Lab 7.2: Adding Open File Dialogs and a Rich Edit View

Objectives

After completing this lab, you will be able to:

- Modify the base classes of views and documents.
- Use a Common Dialogs Library Open dialog box.
- Use a Rich Edit view in an application.
- Process a dropped file on the edit control.

Prerequisites

This lab assumes that you have completed or are familiar with the techniques and concepts covered in Lab 1 of this chapter.

Lab Setup

To run the solution to this lab, click this icon.

To see a demonstration of the solution for this lab, click this icon.

Estimated time to complete this lab: **90 minutes**.

Exercises

The following exercises provide practice working with the concepts and techniques covered in this chapter.

Exercise 1: Modifying the Base Classes of Views and Documents

In this exercise, you will use the text editors of Visual C++ to modify the Document and View classes to inherit from their respective Rich Edit classes.

Exercise 2: Handling the Common Open File Dialog Within an Application

In this exercise, you will add a handler for the Open File dialog box in the **ShowDiff** application.

Exercise 3: Using the Rich Edit Views in the Application

In this exercise, you will use the Rich Edit views to display the file that the user chooses in the File dialog box. You will create a function that serializes the text data into the Rich Edit views, and drive that function from the **OnFileOpen** handler.

Exercise 4: Managing Drag and Drop

When you run the solution to Exercise 3, you will find that Drag and Drop from the Explorer is enabled as part of the Rich Rdit control, but it is enabled as ActiveX rather than File Open as the user may expect. In this exercise, you will modify the behavior of CRichEdit so that it will open the file rather than link it.

Optional Exercise

In this exercise, you will write code to accept one file in the drag and drop. If there is a second file to compare to the first, run the comparison. Otherwise, wait for a drop in the other pane of the splitter.

For this lab, you will need to use the project that you created in Lab 1 of this series of Labs. You can copy this project from \Labs\C07\Lab02\Baseline. The completed code for these exercises is in \Labs\C07\Lab02\Xxx, where Xxx is the exercise number.

Exercise 1: Modifying the Base Classes of Views and Documents

If you do not have a starting point for this exercise, the code that forms the basis for this exercise is in \Labs\C07\Lab02\Baseline.

In this exercise, you will use the text editors of Visual C++ to modify the Document and View classes to inherit from their respective Rich Edit classes.

➢ Replace CDocument with CRichEditDoc

1. Open DiffDoc.Cpp.
2. From the Edit Menu, select Replace.

   Find **CDocument** and replace it with **CRichEditDoc** by either clicking the Find Next button and repeatedly clicking the Replace button or clicking the Replace All button.
3. Save DiffDoc.Cpp.
4. Repeat this procedure with DiffDoc.H.

➢ Replace CView with CRichEditView

1. Open DiffView.Cpp.
2. From the Edit Menu, select Replace.

   Find **CView** and replace it with **CRichEditView** by either clicking the Find Next button and repeatedly clicking the Replace button or clicking the Replace All button.
3. Save DiffView.Cpp.
4. Repeat this procedure with DiffView.H.

➢ Add the #includes that support Rich Edit classes to Stdafx.H.

1. Open Stdafx.H.
2. The **CRichEditDoc** class inherits from **COleDocument**, **COleLinkingDoc**, and **COleServerDoc**. The definitions for these classes and their support functions need to be included before the RichEdit definitions proper. To the end of Stdafx.H, add:

```
#include <afxole.h>
#include <afxodlgs.h>
```

3. The definitions for Rich Edit are in Afxrich.H To the end of Stdafx.H, after Afxodlgs.H, add:

```
#include <afxrich.h>
```

4. Save Stdafx.H.

➢ Add the support for CRichEditCntrItem

Derived classes from **CRichEditDoc** need to provide a pointer to a control item (the class that contains the Rich Edit control itself). This simple function is implemented in DiffDoc.H.

1. Open DiffDoc.H.
2. Add this code to the overrides section (after the //}AFX_VIRTUAL tag):

```
virtual CRichEditCntrItem* CreateClientItem(REOBJECT* preo) const;
```

3. Save DiffDoc.H.
4. Open DiffDoc.Cpp.
5. Add this code right after the destructor (**CDiffDoc::~CDiffDoc**):

```
CRichEditCntrItem* CDiffDoc::CreateClientItem(REOBJECT* preo) const
{
    return new CRichEditCntrItem();
}
```

6. Save DiffDoc.Cpp.

The completed code for this exercise is in \Labs\C07\Lab02\Ex01.

