工程应用与项目实践丛书

Visual C++

工程应用与项目实践

赛奎春 主编
张 雨 阮伟良 李 贺 等编著

附光盘

机械工业出版社
CHINA MACHINE PRESS

工程应用与项目实践丛书

Visual C++工程应用与项目实践

赛奎春 主编

张 雨 阮伟良 李 贺 等编著

机械工业出版社

本书从项目开发必备的知识和原则入手，全面系统地介绍了 Visual C++在工程开发中的编程知识、方法和技巧。全书共分 10 章，内容包括：MFC 编程基础；可视化界面设计；文件与文件系统在程序开发中的应用；图形与多媒体技术在开发中的应用；注册表、操作系统编程；输入/输出技术；Visual C++高级应用；数据库程序设计；网络与 Internet 程序设计；物流综合管理系统等。全书注重所讲知识的工程应用，读者在掌握 Visual C++软件的同时，能够快速掌握工程项目开发的思路、方法和经验，并轻松解决项目开发中的出现的问题。

本书配套光盘提供了书中所有实例的源代码。

本书注重工程实践，实用性强，是各级程序开发人员不可多得的参考书，也非常适合大中专院校师生学习参考。

图书在版编目（CIP）数据

Visual C++工程应用与项目实践 / 赛奎春主编. —北京：机械工业出版社，2005.1

（工程应用与项目实践丛书）

ISBN 7-111-15960-8

Ⅰ. V… Ⅱ. 赛… Ⅲ. C 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2004）第 141811 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策　　划：胡毓坚

责任编辑：李利健

责任印制：李　妍

北京蓝海印刷有限公司印刷 • 新华书店北京发行所发行

2005 年 1 月第 1 版 • 第 1 次印刷

787mm×1092mm 1/16 • 24 印张 • 594 千字

0001-5000 册

定价：41.00 元（含 1CD）

凡购本图书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话（010）68993821、88379646

68326294、68320718

出 版 说 明

随着信息技术突飞猛进的发展，社会对软件人才的需求越来越大，要求也越来越高。为使读者快捷地掌握编程语言，并学以致用。我们组织了一批有丰富编程经验的程序开发人员，精心编写了这套“工程应用与项目实践丛书”。

本套丛书包括《Dreamweaver 工程应用与项目实践》、《Java 工程应用与项目实践》、《ASP 工程应用与项目实践》、《JSP 工程应用与项目实践》、《JBuilder 工程应用与项目实践》、《Visual Basic 工程应用与项目实践》、《Delphi 工程应用与项目实践》、《Visual C++工程应用与项目实践》、《SQL Server 工程应用与项目实践》和《Visual FoxPro 工程应用与项目实践》。

本套丛书具有如下特点：

内容全面，系统性强

系统地介绍了开发工具在工程开发中的应用方法，并对重点和难点问题作了详细阐述，内容详实，覆盖面广，可满足不同层次读者的需要。

结构清晰、分析透彻

紧密围绕相关开发工具在工程开发中的应用选择内容，并通过类比和实际应用案例透彻分析所讲知识，力求使读者抓住本质，迅速掌握相关开发工具。

突出实践，学以致用

注重所讲知识的实践应用。作者从多年开发工程项目的实践入手，将编程必备的知识与工程开发中的实际案例相结合，使读者快速分享作者在开发中的经验、方法和技巧，达到学以致用的目的。

案例完整，轻松上手

丛书中每本书的最后都通过几个完整项目实例循序渐进地介绍项目开发的完整过程，使读者在掌握开发语言的基本开发方法和技巧后，能够快速掌握实用项目的开发思路和方法。

本丛书注重工程实践，实用性强，是各级程序开发人员不可多得的参考书，也非常适合大中专院校师生学习参考。

书中的实例多来源于作者们开发的实际项目，具有实际应用的价值，随书光盘包含了所有实例的源代码程序。

商业管理软件的开发是一项复杂又充满乐趣的创造性活动，希望本书能为读者提供一些商业管理软件开发的思路和技巧，开发出适合我国商业企业的优秀商业管理软件。

机械工业出版社

前　言

Visual C++是 Microsoft（微软）公司开发的可视化编程工具，因其开发出来的应用程序与 Windows 操作系统紧密结合。代码执行效率高。一直是编程专业开发人员开发 Windows 应用程序的首选工具。

