第11章　普 通 窗 口

MFC应用程序中几乎所有的窗口都是特殊的——要么是对话框，要么是视窗口或是控件窗口。但它们都是同一种窗口即普通窗口的不同变化。本章的实例将讲解如何创建一个通用窗口，它适用于任何情况。

Windows应用程序的用户界面完全是由单个的窗口所组成的，这些窗口具有不同的尺寸和风格。多数情况下，使用 Developer Studio中的编辑器和向导自动地将窗口加入到应用程序中去，纵观全书可以找到许多这方面的例子。不幸的是，向导只能创建一些特定种类的窗口，除此之外就需要自己动手去创建自己的窗口了。

本章中包含在应用程序用户界面的任何位置创手工建各种窗口的实例。它们包括：

实例38　创建普通窗口。本例将演示如何只用 MFC的一般窗口过程来创建窗口。

实例39　创建一个窗口类——短调用形式，本例将演示如何使用 MFC的AfxRegisterWnd-Class ()函数创建自定义窗口类，该函数自动填充了很多空白。

实例40　创建一个窗口类——长调用形式，本例将演示如何使用 MFC的AfxRegisterClass()函数创建一个窗口类，该函数可以由用户完全控制窗口的类创建过程。

11.1　实例38：创建普通窗口

1. 目标

创建一个普通窗口，如图11-1所示。

2. 策略

本例将使用MFC的通用窗口类CWnd来创建这些窗口。同时还将用 Windows API来直接创建窗口。然后使用一种将这个CWnd对象连接到已存在窗口上的方法。

3. 步骤

1) 使用CWnd创建一个普通窗口

使用MFC创建一个普通窗口，可以使用：

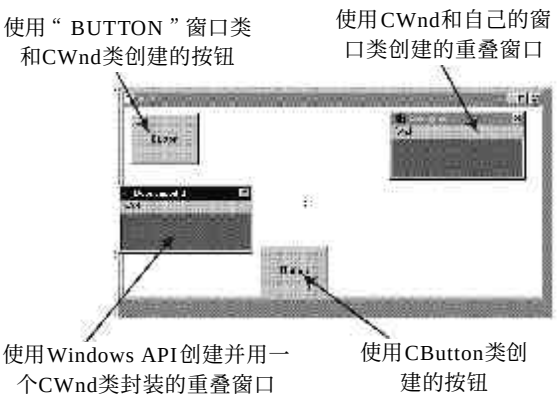

图11-1　四个普通窗口

```
CWnd wnd;
HMENU hMenu = ::LoadMenu( NULL,MAKEINTRESOURCE( IDR_WZD_MENU ) );
wnd.CreateEx(
    0,                          // extended window style
    _T( "AfxWnd" ),             // MFC window class name
    "Caption",                  // window caption
    WS_CHILD|WS_VISIBLE,        // window style
    10,10,                      // x,y position
    100,75,                     // width and height
```

```
    m_hWnd,                     // parent window handle
    hMenu                       // menu handle,
                                //     or if child window, a window id
    );
```

上面使用的的窗口类名是通用MFC窗口。为了创建一个按钮控件，只需要使用 AfxWnd来代替BUTTON：

```
CWnd wndButton;
wndButton.CreateEx( 0,                  // extended window style
    _T( "BUTTON" ),                     // window class name
    "My Button",                        // window caption
    WS_CHILD|WS_VISIBLE,                // window style
    10,10,                              // x,y position
    100,75,                             // width and height
    hWnd,                               // parent window handle
    ( HMENU )IDC_WZD_BUTTON             // in this case, a button id
    );
```

为使用MFC CButton类来创建完全相同的按钮，可使用以下代码， CButton中的Create()成员函数只完成上面我们所完成的工作:

```
CRect rect( 200, 200, 300, 275 );
CButton button;
button.Create(
    "Button",                   // window caption
    WS_CHILD|WS_VISIBLE,        // window style
    rect,                       // position and dimensions
    this,                       // parent window class
    IDC_WZD_BUTTON              // button id
    );
```

