

COMを応用してExcelをコントロール

えムナウ (児玉宏之)



<http://mnow.jp/>

<http://mnow.wankuma.com/>

<http://blogs.wankuma.com/mnow/>

<http://www.ailight.jp/blog/mnow/>

.NET UX Lab

.Net ユーザーエクスペリエンス研究所

えムナウのC#

プログラミングのページ



わんくま同盟 横浜勉強会 #02

アジェンダ

- 基本をおさらいしてみる
 - OLEとCOM
 - COMのインターフェース
- Excelを内部に表示するプログラムの作成
- Excelに制約をつけていこう

OLEとCOM

- OLE (Object Linking and Embedding)
 - アプリケーションソフト間でデータを転送・共有するための仕組み
 - OLEサーバー
 - アプリケーションの機能提供側
 - OLEコンテナ
 - 機能を利用するほう
 - OLEコントロール (ActiveXコントロール)
 - OLEサーバーのみで小型でシンプルな機能、インターネットに対応することでActiveXコントロールに改名

OLEとCOM

- COM (Component Object Model)
 - COMはOLE、OLEオートメーション、OCX、ActiveX、COM+、DCOMをカバーする包括的な用語
 - 部品化されたプログラムを作成・利用するための基盤となる技術
 - データのやりとりができるプログラム部品のこと COMコンポーネント、ネットワークを通じて別のコンピュータ上にあるCOMコンポーネントを利用する技術がDCOM
 - Windowsだけでなく、UNIXやMac OSにも移植

OLEとCOM

- Excelの実行ファイルはCOMです。
- Microsoft Windows SDK 付属の OLE-COM Object Viewer で見てみる。
 - Microsoft Excel Application
 - IConnectionPointContainer
 - IDispatch
 - ISupportErrorInfo
 - IUnknown



- Microsoft DocProp Rating Cont
- Microsoft DocProp Static Prop
- Microsoft DTC Transaction
- Microsoft DTC Transaction Ma
- Microsoft DTC Transaction Ma
- Microsoft DTC Transaction Un
- Microsoft DTV-DVD Audio Dec
- Microsoft DTV-DVD Video Dec
- Microsoft DVBC Network Prov
- Microsoft DVBS Network Prov
- Microsoft DVBT Network Prov
- Microsoft Embedded Common
- Microsoft Embedded Common
- Microsoft Embedded Common
- Microsoft Embedded Search G
- Microsoft Embedded Search G
- Microsoft Equation 2.0
- Microsoft Excel Application**
 - _Global
 - _Global
 - IConnectionPointContainer
 - IDispatch
 - ISupportErrorInfo
 - IUnknown
- Microsoft Expression Web
- Microsoft Expression Web 12.0
- Microsoft Expression Web 12.0
- Microsoft Expression Web 12.0
- Microsoft Expression Web 12.0
- Microsoft Expression Web 12.0

No icon Available
 Microsoft Excel Application
 {00020812-0000-0000-C000-000000000046}

Registry Implementation Activation Launch Permissions Access Permissions

CLSID =

```

{00020812-0000-0000-C000-000000000046} = Microsoft Excel Application
  InprocServer32 [Assembly] = Microsoft.Office.Interop.Excel, Version=12.0.0.0, Culture=neutral, PublicKeyToken=7b7c5947-174f-4222-8b76-78477383e9c4
  InprocServer32 [Class] = Microsoft.Office.Interop.Excel.GlobalClass
  InprocServer32 [RuntimeVersion] = v1.1.4322
    12.0.0.0 [Assembly] = Microsoft.Office.Interop.Excel, Version=12.0.0.0, Culture=neutral, PublicKeyToken=7b7c5947-174f-4222-8b76-78477383e9c4
    12.0.0.0 [Class] = Microsoft.Office.Interop.Excel.GlobalClass
    12.0.0.0 [RuntimeVersion] = v1.1.4322
  LocalServer [LocalServer] = w_1^VV!!!!!!!!!!!!!!MKKSkEXCELFiles>tw{~$4Q}c@II=l2xaTO5 /automation
  LocalServer [<no name>] = C:%PROGRAMFILES%Microsoft Office12\EXCEL\EXCEL.EXE /automation
  LocalServer32 [LocalServer32] = w_1^VV!!!!!!!!!!!!!!MKKSkEXCELFiles>tw{~$4Q}c@II=l2xaTO5 /automation
  LocalServer32 [<no name>] = C:%PROGRAMFILES%Microsoft Office12\EXCEL\EXCEL.EXE /automation
  