本书从项目开发必备的知识和原则入手，全面系统地介绍了 MFC（Microsoft Fundation Classes）在工程开发中的编程知识、方法和技巧。本书前 9 章主要介绍 MFC 程序设计的重点和难点技术。每章首先进行知识讲解，然后结合相关知识在工程开发中的应用实例深化理解应用，实例注重实用性、启发性和典型性，使读者能结合所学知识快速掌握 Visual C++程序的开发方法和技巧。第 10 章通过一个完整工程实例的开发过程，带领读者快速掌握实用项目的开发思路和方法，从而领会 MFC 程序开发的精髓。

本书作者从多年开发工程项目的实践入手，将编程必备的知识与工程开发中的实际案例相结合，使读者在学习 Visual C++的同时，快速分享作者在程序开发中的经验。全书按照深入浅出的原则，对不同的开发知识和工程应用问题进行了详细讲解。

本书中的实例多来源于作者自身开发的实际项目，这些例子都具有实际应用的价值，为使读者能更好地使用本书，随书光盘包含了所有实例的源代码程序。

本书由赛奎春主编，张雨、阮伟良、李贺等编写，参加编写的人员还有：邹天思、顾彦泠、刘欣、高春艳、王国辉、李浩然、高飞、张世辉、郭锐、高茹、王晶莹、高月、郝洪斌、郭铁、王晶洁、邹淑芳、高润岭等。

由于作者的水平有限，书中难免有一些错误或不足之处，敬请广大读者批评指正。

欢迎到明日公司网站与作者交流。

明日网站：www.mingrisoft.com

E-mail：mingrisoft@mingrisoft.com

编　者

光盘使用说明

运行环境及系统设置

1．本书所有实例均是在 Windows 2000 下开发的，程序测试环境为 Windows 2000。

2．系统时间设定，为保证程序查询功能的正常使用，应设置系统日期格式。步骤如下：

1）打开“控制面板”，单击“区域设置”。

2）在弹出的“区域设置 属性”对话框中，单击“日期”选项卡，在“短日期格式”列表框中选择“yyyy-mm-dd”列表项。

源程序使用方法

1．如使用源程序，应在系统上安装 Visual C++ 6.0。

2．本书实例的源程序在配套光盘中。要使用本书源程序，请将该文件夹复制到计算机硬盘上，并去掉所有文件的只读属性。

数据库环境设置

配套光盘中的实例如果使用了 Access 数据库，均可在程序中自动识别路径；如果是 SQL Server 数据库，需要对数据环境进行配置。下面以第 10 章《物流综合管理系统》为例，介绍 SQL Server 数据库程序的具体配置方法：

1）打开 Windows“开始”菜单，执行“程序\Microsoft SQL Server\企业管理器”菜单项。

2）在 SQL 企业管理器中，单击逐级展开“控制台根目录”。

3）鼠标右健单击“数据库”节点，在弹出的菜单中选择“所有任务\附加数据库”菜单项，打开“附加数据库”对话框。

4）在“附加数据库”对话框中的“要附加数据库的 MDF 文件”文本框内输入“wlzhglxt_Data.MDF”文件的完整路径，可以通过【…】按钮完成。

5）单击【确定】按钮，完成数据库的附加工作。

6)数据库附加成功后，还需要配置应用程序的数据源。读者可参阅本书光盘的 Readme.doc 文件。

目　录

第 1 章　MFC 编程基础

1.1　MFC 应用程序的运行

MFC（Microsoft Foundation Classes）是微软提供的封装 Windows API 的 C++类库，利用它，能使 Windows 程序设计的某些方面变得更加容易。

