1.AutoStart = True                                    '设置自动播放
    Command2.Enabled = False: Command3.Enabled = False: Command4.Enabled = False '设置按钮不可用
End Sub
Private Sub Command1_Click()
    CommonDialog1.Action = 1
    MediaPlayer1.FileName = CommonDialog1.FileName                    '指定播放文件

```

```

If CommonDialog1.FileName <> "" Then
    Caption = "正在播放: " & CommonDialog1.FileName
    Command3.Enabled = True: Command4.Enabled = True
End If
End Sub
Private Sub Command2_Click()
    MediaPlayer1.Play
    Command2.Enabled = False: Command3.Enabled = True: Command4.Enabled = True
End Sub
Private Sub Command3_Click()
    MediaPlayer1.Pause
    Command3.Enabled = False: Command2.Enabled = True
End Sub
Private Sub Command4_Click()
    MediaPlayer1.Stop
    Command4.Enabled = False: Command2.Enabled = True
End Sub

```

'如果选择了多媒体文件  
'将多媒体文件名显示在窗体标题栏上  
'设置按钮不可用  
'播放多媒体文件  
'设置按钮不可用  
'暂停播放多媒体文件  
'设置按钮不可用  
'停止播放多媒体文件  
'设置按钮不可用

### 19.6.3 多媒体演示程序

下面使用“MMControl+ShockwaveFlash 控件”制作多媒体演示程序，其中 MMControl 控件用于控制背景音乐，ShockwaveFlash 控件用于制作动态主界面，并与 VB 程序进行交互。运行程序，结果如图 19.17 所示。（实例位置：光盘\TM\sl\19\9）



图 19.17 多媒体演示程序

程序实现步骤如下。

- (1) 新建一个工程，将 MMControl 和 ShockwaveFlash 控件添加到工具箱中。
- (2) 在窗体上添加一个 MMControl 控件和一个 ShockwaveFlash 控件。
- (3) 程序主要代码如下：

```

Private Declare Function WinExec Lib "kernel32" (ByVal lpCmdLine As String, ByVal nCmdShow As Long) As Long
Private Sub Form_Load()

```

'声明 API 函数 WinExec

```

ShockwaveFlash1.Movie = App.Path & "\swf\main.swf"      '播放 Flash
ShockwaveFlash1.Menu = False                            '不显示菜单
ShockwaveFlash1.Playing = True                          '设置正在播放
With MMControl1
    .FileName = App.Path & "\back\mr.wav"                '指定音频文件
    .command = "Open"                                    '打开音频文件
    .command = "play"                                    '播放音频文件
End With
End Sub
'此处省略了循环播放的代码，这部分代码可参见光盘中的源程序
Private Sub ShockwaveFlash1_FSCommand(ByVal command As String, ByVal args As String)
    Select Case command                                  '通过不同的命令按钮，执行不同的功能
        Case "1"
            MsgBox "多媒体教学录像"
        Case "2"
            '编程词典
            MsgBox "编程词典"
        Case "3"
            '浏览光盘
            WinExec "explorer.exe " & Left(App.Path, 3), 10
        Case "4"
            '帮助
            MsgBox "帮助"
        Case "5"
            End
    End Select
End Sub

```

## 19.7 小结

本章介绍了 MMControl 控件、Animation 控件、MediaPlayer 控件、ShockwaveFlash 控件的主要属性、方法和事件，这些知识是编写多媒体程序的基础，读者应重点掌握。另外，本章还介绍了一些 DirectX 的知识，如果读者对这一技术感兴趣，想进一步了解它，可以查阅相关网站或书籍。

## 19.8 练习与实践

1. 使用 MMControl 控件播放 MIDI、WAV 和 AVI 文件。（答案位置：光盘\TM\19\10）
2. 使用 MMControl 控件播放光驱中的 VCD。（答案位置：光盘\TM\19\11）
3. 使用 MediaPlayer 控件连续播放 MP3。（答案位置：光盘\TM\19\12）
4. 制作一个 Flash 播放器。（答案位置：光盘\TM\19\13）
5. 使用 DirectSound 编程实现 3 个 WAV 文件混音效果。（答案位置：光盘\TM\19\14）

# 第20章

## SQL 应用

(教学录像: 1 小时 29 分钟)

SQL 是一种结构化查询语言, 它不仅具有强大的查询功能, 而且还具有创建数据库对象、数据结构、添加数据和修改数据等功能。由于 SQL 与数据库打交道, 本章先介绍数据库基础知识和常用数据库的安装、创建数据库及表的过程, 接着从最常用的 SQL 语句入手, 使读者快速应用 SQL, 从而为编写数据库应用程序奠定坚实的基础。

通过阅读本章, 您可以:

- » 了解什么是数据库
- » 掌握常用数据库软件的安装
- » 学会创建数据库及表的方法
- » 掌握检索数据、排序数据和过滤数据的 SQL 语句
- » 掌握数据汇总、分组统计和子查询
- » 掌握多种向表中插入数据的 SQL 语句
- » 掌握修改和删除数据的 SQL 语句



## 20.1 数据库的基本知识

 教学录像：光盘\TM\lx\20\数据库的基本知识

理解和使用数据库是掌握 SQL 的一个重要部分，下面就先简单地介绍一下数据库的概念及常用数据库软件的安装和使用。

### 20.1.1 什么是数据库

“数据库”从字面上看，就是存放数据的仓库。从本质上讲，数据库是指数据和数据对象的集合。这种集合可以长期存储，具有确定的数据存储结构，同时能以安全和可靠的方法进行数据的检索和存储。数据对象是指表（tabel）、视图（view）、存储过程（stored procedure）和触发器（trigger）等。

### 20.1.2 数据库软件的安装和使用

流行的数据库软件有多种，经常和 VB 配合使用的是 Access 数据库管理系统和 SQL Server 2000 数据库管理系统。下面就详细介绍这两种数据库管理系统的安装和使用。

#### 1. Access 的安装和使用


Access 数据库管理系统是 Microsoft Office 中自带的一个组件，安装 Office 的同时便可以将它安装到系统中。它是当前比较流行的关系型数据库管理系统之一，能够满足小型企业客户/服务器解决方案的要求。因此，在使用 VB 开发中小型管理系统时，一般都使用 Access 数据库。

**例 20.1** 下面以 Access 2003 数据库管理系统为例介绍创建 Access 数据库和表的过程。（实例位置：光盘\TM\sl\20\1）

（1）启动 Access 2003，选择“文件”/“新建”命令，在“新建文件”任务窗格中单击“空数据库”，打开“文件新建数据库”对话框，在“文件名”文本框中输入“db\_books”，单击“确定”按钮数据库创建成功。

（2）在对象列中单击“表”选项，选择“使用设计器创建表”，然后单击工具栏中的“新建”按钮，在打开的“新建表”对话框中选择“设计视图”，单击“确定”按钮。

（3）在“字段名称”列中的第一个单元格中输入字段名称“条形码”，单击“数据类型”列中的第一个单元格，再单击出现的下拉按钮，在下拉列表框中选择数据类型“文本”，“说明”列用来输入一些描述性文字，为了将来数据库的可读性和可维护性，如果是英文字段一定在此说明。在“字段属性”中设置长度为“20”。

 **说明：**设置字段的数据类型即定义用户可以在字段中输入的值类型。例如，如果要使字段存储可以用于计算的数值，要将其数据类型设置为“数字”或“货币”。

（4）按照步骤（3）依次添加“书名”、“作者”、“出版社”、“数量”、“单价”和“金额”等字段，如图 20.1 所示。

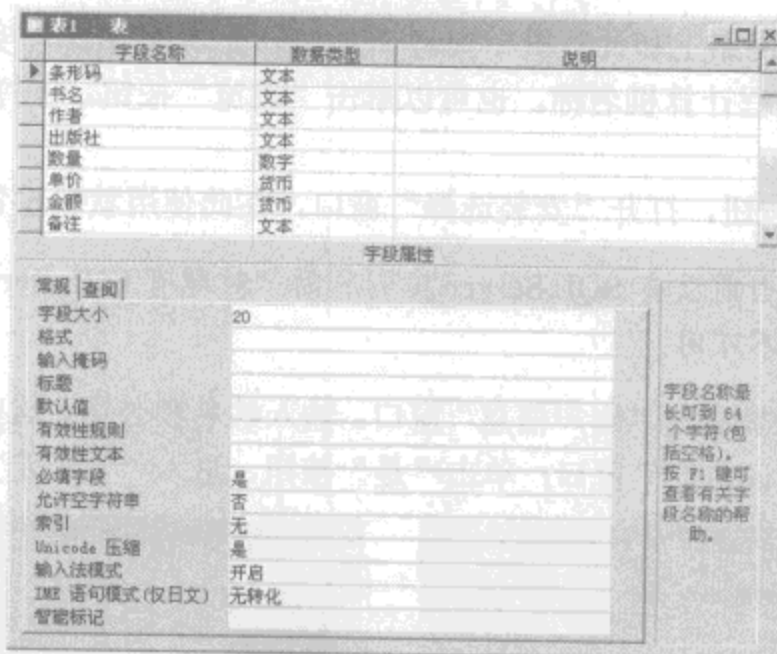


图 20.1 表设计器

(5) 单击工具栏中的“保存”按钮，在“另存为”对话框中输入表名“图书信息表”；单击“确定”按钮，数据表即创建成功。

## 2. SQL Server 2000 的安装和使用

SQL Server 2000 是微软公司开发的，一套功能强大的数据库管理系统，它包括多种版本。

**例 20.2** 下面以安装个人版为例介绍安装 SQL Server 2000 的过程。(实例位置：光盘\TM\sl\20\2)

(1) 将安装盘放入光驱，光盘自动运行，并弹出如图 20.2 所示的窗口。

**说明：**如果光盘没有自动运行，可以使用“我的电脑”或者 Windows 资源管理器，浏览光盘内容，双击光盘根目录下的 Autorun.exe 文件启动安装程序。

(2) 在启动界面中选择“安装 SQL Server 2000 组件”选项，切换至“安装组件”窗口，如图 20.3 所示。



图 20.2 启动界面




图 20.3 安装组件

(3) 在“安装组件”窗口中选择“安装数据库服务器”选项，打开 SQL Server 安装向导，单击“下一步”按钮。

(4) 在“计算机名”窗口中选择安装 SQL Server 2000 数据库的计算机，这里选择“本地计算机”

单选按钮。如果用户想要在网络上和计算机中安装 SQL Server 2000 数据库实例，可以选择“远程计算机”单选按钮，然后输入远程计算机名称。也可以单击“浏览”按钮，在弹出的“选择计算机”窗口中选择计算机。

(5) 单击“下一步”按钮，打开“安装选择”窗口，在此使用默认设置。

 说明：如果用户系统中当前没有 SQL Server 实例，则“对现有 SQL Server 实例进行升级、删除或添加组件”选项不可用。

(6) 单击“下一步”按钮打开“用户信息”窗口，输入姓名和公司。这些信息可以省略。单击“下一步”按钮打开“软件许可证协议”窗口；单击“是”按钮，进入“安装定义”窗口，如图 20.4 所示。

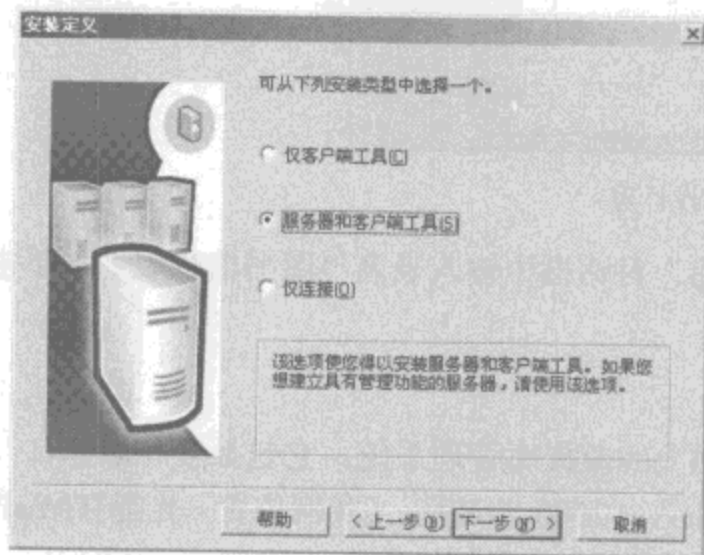


图 20.4 “安装定义”窗口

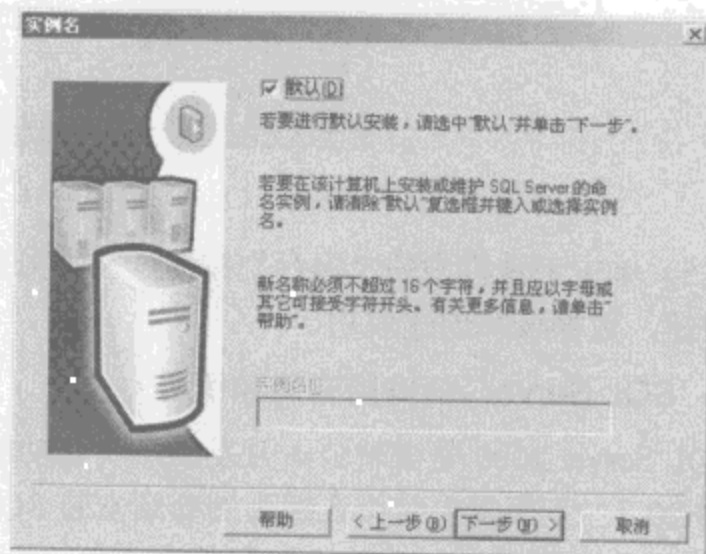


图 20.5 “实例名”窗口

(7) 在“安装定义”窗口中有 3 个安装类型：“仅客户端工具”选项，表示只安装客户端数据库管理工具；“服务器和客户端工具”选项，表示安装服务器和客户端工具以创建具有管理能力的关系数据库服务器；选择“仅连接”选项，表示只安装数据库客户端连接组件。这里选择“服务器和客户端工具”选项，单击“下一步”按钮，打开“实例名”窗口，如图 20.5 所示。

(8) 在“实例名”窗口中设置安装 SQL Server 的实例名称，可以使用默认的实例名，即选中“默认”复选框。如果要自己设置实例名，不选中“默认”复选框，在“实例名”文本框中直接输入实例名，这里选择默认的实例名。

(9) 单击“下一步”按钮，打开“安装类型”窗口，在此选择“典型”单选按钮，即进行典型安装。如果用户要更改默认的安装路径，可以单击“浏览”按钮重新指定安装路径。

(10) 单击“下一步”按钮，打开“服务账户”窗口，如图 20.6 所示。

(11) 在“服务账户”窗口中设置用户账户，在“服务设置”选项组中选择“使用本地系统账户”选项，单击“下一步”按钮，打开“身份验证模式”窗口，如图 20.7 所示。

(12) 在“身份验证模式”对话框中选择需要的身份验证模式。

- ☒ Windows 身份验证模式：该模式表示用户通过 Windows 用户账户连接时，SQL Server 2000 使用 Windows 操作系统中的信息验证账户名和密码。
- ☒ 混合模式：允许用户使用 Windows 身份验证模式或 SQL Server 2000 身份验证进行连接，或混合模式中使用信任连接（由 Windows 验证的连接）。

这里选择“混合模式”单选按钮，同时选中“空密码”复选框，单击“下一步”按钮，打开“开始复制文件”窗口。

注意：为了数据库安全，建议在此设置密码，并且要记住这个密码，因为用 VB 连接数据库时会用到。

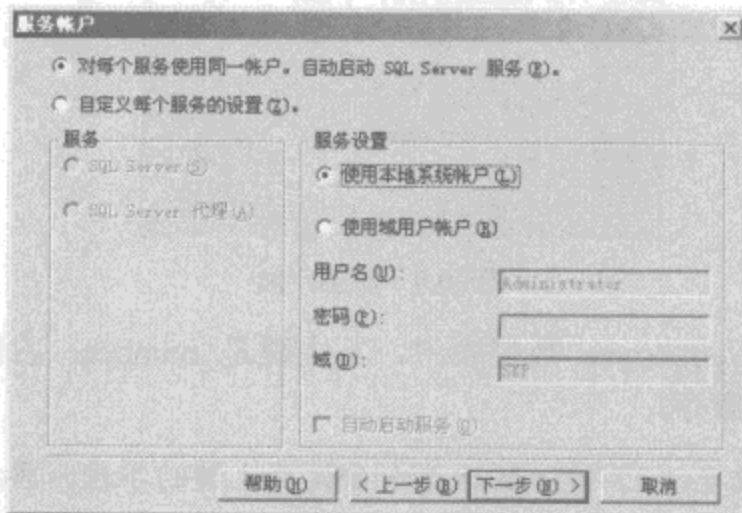


图 20.6 “服务账户”窗口

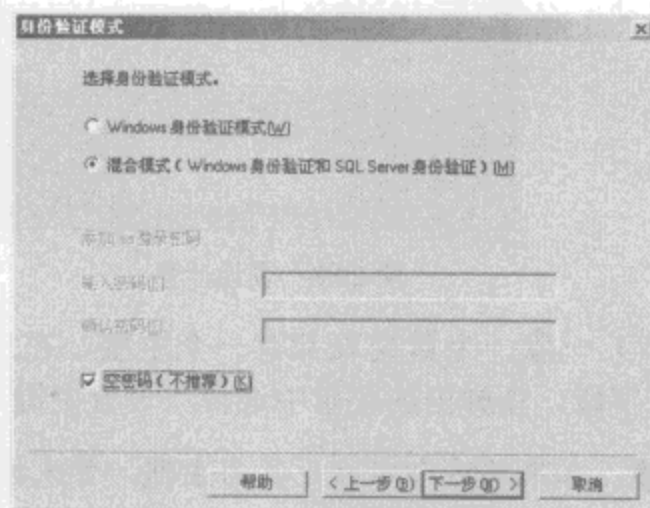


图 20.7 “身份验证模式”对话框

(13) 在“开始复制文件”窗口中单击“下一步”按钮开始复制文件，即对 SQL Server 2000 进行安装，安装成功后，将弹出“安装完毕”窗口。

例 20.3 下面介绍创建数据库和表的过程。要创建的数据库为 db\_manpowerinfo，表为 tb\_employee。

(1) 选择“开始”/“程序”/Microsoft SQL Server/“企业管理器”，打开企业管理器。

(2) 依次展开服务器组和服务器，右击“数据库”，在弹出的菜单中选择“新建数据库”，打开“数据库属性”对话框，在“名称”文本框中输入 db\_manpowerinfo，单击“确定”按钮数据库便创建成功了。

(3) 依次展开“数据库”和 db\_manpowerinfo，右击“表”，在弹出的快捷菜单中选择“新建表”，打开表设计器，如图 20.8 所示。

(4) 在“列名”列中的第一个单元格中输入字段名称“number”，单击“数据类型”列中的第一个单元格，再单击出现的下拉按钮，在下拉列表框中选择数据类型 varchar，“长度”为 50。因为 number 不允许为空，所以将“允许空”选项去掉。

(5) 设置字段属性，以下是一些常用到的字段属性。

- ☒ 说明：用来输入一些描述性文字，为了将来数据库的可读性和可维护性，如果是英文字段一定在此说明。如字段 number，其说明为“对每个员工的编码”。
- ☒ 默认值：每当在表中为字段插入带空值的行时，将显示该字段的默认值，下拉列表框中包含在数据库中定义的所有全局默认值，如果需要可以在此选择。
- ☒ 精度：显示字段值的最大数字位数。
- ☒ 小数位数：显示字段值小数点右边能出现的最大数字位数。
- ☒ 标识：显示 SQL Server 是否将该字段用作标识字段。
- ☒ 标识种子：显示标识字段的种子值。该选项只适用于其“标识”选项设置为“是”的字段。
- ☒ 标识递增量：显示标识字段递增量值。该选项只适用于其“标识”选项设置为“是”的字段。

(6) 按照步骤 (4) 和步骤 (5) 依次添加 name、sex、card、birth、age、nation 等字段，如图 20.9



所示。

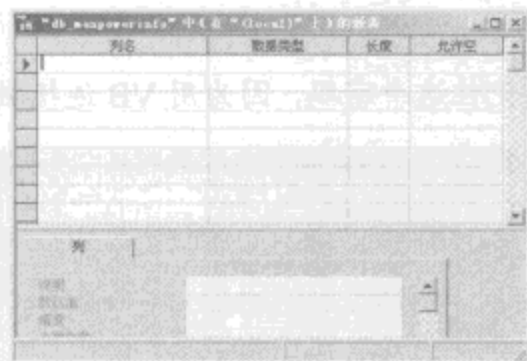


图 20.8 表设计器

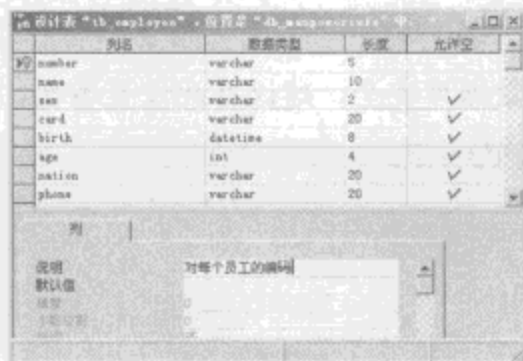


图 20.9 添加字段

(7) 一般一个数据表都有一个主键，在 tb\_employee 数据表中，其主键是 number。右键单击 number 字段，在弹出的菜单中选择“设置主键”命令。

**注意：**如果将多个字段设置为主键，首先按住〈Ctrl〉键，然后选中要设置主键的字段，再按照步骤(7)设置主键。

(8) 保存表，单击工具栏上的“保存”按钮，打开“选择名称”对话框，输入表名 tb\_employee，单击“确定”按钮。

**注意：**表名必须遵守 SQL Server 2000 的命名规则，最好能够准确地表达表的内容。另外，表名不要用 sys 开头，以免与系统表混淆。

至此，tb\_employee 表创建完成。按照该方法再创建一个名为 tb\_pay 的表(相关字段可参见图 20.10)，这两个表就是后面查询中应用的表。在这两个表中添加一些数据，最终效果如图 20.10 所示。

number	name	sex	card	birth	age	nation	phone
00001	张*译	男	220111XXXXXXX	1984-04-01	21	汉族	230110
00002	高*强	男	220104XX				
00003	张*峰	男	220111XX				
00004	李*丽	女	220111XX				
00005	李*雨	女	220111XX				
00006	刘*彬	女	220111XX				
00008	赵*强	男	220104XX				
00009	周*强	男	220104XX				
00010	孙*强	男	220104XX				
00028	郑*志明	男	220123XX				

ID	PayMonth	EmployeeNumber	EmployeeName	BasePay	AdiPay	AgePay	WorkPay	MustPay	DeliPay	RealityPay
15	2005-01	00001	张*译	2300	120	0	100	2620	40	2580
16	2005-01	00002	高*强	1700	0	0	0	1650	10	1640
17	2005-01	00003	张*峰	1800	0	0	0	1920	10	1910
18	2005-01	00004	李*丽	1700	50	0	100	1750	40	1710
19	2005-01	00005	李*雨	1300	0	0	0	1400	10	1390
20	2005-01	00006	刘*彬	1700	0	0	0	1620	10	1610
21	2005-01	00008	赵*强	2600	0	0	100	2800	60	2740
22	2005-01	00009	周*强	1000	0	0	0	1110	10	1100
23	2005-01	00010	孙*强	1700	0	0	0	1620	10	1610
24	2005-01	00028	郑*志明	2000	200	0	0	2300	40	2260

图 20.10 表 tb\_employee 和表 tb\_pay

## 20.2 SQL 基础

**教学录像：**光盘\TM\lx\20\SQL 基础

本节介绍什么是 SQL 和执行 SQL 语句的常用工具。

### 20.2.1 什么是 SQL

SQL 是结构化查询语言 (Structured Query Language) 的缩写。它是一种组织、管理和检索存储在

数据库中数据的工具，是一种可以与数据库交互的结构化查询语言。

SQL 是一种子语言，而不是一种完全的编程语言，它只能告诉数据库管理系统要做什么，至于怎样做则由数据库管理系统完成。大多数数据库管理系统都对标准的 SQL 进行了扩展，使其允许对 SQL 进行编程。例如 Oracle 使用 PL/SQL，SQL Server 使用 Transact-SQL，而 Access 等小型数据库则使用 JET-SQL。VB 经常与 JET-SQL 和 Transact-SQL 打交道，二者语法基本相同，只是个别通配符不同，详细内容可参见后面的章节。

 说明：JET-SQL 和 Transact-SQL 还存在一些不同之处，读者可在实际编程过程中进一步体会。

## 20.2.2 执行 SQL 语句的工具

由于后面大部分章节介绍的是 SQL 语句的应用，因此这里先了解一下执行 SQL 语句的工具，以验证 SQL 语句的准确性和执行效果。下面分别以 Access 2003 和 SQL Server 2000 中执行 SQL 语句为例进行介绍。

### 1. 在 Access 2003 中执行 SQL 语句

启动 Access 2003，打开需要建立查询的数据库，如 db\_books。在对象列中单击“查询”选项，如图 20.11 所示，双击运行“在设计视图中创建查询”，启动查询设计视图。在“显示表”对话框中选择表，单击“添加”按钮，该表便被添加到了查询设计视图中，如图 20.12 所示。

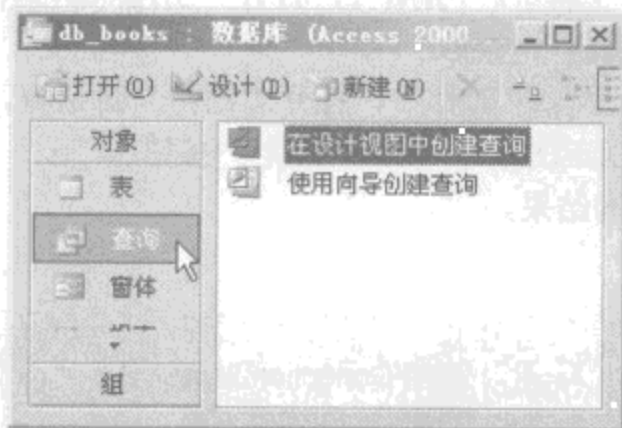


图 20.11 选择“查询”

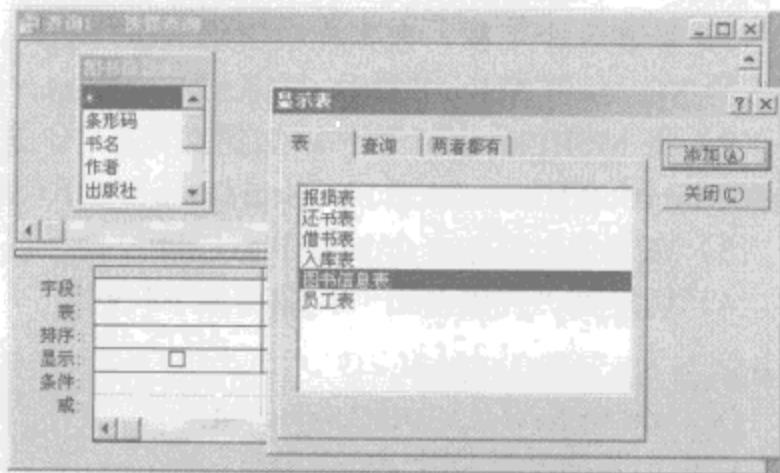


图 20.12 查询设计视图

单击“关闭”按钮，关闭“显示表”对话框，此时选择“视图”/“SQL 视图”，将打开如图 20.13 所示的窗口，在此可以查看或编辑 SQL 语句。

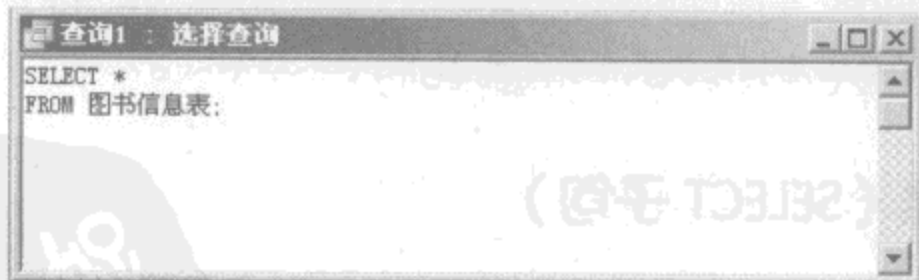


图 20.13 SQL 视图

查看或编辑完 SQL 语句后，单击工具栏中的“运行”按钮，即可显示查询结果集。

## 2. 在 SQL Server 2000 中执行 SQL 语句

启动企业管理器，并从数据库中选择一个数据表，如 tb\_employee。单击鼠标右键，选择“打开表”/“查询”命令，或者在工具栏中单击“显示/隐藏 SQL 窗格”按钮，如图 20.14 所示，在“SQL 窗格”中可以编写 SQL 语句，在“结果窗格”中将显示结果集。

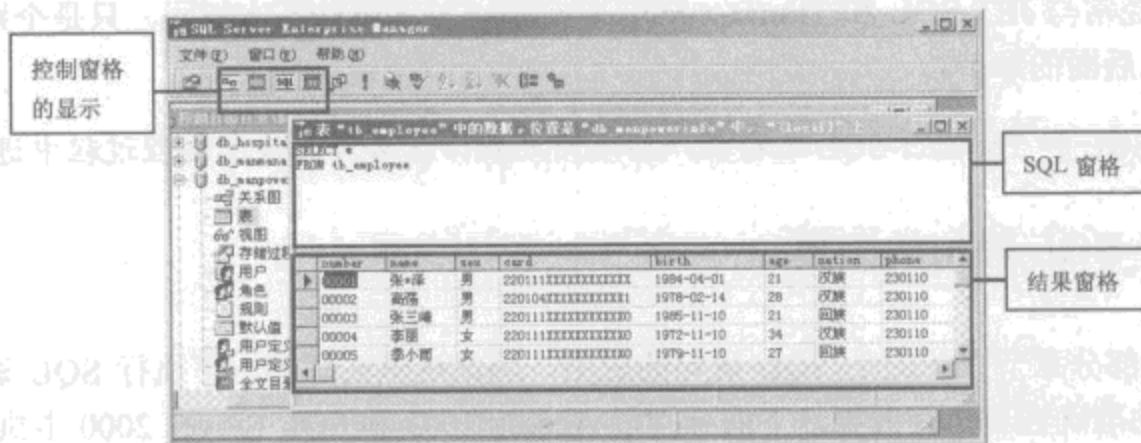


图 20.14 查询窗口

## 3. 在 VB 中执行 SQL 语句

在 VB 中执行 SQL 语句有多种方法。例如，使用 ADO 对象的 Connection 对象执行 SQL 语句，然后使用文本框（单个字段显示）控件或 MSHFlexGrid 等表格控件显示查询结果。

**例 20.4** 下面通过一个简单的例子，介绍在 VB 中执行 SQL 语句。（实例位置：光盘\TM\sl\20\3）

- (1) 新建一个工程，在工程中引用 ADO 对象。选择“工程”/“引用”，打开“引用”对话框，在此选择 Microsoft ActiveX Data Object 2.5 Library，单击“确定”按钮。
- (2) 将 MSHFlexGrid 控件添加到工具箱中。选择“工程”/“部件”，打开“部件”对话框，在此选择 Microsoft Hierarchical FlexGrid Control 6.0 (SP4) (OLEDB)，单击“确定”按钮。
- (3) 在窗体上添加一个 MSHFlexGrid 控件，以显示查询结果。
- (4) 切换到“代码窗口”，编写如下代码。

```
Private Sub Form_Load()  
    Dim cnn As New ADODB.Connection  
    Dim rs As New ADODB.Recordset  
    '连接 SQL Server 2000 数据库 db_manpowerinfo  
    cnn.Open "Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=db_manpowerinfo"  
    Set rs = cnn.Execute("select * from tb_employee")  
    Set MSHFlexGrid1.DataSource = rs  
End Sub
```

'声明 Connection 对象  
'声明 Recordset 对象  
'连接数据表 tb\_employee，并返回查询结果集  
'用 MSHFlexGrid 控件显示查询结果

说明：有关更详细的数据库相关技术可参见第 21 章。

## 20.3 检索数据（SELECT 子句）

教学录像：光盘\TM\lx\20\检索数据（SELECT 子句）

这一节主要介绍如何使用 SELECT 语句从表中检索一个或多个数据列。

## 20.3.1 SELECT 子句

SELECT 子句指定要查询的列。这些列通常被一个选择列表指定，选择列表是中间用逗号分开的选择项列表。选择项可以是字段名、常量或 SQL 表达式。下面是 SELECT 子句的语法：

```
SELECT [ ALL | DISTINCT ]
      [ TOP n [ PERCENT ] [ WITH TIES ] ]
      < select_list >
< select_list > ::=
{
  *
  | { table_name | view_name | table_alias }. *
  | { column_name | expression | IDENTITYCOL | ROWGUIDCOL }
    [[ AS ] column_alias ]
  | column_alias = expression
}
[ ,...n ]
```

参数说明：

ALL：指定在结果集中可以显示重复行。ALL 是默认设置。

DISTINCT：去掉重复记录。

TOP n [PERCENT]：指定只从查询结果集中输出前 n 行。n 是介于 0 和 4294967295 之间的整数。如果还指定了 PERCENT，则只从结果集中输出前百分之 n 行。当指定时带 PERCENT 时，n 必须是介于 0~100 之间的整数。如果查询包含 ORDER BY 子句，将输出由 ORDER BY 子句排序的前 n 行（或前百分之 n 行）。如果查询没有 ORDER BY 子句，行的顺序任意。

WITH TIES：指定从基本结果集中返回附加的行，这些行包含与出现在 TOP n (PERCENT) 行最后的 ORDER BY 列中的值相同的值。如果指定了 ORDER BY 子句，则只能指定 TOP ...WITH TIES。

< select\_list >：为结果集选择的列。选择列表是以逗号分隔的一系列表达式。

\*：指定在 FROM 子句内返回表和视图内的所有列。列按 FROM 子句在所指定的表或视图中的顺序返回，“table\_name | view\_name | table\_alias.\*”将\*的作用域限制为指定的表或视图。

column\_name：是要返回的列名。限定 column\_name 以避免二义性引用，当 FROM 子句中的两个表内有包含重复名的列时会出现这种情况。

expression：是列名、常量、函数以及由运算符连接的列名、常量和函数的任意组合，或者是子查询。

IDENTITYCOL：返回标识列。有关更多信息，请参见 IDENTITY（属性）、ALTER TABLE 和 CREATE TABLE。如果 FROM 子句中的多个表内有包含 IDENTITY 属性的列，则必须用特定的表名（如 T1.IDENTITYCOL）限定 IDENTITYCOL。

ROWGUIDCOL：返回行全局唯一标识列。如果在 FROM 子句中有多个表具有 ROWGUIDCOL 属性，则必须用特定的表名（如 T1.ROWGUIDCOL）限定 ROWGUIDCOL。

column\_alias：是查询结果集内替换列名的可选名。别名还可用于为表达式结果指定名称。




### 20.3.2 检索单个列

例 20.5 检索单个列，SQL 语句如下。

```
SELECT name FROM tb_employee
```


上述语句利用 SELECT 语句从员工表 tb\_employee 中检索一个名称列 name，所需的列名在 SELECT 关键字之后给出，FROM 关键字指出从其中检索数据的表名，此语句输出结果如图 20.15 所示。

 技巧：在检索某个列的同时，还可以将该列重命名，这需要使用 AS 关键字。例如，将 name 重命名为“姓名”，SQL 语句如下。

```
SELECT name AS 姓名 FROM tb_employee
```

### 20.3.3 检索多个列

要从一个表中检索多个列，使用相同的 SELECT 语句。唯一不同的是必须在 SELECT 关键字后给出多个列名（也就是字段名），列名之间必须以逗号分隔。

 注意：在选择多个列时，一定要在列名之间加上逗号，但最后一个列名后不加。如果在最后一个列名后加了逗号，将出现错误。

例 20.6 下面使用 SELECT 语句检索员工表 tb\_employee 中的编号（number）、姓名（name）和性别（sex），语句如下。

```
SELECT number, name, sex FROM tb_employee
```

上述语句的输出结果如图 20.16 所示。

name
张*泽
高强
张三峰
李丽
季小雨

图 20.15 检索单个列

number	name	sex
00004	李丽	女
00002	高强	男
00005	季小雨	女
00001	张*泽	男
00003	张三峰	男


图 20.16 检索多个列

### 20.3.4 检索所有列

前面介绍了检索一个或多个列，下面介绍使用 SELECT 语句检索所有的列。用 SELECT 语句检索所有的列可以不必给出所有字段并有逗号隔开，而是使用一个星号（\*）通配符即可。例如下面的语句。

```
SELECT * FROM tb_employee
```

上述语句，给定一个通配符（\*），则返回表中所有列。列的顺序一般是表中各列出现的物理顺序。

 说明：一般情况下，如果确定需要显示的列，就使用前面介绍的方法，如果不确定就使用\*通配符。这样不仅可以检索出列名未知的列，而且会很省事，不用编写较多的代码，但是系统检索不必要的列也会降低检索效率和应用程序的性能。

## 20.4 排序检索数据 (ORDER BY 子句)

 教学录像：光盘\TM\lx\20\排序检索数据 (ORDER BY 子句)

将检索出来的数据，按一定顺序排列，需要使用 ORDER BY 子句。下面介绍使用 ORDER BY 子句简单排序、按多个列排序、按列位置排序、指定排序方向和对新生成的列进行排序。

### 20.4.1 排序数据

前面的讲解中使用 SELECT 语句实现了检索单个列、多个列和所有列，但实际检索出来的数据，并没有特定的顺序，如例 20.6。

要想让检索出来的数据按一定顺序排列，就需要使用 ORDER BY 子句。ORDER BY 子句取一个或多个列的名字，对输出进行排序。

例 20.7 将例 20.6 中检索的数据，按编号 (number) 字段升序排序。

```
SELECT number,name,sex FROM tb_employee ORDER BY number
```

这条语句与例 20.6 中前面的语句相同，只是在后面加了 ORDER BY number，实现对 number 字段以数字字符顺序排序。

### 20.4.2 按多个列排序

在实际编程中，经常需要多个字段进行排序。例如，显示员工信息并按年龄排序。如果有几个相同年龄的员工，再按编号排序，这样做是非常有用的。

按多个列排序，应指定排序的列名，并在列名之间用逗号分开。

例 20.8 下面的代码检索员工编号、姓名、性别和年龄，并按编号 (number) 和年龄 (sex) 排序，首先按年龄 (sex) 排序，然后再按编号 (number) 排序。

```
SELECT number, name, sex, age FROM tb_employee ORDER BY age, number
```

上述语句，输出结果如图 20.17 所示。

### 20.4.3 按列位置排序

除了用列名指出要排序的列外，ORDER BY 子句还支持用列的位置进行排序。列的位置是指列所在的序号，也就是第几列，如图 20.18 所示。

number	name	sex	age
00001	张*泽	男	21
00003	张三峰	男	21
00009	周全强	男	21
00006	刘*彬	女	26
00005	季小雨	女	27
00002	高强	男	28
00010	孙全强	男	28
00008	赵全强	男	29
00004	李丽	女	34
00028	郑志明	男	34

② 按“编号”排序      ① 按“年龄”排序

图 20.17 按多个列排序


number	① 列	name	② 列	sex	③ 列	age	④ 列
00001		张*泽		男		21	
00003		张三峰		男		21	
00009		周全强		男		21	

图 20.18 列位置

例 20.9 将例 20.8 中指定的列名换成列位置，SQL 语句如下。

```
SELECT number, name, sex, age FROM tb_employee ORDER BY 4, 1
```

上述语句中“4”表示第 4 列也就是年龄（age），“1”表示第 1 列也就是编号（number），该语句与例 20.8 中的语句的输出结果相同。

 说明：使用列位置要比输入列名称方便得多，但它也有缺点，一是不明确给出列的列名，容易错用需要排序的列，二是如果检索字段的顺序发生改变，而排序的列位置忘记改变，会引起排序错误。

#### 20.4.4 指定排序方向

排序方向分为升序和降序。ORDER BY 子句后面加 ASC 关键字为升序排序，该排序方式是默认的。如果 ORDER BY 子句后面什么也不加（如前面举的例子），就是升序排序。ORDER BY 子句后面加 DESC 关键字为降序排序。

例 20.10 检索员工编号、姓名、性别和年龄，并按年龄（age）降序排序。

```
SELECT number, name, sex, age FROM tb_employee ORDER BY age DESC
```

如果需要使用多种排序，例如，按年龄（age）降序排序，然后再按编号（number）排序。

```
SELECT number, name, sex, age FROM tb_employee ORDER BY age DESC, name
```

上述语句，输出结果如图 20.19 所示。

number	name	sex	age
00004	李丽	女	34
00028	郑志明	男	34
00008	赵全强	男	29
00002	高强	男	28
00010	孙全强	男	28
00005	李小雨	女	27
00006	刘彬彬	女	26
00001	张*泽	男	21
00003	张三雄	男	21
00009	周全强	男	21

图 20.19 多种排序

如果需要将多个列降序排序，应在每列都使用 DESC 关键字，例如下面的语句。

```
SELECT number, name, sex, age FROM tb_employee ORDER BY age DESC, name DESC
```

### 20.4.5 对新生成的列进行排序

ORDER BY 子句还可以对新生成的列进行排序，例如将图书数据中，单本版税最高的书排在第一位，SQL 语句如下。

```
SELECT title, price * royalty / 100 as royalty_per_unit FROM titles ORDER BY royalty_per_unit DESC
```

 说明：用于计算每本书单本所赚版税的公式用粗体表示。

## 20.5 过滤数据（WHERE 子句）

 教学录像：光盘\TM\lx\20\过滤数据（WHERE 子句）

对表中数据进行过滤需要使用 WHERE 子句。本节介绍 WHERE 子句的基本使用方法、WHERE 子句中比较运算符的运用，检索指定范围的值，模式条件查询以及组合条件查询。

### 20.5.1 使用 WHERE 子句

数据表中一般都包含大量的数据，如果用户仅需要其中的一部分数据，这时就应使用 WHERE 子句。WHERE 子句在表名（FROM 子句）之后给出。

例 20.11 查询年龄等于 28 的员工，SQL 语句如下，输出结果如图 20.20 所示。

```
SELECT number, name, age FROM tb_employee WHERE age = 28
```

	number	name	age
▶	00002	高强	28
*	00010	孙全强	28

图 20.20 年龄为 28 的员工

### 20.5.2 WHERE 子句比较运算符

SQL 支持所有的比较运算符，这些运算符如表 20.1 所示。



表 20.1 WHERE 子句运算符

运 算 符	说 明	运 算 符	说 明
=	等于	<=	小于等于
>	大于	!>	不大于
<	小于	!<	不小于
>=	大于等于	<>或!=	不等于

例 20.12 下面通过几个例子介绍比较运算符的用法。

查询“年龄”不等于 28 的员工：

```
SELECT number, name, age FROM tb_employee WHERE age <> 28
```

查询“年龄”小于 28 的员工：

```
SELECT number, name, age FROM tb_employee WHERE age < 28
```

查询“年龄”小于 28 并大于 21 的员工：

```
SELECT number, name, age FROM tb_employee WHERE age < 28 AND age > 21
```

查询“年龄”不小于 28 的所有员工：

```
SELECT * FROM tb_employee WHERE age !< 28
```

换一种写法，输出相同的结果集：

```
SELECT * FROM tb_employee WHERE age >= 28
```

注意：查询字符型数据时，要查询的值应使用单引号。例如，查询姓名等于“张\*泽”的，SQL 语句为：

```
SELECT number, name, age FROM tb_employee WHERE name = '张*泽'
```

### 20.5.3 检索指定范围的值

要检索两个给定的值之间的数据，可以使用范围条件进行检索。通常使用 BETWEEN...AND 和 NOT...BETWEEN...AND 来指定范围条件。

使用 BETWEEN...AND 查询条件时，指定的第一个值必须小于第二个值。因为 BETWEEN...AND 实质是查询条件“大于等于第一个值，并且小于等于第二个值”的简写形式，即 BETWEEN...AND 要包括两端的值，等价于比较运算符 ( $\geq \dots \leq$ )。

例 20.13 查询“年龄”在 28~34 之间的员工，SQL 语句如下。

```
SELECT number, name, age FROM tb_employee WHERE age BETWEEN 28 AND 34
```

下面给出查询前和查询后的效果，如图 20.21 和图 20.22 所示。

number	name	age
00001	张*泽	21
00002	高强	28
00003	张三峰	21
00004	李丽	34
00005	季小雨	27
00006	刘*彬	26
00008	赵全强	29
00009	周全强	21
00010	孙全强	28
00028	郑志明	34

图 20.21 查询前



number	name	age
00002	高强	28
00004	李丽	34
00008	赵全强	29
00010	孙全强	28
00028	郑志明	34

图 20.22 查询后

而 NOT...BETWEEN...AND 语句返回某个数据值在两个指定值的范围以外的,但并不包括两个指定的值。

例如,查询“年龄”不在 28~34 之间的员工,SQL 语句如下。

```
SELECT number, name, age FROM tb_employee WHERE age NOT BETWEEN 28 AND 34
```

#### 20.5.4 模式条件查询

模式条件查询是用来返回符合某种匹配格式的所有记录,通常使用 Like 或 NOT Like 关键字来指定模式查询条件。Like 查询条件需要使用通配符在字符串内查找指定的模式,下面先了解一下常用的通配符,如表 20.2 所示。

表 20.2 Like 关键字中的通配符及其含义

通 配 符	说 明
%	由零个或更多字符组成的任意字符串
_	任意单个字符
[ ]	用于指定范围,例如[A~F],表示 A~F 范围内的任何单个字符

##### 1. 百分号 (%) 通配符

百分号 (%) 通配符在 SQL 查询时,经常会用到。%表示任何字符出现任意次数。

例 20.14 查询姓“张”的员工,SQL 语句如下。

```
SELECT number, name FROM tb_employee WHERE name LIKE '张%'
```

也可以在搜索内容的两端加上%,例如查询姓名中包含“强”的员工,SQL 语句如下。

```
SELECT number, name FROM tb_employee WHERE name LIKE '%强%'
```

⚠ 注意: 如果数据库是 Access,需要使用星号 (\*),而不是百分号 (%)。

##### 2. 下划线 (\_) 通配符

下划线 (\_) 通配符的用途与百分号 (%) 通配符一样,但下划线只匹配单个字符而不是多个字符。

例 20.15 查询姓名中第二个字为“强”的员工,SQL 语句如下。

```
SELECT number, name FROM tb_employee WHERE name LIKE '_强'
```

注意：如果数据库是 Access，需要使用问号（?），而不是下划线（\_）。

### 3. 方括号（[]）通配符

在模式查询中可以使用方括号（[]）通配符来查询一定范围内的数据。方括号（[]）通配符用于表示一定范围内的任意单个字符，它包括两端数据。

例 20.16 查询电话号码以“110”结尾并且开头数字位于 1~5 之间的员工信息，SQL 语句如下。

```
SELECT * FROM tb_employee WHERE phone LIKE '[1-5]30110'
```

## 20.5.5 组合条件查询（AND、OR 和 NOT）

如果想把前面讲过的几个单一条件组合成一个复合条件，这就需要使用逻辑运算符 AND、OR 和 NOT，才能完成复合条件查询。使用逻辑运算符时，遵循的指导原则如下所示：

- （1）使用 AND 返回满足所有条件的行。
- （2）使用 OR 返回满足任一条件的行。
- （3）使用 NOT 返回不满足表达式的行。

就像数据运算符乘和除一样，它们之间是具有优先级顺序的：NOT 优先级最高，AND 次之，OR 的优先级最低。

例 20.17 下面通过两个例子介绍 OR 和 AND 的使用。

用 OR 查询员工中姓“张”或者姓“李”的员工信息：

```
SELECT * FROM tb_employee WHERE name LIKE '张%' OR name LIKE '李%'
```

用 AND 查询员工中姓“张”并且“民族”是“汉族”的员工信息：

```
SELECT * FROM tb_employee WHERE name LIKE '张%' AND nation = '汉族'
```

## 20.6 高级查询

教学录像：光盘\TM\lx\20\高级查询

本节介绍的高级查询包括汇总数据、分组统计和子查询。

### 20.6.1 汇总数据

SQL 提供一组聚合函数，它们能够对整个数据集合进行计算，将一组原始数据转换为有用的信息，以便用户使用。例如求成绩表中的总成绩、学生表中的平均年龄等。

SQL 的聚合函数如表 20.3 所示。

表 20.3 聚合函数

聚 合 函 数	支持的数据类型	功 能 描 述
Sum()	数字	对指定列中的所有非空值求和
Avg()	数字	对指定列中的所有非空值求平均值
Min()	数字、字符、日期	返回指定列中的最小数字、最小的字符串和最早的日期时间
Max()	数字、字符、日期	返回指定列中的最大数字、最大的字符串和最近的日期时间
Count()	任意基于行的数据类型	统计结果集中全部记录行的数量。最多可达 2147483647 行

例 20.18 下面通过几个例子介绍 SQL 聚合函数的应用。

使用 AVG 函数求员工平均年龄：

```
SELECT sex, AVG(age) AS 平均年龄 FROM tb_employee GROUP BY sex
```

使用 Count 函数统计员工人数：

```
SELECT COUNT(number) AS 员工人数 FROM tb_employee
```

使用 Max 函数统计最大年龄：

```
SELECT MAX(age) AS 最大年龄 FROM tb_employee
```

使用 SUM 函数统计工资发放总额：

```
SELECT SUM(RealityPay) AS 工资发放总额 FROM tb_pay
```

## 20.6.2 分组统计

在 SQL 语句中，可以使用 GROUP BY 语句来实现按字段值相等的记录值进行的分组统计，语法如下。

```
SELECT filedlist FROM table WHERE criteria[GROUP BY groupfieldlist]
```

- ☑ fieldlist: 同任何字段名的别名、SQL 聚集函数、选择谓词 (ALL、DISTINCT、GROUP BY 语句、DISTINCTROW 或 TOP) 或其他 SELECT 语句选项一起被获取。
- ☑ Table: 从其中获取数据表的名称。
- ☑ Criteria: 选择标准。如果语句包含 WHERE 子句，则 Microsoft Jet 数据库引擎在对记录应用 WHERE 条件后会将这些值分组。
- ☑ Groupfieldlist: 用以记录分组的字段名，最多为 10 个字段。Groupfieldlist 中的字段名的顺序决定组层次，由分组的最高层次至最低层次。

例 20.19 分组统计员工中的男女人数，SQL 语句如下。

```
SELECT sex, COUNT(number) AS 人数 FROM tb_employee GROUP BY sex
```

⚠ 注意：GROUP BY 关键字一般用于同时查询多个字段并对字段进行算术运算的 SQL 命令中。



### 20.6.3 子查询

子查询是 SELECT 语句内的另外一条 SELECT 语句,而且常常被称为内查询或是内 SELECT 语句。SELECT、INSERT、UPDATE 或 DELETE 中允许是一个表达式的地方都可以包含子查询,子查询甚至可以包含在另外一个子查询中。

#### 1. 带有 IN 运算符的子查询

在带有 IN 运算符的子查询中,子查询的结果是一个结果集。父查询通过 IN 运算符将父查询中的一个表达式与子查询结果集中的每一个值进行比较,如果表达式的值与子查询结果集中的任何一个值相等,父查询中的“表达式 IN (子查询)”条件表达式返回 TRUE,否则返回 FALSE。NOT IN 运算符与 IN 运算符结果相反。

例 20.20 在员工信息和工资信息两个表中,查询已发工资的员工信息,SQL 语句如下。

```
SELECT * FROM tb_employee WHERE number IN(SELECT EmployeeNumber FROM tb_pay)
```

#### 2. 带有比较运算符的子查询

在带有比较运算符的子查询中,子查询的结果是一个单值。父查询通过比较运算符将父查询中的一个表达式与子查询结果(单值)进行比较,如果表达式的值与子查询结果做比较运算的结果为 TRUE,父查询中的“表达式比较运算符(子查询)”条件表达式返回 TRUE,否则返回 FALSE。

常用的比较运算符有: >、>=、<、<=、=、<>、!=、!>、!<

例 20.21 列出实发工资大于 1800 的员工的基本信息,SQL 语句如下。

```
SELECT * FROM tb_employee WHERE number IN(SELECT EmployeeNumber FROM tb_pay WHERE RealityPay > 1800)
```

## 20.7 插入数据

### 教学录像: 光盘\TM\lx\20\插入数据

本节介绍插入数据,包括插入完整的行、插入部分行、插入检索出的数据以及将一个表中的数据复制到另一个表。

#### 20.7.1 插入完整的行

插入完整行或部分行,应在 INSERT 语句中使用 VALUES 关键字,语法格式为:

```
INSERT INTO table_name[(column[,column2]...)]  
VALUES(CONSTANT[,CONSTANT2]...)
```

例 20.22 向员工表 tb\_employee 中添加一行新记录,并给每列都赋予一个新值,SQL 语句如下。

```
INSERT INTO tb_employee VALUES ('00030', '小李', '女', '220111XXXXXXXXXXXX', '1980-02-01', '28', '汉族', '96784')
```

⚠ 注意：必须使用与数据库表中字段名称相同的顺序输入数据值（即 number、name、sex、card、birth、age、nation、phone），数据值之间用逗号分开。VALUES 数据要用括号括起来，而且 SQL 要求对字符和日期数据用单引号封闭。

### 20.7.2 插入部分行

给部分列添加数据时，需要对这些列进行指定，那些不放入数据的列必须为默认值或定义为空，以防止出现错误。

例 20.23 只为员工表 tb\_employee 中的编号“number”和姓名“name”两列添加数据，SQL 语句如下。

```
INSERT INTO tb_employee (number, name) VALUES ('00031', '小王')
```

📖 说明：INSERT 语句对列名称顺序没有要求，只要给出的数据值与该顺序匹配即可。

### 20.7.3 插入检索出的数据

使用 SELECT 语句将数据添加到某一行的部分列而不是所有列，就像使用 VALUES 子句一样。在 INSERT 子句中简单地指定要添加数据的列。

例 20.24 如果在员工表中有员工编号等于“00031”的员工，而在工资表中却没有该员工，那么可以使用下面的语句将员工表中的员工编号等于“00031”的员工插入到工资表中，SQL 语句如下。

```
INSERT INTO tb_pay(EmployeeNumber, ID) SELECT number FROM tb_employee WHERE (number = '00031')
```

运行上述语句，将产生一个错误。原因是工资表 tb\_pay 中的 ID 不允许空值，而且没有默认值。这种情况下，可以将 01 作为 ID 的一个哑值（dummy value），并用它作为一个常量，如：

```
INSERT INTO tb_pay(EmployeeNumber, ID) SELECT number, 01 FROM tb_employee WHERE (number = '00031')
```

⚠ 注意：如果在列上使用了唯一索引或使用 UNIQUE 或 PRIMARY KEY 约束，就不能使用上述方法。

### 20.7.4 将一个表中的数据复制到另一个表

可以在 INSERT 语句中使用 SELECT 语句来获取一个或多个表中的值，在 INSERT 语句中使用 SELECT 语句的简单语法为：

```
INSERT INTO table_name[(insert_column)_list]
SELECT column_list
FROM table_list
WHERE search_conditions
```

在 INSERT 语句中的 SELECT 语句允许用户将数据从一个表的所有列或部分列移至另一个表中。如果要在—组列中插入数据，就可以在另一个时间使用 UPDATE 添加该值至其他的列。

如果需要一个表将行插入到另一个表中时，这两个表必须具有匹配的结构。也就是说，对应的列必须为同一个数据类型或系统可以在它们的数据类型上自动转换。

如果两个表中所有列的顺序与结构一致，那么就不必在表中指定列名。

**例 20.25** 将日消费信息表中的日结信息放到月消费信息表中，SQL 语句如下。

```
insert into 月消费信息表 select 箱号,所在大厅,项目编号,名称,单位,单价,数量,简称,消费状态,隐藏状态,登记时间,折扣,金额小计,消费单据号 from 日消费信息表 order by 消费单据号
```

或者：

```
insert into 月消费信息表 select * from 日消费信息表
```

如果这两表中的列在顺序上与数据库中表的结构不一致则可以使用 INSERT 或 SELECT 子句对列重新排序使它们匹配。

如果不匹配，系统就不能进行插入操作或不能正确地进行插入操作，它会将数据放置在错误的列中。

## 20.8 修改和删除数据

 教学录像：光盘\TM\lx\20\修改和删除数据.exe

本节介绍修改数据和删除数据。

### 20.8.1 修改数据

UPDATE 语句可以改变表中单一行、成组行和所有行中的值。下面是简化了的 UPDATE 语法。

```
UPDATE table_name
set column_name=expression
[WHERE search_conditions]
```

#### 1. 指定表：UPDATE 子句

UPDATE 关键字后面跟有表名或视图名，一次只可以改变一个表或一个视图中的数据。

如果 UPDATE 语句违背了完整性约束（例如，添加的某个值，这个值具有错误的数据类型），则系统就不能进行更新并显示一个错误信息。

#### 2. 指定列：SET 子句

SET 子句用于指定列和改变值。

**例 20.26** 计算工资表 tb\_pay 中的实发工资，实发工资等于应发工资减去应扣工资，语句如下。

```
UPDATE tb_pay SET RealityPay = MustPay - DelPay
```

#### 3. 指定行：WHERE 子句

UPDATE 语句中的 WHERE 子句指定要修改的行（类似于 SELECT 语句中的 WHERE 子句）。

例 20.27 将工资表 tb\_pay 中的基本工资在 1500 元的员工涨 200 元, SQL 语句如下。

```
Update tb_pay set BasePay = BasePay +200 where BasePay =1500
```

## 20.8.2 删除数据

删除单行或多行数据可使用 DELETE 语句, 语法格式如下。

```
DELETE FROM table_name WHERE search_conditions
```

例 20.28 删除员工表 tb\_empolyee 中的所有数据, SQL 语句如下。

```
DELETE FROM tb_employee
```

例 20.29 利用 WHERE 子句可以指定要删除哪一行。例如, 要删除员工表 tb\_empolyee 中的“年龄”小于 23 的数据, SQL 语句如下。

```
DELETE FROM tb_employee WHERE age < 23
```

## 20.9 小结

本章涉及的 SQL 语句是一些较为简单且常用的 SQL 语句。在实际编写数据库应用程序时, 还需要掌握大量的较为复杂的 SQL 语句, 这就需要读者多阅读一些有关 SQL 语言的专业图书。

## 20.10 练习与实践

1. 使用 AS 子句将英文字段重命名为中文字段。(答案位置: 光盘\TM\sl\20\4)
2. 创建一个销售表, 然后使用 SQL 语句统计各类商品的销售总额。(答案位置: 光盘\TM\sl\20\5)
3. 按日期查询销售表中某一商品的销售记录。(答案位置: 光盘\TM\sl\20\6)





# 第21章

## 数据库开发技术

(教学录像: 1 小时 4 分钟)

掌握了 SQL 语言, 再熟悉一些数据库开发技术, 就可以开发出数据库应用程序。本章介绍了几种数据库开发技术, 包括 ODBC、早期的 DAO 对象和 Data 控件, 以及当今流行的 ADO 对象和 ADO 控件。

通过阅读本章, 您可以:

- ▶▶ 认识 ODBC, 掌握配置 ODBC 的方法
- ▶▶ 学会引用 DAO 对象, 熟练掌握 DAO 对象, 以及用 DAO 对象实现数据的增、删、改、查
- ▶▶ 认识 Data 控件, 用 Data 控件连接数据库, 以及用 Data 控件实现数据的增、删、改、查
- ▶▶ 学会引用 ADO 对象, 熟练掌握 ADO 对象, 用其操纵数据库
- ▶▶ 认识 ADO 控件, 用 ADO 控件连接各种数据源、记录源
- ▶▶ 掌握 ADO 控件的主要属性、方法和事件, 以及用 ADO 控件实现数据的增、删、改、查



## 21.1 VB 访问数据库

 教学录像：光盘\TM\lx\21\VB 访问数据库.exe

VB 访问数据库有多种方法，如 Data 控件、DAO 对象、RDO 对象、ADO 控件和 ADO 对象等。

### ☒ Data 控件和 DAO 对象

Data 控件和 DAO 对象同属于 DAO (Data Access Objects) 技术，Data 控件是 VB 工具箱中的基本控件，使用 Data 控件可以打开、访问并操纵已有的数据库，它是操纵数据库最简便的方法。

DAO 对象是数据访问对象之一，是 VB 最早引入的数据访问技术。它比 Data 控件功能强大，不仅可以打开、访问并操纵已有的数据库，而且可以创建数据库、表和索引。另外，它不需要添加任何数据控件，只须程序代码就能创建完整的数据库应用程序，但使用该对象前应首先在工程中引用它。


### ☒ RDO 对象

RDO (Remote Data Objects) 远程数据对象是一个到 ODBC 的面向对象的数据访问接口，有了 Visual Basic 6.0 以后，RDO 已逐步被 ADO 替代，因此本章不作更多介绍。

### ☒ ADO 控件和 ADO 对象

ADO 控件从外形看与 Data 控件差不多，但功能却差很多。ADO 控件是最新的数据访问技术，访问更加简单和灵活，支持多种数据库，而且访问的数据类型也更为丰富，特别在 Internet 方面的应用可极大提高系统性能。如果需要少量的代码来创建数据库应用程序，则建议使用该控件。

ADO 对象是 DAO/RDO 对象的后继产物，它用较少的对象、更多的属性、方法（和参数）和事件，对各种数据源进行操作访问。

 说明：由于 ADO 控件和 ADO 对象是 Visual Basic 6.0 的最新数据访问技术，因此本章将重点介绍它们。

## 21.2 ODBC

 教学录像：光盘\TM\lx\21\ODBC.exe

ODBC 是目前访问远程数据库的主要方法，在开发远程数据库或其他数据库程序前，首先要配置好 ODBC 数据源。下面就介绍什么是 ODBC 和如何配置 ODBC 数据源。

### 21.2.1 认识 ODBC

ODBC (Open DataBase Connectivity, 开放数据库互连) 是 Microsoft 公司提供的有关数据库的一个组成部分，它建立一组规范并提供了数据库访问的标准 API (应用程序编程接口)。一个使用 ODBC 操作数据库的应用程序，基本操作都是由 ODBC 驱动程序完成的，不依赖于 DBMS。

应用程序访问数据库时，首先要用 ODBC 管理器注册一个数据源，这个数据源包括数据库位置、数据库类型和 ODBC 驱动程序等信息，管理器根据这些信息建立 ODBC 与数据库的连接。

## 21.2.2 配置 ODBC 数据源

配置 ODBC 数据源, 首先要打开 Windows 操作系统中的“控制面板”/“管理工具”/“数据源 (ODBC)”, 打开“ODBC 数据源管理器”, 如图 21.1 所示。然后单击“添加”按钮, 打开“创建新数据源”对话框, 如图 21.2 所示。在此选择 ODBC 提供驱动程序的数据源, 包括 Access 类驱动程序、dBase 类驱动程序、Excel 类驱动程序、FoxPro 类驱动程序、Visual FoxPro 类驱动程序、Paradox 类驱动程序、Text 类驱动程序、Oracle 类驱动程序和 SQL Server 类驱动程序。

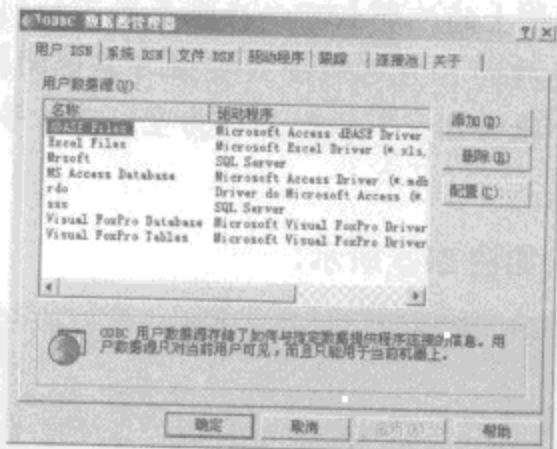


图 21.1 打开“ODBC 数据源管理器”



图 21.2 创建数据源

下面分别以常用数据库 Access 和 SQL Server 为例介绍配置 ODBC 数据源的方法。

## 1. Access 数据库 DSN 的配置方法

(1) 在如图 21.2 所示的对话框中, 拖曳滚动条, 在“名称”列表框中选择 Microsoft Access Driver (\*.mdb), 单击“完成”按钮, 打开“ODBC Microsoft Access 安装”对话框。

(2) 在“ODBC Microsoft Access 安装”对话框中的“数据源名”文本框中输入要创建的数据源名称, 例如 ODBCdb\_kfql。单击“选择”按钮, 打开“选择数据库”对话框, 在此选择数据源连接的 Access 数据库, 单击“确定”按钮, 连接的 Access 数据库路径将显示在“ODBC Microsoft Access 安装”对话框中, 如图 21.3 所示。

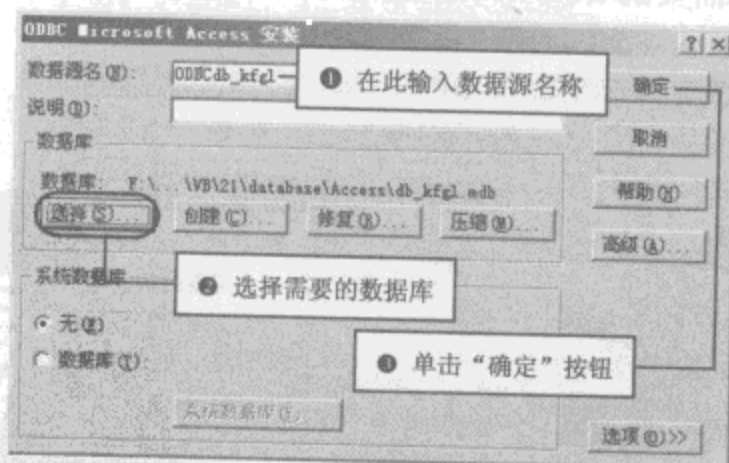


图 21.3 设置数据源

(3) 单击“确定”按钮, 新创建的数据源就会添加到如图 21.1 所示对话框中的数据源列表中了,



此时在如图 21.1 对话框中单击“确定”按钮，一个新的 Access 数据源就创建完成了。

## 2. SQL Server 数据库 DSN 配置方法

(1) 在如图 21.2 所示对话框中的“名称”列表框中选择 SQL Server，单击“完成”按钮，打开如图 21.4 所示的对话框。

- ☒ 在“名称”文本框内输入新的数据源名，例如 ODBCmanpowerinfo 作为新的数据源名称。
- ☒ 在“描述”文本框内输入对数据源的描述，也可以为空。这里没有输入内容。
- ☒ 在“服务器”下拉列表框中选择需要连接的服务器。

**注意：**如果要连接的 SQL Server 是安装在本地计算机上的，那么可以在服务器下拉列表框中选择 local，local 表示连接到本地服务器；如果要连接的 SQL Server 是安装在其他的服务器上的，则在列表框中选择所需的服务器名称；如果“服务器”下拉列表框为空，可以手动输入服务器名称。

(2) 单击“下一步”按钮进行下一步的配置工作，如图 21.5 所示。

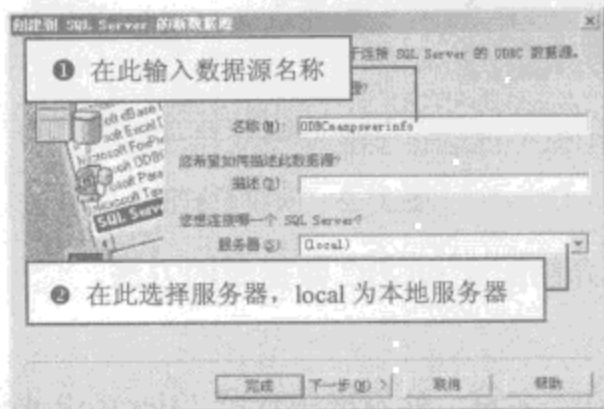


图 21.4 数据源信息设置

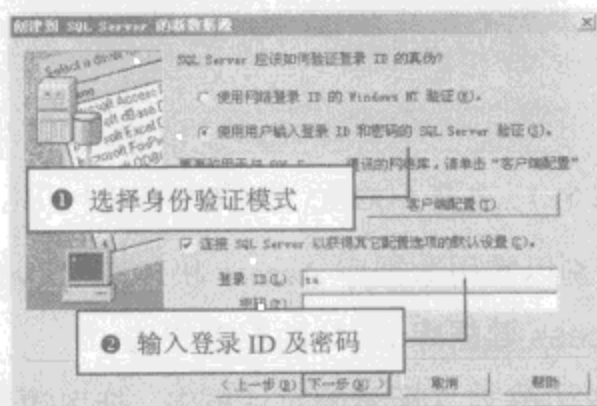


图 21.5 选择数据库验证方式

- ☒ 在“登录 ID”文本框中输入 sa。
- ☒ 在“密码”文本框中输入密码，这个密码是安装 SQL Server 时设置的，如果为空，则不输入。

(3) 单击“下一步”按钮，打开如图 21.6 所示的对话框，在此选中“更改默认的数据库为”复选框，同时在下拉列表框中选择需要的 SQL Server 数据库（例如选择 db\_manpowerinfo），然后单击“下一步”按钮。

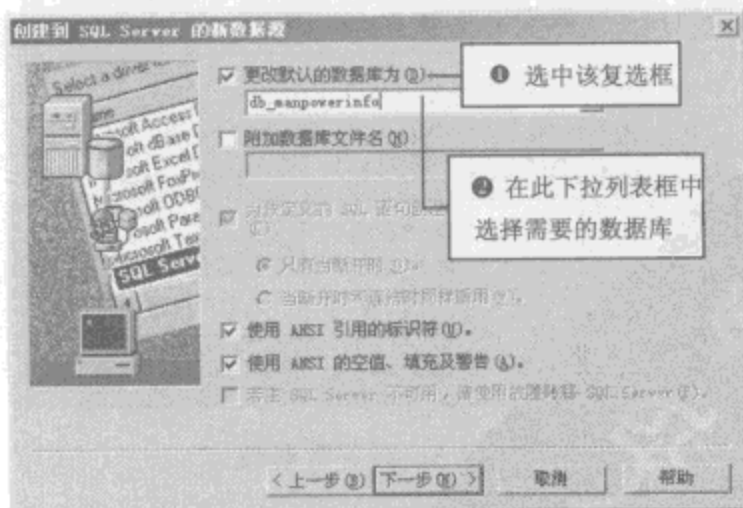


图 21.6 选择数据库

(4) 在打开的对话框中使用默认选项, 然后单击“完成”按钮, 打开“ODBC Microsoft SQL Server 安装”对话框。单击“测试数据源”按钮, 如果正确, 则连接成功; 如果不正确, 系统会指出具体的错误, 用户应该重新检查配置的内容是否正确。

(5) 单击“确定”按钮, 新创建的数据源就会添加到如图 21.1 所示对话框中的数据源列表中了, 此时在如图 21.1 对话框中单击“确定”按钮, 一个新的 SQL Server 数据源就创建完成了。

## 21.3 DAO 对象

 教学录像: 光盘\TM\lx\21\DAO 对象

数据访问对象 DAO (Data Access Objects) 是 Microsoft 公司推出的第一个基于面向对象开发技术基础上的数据库访问技术。可以在 VB 中编写代码来操纵诸如 Access 等一些小型数据库, 实现查询、添加、修改和删除数据, 以及创建数据库、表等功能。

下面主要介绍在程序中引用 DAO 对象、认识 DAO 对象及其子对象, 以及用 DAO 对象实现数据的增、删、改、查。

### 21.3.1 引用 DAO 对象

在使用 DAO 对象前, 应首先在工程中引用它, 具体步骤如下。

(1) 在 VB 工程中选择“工程”/“引用”命令, 打开“引用”对话框。

(2) 在“引用”对话框中选择 Microsoft DAO 3.6 Object Library 选项, 如图 21.7 所示, 然后单击“确定”按钮。

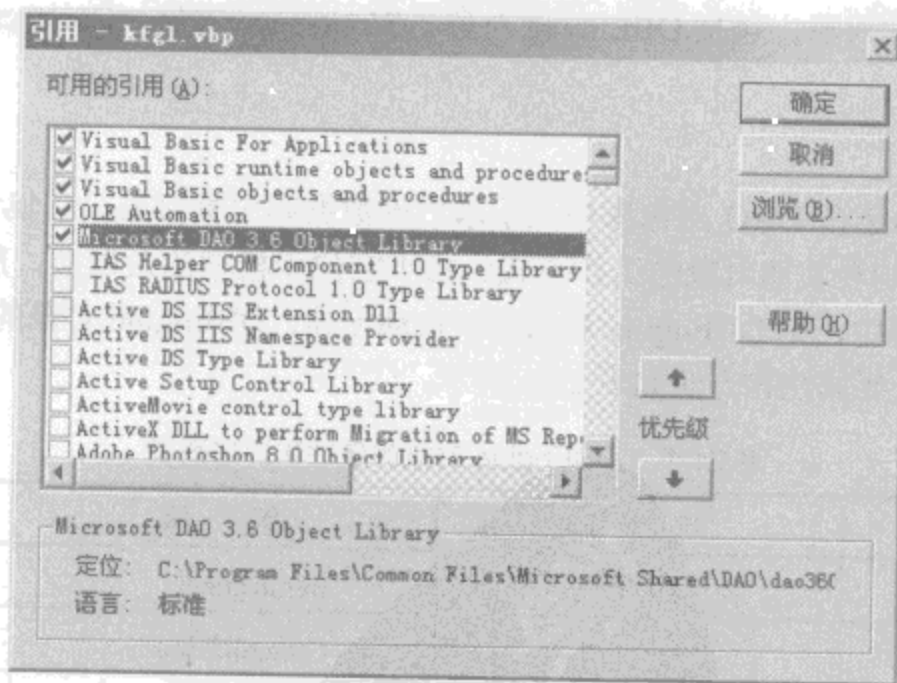



图 21.7 引用 DAO 对象

 说明: DAO 3.6 支持 Access 2000 以上版本的数据库, 如果没有该项, 则应安装 VB 6.0 补丁程序。

### 21.3.2 DAO 对象的子对象

DAO 对象还包含一些子对象，其层次结构如图 21.8 所示。

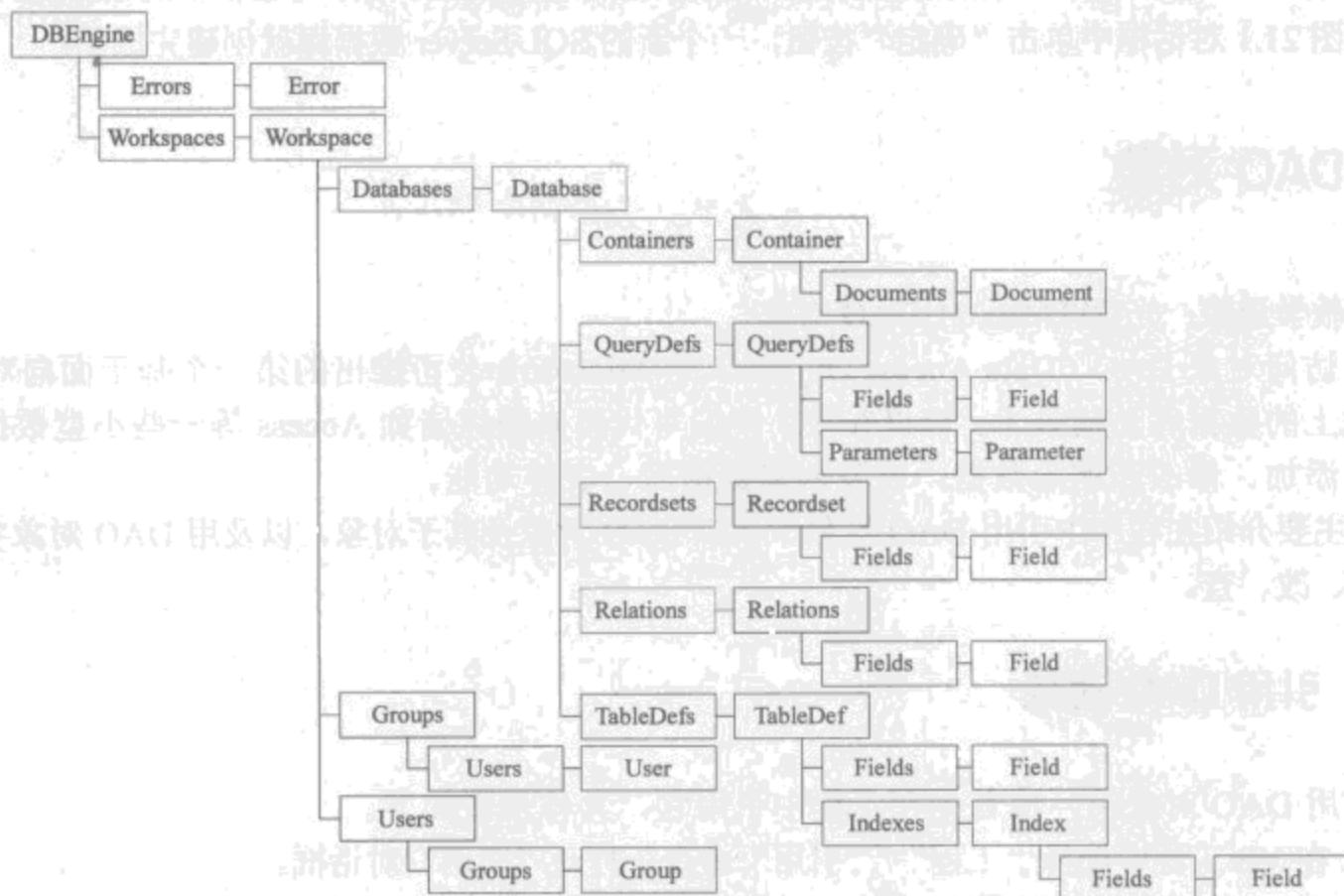


图 21.8 DAO 对象的层次结构

下面介绍几个主要的对象，包括 DBEngine 对象、Workspace 对象、Connection 对象、Database 对象和 Recordset 对象。

#### 1. DBEngine 对象

DBEngine 对象指的是在 DAO 对象库中处于顶层的数据库引擎对象，它包含和控制着 DAO 对象模型里的所有对象，影响着其他对象的工作方式。由于它位于模型的最顶层，所以不需要建立，只要将 DAO 引用到工程项目中，则 DBEngine 对象将被自动创建，DBEngine 对象的方法也可以直接调用。

DBEngine 对象的主要属性和方法如表 21.1 所示。

表 21.1 DBEngine 对象的属性和方法列表

属 性	说 明	方 法	说 明
DefaultPassword	默认密码	BeginTrans	开始事务
DefaultType	默认类型	CommitTrans	提交事务
DefaultUser	默认用户	CompactDatabase	压缩数据库
IniPath	初始路径	CreateDatabase	创建数据库
LoginTimeout	连接超时	CreateWorkspace	创建工作间

续表

属 性	说 明	方 法	说 明
		Idle	保持连接
		OpenConnection	打开连接
		OpenDatabase	打开数据库
		RegisterDatabase	注册数据库
		RepairDatabase	修复数据库
		Rollback	回滚事务
		SetOption	设置属性

## 2. Workspace 对象

一个 Workspace 对象对应于一个数据库用户。该对象是一个具有名字的数据工程区对象，也称“数据库空间对象”。

Workspace 对象为用户定义了一个会话，通过与之关联的用户名和口令建立一个安全级别。

Workspace 对象的主要属性和方法如表 21.2 所示。

表 21.2 Workspace 对象的主要属性和方法列表

属 性	说 明	方 法	说 明
CheckedOutToWorkspace	导出到工作间	Checkin	导入
Interface	接口	Checkout	导出
InternalID	内部标识	Create	创建
IsCheckedOut	是否导出	CreateObject	创建对象
IsFrozen	是否锁定	This	本对象
Name	名字	Delete	删除
Object	对象引用	Lock	锁定
ObjectID	对象标识	Open	打开
MajorDBVersion	数据库版本 1	Refresh	刷新
MinorDBVersion	数据库版本 2		
Repository			
ResolutionType			

Workspace 对象总是以集合 (Workspaces 集合) 的形式从属于 DBEngine 对象的。表示 Workspaces 集合中的对象有以下几种方式：

- ☒ DBEngine.Workspaces(0)
- ☒ DBEngine.Workspaces("name")
- ☒ DBEngine.Workspaces![name]



例 21.1 可以利用 Workspace 对象的 CreateDatabase 方法创建一个新的数据库。在下面的语句中，使用了两个参数，一个用以指定数据库的名称，另一个用以指定区域。

