Windows 2000 应用系列

Windows 2000 编程

李多多 等 编著

人 民 邮 电 出 版 社

需要全本请访问 www.ertongbook.com

第 4 章　Windows 窗口与对话框

Windows 中的窗口是用户与应用程序进行交互的主要单元，在显示屏幕上体现为一块矩形区域。窗口及其控件作为图形用户界面（GUI，Graphics User Interface）的主要元素，可以接受用户的输入并在应用程序的指令作用下作出响应，从这个角度上讲，它是 Windows 环境中事件和消息的目标。窗口的属性决定了它的外观，而其方法决定了其如何响应用户的输入和其他系统事件。

对话框(Dialog)实际上是一个窗口。在 MFC 中，对话框的功能被封装在了 CDialog 类中，CDialog 类是所有窗口的基类 CWnd 类的派生类。对话框是最常用的人机交互工具，在 GUI 编程中非常重要。

本章将就 Windows 中的窗口和对话框进行介绍。

4.1 Windows 窗口概述

4.1.1　窗口类简介

在 MFC 中类 CWnd 是一个很重要的类，它封装了 Windows 窗口句柄 HWND。在 Windows 编程中，窗口句柄唯一地标识了一个窗口。除了直接从 CWnd 派生新的窗口类以外，我们更倾向于从 MFC 中定义其他的类，如 CFrameWnd、CMDIFrameWnd、CMDIChileWnd、CView 和 CDialog 及 CButton 之类的控件类派生新的窗口类。在 MFC 中定义的这些类本身也是从 CWnd 派生的。

Windows 通过分层来组织窗口，所有窗口的根是桌面窗口（Desktop Window），由 Windows 在启动时创建，其他顶层窗口由相应的应用程序线程创建。顶层窗口可以拥有自己的子窗口，并且窗口的父子关系可以嵌套。

在 MFC 中 CWnd 类的层次结构如图 4-1 所示。

在传统的 Windows 应用程序中，消息是通过通常具有 WndProc 之类的函数名的窗口过程(Window Procedure))即回调函数来处理的。这种方式在 MFC 中仍然使用，但为 CWnd 类及其消息映射所隐藏。在类 CWnd 中，Windows 消息会自动的通过消息映射传递到合适的 OnMessage 成员函数进行处理。通常我们都在类 CWnd 的派生类中重载需要处理的特定消息所对应的 OnMessage 成员函数。

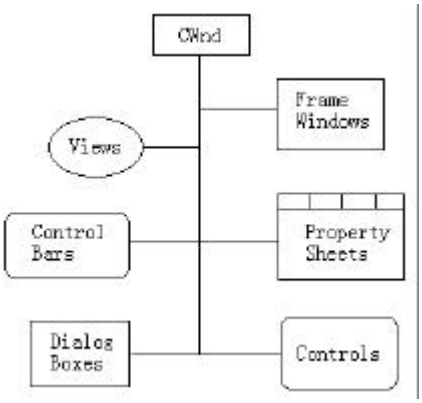


图 4-1　CWnd 类的层次结构

4.1.2　常见窗口类型

1．框架窗口

框架窗口为应用程序的用户界面提供结构框架，它是应用程序的主窗口，负责管理其包容的窗口，一个应用程序的最顶层的框架窗口是应用程序启动时创建的第一个窗口。

MFC 提供 3 种类型的框架窗口：

- 单文档（SDI）窗口　单文档窗口一次只能打开一个文档框架窗口，对应于 CFrameWnd 类。
- 多文档（MDI）窗口　MDI 应用程序运行时，在应用程序的一个实例中可以打开多个不同类型的文档框架窗口，这些窗口称作子窗口（Child Window）。如 Visual Studio 就可以打开资源文件窗口和源程序窗口等不同类型的窗口。此时，激活不同类型的 MDI 子窗口，菜单也将相应变化。多文档（ MDI）窗口由 MFC 中的 CMDIFrameWnd、CMDIChildWnd 类提供支持。
- 对话框　对应于 Cdialog 类。

下面对上述各类做详细介绍。

- CframeWnd　用于 SDI 框架窗口，形成单个文档及其视图的边框。框架窗口既是应用程序的主框架窗口，也是当前文档对应的视图的边框。要生成一个单文档窗口，主窗口必须从 CFrameWnd 派生。
- CMDIFrameWnd　用于 MDI 应用程序的主框架窗口。主框架窗口是所有 MDI 文档窗口的容器，并与它们共享菜单条。MDI 框架窗口是出现在桌面中的顶层窗口。要生成一个多文档窗口，主窗口必须从 CMDIFrameWnd 派生，而且其中的子窗口必须从 CMDIChildWnd 派生。
- CMDIChildWnd　用于在 MDI 主框架窗口中显示打开的各个文档。每个文档及其视图都有一个 MDI 子框架窗口，子框架窗口包含在 MDI 主框架窗口中。子框架窗口看起来类似一般的框架窗口，但它包含在主框架窗口中，而不是位于桌面的，并且为主窗口所裁剪。而且 MDI 子窗口没有自己的菜单，它与主 MDI 框架窗口共享菜单。

