COM 应用程序框架

设计方案 试验 A 版

许宗森

本文只介绍《COM 应用程序框架》的主要设计部分，更多、更详细的文档信息请参见下载文件包中的文档和源代码。

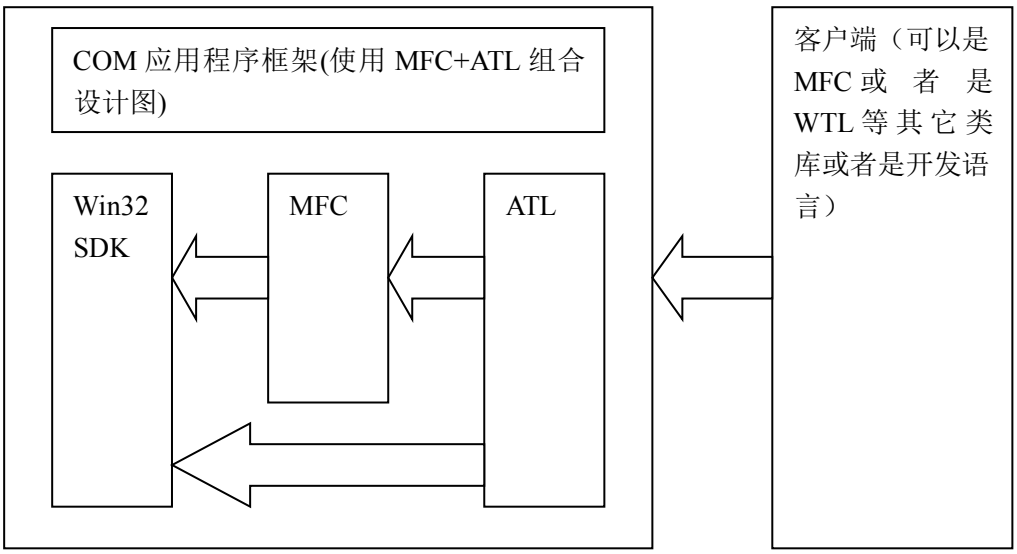
一、 设计说明

《COM 应用程序框架》是把标准的 Microsoft Windows 多文档处理应用程序使用 COM 技术来设计. 所以真对多文档处理应用程序的需求，不在多写。如果您不了解请参见 MSDN 或者是其它编程基础方面的书籍。

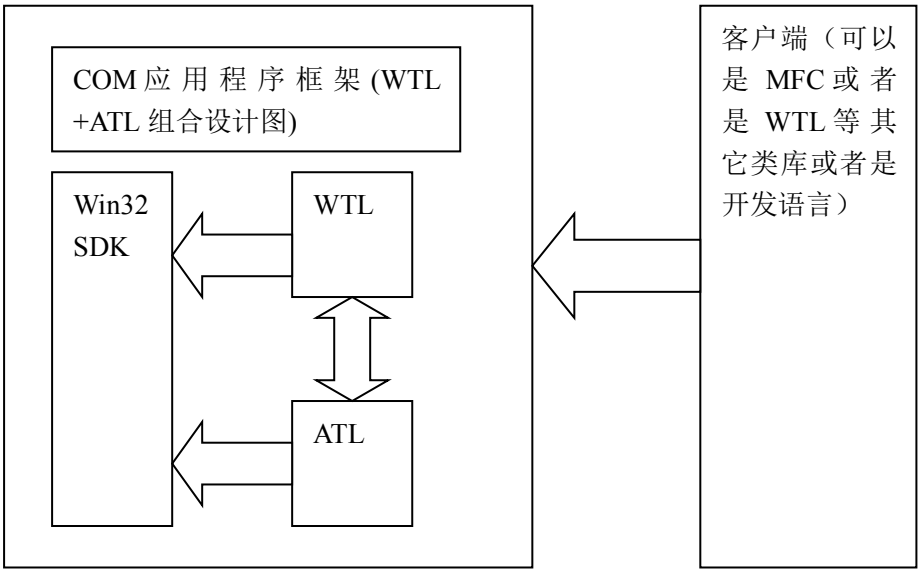
《COM 应用程序框架》分为两种，一种是 COM 多文档应用程序框架，第二种是单文档应用程序框架。在这里我们只介绍多文档应用程序，不介绍单文档应用程序。