```

COMのインターフェース

- IUnknown
 - QueryInterface
 - インターフェイス識別子で識別されるインターフェイスポインタを返す
 - AddRef
 - インターフェイスの参照カウントをインクリメントする。
 - Release
 - インターフェイスの参照カウントをデクリメントする。
- Release不足によりCOMが使っていないのに残ってしまう問題が発生する。

COMのインターフェース

- IDispatch

- IUnkownの構成要素QueryInterface/AddRef/Releaseに以下のものが追加される。
- GetTypeInfo/GetTypeInfoCount
 - オブジェクトの型情報を取得/オブジェクトが提供する型情報インターフェイスの数 (0 または 1) を取得
- GetIDsOfNames
 - 名前(文字列)からDISPIDを取得
- Invoke
 - DISPIDや引数を使いプロパティおよびメソッドにアクセス

COMのインターフェース

- IConnectionPointContainer
 - アプリケーションが外に出す関数(例えばイベント)を一覧や検索できる
 - 見つかったConnectionPointに対してIDispatchを接続すれば、関数の呼び出し(イベント発生)時にIDispatchのInvokeメソッドが呼ばれる

Excelを内部に表示するプログラムの作成

- Excelを埋め込むアプリケーションを作る
 - ExcelはOLEサーバーとしてもOLEコンテナとしても動作する
 - プロジェクトをOLEコンテナとして作成すればExcelをOLEサーバー（OLEオートメーションとも言う）として内部に表示することができる

Excelを内部に表示するプログラムの作成

- MFC アプリケーションを作成する
 - シングルドキュメント
 - プロジェクト形式: MFC 標準
 - 視覚スタイルと色: 規定
 - 複合ドキュメント サポート: コンテナ
 - データベースサポート: なし
 - コマンドバー: クラシックメニューを使用する
 - 高度な機能: コモンコントロール マニフェスト

Excelを内部に表示するプログラムの作成

- 出来上がったファイル
 - stdafx
 - CxxApp CWinAppを継承
 - CMainFrame CFrameWndを継承
 - CxxDoc COleDocumentを継承
 - CxxView CViewを継承
 - CntrlItem COleDocObjectItemを継承

Excelを内部に表示するプログラムの作成

- 既存のExcelファイルを初期読み込み
 - CxxDoc::OnNewDocument

```
CString filepath("C:¥¥Users¥¥Public¥¥Documents¥¥  
    Demo¥¥OleDemo¥¥ExTest¥¥Text.xlsx");  
m_pltem = new CxxCntrlItem(this);  
m_pltem->CreateFromFile(filepath);
```

Excelを内部に表示するプログラムの作成

- 既存のドキュメントをView内部に表示
 - CxxView::OnInitialUpdate

```
COleDocument* pDoc = (COleDocument*) GetDocument();
if (pDoc != NULL) {
    POSITION posItem = pDoc->GetStartPosition();
    if (posItem != NULL) {
        CDocItem* pItem = pDoc->GetNextItem(posItem);
        COleDocObjectItem *pDocObjectItem =
            DYNAMIC_DOWNCAST(COleDocObjectItem, pItem);
        if (pDocObjectItem != NULL) {
            pDocObjectItem->DoVerb(OLEIVERB_SHOW, this);
        }
        m_pSelection = DYNAMIC_DOWNCAST(CExTestCntrlItem, pItem);
        pDoc->UpdateAllViews(NULL);
    }
}
```



Excelを内部に表示するプログラムの作成

- COleClientItem::CreateFromFile
 - ファイルを使って埋め込みアイテムを作成
- COleClientItem::DoVerb
 - サーバーアプリケーションが起動され、OLE アイテムが編集できるようになる
 - OLEIVERB_SHOW : 可能ならView内で開く
 - OLEIVERB_OPEN : 別Windowで開く