1.1.1　知识讲解

使用 Windows API 开发应用程序的读者都可能有这样的体会，即使是开发一个很小的程序，也需要对 Windows 的编程原理有很深刻的认识，而且随着程序代码长度的增长，出错率也越来越高，调试工作也变得越来越复杂。利用 MFC 开发应用程序，就可以很好地改善这个问题，利用类的可重用性和可扩充性，可以大大降低 Windows 应用程序设计的难度，减少工作量。

MFC 是一个非常庞大的体系，拥有 200 多个类，此外，还包括大量的宏及全局函数。但它在结构和逻辑上是清晰的。类是 MFC 中最主要的内容，虽然每个 MFC 类都有自己独特的构造和功能，但这些类之间并不是毫不相关的，它们以层次结构的方式组织起来。其中，CObject 位于继承关系的最底层，大多数 MFC 类都是从 CObject 直接或间接地派生出来的。CObject 类给那些继承它的类提供了如串行化支持、运行时类信息支持、诊断和调试支持等重要的特性。在实际的工程开发中，程序员也常常需要定义一些从 CObject 派生的类，从而继承 CObject 类的特性。

图 1-1 列出了 MFC 中几个最重要的类的层次关系。

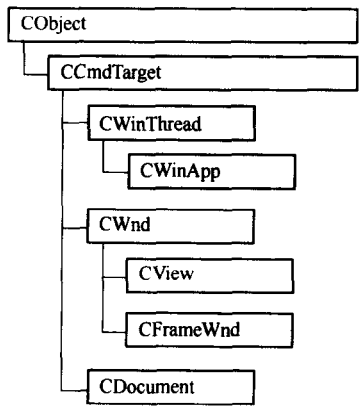

图 1-1　MFC 中最重要的类的层次关系

1.1.2　理解 CWinApp

CWinApp 类封装了 MFC 应用程序的核心对象，它提供了消息循环来检索消息，并将消息调度给应用程序的窗口，以及包括可被覆盖的、用来自定义应用程序行为的主要虚函数。

在 MFC 应用程序中有一个 CWinApp 派生类的对象 theApp（只有包含头文件 Afxwin.h，才能将 CWinApp 类引入到应用程序中），它是一个全局变量，代表了应用程序运行的主线程。每个 MFC 应用程序都会有一个 theApp，它必须声明为在全局范围内有效，这样它才可以在程序开始时（即在内存中）被实例化。

MFC 应用程序启动时，首先创建应用程序对象 theApp，这时将自动调用应用程序类的

构造函数来初始化 theApp，然后由应用程序框架调用 MFC 提供的 AfxWinMain 主函数。在 AfxWinMain 主函数中，首先通过调用全局函数 AfxGetApp 获取应用程序对象 theApp 的指针，然后通过这个指针调用应用程序对象的有关函数，完成程序的初始化和启动工作。最后调用函数 Run，进入消息循环。应用程序运行后，各函数的执行顺序和调用关系如图 1-2 所示。

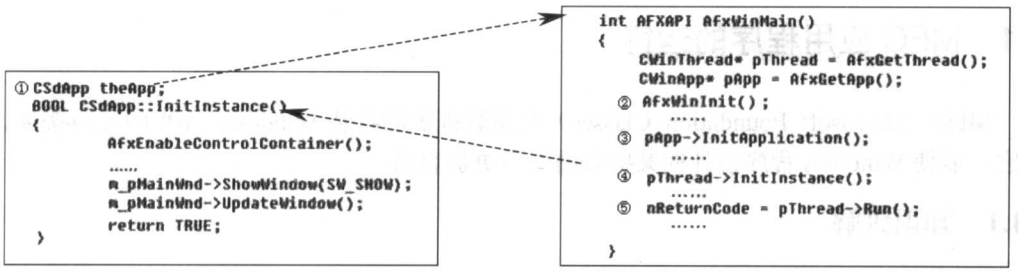

图 1-2 应用程序运行后各函数的执行顺序和调用关系

成员函数 Run 负责处理消息循环，它不断地检查消息队列中有没有消息。如果有消息，它会根据消息内容作出适当的处理；如果没有消息，这个函数会调用 OnIdle 函数进行空闲时间处理。

