EBook Gratis

APRENDIZAJE mfc

Free unaffiliated eBook created from
Stack Overflow contributors.

#mfc

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: mfc

It is an unofficial and free mfc ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official mfc.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con mfc

Observaciones

Microsoft Foundation Classes , o **MFC** , es una biblioteca que proporciona un envoltorio orientado a objetos alrededor de la API de Win32. Al encapsular la API Win32 "sin procesar" en las clases de C ++, MFC hace que sea mucho más fácil crear aplicaciones GUI y administrar recursos.

MFC ha existido desde hace mucho tiempo. Se introdujo por primera vez en 1992 con la versión 7 del compilador C / C ++ de Microsoft. En este momento, el desarrollo de C ++ estaba empezando a despegar. Las versiones posteriores de Visual Studio se han enviado con versiones significativamente mejoradas de MFC. Todavía se incluye con la última versión de Visual Studio 2015. Pero, lamentablemente, sus raíces heredadas son bastante visibles. Dado que la mayor parte se desarrolló antes de la estandarización del lenguaje C ++, las clases MFC no hacen uso completo de las características modernas de C ++ como las plantillas, proporcionan sus propias implementaciones personalizadas de otras funciones estándar de C ++ como RTTI, y utilizan muchos modismos no estándar . Estos hechos hacen que sea casi imposible compilar una aplicación MFC con cualquier compilador que no sea el de Microsoft. Sin embargo, en el lado positivo, MFC está bien integrado en Visual Studio, lo que hace que el desarrollo sea mucho más fácil.

Durante el desarrollo inicial, la biblioteca se conocía como Application Framework Extensions (abreviada a AFX). El departamento de marketing más tarde cambió su nombre a MFC, pero era demasiado tarde para cambiar cualquiera de los códigos, por lo que gran parte del código todavía hace referencia a "Afx" en lugar de "Mfc". Un ejemplo notable es el archivo de encabezado precompilado estándar que se genera automáticamente por Visual Studio: se llama `StdAfx.h` .

El 7 de abril de 2008, Microsoft lanzó una actualización de las clases MFC como una actualización fuera de banda para Visual Studio 2008 y MFC 9. La actualización presenta nuevas construcciones de interfaz de usuario, incluidas las cintas (similares a las de Microsoft Office 2007) y los widgets de UI asociados, las barras de herramientas totalmente personalizables, los paneles de acoplamiento (como Visual Studio 2005) que pueden flotar libremente o acoplarse a cualquier lado y las pestañas de documentos. La nueva funcionalidad se proporciona en nuevas clases para que las aplicaciones antiguas aún continúen ejecutándose. Esta actualización está construyendo en la parte superior de la BCGSoft Biblioteca BCGControlBar Professional Edition y fue nombrado como **MFC Feature Pack.**

Así que ahora MFC consta de dos bibliotecas con diferentes enfoques:

- Classic MFC (envoltorio para Win32 API).
- Paquete de características de MFC (conjunto mixto de controles de API de Win32 y nuevos controles de dibujo automático, como Ribbon).

Ver también:

- Aplicaciones de escritorio MFC (descripción general)
- Referencia MFC (referencia API)
- Paquete de características de MFC (descripción general)
- Referencia del paquete de características de MFC (referencia de la API del paquete de características)
- Muestras del paquete de características de MFC

Examples

Un programa básico de MFC

```
// Include the MFC header:
// (you do not need to and should not include the standard Windows headers, e.g.
// Windows.h)
#include <AfxWin.h>               // MFC core and standard components
// The following header defines resource constants, such as dialog and control IDs:
#include "resource.h"

// The basic element of an MFC application is a class that inherits from CWinApp.
class CMyApp : public CWinApp
{
    // This gets called as the application gets initialized.
    virtual BOOL InitInstance()
    {
        // Initialize a CDialog object to show in a moment.
        CDialog dlg(IDD_DIALOG1);
        // Display the dialog box as a modal dialog box.
        dlg.DoModal();

        // Return FALSE from this method to exit the application.
        return FALSE;
    }
};

// The one and only application object.
CMyWinApp theApp;
```

Resumen:

IDD_DIALOG1 debe ser el ID de un cuadro de diálogo definido en un archivo de recursos de proyecto creado por un editor de recursos, como el integrado en Visual Studio. (Un archivo de recursos generalmente tiene la extensión .rc). Para personalizar el comportamiento de un diálogo, puede derivar una nueva clase de CDialog.

Un cuadro de diálogo modal ejecuta su propio bucle de mensaje. La llamada "dlg.DoModal ();" no vuelve hasta que el diálogo haya sido cerrado por el usuario.