```
Set MyWs = DBEngine.Workspaces(0)
Set MyDB = MyWs.CreateDatabase("C:\Data\Biblio.mdb", dbLangGeneral)
```

### 3. Connection 对象

Connection 对象的作用是管理 DAO 程序与 ODBC 数据源的连接。Connection 对象是与远程数据库连接的过程同时存在的，只要连接结束，Connection 对象的生命期就结束了。

Connection 对象的主要集合、属性和方法如表 21.3 所示。

表 21.3 Connection 对象的主要集合、属性和方法列表

集 合 (Collections)	说 明	属 性	说 明	方 法	说 明
Fields	字段集合	Attributes	属性	CreateField	创建字段
Indexes	索引集合	ConflictTable	冲突的表	CreateIndex	创建索引
Properties	属性集合	Connect	连接引用	CreateProperty	创建属性
		DateCreated	创建日期	OpenRecordset	打开数据集
		KeepLocal(user-defined)	保持在本地	RefreshLink	刷新连接
		LastUpdated	最后更新		
		Name	名字属性		
		RecordCount	记录数		

Connection 对象只在 ODBC Direct Workspaces 对象中存在，这种 Workspaces 对象构造时对于类型的选择需要使用 dbUseODBC 关键字。

例 21.2 用 Connection 对象、Workspace 对象和 Recordset 对象连接数据库和数据表。(实例位置：光盘\TM\sl\21\1)

```
Dim wrkODBC As Workspace      '定义一个 Workspace 对象
Dim cnKFGL As Connection      '定义一个 Connection 对象
Dim myrs As Recordset         '定义一个 Recordset 对象
Set wrkODBC = CreateWorkspace("NewODBCWorkspace", "admin", "", dbUseODBC) '创建一个工作空间
Set cnKFGL = wrkODBC.OpenConnection("Connection1", dbDriverNoPrompt,
"ODBC;DATABASE=db_kfgl;DSN=ODBCdb_kfgl") '创建一个用于连接 ODBC 的 Connection 对象
Set myrs = cnKFGL.OpenRecordset("kf", dbOpenSnapshot, dbRunAsync) '连接数据表 kf
myrs.MoveFirst                '移到第一条记录
Debug.Print myrs.Fields("房间号") & myrs.Fields("房态") '将记录输出到立即窗口
cnKFGL.Close: wrkODBC.Close '关闭连接，关闭工作空间
```

### 4. Database 对象

Database 对象即数据库对象，用来管理一个打开的数据库连接。Database 对象是对数据库实施操作时首先要使用的对象。

Database 对象用于创建一个永久性的数据库连接（相对于 Connection 对象而言）。使用 DBEngine

对象的 OpenDatabase 方法可以打开一个数据库连接,但是执行 OpenDatabase 方法后,DAO 对象并不是真正与该数据库进行连接,而是在打开相应的 Recordset 对象或者把 TableRef 对象与具体的表连接时才会真正连接目标数据库。

Database 对象的主要集合、方法和属性如表 21.4 所示。

表 21.4 Database 对象的主要集合、属性和方法列表

集合 (Collections)	说 明	属 性	说 明	方 法	说 明
Containers	包含数据库对象的容器集合	Connect	连接引用	Close	关闭连接
Indexes	索引集合	QueryTimeout	查询超时设置	CreateQueryDef	创建查询
QueryDefs	查询集合	LastUpdated	最后更新	CreateProperty	创建属性
TableDefs	表集合	Name	名字属性	CreateTableDef	创建表对象
				Execute	运行 SQL
				NewPassword	新密码
				OpenRecordset	打开数据集

例 21.3 定义一个数据库对象 mydb1,使其连接数据库 db\_kfgl.mdb,代码如下。

```
Dim mydb1 As Database           '定义一个 Database 对象
Set mydb1 = Workspaces(0).OpenDatabase(App.Path & "\db_kfgl.mdb") '使其连接数据库 db_kfgl.mdb
```

## 5. Recordset 对象

Recordset 对象即记录集对象,是进行数据库操作中最常用的对象,它用于管理从数据表中或者从自定义查询中返回的记录集。Recordset 对象是 Database 对象的子对象,因而在建立 Recordset 对象之前需要先建立 Database 对象。

Recordset 对象的主要集合、属性和方法如表 21.5 所示。

表 21.5 Recordset 对象的主要集合、属性和方法列表

集合 (Collections)	说 明	属 性	说 明	方 法	说 明
Fields	字段集合	EOF	记录集最后一条之后	AddNew	添加记录
		BOF	记录集第一条之前	Cancel	取消
		Connection	连接引用	CancelUpdate	取消更新
		EditMode	编辑模式	Close	关闭记录集
		LastModified	最后更新	Delete	删除记录集
		Name	名字	Edit	编辑记录集
		RecordCount	记录集数	etRows	获得记录集数组
		RecordStatus	记录状态	Move	记录集移动
		Restartable	重新启动	MoveFirst	移到第一条

续表

集合 (Collections)	说 明	属 性	说 明	方 法	说 明
Fields	字段集合	Updatable	记录可更新性	MoveLast	移到最后一条
		BookMark	书签	MoveNext	移到下一条
				MovePrevious	移到前一条
				NextRecordset	下一个记录集
				Requery	重新查询
				Update	更新记录集

Recordset 对象是 DAO 编程中使用频率最高的对象。所有的 Recordset 对象都是由记录 (records) 和字段 (fields) 组成的。

例 21.4 定义一个记录集对象 myrs1, 使其连接一个数据表, 代码如下。

```
Dim myrs1 As Recordset                                '定义一个 Recordset 对象
Set myrs1 = mydb1.OpenRecordset("kf", dbOpenTable)    '使其连接数据表 kf
```

### 21.3.3 DAO 对象的综合应用

使用 DAO 对象/Recordset 对象的 AddNew、Edit、Update 和 Delete 方法可以实现数据的增加、修改和删除, 使用 MoveFirst、MoveLast、MoveNext 和 MovePrevious 方法可以查看指定的数据。

例 21.5 下面使用 DAO 对象实现数据的增、删、改、查。以客房管理为例, 运行程序, 单击“添加”按钮, 添加客房信息, 如图 21.9 所示。单击“保存”按钮, 将该信息保存到数据表中; 单击“修改”按钮, 首先浏览需要修改的数据, 如图 21.10 所示, 然后进行修改或删除。(实例位置: 光盘\TM\sl\21\2)

图 21.9 添加客房信息

图 21.10 修改或删除客房信息

说明: 修改数据后, 同样单击“保存”按钮, 保存修改后的数据。

程序实现步骤如下。

(1) 新建一个工程, 在该工程中引用 DAO 对象。

(2) 在窗体中添加两个 Frame 控件, 均使用默认名称。在 Frame1 中添加一些用于录入数据的 Label、TextBox 和 ComboBox 控件, 在 Frame2 中添加一个名为 lblRecordCount 的 Label 控件和一个包含 4 个元素的名为 cmdMove 的 CommandButton 控件数组。

(3) 在窗体中添加 6 个 CommandButton 控件, Caption 属性分别为“添加”、“修改”、“保存”、“取消”、“删除”和“退出”; 名称属性分别为 cmdAdd、cmdModify、cmdSave、cmdCancel、cmdDelete 和 cmdExit。

(4) 程序主要代码如下。

在通用声明部分, 声明数据库对象和记录集对象, 代码如下。

```
Dim DAOmydb1 As Database           '定义一个数据库对象
Dim DAOmyrs1 As Recordset         '定义一个记录集对象
```

定义一个布尔型标记, 用于判断是新增数据还是修改数据。其值为 True, 表示新增数据; 其值为 False, 表示修改数据, 代码如下。

```
Dim Addbln As Boolean
```

窗体载入时, 连接数据库, 初始化“房间类型”和“房态”, 代码如下。

```
Private Sub Form_Load()
    '连接客房管理数据库 db_kfgl.mdb
    Set DAOmydb1 = Workspaces(0).OpenDatabase(App.Path & "\db_kfgl.mdb")
    Controls_State False
    '向“房间类型”下拉列表框 Combo1 中添加房间类型
    Combo1.AddItem "普房"
    Combo1.AddItem "标房"
    Combo1.AddItem "双人间"
    Combo1.AddItem "套房"
    Combo1.ListIndex = 0
    '向“房态”下拉列表框 Combo2 中添加房间状态
    Combo2.AddItem "空房"
    Combo2.AddItem "入住"
    Combo2.AddItem "维修"
    Combo2.AddItem "打扫中"
    Combo2.ListIndex = 0
    '初始化窗体的高度
    Me.Height = 3045
End Sub
```

单击“添加”按钮, 设置标记 Addbln 为 True, 也就是“新增数据”。初始化控件, 同时使“房间号”文本框 txtfjh 获得焦点。代码如下。

```
Private Sub cmdAdd_Click()
    Addbln = True           '设置标记为 True
    Controls_State True     '调用 Controls_State 过程, 设置控件状态
    '清空控件中的数据
    txtfjh.Text = ""
    txtjg.Text = ""
    txtpz.Text = ""
    txtbz.Text = ""
    txtfjh.SetFocus        '“房间号”文本框 txtfjh 获得焦点
End Sub
```



单击“修改”按钮，设置标记 Addbln 为 False，也就是“修改数据”。设置相应控件的状态，连接表，调用 cmdMove\_Click(0)事件过程，在控件中显示第一条记录。代码如下。

```
Private Sub cmdModify_Click()
    Addbln = False
    Me.Height = 4065
    Controls_State True
    cmdDelete.Visible = True
    Set DAOmyrs1 = DAOmydb1.OpenRecordset("select * from kf order by 房间号", dbOpenSnapshot)
    Call cmdMove_Click(0)
End Sub
```

单击“第一条”、“上一条”等按钮，浏览记录。调用 ViewData 子过程，将表中数据显示在控件中，同时显示记录总数和当前记录位置。代码如下。

```
Private Sub cmdMove_Click(Index As Integer)
    With DAOmyrs1
        Select Case Index
            Case Is = 0
                .MoveFirst '移到第一条记录
            Case Is = 1
                .MovePrevious '移到上一条记录
                If .BOF Then '如果记录到头
                    MsgBox "记录已到头！" '提示用户
                    .MoveFirst '移到第一条记录
                End If
            Case Is = 2
                .MoveNext '移到下一条记录
                If .EOF Then '如果记录到尾
                    MsgBox "记录已到尾！" '提示用户
                    .MoveLast '移到最后一条记录
                End If
            Case Is = 3
                .MoveLast '移到最后一条记录
        End Select
        Call ViewData '调用 ViewData 子过程，在控件中显示记录
        '在 lblRecordCount 中显示总记录数和当前记录位置
        lblRecordCount.Caption = "共 " & .RecordCount & " 条记录 第 " & .AbsolutePosition + 1 & " 条记录"
    End With
End Sub
```

单击“保存”按钮，将当前记录保存到表中。如果标记 Addbln 为 True，则将用户新录入的数据使用 AddNew 方法和 Update 方法保存到表中；否则先使用 FindFirst 方法查找当前记录，然后使用 Edit 方法和 Update 方法修改该记录。代码如下。

```
Private Sub cmdSave_Click()
    '不允许“房间号”、“房间类型”和“房态”为空
    If txtfjh.Text = "" Or Combo1.Text = "" Or Combo2.Text = "" Or txtjg.Text = "" Then
        MsgBox "此项不允许为空！"
```

```

Exit Sub
End If
If Addbln = True Then
    Set DAOmyrs1 = DAOmydb1.OpenRecordset("kf", dbOpenTable)
    DAOmyrs1.AddNew
    '给表中各字段赋值
    DAOmyrs1.Fields("房间号") = txtfjh.Text
    DAOmyrs1.Fields("房间类型") = Combo1.Text
    DAOmyrs1.Fields("房态") = Combo2.Text
    DAOmyrs1.Fields("单价") = txtjg.Text
    DAOmyrs1.Fields("配置") = txtpz.Text
    DAOmyrs1.Fields("备注") = txtbz.Text
    DAOmyrs1.Update
Else
    Set DAOmyrs1 = DAOmydb1.OpenRecordset("kf", dbOpenDynaset)
    DAOmyrs1.FindFirst "房间号 like " + Chr(34) + txtfjh + Chr(34) + ""
    DAOmyrs1.Edit
    '给表中各字段赋值
    DAOmyrs1.Fields("房间号") = txtfjh.Text
    DAOmyrs1.Fields("房间类型") = Combo1.Text
    DAOmyrs1.Fields("房态") = Combo2.Text
    DAOmyrs1.Fields("单价") = txtjg.Text
    DAOmyrs1.Fields("配置") = txtpz.Text
    DAOmyrs1.Fields("备注") = txtbz.Text
    DAOmyrs1.Update
    cmdDelete.Visible = False
    Me.Height = 3045
End If
'调用 Controls_State 子过程，设置控件状态
Controls_State False
End Sub

```

'如果标记 Addbln 为 True，也就是“新增数据”  
'连接表  
'增加记录  
'更新数据表  
'否则标记 Addbln 为 False，也就是“修改数据”  
'连接表  
'查找记录  
'编辑记录  
'更新数据表  
'“删除”按钮不可见  
'设置窗体的高度

单击“删除”按钮，首先使用 FindFirst 方法查找当前记录，然后使用 Delete 方法删除它。删除完成后调用事件过程 cmdMove\_Click (2)，将记录移到下一条。代码如下。

```

Private Sub cmdDelete_Click()
    Set DAOmyrs1 = DAOmydb1.OpenRecordset("kf", dbOpenDynaset)
    DAOmyrs1.FindFirst "房间号 like " + Chr(34) + txtfjh + Chr(34) + ""
    DAOmyrs1.Delete
    cmdMove_Click (2)
End Sub

```

'连接表  
'查找记录  
'删除记录  
'调用事件过程 cmdMove\_Click (2)，显示下一条记录

## 21.4 Data 控件

 教学录像：光盘\TM\lx\21\Data 控件

Data 控件是 VB 早期在数据库编程方面的技术。本节从认识 Data 控件开始，到使用 Data 控件连接数据库，以及使用 Data 控件添加、修改和删除数据，使读者快速掌握 Data 控件。

### 21.4.1 认识 Data 控件

Data 控件是 VB 中的基本控件，可以直接从工具箱中添加到工程中并使用。下面是 Data 控件在工具箱中及其添加到窗体上的图标，如图 21.11 和图 21.12 所示。



图 21.11 Data 控件在工具箱中的图标

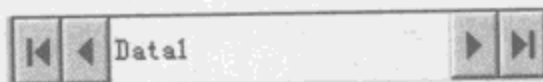


图 21.12 Data 控件在窗体中的图标

### 21.4.2 用 Data 控件连接数据库

要使用 Data 控件对表中的记录进行增加、修改、查询和删除等操作，首先应将 Data 控件连接到指定的数据库，这就要求读者熟练掌握 Data 控件的 DatabaseName 属性和 RecordSource 属性。

#### 1. 通过 DatabaseName 属性连接数据库

使用 DatabaseName 属性，可以连接到指定的 Access 数据库，具体步骤如下。

(1) 用鼠标选中窗体中的 Data 控件，然后在“属性”窗口中选择 DatabaseName 属性，并且单击旁边的按钮，如图 21.13 所示。

(2) 在打开的对话框中选择所要连接的数据库，然后单击“打开”按钮，如图 21.14 所示。



图 21.13 设置 Data 控件的 DatabaseName 属性

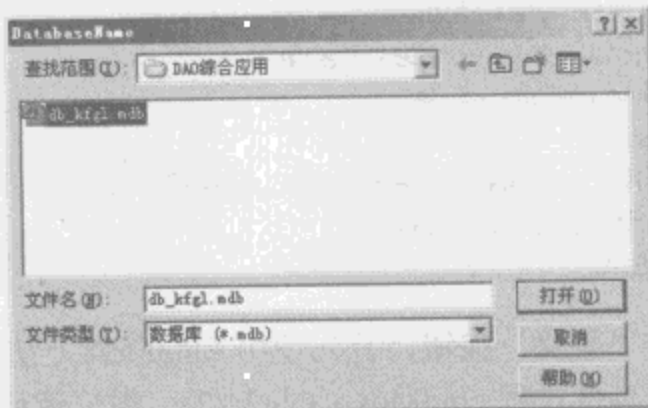


图 21.14 选择所要连接的 Access 数据库

#### 2. 通过 RecordSource 属性连接表

在通过 DatabaseName 属性连接完数据库以后，紧接着的操作就是通过 RecordSource 属性连接数据表了。

RecordSource 属性确定具体可访问的数据，这些数据构成记录集对象 Recordset。该属性值能够返回数据库中的单个表。可以在 RecordSource 属性中选择所要连接的数据表，如图 21.15 所示。

### 21.4.3 Data 控件的综合应用

利用数据绑定控件，编写少量代码就可以实现数据的增、删、改、查，因为绑定的数据控件已连接到数据表中的不同字段。录入数据时，只须使用 AddNew 方法添加一条新记录，然后在绑定控件中

录入相关数据, 录入完成后, 使用 Update 方法更新即可完成数据的增加; 修改数据则首先使用 Edit 方法使当前记录处于编辑状态, 修改完成后, 同样使用 Update 方法更新即可完成数据的修改; 删除数据时首先使用 Delete 方法删除当前数据, 然后使用 Refresh 方法刷新数据表。

例 21.6 下面改写例 21.5 使用 Data 控件绑定 TextBox 控件和 ComboBox 控件, 实现客房管理。运行程序, 结果如图 21.16 所示。(实例位置: 光盘\TM\sl\21\3)



图 21.15 连接数据表

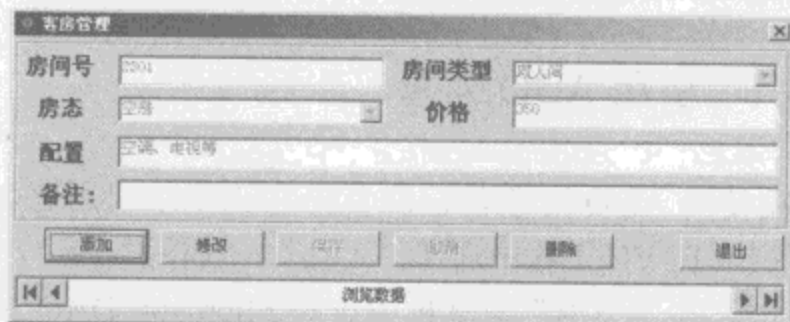


图 21.16 Data 控件的综合应用

程序具体实现步骤如下。

- (1) 在工程中添加一个 Data 控件, 通过前面介绍的方法连接数据库 db\_kfgl 和表 kf。
- (2) 设置各个控件的 DataSource 属性为 Data1, DataField 属性为对应的字段名称。
- (3) 程序主要代码如下。

```
Private Sub cmdAdd_Click()
    Controls_State True           '调用 Controls_State 子过程, 设置控件状态
    Data1.Recordset.AddNew        '添加新记录
    txtfjh.Text = "": txtjg.Text = "": txtpz.Text = "": txtbz.Text = ""    '清空文本框
    txtfjh.SetFocus               '房间号文本框 txtfjh 获得焦点
End Sub

Private Sub cmdModify_Click()
    Controls_State True           '调用 Controls_State 子过程, 设置控件状态
    Data1.Recordset.Edit          '编辑记录
End Sub

Private Sub cmdSave_Click()
    '不允许“房间号”、“房间类型”和“房态”为空
    If txtfjh.Text = "" Or Combo1.Text = "" Or Combo2.Text = "" Or txtjg.Text = "" Then
        MsgBox "此项不允许为空!"
        Exit Sub
    End If
    Data1.Recordset.Update
    Controls_State False          '调用 Controls_State 子过程, 设置控件状态
End Sub

Private Sub cmdDelete_Click()
    Data1.Recordset.Delete        '删除记录
    Data1.Refresh                 '刷新数据表
End Sub
```



## 21.5 ADO 对象

 教学录像：光盘\TM\lx\21\ADO 对象

本节将介绍 ADO 对象的使用方法，认识 ADO 对象的子对象。重点介绍 ADO 对象的 3 种对象——Connection 对象、Recordset 对象和 Command 对象。

### 21.5.1 引用 ADO 对象

在使用 ADO 对象前，应首先在工程中引用它，具体步骤如下。

- (1) 在 VB 工程中选择“工程”/“引用”命令，打开“引用”对话框。
- (2) 在“引用”对话框中选择 Microsoft DAO 3.6 Object Library 选项，如图 21.17 所示，然后单击“确定”按钮。

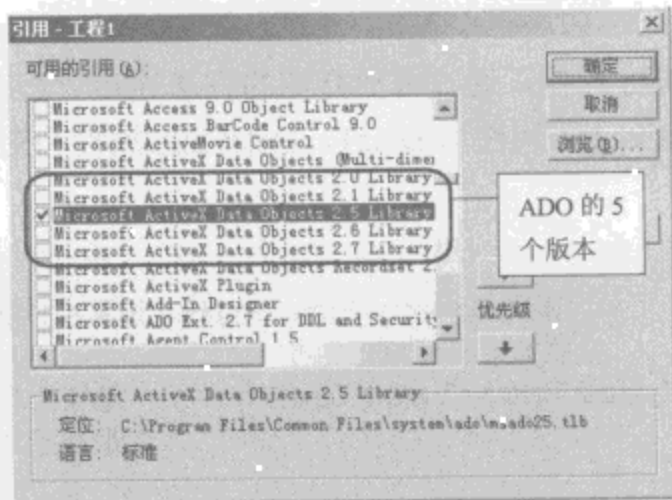



图 21.17 引用 ADO 对象

在“引用”对话框中选择 Microsoft ActiveX Data Object 2.5 Library。“2.5”是 ADO 的版本号。在如图 21.17 所示的“可用的引用”列表中，从 ADO 2.0 到 ADO 2.7 一系列的产品都可以选用。ADO 是向下兼容的，新的 ADO 组件版本兼容使用低版本 ADO 开发出来的程序。

设置完成后，就可以在工程中应用 ADO 了。ADO 组件库的前缀是 ADODB，例如使用 Connection 对象时，应表示为 ADODB.Connection；使用 Recordset 对象时，应表示为 ADODB.Recordset。

### 21.5.2 ADO 对象的子对象

ADO 对象还包含一些子对象，其层次结构如图 21.18 所示。

 说明：ADO 对象的层次结构图和 DAO 对象的层次结构图，验证了我们开头所说的 ADO 对象是 DAO/RDO 对象的后继产物。它用较少的对象、更多的属性、方法（与参数）和事件，对各种数据源进行操作访问。

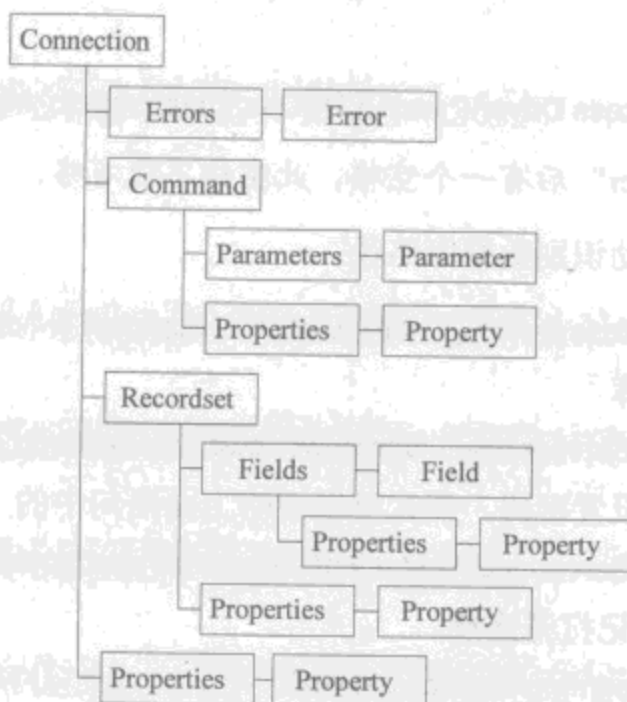


图 21.18 ADO 对象的层次结构

### 21.5.3 连接多种数据库（Connection 对象）

Connection 对象的 Open 方法可以连接多种数据源，其语法格式如下。

```
connection.Open ConnectionString, UserID, Password, OpenOptions
```

- ☑ ConnectionString: 可选参数，一个字符串，包含连接信息。可参见 ConnectionString 属性提供的 4 个参数，如表 21.6 所示。
- ☑ UserID: 可选参数，一个字符串，包含建立连接时所使用的用户名称。
- ☑ Password: 可选参数，一个字符串，包含建立连接时所用密码。
- ☑ OpenOptions: 可选参数。如果其值为 adAsyncConnect，则异步打开连接；如果其值为 adConnectUnspecified（默认值），则同步打开连接。

下面介绍 ADO 支持的 ConnectionString 属性的 4 个参数，参数说明如表 21.6 所示。

表 21.6 ADO 支持的 ConnectionString 属性的 4 个参数


参 数	说 明
Provider	指定用来连接的提供者名称
File Name	指定包含预先设置连接信息的特定提供者的文件名称（例如，持久数据源对象）
Remote Provider	指定打开客户端连接时使用的提供者名称（仅限于 Remote Data Service）
Remote Server	指定打开客户端连接时使用的服务器的路径名称（仅限于 Remote Data Service）

下面介绍使用 Connection 对象连接数据库的方法。在使用 Connection 对象连接数据库之前，应首先声明该对象，声明如下：

```
Dim cn As New ADODB.Connection
```

## 1. 连接 Access 数据库

```
cn.Open "Driver={Microsoft Access Driver (*.mdb)};DBQ=c:\database\db_kfjl.mdb"
```

 注意：上面的语句中“Driver”后有一个空格，此空格不能省略。

连接 Access 数据库时，自动识别数据库路径：

```
cn.Open "Driver={Microsoft Access Driver (*.mdb)};DBQ=" & app.path & "\db_kfjl.mdb"
```

## 2. 连接 SQL Server 数据库

```
cn.Open "Driver={SQL Server};Server=local;uid=sa;pwd=pwd;database=db_manpowerinfo"
```

如果在 SQL Server 中没有设置密码，那么可以省略上面语句中的“pwd”。如：

```
cn.Open "Driver={SQL Server};Server=local;uid=sa;database=db_manpowerinfo "
```

## 3. 使用 DSN 和 ODBC 标记打开连接

```
cn.Open "DSN=ODBCmanpowerinfo;uid=sa;pwd=pwd;"
```

如果数据库中没有设置密码，那么可以省略上面语句中的“pwd”。如：

```
cn.Open "DSN= ODBCmanpowerinfo;uid=sa;"
```

## 4. 使用 DSN 和 OLE DB 标记打开连接

```
cn.Open "DataSource=ODBCmanpowerinfo;user ID=sa;"
```

## 5. 使用 DSN 和单个参数而非连接字符串打开连接

```
cn.Open "ODBCmanpowerinfo","sa"
```

在对打开的 Connection 对象的操作结束后，可使用 Close 方法释放所有关联的系统资源。关闭对象并将它从内存中删除。可以更改它的属性设置并在以后再次使用 Open 方法打开它。要将对象完全从内存中删除，可将对象变量设置为 Nothing，如 set cn=Nothing。

## 21.5.4 连接记录源（Recordset 对象）

Recordset 对象表示的是来自基本表或命令执行结果的记录全集。任何时候，Recordset 对象所指的当前记录均为集合内的单个记录。

使用 Recordset 对象的 Open 方法可以打开代表基本表、查询结果或者以前保存的 Recordset 中记录的游标，其语法格式如下。

```
recordset.Open Source, ActiveConnection, CursorType, LockType, Options
```

- ☒ Source: 可选参数，变体型，表示对象的变量名、SQL 语句、表名、存储过程调用或持久 Recordset 文件名。
- ☒ ActiveConnection: 可选参数，变体型，表示有效的 Connection 对象变量名或字符串，包含 ConnectionString 参数。
- ☒ CursorType: 可选参数，用于确定提供者打开 Recordset 对象时应该使用的游标类型，这些类型如表 21.7 所示。

表 21.7 打开 Recordset 时可使用游标类型

常 量	说 明
adOpenForwardOnly	(默认值) 打开仅向前类型游标
adOpenKeyset	打开键集类型游标
adOpenDynamic	打开动态类型游标
adOpenStatic	打开静态类型游标

- ☒ LockType: 可选参数, 用于确定提供者打开 Recordset 对象时应该使用的锁定 (并发) 类型的 LockTypeEnum 值, 这些值如表 21.8 所示。

表 21.8 打开 Recordset 时应使用的锁定类型

常 量	说 明
adLockReadOnly	(默认值) 只读。不能改变数据
adLockPessimistic	保守式锁定 (逐个)。提供者完成确保成功编辑记录所需的工作, 通常通过在编辑时立即锁定数据源的记录来完成
adLockOptimistic	开放式锁定 (逐个)。提供者使用开放式锁定, 只在调用 Update 方法时才锁定记录
adLockBatchOptimistic	开放式批更新。用于批更新模式 (与立即更新模式相对)

- ☒ Options: 可选参数, 长整型值, 用于指示提供者如何指定 Source 参数, 其值如表 21.9 所示。

表 21.9 Options 常量

常 量	说 明
adCmdText	指示提供者应该将 Source 作为命令的文本定义来计算
adCmdTable	指示 ADO 生成 SQL 查询以便从 Source 命名的表返回所有行
adCmdTableDirect	指示提供者更改从 Source 命名的表返回的所有行
adCmdStoredProc	指示提供者应该将 Source 视为存储的过程
adCmdUnknown	指示 Source 参数中的命令类型为未知
adCommandFile	指示应从 Source 命名的文件中恢复持久 (保存的) Recordset
adExecuteAsync	指示应异步执行 Source
adFetchAsync	指示在提取 CacheSize 属性中指定的初始数量后, 应该异步提取所有剩余的行

使用 Recordset 对象打开表之前, 应首先声明该对象, 声明如下:

```
Dim rs1 As New ADODB.Recordset
```

例 21.7 下面使用 Recordset 对象打开“客房信息表”kf, 代码如下。(实例位置: 光盘\TM\sl\21\4)

```
Dim cnn As New ADODB.Connection      '定义一个 Connection 对象
Dim rs As New ADODB.Recordset        '定义一个 Recordset 对象
```



```

Private Sub Form_Load()
    cnn.Open "Driver={Microsoft Access Driver (*.mdb)};DBQ=" & App.Path & "\db_kfgl.mdb" '连接数据库
    rs.Open "kf", cnn, adOpenKeyset, adLockOptimistic '连接数据表
    rs.MoveFirst '移动到第一条记录
    '使用循环语句将所有房间号输出到立即窗口
    Do While rs.EOF = False
        Debug.Print rs.Fields("房间号")
        rs.MoveNext
    Loop
    rs.close '关闭数据集对象
End Sub

```

例 21.8 Recordset 对象还可以打开带查询结果的表。例如，显示所有“空房”，只须将上一例中的 rs.Open "kf", cnn, adOpenKeyset, adLockOptimistic，改写为如下代码。（实例位置：光盘\TM\sl\21\5）

```
rs.Open "select * from kf where 房态='空房'", cnn, adOpenKeyset, adLockOptimistic
```

⚠ 注意：使用 Recordset 对象的 Open 方法打开数据表，程序结束后应使用 Close 方法关闭该对象。如果不关闭，再次使用该对象时，将出现运行时错误，并提示对象被打开。

### 21.5.5 执行 SQL 语句（Command 对象）

使用 Command 对象查询数据库并返回 Recordset 对象中的记录，以便执行大量操作或处理数据库结构。某些 Command 集合、方法或属性被引用时可能会产生错误。这取决于提供者的功能。

可以使用 Command 对象的集合、方法、属性进行下列操作：

- ☒ 使用 CommandText 属性定义命令（例如 SQL 语句）的可执行文本。
- ☒ 通过 Parameter 对象和 Parameters 集合定义参数化查询或存储过程参数。
- ☒ 可使用 Execute 方法执行命令并在适当的时候返回 Recordset 对象。
- ☒ 执行前应使用 CommandType 属性指定命令类型以优化性能。
- ☒ 使用 Prepared 属性决定提供者是否在执行前保存准备好（或编译好）的命令版本。
- ☒ 使用 CommandTimeout 属性设置提供者等待命令执行的秒数。
- ☒ 通过设置 ActiveConnection 属性使打开的连接与 Command 对象关联。
- ☒ 设置 Name 属性将 Command 标识为与 Connection 对象关联的方法。
- ☒ 将 Command 对象传送给 Recordset 的 Source 属性以便获取数据。

要独立于先前已定义的 Connection 对象创建 Command 对象，请将它的 ActiveConnection 属性设置为有效的连接字符串。此时 ADO 仍将创建 Connection 对象，只是它不会将该对象赋给对象变量。但是，如果将多个 Command 对象与同一个连接关联，则必须显式创建并打开 Connection 对象，这样即可将 Connection 对象赋给对象变量。如果没有将 Command 对象的 ActiveConnection 属性设置为该对象变量，则即使使用相同的连接字符串，ADO 也会为每个 Command 对象创建新的 Connection 对象。

要执行 Command，只需通过它所关联的 Connection 对象的 Name 属性，将其简单调用即可，但必须将 Command 的 ActiveConnection 属性设置为 Connection 对象。如果 Command 带有参数，还要将这些参数的值作为参数传送给方法。

例 21.9 下面使用 Command 对象执行 SQL 语句。例如, 查询“客房信息表”kf 中, 房间号为“2301”的记录, 代码如下。(实例位置: 光盘\TM\sl\21\6)

```
Dim cnn As New ADODB.Connection           '定义一个 Connection 对象
Dim rs As New ADODB.Recordset             '定义一个 Recordset 对象
Dim cmd As New ADODB.Command              '定义一个 Command 对象
Private Sub Form_Load()
    cnn.Open "Driver={Microsoft Access Driver (*.mdb)};DBQ=" & App.Path & "\db_kfgl.mdb" '连接数据库
    Set cmd.ActiveConnection = cnn          '给 Command 对象指定连接对象
    cmd.CommandText = "select * from kf where 房间号='2301'" '设置命令文本
    cmd.CommandType = adCmdText            '设置命令类型
    cmd.CommandTimeout = 15                '设置执行命令需等待的时间
    Set rs = cmd.Execute                   '返回记录集对象
    MsgBox "该房间已找到, 房间号为【" & rs.Fields("房间号") & "】" '输出记录
End Sub
```

注意: 在程序中引用 ADO 对象。

### 21.5.6 ADO 对象的综合应用

使用 ADO 对象实现增、删、改应首先引用 ADO 对象, 然后使用 ADO 对象的 AddNew、Update 和 Delete 方法。

例 21.10 下面将使用 ADO 对象实现经手人信息的添加、修改和删除。运行程序, 结果如图 21.19 所示。(实例位置: 光盘\TM\sl\21\7)

经手人编号	经手人姓名	联系方式	联系地址	身份证号	备注
002	李*民	13009224912	长春市卫星路113号	22010319791	无
003	王华山	8988699	长春市红旗街99号	22018319801	暂时经手人
005	李津	12121212	东大桥附近	22010319820	无
007	鲁伟	1300900898	建设街131号	22010319830	无
008	张三	23343434	长春	220*****	无

图 21.19 ADO 对象的综合应用

程序主要代码如下。

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Key
        Case "add"
            blnadd = True '设置标记为“添加数据”
            '清空文本框, 解除锁定
            For i = 1 To Text1.UBound
                Text1(i) = ""
                Text1(i).Locked = False
            Next i
    End Select
End Sub
```

```

Text1(0).SetFocus           '设置经手人编号 Text1(0)获得焦点
Case "save"
    If blnadd = True Then    '如果标记为 True, 即添加新记录
        rs.Open "select * from 经手人表", cn, adOpenKeyset, adLockOptimistic '连接数据表
        '向表中添加新记录
        With rs
            .AddNew
            .Fields("经手人编号") = Text1(0).Text: .Fields("经手人姓名") = Text1(1).Text
            .Fields("联系方式") = Text1(2).Text: .Fields("联系地址") = Text1(3).Text
            .Fields("身份证号") = Text1(4).Text: .Fields("备注") = Text1(5).Text
            .Update
        End With
        rs.Close
    End If
    If blnadd = False Then   '如果标记为 False, 即修改记录
        rs.Open "select * from 经手人表 where 经手人编号=" + Text1(0).Text + "'", cn, adOpenKeyset,
adLockOptimistic           '按经手人编号查询指定的经手人
        If rs.RecordCount > 0 Then '如果记录大于零, 则修改该记录
            With rs
                .Fields("经手人编号") = Text1(0).Text: .Fields("经手人姓名") = Text1(1).Text
                .Fields("联系方式") = Text1(2).Text: .Fields("联系地址") = Text1(3).Text
                .Fields("身份证号") = Text1(4).Text: .Fields("备注") = Text1(5).Text
                .Update
            End With
        End If
        rs.Close
    End If
Case "xg"
    blnadd = False          '设置标记为 False, 即修改记录
    '此处省略了解除锁定和取消操作的代码, 这部分代码可参见光盘中的源程序
Case "del"
    rs.Open "select * from 经手人表 where 经手人编号=" + Text1(0).Text + "'", cn, adOpenKeyset,
adLockOptimistic           '按经手人编号查询指定的记录
    If rs.RecordCount > 0 Then '如果记录大于零
        rs.Delete             '删除该记录
        rs.Update             '更新数据表
    End If
    rs.Close                 '关闭记录集对象
Case "refresh"
    '刷新数据表
    Adodc1.RecordSource = "经手人表"
    Adodc1.Refresh
Case "close"
    Unload Me                '关闭窗口体
End Select
End Sub

```

## 21.6 ADO 控件

### 教学录像：光盘\TM\lx\21\ADO 控件

本节主要介绍认识 ADO 控件、用 ADO 控件连接各种数据源、用 ADO 控件连接记录源、ADO 控件常用属性、方法和事件，以及使用 ADO 控件实现数据的增、删、改、查。

### 21.6.1 认识 ADO 控件

ADO 是 ActiveX 控件，使用时应首先将其添加到工具箱中。选择“工程”/“部件”，打开“部件”对话框，选择 Microsoft ADO Data Control 6.0 (SP4) (OLEDB)，单击“确定”按钮，ADO 控件将添加到工具箱中，双击 ADO 控件图标，将其添加到窗体。添加过程如图 21.20 所示。

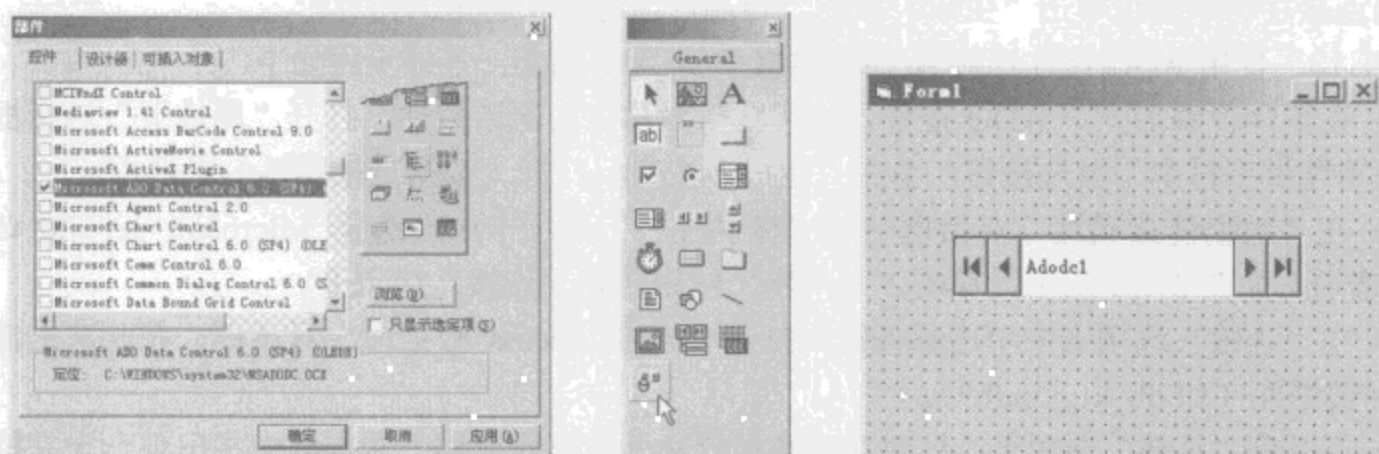


图 21.20 ADO 控件添加过程

使用 ADO 控件可以实现以下功能：

- ☒ 连接一个本地数据库或远程数据库。
- ☒ 打开一个指定的数据库表或定义一个基于结构化查询语言 (SQL) 的查询、存储过程或该数据库中表视图的记录集合。
- ☒ 将数据字段的数值传递给数据绑定控件，并在这些控件中显示或更改这些数据字段的数值。
- ☒ 添加新的记录，或者根据显示在数据绑定控件中的数据的数据的任何更改来更新一个数据库。

### 21.6.2 用 ADO 控件连接各种数据源

连接数据源是访问数据库的第一步，ADO 控件通过其 `ConnectionString` 属性来连接各种数据源。方法是右击 ADO 控件，打开“属性页”对话框，在此对话框中允许用户通过 3 种不同的方式来连接数据源，下面分别进行介绍。

#### 1. 使用 Data Link 文件

表示通过一个 ODBC 文件数据源连接文件来完成。这个文件是在 ODBC 数据源（文件 DSN）中



事先创建好的。

**例 21.11** 下面以 test.dsn 文件为例介绍使用 Data Link 文件连接数据源。

在 ADO 控件的“属性页”对话框中选择“使用 Data Link 文件”单选按钮，单击“浏览”选择需要的 ODBC 文件数据源（如光盘中提供的 test.dsn 文件）。选择完成后，返回到“属性页”对话框，此时“使用 Data Link 文件”单选按钮下的文本框中将出现一个字符串，单击“确定”按钮完成设置。

## 2. 使用 ODBC 数据资源名称

通过下拉菜单选择已创建好的 ODBC 数据源名称（用户 DSN）作为数据来源。

**例 21.12** 下面以 21.2.2 节（配置 ODBC 数据源）中创建的 ODBC 数据源 ODBCmanpowerinfo 为例介绍使用 ODBC 数据资源名称连接数据源。

在 ADO 控件的“属性页”对话框中选择“使用 ODBC 数据资源名称”单选按钮，然后在其下拉列表框中选择 ODBCmanpowerinfo，单击“确定”按钮完成设置。


## 3. 使用连接字符串

使用连接字符串通过单击“生成”按钮，自动产生连接字符串的内容。

**例 21.13** 使用 ADO 控件连接 Access 数据库 db\_kfgl.mdb，具体步骤如下。

(1) 在窗体上添加一个 ADO 控件，使用默认名称，右击该控件，打开“属性页”对话框，在此选择“使用连接字符串”单选按钮，单击“生成”按钮，打开“数据链接属性”对话框。

(2) 在“数据链接属性”对话框中的“提供程序”选项卡中，选择 Microsoft Jet 4.0 OLE DB Provider 选项，如图 21.21 所示，单击“下一步”按钮转到“连接”选项卡。

(3) 在“连接”选项卡中单击  按钮，打开“连接 Access 数据库”对话框，在此双击需要连接的 Access 数据库（如 db\_kfgl.mdb）。

(4) 返回到“连接”选项卡，这时在“选择或输入数据库名称”文本框中将出现一个完整的数据库路径，如图 21.22 所示。



图 21.21 选择数据提供者

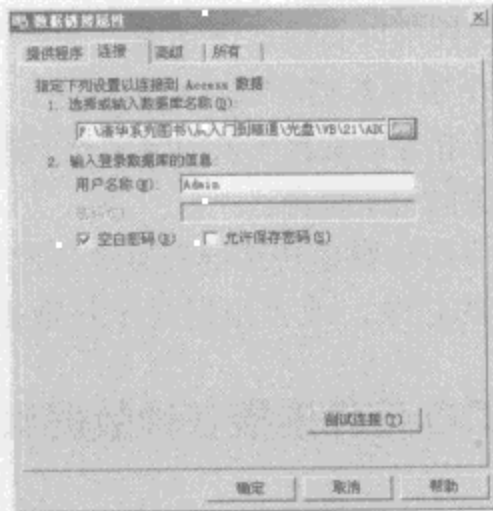


图 21.22 选择 Access 数据库

(5) 在“输入登录数据库的信息”文本框中输入用户名和密码，一般使用默认设置，单击“测试连接”按钮，测试连接是否成功。如果连接成功，单击“确定”按钮，返回到“属性页”对话框，在“使用连接字符串”文本框中将会看到已经生成的连接字符串，该字符串如下：

Provider=Microsoft.Jet.OLEDB.4.0;Data Source=F:\清华系列图书\从入门到精通\光盘\VB\21\ADO 控件

```
\db_kfgl.mdb;Persist Security Info=False
```

至此, ADO 控件便成功地连接了 db\_kfgl.mdb 数据库。

另外, 除了使用上面介绍的方法设置 ConnectionString 属性外, 还可以通过代码设置该属性。例如, 窗体载入时, 连接数据库, 代码如下。

```
Adodc1.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=F:\清华系列图书\从入门到精通\光盘\VB21\ADO 控件\db_kfgl.mdb;Persist Security Info=False"
```

若需要自动识别数据库路径, 可以使用 App.Path, 代码如下:

```
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path &
"\db_kfgl.mdb;Persist Security Info=False"
```

通过 ADO 控件的“使用连接字符串”还可以连接 SQL Server。首先选择数据提供者 Microsoft OLE DB Provider for SQL Server, 然后选择服务器, 输入用户名称和密码(一般为 sa, 密码为空), 最后选择需要连接的数据库。

### 21.6.3 用 ADO 控件连接记录源

使用 ADO 控件的 RecordSource 属性可以连接指定的记录源。

**例 21.14** 使用 RecordSource 属性连接 db\_kfgl.mdb 数据库中的 kf 表。

(1) 右击 ADO 控件, 打开“属性页”对话框, 选择“记录源”选项卡, 如图 21.23 所示。

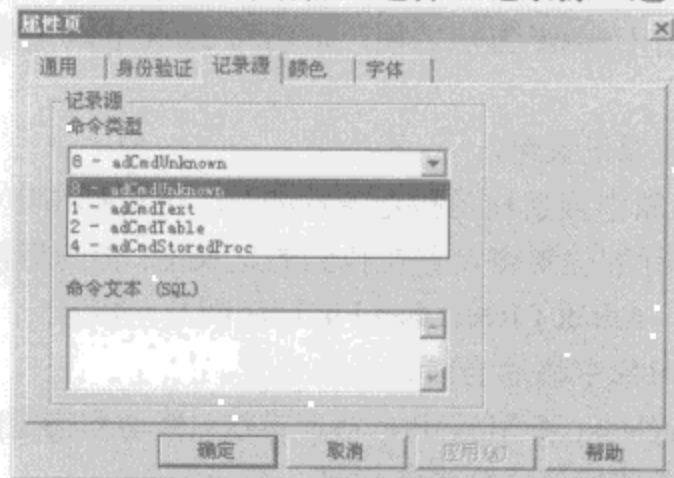


图 21.23 记录源

(2) “命令类型”下拉列表中有 4 个选项供用户选择。它们的说明如表 21.10 所示。

表 21.10 命令类型说明

类 型 名 称	类 型 说 明
8-adCmdUnknown	CommandText 属性中的命令类型未知, 是默认类型
1-adCmdText	将 CommandText 作为命令或存储过程调用的文本定义进行计算
2-adCmdTable	将 CommandText 作为全部由内部生成的 SQL 语句返回的表格的名称进行计算
4-adCmdStoredProc	将 CommandText 作为存储过程名进行计算

下面使用“1-adCmdText”类型，在“命令文本”中输入 SQL 语句，该语句如下。

```
select * from kf where 房态='空房'
```

(3) 单击“确定”按钮，关闭记录源属性页。

至此，连接表的工作便完成了。

另外，连接记录源还可以通过代码实现。例如，窗体载入时，连接记录源代码如下：

```
Adodc1.CommandType = adCmdText
Adodc1.RecordSource = "select * from kf where 房态='空房'"
```

## 21.6.4 ADO 控件常用属性、方法和事件

ADO 控件有很多属性、方法和事件，下面只介绍几个重点的属性、方法和事件。

**AbsolutePosition 属性：**用于设置或返回当前记录的序号位置，如 Adodc1.Recordset.AbsolutePosition。

**ActiveConnection 属性：**用于指定 Command、Recordset 或 Record 对象当前所属的 Connection 对象。

**BOF 属性：**表示当前记录位置位于 Recordset 对象的第一个记录之前。

**EOF 属性：**表示当前记录位置位于 Recordset 对象的最后一个记录之后。

**RecordCount 属性：**返回记录总数，如 a=Adodc1.Recordset.RecordCount。

**AddNew 方法：**用于创建新记录。

**CancelUpdate 方法：**在调用 Update 方法之前，取消对 Recordset 对象的当前行、新行或者 Record 对象的 Fields 集合所做的更改。

**Delete 方法：**删除当前记录或记录组。

**Find 方法：**在 Recordset 对象中搜索满足指定条件的行。可选择指定搜索方向、起始行和从起始行的偏移量。如果满足条件，当前行的位置将设置在找到的记录上；否则将把当前行位置设置为 Recordset 对象的结尾（或开始）处。如：Adodc1.Recordset.Find "房间号> 4001"

**Move 方法：**在 Recordset 对象中移动当前记录的位置。

**MoveFirst、MoveLast、MoveNext 和 MovePrevious 方法：**移动到指定的 Recordset 对象中的第一个、最后一个、下一个或上一个记录并使其成为当前记录。

**Update 方法：**保存对 Recordset 对象的当前行或者 Record 对象的 Fields 集合所做的更改。

**Error 事件：**在执行 VB 代码而发生了一个数据访问错误的情况下，会发生这个事件。

**例 21.15** 如果提供给用户输入（修改）数据的控件是数据绑定控件，那么唯一索引字段的值发生重复时，ADO 控件会产生 Error 事件，通过该事件，可以处理错误的发生。如下面的代码：

```
Private Sub Adodc1_Error(ByVal ErrorNumber As Long, Description As String, ByVal Scode As Long, ByVal
Source As String, ByVal HelpFile As String, ByVal HelpContext As Long, fCancelDisplay As Boolean)
    If ErrorNumber = -2147217873 Then
        Scode = 0
        MsgBox Description
    End If
End Sub
```

## 21.6.5 ADO 控件的综合应用

利用数据绑定控件，只用少量代码即可实现数据的增、删、改，因为绑定的数据控件已连接到数据表中的不同字段。录入数据时，只须使用 AddNew 方法添加一条新记录，然后在绑定控件中录入相关数据，录入完成后，使用 Update 更新即可完成数据的增加。修改数据则可以直接进行，删除数据可以使用 Delete 方法。

例 21.16 下面使用 ADO 控件实现经手人数据的增、删、改。运行程序，结果如图 21.24 所示。  
(实例位置：光盘\TM\sl21\8)

经手人编号	经手人姓名	联系方式	联系地址	身份证号	备注
002	李*民	13009224****	长春市卫星路113号	22010319791****	无
003	王华山	8968699	长春市红旗街99号	22018319801102	暂时经手人
005	李津	7977899	东大桥附近	22010319820301	无
006	刘伟				

