Win Form 笔记

郑华

2018 年 5 月 13 日

1 新数据类型

主要目的是容易标识变量的功能，其实现类似于宏定义“Define UINT unsigned int”

- - MSG: 消息类型
- - HWND：句柄，就是一个资源标识，类似与指针，通过其找到对应的资源
- - UNIT：unsigned int
- - WM_: windows message 标识前缀
- - WPARAM 、LPARAM：附加消息
- - DWORD: 32 位的整数
- - POINT: 点，位置类型，成员 x、y

2 Windows 应用关系

0. 设计一个 WinFrame 的基本步骤 ：

1- 设计一个窗口类

WNDCLASS winClass; WNDCLASSEX

2- 注册窗口类

RegisterClass(&winClass);

3- 创建窗口

hwnd = CreateWindow(...);

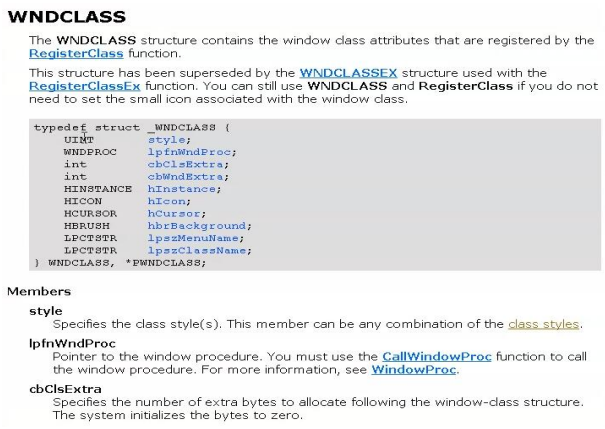
4- 显示和更新窗口

ShowWindow(hwnd,nCmdShow);

UpdateWindow(hwnd);

5- 消息循环

while(GetMessage(&msg,NULL,0,0)){



WNDCLASS

The **WNDCLASS** structure contains the window class attributes that are registered by the RegisterClass function.

This structure has been superseded by the WNDCLASSEX structure used with the RegisterClassEx function. You can still use **WNDCLASS** and **RegisterClass** if you do not need to set the small icon associated with the window class.

```
typedef struct _WNDCLASS {
    UINT        style;
    WNDPROC     lpfnWndProc;
    int         cbClsExtra;
    int         cbWndExtra;
    HINSTANCE   hInstance;
    HICON       hIcon;
    HCURSOR     hCursor;
    HBRUSH      hbrBackground;
    LPCTSTR     lpszMenuName;
    LPCTSTR     lpszClassName;
} WNDCLASS, *PWNDCLASS;
```

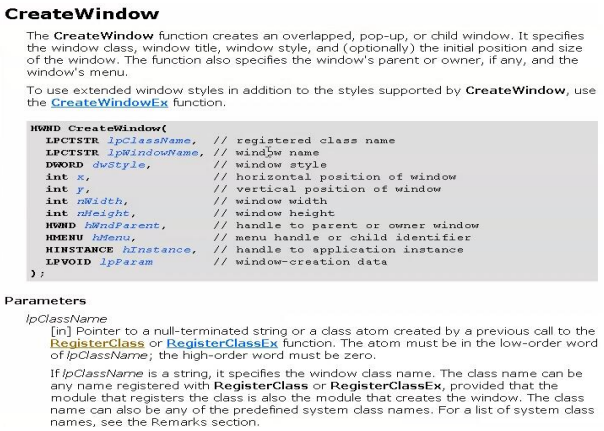
Members

style
Specifies the class style(s). This member can be any combination of the class styles.

lpfnWndProc
Pointer to the window procedure. You must use the CallWindowProc function to call the window procedure. For more information, see WindowProc.

cbClsExtra
Specifies the number of extra bytes to allocate following the window-class structure. The system initializes the bytes to zero.

