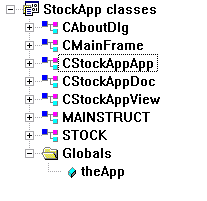
MFC单文档程序架构解析

 https://www.cnblogs.com/HappyXie/archive/2011/05/06/mfc.html

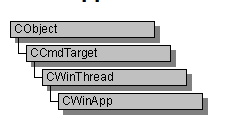
这里我以科院杨老师的单文档程序来分析一下MFC单文档的程序架构，纯属个人见解，不当之处烦请指教！

首先我们了解到的是



图（一）

theApp 是唯一一个在程序形成的时候就存在的全局变量，它属于CstockAppApp类，而CstockAppApp 继承于CwinApp类，我们看一下MSDN中CwinApp的继承关系如下：



图（二）

从继承关系当中，我们发现theApp是作为程序的实体而存在的，是单文档程序的核心。

首先分析一下的是CsockAppApp这个类，这里面有一个重要的函数

BOOL CStockAppApp::InitInstance()这个函数，包含了单文档程序中重要的信息，特别要提到的是一下的一段代码：

CSingleDocTemplate\* pDocTemplate;

pDocTemplate = new CSingleDocTemplate(

IDR\_MAINFRAME,

RUNTIME\_CLASS(CStockAppDoc),

RUNTIME\_CLASS(CMainFrame),       // main SDI frame windon

RUNTIME\_CLASS(CStockAppView));

这里体现了几个重要的思想，第一动态创建和动态附加的一种思想，RUNTIME\_CLASS是一个宏定义，这里不做展开，可以理解为最后的函数返回了实现了如下的语句

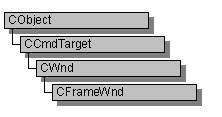
Retrun new CmainFramw 的功能，这里将三个不同的类对象附加到CsingleDocTemplate对象中，这样使得整个程序得以整体成立。

接下来对剩下的三个类对象做分析

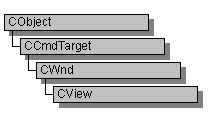
1. CmainFrame类

CmianFrame类作为程序的框架类，其起到了一种容器的作用，在这个容器当中可以装在多个视图，菜单，工具等。要注意的有几点，第一，因为mianfram是没有视图的，因此如果在mainframe相应Onpaint消息，自然是可以响应这个消息的，但是你会惊奇的发现，在你的绘制当中如何也不会出现你预想出现的绘制内容，为什么呢？ 嘿嘿，可以想象一下你在一个word程序里面，当你关闭了所有的白板（视图）的时候，你会发现你已经无法在编写文字，道理是一样的，在mainframe里面进行绘制，程序是没有问题，但是绘制的内容是在灰色上面，windows不予显示的。

第二点你可以发现在对菜单，和工具条的单击消息进行相应的时候你可以将消息响应函数添加到cmainframe也没有将消息响应函数调价到cview当中，但是你会惊奇的发现，两者只在一个地方相应的时候，消息响应函数没有问题都能正确的执行，但是如果同时对一个按钮或者菜单进行单击消息响应的时候，你会发现windows会执行的Cview里面的消息响应函数。分析两者之间的关系，首先Cmianframe的继承关系如下：



而对Cview类而言其继承类关系如下图所示：



可以看出两个类之间并没有直接的关系，但是我们通过程序的运行可以知道消息的响应是以Cview中为主的，可以认为存在一种类似于多态的关系。

这里具体的消息路径如下：

  在SDI(单文档)界面中，菜单响应遵循这样一个顺序：菜单消息先由CMainFrame类接收，CMainFrame并不直接在内部寻找对应的相应函数，而是到CView类寻找。如果CView类有该消息的响应函数，那么就直接调用CView类中的响应函数，否则，转到CDoc类寻找，如果CDoc类中存在该消息的响应函数，那么就直接调用CDoc类中的响应函数，否则，返回到CMainFrame类寻找。如果CMainFrame类中也没有，返回到CApp类中寻找。如果在CApp类中也没有找到，表示没有该菜单的响应函数。

