eBook Gratuit

APPRENEZ mfc

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#mfc

Table des matières

À propos

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

Chapitre 1: Commencer avec mfc

Remarques

Microsoft Foundation Classes ou **MFC** est une bibliothèque qui fournit un wrapper orienté objet autour de l'API Win32. En encapsulant l'API Win32 "brute" dans les classes C ++, MFC facilite considérablement la création d'applications d'interface graphique et la gestion des ressources.

MFC existe depuis très longtemps. Il a été introduit en 1992 avec la version 7 du compilateur C / C ++ de Microsoft. A cette époque, le développement C ++ commençait à peine à prendre son envol. Les versions ultérieures de Visual Studio ont été fournies avec des versions considérablement améliorées de MFC. Il est toujours inclus dans la dernière version de Visual Studio 2015. Mais ses racines héritées sont malheureusement tout à fait visibles. Comme la plupart ont été développés avant la standardisation du langage C ++, les classes MFC n'utilisent pas pleinement les fonctionnalités modernes du C ++ telles que les modèles, fournissent leurs propres implémentations personnalisées d'autres fonctionnalités C ++ standard telles que RTTI et utilisent de nombreux idiomes non standard . Ces faits rendent presque impossible la compilation d'une application MFC avec un compilateur autre que celui de Microsoft. Du côté des avantages, MFC est bien intégré à Visual Studio, ce qui facilite considérablement le développement.

Au début du développement, la bibliothèque était appelée Application Framework Extensions (en abrégé AFX). Le service marketing a ensuite changé son nom pour MFC, mais il était trop tard pour en modifier le code, une grande partie du code faisant toujours référence à "Afx" au lieu de "Mfc". Un exemple notable est le fichier d'en-tête pré-compilé standard généré automatiquement par Visual Studio: il s'appelle `StdAfx.h` .

Le 7 avril 2008, Microsoft a publié une mise à jour des classes MFC en tant que mise à jour hors bande de Visual Studio 2008 et MFC 9. La mise à jour comporte de nouvelles constructions d'interface utilisateur, notamment les rubans (similaires à Microsoft Office 2007). des widgets d'interface utilisateur associés, des barres d'outils entièrement personnalisables, des panneaux d'ancrage (comme Visual Studio 2005) qui peuvent être librement flottés ou ancrés sur n'importe quel côté et onglets du document. La nouvelle fonctionnalité est fournie dans les nouvelles classes afin que les anciennes applications continuent à s'exécuter. Cette mise à jour est basée sur BCGControlBar Library Professional Edition de BCGSoft et a été nommée **MFC Feature Pack** .

Donc, maintenant, MFC est composé de deux bibliothèques avec des approches différentes:

- Classic MFC (wrapper pour Win32 API).
- MFC Feature Pack (ensemble mixte de contrôles API Win32 et de nouveaux contrôles de dessin automatique, tels que le ruban).

Voir également:

- Applications de bureau MFC (vue d'ensemble)
- Référence MFC (référence API)

- Feature Pack MFC (aperçu)
- MFC Feature Pack Reference (Référence de l' API Feature Pack)
- Exemples de packs MFC

Examples

Un programme de base 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;
```

Résumé:

IDD_DIALOG1 doit être l'ID d'une boîte de dialogue définie dans un fichier de ressources de projet créé par un éditeur de ressources, tel que celui intégré à Visual Studio. (Un fichier de ressources a généralement l'extension .rc.) Pour personnaliser le comportement d'une boîte de dialogue, vous pouvez dériver une nouvelle classe à partir de CDialog.

Une boîte de dialogue modale exécute sa propre boucle de messages. L'appel "dlg.DoModal ();" ne retourne pas tant que la boîte de dialogue n'a pas été fermée par l'utilisateur.

Si nous avions renvoyé TRUE à partir d'InitInstance (), la boucle de messages de l'application aurait été lancée. Ceci est utilisé lorsque vous avez une application plus complexe et sans dialogue.

Lire Commencer avec mfc en ligne: https://riptutorial.com/fr/mfc/topic/2011/commencer-avec-mfc

Chapitre 2: Barres de contrôle ancrables (vitres)

Remarques

Presque toutes les applications MFC ont une barre d'outils et une barre d'état - des types spéciaux de la barre de contrôle ancrés dans la partie supérieure et inférieure du cadre principal de l'application. Mais souvent, la logique de l'application nécessite davantage de barres d'information ancrées sur un côté du cadre, par exemple la barre de propriétés ou la barre de classes, la barre de prévisualisation, la barre de sortie et bien d'autres. Classic MFC ont une bonne solution que pour les barres d'outils et d'autres contrôles qui ne peuvent pas être redimensionnés dynamiquement. Mais MFC Feature Pack dispose d'un gestionnaire d'amarrage avancé qui permet:

- Redimensionner dynamiquement les barres ancrées.
- Recomposition sur n'importe quel côté de l'image avec aperçu en direct.
- Sérialiser l'état et la position des barres ancrées dans le registre.
- Les barres de fixation se trouvent du côté de la même image et sont collectées dans le volet à onglets.

Tout ce dont vous avez besoin est d'hériter votre barre d' ancrage de la classe CDockablePane .

Examples

Volet d'accueil sur le côté gauche du cadre.

```
// 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;
}
```

Ancrage des volets dans le cadre Enfant.

Parfois, l'application doit avoir des volets ancrés non pas dans le cadre principal, mais dans le cadre enfant. C'est généralement l'application MDI. Dans le pack de fonctions MFC, un tel cadre enfant est hérité de la classe `CMDIChildWndEx` et, en tant que cadre principal (hérité de `CMDIFrameWndEx`), tous les codes sont requis pour un tel ancrage.

Mais il y a quelques astuces pour la trame enfant. Et cet exemple les montre.