Excelを内部に表示するプログラムの作成

• Excelのヘッダーを作成する

```
#import "C:\Program Files\Common Files\Microsoft Shared\Office12\MSO.DLL"  
rename("DocumentProperties", "DocumentPropertiesXL")  
rename("RGB", "MSO_RBGXL")
```

```
#import "C:\Program Files\Common Files\Microsoft  
Shared\VBA\VBA6\Vbe6ext.olb"
```

```
#import "C:\Program Files\Microsoft Office\Office12\EXCEL.EXE"  
rename("ReplaceText", "ReplaceTextXL")  
rename("CopyFile", "CopyFileXL")  
rename("DialogBox", "DialogBoxXL")  
rename("RGB", "RBGXL")  
rename("DocumentProperties", "DocumentPropertiesXL")  
rename("Font", "FontXL")  
rename("Picture", "PictureXL")  
exclude("IFont") exclude("IPicture")  
no_dual_interfaces
```



Excelを内部に表示するプログラムの作成

- Excelのヘッダーとは
 - ExcelVbaメソッド呼び出しと同じ形式で呼び出し
てくれる為の QueryInterface や Invoke を隠す
実装とそのヘッダー
 - mso.tlh mso.thi
 - vbe6ext.tlh vbe6ext.thi
 - excel.tlh excel.tli

Excelを内部に表示するプログラムの作成

- Excelのインスタンスを取得する
 - COleClientItem::m_IpObject から

```
LPOLELINK IpOleLink = NULL;
m_IpObject->QueryInterface(IID_IOleLink, (LPVOID FAR*)&IpOleLink);
IpUnk = NULL;
if (IpOleLink->GetBoundSource(&IpUnk) != NOERROR)
{
    TRACE0("Warning: Link is not connected!¥n");
    IpOleLink->Release();
    return NULL;
}
IpOleLink->Release();

LPDISPATCH IpDispatch = NULL;
IpUnk->QueryInterface(IID_IDispatch, (void**)&IpDispatch);

_Workbook wb;
wb.AttachDispatch(IpDispatch);
_Application app;
app = wb.GetApplication();
:
:
IpDispatch->Release();
```



Excelを内部に表示するプログラムの作成

- Excelヘッダーを使ってプログラムするには
 - オブジェクトの意味やメソッドの意味がわからないとヘッダーだけではできない
 - Excelヘッダーにはクラスやメソッドやパラメータの意味も使い方の例も載っていない
 - Excelの開発用VisualBasicのHELPを参照してExcelヘッダーの使い方を推測することになる
 - パラメータの使わない部分は省略値としてvtMissing を利用する

Excelを内部に表示するプログラムの作成

- excel.tlh

- Excelの開発タブからVisualBasicを表示してHelpを出すと開発者用リファレンスが表示される
- HelpのExcel 2007 開発者用リファレンスーリファレンスに excel.tlh に記述されているオブジェクトが一覧で表示される
- 参照したいオブジェクトをクリックしてさらにメンバーを確認すればメソッドやプロパティやイベントが確認できる

Excelを内部に表示するプログラムの作成

- mso.tlh

- Excelの開発タブからVisualBasicを表示してHelpを出すと開発者用リファレンスが表示される
- HelpのExcel 2007 開発者用リファレンスー2007 Microsoft Office system オブジェクト ライブラリリファレンスーリファレンスに mso.tlh に記述されているオブジェクトが一覧で表示される
- 参照したいオブジェクトをクリックしてさらにメンバーを確認すればメソッドやプロパティやイベントが確認できる

Excelに制約をつけていこう

- メニューを非表示にする
- OLEサーバー(Excel)のメニューをOLEコンテナ(作成プロジェクト)に挿入変更削除する呼び出しを無処理にするとExcelメニューが出ない
 - COleClientItem::OnInsertMenus
 - COleClientItem::OnSetMenu
 - COleClientItem::OnRemoveMenus

Excelに制約をつけていこう

- ツールバーを非表示にする
 - すべて非表示にするなら _Application から Toolbars を取得して配列の1から個数分 Toolbar を取得し Visible プロパティを VARIANT_FALSE に設定する
 - Toolbar の Name プロパティを確認すればひとつのツールバーについて非表示にできる

