Articles » Platforms, Frameworks & Libraries » COM / COM+ » COM

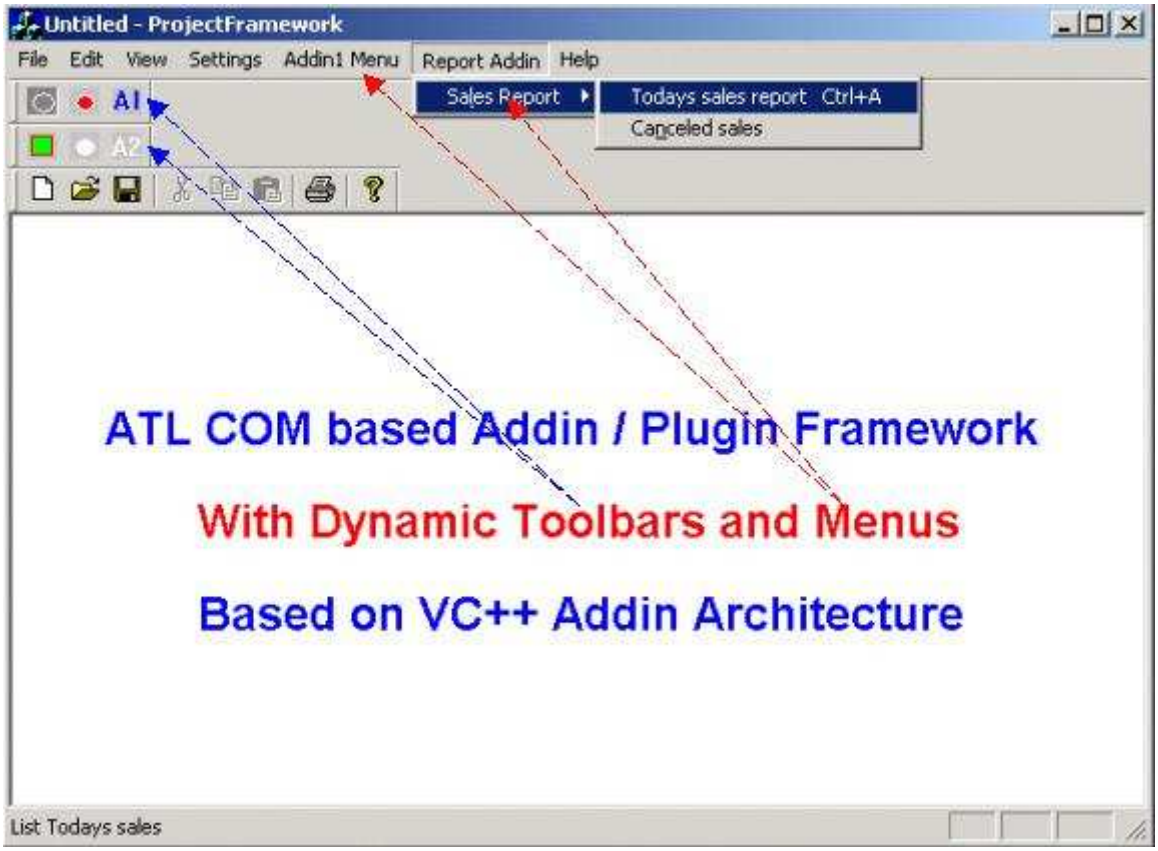
ATL COM Based Addin / Plugin Framework With Dynamic Toolbars and Menus

thomas_tom99, 10 Dec 2004

An article on ATL COM Based Addin / Plugin Framework With Dynamic Toolbars and Menus, based on VC++ Addin Architecture.