2) 使用Windows API创建一个普通窗口

为了直接用Windows API创建一个普通窗口，可以直接使用下面的代码。这个例子中将创建一个重叠窗口。请参照后面的注意以了解重叠窗口、弹出窗口和子窗口的不同之处。

```
HWND hWnd = ::CreateWindowEx(
    WS_EX_CLIENTEDGE,           // extended window style
    "AfxWnd",                   // windows class name
    "Overlapped",               // window caption
    WS_CAPTION|WS_SYSMENU|WS_OVERLAPPED|WS_VISIBLE|WS_DLGFRAME,
                                // window style
    220, 220, 200, 100,         // position and dimensions
    NULL,                       // owner window handle—NULL is Desktop
    hMenu,                      // for popup and overlapped windows
    AfxGetInstanceHandle(),     // handle to application instance
    NULL                        // pointer to window-creation data
    );
```

3) 使用CWnd对象封装Windows对象

可以使用下面的代码，使用Wnd类来封装刚才创建的窗口：

```
CWnd wndWrapper;
wndWrapper.Attach( hWnd );
```

一旦封装该窗口，当CWnd类被析构时，CWnd将会销毁这个窗口。

为了在析构CWnd类或其派生类(例如：Cbutton)时，CWnd附属的窗口不被销毁，可以在开始时用下面的代码将进行分离：

```
HWND hWnd = wndWrapper.Detach();
::DestroyWindow( hWnd );
```

如果希望窗口销毁该窗口所附属的MFC类，可使用ClassWizard加入WM_NCDESTROY消息处理函数到这个类中。然后在消息处理函数中加入如下的代码：

```
void CWzdWnd::OnNcDestroy()
{
   CWnd::OnNcDestroy();

   delete this;
}
```

4. 注意

■ 普通窗口是在Windows界面中所看到的所有窗口的基础。共有三种普通类型的窗口：重叠窗口、弹出窗口以及子窗口。它们是主窗口、对话框和消息框、控件窗口 (例如控件)的基础。Windows API通常在重叠窗口或弹出窗口所谓的非客户区进行大量的绘制工作。另一方面，子窗口是在窗口类中定义的，它通常是在窗口过程中绘制。控件窗口都是具有用于绘制各自控件的唯一窗口过程的子窗口。关于窗口的更多的细节，请参阅第 1章。

■ 在调用了CreateEx()和CreateWindow()函数之后，使用一系列从Windows API到该窗口的窗口过程的消息便创建了一个MFC窗口。关于该事件发生序列，请参阅附录 A。

■ 添加一个WM_NCDESTROY消息处理函数来析构一个MFC类在大多数情况下是不必要的。这是因为该类要么是嵌入到另一个类中，而后者在应用程序结束时将自动析构前者，要么是被分配到某一函数的堆栈中，当函数返回时，这个类也将被析构。但是有一种情况需要这样做，就是无模式对话框。只要用户单击 Close按钮，无模式对话框窗口将会被销毁。而该窗口附属的类将被剩下，它没有窗口可访问也无法析构，结果将导致内存泄漏。

■ 本例使用的类由Windows 操作系统和MFC所支持。为创建自己的窗口类，请参考实例39和实例40。

5. 使用光盘时注意

执行附带光盘上的工程时，会发现视将由不同方式创建的四个基本窗口所填充。

11.2 实例39：创建短调用形式窗口类

1. 目标

创建一个通用窗口类以便于创建窗口时所使用。

2. 策略

MFC框架为创建和注册一个窗口类提供了两个函数，本例将使用短版本的调用：AfxRegisterWndClass()。为了在创建窗口类的过程具有更多的控制，请参考实例 40。