图 21.24 ADO 控件的综合应用

程序主要代码如下：

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Key
        Case "add"
            Adodc1.Recordset.AddNew                                '添加新记录
            '此处省略了清空文本框和解除锁定的代码，这部分代码可参见光盘中的源程序
            Text1(0).SetFocus                                       '使经手人编号 text1(0)获得焦点
        Case "save"
            Adodc1.Recordset.Update                                '更新数据表
        Case "xg"
            '此处省略了解除锁定的代码，这部分代码可参见光盘中的源程序
        Case "cancel"
            '此处省略了锁定文本框的代码，这部分代码可参见光盘中的源程序
        Case "del"
            Adodc1.Recordset.Delete                                '删除记录
            Adodc1.Recordset.Update                                '更新数据表
        Case "close"
            Unload Me                                              '关闭窗口体
    End Select
End Sub
```

⚠ 注意：利用绑定控件录入数据时，如果输入非法数据，程序将报错，因此要设计遇错处理程序。



## 21.7 小结

本章介绍了几种数据库访问技术，重点介绍了 ADO 控件和 ADO 对象。这两种访问数据库的方法，究竟使用哪种，要根据对数据库技术掌握的熟练程度以及自身编程习惯。但是笔者还是提出一点建议：对于初学者，建议使用 ADO 控件（数据绑定方法），因为编写少量代码就可以快速地创建数据库应用程序；对于有一些编程经验的程序员，建议使用 ADO 对象，以方便程序日后维护和移植。当然，二者也可以结合应用。

## 21.8 练习与实践

1. 使用 DAO 对象创建一个 Access 数据库，其中包括一个表，表中有编号、商品名称、单位、单价和数量字段。（答案位置：光盘\TM\sl\21\9）
2. 使用 ADO 的 Connection 对象执行 SQL 语句，更新销售表（xsd）中的金额。光盘中提供数据库 db\_zbjxc.mdb。（答案位置：光盘\TM\sl\21\10）
3. 使用 ADO、DataGrid 和 DTPicker（日期控件）控件，按日期查询销售表（xsd）中的数据。光盘中提供数据库 db\_zbjxc.mdb。（答案位置：光盘\TM\sl\21\11）

# 第22章

## 数据库控件

(教学录像: 43 分钟)

本章主要介绍数据库控件,包括早期的和当前流行的数据库控件,如 DBCombo 和 DBList 控件, DataCombo 和 DataList 控件, DataGrid 控件, MSFlexGrid 和 MSHFlexGrid 控件。它们是编写数据库应用程序所必须掌握的知识,有了这些数据库控件,操纵数据将变得更加方便快捷。

通过阅读本章,您可以:

- » 学会 DBCombo 和 DBList 控件的日常应用
- » 认识 DataCombo 和 DataList 控件
- » 掌握 DataCombo 和 DataList 控件显示普通数据和关系数据的方法
- » 认识 DataGrid 控件,学会用 DataGrid 控件显示数据、格式化数据和锁定数据
- » 学会将 DataGrid 控件中的数据显示在文本框中
- » 认识 MSFlexGrid 和 MSHFlexGrid 控件,学会用 MSHFlexGrid 控件显示数据
- » 掌握 MSHFlexGrid 控件对数据进行排序、合并的方法
- » 掌握隐藏 MSHFlexGrid 控件的行、列和冻结字段的方法

## 22.1 DBCombo 和 DBList 控件

 教学录像：光盘\TM\lx\22\DBCombo 和 DBList 控件.exe

DBCombo 和 DBList 控件与 Data 控件绑定可以快速方便地实现在下拉列表框和列表框中显示数据表中的数据。这两个控件都属于 ActiveX 控件，使用时应首先将其添加到工具箱中。选择“工程”/“部件”，打开“部件”对话框，在此选择 Microsoft Data Bound List Controls 6.0，单击“确定”按钮，将其添加到工具箱中，然后在工具箱中分别双击这两个控件的图标，即可将它们添加到窗体上。添加过程如图 22.1 所示。

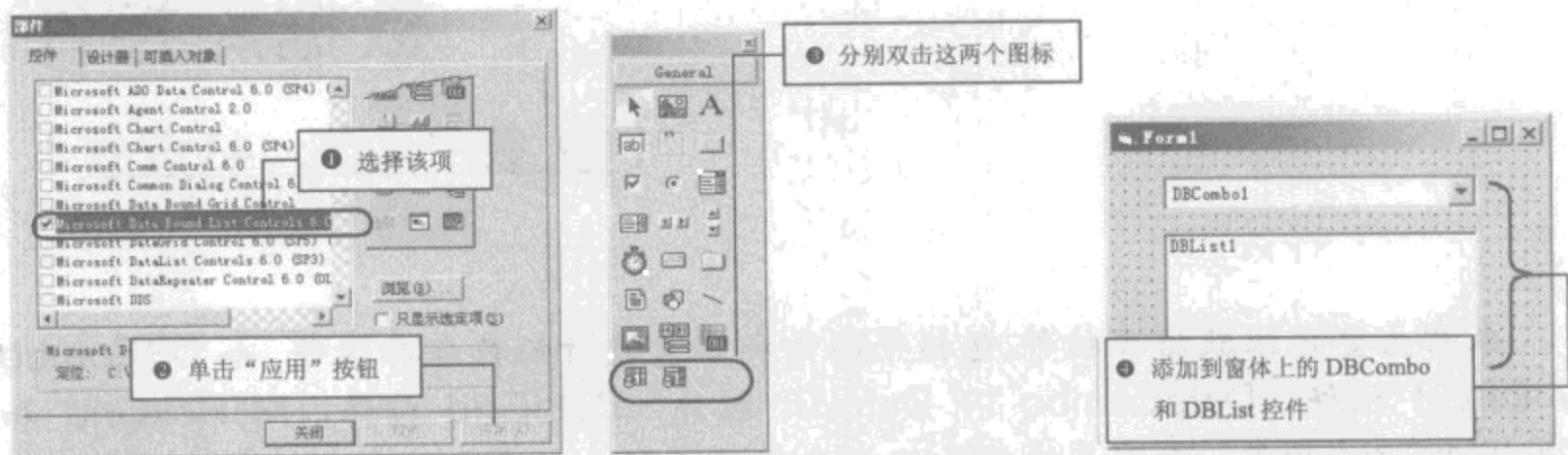


图 22.1 DBCombo 和 DBList 控件的添加过程

将 DBCombo 和 DBList 控件添加到窗体上后，接着在窗体上添加 Data 控件，然后简单地设置一下这几个控件的属性，就可以显示数据表中的数据了。

**例 22.1** 下面分别介绍使用 DBCombo 和 DBList 控件绑定到 Data 控件显示客房信息的过程。


(1) 新建一个工程，在窗体上添加一个 Data 控件，通过上一章介绍的方法，使其连接数据库 db\_kfgl.mdb 和其中的数据表 kf。为了使其运行时不可见，设置 Visible 属性为 False。（实例位置：光盘\TM\sl\22\1）

(2) 在窗体上添加一个 DBCombo 控件和一个 DBList 控件。

(3) 单击 DBCombo 控件，在“属性”窗口中找到 RowSource 属性，在其旁边的下拉列表框中选择 Data1，接下来找到 ListField 属性，在其旁边的下拉列表框中选择“房间号”。

(4) DBList 控件的设置方法与 DBCombo 控件一样，这里就不介绍了。

(5) 按〈F5〉键，运行程序，效果如图 22.2 所示。

 **说明：**由于 DBCombo 和 DBList 控件只能绑定到 Data 控件、RDO 控件等旧的数据源中，因此这里就不做更多地介绍，本章重点介绍的是下面几种数据控件。

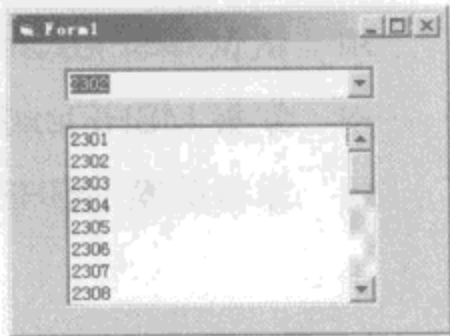


图 22.2 在列表中显示房间号

## 22.2 DataCombo 和 DataList 控件

 教学录像：光盘\TM\lx\22\DataCombo 和 DataList 控件

DataCombo 控件是一个数据绑定组合框，而 DataList 控件是一个数据绑定列表框。它们可以自动地由一个附加数据源中的一个字段填充，并且可选择地更新另一个数据源相关表中的一个字段。

### 22.2.1 认识 DataCombo 和 DataList 控件

DataCombo 和 DataList 控件与 ADO 控件绑定可以快速方便地实现在下拉列表框和列表框中显示数据表中的数据。这两个控件都属于 ActiveX 控件，使用时应首先将其添加到工具箱中。选择“工程”/“部件”，打开“部件”对话框，在此选择 Microsoft DataList Controls 6.0 (SP3)，单击“确定”按钮，将其添加到工具箱中，然后在工具箱中分别双击这两个控件的图标，即可将它们添加到窗体上。添加过程如图 22.3 所示。

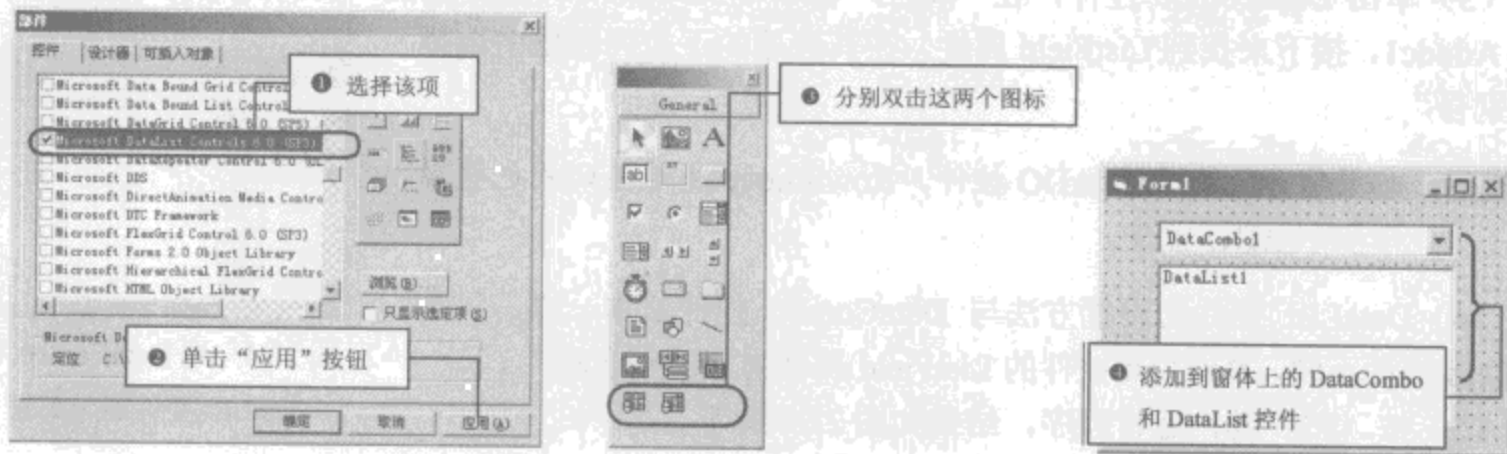


图 22.3 DataCombo 和 DataList 控件的添加过程

### 22.2.2 DataCombo 和 DataList 控件的属性

DataCombo 和 DataList 控件有一些特殊的属性，这使得它们与其他的数据控件不同。下面介绍几个重点的属性。

☒ **DataSource 属性**

指定 DataCombo 和 DataList 控件被绑定到的 ADO 控件的名称。

☒ **DataList 属性**

指定由 DataSource 属性所指定的记录集中的字段名称。该字段将用来决定数据列表中哪个元素将被突出显示。如果需要做出新的选择，则在移动到新记录的时候，该字段被更新。

☒ **RowSource 属性**

用于填充列表的 ADO 控件绑定的数据库的名称。

☒ **ListField 属性**

由 RowSource 属性所指定的记录集中的字段名来填充列表。




### ☒ BoundColumn 属性

用于返回或者设置一个由 RowSource 属性指定的在记录集中的字段名, 该记录集用来为另一个 Recordset 对象提供数据。选择确定后, 回传到 DataField。

### ☒ BoundText 属性

用于返回或者设置由 BoundColumn 属性指定的字段的值。选择确定后, 该值被回传从而更新由 DataSource 和 DataField 属性指定的 Recordset 对象。


 **注意:** DataCombo 和 DataList 控件与 ADO 控件绑定在一起使用的, 因此当使用它们时, 应该先使用 ADO 控件指定数据库和表。

**例 22.2** 下面使用 DataCombo 和 DataList 控件显示“图书类别表”中的类别编号和类别名称, 具体实现步骤如下。(实例位置: 光盘\TM\sl\22\2)

(1) 新建一个工程, 在窗体中添加 ADO 控件, 通过上一章介绍的方法, 使其连接数据库 db\_book 和数据表“图书类别表”。

(2) 按照 22.2.1 节介绍的方法在窗体上添加一个 DataCombo 控件和一个 DataList 控件。

(3) 单击 DataCombo 控件, 在“属性”窗口中找到 RowSource 属性, 在其旁边的下拉列表框中选择 Adodc1, 接下来找到 ListField 属性, 在其旁边的下拉列表框中选择“类别号”。

 **说明:** 如果窗体上有几个 ADO 控件, RowSource 属性值列表也将显示多个 ADO 控件。

(4) DataList 控件的设置方法与 DataCombo 控件基本一样, 只是 ListField 属性不同, DataList 控件的 ListField 属性为“类别名称”。

(5) 按〈F5〉键, 运行程序, 结果如图 22.4 所示。

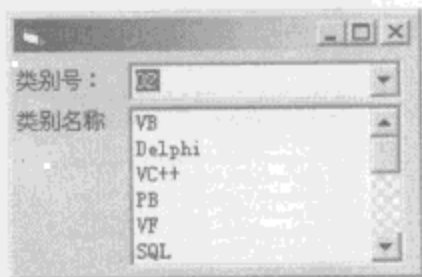


图 22.4 用 DataCombo 和 DataList 控件显示数据

## 22.2.3 显示关系表中的数据

DataCombo 控件和 DataList 控件具有与众不同的特性。它们可以通过自身提供的 BoundColumn 属性和 BoundText 属性很方便地显示和查询关系表中数据。

**例 22.3** 在图书管理数据库中, 图书类别名称存储在一个表中, 每个类别都有一个唯一的标识, 即类别号; 另一个显示图书信息的表则使用类别号来表明是哪类图书。图书信息存储在一个表中, 每本图书都有一个唯一的标识, 即编号; 另一个显示图书目录的表则使用编号来表明是哪本书的目录。它们的关系如图 22.5 所示。(实例位置: 光盘\TM\sl\22\3)

由图 22.5 可以得出库存表、目录表和图书类别表之间的关系, 这时可以使用 DataCombo 控件显示图书类别名称, 而不可见地将图书类别的唯一标识(类别号)提供给“库存表”; 使用 DataList 控件显示图书信息, 而不可见地将图书的唯一标识(编号)提供给“目录表”; 使用 DataGrid 控件显示最终的图书目录信息。结果如图 22.6 所示。

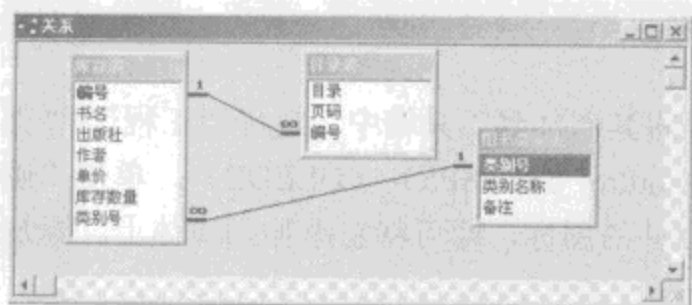


图 22.5 库存表、目录表和图书类别

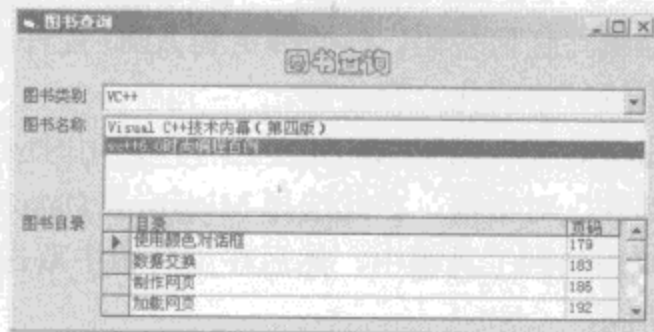


图 22.6 显示关系表中的数据

程序具体实现步骤如下。

- (1) 新建一个工程，在窗体上添加 3 个 ADO 控件，默认名为 Adodc1、Adodc2 和 Adodc3，将它们都连接到 db\_books.mdb 数据库，然后分别连接图书类别表、库存表和目录表。
- (2) 在窗体上添加一个 DataCombo 控件、一个 DataList 控件和一个 DataGrid 控件。
- (3) 设置 DataCombo1 控件的 RowSource 属性为 Adodc1，ListField 属性为“类别名称”，BoundColumn 属性为“类别号”。
- (4) 设置 DataList1 控件的 RowSource 属性为 Adodc2，ListField 属性为“书名”，BoundColumn 属性为“编号”。
- (5) 设置 DataGrid1 控件的 DataSource 属性为 Adodc3。
- (6) 切换到“代码窗口”，编写如下代码。

```
Private Sub DataCombo1_Click(Area As Integer)
    Adodc2.RecordSource = "select * from 库存表 where 类别号=" + DataCombo1.BoundText + ""
    Adodc2.Refresh
End Sub
Private Sub DataList1_Click()
    Adodc3.RecordSource = "select * from 目录表 where 编号=" + DataList1.BoundText + " order by 页码"
    Adodc3.Refresh
End Sub
```

说明：虽然在 DataCombo1 和 DataList1 控件中没有“类别号”和“编号”，但可以利用它们的 BoundColumn 属性和 BoundText 属性实现查询并显示查询结果，这是 DataCombo 和 DataList 控件固有的特性。

## 22.3 DataGrid 控件

教学录像：光盘\TM\lx\22\DataGrid 控件

虽然 ADO 控件具有存取数据库数据的能力，但却没有提供显示数据的功能。如果要显示数据库中的内容，可以使用绑定其他控件的方法。例如，绑定 TextBox 控件、DataGrid 控件、MSHFlexGrid 控件等。其中，若显示表格数据，使用 DataGrid 控件比较简便。

### 22.3.1 认识 DataGrid 控件

在 VB 6.0 中的众多数据控件中，DataGrid 控件是最灵活、功能最强大的控件之一。使用 DataGrid

控件无须编写任何代码，只要绑定到 ADO 控件上，就可以实现数据的新增、修改、删除和浏览，还可以对数据进行格式化、锁定等。

DataGrid 控件属于 ActiveX 控件，使用时应首先将其添加到工具箱中。选择“工程”/“部件”，打开“部件”对话框，在此选择 Microsoft DataGrid Control 6.0 (SP5) (OLEDB)，单击“确定”按钮，将其添加到工具箱中，然后在工具箱中双击该控件的图标，即可将它添加到窗体上。添加过程如图 22.7 所示。

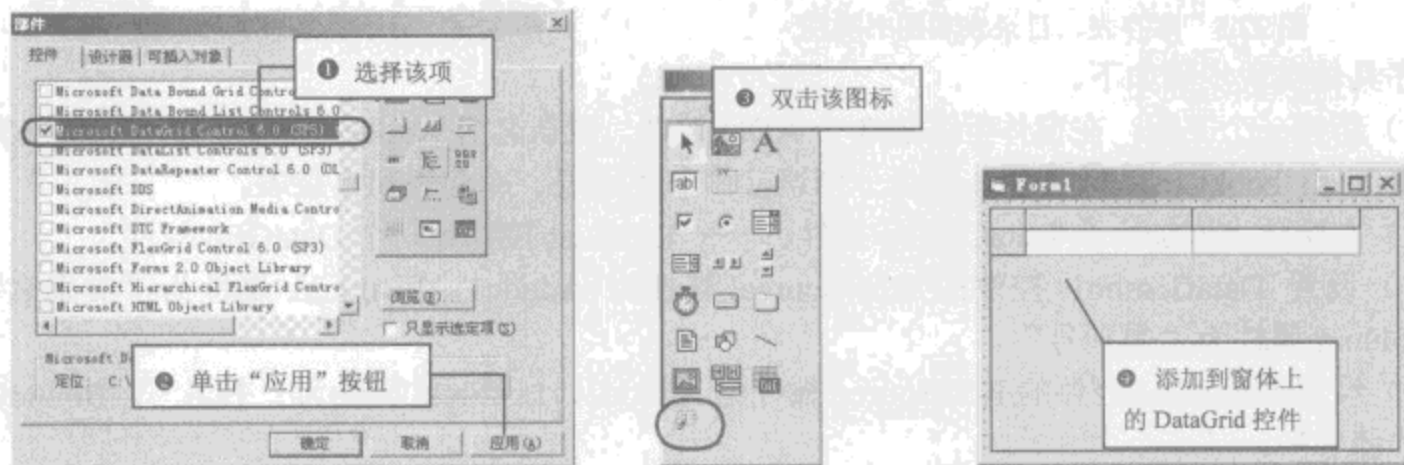


图 22.7 DataGrid 控件的添加过程

### 22.3.2 用 DataGrid 控件显示数据

用 DataGrid 控件显示数据，主要使用 DataSource 属性。它是 DataGrid 控件的主要属性，决定 DataGrid 控件指向的数据库。通常将它设置为 ADO 控件，通过这个 ADO 控件连接到数据库上。

**例 22.4** 下面使用 DataGrid 控件显示库存图书信息。(实例位置：光盘\TM\sl\22\4)

(1) 新建一个工程，在窗体中添加一个 ADO 控件，按照 21 章介绍的方法，使其连接数据库 db\_books.mdb 和数据表“库存表”。

(2) 在窗体上添加一个 DataGrid 控件，单击该控件，在“属性”窗口中找到 DataSource 属性，在其旁边的下拉列表框中选择“Adodc1”，如图 22.8 所示。

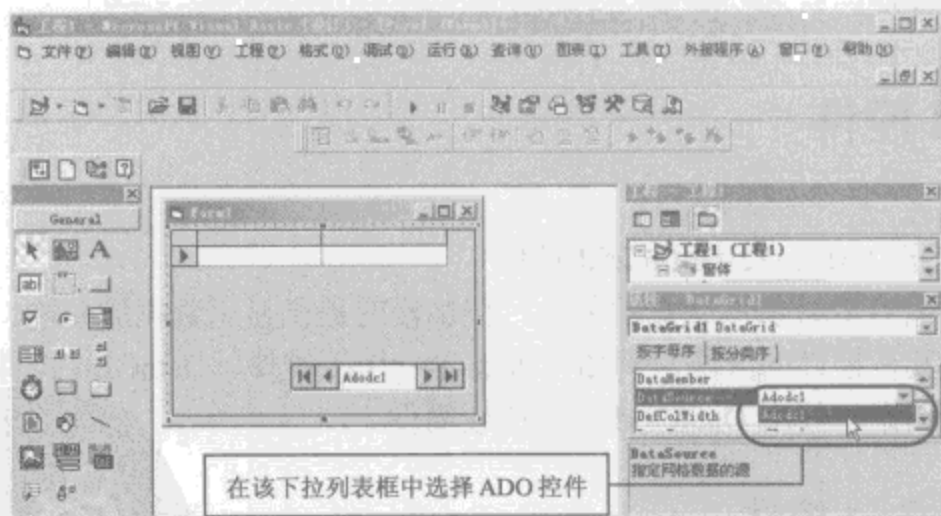


图 22.8 设置 DataGrid 控件的 DataSource 属性

(3) 右击 DataGrid 控件，在弹出的快捷菜单中选择“检索字段”，此时将检索库存表中的所有字段，如图 22.9 所示。

(4) 在“属性页”对话框中，如果将“允许添加”、“允许删除”和“允许更新”复选框中的“√”符号去掉，如图 22.10 所示，则 DataGrid 控件将不允许用户添加、删除和修改数据；反之则允许用户添加、删除和修改数据。

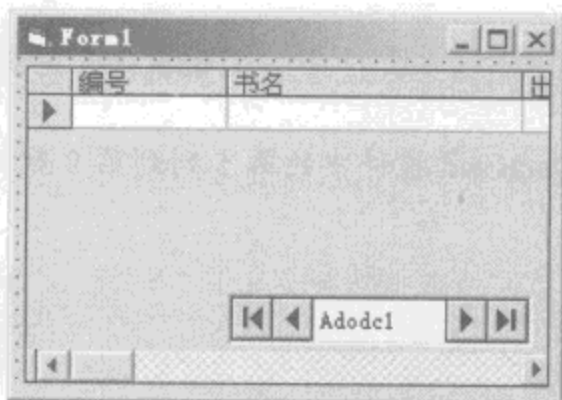


图 22.9 检索字段

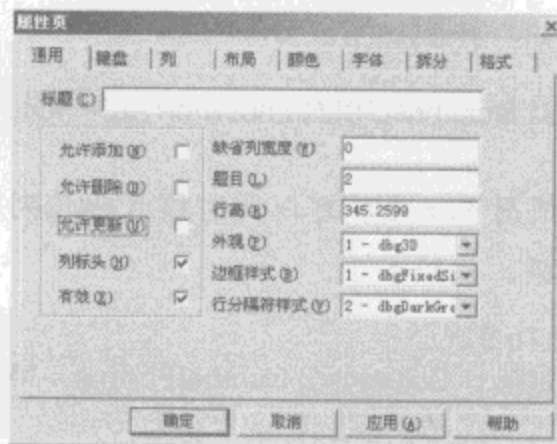


图 22.10 “属性页”对话框

技巧：也可以使用 DataGrid 控件的 AllowAddNew、AllowDelete 和 AllowUpdate 属性设置 DataGrid 控件允许或不允许用户添加、删除和修改数据，代码如下。

```
DataGrid1.AllowAddNew = True      '允许添加
DataGrid1.AllowDelete = True      '允许删除
DataGrid1.AllowUpdate = True      '允许更新
```

(5) 按〈F5〉键，运行程序，DataGrid 控件将显示库存图书信息，如图 22.11 所示。

编号	书名	出版社	作者	单价	库存数量	类别号
1	专家门诊——Visual C++开发	人民邮电出版社	肖宏伟	38	10	07
2	Visual Basic数据库开发关键技术	人民邮电出版社	高春艳等	39	9	01
3	Visual C++技术内幕(第四版)	清华大学出版社	潘爱民	96	14	03
4	JSP程序设计实例教程	清华大学出版社	姜晓铭 陈	38	20	06
5	网站开发新动力HTML JavaScript	北京希望电子出版	金坚信等	39	10	06
6	网页设计梦幻作坊	机械工业出版社	苏国彬	36	23	07
7	网页设计精彩实例	电子工业出版社	彭万波 张	42	12	07
8	完全参考(上)应用开发篇Dre	清华大学出版社	李明	37	9	07
9	实用参考手册SQL(第四版)	清华大学出版社	Judith S.	46	3	13
10	Visual C++6.0时尚编程百例	机械工业出版社	网冠科技	33	10	03

图 22.11 用 DataGrid 控件显示库存图书信息

说明：当使用 ADO 控件查询数据时，如果该控件绑定了 DataGrid 控件，那么 DataGrid 控件中的数据将自动更新。

### 22.3.3 格式化数据

从图 22.11 可以看出，通过 DataGrid 控件显示出的金额没有被格式化，使其难以同“库存”进行区分。下面将使用 Column 对象的 NumberFormat 属性对金额进行格式化。




例 22.5 下面使用 NumberFormat 属性将库存图书信息中的“单价”格式化为金额，代码如下。

```
Private Sub Form_Load()  
    DataGrid1.Columns("单价").NumberFormat = "0.00" '格式化“单价”列  
End Sub
```

例 22.6 用 NumberFormat 属性还可以格式化日期，例如将 DataGrid 控件中的第 10 列中的日期格式化为长日期，代码如下。

```
DataGrid1.Columns(10).NumberFormat = "long date" '格式化第 10 列
```

 技巧：使用 For 循环可以同时格式化几列，例如将 DataGrid 控件中的第 5 列到第 9 列格式化为金额，代码如下。

```
For i = 5 To 9  
    DataGrid1.Columns(i).NumberFormat = "0.00"  
Next i
```

以上设置，也可以通过“属性页”对话框中的“格式”选项卡设置完成，如图 22.12 所示。

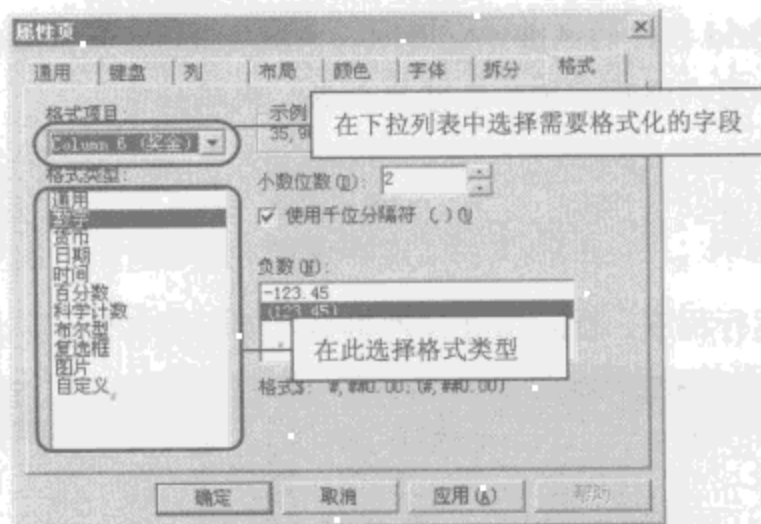


图 22.12 “格式”选项卡

#### 22.3.4 锁定数据

Column 对象的 Locked 属性，可以将表格中的数据锁定。

例 22.7 将 DataGrid 控件 3 列到 5 列中的数据锁定，代码如下。

```
For i = 3 To 5  
    DataGrid1.Columns(i).Locked = True  
Next i
```

#### 22.3.5 将 DataGrid 控件中的数据显示在文本框中

为了方便用户更详细地浏览和修改数据，可以将 DataGrid 控件中的数据显示在文本框中，效果如

图 22.13 所示。这主要通过 Column 对象的 Text 属性完成。

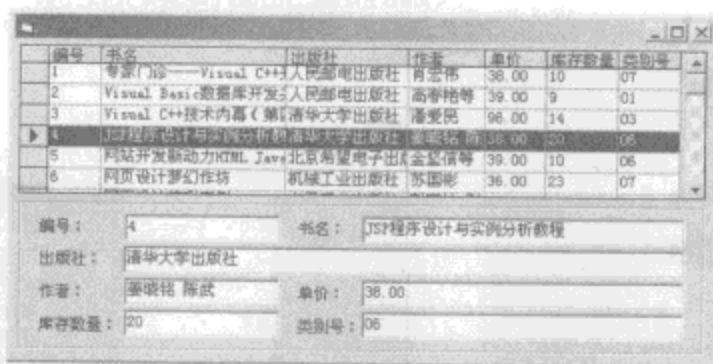


图 22.13 将 DataGrid 控件中的数据显示在文本框中

**例 22.8** 将 DataGrid 控件中的“书名”列中的数据显示在文本框 Text1 中，可以使用如下语句。

```
Text1.Text = DataGrid1.Columns("书名").Text
```

**例 22.9** 若选择某行时，将 DataGrid 控件中每行中各单元格中的数据显示对应的文本框中，则可以利用控件数组和循环语句，代码如下。（实例位置：光盘\TM\sl\22\5）

```
Dim i As Integer                                '定义一个整型变量
Private Sub Form_Load()
    Call DataGrid1_RowColChange>LastRow, LastCol)    '调用 DataGrid1_RowColChange 过程
End Sub
Private Sub DataGrid1_RowColChange>LastRow As Variant, ByVal LastCol As Integer)
    For i = 0 To 6
        Text1(i).Text = DataGrid1.Columns(i).Text    '将 DataGrid 控件中的数据赋值给文本框
    Next i
End Sub
```

## 22.4 MSFlexGrid 和 MSHFlexGrid 控件

 **教学录像：**光盘\TM\lx\22\MSFlexGrid 和 MSHFlexGrid 控件

MSFlexGrid 和 MSHFlexGrid 控件都用于以表格形式显示数据库中的数据，并可以操作数据。由于 MSHFlexGrid 控件是在 MSFlexGrid 控件的基础上发展而来的，所以 MSHFlexGrid 控件比 MSFlexGrid 控件的功能更强大、使用更灵活。

两个控件都提供了高度灵活的排序、合并和格式设置功能。MSFlexGrid 控件绑定到 Data 控件上，数据是只读的；MSHFlexGrid 控件绑定到 ADO 控件、ADO 对象或数据环境，数据也是只读的，更主要的一个特性是 MSHFlexGrid 控件与数据环境绑定在一起能够显示关系层次结构记录集。下面重点介绍 MSHFlexGrid 控件。

### 22.4.1 认识 MSHFlexGrid 控件

MSHFlexGrid 控件属于 ActiveX 控件，使用时应首先将其添加到工具箱中。选择“工程”/“部件”，打开“部件”对话框，在此选择 Microsoft Hierarchical FlexGrid Control 6.0 (SP4) (OLE DB)，单击

“确定”按钮，将其添加到工具箱中，然后在工具箱中双击该控件的图标，即可将它添加到窗体上。添加过程如图 22.14 所示。

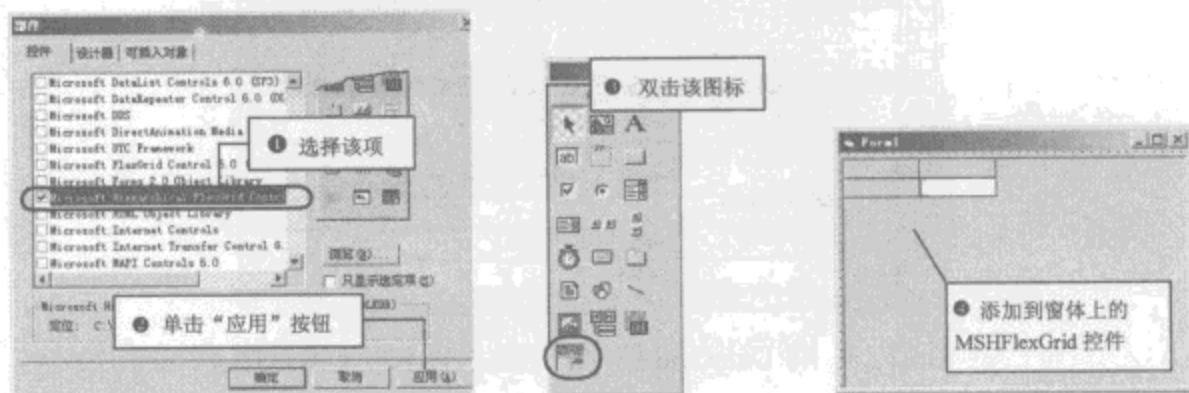


图 22.14 MSHFlexGrid 控件的添加过程

## 22.4.2 用 MSHFlexGrid 控件显示数据

MSHFlexGrid 控件同 ADO 控件、数据环境和 ADO 对象进行绑定，来显示数据。如果用户使用 ADO 控件作为数据源，只设置 DataSource 属性为 ADO 控件（如 Adodc1）即可；如果用户使用数据环境作为数据源，除了设置 DataSource 属性为数据环境（如 DataEnvironment1）外，还要设置 DataMember 属性为 Command 对象（如 Command1）；如果使用 ADO 对象作为数据源，应首先引用 ADO 对象，然后将 Recordset 对象（如 rs1）赋值给 DataSource 属性。

**例 22.10** 下面以 ADO 控件为例介绍使用 MSHFlexGrid 控件显示库存图书信息。（实例位置：光盘\TM\sl\22\6）

- (1) 新建一个工程，将 ADO 控件和 MSHFlexGrid 控件添加到工具箱中。
- (2) 在窗体上添加一个 ADO 控件，使其连接数据库 db\_books.mdb 和数据表“库存表”。
- (3) 在窗体上添加一个 MSHFlexGrid 控件，设置 DataSource 属性为 Adodc1。
- (4) 检索结构，显示数据字段。

MSHFlexGrid 控件不能在其单元格中自动显示数据字段，因此需要通过右击 MSHFlexGrid 控件，在弹出的快捷菜单中选择“检索结构”，以实现显示数据字段。

- (5) 调整 MSHFlexGrid 控件的“外观”。

右击 MSHFlexGrid 控件，在“属性页”对话框中单击“通用”选项卡，如图 22.15 所示。

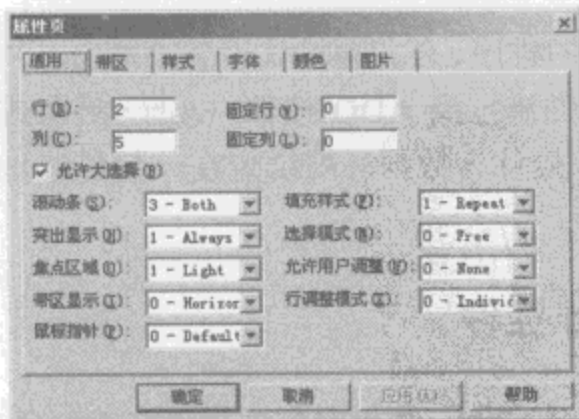


图 22.15 “通用”选项卡

在“通用”选项卡中可以设置 MSFlexGrid 控件的如下属性。

行、列：对应 MSFlexGrid 控件的 Rows 属性和 Cols 属性，主要设置控件的行数和列数。

固定行、固定列：决定 MSFlexGrid 控件最上面有多少固定行和最左边有多少固定列。固定行上可以自动显示字段的名称。

突出显示：决定选定的单元格是否在 MSFlexGrid 中突出显示。有以下 3 种选择。

- ① 0-FlexHighlightNever：表明选定的单元格上没有突出显示。
- ② 1-FlexHighlightAlways：默认值，表明选定的单元格突出显示。
- ③ 2-FlexHighlightWithFocus：表明突出显示只在控件有焦点时有效。

网格线：决定在 MSFlexGrid 中绘制网格线的类型。有以下 4 种选择。

- ① 0-FlexGridNone：在单元格之间没有线。
- ② 1-FlexGridFlat：默认值，表明单元格之间的线样式被设置为正常的、平面的线。
- ③ 2-FlexGridInset：单元格之间的线样式为凹入线。
- ④ 3-FlexGridRaised：单元格之间的线样式为凸起线。

文本样式：决定本带区中文本显示的风格。有以下 5 种选择。

- ① 0-FlexTextflat：默认值，文本正常显示，平面文本。
- ② 1-FlexTextRaised：文本看起来凸起。
- ③ 2-FlexTextInset：文本看起来凹入。
- ④ 3-FlexTextRaisedLight：文本看起来轻微凸起。
- ⑤ 4-FlexTextInsetLight：文本看起来轻微凹入。

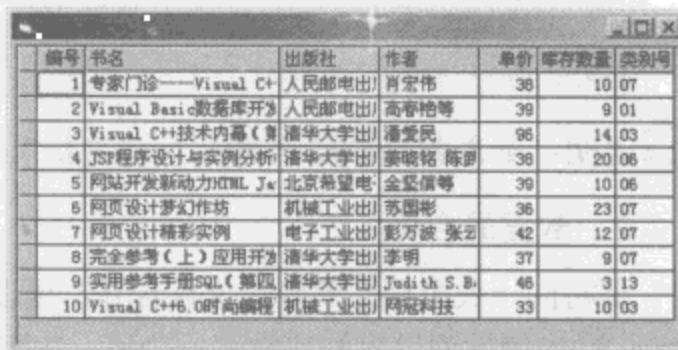
文本样式表头：决定本带区的列标头文本显示的风格，与“文本样式”相同。

(6) 按〈F5〉键，运行程序，结果如图 22.16 所示。

在调整列宽上，MSFlexGrid 控件没有 DataGrid 控件灵活，DataGrid 控件在设计时就可以调整列宽，而 MSFlexGrid 控件只能通过设置 AllowUserResizing 属性（运行时用鼠标调整）或 ColWidth 属性（编写代码调整）调整列宽。下面使用 ColWidth 属性调整 MSFlexGrid 控件的列宽（效果如图 22.17 所示），代码如下。



编号	书名	出版社	作者	单价	库存数量	类别号
1	专家门诊——Visual C++	人民邮电	肖宏伟	38	10	07
2	Visual Basic 数据库开发	人民邮电	高春艳等	39	9	01
3	Visual C++ 技术内幕	清华大学	潘爱民	96	14	03
4	JSP 程序设计	清华大学	姜晓铭 陈静	38	20	06
5	网站开发新动力 HTML、Java	北京希望电	金坚信等	39	10	06
6	网页设计梦幻作坊	机械工业	苏国彬	36	23	07
7	网页设计精彩实例	电子工业	彭万波 张云	42	12	07
8	完全参考（上）应用开发	清华大学	李明	37	9	07
9	实用参考手册 SQL（第四版）	清华大学	Judith S. B.	46	3	13
10	Visual C++ 6.0 时尚编程	机械工业	网冠科技	33	10	03



编号	书名	出版社	作者	单价	库存数量	类别号
1	专家门诊——Visual C++	人民邮电	肖宏伟	38	10	07
2	Visual Basic 数据库开发	人民邮电	高春艳等	39	9	01
3	Visual C++ 技术内幕	清华大学	潘爱民	96	14	03
4	JSP 程序设计	清华大学	姜晓铭 陈静	38	20	06
5	网站开发新动力 HTML、Java	北京希望电	金坚信等	39	10	06
6	网页设计梦幻作坊	机械工业	苏国彬	36	23	07
7	网页设计精彩实例	电子工业	彭万波 张云	42	12	07
8	完全参考（上）应用开发	清华大学	李明	37	9	07
9	实用参考手册 SQL（第四版）	清华大学	Judith S. B.	46	3	13
10	Visual C++ 6.0 时尚编程	机械工业	网冠科技	33	10	03

图 22.16 用 MSFlexGrid 控件显示数据（设置列宽前） 图 22.17 用 MSFlexGrid 控件显示数据（设置列宽后）

```
Private Sub Form_Load()
    With MSFlexGrid1
        .ColWidth(0) = 200: .ColWidth(1) = 500: .ColWidth(2) = 2000
        .ColWidth(5) = 600: .ColWidth(6) = 800: .ColWidth(7) = 600
    End With
End Sub
```



### 22.4.3 数据排序与合并

#### 1. 数据的排序

MSHFlexGrid 控件的 Sort 属性可以对 MSHFlexGrid 表格中的数据进行多种排序操作。下面介绍 Sort 属性, 如表 22.1 所示。

表 22.1 Sort 属性值及其功能

值	常 数	功 能
0	flexsortnone	无, 不执行排序
1	flexsortgenericAscending	一般升序。执行估计文本不管是字符串或者是数字的升序排序
2	FlexsortGenericDescending	一般降序。执行估计文本不管是字符串或者是数字的降序排序
3	FlexsortNumericAscending	数值升序。执行将字符串转换为数值的升序排序
4	FlexsortNumericDesending	数值降序。执行将字符串转换为数值的降序排序
5	FlexsortstringnocaseAsending	字符串升序。执行不区分字符串大小写比较的升序排序
6	FlexsortstringNocaseAsending	字符串降序。执行不区分字符串大小写比较的降序排序
7	FlexsortnocaseDescending	字符串升序。执行区分字符串大小写比较的升序排序
8	FlexsortstringDescending	字符串降序。执行区分字符串大小写比较的降序排序
9	Flexsortcustom	自定义。使用 Compare 事件比较行

例 22.11 按“出版社”升序排序例 22.10 中的库存图书信息, 代码如下。

```
Private Sub Form_Load()
    With MSHFlexGrid1
        .Col = 3                '排序第 3 列
        .Sort = 1              '按一般升序排序
    End With
End Sub
```

 说明: 如果排序数值数据, 如“单价”, 可以设置 Sort 属性值为 3 或 4。

#### 2. 数据的合并

要将同一表中相同的数据进行合并, 可以使用 MergeCol 属性和 MergeRow 属性。MergeCol 属性和 MergeRow 属性通过返回或设置一个值, 决定哪些行和列可以把它们的内容合并。

例 22.12 将例 22.10 中, 相同出版社合并在一起, 效果如图 22.18 所示。(实例位置: 光盘 \TM\sl\22\7)

要将所有相同的出版社都合并在一起, 应首先按“出版社”进行排序, 然后再合并, 代码如下。

```
Private Sub Form_Load()
    With MSHFlexGrid1
        .Col = 3                '排序第 3 列
```

```

.Sort = 1           '按一般升序排序
.MergeCells = 1     '自由合并
.MergeCol(3) = True '合并第 3 列
End With
End Sub

```

编号	书名	出版社	作者	单价	库存数量	类别号
5	网站开发新动力HTML、JSP	北京希望电	金坚信等	39	10	06
7	网页设计精彩实例	电子工业出	彭万波 张云	42	12	07
6	网页设计梦幻作坊	机械工业出	苏国彬	36	23	07
10	Visual C++6.0时尚编程	网冠科技	33	10	03	
9	实用参考手册SQL(第四版)	Judith S. B.	46	3	13	
3	Visual C++技术内幕(第2版)	潘爱民	96	14	03	
4	JSP程序设计与实例分析	清华大学出	姜晓铭 陈武	38	20	06
8	完全参考(上)应用开发	李明	37	9	07	
2	Visual Basic数据库开发	高孝松等	39	9	01	
1	专家门诊——Visual C++	人民邮电出	肖宏伟	38	10	07

图 22.18 合并出版社

#### 22.4.4 隐藏行或列

隐藏 MSFlexGrid 其中的某些行或列，方法非常简单，只需设置某行的行高或某列的列宽为 0 就可以了。

例 22.13 隐藏 MSFlexGrid 表格的第 0 行和第 5 列，代码如下。

```

MSFlexGrid1.RowHeight(0) = 0   '隐藏第 0 行
MSFlexGrid1.ColWidth(5) = 0    '隐藏第 5 列

```

#### 22.4.5 冻结字段

所谓冻结的字段指的是不会随着滚动条一起移动的字段（也就是固定字段），如图 22.19 所示，其中灰色背景的字段就是被冻结的字段。

编号	书名	作者	单价	库存数量	类别
5	网站开发新动力HTML、JSP	金坚信等	39	10	06
7	网页设计精彩实例	彭万波 张云	42	12	07
6	网页设计梦幻作坊	苏国彬	36	23	07
10	Visual C++6.0时尚编程	网冠科技	33	10	03
9	实用参考手册SQL(第四版)	Judith S. B.	46	3	13
3	Visual C++技术内幕(第2版)	潘爱民	96	14	03
4	JSP程序设计与实例分析	姜晓铭 陈武	38	20	06

图 22.19 冻结“编号”和“书名”

例 22.14 冻结字段需要使用 FixedCols 属性，下面冻结“编号”和“书名”，代码如下。（实例位置：光盘\TM\sl\22\8）

```

Private Sub Form_Load()
    MSFlexGrid1.FixedCols = 3   '冻结前 3 列
End Sub

```

上面的代码将前 3 列设置为冻结字段，如果要解除冻结字段，只需将 FixedCols 属性值设置为 0。

## 22.5 小结

本章介绍了常用数据库控件的基本应用，重点应掌握 DataGrid 控件和 MSHFlexGrid 控件。这两个控件都可以显示表格数据，MSHFlexGrid 控件通用性比 DataGrid 控件高些，但它不具备像 DataGrid 控件一样的新增、修改和删除数据的功能。另外，用 MSHFlexGrid 控件显示过多的数据（上万条），就会浪费内存和时间，而 DataGrid 控件相对会好些，因为 DataGrid 控件是需要显示数据时才向 ADO 控件读取必要的数据库，所以速度快又不会浪费太多的内存。

## 22.6 练习与实践

1. 用 DataCombo 控件显示“房间号”，然后按房间号查询并显示房间详细信息。（答案位置：光盘\TM\sl\22\9）

2. 使用 DataGrid 控件显示商品信息，然后将“单据号”和“单价”进行格式化，将“单据号”格式化为“X-0000”，将“单价”格式化为“0.00”，如下图所示。（答案位置：光盘\TM\sl\22\10）

单据号	商品编号	商品名称	型号	规格	产地	单位	数量	单价	日期
X-0001	T1006	复印纸	A4	1*8	江苏	箱	4	450.00	2005-05-17
X-0002	T1007	打印纸	16K	1*8	江苏	箱	50	120.00	2005-03-25
X-0004	T1006	复印纸	A3	1*8	江苏	箱	5	130.00	2005-05-17
X-0005	T1008	复印纸	16K	1*8	江苏	箱	90	115.00	2005-05-23
X-0003	T1007	打印纸	16K	1*8	江苏	箱	50	120.00	2005-03-25
X-0006	T1005	复印纸	16K	1*8	江苏	箱	5	140.00	2005-03-15

3. 用 MSHFlexGrid 控件显示数据，并设置奇数行背景色为“黄色”，偶数行背景色为“绿色”。（答案位置：光盘\TM\sl\22\11）

4. 用 MSHFlexGrid 控件显示数据，并根据用户选择的字段和排序方式进行排序。（答案位置：光盘\TM\sl\22\12）

# 第23章

## 网络编程技术

(教学录像: 28 分钟)

随着计算机软、硬件技术的不断提高,近10年来,计算机网络得到了长足的发展。尤其是Internet(互联网)的兴起,使计算机网络技术发展到了一个新的里程碑。Internet和WWW已被人们所熟知,互联网技术发展日新月异,新技术、新方法层出不穷。相应的,网络应用程序开发变得越来越复杂。在本章的讲解过程中,为了更加方便读者理解,笔者结合了相关实例来介绍网络编程的基本知识及思路与过程。

通过阅读本章,您可以:

- » 了解 OSI 参考模型
- » 了解 HTTP 协议、FTP 协议
- » 了解 TCP 基础、UDP 基础
- » 掌握 Winsock 控件的使用
- » 掌握 Internet Transfer 控件的使用
- » 掌握 WebBrowser 控件的使用
- » 学会开发简单的聊天程序
- » 学会设计文件上传与下载程序
- » 学会制作自己的浏览器





## 23.1 网络基础知识

 教学录像：光盘\TM\lx\23\网络基础知识.exe

在 VB 中应用网络技术进行编程，首先要了解一些有关 Internet 的基础知识，如应用层的有关协议、网络层次模型等。本节将主要介绍 OSI 参考模型以及 HTTP、FTP 和 IP 协议这几个方面的内容。

### 23.1.1 OSI 参考模型

OSI 参考模型（Open System Interconnection，开发系统互联模型）是一个将不同机种的计算机系统联合起来，使其可以进行相互通信的规范。OSI 采用分层的构造技术，它由 7 层组成，每一层为上一层提供服务。其中 OSI 网络参考模型如图 23.1 所示。

OSI 参考模型层次	OSI 参考模型名称
第 7 层	应用层 (Application)
第 6 层	表示层 (Presentation)
第 5 层	会话层 (Session)
第 4 层	传输层 (Transport)
第 3 层	网络层 (Network)
第 2 层	数据链路层 (Data Link)
第 1 层	物理层 (Physical)

图 23.1 OSI 网络参考模型

### 23.1.2 HTTP 协议

HTTP 协议（Hypertext Transfer Protocol，超文本传输协议），用于从 WWW 服务器传输超文本到本地浏览器的传送协议。它可以使浏览器更加高效，使网络传输减少。它不仅保证正确的传输超文本文档，还确定传输文档中哪一部分，以及哪部分内容首先显示。

### 23.1.3 FTP 协议

FTP 协议（File Transfer Protocol，文件传输协议），允许用户将某个系统中的文件复制到另一个系统中。在设置 FTP 服务器属性的目录安全性时，如果把 FTP 站点设置为“默认情况下，所有计算机将被授权访问”选项，则除了加入列表的 IP 地址以外，来自其他 IP 地址的客户端都被允许访问；如果把 FTP 站点设置为“默认情况下，所有计算机将被拒绝访问”选项，则除了加入列表的 IP 地址以外，来自其他 IP 地址的客户端都被拒绝访问。还要补充一点说明，如果客户端是通过代理服务器的方式来访问 FTP 服务器，那么 IIS 接收并进行处理的是代理服务器的 IP 地址，而不是用户计算机的 IP 地址。

由此可以看出, 登录到某些网站上时可以使用匿名账号。

## 23.2 Winsock 控件编程

 教学录像: 光盘\TM\lx\23\Winsock 控件编程.exe

Winsock (Windows Socket) 是 Microsoft 为 Win32 环境下的网络编程提供的接口, 这些接口是以 API 的形式出现的。Winsock 控件的工作原理为: 服务器不停地监听检测客户端的请求, 客户端则向服务器端发出连接请求, 当两者的协议沟通时, 客户端与服务器端就建立起了连接。此时, 客户端继续请求服务器端发送或接收数据, 服务器则在等待客户端的这些请求。


### 23.2.1 TCP 与 UDP 基础

TCP (Transmission Control Protocol, 传输控制协议), 允许创建和维护与远程计算机的连接。在连接之后, 两台计算机之间就可以把数据当作一个双向字节流进行交换。数据传输完成后, 还要关闭连接。

UDP (User Datagram Protocol, 用户数据报协议), 是一个面向无连接的协议。采用该协议, 计算机并不需要建立连接。它为应用程序提供一次性的数据传输服务。UDP 协议不提供差错恢复, 不能提供数据重传, 因此该协议传输数据安全性略差。

### 23.2.2 Winsock 控件

Winsock 控件提供了访问 TCP 和 UDP 网络服务的方便途径。当利用它编写网络程序时, 不必了解 TCP 等协议的细节或调用低级的 Winsock API 函数, 只需通过设置控件的属性并调用其方法就可以轻松连接到一台远程机器上, 并实现网络连接进行信息交换。

如果要在 VB 工程中使用 Winsock 控件, 则应在工程中选择“工程”/“部件”命令, 打开“部件”对话框, 在该对话框中选中 Microsoft Winsock Control 6.0 (SP5) 复选框, 单击“确定”按钮即可将  图标添加到工具箱中。“部件”对话框如图 23.2 所示。

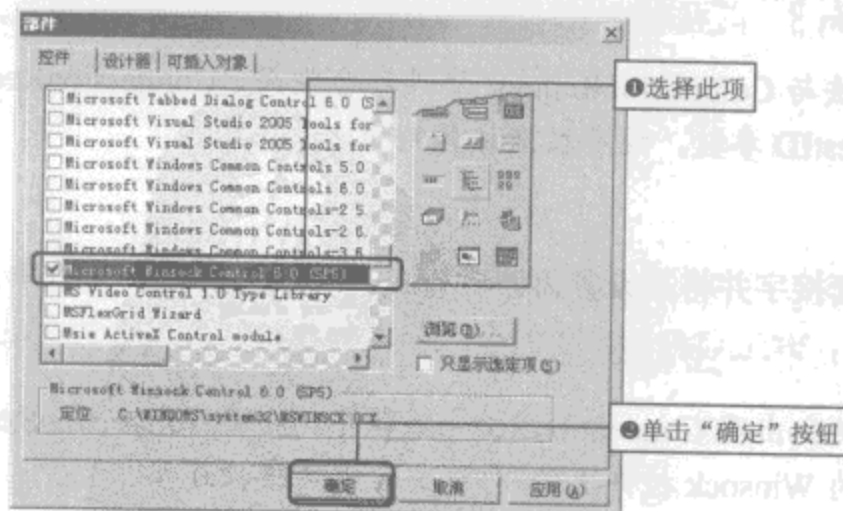


图 23.2 “部件”对话框

下面主要介绍 Winsock 控件的常用属性、方法和事件。

### 1. LocalPort 属性

LocalPort 属性用于返回或设置所用到的本地端口。对客户来说，该属性指定发送数据的本地端口，如果应用程序不需要特定端口，则指定 0 为端口号；对于服务器来说，用于监听的本地端口，如果指定的是端口 0，就使用一个随机端口。在调用了 Listen 方法后，该属性就包含了已选定的实际端口。

### 2. RemotePort 属性

RemotePort 属性用于返回或设置所连接应用程序的远程端口号。

### 3. State 属性

State 属性用于返回控件的状态，其取值用表 23.1 所示的枚举类型来表示。

表 23.1 State 属性的设置值

常 数	值	说 明
sckClosed	0	默认值，关闭
sckOpen	1	打开
sckListening	2	监听
sckConnectionPending	3	连接挂起
sckResolvingHost	4	识别主机
sckHostResolved	5	已识别主机
sckConnecting	6	正在连接
sckConnected	7	已连接
sckClosing	8	同级人员正在关闭连接
sckError	9	错误

注意：该属性在设计时是只读的。

### 4. Accept 方法

Accept 方法用于接收新的连接。该方法仅适用于 TCP 服务器应用程序。其语法格式如下：

```
object.Accept RequestID
```

注意：一般 Accept 方法与 ConnectionRequest 事件配合使用。ConnectionRequest 事件有一个对应的参数，即 RequestID 参数，该参数应该传给 Accept 方法。

### 5. Listen 方法

Listen 方法用于创建套接字并将其设置为监听模式。该方法仅适用于 TCP 连接。其语法格式如下：

```
object.Listen
```

说明：当有新连接时就会出现 ConnectionRequest 事件。处理 ConnectionRequest 事件时，应用程序应该在一个新的 Winsock 控件上使用 Accept 方法接收连接。

## 6. SendData 方法

SendData 方法用于将数据发送给远程计算机。其语法格式如下：

```
object.SendData data
```

其中 data 是要发送的数据。对于二进制数据应使用字节数组。当 UniCode 字符串用 SendData 方法在网络上发送之前，将被转化成 ANSI 字符串。

## 7. GetData 方法

GetData 方法用于获取当前的数据块并将其存储在 Variant 变体类型的变量中。当本地计算机接收到远程计算机的数据时，数据存放在接收缓存中。要从接收缓存中取得数据，可以使用 GetData 方法。其语法格式如下，参数说明如表 23.2 所示。


```
object.GetData data, [type,] [maxLen]
```

表 23.2 GetData 方法的参数说明

参 数	描 述
object	对象表达式
data	在 GetData 方法成功返回之后存储获取数据的地方。如果对请求的类型没有足够可用的数据，则将 data 设置为 Empty
type	可选参数。获取的数据类型，其设置值如表 23.3 所示
maxLen	可选参数。指定接收到的字节数组或字符串的大小

表 23.3 type 参数的设置值

参 数	类 型 说 明
VbByte	Byte
vbInteger	Integer
vbLong	Long
vbSingle	Single
vbDouble	Double
vbCurrency	Currency
vbDate	Date
vbBoolean	Boolean
vbError	SCODE
vbString	String
vbArray + vbByte	Byte Array

 说明：通常总是将 GetData 方法与 DataArrival 事件并用，而 DataArrival 事件包含 bytesTotal 参数。如果指定一个比 bytesTotal 参数小的 maxlen，则会有一个错误号为 10040 的错误提示信息，以提示用户剩余的字节将丢失。

## 8. ConnectionRequest 事件

当远程计算机发出请求连接时触发该事件，该事件仅适用于 TCP 服务器应用程序。在请求一个新



连接时 ConnectionRequest 事件被激活。激活之后, RemoteHostIP 和 RemotePort 属性将存储有关客户端计算机的 IP 地址和端口等信息。

例如, 在 Winsock 控件的 ConnectionRequest 事件中, 使用 State 属性判断该控件是否已关闭, 然后再通过 Accept 方法接受新的连接。其程序代码如下:

```
Private Sub Winsock1_ConnectionRequest(ByVal RequestID As Long)
    If Winsock1.State <> 0 Then Winsock1.Close
    Winsock1.Accept RequestID
End Sub
```

'用于远程计算机请求连接  
'关闭 Winsock 控件  
'表示客户请求连接的 ID 号

### 9. DataArrival 事件

当新数据到达时触发该事件, 通过该事件可以接收远程计算机发送过来的数据信息。

例如, 在 Winsock 控件的 DataArrival 事件中, 使用 GetData 方法获得远程计算机的数据信息并将其显示在文本框中。其程序代码如下:

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
    Dim strdata As String : Dim sdata As String
    Winsock1.GetData strdata
    sdata = Left$(strdata, 7)
    sendxx = Right$(strdata, Len(strdata) - 7)
    aa = "远程计算机 " + sendxx
    Txt_incept.Text = Txt_incept.Text & aa & vbCrLf
End Sub
```

'接收新数据信息  
'定义字符串变量  
'获得消息数据  
'接收到的信息显示在文本框中

## 23.2.3 开发客户端/服务器端聊天程序

**例 23.1** 开发一个客户端/服务器端聊天程序, 该程序主要利用服务器端的 Winsock 控件绑定一个端口进行监听, 当有来自客户端的请求发生时, 就建立一个新的连接, 之后就可以实现客户端和服务器的通信。(实例位置: 光盘\TM\sl\23\1)

### 1. 服务器端

在服务器端的应用程序中, 输入客户机的 IP 地址, 然后单击“设置服务器”按钮, 将其程序设置为服务器端运行程序。其运行效果如图 23.3 所示。

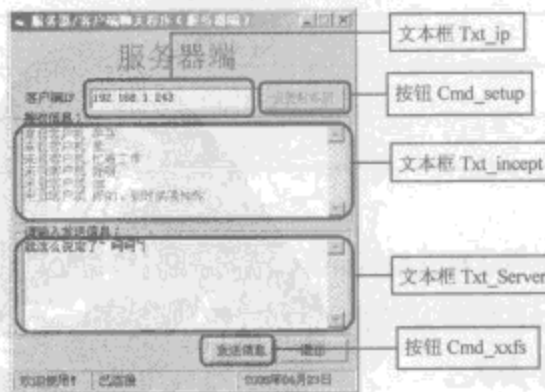


图 23.3 服务器端

下面是服务器端应用程序的关键代码。

```

Private Sub Cmd_setup_Click()
    Winsock1.LocalPort = 100
    Winsock1.RemotePort = 200
    Winsock1.Listen
End Sub
Private Sub Cmd_xxfS_Click()
    If Winsock1.State = 7 Then
        Dim BB, DD
        If Txt_Server.Text = "" Then
            MsgBox "不能发送空信息", , "系统提示"
        Else
            DD = Txt_Server.Text
            Winsock1.SendData "SENDINF" & DD
        End If
    Else
        MsgBox "没有连接, 请查正后再试", vbInformation, "错误"
    End If
End Sub
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
    Dim strdata As String : Dim sdata As String
    Winsock1.GetData strdata
    sdata = Left$(strdata, 7)
    Select Case sdata
        Case "SYSINFO"
            txtxx = Right$(strdata, Len(strdata) - 7)
        Case "SENDINF"
            Dim aa, temp1
            sendxx = Right$(strdata, Len(strdata) - 7)
            aa = "来自客户机 " + sendxx
            Txt_incept.Text = Txt_incept.Text & aa & vbCrLf
        Case "QUITMYF"
    End Select
End Sub

```

'设置服务器按钮  
'本地服务器端口号  
'客户端端口号  
'监听  
'“发送信息”按钮  
'发送消息  
'接收新数据信息  
'定义字符串变量  
'获得消息数据  
'系统消息  
'发送消息  
'关闭服务端

## 2. 客户端

在客户端的应用程序中, 输入服务器端程序的 IP 地址, 单击“网络连接”按钮, 使得客户端程序与服务器端程序取得连接, 这时才可以发送数据信息来进行聊天, 其运行效果如图 23.4 所示。

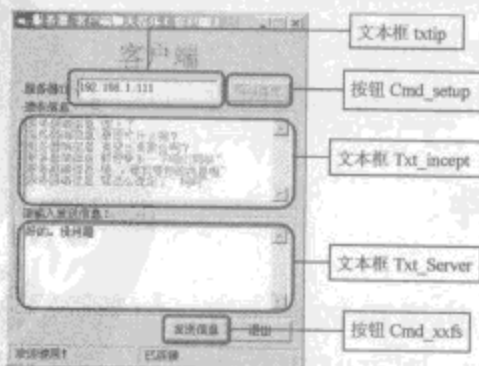


图 23.4 客户端

下面是客户端应用到的关键代码。

```

Private Sub Cmd_setup_Click()           '设置连接
    Winsock1.RemoteHost = txtip.Text    '设置远程计算机
    Winsock1.LocalPort = 200           '本地服务器端口号
    Winsock1.RemotePort = 100          '客户端端口号
    Winsock1.Connect                   '连接到远程计算机
End Sub
Private Sub Cmd_xxfs_Click()           '“发送信息”按钮
    If Winsock1.State = 7 Then
        Dim BB
        If Txt_Server.Text = "" Then
            MsgBox "不能发送空信息", , "系统提示"
        Else
            Winsock1.SendData "SENDINF" & Txt_Server.Text    '发送消息
        End If
    Else
        MsgBox "没有连接, 请查正后再试", vbInformation, "错误"
    End If
End Sub
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)    '接收新数据信息
    Dim sdata As String : Dim strdata As String : Dim mycommand As String    '定义字符串变量
    Winsock1.GetData sdata    '获得消息数据
    mycommand = Left$(sdata, 7)
    Select Case mycommand    '消息
        Case "SENDINF"
            Dim aa
            sendxx = Right$(sdata, Len(sdata) - 7)
            aa = "服务器端信息 " + sendxx
            Txt_incept.Text = Txt_incept.Text & aa & vbCrLf
        End Select
    End Sub

```


## 23.3 Internet Transfer 控件编程

 教学录像: 光盘\TM\lx\23\Internet Transfer 控件编程.exe

Internet Transfer 控件提供两种 Internet 协议, 即超文本传送协议 (HyperText Transfer Protocol, HTTP) 和文件传输协议 (File Transfer Protocol, FTP)。该控件主要用于数据传输, 使用 Internet Transfer 控件可以通过 OpenURL 或 Execute 方法连接到任何使用这两个协议的站点并检索文件。

### 23.3.1 Internet Transfer 控件

Internet Transfer 控件支持 HTTP 协议和 FTP 协议。如果使用 HTTP 协议, 可以从 WWW 服务器上获取 HTML 文档。如果使用 FTP 协议, 则可以到 FTP 服务器上上传或下载文件。在 VB 中使用 Internet

Transfer 控件时,首先需要选择菜单栏中的“工程”/“部件”命令,在弹出的对话框中选中“Microsoft Internet Transfer Control 6.0 (SP4)”复选框,单击“确定”按钮即可将图标添加到工具中。“部件”对话框如图 23.5 所示。

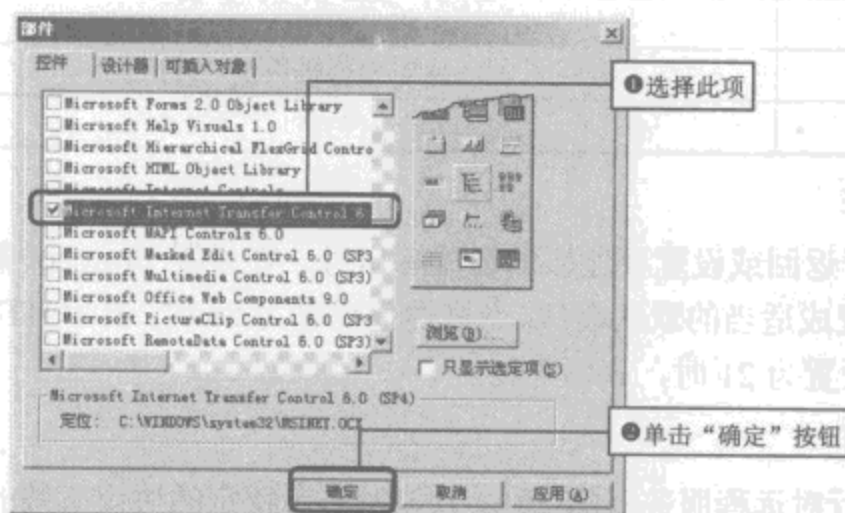


图 23.5 “部件”对话框

注意: Internet Transfer 控件的功能依赖于所使用的协议,协议不同所用的属性和方法不同,所能够进行的操作也不同。

下面主要介绍 Internet Transfer 控件的常用属性和方法。

#### 1. AccessType 属性

AccessType 属性用于设置或返回一个值,决定该控件用来与 Internet 进行通讯的访问类型(通过代理访问或直接访问)。正在处理异步请求时,该值可以改变,但直到创建了下一个连接时,改变才会生效。AccessType 属性的设置值如表 23.4 所示。

表 23.4 AccessType 属性值的设置

常 数	值	描 述
icUseDefault	0	使用默认值。控件使用在注册表中找到的默认设置值来访问 Internet
icDirect	1	直接连到 Internet
icNamedProxy	2	命名代理。指示控件使用 Proxy 属性中指定的代理服务器

#### 2. Protocol 属性

Protocol 属性用于设置或返回一个值,指定和 Execute 方法一起使用的协议。在编程中指定该属性后,URL 属性被更新以显示新值。另外,如果此 URL 的协议部分被更新,Protocol 属性也将被更新以体现新值。OpenURL 和 Execute 方法都可能会修改该属性值。直到调用下一个 Execute 或 OpenURL 方法时,该属性值的改变才会有效。其中,Protocol 属性的设置值如表 23.5 所示。

表 23.5 Protocol 属性值设置

常 数	值	描 述
icUnknown	0	未知的
icDefault	1	默认协议



续表

常 数	值	描 述
icFTP	2	文件传输协议 (FTP)
icReserved	3	为将来预留
icHTTP	4	超文本传输协议 (HTTP)
icHTTPS	5	安全 HTTP

### 3. RemotePort 属性

RemotePort 属性用于返回或设置要连接的远程端口号。在设置 Protocol 属性时, 将对每个协议自动把 RemotePort 属性设置成适当的默认端口。当属性值设置为 80 时, 表示 HTTP, 通常用于 Word Wide Web 的连接; 当属性值设置为 21 时, 表示 FTP。

### 4. Execute 方法

Execute 方法用于执行对远程服务器的请求。只能发送对特定的协议有效的请求。其语法格式如下, 参数说明如表 23.6 所示。

object.Execute url, operation, data, requestHeaders

表 23.6 Execute 方法的参数说明

参 数	描 述
object	对象表达式
url	可选项。字符串, 指定控件将要连接的 URL。如果这里未指定 URL, 将使用 URL 属性中指定的 URL
operation	可选项。字符串, 指定将要执行的操作类型。operation 的有效设置值由所用的协议决定
data	可选项。字符串, 指定用于操作的数据
requestHeaders	可选项。字符串, 指定由远程服务器传来的附加的标头

### 5. OpenURL 方法

OpenURL 方法用于打开并返回指定 URL 的文档。文档以变体型返回。该方法完成时, URL 的各种属性 (以及该 URL 的一些部分, 如协议) 将被更新, 以符合当前的 URL。其语法格式如下, 参数说明如表 23.7 所示。

object.OpenUrl url [,datatype]

表 23.7 OpenURL 方法的参数说明

参 数	描 述
object	对象表达式
url	必选项。被检索文档的 URL
datatype	可选项。整数, 指定数据类型。其中 datatype 的设置值为 icString 时, 表示把数据作为字符串来检索; datatype 的设置值为 icByteArray 时, 表示把数据作为字节数组来检索

### 23.3.2 文件上传与下载

**例 23.2** 开发一个文件上传与下载的应用程序。该程序主要利用 FTP 协议连接 Internet，并检索文档位置，来实现其上传与下载。具体设置步骤如下：（实例位置：光盘\TM\sl\23\2）

#### 1. 设置 FTP 服务器

（1）选择操作系统“开始”菜单中的“控制面板”/“管理工具”/“Internet 信息服务（IIS）管理器”命令。在弹出的“Internet 信息服务（IIS）管理器”窗口中展开“FTP 站点”子节点，如图 23.6 所示。

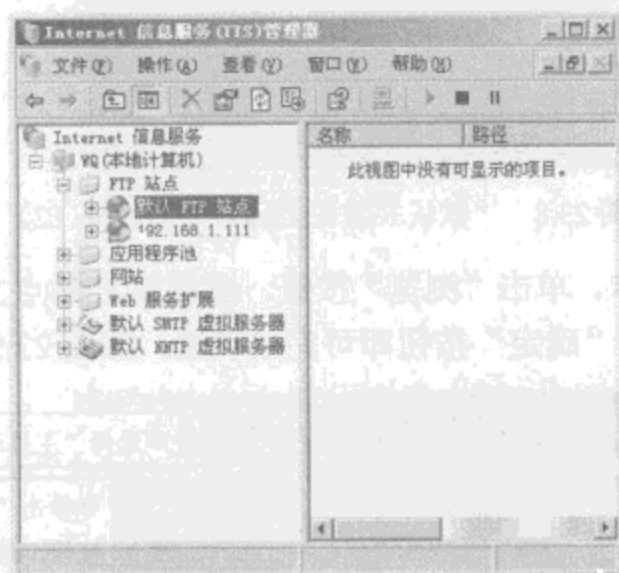


图 23.6 “Internet 信息服务（IIS）管理器”窗口

（2）选择“默认 FTP 站点”并单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，弹出“默认 FTP 站点属性”对话框。在该对话框中选择“FTP 站点”选项卡，并在 IP 地址下拉文本框中输入本机 IP 地址，如图 23.7 所示。

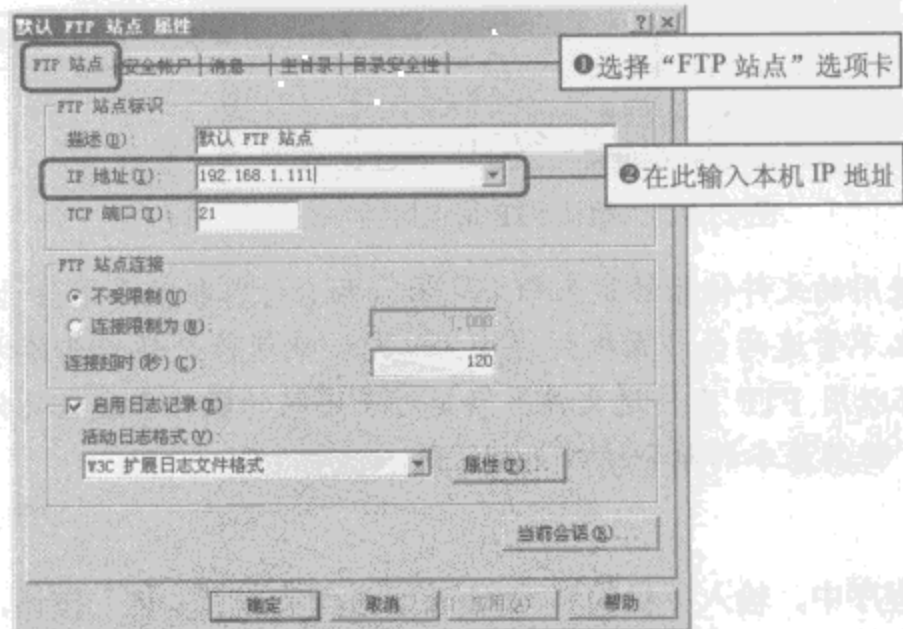


图 23.7 “默认 FTP 站点属性”对话框（1）

(3) 选择“安全账户”选项卡，并选中“允许匿名连接”复选框，如图 23.8 所示。

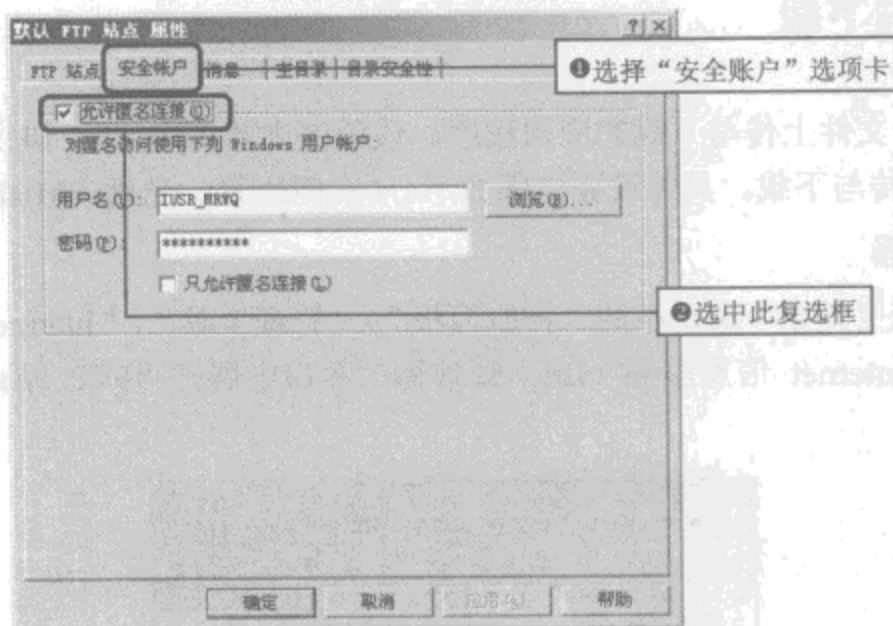


图 23.8 “默认 FTP 站点属性”对话框 (2)

(4) 选择“主目录”选项卡，单击“浏览”按钮，选择 FTP 站点文件的存放路径，并在下方的复选框中设置访问权限，最后单击“确定”按钮即可完成设置，如图 23.9 所示。