Download demo project - (Addin_Project_Framework_Bin.zip) 90Kb

Download source - (Addin_Project_Framework_Src.zip) 338 Kb



Links

- **Len Holgate's** Writing extensible applications

- **Len Holgate's** Component Category Manager wrapper classes
- Pluggable Components using Component Categories - Part 1 (**By Zac Howland**)
- Pluggable Components using Component Categories - Part II (**By Zac Howland**)
- **Roger Allen's** An MFC extension library to enable DLL plug-in technology for your application using MESSAGE_MAPs

Introduction

Building a product which suits to every customers requirement is every Companies dream. But in practice this may not be the case. This is because in most of the cases the requirement keep on changing drastically so that we have to make changes in the code level , recompile it, test it and ship it as another version. This way of development is not efficient in terms of project cost and developmental effort. This is where the **Addin / Plugin** architecture wins. In this article I have written a ATL COM Addin / Plugin Framework **with dynamic toolbars and menu** support . This framework is actually based on the VC++ 6.0 Addin Architecture which Microsoft used in every MS Office based applications (not sure of the exact implementation , just a guess work, because I followed the same path of VC++ Addin wizard generated code). One difference, instead of adding menus , their description and other details in a new line delimited fashion in VC++ Addin , I used XML format for adding the Plugin details (Don't know why Microsoft didn't use XML?, perhaps at that time **XML** was not standardized , or is there any other specific reason??) along with **Component Category** for categorizing the plugins. Click **here** for seeing the XML format which I used in this project. You can add more properties as leaf nodes if you want (I followed the Idea of putting menu properties as nodes. Alternatively you can put the properties in some other fashion too).

Background

This article assumes that you have a basic understanding of COM and MFC. This article is actually a continuation of the excellent article written by **Zac Howland** Pluggable Components using Component Categories - Part 1 . So the user is advised to go through this article first before coming to my article.

Using the code

The demo project attached contains 3 project workspaces written in VC++ 6.0. The application **ProjectFramework** is the one which loads all the addins , their menus, toolbars and other details. This application uses one core class called **CAddinManager** for various addin manipulations. The main header file of **CAddinManager** class is given below**.**

```
class CAddinManager
{
private:
    BOOL m_bLoadAllAdins;
    CArray<CAddinInfo,CAddinInfo> m_AddinInfoArray;
    CComPtr<IProjectFramework> m_ProjectFrameworkObject;
    HRESULT LoadAllAdins(BSTR strAddinLoadingInfo,
                         IProjectFramework* pProjectFramework);

public:
    BOOL InvokeAddinMenuItem(UINT iCommandID);
    BOOL SetAddinVisible(CString strAddinName, BOOL bVisible);
    const CAddinInfo& GetAddinInformation(UINT iCommandID);
    const CAddinCommadInfo& GetAddinCommadInfo(UINT iCommandID);
    CAddinCommadInfo GetAddinCommadInfo(long iAddinIndex,long lIndex);
    BOOL AddAddinCommandInfo(long iAddinIndex,
                             CAddinCommadInfo AddinCommadInfo);
    BOOL SetAddinCount(long lCount);
    BOOL UnloadAllAddins();
    CAddinInfo GetAddinInfo(long iAddinIndex);
    BOOL SetAddinInfo(long iAddinIndex,CAddinInfo AddinInfo);
    long GetAddinCount();
    void SetLoadAllAddinStatus(BOOL bLoadAllAddins);
    virtual BOOL GetLoadAllAddinStatus();
    CAddinInfo GetAddinIffo(CLSID clsID);

    BOOL LoadAllAddins();
```

```
    BOOL SaveAddinDefaultSettings();
    BOOL LoadAddinDefaultSettings();

    CAddinManager();
    virtual ~CAddinManager();

};
```

This class uses a lot of other related classes like `CAddinInfo` which contains information's about each addin, `IProjectFramework` which is the interface exposed by the ProjectFramework application and `IProjectFrameworkAddin` interface which every pluggin should implement. Please see the source code for more details.The important blocks of codes in which the application built up is given below.

