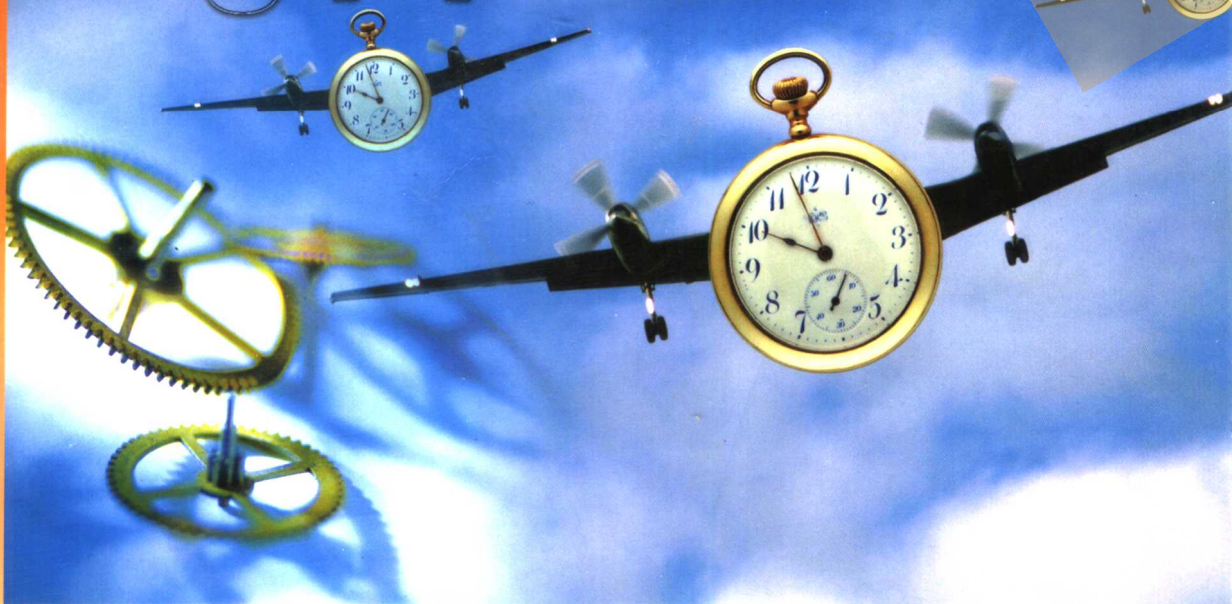


罗强
蔡兵兵
编著

Visual C++

doosriver 文川网
入驻商家
在文川网搜索古籍书城 发现更多电子书



循序渐进学

Visual C++ .NET

编程



- ◆从入门与实用的角度出发讲解，引导读者及时地掌握Visual C++ .NET的新技术，并配有大量的精选实例
- ◆条理化的结构安排，让读者快速查找解决各种问题的方法
- ◆提供最新的编程思想，能大幅度缩短您摸索的时间

北京科海电子出版社
<http://www.KHP.com.cn>

科海电脑技术丛书

循序渐进学 Visual C++.NET 编程

罗强 蔡兵兵 编著

北京科海电子出版社

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

15
32

15

内 容 提 要

读者对象:

- 希望快速掌握 Visual C++ .NET 的初学者
- 适用于从初级到高级水平的读者朋友

达成目标:

- 学会使用 Visual C++ .NET 设计程序
- 设计出属于您的网络程序

技能要求:

- 对 C 语言有深入了解
- 接触过 C++ 编程

本书特点:

- 从入门与实用角度出发讲解
- 条理化的结构安排,让读者快速查找解决各种问题的方法
- 提供最新的编程思想,能大幅度缩短您摸索的时间

这是一本有关 Visual C++ .NET 和 Microsoft .NET 的书籍。书中集成了大量 Visual C++ .NET 的新知识、新技术,倾注了笔者多年来的编程经验。

.NET 的平台及框架是基于微软软件工业基础的又一次升级和演化,在网络飞速发展的今天,.NET 将很快成为主流。

本书分为 5 大部分,共 16 章,内容包括: Visual C++ .NET 概述、Visual C++ .NET 的开发环境、面向对象编程与 C++ 语言、MFC 及 Framework、文档视图结构、图形设备接口 GDI+、公用控制、动态链接库、多线程编程技术、Internet 编程、.NET 编程等技术,每个技术都配有精选的实例和大量图片。本书由浅入深,简单易学,能快速引导您顺利进入 .NET 的编程世界。

本书可作为高等院校师生教学和自学参考用书,也可作为各类计算机培训班的教材。

品 名: 循序渐进学 Visual C++ .NET 编程
作 者: 罗 强 蔡兵兵
责任编辑: 徐建军
排 版: 吴文娟
出 品: 北京科海电子出版社
印 刷 者: 北京朝阳科普印刷厂印刷
发 行: 新华书店总店北京科技发行所
开 本: 787×1092 1/16 印张: 33.375 字数: 811 千字
版 次: 2002 年 8 月第 1 版 2002 年 8 月第 1 次印刷
印 数: 0001~5000
盘 号: ISBN 7-900107-37-1
定 价: 48.00 元 (1CD)

docsriver 文川网
入驻商家 古籍书城

在文川网搜索古籍书城 获取更多电子书

目 录

第 1 部分 Visual C++.NET 简介

第 1 章 Visual C++.NET 概述 1	(应用程序向导)..... 16
1.1 MFC 历史和 Visual C++..... 1	2.1.2 Solution Explorer..... 16
1.1.1 MFC 发展史..... 2	2.1.3 Properties Window(属性窗口)..... 22
1.1.2 MFC 类库概念和组成..... 3	2.1.4 Developer Studio.Net 的 一些快捷特性..... 22
1.1.3 MFC 的优点..... 9	2.2 创建、组织文件、工程和工作区..... 24
1.2 Visual C++ .NET 与老版本的区别..... 9	2.2.1 新建工程..... 24
1.3 Visual Studio.NET 新特性..... 11	2.2.2 打开工程项目..... 30
1.3.1 .NET 的创新..... 11	2.2.3 增加已有文件到工程中..... 30
1.3.2 .NET 与同类产品的比较..... 13	2.3 自定义 Developer Studio.Net..... 31
1.3.3 .NET 对 IT 专业人员的重要意义.. 13	2.3.1 自定义工具栏和菜单..... 31
1.4 本章小结..... 14	2.3.2 自定义快捷键..... 33
第 2 章 熟悉 Visual C++.NET 的 开发环境 15	2.3.3 使用宏..... 33
2.1 Visual C++.NET 可视化集成开发环境... 15	2.4 本章小结..... 35
2.1.1 Application Wizard	

第 2 部分 Visual C++语言基础

第 3 章 面向对象编程与 C++语言 36	3.6 模板..... 60
3.1 面向对象的编程技术..... 36	3.6.1 模板的概念..... 60
3.1.1 封装..... 37	3.6.2 为什么使用模板..... 62
3.1.2 继承..... 38	3.6.3 函数模板..... 63
3.1.3 多态和虚函数..... 40	3.6.4 类模板..... 64
3.2 类的声明和定义..... 42	3.7 本章小结..... 66
3.3 构造函数和析构函数..... 47	第 4 章 Windows 编程简介 67
3.3.1 类与对象..... 47	4.1 消息驱动的应用程序..... 67
3.3.2 构造函数..... 49	4.2 Win32 API 和 SDK..... 70
3.3.3 析构函数..... 54	4.3 Windows 应用程序的基本结构..... 71
3.4 友元..... 55	4.4 32 位编程的特点..... 85
3.5 C++语言的输入输出..... 58	4.5 本章小结..... 93
3.5.1 I/O 流结构..... 58	第 5 章 MFC 及 Framework 简介 94
3.5.2 其他输入/输出函数..... 59	

03517/101

5.1 MFC 概述.....	94	5.3 MFC 应用程序框架的运行机制.....	103
5.2 MFC 程序结构分析.....	95	5.4 .NET 框架类简介.....	108
5.2.1 类 CMyApp.....	97	5.4.1 公共语言运行时环境.....	109
5.2.2 类 CMainFrame.....	98	5.4.2 .NET Framework 类库.....	110
5.2.3 类 CMyView 与 CMyDoc.....	99	5.5 本章小结.....	117