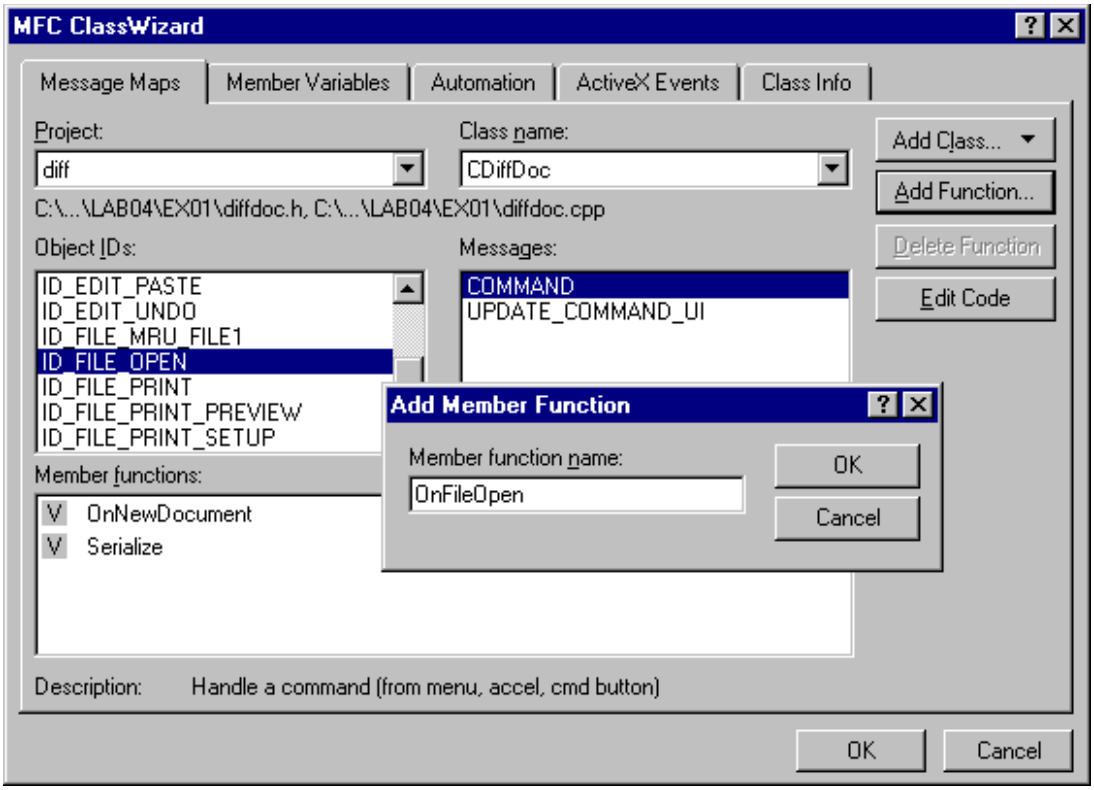
Exercise 2: Handling the Common Open File Dialog

Continue with the files you created in Exercise 1 or, if you do not have a starting point for this exercise, the code that forms the basis for this exercise is in \Labs\C07\Lab02\Ex01.

In this exercise, you will add a handler for the Open File dialog box to the ShowDiff application.

➢ Add an OnOpenFile handler for CDiffDoc

1. From the View menu, choose ClassWizard or press CTRL+W. Select the Message Maps tab.
2. Select the **CDiffDoc** class, the ID_FILE_OPEN menu object ID, and the COMMAND message, and click the Add Function button, as shown in this figure.



3. Accept the default OnFileOpen function name in the Add Member Function dialog box, and click OK.
4. In ClassWizard, click the OK button to create the function.

➢ Declare protected filename members

Right-click **CDiffDoc** in ClassView and add these two protected variables:

```
CString m_File1
CString m_File2
```

➢ Complete the OnOpenFile handler

1. Open DiffDoc.Cpp and scroll to the **OnOpenFile** handler. Alternatively, start ClassWizard, choose the **OnOpenFile** handler for **CDiffDoc**, and click the Edit Code button.
2. Define the filter for the File dialog. In this example, you will filter for C++ source files (*.Cpp and *.H) and for all files.

```
static char BASED_CODE szFilter[] =
    "All Files (*.*)|*.*|C++ Files (*.cpp, *.h)"\
```

```
    "|*.cpp;*.h||";
```

3. Construct the File dialog as an open dialog box with this filter:

```
CFileDialog dlg(TRUE,NULL,NULL,NULL,szFilter);
```

4. Show the File dialog as a modal dialog box, and check to see whether the OK button was pressed.

```
if (dlg.DoModal() == IDOK)
```

5. When the user clicks OK, assign the path to m_File1 and m_File2.

```
m_File1 = m_File2 = dlg.GetPathName();
```

6. Save DiffDoc.Cpp.

The complete code for the **OnOpenFile** handler is:

```
void CDiffDoc::OnFileOpen()
{

    static char BASED_CODE szFilter[] =
        "All Files (*.*)|*.*|C++ Files (*.cpp, *.h)| "\
        "*.cpp;*.h||";

    CFileDialog dlg(TRUE,NULL,NULL,NULL,szFilter);

    if (dlg.DoModal() == IDOK)
    {
        m_File1 = m_File2 = dlg.GetPathName();
    }
}
```

The completed code for this exercise is in \Labs\C07\Lab02\Ex02.