在应用程序类的所有成员函数中，只有 InitInstance 函数是派生类需要重载的函数，除非 InitInstance 创建一个窗口，否则应用程序是不会有窗口的。这就是为什么即使最小的 MFC 应用程序也必须从 CWinApp 派生出一个类并覆盖成员函数 CWinApp::InitInstance 的原因。

函数 InitInstance 的作用是为应用程序提供一个自身初始化的机会。由 InitInstance 返回的值决定了框架结构接下来要执行的内容。从 InitInstance 返回 FALSE 将关闭应用程序。如果初始化正常，InitInstance 函数将返回 TRUE，以便允许程序继续进行。

通常情况下，InitInstance 函数中都需要包含类似下面的代码：

```
.....
1  m_pMainWnd=new CMainWindow;
2  m_pMainWnd->ShowWindow(SW_SHOW);
3  m_pMainWnd->UpdateWindow();
.....
```

在第 1 句代码中，将一个已构造的 CMainWindow 对象的地址复制到了应用程序对象的 m_pMainWnd 数据成员中。在窗口创建以后，InitInstance 就会通过 CMainWindow 指针调用 ShowWindow 和 UpdateWindow 函数来显示它。要记住：如果不使用 SW_SHOW 属性，窗口可能是不可见的，这也正是第 2、3 句代码所要说明的问题。

ShowWindow 和 UpdateWindow 是所有窗口对象共用的 CWnd 成员函数，其中包括 CFrameWnd 类的对象，CMainWindow 就是从 CFrameWnd 派生出来的。这些函数几乎就是对同样名称的 API 函数的封装。要从 MFC 程序中调用一个常规的 Windows API 函数，需要在函数名称前添加一个全局运算符“::”，这个运算符号能确保即使对象具有与 API 函数相同名称的成员函数，也可以调用 API 函数。注意，这时的 API 函数与前面用到的 CWnd 类中

的函数同名，但其参数却是不同的，请读者加以区分。

除了 InitInstance 函数，CWinApp 还提供一些虚函数，例如，ExitInstance 函数的默认操作是做一些框架结构要求的清除事务；OnIdle 函数可以方便地执行如垃圾回收这样的后台处理事务；PreTranslateMessage 函数可以在消息被调度以前执行一些专门的预处理等工作。

1.1.3　简单的 MFC 应用程序“Hello MFC!”

经过前面的学习，想必读者已经对 MFC 有所了解，本节将通过一个小程序“Hello MFC!”来揭开 MFC 应用程序运行的秘密。

```
//hello.h
class CTheApp : public CWinApp
{
public:
    virtual BOOL InitInstance ();
};
class CMainWindow : public CFrameWnd
{
public:
    CMainWindow ();
protected:
    afx_msg void OnPaint ();
    DECLARE_MESSAGE_MAP ()
};
//hello.cpp
#include <afxwin.h>
#include "Hello.h"
CTheApp theApp;
BOOL CTheApp::InitInstance ()
{
    m_pMainWnd = new CMainWindow;
    m_pMainWnd->ShowWindow (SW_SHOW);
     m_pMainWnd->UpdateWindow ();
    return TRUE;
}
BEGIN_MESSAGE_MAP (CMainWindow, CFrameWnd)
    ON_WM_PAINT ()
END_MESSAGE_MAP ()
CMainWindow::CMainWindow ()
{
    Create (NULL, _T ("Hello MFC!"));
}
void CMainWindow::OnPaint ()
{
    CPaintDC dc (this);
```

```
        CRect rect;
        GetClientRect (&rect);
        dc.DrawText (_T ("Hello, MFC"), -1, &rect,
            DT_SINGLELINE | DT_CENTER | DT_VCENTER);
    }
```

上述代码与一般的 C++程序相比，只是没有包含自定义类以外的任何代码。例如，它没有 main 或 WinMain 函数。在整个程序中，惟一起作用的语句就是具有全局有效性的、用来实例化应用程序对象的 CTheApp theApp。到底是什么启动了程序，应用程序对象又是何时起作用的呢？

