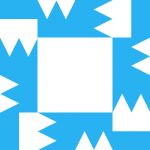
Creating a Simple Drives Explorer Program

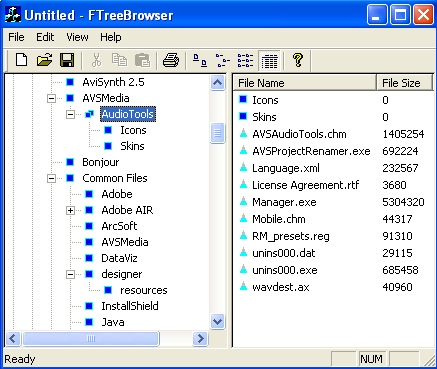


[**Ben Aldhouse**](https://www.codeproject.com/script/Membership/View.aspx?mid=2690007), 5 May 2009

|  |  |  |
| --- | --- | --- |
| https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-fill-md.png  https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-empty-md.png | | 4.44 (10 votes) |
|  | |  |  |  | | --- | --- | --- | | Rate this: | [vote 1vote 2vote 3vote 4vote 5](https://www.codeproject.com/Articles/13366/Creating-a-Simple-Drives-Explorer-Program) |  | | | |

A project using the Windows Explorer Framework and some API stuff

* [**Download demo project - 28.9 KB**](https://www.codeproject.com/KB/tree/SimpleDrvExplorer/FTreeBrowser.zip)
* [**Download demo application - 10.81 KB**](https://www.codeproject.com/KB/tree/SimpleDrvExplorer/FTreeBrowserApp.zip)



Introduction

In this article, I show you how to recreate some of the browsing functionality that is seen in Windows Explorer.

There is a tree control on the left with which the user could navigate folders - causing the files in whichever folder was currently selected to appear in the right hand pane.

Of course, if you want to have a program that enhances Windows Explorer, then you should probably look at working with the Shell. I haven't done that because I want to explore the way the user interface works and potentially build similar interfaces for other applications which navigate other sorts of information than folders and files in storage devices.

Here is a description of how to build my project from scratch...

Select 'File' then 'New'. Give your project the name FTreeBrowser.

Click 'OK'. Choose whether you want a single document or a multiple document interface. Check the 'Document/View architecture support?' (When I did this, it was already checked).

Click through steps 2, 3 and 4, no action is required here. In step 5, select the 'Windows Explorer' option under the question 'What style of project would you like?'

Click 'Next'.

Next you get the option to change some class and file names. For CFTreeBrowserView, use the following names instead of the automatically generated ones:

Hide   Copy Code

CFTreeBrowserView

Class Name: CRightView

Header File: RightView.h

Implementation File: RightView.cpp

Base Class: CListView

Click 'OK' in the 'New Project Information' pane.

In the Workspace pane, click on the FileView tab. Open up the *Header Files* folder and open the file *FTreeBrowserDoc.h*. In this file, insert the lines...

Hide   Copy Code

class CLeftView;

class CRightView;

... and...

Hide   Copy Code

CLeftView \*pLeftView;

CRightView \*pRightView;

... in the positions shown in the code section below:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

*// FTreeBrowserDoc.h : interface of the CFTreeBrowserDoc class*

*//*

///*/////////////////////////////////////////////////////////////////*

....

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

class CLeftView;

class CRightView;

class CFTreeBrowserDoc : public CDocument

{

protected: *// create from serialization only*

CFTreeBrowserDoc();

DECLARE\_DYNCREATE(CFTreeBrowserDoc)

*// Attributes*

public:

*// Operations*

public:

CLeftView \*pLeftView;

CRightView \*pRightView;

*// Overrides*

*// ClassWizard generated virtual function overrides*

*//{{AFX\_VIRTUAL(CFTreeBrowserDoc)*

public:

virtual BOOL OnNewDocument();

virtual void Serialize(CArchive& ar);

*//}}AFX\_VIRTUAL*

*// Implementation*

public:

virtual ~CFTreeBrowserDoc();

#ifdef \_DEBUG

virtual void AssertValid() const;

virtual void Dump(CDumpContext& dc) const;

#endif

protected:

*// Generated message map functions*

protected:

*//{{AFX\_MSG(CFTreeBrowserDoc)*

*// NOTE - the ClassWizard will add and remove member*

functions here.