《COM 应用程序框架》建立在一个单独的 AIFrame.DLL 文件中，所有的功能都通过 COM 接口进行操作. 《COM 应用程序框架》将使用如下两种库的组合进行设计，1.使用 MFC+ATL 组合, 2.使用 WTL+ATL 组合，下面分别对这两种组合的优点和缺点说明一下，最后选择一种最优组合。

1. 使用 MFC+ATL 组合图:



2. 使用 WTL+ATL 组合图:



从上面两附图中很容易看出，使用 MFC+ATL 组合开发 COM 应用程序框架，MFC 存在一层函数调用，代码执行速度会慢一些。小程序可能看不出来，大程序也就明显了。

如果使用 WTL+ATL 组合开发 COM 应用程序框架，就不会多一层调用，代码执行速度非常快，就像是用 Win32 SDK 编写代码一样没有什么区别，因为 WTL 是模板代码，在编译后不会存在一层函数调用。所以《COM 应用程序框架》将采用 WTL +ATL 组合进行设计，这可能是最佳方案。

二、 通用设计

1. 数据视图

所谓数据视图，就是添加到《COM 应用程序框架》中每一个窗口，无论这个窗口是用做什么，还是什么形状的，统称为数据视图。

所有客户端程序向《COM 应用程序框架》添加的数据视图必须从 IDataView 纯虚接口继承下来，必须是. 这样《COM 应用程序框架》才能工作正常。

数据视图纯虚接口:IDataView。父类是 IDispatch。

2. 命令的响应函数

函数名称：NotifyCommand(UINT codeNotify, UINT cmdID, VARIANT_BOOL * bHandle);

参数： codeNotify - 通报代码，现在没有使用。

cmdID - 某个命令 ID，可以是菜单也可以是工具格中的按钮。

bHandle - 如果命令还继续向下路径设置为 VARIANT_TRUE,不向下路径设置为 VARIANT_FALSE.

三、 《COM 多文档应用程序框架》设计

1. 框架

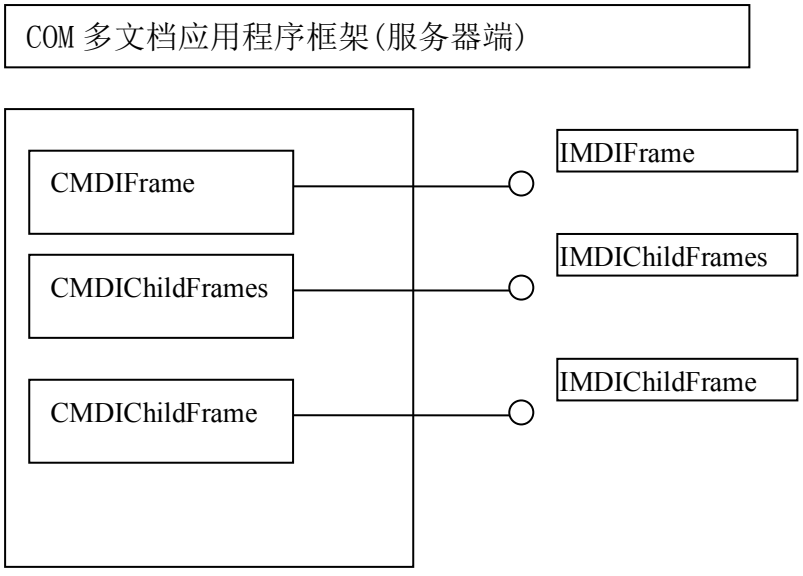
《COM 多文档(MDI)应用程序框架》只有一个 MDI 主窗口，但是可以拥有 N 多 MDI 子窗口，每个 MDI 子窗口可以拥有多个数据视图，每一个数据视图必须从 IDataView 接口继承下来。

MDI 主窗口框架类名：CMDIFrame,接口是 IMDIFrame，是所有 MDI 子窗口、停靠数据视图、工具条、菜单的容器，并负责把命令分发给当前活动的 MDI 子窗口或者是数据视图。

MDI 子窗口框架类名：CMDIChildFrame,接口是 IMDIChildFrame。可以拥有 N 多数据视图，并把命令分发给当前活动的数据视图。

MDI 子窗口框架集合类名：CMDIChildFrames，接口是 IMDIChildFrames，用来管理所有的 MDI 子窗口和创建 DMI 子窗口。

《COM 多文档应用程序框架》接口图：



2. 事件

- IMDIFrame 事件

  目前仅提供两个事件。

  OnQuit(VARIANT_BOOL *vbQuit)