看看主框架的源代码就可以找到答案。MFC 提供的源代码文件（Winmain.cpp）中包含一个 AfxWinMain 函数，它在 MFC 中的作用相当于 WinMain。AfxWinMain 广泛使用应用程序对象，这就是为什么应用程序对象必须作全局声明的原因。全局变量和对象在任何其他代码执行以前被创建，在 AfxWinMain 运行以前，应用程序对象必须在内存中存在。

运行一开始，AfxWinMain 就调用 AfxWinInit 函数来初始化主框架，并将 hInstance、cCmdShow 以及其他 AfxWinMain 函数参数复制给应用程序对象的数据成员，然后它调用 InitApplication 和 InitInstance。在 MFC 的 16 位版本中，只有参数为空时，才调用 InitApplication，这表明当前运行的进程是应用程序的惟一实例；在 Win32 环境下，hPrevInstance 总是空的，因此主框架不必再去检查，32 位应用程序能够像使用 InitInstance 一样方便地使用 InitApplication 初始化自身，但是为了保证 MFC 与先前版本的兼容性，还是提供了 InitApplication。如果 AfxWinInit、InitApplication 或者 InitInstance 返回 0 值，则 AfxWinMain 终止而不是继续运行，应用程序被关闭。

只有在上述所有函数都返回非零值时，AfxWinMain 才执行以下关键的步骤。

```
    pThread->Run();
```

Run 函数执行消息循环并开始向应用程序窗口发送信息。消息循环重复执行，直到 WM_QUIT 消息从消息队列中被检索到。这时 Run 跳出循环，并调用 ExitInstance 做一些清理工作。最后，AfxWinMain 执行一个 return 语句结束应用程序。

1.2 文档视图结构程序在工程中的应用

MFC 中大量的应用程序都是建立在文档视图结构的基础上的。通过 Visual C++提供的 MFC AppWizard，读者可以很轻松地生成文档视图工程。这种由 AppWizard 生成的文档视图工程不仅为用户封装了一些基本功能，而且形成了高度模块化的作业。但是文档视图结构对于一个 MFC 新手来说是难以驾驭的，例如，不同的变量应放在哪个类中、不同类之间如何引用类成员、自定义类实例放在哪里等。本节将为读者解答这些问题。

1.2.1 知识讲解

使用 MFC AppWizard 生成的单文档（SDI）应用程序中含有应用类、窗口框架类、视图

类和文档类。如果读者刚刚接触 MFC，那就应该特别注意：这些类是一个有机整体，既有分工又有协作，很少由单独一部分去完成一项实际任务，而是由每个对象负责一定的行为。为了便于讲解，这里用 MFC AppWizard 生成一个单文档程序，工程名字为“FirstSDI”，并且保留 MFC AppWizard 中的默认设置。

在工程的 ClassView 窗口中可看到应用类 CFirstSDIApp、框架窗口类 CMainFrame、视图类 CFirstSDIView 和文档类 CfirstDoc。按〈Ctrl+F5〉键运行程序，如图 1-3 所示。