第 3 部分 Visual C++ 编程初步

第 6 章 资源的使用.....	118	第 8 章 图形设备接口 GDI+.....	188
6.1 图标和光标.....	118	8.1 设备环境类.....	188
6.2 菜单与加速键.....	124	8.1.1 设备上下文工作原理.....	188
6.2.1 简介.....	124	8.1.2 实例绘图原理剖析.....	191
6.2.2 使用菜单.....	128	8.1.3 绘图操作实现.....	192
6.3 位图.....	132	8.1.4 基本绘图函数.....	193
6.3.1 位图简介.....	132	8.2 映射模式.....	195
6.3.2 使用工具栏.....	136	8.2.1 坐标与坐标模式.....	195
6.4 本章小结.....	142	8.2.2 生成 MyCoordinate 程序框架.....	196
第 7 章 文档视图结构.....	143	8.2.3 坐标方向.....	200
7.1 文档/视图概述.....	143	8.3 GDI+ 对象.....	206
7.2 文档/视图结构各类之间的关系.....	146	8.3.1 3 种图形输出类型.....	206
7.3 生成文档.....	150	8.3.2 MFC 中与 GDI+ 有关的类.....	207
7.3.1 概述.....	150	8.3.3 常见的绘图任务.....	210
7.3.2 保存文档数据.....	150	8.4 绘图程序.....	212
7.3.3 串行化数据.....	154	8.4.1 MDI 应用程序框架.....	212
7.3.4 使用集合类管理数据.....	159	8.4.2 设计绘图程序的文档类.....	214
7.3.5 串行化对象.....	168	8.4.3 设计绘图程序的视图类.....	220
7.4 CView 类与 CDC 类.....	169	8.5 使用画笔和画刷.....	228
7.4.1 概述.....	169	8.5.1 画笔.....	228
7.4.2 CView 类对象的工作机制.....	169	8.5.2 画刷.....	231
7.4.3 CView 类的应用.....	172	8.5.3 位图画刷.....	233
7.4.4 消息映射的使用.....	177	8.5.4 使用点、画刷和画笔进行绘图.....	235
7.5 常用 CView 类的其他派生类的应用.....	179	8.5.5 在窗口中绘制设备相关位图、 图标和设备无关位图.....	236
7.5.1 CScrollView 类.....	179	8.6 字体对象.....	238
7.5.2 CEditView 类、CRichEditView 类和 CFormView 类.....	186	8.6.1 在窗口中输出文字.....	238
7.6 添加多文档支持.....	186	8.7 本章小结.....	241
7.7 本章小结.....	187		

第 4 部分 基本控件

第 9 章 WinForm 与对话框	242	10.1.1 静态控件	289
9.1 对话框基类简介	242	10.1.2 按钮控件	289
9.2 对话框设计和应用	243	10.1.3 编辑框控件	292
9.2.1 对话框模板	243	10.1.4 滚动条控件	296
9.2.2 对话框类的创建	247	10.1.5 列表框控件	298
9.2.3 为对话框类加入成员变量	249	10.1.6 组合框控件	302
9.2.4 对话框的初始化	250	10.2 标准控件的使用	305
9.2.5 对话框的数据交换机制	252	10.3 本章小结	311
9.2.6 对话框的运行机制	253	第 11 章 公用控件	312
9.2.7 处理控件通知消息	255	11.1 公用控件类简介	312
9.3 非模态对话框	261	11.1.1 Win32 控件的通知消息	312
9.3.1 非模态对话框的特点	262	11.1.2 旋转按钮控件	314
9.3.2 窗口对象的自动清除	264	11.1.3 滑尺控件	317
9.4 Windows Forms	266	11.1.4 进度条控件	319
9.4.1 Windows Forms 的编程模型	267	11.1.5 树形视图控件	320
9.4.2 TuneTown 应用程序	270	11.1.6 列表视图控件	325
9.5 本章小结	287	11.2 测试新型 Win32 控件的一个例子	329
第 10 章 标准控件	288	11.3 本章小结	335
10.1 标准控件类简介	288		

第 5 部分 高级编程

第 12 章 动态链接库	336	第 14 章 数据库管理	368
12.1 概述	336	14.1 ODBC 编程简介	368
12.2 创建和使用动态链接库	338	14.1.1 概述	368
12.2.1 DLL 的结构和导出方式	339	14.1.2 MFC 提供的 ODBC 数据库类	369
12.2.2 链接应用程序到 DLL	343	14.1.3 应用 ODBC 编程	369
12.3 使用动态链接库扩展 MFC	346	14.2 DAO	389
12.4 本章小结	351	14.2.1 DAO 概述	389
第 13 章 多线程编程技术	352	14.2.2 DAO 和 ODBC 的相似之处	389
13.1 概述	352	14.2.3 DAO 的特色	390
13.2 基于 Visual C++ .NET 的多线程编程	353	14.2.4 使用 ODBC 还是 DAO	391
13.3 基于 MFC 的多线程编程	356	14.3 ADO 简介	391
13.4 .NET Framework 对多线程的支持	364	14.3.1 ADO 概述	392
13.5 本章小结	367	14.3.2 在 VC 中使用 ADO	392

14.4	ADO.NET 简介	396	15.3.3	使用 Global.aspx 文件	440
14.5	对于 SQL Server 的编程	399	15.3.4	管理应用状态	443
14.6	本章小结	402	15.3.5	HttpHandlers 和 Factories	466
第 15 章	Internet 编程	403	15.3.6	用 ASP.NET 来创建 Web 程序 ..	468
15.1	ISAPI	403	15.4	Web Service	472
15.1.1	在现实世界中使用 ISAPI	404	15.4.1	Web Service 简介	472
15.1.2	使用 ISAPI 的 5 个类	405	15.4.2	建立 Web Service 最简单的方式	478
15.1.3	创建 ISAPI 扩展	405	15.5	COM+: Internet 的未来	482
15.1.4	ISAPI 扩展测试	409	15.6	本章小结	483
15.1.5	创建 ISAPI 过滤器	412	第 16 章	Visual C++.NET 的	
15.1.6	使用 ISAPI 过滤器划分 Web 站点的方法	414		.NET 编程	484
15.1.7	使用 ISAPI 扩展转发 服务器信息	415	16.1	.NET 简介与基本知识	484
15.1.8	设计 Web 页	420	16.2	SOAP 协议简介	489
15.1.9	HTML 概要	422	16.2.1	SOAP 定义	489
15.1.10	链接与锚地	424	16.2.2	ASP.NET Web Services	491
15.2	WinInet	429	16.2.3	Microsoft .NET Remoting	494
15.2.1	简介	429	16.2.4	ATL Server Web Services	497
15.2.2	方法与技巧 (Tips & Knacks)	430	16.3	XML 简介	500
15.2.3	HTTP 应用实现步骤	433	16.3.1	基本概念	500
15.2.4	FTP 应用实现步骤	433	16.3.2	XML 和 Web 服务在 .NET 中 的实际应用	504
15.2.5	Gopher 应用实现步骤	434	16.3.3	在 Visual C++.NET 中编写一个 综合的 .NET 应用程序	513
15.3	ASP.NET 应用	438	16.4	Microsoft .NET 的现实意义	523
15.3.1	创建应用	439	16.5	本章小结	525
15.3.2	应用的生存期	440			

第 1 章 Visual C++.NET 概述

MFC 类库是 Visual C++ 的基础与核心部分,可以说学习使用 Visual C++ 的过程就是学习使用 MFC 类库的过程, MFC 类库庞大而繁杂由数百个类组成,类与类之间的层次关系也颇为复杂,后面将有专门的章节讲述 MFC 类库的基本结构,本章主要介绍其发展史及其基本概念,此外还将简要介绍 Visual Studio.NET 集成开发环境的一些新特性与功能。

1.1 MFC 历史和 Visual C++

微软基础类库 (MFC, Microsoft Foundation Class) 是微软为 Windows 程序员提供的一个面向对象的 Windows 编程接口,它大大简化了 Windows 的编程工作。使用 MFC 类库的好处是:首先, MFC 提供了一个标准化的结构,这样开发人员不必从头设计创建和管理一个标准 Windows 应用程序所需的程序,而是“站在巨人肩膀上”,从一个比较高的起点编程,故节省了大量的时间;其次,它提供了大量的代码,指导用户编程时实现某些技术和功能。MFC 库充分利用了 Microsoft 开发人员多年开发 Windows 程序的经验,并可以将这些经验融入到你自己开发的应用程序中去。

MFC 相当彻底地封装了 Win32 软件开发工具包 (Software Development Kit, 即通常所说的 SDK) 中的结构、功能,它为编程者提供了一个应用程序框架,这个应用程序框架为编程者完成了很多 Windows 编程中的例行性工作,如管理窗口、菜单和对话框,执行基本的输入和输出、使用集合类来保存数据对象等等;并且, MFC 使得在程序中使用很多过去很专业、很复杂的编程课题,如 ActiveX、OLE、本地数据库和开放数据库连接性 (Open Database Connectivity, 简称为 ODBC)、Windows 套接字和 Internet 应用程序设计等,以及其他的应用程序界面特性,如属性页 (也叫标签页)、打印和打印预览、浮动的和可定制的工具栏变得更加容易。

对用户来说,用 MFC 开发的最终应用程序具有标准的、熟悉的 Windows 界面,这样的应用程序易学易用;另外,新的应用程序还能立即支持所有标准 Windows 特性,而且是用普通的、明确定义的形式。事实上,也就是在 Windows 应用程序界面基础上定义了一种新的标准——MFC 标准。