- Cdialog 对话框是一种特殊类型的窗口，它的边框一般不可以调整，而且内部包含一些控件窗口。基于对话框的窗口要从 CDialog 派生出主窗口类。

2．父窗口

术语父窗口(Parent)是相对于子窗口而言。若某一个窗口拥有一个子窗口(Child)，则该窗口就被称为子窗口的父窗口。父窗口是拥有子窗口的窗口。

3．子窗口

子窗口就是具有 WS_CHILD 风格的窗口，且一定有一个父窗口。所有的控件都是子窗口。子窗口可以没有边框。子窗口依赖于父窗口且完全被限制在父窗口内部。

4．拥有者窗口与被拥有者窗口

拥有者窗口(Owner)相对于被拥有者窗口而言。若某一个窗口拥有一个非子窗口，则该窗口被称为拥有者窗口。被拥有窗口(Owned)不具有 WS_CHILD 风格，可在屏幕上任意移动。

5．弹出式窗口

具有 WS_POPUP 风格，它可以没有父窗口。这种窗口几乎什么都没有，可看作一个矩形区域。

通常我们使用两个步骤来创建并使用一个窗口：

(1) 调用类 CWnd 的构造函数来构造一个 CWnd 对象。

(2) 调用其成员函数 Create 来创建窗口并将该窗口与所创建的 CWnd 对象相关联。

(3) 当用户终止该窗口时，消毁与之相关联的 CWnd 对象，或者调用 CWnd 对象的成员函数 DestroyWindow 删除窗口并消毁其数据结构。

4.1.3 CWnd 类数据成员及成员函数

大多数以 HWND 为参数的 Win32 API 函数都已作为类 CWnd 的成员函数进行了封装，很多时候我们通过类 CWnd 的派生类调用的成员函数并不是由派生类本身所提供的，而是在类 CWnd 中进行定义的。下面我们分类给出在 CWnd 类中定义的各类成员函数。

1．类 CWnd 的数据成员

类 CWnd 最关键的数据成员是 m_hWnd，它对应于与该 CWnd 对象相关联的 Windows 窗口句柄(HWND)。

2．类 CWnd 的构造/析构函数

- 构造函数：CWnd；
- 消毁与 CWnd 对象相关联的 Windows 窗口：DestroyWindow。

3．类 CWnd 的初始化函数

类 CWnd 的初始化函数如表 4-1 所示。

表 4-1　　类 CWnd 的初始化函数

成员函数	描　述
Create	创建并初始化与 CWnd 对象相关联的窗口
PreCreateWindow	在与 CWnd 相关联的 Windows 窗口被创建之前调用
CalcWindowRect	用来计算窗口矩形
GetStyle	返回当前窗口样式
GetExStyle	返回当前窗口的扩展样式
Attach	将一个 Windows 窗口句柄与 CWnd 对象相关联
Detach	解除一个 Windows 窗口句柄与 CWnd 对象的关联，并返回该句柄
PreSubclassWindow	在 SubclassWindow 调用前允许其他必要的 Subclass 操作
SubclassWindow	将一个窗口与一个 CWnd 对象相关联，并使其通过 CWnd 的消息映射传递消息
UnsubclassWindow	解除一个窗口与 CWnd 对象的关联
FromHandle	当给定一个窗口句柄时返回一个指向 CWnd 对象的指针。如果没有 CWnd 对象与该句柄相关联，则创建一个与之相关联的临时 CWnd 对象
FromHandlePermanent	给定一个窗口句柄时返回一个指向 CWnd 对象的指针。如果没有 CWnd 对象与该句柄相关联，则返回 NULL
DeleteTempMap	由 CWinApp 的空闲处理程序自动调用，以删除由 FromHandle 创建的临时 CWnd 对象
GetSafeHwnd	this 指针为 NULL 时返回 NULL，否则返回 m_hWnd
CreateEx	创建一个 Windows 覆盖窗口、弹出窗口或子窗口并将其与 CWnd 对象相关联
CreateControl	创建一个 OLE 控件，使该控件在 MFC 应用程序中可以使用一个 CWnd 对象表示

4．类 CWnd 的窗口访问函数

类 CWnd 的窗口访问函数如表 4-2 所示。

表 4-2　　类 CWnd 的窗口访问成员函数

成员函数	描　述
ChildWindowFromPoint	判断包含指定点的子窗口
FindWindow	返回由其窗口名称和窗口类标识的窗口的句柄
GetNextWindow	返回窗口管理器列表中的下一个(或上一个)窗口
GetOwner	返回指向 CWnd 对象的所有者的指针
SetOwner	改变 CWnd 对象的所有者
GetTopWindow	返回属于 CWnd 对象的第一个子窗口
GetWindow	返回与当前窗口有指定关系的窗口
GetLastActivePopup	判断 CWnd 对象所有的弹出窗口中最近激活的窗口
IsChild	判断 CWnd 对象是否为一个子窗口
GetParent	如果存在的话，获得 CWnd 对象的父窗口
GetSafeOwner	获得给定窗口的安全的所有者