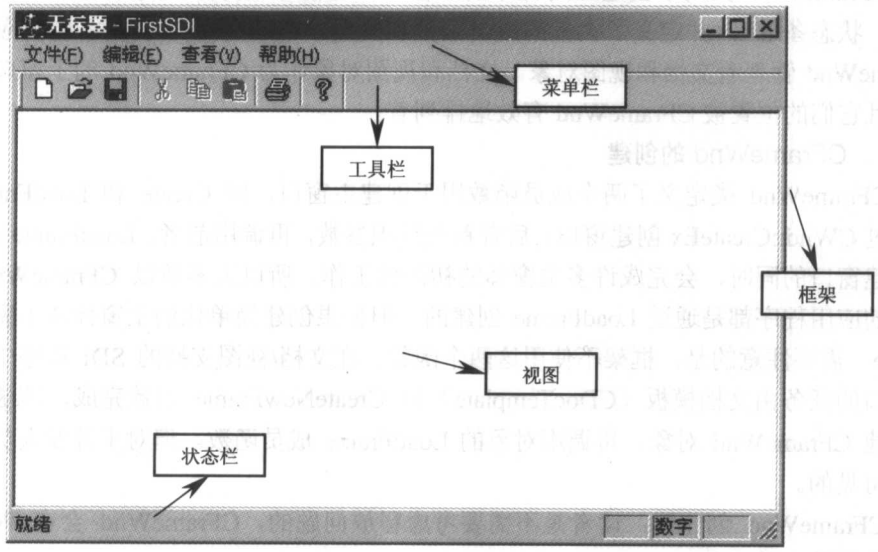

图 1-3　FirstSDI 应用程序运行结果

读者可以把整个窗口看作是主窗口框架，其中央部分为视图，即视图是主窗口框架的子窗口。主窗口框架除了视图，还有边框、标题栏、系统菜单、工具栏和状态栏。主窗口框架类和视图类都是 CWnd 的派生类，所以它们有很多共同的方法。

为什么 MFC 要使用文档/视图这种结构呢？应用程序又是怎样将它们有效地结合起来的呢？

MFC 使用文档/视图结构，最重要的一个特征就是它能够将数据管理和数据显示分离开；至于第 2 个问题，读者有必要看一看下面的代码段：

```
BOOL CFirstSDIApp::InitInstance()
{
……
    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,
        RUNTIME_CLASS(CFirstSDIDoc),
        RUNTIME_CLASS(CMainFrame),       // main SDI frame window
        RUNTIME_CLASS(CFirstSDIView));
    AddDocTemplate(pDocTemplate);
……
```

```
    return TRUE;
}
```

CSingleDocTemplate 是单文档模板，框架利用它把各个类组织到一起，使之协调工作。

1.2.2 CFrameWnd 类

CFrameWnd 类用于创建应用程序的主窗口，它能够很好地支持系统菜单和控制条（工具条、状态条等），并定义了大量的成员函数和变量。在编写文档/视图结构的应用程序时，CFrameWnd 管理着文档和视图对象。文档和视图对象作为 CFrameWnd 的子窗口分享着客户区，且它们的位置被 CFrameWnd 有效地排列着。

1．CFrameWnd 的创建

CFrameWnd 类定义了两个成员函数用于创建主窗口，即 Create 和 LoadFrame。前者主要通过 CWnd::CreateEx 创建窗口；后者首先组织参数，再调用前者。LoadFrame 的形参简洁，在创建窗口的同时，会完成许多主窗体的初始化工作。所以大多数以 CFrameWnd 为程序主窗体的应用程序都是通过 LoadFrame 创建的。但如果创建简单化的主窗体或子窗体，可调用 Create。需要注意的是，框架不使用这两个函数。在文档/视图支持的 SDI 程序中，创建主框架窗口的任务由文档模板（CDocTemplate）的 CreateNewFrame 函数完成，该函数首先动态地创建 CFrameWnd 对象，再调用对象的 LoadFrame 成员函数。但对于开发人员来说，这些是不可见的。

CFrameWnd 创建后，读者是不需要考虑释放问题的，CFrameWnd 会在 PostNcDestroy 函数中清除当前对象。

2．管理视图对象

视图是主框架窗口中一个 ID 为 AFX_IDW_PANE_FIRST 的子窗口，与主框架窗口类似，视图也是由 CFrameWnd 创建的。CFrameWnd 中提供了成员函数 OnCreateClient 专门用于完成此工作。这个函数是在 WM_CREATE 消息响应函数中被调用的。

一个主窗口可能包含多个视图，它们或者通过 CSplitterWnd 在客户区拆分创建，或者直接以子窗口形式创建。但是，框架规定只能有一个活动视图，这也是不使用拆分条，同时只能显示一个视图的原因。

主框架窗口创建后（视图也已创建），一般要调用 CFrameWnd::InitialUpdateFrame 进行初始化，该函数首先设置第一视图（ID 为 AFX_IDW_PANE_FIRST）为活动视图，然后向所有视图发送初始化消息，确保每个视图的 CView::OnInitialUpdate 都被调用。