为了更好地理解 MFC,我们有必要了解一下 MFC 的发展史。

1.1.1 MFC 发展史

开始, Microsoft 建立了一个 AFX 小组, AFX 代表 Application Framework, 即应用程序框架。据说创建该小组原意是为了发布一个 Borland C++ 的 OWL (Object Windows Language) 的竞争性产品, 因为那时候 Borland 公司的应用程序框架 OWL 已经做得相当成功。AFX 小组像 OWL 那样, 提出了一个高度抽象 Windows API 的一个类库。

他们采用自顶向下的设计方法, 逐步将对象抽象出来, 并施加到 Windows 上。然后, 他们试着花了几个月的时间用这个类库来编写应用程序, 结果发现这个类库偏离 Windows API 实在太远, 过分抽象并没有太大的实用性, 相反大大降低了应用程序的效率。

于是, 他们干脆放弃了整个 AFX 类库, 对类库进行重新设计。这次, 他们采用了自底向上的方法, 从已有的 Windows API 着手, 将类建立在 Windows API 对象基础上, 设计出后来成为 MFC 1.0 的一个类库。但是, 你现在仍然可以看到 AFX 时期的痕迹, 许多源程序文件有 `afx` 前缀, 如 `afxabort.cpp`, `afxmem.cpp`。MFC 延用了许多 AFX 类库的宏, 因此我们经常会看到以 AFX 开头的宏。

AFX 小组实际上做了两件工作: MFC 类库和对 MFC 的 IDE 支持 (即资源编译器和操作向导)。在 1994 年 4 月份之后, AFX 的名字停止使用, 该小组成员成为 Visual C++ 开发组的一部分, 即现在的 MFC 小组。

MFC 1.0 版于 1992 年同 Microsoft C/C++ 同时发布。它提供了对 Windows API 简单的抽象和封装, 还没有我们现在常用的文档/视结构特性。但它引入了 COBJECT, 通过 CArchive 的持续化和其他一些 MFC 中仍然使用的特性, 从而奠定了 MFC 的基础。

MFC 2.0 在 MFC 1.0 基础上增加了文档/视结构框架、OLE 1.0 类、消息映射和公用对话框类, 废弃了 1.0 版中的 CModalDialog 类并将它的功能移入到 CDialog 中, 并增加了工具栏、对话框、分割视窗的支持。MFC 2.1 随同 Visual C++ 1.1 for NT 发布, 它把 MFC 2.0 移植到了 Win32 上。MFC 2.5 随同 Visual C++ 1.5 一起发布, 它引入了 OLE 2.0 和 ODBC 类。它是最后官方的 16 位发行版, 于 1993 年 12 月发布。目前, 在开发 16 位 Windows 程序时, Visual C++ 1.5 和 MFC 2.5 仍然有大量的用户。随后的 MFC 2.51、2.52 纠正了 MFC 2.5 中的一些错误, 增加了标签式对话框、WinSock 和 MAPI (Microsoft 电子邮件应用程序接口) 支持。MFC 3.1 同 Visual C++ 2.1 一起于 1995 年 1 月份发布, 它引入了 Windows 95 公共控件 (包括动画、热键、图像列表和工具栏提示等)。MFC 4.0 于 1995 年 12 月份同 Visual C++ 4.0 一起发布。Microsoft 直接从 Visual C++ 2.0 一下子跳过一个版本号, 升级到了 4.0, 以保持 MFC 版本号和 Visual C++ 版本号的一致性, 但这种一致性又在 Visual C++ 5.0 中打破了。在 MFC 4.0 中增加了 CSynchronize, CMutex, CEvent, CMultiLock, CShellNew 以更好地支持多线程以及 Windows 95 的其他一些特性。Visual C++ 还引入了 Component Gallery (组件画廊)、STL 支持和大量的新特性。MFC 4.1 最重要的特性是支持 Win32。许多 MFC 开发者一直都在使用该版本。MFC 4.1 修正了 4.0 的一些错误并增加了 Internet 特性。MFC 4.2 增加了 ISAPI 和 OCX 容器支持。

MFC 4.21 于 1997 年 3 月 19 日同 Visual C++ 5.0 一起发布, 它是目前最新和最完善的 MFC 版本。它只增加了对微软的 IntelliMouse (智能鼠标器) 的支持。Visual C++ 6.0 再次和 MFC 的版本同步。它包括了许多新特性: 封装了作为 IE 4.0 一部分引入的 Windows 通

用控件的新的 MFC 类；对动态的 HTML 的支持；Active Document Containment:OLE DB 与 Provider 的绑定。

MFC 发行版列表如下：

- MFC Release MSVC Release 16 位或 32 位备注
- 1.0 16 简单的封装 Windows
- 2.0 1.0 16 增加了文档/视结构
- 2.1 1.1 for NT 32 第一个 NT 的发行版
- 2.5 1.5 16 OLE/ODBC, 最后一个 16 位版本
- 2.51 2.0 16 修正错误
- 2.52 2.1 16 增加标签式对话框
- 3.0 2.0 32 标签式对话框、可停泊工具栏
- 3.1 2.1 32 Winsock/MAPI, Windows 公共控制
- 4.0 4.0 32 Win 95, 线程类, OCX 容器
- 4.1 4.1 32 sweeper (WinInet) classes

以上是最后支持 Win32s 的版本

- 4.2 4.2 32 修正错误, ISAPI classes
- 4.2b internet dl 32 修正错误
- 4.21 5.0 32 IntelliMouse™ support
- 6.0 动态 HTML;Active Document Containter
- .NET 全面基于.NET

1.1.2 MFC 类库概念和组成

类库是一个可以在应用中使用的相互关联的 C++类的集合。类库有些随编译器提供，如 Borland C++ Turbo Vision 等；有的是由其他软件公司销售，如用于数据库开发的 Code Base；有的则是由用户自己开发的。比如图像处理类库完成图像显示、格式转换和量化等；串行通信类库用于支持串行口输入输出。有些情况下用户可以直接利用类库中包含的类定义应用程序所需的变量，有时则需要从类库所提供的类中派生出新的类，这依赖于类库的设计和具体的应用程序。

Microsoft 提供了一个基础类库 MFC，其中包含用来开发 C++和 C++ Windows 应用程序的一组类。基础类库的核心是以 C++形式封装了大部分的 Windows API。类库表示窗口、对话框、设备上下文、公共 GDI 对象，例如画笔、调色板、控制框和其他标准的 Windows 部件。这些类提供了一个面向 Windows 中结构的简单的 C++成员函数的接口。

MFC 可分为两个主要部分：基础类、宏和全局函数。

MFC 基础类

MFC 中的类按功能来分可划分为以下几类：

- 基类
- 应用程序框架类

- 应用程序类
- 命令相关类
- 文档/视类
- 线程类
- 可视对象类
- 窗口类
- 视类
- 对话框类
- 属性表
- 控制类
- 菜单类
- 设备描述表
- 绘画对象类
- 通用类
- 文件
- 诊断
- 异常
- 收集
- 模板收集
- 其他支持类
- OLE2 类
- OLE 基类
- OLE 可视编辑包装程序类
- OLE 可视编辑服务器程序类
- OLE 数据传输类
- OLE 对话框类
- 其他 OLE 类
- 数据库类

宏和全局函数

若某个函数或变量不是某个类的一个成员，那么它是一个全局函数或变量。Microsoft 基本宏和全局函数提供了以下功能：

- 数据类型
- 运行时刻对象类型服务
- 诊断服务
- 异常处理
- CString 格式化及信息框显示
- 消息映射

- 应用消息和管理
- 对象连接和嵌入 (OLE) 服务
- 标准命令和 Windows IDs

注意：全局函数以“Afx”为前缀，所有全局变量都是以“Afx”为前缀，宏不带任何特别前缀，但是全部大写。

常见的全局函数和宏有：AfxGetApp，AfxGetMainWnd，AfxMessageBox 和 DEBUG_NEW 等，我们会在后面的章节中用到并对它们进行介绍。

从继承关系来看，又可将 MFC 中的类分成两大类：大多数的 MFC 类是从 CObject 继承下来；另外一些类则不是从 CObject 类继承下来，这些类包括：CString（字符串类），CTime（日期时间类），CRect（矩形类）和 CPoint（点）等，它们提供程序辅助功能。

由于 MFC 中大部分类是从 CObject 继承下来的，CObject 类描述了几乎所有的 MFC 中其他类的一些公共特性，因此我们有必要理解 CObject 类。