*// DO NOT EDIT what you see in these blocks of*

generated code !

*//}}AFX\_MSG*

DECLARE\_MESSAGE\_MAP()

};

Open up the ClassWizard (press CTRL-W), under the Message Maps tab, select CLeftView in the 'Class name' list, CLeftView in the 'Object IDs' list, WM\_CREATE in the 'Messages:' list. If you double-click on WM\_CREATE, the OnCreate function is created and highlighted for you in the member functions list. Click on 'OK'.

Go back to the workspace, click the ClassView tab and open up the CLeftView class. If you double-click on the OnCreate function, the new function will be presented to you in the file editor pane.

Add the line GetDocument()->pLeftView = this; so that the function appears as below:

Hide   Copy Code

int CLeftView::OnCreate(LPCREATESTRUCT lpCreateStruct)

{

if (CTreeView::OnCreate(lpCreateStruct) == -1)

return -1;

*// TODO: Add your specialized creation code here*

GetDocument()->pLeftView = this;

return 0;

}

Repeat the last few steps from 'Open up the ClassWizard' but this time select CRightView in the 'Class name' list and CRightView in the 'Object IDs' list. Next you will need to add the line GetDocument()->pRightView = this; to the CRightView::OnCreate function as follows:

Hide   Copy Code

int CRightView::OnCreate(LPCREATESTRUCT lpCreateStruct)

{

if (CListView::OnCreate(lpCreateStruct) == -1)

return -1;

*// TODO: Add your specialized creation code here*

GetDocument()->pRightView = this;

return 0;

}

Next, add five icons. You create these side by side in a bitmap 16 pixels high and 80 pixels wide. The first two will be used for the root nodes (unselected and selected respectively), the third and fourth icons will be used for branch nodes (unselected and selected respectively) and the last icon will be used for leaf nodes. You can do this by clicking 'Insert' on the main menu and then selecting resource. Double click on 'Bitmap'.

Right hand click on the icon for the bitmap that has appeared in the resource tab of the ClassView pane. Change the ID of the bitmap to IDB\_TREE\_BMP. By using the grab buttons at the edge of the pixel grid, set its width to 90 and its height to 16. You don't have to count the pixels, the size of the bitmap appears in the bottom right hand corner of the IDE main frame. Draw your icons in boxes of 16 X 16 pixels across the bitmap.

Once you have made your icons, you make them accessible to your tree control. The first step in this process is declaring a CImageList variable. Open up ClassView. Right click over CLeftView and click on 'Add Member Variable'. Declare your variable type as CImageList, variable Name as m\_TreeImages and access as Public.

Now to populate the root items in the tree we create a function called CreateRoots. In ClassView, right-click CLeftView-> Add Member Function. Set the return type to void. Set the Function Declaration to CreateRoots. Leave the access as Public.

Put the following code in the new function:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

void CLeftView::CreateRoots()

{

*// If there is anything in the tree, remove it*

GetTreeCtrl().DeleteAllItems();

CTreeCtrl &ctlDrives = this->GetTreeCtrl();

m\_TreeImages.Create(IDB\_TREE\_BMP, 16, 1, RGB(255, 255, 255));

ctlDrives.SetImageList(&m\_TreeImages, TVSIL\_NORMAL);

HTREEITEM hRoot;

char \* strBuffer= NULL;

CString strMessage;

int nPos = 0;

UINT nCount = 0;

CString strDrive = "?:\\";

DWORD dwDriveList = ::GetLogicalDrives ();

CString cTmp;

while (dwDriveList) {

if (dwDriveList & 1) {

cTmp = strDrive;

strDrive.SetAt (0, 0x41 + nPos);

strDrive = strDrive.Left(2);

hRoot = ctlDrives.InsertItem(strDrive,0, 1);

}

dwDriveList >>= 1;

nPos++;

}

}

We want this function to be called when the application is started up or whenever a new document is made. In ClassView, open up the CFTreeBrowserDoc node and double-click on the OnNewDocument node.