Exercise 3: Using the Rich Edit Views in the Application

Continue with the files you created in Exercise 2 or, if you do not have a starting point for this exercise, the code that forms the basis for this exercise is in \Labs\C07\Lab02\Ex02.

In this exercise, you will use the Rich Edit views to display the file that the user chooses in the File dialog box. You will create a function that serializes text data into the Rich Edit views, and drive that function from the **OnFileOpen** handler.

➢ **Get the application and the main frame**

1. Open Diff.H.
2. Add a forward reference to **CMainFrame** before the definition of **CDiffApp**.

```
    class CMainFrame;
```

3. Define a public method to return a pointer to the application.

```
public:
        static CDiffApp * GetApp()
                        {return (CDiffApp *)AfxGetApp(); }
```

4. Define a public method to a pointer to the main frame.

```
CMainFrame * GetMainFrame()
                {return (CMainFrame *)m_pMainWnd; }
```

Note that the purpose of steps 3 and 4 is to simplify writing the code for the rest of this lab.

5. Save Diff.H.

➢ **Initialize the file members and set the views to text only**

1. Open DiffDoc.Cpp and scroll to **CDiffDoc::CDiffDoc**.
2. Initialize m_File1 and m_File2 as empty strings.

```
m_File1 = _T("");
m_File2 = _T("");
```

3. The m_bRTF member of **CRichEditDoc** defines whether the streams should store formatting. Because you will be dealing with only text, set this member to FALSE.

```
m_bRTF  = FALSE;
```

4. Save DiffDoc.Cpp.

➢ **Add the function to display the files in the views**

1. Right-click **CDiffDoc** in ClassView and add the following public function:

```
void RunComparison(LPCSTR lpszFile1, LPCSTR lpszFile2)
```

2. Open DiffDoc.Cpp. Include DiffView.H, Splitter.H, and MainFrm.H after the statement that includes DiffDoc.H.

```
#include "diffdoc.h"
#include "diffview.h"
#include "splitter.h"
#include "mainfrm.h"
```

➢ **Code the RunComparison function**

1. Get the splitter.

```
CMainFrame * pFrame = CDiffApp::GetApp()->GetMainFrame();
CSplitter * pSplitter = pFrame->GetSplitter();
```

2. Get the first pane.

```
CDiffView * pView;
pView = (CDiffView *)pSplitter->GetPane(0,0);
```

3. Stream the first file into the first pane.

```
CFile   file(lpszFile1, CFile::modeRead);
CArchive ar(&file, CArchive::load);
pView->Serialize(ar);
```

4. Repeat the procedure for the second pane.
5. Mark the document as “clean” so as to not save it.

```
SetModifiedFlag(FALSE);
```

6. Save DiffDoc.Cpp.

The complete **RunComparison** function, including some error-checking, is shown in this sample code.

```
void CDiffDoc::RunComparison (LPCSTR lpszFile1,
                              LPCSTR lpszFile2)
{
    CMainFrame * pFrame = CDiffApp::GetApp()->GetMainFrame();

    if(pFrame)
    {
        CSplitter * pSplitter = pFrame->GetSplitter();
```

```
        if(pSplitter)
        {
            CDiffView * pView;

            pView = (CDiffView *)pSplitter->GetPane(0,0);

            if(pView)
            {
                CFile   file(lpszFile1, CFile::modeRead);
                CArchive ar(&file, CArchive::load);
                pView->Serialize(ar);
            }

            pView = (CDiffView *)pSplitter->GetPane(0,1);

            if(pView)
            {
                CFile   file(lpszFile2, CFile::modeRead);
                CArchive ar(&file, CArchive::load);
                pView->Serialize(ar);
            }
            //Flag as clean so that we won't get
            //prompted to save
            SetModifiedFlag(FALSE);
        }
    }
}
```

➢ **Call RunComparison from OnFileOpen**

1. Scroll to the end of **CDiffDoc::OnFileOpen**.
2. Add a call to **RunComparison** with the two files.

```
RunComparison (m_File1, m_File2);
```

3. Save DiffDoc.Cpp.

➢ **Set the Rich Edit control to read only**

1. Open ClassWizard.
2. Add a handler for **OnInitialUpdate** to **CDiffView**.
3. Open DiffView.Cpp.
4. Add a call to **SetReadOnly** to the end of **OnInitialUpdate**.

```
GetRichEditCtrl().SetReadOnly();
```

5. Save DiffView.Cpp.

➢ **Make the changes to CMainFrame**

1. Open MainFrm.H.
2. Add a public attribute method to return the splitter.

```
CSplitter * GetSplitter() { return &m_wndSplitter; }
```

3. Save MainFrm.H.
4. Open MainFrm.Cpp to **CMainFrame::OnCreateClient**.
5. Comment out SetActiveView.

```
//SetActiveView((CView *)m_wndSplitter.GetPane(0,1));
```

6. Save MainFrm.Cpp.

➢ **Build and test the application**

The completed code for this exercise is in \Labs\C07\Lab02\Ex03.