  NotifyCommand(UINT codeNotify, UINT cmdID, VARIANT_BOOL *bHandle)

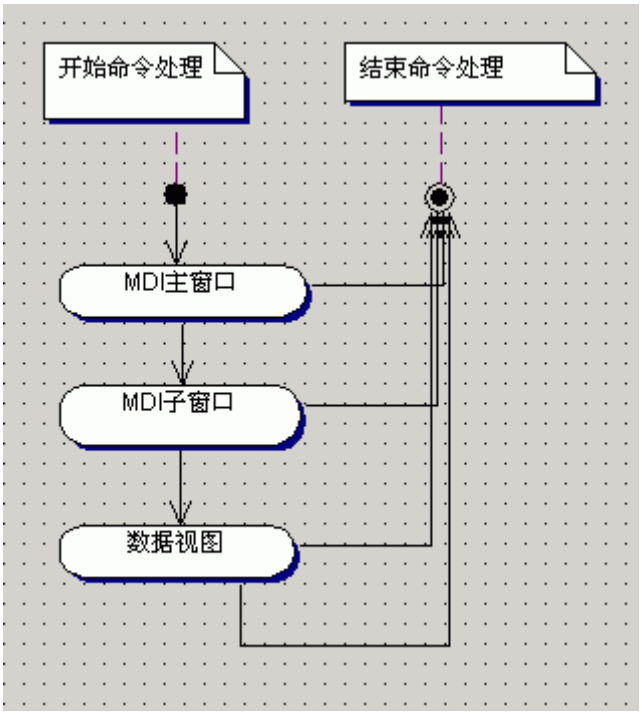
- IMDIChildFrame 事件
  目前仅提供一个事件。
  NotifyCommand(UINT codeNotify, UINT cmdID, VARIANT_BOOL *bHandle)

3. 命令路由

MDI 主窗口框架、MDI 子窗口框架事件类和数据视图都拥有一个命令处理方法 NotifyCommand。《COM 多文档应用程序框架》就是通过 NotifyCommand 向客户端发送命令处理。

MDI 主窗口事件首先接收到菜单或者是工具条命令，处理完必后返回或者是发送给 MDI 子窗口框架事件处理，在 MDI 子窗口框架处理完必后返回或者是发送给让当前活动的数据视图处理. 完成后命令返回 。

命令是否向下路由是有 NotifyCommand 方法的 bHandle 变量决定，设置为 VARIANT_TRUE 命令向下路由，设置为 VARIANT_FALSE 命令不向下路由。