我们首先查看一下 CObject 类的定义，CObject 类的定义如下所示：

```
// class CObject is the root of all compliant objects
class CObject
{
public:
// Object model (types, destruction, allocation)
virtual CRuntimeClass* GetRuntimeClass() const;
virtual ~CObject(); // virtual destructors are necessary
// Diagnostic allocations
void* PASCAL operator new(size_t nSize);
void* PASCAL operator new(size_t, void* p);
void PASCAL operator delete(void* p);

#ifdef _DEBUG && !defined(_AFX_NO_DEBUG_CRT)
// for file name/line number tracking using DEBUG_NEW
void* PASCAL operator new(size_t nSize, LPCSTR lpszFileName, int nLine);
#endif

// Disable the copy constructor and assignment by default so you will get
// compiler errors instead of unexpected behaviour if you pass objects
// by value or assign objects.
protected:
CObject();
private:
CObject(const CObject& objectSrc); // no implementation
void operator=(const CObject& objectSrc); // no implementation
```

```

// Attributes
public:
BOOL IsSerializable() const;
BOOL IsKindOf(const CRuntimeClass* pClass) const;
// Overridables
virtual void Serialize(CArchive& ar);
// Diagnostic Support
virtual void AssertValid() const;
virtual void Dump(CDumpContext& dc) const;
// Implementation
public:
static const AFX_DATA CRuntimeClass classCObject;
#ifdef _AFXDLL
static CRuntimeClass* PASCAL _GetBaseClass();
#endif
};

```

CObject 类为派生类提供了下述服务：

对象诊断

MFC 提供了许多诊断特性，它可以：

输出对象内部信息；**CDumpContext** 类与 **CObject** 的成员函数 **Dump** 配合，用于在调试程序时输出对象的内部数据。

对象有效性检查

重载基类的 **AssertValid** 成员函数，可以为派生类的对象提供有效性检查。

运行时访问类的信息

MFC 提供了一个非常有用的特性，它可以进行运行时的类型检查。如果从 **CObject** 派生出一个类，并使用了以下 3 个宏 (**IMPLEMENT_DYNAMIC**, **IMPLEMENT_DYNCREATE** 或 **IMPLEMENT_SERIAL**) 之一，就可以：

运行时访问类名

安全可靠地把通用的 **CObject** 指针转化为派生类的指针。比如，我们定义一个主窗口类：

```

CMyFrame:public CFrameWnd
{
.....
}

```

然后我们使用这个类：

```
CMyFrame *pFrame=(CMyFrame*)AfxGetMainWnd();
pFrame->DoSomeOperation();
```

AfxGetMainWnd 是一个全局函数，返回指向应用程序主窗口的指针，类型为 CWnd*。因此我们必须对它进行强制类型转换，但我们如何知道是否转换成功了呢？我们可以使用 CObject 的 IsKindOf()成员函数检查 pFrame 的类型，用法如下：

```
ASSERT(pFrame->IsKindOf(RUN_TIMECLASS(CMyFrame)));
```

将上一语句插入到 pFrame-> DoSomeOperation()之前，就可以在运行时作类型检查，当类型检查失败时，引发一个断言 (ASSERT)，中断程序执行。

对象持续性

通过与非 CObject 派生的档案类 CArchive 相结合，提供将多个不同对象以二进制形式保存到磁盘文件 (Serialization) 中以及根据磁盘文件中的对象状态数据在内存中重建对象 (Deserialization) 的功能。

然而，MFC 不仅仅是一个类库，它还提供了一层建立在 Windows API 的 C++封装上的附加应用程序框架。该框架提供了 Windows 程序需要的多数公共用户界面。

注意：所谓应用程序框架指的是为了生成一般的应用所必须的各种组件的集成。应用框架是类库的一种超集。一般的类库只是一种可以用来嵌入任何程序中的、提供某些特定功能（如图像处理、串行通信）的孤立的类的集合，但应用框架却定义了应用程序的结构，它的类既相互独立，又相互依赖，形成一个统一的整体，可以用来构造大多数应用程序。中国用户熟悉的 Borland C++ 的 DOS 下的 Turbo Vision 和 Windows 下的 OWL (Object Windows Language) 都是应用框架的例子。

下面我们举个具体的例子来说明 MFC 所提供的应用程序框架，程序如下：

```
#include<afxwin.h>
//derived an application class
class CMinMFCApp:public CWinApp
{
public:
    BOOL InitInstance();
};

//Derive the main window class
class CMainWindow:public CFrameWnd
{
public:
    CMainWindow();
```

```
DECLARE_MESSAGE_MAP()
};
BEGIN_MESSAGE_MAP(CMainWindow, CFrameWnd)
END_MESSAGE_MAP()

/*CMinMFCApp Member Functions*/
BOOL CMinMFCApp::InitInstance()
{
    m_pMainWnd=new CMainWindow();
    m_pMainWnd->ShowWindow(m_nCmdShow);
    m_pMainWnd->UpdateWindow();
    return TRUE;
}

/*CMainWindow member functions*/
CMainWindow::CMainWindow()//constructor
{
    Create(NULL,
    "Min MFC Application",
    WS_OVERLAPPEDWINDOW,
    rectDefault,
    NULL,
    NULL);
}

/*an instance of type CMinMFCApp*/
CMinMFCApp ThisApp;
```

程序段定义了一个最小的 MFC 应用程序所需的框架程序。其中声明了 CMinMFCApp 类，它是从应用程序类 CWinApp 中派生出来的；而窗口 CMainWindow 类，是从框架窗口 CFrameWnd 类派生出来的。我们还用 CMinMFCApp 定义了一个全局对象 ThisApp。读者也许会问，为什么没有 WinMain 函数？因为 MFC 已经把它封装起来了。在程序运行时，MFC 应用程序首先调用由框架提供的标准的 WinMain 函数。在 WinMain 函数中，首先初始化由 CMinMFCApp 定义的惟一的实例，然后调用 CMinMFCApp 继承 CWinApp 的 Run 成员函数，进入消息循环。退出时调用 CWinApp 的 ExitInstance 函数。

从上面的说明可以看出，应用程序框架不仅提供了构建应用程序所需要的类（CWinApp, CFrameWnd 等），还定义了程序的基本执行结构。所有的应用程序都在这个基本结构的基础上完成不同的功能。

MFC 除了定义程序执行的结构之外，还定义了 3 种基本的主窗口模型：单文档窗口，多文档窗口和对话框作为主窗口。

注意：Visual C++提供了两个重要的工具，用于支持应用程序框架，它们就是前面提到 Application Wizard 和 ClassWizard。Application Wizard 用于在应用程序框架基础上迅速生成用户的应用程序基本结构。ClassWizard 用于维护这种应用程序结构。

1.1.3 MFC 的优点

Microsoft MFC 具有以下不同于其他类库的优势：

- 完全支持 Windows 所有的函数、控件、消息、GDI 基本图形函数、菜单及对话框。类的设计以及同 API 函数的结合相当合理。
- 使用与传统的 Windows API 同样的命名规则，即匈牙利命名法。
- 进行消息处理时，不使用易产生错误的 switch/case 语句，所有消息映射到类的成员函数，这种直接消息到方法的映射对所有的消息都适用。它通过宏来实现消息到成员函数的映射，而且这些函数不必是虚拟的成员函数，这样不需要为消息映射函数生成一个很大的虚拟函数表（V 表），节省内存。
- 通过发送有关对象信息到文件的能力提供更好的判定支持，也可确认成员变量。
- 支持异常错误的处理，减少了程序出错的机会。
- 运行时确定数据对象的类型，这允许实例化时动态操作各域。
- 有较少的代码和较快的速度。MFC 库只增加了少于 40kB 的目标代码，效率只比传统的 C Windows 程序低 5%。
- 可以利用与 MFC 紧密结合的 Application Wizard 和 ClassWizard 等工具快速开发出功能强大的应用程序。

另外，在使用 MFC 时还允许混合使用传统的函数调用。

1.2 Visual C++ .NET 与老版本的区别

1. 工程种类更为丰富

新建的工程分为三大类：.NET projects, Win32 projects 和 Utility projects。

在 .NET projects 工程中新加了 Managed C++ Application, Managed C++ class library, Managed C++ empty project 和 Managed C++ Web Service。在 Win32 projects 工程中新加了 ATL project, ATL server project, ATL Server Web service project, MFC Application 和 MFC DLL。在 Utility projects 工程中增加了 Custom Application Wizard 和 Makefile project。

2. 编译器和链接器的性能得到提高

使用/OI 选项编译生成的代码大小将比 Visual C++ 6 版小 5%到 10%。

链接器使用了/FIXED 选项来创建更小的供发行的应用程序。因此，在使用剖析程序时，由于需要重定位信息，链接器必须使用/PROFILE 和/FIXED:NO 选项。这同样适用于其他

docsriver 文川网
入驻商家 古籍书城

在文川网搜索古籍书城 获取更多电子书

如 Bounds Checker 或 Purify 之类的链接后 (post-link) 工具。

新增的/EH 编译选项可以更有效的控制 C++异常处理。C++同步异常处理允许编译器生成更小的代码, 因此它是 Visual C++.NET 新的默认 C++异常处理模式。