3. 步骤

1) 用AfxRegisterWndClass()函数创建新窗口类

为创建窗口类，可以使用：

```
lpszClass = AfxRegisterWndClass(
    // window class styles
    CS_DBLCLKS |                        // convert two mouse clicks into
                                        //     a double click to this
                                        //     window's process
    CS_HREDRAW |                        // send WM_PAINT to window
                                        //     if horizontal size changes
    CS_VREDRAW                          // send WM_PAINT to window
                                        //     if vertical size changes
    // CS_OWNDC |                       // every window created from
                                        //     this class gets its very
                                        //     own device context
    // CS_PARENTDC |                    // device context created for
                                        //     this window allows drawing
                                        //     in parent window too
    // CS_NOCLOSE |                     // disable the close command
                                        //     on the System menu
    ::LoadCursor(NULL,IDC_CROSS),       // window class cursor
                                        //     or NULL for default arrow
                                        //     cursor (this cursor is
                                        //     displayed the when mouse
                                        //     cursor is over a window
                                        //     created with this class)
    (HBRUSH)(COLOR_BACKGROUND+1),       // background color
                                        //     or NULL for no background
                                        //     erase (if NULL, window
                                        //     will not erase
                                        //     background for you)
    AfxGetApp() -> LoadIcon(IDI_WZD_ICON)   // window icon or
                                        //     NULL for default
                                        //     icon (icon
                                        //     displayed in
                                        //     window caption
                                        //     or in minimized
                                        //     window)
    );
```

AfxRegisterWndClass()自动生成一个新的窗口类名。为了使用这个新的窗口类创建窗口，只需使用这个已生成的名字来创建即可。

2) 使用由AfxRegisterWndClass()创建的窗口类

为使用新的窗口类，将AfxRegisterWndClass()创建的类名加入到CWnd::CreateEx()函数中，如下所示：

```
CWnd wnd;
wnd.CreateEx( 0,lpszClass," ",WS_OVERLAPPEDWINDOW|WS_VISIBLE,
    100, 100, 200, 100, NULL, NULL );
```

3) 创建最简单的窗口类

为使用AfxRegisterWndClass()创建最简单的窗口类，可以使用如下代码：

```
lpszClass = AfxRegisterWndClass( 0 );
```

通过该窗口类创建的窗口将有一个箭头光标和一个缺省图标，其背景不能擦除。

4. 注意

■ 窗口类名只是一个文本字符串，用于标识已在系统中注册的窗口类结构。窗口类结构用于维护用户窗口类的风格、背景色以及在窗口创建时进行初始化的窗口过程。关于窗口类的更多信息，请参阅第1章。

■ AfxRegisterWndClass()自动创建和初始化一个窗口结构。对于窗口过程，它使用一个名为AfxWndProc的普通MFC窗口过程。AfxRegisterWndClass()还可以根据所传递的参数自动创建窗口类名。然而这种方式的一个缺点是如果使用完全相同的参数分两次调用该函数时，将只能创建一个窗口类。一般情况下这将无伤大雅，除非使用不连续的 CS_CLASSDC。但即使在这种情况下，也只是在两个由该窗口类创建的窗口试图同时进行绘制时才会出现问题。关于这方面问题的详细信息，请参阅第1章。

■ 如果希望在创建窗口类具有更多的自主控制，包括给自行确定窗口类名等等，请参考实例40。

5. 使用光盘时注意

执行附送光盘上的工程时，在 WzdView中的OnTestWzd1()函数中设置一个断点。单击Test和Wzd1菜单命令，在应用程序中创建两个窗口类及用这两个窗口类创建两个窗口时，跟踪该过程。

11.3 实例40：创建长调用形式窗口类

1. 目标

创建一个特定的窗口类并将能够自行确定窗口类名。

2. 策略

MFC框架中提供了两个函数以创建和注册一个窗口类。在本例中将使用长形式的调用：AfxRegisterClass()。它允许在窗口类创建过程中用户具有更多的控制权。为创建一个快速、通用的窗口类，请参考实例39。