需要全本请访问 www.ertongbook.com

续表

成员函数	描　述
SetParent	改变父窗口
WindowFromPoint	标识包括给定点的窗口
GetDlgItem	从指定的对话框获得标识符为指定 ID 的控件
GetDlgCtrlID	如果 CWnd 为一子窗口，返回其 ID 值
SetDlgCtrlID	当 CWnd 对象为一子窗口，不仅指对话框中的控件时，为其指定控件 ID 或窗口 ID
GetDescendantWindow	检查所有下级窗口(Descendant Window)并返回具有指定 ID 的窗口
GetParentFrame	获得 CWnd 对象的父框架窗口
SendMessageToDescendants	发送一条消息到窗口的所有下级窗口
GetTopLevelParent	获得窗口的顶层父窗口
GetTopLevelOwner	获得窗口的顶层所有者窗口
GetParentOwner	返回指向子窗口的父窗口的指针
GetTopLevelFrame	获得窗口的顶层框架窗口
UpdateDialogControls	用来更新对话框按钮或其他控件的状态
UpdateData	初始化对话框或从对话框中获取数据
CenterWindow	相对于父窗口使窗口居中

5．类 CWnd 的窗口大小和位置函数

类 CWnd 的窗口大小和位置函数如表 4-3 所示。

表 4-3　　类 CWnd 的窗口大小和位置成员函数

成员函数	描　述
GetWindowPlacement	获得显示状态和窗口的正常、最小化和最大化位置
SetWindowPlacement	设置显示状态和窗口的正常、最小化和最大化位置
GetWindowRgn	获得窗口的窗口区域的拷贝
SetWindowRgn	设置窗口区域
IsIconic	判断窗口是否被最小化(图标化)
IsZoomed	判断窗口是否被最大化
MoveWindow	改变窗口的位置和度量
SetWindowPos	改变子窗口、弹出式窗口或顶层窗口的大小、位置和顺序
ArrangeIconicWindows	排列所有最小化的子窗口
BringWindowToTop	将 CWnd 对象放到窗口栈的顶部
GetWindowRect	获得 CWnd 对象的屏幕坐标
GetClientRect	获得 CWnd 对象客户区的度量

6．类 CWnd 的更新和绘制函数

类 CWnd 的更新和绘制函数如表 4-4 所示。

表 4-4　　类 CWnd 的更新和绘制函数

成员函数	描　述
BeginPaint	为重绘操作准备 CWnd 对象
EndPaint	标记重绘操作的结束
Print	在指定的设备描述表绘制当前窗口
PrintClient	在指定的设备描述表绘制所有窗口
LockWindowUpdate	禁止或重新允许绘制指定的窗口
UnlockWindowUpdate	解除 CWnd::LockWindowUpdate 对窗口的锁定
GetDC	获得客户区的显示上下文
GetDCEx	获得客户区的显示上下文，并在绘制过程中允许裁剪
RedrawWindow	在客户区中更新指定的矩形或区域
GetWindowDC	获得整个窗口的显示上下文，包括标题条，菜单和滚动条
ReleaseDC	释放客户区或窗口设备描述表，并使其可为其他程序所使用
UpdateWindow	更新客户区
SetRedraw	决定在 CWnd 对象中的改变是否被重绘
GetUpdateRect	获得完全覆盖 CWnd 对象的更新区域的最小矩形坐标
GetUpdateRgn	获得 CWnd 对象的更新区域
Invalidate	使用整个客户区无效
InvalidateRect	通过将给定矩形添加到当前更新区域来使包括在给定矩形内的客户区无效
InvalidateRgn	通过将给定区域添加到当前更新区域来使包括在给定区域内的客户区无效
ValidateRect	通过将给定矩形从当前更新区域中移出来使包括在给定矩形内的客户区有效
ValidateRgn	通过将给定区域从当前更新区域中移出来使包括在给定区域内的窗户区有效
ShowWindow	显示或隐藏窗口
IsWindowVisible	判断窗口是否可见
ShowOwnedPopups	显示或隐藏窗口拥有的所有弹出式窗口
EnableScrollBar	允许或禁止滚动条上的一个或两个箭头

7．类 CWnd 的窗口状态函数

类 CWnd 的窗口状态成员函数如表 4-5 所示。

8．类 CWnd 的坐标映射函数

类 CWnd 的坐标映射函数主要有如下几个：

- MapWindowPoints　从 CWnd 对象的坐标空间映射一系列点到另一窗口的坐标空间。
- ClientToScreen　转换给定点的客户坐标或显示矩形到屏幕坐标。