Exercise 4: Managing Drag and Drop

Continue with the files you created in Exercise 3 or, if you do not have a starting point for this exercise, the code that forms the basis for this exercise is in \Labs\C07\Lab02\Ex03.

When you run the results of Exercise 3, you will find that Drag and Drop from the Explorer is enabled as part of the Rich Edit control, but as OLE rather than File Open as a user may expect. In this exercise, you will modify the behavior of **CRichEdit** so that it will open the file rather than link the file. For purposes of this exercise, you will process dropped files only if there are two or more files dropped. Ignore any additional files beyond the first two.

➢ **Add a handler for WM_DROPFILES to CMainFrame using ClassWizard**

➢ **Implement CMainFrame::OnDropFiles**

1. Determine the number of files being dropped.

```
UINT nFileCount =
    ::DragQueryFile (hDropInfo,0xFFFFFFFF, NULL, 0);
```

2. Determine whether two or more files have been dropped.

```
if (nFileCount >= 2)
```

➢ **Edit CMainFrame::OnDropFiles to get the first two dropped files and pass them to CDiffDoc::RunComparison**

1. Get the path of the first file.

```
CString File1;

::DragQueryFile(hDropInfo,
                0,
                File1.GetBufferSetLength(_MAX_PATH),
                _MAX_PATH);
File1.ReleaseBuffer();
```

2. Get the path for the second file.
3. Call **CDiffDoc::RunComparison**.

```
((CDiffDoc *)GetActiveDocument())->
                RunComparison(File1, File2);
```

4. Finish drag and drop processing.

```
::DragFinish(hDropInfo);
```

The complete function is shown in this sample code.

```
void CMainFrame::OnDropFiles(HDROP hDropInfo)
{
    UINT nFileCount = ::DragQueryFile (hDropInfo,
                                       0xFFFFFFFF,
                                       NULL, 0);
    ASSERT(nFileCount !=0);
```

```
    // We must have at least two files for this function;
    // we will grab only the first two

    if (nFileCount >= 2)
    {
        CString File1;
        CString File2;

        ::DragQueryFile(hDropInfo,
                        0,
                        File1.GetBufferSetLength(_MAX_PATH),
                        _MAX_PATH);
        File1.ReleaseBuffer();

        ::DragQueryFile(hDropInfo,
                        1,
                        File2.GetBufferSetLength(_MAX_PATH),
                        _MAX_PATH);
        File2.ReleaseBuffer();

        ((CDiffDoc *)GetActiveDocument())->
                            RunComparison(File1, File2);

        ::DragFinish(hDropInfo);
    }
}
```

➢ **Add a handler for WM_DROPFILES to CDiffView using ClassWizard**

➢ **Pass the WM_DROPFILES message to CMainFrame from CDiffView**

1. In **CDiffView::OnDropFiles**, get the main frame window.

```
CDiffApp::GetApp()->GetMainFrame()
```

2. Send WM_DROPFILES to it.

```
->SendMessage(  WM_DROPFILES,
                    (WPARAM)hDropInfo );
```

3. Comment out the default **OnDropFiles** behavior so OLE embedding will not be invoked.

```
//CrichEditView::OnDropFiles(hDropInfo);
```

The complete function is shown in this sample code.

```
void CDiffView::OnDropFiles(HDROP hDropInfo)
{

    CDiffApp::GetApp()->GetMainFrame()->
                SendMessage (WM_DROPFILES,
                             (WPARAM)hDropInfo);
    // Don't call the default--we don't want OLE embedding
    //behavior
    //CRichEditView::OnDropFiles(hDropInfo);
}
```

4. At the top of DiffView.Cpp, include Splitter.H and MainFrm.H.

```
#include "splitter.h"
#include "mainfrm.h"
```

➢ **Build and test the application**

The completed code for this exercise is in \Labs\C07\Lab02\Ex04.

Optional Exercise

You can complete this exercise if you want more practice working with the concepts and techniques presented in this chapter. Write code to accept one file in the drag and drop. Next, if there is a second file to compare to the first, run the comparison. Otherwise, wait for a drop in the other pane of the splitter.