对用来控制代码优化所面向处理器的编译器选项/G3, /G4, /G5, /G6 和/GB 作了修改。

将/GX 编译器选项映射为/EHsc。允许使用链接器选项/PDBTYPE 指定包括调试信息的程序数据库 (PDB)。该选项可以节省磁盘空间并加快链接。

在 NMAKE 中支持批处理规则。

3. 增加了部分类的功能

可以在 Web 浏览器或支持 ActiveX 文档的 OLE 容器 (例如 Microsoft Office Binder) 的整个客户区显示活动的文档。Win32 Internet API (WinInet)使 Internet 成为任意应用程序的一个完整部分并简化了 Internet 服务, 如 FTP, HTTP 和 Gopher 的访问。


增加了对 DAO 的支持。增加了对 ODBC 的支持, 并对 MFC ODBC 类作了几个重要的修改。

COleDateTime 成员函数 SetDate, SetDateTime 和 SetTime 的返回值从 BOOL 改变为 int。每一个成员函数当 COleDateTime 对象被正确设置时返回 0, 否则返回 1。该返回值基于 DateTimeStatus 枚举类型。

新增示例程序 IMAGE。该程序生成一个可以异步下载数据的 ActiveX 控件。ATL 新版支持创建既小又快的 ActiveX 控件。

改善了下列通用浮点超越函数的性能: pow, sqrt, log, log10, sin, cos, tan, asin, acos 和 atan。

改善了内存移动和内存拷贝函数的性能。

 **注意:** OLE DB 是一组 OLE 接口, 它使应用程序可以以统一的方式访问保存在不同信息源中的数据。这些接口支持适合于数据源的大量数据库功能, 并允许数据源共享其数据。所配套的 OLE DB 软件开发工具包所提供的一组软件部件、工具和文档可以在开发 OLE DB 客户和提供程序提供帮助。

4. 集成开发环境的功能更为强大

新的 Application Wizard 可以自动管理基于对话框的应用程序中的对话框类。只需要简单的创建一个基于对话框的应用程序, 并选择对自动化的支持, 就可以像早期版本的 AppWizard 一样, 得到一个支持基本自动化的基于对话框的应用程序。通过单独的代理类, 对话框类也可以通过自动化导出。你可以添加方法和属性来导出对话框中的元素。

定制的 Application Wizard 可以改变工程创建时的设置。例如, 你可以在目标创建之后调整编译器、链接器和查看设定或者添加定制的建立步骤。

asynchronous (URL) moniker 允许应用程序异步地下载文件和控件属性, 以便在任务完成后为其他进程释放系统资源。

ERRLOOK 工具可以使用系统错误的值来检索相应的错误消息, 其中包括 OLE HRESULT。错误值可以通过包括拖放、编辑命令等的多种方法给出。由 ERRLOOK 所返

回的错误消息文本可以复制并粘贴到其他应用程序中。

在 Visual C++ 中, 可以使用 WizardBar 来将代码与程序中的可视元素挂钩。快捷键、二进制、对话框和字符串编辑器支持定位至快捷键、ASCII 字符串、三十二进制字符串、控件 ID 和标签及指定字符串的 Find 命令能更方便的一次修改多个项(可以为快捷键、对话框、菜单和字符串)的属性。

改进了的 WizardBar 采用浮动条显示, 大大方便了编程人员。新增的向导添加了集成到 Visual C# 和 Visual InterDev 中的新增向导。

工具栏采用浮动显示, 新增了 Solution 方案。采用了工作区右置, 更加符合视觉效果。

1.3 Visual Studio.NET 新特性

.NET 的平台及框架是基于微软软件工业基础的又一次升级和演化。然而, .NET 还是要尽力保证 Windows 系统及系列产品和 .NET 能够融为一体, 尽量在微软公司原有的软件资产基础上, 使 .NET 继续成为 Internet 的中心。

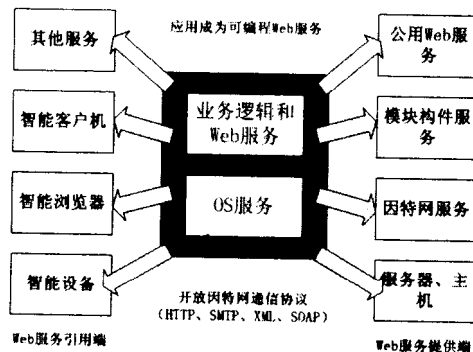


图 1.1 在 .NET 体系结构中, XML 是各应用之间无缝接合的关键

1.3.1 .NET 的创新

.NET 主要组成部分及其作用介绍如下:

1. Windows.NET

Windows.NET 是融入 .NET 技术的 Windows, 它将紧密地整合了 .NET 的一系列核心构造模块, 为数字媒体及应用间协同工作提供支持, 是微软公司的下一代 Windows 桌面平台。

2. .NET Framework

.NET Framework 的目的是便于开发商更容易地建立网络应用程序和 Web 服务, 它的关键特色是提供一个多语言组件开发和执行的环境。从层次结构来看 .NET Framework 又包括 3 个主要组成部分: 通用语言运行环境 (Common Language Runtime)、服务框架 (Services Framework)、上层的两类应用模板——面向 Web 的网络应用程序模板 (Web Forms 或 Web

Services) 和 Windows 应用程序模板 (Win Forms)。

其中通用语言运行环境在组件运行时, 负责管理内存分配、启动和中止线程和进程、强化安全系数, 同时还调整任何该组件涉及到的其他组件的附件配置。在通用语言运行环境上是服务框架, 它为开发人员提供了一套能够被任何现代编程语言调用的、统一的面向对象、异步、层次结构的可扩展类库, 包括集合、输入/输出、字符串、图画、网络、线程、全球化、安全加密、数据库访问、调试相关服务等类库。在服务框架之上是两种应用类型的模板, 一类是传统的 Windows 应用程序模板, 另一类是基于 ASP+ 的 Web 网络应用程序模板。其中 ASP+ 以一组控件和体系结构方式提供了一个 Web 应用模型, 由 .NET 框架提供的类库构建而成, 通过它可以简化 Web 应用的实现过程。

3. .NET 企业服务器

在微软宣称的“第三代互联网”中, .NET 企业服务器是企业集成和管理所有基于 Web 的各种应用的基础, 它提供企业未来开展电子商务的高可靠性、高性能、高可伸缩性以及高可管理性。.NET 企业服务器的构成异常庞大而复杂, 它共包括 8 个各司其职的服务器, 如下所示。

.NET 企业服务器功能描述:

- Application center 2000 部署和管理基于 Windows 2000 之上的 Web 应用;
- BizTalk Server 2000 用于企业间交换商务信息;
- Commerce Server 2000 用于快速创建在线电子商务;
- Exchange 2000 提供基于 Windows 2000 的通信和协作功能;
- Host Integration Server 2000 为主机系统的组件集成提供方便;
- Internet Security & Acceleration Server 2000 主要解决企业应用安全性和可管理性的问题;
- Mobile Information 2001 Server 为移动解决方案提供可靠而具伸缩性的平台;
- SQL Server 2000 则提供完全的数据库和数据分析解决方案。

其中, 目前被关注最多的是 BizTalk Server 2000, 它提供了企业间进行电子商务所需的自动、集成的业务处理工具和架构。BizTalk Server 2000 利用 XML 作为企业内部及企业间文档传输的数据格式, 可以屏蔽平台、操作系统不同的差异, 使商业系统的集成成为可能。它主要完成的功能有: 企业间可靠的文档交换、ERP 应用集成、商业流程自动化定制、管理和监控等等。

4. 模块构建服务

模块构建服务 (Building Block Services) 是 .NET 平台中的核心网络服务集合, 它主要包括以下几个组成部分: Internet XML 通信, 使 Web 站点变成灵活的服务来交换和处理数据; Internet XML 数据空间, 在 Web 商提供安全的和可编程的 XML 存储空间; Internet 动态更新, 为快速开发和动态配置应用提供服务。

Internet 日程安排, 集成工作、社会和私人的日历; Internet 身份认证, 提供从口令、钱包到生理数据等多级身份认证手段, 还有 Internet 目录服务和 Internet 即时信息传递等服务。

5. Orchestration

Orchestration 是一种基于 XML 的面向应用的软件集成和自动化处理技术。它的目标是尽量不受时间、组织、应用及个人的限制，来最大程度和最好地把集成技术和自动处理技术接合起来，以便商业事务能够交互、动态和可靠地进行下去。

注意：Orchestration 有 3 个基本要求：处理与执行过程分离，即整个处理并不一定非要同执行的细节及途径绑定起来；动态处理，即随着数据不同及交换的变化，整个操作过程必须随时动态更新改变；Any to Any 集成，即整个处理过程不能对参与的平台、应用及协议等作出限制。.NET 的 BizTalk Orchestration 是上述技术的一个实现，它包括一个可视化的设计环境、一套捆绑的工具和一个 Orchestration 引擎，用于业务流程处理、管理和调试。

6. Visual Studio.NET

Visual Studio.NET 是基于 XML 的编程工具和环境，它便于快速开发符合 .NET 体系的软件服务，使其在独立设备、企业数据中心和因特网之间的传送更加容易。