图 1: wndClass 图



CreateWindow

The **CreateWindow** function creates an overlapped, pop-up, or child window. It specifies the window class, window title, window style, and (optionally) the initial position and size of the window. The function also specifies the window's parent or owner, if any, and the window's menu.

To use extended window styles in addition to the styles supported by **CreateWindow**, use the CreateWindowEx function.

```
HWND CreateWindow(
  LPCTSTR lpClassName,  // registered class name
  LPCTSTR lpWindowName, // window name
  DWORD dwStyle,        // window style
  int x,                // horizontal position of window
  int y,                // vertical position of window
  int nWidth,           // window width
  int nHeight,          // window height
  HWND hWndParent,      // handle to parent or owner window
  HMENU hMenu,          // menu handle or child identifier
  HINSTANCE hInstance,  // handle to application instance
  LPVOID lpParam        // window-creation data
);
```

Parameters

lpClassName
[in] Pointer to a null-terminated string or a class atom created by a previous call to the RegisterClass or RegisterClassEx function. The atom must be in the low-order word of *lpClassName*; the high-order word must be zero.

If *lpClassName* is a string, it specifies the window class name. The class name can be any name registered with **RegisterClass** or **RegisterClassEx**, provided that the module that registers the class is also the module that creates the window. The class name can also be any of the predefined system class names. For a list of system class names, see the Remarks section.

图 2: createWindow 图

TranslateMessage(&msg);

DispatchMessage(&msg);

}

在 vs2010 中的创建 ：

1- 新建 win32

2- 选择 windows Application

3- 下一步不用选择 Empty 工程，直接生成即可

1.WinMain 函数 ：Windows 程序的入口函数，WINAPI 是一个 Windows 定义的宏，将使系统以特定于 Windows

2. 窗口类 ：利用**结构体 WNDCLASS 进行定义窗口样式**，如图1 ，消息处理函数的函数指针 lpfnWndoroc，背景画刷 hbrBac = (HBBRUSH)GetStockObject(颜色)，菜单，窗口名等，然后利用 Register（&winclass）进行注册，然后利用 CreateWindow 创建, 并利用句柄进行保存存或指向它，如图2

3. 退出 利用 PostQuitMessage(0) 使程序结束

4. 示例代码

```
#include<Windows.h>

LRESULT CALLBACK WndProc( HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam );

int WINAPI wWinMain( HINSTANCE hInstance, HINSTANCE prevInstance, LPWSTR cmdLine,int cmdShow )
{
UNREFERENCED_PARAMETER( prevInstance );
UNREFERENCED_PARAMETER( cmdLine );

WNDCLASSEX wndClass = { 0 };
wndClass.cbSize = sizeof( WNDCLASSEX ) ;
wndClass.style = CS_HREDRAW | CS_VREDRAW;
wndClass.lpfnWndProc = WndProc;
```

```
wndClass.hInstance = hInstance;
wndClass.hCursor = LoadCursor( NULL, IDC_ARROW);
wndClass.hbrBackground = ( HBRUSH )(COLOR_WINDOW + 1 );
wndClass.lpszMenuName = NULL;
wndClass.lpszClassName = "DIRECTX11BookWindowClass";

if( !RegisterClassEx( &wndClass ) )
return -1;

RECT rc = { 0, 0, 640, 480 };
AdjustWindowRect( &rc,WS_OVERLAPPEDWINDOW, FALSE );

HWND hwnd = CreateWindowA( "DIRECTX11BookWindowClass","Blank␣Win32␣Window",
WS_OVERLAPPEDWINDOW, CW_USEDEFAULT,CW_USEDEFAULT, rc.right - rc.left,
rc.bottom - rc.top, NULL, NULL,hInstance, NULL );

if( !hwnd )
return -1;

ShowWindow( hwnd, cmdShow );


MSG msg = { 0 };

while( msg.message != WM_QUIT )
{
if( PeekMessage( &msg, 0, 0, 0,PM_REMOVE ) )
{
TranslateMessage( &msg );
DispatchMessage( &msg );
}
else
{

}
}


return static_cast<int>( msg.wParam);
}




LRESULT CALLBACK WndProc( HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam )
{
PAINTSTRUCT paintStruct;
HDC hDC;

switch( message )
{
case WM_PAINT:
hDC = BeginPaint( hwnd,&paintStruct );
EndPaint( hwnd, &paintStruct);
break;
```

```
        case WM_DESTROY:
        PostQuitMessage( 0 );
        break;

        default:
        return DefWindowProc( hwnd,message, wParam, lParam );
        }

        return 0;
        }
```