- ScreenToClient　转换给定点的屏幕坐标或显示矩形到客户坐标。

表 4-5　类 CWnd 的窗口状态成员函数

成员函数	描　述
IsWindowEnabled	判断窗口是否允许鼠标或键盘输入
EnableWindow	允许或禁止使用鼠标和键盘进行输入
GetActiveWindow	获得激活窗口
SetActiveWindow	激活窗口
GetCapture	获得拥有鼠标捕获的窗口
SetCapture	使 CWnd 对象获得在此之后的所有鼠标输入事件
GetFocus	获得当前拥有输入焦点的 CWnd 对象
SetFocus	获得当前输入焦点
GetDesktopWindow	获得 Windows 桌面窗口
GetForegroundWindow	返回指向前台窗口(当前用户正在工作的顶层窗口)的指针
SetForegroundWindow	将创建该窗口的线程放至前台并激活窗口
GetIcon	获得图标句柄
SetIcon	设置句柄为一指定图标
GetWindowContextHelpId	获得帮助上下文标识符
SetWindowContextHelpId	设置帮助上下文标识符
ModifyStyle	修改当前窗口样式
ModifyStyleEx	修改当前窗口的扩展样式

9．类 CWnd 的窗口文本函数

类 CWnd 的窗口文本函数主要有如下几个：

- SetWindowText　设置窗口文本或标题条(如果有的话)为指定文本。
- GetWindowText　获得窗口文本或标题条。
- GetWindowTextLength　返回窗口文本或标题条的长度。
- SetFont　设置当前字体。
- GetFont　获得当前字体。

10．类 CWnd 的窗口滚动函数

类 CWnd 的窗口滚动函数如表 4-6 所示。

11．类 CWnd 的插入符函数

类 CWnd 的插入符成员函数主要有如下几个：

- CreateCaret　创建新的插入符形状，并获得该插入符的所有权。
- CreateSolidCaret　创建方块形状的插入符，并获得该插入符的所有权。
- CreateGrayCaret　创建变灰方块形状的插入符，并获得该插入符的所有权。

- GetCaretPos　获得插入符当前位置的客户坐标。
- SetCaretPos　移动插入符到指定的位置。
- HideCaret　隐藏插入符。
- ShowCaret　在插入符的当前位置显示插入符。

表 4-6　　类 CWnd 的滚动成员函数

成员函数	描　述
GetScrollPos	获得滚动框的当前位置
GetScrollRange	拷贝给定滚动框中滚动块的当前最大和最小位置
ScrollWindow	滚动客户区的内容
ScrollWindowEx	滚动客户区内容。与 ScrollWindowEx 类似，但具有一些附加特性
GetScrollInfo	获得关于某一滚动条的由 SCROLLINFO 结构维护的信息
GetScrollLimit	获得滚动条的限制
SetScrollInfo	设置关于滚动条的信息
SetScrollPos	设置滚动条的当前位置，并在指定的情况下重绘滚动条以反映新的位置
SetScrollRange	设置给定滚动条的最小和最大位置值
ShowScrollBar	显示或隐藏滚动条
EnableScrollBarCtrl	允许或禁止兄弟滚动条控件
GetScrollBarCtrl	返回兄弟滚动条控件
RepositionBars	在客户区中对控件条重定位

12．类 CWnd 的窗口消息函数

类 CWnd 的窗口消息成员函数主要有如下几个：

- GetCurrentMessage　返回窗口正在处理的消息的指针。仅在一个 OnMessage 消息处理函数中调用该成员函数。
- Default　调用默认窗口过程，该过程提供对所有应用程序未处理的消息的默认处理。
- PreTranslateMessage　由 CWinApp 使用，在窗口消息被发送到 TranslateMessage 和 DispatchMessage 之前对其进行过滤。
- SendMessage　将一条消息发送到CWnd对象，直至该对象处理该消息之后才返回。
- PostMessage　将一条消息放入程序的消息队列，不等待窗口处理该消息就立即返回。
- SendNotifyMessage　将指定消息发送到窗口，并尽可能快地返回，这依赖于调用线程如何创建窗口。

13．类 CWnd 的可重载函数

类 CWnd 的可重载成员函数主要有如下几个：

- WindowProc　为 CWnd 对象提供一个窗口过程。默认的窗口过程通过消息映射发

送消息。

- DefWindowProc　调用默认窗口过程，该过程提供应用程序未处理的所有窗口消息的默认处理。
- PostNcDestroy　在窗口被消毁后由 OnNcDestroy 函数调用。
- OnNotify　由框架调用以通知父窗口某一事件在某一控件中发生或者该控件需要信息。
- OnChildNotify　由父窗口调用以给通知控件一个响应控件通知的机会。
- DoDataExchange　用于对话框数据交换和验证。由 UpdateData 调用。

4.2 Windows窗口操作

4.2.1 窗口的创建

一、窗口的创建步骤

正如 4.1.2 节中所述，窗口的创建分为两步：

(1) 创建一个 C++的窗口对象，但是此时只是初始化窗口的数据成员，并没有真正创建窗口。

可以有以下 2 种方法：

- 用 new 操作符创建窗口对象：

CMainFrame* pMyFrame=new CMainFrame()；

- 定义一个窗口对象，自动调用其构造函数：

CMainFrame MyFrame;