1.3.2 .NET 与同类产品的比较

针对 .NET 计划的推出，业界早已有多种说法，其中一个最重要的观点是 .NET 并没有发明什么先进的技术，而只是多家思想的拷贝和整合。从某种角度来看，这的确不无道理。下面我们就 SUN 的 Java&&J2EE 体系结构和 .NET 的框架作一简单的对比。

我们可以看出，对于 .NET 框架的每一部分，Java&&J2EE 体系结构多少都有所表现，此外，.NET 计划中的很多产品也能从其竞争对手，如 SUN，ORACLE，IBM 和 HP 的产品中发现一些相似痕迹。因此的确不能说 .NET 首创了多少多少创新，更合适的说法是它对很多公司的思想和自己的工业基础作了较好的整合，并有所创新。

虽然如此，但微软的 .NET 计划及思想还是为整个信息产业和未来软件世界的发展提供了一个可供分析参考的标准，因为技术本身很多是相互融和和渗透的，所有技术争论并不能决定未来，只有通过市场才能最后决定什么是真正的互联网未来。

1.3.3 .NET 对 IT 专业人员的重要意义

目前，IT 专业人员能够利用与构建 .NET 平台相同的技术。.NET Enterprise Servers 和 Windows®2000 操作系统，为创建具有高度可管理性的、能迅速投入市场的应用程序提供了坚实基础。它们利用的是可扩展标记语言 (XML)，因此随着 Web 体系结构的革新，在此平台上创建的程序依然很有价值。

注意：.NET 平台的核心是，采用有效的、分门别类的方式来构建应用程序，达到其前所未有的规模。该平台上的 Web 服务模型指的是：企业应用程序的中心业务要素通常由本地管理，而支持它们的服务（如用户认证、文件存储、用户首选项管理、日历、邮件等等）却无须本地管理，可以被无缝订购。为了存储用户文件和邮件，IT 专业人员往往在服务器上安装新的独立磁盘冗余阵列 (RAID 阵

列), 而有了 .NET, 他们在这一方面将会花费较少的精力, 而更多地致力于怎样为公司增加效益。

.NET 的最终目的就是让用户在任何地方、任何时间, 以及利用任何设备都能访问他们所需要的信息、文件和程序。而用户不需要知道这些东西存在什么地方, 甚至连如何获得等具体细节都不知道。他们只需发出请求, 然后只管接收就是了, 而所有后台的复杂性是完全屏蔽起来的。所以对于企业的 IT 人员来说, 他们也不需要管理复杂的平台以及各种分布应用之间的工作是如何协调的。

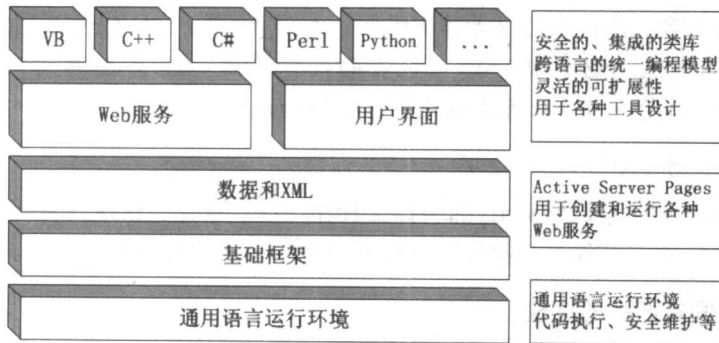


图 1.2 .NET Framework 包括通用语言运行环境、Framework 类库和 Active Server Pages+

可见, 与以前的版本相比, VC++.NET 提供了更为强大的功能, 这些在后续章节中会作更为详尽的介绍。

1.4 本章小结

本章主要介绍了 MFC 类库的发展历史、基本概念及其组成, MFC 类库是 Visual C++ 的核心, 也是本书的重点, MFC 类的功能强大, 结构精巧, 可以极大的简化编程工作和提高开发效率, 所以建议读者一定要熟悉和掌握其基本结构及其基本类的使用方法, Visual Studio.NET 集成开发环境为编程开发工作提供了更为强大、快捷方便的开发工具, 熟练掌握其使用方法可以极大地提高编程效率, 在下一章中将详细介绍其使用方法, 本章的要点为:

- 了解 MFC 类库各个版本的差别及发展历史;
- 掌握 MFC 类库的基本概念;
- 了解 Visual C++.NET 的新增功能及其集成开发环境的使用方法。

第 2 章 熟悉 Visual C++.NET 的开发环境

全新的 Developer Studio.Net, VC++.NET 由于采用了可视化编程的概念使得开发环境变化很大, 界面做得更友好了, 使用更方便了。既保留了以前 Visual C++ 6.0 的优秀特性, 又新增了很多功能; 在界面上集合了以前 Visual Studio 的各种语言的开发特征。本章着重介绍了 VC++.NET 最新的界面开发环境, 逐步深入, 使您对于全新的 VC++.NET 有更深入的了解。本章是用 VC++.NET 开发 Windows 程序的基础。

2.1 Visual C++.NET 可视化集成开发环境

Visual C++ .NET 提供了一个不同于以往版本的支持可视化编程的集成开发环境: Visual Studio.Net (又名 Developer Studio.Net)。Developer Studio.Net 是一个通用的应用程序集成开发环境, 它不仅支持 Visual C++, 还支持 Visual Basic, Visual InterDev 和 Visual Java 等 Microsoft 系列开发工具。Developer Studio.Net 包含了一个文本编辑器、资源编辑器、工程编译工具、一个增量连接器、源代码浏览器、集成调试工具, 以及一套联机文档。使用 Developer Studio.Net, 可以完成创建、调试、修改应用程序等的各种操作。

Developer Studio.Net 采用标准的多窗口 Windows 用户界面, 与 Visual Studio 6.0 相比又增加了一些新特性, 使得开发环境更易于使用, 用户能很容易学会它的使用方法。一个典型的 Developer Studio.Net 用户界面如图 2.1 所示。

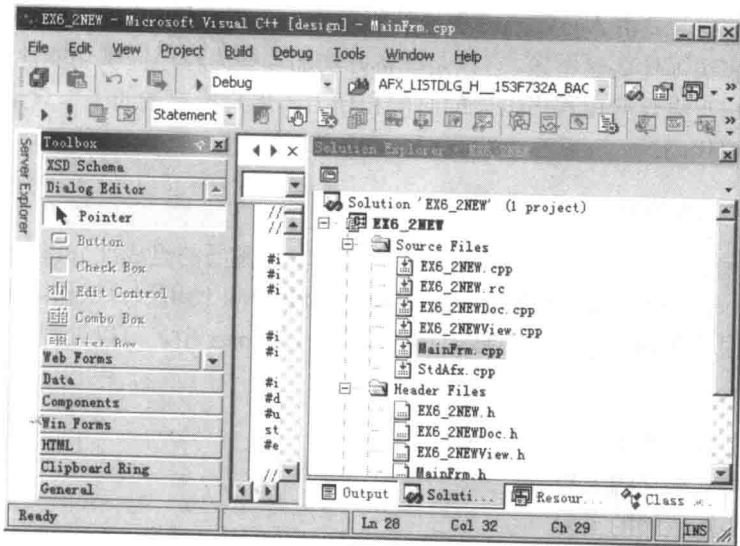



图 2.1 Developer Studio.Net 用户界面

由于 Developer Studio.Net 是一个可视化的开发工具,在介绍 Developer Studio.Net 的各个组成部分之前,首先了解一下可视化编程的概念。可视化技术是当前发展迅速并引人注目的技术之一,它的特点是把原来抽象的数字、表格、功能逻辑等用直观的图形、图像的形式表现出来。可视化编程是它的重要应用之一。所谓可视化编程,就是指:在软件开发过程中,用直观的具有一定含义的图标按钮、图形化的对象取代原来手工的抽象的编辑、运行和浏览操作,软件开发过程表现为鼠标单击按钮和拖放图形化的对象以及指定对象的属性、行为的过程。这种可视化的编程方法易学易用,而且大大提高了工作效率。

Visual C++.NET 的集成开发环境 Developer Studio.Net 提供了大量的实用工具以支持可视化编程特性,它们包括: Solution Explorer, ClassWizard, Application Wizard, Property Windows 和 Component Gallery 等。下面我们将对它们作逐一介绍。

2.1.1 Application Wizard(应用程序向导)

 **注意:** Application Wizard 同 Visual Studio 6.0 中的 AppWizard 作用相同,是 Visual C++ 提供的一个高级编程工具,它可以产生应用的 C++ 源代码框架。在 Visual Studio 6.0 中,AppWizard 通过与另一个工具 ClassWizard 一起配合使用,可大大节省开发应用程序的时间和精力。而在 Visual Studio.Net 中,ClassWizard 的作用基本上是通过 Properties 窗口来完成的。