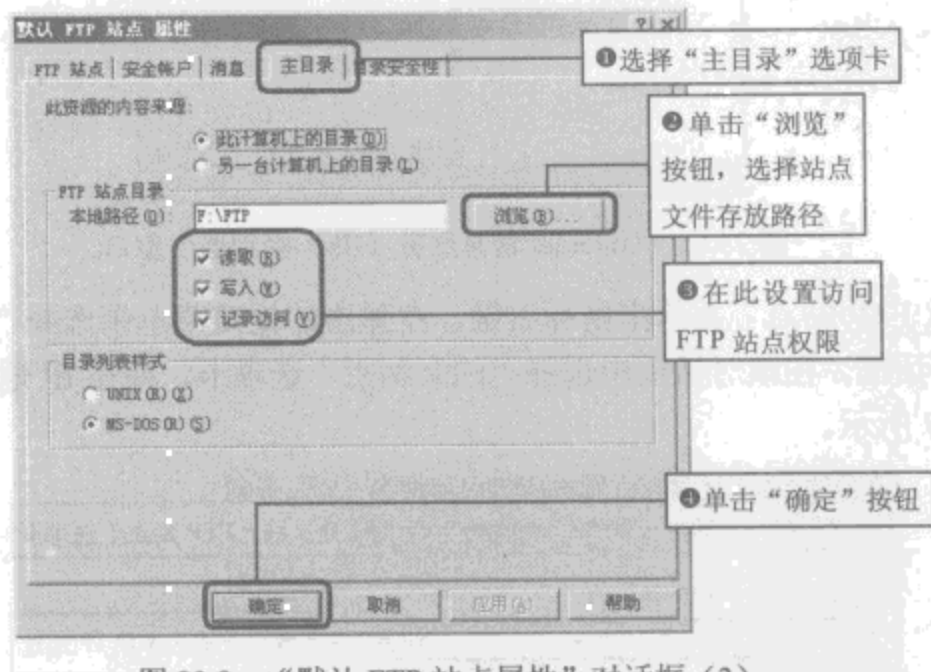


图 23.9 “默认 FTP 站点属性”对话框 (3)

**说明：**FTP 以它所使用的文件传输协议来命名。假如两台计算机能与 FTP 协议对话，并且能访问 Internet，那么不管这两台计算机处于什么位置、采用什么样的连接方式和使用什么样的操作系统，都可以用 FTP 来传送文件，只是对于不同的操作系统在具体操作上可能会有一些细微的差别，但其基本的命令结构是相同的。

## 2. 上传文件

在上传文件的应用程序中，输入主机的 IP 地址或域名，单击“上传”按钮，程序将连接服务器，如果连接成功，即可将所选文件上传至 FTP 站点所设置的文件夹中，其运行效果如图 23.10 所示。

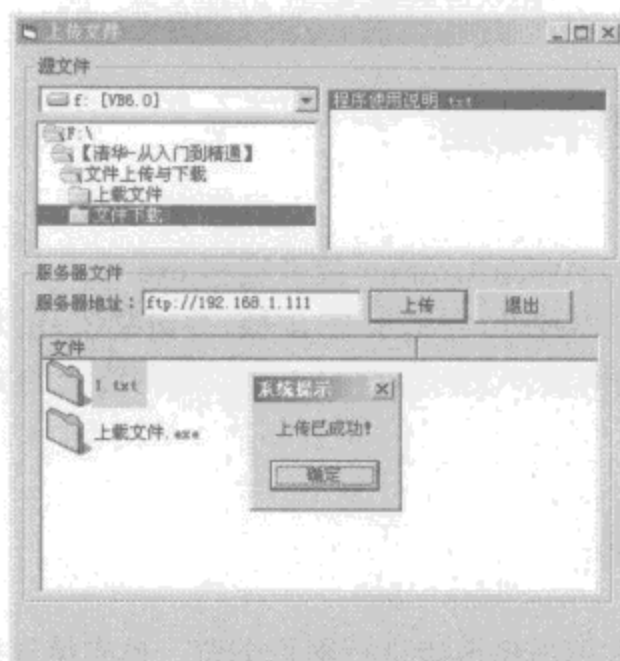


图 23.10 上传文件

下面是上传文件端应用到的关键代码。

```
Public StateStyle As Integer
Private Sub cmdSend_Click()                                '“上传”按钮
    Dim myfilepath As String
    myfilepath = File1.Path & "\" & File1.FileName        '文件位置
    StateStyle = 1
    Inet1.Execute txtURL.Text, "SEND " & myfilepath & " " & File1.FileName '上传文件
    MsgBox "上传已成功!", , "系统提示"
    StateStyle = 0
    Inet1.AccessType = icUseDefault                        '设置与 Internet 连接的类型
    Inet1.Protocol = icFTP                                 '指定 FTP 协议
    Inet1.RemotePort = 21                                  '设置连接远程端口号为 21
    Inet1.Execute txtURL.Text, "DIR "                      '检索目录
End Sub
```

### 3. 下载文件

在文件下载的应用程序中,分别输入服务器端的 IP 地址、下载文件名和另存为本机上的文件名,然后单击“下载”按钮,使得文件下载端的应用程序与服务器端的程序取得连接,这时才可以进行下载文件操作,并将其文件下载到该程序所在路径下。运行效果如图 23.11 所示。

下面是文件下载端应用到的关键代码。

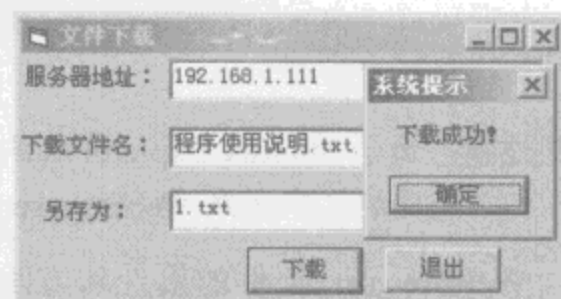


图 23.11 文件下载

```
Private Sub Command1_Click()                                '“下载”按钮
    Inet1.AccessType = icUseDefault                        '设置与 Internet 连接的类型
    Inet1.Protocol = icFTP                                 '指定 FTP 协议
    Inet1.RemotePort = 21                                  '设置连接远程端口号为 21
    Inet1.Execute txtURL.Text, "GET " & txtFile.Text & " " & txtSave.Text '下载文件
End Sub
```




## 23.4 WebBrowser 控件编程

 教学录像：光盘\TM\lx\23\WebBrowser 控件编程.exe

WebBrowser 控件是一个浏览器控件，它基于 IE 内核，并封装了 IE 大部分的功能。利用 WebBrowser 控件不仅可以浏览 Internet 上的网页，又可以查看本地或者网络上的文件，同时还可以开发自己的浏览器程序。

### 23.4.1 WebBrowser 控件

WebBrowser 控件既支持通过单击超链接进行网页浏览，也支持通过输入 URL 地址进行浏览。该控件还能保存一个历史列表，以使用户进行向前、向后或者返回到前若干个浏览过的站点、文件夹和文件等信息。在 VB 中使用 WebBrowser 控件之前，需要选择菜单栏中的“工程”/“部件”命令，在弹出的对话框中，选中 Microsoft Internet Controls 复选框，单击“确定”按钮即可将  图标添加到工具箱中。“部件”对话框如图 23.12 所示。

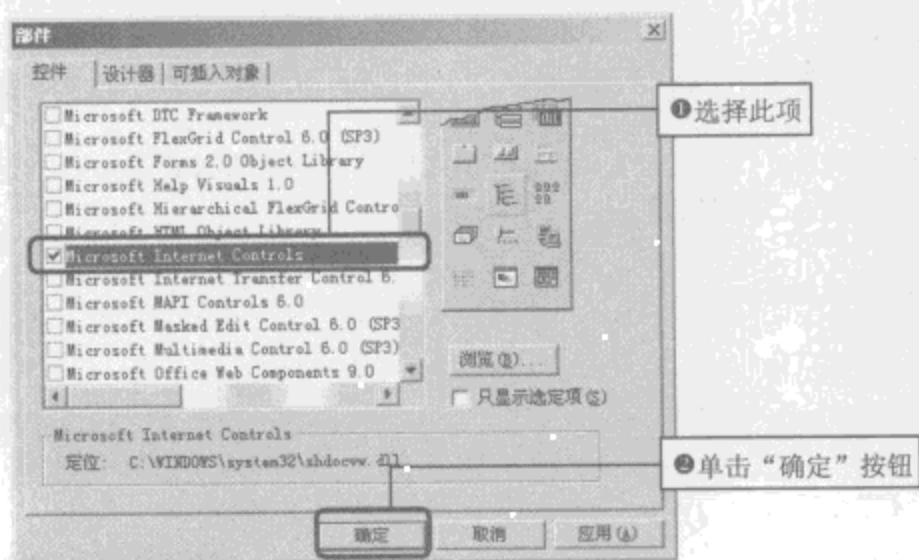



图 23.12 “部件”对话框

 注意：由于 Internet Explorer 版本的不同，可能该控件的名字也有所不同。在较低的版本中，该控件的名字叫做 WebBrowser 控件。

WebBrowser 控件的属性和方法将在 23.4.2 小节的实例中进行体现，在此只简单进行介绍。WebBrowser 控件的属性和方法具体说明如表 23.8 所示。

表 23.8 WebBrowser 控件的属性和方法说明

属性/方法	说 明
LocationName 属性	用于返回访问 Web 页的标题名称
LocationURL 属性	用于设置或返回 Web 浏览器控件浏览网页时被浏览网页的 URL 地址

续表

属性/方法	说 明
Navigate 方法	用于打开所要浏览的网页
Refresh 方法	用于刷新正在浏览的网页
GoBack 方法	用于设置或返回上一页浏览过的网页页面
GoForward 方法	用于设置或返回下一页浏览过的网页页面
GoHome 方法	用于显示或设置网站的主页
Stop 方法	用于停止或设置正在显示的网页

### 23.4.2 制作自己的浏览器

**例 23.3** 开发制作一个自己的浏览器程序。在这个浏览器中,可以实现浏览网页、设置主页、停止显示网页、刷新网页等功能。在程序运行时,首先在“浏览地址”右侧的文本框中输入要访问网站的地址,然后按下〈Enter〉键即可在下方显示该网页信息内容。其实现效果如图 23.13 所示。(实例位置:光盘\TM\sl\23\3)

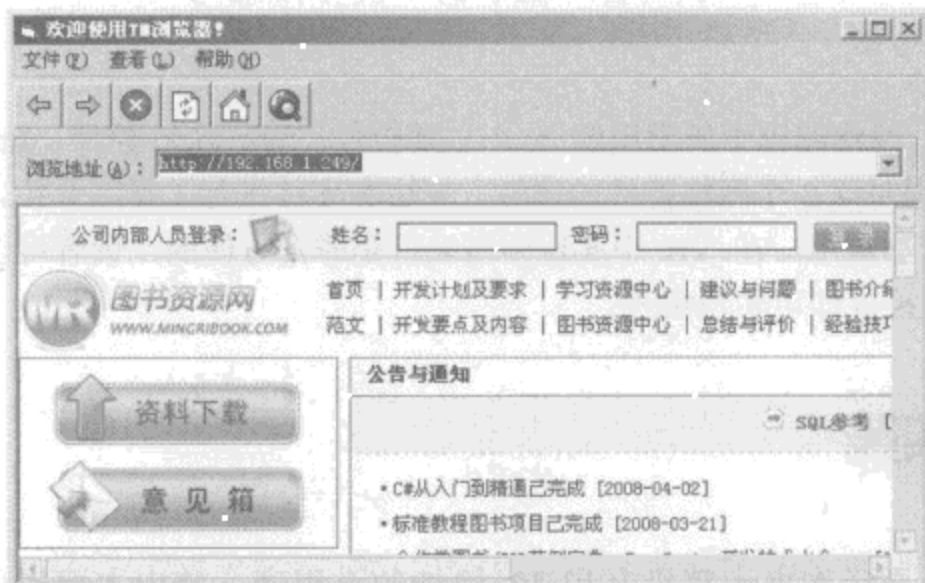


图 23.13 浏览器

1. 下面是显示网页信息所应用到的关键代码。

```
Private Sub Form_Load()  
    StrURL = Combo1.Text  
    WebBrowser1.Navigate StrURL  
End Sub
```

'浏览网页'

2. 下面是控制工具栏向前、后退、停止、刷新和设置为主页信息所应用到的关键代码。

```
Private Sub tbToolBar_ButtonClick(ByVal Button As Button)  
    On Error Resume Next  
    Select Case Button.Key  
        Case "Back"
```

'后退'

```

WebBrowser1.GoBack
Case "Forward"           '前进
WebBrowser1.GoForward
Case "Refresh"           '刷新
WebBrowser1.Refresh
Me.Caption = WebBrowser1.LocationName
Case "Home"              '主页
WebBrowser1.GoHome
Combo1.Text = WebBrowser1.LocationURL
Case "Search"            '搜索
WebBrowser1.GoSearch
Me.Caption = WebBrowser1.LocationName
Combo1.Text = WebBrowser1.LocationURL
Case "Stop"              '停止
WebBrowser1.Stop
Me.Caption = WebBrowser1.LocationName
End Select
End Sub

```

## 23.5 小结

本章主要介绍 VB 网络编程中所常用的 3 个主要控件：Winsock 控件、Internet Transfer 控件和 WebBrowser 控件，分别对这 3 个网络控件的属性、方法和事件进行了详细介绍，并通过简单的实例加深了对这些控件的理解与运用。在 VB 中实现网络编程是非常方便的，读者可以不了解网络通信的基础知识，只需了解网络控件即可实现编程。

## 23.6 练习与实践

1. 尝试开发一个显示本机计算机名和 IP 地址的小程序。其中主要应用到 Winsock 控件的 LocalHostName 属性和 LocalIP 属性。（答案位置：光盘\TM\sl\23\4）
2. 尝试开发一个提取网页标题名称的小程序。其中主要应用到 WebBrowser 控件的 Navigate 方法和 LocationName 属性。（答案位置：光盘\TM\sl\23\5）
3. 尝试开发一个提取网页源码的小程序。其中主要应用到 WebBrowser 控件的 Navigate 方法和 Internet Transfer 控件的 OpenURL 方法。（答案位置：光盘\TM\sl\23\6）

# 第24章

## 自动化控制

(教学录像: 35 分钟)

具有自动功能的应用程序和 COM 组件可以是自动服务程序、客户应用程序 (Client)，或者两者兼有。作为服务程序的组件可以向其他应用程序提供对象；作为客户应用程序的组件可以创建对象。

可以很容易地在 VB 应用程序中合并 Microsoft Excel 和 Microsoft Word 等应用程序的功能和灵活性。因为 VB 也可以作为服务程序，所以也可以把它的功能集成到基于 Microsoft Office 或其他 COM 组件的解决方案组中。

可插入的 OLE 对象来自于支持 OLE 的应用程序，例如 Microsoft Excel 和 Microsoft Word。这样的对象包括 Word 文档和 Excel 工作表，在表单上可以使用 OLE 容器控件链接或嵌入这些对象，并且可以在表的通用型字段中保存这些对象，使用 OLE 绑定型控件在表单上进行显示。

通过阅读本章，您可以：

- » 了解 OLE 控件的基本功能
- » 掌握 OLE 控件的常用属性
- » 掌握 OLE 控件的常用方法和事件
- » 了解常用的 Word 对象
- » 掌握如何使用 Word 对象
- » 掌握如何使用 VB 控制 Word
- » 了解常用的 Excel 对象
- » 掌握如何使用 Excel 对象
- » 掌握如何使用 VB 控制 Excel




## 24.1 OLE 控件

 教学录像：光盘\TM\lx\24\OLE 控件.exe

OLE 的含义是对象的链接和嵌入 (Object Linked Embedded)。如果部件支持对象链接和嵌入，则通过在应用程序中插入 OLE 对象，可以不用任何代码就把对象链接和嵌入到应用程序中。用 OLE 容器控件可以包含来自另一个应用程序的数据。

### 24.1.1 OLE 控件的功能

VB 的控件箱中提供了 OLE 控件 ，为使用对象的可视化提供了很大的灵活性。如图 24.1 中的 OLE 控件中嵌入了 Excel 对象。

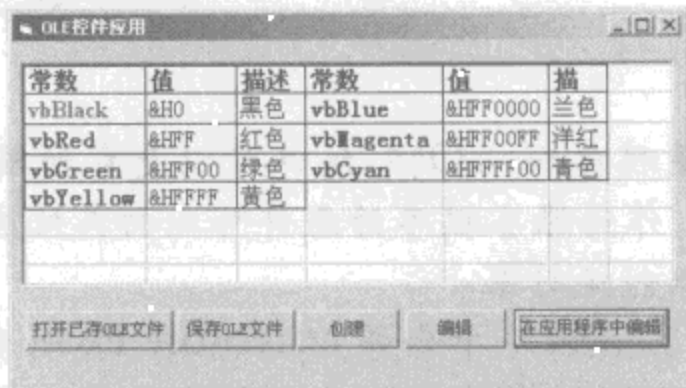


图 24.1 OLE 控件中嵌入 Excel 对象

OLE 控件的主要功能有：

- ☒ 在应用程序中为可插入对象建立占位符，在程序运行时动态地创建或者改变 OLE 控件中的对象。
- ☒ 在应用程序中创建链接对象。
- ☒ 可以在设计时使用插入对象对话框（包含插入对象、特殊粘贴等命令）来创建对象，也可以在运行时通过设置相应的属性来创建对象。
- ☒ 将对象显示为图标。
- ☒ 用户可以在 OLE 控件中执行相应的操作来移动、调整对象尺寸或者更新对象等。
- ☒ 从剪贴板的数据中创建对象。
- ☒ OLE 控件还可以绑定数据库。

OLE 容器控件一次只能包含一个对象。创建一个链接的对象或者内嵌对象时，可以使用以下几种方法：

- ☒ 使用“插入对象”或“特殊粘贴”对话框（在运行时或设计时）。
- ☒ 在属性窗口设置 Class 属性，用鼠标右键单击 OLE 容器控件，然后再选择相应的命令（只能在设计时）。
- ☒ 使用 OLE 容器控件的相应的方法。

☑ 在属性窗口选择 Class 属性，并且单击“属性”按钮，可以得到应用程序可用的类名的列表。

🔊 注意：插入对象对话框并不显示类名的列表。这个对话框显示每个对象类的名称，该名称一般比较长，也比较容易理解。

## 24.1.2 OLE 控件的常用属性

### 1. AutoActivate 属性

AutoActivate 属性返回或设置一个值，允许通过双击 OLE 容器控件或将焦点移到 OLE 容器控件来激活对象，换句话说就是用于设置激活 OLE 控件的对象的方式。OLE 对象的激活方式如表 24.1 所示。

表 24.1 OLE 控件的对象的激活方式

属 性 常 数	属 性 值	含 义
vbOLEActivateManual	0	手工激活。对象不能自动激活，可以使用程序的 DoVerb 方法激活对象
vbOLEActivateGetFocus	1	接受焦点激活。如果 OLE 容器控件包含的对象支持单击激活，当 OLE 容器控件接收焦点时，将提供对象的应用程序激活
vbOLEActivateDoubleClick	2	（默认值）双击或按〈Enter〉键激活。在 OLE 容器控件获得焦点以后，在其上双击或按〈ENTER〉键时，将提供对象的应用程序激活
vbOLEActivateAuto	3	自动的。如果 OLE 容器控件包含对象，当控件接收焦点或当双击控件时，均根据对象规范的激活方法，将提供对象的应用程序激活

### 2. AppIsRunning 属性

AppIsRunning 属性返回或设置一个值，指示在 OLE 容器控件中创建对象的应用程序是否正在运行，在设计时不可用。

可以设置 AppIsRunning 属性的值，来启动在 OLE 容器控件中创建对象的应用程序。这样可以使对象激活得更快。当对象丢失焦点后，也可以将这个属性设置为 False 来关闭应用程序。

### 3. Object 属性

Object 属性返回对象或 OLE 容器控件中对象的方法或属性的设置。通过使用 OLE 控件的 Object 属性，可以使用链接或者嵌入对象的属性和方法。Object 属性在运行时是只读的，保证了对 OLE 控件中的对象的引用。如要使用此属性，OLE 控件必须包含可编程的对象。

使用语法是：

```
object.Object[.property | .method]
```

参数说明如表 24.2 所示。

表 24.2 Object 属性参数说明

参 数	说 明
object	对象表达式，其值是 OLE 控件
Property	参数指明对象支持的属性
Method	参数指明对象支持的方法

#### 4. ObjectVerbs 属性

返回对象所支持的谓词列表。

语法:

```
object.ObjectVerbs(number)
```

语法参数如下。

Object: 对象表达式, 其值是 OLE 控件。

Number: 一个指示数组中元素的数值表达式。

例如, 打印所有可用的谓词列表的语句如下:

```
For i = 0 To OLE1.ObjectVerbsCount - 1      '循环读取对象支持的谓词
    Print OLE1.ObjectVerbs(i)                '如果支持 Open 谓词
Next i
```

#### 5. ObjectVerbsCount 属性

ObjectVerbsCount 属性返回对象支持的谓词的个数。

语法:

```
object.ObjectVerbsCount
```

object: 对象表达式, 其值是一个 OLE 容器控件。

使用这个属性确定 ObjectVerbs 属性数组中元素的个数。对象支持的谓词列表会随着对象的状态而变化, 为了更新对象所支持的谓词列表, 可使用 FetchVerbs 方法。

**例 24.1** 检查 OLE 对象的动词列表, 如果找到 Open 动词, 就执行该动词, 在 OLE 控件的对象对应的应用程序中打开该对象。(实例位置: 光盘\TM\sl\24\1)

新建一个工程, 在打开的窗体中添加一个 OLE 控件并嵌入一个可用的对象 (如 Microsoft Excel), 然后再添加一个 Command 控件。打开代码编辑窗口输入以下的代码:

```
Private Sub Command1_Click()
    Dim i As Integer '定义一个整型变量
    For i = 0 To OLE1.ObjectVerbsCount - 1      '循环读取对象支持的谓词
        If OLE1.ObjectVerbs(i) = "打开(&O)" Then '如果支持 Open 谓词
            OLE1.DoVerb i                        '打开该对象
            Exit Sub                             '跳出 Sub 过程
        End If                                  '结束 If 语句
    Next i                                       'i 值加 1
End Sub
```

#### 6. SizeMode 属性

SizeMode 属性返回或设置一个值, 它指定 OLE 容器控件如何调整大小, 或者如何显示图象。可取值及意义参见表 24.3 所示。

表 24.3 SizeMode 属性的取值及功能描述

常 数	值	功 能 描 述
VbOLESizeClip	0	(默认值) 剪裁。对象按实际大小显示。如果对象比 OLE 容器控件大, 它的图像被控件的边框剪裁掉


续表

常 数	值	功 能 描 述
VbOLESizeStretch	1	伸展。调整对象图象的大小使其充满 OLE 容器控件。图象也许不能维持对象原来的比例
VbOLESizeAutoSize	2	自动。OLE 容器控件重新调整大小以显示整个对象
VbOLESizeZoom	3	缩放。重新调整对象的大小使其尽可能充满 OLE 容器控件，并且仍然维持该对象原来的比例

例如，在加载时让系统将重新调整对象的大小使其尽可能充满 OLE1 容器控件，并且仍然维持该对象原来的比例。代码如下。

```
OLE1.SizeMode = 3
```

'自动调整控件内对象的大小'

 说明：当 SizeMode 设置为 2（自动）时，如果显示对象的大小作了改变，OLE 容器控件自动调整大小。此时，在 OLE 容器控件自动调整大小之前，先调用 Resize 事件。在 Resize 事件过程中的 heightnew 和 widthnew 参数，指示显示对象的最佳大小（其大小由创建该对象的应用程序决定）。可以改变 Resize 事件过程中的 heightnew 和 widthnew 参数的值，来调整控件的大小。

### 7. UpdateOptions 属性

返回或设置一个值，指示当链接数据修改后如何更新对象。

语法格式如下：

```
object.UpdateOptions [= number]
```

其参数说明如下。

Object：对象表达式。

Number：整数，指示如何更新对象，其取值及描述如表 24.4 所示。

表 24.4 UpdateOptions 属性取值及功能描述

常 数	值	描 述
vbOLEAutomatic	0	（默认值）自动的。每次改变链接数据时均更新对象
vbOLEFrozen	1	冻结的。无论何时从生成对象的应用程序内保存链接数据，均更新对象
vbOLEManual	2	手动的。只有使用 Update 方法才更新对象

 说明：对链接的对象来说，当其他用户或应用程序能够存取和修改链接数据时，这个属性是很有用的。当改变对象的数据时，调用 Updated 事件。

## 24.1.3 OLE 控件的常用方法和事件

### 1. 常用的方法

#### ☒ Delete 方法

删除指定的对象，而释放与之关联的内存。用于显式地删除对象。当关闭窗体时或对象被一个新对象取代时，对象都将被自动删除。



语法:

```
object.Delete
```

其中, object 是一个对象表达式, 其值是一个 OLE 控件。

例如, 运行时删除 OLE1 控件中的对象, 以释放与之关联的内存, 语句为:

```
OLE1.Delete           '删除与 OLE1 关联的对象
```

#### ☒ DoVerb 方法

打开一个对象, 用于进行诸如编辑操作。不支持命名的参数。

语法:

```
object.DoVerb (verb)
```

其中 object 是对象表达式, 其值是一个 OLE 控件。verb 可选, 是在 OLE 容器控件内要执行的对象的谓词。如不指定, 就执行默认谓词。这个参数的值可以是一个所有对象都能支持的标准谓词, 也可以是 ObjectVerbs 属性数组的一个索引。例如在例 24.1 中使用 DoVerb 方法打开嵌入的对象。

#### ☒ ReadFromFile 方法

从用 SaveToFile 方法创建的数据文件中加载对象, 不支持命名的参数。

语法:

```
object.ReadFromFile filename
```

其中, object 是一个对象表达式, 其值是一个 OLE 控件。Filename 参数是必选的参数, 是一个数值表达式, 指定用于加载对象的文件号。这个号必须与一个打开的二进制文件相对应。

例如, 以二进制形式读取 “C:\Test.ole” 文件的语句如下:

```
Open "c:\test.ole" For Binary As #1      '打开二进制文件
OLE1.ReadFromFile 1                      '将对象装入到 OLE1 控件中
Close #1                                '关闭打开的文件
```

#### ☒ SaveToFile 方法

SaveToFile 方法将对象保存到数据文件中, 不支持命名的参数。当含有 OLE 控件的窗体被关闭时, 与该控件相关的数据的任何变化将丢失。SaveToFile 方法只用于嵌入的对象, OLE 控件中的对象只能保存于打开的二进制文件中。

语法:

```
object.SaveToFile filename
```

其中, object 是一个对象表达式, 其值是一个 OLE 控件。Filename 参数是必选的参数, 是一个数值表达式, 指定用于加载对象的文件号。这个号必须与一个打开的二进制文件相对应。

例 24.2 开发一个程序可以保存在 OLE 控件中的对象的内容到数据文件 “C:\Test.data” 中, 又可以读取该数据文件的内容到 OLE 控件的程序。(实例位置: 光盘\TM\sl\24\2)

程序代码如下:

```
Dim fNum As Integer                    '定义一个整型变量
Private Sub Command1_Click()
    fNum = FreeFile                    '返回可用的文件号
```

```

Open "c:\test.data" For Binary As #fNum      '打开二进制文件
OLE1.SaveToFile fNum                          '将对象中的数据保存到二进制文件
Close # fNum                                 '关闭打开的文件
End Sub
Private Sub Command2_Click()
    fNum = FreeFile                          '返回可用的文件号
    Open "c:\test.data" For Binary As #fNum  '打开二进制文件
    OLE1.ReadFromFile fNum                  '将对象装入到 OLE1 控件中
    Close # fNum                             '关闭打开的文件
End Sub

```

## 2. 常用的事件

除具有控件的通用事件以外, OLE 控件还具有 Updated 事件。当对象的内容改变时, 触发 Updated 事件。该事件用于决定对象的数据在最后一次保存时是否有效。

语法:


```
Sub object_Updated (code As Integer)
```

Object: 对象表达式。

Code: 整数, 指定对象是如何更新的, 具体设置值及意义如表 24.5 所示。

表 24.5 code 属性设置及功能描述

常 数	值	描 述
vbOLEChanged	0	对象的数据已经改变
vbOLESaved	1	对象的数据已由创建该对象的应用程序保存
vbOLEClosed	2	含有链接对象数据的文件已被创建该对象的应用程序关闭
vbOLERenamed	3	含有链接对象数据的文件已被创建该对象的应用程序重命名

 说明: 这些常数列在“对象浏览器”的 VB 对象库中。可以使用这个事件, 确定在上次保存之后对象的数据是否已被修改。为此, 在 Updated 事件中设置一个全局变量, 来指示需要保存的数据。在保存对象之后, 将该变量复位。

## 24.2 利用 VB 控制 Word

 教学录像: 光盘\TM\lx\24\利用 VB 控制 Word

Word 对象封装了 Microsoft Word 的全部元素, 例如 Application 对象表示 Word 应用程序, Document 对象表示一个 Word 文档, Table 对象表示一个表格等等。下面详细讲述如何在 VB 中引用 Word 对象以及 Word 对象的使用。

### 24.2.1 如何在 VB 中使用 Word 对象

#### 1. 添加 Word 对象引用

Word 对象不是 VB 默认引用的对象, 需要自行将其添加到引用列表中。添加的方法是:

选择 VB 菜单的“工程”/“引用”命令，打开“引用-工程 1”对话框，对话框中列出了所有可用的引用对象，如图 24.2 所示。

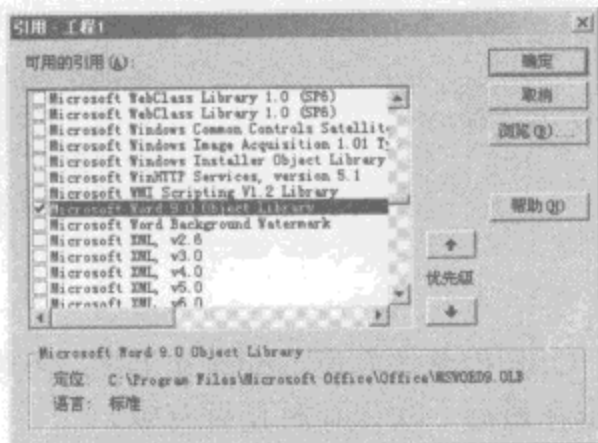



图 24.2 添加对 Word 对象的引用

在对话框的列表中选择 Microsoft Word 9.0 Object Library 列表项，单击“确定”按钮添加 Word 对象的引用。

 **说明：**由于 Word 应用程序版本的不同，列表中的 Word 对象也有所不同。如 Microsoft Word 2000 对象为“Microsoft Word 9.0 Object Library”，Microsoft Word 2003 对象为“Microsoft Word 11.0 Object Library”等，读者添加对象时应根据实际情况选择。

## 2. 使用 Word 对象

### (1) 声明 Word 对象

Application 对象、Document 对象、Table 对象等 Word 对象使用之前都需要首先声明。

例如，声明 Application 对象、Document 对象、Table 对象，代码如下：

```
Dim wdapp As Word.Application      '声明一个 Application 对象的实例
Dim wddoc As Word.Document        '声明一个 Document 对象的实例
Dim atable As Word.Table          '声明一个 Table 对象的实例
```

### (2) 创建新的 Word 对象

可以在 Set 语句中，用 New 关键字创建一个新对象并将对象引用赋予对象变量。声明 Word 对象变量时使用 New 关键字，第一次使用该变量时，VB 会自动地创建一个新对象。

例如，将新的 Application 对象赋给变量 wdapp，同时添加一个新文档的语句如下：

```
Set wdapp = New Word.Application    '声明并创建一个 Word 应用程序
Set wddoc = wdapp.Documents.Add     '在 Word 中添加一个新文档
```

除了在声明时创建 Word 对象外，还可以使用 CreateObject 方法创建一个新对象。

例如，将新的 Application 对象的引用赋予变量 wdapp，同时同时添加一个新文档，语句如下：

```
Set wdapp = CreateObject("Word.Application") '创建 Word 应用程序
Set wddoc = wdapp.Documents.Add             '在 Word 中添加一个新文档
```

### (3) 打开创建好的 Word 文件

打开已经创建的 Word 文件使用 Open 方法。

例如, 打开 Test1.doc 文件的代码如下:

```
wdapp.Documents.Open App.Path & "\Test1.doc" '打开 Test1.doc 文件
```

#### (4) 显示创建或者打开的 Word 对象

创建一个 Word 对象以后并不能马上显示, 需要调用 Word 对象的 Visible 属性进行显示。

例如, 显示创建的 Word 对象 wdapp, 代码如下:

```
wdapp.Visible = True '显示创建的 Word 应用文件
```

#### (5) 释放 Word 对象

使用 Word 对象的 Quit 方法, 可以释放 Word 报表。

例如, 释放已经声明的 Word 对象 newxls, 代码如下:

```
wdapp.Quit '释放已经创建的 Word 对象
```

### 24.2.2 常用的 Word 对象

(1) Application 对象表示 Word 应用程序。在 Word 对象模型中, Application 对象处于模型的顶端。通过使用 Application 对象, 可以访问模型中的其他对象, 从而可以控制 Word 应用程序的外观或功能等。

例如, 打开 VB 程序所在路径下的 Word 文件可以使用 Documents 对象的 Open 方法, 代码如下:

```
Application.Documents.Open App.Path & "\Doc1.doc" '打开程序所在路径下的 Word 文件
Application.Visible=True '设置打开的 Word 文件可见
```

退出 Word 可以使用 Quit 方法, 代码如下:

```
Application.Quit '退出 Word 应用程序
```

(2) Documents 对象是由 Word 当前打开的所有 Document 对象所组成的集合。

**例 24.3** 在窗口中显示已经打开的 Word 文档的名称。(实例位置: 光盘\TM\sl\24\3)

新建一个工程, 打开新建窗体的代码编辑器, 输入以下代码:

```
Dim newWord As New Word.Application '声明 Word 对象
Private Sub Form_Activate()
    Dim strDoc As String '定义一个字符型变量
    '在立即窗口中显示打开文档的名称
    For Each doc In Documents '遍历每一个打开的 Word 文档
        strDoc = doc.Name & vbCrLf & strDoc '记录打开的 Word 文档的名称
    Next doc
    Print strDoc '在立即窗口中显示打开的 Word 文档的名称
End Sub
```

用 Add 方法可以创建新的空文档, 并将其添加到 Documents 集合中。用 Open 方法可以打开文档。下面的语句用来关闭 Test.doc 文档。

```
Documents("Test.doc").Close '关闭指定的文档
```

用 Documents(index) 可以返回单个的 Document 对象, 其中的 index 是文档的名称或索引序号。索



引序号代表文档在 Documents 集合中的位置, 例如 Documents(1).Activate 是代表激活 Documents 集合中的第一个文档, 但建议使用 Documents(Test.doc)这样的显示引用, 这样可以增加代码的可读性, 而且不易出错。

(3) Document 对象表示一篇文档。它是 Documents 集合中的一个元素, Documents 集合包含 Word 当前打开的所有 Document 对象。可用 ActiveDocument 属性引用处于活动状态的文档。

例如, 使用 Activate 方法激活名为 Doc1 的文档, 然后打印该文档, 代码如下。

```
Documents("Doc1.doc").Activate      '激活 Doc.doc 文档
ActiveDocument.PrintOut              '打印激活的 Word 文档
```

(4) Range 对象代表文档中的一个范围。每一个 Range 对象由一个起始和一个终止字符位置定义。例如, 返回代表活动文档前 20 个字符的 Range 对象, 代码如下。

```
Set myRange=ActiveDocument.Range(Start:=0,End:=20)  '返回活动文档的前 20 个字符
```

(5) Table 对象代表一个表格。它是 Tables 集合的一个成员。Tables 集合包含了指定选定内容、范围或文档中的所有表格。

使用 Add 方法可以在指定范围内添加表格。

例如, 在一个名称为 Doc1.doc 的 Word 文档中创建一个 3 行 7 列的表格, 代码如下。

```
Dim newword As New Word.Application      '声明一个 Word.Application 对象
Dim newtable As Word.Table               '声明一个 Word.Table 对象
newword.Documents.Open App.Path & "\Doc1.doc"  '打开 Doc1.doc 文档
Set newtable = newword.ActiveDocument.Tables.Add(newword.Selection.Range, 3, 7) '创建一个的表格
newword.Visible = True                   '显示该 Word 文档
```

(6) Cell 对象代表单个表格单元格。它是 Cells 集合中的元素。Cells 集合代表指定对象中所有的单元格。使用语法如下:

```
Cell(row, column)
```

其中, 参数 row 为行号, 参数 column 为列号。返回 Cell 对象。

例如, 为第一单元格设置底纹, 代码如下。

```
newdoc.Tables(1).Rows(1).Cells(1).Shading.Texture = wdTexture20Percent  '为第一个单元格设置底纹
```

### 24.2.3 提取指定目录下的文件夹

如果计算机上的文件夹较多, 并且都经常被使用, 可以将计算机上的文件夹提取出来放到一个 Word 文档中, 利用 Microsoft Word 强大的排版功能对文档重新编辑, 以提高效率。手工进行提取这些信息是非常费时费力的。通过本例可以很方便地将指定目录下的文件夹提取到 Word 表格中。运行程序, 选择要提取的文件夹所在的路径, 并设置是否提取完全目录信息, 单击“提取”按钮将信息提取到 Word 表格当中。实例运行结果如图 24.3 所示。

例 24.4 提取指定目录下的文件夹。(实例位置: 光盘\TM\sl\24\4)



图 24.3 提取指定目录中的文件夹

下面讲述一下本程序的开发步骤:

- (1) 新建一个标准工程, 创建一个新窗体。
- (2) 在窗体上添加一个 Frame 控件, 默认的“名称”属性为 Frame1, 设置其 Caption 属性为空。
- (3) 在 Frame1 控件中添加一个 DriveListBox 控件、一个 DirListBox 控件、一个 CheckBox 控件和两个 CommandButton 控件, 设置 CheckBox 控件的 Caption 属性值为“提取文件夹的完整路径”, 分别设置两个 CommandButton 控件的 Caption 属性为“提取”和“退出”。
- (4) 按照前面添加 Word 对象引用的方法添加 Word 对象的引用。
- (5) 打开代码编辑器, 输入以下代码:

```
Private Sub Form_Load()
    Me.Left = (Screen.Width - Me.Width) / 2           '控制窗体启动时在水平方向上位于屏幕中心
    Me.Top = (Screen.Height - Me.Height) / 2          '控制窗体启动时在竖直方向上位于屏幕中心
End Sub
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive                          '设置 DirListBox 控件的路径
End Sub
Private Sub Command1_Click()
    Dim i, ifieldcount As Integer, irecordcount As Integer '定义整形变量
    Dim wdapp As Word.Application                     '声明一个 Application 对象的实例
    Dim wddoc As Word.Document                       '声明一个 Document 对象的实例
    Dim atable As Word.Table                          '声明一个 Table 对象的实例
    irecordcount = Dir1.ListCount                     '取得指定目录下文件夹的个数
    Set wdapp = CreateObject("Word.Application")      '创建 Word 应用程序, 这一句打开 Word 2000
    Set wddoc = wdapp.Documents.Add                  '在 Word 中添加一个新文档
    With wdapp                                        '使用 With 结构
        .Visible = True                             '显示创建的 Word 文档
        .Activate                                    '是创建的 Word 文档为当前活动文档
        '在 Word 文档中添加一个表格
        Set atable = .ActiveDocument.Tables.Add(.Selection.Range, irecordcount + 1, 2)
        '添加表格的表头
        atable.Cell(1, 1).Range.InsertAfter "文件夹名" '设置表格中第一行第一列单元格的内容
        atable.Cell(1, 2).Range.InsertAfter "说明"     '设置表格第一行第二列单元格的内容
        '指定表格内容
        For i = 0 To irecordcount + 1                 '遍历指定目录下的文件夹
            If Check1.Value = 1 Then                  '如果要提取文件夹的完全路径
                atable.Cell(i + 2, 1).Range.InsertAfter Dir1.List(i) '向表格中插入提取的信息
            Else                                        '如果不需要提取文件夹的完全路径
                atable.Cell(i + 2, 1).Range.InsertAfter RName(Dir1.List(i)) '向表格中插入提取的信息
            End If
        Next i
    End With
    Set wdapp = Nothing                               '清除 Word.Application 对象
    Set wddoc = Nothing                               '清除 Word.Document 对象
End Sub
Private Function RName(str As String) As String
    '提取文件夹的名称
    Dim k As Integer                                 '定义整形变量
```

Dim RigStr As String	'定义字符串变量
k = 1	'设置整型变量的初值
If str <> "" Then	'如果目录不为空
Do	
RigStr = Left(Right(str, k), 1)	'取出字符串右边第 k 个字符
If RigStr <> "\" Then	'如果取出的字符不等以 "\"
RName = RigStr & Rname	'把该字符赋给变量
End If	
k = k + 1	
Loop While RigStr <> "\"	'当取出的字符为 "\" 时结束循环
End If	
End Function	

## 24.3 利用 VB 控制 Excel

### 教学录像：光盘\TM\lx\24\利用 VB 控制 Excel

使用 VB 的数据报表设计器可以很方便地创建数据报表，然而，用数据报表设计器设计的报表格式一般较简单，功能也较少。现在人们大多熟悉 Office 办公软件，习惯于使用 Excel 报表处理表格数据，用户希望数据应用系统也能生成 Excel 格式的报表。因此，使用 VB 创建 Excel 报表是开发一个数据库应用系统的重要任务之一。

Excel 对象封装了 Microsoft Excel 的全部元素，例如 Application 对象表示 Excel 应用程序，Worksheet 对象表示 Excel 工作表等等。利用 Excel 对象提供的属性、方法和事件，程序的最终用户可以用 Excel 程序打开数据库应用系统生成的报表，并利用 Excel 本身的功能完成报表的打印预览、打印、保存等功能。下面介绍如何在 VB 中引用 Excel 对象和 Excel 对象的使用。

### 24.3.1 如何在 VB 中使用 Excel 对象

#### 1. 添加 Excel 对象引用

Excel 对象不是 VB 默认引用的对象，需要自行将其添加到引用列表中。添加的方法是：

(1) 选择 VB 菜单的“工程”/“引用”命令，打开“引用-工程 1”对话框，对话框中列出了所有可用的引用对象，如图 24.4 所示。

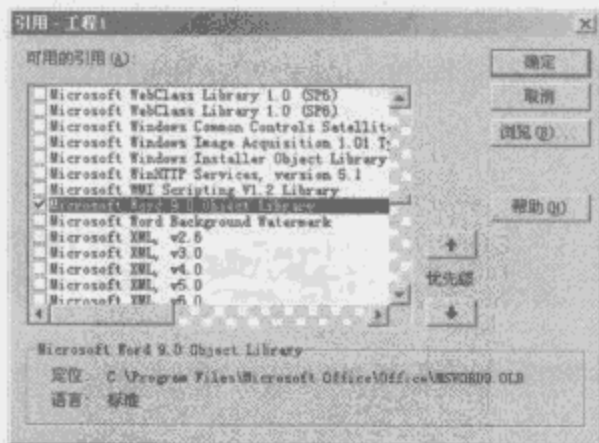



图 24.4 添加对 Excel 对象的引用

(2) 在对话框的列表中选择 Microsoft Excel 9.0 Object Library 列表项, 单击“确定”按钮添加 Excel 对象的引用。

 说明: 由于 Excel 应用程序版本的不同, 列表中的 Microsoft Excel 对象也有所不同。如 Microsoft Excel 2000 对象为 Microsoft Excel 9.0 Object Library, Microsoft Excel 2003 对象为 Microsoft Excel 11.0 Object Library 等, 读者添加对象时应根据实际情况选择。

## 2. 使用 Excel 对象

### (1) 声明 Excel 对象

声明 Excel 对象变量时使用 New 关键字, 第一次使用该变量时, VB 会自动地创建一个新对象。例如, 声明 Application 对象、Workbook 对象和 Worksheet 对象, 代码如下:

Dim newxls As New Excel.Application	'声明 Application 对象
Dim newbook As New Excel.Workbook	'声明 Workbook 对象
Dim newsheet As New Excel.Worksheet	'声明 Worksheet 对象

### (2) 创建新的 Excel 对象

可以在 Set 语句中, 用 New 关键字创建一个新对象并将对象引用赋予对象变量。

例如, 将新的 Application 对象的引用赋予变量 newxls, 同时创建一个工作簿和一个工作表, 语句如下。

Dim newxls As Excel.Application	'声明 Application 对象
Set newxls = New Excel.Application	'创建一个新 Application 对象赋给变量
Set newbook = newxls.Workbooks.Add	'创建工作簿
Set newsheet = newbook.Worksheets(1)	'创建工作表

### (3) 打开创建好的 Excel 文件

打开已经创建的 Excel 文件, 使用 Open 方法。

例如, 打开 book1.xls 文件的代码如下:

newxls.Workbooks.Open App.Path & "\book1.xls"	'打开 book1.xls 文件
---	------------------

### (4) 显示创建或者打开的 Excel 对象

创建一个 Excel 对象以后并不能马上显示, 需要调用 Excel 对象的 Visible 属性进行显示。

例如, 显示创建的 Excel 对象 newxls:

newxls.Visible = True	'显示创建的 Excel 应用文件
-----------------------	-------------------

### (5) 释放 Excel 对象

使用 Excel 对象的 Quit 方法, 可以释放 Excel 报表。

例如, 释放已经声明的 Excel 对象 newxls, 代码如下:

newxls.Quit	'释放已经创建的 Excel 对象
-------------	-------------------

## 24.3.2 常用的 Excel 对象

(1) Application 对象表示 Excel 应用程序。在 Excel 对象模型中, Application 对象处于模型的顶



端。通过使用 Application 对象，可以访问模型中的其他对象，从而可以控制 Excel 应用程序的外观或功能等。

例如，打开 VB 程序所在路径下的 Excel 文件，代码如下：

Application.Workbooks.Open App.Path & "\book1.xls"	'打开 book1.xls 的 Excel 文件
Application.Visible = True	'使 Excel 窗口可见

退出已经打开的 Excel 应用程序，代码如下：

Application.Quit	'退出 Excel 应用程序
------------------	----------------

(2) Workbook 对象表示 Excel 中的工作簿，即对应一个 Excel 文件。通过使用 Workbook 对象，可以实现对 Excel 工作簿的各种控制。

例如，创建一个工作簿，代码如下：

Application.Workbooks.add	'创建一个工作簿
Application.Visible = True	'使 Excel 窗口可见

(3) Worksheet 对象表示 Excel 中的工作表。通过使用 Worksheet 对象，可以实现对 Excel 工作表的各种控制。

例如，打开一个 Excel 文件并创建一个工作表，代码如下：

Application.Workbooks.Open App.Path & "\book1.xls"	'打开 book1.xls 的 Excel 文件
Application.Worksheets.Add	'创建一个工作表
Application.Visible = True	'使 Excel 窗口可见

(4) Range 对象表示 Excel 中的区间，例如可以代表 Excel 的某一单元格、某一行、某一列、某一选定区域（该选定区域可包含一个或若干连续单元格区域）或者某一个三维区域。

**例 24.5** 创建一个 Excel 工作表，并将 Excel 单元格 A1 中的值设置为“控制 Excel”。（实例位置：光盘\TM\sl\24\5）

新建一个标准工程，创建一个新窗体。

Dim newxls As New Excel.Application	'使用 New 关键字声明 Application 对象
Dim newbook As New Excel.Workbook	'使用 New 关键字声明 Workbook 对象
Dim newsheet As New Excel.Worksheet	'使用 New 关键字声明 Worksheet 对象
Private Sub Form_Load()	
Set newbook = newxls.Workbooks.Add	'创建新的工作簿并赋给变量 newbook
Set newsheet = newbook.Worksheets(1)	'创建新的工作表并付给变量 newsheet
newsheet.Range("A1").Value = "控制 Excel"	'将工作表 A1 中的值设为"控制 Excel"
newxls.Visible = True	'使 Excel 窗口可见
End Sub	

### 24.3.3 把数据导出到 Excel 中

下面根据前面所学的知识，来制作一个将程序中的数据导出到 Excel 中的实例。运行程序，如图 24.5 所示，单击“查询”按钮，根据给定的查询条件查询数据，单击“导出”按钮，将查询结果导出到 Excel 表中。（实例位置：光盘\TM\sl\24\6）

查询到符合条件的数据以后，单击“导出到 Excel”按钮，系统将自动创建一个 Excel 文件并把表

格中的内容导入到 Excel 文件中, 导出完成以后的 Excel 文件如图 24.6 所示。

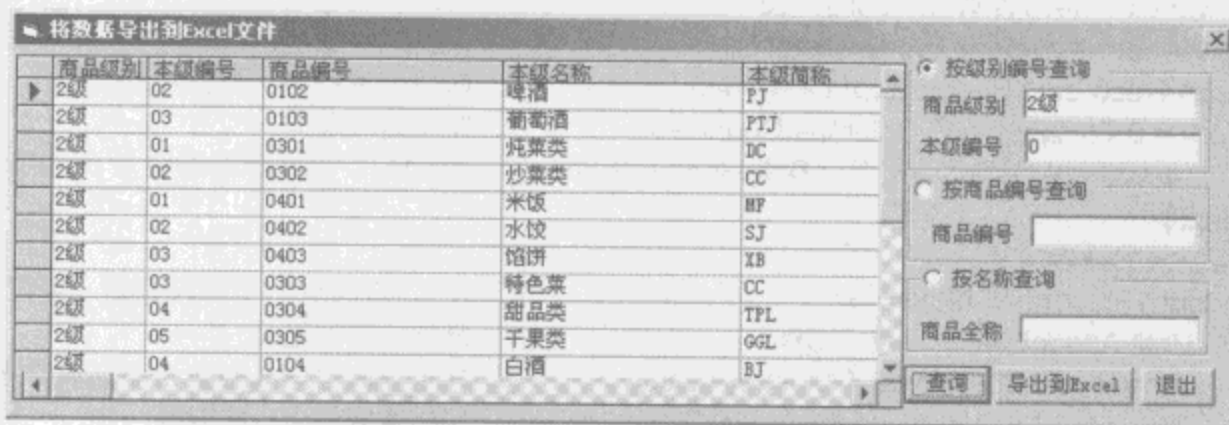


图 24.5 将数据导出到 Excel 文件

	A	B	C	D	E	F	G	H
1	商品级别	本级编号	商品编号	本级名称	本级简称	商品全称	单位	单价
2	2级	02	0102	啤酒	PJ	啤酒	个	100
3	2级	03	0103	葡萄酒	PTJ	葡萄酒	个	100
4	2级	01	0301	炖菜类	DC	炖菜类	个	100
5	2级	02	0302	炒菜类	CC	炒菜类	个	100
6	2级	01	0401	米饭	MF	米饭	个	100
7	2级	02	0402	水饺	SJ	水饺	个	100
8	2级	03	0403	馅饼	XB	馅饼	个	100
9	2级	03	0303	特色菜	CC	特色菜	个	100

图 24.6 自动创建的 Excel 文件

下面讲述一下本实例的开发步骤:

(1) 新建一个标准工程, 创建一个新窗体, 默认的“名称”属性为 Form1, 将其 Caption 属性设置为“将数据导出到 Excel 文件”。

(2) 在 Form1 窗体上添加一个 Frame 控件数组、一个 ADO 控件、一个 DataGrid 控件、3 个 CommandButton 控件和一个 OptionButton 控件数组。其中 ADO 控件和 DataGrid 控件不是工具箱中默认的控制, 需要自行添加, 添加的步骤是:

在工程中选择“工程”/“部件”, 打开“部件”对话框, 在此选中 Microsoft ADO Data Control 6.0 (SP4) (OLEDB) 和 Microsoft DataGrid Control 6.0 (SP5) (OLEDB) 复选框, 单击“确定”按钮。此时, ADO 控件和 DataGrid 控件将出现在工具箱中。

(3) 设置 3 个 CommandButton 控件的 Name 属性分别为 cmdFind、cmdExport 和 cmdExit, Caption 属性分别为“查询”、“导出到 Excel”和“退出”。

(4) 按照上图添加 TextBox 控件和 Label 控件, 同时设置他们的 Caption 属性。

(5) 按照前面介绍的方法引用 Excel 对象。

(6) 打开代码编辑器窗口, 输入以下代码。

'此处实现窗体初始化和查询功能的代码省略, 详细代码参见光盘源程序

```
Private Sub cmdExport_Click()
```

```
    Dim i As Integer, r As Integer, c As Integer
```

```
    Dim newxls As New Excel.Application
```

```
    Dim newbook As New Excel.Workbook
```

```
'声明局部变量
```

```
'声明 Excel 的 Application 对象的实例
```

```
'声明 Excel 的 Workbook 对象的实例
```

Dim newsheet As New Excel.Worksheet	声明 Excel 的 Worksheet 对象的实例
Set newbook = newxls.Workbooks.Add	'创建工作簿
Set newsheet = newbook.Worksheets(1)	'创建工作表
If Adodc1.Recordset.RecordCount > 0 Then	'如果 ADO 控件的记录集大于 0
For i = 0 To DataGrid1.Columns.Count - 1	'循环添加表头
newsheet.Cells(1, i + 1) = DataGrid1.Columns(i).Caption	'Excel 的表头为 DataGrid 控件的表头
Next i	
'指定表格内容	
Adodc1.Recordset.MoveFirst	'把 ADO 控件的指针移到第一条记录
Do Until Adodc1.Recordset.EOF	'复制所有的记录
r = Adodc1.Recordset.AbsolutePosition	'指定 Recordset 对象当前记录的序号位置
For c = 0 To DataGrid1.Columns.Count - 1	'循环读取每一行的数据
DataGrid1.Col = c	'读取第 c 列的数据
newsheet.Cells(r + 1, c + 1) = DataGrid1.Columns(c)	'复制单元格的内容
Next c	
Adodc1.Recordset.MoveNext	'ADO 控件的指针移到下一条记录
Loop	
newxls.Visible = True	'显示创建的 Excel 应用文件
End If	
End Sub	
Private Sub Option1_Click(Index As Integer)	
Dim i As Integer	'声明局部变量
For i = 0 To 2	'循环设置 Frame 框架
If Option1(i).Value = True Then	'如果单选按钮被选中
Frame1(i).Enabled = True	'Frame 控件可用
Else	'如果单选按钮没用被选中
Frame1(i).Enabled = False	'Frame 控件不可用
End If	
Next	
End Sub	

## 24.4 小结

本章介绍了 OLE 控件的常用属性、方法、事件，利用 VB 控制 Word 和 Excel。通过本章的学习，读者应该学会如何使用 OLE 控件，重点掌握通过 VB 应用程序创建、使用 Word 和 Excel 对象。

## 24.5 练习与实践

1. 创建一个应用程序，可以在 OLE 控件中动态创建一个 Excel 对象，同时还可以打开已经保存过的 OLE 对象，并可以对创建或者打开的 OLE 控件中的对象进行编辑，运行结果如图 24.7 所示。（答案位置：光盘\TM\sl\24\7）



图 24.7 OLE 控件应用

2. 使用程序的过程中,经常需要把一些文本文件导出到 Word 文档中保存,创建一个应用程序,单击“导出到 Word”按钮,可以把文本框中的内容导入到 Word 中,运行界面如图 24.8 所示。(答案位置:光盘\TM\sl\24\8)



图 24.8 导出到 Word 文档

3. 有时候文件夹中的文件非常多,查找起来非常的麻烦,如果把他们的名称存放到 Excel 表格中,查找起来就会方便很多。编写一个程序,提取文件夹中的文件,并把他们导入到 Excel 文件中。程序运行界面如图 24.9 所示。(答案位置:光盘\TM\sl\24\9)

4. 有时候创建完成 Word 文档以后需要知道 Word 文档的页码,试编写一个程序,批量地统计指定文件夹中 Word 文档的页码,并将统计结果显示在 Excel 表格中。程序运行界面如图 24.10 所示。(答案位置:光盘\TM\sl\24\10)

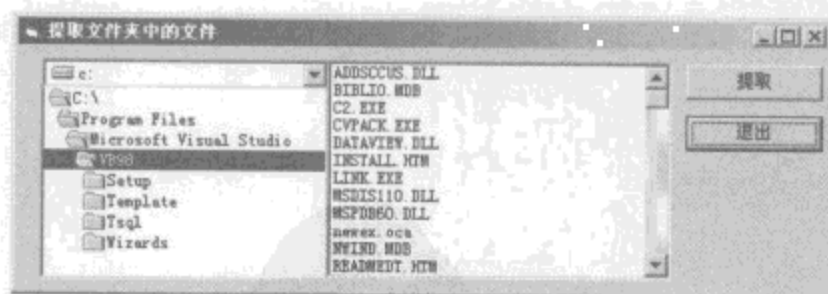


图 24.9 提取指定文件夹中的文件

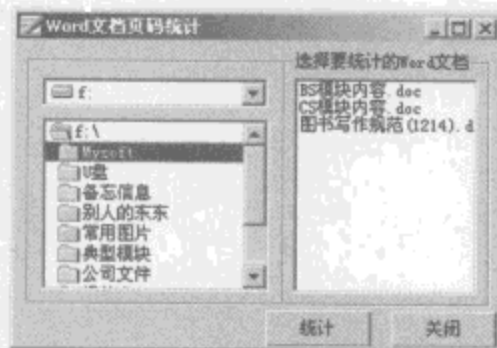


图 24.10 Word 文档页码统计





# 第25章

## 创建和使用帮助文件

(教学录像: 40 分钟)

不管一个应用程序编写得如何简单实用, 用户还是会对程序中的某一部分的使用提出疑问。在这样的情况下, 如果编程人员能为用户提供联机参考文档, 即可以帮助用户快速熟悉和使用此程序, 从而使作品更加完美实用。这种联机帮助文档在使用时, 可以减少用户去翻阅参考资料的时间, 同时也可以大大加强获得帮助信息的效率, 最终为用户提供一个快速高效的参考平台。本章详细地介绍了如何制作帮助文件以及在 Visual Basic 中调用帮助文档的 5 种方法。

通过阅读本章, 您可以:

- » 掌握制作 HLP 帮助文件的方法
- » 掌握制作 CHM 帮助文件的方法
- » 学会在应用程序中使用〈F1〉键调用帮助
- » 学会在应用程序中使用 SENDKEYS 方法调用帮助
- » 学会在应用程序中使用 SHELL 函数调用帮助
- » 学会在应用程序中使用 HTMLHELP 函数调用帮助
- » 学会在应用程序中使用 SHELLEXECUTE 函数调用帮助

## 25.1 Help Workshop 帮助文件

 教学录像：光盘\TM\lx\25\Help Workshop 帮助文件.exe

Help Workshop 是一款可以将 RTF 文档编译为 HLP 帮助文件的软件。此软件除了可以制作帮助文件外，还可以编辑帮助文件的项目文件、帮助文件的内容文件，以及测试和输出帮助报告文件。

Help Workshop 软件还可以将帮助文件的文字信息和位图，以及其他相关的帮助信息资料文件编译成其他格式的文件，然后使用 Microsoft Windows Help 程序就可以浏览这些帮助信息。

本节将介绍如何使用 Help Workshop 软件制作扩展名为.HLP 的帮助文件。

### 25.1.1 安装 Help Workshop 帮助文件

安装 Microsoft Help Workshop 软件的操作步骤如下：

(1) 双击 VB 安装光盘中 Common\Tools\VB\HCW 目录下的 setup.exe 文件，在弹出的对话框中单击“下一步”按钮，这时会弹出欢迎窗口。

(2) 单击欢迎窗口中的 Next 按钮，继续进行下一步安装。

(3) 通过单击 Choose Destination Location 窗口中的 Browse 按钮，选择 Microsoft Help Workshop 所要安装的路径，这里采用默认的安装路径，如图 25.1 所示。然后单击 Next 按钮，继续进行下一步安装。

(4) 在弹出的 Setup Type 窗口中选择 Microsoft Help Workshop 的安装类型，其安装类型主要包括：Typical（典型安装）、Compact（简洁安装）和 Custom（自定义安装）。这里选择 Typical 选项，如图 25.2 所示。然后单击 Next 按钮，开始安装 Microsoft Help Workshop。

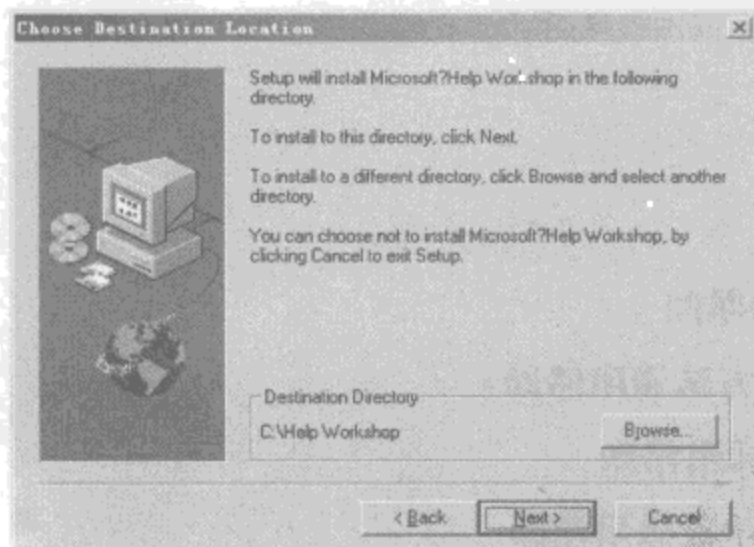


图 25.1 Choose Destination Location 窗口

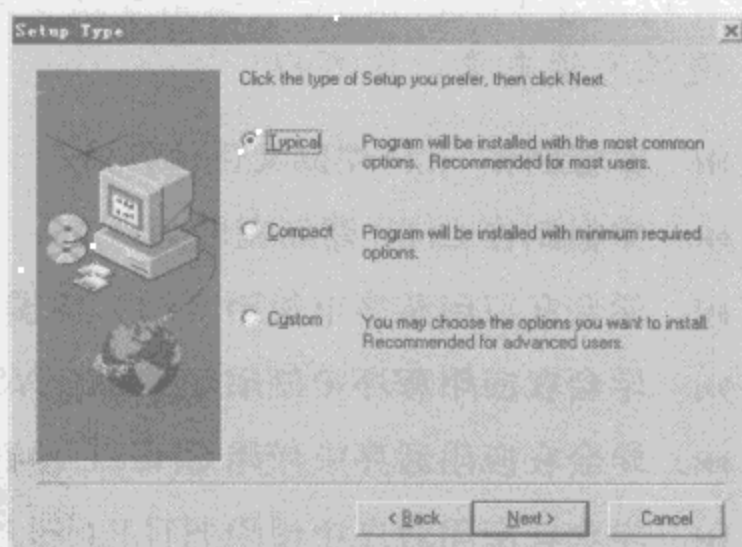


图 25.2 Setup Type 窗口

(5) 文件安装完成后，在弹出的对话框中，单击“确定”按钮，即可完成 Microsoft Help Workshop 安装。Microsoft Help Workshop 软件安装完成以后，在“开始”菜单中只有该软件的卸载程序，要运行 Microsoft Help Workshop 软件，需执行其安装目录下的 hcrtf.exe 文件。

### 25.1.2 编辑 RTF 文件

RTF 文档可以由微软公司的 Word 来创建。制作 RTF 文档主要步骤如下:

(1) 首先新建一个 Word 文档, 并撰写程序的使用说明书。撰写完成以后, 选择菜单栏中“文件”/“另存为”命令, 打开“另存为”对话框, 然后在“保存类型”下拉列表框中选择“RTF 格式 (\*.rtf)”, 如图 25.3 所示, 同时选择保存路径并输入一个文件名, 单击“保存”按钮, 即可创建一个 RTF 格式的文档。

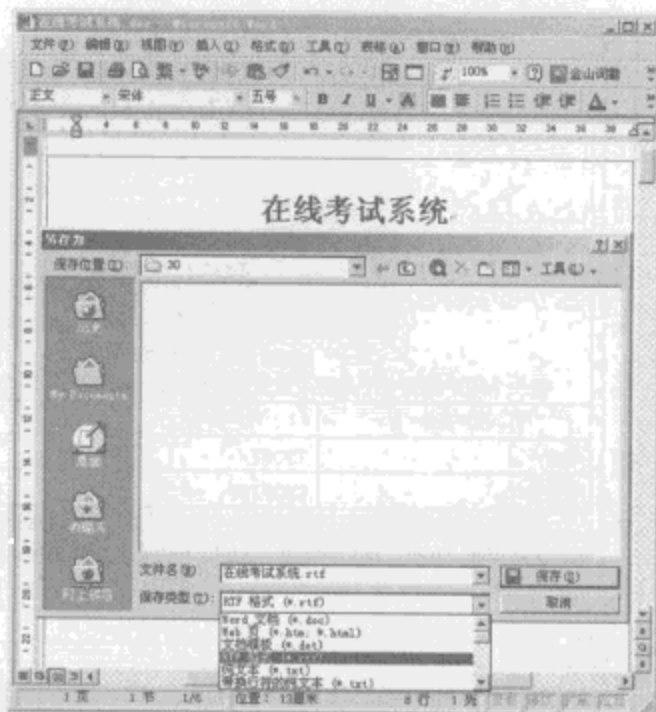
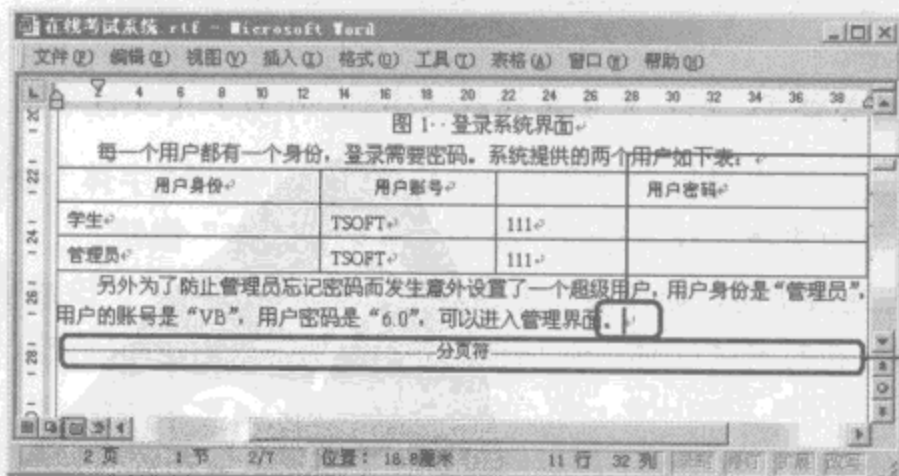


图 25.3 另存为.rtf 格式

(2) 单击工具栏中的“显示/隐藏”按钮 。这样做的目的是将文档中所有的编辑标记都显示出来, 以便于编辑 RTF 文件。

(3) 将输入光标置于某一主题内容的结尾处。单击菜单栏中的“插入”/“分隔符”命令, 将弹出“分隔符”对话框, 选择该对话框中选择“分页符”单选按钮, 单击“确定”按钮, 即可完成分页符的插入操作, 插入分页符后的效果如图 25.4 所示。



① 将输入光标置于此处, 单击菜单栏中的“插入”/“分隔符”命令。选择“分页符”单选按钮, 单击“确定”按钮, 即可完成分页符的插入操作

② 插入分页符后的效果

图 25.4 设置分页之后的 RTF 文档



(4) 将光标置于某一主题标题的左侧，单击菜单栏中的“插入”/“脚注和尾注”命令，弹出“脚注和尾注”对话框。在“插入”区域中选择“脚注”单选按钮；在“编号方式”区域中选择“自定义标记”单选按钮，然后在后面的文本框中输入一个自定义脚注符号“#”，单击“确定”按钮，在页面的下端输入对应的主题 ID 号，即可完成脚注的添加。最终插入效果如图 25.5 所示。

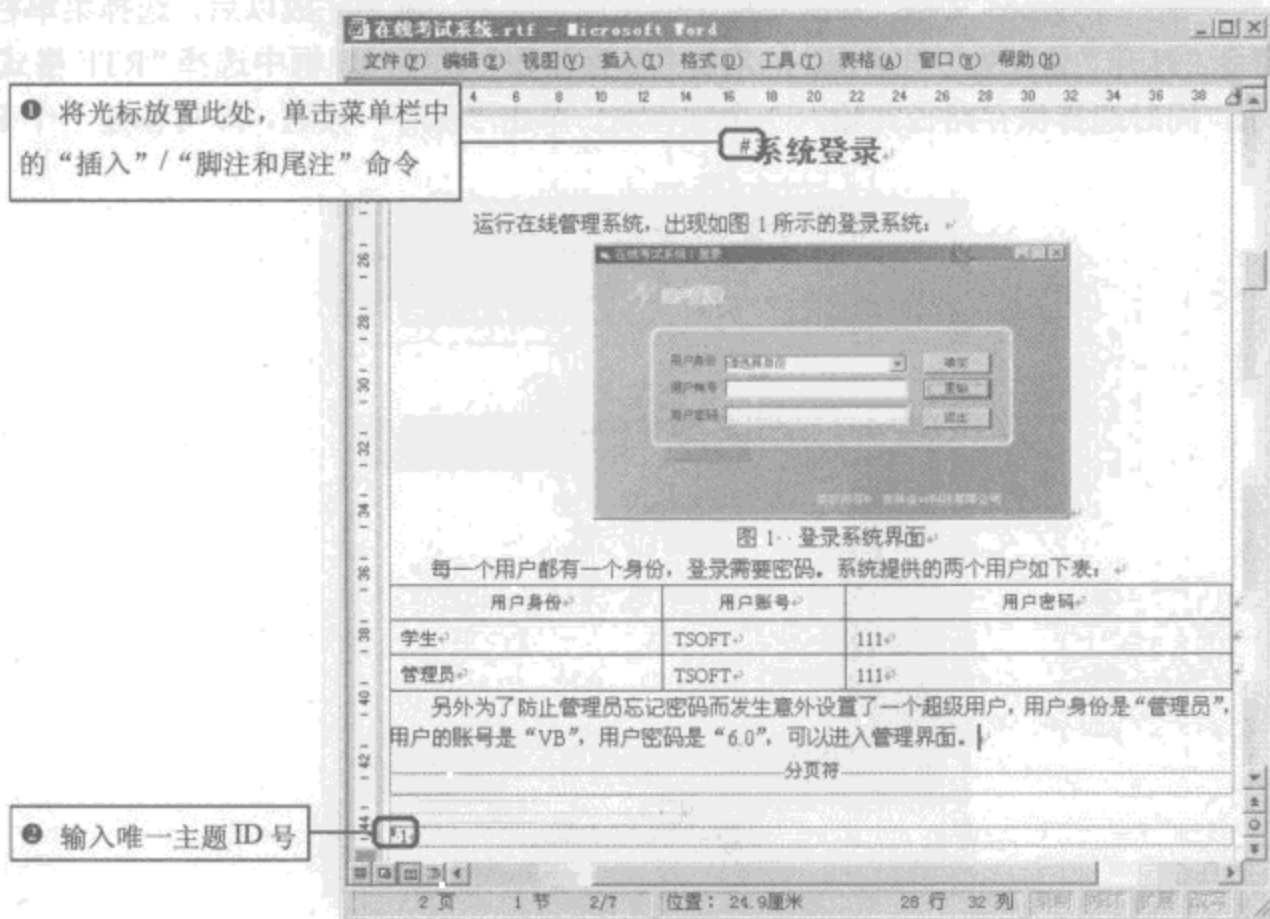


图 25.5 设置脚注

**说明：**在创建 RTF 文档时，会使用到许多“自定义脚注”。常用的“自定义脚注”标识有以下 5 种。

- ①#号：标注上下文字符串，定义一个唯一能标识主题页的字符串。主要实现主题页的链接跳转。
- ②\$号：标注主题标题，通过该脚注可以知道本专题的具体内容，并实现超链接到跳转页。
- ③\*号：主题页建立标志，通过建立标志的真假条件来选择包括或排除主题页。
- ④+号：指出主题的浏览顺序。
- ⑤K（大写）：标注主题页的关键字，将一个主题标题链接到一个或多个关键字上。