Si hubiéramos regresado TRUE desde InitInstance (), habría iniciado el bucle de mensajes de la aplicación. Esto se usa cuando tienes una aplicación más compleja, no basada en cuadros de diálogo.

Lea Empezando con mfc en línea: https://riptutorial.com/es/mfc/topic/2011/empezando-con-mfc

Capítulo 2: Barras de control acoplables (paneles)

Observaciones

Casi todas las aplicaciones MFC tienen barra de herramientas y barra de estado: tipos especiales de la barra de control que se acoplan a la parte superior e inferior del marco principal de la aplicación. Pero a menudo la lógica de la aplicación requiere más barras de información acopladas a algún lado del marco, por ejemplo, puede ser la barra de propiedades o la barra de clases, la barra de vista previa, la barra de salida y muchos otros. El MFC clásico tiene una buena solución solo para barras de herramientas y otros controles que no se pueden redimensionar dinámicamente. Pero MFC Feature Pack tiene un administrador de acoplamiento avanzado que permite:

- Dinamizar dinámicamente las barras acopladas.
- Redocking a cualquier lado del marco con vista previa en vivo.
- Serializar el estado y la posición de las barras acopladas en el registro.
- Acoplar las barras al mismo lado del marco y recogerlas en el panel con pestañas.

Todo lo que necesita es heredar su barra de acoplamiento de la clase CDockablePane .

Examples

Panel de acoplamiento al lado izquierdo del marco.

```
// Main frame class declaration
class CMainFrame
   : public CMDIFrameWndEx
{
    DECLARE_DYNAMIC(CMainFrame)

protected:
    // declare our pane
    CDockablePane m_wndPane;

    // .... other class memebers

public:
    CMainFrame();

protected:
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    DECLARE_MESSAGE_MAP()
};


// Main frame class implementation
IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWndEx)
```

```
BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWndEx)
    ON_WM_CREATE()
END_MESSAGE_MAP()

CMainFrame::CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CMDIFrameWndEx::OnCreate(lpCreateStruct) == -1)
        return -1;

    // enable frame docking to any sides
    EnableDocking(CBRS_ALIGN_ANY);

    // set the visual manager used to draw all user interface elements
    CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS(CMFCVisualManagerWindows));

    // enable smart docking style window behavior
    CDockingManager::SetDockingMode(DT_SMART);


    // Other frame initialization code
    // ....


    // Creating the pane.
    // ID_VIEW_PANE_ID - pane ID, must be declared in resource.h
    // CRect(0, 0, 100, 100) - default pane size in floating state (pane is not docked to any
frame sides).
    // CBRS_LEFT - default side for pane docking.
    if (!m_wndPane.Create(_T("My Pane"), this, CRect(0, 0, 100, 100), TRUE, ID_VIEW_PANE_ID,
WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS | WS_CLIPCHILDREN | CBRS_LEFT | CBRS_FLOAT_MULTI)) {
        TRACE0("Failed to create Pane\n");
        return -1; // failed to create
    }
    // Enable docking and redocking pane to frame any sides (you can pass a combination of
CBRS_ALIGN_ flags)
    m_wndPane.EnableDocking(CBRS_ALIGN_ANY);

    // Dock pane to the default (left) side of the frame.
    DockPane(&m_wndPane);

    return 0;
}
```

Acoplando los paneles al marco hijo.

Algunas veces la aplicación debe tener paneles que no se acoplen al marco principal, sino al marco secundario. Por lo general es una aplicación MDI. En el paquete de características de MFC, dicho marco secundario se hereda de la clase `CMDIChildWndEx` y como marco principal (heredado de `CMDIFrameWndEx`) tiene todo el código requerido para tal acoplamiento.