3 消息映射

1- 头文件要做的 :

afx_msg void OnPaint();

DECLARE_MESSAGE_MAP();

2- 源文件要做的 :

BEGIN_MESSAGE_MAP()

ON_WM_PAINT()

ON_WM_LBUTTONDOWN()

END_MESSAGE_MAP()

3- 右键类点击属性进行添加消息

4 字符集与 TEXT 宏

8 位 ANSI 字符集

16 位 unicode 字符集 - 宽字符集

TEXT 宏 (_T 宏)

TCHAR、TCHAR*、LPTSTR、LPCTSTR

API 分 ANSI 和 Unicode 两个版本

1.ANSI 8 位 :

char szMsgA[256] = ”hello”;

strcat(szMsgA,”ss”);

::MessageBoxA(NULL,szMsgA,” 窗口名”，MB_OK);

2.Unicode 16 位 ：

wchar_t szMsgW[256] = L”hello”;

宽型字符串直接添加 L

lstrcatW(szMsgW,L”, unicode”);

::MessageBoxW(NULL, szMsgW,L”hello”，MB_OK);

3. 自适应类型 ：

TCHAR szMsgT[256] = TEXT(”Hello”);

LPTSTR st = TEXT(”hello”);

LPCTSTR sct = TEXT(”hello”);

_tcscat(szMsgT,TEXT(”,TCHAR”));

::MessageBox(NULL„ szMsgT,TEXT(”hello”)，MB_OK)

5 常规空 MFC 的建立

1.Visual C++ - General

2.Empty Project

3. 右键项目–属性

4. 配置 MFC 的使用 dll

5. 配置字符集 unicode

6 MFC 设备绘图类

：

1.windows GDI ：

1- GDI

2- DC

最后的一个点不画【左闭右开】

2.MFC 绘图类 ：

1- CDC

2- CPaintDC

（1）CPaintDC 类是 CDC 类的一个派生类，该类一般用在响应 WM_PAINT 消息的函数 OnPaint() 中。

（2）WM_PAINT 消息是当窗口的某个区域需要重画时激发的窗口消息。当程序中的消息循环接到 WM_PAINT 消息时就自动调用消息处理函数 OnPaint()，如果在 OnPaint 函数内定义了 CPaintDC 类的对象，通过这个类对象就可以使用 CDC 类的成员函数完成视图客户区中的图形绘制操作。

（3）CPaintDC 用于响应窗口重绘消息（WM_PAINT）时的绘图输出。CPaintDC 在构造函数中调用 BeginPaint() 取得设备上下文，在析构函数中调用 EndPaint() 释放设备上下文。EndPaint() 除了释放设备上下文外，还负责从消息队列中清除 WM_PAINT 消息。因此，在处理窗口重画时，必须使用 CPaintDC，否则 WM_PAINT 消息无法从消息队列中清除，将引起不断的窗口重画。**CPaintDC 也只能用在 WM_PAINT 消息处理之中。**

3- CClientDC

CClientDC 类也是 CDC 类的派生类。**它只能在窗口的客户区（即窗口中除了边框、标题栏、菜单栏以及状态栏外的中间部分）中进行绘图**，坐标点（0,0）通常指的是客户区的左上角。它的构造函数调用 GegDC 函数，而析构函数调用 ReleaseDC 函数。CClientDC（客户区设备上下文）用于客户区的输出，它在构造函数中封装了 GetDC()，在析构函数中封装了 ReleaseDC() 函数。一般在响应非窗口重画消息（如键盘输入时绘制文本、鼠标绘图）绘图时要用到它。用法是：