(5) 选择将要进行连接的主题标题，单击鼠标右键在弹出的快捷菜单中选择“字体”命令，在“字体”对话框中的“下划线”下拉列表框中选择“双下划线”类型，单击“确定”按钮。

(6) 在需要的主题标题后面设置连接字符，如在某一主题标题后添加一个数字，该数字即是与其所对应内容页的脚注编号。选中连接字符打开“字体”对话框，选中“隐藏文字”复选框，单击“确定”按钮，将其设置为隐藏文字。

(7) 重复步骤 (5) 和步骤 (6) 的操作，为其他主题标题设置连接字符，最终效果如图 25.6 所示。

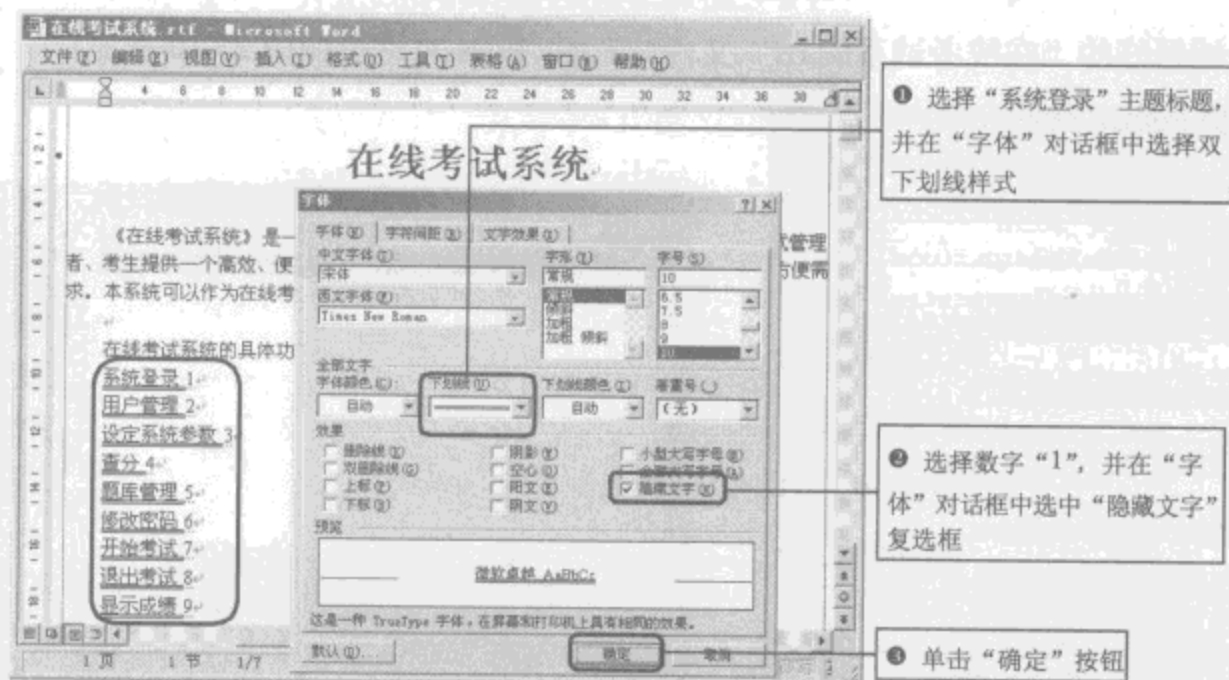


图 25.6 设置连接字符

### 25.1.3 制作简单的帮助文件

**例 25.1** 使用 Help Workshop 软件将 RTF 文件编译成帮助文件，具体步骤如下：（实例位置：光盘 \TM\sl\25\1）

（1）运行 Microsoft Help Workshop 安装目录下的 hcrtf.exe 文件，在弹出的操作界面中选择菜单栏中的 File/New 命令，将弹出 New 对话框，如图 25.7 所示。

（2）选择 New 对话框中的 Help Project 选项，单击 OK 按钮，将弹出 Project File Name 对话框。在该对话框中输入.hlp 文件的文件名（如输入 Help），同时单击工具栏中的“保存”按钮，进入如图 25.8 所示的 Microsoft Help Workshop-[help.hpj]对话框。

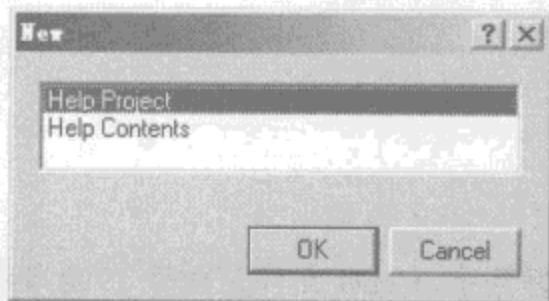


图 25.7 New 对话框

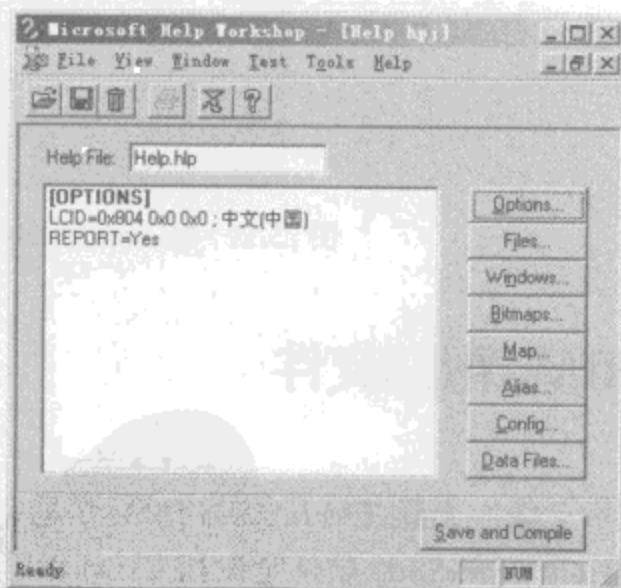


图 25.8 Microsoft Help Workshop-[Help.hpj]对话框

（3）单击图 25.8 中的 Options 按钮，将弹出 Options 对话框，如图 25.9 所示。选择 Options 对话框中的 Files 选项卡，单击 Change 按钮，将弹出 Topic Files 对话框，单击 Topic Files 对话框中的 Add

按钮来添加已编辑好的“在线考试系统.rtf”文件。Topic Files 对话框如图 25.10 所示。

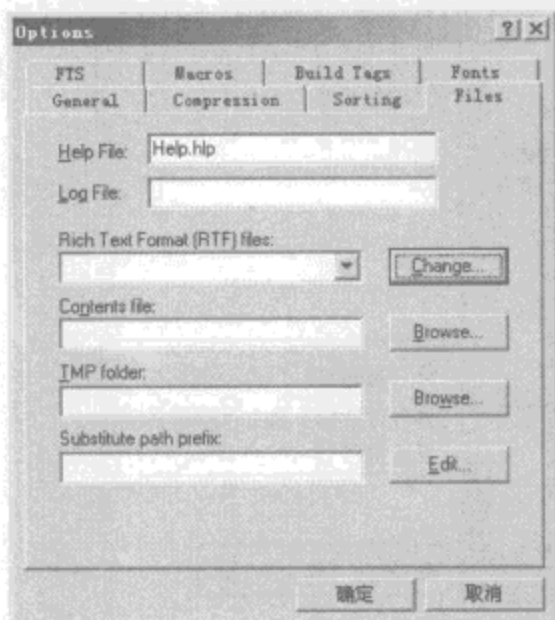


图 25.9 Options 对话框

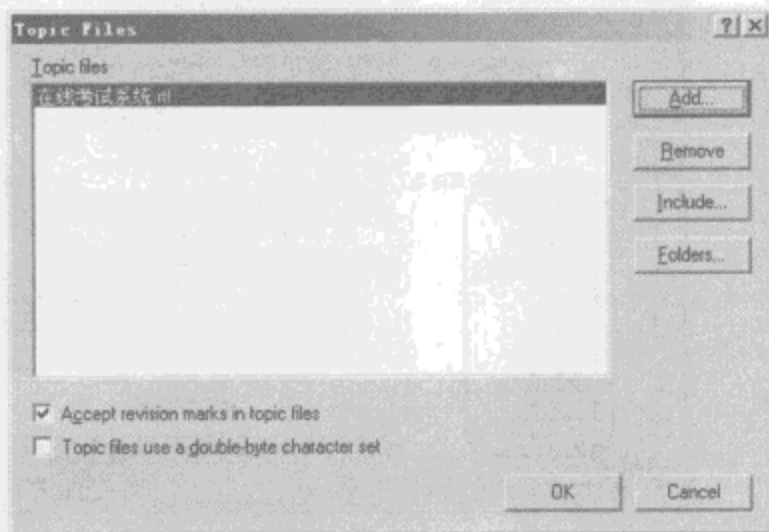


图 25.10 Topic Files 对话框

(4) 单击图 25.10 中的 OK 按钮, 返回到 Microsoft Help Workshop-[Help.hpj]对话框中, 如图 25.11 所示。单击 Save and Compile 按钮, 开始编译 Help.hlp 文件。

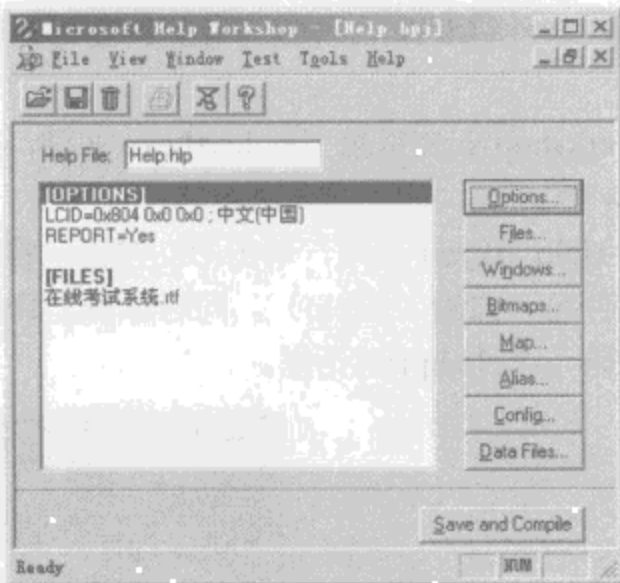


图 25.11 Microsoft Help Workshop-[Help.hpj]对话框

## 25.2 HTML 帮助文件

 教学录像: 光盘\TM\lx\25\HTML 帮助文件.exe

HTML Help WorkShop 软件是制作 CHM 帮助文件的好帮手。CHM 文件是微软 1998 年推出的基于 HTML 文件特性的帮助文件系统, 被 Internet Explorer 支持的 JavaScript、VBScript、ActiveX、Java Applet、Flash、Html 图像文件 (GIF、JPEG、PNG) 和音频视频 (AU、MIDI、WAV、AVI) 等文件, CHM 同样支持, 并且可以通过 URL 地址与因特网联系在一起。CHM 帮助文件是使用 HTML (超文本标记语

言)编写的,其扩展名为.html或.htm。CHM 帮助文件也可以通过 Word 编写正常的文档,再将其另存为 HTML 文件,这样文档会自动转换为 HTML 的源码。制作 CHM 帮助文件主要是因为此文件占用空间较小,方便网络传输,因此得到了广泛传播。

### 25.2.1 安装 Microsoft HTML Help Workshop

安装 Microsoft HTML Help Workshop 软件的操作步骤如下:

- (1) 执行 VB 安装盘中 HTMLHELP 文件夹内的 HTMLHELP.EXE 程序安装文件。
- (2) 在弹出的 License Agreement 窗口中单击 Accept 按钮接受安装许可协议,进行下一步安装。
- (3) 在弹出的 Welcome 窗口中单击 Next 按钮,进行下一步安装。
- (4) 在弹出的 Select Destination Folder 窗口中单击 Browse 按钮,在弹出的对话框中选择 Microsoft HTML Help Workshop 的安装路径后,单击 OK 按钮。此时 Select Destination Folder 窗口中的 Location 文本框中将显示刚刚所选择的安装路径,如图 25.12 所示。单击 Next 按钮,进行下一步操作。
- (5) 在弹出的 Choose Type of Setup 窗口中选择安装类型,选择 Complete (完全安装) 单选按钮,如图 25.13 所示。单击 Next 按钮,进行下一步操作。

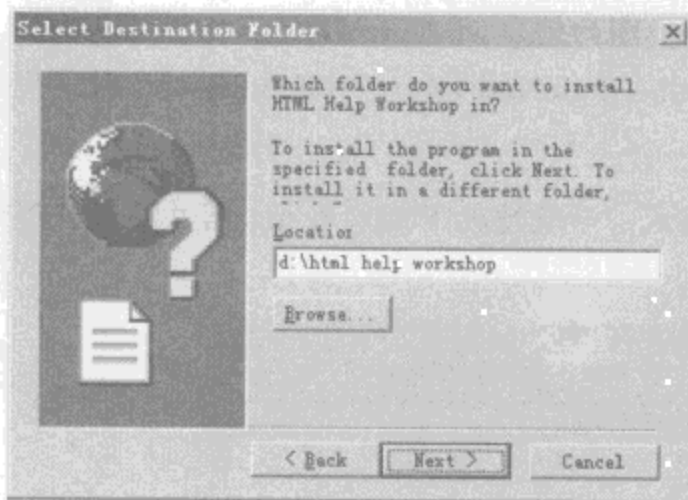


图 25.12 Select Destination Folder 窗口

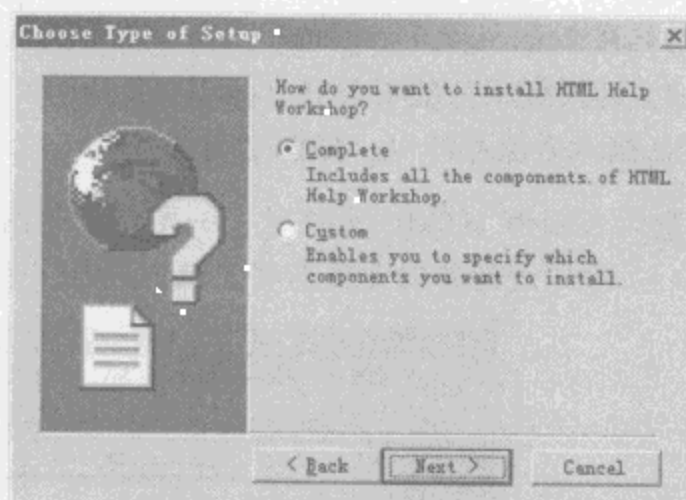


图 25.13 Choose Type of Setup 窗口

(6) 在弹出的 Select Program Item Group 窗口中选择 Create a new submenu call (在“开始”菜单中创建一个新的子项) 单选按钮,单击 Next 按钮,进行下一步操作。Select Program Item Group 窗口如图 25.14 所示。

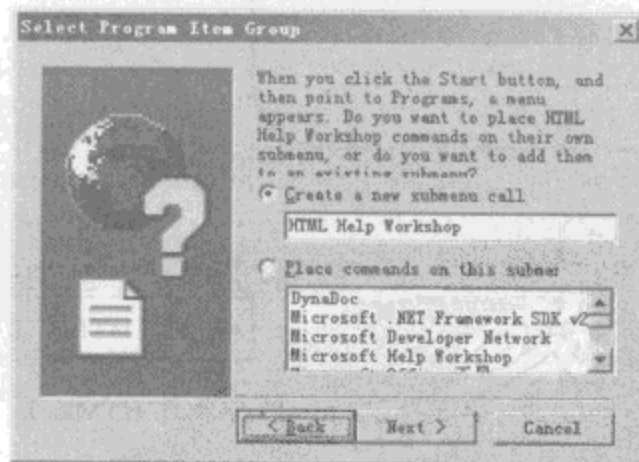


图 25.14 Select Program Item Group 窗口



(7) 在弹出的 Personalize HTML Help Workshop 窗口中输入名称 HTML Help Workshop 后, 单击 Next 按钮, 开始安装 Microsoft HTML Help Workshop 软件。安装完成后会弹出对话框提示安装成功, 单击 OK 按钮即可完成安装。

### 25.2.2 编辑 HTML 文件

编辑 HTML 文件首先需要撰写程序的说明书, 即主要功能和相应问题。一般情况下, 撰写的程序使用说明书, 都是利用 Word 编写的, 并且将其另存为 Web 页 (.htm 格式文件), 因为在使用 HTML Help Workshop 软件编译帮助文件的时候需要使用 .htm 格式的文件。帮助主题文件是将帮助文件中应用的所有元素都联系在一起, 如项目文件、项目目录文件、图像文件和索引文件等。具体编辑 HTML 文件的步骤如下:

(1) 首先新建一个 Word 文档, 并撰写程序的使用说明书, 撰写完成以后, 选择菜单栏中“文件”/“另存为 Web 页”命令。

(2) 找到刚刚保存过的 HTML 文件, 并单击鼠标右键, 在弹出的快捷菜单中选择“打开方式”, 在其下拉菜单中选择 Microsoft FrontPage 软件。

(3) 在 Microsoft FrontPage 软件中重新插入并替换图片信息。

(4) 切换到下方的 HTML 页, 在 HTML 页中将 `<!--[if gte vml 1]>.....<![endif]-->` 及灰色代码里的全部信息删除后保存即可。如图 25.15 所示。

**注意:** 在制作 HTML 文件时, 一定要注意图片格式, 一般情况下都采用 .jpg 图片格式, 否则在制作完成的帮助文件后将无法显示图片信息。




图 25.15 Microsoft FrontPage 软件 HTML 页

## 25.2.3 制作目录和目录文件

例如, 编辑完 HTML 文件以后, 使用 HTML Help Workshop 软件对 HTML 文件进行编译, 编译以后生成的.chm 文件即为需要的帮助文件, 其具体步骤如下:

## (1) 创建工程

①单击“开始”按钮, 选择“程序”/HTML Help Workshop/HTML Help Workshop 选项, 启动 HTML Help Workshop 软件, 单击菜单栏中的 File/New 命令, 或者直接单击工具栏上的新建  按钮, 打开 New 对话框, 选择列表框中的 Project 选项, 如图 25.16 所示。

单击 OK 按钮, 开始创建工程。

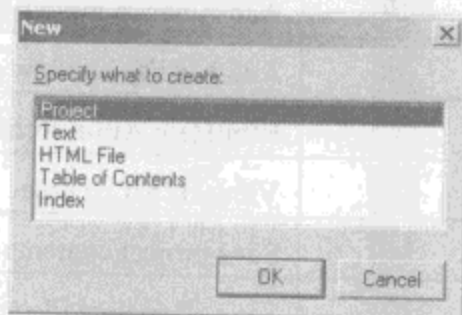


图 25.16 New 对话框

New 对话框中列表框内的各选项类型如下。

☒ Project: 工程文件。

☒ Text: 文本文件。

☒ HTML File: HTML 文件。

☒ Table of Contents: 目录文件。

☒ Index: 索引文件。

②在 New Project 窗口中, 默认条件下单击“下一步”按钮, 弹出 New Project—Destination 窗口, 单击 Browse 按钮, 在弹出的“打开”对话框中设置要创建工程文件的路径及文件名, 工程文件的扩展名为.hhp。

③在 New Project—Existing Files 窗口中设置是否将现有的目录文件 (.hhc)、索引文件 (.hhk) 和主题文件 (.htm) 添加到项目中。如果选中这些复选框, 则需要指定该文件的名称和位置, 在此不选择这些选项, 如图 25.17 所示。单击“下一步”按钮, 在弹出的窗口中单击“完成”按钮, 完成创建工程文件。新创建的工程如图 25.18 所示。

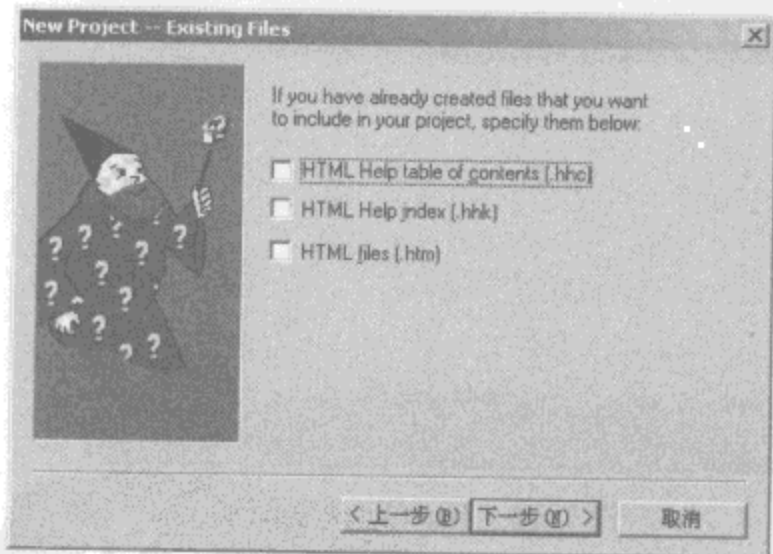


图 25.17 New Project—Existing Files 窗口

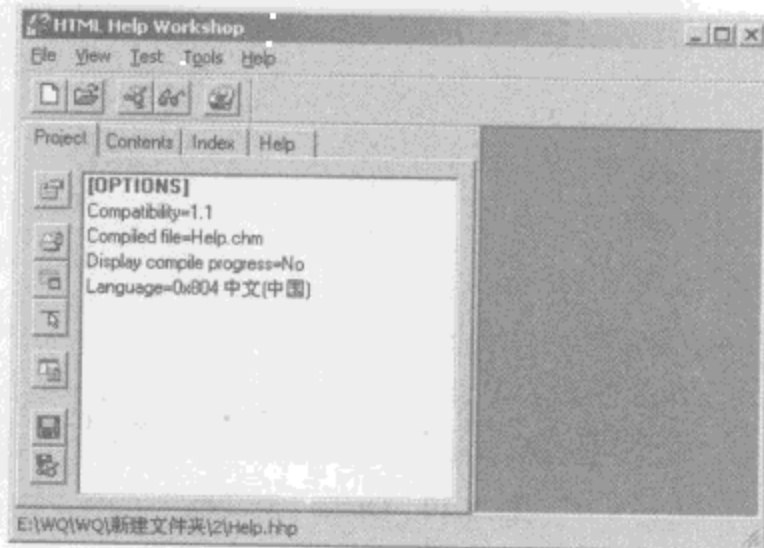









图 25.18 HTML Help Workshop 项目窗口


在工程窗口的 Project 选项卡的左侧有 7 个功能按钮, 主要功能说明如表 25.1 所示。


表 25.1 Project 选项卡的功能按钮

图 标	名 称	说 明
	Change project options	改变项目选项
	Add/Remove topic files	添加/删除主题文件
	Add/Modify window definitions	添加/修改、定义窗口
	HtmlHelp API information	HtmlHelp API 信息
	View HTML source	查看 HTML 源文件
	Save project, contents and index files	保存项目、目录和索引文件
	Save all project file and compile	保存和编译所有文件

## (2) 创建项目文件

完成创建工程文件后，就可以开始设置帮助文件了。创建项目文件步骤如下：

①单击左侧的 Change project options 按钮，打开 Options 对话框，如图 25.19 所示。选择 General 选项卡，在 General 选项卡的 Title 编辑框中输入标题“小区物业管理系统”，单击“确定”按钮。

②添加主题文件，单击左侧的 Add/Remove topic files 按钮，弹出 Topic Files 对话框，如图 25.20 所示。单击 Add 按钮，弹出“打开”对话框，选择“小区物业管理系统.htm”文件添加到工程中（前提是 HTML 文件必须存在，否则编译的帮助文件运行后，系统会报错），然后单击 OK 按钮，完成主题文件的添加操作。

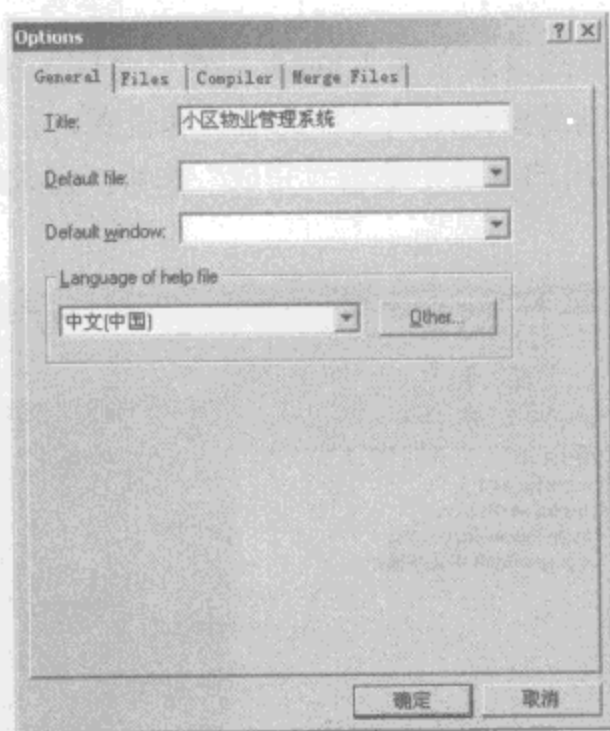


图 25.19 Options 对话框

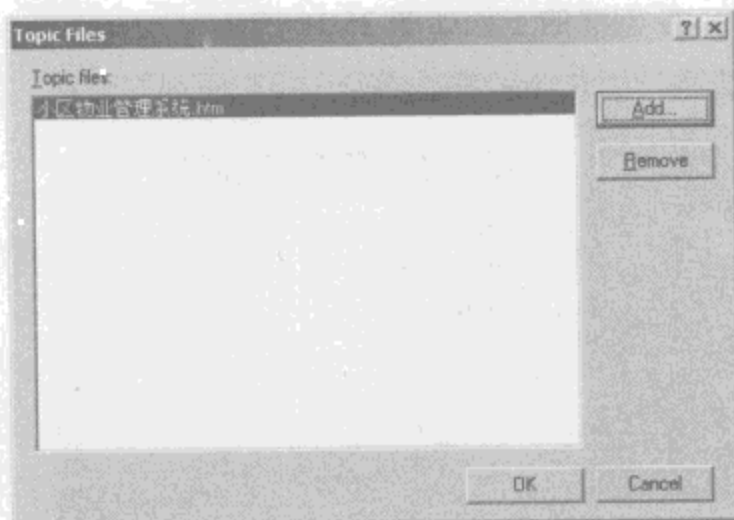


图 25.20 Topic Files 对话框

## (3) 创建项目目录

①选择 HTML Help Workshop 工程窗口中的 Contents 选项卡，弹出 Table of Contents Not Specified 提示对话框，选择 Create a new contents file 单选按钮，单击 OK 按钮创建目录文件，如图 25.21 所示。将目录文件保存到工程目录下，目录文件保存对话框如图 25.22 所示。

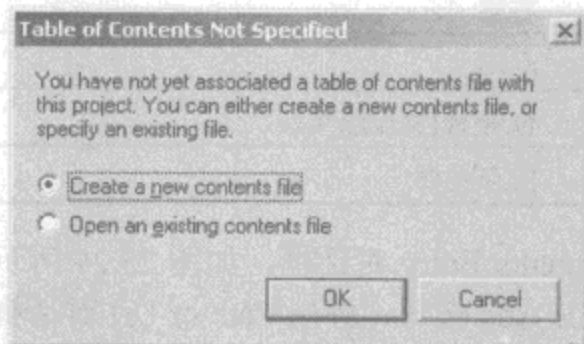


图 25.21 Table of Contents Not Specified 对话框

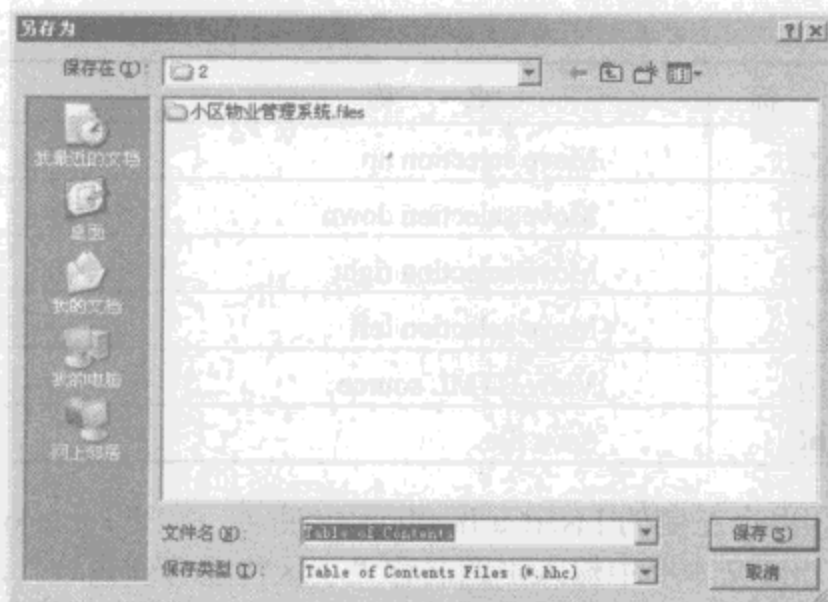


图 25.22 目录文件保存对话框

②创建目录文件后的 Contents 选项卡，如图 25.23 所示。Contents 选项卡的左侧有 11 个按钮，其主要功能说明如表 25.2 所示。

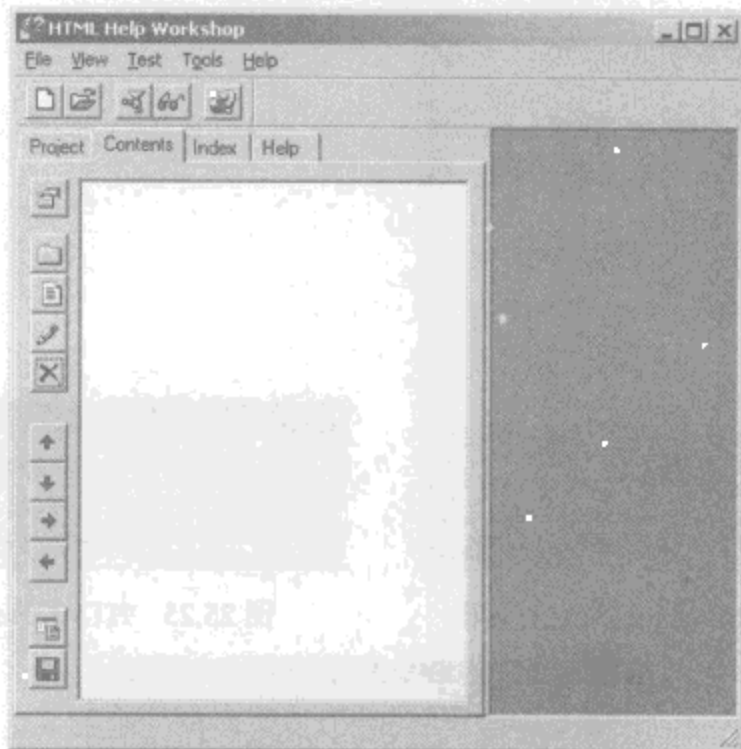









图 25.23 创建目录文件后的 Contents 选项卡


表 25.2 Contents 选项卡中的功能按钮说明

图 标	名 称	说 明
	Contents properties	目录属性
	Insert a heading	插入标题
	Insert a page	插入主题
	Edit selection	编辑
	Delete selection	删除



图 标	名 称	说 明
	Move selection up	上移
	Move selection down	下移
	Move selection right	右移
	Move selection left	左移
	View HTML source	查看 HTML 源文件
	Save file	保存文件

③单击左侧的 Insert a heading 按钮, 弹出 Table of Contents Entry 对话框, 如图 25.24 所示。在 Entry title 文本框中输入主题名称“小区物业管理系统”, 单击 Add 按钮, 弹出 Path or Url 对话框, 在该对话框中单击 Browse 按钮, 弹出“打开”对话框, 选择要添加的相关主题文件。然后通过图 25.24 中的 Add/Edit 按钮连接主题文件, 单击“确定”按钮完成添加。

④单击 Insert a page 按钮, 弹出 HTML Help Workshop 对话框, 询问是否继续添加操作, 如图 25.25 所示, 单击“是”按钮。重复步骤③的操作, 依次添加所有文件目录。创建完成后的目录文件如图 25.26 所示。

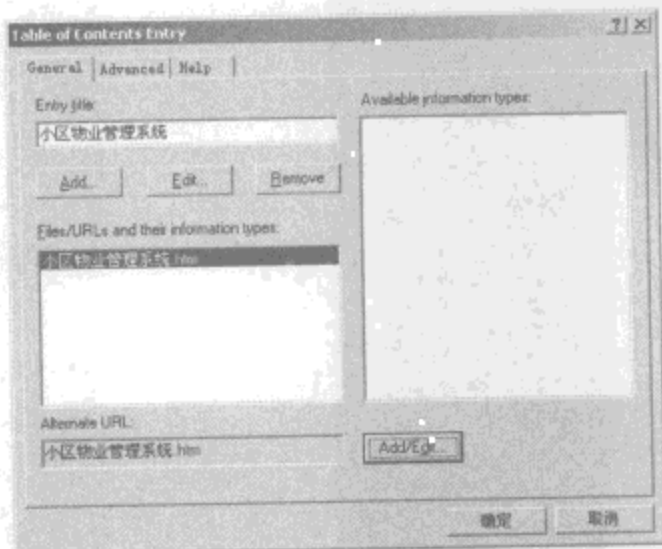


图 25.24 Table of Contents Entry 对话框

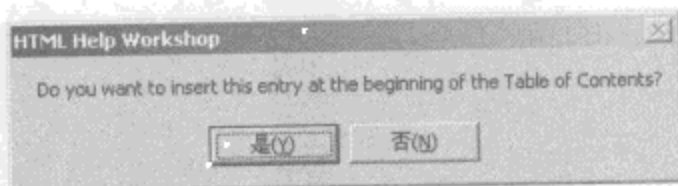


图 25.25 HTML Help Workshop 对话框

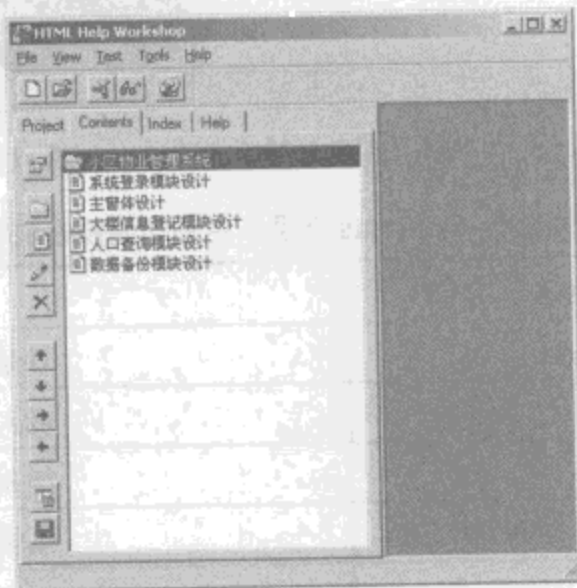



图 25.26 创建目录文件后的 Contents 选项卡

### 25.2.4 制作 Chm 文件

**例 25.2** 制作完目录及目录文件还不算最终的目标, 因为此时的工程并没有编译生成.chm 文件。本节将介绍如何生成.chm 文件的制作过程。(实例位置: 光盘\TM\sl\25\2)

(1) 经过以上的操作后, 还要将创建完的目录文件添加到工程文件中。在 Project 选项卡中, 单击 Add/Modify window definitions 按钮, 弹出 Add a New Window Type 对话框, 如图 25.27 所示。在编辑框中自定义显示窗口的名称, 单击 OK 按钮, 在弹出的 Window Types 对话框中设置帮助文件窗口的显示风格, 如图 25.28 所示。

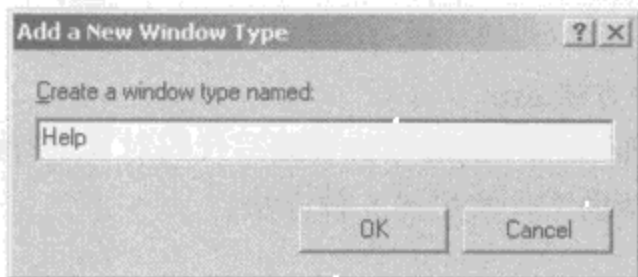


图 25.27 Add a New Window Type 对话框

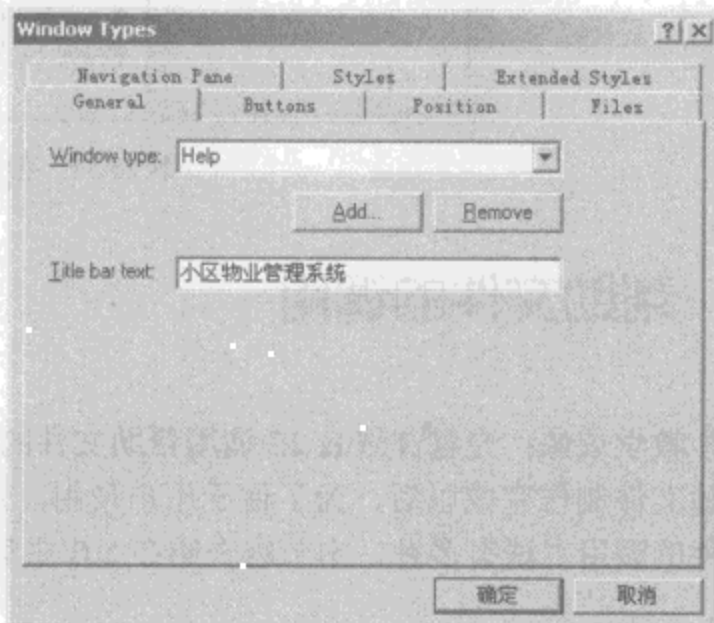


图 25.28 Window Types 对话框

(2) 设置完帮助文件窗口的显示样式以后, 就可以开始编译文件了。在工程窗口的菜单栏中选择 File/Compile 命令, 弹出 Create a Compiled file 对话框, 选择要编译文件的路径, 如图 25.29 所示, 默认为工程路径下。单击 Compile 按钮, 弹出提示对话框提示是否要编译工程, 单击“是”按钮, 开始编译文件, 如图 25.30 所示。

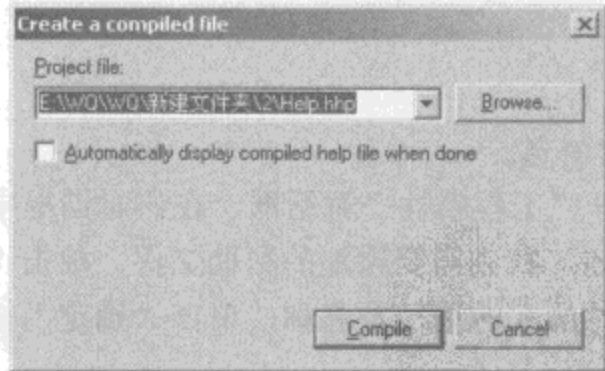


图 25.29 Create a Compiled file 对话框

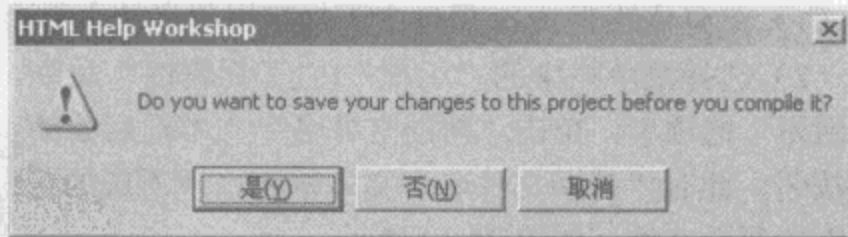


图 25.30 选择是否要编译工程

(3) 文件编译成功后, 在工程窗口的右侧窗格中将显示帮助文件的路径。在工程的根目录下找到经过编译的帮助文件 Help.chm, 双击打开, 如图 25.31 所示。

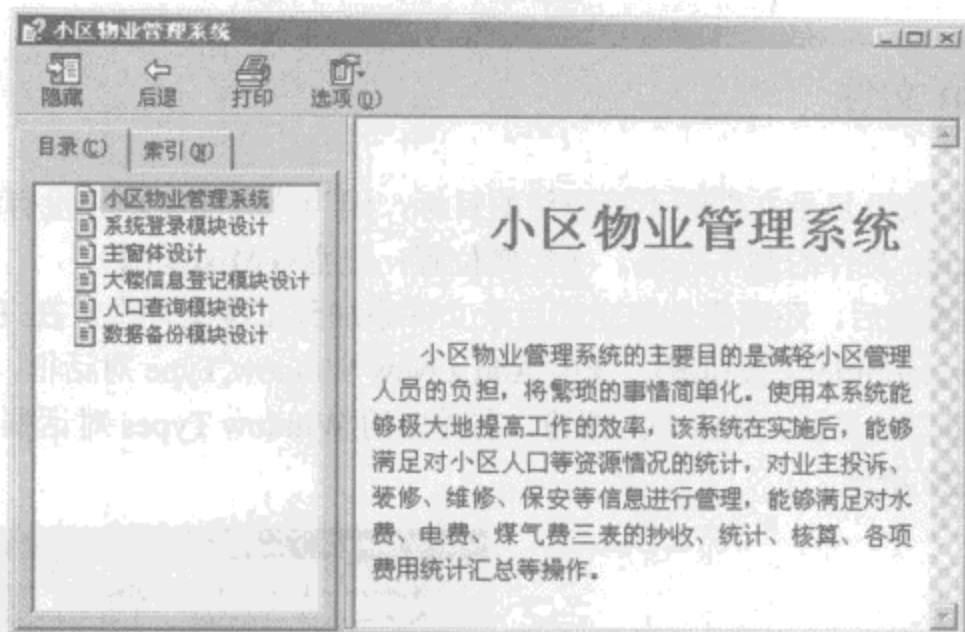


图 25.31 帮助文件效果

## 25.3 帮助文件的调用

教学录像：光盘\TM\lx\25\调用帮助文件的 5 种方法.exe

帮助文件制作完成以后，为了便于用户使用，还需要在程序的适当位置编写代码对其进行调用。帮助文件的调用方法有多种，本节将介绍在 VB 中调用帮助文件常用的 5 种方法。

### 25.3.1 使用〈F1〉键调用帮助

快捷键对于编程人员来说并不陌生，通过快捷键可以使非常繁琐的工作变得简单快捷，快捷键也在软件开发过程中得到广泛的应用。为了方便用户能够快速得到帮助信息，可以将〈F1〉键设置为调用帮助文件的快捷键，用户通过按下〈F1〉键即可调用相关联的帮助信息。

**例 25.3** 在 VB 中实现使用〈F1〉键来调用帮助文件。这里分别利用两种方法来加以说明，其具体步骤如下。（实例位置：光盘\TM\sl\25\3）

☒ 方法一：

(1) 在窗体的 Picture 属性中添加应用程序的主界面图片信息。

(2) 选择菜单栏中的“工程”/“工程属性”命令，打开“工程属性”对话框，在该对话框中选择“通用”选项卡，单击“帮助文件名”文本框后面的...按钮，找到需要添加的帮助文件，单击“打开”按钮，此时“帮助文件名”文本框将出现所选的帮助文件所在的路径及名称，单击“确定”按钮即可添加帮助文件。如图 25.32 所示。

(3) 运行程序，按下〈F1〉键即可调用帮助文件。

☒ 方法二：

(1) 在窗体的 Picture 属性中添加应用程序的主界面图片信息，在“菜单编辑器”中添加应用程序的菜单栏，再在窗体上添加一个 CommonDialog 控件。

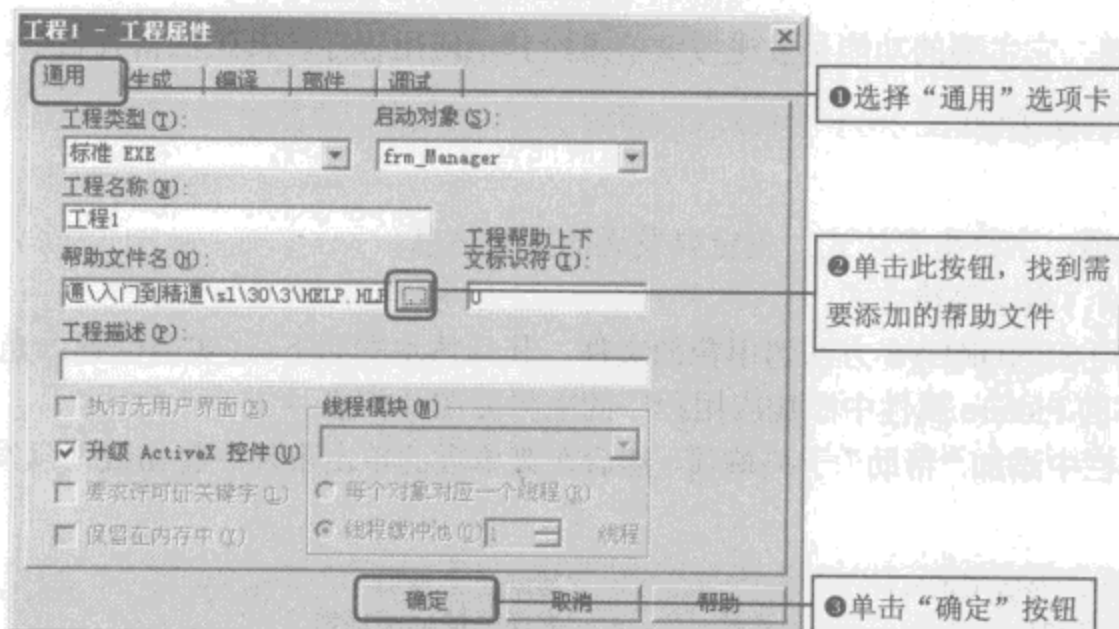


图 25.32 “工程属性”对话框

(2) 在“菜单编辑器”中，设置一个标题为“帮助”的子菜单项，在快捷键下拉列表框中选择 F1，单击“确定”按钮即可添加调用帮助文件的快捷键，如图 25.33 所示。

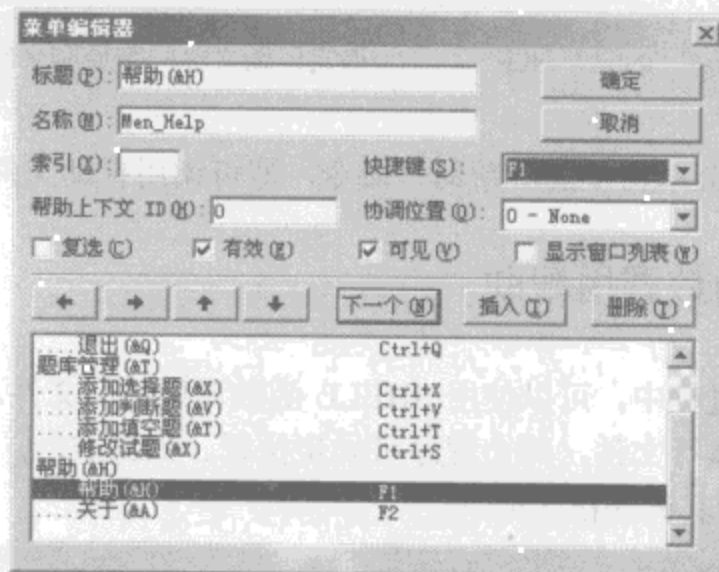


图 25.33 “菜单编辑器”对话框

(3) 在“帮助”子菜单项的单击事件下，添加如下代码。

```
Private Sub Men_Help_Click()  
    CommonDialog1.HelpFile = App.Path & "/HELP.HLP"  
    CommonDialog1.HelpCommand = cdlHelpContents  
    CommonDialog1.ShowHelp  
End Sub
```

'帮助子菜单项  
'确定帮助文件的路径和文件名  
'设置需要联机帮助文件的类型  
'显示帮助文件

(4) 运行程序，按下〈F1〉键即可调用帮助文件。

## 25.3.2 使用 SENDKEYS 方法调用帮助

在 VB 应用程序的开发过程中，可以使用 SENDKEYS 语句调用帮助文件。SENDKEYS 语句用于



激活菜单控件的键，它主要的功能是发送按键信息给其他应用程序。其语法格式如下：

**SENDKEYS** String[,Wait]

参数说明：

String: 必选项，字符串表达式，指定要发送的按键信息。

Wait: 可选项，指定等待方式。

例 25.4 使用 SENDKEYS 方法调用帮助文件，其具体步骤如下：（实例位置：光盘\TM\sl\25\4）

（1）在窗体的 Picture 属性中添加应用程序的主界面图片信息。

（2）在菜单栏中添加“帮助”子菜单项，然后在菜单项的单击事件中将击键消息发送到活动窗口，其代码如下：

```
Private Sub Men_Help_Click()           '帮助子菜单项
    SendKeys "{F1}"                   '发送击键到活动窗
End Sub
```

（3）在窗体的 Load 事件中通过 App 对象中的 HelpFile 属性来设置程序所使用的帮助文件。其代码如下。

```
Private Sub Form_Load()
    App.HelpFile = App.Path & "\HELP.HLP" '确定帮助文件的路径和文件名
End Sub
```

（4）程序运行后，单击菜单栏中的“帮助”命令，即可调用帮助文件。

### 25.3.3 使用 SHELL 函数调用帮助

在 VB 应用程序的开发过程中，可以使用 SHELL 函数调用帮助文件。SHELL 函数主要用于在程序执行的同时打开另一个可执行文件，两个程序可异步执行。如果成功，则返回代表这个程序的任务 ID；若不成功，则会返回 0。其语法格式如下：

**SHELL**(PathName[,WindowStyle])

SHELL 函数的语法中含有如表 25.3 所示的命名参数。

表 25.3 SHELL 函数的语法命名参数

参 数	描 述
PathName	必选项。String 类型（字符串）值，要执行的程序名以及任何必需的参数或命令行变量，可能还包括目录或文件夹以及驱动器
WindowStyle	可选项。Integer 类型（整数）值，表示在程序运行时窗口的样式。如果省略，则程序是以具有焦点的最小化窗口来执行的。其 WindowStyle 命名参数的值，如表 25.4 所示

表 25.4 WindowStyle 命名参数的值

常 量	值	描 述
vbHide	0	窗口被隐藏，并且焦点会移到隐式窗口
VbNormalFocus	1	窗口具有焦点，并且会还原到它原来的大小和位置

续表

常 量	值	描 述
VbMinimizedFocus	2	窗口会以一个具有焦点的图标来显示
VbMaximizedFocus	3	窗口是一个具有焦点的最大化窗口
VbNormalNoFocus	4	窗口会被还原到最近使用的大小和位置, 而当前活动的窗口仍然保持活动
VbMinimizedNoFocus	6	窗口会以一个图标来显示。而当前活动的窗口仍然保持活动

例 25.5 使用 SHELL 函数调用帮助文件。具体步骤如下: (实例位置: 光盘\TM\sl\25\5)

(1) 在窗体的 Picture 属性中添加应用程序的主界面图片信息。

(2) 在菜单栏中添加“帮助”子菜单项, 然后在菜单项的单击事件中将击键消息发送到活动窗口, 然后通过 SHELL 函数来调用此程序所使用的帮助文件, 其代码如下。

```
Private Sub Men_Help_Click()           '帮助子菜单项
    Shell "winhlp32.exe HELP.HLP", vbNormalFocus '确定帮助文件的路径和文件名
End Sub
```

### 25.3.4 使用 HTMLHELP 函数调用帮助

在前面几小节中, 我们讲解了如何在 VB 应用程序中调用 HLP 格式的帮助文件, 本节将介绍如何调用 CHM 格式的帮助文件。在 VB 应用程序的开发过程中, 可以使用 API 函数的 HTMLHELP 函数调用帮助文件, 通过 HTMLHELP 函数可以达到调用帮助文件的功能, 并对其设置帮助文件的标题、坐标、大小、窗口样式等效果。其语法格式如下:

HWND HTMLHELP (HWND hwndCaller, LPCSTR pszFile, UINT uCommand, DWORD dwData)

API 函数的 HTMLHELP 参数值如表 25.5 所示。

表 25.5 API 函数的 HTMLHELP 参数值

参 数	说 明
HwndCaller	指定一个窗口的句柄, 如果帮助文件存在, HTMLHELP 函数将消息送到这个窗口句柄中
PszFile	规定 HTML 文件, URL, 编译 HTML 文件或窗口定义
UCommand	命令动作, 补充 pszFile 和 dwData 参数值
DwData	规定任意数据, 它必须在 uCommand 参数的基础上

例 25.6 使用 HTMLHELP 函数调用帮助文件。具体步骤如下: (实例位置: 光盘\TM\sl\25\6)

(1) 在窗体的 Picture 属性中添加应用程序的主界面图片信息。

(2) 在使用 HTMLHELP 函数之前需要声明, 其声明语句如下。

```
Private Declare Function HtmlHelpA Lib "hhctrl.ocx" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As Long, ByVal dwData As Long) As Long
'hwndCaller 指定调用者的窗口, pszFile 指定要调用的文件,
```

'uCommand 是发送给 HtmlHelp 的命令, dwData 是 uCommand 的参数

(3) 在菜单栏中添加“帮助”子菜单项, 在该菜单项的单击事件中将击键消息发送到活动窗口, 然后在窗体的单击事件中通过 API 中的 HTMLHELP 函数来设置程序所要寻找的相关帮助文件, 其代码如下。

```
Private Sub Help_Click(Index As Integer)           '帮助子菜单项
    HtmlHelpA Frm_main.hWnd, App.Path & "\Help.chm", 0, 0 '调用与主程序同目录下的 Help.chm 帮助文件
End Sub
```

### 25.3.5 使用 SHELLEXECUTE 函数调用帮助

在 VB 应用程序的开发过程中, 可以使用 API 的 SHELLEXECUTE 函数调用帮助文件。SHELLEXECUTE 函数的功能是运行一个外部程序, 或者打开一个已注册的文件、打开一个目录、打印一个文件等, 并对外部程序有一定的控制。

另外, 还有几个 API 函数也可以实现这些功能, 但是在大多数情况下 SHELLEXECUTE 函数使用的频率更多些, 因为它的使用并不复杂。该函数的声明格式如下:

```
Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
```

API 函数的 SHELLEXECUTE 参数值如表 25.6 所示。

表 25.6 API 函数的 SHELLEXECUTE 参数值

参 数	类 型	说 明
hwnd	Long	指定一个窗口的句柄, 有时候 Windows 程序有必要在创建自己的主窗口前显示一个消息框
lpOperation	String	指定字符串 open 来打开 lpFile 文档, 或指定 Print 来打印它
lpFile	String	用关联程序打印或打开一个程序名或文件名
lpParameters	String	如 lpFile 是可执行文件, 则这个字符串包含传递给执行程序的参数
lpDirectory	String	使用的完整路径
nShowCmd	Long	定义了如何显示启动程序的常数

**例 25.7** 使用 SHELLEXECUTE 函数调用帮助文件。具体步骤如下: (实例位置: 光盘\TM\sl\25\7)

(1) 在窗体的 Picture 属性中添加应用程序的主界面图片信息。

(2) 在使用 SHELLEXECUTE 函数之前需要声明, 其声明语句如下。

'声明 API 函数用于异步打开一个文档

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" _
    (ByVal hwnd As Long, ByVal lpOperation As String, _
    ByVal lpFile As String, ByVal lpParameters As String, _
    ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
'用原来的大小和位置显示一个窗口, 同时令其进入活动状态
```

```
Private Const SW_SHOWNORMAL = 1
```

(3) 在菜单栏中添加“帮助”子菜单项,在该菜单项的单击事件中将击键消息发送到活动窗口,然后在窗体的单击事件中通过 API 中的 SHELLEXECUTE 函数来设置程序所要调用的帮助文件,其代码如下。