In CProjectFrameworkApp declare a variable of `CAddinManager` statically.

```
static CAddinManager m_AddinManager;
static CProjectFrameworkView* m_pView;
```

in

```
BOOL CProjectFrameworkApp::InitInstance()
{
    ...
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;
    ((CMainFrame*)m_pMainWnd)->LoadAditionalAccelerators();
}
```

in `CMainFrame`

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
//{{AFX_MSG_MAP(CMainFrame)
ON_WM_CREATE()
ON_COMMAND(ID_ADDIN_ADDINSETTINGS, OnAddinAddinsettings)
ON_WM_CLOSE()
ON_WM_MENUSELECT()
ON_WM_INITMENUPOPUP()
//}}AFX_MSG_MAP

ON_COMMAND_RANGE(PF_ADDIN_CMD_MIN_MSG_ID, PF_ADDIN_CMD_MAX_MSG_ID, <BR>
OnAddinMenuItems)
ON_UPDATE_COMMAND_UI_RANGE(PF_ADDIN_CMD_MIN_MSG_ID, PF_ADDIN_CMD_MAX_MSG_ID, <BR>
OnUpdateAddinMenuItems)
ON_UPDATE_COMMAND_UI_RANGE(ID_FILE_NEW, ID_APP_ABOUT, OnUpdateMenuItems)

ON_NOTIFY_EX_RANGE(TTN_NEEDTEXTW, PF_MIN_MSG, PF_MAX_MSG, OnToolTipText)
ON_NOTIFY_EX_RANGE(TTN_NEEDTEXTA, PF_MIN_MSG, PF_MAX_MSG, OnToolTipText)

END_MESSAGE_MAP()
```

and

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


    //Load all addins
    if(CProjectFrameworkApp::m_AddinManager.LoadAddinDefaultSettings())
    {
        if(CProjectFrameworkApp::m_AddinManager.GetLoadAllAddinStatus())
        {
            CProjectFrameworkApp::m_AddinManager.LoadAllAddins();
        }
```

```
    }
    LoadAllAddinCommands();

    if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD |
                                        WS_VISIBLE | CBRS_TOP
                    | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY
                     | CBRS_SIZE_DYNAMIC)
        || !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1; // fail to create
    }

    if (!m_wndStatusBar.Create(this) ||
        !m_wndStatusBar.SetIndicators(indicators,
                                      sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("Failed to create status bar\n");
        return -1; // fail to create
    }

    // TODO: Delete these three lines if you don't want the toolbar to
    // be dockable

    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);

    //Remove File new and Open Commands
    RemoveCommands(ID_FILE_NEW);
    RemoveCommands(ID_FILE_OPEN);
    return 0;
}
```

Points of Interest

In this version features implemented are.

a. Dynamic Menus.
b. Dynamic Toolbars.
c. Help string and tool tip support.
d. Invoking of methods in plugins from addin menu and toolbar.
e. Automation support explained using a dialog box invoked from one of the plugins (Report Addin -> Sales report -> Todays Sales Report Ctrl + B menu).
f. Hiding of Menus and toolbar buttons (eg : File ->New and Open).
g. Key Board accelerator support for plugin.
h. Loading / Unloading of addins.
i. Connection point support for getting notification events from addins.

History

- Initial Release : Dec 2, 2004
- Added Connection Point support : Dec 10, 2004 <!------------------------------ That's it! ---------------------------->

License

Share

About the Author

thomas_tom99
Chief Technology Officer KTS INFOTECH PVT LTD
India

->9+ Years of Experience in IT Field.
->Basically a C++ Programmer migrating to .NET
->Have Masters degree in Physics and Computer Scince.
-> Doing his Ph.D(Part Time) in Optical Networking)
->Interests: Software product development,Networking, Robotics,Sports Physics, Learning musical instruments, Cricket.

->Resides in kerala ,the gods own country, with his mother and wife.

Home page

You may also be interested in...

.NET Based Add-in/Plug-in Framework with Dynamic Toolbars and Menus

Window Tabs (WndTabs) Add-In for DevStudio

Building an Office2K COM addin with VC++/ATL

Introduction to D3DImage

SAPrefs - Netscape-like Preferences Dialog

OLE DB - First steps

Comments and Discussions

48 messages have been posted for this article Visit **https://www.codeproject.com/Articles/8955/ATL-COM-Based-**

Addin-Plugin-Framework-With-Dynamic to post and view comments on this article, or click **here** to get a print view with messages.