CClientDC dc(this);//this 一般指向本窗口或当前活动视图

dc.TextOut(10,10,str,str.GetLength());

//利用 dc 输出文本，如果是在 CScrollView 中使用，还要注意调用 OnPrepareDC(&dc) 调整设备上下文的坐标。

4- CWindowDC

CWindowDC 类也是 CDC 类的派生类。**其成员函数可以在窗口的客户区和非客户区（即窗口的边框、标题栏、菜单栏以及状态栏）中绘图，坐标点（0,0）是指整个屏幕的左上角。**同 CClientDC 类一样，它的构造函数调用 GegDC 函数，而析构函数调用 ReleaseDC 函数。

3. 绘图函数 ：

POINT points[5] = 10,10, 20,20, 30,30, 40,40, 50,50;

dc.Polyline(points,5);

贝塞尔曲线：

dc.PolyBezier(points,4);

矩形区域：

CRect rect(左 _x, 左 _y, 右下 _x, 右下 _y);

画矩形：

dc.Rectangle(rect);

画椭圆：

dc.Ellipse(rect);

画弧线: 截取椭圆

dc.Arc(10,10, 200,100, 0,0, 80,200);

画扇形:

dc.Pie(rect,point1, point2);

画弦:

dc.Chord(rect,point1, point2);

4. 画笔与画刷 ：

使用画笔–方式 1：

CPen pen(PS_SOLID,6,RGB(255,0,0));

PS_DASH : 虚线

PS_DOT : 点线

dc.SelectObject(&pen);

使用画笔–方式 2：

CPen pen3;

LOGPEN lp;

lp.lopnStyle = PS_DASHDOT;

lp.lopnWidth.x = 1;

lp.lopnColor = RGB(0,0,255);

pen3.CreatePenIndirect(&lp);

dc.SelectObject(&pen3);

使用画刷：背景色，填充色

CBrush brush(RGB(0,0,255));

或 brush(HS_DIAGCROSS 斜网格,RGB(0,255,255));

HS_BDIAGONAL : 斜线

HS_CROSS: 正网格

HS_FDIAGONAL: 反斜线

HS_HORIZONTAL: 正斜线

dc.SelectObject(&brush);

dc.Rectangle(...);

5. 画文本 ：

1- dc.DrawText:

dc.DrawText(TEXT(”Hello”),-1,&rect,DT_SINGLELINE | DT_CENTER | DT_VCENTER);

2- dc.TextOut

dc.TextOut(100,100,TEXT(”Hello”));

3- 字体：

方式 1:

CFont font;

font.CreatePointFont(72*10,TEXT(”Arial”));

dc.SelectObject(&font);

方式 2:

LOGFONT lf;

::ZeroMemory(&lf,sizeof(lf));

lf.lfHeight = 120;

lf.lfWeight = FW_BOLD;

lf.lfItalic = TRUE;

::lstrcpy(lf.lfFaceName, TEXT(”Times New Roman”));

CFont font_Indirect;

font_Indirect.CreatePointFontIndirect(&lf);

偏移：rect.Offset

旋转：

lf.lfEscapement = 45 *10;

lf.lfOrientation = 45 *10;

6. 备用对象画笔画刷 ：

选择备用的画笔：

dc.SelectStockObject(NULL_PEN);

选择备用的画刷：

dc.SelectStockObject(LTGRAY_BRUSH);

dc.SetMapMode(MM_LOENGLISH);// 将坐标系转换为数学类型，但是左上角还为 0,0

dc.SetTextAlign(TA_CENTER | TA_BOTTOM); //文字的对齐模式

dc.SetBkMode(TRANSPARENT); //设置透明

7.win32 画图不更新 http://zhidao.baidu.com/link?url=BxIG_kklq269UNNlR1HAAIk9fGlL2HtallG4_zSSoRGOrRr3cnJ_F

要设置失效区域