Pero hay algunos trucos para marco infantil. Y este ejemplo los muestra.

```
// Declare child frame
class CChildFrame : public CMDIChildWndEx
```

```
{
    DECLARE_DYNCREATE(CChildFrame)

protected:
    // declare our pane
    CDockablePane m_wndPane;

public:
    CChildFrame();

protected:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

    // CMDIChildWndEx class haven't serialization for state of the docking manager,
    // so we need to realize it manually.
    //
    // Docking state serialization methods:
    virtual void SaveBarState(LPCTSTR lpszProfileName) const;
    virtual void LoadBarState(LPCTSTR lpszProfileName);

protected:
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnDestroy();

    DECLARE_MESSAGE_MAP()
};



// CChildFrame Implementation

IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWndEx)

BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWndEx)
    ON_WM_CREATE()
    ON_WM_DESTROY()
END_MESSAGE_MAP()

CChildFrame::CChildFrame()
{
    // Trick#1: Add this line for enable floating toolbars
    m_bEnableFloatingBars = TRUE;
}

BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CMDIChildWndEx::PreCreateWindow(cs) )
        return FALSE;

    cs.style = WS_CHILD | WS_VISIBLE | WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU
        | FWS_ADDTOTITLE | WS_THICKFRAME | WS_MINIMIZEBOX | WS_MAXIMIZEBOX | WS_MAXIMIZE;

    // Trick#2: Add this line for remove the ugly client edge of the child frame.
    cs.dwExStyle &= (~WS_EX_CLIENTEDGE);

    return TRUE;
}

void CChildFrame::SaveBarState(LPCTSTR lpszProfileName) const
{
    const_cast<CChildFrame*>(this)->GetDockingManager()->SaveState(lpszProfileName);
```

```
    // Trick#3: we need to call serialization method of CMFCToolBar panes that may be docked
to the child frame.
    CObList list;
    const_cast<CChildFrame*>(this)->GetDockingManager()->GetPaneList(list, FALSE, NULL,
FALSE);
    if (list.GetCount() > 0) {
        POSITION pos = list.GetTailPosition();
        while (pos != NULL) {
            CMFCToolBar* pToolBar = DYNAMIC_DOWNCAST(CMFCToolBar, list.GetPrev(pos));
            if (pToolBar != nullptr) {
                pToolBar->SaveState(lpszProfileName);
            }
        }
    }
}

void CChildFrame::LoadBarState(LPCTSTR lpszProfileName)
{
    // Trick#3: we need to call serialization method of CMFCToolBar panes that may be docked
to the child frame.
    CObList list;
    GetDockingManager()->GetPaneList(list, FALSE, NULL, FALSE);
    if (list.GetCount() > 0) {
        POSITION pos = list.GetTailPosition();
        while (pos != NULL) {
            CMFCToolBar* pToolBar = DYNAMIC_DOWNCAST(CMFCToolBar, list.GetPrev(pos));
            if (pToolBar != nullptr) {
                pToolBar->LoadState(lpszProfileName);
            }
        }
    }

    GetDockingManager()->LoadState(lpszProfileName);
    GetDockingManager()->SetDockState();
    GetDockingManager()->ShowDelayShowMiniFrames(TRUE);

    // Trick#4: MFC BUGFIX: force assigning the child frame docking manager to all miniframes
(for details look at http://stackoverflow.com/q/39253843/987850).
    for (POSITION pos = GetDockingManager()->GetMiniFrames().GetHeadPosition(); pos != NULL;)
    {
        CWnd* pWndNext = (CWnd*)GetDockingManager()->GetMiniFrames().GetNext(pos);
        if (pWndNext != nullptr && pWndNext->IsKindOf(RUNTIME_CLASS(CPaneFrameWnd))) {
            STATIC_DOWNCAST(CPaneFrameWnd, pWndNext)->SetDockingManager(GetDockingManager());
        }
    }
}


int CChildFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    bool bRes = CMDIChildWndEx::OnCreate(lpCreateStruct) == 0;
    if (bRes)
    {
        // enable docking
        EnableDocking(CBRS_ALIGN_ANY);

        // enable Visual Studio 2005 style docking window behavior
        CDockingManager::SetDockingMode(DT_SMART);

        // Creating the pane.
```

```
        // ID_VIEW_PANE_ID - pane ID, must be declared in resource.h
        // CRect(0, 0, 100, 100) - default pane size in floating state (pane is not docked to
any frame sides).
        // CBRS_LEFT - default side for pane docking.
        if (!m_wndPane.Create(_T("My Pane"), this, CRect(0, 0, 100, 100), TRUE,
ID_VIEW_PANE_ID, WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS | WS_CLIPCHILDREN | CBRS_LEFT |
CBRS_FLOAT_MULTI)) {
            TRACE0("Failed to create Pane\n");
            return -1; // failed to create
        }
        // Enable docking and redocking pane to frame any sides (you can pass a combination of
CBRS_ALIGN_ flags)
        m_wndPane.EnableDocking(CBRS_ALIGN_ANY);

        // Dock pane to the default (left) side of the frame.
        DockPane(&m_wndPane);
    }

    // Loading dock manager state
    if (bRes) {
       LoadBarState(theApp.GetRegSectionPath(_T("ChildFrame")));
    }

    return bRes ? 0 : 1;
}

void CChildFrame::OnDestroy()
{
    // Save dock manager state
    SaveBarState(theApp.GetRegSectionPath(_T("ChildFrame")));

    CMDIChildWndEx::OnDestroy();
}
```

Lea Barras de control acoplables (paneles) en línea:
https://riptutorial.com/es/mfc/topic/6706/barras-de-control-acoplables--paneles-

Capítulo 3: Migración de la extensión ISAPI MFC (C ++) VS2005 DLL project a VS2015.