Place a call to the CreateRoots function here. The OnNewDocument function should now look like this:

Hide   Copy Code

BOOL CFTreeBrowserDoc::OnNewDocument()

{

if (!CDocument::OnNewDocument())

return FALSE;

*// TODO: add reinitialization code here*

*// (SDI documents will reuse this document)*

this->pLeftView->CreateRoots();

return TRUE;

}

Next, scroll up to the top of the file and add an #include line for *LeftView.h*.

Hide   Copy Code

*// FTreeBrowserDoc.cpp : implementation of the CFTreeBrowserDoc class*

*//*

#include "stdafx.h"

#include "FTreeBrowser.h"

#include "FTreeBrowserDoc.h"

#include "LeftView.h"

You should now be able to compile your project. The resulting application should show your system drives displayed in the left hand pane - but nothing else yet.

Go back to the CLeftView class and add a call to the tree's ModifyStyle method in the OnInitialUpdatefunction.

Hide   Copy Code

void CLeftView::OnInitialUpdate()

{

CTreeView::OnInitialUpdate();

*// TODO: You may populate your TreeView with items by directly accessing*

*// its tree control through a call to GetTreeCtrl().*

GetTreeCtrl().ModifyStyle(NULL, TVS\_HASBUTTONS | TVS\_HASLINES | TVS\_LINESATROOT);

}

To set the width of the left pane, open up the CMainFrame class (or CChildFrame class if you are writing an MDI application) and edit the OnCreateClient function. If you had created an SDI application, rather than an MDI application, you would find this function in the CMainFrame class.

To set a width of 250, place '250' in the conditional line below, as follows:

Hide   Copy Code

if (!m\_wndSplitter.CreateView(0, 0, RUNTIME\_CLASS(CLeftView),

CSize(250, 100), pContext) ||

!m\_wndSplitter.CreateView(0, 1, RUNTIME\_CLASS(CRightView),

CSize(100, 100), pContext))

Now we will pay some attention to the right hand pane. First we'll add a function to clear the list control. In ClassView, right-click CRightView and select AddMemberFunction. Set the return type to void and the function name to ResetFiles. Keep the function's access as public and click enter. You can now add the following code:

Hide   Copy Code

void CRightView::ResetFiles()

{

CListCtrl &ctlLView = GetListCtrl();

ctlLView.DeleteAllItems();

while(ctlLView.DeleteColumn(0))

;

UpdateWindow();

}

It is now time to add the functionality which will display the contents of whatever folder is selected in the left pane. The contents are displayed in the right pane and the new function is added to CRightView. The type of this function is void, the implementation is DisplayFiles(LPTSTR Path) and the access is public.

Fill the function with the following code:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

void CRightView::DisplayFiles(LPTSTR Path)

{

CListCtrl &ctlRightView = this->GetListCtrl();

ResetFiles();

ctlRightView.InsertColumn(0, \_T("File Name"), LVCFMT\_LEFT, 160);

ctlRightView.InsertColumn(1, \_T("File Size"), LVCFMT\_LEFT, 40);

*//AfxMessageBox(Path);*

WIN32\_FIND\_DATA FindFileData;

HANDLE hFind;

int nItem;

hFind = FindFirstFile(Path, &FindFileData);

int n = 0;

if (hFind != INVALID\_HANDLE\_VALUE){

do {

if (FindFileData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY

&& FindFileData.cFileName != CString (".")

&& FindFileData.cFileName != CString (".."))