若读者需要设置或返回活动视图，就需要使用 CFrameWnd 提供的 SetActiveView 和 GetActiveView 来完成。在设置活动视图后，应该将活动视图的 ID 切换为 AFX_IDW_PANE_FIRST，因为有些操作是只针对第一视图的。例如，只有第一视图才能与控制条争夺主窗口客户区的空间，所以其他视图无法在主框架窗口中正常显示（如果不使用拆分条）。

3．管理控制条

主框架窗口的直观特点是被丰富的控制条装饰着，如工具条、状态条、扩展的停靠窗口等，它们都派生于基类 CControlBar。当鼠标移到工具条按钮或某菜单项区域时，相应的提示信息会在状态栏中显示或以 Tip 形式弹出；没有建立消息映射的命令会自动禁止；客户区

发生变化时视图和控制条会自动排列。这一切都是 CFrameWnd 封装的功能。下面列举几个重要的控制条操作函数：

EnableDocking：允许控制条在自己的客户区停靠。

DockControlBar：将控制条停靠在客户区周边。

FloatControlBar：将控制条浮动在屏幕上，而不是停靠在客户区。

ShowControlBar：显示或隐藏控制条。

SaveBarState：将所有控制条的状态存入初始化文件或注册表。

LoadBarState：从初始化文件或注册表中恢复所有控制条状态。

GetDockState：将控制条状态信息存入一个 CDockState 对象。

SetDockState：从一个 CDockState 对象中恢复控制条状态。

SetMessageText：在状态栏的第一个面板区域显示一个信息串。

RecalcLayout：虚函数，当控制条位置变化或客户区尺寸变化时被调用，重新设置视图及控制条在客户区的位置。可根据需要重载它或主动调用它。

4．分发命令消息

命令消息是指菜单、工具栏、加速键及命令按钮向其所在窗口发送的 WM_COMMAND 消息。主框架窗口通常包含应用程序的系统主菜单和工具栏，而加速键往往在 LoadFrame 中装入主窗口，它们都要向主窗口发送命令消息。

命令消息与窗口消息（除 WM_COMMAND 之外，前缀是 WM_的消息）不同，窗口消息与某一窗口（句柄）紧密相关，应该由接收消息的窗口来处理；而命令消息往往与具体的窗口无关，只是为程序完成一个功能操作。主框架窗口的系统菜单（工具按钮）尤其如此，一个主菜单命令在其他窗口中（如视图）或在其他模块中（如文档）处理也许更合理。为了解决这个矛盾，CFrameWnd 实现了分发命令消息的机制，它能够将本窗口收到的命令消息分发给视图、文档和应用类。

CCmdTarget 类定义了一个 OnCmdMsg 虚函数，用于处理命令消息，派生类可以重载它，实现自己的命令消息处理方式。CFrameWnd 的命令消息分发机制就是通过重载这个函数实现的。

最后，要注意这样一个事实：主窗口直接调用视图（间接调用文档）、应用类的 OnCmdMsg 虚函数处理命令消息，并没有通过 SendMessage 或 PostMessage 将命令消息转发，而 OnCmdMsg 仅仅在类中搜索消息映射表，查找该命令的处理函数，找不到则返回 FALSE。所以视图类只有通过消息映射，才能处理主窗口转发的命令消息，如果使用 CView::WindowProc 捕捉该类消息，会一无所获。

5．必要的消息处理

为了管理控制条和视图，CFrameWnd 为几个窗口消息建立了消息映射，专门进行处理。下面列举出几个消息处理函数以及它们的说明：

OnInitMenuPopup：处理 WM_INITMENUPOPUP 消息，设置弹出菜单的各项目的启用/禁止状态。

OnEnterIdle：处理 WM_ENTERIDLE 消息，设置状态条空闲时的提示信息。

OnMenuSelect：处理 WM_MENUSELECT 消息，当某菜单项被选择时更新状态条提示。

OnToolTipText：处理 TTN_NEEDTEXT 通知消息，显示工具条的工具提示。