Introducción

Es posible que haya visto varios sitios web que muestren cómo crear un proyecto de extensión ISAPI, pero ninguno de ellos demostrará cómo migrar el proyecto existente de extensión ISAPI (VS2005) existente a VS2015. Me había enfrentado a un problema similar mientras trabajaba en uno de esos requisitos. Este artículo demuestra el camino experimental que había tomado para resolver mi problema.

Observaciones

Estaba trabajando en una tarea de migración en la que encontré un proyecto. El proyecto ISAPI heredado era un proyecto DLL de extensión MFC integrado en VS2005 y después de buscar en Google, me di cuenta de que las clases ISAPI MFC (CHttpServerContext, CHttpServer, etc.) no se enviarán después del VS2005. Microsoft recomienda usar las funciones de punto de entrada ISAPI del kit de desarrollo de software (SDK) de los Servicios de Internet Information Server (IK) de Microsoft en lugar de las clases ISAPI de MFC. Pensé que podría requerir una reescritura completa del proyecto. Sin embargo, afortunadamente, tomé un camino experimental que funcionó en mi caso. Pensé en escribir este artículo que podría ayudar a alguien a resolver el problema similar.

El propósito de este tema es demostrar los cambios necesarios para eliminar la dependencia de MFC de la DLL de extensión ISAPI de MFC sin tener que volver a escribir el código completo. El ejemplo que se muestra arriba no contiene la implementación completa, es solo para fines de demostración.

Examples

Ejemplo:

Propósito del proyecto: descargar un archivo particular basado en la identificación del mensaje enviado como una solicitud del cliente al servidor. El archivo se creará en el servidor y se transmitirá al cliente.

Aquí, no he compartido todo el código, pero he mostrado lo que es necesario ya que el propósito es demostrar los cambios necesarios para la migración.

Para hacer los cambios. Tomé la referencia de afxisapi.h, afxisapi.inl y HttpExt.h existentes que se envían con VS2005.

- Use EXTENSION_CONTROL_BLOCK en lugar de CHttpServerContext. EXTENSION_CONTROL_BLOCK es una estructura principal que utiliza IIS y la extensión

ISAPI para intercambiar la información. Contiene toda la información sobre nueva solicitud.

- Elimine la dependencia de la clase CHttpServer de su clase de extensión MFC. Elimine la asignación de mensajes de análisis de comandos http si se utiliza en su proyecto. Las macros como ON_PARSE_COMMAND, BEGIN_PARSE_MAP, END_PARSE_MAP y DEFAULT_PARSE_COMMAND se eliminaron después de VS2005.
- Implemente sus propias funciones GetExtensionVersion, TerminateExtension y HttpExtensionProc en su clase de extensión MFC que se derivó previamente de la clase CHttpSever. Estas funciones
- GetExtensionVersion () La función GetExtensionVersion es la primera función de punto de entrada en IIS. Esta función le permite a su extensión ISAPI registrar su información de versión con IIS.

```
BOOL  CISAPIExtension::GetExtensionVersion( HSE_VERSION_INFO* pVer )
{
    ISAPIVERIFY(
       ::LoadString( AfxGetResourceHandle(),
                           IDS_SERVER,
                           sz,
                           HSE_MAX_EXT_DLL_NAME_LEN) );

       _tcscpy_s( pVer->lpszExtensionDesc, ( strlen( sz ) + 1U ), sz )
}
```

- HttpExtensionProc (): la función HttpExtensionProc es el punto de entrada principal para una extensión ISAPI llamada por IIS. Expone los métodos que utiliza IIS para acceder a la funcionalidad expuesta por la extensión. Extraiga la información de la solicitud del bloque de control de extensión y haga su procesamiento.

```
DWORD CISAPIExtension::HttpExtensionProc(LPEXTENSION_CONTROL_BLOCK pECB)
{
try
{
    if (strcmp(pECB->lpszMethod, "POST") == 0)
    {
        if (0 < pECB->cbTotalBytes)
        {
            unsigned long ulSize = pECB->cbTotalBytes;
            char* lpszContent = new char[ulSize + 1];
            if (NULL != lpszContent)
            {
                memcpy(lpszContent, pECB->lpbData, ulSize);
                lpszContent[ulSize] = '\0';

                std::string message_id;
                // Extract Message id from lpszContent using string operation.

                if(strcmp(pECB->lpszQueryString, "DownloadFile") == 0)
                {
                    DownloadFileFunction(pECB, message_id));
                }

                delete[] lpszContent;
```

```
                lpszContent = NULL;
            }
        }
    }
}
catch (...)
{

}
return HSE_STATUS_SUCCESS;
}
```