Application Wizard 是一个标准的 C++ 源代码生成器。它通过一系列的对话框来提示用户输入所需创建的程序的信息,如它的名字和位置。用户还可以指定它是否具有一些特性,如多文档接口或工具栏,对数据库、OLE 的支持等。然后 AppWizard 生成一些文件,这些文件构成程序的框架。由 AppWizard 生成的程序是一个基本的 Windows 程序,用户可以编译并运行——它实际什么也不做。它只是准备好增加那些为程序提供功能性的资源和代码。这样就节省了用户设计应用程序框架的时间和精力,用户所要做的工作只是直接往框架中添加自己的处理代码。

2.1.2 Solution Explorer

Solution Explorer 是 Developer Studio.Net 的一个最重要的组成部分,程序员的大部分工作都在 Developer Studio.Net 中完成。Developer Studio.Net 使用项目工作区来组织项目、元素以及项目信息在屏幕上出现的方式。在 Solution Explorer 中,可以处理:

- 一个工程和它所包含的文件;
- 一个工程的子工程;
- 多个相互独立的工程;
- 多个相互依赖的工程。

Solution Explorer 可包含由不同的开发工具包生成的工程,如 Visual C++ 和 Visual C#。在桌面上, Solution Explorer 以窗口方式组织项目、文件和项目设置。Solution Explorer 窗口一般位于屏幕右侧,如图 2.2 所示。Developer Studio.Net 提供了一组视图,用于从不同

的角度（视图）查看项目中包含的工程和联机文档。

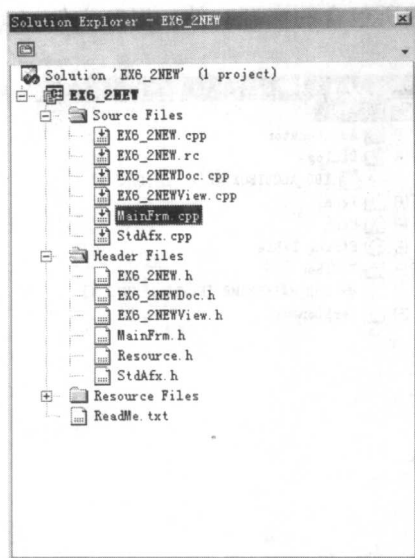


图 2.2 Solution Explorer 窗口

每个项目视图都有一个相应的文件夹，包含了关于该项目的各种元素。展开该文件夹可以显示该视图方式下工作区的详细信息，Solution Explorer 有 5 种辅助视图。

ClassView(类视图)

如图 2.3 所示显示项目中定义的 C++ 类，展开文件夹显示工程中所定义的所有类，展开类可查看类的数据成员和成员函数以及全局变量、函数和类型定义。

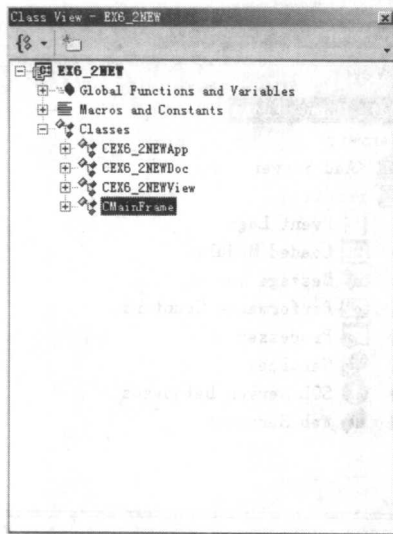


图 2.3 ClassView(类视图)

ResourceView(资源视图)

如图 2.4 所示显示项目中所包含的资源文件。展开文件夹可显示所有的资源类型。

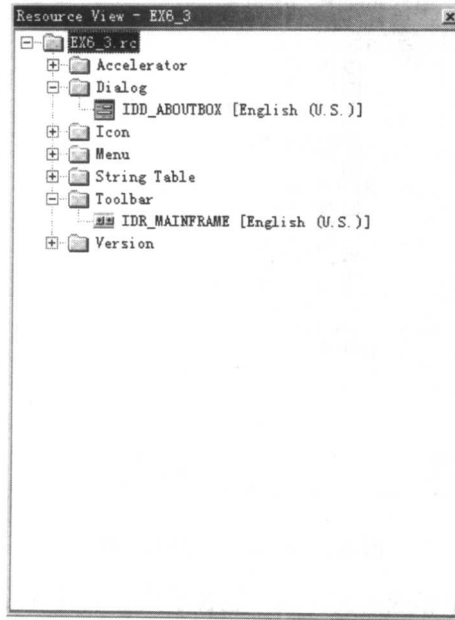


图 2.4 ResourceView(资源视图)

Server Explorer(服务器资源管理器)

如图 2.5 所示，用于管理基于服务器组件开发应用的工程项目的组件资源。

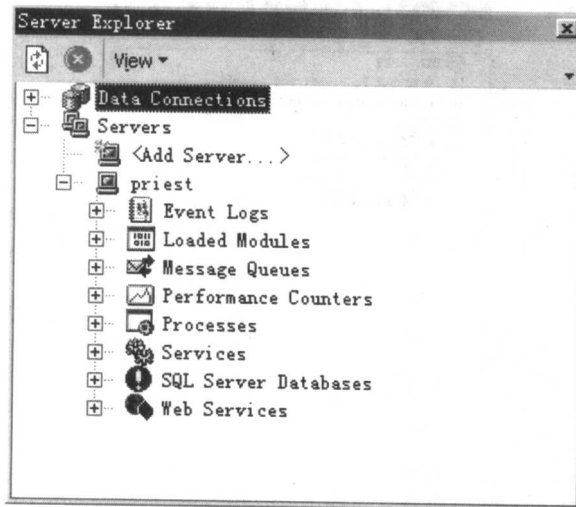


图 2.5 Server Explorer(服务器资源管理器)

Properties Window(属性窗口)

Properties 窗口如图 2.6 所示, 取代了 Visual Studio 6.0 中的 Class Wizard, 用来组织与类相关的一切资源和属性等等, 详细用法将在后面介绍。

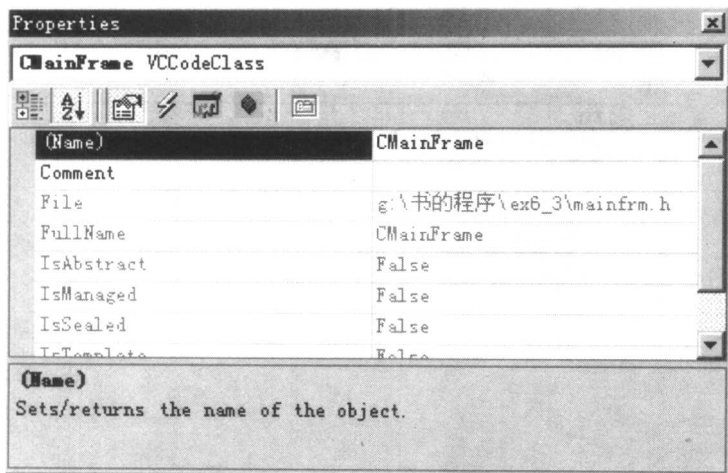


图 2.6 Properties Window(属性窗口)

Web Sever

Web Sever 页如图 2.7 所示, 用于组织工程中的 Web 页资源。

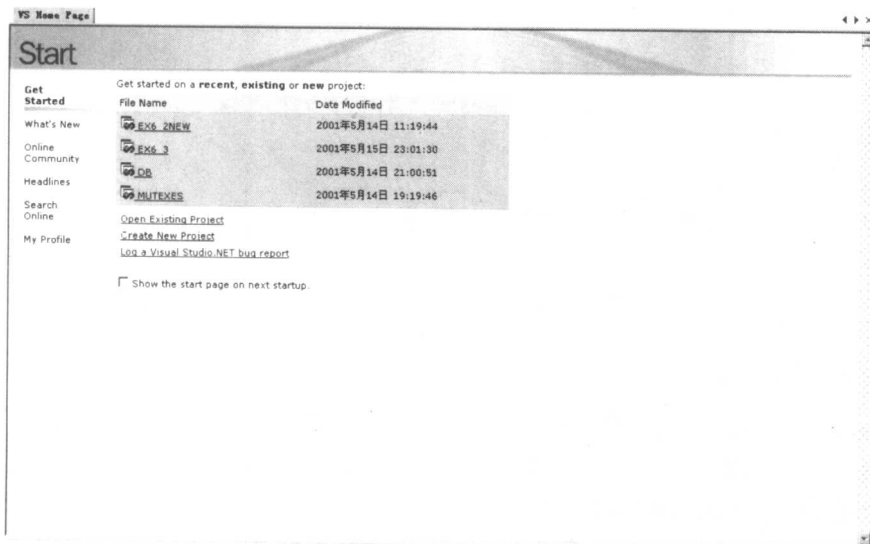


图 2.7 Web Sever 页

可以将所有的视图放在一起, 单击底部的标签可以从一个视图切换到另一个视图, 如图 2.8 所示。

每个视图都是按层次方式组织的。可以展开文件夹和其中的项来查看其内容，或折叠起来查看其组织结构。在项目视图中，如果一项不可以再展开，那么它是可编辑的。双击这一项便可以打开相应的文档编辑器进行编辑。对类和源程序文件来说，是打开文本编辑器，对于对话框来说是打开对话框编辑器。每个视图还支持右键快捷菜单。