其中 CMainFrame 是从 CFrameWnd 派生的对象。

(2) 利用 CFrameWnd 的 Create 成员函数做出窗口，并将其句柄 HWND 保存在 C++对象的公共数据成员 m_hWnd 中。

可以有以下 2 种方法：

- 利用指向 CmainFrame 类窗口对象的指针作出窗口：

pMyFrame->Create(NULL,“My Window”)；

- 利用 CmainFrame 类窗口对象变量名作出窗口：

MyFrame.Create(NULL,“My Window”)；

二、窗口的不同创建方法

1．利用 Create 函数创建

Create 函数的原形如下：

BOOL Create(LPCTSTR lpszClassName, LPCTSTR lpszWindowName, DWORD dwStyle = WS_OVERLAPPEDWINDOW, const RECT& rect = rectDefault, CWnd* pParentWnd = NULL, LPCTSTR lpszMenuName = NULL, DWORD dwExStyle = 0, CCreateContext*

pContext = NULL);

参数说明：

- lpszClassName　**窗口注册类名，它指定了窗口的图标和类风格。这里我们使用** NULL **作为其值，表明使用默认属性。**
- LpszWindowName　**窗口标题。**
- DwStyle　**窗口的风格。**
- RECT& rect　**窗口对应矩形区域的大小。**
- PParentWnd　**窗口的父窗口。**
- lpszMenuName　**窗口的菜单名。**

当调用 Create **或** CreateEx **函数时，操作系统会向窗口发送一条** WM_CREATE **消息。** OnCreate **函数就是用来响应** WM_CREATE **消息的。**

对于特定的应用程序，在 OnCreate **函数中，首先调用** CFrameWnd **的默认处理方法** OnCreate **完成窗口创建工作，后面是应用程序主窗口的特定工作。例如可以在用户自己的应用程序主窗口中，创建工具条和状态栏等。**

2．**使用** LoadFrame **从资源文件中创建窗口**

对于从 CFrameWnd **派生出来的窗口，可以使用** LoadFrame **从资源文件中创建窗口，它只需要一个参数。其调用形式如下：**

pMyFrame->LoadFrame(IDR_FRAME)；

LoadFrame **使用该参数从资源中获取许多默认值，包括主边框窗口的标题、图标、菜单、加速键等。但是，在使用** LoadFrame **时，必须确保标题字符串、图标、菜单、加速键等资源使用同一个** ID **标识符。在本例中，我们使用** IDR_FRAME**。**

4.2.2　窗口的显示

创建完窗口后，可调用 ShowWindow **成员函数来显示窗口。**ShowWindow **带一个参数，指示窗口以何种方式显示。**

CWnd:: ShowWindow **函数原型为：**

BOOL ShowWindow(int nCmdShow);

返回值：如果窗口已可见，则返回非 0 **值，否则返回** 0**。**

参数说明：

nCmdShow **决定** CWnd **对象的显示方式，其取值必须为以下各值之一：**

- SW_HIDE　**隐藏该窗口，并将激活状态传递至另外的窗口。**
- SW_MINIMIZE　**最小化该窗口并激活系统列表中的顶层窗口。**
- SW_RESTORE　**激活并显示该窗口。如果该窗口已被最小化或最大化，则** Windows **会将该窗口重置为其初始状态，包括原始大小和位置。**
- SW_SHOW　**激活窗口并将其以当前大小和位置显示。**
- SW_SHOWMAXIMIZED　**激活窗口并最大化显示。**
- SW_SHOWMINIMIZED　**激活窗口并最小化显示。**

需要全本请访问 www.ertongbook.com

- SW_SHOWMINNOACTIVE 以图标方式显示窗口，如果窗口已被激活，则其激活状态不变。
- SW_SHOWNA 以当前状态显示窗口，其激活状态保持不变。
- SW_SHOWNOACTIVATE 以最近状态(包括大小和位置)显示窗口，其激活状态保持不变。
- SW_SHOWNORMAL 激活并显示窗口，如果该窗口已被最小化或最大化，则 Windows 会将该窗口重置为其初始状态，包括原始大小和位置。

该函数用于设定窗口的可见性。对于每个应用程序的主窗口，ShowWindow 只能与 CWinApp::m_nCmdShow 一起调用一次。在此之后对于 ShowWindow 的调用必须采用上述各值中的一个来代替 CWinApp::m_nCmdShow 所指定的值。

默认方式为 SW_SHOW，但实际上我们经常希望应用程序启动时窗口最大化，此时可以将该参数改为 SW_SHOWMAXMIZED。

4.2.3 窗口的注册

在传统的 Windows C程序中，送给一个窗口的所有消息是在它的窗口函数中处理的。把一个窗口同它的窗口函数联系起来的过程称为注册窗口类。注册窗口包括对窗口指定一个窗口函数（给出窗口函数的指针）以及设定窗口的光标、背景刷子等内容。一个注册窗口类可以被多个窗口共享。注册窗口通过调用 API 函数 RegisterClass 来完成。