```
// 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();
}
```

Lire Barres de contrôle ancrables (vitres) en ligne: https://riptutorial.com/fr/mfc/topic/6706/barres-de-controle-ancrables--vitres-

Chapitre 3: Migration du projet DLL d'extension C ++ (ISAPI MFC) VS2005 vers VS2015.

Introduction

Vous avez peut-être vu plusieurs sites Web qui montreront comment créer un projet d'extension ISAPI, mais aucun ne démontrera comment migrer le projet existant d'extension ISAPI (VS2005) vers VS2015. J'avais fait face à un problème similaire pendant que je travaillais sur l'une de ces exigences. Cet article démontre la manière expérimentale que j'avais prise pour résoudre mon problème.

Remarques

Je travaillais sur une tâche de migration où je suis tombé sur un projet. Le projet ISAPI hérité était un projet DLL d'extension MFC intégré à VS2005 et, après quelques recherches sur Google, j'ai appris que les classes ISAPI MFC (CHttpServerContext, CHttpServer etc.) ne sont pas livrées après VS2005. Microsoft recommande d'utiliser les fonctions ISAPI Entry-Point à partir du kit de développement logiciel (SDK) Microsoft Internet Information Services (IIS) au lieu des classes ISAPI MFC. Je pensais que cela pourrait nécessiter une réécriture complète du projet. Heureusement, j'ai pris un chemin expérimental qui a fonctionné dans mon cas. J'ai pensé écrire cet article qui pourrait aider quelqu'un à résoudre le problème similaire.

L'objet de cette rubrique est de démontrer les modifications requises pour supprimer la dépendance de MFC de la DLL d'extension ISAPI MFC sans réécriture du code complet. L'exemple ci-dessus ne contient pas d'implémentation complète, mais uniquement à des fins de démonstration.

Examples

Exemple:

Objectif du projet: Télécharger un fichier particulier en fonction de l'ID du message envoyé en tant que requête du client au serveur. Le fichier sera créé sur le serveur et transmis au client.

Ici, je n'ai pas partagé tout le code, mais j'ai montré ce qui est nécessaire car l'objectif est de démontrer les modifications requises pour la migration.

Pour faire les changements J'ai pris la référence des fichiers afxisapi.h, afxisapi.inl et HttpExt.h existants qui sont livrés avec VS2005.

- Utilisez EXTENSION_CONTROL_BLOCK au lieu de CHttpServerContext. EXTENSION_CONTROL_BLOCK est une structure principale utilisée par IIS et l'extension

ISAPI pour échanger les informations. Il contient toutes les informations sur la nouvelle demande.

- Supprimez la dépendance de la classe CHttpServer de votre classe d'extension MFC. Supprimez le mappage des messages d'analyse des commandes http s'il est utilisé dans votre projet. Des macros comme ON_PARSE_COMMAND, BEGIN_PARSE_MAP, END_PARSE_MAP et DEFAULT_PARSE_COMMAND étaient en cours de suppression après VS2005.

- Implémentez vos propres fonctions GetExtensionVersion, TerminateExtension et HttpExtensionProc dans votre classe d'extension MFC précédemment dérivée de la classe CHttpSever. Ces fonctions

- GetExtensionVersion () La fonction GetExtensionVersion est la première fonction de point d'entrée dans IIS. Cette fonction permet à votre extension ISAPI d'enregistrer ses informations de version avec 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 fonction HttpExtensionProc est le point d'entrée principal d'une extension ISAPI appelée par IIS. Il expose les méthodes utilisées par IIS pour accéder aux fonctionnalités exposées par l'extension. Extrayez les informations de demande du bloc de contrôle d'extension et effectuez votre traitement.

```
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;
}
```

- Utilisez ServerSupportFunction de EXTENSION_CONTROL_BLOCK au lieu de la fonction TransmitFile de CHttepServerContext. Auparavant, TransmitFile () était utilisé pour le transfert de fichiers.
- Utilisez HSE_REQ_TRANSMIT_FILE dans ServerSupportFunction pour le transfert de fichiers.
- Tirez les informations du fichier de transmission dans la structure HSE_TF_INFO.
- Créez une fonction de rappel (AsyncIOCompletionFunction) qui sera appelée lors de l'achèvement des E / S asynchrones.

```
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);
}
```

Extrait de code pour 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);
```

Lire Migration du projet DLL d'extension C ++ (ISAPI MFC) VS2005 vers VS2015. en ligne: https://riptutorial.com/fr/mfc/topic/10026/migration-du-projet-dll-d-extension-c-plusplus--isapi-mfc--vs2005-vers-vs2015-

Chapitre 4: Multithreading

Remarques

MFC prend en charge les threads de travail et les threads d'interface graphique (threads avec des boucles de message). Voir https://msdn.microsoft.com/en-us/library/975t8ks0.aspx pour plus de documentation.

Examples

Exemple simple de thread de travail AfxBeginThread

Cet exemple montre un appel de AfxBeginThread qui démarre le thread de travail et un exemple de procédure de thread de travail pour ce thread.

```
// 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.
```

Lire Multithreading en ligne: https://riptutorial.com/fr/mfc/topic/6577/multithreading

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec mfc	23W, aquirdturtle, Cody Gray, Community
2	Barres de contrôle ancrables (vitres)	23W
3	Migration du projet DLL d'extension C ++ (ISAPI MFC) VS2005 vers VS2015.	Rockford
4	Multithreading	aquirdturtle