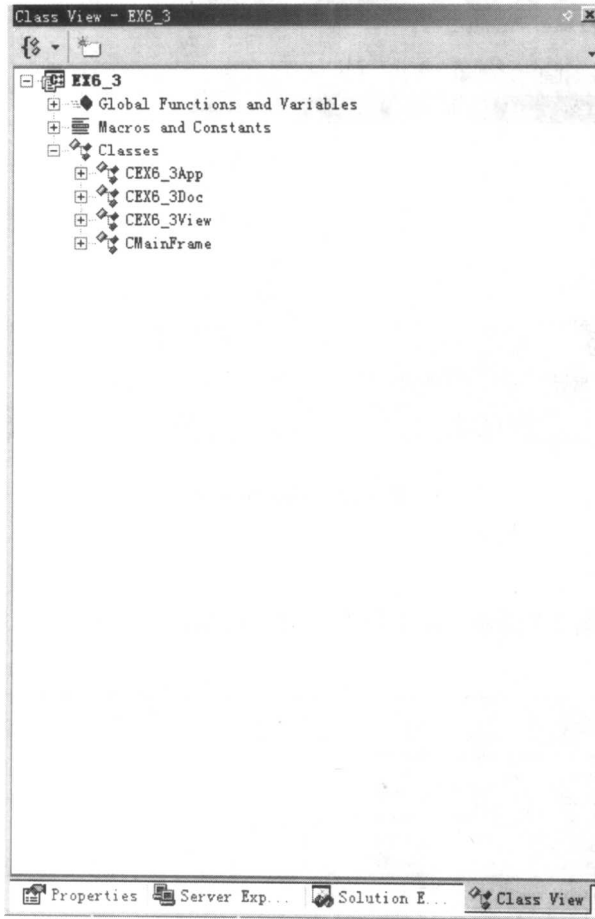


图 2.8 视图集

使用 Solution Explorer

Solution Explorer 中显示了工程文件所包含的文件的逻辑关系。一个 Solution 可以包含多个工程，其中活动工程以黑体显示。活动配置决定了编译活动工程时的编译选项。活动工程是使用 Build 或 Rebuild All 时要编译的那一个工程。可以用 Build 菜单上的 Set Active Configuration 选择不同的活动配置；也可以在 Project 菜单上用 Set Active Project 选择不同的活动工程。

使用 Solution Explorer 可以：

- 查看文件；
- 管理文件，包括增加、删除、移动、重命名和拷贝文件等。

要增加一个文件到过程中，可以选择 Project->Add to Project->Files 菜单，则弹出 File 对话框，选择相应文件即可；要从工程中删除一个文件，可打开工程文件夹，选择相应文件，然后按 DEL 键。

使用类视图 (ClassView)

ClassView 显示所有已定义的类以及这些类中的数据成员和成员变量。Visual C++.NET 自动从 Solution Explorer 中所包含的源程序文件中分离出类。

在 ClassView 中，文件夹代表工程文件名。展开 ClassView 顶层的文件夹后，显示工程中所包含的所有类，如图 2.9 所示。双击一个类的图标（或单击图标旁的+号）时，ClassView 展开该类并显示其类成员。

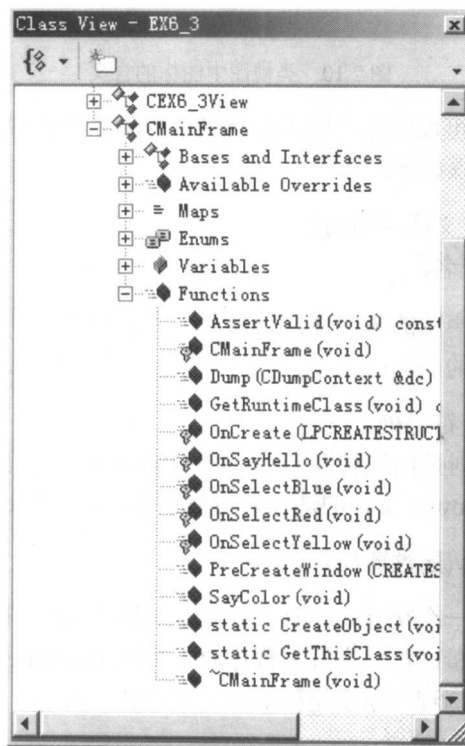


图 2.9 类视图

ClassView 使用图标标识类、类成员和工程中的其他项，图 2.10 显示了所有的图标和含义。

docsriver 文川网
入驻商家 古籍书城

在文川网搜索古籍书城 获取更多电子书

图标	含义
	类
	保护类成员函数
	私有类成员函数
	公有类成员函数
	保护类数据成员
	私有类数据成员
	公有类数据成员

图 2.10 类视图中图标的含义

利用 ClassView 不仅可以浏览应用程序所包含的类以及类中的成员，还可以快速跳到一个类或成员的定义，步骤如下：

- (1) 选择要查找的定义或声明的符号。
- (2) 双击所选的符号名。

要打开关于某一个类声明的头文件，只需双击类名即可。

要想查找某一个变量的参考：

- (1) 将光标定位在该符号处。
- (2) 按鼠标右键，则弹出快捷菜单，选择 Reference 选项(此时应确保已经生成了 Browse(浏览)文件,关于 Browse 文件的选项，可以在 Project→Settings 中设置)。

2.1.3 Properties Window(属性窗口)

Properties Window 是一个交互式工具，用来定制类的属性，把消息映射成类成员函数，或者把控制框映射为类变量成员。在开发程序时，可用 Properties Window 控制类的属性，包括消息处理和消息映射例程（用于定位处理消息的代码）。

使用 Properties Window 可以：

- 映射消息到函数；
- 新建、删除消息处理函数；
- 查看已被处理的消息并跳到消息处理代码处；
- 创建新类时，自动加入方法和属性；
- 处理现有的类和类库。

在后续章节中，我们将举例说明 AppWizard 和 ClassWizard 这两个工具的用法。

2.1.4 Developer Studio.Net 的一些快捷特性

为了使开发环境更易于使用，Developer Studio.Net 还提供了一些快捷特性，包括：快

速访问常用对话框、快捷菜单、属性对话框和键盘快捷键等。

快速访问常用对话框

在列表出现的窗口中,通常可以通过双击列表中的一项显示与该项相关的常用对话框。比如,要编辑字符串列表中的一项,可以双击它,弹出字符串编辑对话框,用于编辑双击的那一项。

右键菜单(快捷菜单)

在 Developer Studio.Net 的许多窗口中,可以在一个选中的对象或窗口背景上单击鼠标右键弹出快捷菜单。快捷菜单包含了与当前区域或所选项相关的一组常用命令。大多数命令都可以从菜单栏上访问,但用快捷菜单更加方便。

属性对话框

选择一个项,单击右键,从快捷菜单上选择 **Properties** 选项,就显示出关于该对象的属性对话框。属性对话框显示在一个属性页中,可能包含许多标签。弹出属性对话框的快捷键是 Alt+Enter。如果有可编辑的属性,就可以在属性页中直接修改。图 2.11 是修改某个编辑框属性的一个例子。

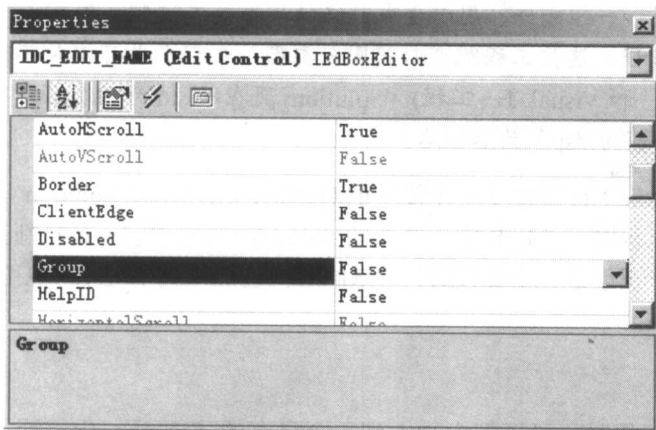


图 2.11 Properties 对话框

键盘快捷键

键盘快捷键为那些喜欢键盘而不喜欢用鼠标的用户提供了执行某一功能的替代方式。Developer Studio.Net 为一些常用的命令分配了缺省的快捷键。用户可以自定义快捷键。

为了高效地使用 Developer Studio.Net,建议读者记住以下一些常用的快捷键:

- 打开文件 (File→Open): CTRL+O, 表示打开一个已有的文件;
- 保存文件 (File→Save): CTRL+S, 表示保存活动文档;
- 编辑复制 (Edit→Copy): CTRL+C;
- 编辑剪切 (Edit→Cut): CTRL+T;