在 MFC 下提供了默认的自动窗口注册过程。框架仍然使用传统的注册类，而且提供了几个标准的注册类，它们在标准的应用程序初始化函数中注册。调用 AfxRegisterWndClass 全局函数就可以注册附加的窗口类，然后把已经注册的类传给 CWnd 的 Create 成员函数。用户可以定制自己的注册过程，以提供一些附加的特性。比如设置窗口的图标、背景、光标等。下面是注册窗口的例子。

```
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
//修改窗口类或风格
UINT ClassStyle=CS_VREDRAW|CS_HREDRAW;
//指定窗口标题风格，关闭自动添加文档标题的功能
cs.style=cs.style&(~FWS_ADDTOTITLE);
//AfxRegisterWndClass 指定窗口使用箭头光标
cs.lpszClass = AfxRegisterWndClass(ClassStyle,
AfxGetApp()->LoadStandardCursor(IDC_ARROW),
//背景刷子使用比窗口颜色标号大一号的颜色
(HBRUSH)(COLOR_WINDOW+1),
//使用 IDR_MAINFRAME 标识符指定的图标
AfxGetApp()->LoadIcon(IDR_MAINFRAME));
return TRUE;
}
```

注册窗口在 CFrameWnd 的 PreCreateWindow 方法中完成，且注册窗口的工作必须在调用 Create 函数创建窗口之前完成。

4.2.4　激活窗口

活动窗口必定是一个没有父窗口的顶层窗口，包括框架窗口和对话框。当顶层窗口被激活时，Windows 向窗口发送 WM_ACTIVATE 消息，对此消息的默认处理是将活动窗口设为输入焦点。

输入焦点用于表示哪个窗口有资格接收键盘输入消息。带有输入焦点的窗口是一个活动窗口，或者是该活动窗口的子窗口。当一个顶层窗口获得输入焦点时，Windows 向该窗口发送 WM_SETFOCUS 消息，此窗口可将输入焦点重定位到它的子窗口上。子窗口不会自动获得输入焦点。失去输入焦点的窗口会收到 WM_KILLFOCUS 消息。当子窗口拥有输入焦点时，父窗口就不会处理键盘输入了。

4.2.5　关闭和销毁窗口

一个 MFC 窗口对象包括两方面的内容：一是窗口对象封装的窗口，即存放在 m_hWnd 成员中的 HWND（窗口句柄）；二是窗口对象本身是一个 C++对象。

要删除一个 MFC 窗口对象，应该先删除窗口对象封装的窗口，然后删除窗口对象本身。

关闭窗口时，动作如下：

(1) 操作系统依次向被关闭的窗口发送 WM_CLOSE 和 WM_DESTROY 消息。

(2) WM_CLOSE 消息的默认处理函数 OnClose 将调用 DestroyWindow 销毁窗口。OnClose()常用于保存窗口的状态，提示保存未保存的数据等等。

(3) 框架调用窗口的析构函数作清理工作并删除 C++窗口对象。

注意不要使用 C++的 delete 操作符来销毁框架窗口，而应当采用 CWnd 的 DestroyWindow 成员函数来销毁。DestroyWindow 首先删除子窗口，再删除窗口本身。对于窗口的不同创建方式，窗口对象的清除工作可分为以下两类：

- 若窗口对象以变量方式产生，即在栈上分配内存，该窗口对象会被自动清除。这些窗口对象以变量的形式嵌入在别的对象内或以局部变量的形式创建在栈上，对于一个以变量形式创建的窗口对象，程序员不必关心它的删除问题，因为该对象的生命期总是有限的，若该对象是某个对象的成员变量，它会随着父对象的消失而消失。若该对象是一个局部变量，那么它会在函数返回时被清除。
- 若窗口对象是用 new 操作符创建的，也就是在堆上分配内存，则需要用户自己处理。因为一个在堆中动态创建的窗口对象，其生命期是任意长的。

删除窗口最直接的方法是调用 CWnd::DestroyWindow 或::DestroyWindow，前者封装了后者的功能。前者不仅会调用后者，而且会使成员 m_hWnd 保存的 HWND 无效(NULL)。如果 DestroyWindow 删除的是一个父窗口或拥有者窗口，则该函数会先自动删除所有的子窗口或被拥有者，然后再删除父窗口或拥有者。

在一般情况下，在程序中不必直接调用 DestroyWindow 来删除窗口，因为 MFC 会自动调用 DestroyWindow 来删除窗口，这点在前面已作了阐述。

初学者在学习C++编程时，对new操作符的使用往往不太踏实，因为用new在堆中创建对象，就不能忘记用delete删除对象。读者在学习MFC的例程时，可能会产生这样的疑问，为什么有些程序用new创建了一个窗口对象，却未显示地用delete来删除它呢？问题的答案就是有些MFC窗口对象具有自动清除的功能。

不具有自动清除功能的窗口对象通常是以变量的形式创建的，无需自动清除功能。这些窗口类如下：

- 所有标准的Windows控件类；
- 从CWnd类直接派生出来的子窗口对象（如用户定制的控件）；
- 切分窗口类CsplitterWnd；
- 默认的控制条类（包括工具条、状态栏和对话条）；
- 模态对话框类。

具有自动清除功能的窗口对象通常是在堆中创建的，这些窗口类如下：

- 主框架窗口类，直接或间接从CFrameWnd类派生；
- 视图类，直接或间接从CView类派生。

读者在设计自己的派生窗口类时，可根据窗口对象的创建方法来决定是否将窗口类设计成可以自动清除的。例如，对于一个非模态对话框来说，其对象是创建在堆中的，因此应该具有自动清除功能。