同时CmainFrame作为整个程序的框架，它提供了程序运行的基础环境，这里再强调介绍一下两点

1. 在CmainFrame中访问Cview对象和Cdoc对象

要访问这两个对象可以使用全局函数GetActiveDocument()和GetActiveView（）这样可以获得Doc对象和View对象的实体了

2. 在CmainFrame中调用Cview对象更新窗口，这里使用方法如下：GetActiveView()->Invalidate(FALSE);// 这一句会是cview调用OnDraw消息响应函数

GetActiveView()->UpdateWindow();//这句可以加上也可以不加，暂时没有发现不加会出现什么问题。

2.Cview类

   Cview类作为视图类，其可以理解为一张画布，在这张画布上可以画图，也可以画控件，其重要的函数有OnDraw这是主要绘图出现的地方。在View中要实现重画的时候可以按如下方式实现调用：

   void CTestView::OnChangeRect()

{

// Change Rectangle size.

m\_rcBox = CRect(20, 20, 210, 210);

// Invalidate window so entire client area

// is redrawn when UpdateWindow is called.

Invalidate();

// Update Window to cause View to redraw.

UpdateWindow();

}

// On Draw function draws the rectangle.

void CTestView::OnDraw(CDC\* pDC)

{

//  ..  Other draw code here.

pDC->Draw3dRect(m\_rcBox, 0x00FF0000, 0x0000FF00);

}

 在Cview内中要获取到CmianFrame可以使用下面的函数：

AfxGetMainWnd()

3.doc类对象。

个人觉得单文档视图的设计模式很接近MVC的设计模式，Model层可以粗略的认为是在Doc这个类对象里面，在单文档视图当中Doc对象充当的也是数据存储的类，可见确实是有一些MVC设计模式的意思。这里介绍在其他类中要获取Doc对象的方法

在Cmianframe中可以使用

   GetActiveDocument()

在Cview类中可以使用

   这里要注意一点，在Cview类中包含了一个Cdocument的对象m\_pDocument这个对象即使指向Doc类的基类对象的，而要实现基类对象到现在doc类对象的转换只要添加如下函数即可

inline CStockAppDoc\* CStockAppView::GetDocument()

{ return (CStockAppDoc\*)m\_pDocument; }

则可以实现。

附上三个类对象相互访问方法：

1、主框架(CFrameWnd)中访问视图(CView)  
CView\* GetActiveView() const;  
通常定义的视图为CView的派生类，在调用自定义视图对象的方法时

应该这样写：((CMouseKeyView\*)GetActiveView())->MyFunc();

2、主框架(CFrameWnd)中访问文档(CDocument)  
GetActiveDocument，返回CDocument对象；

3、在视图(CView)中访问文档(CDocument)  
inline CMouseKeyDoc\* CMouseKeyView::GetDocument()  
{return (CMouseKeyDoc\*)m\_pDocument;}

4、在视图(CView)中访问框架(CFrameWnd)  
CFrameWnd\* GetParentFrame() const;

这里修改一下，因为上述代码获取的只是CMainFrame的父类对象CFrameWnd对象，要获取实际的CMainFrame对象可以进行如下操作首先在头文件中添加#include "MainFrm.h" 然后代码如下：

CMainFrame\* frm=(CMainFrame\*)::AfxGetMainWnd();frm->test(); 就能成功咯

5、在文档(CDocument)中访问框架(CFrameWnd)  
CWnd\* AfxGetMainWnd();  
CWnd\* AfxGetApp()->m\_pMainWnd;

6、在文档(CDocument)中访问视图(CView)  
UpdateAllViews  
功能：通知所有的视图文档已被修改的信息  
原型：  
void UpdateAllViews(  
CView\* pSender, // 要更新的视图指针，如果希望更新所有视

图，将此参数设为NULL  
LPARAM lHint=0L, // 包含更改消息的通知  
CObject\* pHint=NULL // 保管更改消息的对象  
    }

7、在其他类中访问文档类(CDocument)  
CDocument\* GetDocument()  
{  
CFrameWnd\* frm=(CFrameWnd\*)::AfxGetMainWnd();  
ASSERT(frm);  
CDocument\* pDoc=frm->GetActiveDocument();  
ASSERT(pDoc);  
ASSERT(pDoc->IsKindOf(RUNTIME\_CLASS(CMouseKeyDoc)));  
return (CMouseKeyDoc\*)pDoc;  
}