- Use ServerSupportFunction de EXTENSION_CONTROL_BLOCK en lugar de la función TransmitFile de CHttepServerContext. Anteriormente, TransmitFile () se utilizaba para la transferencia de archivos.
- Use HSE_REQ_TRANSMIT_FILE en ServerSupportFunction para la transferencia de archivos.
- Tire de la información del archivo de transmisión en la estructura HSE_TF_INFO.
- Cree una función de devolución de llamada (AsyncIOCompletionFunction) que se llamará en la finalización asíncrona de E / S.

```
void WINAPI  AsyncIOCompletionFunction(
EXTENSION_CONTROL_BLOCK * pECB,
PVOID    pContext,
DWORD    cbIO,
DWORD    dwError)
{
    HSE_CUSTOM_ERROR_INFO* pErrorInfo = (HSE_CUSTOM_ERROR_INFO*)pContext;
    DWORD dwIOStatus = dwError == NO_ERROR ?
        HSE_STATUS_SUCCESS :
        HSE_STATUS_ERROR;

    if (NULL != pErrorInfo)
    {
        //
        // log HTTP status code
        //
        pECB->dwHttpStatusCode = atoi(pErrorInfo->pszStatus);

        delete pErrorInfo;
        pErrorInfo = NULL;
    }

    pECB->ServerSupportFunction(
        pECB->ConnID,
        HSE_REQ_DONE_WITH_SESSION,
        &dwIOStatus,
        (LPDWORD)NULL,
        (LPDWORD)NULL);
}
```

Fragmento de código para DownloadFileFunction ()

```
HSE_TF_INFO pHSEInfo;

pHSEInfo.hFile            = hFile;
```

```
pHSEInfo.dwFlags          = HSE_IO_ASYNC;
pHSEInfo.pHead            = (PVOID) pvHeader;
pHSEInfo.HeadLength       = dwHeaderLen;
pHSEInfo.pContext         = 0;
pHSEInfo.pTail            = 0;
pHSEInfo.TailLength       = 0;
pHSEInfo.BytesToWrite     = 0;
pHSEInfo.Offset           = 0;
pHSEInfo.pfnHseIO         = AsyncIOCompletionFunction; // callback function that will     be
called on asynchronous I/O completion
pHSEInfo.pszStatusCode    = LPCSTR("200 OK");

bResult = m_pCtxt->ServerSupportFunction(m_pCtxt->ConnID, HSE_REQ_TRANSMIT_FILE, &pHSEInfo, 0,
0);
```

Lea Migración de la extensión ISAPI MFC (C ++) VS2005 DLL project a VS2015. en línea: https://riptutorial.com/es/mfc/topic/10026/migracion-de-la-extension-isapi-mfc--c-plusplus--vs2005-dll-project-a-vs2015-

Capítulo 4: Multihilo

Observaciones

MFC admite subprocesos de trabajo y subprocesos de interfaz gráfica de usuario (subprocesos con bucles de mensajes). Consulte https://msdn.microsoft.com/en-us/library/975t8ks0.aspx para obtener más documentación.

Examples

Ejemplo simple de subproceso de trabajo de AfxBeginThread

Este ejemplo muestra una llamada de AfxBeginThread que inicia el subproceso de trabajo y un procedimiento de ejemplo de subproceso de trabajo para ese subproceso.

```
// example simple thread procedure.
UINT __cdecl threadProc(LPVOID rawInput)
{
    // convert it to the correct data type. It's common to pass entire structures this way.
    int* input = (int*)rawInput;
    // TODO: Add your worker code...
    MessageBox(0,"Inside thread!",0,0);
    // avoid memory leak.
    delete input;
    return 0;
}
// ...
// somewhere that gets called when you want to start the thread...
int* input = new int;
*input = 9001;
AfxBeginThread(threadProc, input);
// after this, the message box should appear, and the rest of your code should continue
// running.
```

Lea Multihilo en línea: https://riptutorial.com/es/mfc/topic/6577/multihilo

Creditos

S. No	Capítulos	Contributors
1	Empezando con mfc	23W, aquirdturtle, Cody Gray, Community
2	Barras de control acoplables (paneles)	23W
3	Migración de la extensión ISAPI MFC (C ++) VS2005 DLL project a VS2015.	Rockford
4	Multihilo	aquirdturtle