4. UML 类图

用 COM 设计的类比较长抓图不太方便，UML 图、类的信息和函数成员请参见源代码。

四、 客户端设计

客户端应用程序必须实现两个功能块，第一个是数据视图，第二个是命令处理。

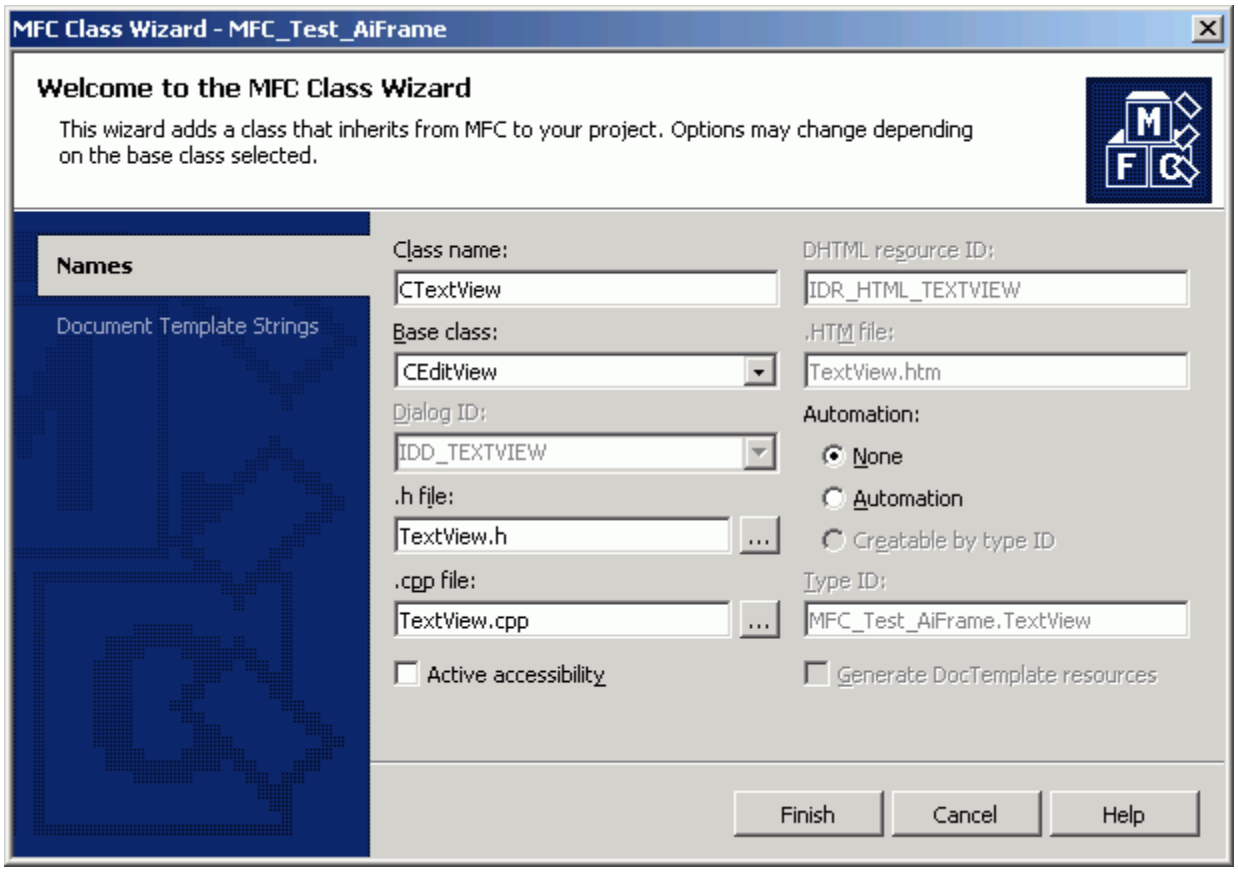
五、 项目文件夹说明

文件夹名称	类型	说明
AiFrame	.dll	《COM 应用程序框架》全部代码所在文件夹。
inc	.h	《COM应用程序框架》客户端使用的一些公用头文件和已经设计的类。
MFC_Test_AiFrame	.exe	使用 MFC 调用《COM 应用程序框架》的实例
WTL_Test_AiFrame	.exe	使用 WTL 调用《COM 应用程序框架》的实例
Bin	Bin	生成的.dll 和.exe 文件存放的文件夹

六、 讲解使用 MFC 调用《COM 应用程序框架》的实例

这个实例使用 MFC 库调用《COM 应用程序框架》建立起的一个简单的文本程序。以方便了解如何使用《COM 应用程序框架》的流程和主要部分，和《COM 应用程序框架》方便之处。

- 使用 MFC 项目向导，建立一个对话框项目应用(项目名称自己定义)假如是 MFC_Test_AiFrame 项目。
- 在 stdafx.h 文件加入。
  #include "..\inc\aiframeimpl.h"
- 在某个 .cpp 中加入
  #include "..\inc\AiFrame_i.c"
- 向项目中添加一个文本编辑视图类,如下图.



新建立的类名是 CTextView, 基类(Base class)是 CeditView 类。单击完成按钮。
打开 TextView.h 文件加入如下代码：
CDataViewImp 是 IDataView 数据视图接口实现模板类。

```
 class CTextView : public CEditView
            ,public CDataViewImpl<CTextView>
{
....
NC_BEGIN_MAP()
    //在这里加入您的视图命令处理.
NC_END_MAP()
virtual HRESULT STDMETHODCALLTYPE CreateWnd(HWND hWndParent);...
}
```

打开 TextView.cpp 文件加入如下代码：

```
//必须有 CreateWnd 方法，有《COM 应用程序框架》调用。
HRESULT STDMETHODCALLTYPE CTextView::CreateWnd(HWND hWndParent)
{
CWnd *pWnd = (CWnd*)this;
CWnd *wndParent = CWnd::FromHandle(hWndParent);
CRect _Rect(0,0, 100, 100);
```

```
//建立视图
if (pWnd->Create(NULL,
("MFC CEditView "),
WS_HSCROLL|WS_VSCROLL|WS_CLIPSIBLINGS|WS_CLIPCHILDREN|WS_VISIBLE|ES_
MULTILINE|ES_AUTOHSCROLL,
_Rect, wndParent, 0))
{
m_pEdit = &GetEditCtrl();
return S_OK;
}

return S_FALSE;
}
```