Excelに制約をつけていこう

- 入力できる文字を制限する
 - 例えば特定のコントロールキーを無効にしたい場合など
 - user32.dll の SetWindowsHookEx 関数を WH_KEYBOARD と WindowHandle とフックルーチンのアドレス をパラメータとしてフックすればExcelでキー入力があったときにフックルーチンがコールされる
 - WH_KEYBOARD_LL は全てのプロセスのキー入力がフックできる

Excelに制約をつけていこう

- マウスクリックを制限する
 - 右クリックでコンテキストメニューが表示されることを防止する
 - user32.dll の SetWindowsHookEx 関数を WH_MOUSE と WindowHandle とフックルーチンのアドレスをパラメータとしてフックすれば Excelでマウス操作があったときにフックルーチンがコールされる
 - WH_MOUSE_LL は全てのプロセスのマウス操作がフックできる

Excellに制約をつけていこう

- user32.dll の SetWindowsHookEx 関数は注意して使用しないとOSの誤動作につながるなので注意深く実装する必要がある
- ウィルス監視をするプログラムが user32.dll の SetWindowsHookEx 関数を監視して警告を発生する場合がありますので注意が必要

Excelに制約をつけていこう

- Excelのイベントを拾う

- Excelのイベントで保存・印刷・右クリックなどを抑止できる
- Excel のイベントを拾うクラスを作成する
- IConnectionPointContainer を利用して Excel の Application とイベントを拾うクラスを結ぶ
- Excel のイベントが入るとイベントを拾うクラスの Invoke が呼ばれるので DISPID を識別してイベントハンドラを書く
- 終了時には IConnectionPointContainer で結んだのを解除する

Excelに制約をつけていこう

- Excel のイベントを拾うクラス
 - IDispatch を継承して QueryInterface / AddRef / Release / Invoke を実装する
 - QueryInterface は IID_IDispatch / IID_Iunknown / IID_IExcelEventSink で自分を返す
 - Invoke はイベントハンドラ
 - GetTypeInfo / GetTypeInfoCount / GetIDsOfNames は E_NOTIMPL を返して未実装にする

Excelに制約をつけていこう

- Excel の Application とイベントを拾うクラスを結ぶ

```
IConnectionPointContainer *pConnPtContainer;  
app.QueryInterface(IID_IConnectionPointContainer,  
    (void **)&pConnPtContainer);  
pConnPtContainer->FindConnectionPoint(IID_IExcelEventSink,  
    &pConnectionPoint);  
pConnectionPoint->Advise(this, &adviseCookie);  
pConnPtContainer->Release();
```

Excelに制約をつけていこう

- DISPID を識別してイベントハンドラを書く

```
HRESULT _stdcall ExcelEventSink::Invoke(  
    DISPID dispIdMember, REFIID riid, LCID lcid, WORD wFlags,  
    DISPPARAMS *pDispParams, VARIANT *pVarResult, EXCEPINFO *pExcepInfo, UINT *puArgErr)  
{  
    if (dispIdMember == 0x624) // WorkbookBeforePrint  
    {  
        Excel::_WorkbookPtr pWorkbook = pDispParams->rgvarg[0].pdispVal;  
        if (m_pWorkbook != NULL)  
        {  
            if (m_pWorkbook == pWorkbook)  
            {  
                VARIANT_BOOL * Cancel = pDispParams->rgvarg[1].pboolVal;  
                *Cancel = VARIANT_TRUE;  
            }  
        }  
    }  
    return S_OK;  
}
```



Excelに制約をつけていこう

- 以下のように順を追って制限をした
 - メニューを非表示にする
 - ツールバーを非表示にする
 - 入力できる文字を制限する
 - マウスクリックを制限する
 - Excelのイベントを拾って操作をキャンセルする
- COMやOLEを使ってVisualBasicで操作する
以上のことができるのを理解してもらえただろうか

まとめ

- Excel を題材にして OLEとCOM の深遠なる世界の一端を見てきました
- 今回はなるべくラッパーの少ない道を通ってきましたが最小限のラッパー(Excelヘッダー)は使っています
- VisualBasicやC#でExcelをコントロールする方は今後もいると思いますがこんな世界が基礎としてあることを理解してほしいと思います。