```
Private Sub Help_Click(Index As Integer)           '帮助子菜单项
    Dim a As Long : Dim b As String
    b = App.Path & "\Help.chm"                   '用变量 b 记录与主程序同目录下的 Help.chm 帮助文件
    a = ShellExecute(0, "open", b, "", "", SW_SHOWNORMAL)
End Sub
```

## 25.4 小结

本章主要介绍了如何制作帮助文件,以及如何在 VB 应用程序中调用该文件。在此只简单介绍了 VB 自带的帮助文件制作工具,在实际项目开发过程中,运用 VB 自带的工具制作帮助文件耗时太长,所以用户可以自行选择使用第三方提供的工具来制作。

## 25.5 练习与实践

1. 设计一个简单的 HLP 格式的帮助用户,设计完成的效果如图 25.34 所示,其中,需要的 Word 文档到光盘对应路径下即可查找。(答案位置:光盘\TM\sl\25\8)
2. 设计一个简单的 CHM 格式的帮助用户,设计完成的效果如图 25.35 所示,其中,需要的 Word 文档到光盘对应路径下即可查找。(答案位置:光盘\TM\sl\25\9)

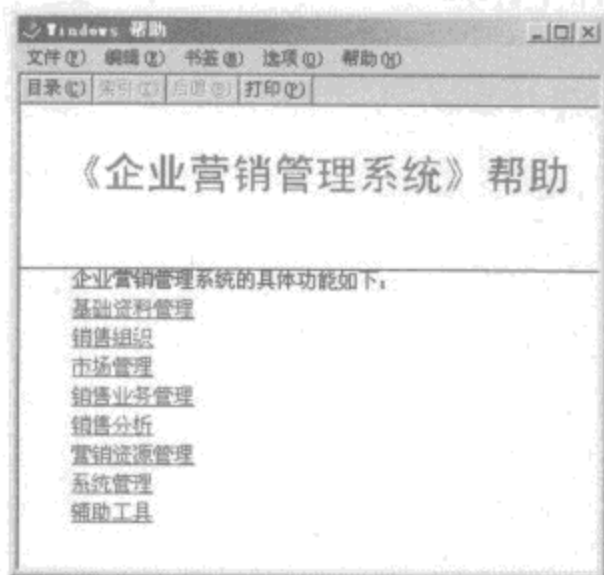


图 25.34 “企业营销管理系统”帮助文件

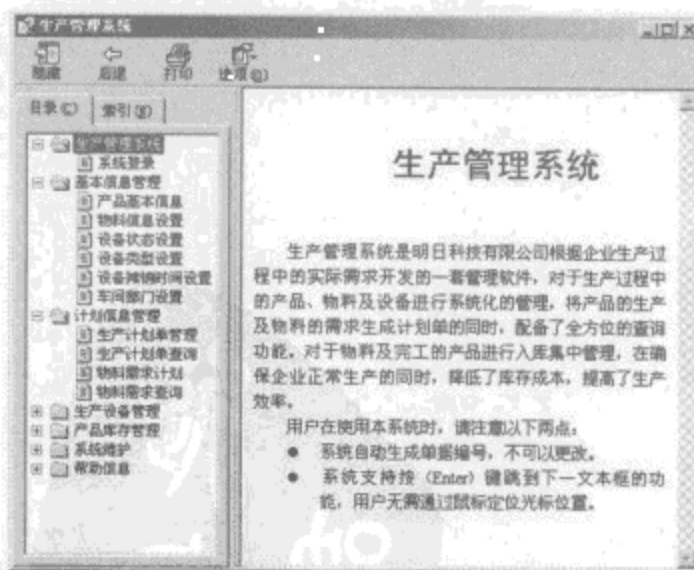


图 25.35 “生产管理系统”帮助文件





# 第26章

## 应用程序打包

(教学录像: 23 分钟)

使用 VB 所开发的工程, 包含许多分散的、与应用程序相关的文件, 在没有经过编译之前是不能在没有 VB 开发环境的计算机中运行的, 而在编译以后生成的 EXE 文件, 又不能包含所有分散的与应用程序相关的文件。如果能让应用程序在其他计算机上正常运行, 就需要将这些相关的文件集中起来。而打包就是将这些相关文件集中起来, 形成一个 Setup.exe (可执行) 安装包文件的过程。

在其他没有 VB 环境的计算机上, 通过执行 Setup.exe 安装包文件, 将应用程序和与其相关的文件一起安装到所在计算机上, 这样应用程序就能正常地在该计算机中运行了。

通过阅读本章, 您可以:

- » 了解设置数据源的过程
- » 掌握如何编译工程
- » 掌握打包的详细过程
- » 掌握如何安装应用程序
- » 掌握如何卸载安装的应用程序
- » 掌握打包时的注意事项



## 26.1 打包前的工作

教学录像：光盘\TM\lx\26\打包前的工作.exe

本节介绍程序打包前的工作，包括设置数据源和编译工程。

### 26.1.1 设置数据源

在开发数据库应用程序时，由于程序连接数据源要求具有完整的路径信息，安装过程中如果处理不好，将造成应用程序执行时发生找不到数据源的错误。如果访问的数据库为本机的数据库，如 Access 数据库，则这个数据源必须随同安装文件一起复制到该计算机中。

因此当我们在工程中指定数据源位置时，就需要注意后面应用程序与数据源间的交互。例如在工程中利用数据环境设计器创建与数据库的连接，启动 Connection 对象的数据连接属性对话框，切换到“连接”选项卡中指定数据来源的位置时，可以直接输入数据库的名称。例如输入数据库的名称为 db\_Data.mdb。

当编译了此工程获得执行文件时，只要将 db\_Data.mdb 数据库与执行文件存放在同一目录中，启动应用程序便能够顺利连接数据库同时进行访问数据；若要通过打包，安装这个应用程序到其他计算机，同样只要将 db\_Data.mdb 数据库复制到安装目录中即可。

### 26.1.2 编译工程

在 VB 环境中选择“文件”/“生成\*.exe”项，启动“生成工程”对话框，从中指定可执行文件的名称以及存放的位置，如图 26.1 所示。

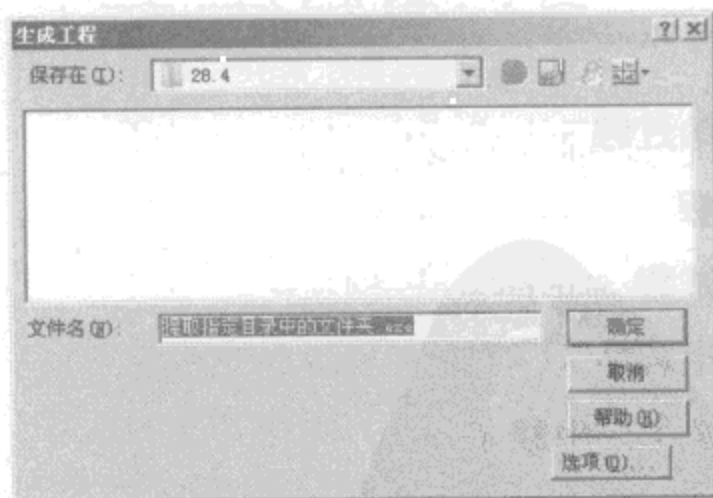



图 26.1 “生成工程”对话框

说明：“生成\*.exe”命令中的“\*”根据编译的工程不同而改变，这里以例 24.4 提取指定目录中的文件夹为例，所以选择“生成提取指定目录中的文件夹.exe”命令。

单击“选项”按钮，打开“工程属性”对话框，对话框中包含“生成”与“编译”两个选项卡。在“生成”选项卡中可以指定工程的版本信息以及程序名称、图标等项目，如图 26.2 所示。

 说明：“工程属性”对话框可以在 VB 环境中选择“工程”/“\*属性”命令打开。如打开“提取指定目录中的文件夹”工程的属性对话框：选择“工程”/“提取指定目录中的文件夹 属性”命令。

“编译”选项卡如图 26.3 所示，在“编译”选项卡中可指定这个工程的编译类型，包含的选项除了伪代码（P-Code）与本机代码（Native-Code）两大主要类型外，还提供了一些详细设置项目，分别说明如下。

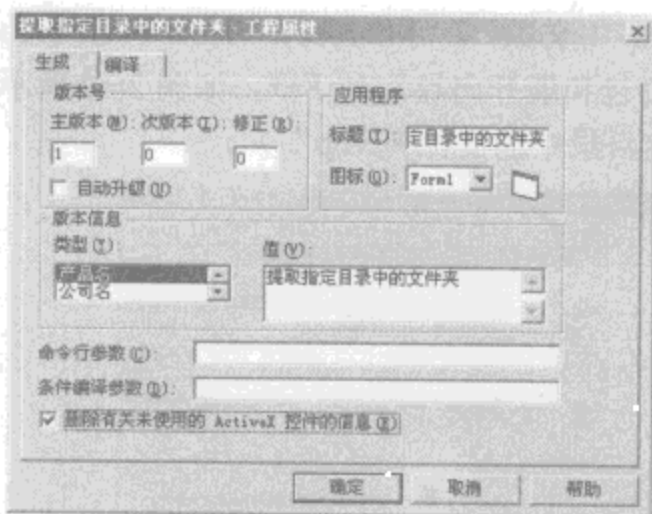


图 26.2 “工程属性”对话框中的“生成”选项卡

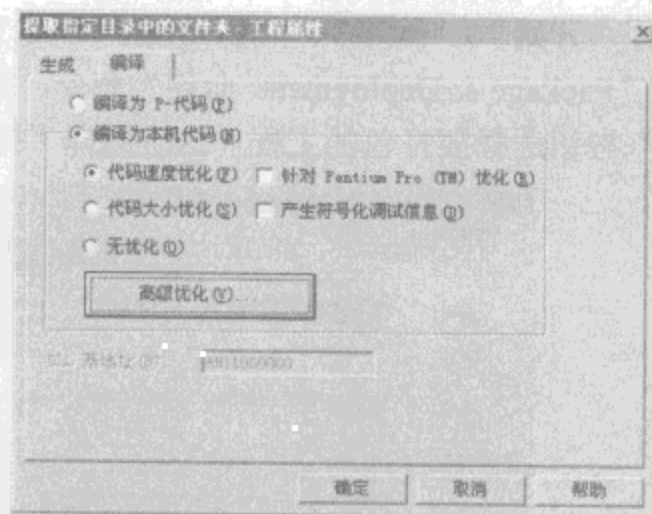


图 26.3 “工程属性”对话框“编译”选项卡

#### ☒ 本机代码类型

将应用程序编译成机器码格式，具有较好的执行效率，也可以利用系统提供的标准机器码调试工具，来调试编译完成的机器码执行文件。在优化设置区域提供了一些选择项目，如表 26.1 所示。

表 26.1 机器码执行文件中优化设置区域提供的选择项目

选 项	描 述
代码速度优化	使编译后的程序具有最快的执行速度
代码大小优化	使编译完成的执行文件达到最优化
无优化	关闭所有最优化
针对 Pentium Pro(TM)优化	产生 Pentium Pro(P6)处理器最优化的程序代码
产生符号化调试信息	编译成的执行文件可运用 Visual C++或其他兼容的调试程序进行调试

#### ☒ 伪代码类型

伪代码类型是介于 VB 的高阶命令（设计者输入程序代码）和供电脑辨识执行的低阶机器码之间的执行文件格式，其执行方式是连接一个 Runtime 程序库，将每一行的伪代码翻译为电脑看得懂的机器码然后去执行功能，这种编译类型的特点可以使执行文件变得很小。

设置完成工程的属性以后，单击“确定”按钮关闭“工程属性”对话框，回到“生成工程”对话框。在“生成工程”对话框中单击“确定”按钮，开始编译程序并生成“提取指定目录中的文件夹.exe”文件。



## 26.2 工程打包的详细过程

 教学录像：光盘\TM\lx\26\工程打包的详细过程.exe

本节将以生成提取指定目录中的文件夹的系统为例，详细介绍“打包和展开向导”每个程序的设置步骤和技巧。打包的工程对象为“提取指定目录中的文件夹.vbp”，通过下面对于每个程序中各个项目的说明，将相关的文件打包，使其能够安装于其他计算机上且能够在脱离 VB 环境下执行。

### (1) 打开打包程序向导并打开被打包的工程

选择“开始”/“程序”/“Microsoft Visual Basic 6.0 中文版”/“Microsoft Visual Basic 6.0 中文版工具”/“Package & Deployment 向导”命令，打开“打包和展开向导”对话框，如图 26.4 所示。单击“浏览”按钮选择要打包的工程（这里选择“提取指定目录中的文件夹.vbp”）。

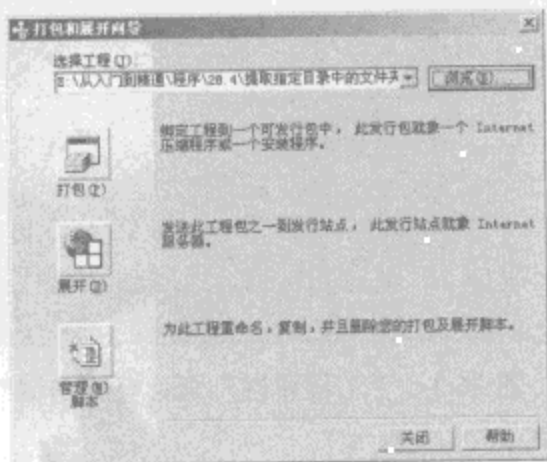


图 26.4 指定工程及操作类型

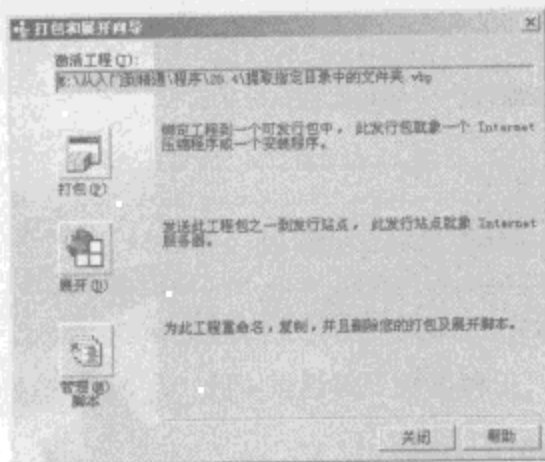



图 26.5 在 VB 打包向导中启动打包程序的初始界面

在打开的 VB 工程中选择“外接程序”/“外接程序管理器”命令，打开“外接程序管理器”对话框。在“可用外接程序”列表中选择“打包和展开向导”项，并在“加载行为”区域中选中“加载/卸载”复选框，单击“确定”按钮，将“打包和展开向导”加载到外接程序中。在外接程序菜单中将列出“打包和展开向导”命令，执行这个命令即可启动“打包和展开向导”，如图 26.5 所示。

 注意：命令只可以为当前启动的工程进行打包和展开等操作。

### (2) 重新编译可执行文件

单击“打包”按钮，开始打包操作。如果在打包之前应用工程中的文件比可执行文件更新，则在单击“打包”按钮时，会弹出重新编译可执行文件对话框，如图 26.6 所示。在该对话框中，如果单击“是”按钮，开始重新编译可执行文件；单击“否”按钮，则在打包的过程中，使用原有的可执行文件。

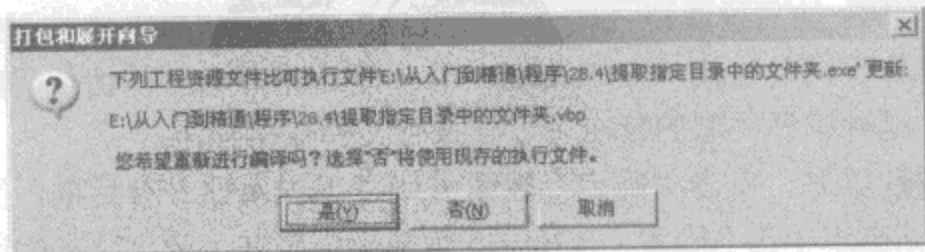


图 26.6 重新编译可执行文件对话框

## (3) 打开“包类型”对话框

在第(2)步中,无论是单击“是”按钮还是单击“否”按钮都会打开“包类型”对话框,在“包类型”对话框中的“包类型”列表中包含两个选项,其中“标准安装包”选项用于创建由 setup.exe 程序安装的包,“相关文件”用来创建应用程序运行时要求的部件文件列表信息。这里选择“标准安装包”选项,如图 26.7 所示。

## (4) 设置包文件的存储位置

设置完“包类型”对话框以后,单击“下一步”按钮进入“打包文件夹”对话框,如图 26.8 所示。“打包文件夹”对话框用来设置打包后的文件存放的路径。单击“新建文件夹”可以新建一个文件夹来存放打包文件;单击“网络”按钮可以把打包后的文件存储到网络上的公共文件夹。

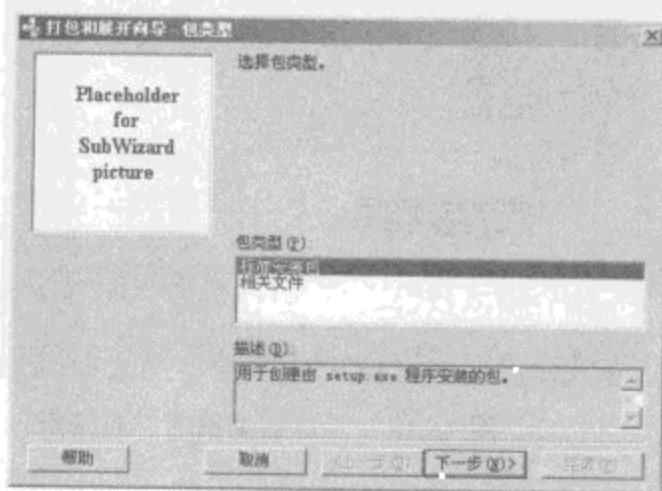


图 26.7 “包类型”对话框

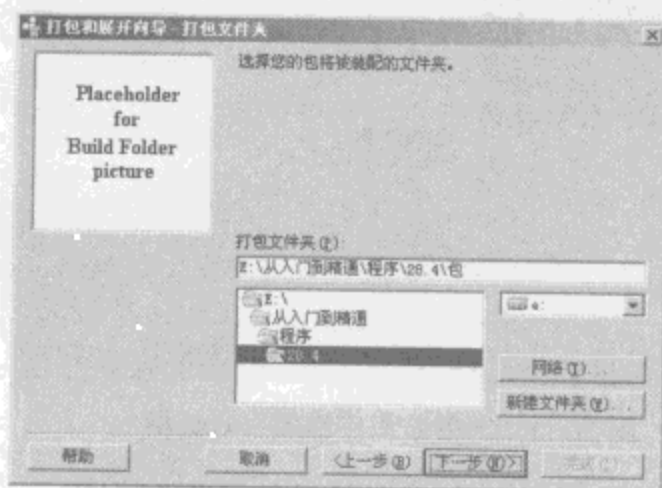


图 26.8 “打包文件夹”对话框

## (5) 添加或删除包含的文件

设置完打包后文件存储的文件夹位置后,单击“下一步”按钮,打开“包含文件”对话框,“打包和展开向导”将依据打包的工程对象,自动将执行文件以及相关的 Runtime 文件、控件对象库等列在界面的清单中,如图 26.9 所示。可以根据需要添加(单击“添加”按钮)或者删除(取消选中文件前的复选框)包含的文件。

## (6) 指定打包选项

完成“包含文件”对话框的设置以后,单击“下一步”按钮打开“压缩文件选项”对话框,如图 26.10 所示。随着存储技术的飞速发展,软盘已经退出了市场,所以在“压缩文件选项”对话框中选择“单个的压缩文件”选项即可。

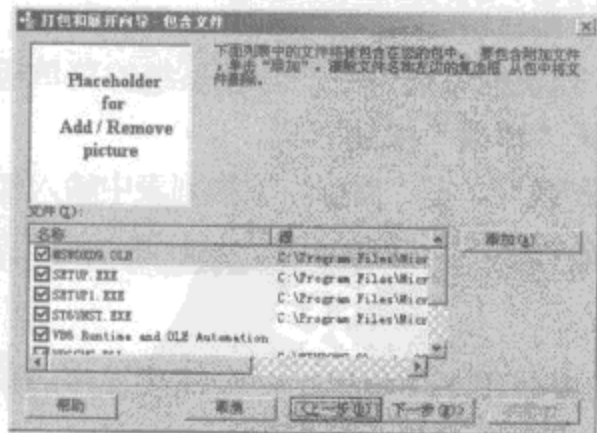


图 26.9 “包含文件”对话框

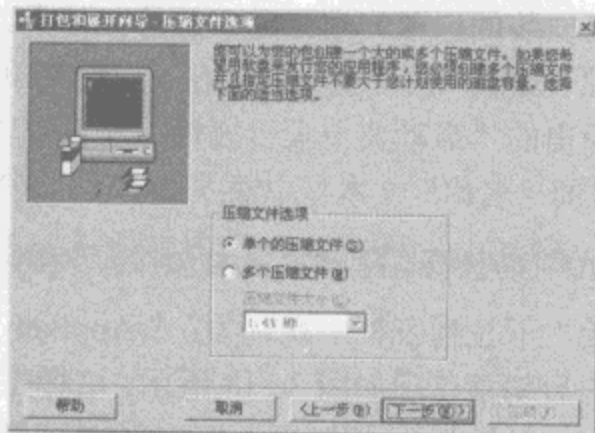


图 26.10 “压缩文件选项”对话框

### (7) 设定安装标题

设置完压缩文件选项以后单击“下一步”按钮，打开“安装程序标题”对话框，如图 26.11 所示。在这里可以为安装程序设置安装标题，在窗体界面的文本框中直接输入安装程序的安装标题即可。

### (8) 指定工作组与项目

设置完成安装程序标题以后单击“下一步”按钮，打开“启动菜单项”对话框，如图 26.12 所示。此对话框的“新建组”按钮可以创建程序的组；“新建项”按钮可以为组创建新的项目；“属性”按钮可对组和项进行相应的设置；“删除”按钮可以删除选定的组或项。

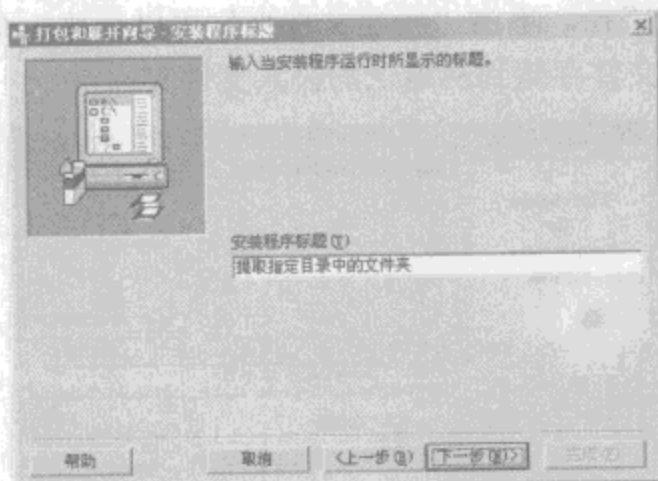


图 26.11 “安装程序标题”对话框

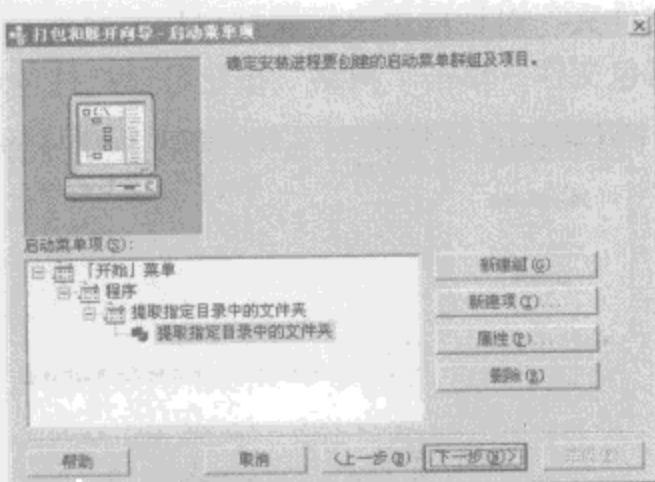


图 26.12 “启动菜单项”对话框

下面对菜单的组和项进行具体分析和创建。

#### ☑ 菜单组

如果要将应用程序的菜单组直接创建在“开始”菜单的主层次中，就要在“启动菜单项”中选中“开始菜单”，然后单击“新建组”按钮，这个新建的组即被创建在“开始”菜单主层次中。同样选中“程序”节点，然后单击“新建组”按钮就可把组建立在“开始”菜单的“程序”子目录中。

#### ☑ 菜单项目

项一般都创建在相应的组中，也可直接创建在“程序”子目录中。选中要创建项的组，单击“新建项”按钮，弹出新建项的对话框，默认状况如图 26.13 所示。其中，“名称”用于指定创建的菜单项的名称；“目标”显示由创建的项所指的文件，如果指定的文件有命令行参数，也可以直接输入命令，“开始”用于指定包含原始项目或一些相关文件的文件夹，程序有时需要使用其他位置的文件，就需要指定这些文件所在的文件夹，以便程序可以找到它们。在这个字段选择宏（打包和展开向导使用宏来指示包装要安装的位置）来指定开始的文件夹。

下面设置“卸载”的快捷方式。在“启动菜单项”列表框中选中生产管理系统的菜单组，单击“新建项”，在弹出的“启动菜单项目属性”对话框内设置“卸载”快捷方式。

在对话框的“名称”文本框中输入菜单项的名称“卸载”。在“目标”下拉列表中输入卸载命令为：

```
$(WinPath)ST6UNST.EXE -n "$(AppPath)ST6UNST.LOG"
```

在“开始”下拉列表框中默认为：\$(AppPath)

设置后的启动菜单项如图 26.14 所示。这样生成的安装程序中将包括一个卸载程序的菜单项。应用程序还可以在“开始”/“控制面板”/“添加或删除程序”中进行手动删除。



图 26.13 设置“卸载”菜单属性

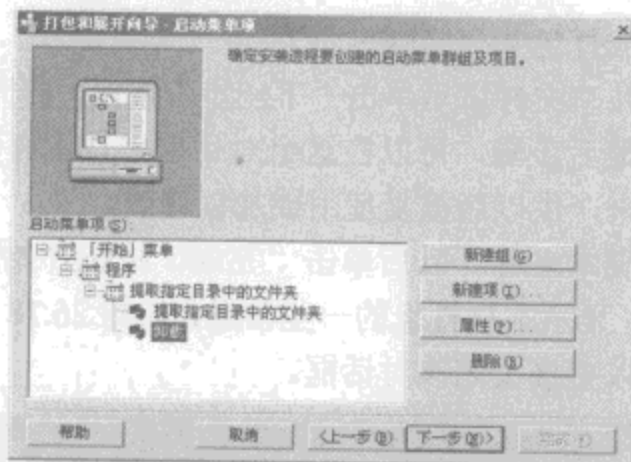


图 26.14 添加了“卸载”项的菜单项对话框

### (9) 调整安装位置

完成“启动菜单项”的设置后单击“下一步”按钮，打开“安装位置”对话框。“打包和展开向导”将指定相关文件的默认安装位置，也可以自行设置文件的安装位置。单击需要改变安装路径文件的“安装路径列”，打开下拉列表框，选择相应的安装位置模式，安装位置模式的说明如表 26.2 所示。

表 26.2 安装位置模式说明

选 项	描 述
\$(WinSysPath)Windows	根据 Windows 操作系统版本不同会安装在\Windows\System 或\Winnt\System32 目录下
\$(WinPath)	根据 Windows 操作系统版本的不同会将文件安装在\Windows 或\Winnt 目录中
\$(CommonFiles)	有时安装共享文件的公用目录。如：C:\Program Files\Common Files
\$(ProgramFiles)	应用程序通常所安装到的目录。如：C:\Program Files
\$(AppPath)	用户指定的应用程序目录，或 Setup.lst 中[SETUP]部分指定的 DefaultDir 值
\$(MSDAOPath)	数据访问对象（DAO）部件在注册表中的位置

打包应用程序时，可以针对这些文件的安装位置进行适当的调整，如图 26.15 所示。

### (10) 指定共享文件

设置完文件的安装位置以后单击“下一步”按钮，打开“共享文件”对话框，如图 26.16 所示。指定共享文件主要用来供设计者决定哪些文件是以共享模式安装进来的。共享文件是在用户机器上允许被其他应用程序所使用，当一般用户解除安装应用程序时，如果电脑上还有其他应用程序在使用该文件，这种文件将不会被删除。

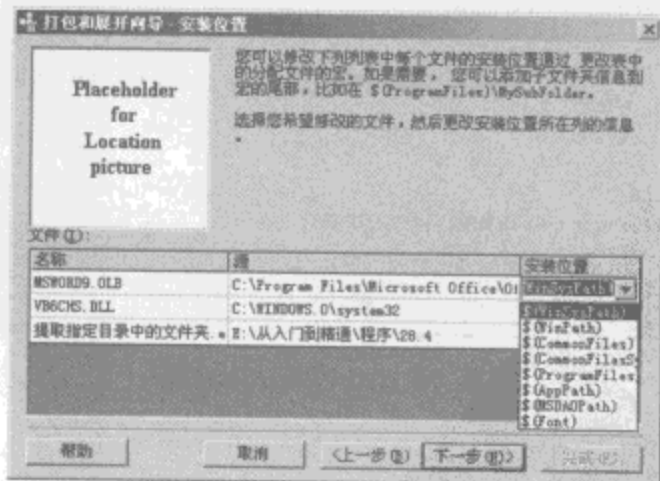


图 26.15 “安装位置”对话框

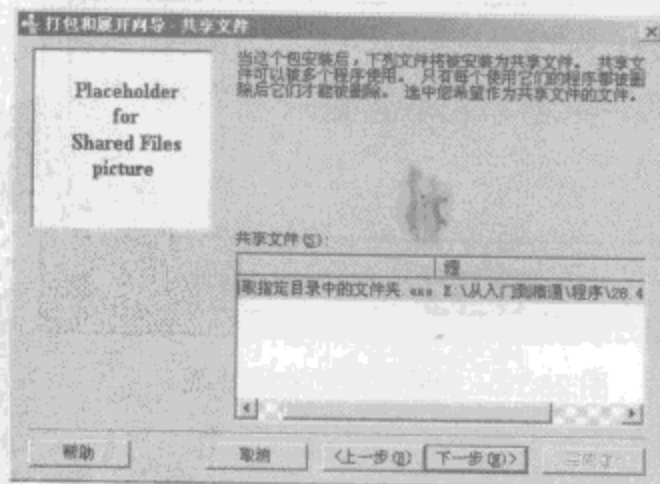


图 26.16 “共享文件”对话框



### (11) 完成打包并存储脚本

设置完成共享文件以后单击“下一步”按钮，打开“已完成”对话框，如图 26.17 所示。这一步是打包过程中的最后步骤，可以在这个对话框中设置存储脚本的“脚本名称”，以便将本次打包所作的设置值储存起来，供以后重复使用或加以修改。

设置完“脚本名称”以后单击“完成”按钮开始进行打包，完成打包工作以后系统将弹出“打包报告”对话框，显示本次打包的一些信息，如图 26.18 所示。如果想保存该报告，可以单击“保存报告”按钮，否则可以直接关闭该对话框。

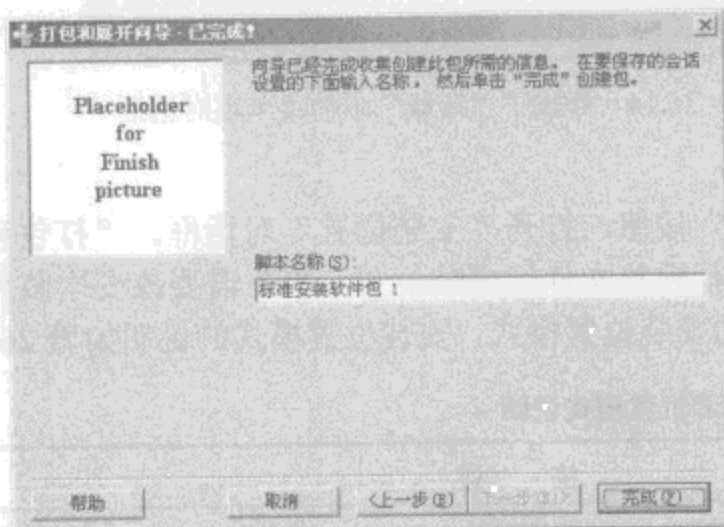


图 26.17 设置存储脚本的名称

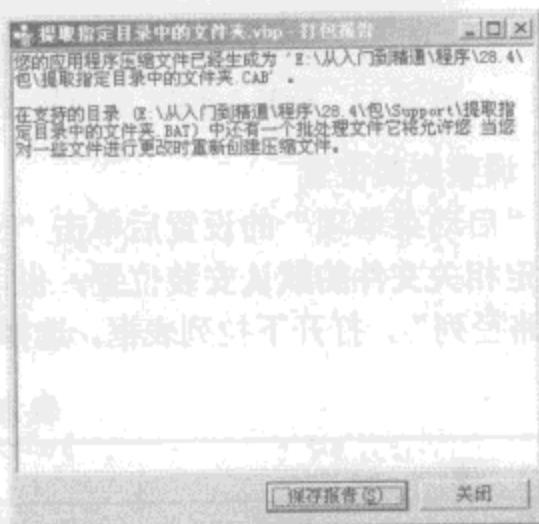



图 26.18 “打包报告”对话框

## 26.3 安装应用程序

### 教学录像：光盘\TM\lx\26\安装应用程序.exe

将打包时指定打包文件夹内的文件复制到所要安装程序的计算机上。执行文件夹内的 setup.exe 文件开始安装程序，将弹出“安装程序”的欢迎界面，如图 26.19 所示。

在“安装程序”的欢迎界面中单击“确定”按钮，打开“安装程序”的更改目录对话框，设置程序安装的路径，如图 26.20 所示。设置完程序的安装路径以后，单击图标开始安装程序。

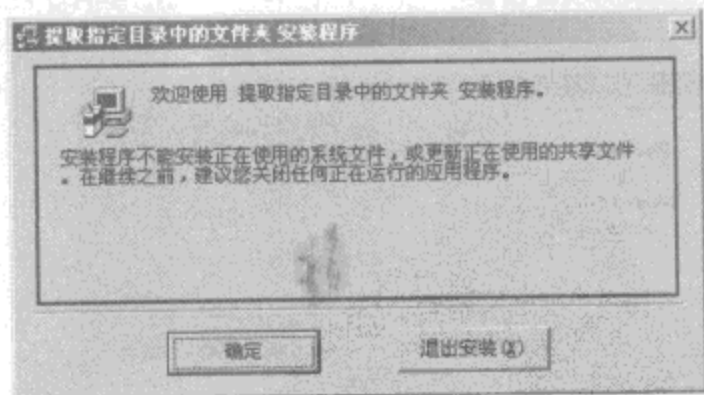


图 26.19 安装程序的欢迎界面

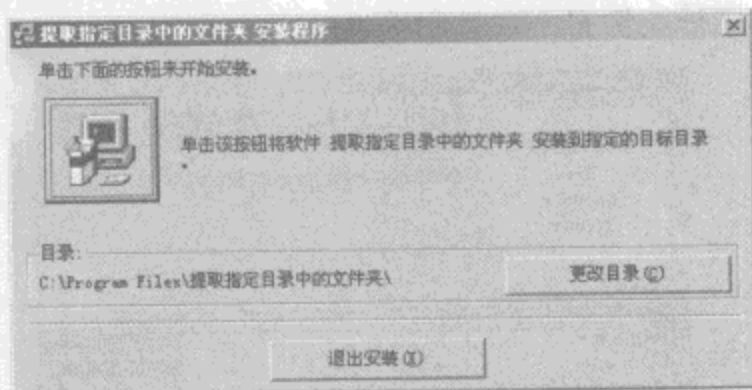


图 26.20 设置安装路径

## 26.4 卸载应用程序

 教学录像：光盘\TM\lx\26\卸载应用程序.exe

如果一个应用程序长期不使用，为了节约系统资源，释放磁盘空间，需要把该应用程序从计算机上卸载下来。大多数的应用程序都提供了卸载、删除的功能，几乎所有的应用程序都可以在“添加删除程序”对话框中创建一个条目，并可以通过“添加删除程序”对话框来卸载该程序。通过 VB 自带的“打包和展开向导”创建的安装文件，如果在“启动菜单”中添加了“卸载”的功能，直接在开始菜单中找到对应的“卸载”命令，启动该命令后系统将弹出如图 26.21 所示的“应用程序删除”对话框，单击“是”按钮开始卸载程序。如果在打包应用程序时没有创建“卸载”启动菜单项，可以通过“添加删除程序”对话框来卸载该程序。

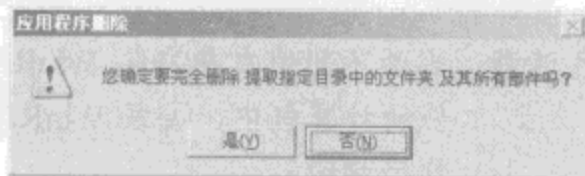


图 26.21 “应用程序删除”对话框

## 26.5 打包时应注意的问题

 教学录像：光盘\TM\lx\26\打包时应注意的问题.exe

前面详细介绍了工程打包、应用程序的安装和卸载，但在实际应用中，经常会遇到一些问题，本节对这些常见问题进行讲解。

### 26.5.1 打包 SQL Server 的数据库应用程序

前台使用 VB、后台使用 SQL Server 数据库开发的应用程序，在打包时是不能将数据库文件打包到安装程序中的，只能打包应用程序以及其中的组件等。但是可以先用 bcp 把数据导出到文件，在将库和表生成 SQL 脚本后，再进行程序的打包。bcp 实用工具能够在 SQL Server 与一个数据文件之间导入与导出数据。例如，Northwind 数据库内有表 Jane's Orders，该表由用户 Jane Doe 所拥有。若要使用登录名 Jane Doe 和密码 go dba 将该表从 Northwind 数据库大容量复制到 Orders.txt 文件，可执行下列命令之一。

```
bcp "Northwind.Jane Doe.Jane's Orders" out "Jane's Orders.txt" -c -q -U"Jane Doe" -P"go dba"
bcp "Northwind.[Jane Doe].[Jane's Orders]" out "Jane's Orders.txt" -c -U"Jane Doe" -P"go dba"
```

### 26.5.2 打包程序时如何添加文件夹

在打包文件时向包含文件清单中添加附加文件时，是无法添加文件夹的。如果需要添加文件夹，应该将文件夹压缩之后再添加到清单中。在安装应用程序时，通过应用程序将该文件夹解压之后，这样才能保证应用程序的正常运行。在应用程序中解压压缩文件夹的事件代码如下：

```

Dim mystr As String
Dim Source As String      '源文件
Dim Target As String      '目标文件
Dim retval
Private Sub Cmd_jys_Click()
    mystr = "C:\Program Files\WinRAR\WinRAR.exe"
    Source = App.Path & "\Picture.rar"      '原压缩文件的位置
    Target = App.Path & "\NewFile"          '存放解压缩文件的位置
    mystr = mystr & " X " & Source & " " & Target
    retval = Shell(mystr, vbHide)           '调用 RAR 文件解压缩
End Sub

```

注意：因为不同用户在安装 WinRAR 文件时的安装路径可能不同，所以应该注意在运行解压缩文件的计算机中，安装 WinRAR 程序的路径必须与解压缩文件程序中所调用 WinRAR 文件的路径相同。

### 26.5.3 解决文件未找到的问题

任何一个应用软件在开发完成之后，都会附带许多的安装文件。这些文件在打包应用程序时，需要全部添加到打包程序当中，否则在安装完应用软件后运行程序时，找不到程序运行时所需要的文件，这样程序运行时就会出现文件未找到的提示信息。

### 26.5.4 修改文件安装的默认路径

使用 VB 打包程序安装时，通常默认的安装路径为“C:\Program Files\”，要更改这个默认的安装路径，可以通过修改安装文件目录下的 Setup.lst 文件实现。首先使用记事本打开该文件，在其中找到“DefaultDir= \$(ProgramFiles)”字符串，如果想把程序默认的安装路径修改到 C 盘根目录下，就将该段字符串修改为“DefaultDir=C:\”即可。

## 26.6 小结

本章介绍了应用程序的编译、打包以及安装卸载的相关信息，重点讲述了工程打包的详细流程以及打包时应注意的问题。通过本章的学习读者应该掌握编写完程序以后如何进行编译、如何打包以及如何安装打包程序。

## 26.7 练习与实践

根据本节所讲内容打包“光盘\TM\sl\24\10”目录下的应用程序。（答案位置：光盘\TM\lx\26\应用实践.exe）

# 第4篇

## 项目实战

### » 第27章 企业进销存管理系统

本篇通过一个大型、完整的企业进销存管理系统，运用软件工程的设计思想，让读者学习如何进行软件项目的实践开发。书中按照编写项目计划书→系统设计→数据库设计→创建项目→实现项目→运行项目→项目打包部署→解决开发常见问题的过程进行介绍，带领读者一步一步亲身体会开发项目的全过程。





# 第27章

## 企业进销存管理系统

(教学录像: 2小时 48分钟)

企业进销存管理系统是一个信息化管理软件。实现企业信息化管理是现代企业稳步发展的必要条件,它可以提高企业的管理水平和工作效率,最大限度地减少手工操作带来的失误。企业进销存管理系统可以实现企业的进货、销售、库存管理等各项业务的信息化管理。

通过阅读本章,您可以:

- 如何需求分析和编写项目计划书
- 进销存管理系统的一般设计理念
- 如何实现 Flash 与 VB 6.0 的交互
- 如何实现无标题窗体移动
- 如何使用 MSChart 控件显示不同类型的图表

## 27.1 系统分析

 教学录像：光盘\TM\lx\27\系统分析.exe

系统分析是开发程序前必须完成的工作，主要包括需求分析、可行性分析和编写项目计划书等。

### 27.1.1 需求分析

加入 WTO 之后，随着国内经济的高速发展，中小型的商品流通企业越来越多。这些企业经营的商品种类繁多，难以管理。×××有限公司是一家以商业经营为主的私有企业，公司为了完善管理制度，增强企业的竞争力，决定开发进销存管理系统，以实现商品管理的信息化。进销存管理系统是商业企业经营和管理中的核心环节，也是企业取得效益的关键。现需要委托其他单位开发一个企业进销存管理系统。

### 27.1.2 可行性分析

根据《GB8567—88 计算机软件产品开发文件编制指南》中可行性分析的要求，制定可行性研究报告如下。

#### 1. 引言

##### ☒ 编写目的

为了给软件开发企业的决策层提供是否进行项目实施的参考依据，现以文件的形式分析项目的风险、项目需要的投资与效益。

##### ☒ 背景

×××有限公司是一家以商业经营为主的私有企业。为了完善管理制度和增强企业的竞争力，实现信息化管理，现需要委托其他公司开发一个将商品进货、销售与库存管理一体化的管理软件，项目名称为“企业进销存管理系统”。

#### 2. 可行性研究的前提

##### ☒ 要求

- 可以真正实现对企业商品进销存的管理。
- 系统的功能要符合本企业的实际情况。
- 系统的功能操作要方便、易懂，不要有多余或复杂的操作。
- 可以方便地对进销存信息进行输出打印。

##### ☒ 目标

便于对企业内部的进销存情况进行管理。

##### ☒ 评价尺度

项目需要在两个月内交付用户使用。系统分析人员需要 3 天内到位；用户需要 5 天时间确认需求

分析文档, 去除其中可能出现的问题, 例如用户可能临时有事, 占用 7 天时间确认需求分析。那么程序开发人员需要在 50 天的时间内进行系统设计、程序编码、系统测试、程序调试和系统打包等部署工作, 其间还包括了员工每周的休息时间。

### 3. 投资及效益分析

#### ☒ 支出

根据预算, 公司计划投入 8 个人, 为此需要支付 9 万元的工资及各种福利待遇; 项目的安装、调试以及用户培训、员工出差等费用支出需要 2.5 万元; 在项目后期维护阶段预计投入 3 万元的资金。累计项目投入需要 14.5 万元资金。

#### ☒ 收益

客户提供项目开发资金 30 万元, 对于项目后期进行的改动, 采取协商的原则, 根据改动规模额外提供资金。因此, 从投资与收益的效益比上, 公司大致可以获得 15.5 万元的利润。

项目完成后, 会给公司提供资源储备, 包括技术、经验的积累。

### 4. 结论

根据上面的分析, 在技术上不会存在问题, 因此项目延期的可能性很小; 在效益上, 公司投入 8 个人、两个月的时间获利 15.5 万元, 比较可观; 另外, 公司还可以储备项目开发的经验和资源。因此, 认为该项目可以开发。

## 27.1.3 编写项目计划书

根据《GB8567—88 计算机软件产品开发文件编制指南》中的项目开发计划要求, 结合单位实际情况, 设计项目计划书如下。

### 1. 引言

#### ☒ 编写目的

为了能使项目按照合理的顺序开展, 并保证按时、高质量地完成, 现拟订项目计划书。将项目开发生命周期中的任务范围、团队组织结构、团队成员的工作任务、团队内外沟通协作方式、开发进度、检查项目工作等内容描述出来, 作为项目相关人员之间的共识、约定, 以及项目生命周期内的所有项目活动的行动基础。

#### ☒ 背景

企业进销存管理系统是由×××有限公司委托本公司开发的大型管理系统。主要功能是实现企业进销存的信息化管理, 包括统计查询、进货、退货、销售和库存盘点等功能。项目周期两个月。项目背景规划如表 27.1 所示。

表 27.1 项目背景规划

项 目 名 称	签定项目单位	项目负责人	参与开发部门
企业进销存管理系统	甲方: ×××科技有限公司	甲方: 王经理	设计部门
	乙方: TM 科技有限公司	乙方: 高经理	开发部门 测试部门



## 2. 概述

### ☑ 项目目标

项目应当符合 SMART 原则, 把项目要完成的工作用清晰的语言描述出来。企业进销存管理系统的主要目标是为企业的管理者提供一套能够方便地对企业内部人员的变更及调动等进行管理的软件。

### ☑ 应交付成果

项目开发完成后, 交付的内容如下。

- 以光盘的形式提供企业进销存管理系统的源程序、系统数据库文件、系统打包文件和系统使用说明书。
- 系统发布后, 进行无偿维护和服务 6 个月, 超过 6 个月进行系统有偿维护与服务。

### ☑ 项目开发环境

开发本项目所用的操作系统可以是 Windows Server 2000、Windows XP 或 Windows Server 2003, 开发工具为 VB 6.0, 数据库采用 SQL Server 2000。

### ☑ 项目验收方式与依据

项目验收分为内部验收和外部验收两种方式。项目开发完成后, 首先进行内部验收, 由测试人员根据用户需求和项目目标进行验收。项目在通过内部验收后, 交给客户进行外部验收, 验收的主要依据为需求规格说明书。

## 3. 项目团队组织

### ☑ 组织结构

本公司针对该项目组建了一个由公司副经理、项目经理、系统分析员、软件工程师、美工人员和测试人员构成的开发团队, 团队结构如图 27.1 所示。

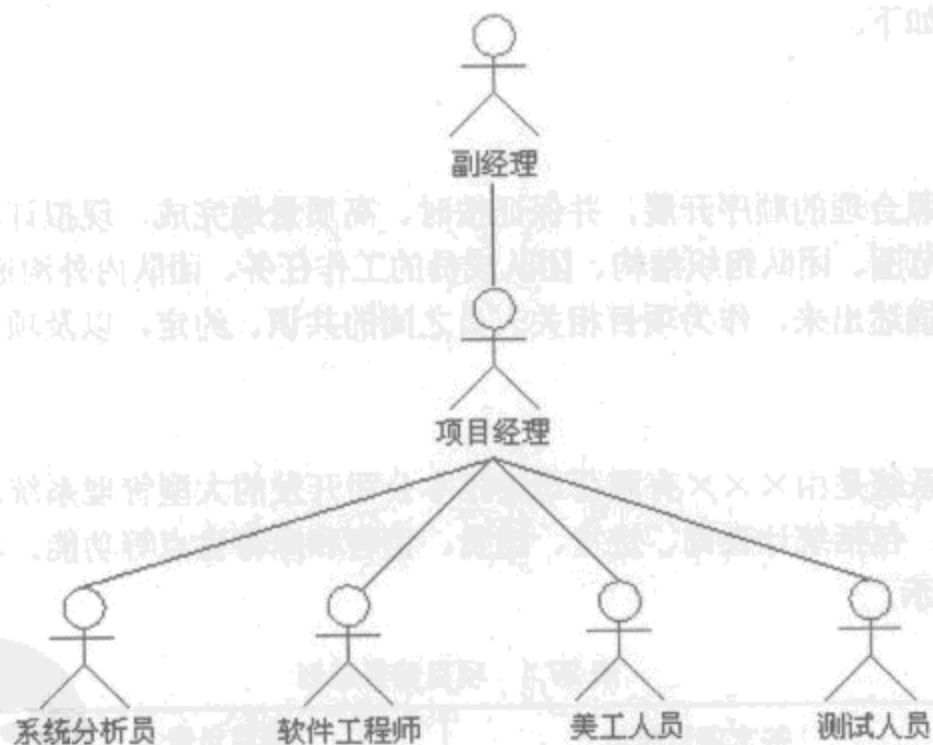


图 27.1 项目开发团队结构

### ☑ 人员分工

为了明确项目团队中每个人的任务分工,现制定人员分工表,如表 27.2 所示。

表 27.2 人员分工表

姓 名	技 术 水 平	所 属 部 门	角 色	工 作 描 述
张某	系统分析师	经理部	副经理	负责项目的审批、决策的实施
王某	系统分析师	项目开发部	项目经理	负责项目的前期分析、策划、项目开发进度的跟踪、项目质量的检查
孙某	中级系统分析员	项目开发部	系统分析员	负责系统功能分析、系统框架设计
刘某	中级软件工程师	项目开发部	软件工程师	负责软件设计与编码
高某	中级软件工程师	项目开发部	软件工程师	负责软件设计与编码
迟某	初级软件工程师	项目开发部	软件工程师	负责软件编码
卢某	中级美工设计师	设计部	美工人员	负责软件的界面设计
李某	中级系统测试工程师	项目开发部	测试人员	对软件进行测试、编写软件测试文档

## 27.2 系统设计

📺 教学录像: 光盘\TM\lx\27\系统设计.exe

系统设计包括系统目标、系统功能结构、系统业务流程和系统编码规范。

### 27.2.1 系统目标

根据需求分析的描述以及与用户的沟通,现制定系统实现目标如下:

- ☑ 界面设计简洁、友好、美观大方。
- ☑ 操作简单、快捷方便。
- ☑ 数据存储安全、可靠。
- ☑ 信息分类清晰、准确。
- ☑ 强大的模糊查询功能,保证数据查询的灵活性。
- ☑ 提供销售排行榜,为管理员提供真实的数据信息。
- ☑ 提供灵活、方便的权限设置功能,使整个系统的管理分工明确。
- ☑ 对用户输入的数据,系统进行严格的数据检验,尽可能排除人为的错误。

### 27.2.2 系统功能结构

企业进销存管理系统的功能结构如图 27.2 所示。

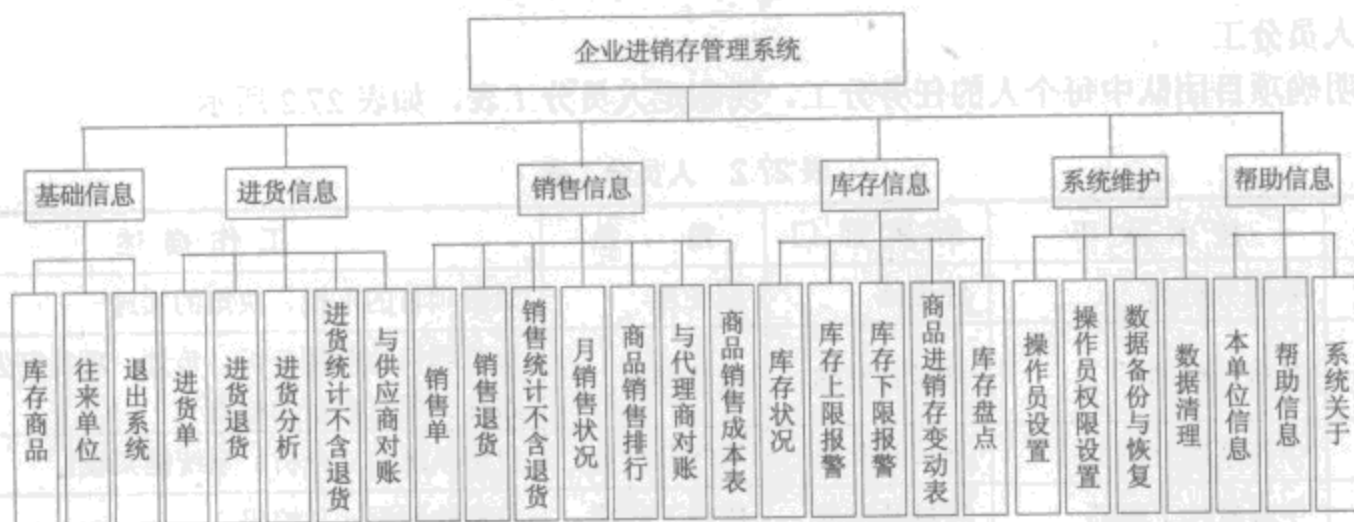


图 27.2 企业进销存管理系统功能结构图

### 27.2.3 系统业务流程图

企业进销存管理系统的业务流程图如图 27.3 所示。

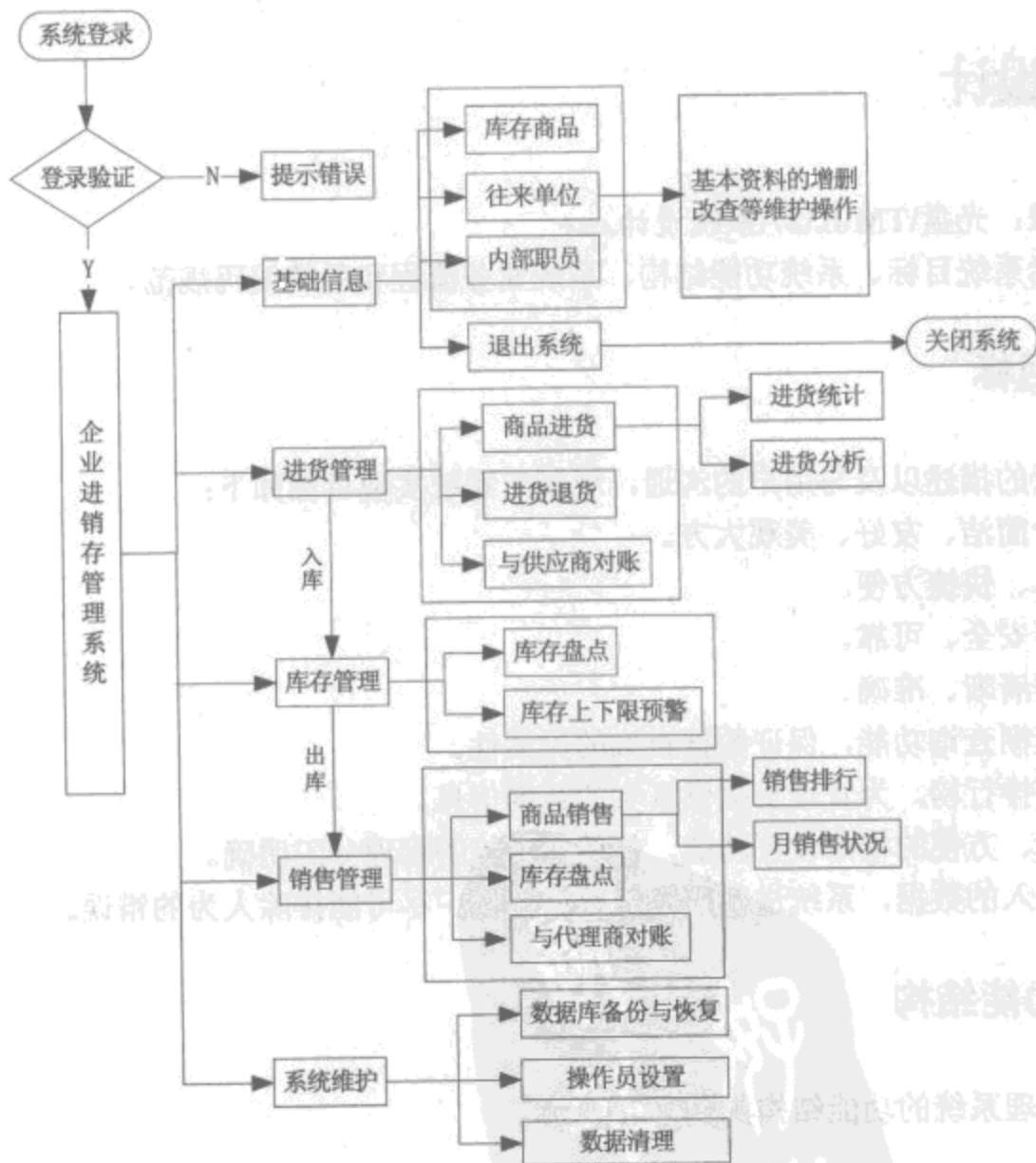


图 27.3 企业进销存管理系统的业务流程图

### 27.2.4 系统编码规范

开发程序时，往往会有多人参与，为了使程序的结构与代码风格标准化，以便于使每个参与开发的人员尽可能直观地查看和理解其他人编写的代码，需要在编码之前制定一套统一的编码规范。下面介绍了一套在程序开发中常用的编码规范供读者参考。

#### 1. 数据库命名规范

##### ☒ 数据库

数据库命名以字母 db（小写）开头，后面加数据库相关英文单词或缩写。下面将举例进行说明，如表 27.3 所示。

表 27.3 数据库命名

数据库名称	描 述
db_SSS	企业进销存管理系统数据库

##### ☒ 数据表

数据表以字母 tb（小写）开头，后面加数据表相关英文单词或缩写。下面将举例进行说明，如表 27.4 所示。

表 27.4 数据表命名

数据表名称	描 述
tbS_sell_main	销售主表
tbS_sell_detailed	销售明细表

##### ☒ 字段

字段一律采用英文单词或词组（可利用翻译软件）命名，如找不到专业的英文单词或词组，可以用相同意义的英文单词或词组代替。下面将举例进行说明，如表 27.5 所示。

表 27.5 字段命名

字 段 名 称	描 述
Tradecode	商品编号
Fullname	商品全称

#### 2. 程序代码命名规范

##### (1) 变量及对象名称定义规则

根据不同的程序需要，编写代码时都需要定义一定的变量或常量，下面介绍一种常见的变量及常量命名规则，如表 27.6 所示。



表 27.6 变量及常量命名规则

变量及常量级别	命名规则	举 例
模块级变量	M_+数据类型简写+变量名称	M_int_xx
全局变量	G_+数据类型简写+变量名称	G_int_xx
局部变量	P_+数据类型简写+变量名称	P_dbl_sl
模块级常量	Mc_+数据类型简写+常量名称	Mc_str_xx
全局常量	Gc_+数据类型简写+常量名称	Gc_str_xx
过程级常量	Pc_+数据类型简写+常量名称	Pc_str_xx

## (2) 控件命名规则

窗体和控件的命名应采用统一的规范, 一般采用具有实际意义的英文单词或标识, 也可以采取多个单词的组合。控件命名规则如表 27.7 所示。

一般, 窗体应采用 frm\_\*\*或 Frm\_main 的形式, 如 frm\_main 等。

表 27.7 控件命名规则

VB 控件	命名形式
Form	Frm
Module	Mdl
Class	Cls
Label (大量的标签不用命名)	Lbl
Text (大量的文本框不用命名)	Txt
ComboBox	Cbx
ListBox	Lit
ListView	Lvw
TreeView	Tvw
Frame	Fam
PictureBox	Pte
Image	Ige
Timer	Tmr
ToolBar	Tbr
CheckBox	Cek
OptionButton	Otn
CommonDialog	Cdg
DTPicker	Dtp
DataGrid	Dgr
ImageList	Imt
CoolBar	Cbr
CommandButton	Cmd
ProgressBar	Pgb

续表

VB 控件	命名形式
SSTab	Stb
StatusBar	Sbr
RichTextBox	Rtb
MaskedTextBox	Mex
TabStrip	Tsp

上面介绍的是一套 VB 6.0 中常用的编码规范, 希望对读者的程序开发有一定的帮助。


## 27.3 系统运行环境

 教学录像: 光盘\TM\lx\27\系统运行环境

本系统的程序运行环境具体如下。

- ☒ 系统开发语言: Microsoft Visual Basic 6.0。
- ☒ 数据库管理软件: Microsoft SQL Server 2000。
- ☒ 运行平台: Windows XP/Windows 2000/Windows Server 2003。
- ☒ 分辨率: 最佳效果 1024×768 像素。

## 27.4 数据库与数据表设计

 教学录像: 光盘\TM\lx\27\数据库与数据表设计.exe

开发应用程序时, 对数据库的操作是必不可少的。数据库设计是根据程序的需求及其实现功能所制定的, 数据库设计的合理性, 将直接影响到程序的开发过程。

### 27.4.1 数据库分析

企业进销存管理系统中采用的是 SQL Server 2000 数据库。SQL Server 2000 数据库在安全性、准确性和运行速度方面有绝对的优势, 并且处理数据量大、效率高, 所以本系统采用了 SQL Server 2000 数据库作为后台数据库。数据库命名为 db\_SSS, 其中包含了 16 张数据表, 用于存储不同的信息, 详细信息如图 27.4 所示。

### 27.4.2 创建数据库

在 SQL Server 2000 中创建数据库 db\_SSS 的具体步骤如下。

- (1) 选择“开始”/“所有程序”/Microsoft SQL Server/“企业管理器”选项, 如图 27.5 所示。



图 27.4 企业进销存管理系统中用到的数据表

(2) 启动 SQL Server 2000 的企业管理器，展开“控制台根目录”树形结构。选中“数据库”节点，单击鼠标右键，在弹出的快捷菜单中选择“新建数据库”命令，如图 27.6 所示。

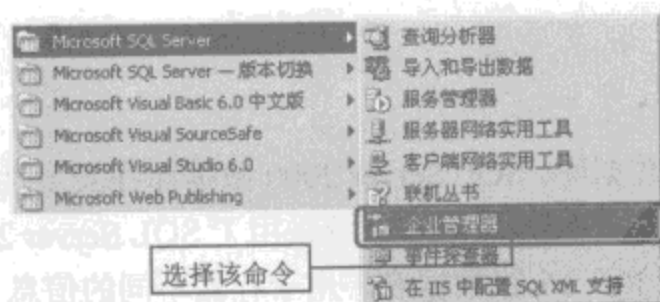


图 27.5 选择“企业管理器”选项



图 27.6 选择“新建数据库”选项

(3) 打开如图 27.7 所示的“数据库属性”对话框，该对话框中，输入新建的数据库的名称 db\_SSS。

(4) 单击“确定”按钮，即可新建一个 db\_SSS 数据库，如图 27.8 所示。

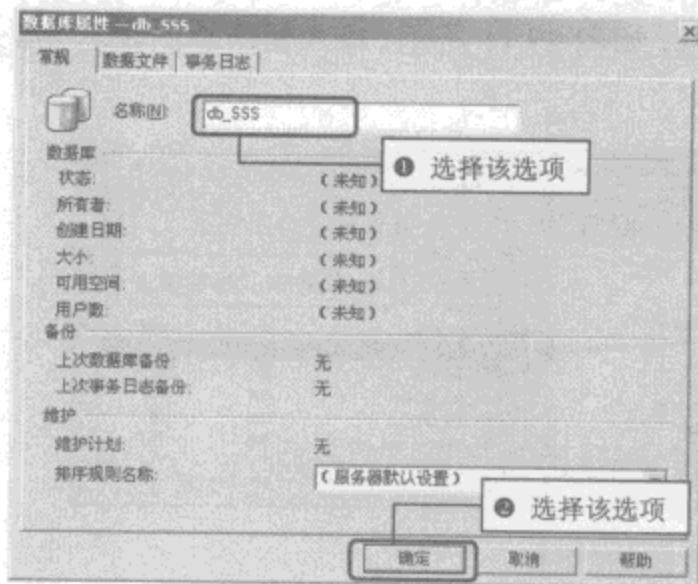


图 27.7 新建数据库名称




图 27.8 新建的 db\_SSS 数据库

### 27.4.3 创建数据表

在已经创建的数据库 db\_SSS 中创建 16 个数据表。

下面以 tbS\_stock 表为例介绍创建数据表的过程。

(1) 展开新建的 db\_SSS 数据库节点，选中“表”节点，单击鼠标右键，在弹出的快捷菜单中选择“新建表”命令，如图 27.9 所示。

(2) 在弹出的“新表”对话框中创建新表的各个字段。这里输入要创建的表中所需要的字段，并设置主键。以 fullname 字段为例，介绍如何创建字段。首先在“列名”列中输入字段名称，这里为 fullname。在“数据类型”列中选择需要的数据类型，这里选择 varchar 类型。在“长度”列中输入该字段的长度，这里为 20。在“允许空”列中，取消选中，因为该字段要被设置为主键，因此该字段不能为空。单击工具栏上  图标，设置该字段为主键，如图 27.10 所示。

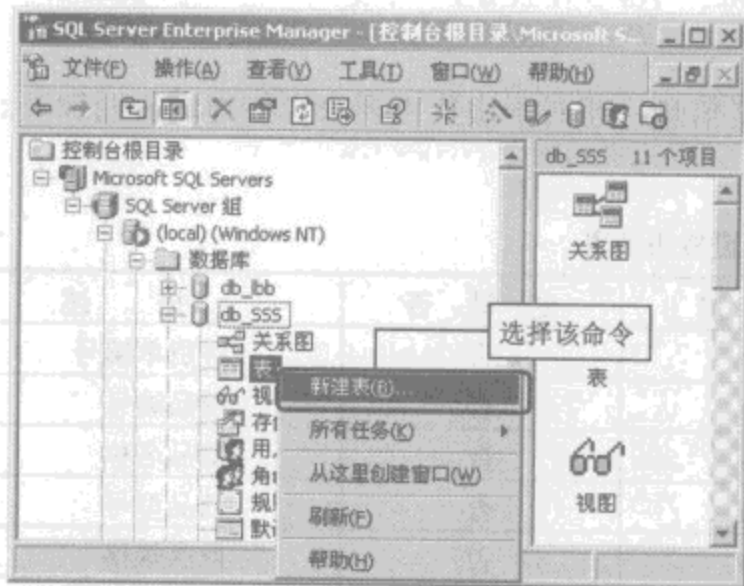


图 27.9 选择“新建表”选项

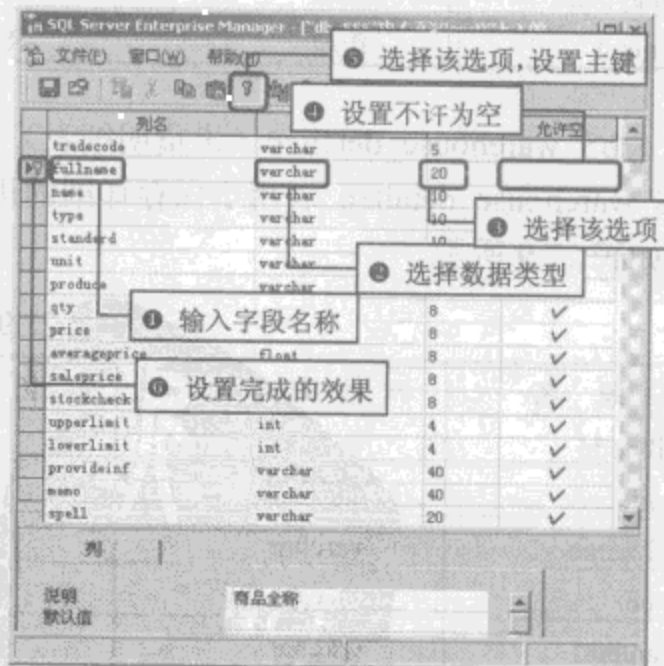



图 27.10 添加字段



(3) 单击“保存”按钮, 弹出“选择名称”对话框, 如图 27.11 所示, 输入要新建的表名 tbS\_stock, 单击“确定”按钮, 即可在数据库中添加一个表。

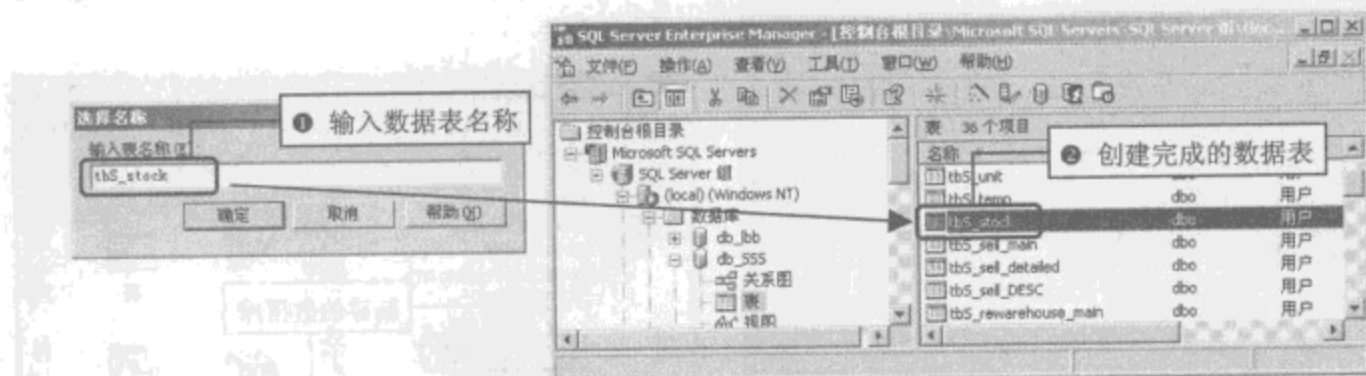


图 27.11 输入数据表的名称

由于篇幅有限, 其他数据表的创建过程就不予介绍, 相信读者能够举一反三, 结合下面给出的数据表结构, 创建数据表。

#### ☒ tbS\_warehouse\_main (进货主表)

tbS\_warehouse\_main 表用于保存商品进货的主要信息, 该表相对于进货明细表来说称为主表, 记录的信息比明细表要简略。该表的结构如表 27.8 所示。

表 27.8 进货主表

字段名	数据类型	长度	主键否	描述
billdate	datetime	8	否	进货日期
billcode	varchar	20	否	进货单编号
units	varchar	20	否	供应商名称
handle	varchar	10	否	经手人
summary	varchar	100	否	摘要
fullpayment	float	8	否	应付金额
payment	float	8	否	实付金额

#### ☒ tbS\_warehouse\_detailed (进货明细表)

tbS\_warehouse\_detailed 表用于保存详细的进货信息, 相对于进货主表来说称为进货明细表。该表的结构如表 27.9 所示。

表 27.9 进货明细表

字段名	数据类型	长度	主键否	描述
billcode	varchar	20	否	进货单编号
trade code	varchar	20	否	商品编号
fullname	varchar	20	否	商品名称
type	varchar	10	否	商品型号
standard	varchar	10	否	商品规格
produce	varchar	20	否	商品产地
unit	char	4	否	商品单位

续表

字段名	数据类型	长度	主键否	描述
qty	float	8	否	商品数量
price	float	8	否	商品进价
tsum	float	8	否	进货金额
billdate	datetime	8	否	进货日期

☒ tbS\_stock (库存表)

tbS\_stock 表用于保存商品的库存信息, 该表的结构如表 27.10 所示。

表 27.10 库存表

字段名	数据类型	长度	主键否	描述
trade code	varchar	5	否	商品编号
full name	varchar	20	是	商品全程
name	varchar	10	否	商品简称
type	varchar	10	否	商品型号
standard	varchar	10	否	商品规格
unit	varchar	10	否	商品单位
produce	varchar	20	否	商品产地
qty	float	8	否	库存数量
price	float	8	否	最后一次进货价格
average price	float	8	否	加权平均价
sale price	float	8	否	最后一次销售价格
stock check	float	8	否	盘点数量
upper limit	int	8	否	库存上限
lower limit	int	8	否	库存下限
provide inf	varchar	40	否	商品供货信息
memo	varchar	40	否	备注信息
spell	varchar	20	否	拼音简码

☒ tbS\_sell\_main (销售主表)

tbS\_sell\_main 表用于保存商品销售的主要信息, 该表的结构如表 27.11 所示。

表 27.11 销售主表


字段名	数据类型	长度	主键否	描述
billdate	datetime	8	否	销售日期
billcode	varchar	20	否	销售单编号
units	varchar	20	否	购货单位
handle	varchar	10	否	经手人
summary	varchar	100	否	摘要
full gathering	float	8	否	应收金额
gathering	float	8	否	实收金额

☒ tbS\_sell\_detailed (销售明细表)

tbS\_sell\_detailed 表用于保存商品销售的详细信息, 该表的结构如表 27.12 所示。

表 27.12 销售明细表

字段名	数据类型	长度	主键否	描述
billcode	varchar	20	否	销售单编号
tradeCode	varchar	20	否	商品编号
fullname	varchar	20	否	商品全称
type	varchar	10	否	商品类型
standard	varchar	10	否	商品规格
produce	varchar	20	否	商品产地
unit	varchar	4	否	商品单位
qty	float	8	否	销售数量
price	float	8	否	销售单价
tsum	float	8	否	销售金额
billdate	datetime	8	否	销售日期

 说明: 由于篇幅有限, 这里只列举了重要的数据表的结构, 其他的数据表结构请参见光盘中的数据库文件。

#### 27.4.4 数据表逻辑关系

为了使读者能够更好地了解库存商品信息表与其他各表之间的关系, 在这里给出数据表关系图, 如图 27.12 所示。通过图 27.12 的表间关系可以看出, 库存商品信息表和进货明细表、进货退货明细表、销售明细表以及销售退货明细表等都存在着一定的联系, 这些数据表通过 fullname 字段 (商品全称字段) 联系起来。

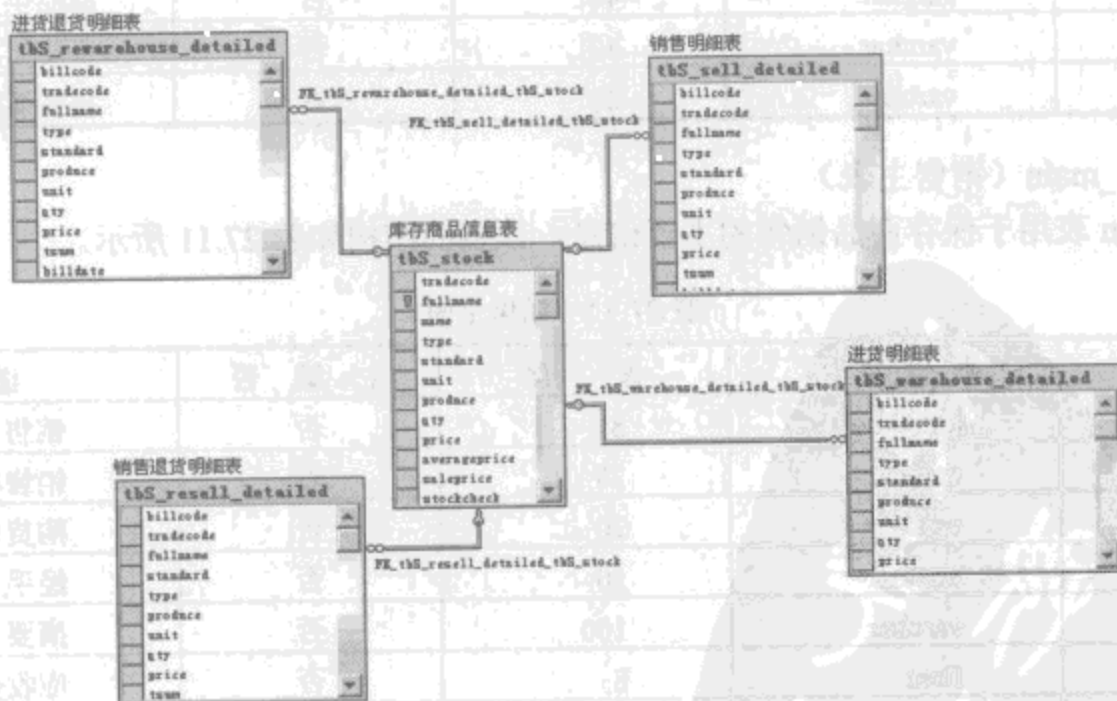


图 27.12 数据表关系 1

在商品的进货销货的一系列数据操作中,如果其中某些信息(如往来单位的相关信息)发生变化,其他的数据表中的相关信息也应该发生变化,这样才能保证数据的完整性和一致性。图 27.13 中列出了职员信息表、进货退货表、销售退货表、销售表、进货退货表、往来单位信息表以及往来对账明细表之间的关系,这些数据表之间利用 fullname 字段(商品全称字段)联系起来。

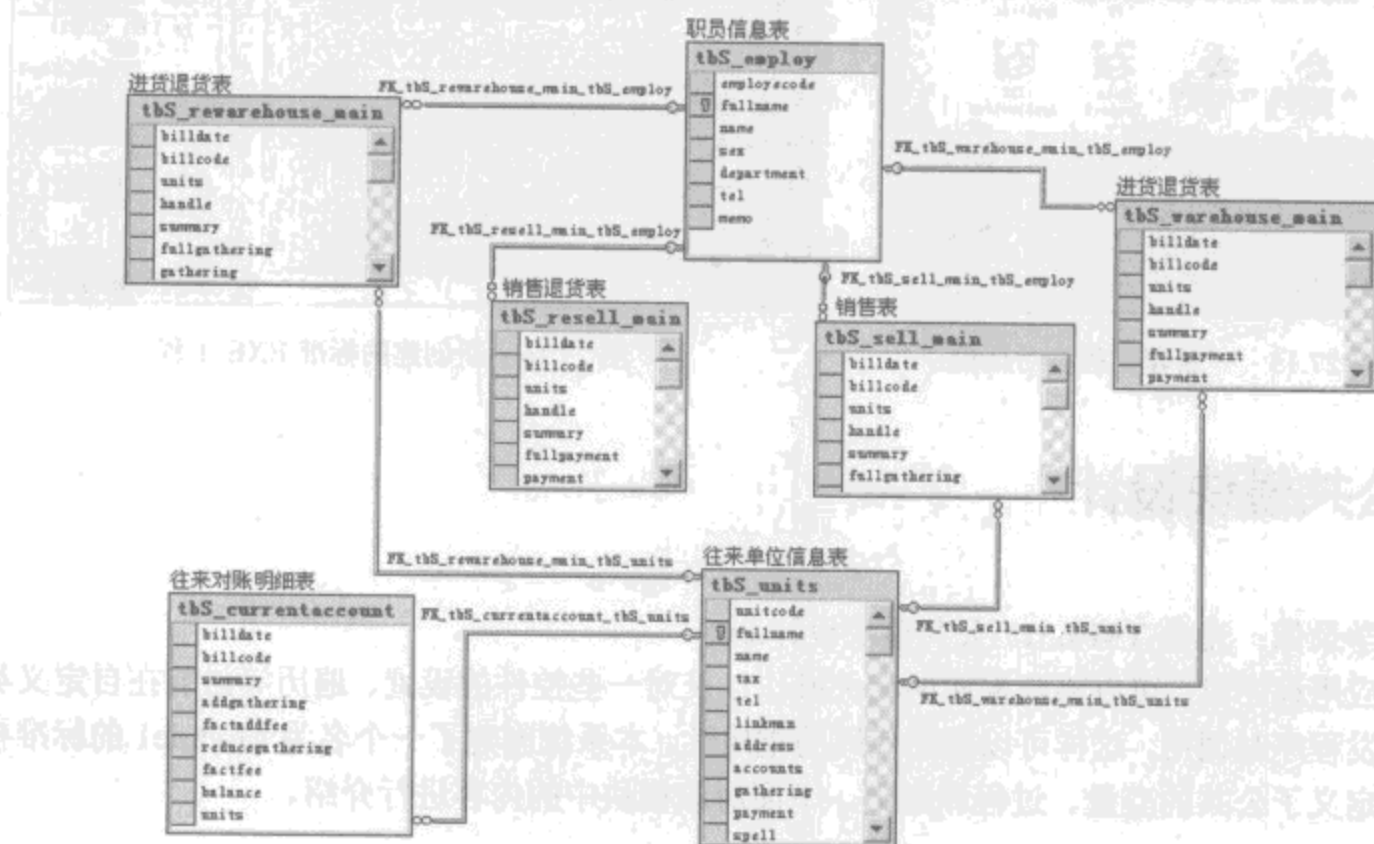


图 27.13 数据表关系 2

## 27.5 创建项目

教学录像: 光盘\TM\lx\27\创建项目.exe

在 VB 6.0 开发环境中创建项目的具体步骤如下。

(1) 选择“开始”/“所有程序”/“Microsoft Visual Basic 6.0 中文版”/“Microsoft Visual Basic 6.0 中文版”选项,如图 27.14 所示。



图 27.14 选择“Microsoft Visual Basic 6.0 中文版”选项

(2) 在启动 VB 开发环境时,会首先打开“新建工程”对话框。在该对话框中选择“新建”选项卡,选择“标准 EXE”图标,单击“打开”按钮,如图 27.15 所示,即可新建一个标准的 EXE 工程。

(3) 新创建的标准 EXE 工程,如图 27.16 所示。



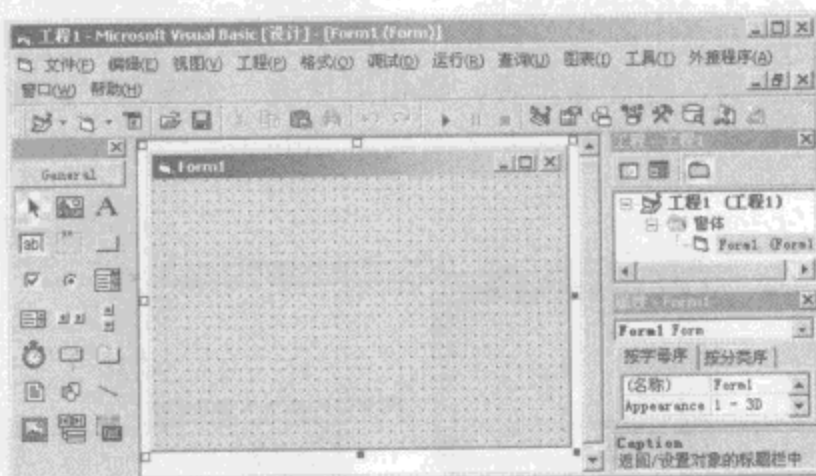


图 27.16 新创建的标准 EXE 工程

## 27.6 公共模块设计

在开发应用程序时,可以将数据库的相关操作以及对一些控件的设置、遍历等封装在自定义类中,便于在开发程序时调用,这样可以提高代码的重用性。本系统创建了一个名为 `Module1` 的标准模块,该模块中定义了公共的变量、过程和函数,下面对该模块中的内容进行介绍。

### 27.6.1 主函数

在一个应用程序中，一般都有一个主程序 Sub Main，程序的执行从这个主程序开始，当然也可以设置为其他的窗体。下面介绍将启动对象设置为 Sub Main 函数。

- (1) 选择“工程”/“进销存管理系统 属性”命令,将弹出“工程”对话框。
- (2) 在该对话框中选择“通用”选项卡,在“启动对象”组合框中选择 Sub Main 选项,单击“确定”按钮,完成启动对象的设置,如图 27.17 所示。

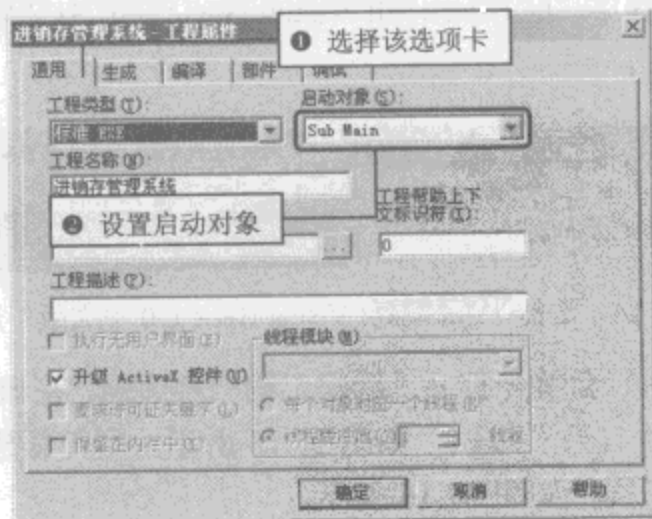


图 27.17 设置启动对象

本系统的主函数主要用于自动附加数据库，在数据库附加完成以后调用启动窗体，开始程序的正常执行。在数据库附加时，利用 `sp_attach_db` 语句进行附加，并利用 `Con` 的 `Execute` 方法执行数据库附加的 SQL 语句。

完成数据库附加以后，将数据连接字符串赋给字符串变量 `PublicStr`，方便其他窗体的调用。最后，即可显示出系统的启动窗体，关键代码如下：

