**mfc 框架 元素之间通信方式**

https://blog.csdn.net/qiangxyz/article/details/5836254

Gilbert是个喜欢收集的人, 有些东西还是收了过来.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 获得CWinApp | 获得CMainFrame | 获得CChildFrame | 获得CDocument | 获得CView |
| 在CWinApp  中 |  | AfxGetMainWnd()  m\_pMainWnd | AfxGetMainWnd()->MDIGetActive()  AfxGetMainWnd()->GetActiveFrame() | SDI:AfxGetMainWnd()->GetActiveView()->GetDocument()  MDI:AfxGetMainWnd()->MDIGetActive()->GetActiveView()->GetDocument() | SDI:AfxGetMainWnd()->GetActiveView()  MDI:AfxGetMainWnd()->MDIGetActive()->GetActiveView() |
| 在CMainFrame中 | AfxGetApp()  theApp |  | MDIGetActive()  GetActiveFrame() | SDI:GetActiveView()->GetDocument()  MDI:MDIGetActive()->GetActiveView()->GetDocument() | SDI:GetActiveView()  MDI:MDIGetActive()->GetActiveView() |
| 在CChildFrame中 | AfxGetApp()  theApp | GetParentFrame() |  | GetActiveView()->GetDocument() | GetActiveView() |
| 在CDocument中 | AfxGetApp()  theApp | AfxGetMainWnd() | AfxGetMainWnd()->MDIGetActive()  AfxGetMainWnd()->GetActiveFrame() |  | POSITION   pos   =   GetFirstViewPosition();GetNextView(pos) |
| 在CView中 | AfxGetApp()  theApp | AfxGetMainWnd() | GetParentFrame() | GetDocument() |  |
| 在其他类中 | AfxGetApp() | AfxGetMainWnd() | AfxGetMainWnd()->MDIGetActive()  AfxGetMainWnd()->GetActiveFrame() | SDI:AfxGetMainWnd()->GetActiveView()->GetDocument()  MDI:AfxGetMainWnd()->MDIGetActive()->GetActiveView()->GetDocument() | SDI:AfxGetMainWnd()->GetActiveView()  MDI:AfxGetMainWnd()->MDIGetActive()->GetActiveView() |

MainFrame是主框架，也基本可以用全局函数访问到。MainFrame下是若干个ChildFrame，ChildFrame中若干个View和Document（可能不成对），ChildFrame管理着View，View和Document进行互操作。因此整体框架就出来了，一般除了直接应用的关系都可以通过MainFrame-->Active ChildFrame-->Active View-->Document这条线进行访问

一、SDI

1：在CMainFrame框架中如何得到视图类的指针：

C \*\*View \* pView;

pView=(C\*\*View\*)((CFrameWnd\*)AfxGetApp()

->m\_pMainWnd)->GetActiveView();

    CWinApp中的m\_pMainWnd变量就是CMainFrame的指针。每一个MFC应用程序都有一个CWinApp派生类的对象。而 CWinApp 类中有一个 CWnd \* m\_pMainWnd 成员变量。

    当新建一个MFC应用程序的时候，在 InitInstance虚函数里都会出现对 m\_pMainWnd 赋值的语句.唯一的例外是单文档界面的MFC应用程序，你无法在 InitInstance 函数里看到这段代码，因为它已经被隐藏在 ProcessShellCommand 这个函数里了。

    因为AfxGetApp()得到的是CWinApp类的对象,且AfxGetApp返回值为CWinApp对象指针，因为你是在自己创建的项目中得到CWndApp成员函数或者成员变量,所以你必须强制转换为你自己的项目中的类，才能找到成员函数或者变量。

    注: 在单文档中,获得视指针的最简单的方法还是：

    ((C\*\*View \*)CFrameWnd::GetActiveView())

2：当然在CMainFrame中也可以得到文档类的指针:

CMyDocument\* pDoc;

pDoc=(CMyDocument\*)((CFrameWnd\*)AfxGetApp()->m\_pMainWnd)

->GetActiveDocument();

3：由上面可以知道:在View中怎样获得MainFrame指针

CMainFrame \*pMain=(CMainFrame \*)AfxGetApp()->m\_pMainWnd;

注: 从视图类中获得MainFrame指针也可以：

((CMainFrame \*)CWnd::GetParentFrame())或者

((CMainFrame \*)AfxGetMainWnd())

二、MDI

当然对于MDI程序，由于子窗口才是文档框窗，因此首先要用GetActiveFrame()取得活动子框架窗口，然后通过该子窗口获取活动视图和文档：

CMDIChildWnd\* pChild=(CMDIChildWnd\*)((CFrameWnd\*)AfxGetApp()

                                ->m\_pMainWnd)->GetActiveFrame();

取得活动视图:

CMyView\* pView=(CMyView\*)pChild->GetActiveView();

取得活动文档：

CMyDocument\* pDoc=pChild->GetActiveDocument();

注: 也可以用这种方法来得到多文档中的视指针

获得活动子框架窗口

CMDIChildWnd\* pChild=(CMDIChildWnd\*)GetActiveFrame();

CMDIChildWnd\* pChild=MDIGetActive();

获得活动子帧窗口的活动视图

CMyView\* pView=(CMyView\*)pChild->GetActiveView();

三、还是MDI

1.从视图类获得文档类的指针

在视图类中需要引用文档类的地方之前，使用以下语句：

C\*Doc \*pDoc=(C\*Doc\*)GetDocument();

2.从文档类取得视图类的指针CDocument类提供了两个函数用于视图类的定位：

GetFirstViewPosition()和GetNextView()

注意：GetFirstViewPosition()用于返回第一个视图位置（返回的并非视图类指针，而是一个POSITION类型值），GetNextView()有两个功能：返回下一个视图类的指针以及用引用调动的方式来改变传入的POSITION类型参数的值。在Test程序中，只有一个视图类，因此只需将这两个函数调用一次即可得到 CTestView的指针如下：

C\*View\* pView;

POSITION pos=GetFirstViewPosition();

pView=GetNextView(pos);

这样便可到了C\*View类的指针pView。执行完成后，变量pos=NULL，因为没有下一个视图类，自然也没有下一个视图类的 POSITION。但是这几条语句太简单，不具有太强的通用性和安全特征；当要在多个视图为中返回某个指定类的指针时，我们需要遍历所有视图类，直到找到指定类为止。判断一个类指针指向的是否某个类的实例时，可用IsKindOf()成员函数时行检查．

如：

pView->IsKindOf(RUNTIME\_CLASS(C\*View));

即可检查pView所指是否是C\*View类。

有了以上基础，我们已经可以从文档类取得任何类的指针。为了方便，我们将其作为一个文档类的成员函数，它有一个参数，表示要获得哪个类的指针。实现如下：

CView\* C\*Doc::GetVieww(CRuntimeClass\* pClass)

{

CView\* pView;

POSITION pos=GetFirstViewPosition();

while(pos!=NULL)

{

pView=GetNextView(pos);

if(!pView->IsKindOf(pClass))

break;

}

if(!pView->IsKindOf(pClass))

{

AfxMessageBox("Connt Locate the View.");

return NULL;

}

return pView;

}

其中用了两次视图类的成员函数IsKindOf()来判断，是因为退出while循环有三种可能：  
1.pos为NULL，即已经不存在下一个视图类供操作；

2.pView已符合要求。

3.1和2同是满足。这是因为GetNextView()的功能是将当前视图指针改变成一下个视图的位置同时返回当前视图指针，因此pos是pView的下一个视图类的POSITION，完全有可能既是pos==NULL又是pView符合需要。因此需采用两次判断。