3. 步骤

1) 用AfxRegisterClass()创建一个窗口类

为创建一个窗口类，必须首先如下所示初始化 WNDCLASS结构：

```
WNDCLASS wndclass =
{
   // window class styles
   CS_DBLCLKS |              // convert two mouse clicks into a double
                             //    click to this window's process
   CS_HREDRAW |              // send WM_PAINT to window if horizontal
                             //    size changes
   CS_VREDRAW                // send WM_PAINT to window if vertical
                             //    size changes
   // CS_GLOBALCLASS |       // class is available to all process threads
   // CS_OWNDC |             // every window created from this class
                             //    gets its very own device context
   // CS_PARENTDC |          // device context created for this window
                             //    allows drawing in parent window too
```

```
    // CS_NOCLOSE |              // disable the close command on the
                                 //    System menu
    AfxWndProc,                  // window process for every window created
                                 //    from this class
    0,0,                         // extra window and class bytes unused
                                 //    in MFC
    AfxGetInstanceHandle(),      // handle of this application's instance
    AfxGetApp() -> LoadIcon( IDI_WZD_ICON ),        // window icon or NULL
    ::LoadCursor( NULL,IDC_CROSS ),                 // window class cursor or
                                                    //    NULL for default
                                                    //    arrow cursor
    ( HBRUSH )( COLOR_BACKGROUND + 1 ),             // background color or NULL
                                                    //    for no background
                                                    //    erase
    MAKEINTRESOURCE( IDR_WZD_MENU ),                // menu to be used when
                                                    //    creating windows
                                                    //    using this class
    "MyClassName"                                   // a class name you are
                                                    //    assigning this
                                                    //    windows class
};
```

然后可以在系统中注册窗口类，使用以下代码：

```
AfxRegisterClass( &wndclass );
```

2) 使用由AfxRegisterClass()创建的窗口类

为使用这个新的窗口类，将其名字加入到 CWnd::CreateEx()函数中，如下所示：

```
CWnd wnd;
wnd.CreateEx( 0, "MyClassName",          <<< new class name
   " ",WS_OVERLAPPEDWINDOW|WS_VISIBLE,
   100, 100, 200, 100, NULL, NULL );
```

4. 注意

■ 所有由MFC创建的窗口都使用 AfxWndProc窗口过程。这允许发送到 MFC窗口的消息以同样的方式进行处理。该窗口过程确定了窗口的外观和使用感觉。一个 BUTTON窗口类则具有一个特殊的窗口过程，该过程将绘制一个按钮以作为对发送给该过程的 WM_PAINT消息的响应。关于这方面的详细信息，请参阅第 1章。

■ 本例中定义的图标将在具有标题的窗口或是最小化窗口的左上角上显示。缺省的图标由系统提供，并随操作系统的不同而不同。

■ 当鼠标在该窗口类创建的窗口的客户区移动时，这里定义的光标将显示。若未定义，则使用缺省的箭头光标。

■ 如果将背景画刷设置成 NULL，则系统不能擦除窗口的背景。这意味着当窗口创建时，窗口的所有其他非客户区仍将被绘制，而客户区内容将保留。即窗口下面的内容仍将显示在客户区中。为了自行擦除客户区的背景，可使用 ClassWizard添加WM_ERASEBKGRND消息处理函数到 MFC类中。在其中可以绘制一个矩形以填充背景，或提供一些其他的有趣的背景。

■ 这里提供的菜单句柄将被分配到由该窗口类创建的每一个窗口中。在创建窗口时，只

需为CWnd::CreateEx()提供一个新的菜单句柄就可以替代缺省菜单。如果菜单句柄在窗口类和CWnd::CreateEx()中被省略，那么窗口在创建时就没有菜单。由于子窗口不能有菜单——因此在窗口类没有使用菜单句柄，但在CWnd::CreateEx()中它被用作一个控件标识号。

■ 这里提供的类名可以是任何的文本名，包括现有的窗口类名，例如 BUTTON。但是，使用现有的类名来命名新的窗口类将意味着以后所有的窗口都将从这个新类中创建。例如，如果将新的窗口类命名为BUTTON，则将来所有的按钮控件都将使用该窗口类创建，而不是从系统缺省的BUTTON窗口类创建。有关超类化的信息请参阅第1章。

■ 应用程序注册的窗口类只对该应用程序才可用的。如果使用 CS_GLOBALCLASS类风格，那么窗口类对每一个该应用程序中创建的线程则都将是可用的，但是对系统中其他的应用程序是不可用的，至少在Windows 3.1是这样。当应用程序结束时，所创建的窗口类都将被注销和销毁。