```
Public Sub Main()                                '主函数
    '附加数据库
    On Error Resume Next
    con.ConnectionString = "Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa" '连接数据库
    con.Open                                       '打开数据库连接
    Set rs = New ADODB.Recordset                 '实例化 rs 对象
    '开始附加数据库
    str = "EXEC sp_attach_db @dbname = N'db_SSS', @filename1 = N" + App.Path + "\DataSource\db_SSS_
Data.MDF" + ", @filename2 = N" + App.Path + "\DataSource\db_SSS_Log.LDF" + ""
    Set rs = con.Execute(str)                    '执行附加语句
    con.Close                                    '关闭数据库连接
    '将数据库连接串赋给公共变量 PublicStr
    PublicStr = "Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=db_SSS"
    frm_Star.Show                                '显示启动窗体
End Sub
```

## 27.6.2 数据库连接函数

数据连接是在开发数据库管理系统中经常需要使用到的。在本程序中为了优化数据库连接，减少代码的使用，在公共模块中定义一个函数用于执行数据库连接。在编程过程中，如果需要使用数据库连接操作时，就可以直接调用数据库连接函数 `cnn()` 来操作数据库。也可以利用公共的变量 `PublicStr`，来给 ADO 控件的 `ConnectionString` 属性赋值，关键代码如下：

```
Public Function cnn() As ADODB.Connection        '共享数据库连接
    Set cnn = New ADODB.Connection              '实例化 cnn 对象
    '利用 cnn 执行数据连接操作
    cnn.Open "Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=db_SSS"
End Function
```

在 `Function cnn()` 函数中，`cnn` 对应的连接字符串参数说明如表 27.13 所示。

表 27.13 `cnn` 连接串的参数说明

参 数	描 述
<code>Provider=SQLOLEDB.1;</code>	代表数据库的提供者（本系统使用的 SQL Server）
<code>Persist Security Info=False;</code>	代表是否设置数据库的安全信息
<code>User ID=sa;</code>	代表 SQL Server 中用户名，安装 SQL Server 默认名称为 sa，密码为空
<code>Initial Catalog=db_SSS</code>	代表本系统使用的数据库

### 27.6.3 拼音简码函数

在进销存管理系统中进行商品的进货和销售的时候，可以直接输入商品的拼音简码来完成商品的检索功能，这样可以为信息的录入工作带来极大的方便。如，汽车厂的拼音简码为 QCC。下面介绍一下拼音简码函数的实现。

汉字属于 DBCS（双字节字符集），该集合中的字符用一个或两个字节来表示。它表示的字符允许多于 256 个。DBCS 字符集一般用于表示的书写系统环境中，例如日文、朝鲜文和中文。

常用汉字的机内码共 3989 个，按英语字母的顺序排列并编码，首汉字为“啊”对应首字母为 A，首汉字为“芭”对应首字母为 B，首汉字为“擦”对应首字母为 C。以 A 为首字母的汉字在“啊”和“芭”之间，以 B 为首字母的汉字在“芭”和“擦”之间，按照这个规律，可以获得常用汉字的拼音简码。利用汉字机内码编码规则，用程序生成汉字的拼音简码。

在判断汉字拼音简码时，使用了 Asc 函数。Asc 函数可以返回字符串首字母的字符值(ASCII 码值)。关键代码如下：

Public Function py(Mystr As String) As String	'获得汉字的拼音简码
On Error Resume Next	'错误处理
If Asc(Mystr) < 0 Then	'如果 ASCII 值小于 0
If Asc(Left(Mystr, 1)) < Asc("啊") Then	'如果 ASCII 值小于 A
py = "0"	'拼音码为 0
Exit Function	'退出过程
End If	
If Asc(Left(Mystr, 1)) >= Asc("啊") And Asc(Left(Mystr, 1)) < Asc("芭") Then	'ASCII 介于啊芭之间
py = "A"	'拼音码为 A
Exit Function	'退出过程
End If	
If Asc(Left(Mystr, 1)) >= Asc("芭") And Asc(Left(Mystr, 1)) < Asc("擦") Then	'如果介于芭擦之间
py = "B"	'拼音码为 B
Exit Function	'退出系统
End If	
'...此处代码有省略	
Else	'如果 ASCII 不小于 0
If UCase(Mystr) <= "Z" And UCase(Mystr) >= "A" Then	'如果介于 A 和 Z 之间
py = UCase(Left(Mystr, 1))	'拼音码为该字母大写
Else	'否则
py = Mystr	'拼音码等于输入的值
End If	
End If	
End Function	

## 27.7 启动窗体的设计

 教学录像：光盘\TM\lx\27\启动窗体的设计.exe

启动窗体也称为闪屏或者欢迎屏，是在应用程序启动的时候一闪而过的窗体界面，它可以为用户

提示一定的信息,用户无须对其进行任何的操作。

闪屏是应用程序中最先显示给用户的一个界面,主要用于数据加载的延时情况。在数据加载时,用这样一个闪屏显示出来,避免由于等待时间过长而产生的焦虑。企业进销存管理系统中的启动窗体如图 27.18 所示。



图 27.18 启动窗体

### 27.7.1 设计窗体界面

启动窗体的界面设计过程如下:

(1) 在工程中新建一个窗体,将窗体的名称命名为 `frm_Star`, `BorderStyle` 属性设置为 `0-None`, `Picture` 属性设置为指定的图片, `StartPosition` 属性设置为“2-屏幕中心”。

(2) 在窗体上添加一个 `ShockwaveFlash` 控件,使用其默认名,用于显示事先设计好的 Flash 动画。该控件不是 VB 的标准控件,因此在使用时,需要通过安装 Flash 软件或者注册 Flash 组件来获得。这里使用的是 Flash 8 提供的 OCX 控件 `Flash8.ocx`。在 VB 中并没有提供这个控件,但是可以通过下面两种方式获取:一种是安装 Flash 软件,另一种是直接注册 `Flash8.ocx` 控件。注册 `ShockwaveFlash` 控件的方法如下:

将 `Flash8.ocx` 复制到 `C:\WINDOWS\system32` 目录下,然后选择“开始”/“运行”命令,在“运行”对话框中,输入“`regsvr32 C:\WINDOWS\system32\Flash8.ocx`”,如图 27.19 所示。

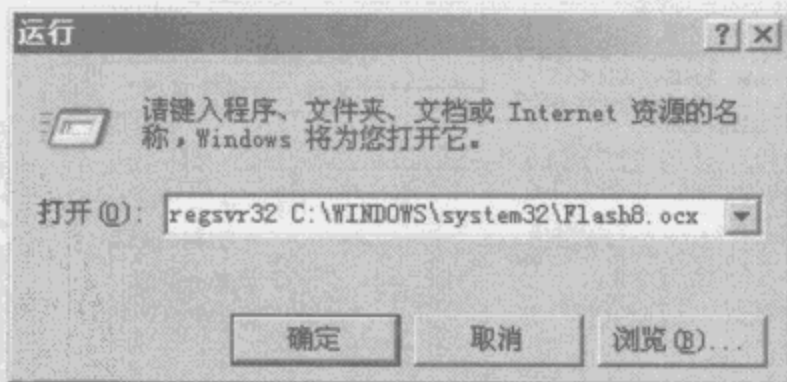


图 27.19 注册 Flash 控件

说明: 此处的 C 盘为系统盘,如果读者的系统盘不是 C 盘,可以将其放置在系统盘的对应目录下。



单击“确定”按钮，注册控件，当弹出如图 27.20 所示的对话框时，则说明控件注册成功。

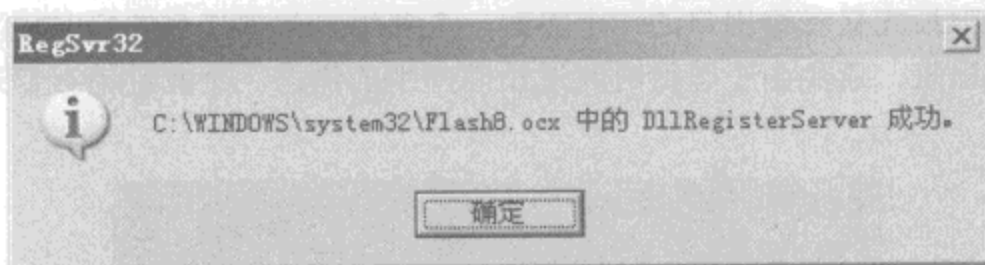


图 27.20 控件注册成功

控件注册成功以后，需要将其添加到 VB 的工程中。选择“工程”/“部件”菜单项，在弹出的“部件”对话框中选择“控件”选项卡，选中 Shockwave Flash 复选框，将其添加到 VB 的工程中，其中 ☐ 就是 ShockwaveFlash 控件。

(3) 在窗体上添加一个 Timer 控件，使用其默认名，用于调用登录窗体，设置 Interval 属性为 1500。

## 27.7.2 添加资源文件

在“资源编辑器”窗口中，可以对与工程相关联的资源文件(.res)中的资源进行添加、删除和编辑。要进行说明的是，资源编辑器一次只能编辑一个资源文件，而在一个工程中只能包含一个资源文件。

### 1. 资源编辑器的加载

在默认情况下，资源管理器没有被加载到 VB 的集成开发环境中。如果要使用资源管理器，需要首先将其添加到 VB 的集成开发环境中，具体的添加方法如下：

(1) 选择“外接程序”/“外界程序管理器”菜单项，将弹出“外接程序管理器”对话框。

(2) 在该对话框中，双击“VB 6 资源编辑器”，当在对应的“加载行为”栏中出现“加载”字样时，单击“确定”按钮，完成资源编辑器的加载，如图 27.21 所示。

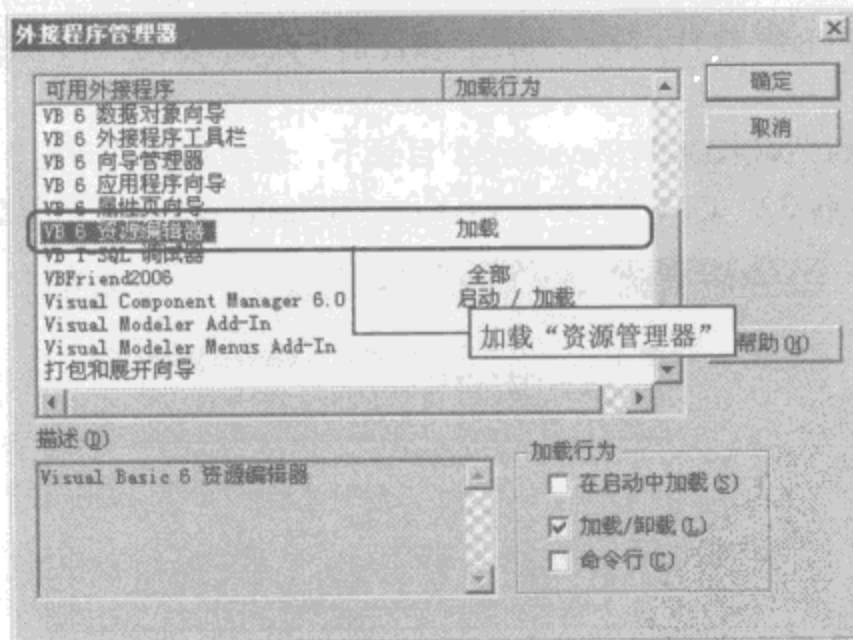


图 27.21 加载资源编辑器

(3) 加载以后，在 VB 工程的“标准”工具栏中就可以看到资源编辑器的图标，如图 27.22 所示。

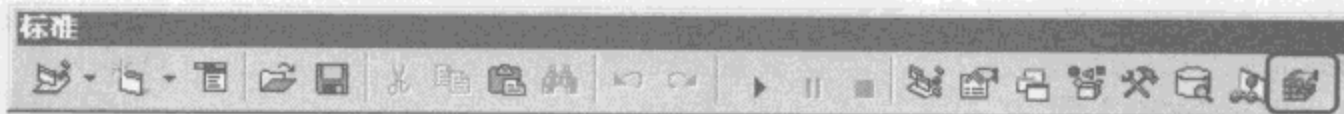




图 27.22 在“标准”工具栏中的“资源编辑器”

将“资源编辑器”加载到工程中就可以使用它了。“资源编辑器”窗口是可连接的。可用如下方法打开“资源编辑器”窗口：

- ☒ 单击“工具”菜单中“资源编辑器”命令。
- ☒ 使用 VB 的“标准”工具栏上的“资源编辑器”工具栏按钮。
- ☒ 如果资源文件已经存在，可以通过在“工程”窗口中，双击资源文件或选中资源文件并按下〈Enter〉键。

## 2. 添加自定义资源

利用资源管理器可以添加的资源很多，如文字、位图、图标等，也可以利用自定义资源向工程中添加 OCX 文件。这里利用该功能，向工程中添加自定义资源，具体的步骤如下：

- (1) 打开 VB 的资源编辑器，单击工具栏按钮，添加自定义资源。
- (2) 弹出“打开一个自定义资源”对话框。在该对话框中选择需要添加的自定义资源，这里为 Flash8.ocx 组件，单击“打开”按钮，即可将该资源添加到资源编辑器中。
- (3) 单击资源编辑器工具栏上的保存按钮，即可将该资源保存到 VB 的工程中，执行过程如图 27.23 所示。

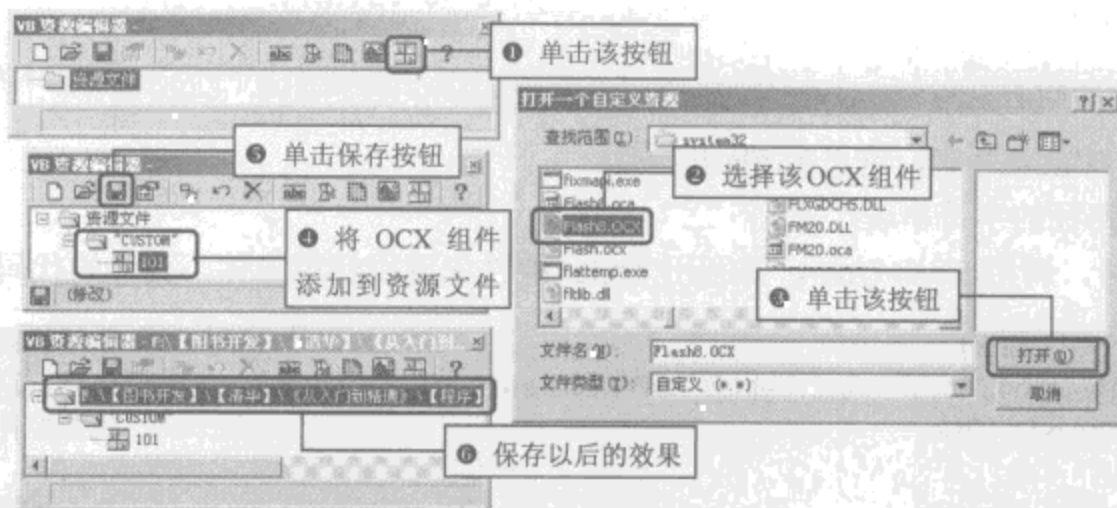


图 27.23 添加资源文件

### 27.7.3 代码注册 Flash 控件

Flash 控件的注册和一般的第三方控件的注册是一样的，可以利用 Regsvr32.exe 工具来手动注册，注册以后就可以正常在工程中使用。这里介绍直接从资源文件中将其复制到系统路径下，并利用 Shell 函数，在 DOS 下执行 regsvr32 方法将该控件注册到所在计算机中。上述过程在窗体的初始化事件中被执行，具体的程序代码如下：

```
Private Declare Function GetSystemDirectory Lib "kernel32" Alias "GetSystemDirectoryA" (ByVal lpBuffer As String, ByVal nSize As Long) As Long
```

Private Sub Form_Initialize()	'窗体初始化
On Error Resume Next	'错误处理
Dim str As String * 255	'定义字符型变量
GetSystemDirectory str, Len(str)	'获得系统路径
Dim str2 As String	'定义变量
str2 = Trim(Replace(str, Chr(0), ""))	'将去除路径中的空格
Dim i() As Byte	'定义数组变量
i = LoadResData(101, "CUSTOM")	'装载资源
Open str2 & "\Flash8.OCX" For Binary Access Write As #1	'打开 Flash8.ocx 文件
Put #1, , i	'向文件中写入数据
Close #1	'关闭文件
Do Until str2 & "\Flash8.OCX" <> ""	'循环执行
DoEvents	'转让控制权
Loop	
Shell "cmd.exe /c regsvr32 " & str2 & "\Flash8.OCX" & " /s", vbHide	'执行注册语句
End Sub	

#### 27.7.4 调用 Flash 动画

在本窗体中使用了 Flash 控件，在该窗体启动时，加载事先设计好的 Flash 动画，这里利用 Flash 控件的 Movie 属性来实现。该属性用于设置一个路径，确定 Flash 控件播放的 Flash 动画文件的所在位置。

语法：

object.Movie [=string]

object: 一个对象表达式，这里为 Flash 控件。

string: 一个路径的字符串表达式。

在窗体启动时，调用 Flash 动画，并利用 Timer 控件来调用登录窗体，具体的代码如下：

Private Sub Form_Load()	'窗体加载
ShockwaveFlash1.Movie = App.Path & "\SWF\sss.swf"	'自动识别 Falsh 文件路径
Timer1.Enabled = True	'设置控件可用
End Sub	
Private Sub Timer1_Timer()	'Timer 事件
Frm_mm.Show	'显示登录窗体
Unload Me	'关闭本窗体
End Sub	

### 27.8 系统登录窗体设计

 教学录像：光盘\TM\lx\27\系统登录窗体设计.exe

 本模块使用的数据表：tbS\_power

系统登录窗体主要用于对登录到企业进销存管理系统中的用户进行安全检查，以防止非法用户进

入到本系统中。只有合法的用户才能登录本系统。

系统登录窗体的主要功能是通过数据库表 tbS\_powe 的查询, 结合 IF 语句判断选定的用户及输入的密码是否符合数据库中用户名和密码, 如果符合则运行登录, 否则提示错误信息, 如果错误输入超过 3 次, 则强行退出系统。系统登录窗体的界面效果如图 27.24 所示。

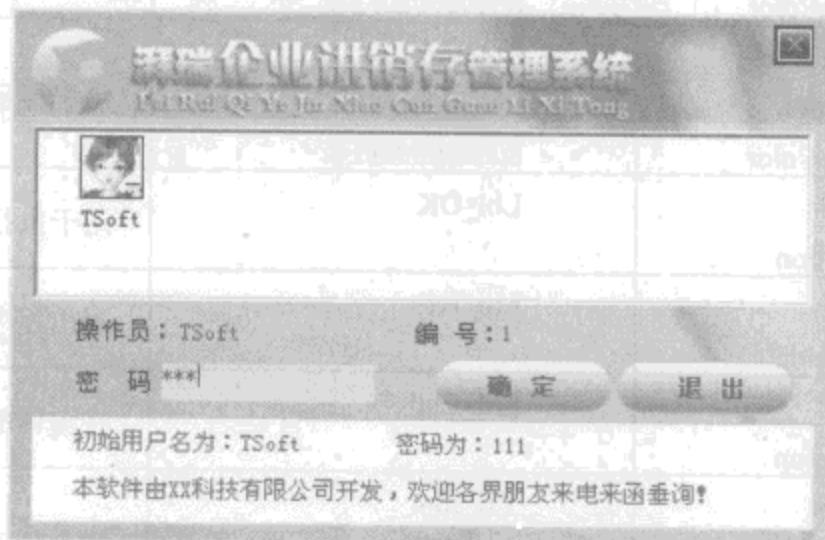


图 27.24 系统登录界面

### 27.8.1 设计窗体界面

系统登录窗体的界面设计过程如下。

(1) 在工程中新建一个窗体, 将窗体的名称命名为 Frm\_mm, BorderStyle 属性设置为 0-None, StartUpPosition 属性设置为“2-屏幕中心”。

(2) 在窗体上添加一个 PictureBox 控件, 使用其默认名, 用于显示事先设计好的图片。

(3) 在 Picture1 上添加一个 ListView 控件和一个 ImageList 控件, 使用默认名。这两个控件都是 ActiveX 控件, 在使用前需要将其添加到 VB 的工程中, 具体的添加方法如下:

选择“工程”/“部件”命令, 在弹出的“部件”对话框中选择“控件”选项卡, 选中 Microsoft Windows Common Controls 6.0 (SP6) 项, 单击“确定”按钮, 即可将 ListView 控件和 ImageList 控件添加到 VB 的工程中。

(4) 在窗体上添加一个 ADO 控件, 使用其默认名, 该控件用于连接用户权限表。该控件属于 ActiveX 控件, 在使用前需要将其添加到 VB 的工程中, 具体的添加方法: 选择“工程”/“部件”命令, 在弹出的“部件”对话框中选择“控件”选项卡, 选中 Microsoft ADO Data Control 6.0 (SP6) 项, 单击“确定”按钮, 即可将 ADO 控件添加到 VB 的工程中。

(5) 在 Picture1 上添加 6 个 Label 控件和一个 TextBox 控件, 具体的设置如表 27.14 所示。

表 27.14 Label 控件和 TextBox 控件的属性设置

对 象	属 性	值	功 能
LabelA	名称	Lbl_Name	用于显示用户姓名
	Caption	Lbl_Name	
	ForeColor	&H000080FF&	



续表

对 象	属 性	值	功 能
LabelA	名称	Lbl_Czyid	用于显示用户编号
	Caption	Lbl_Czyid	
	ForeColor	&H000080FF&	
	名称	Lbl_Infor	用于显示初始的用户名和密码信息
	Caption	Lbl_Infor	
	ForeColor	&H000080FF&	
LabelA	名称	Lbl_OK	用于执行“确定”按钮操作
	Caption	Lbl_OK	
	名称	Lbl_Exit	退出程序的执行
	Caption	Lbl_Exit	
	名称	Lbl_End	在窗体的右上角，用于退出程序
TextBoxlabl	名称	Txt_mm	用于输入用户的密码
	Text	Txt_mm	
	ForeColor	&H000080FF&	

(6) 在窗体上添加一个 TextBox 控件，命名为 Txt\_Time，用于记录用户输入密码的次数，如果超过 3 次将强制退出。

## 27.8.2 向 ListView 控件中添加用户名

在窗体加载时，在 ListView 控件中将显示出当前系统中的所有操作员以及操作员的图像。这里在窗体加载的时候利用 ADO 控件和 SQL 语句查询数据表 tbS\_powe 中的操作员信息，并将其添加到 ListView 控件中。利用在数据表中存储的头像信息，获取出与 ImageList 控件中相对应的图片，将其添加到 ListView 控件中，并将“密码”文本框中的内容清空。关键代码如下：

```

Dim rs As New ADODB.Recordset           '定义记录集变量
Dim itmX As ListItem                    '声明一个 ListItem 对象
Dim Mylcon As Integer                  '声明一个整型变量
Dim Mystr As String                    '声明字符串变量
Private Sub Form_Load()                '窗体加载
    '设置标签内容
    Lbl_Infor.Caption = "初始用户名为: TSoft      密码为: 111      " + Chr(10) + Chr(10) + "本软件由 XX
    科技有限公司开发，欢迎各界朋友来电来函垂询！"
    rs.Open "select * from tbS_power", cnn, adOpenKeyset           '查询用户权限表
    If rs.RecordCount > 0 Then           '如果查询记录大于零
        rs.MoveFirst                    '移动到记录头
        Lbl_Czyid.Caption = rs.Fields("userid")                    '显示用户编号
        Lbl_Name.Caption = rs.Fields("sysuser")                    '显示用户姓名
        '向 ListView 中添加图片
        Do While rs.EOF = False           '循环到记录尾

```

```

        Mystr = rs.Fields("sysuser")           '将用户名赋给变量
        Mylcon = Val(Right(rs.Fields("head"), Val(Len(rs.Fields("head")) - 2))) '将头像编号赋给变量
        Set itmX = ListView1.ListItems.Add(, , Mystr, Mylcon) '向 ListView 添加项目
        rs.MoveNext                             '移动到下一条
    Loop
End If
rs.Close                                     '关闭记录
Txt_mm.Text = ""                             '清空“密码”文本框
End Sub

```

### 27.8.3 添加用户名和编号

当窗体启动以后，用户单击用户头像，将显示用户的用户名和编号，并将焦点设置在“密码”文本框中。这里主要应用了 ListView 控件的 Click 事件，以及 SelectedItem 属性。Click 事件用于处理 ListView 控件的单击事件，当用户单击 ListView 控件中的项目时将触发该事件，同时利用 SelectedItem 属性获得所选项目的名称，并利用该名称查询出用户名和用户编号。关键的代码如下：

```

Private Sub ListView1_Click()                 '单击事件
    If ListView1.ListItems.Count > 0 Then     '如果 ListView 中有项目
        '查询操作员信息
        rs.Open "select * from tbS_power where sysuser =" + ListView1.SelectedItem + "", cnn, adOpen
        Keyset
        If rs.RecordCount > 0 Then           '如果查询结果大于零
            '给相应的控件赋值
            Lbl_Czyid.Caption = rs.Fields("userid") '显示用户编号
            Lbl_Name.Caption = rs.Fields("sysuser") '显示用户姓名
        End If
        rs.Close                             '关闭记录集
        Txt_mm.SetFocus                       '“密码”文本框获得焦点
    End If
End Sub

```

### 27.8.4 判断用户名和密码

用户选中用户名以后，在“密码”文本框中输入该用户的密码，然后单击“确定”按钮，进入系统内部。当用户单击“确定”按钮时，系统将核对输入的用户密码和数据库中的密码是否相同，如果输入正确，则进入到系统内部，如果不正确则提示信息，并将输入的次数添加到 Txt\_Time 文本框中。当 Txt\_Time 文本框中的次数超过 3 时，则提示信息，退出系统。关键代码如下：

```

Private Sub Lbl_OK_Click()                   '确定
    '打开 rs 记录集
    rs.Open "select * from tbS_power where sysuser =" + Trim(Lbl_Name.Caption) + "", cnn, adOpenKeyset
    If rs.RecordCount > 0 Then               '如果记录数大于零
        '验证操作员及密码
        If Txt_mm.Text = rs.Fields("password") Then '如果输入的密码正确
            Load MDIForm1                   '加载主窗体
        End If
    End If
End Sub

```

```

MDIForm1.Show
Unload Me
Else
    If Lbl_Name.Caption = "" Then
        MsgBox "请选择操作员!", , "信息提示"
        ListView1.SetFocus
    Else
        If Txt_mm.Text <> rs.Fields("password") Then
            MsgBox "密码错误,请重新输入密码!", , "信息提示"
            Txt_Time.Text = Val(Txt_Time.Text) + 1
            Txt_mm.SetFocus
        End If
    End If
    If Txt_Time.Text = "3" Then
        MyMsg = MsgBox("密码输入错误,请向系统管理员查询!", , "信息提示")
        If MyMsg = vbOK Then End
    End If
End If
rs.Close
End Sub

```

'显示主窗体  
'关闭登录  
'否则  
'如果用户名为空  
'提示信息  
'将焦点设置在 ListView 中  
'如果用户名不为空  
'如果密码不相同  
'提示错误信息  
'错误次数加一  
'“密码”文本框获得焦点  
'密码错误 3 次, 退出系统  
'提示信息  
'如果用户单击“确定”按钮, 退出系统  
'关闭记录集

### 27.8.5 移动无标题栏窗体

在设计程序的时候, 为了窗体界面的美观, 会通过设置窗体的 `BorderStyle` 属性将窗体的标题栏设置为隐藏效果, 用一张图片覆盖整个窗体, 以达到美观的效果。但是这样也给用户的操作带来了一定的麻烦, 因为窗体没有标题栏, 用户不能通过鼠标移动窗体, 窗体只能显示在启动的位置, 不能移动, 直到被关闭为止。

这里可以利用 API 函数来实现窗体移动的功能, 首先利用 `ReleaseCapture` 函数释放鼠标捕获, 然后利用 `SendMessage` 函数向窗体发送消息, 达到窗体移动的效果。关键代码如下:

```

Const HTCAPTION = 2
Const WM_NCLBUTTONDOWN = &HA1
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wParam As Long, ByVal wParam As Long, lParam As Any) As Long
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 1 Then
        Dim ReturnVal As Long
        X = ReleaseCapture()
        ReturnVal = SendMessage(hwnd, WM_NCLBUTTONDOWN, HTCAPTION, 0)
    End If
End Sub

```

'定义常数  
'定义常数  
'声明 API 函数  
'声明 API 函数  
'如果按下鼠标左键  
'定义长整型变量  
'释放鼠标捕获  
'发送消息

## 27.9 主窗体设计

 教学录像：光盘\TM\lx\27\主窗体设计.exe

主窗体是程序操作过程中必不可少的，它是人机交互中的重要环节。通过主窗体，用户可以调用系统相关的各子模块，快速掌握本系统中所实现的各个功能。企业进销存管理系统中，当登录窗体验证成功后，用户将进入主窗体。主窗体被分为 4 个部分：最上面是系统菜单栏，可以通过它调用系统中的所有子窗体；菜单栏下面是利用 Flash 控件制作的工具栏，它以按钮的形式使用户能够方便地调用最常用的子窗体；窗体的中间区域是一个和程序主题相关的背景图片；窗体的最下面，用状态栏显示当前所打开窗体、当前登录的用户名等信息。主窗体运行结果如图 27.25 所示。



图 27.25 企业进销存管理系统主界面

### 27.9.1 设计窗体界面

主窗体的界面设计过程如下：

- (1) 在工程中新建一个 MDI 窗体，使用窗体的默认 MDIForm1，Caption 属性设置为“企业进销存管理系统”。
- (2) 利用菜单编辑器在窗体上设计菜单栏，具体的设计方法请参见 27.9.2 节相关内容。
- (3) 在窗体上添加一个 PictureBox 控件，使用该控件的默认名，设置 Align 属性为“1-Align Top”，设置 Picture 属性为指定的图片。该图片框用于充当其他控件的容器。
- (4) 在 Picture1 中添加一个 ShockwaveFlash 控件，用于显示事先设计好 Flash 动画。将其命名为 Flash1。
- (5) 在 Picture1 上添加一个 PictureBox 控件，将其命名为 Pic\_Flex，用于显示辅助工具栏。设置 Picture 属性为事先设置好的图片。



(6) 在 Pic\_Flex 上添加 Lbl\_Pic 控件数组 (0~5) 和 Lbl\_info 控件数组 (0~5), 设置 BackStyle 属性为 “0-Transparent”。

(7) 在窗体上添加一个 Timer 控件, 设置 Interval 属性为 30, 用于控制浮动工具栏的伸展和收缩。

(8) 在窗体上添加一个 StatusBar 控件, 充当窗体的状态栏, 具体请参见第 27.9.5 节的相关内容。

## 27.9.2 设计菜单栏

利用 VB 自带的菜单编辑器给窗体添加菜单。通过选择 “工具” / “菜单编辑器” 命令来打开菜单编辑器, 菜单编辑器的使用和菜单的设计读者可以参见本书第 10 章的相关内容, 这里通过菜单编辑器设计的菜单如图 27.26 所示。

标题	名称	快捷键	标题	名称	快捷键
基础信息(&J)	menu1		....往来分析(与代理商对账)	file_supply	
....库存商品	Stock		....商品销售成本表	file_sellCost	
....往来单位	Unit		库存信息(&B)	menu4	
....内部职员	Employee		....库存状况	file_stockStatus	F4
....-	w1		....-	ll5	
....退出系统(&X)	file_exit		....库存商品数量上限报警	file_UL	Ctrl+U
进货信息(&C)	menu2		....库存商品数量下限报警	file_FL	
....进货单	file_bill	F2	....-	ll6	
....进货退货	file_stockExit	Ctrl+F2	....商品进销存变动表	file_buySellChange	
....-	ll1		....库存盘点(自动盘盈盘亏)	file_stockCheck	
....进货分析	file_analyse	F9	系统维护(&T)	menu5	
....进货统计(不含退货)	file_stat		....操作员设置	file_op	
....-	ll2		....操作员权限设置	file_setSys	Ctrl+Q
....往来分析(与供应商对账)	file_CAG	F12	....-	lp2	
销售信息(&S)	menu3		....数据备份与恢复	file_BR	Ctrl+B
....销售单	file_sell	F3	....数据清理	file_dataClear	
....销售退货	file_sellStock	Ctrl+F3	帮助信息(&S)	menu6	
....-	ll3		....本单位信息	file_unit	
....销售统计(不含退货)	file_sellStat		....-	lq1	
....月销售状况	file_monthSell	F8	....帮助信息	file_help	F1
....商品销售排行	file_sellTaxis		....-	q2	
....-	ll4		....系统关于	file_softEdition	

图 27.26 企业进销存管理系统菜单

设计好菜单以后, 就可以给菜单添加代码了。给菜单添加代码的方法非常简单, 如选择 “基础信息” / “库存商品” 命令, 即可进入到代码编辑区域中, 在该区域中编写代码如下, 实现的功能是调用库存信息窗体。

```
Private Sub Stock_Click()  
    frm_basic_stock.Show  
End Sub
```

'单击“库存信息”命令  
'显示库存信息窗体

### 27.9.3 利用 Flash 设计工具栏

窗体的工具栏是利用 Flash 动画设计的,但是在 MDI 主窗体上不能添加 ShockwaveFlash 动画,因此只能将该控件放置在 PictureBox 控件中。这里利用 Flash 动画调用系统中的常用功能,具体的效果如图 27.27 所示。



图 27.27 Flash 效果

利用 Flash 和 VB 进行交互。实现 VB 和 Flash 交互时,需要用到一个 FSCommand 命令,在 Flash 的 ActionScript 里面有一个 FSCommand()函数,该函数的主要功能就是发送 FSCommand 命令。利用 Flash 中的 FSCommand()函数向 VB 应用程序发送 Command 命令,在 VB 中应用程序捕获 ShockWaveFlash 控件的 FSCommand 事件,接收 Command 命令,从而达到 Flash 和 VB 交互的目的,具体的执行流程如图 27.28 所示。



图 27.28 Flash 动画控制应用程序的执行过程

首先,设置 FSCommand 命令。下面以图 27.27 所示的“商品进货”按钮为例进行介绍。在 Flash 中,设计“商品进货”按钮时,添加如下代码。这里利用序号作为参数进行传递。

```
on(release){
fscommand("1","1");
}
```

其中,第 1 个“1”为 Command 命令,第 2 个“1”为参数,这里利用“1”与 VB 进行交互。将 Flash 都设计好以后,保存输出。

下面介绍在 VB 中接收 FSCommand 命令。

在 VB 工程中,将 ShockwaveFlash 控件添加到窗体上以后,将其改名为 Flash1,在代码编辑区中找到 Flash 控件的 FSCommand 事件,代码如下:

```
Private Sub Flash1_FSCommand(ByVal command As String, ByVal args As String)

End Sub
```

该事件用于接收 Flash 发送的 FSCommand 命令的事件,当 Flash 发送的 FSCommand 命令和 VB 中 FSCommand 事件的 command 参数相一致时,将触发其后面的操作。如当用户单击“商品进货”按钮时,Flash 动画将发出“1”这个 FSCommand 命令,VB 中的 FSCommand 事件捕获该 FSCommand 命令,并执行其后面的操作,这里为调用商品进货窗体。

其他按钮操作和“商品进货”按钮的实施过程基本一致,其具体的代码如下。

```

Private Sub Flash1_FSCommand(ByVal command As String, ByVal args As String)
    Select Case command
    Case 1
        frm_stockBill.Show
    Case 2
        frm_saleBill.Show
    Case 3
        frm_Stock.Show
    Case 4
        frm_checkStock.Show
    Case 5
        bTimeFlag = True
        Pic_Flex.Width = 0
        Pic_Flex.Visible = True
    End Select
End Sub

```

'判断单击的是那个按钮  
 '商品进货  
 '显示“商品进货”窗体  
 '商品销售  
 '显示“商品销售”窗体  
 '库存情况  
 '显示“库存情况”窗体  
 '库存盘点  
 '显示“库存盘点”窗体  
 '辅助工具  
 '标识变量设置为 True  
 '浮动工具栏宽度设置为零  
 '浮动工具栏可见

#### 27.9.4 利用图片设计浮动工具栏

在 Flash 工具栏上单击“辅助工具”按钮，将弹出浮动工具栏，如图 27.29 所示。

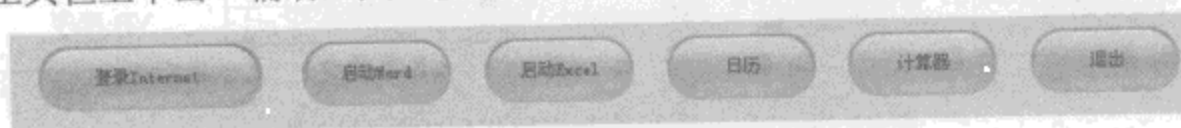


图 27.29 浮动工具栏

该工具栏利用 PictureBox 控件以及 Label 控件数组设计而成。在 Picture1 中添加一个 PictureBox 控件，命名为 Pic\_Flex，用于设置为浮动的工具栏；在 Pic\_Flex 中添加事先设计好的图片，该图片中，只留下了按钮框，并没有留下按钮的文字；这里在按钮上添加一个 Label 控件，命名为 Lbl\_info，用于显示按钮名称。在 Lbl\_info 控件数组的上面添加一个 Lbl\_Pic 控件数组，用于触发单击事件和鼠标移动事件。

在使用浮动工具栏时，利用 Timer 控件来调整 Pic\_Flex 控件的宽度。首先定义一个窗体级 Boolean 型变量 bTimeFlag，用于标识浮动工具栏是伸展还是收缩。当 bTimeFlag 设置为 True，标识浮动工具栏伸展，当 bTimeFlag 设置为 False，标识浮动工具栏收缩，关键代码如下：

```

Dim bTimeFlag As Boolean
Private Sub Timer1_Timer()
    Dim i As Integer
    If bTimeFlag = True Then
        If Pic_Flex.Width < 11934 Then
            For i = 1 To 100
                Pic_Flex.Width = Pic_Flex.Width + 2
            Next i
        End If
    Else
        If Pic_Flex.Width > 10 Then
            For i = 1 To 100
                Pic_Flex.Width = Pic_Flex.Width - 2
            Next i
        End If
    End Sub

```

'定义变量用于标识浮动工具条的伸缩  
 '定义整型变量  
 '如果标识变量为 True  
 '如果宽度小于 11934  
 '从 1 到 100 做循环  
 '浮动工具栏宽度自加 2  
 '如果标识变量为 False  
 '如果宽度大于 10  
 '从 1 到 100 做循环  
 '浮动工具栏宽度自减 2

```

End If
End If
End Sub

```

当用户单击浮动工具栏上的辅助工具按钮时,触发 Lbl\_Pic 控件数组的单击事件,执行对应的功能,关键代码如下:

```

Private Sub Lbl_Pic_Click(Index As Integer)
    Select Case Index
        Case 0
            ShellExecute Me.hwnd, "open", "http://www.mrbccd.com", 1, 1, 5
        Case 1
            ShellExecute Me.hwnd, "open", "winword.exe", "", 1, 5
        Case 2
            ShellExecute Me.hwnd, "open", "excel.exe", "", 1, 5
        Case 3
            frm_DATE.Show
        Case 4
            Dim ReturnValue, I
            ReturnValue = Shell("Calc.EXE", 1)
            AppActivate ReturnValue
        Case 5
            bTimeFlag = False
    End Select
End Sub

```

'Lbl\_Pic 控件数组单击事件  
'判断数组序号  
'单击 Lbl\_Pic (0)  
'登录 Internet  
'单击 Lbl\_Pic (1)  
'启动 Word  
'单击 Lbl\_Pic (2)  
'启动 Excel  
'单击 Lbl\_Pic (3)  
'调用“日历”窗体  
'单击 Lbl\_Pic (4)  
'定义变量  
'运行计算器  
'激活计算器  
'单击 Lbl\_Pic (5)  
'标识变量设置为 False

当用户鼠标在按钮上移动时,对应的该按钮上的文字变成红色;当鼠标移开时,该按钮上的文字颜色又变回为黑色。在 Lbl\_Pic 控件数组上移动时,按钮文字的颜色变为红色,代码如下:

```

Private Sub Lbl_Pic_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    Lbl_info(Index).ForeColor = RGB(255, 0, 0)
End Sub

```

'文字颜色设置为红色

当鼠标在窗体或 Pic\_Flex 控件上移动时,文字的颜色又被设置回黑色,关键代码如下:

```

Private Sub MDIForm_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    '鼠标在窗体上移动
    Dim i As Integer
    For i = 0 To Lbl_info.UBound
        Lbl_info(i).ForeColor = RGB(0, 0, 0)
    Next i
End Sub

Private Sub Pic_Flex_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    '鼠标在 Pic_Flex 上移动
    Dim i As Integer
    For i = 0 To Lbl_info.UBound
        Lbl_info(i).ForeColor = RGB(0, 0, 0)
    Next i
End Sub

```

'定义变量  
'循环控件数组  
'设置文字颜色为黑色  
  
'定义变量  
'循环控件数组  
'设置文字颜色为黑色



## 27.9.5 设计状态栏

窗体中的状态栏利用 StatusBar 控件设计而成。该控件不是标准的 ActiveX 控件，在使用前需要将其添加到窗体上，具体的添加方法如下：

选择“工程”/“部件”命令，在弹出的“部件”对话框中选中 Microsoft Windows Common Controls 6.0 (SP6) 项，将 StatusBar 控件添加到工具箱中。将该控件添加到窗体上。

用鼠标右键单击该窗体，在弹出的快捷菜单中选择“属性”命令，即可弹出“属性页”对话框，通过“属性页”对话框将状态栏设置为 5 个窗格，分别用于显示当前的窗体信息、当前的操作员信息、网址、日期和时间，如图 27.30 所示。



图 27.30 状态栏效果

其中，网址是通过在“属性页”对话框中的“窗格”选项卡中的“文本”文本框中进行设置的，如图 27.31 所示。



图 27.31 通过“属性页”对话框设置网址

日期窗格是通过在“窗格”选项卡中设置“样式”为 6-sbrDate。时间窗格是通过设置“样式”为 5-sbrTime。

“当前窗体”和“当前操作员”窗格中的内容是通过代码进行设置的。首先来介绍一下“当前窗体”窗格的实现。在调用“商品进货”窗体时，设置主窗体的状态栏，将当前窗体的名称和状态栏窗格中的内容重新写入窗体的状态栏中。关键代码如下：

```
Private Sub Form_Load() '加载“商品进货”窗体
    MDIForm1.SBar1.Panels(1).Text = MDIForm1.SBar1.Panels(1).Text + Me.Caption
End Sub
```

“当前操作员”窗格的实现：在登录窗体中，当用户登录到系统中时，将登录系统的操作员的姓名写入到主窗体状态栏的“当前操作员”窗格中，关键代码如下：

```
Private Sub Lbl_OK_Click() '确定
    MDIForm1.SBar1.Panels(2).Text = MDIForm1.SBar1.Panels(2).Text & Lbl_Name.Caption
End Sub
```

## 27.10 商品进货模块设计

教学录像：光盘\TM\lx\27\商品进货模块设计.exe

本模块使用的数据表：tbS\_warehouse\_main、tbS\_warehouse\_detailed、tbS\_stock、

tbS\_currentaccount、tbS\_units、tbS\_employ

商品进货模块主要完成将所采购商品的信息批量保存到入库表和库存表中。为了提高录入入库商品信息的速度,这里使用了 MSFlexGrid 表格控件。

MSFlexGrid 控件可以对表格数据进行显示和操作,具有完全的灵活性,但它不能像 DataGrid 表格那样在运行时直接对表格中的数据进行编辑操作。如果要对 MSFlexGrid 表格中的数据进行编辑,必须借用 TextBox 控件。

运行程序,在程序的主窗体中选择“进货信息”/“商品进货”命令,即可进入到“商品进货”模块中。在该模块中双击“经手人”文本框,即可弹出职员表中的相关信息;双击经手人姓名,即可将其添加到“经手人”文本框中。双击“进货单位”文本框,在弹出的供货商表格中,选择供货单位,按回车键,将选择的供应商名称添加到“供货单位”文本框中。

当光标进入到“商品名称”文本框中,用户双击该文本框将弹出所有的商品信息,用户可以选择需要进货的商品信息。选中之后,将直接添加到商品进货单中,用户只需输入进货商品的数量信息即可。金额、入库品种、合计数量等都由系统自动运算得出。当用户输入完成要进货的商品信息,单击“确定”按钮,即可将其保存到数据库中,其实现效果如图 27.32 所示。

图 27.32 商品进货模块

### 27.10.1 设计窗体界面

商品进货模块的界面设计过程如下:

(1) 在工程中新建一个窗体,将窗体的名称命名为 frm\_stockBill, BorderStyle 属性设置为 1-Fixed Single, MDIChild 属性为 True, Caption 属性设置为“进货单”, MaxButton 属性为 False。

(2) 在窗体上添加 Adodc 控件和 DataGrid 控件。由于这两个控件属于 ActiveX 控件,在使用前需要将其添加到 VB 工程中,具体方法为:

①选择“工程”/“部件”命令,在弹出的“部件”对话框中选中 Microsoft Ado Data Controls 6.0(SP4)

项, 添加 ADO 控件, 选中 Microsoft DataGrid Controls 6.0(SP5)项, 添加 DataGrid 控件。单击“确定”按钮, 即可将这两个控件添加到 VB 工程中。

②在窗体中添加 3 个 DataGrid 控件, 使用默认名称, 分别设置控件的选取框样式。在 DataGrid 控件上单击鼠标右键, 选择“属性”命令, 弹出“属性页”对话框, 选择“拆分”选项卡, 将 DataGrid 控件的选取边框样式设置为 4-dbgHighlightRowRaise, 如图 27.33 所示。

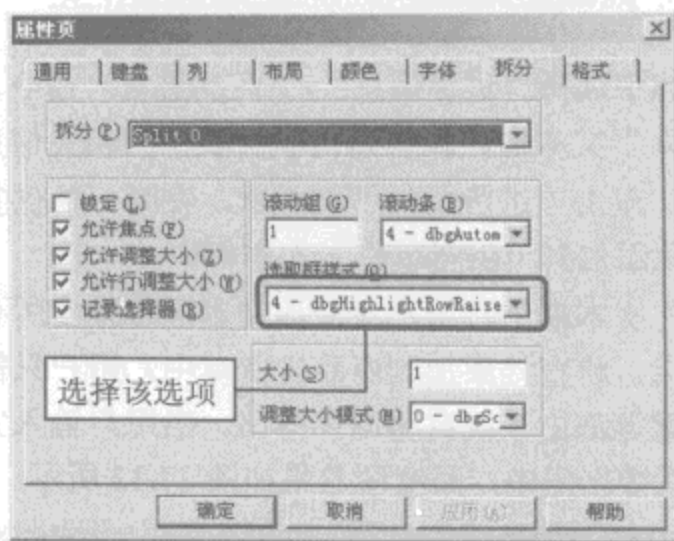


图 27.33 设置边框样式

注意: 在没有设置选取框样式的情况下, 如果 DataGrid 控件获得焦点, 焦点将在控件的第一个单元格中, 而不是整行获得焦点 (整行选定数据)。

(3) 在窗体中添加 6 个 ADO 控件, 名称分别为 Adodc1、Adodc2、AdoCount、AdoStock、AdoEmploy、AdoUnits, Visible 属性均设置为 False。

(4) 在窗体中添加 TextBox 控件、Label 控件和 CommandButton 控件, 如图 27.34 所示。



图 27.34 商品进货窗体的设计

(5) 添加 DTPicker 控件。该控件属于 ActiveX 控件, 在使用之前需要将其添加到 VB 的工程中, 具体的添加方法如下:

选择“工程”/“部件”命令, 在弹出的“部件”对话框中, 选中 Microsoft Windows Common Controls

2.6.0 项, 单击“确定”按钮, 即可将 DTPicker 控件添加到 VB 工程中。

(6) 添加 MSFlexGrid 控件。由于该控件属于 ActiveX 控件, 在使用之前需要将其添加到 VB 工程中, 具体的添加方法如下:

选择“工程”/“部件”命令, 在弹出的对话框中, 选中 Microsoft FlexGrid Controls 6.0(SP3)项。单击“确定”按钮, 即可将该控件添加到 VB 工程中。将该控件添加到窗体上, 设置名称为 MS1, 其他属性均为默认设置。

商品进货模块的设计结果如图 27.34 所示。

## 27.10.2 窗体初始化

在窗体初始化的时候, 首先将本窗体中需要使用到的数据库都连接上, 利用公共变量 PublicStr 初始化 ADO 数据源, 并将职员表、往来单位表、库存表都绑定到对应的 DataGrid 控件上, 以方便在程序运行时进行调用。将用于显示进货信息的 MSFlexGrid 控件的行数、列数以及列标题等都进行初始化, 关键的代码如下:

Private Sub Form_Load()	'窗体加载
Adodc1.ConnectionString = PublicStr	'连接数据库
Adodc1.RecordSource = "select * from tbS_warehouse_main"	'查询进货主表中的数据
Adodc1.Refresh	'刷新
Adodc2.ConnectionString = PublicStr	'连接数据库
Adodc2.RecordSource = "select * from tbS_warehouse_detailed"	'查询进货明细表
Adodc2.Refresh	'刷新
adoCount.ConnectionString = PublicStr	'连接数据库
adoCount.RecordSource = "select * from tbS_currentaccount"	'查询往来对账明细表
adoCount.Refresh	'刷新
AdoUnits.ConnectionString = PublicStr	'连接数据库
AdoUnits.RecordSource = "select unitcode,fullname from tbS_units"	'查询往来单位信息表
AdoUnits.Refresh	'刷新
Set DataGrid2.DataSource = AdoUnits	'绑定往来单位信息表
AdoEmploy.ConnectionString = PublicStr	'连接数据库
AdoEmploy.RecordSource = "select employecode,fullname from tbS_employ"	'查询职员表
AdoEmploy.Refresh	'刷新
Set DataGrid1.DataSource = AdoEmploy	'绑定职员表
AdoStock.ConnectionString = PublicStr	'连接数据库
AdoStock.RecordSource = "select * from tbS_stock"	'查询库存表
AdoStock.Refresh	'刷新
Set DataGrid3.DataSource = AdoStock	'绑定库存表
Adodc1.RecordSource = "select * from tbS_warehouse_main order by billcode"	'查询进货主表中的信息
Adodc1.Refresh	'刷新
If Adodc1.Recordset.RecordCount > 0 Then	'如果记录数大于零
Adodc1.Recordset.MoveLast	'移动到最后一行
'生成进货编号	
Text1(0).Text = Format(Now, "yyyymmdd") & "JH" & Left(Mid(Adodc1.Recordset.Fields(1), 11, 17), 7) + 1	
Else	'否则
Text1(0).Text = Format(Now, "yyyymmdd") & "JH1000001"	'为今天第一条编号



```

End If
DTPicker1.Value = Format(Now, "yyyy-mm-dd")           '格式化日期
MS1.Rows = 100: MS1.Cols = 7                         '定义 MSI 总行/列数
'定义 MS1 表的宽度
MS1.ColWidth(0) = 12 * 25 * 1: MS1.ColWidth(1) = 12 * 25 * 4: MS1.ColWidth(2) = 12 * 25 * 9
MS1.ColWidth(3) = 12 * 25 * 3: MS1.ColWidth(4) = 12 * 25 * 3: MS1.ColWidth(5) = 12 * 25 * 3
MS1.ColWidth(6) = 12 * 25 * 4
MS1.FixedRows = 1: MS1.FixedCols = 1                 '设置固定行、列
'定义 MS1 表的表头
MS1.TextMatrix(0, 0) = "NO. ": MS1.TextMatrix(0, 1) = "商品编号": MS1.TextMatrix(0, 2) = "商品全名"
MS1.TextMatrix(0, 3) = "单位": MS1.TextMatrix(0, 4) = "数量": MS1.TextMatrix(0, 5) = "单价"
MS1.TextMatrix(0, 6) = "金额"
'定义 MS1 表的列序号
For i = 1 To 99
    MS1.TextMatrix(i, 0) = i
Next i
'装载窗体时, 确定 text3 的位置
Text3.Text = ""                                       '设置 Text3 为空
Text3.Width = MS1.CellWidth: Text3.Height = MS1.CellHeight '设置 Text3 的长宽
Text3.Left = MS1.CellLeft + MS1.Left: Text3.Top = MS1.CellTop + MS1.Top '设置 Text3 坐标
End Sub

```

### 27.10.3 商品信息录入

在商品进货过程中, 如果进货的商品品种较多、数量也很大, 那么使用文本框一条一条的录入数据, 效率将会很低。利用 VB 提供的 MSFlexGrid 控件, 处理数据比较灵活, 虽然显示的数据是只读的, 但可以通过 TextBox 控件向 MSFlexGrid 控件中输入数据, 然后使用 For 循环逐一将表格中数据添加到数据表。

本窗体中利用 Text3 来辅助商品信息录入, 这里主要应用了 Text3 控件的 Change 事件和 KeyDown 事件。

#### 1. Text3 的 Chang 事件

下面先介绍一下 Change 事件, 在 Text3 的 Change 事件中, 首先根据商品名称和拼音简码检索需要的商品信息。然后, 动态统计进货单的商品数量和商品金额。关键的代码如下:

```

Private Sub Text3_Change()
    If MS1.Col = 2 Then                                '如果是在“商品全名”列中
        If Text3.Text <> "" Then                      '如果 text3 不为空
            '筛选商品名称和拼音简码
            AdoStock.RecordSource = "select * from tbS_stock where fullname like '" + Text3.Text + "' +
            '%or spell like '" + Text3.Text + "' + '%'"
            AdoStock.Refresh                            '刷新
            If AdoStock.Recordset.RecordCount > 0 Then '如果记录数大于 0
                DataGrid3.Visible = True              '商品名称表可见
                If DataGrid3.Visible Then DataGrid3.SetFocus '使控件获得焦点
            End If
        End If
    End If
End Sub

```

```

End If
Dim qtp As Integer
Dim tsum As Single
For i = 1 To 99
    If MS1.TextMatrix(i, 4) <> "" Then
        qty = Val(MS1.TextMatrix(i, 4)) + qty
        Label5.Caption = qty
    End If
    If MS1.TextMatrix(i, 5) <> "" Then
        '将数值格式化
        MS1.TextMatrix(i, 6) = Format(Val(MS1.TextMatrix(i, 4)) * Val(MS1.TextMatrix(i, 5)), "#0.00")
    Else
        MS1.TextMatrix(i, 6) = ""
    End If
    If MS1.TextMatrix(i, 6) <> "" Then
        tsum = Val(MS1.TextMatrix(i, 6)) + tsum
        Label6.Caption = Format(tsum, "#0.00")
    End If
Next i
End Sub

```

'定义整型变量  
'定义单精度变量  
'从 1 到 99 循环  
'统计商品数量  
'给变量赋值  
'显示数量合计  
'计算商品金额  
'否则  
'表格内数值为空  
'统计商品金额  
'给变量赋值  
'将合计金额赋给变量

## 2. Text3 的 KeyPress 事件

Text3 控件的 KeyPress 事件主要实现了以下两项功能。

- (1) 将 Text 控件中的数据传输给 MSFlexGrid 控件。
- (2) 将 Text 控件移动到表格的下一个单元格或者表格的下一行。

关键代码如下：

```

Private Sub Text3_KeyPress(KeyAscii As Integer)
    'vbKeyReturn 常数为键盘上的"回车键"
    If KeyAscii = vbKeyReturn Then
        MS1.Text = Text3.Text
        Text3.Text = MS1.Text
        If MS1.Col = 5 Then
            MS1.Row = MS1.Row + 1
            MS1.Col = 1
        Else
            If MS1.Col + 1 <= MS1.Cols - 1 Then
                MS1.Col = MS1.Col + 1
            Else
                If MS1.Row + 1 <= MS1.Rows - 1 Then
                    MS1.Row = MS1.Row + 1
                    MS1.Col = 1
                End If
            End If
        End If
    End If
End Sub

```

'键盘按下事件  
'按回车键, Text3 向右移动  
'将 Text3 的内容赋给 MS1  
'MS1 内容赋给 Text3  
'如果是第 5 列  
'行数加一  
'列为一  
'否则  
'列数加一的值小于总列数  
'列加一  
'否则  
'行数加一小于总行数  
'行加一  
'列值为一

## 27.11 库存状况模块设计

教学录像：光盘\TM\lx\27\库存状况模块设计.exe

本模块使用的数据表：tbS\_stock

库存状况模块的主要功能是查看库存商品数量、商品成本均价、库存商品总价以及库存上下限设置，以及调用子窗体库存明细账本。

在主窗体中选择“库存信息”/“库存状况”命令，即可进入到如图 27.35 所示的库存状况模块中。

NO.	商品编号	商品全名	库存总价
1	T1309	MD电饭煲	1000
2	T1010	TM计算机	11000
3	T1004	SX彩电52寸	80000
4	T1002	XH彩电25寸	58400
5	T1003	CH彩电32寸	148000
6	T1007	LB电磁炉	8000
7	T1008	LL热水器	8000
8	T1011	YS手机	100000
9	T1006	HE冰箱	200000
10	T1005	HE全自动洗衣机	100000
11	T1001	XH彩电21英寸	275400
合计		库存总数量：1147	库存总价值：985800.00 元

图 27.35 库存状况模块运行结果

### 27.11.1 设计窗体界面

库存情况模块的界面设计过程如下：

(1) 在工程中新建一个窗体，将窗体的名称命名为 frm\_Stock，BorderStyle 属性设置为 1-Fixed Single，Caption 属性设置为“库存状况”，MaxButton 属性为 False，MDIChild 属性为 True。

(2) 在窗体上添加两个 Adodc 控件和一个 MSFlexGrid 控件。由于二者属于 ActiveX 控件，在使用前需要将其添加到 VB 工具箱中，具体的添加方法如下：

选择“工程”/“部件”命令，在弹出的“部件”对话框中选中 Microsoft ADO Data Controls 6.0(SP4)项，添加 ADO 控件，选中 Microsoft FlexGrid Controls 6.0(SP6)项，添加 MSFlexGrid 控件。单击“确定”按钮，即可将这两个控件添加到 VB 工程中。

(3) 在窗体上添加一个 ToolBar 控件和一个 ImageList 控件。这两个控件都是 ActiveX 控件，在使用前需要先将其添加到 VB 的工具箱中，具体的添加方法如下：

#### ① 添加 ToolBar 控件和 ImageList 控件

选择“工程”/“部件”命令，在弹出的“部件”对话框中选中 Microsoft Windows Common Controls 6.0 项，单击“确定”按钮，此时将 Toolbar 控件和 ImageList 控件添加到工具箱当中，如图 27.36 所示。

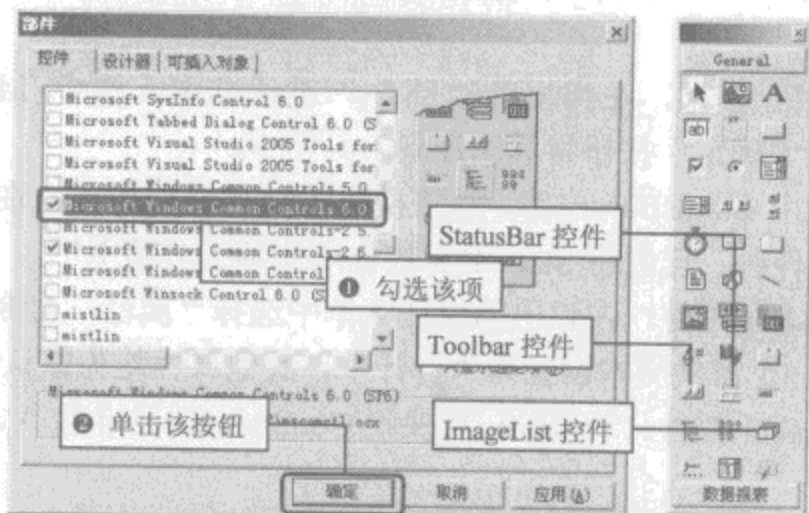


图 27.36 添加控件到工具箱中

## ② 设置控件 ImageList 和 Toolbar 的属性

分别在控件 ImageList 和 Toolbar 上单击鼠标右键选择“属性”，设置控件 ImageList 属性只需一步就可以完成，如图 27.37 所示；设置 Toolbar 控件的属性共需要两步，如图 27.38 所示。

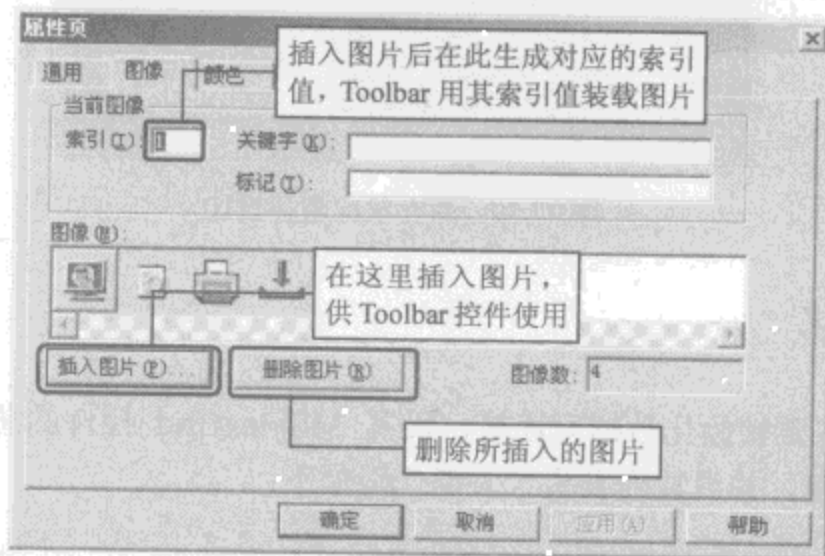


图 27.37 ImageList 属性设置

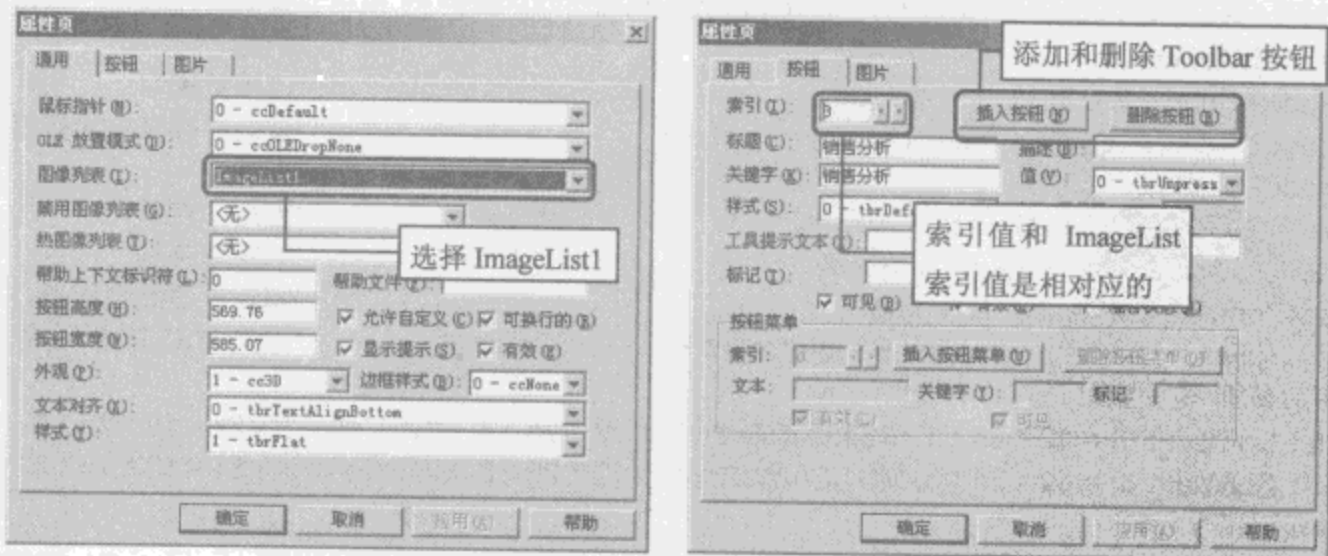


图 27.38 ToolBar 控件属性设置

(4) 在窗体上添加一个 DataCombo 控件。该控件属于 ActiveX 控件，在使用之前需要将其添加到



工具箱，添加方法为：

选择“工程”/“部件”命令，在弹出的“部件”对话框中，选中 Microsoft DataList Controls 6.0(SP3)(OLEDB)项。单击“确定”按钮将其添加到工具箱中。

(5) 在窗体上添加一个 StatusBar 控件。该控件属于 ActiveX 控件，在使用前需要将其添加到工具箱中，在添加 Toobar 控件时，已经将 StatusBar 控件添加到工具箱中了。

用鼠标右键单击 StatusBar 控件，在弹出的快捷菜单中选择“属性”命令，在弹出的“属性页”对话框中添加 3 个窗格，第一个窗格文本设置为“合计”，其他属性均为默认设置。

(6) 在窗体上添加 Label 控件、TextBox 控件和 CommandButton 控件，相关属性设置如图 27.39 所示。

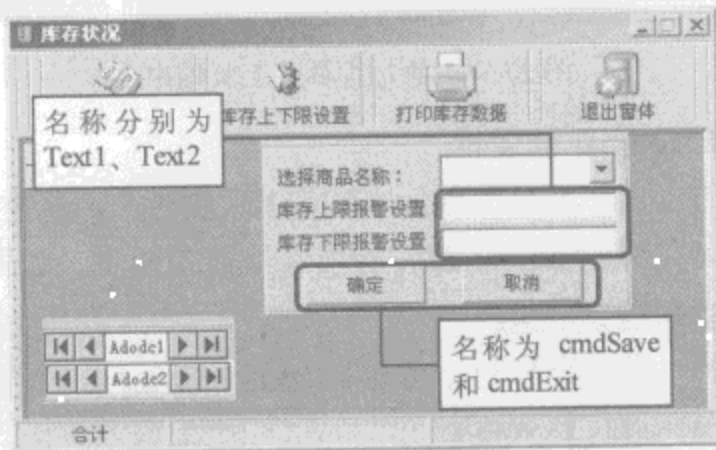


图 27.39 库存状况窗体设计

## 27.11.2 窗体初始化

窗体的 Load 事件主要初始化数据库连接，设置 MSFlexGrid 控件的总行数、总列数，以及定义 MSFlexGrid 控件的列宽度，并设置列表头。关键代码如下：