综上所述，对于MFC窗口类及其派生类来说，在程序中一般不必显式删除窗口对象。也就是说，既不必调用DestroyWindow来删除窗口对象封装的窗口，也不必显式地用delete操作符来删除窗口对象本身。只要保证非自动清除的窗口对象是以变量的形式创建的，自动清除的窗口对象是在堆中创建的，MFC的运行机制就可以保证窗口对象的彻底删除。

如果需要手工删除窗口对象，则应该先调用相应的函数，如CWnd::DestroyWindow删除窗口，然后再删除窗口对象。对于以变量形式创建的窗口对象，窗口对象的删除是框架自动完成的。对于在堆中动态创建的非自动清除的窗口对象，必须在窗口被删除后，显式地调用delete来删除对象，一般在拥有者或父窗口的析构函数中进行。对于具有自动清除功能的窗口对象，只需调用CWnd::DestroyWindow即可删除窗口和窗口对象。注意，对于在堆中创建的窗口对象，不要在窗口还未关闭的情况下就用delete操作符来删除窗口对象。

4.2.6 创建一个不规则形状的窗口

可以使用新的SDK函数SetWindowRgn创建一个不规则形状的窗口。该函数将绘图和鼠标消息限定在窗口的一个指定区域，实际上使窗口成为指定的不规则形状。其具体步骤如下。

(1) 使用AppWizard创建一个基于对话框的应用程序并使用资源编辑器从主对话框资源中删除所有的默认控件、标题以及边界。

(2) 给对话类增加一个CRgn数据成员，以后要使用该数据成员建立窗口区域。

```
Class CMyDlg :: public CDialog
{
…
```

```
private :
Crgn m_rgn : // window region
…
} ;
```

(3) **修改** OnInitDialog **函数建立一个椭圆区域并调用** SetWindowRgn **将该区域分配给窗口。**

```
BOOL CMyDlg : : OnInitDialog ( )
{
CDialog : : OnInitDialog ( ) ;
//获得对话框大小
CRect rcDialog ;
GetClientRect (rcDialog );
//  创建区域对象
m_rgn . CreateEllipticRgn (0 , 0 , rcDialog.Width ( ) , rcDialog .Height ( ) );
//将其分配给窗口
SetWindowRgn (GetSafeHwnd ( ) , (HRGN) m_ rgn , TRUE );
return TRUE ;
}
```

通过建立区域和调用 SetWindowRgn**，即可建立一个不规则形状的窗口。**

4.3 Windows 对话框概述

对话框(Dialog)**实际上是一个窗口。在** MFC **中，对话框的功能被封装在** CDialog **类中，**CDialog **类是** CWnd **类的派生类。**CDialog **类在** MFC **中的类层次结构如图** 4-2 **所示。**

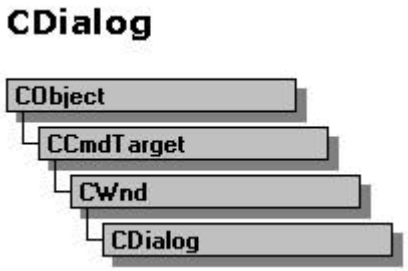


图 4-2　CDialog **类在** MFC **中的类层次结构**

对话框常可以分为以下两种：

- **模态对话框　例如通过** File->Open **命令打开的“打开文件”对话框就是一个模态对话框。模态对话框垄断了用户的输入，当一个模态对话框打开时，用户只能与该对话框进行交互，而其他用户界面对象收不到输入信息**

- 非模态对话框　非模态对话框的典型例子是 Windows 2000 提供的写字板程序中的查找对话框，查找对话框不垄断用户的输入，打开查找对话框后，仍可与其他用户界面对象进行交互，用户可以一边查找，一边修改文章。

从 MFC 编程的角度来看，一个对话框由两部分组成：

- 对话框模板资源　对话框模板用于指定对话框的控件及其分布，Windows 根据对话框模板来创建并显示对话框。
- 对话框类　对话框类用来实现对话框的功能，由于不同的应用程序对对话框的功能要求各不相同，因此一般需要从 CDialog 类派生一个新类，以完成特定的功能。

相应地，对话框的设计包括对话框模板的设计和对话框类的设计两个主要方面。

与对话框有关的消息主要包括以下两类：

- WM_INITDIALOG 消息　在对话框创建时，会收到 WM_INITDIALOG 消息，对话框对该消息的处理函数是 OnInitDialog 用处是初始化对话框。。
- 对话框内控件通知的消息　对话框的控件会向对话框发送控件通知消息，以表明控件的状态发生了变化，这是对话框进行人机交互的主要手段。

4.4 对话框操作

4.4.1 对话框模板设计

利用 Developer Studio 提供的可视化设计工具，用户可以方便地设计对话框模板。其主要步骤如下：

(1) 切换至资源视图，选择 Insert->Resource 命令，并在 Insert Resource 对话框中双击 Dialog 项。完成后在资源视图中会出现一个名为 IDD_DIALOG1 的新的对话框模板资源。