◆ CxxxxxApp 类中主要完成一个功能．继承 CNotifyObjectImpl 对象，用来处理《COM 应用程序框架》发送来的命令．

CNotifyObjectImpl 类是 InotifyObject 接口的实现模板类。

建立主窗口和一个新建文件的子菜单响应该命令．

```
class CFrameApp : public CwinApp
                  ,public CComObjectRootEx<CComSingleThreadModel>
                  ,public IDispEventImpl<EventType, CFrameApp,
              &DIID__IMDIFrameEvents, &LIBID_AiFrameLib>      //MDI 主窗口事件处理
              对象.
{
        IMDIFrame *m_lpMdiFrame;          //主窗口 com 接口类.
        HRESULT CFrameApp::CreateMain()

        //《COM 应用程序框架》发送来的命令处理宏代码.
    BEGIN_SINK_MAP(CFrameApp)
        SINK_ENTRY_INFO(EventType, DIID__IMDIFrameEvents, DISPID_SHOW, OnQuit,
&OnShowInfo1)
        SINK_ENTRY_INFO(EventType,    DIID__IMDIFrameEvents,    DISPID_SHOW2,
NotifyCommand, &OnShowInfo3)
    END_SINK_MAP()
        NC_BEGIN_MAP()
                        NC_COMMAND_ID_HANDLER(ID_HELP_ABOUT,OnAbout)
        NC_COMMAND_ID_HANDLER(ID_FILE_NEW,FileNew)
        NC_END_MAP()
        …
}

HRESULT CFrameApp::CreateMain()
```

```
{

CreateMDIStruct lpMDI={0};
HRESULT hr = 0;
//建立《COM 应用程序框架》IMDIFrame 接口类.
hr = CoCreateInstance(CLSID_MDIFrame, NULL, CLSCTX_ALL, IID_IMDIFrame,
(VOID**)&m_lpMdiFrame););

if (FAILED(hr))
{
	ATLASSERT(0);
	return hr;
}

//建立主窗口.
lpMDI.cbSize = sizeof(CreateMDIStruct);
lpMDI.lParam = NULL;
lpMDI.lpszWindowName = L"Test COM MDIFrame";
lpMDI.hMenu=LoadMenu(AfxGetResourceHandle(),
MAKEINTRESOURCE(IDR_MAINFRAME));
hr = m_lpMdiFrame->CreateWnd(&lpMDI);
m_lpMdiFrame->ShowMe(VARIANT_TRUE);

// IMDIFrame  接口对象添加命令通报对象。
	hr = this->DispEventAdvise((IUnknown*)m_lpMdiFrame);
	if (FAILED(hr))
	{
		ATLASSERT(0);
	}
return hr;
}
```

处理新建文件菜单的命令：

```
void CFrameApp::FileNew(UINT codeNotify, UINT cmdID, VARIANT_BOOL *bHandle)
{
IMDIChildFrames *lpFrames = NULL;
IMDIChildFrame *lpChildFrame = NULL;
HRESULT hr = 0;

//从主窗口 IMDIFrame 接口引出 IMDIChildFrames  集合.
hr = m_lpMdiFrame->get_MDIChildFrames((IDispatch**)&lpFrames);

if (FAILED(hr))
{
```

```
ATLASSERT(0);
}

//用 IMDIChildFrames 集合建立一个 mdi 子窗口．lpChildFrame，反回子窗口接口类．
CTextView *lpTexView = NULL;
lpTexView = new CTextView();
lpFrames->CreateChildFrame((IDataView*)lpTexView, &lpChildFrame);

//设置子窗口标题．
lpChildFrame->put_Title(L"MFC TextView");
}
```

七、　　其它信息

Blog : http://blog.csdn.net/netlinux
Web site :http://www.willspace.net
E-Mail: ytf1978@163.com

《COM 应用程序框架》设计下载地址
http://www.willspace.net/Html/Down/20060619066191012252685 2.htm
http://www.willspace.net/Down.asp?ID=0661923052853096&DownID=0661910122526852