```
Private Sub Form_Load()
    Adodc1.ConnectionString = PublicStr
    Adodc1.RecordSource = "SELECT * FROM tbS_stock"
    Adodc1.Refresh
    Adodc2.ConnectionString = PublicStr
    Adodc2.RecordSource = "select * from tbS_stock"
    Adodc2.Refresh
    Set DataCombo1.DataSource = Adodc2
    Set DataCombo1.RowSource = Adodc2
    DataCombo1.ListField = "fullname"
    MS1.Rows = Adodc1.Recordset.RecordCount + 1: MS1.Cols = 6
    '定义 MS1 表的宽度
    MS1.ColWidth(0) = 12 * 25 * 1: MS1.ColWidth(1) = 12 * 25 * 4: MS1.ColWidth(2) = 12 * 25 * 8
    MS1.ColWidth(3) = 12 * 25 * 3: MS1.ColWidth(4) = 12 * 25 * 3: MS1.ColWidth(5) = 12 * 25 * 3
    '设置固定行、列
    MS1.FixedRows = 1: MS1.FixedCols = 1
    '定义 MS1 表的表头
    MS1.TextMatrix(0, 0) = "NO. ": MS1.TextMatrix(0, 1) = "商品编号": MS1.TextMatrix(0, 2) = "商品全名"
    '连接数据库
    '查询库存表中的数据
    '刷新
    '连接数据库
    '查询库存表中的数据
    '刷新
    '绑定库存表
    '绑定库存表
    '绑定到商品全程表
    '定义 MS1 的总行数、总列数
```

```

MS1.TextMatrix(0, 3) = "库存数量": MS1.TextMatrix(0, 4) = "成本均价": MS1.TextMatrix(0, 5) = "库存总价"
'定义 MS1 表的列序号
For i = 1 To Adodc1.Recordset.RecordCount
    MS1.TextMatrix(i, 0) = i
Next i
Call AddData                                     '调用自定义函数
End Sub

```

### 27.11.3 库存上下限设置

在本模块中也可以对库存的上下限进行设置。运行程序，单击“库存上下限设置”按钮，即可弹出“库存上下限设置”框。在“商品名称”组合框中，选择要设置的商品名称，然后在“库存上限报警设置”文本框中输入库存上限，在“库存下限报警设置”文本框中输入库存下限，单击“确定”按钮，完成库存上下限的设置。这里利用 ADO 控件执行 SQL 语句将文本框中的内容写入到数据库中，关键代码如下：

```

Private Sub cmdsave_Click()                                     '“确定”按钮
    'ADO 控件的 RecordSource 执行 SQL 语句
    Adodc2.RecordSource = "select * from tbS_stock where tradecode=" & Adodc1.Recordset.Fields(0) & ""
    Adodc2.Refresh                                             '刷新
    If Err.Number > 0 Then                                     '如果错误号不为空
        '如果出现错误，用户不继续执行
        If MsgBox("出现“数据库中无有效数据”错误，是否继续？", vbQuestion + vbYesNo, "系统提示") = vbNo
        Then
            Frame2.Visible = False                             'Frame2 控件隐藏
            Exit Sub                                           '退出本过程
        End If
    End If
    If Adodc2.Recordset.RecordCount > 0 Then                  '如果记录数大于零
        Adodc2.Recordset.Fields("upperlimit").Value = Val(Text1.Text) '设置库存上限
        Adodc2.Recordset.Fields("lowerlimit").Value = Val(Text2.Text) '设置库存下限
        Adodc2.Recordset.Update                               '更新记录
        MsgBox "设置成功！"                                    '弹出对话框
        Frame2.Visible = False                                'Frame2 控件隐藏
        Text1.Text = "": Text2.Text = ""                      '设置控件内容为空
    Else                                                        '如果没有记录数
        MsgBox "无库存商品"                                    '弹出提示对话框
    End If
End Sub

```

在“商品名称”组合框中，选择商品名称，来提取上次设置的库存上下限信息。这里利用 DataCombo1 控件的 Change 事件以及 SQL 语句，将数据库中的库存上下限信息显示在文本框中，关键代码如下：

```

Private Sub DataCombo1_Change()                                '选择商品名称
    '如果选择为空，提示信息
    If DataCombo1.Text = "" Then MsgBox "请选择商品名称！": Exit Sub
    '在库存表中查询

```

```

Adodc1.RecordSource = "select * from tbS_stock where fullname=" + DataCombo1.Text + ""
Adodc1.Refresh
If Adodc1.Recordset.RecordCount > 0 Then
    On Error Resume Next
    Text1.Text = Adodc1.Recordset.Fields("upperlimit")
    Text2.Text = Adodc1.Recordset.Fields("lowerlimit")
Else
    Text1.Text = "": Text2.Text = ""
End If
End Sub

```

'刷新  
'如果记录数大于零  
'错误处理  
'显示库存上限  
'显示库存下限  
'如果没有记录  
'清空文本框内容

#### 27.11.4 自定义过程向 MSFlexGrid 控件中添加数据

在本窗体中定义了一个自定义过程 AddData(), 用于将库存商品数据动态显示在 MSFlexGrid 控件中, 并将库存总数量和库存总价值都累加并显示在状态栏中。关键代码如下:

```

Sub AddData()
    Dim qty As Integer
    Dim price As Single
    Adodc1.RecordSource = "SELECT * FROM tbS_stock ORDER BY qty"
    Adodc1.Refresh
    For i = 1 To Adodc1.Recordset.RecordCount
        MS1.TextMatrix(i, 1) = Adodc1.Recordset.Fields("tradeCode").Value
        MS1.TextMatrix(i, 2) = Adodc1.Recordset.Fields("fullName").Value
        MS1.TextMatrix(i, 3) = Val(Adodc1.Recordset.Fields("qty").Value)
        If Adodc1.Recordset.Fields("averageprice").Value = 0 Then
            MS1.TextMatrix(i, 4) = Adodc1.Recordset.Fields("price").Value
        Else
            MS1.TextMatrix(i, 4) = Val(Adodc1.Recordset.Fields("averageprice").Value)
        End If
        MS1.TextMatrix(i, 5) = Val(MS1.TextMatrix(i, 4)) * Val(MS1.TextMatrix(i, 3))
        Adodc1.Recordset.MoveNext
        qty = qty + Val(MS1.TextMatrix(i, 3))
        price = price + MS1.TextMatrix(i, 5)
    Next i
    SBar1.Panels(2).Text = "库存总数量: " & qty
    SBar1.Panels(3).Text = "库存总价值: " & Format(price, "#0.00") & " 元"
End Sub

```

'自定义过程  
'定义整型变量  
'定义浮点行变量  
'查询库存表中数据  
'刷新  
'从 1 到记录总数循环  
'显示商品编号  
'显示商品全称  
'显示商品数量  
'如果平均价格为零  
'显示单价  
'否则  
'显示平均价格  
'显示总金额  
'记录集下移  
'总数量累加赋给变量  
'总金额累加赋给变量  
'显示库存总量  
'显示库存总价值

## 27.12 销售情况分析模块设计

 教学录像: 光盘\TM\lx\27\销售情况分析模块设计.exe

 本模块使用的数据表: tbS\_sell\_detailed、tbS\_temp

在主窗体中选择“销售信息”/“月销售状况”命令, 即可进入到如图 27.40 所示的“月销售状况”

模块中。在月销售状况模块中先统计当年所有商品的销售数据信息（含退货）和净销售商品的数据信息，然后根据分析出的基础数据，以商品为单位利用图表进行月销售分析；或者以商品为单位分析商品的销售明细和销售退货明细。

商品编号	商品名称	销售数量	销售均价	销售金额	销售数量【含退货】	销售金额【含退货】
T1001	XH彩电21英寸	164	950	155800	154	146900
T1002	XH彩电25英寸	81	1300	105300		
T1003	CH彩电32英寸	125	2500	312500	120	302000
T1004	SH彩电52英寸	60	3000	180000		
T1007	LB电饭锅	120	599.33333	71890		
T1008	LL热水器	120	0	0		
T1009	WD电饭煲	1210	700	1048000		
T1010	TH计算机	210	5000	1050000		
合计:		2075		2901990.00		

图 27.40 销售情况分析

### 27.12.1 设计月销售情况窗体界面

月销售情况窗体的界面设计过程如下：

(1) 在工程中新建一个窗体，将窗体的名称命名为 `frm_saleStatus`，`BorderStyle` 属性设置为 1-Fixed Single，`Caption` 属性设置为“月销售状况”，`MaxButton` 属性为 False，`MDIChild` 属性为 True。

(2) 在窗体中添加一个 `DataGrid` 控件，使用默认名称；在窗体中添加 3 个 `Adodc` 控件，均使用默认名称。

(3) 添加一个 `ToolBar` 控件和一个 `ImageList` 控件到窗体上。在窗体上制作工具栏是通过 `ToolBar` 控件和 `ImageList` 控件共同来实现的。

(4) 在窗体上添加一个 `StatusBar` 控件，设计窗体的状态栏，在状态栏的窗格中显示合计商品销售数量和销售金额。

将 `StatusBar` 控件添加到窗体上，用鼠标右键单击该控件，在弹出的快捷菜单中选择“属性”命令，设置属性如图 27.41 所示。

月销售状况模块的设计结果如图 27.42 所示。

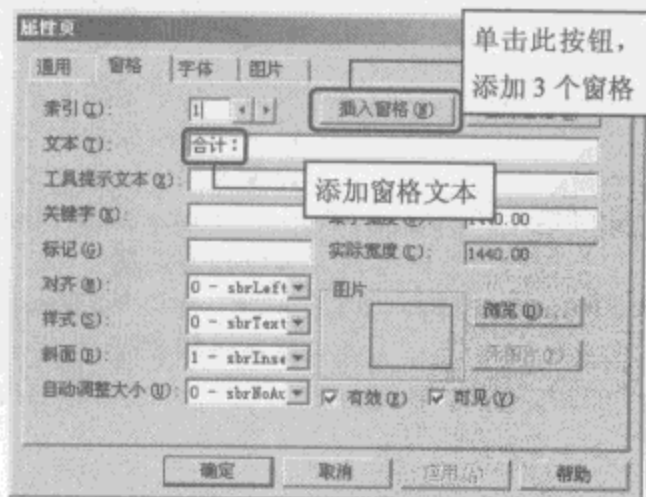


图 27.41 StatusBar 控件属性设置

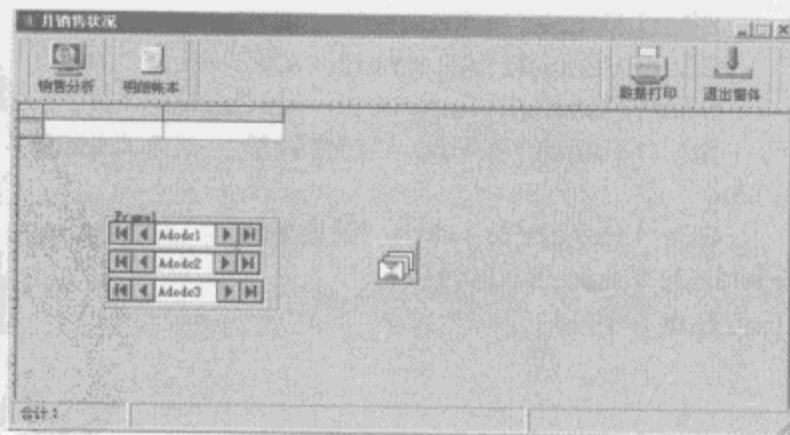


图 27.42 月销售状况窗体设计结果



## 27.12.2 统计全年商品销售状况

在窗体加载时统计全年的商品销售状况。主要使用 SQL 语句左外联接技术（左向外联接的结果集包括 Left 子句中指定的左表的所有行，而不仅仅是联接列所匹配的行。如果左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为空值）。将销售明细表（tbS\_sell\_detailed）和销售退货明细表（tbS\_resell\_detailed）进行统计计算，并显示销售数量和销售金额及净销售数量和销售金额（不含退货）。关键代码如下：

```
Private Sub Form_Load()                                '窗体加载
    Adodc1.ConnectionString = PublicStr                 '连接数据库
    Adodc2.ConnectionString = PublicStr                 '连接数据库
    Adodc3.ConnectionString = PublicStr                 '连接数据库
    'ADO 控件的 RecordSource 执行 SQL 语句，统计销售状况
    Adodc1.RecordSource = "select a.tradecode as 商品编号,a.fullname as 商品名称,a.qty as 销售数量,a.price AS 销售均价,a.tsum as 销售金额,a.qty-b.qty2 as '销售数量【含退货】',a.tsum-b.tsum2 as '销售金额【含退货】' from (SELECT tradecode,fullname,avg(price)as price,sum(qty) AS qty, sum(tsum) as tsum from tbS_sell_detailed group by tradecode,fullname) a left join (SELECT tradecode,fullname,sum(qty) AS qty2,sum(tsum) as tsum2 from tbS_resell_detailed group by tradecode,fullname) b on a.tradecode=b.tradecode "
    Adodc1.Refresh                                     '刷新
    Set DataGrid1.DataSource = Adodc1                  '绑定 DataGrid 控件
    '设置控件宽度
    DataGrid1.Columns(0).Width = 12 * 25 * 3
    DataGrid1.Columns(1).Width = 12 * 25 * 7
    DataGrid1.Columns(2).Width = 12 * 25 * 3
    DataGrid1.Columns(3).Width = 12 * 25 * 3
    DataGrid1.Columns(4).Width = 12 * 25 * 3
    DataGrid1.Columns(5).Width = 12 * 25 * 6
    DataGrid1.Columns(6).Width = 12 * 25 * 6
    'ADO 控件的 RecordSource 执行 SQL 语句，统计销售数量和销售金额
    Adodc2.RecordSource = "SELECT SUM(qty) AS 销售数量, SUM(tsum) AS 销售金额 FROM tbS_sell_detailed"
    Adodc2.Refresh                                     '刷新
    Adodc3.RecordSource = "SELECT SUM(qty) AS 退货数量, SUM(tsum) AS 退货金额 FROM tbS_resell_detailed"
    Adodc3.Refresh                                     '刷新
    SBar1.Panels(1).Alignment = sbrCenter              '设置为居中显示文本
    SBar1.Panels(2).Alignment = sbrCenter              '设置为居中显示文本
    SBar1.Panels(3).Alignment = sbrCenter              '设置为居中显示文本
    SBar1.Panels(2).Text = "销售数量: " & Adodc2.Recordset.Fields(0).Value - Adodc3.Recordset.Fields(0).Value
    '显示销售数量
    SBar1.Panels(3).Text = "销售金额: " & Format(Adodc2.Recordset.Fields(1).Value - Adodc3.Recordset.Fields(1).Value, "#0.00")
    '显示销售金额
End Sub
```

## 27.12.3 设计月销售分析窗体界面

当用户进入到月销售情况窗体界面时,单击工具栏上的“销售分析”按钮,即可进入到“每月销售比较”窗体中,在这里可以对已经选中的窗体进行销售分析,并可以以二维或三维的形式显示。

单击“销售分析”按钮,就可以对当前商品进行按月的分析统计,如图 27.43 所示。

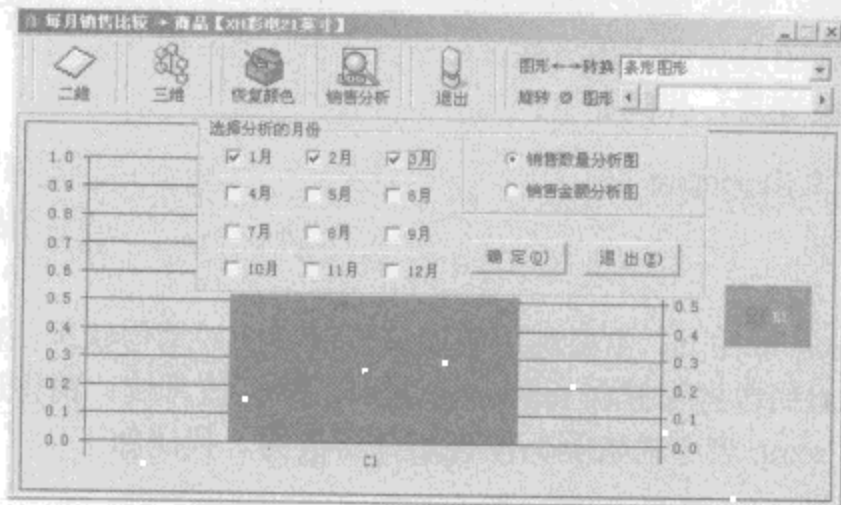


图 27.43 每月销售分析运行效果

月销售分析窗体的界面设计过程如下:

- (1) 在工程中新建一个窗体,将窗体的名称命名为 `frm_saleImage`, `BorderStyle` 属性设置为 `1-Fixed Single`, `Caption` 属性设置为“每月销售比较”, `MaxButton` 属性为 `False`, `MDIChild` 属性为 `True`。
- (2) 在窗体上添加一个 `ToolBar` 控件和一个 `ImageList` 控件,并将这两个控件连接起来。
- (3) 添加 3 个 `Adodc` 控件和一个 `ComboBox` 控件,设置其属性为默认值。
- (4) 在窗体上添加一个 `MSChart` 控件,用于图表显示。由于该控件属于 `ActiveX` 控件,因此在使用时,应首先将其添加 `VB` 的工具箱中。具体的添加方法如下:
  - ① 选择“工程”/“部件”命令,在弹出的“部件”对话框中选中 `Microsoft Chart Controls 6.0(SP4)` 项,单击“确定”按钮,即可将该控件添加到工具箱中。
  - ② 设置该控件的名称为 `MSC1`, `ColumnCount` 属性设置为 1, `RowCount` 属性设置为 1。
- (5) 在窗体上再添加 `CheckBox` 控件、`HScrollBar` 控件和 `OptionButton` 控件,属性设置如图 27.44 所示。

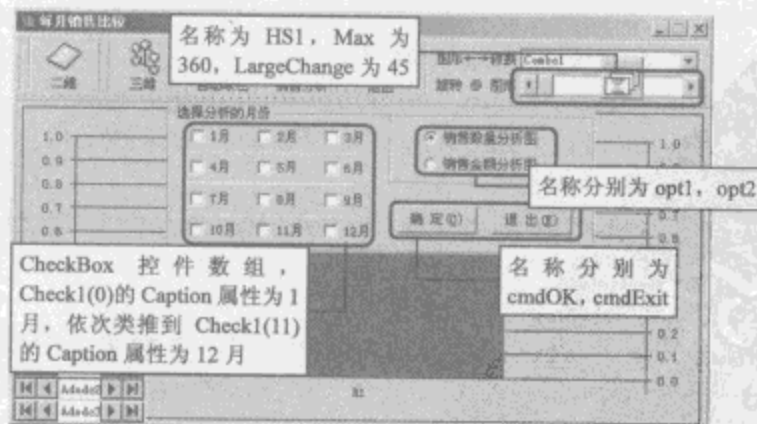


图 27.44 销售分析窗体设计结果

## 27.12.4 利用图表分析月销售状况

利用图表来分析月营业情况。实现方法：定义一个动态二维数组，将数据表中月份字段的值和月营业额的值赋给动态二维数组（例如：arrValues(i, 1) = Adodc2.Recordset!billdate & "月份"和 arrValues(i, 2) = Adodc2.Recordset!qty 月份营业额），将赋值的数据返回给控件 MSChart 的 ChartData 属性，便可以显示月营业分析图（例如：MSChart1.ChartData = arrValues）。另外，图表的 X 轴和 Y 轴分别代表月份和月营业额。下面来介绍具体的实现方法。

在代码窗口中定义相关变量，代码如下：

```
Private rs As New ADODB.Recordset      '声明记录集变量
Private str1 As String                 '声明字符型变量
Dim i As Integer                      '声明整型变量
```

在“确定”按钮的 Click 事件下，主要利用图表分析商品的月销售状况，设计思路为：将销售商品信息（含退货）按月统计出的数据存储在 tbS\_temp 表中，然后按月统计销售退货数据，二者做差，更新 tbS\_temp 表。最后 tbS\_temp 表中的数据为图表所分析数据。代码如下：

```
Private Sub cmdOK_Click()
    Call cnn
    Set rs = New ADODB.Recordset
    str1 = "delete from tbS_temp"
    rs.Open str1, cnn, adOpenDynamic, adLockOptimistic
    cnn.Close
    '向临时表添加数据
    For i = 0 To Check1.Count - 1
        If Check1(i).Value = 1 Then
            dd = Mid(Date, 1, 4) & "-0" & Check1(i).Index + 1
            dd = Format(dd, "yyyy-mm")
            'ADO 控件的 RecordSource 执行 SQL 语句，销售总数量和总价格
            Adodc1.RecordSource = "SELECT SUM(qty) AS 总数量, SUM(tsum) AS 总价格 FROM
tbS_sell_detailed WHERE year(billdate)=" & str(Year(Date)) & " and month(billdate)=" & str(Check1(i).Index + 1)
+ "AND (tradeCode = " & frm_saleStatus.Adodc1.Recordset.Fields(0) & ")"
            Adodc1.Refresh
            Adodc2.RecordSource = "select * from tbS_temp"
            Adodc2.Refresh
            Adodc2.Recordset.AddNew
            Adodc2.Recordset.Fields(0).Value = dd
            Adodc2.Recordset.Fields(1).Value = Adodc1.Recordset.Fields(0)
            Adodc2.Recordset.Fields(2).Value = Adodc1.Recordset.Fields(1)
            '将商品编号写入数据库
            Adodc2.Recordset.Fields(3).Value = frm_saleStatus.Adodc1.Recordset.Fields(0)
            Adodc2.Recordset.Update
            'ADO 控件的 RecordSource 执行 SQL 语句
            Adodc3.RecordSource = "SELECT SUM(qty) AS 总数量, SUM(tsum) AS 总价格, tradeCode
FROM tbS_resell_detailed WHERE year(billdate)=" & str(Year(Date)) & " and month(billdate)=" &
str(Check1(i).Index + 1) & "AND (tradeCode = " & frm_saleStatus.Adodc1.Recordset.Fields(0) & ")group by
tradeCode"
```

'单击“确定”按钮

'调用自定义过程

'实例化记录集对象

'将删除语句赋给变量

'执行删除语句

'关闭连接

'1 至 12 月做循环

'如果复选框被选中

'取日期

'格式化日期

'刷新

'查询临时表数据

'刷新

'添加新记录

'将日期写入数据库

'将总数量写入数据库

'将总价格写入数据库

'更新数据

```

Adodc3.Refresh                                '刷新
'将销售退货的数量和金额刨除
If Adodc3.Recordset.RecordCount > 0 Then        '如果记录数大于零
    '查询临时表中的数据
    Adodc2.RecordSource = "select * from tbS_temp where billdate=" + dd + ""
    Adodc2.Refresh                            '刷新
    '刨除退货数量
    Adodc2.Recordset.Fields(1) = Adodc2.Recordset.Fields(1) - Adodc3.Recordset.Fields(0)
    '刨除退货金额
    Adodc2.Recordset.Fields(2) = Adodc2.Recordset.Fields(2) - Adodc3.Recordset.Fields(1)
    Adodc2.Recordset.Update                    '更新数据库
End If
'ADO 控件的 RecordSource 执行 SQL 语句
Adodc2.RecordSource = "select * from tbS_temp"    '查询临时表中的数据
Adodc2.Refresh                                '刷新
If Opt1.Value = True Then                      '如果选中按销售数量分析
    If Adodc2.Recordset.RecordCount > 0 Then    '如果记录数大于零
        Adodc2.Recordset.MoveFirst              '移动到第一条记录
        nums = Adodc2.Recordset.RecordCount      '将记录数赋给变量
        ReDim arrValues(1 To nums, 1 To 2)      '定义动态数组
        For j = 1 To nums                       '从一到记录总数做循环
            arrValues(j, 1) = " " & Adodc2.Recordset!billdate & "月份" '给数组赋值
            arrValues(j, 2) = Adodc2.Recordset!qty '给数组赋值
            Adodc2.Recordset.MoveNext            '记录集下移
        Next j
        MSC1.ChartData = arrValues               '图表显示数据
        MSC1.Plot.DataSeriesInRow = False        '为统一颜色
        MSC1.Title = "销售数量分析图"           '设置图表标题
        Frame2.Visible = False                  'Frame2 不可见
    End If
End If
If Opt2.Value = True Then                      '当选中销售金额单选框
    If Adodc2.Recordset.RecordCount > 0 Then    '如果记录数大于零
        Adodc2.Recordset.MoveFirst              '移动到第一条记录
        nums = Adodc2.Recordset.RecordCount      '将记录数赋给变量
        ReDim arrValues(1 To nums, 1 To 2)      '定义动态数组
        For j = 1 To nums                       '从一到记录总数做循环
            arrValues(j, 1) = " " & Adodc2.Recordset!billdate & "月份" '给数组赋值
            arrValues(j, 2) = Adodc2.Recordset!tsum '给数组赋值
            Adodc2.Recordset.MoveNext            '记录集下移
        Next j
        MSC1.ChartData = arrValues               '图表显示数据
        MSC1.Plot.DataSeriesInRow = False        '为统一颜色
        MSC1.Title = "销售金额分析图"           '设置图表标题
        Frame2.Visible = False                  'Frame2 不可见
    End If
End If
End If
Next i

```



End Sub

在窗体的 Load 事件下, 主要完成数据控件的连接、初始化工作, 以及向 ComboBox 控件中添加项目。关键代码如下:

```
Private Sub Form_Load()
    Adodc1.ConnectionString = PublicStr
    Adodc1.RecordSource = "select * from tbS_sell_detailed"
    Adodc1.Refresh
    Adodc2.ConnectionString = PublicStr
    Adodc2.RecordSource = "select * from tbS_temp"
    Adodc2.Refresh
    Adodc3.ConnectionString = PublicStr
    Adodc3.RecordSource = "select * from tbS_resell_detailed"
    Adodc3.Refresh
    MSC1.Plot.DataSeriesInRow = True
    '设置窗体标题栏名称
    frm_saleImage.Caption = frm_saleImage.Caption & " 商品【" & frm_saleStatus.Adodc1.Recordset.Fields(1) & "】"
    '向 ComboBox 控件中添加项目
    Combo1.AddItem "条形图形", 0
    Combo1.AddItem "折线图形", 1
    Combo1.AddItem "面积图形", 2
    Combo1.AddItem "阶梯图形", 3
    Combo1.AddItem "组合图形", 4
    Combo1.ListIndex = 0
End Sub
```

'窗体加载事件  
'连接数据库  
'查询销售明细表中的数据  
'刷新  
'连接数据库  
'查询临时表中的数据  
'刷新  
'连接数据库  
'查询销售明细表  
'刷新  
'按数据网格的行读取  
'显示第一个项目的名称

设置三维图表的转角及仰角度数, 在 HScrollbar 控件的 Change 事件下添加代码如下:

```
Private Sub HS1_Change()
    MSC1.Plot.View3d.Set HS1.Value + 45, 15
End Sub
```

'设置三维图表的转角及仰角度数

本窗体中为了使窗体界面更加规范和清晰, 使用了 Toolbar 控件, 将所有的按钮都统一放置在功能工具栏上。在代码编写时, 利用在工具栏按钮中设置的按钮关键字, 来判断用户单击的是哪个按钮, 关键的代码如下:

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Key
        Case Is = "二维"
            '设置 ComboBox 控件的相关属性
            If Combo1.ListIndex = 0 Then MSC1.chartType = 1
            If Combo1.ListIndex = 1 Then MSC1.chartType = 3
            If Combo1.ListIndex = 2 Then MSC1.chartType = 5
            If Combo1.ListIndex = 3 Then MSC1.chartType = 7
            If Combo1.ListIndex = 4 Then MSC1.chartType = 9
        Case Is = "三维"
            '设置 ComboBox 控件的相关属性
            If Combo1.ListIndex = 0 Then MSC1.chartType = 0
            If Combo1.ListIndex = 1 Then MSC1.chartType = 2
    End Select
End Sub
```

'功能工具栏按钮事件  
'判断按钮的关键字  
'如果单击“二维”按钮  
'如果单击“三维”按钮

```

If Combo1.ListIndex = 2 Then MSC1.chartType = 4
If Combo1.ListIndex = 3 Then MSC1.chartType = 6
If Combo1.ListIndex = 4 Then MSC1.chartType = 8
Case Is = "自动取色"
    MSC1.Plot.DataSeriesInRow = True
    Toolbar1.Buttons(9).Caption = "恢复颜色"
    Toolbar1.Buttons(9).Key = "恢复颜色"
    With MSC1.Legend
        .Location.Visible = True
        .Location.LocationType = VtChLocationTypeRight
        .TextLayout.HorzAlignment = VtHorizontalAlignmentRight
        .VtFont.VtColor.Set 255, 255, 0
        .Backdrop.Fill.style = VtFillStyleBrush
        .Backdrop.Fill.Brush.style = VtBrushStyleSolid
        .Backdrop.Fill.Brush.FillColor.Set 255, 0, 255
    End With
Case Is = "恢复颜色"
    MSC1.Plot.DataSeriesInRow = False
    Toolbar1.Buttons(9).Caption = "自动取色"
    Toolbar1.Buttons(9).Key = "自动取色"
Case Is = "分析月份"
    Frame2.Visible = True
Case Is = "退出"
    Unload Me
End Select
End Sub

```

'如果单击“自动取色”按钮  
'按数据网格的行读取  
'设置工具栏按钮文字  
'设置工具栏按钮关键字  
'With 语句用来在一个单一对象或一个用户定义类型上执行一系列的语句  
'将图例设置为可见  
'设置图例在右边  
'右对齐  
'使用黄色文本  
'纯色填充  
'纯色画笔  
'设置颜色为紫色

'如果单击“恢复颜色”按钮  
'按数据网格的列读取  
'设置工具栏按钮文字  
'设置工具栏按钮关键字  
'如果单击“分析月份”按钮  
'Frame2 可见  
'如果单击“退出”按钮  
'退出本窗体

## 27.13 用户权限设置模块设计

教学录像：光盘\TM\lx\27\用户权限设置模块设计.exe

本模块使用的数据表：tbS\_power

用户权限设置模块可以对系统中的用户进行增、删、改、查操作，并可以对登录系统的用户进行权限设置操作。用户在主界面中选择“系统维护”/“操作权限设置”命令，即可进入到如图 27.45 所示的系统用户及权限设置窗体中。

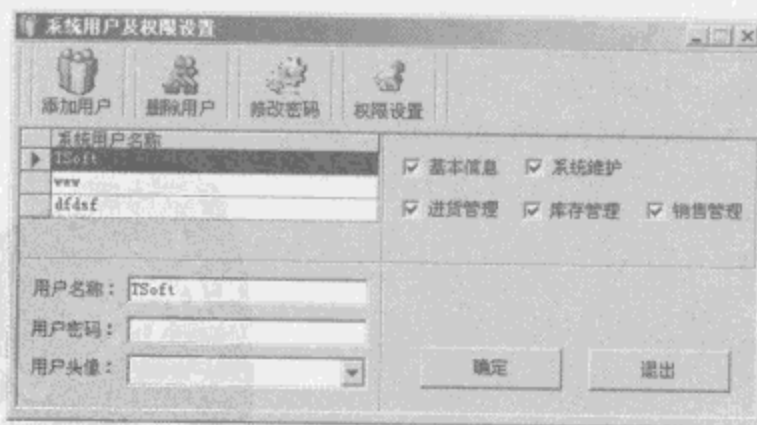


图 27.45 用户权限设置模块

### 27.13.1 设计窗体界面

用户权限设置窗体的界面设计过程如下。

(1) 在工程中新建一个窗体，将窗体的名称命名为 frm\_setOP，BorderStyle 属性设置为 1-Fixed Single，Caption 属性设置为“系统用户及权限设置”，MaxButton 属性为 False，MDIChild 属性为 True。

(2) 在窗体上添加一个 Toolbar 控件和一个 ImageList 控件，向 ImageList 控件中添加图片，并将这两个控件连接起来。

(3) 在窗体上添加两个 ADO 控件，使用其默认名，用于连接数据库。

(4) 在窗体上添加一个 PictureBox 控件，使用其默认名，充当一个容器的作用，用于容纳其他的控件。

(5) 在 Picture1 上添加两个 Frame 控件，用于将 Picture1 分割成 4 个区域：

①在第一象限区域放置 5 个 CheckBox 控件，设置为控件数组，用于设置操作员的权限。

②在第二象限的区域放置一个 DataGrid 控件，用于显示系统用户的名称。

③在第三象限的区域放置 3 个 Label 控件，两个 TextBox 控件；一个 ImageList 控件，命名为 Imt\_Tx；一个 ImageCombo 控件。该控件是 ActiveX 控件，在使用前需要将其添加到工具箱中。具体的方法选择“工程”/“部件”命令，在弹出的“部件”对话框中，选中 Microsoft DataList Controls 6.0 (SP3) (OLEDB) 项，即可将该控件添加到工具箱上。

④在第四象限的区域放置两个 CommandButton 控件，分别命名为 CmdSave 和 cmdExit，设置 Caption 属性为“确定”和“退出”。

### 27.13.2 窗体初始化

在启动该窗体时，要对窗体中的信息进行初始化，首先将 ADO 控件与数据库进行连接，然后设置控件的状态，并向 ImageCombo 控件中添加用户头像，关键代码如下：

```
Private Sub Form_Load()  
    Adodc1.ConnectionString = PublicStr  
    Adodc1.RecordSource = "select sysuser from tbS_power"  
    Adodc1.Refresh  
    Set DataGrid1.DataSource = Adodc1  
    Adodc2.ConnectionString = PublicStr  
    Adodc2.RecordSource = "select sysuser from tbS_power"  
    Adodc2.Refresh  
    Frame1.Enabled = False  
    Frame2.Enabled = False  
    DataGrid1.Enabled = False  
    Dim NewItem As Comboltem  
    Dim i As Integer  
    For i = 1 To 10  
        '连接数据库  
        '查询权限表内容  
        '刷新  
        '绑定  
        '连接数据库  
        '查询权限表中内容  
        '刷新  
        'Frame1 控件不可用  
        'Frame2 控件不可用  
        'DataGrid1 控件不可用  
        '声明一个 Comboltem 对象  
        '声明一个整型变量  
        '循环添加头像  
    Next i  
End Sub
```

```

Set NewItem = ImageCombo1.Comboltems.Add(i, "头像" & i, "头像" & i, "头像" & i)
Next i
End Sub

```

### 27.13.3 工具栏按钮

在本窗体中使用了工具栏, 该工具栏主要用于设置相关控件的状态, 并设置窗体级标识变量 B 的值, 在“确定”按钮中, 将根据标识变量 B 的值的不同来执行相关的操作。关键代码如下:

```

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Key
        Case Is = "添加用户"
            B = "TJYH"
            Frame1.Enabled = True
            Text1(0).Text = "": Text1(1).Text = ""
            ImageCombo1.Text = ""
            If Text1(0).Enabled = True Then Text1(0).SetFocus
            Frame2.Enabled = False
            DataGrid1.Enabled = False
            Case Is = "删除用户"
                B = "SCYH"
                DataGrid1.Enabled = True: Frame1.Enabled = False: Frame2.Enabled = False
            Case Is = "修改密码"
                B = "XGMM"
                Text1(0).Enabled = False: DataGrid1.Enabled = True: Frame1.Enabled = True: Frame2.Enabled = False
            Case Is = "权限设置"
                B = "QXSZ"
                Frame2.Enabled = True: DataGrid1.Enabled = True: Frame1.Enabled = False
    End Select
End Sub

```

'利用关键字查询  
'如果单击“添加用户”按钮  
'设置标识变量  
'设置 Frame1 控件可用  
'设置文本框内容为空  
'设置头像框内容为空  
'使控件获得焦点  
'设置 Frame2 控件不可用  
'设置 DataGrid 控件不可用  
'如果单击“删除用户”按钮  
'设置标识变量  
'设置控件状态  
'如果单击“修改密码”按钮  
'设置标识变量  
'如果单击“权限设置”按钮  
'设置标识变量

### 27.13.4 执行操作

当用户设置完相应的操作以后, 可以通过单击“确定”按钮来执行该操作。在“确定”按钮的单击事件中, 可根据标识变量 B 的值的不同, 判断执行的是什么操作, 关键代码如下:

```

Private Sub cmdsave_Click()
    If B = "TJYH" Then
        If Text1(0).Text = "" Then MsgBox "用户名称不能为空!": Exit Sub
        Adodc1.RecordSource = "select * from tbS_power where sysuser=" & Text1(0).Text & ""
        Adodc1.Refresh
        If Adodc1.Recordset.RecordCount > 0 Then
            MsgBox "此用户名称已经存在!"
            Exit Sub
    End If
End Sub

```

'单击“确定”按钮  
'如果是添加用户操作  
'限制用户名称不能为空  
'刷新  
'如果记录数大于零  
'弹出提示信息  
'如果此用户名存在, 将结束当前过程, 重新添加一新用户名



```

Text1(0).Text = ""
Text1(0).SetFocus
End If
Adodc1.Recordset.AddNew
Adodc1.Recordset.Fields("sysuser") = Text1(0).Text
Adodc1.Recordset.Fields("password") = Text1(1).Text
Adodc1.Recordset.Fields("head") = ImageCombo1.Text
Adodc1.Recordset.Update
MsgBox "系统操作用户添加成功！"
Adodc1.Refresh
Text1(0) = ""; Text1(1) = ""
Frame1.Enabled = True
B = ""

Elseif B = "SCYH" Then
    '提示是否删除
    If MsgBox("确定要删除用户名称为：" + Adodc1.Recordset.Fields(0) + "吗？", vbYesNo + vbQuestion) =
vbYes Then
        Adodc1.Recordset.Delete
        End If
        DataGrid1.Enabled = False
        B = ""
    Elseif B = "XGMM" Then
        '查询数据库中该操作员的信息
        Adodc1.RecordSource = "select * from tbS_power where sysuser=" + Text1(0).Text + ""
        Adodc1.Refresh
        Adodc1.Recordset.Fields("password") = Text1(1).Text
        Adodc1.Recordset.Update
        MsgBox "密码修改成功！"
        Adodc1.RecordSource = "select * from tbS_power"
        Adodc1.Refresh
        '设置控件状态
        DataGrid1.Enabled = False : Text1(0).Enabled = True: Frame1.Enabled = False
        Text1(0).Text = ""; Text1(1).Text = ""
        B = ""
    Elseif B = "QXSZ" Then
        '查询数据表中操作员的信息
        Adodc2.RecordSource = "select * from tbS_power where sysuser=" + Adodc1.Recordset.Fields(0) + ""
        Adodc2.Refresh
        For i = 2 To Check1.Count + 1
            If Check1(i - 2).Value = 1 Then Adodc2.Recordset.Fields(i).Value = 1
            If Check1(i - 2).Value = 0 Then Adodc2.Recordset.Fields(i).Value = 0
            Adodc2.Recordset.Update
        Next
        MsgBox "权限设置成功！"
        Frame2.Enabled = False: DataGrid1.Enabled = False
        B = ""
    End If
End Sub

```

'清空文本框  
'使控件获得焦点

'新增一条记录  
'向数据库中写入用户名  
'向数据库中写入密码  
'向数据库中写入用户头像  
'更新数据  
'提示保存成功  
'刷新  
'清空文本框  
'设置 Frame1 可用  
'清空标识变量  
'如果为删除用户操作

'删除该用户

'设置 DataGrid 控件不可用  
'清空标识变量  
'如果执行修改密码操作

'刷新  
'向数据库中写入新密码  
'更新数据库  
'弹出提示对话框  
'查询权限表中数据  
'刷新


'设置文本框内容  
'清空标识变量  
'如果执行权限设置操作

'刷新  
'从 2 到数组下标做循环  
'设置操作员的权限  
'设置操作员的权限  
'更新数据库

'弹出提示框  
'设置控件状态  
'清空标识变量

## 27.14 运行项目

 教学录像：光盘\TM\lx\27\运行项目.exe

模块设计及代码编写完成之后，单击 VB 开发环境工具栏中的  图标，或者在菜单中选择“运行”/“启动”命令，运行该项目，弹出企业进销存管理系统“启动”界面，如图 27.46 所示。

启动界面过后，就进入到系统的“登录”对话框中，如图 27.47 所示。



图 27.46 “启动”对话框

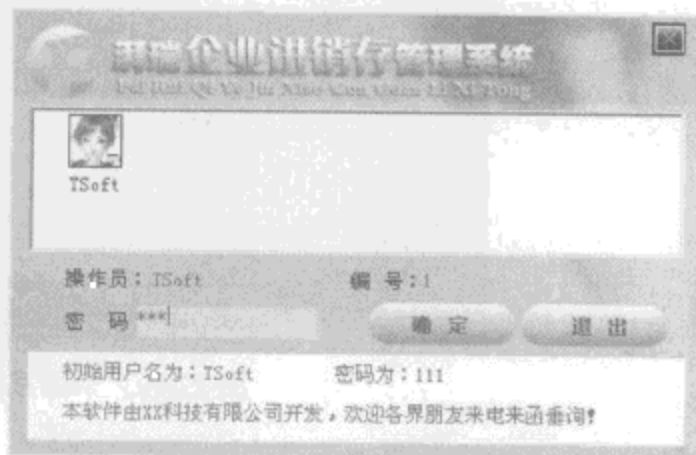


图 27.47 系统登录界面

在“登录”对话框中输入用户名和密码，单击“确定”按钮，进入企业进销存管理系统的主窗体。然后用户可以对主窗体中的菜单栏、工具栏进行操作，以调用其各个子模块。如，在主窗体中单击工具栏中的“商品进货”按钮，可以弹出“商品进货”窗体，如图 27.48 所示，该窗体中，用户可以将所采购商品的信息批量保存到入库表和库存表中。

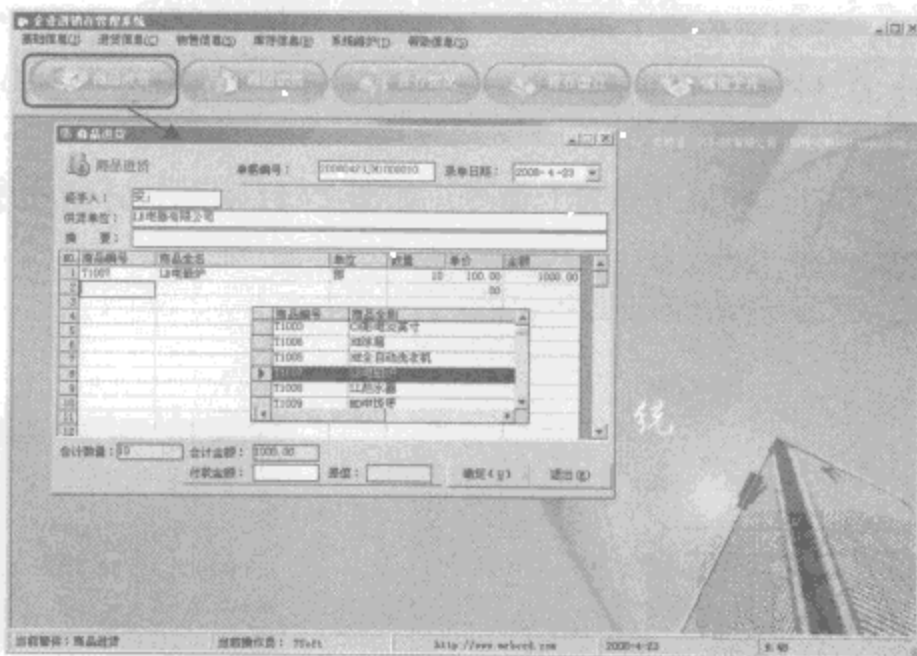


图 27.48 “商品进货”窗体

又如，在主窗体中选择“销售信息”/“月销售状况”命令，即可弹出“月销售状况”对话框，在该对话框中单击“销售分析”按钮，将弹出“每月销售比较”对话框，如图 27.49 所示。在该对话框中

可以对销售信息进行分析比较。

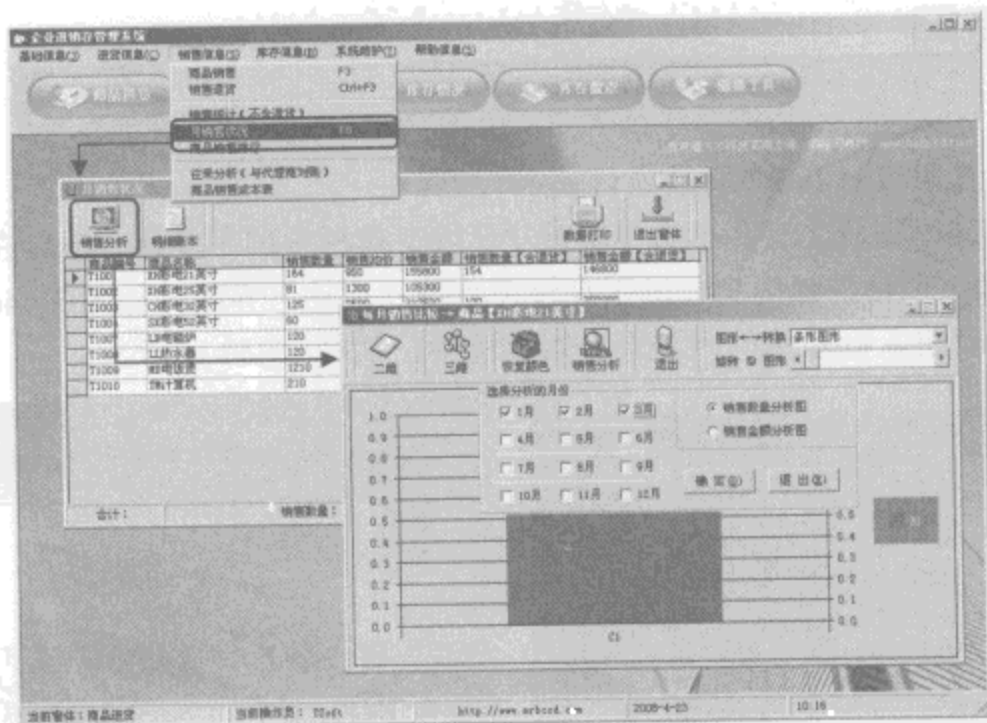


图 27.49 “每月销售比较”对话框

## 27.15 程序打包

**教学录像：**光盘\TM\lx\27\程序打包.exe

系统开发完成之后，如何将本系统打包并制作成安装程序在客户机上安装运行呢？这就需要已经写好的程序进行打包，然后利用打包生成的 SetUp.exe 文件进行安装。

当项目开发完成以后，需要首先将工程保存，然后生成可执行文件。选择“外接程序”/“外接程序管理器”命令，在弹出的“外接程序”对话框中，加载“打包和展开向导”，如图 27.50 所示。单击“确定”按钮完成加载设置。

再选择“外接程序”/“打包和展开向导”命令，如图 27.51 所示，开始程序打包。

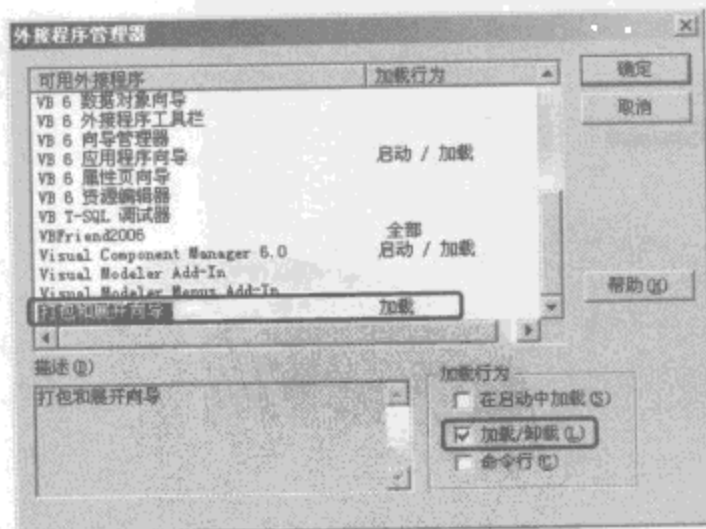


图 27.50 外接程序管理器

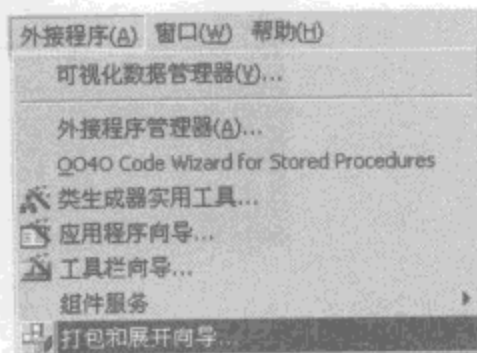


图 27.51 打包和展开向导命令

程序的打包在第 26 章已经进行了详细介绍,读者可以参见相关的知识。

## 27.16 开发常见问题与解决

教学录像: 光盘\TM\lx\27\开发常见问题与解决.exe

### 27.16.1 书写错误的函数名

运行程序之后,进入到“商品进货”模块中,在窗体加载时,弹出如图 27.52 所示的对话框。单击“确定”按钮,进入调试界面,如图 27.53 所示。

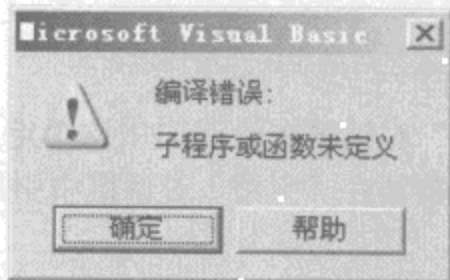


图 27.52 添加信息时弹出的错误信息

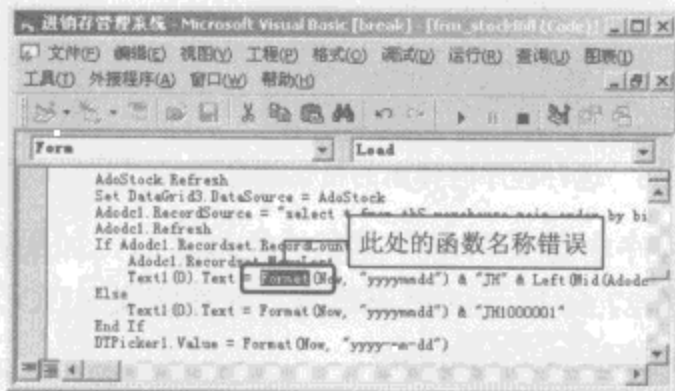


图 27.53 调试之后的 VB 开放环境窗口

发现在编写应用程序的过程中,函数的名称书写错误:在图 27.53 中应用格式化函数 Format 时,错误地将函数名称书写为 Formet,因此在运行程序时弹出了错误信息。

将出错处的程序代码中的函数名称书写正确(将函数名 Formet 修改为 Format),即可排除错误。修改后相关的程序代码如下:

```
Txt_id.Text = Format(Date, "yyyymmdd") & "00001"
```

### 27.16.2 提示文件未找到的错误信息

运行程序之后,选择辅助工具中的计算器命令,即可调用辅助工具计算器。在调试时发现,在单击“计算器”按钮时,弹出如图 27.54 所示的错误提示信息,单击调试按钮,进入到代码编辑器如图 27.55 所示。发现在利用 Shell 函数调用可执行文件的时候,可执行文件的名称书写不正确。

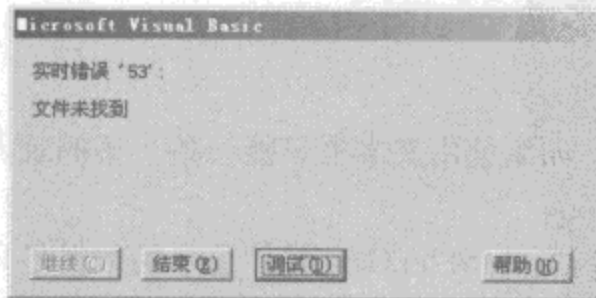


图 27.54 在进行数据恢复时弹出的错误信息

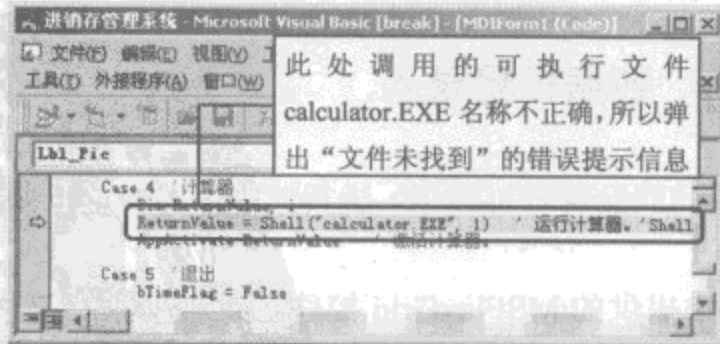


图 27.55 调试之后的 VB 开放环境窗口



在程序编写时,如果输入的文件名错误,或者该路径下不存在这个文件时,将出现文件未找到的错误信息,这里只需将 calculator.EXE 文件名改为正确的文件名 calc.EXE 即可正确执行。

### 27.16.3 解决用户定义类型未定义的问题

在运行程序时,弹出如图 27.56 所示的提示信息,调试之后的 VB 开发环境如图 27.57 所示。

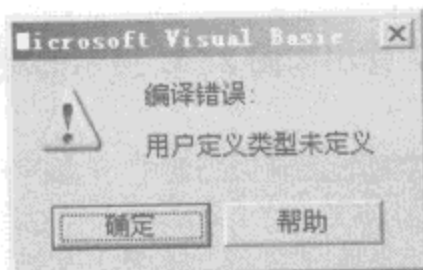


图 27.56 运行程序时弹出的错误信息

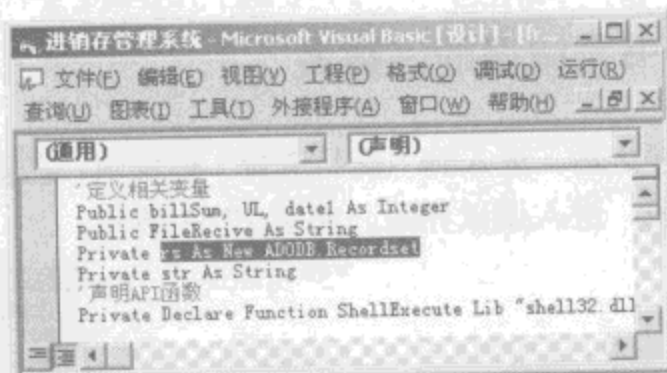


图 27.57 调试之后的 VB 开发环境窗口

在应用程序中定义了用户需要的类型之后,如果在“引用”对话框中没有引用这个定义的类型,就会弹出如图 27.56 所示的错误信息。如在定义 ADO 对象时,需要在“引用”对话框中引用 Microsoft ActiveX Data Objects 2.5 Library。

在 VB 开发环境中,选择“工程”/“引用”命令,在弹出的“引用”对话框中将 ADO 对象引用到 VB 开发环境中,就可解决这个错误,如图 27.58 所示。

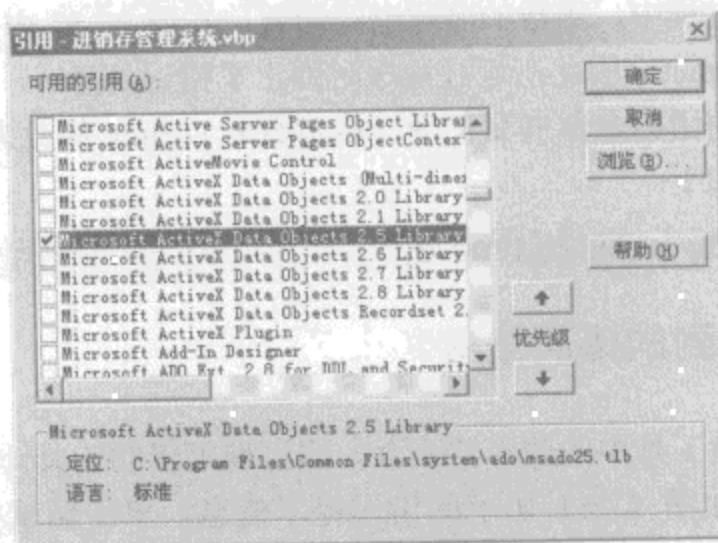


图 27.58 引用 ADO 对象

### 27.16.4 数据批量录入

实际的商品进货过程中,进货的商品品种多且数量大,如果使用文本框只能一条一条的录入数据,效率太低。

VB 中提供的 MSFlexGrid 控件,处理数据比较灵活。虽然 MSFlexGrid 控件显示的数据是只读的,但可以通过 TextBox 控件向 MSFlexGrid 控件中输入数据,然后使用 For 循环逐一将表格中数据添加到

数据表中。具体的实现方法如下。

(1) 初始化 MSFlexGrid 控件。关键代码如下:

```
MS1.Rows = 100: MS1.Cols = 7                                '定义 MS1 控件的总行数、总列数
'定义 MS1 表的宽度
MS1.ColWidth(0) = 12 * 25 * 1: MS1.ColWidth(1) = 12 * 25 * 4: MS1.ColWidth(2) = 12 * 25 * 9
MS1.ColWidth(3) = 12 * 25 * 3: MS1.ColWidth(4) = 12 * 25 * 3: MS1.ColWidth(5) = 12 * 25 * 3
MS1.ColWidth(6) = 12 * 25 * 4
MS1.FixedRows = 1: MS1.FixedCols = 1                        '设置固定行、列
'定义 MS1 表的表头
MS1.TextMatrix(0, 0) = "NO. ": MS1.TextMatrix(0, 1) = "商品编号": MS1.TextMatrix(0, 2) = "商品全名"
MS1.TextMatrix(0, 3) = "单位": MS1.TextMatrix(0, 4) = "数量": MS1.TextMatrix(0, 5) = "单价"
MS1.TextMatrix(0, 6) = "金额"
For i = 1 To 99
    MS1.TextMatrix(i, 0) = i                                '定义 MS1 表的列序号
Next i
'装载窗体时, 确定 text3 的位置
Text3.Text = ""                                             '清空文本内容
Text3.Width = MS1.CellWidth: Text3.Height = MS1.CellHeight '设置文本框宽度和高度
Text3.Left = MS1.CellLeft + MS1.Left: Text3.Top = MS1.CellTop + MS1.Top '设置文本框坐标
```

(2) 确定 TextBox 控件在 MSFlexGrid 表格上的位置及大小。

```
Private Sub MS1_EnterCell()
    Dim X As String, y As String, p As String                '定义变量
    If MS1.CellWidth <= 0 Or MS1.CellHeight <= 0 Then Exit Sub '如果单元格宽和高为 0, 则退出
    X = MS1.TextMatrix(MS1.FixedRows, MS1.Col)              '给变量赋值
    y = MS1.TextMatrix(MS1.Row, 0)                          '给变量赋值
    If y <> "" Then                                           '如果变量值不为空
        If MS1.Col - MS1.LeftCol <= 3 Then                   '列数与左端可见列数差值小于或等于 3
            MS1.LeftCol = MS1.LeftCol + 1                    '左端可见列数加 1
        End If
        If MS1.CellWidth > 0 And MS1.CellHeight > 0 Then    '如果单元格宽度大于 0
            Text3.Width = MS1.CellWidth: Text3.Height = MS1.CellHeight '设置文本框宽度
            Text3.Left = MS1.CellLeft + MS1.Left              '设置文本框 Left 属性
            Text3.Top = MS1.CellTop + MS1.Top                  '设置文本框 Top 属性
        End If
        X = MS1.TextMatrix(MS1.FixedRows, MS1.Col)          '给变量赋值
        y = MS1.TextMatrix(MS1.Row, 0)                       '给变量赋值
        p = MS1.TextMatrix(MS1.Row, MS1.Col)                 '给变量赋值
        Text3.Text = MS1.Text                                 '赋值给 text3.text
    End If
End Sub
```

(3) 在 TextBox 控件上输入数据后, 按〈Enter〉键 TextBox 控件移动到一个单元格或换行。关键代码如下:

```
Private Sub Text3_KeyPress(KeyAscii As Integer)
    'vbKeyReturn 常数为键盘上的"回车键"
    If KeyAscii = vbKeyReturn Then                           '按〈Enter〉键, text3 向右移动
```

```

MS1.Text = Text3.Text           '设置单元格内容
Text3.Text = MS1.Text          '设置文本框内容
If MS1.Col = 5 Then             '如果列值为 5
    MS1.Row = MS1.Row + 1      '行值加 1
    MS1.Col = 1                '列值为 1
Else
    If MS1.Col + 1 <= MS1.Cols - 1 Then '如果列数不是最大值
        MS1.Col = MS1.Col + 1    '列值加 1
    Else
        If MS1.Row + 1 <= MS1.Rows - 1 Then '行值加 1 不是最大值
            MS1.Row = MS1.Row + 1    '行数加 1
            MS1.Col = 1              '列数为 1
        End If
    End If
End If
End If
End Sub

```

(4) 使用 For 循环将 MSFlexGrid 表格中的数据添加到数据表中。关键代码如下：

```

For i = 1 To 99
    With Adodc2.Recordset
        .AddNew                'With 语句用来在一个单一对象上执行一系列的语句
                                '开辟数据存储空间
        .Fields("billcode") = Text1(0).Text    '单据号
        .Fields("billdate") = DTPicker1.Value  '开单日期
        If MS1.TextMatrix(i, 1) <> "" Then .Fields("trade") = Trim(MS1.TextMatrix(i, 1))
        If MS1.TextMatrix(i, 2) <> "" Then .Fields("fullname") = Trim(MS1.TextMatrix(i, 2))
        If MS1.TextMatrix(i, 3) <> "" Then .Fields("unit") = Trim(MS1.TextMatrix(i, 3))
        If MS1.TextMatrix(i, 4) <> "" Then .Fields("qty") = Trim(MS1.TextMatrix(i, 4))
        If MS1.TextMatrix(i, 5) <> "" Then .Fields("price") = Trim(MS1.TextMatrix(i, 5))
        If MS1.TextMatrix(i, 6) <> "" Then .Fields("tsum") = Trim(MS1.TextMatrix(i, 6))
        .Update                '将数据保存到 AddNew 方法开辟的存储空间
    End With
Next i

```

### 27.16.5 使用数据回滚来保护数据恢复工作

通常恢复系统数据时直接使用 ("use master RESTORE DATABASE db\_SSS from disk="" + Text2.Text + """) 来恢复数据。当使用 ADO 控件连接数据库时，使用该方法恢复数据将会失败。

若解决上述问题，应在执行恢复 SQL 语句前，先设置为单用户访问数据库，然后进行数据回滚恢复数据库备份 ("Alter DATABASE db\_SSS set single\_user with rollback immediate use master RESTORE DATABASE db\_SSS from disk="" + Text2.Text + """)，最后将数据库设置回原来的多用户访问数据库 ("Alter DATABASE db\_SSS set multi\_user")，否则，单用户访问数据库失败。本系统实现恢复数据功能的主要代码如下：

```
Private Sub cmdRestore_Click()
```

```

If Text1.Text = "" Then                                '如果文本框为空
    MsgBox "请选择数据恢复文件!"                     '提示信息
    Exit Sub                                           '退出程序
End If
On Error GoTo MyErr                                   '错误处理
Call cnn                                              '调用自定义过程
Set rs = New ADODB.Recordset                         '实例化数据集
'执行 SQL 语句修改数据库
cnn.Execute ("Alter DATABASE db_SSS set single_user with rollback immediate use master RESTORE
DATABASE db_SSS from disk='" + Text2.Text + "'")
Str = "Alter DATABASE db_SSS set multi_user"          '给变量赋值
'Connection 对象的 Execute 方法, 执行指定的查询、SQL 语句、存储过程或特定提供者的文本等内容
Set rs = cnn.Execute(Str)                             '执行 SQL 语句
MsgBox "数据库恢复成功!"                             '弹出提示框
cnn.Close                                             '关闭数据连接
Exit Sub                                              '退出过程
MyErr:
'弹出提示框
MsgBox "由于系统数据量大,数据库恢复已经超时,请退出系统重新操作数据备份!!!", "系统提示"
cnn.Close                                             '关闭数据连接
End
End Sub                                              '结束

```

### 27.16.6 字段大小问题导致数据添加失败

例如：库存商品表中商品全名字段，如表 27.15 所示。

表 27.15 库存商品表中商品全名字段

字 段	类 型	大 小	描 述
fullname	varchar	10	商品全称

当用户输入的商品名称大于 10 个字符时，系统将会出错，如图 27.59 所示。

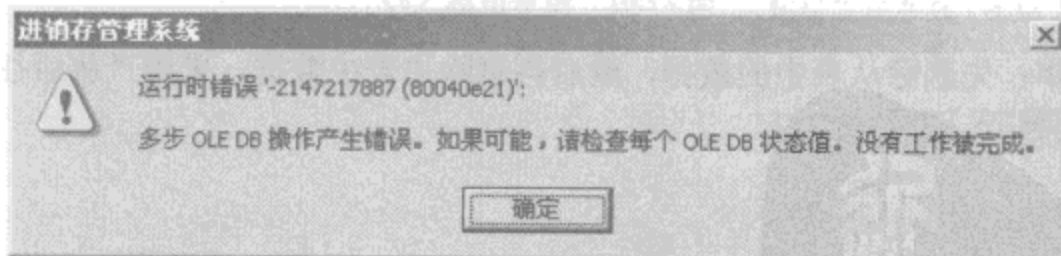


图 27.59 错误信息 (1)

解决方法为将字段大小更改为 20 或更大（合理即可），如表 27.16 所示。

表 27.16 改变字段大小

字 段	类 型	大 小	描 述
fullname	Varchar	20 或更大	商品全称



### 27.16.7 字段设置主键后不能插入重复值

当员工信息表 (tbS\_employ) 中 fullname (员工姓名) 字段设置主键后, 如果重复向 fullname 字段中添加员工姓名后, 将出现如图 27.60 所示的错误。

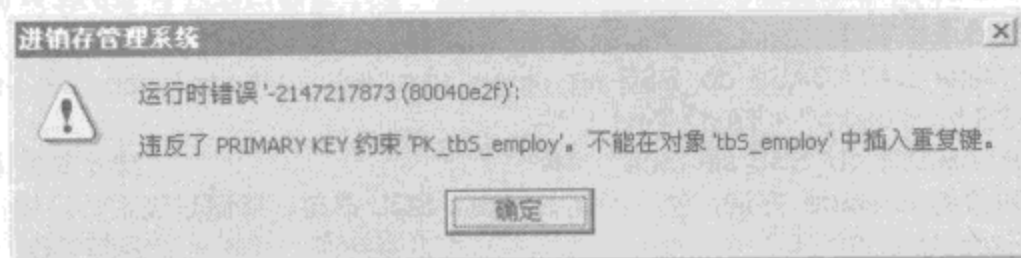


图 27.60 错误信息 (2)

解决方法: 在添加数据库时, 先进行查询, 看看设置主键的字段中是否已经添加了该数据, 然后作出相应的处理。

### 27.16.8 数据库中表存在关系, 如何进行数据库清理

在开发数据清理模块 (用来清理系统数据库数据) 时, 经常会出现如图 27.61 所示的错。根据笔者经验, 原因是数据库中存在表关系。

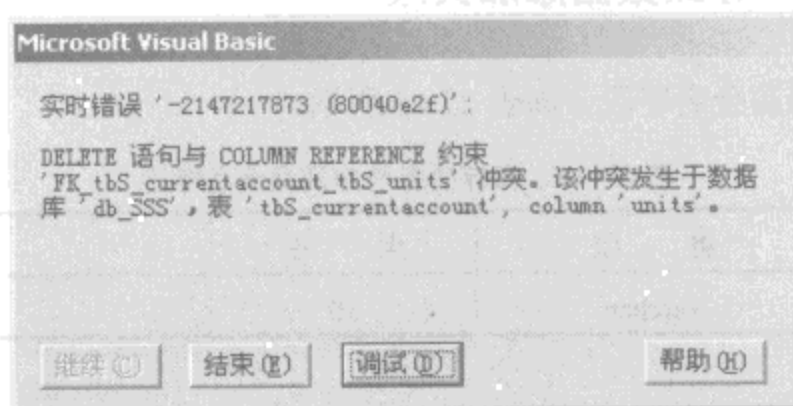


图 27.61 错误信息 (3)

解决方法很简单: 先删除从表中的数据, 然后再删除主表中数据, 采用“从后往前清理”的方法。

## 27.17 小结

至此, 一个完整的企业进销存管理系统就介绍完了。本系统利用前面介绍的知识, 并结合软件工程的设计思想开发了一个完整的企业进销存管理系统。(实例位置: 光盘\TM\27\企业进销存管理系统)

本章从系统分析、系统设计到数据库、数据表的设计, 以及各个模块的设计实现, 为读者详细介绍了一个系统的开发流程。本章着重介绍了利用 Flash 控件的 FSCommand 事件实现 Flash 和 VB 的交互的知识, 希望可以拓展读者的开发思路, 并可以对读者日后的程序开发有一定的帮助。

## 同步语音视频讲解，在线立体全程服务



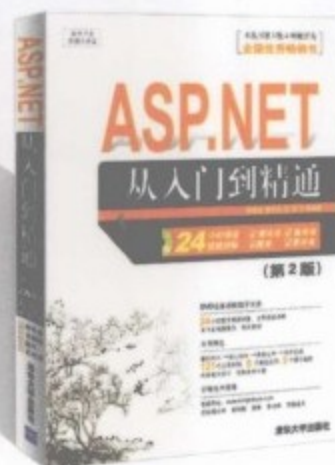
书号: 978-7-302-22661-1  
定价: 69.80元



书号: 978-7-302-22662-8  
定价: 69.80元



书号: 978-7-302-22744-1  
定价: 49.80元



书号: 978-7-302-22745-8  
定价: 79.80元



书号: 978-7-302-22746-5  
定价: 59.80元



书号: 978-7-302-22747-2  
定价: 69.80元



书号: 978-7-302-22792-2  
定价: 69.80元



书号: 978-7-302-22838-7  
定价: 49.80元

查阅本丛书详细信息:

<http://www.tup.com.cn>

<http://www.mingribook.com>

购买本丛书:

全国内地各大书城、书店

<http://www.dangdang.com>

<http://www.amazon.cn>

<http://www.china-pub.com>

新华书店  
PDG

ISBN 978-7-302-22661-1



9 787302 226611 >

定价: 69.80元 (附DVD视频光盘1张)