(2) 双击 IDD_DIALOG1，则会打开该对话框模板的编辑窗口。默认的对话框模板有 OK 和 Cancel 两个按钮，如图 4-3 所示。在窗口的旁边有一个控件面板，在控件面板上用鼠标选择一个控件，然后在对话框中单击，则相应的控件就被放置到了对话框模板中。图 4-4 显示了控件面板。

用户可以一次选择多个控件，选择的方法有两个：

- 在对话框的空白处拖动鼠标，则拖动出来的虚线框内的控件将被选中。
- 在选择控件时按住 Ctrl 键，则可以多重选择。

选中控件或对话框后按回车键，则会弹出一个属性对话框，属性对话框用来设置控件或对话框的各种属性。属性对话框是属性表，如图 4-5 所示。

需要注意的是，如果要在对话框中显示汉字，必须设定正确的语种和字体。请读者尝试在工作区资源视图的 Dialog 类型中单击鼠标选中需要设定的对话框 ID 项，然后按 Alt+Enter 键，并在弹出的属性对话框中的 Language 栏中选择 Chinese(P.R.C.)，如图 4-6 所示。接着，打开如图 4-5 所示的属性对话框，单击 Font...按钮，并选择“宋体”。

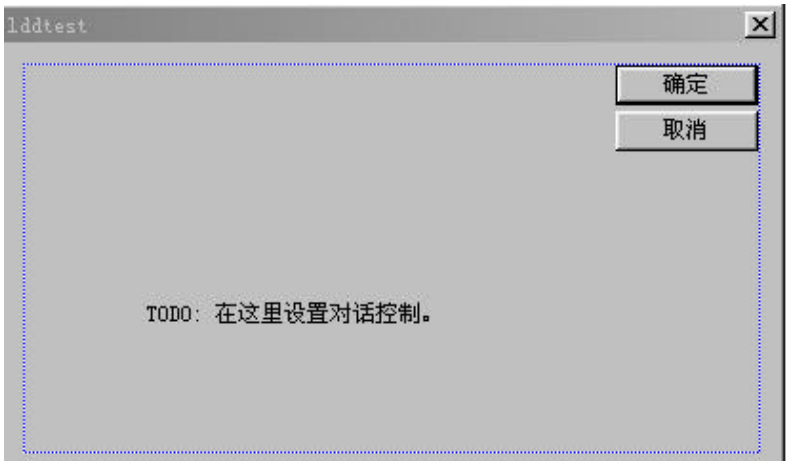


图 4-3　默认的对话框模板

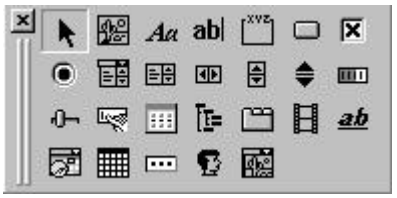


图 4-4　对话框控件面板

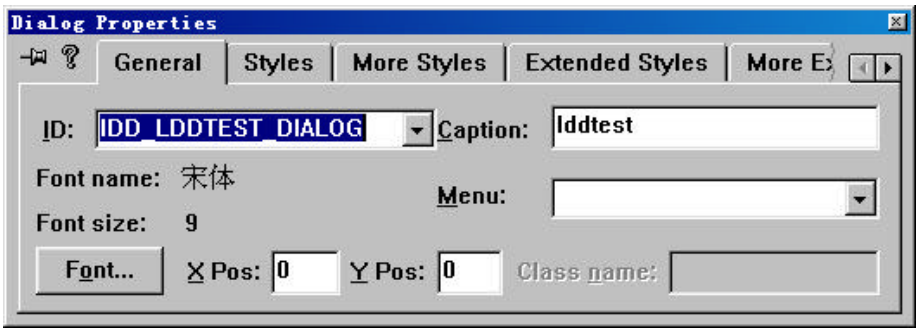


图 4-5　对话框属性对话框

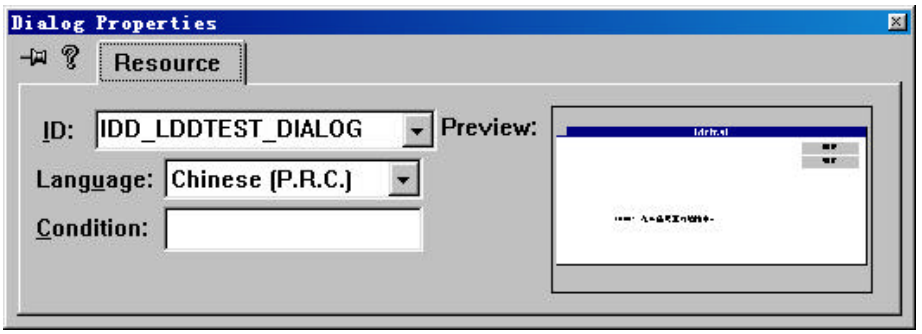


图 4-6　对话框属性对话框 Resource 选项卡

需要全本请访问 www.ertongbook.com