{

++n;

nItem = ctlRightView.InsertItem(n, "File",2);

ctlRightView.SetItemText(nItem, 0, FindFileData.cFileName);

long lFSize = FindFileData.nFileSizeLow;

CString strFSize="";

strFSize.Format("%d",lFSize);

ctlRightView.SetItemText(nItem, 1, strFSize.GetBuffer(1));

}

}while((::WaitForSingleObject(m\_hStopEvent, 0) !=

WAIT\_OBJECT\_0) && (::FindNextFile(hFind, &FindFileData)));

::FindClose(hFind);;

hFind = FindFirstFile(Path, &FindFileData);

*//n = 0;*

do {

if (!(FindFileData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY)){

++n;

nItem = ctlRightView.InsertItem(n, "File",4);

ctlRightView.SetItemText(nItem, 0, FindFileData.cFileName);

long lFSize = FindFileData.nFileSizeLow;

CString strFSize="";

strFSize.Format("%d",lFSize);

ctlRightView.SetItemText(nItem, 1, strFSize.GetBuffer(1));

}

}while((::WaitForSingleObject(m\_hStopEvent, 0) !=

WAIT\_OBJECT\_0) && (::FindNextFile(hFind, &FindFileData)));

::FindClose(hFind);;

}

}

Before you can compile and run the application at this stage, you need to add the m\_hStopEvent handle. To do this, you need to open up the file *RightView.h* and add the line HANDLE m\_hStopEvent; to the class definition in the private: list as can be seen in this listing of the file:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

*// RightView.h : interface of the CRightView class*

*//*

///*////////////////////////////////////////////////////////////////*

...

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

class CRightView : public CListView

{

protected: *// create from serialization only*

CRightView();

DECLARE\_DYNCREATE(CRightView);

*// Attributes*

public:

CExerciseDoc\* GetDocument();

*// Operations*

public:

*// Overrides*

*// ClassWizard generated virtual function overrides*

*//{{AFX\_VIRTUAL(CRightView)*

public:

virtual void OnDraw(CDC\* pDC); *// overridden to draw this view*

virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

protected:

virtual void OnInitialUpdate(); *// called first time after construct*

virtual BOOL OnPreparePrinting(CPrintInfo\* pInfo);

virtual void OnBeginPrinting(CDC\* pDC, CPrintInfo\* pInfo);

virtual void OnEndPrinting(CDC\* pDC, CPrintInfo\* pInfo);

*//}}AFX\_VIRTUAL*

*// Implementation*

public:

CImageList m\_ListImages;

void DisplayFiles(LPTSTR Path);

void DisplaySeason(CString League, CString Season);

void DisplayLeague(CString League);

void DisplayLeagues();

void ResetLeagues();

virtual ~CRightView();

#ifdef \_DEBUG

virtual void AssertValid() const;

virtual void Dump(CDumpContext& dc) const;

#endif

protected:

*// Generated message map functions*

protected:

*//{{AFX\_MSG(CRightView)*

afx\_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);

afx\_msg void OnDblclk(NMHDR\* pNMHDR, LRESULT\* pResult);

*//}}AFX\_MSG*

afx\_msg void OnStyleChanged(int nStyleType, LPSTYLESTRUCT lpStyleStruct);

DECLARE\_MESSAGE\_MAP()

private:

HANDLE m\_hStopEvent;

};

#ifndef \_DEBUG // debug version in RightView.cpp

inline CExerciseDoc\* CRightView::GetDocument()

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

#endif

///*///////////////////////////////////////////////////////////////////*

*//{{AFX\_INSERT\_LOCATION}}*

*// Microsoft Visual C++ will insert additional declarations*

*// immediately before the previous line.*

...

Next we look at what happens when a folder is selected (opened) in the left hand pane. Some of this functionality is required when a folder is opened in the right hand pane, so we will make a separate function outside of the LeftView event handler which deals with this section of the code. This function is void, has a declaration of OpenFolder(CString CStrPath) and should be left as public.

Again, the lines...

Hide   Copy Code

private:

HANDLE m\_hStopEvent;

... must be inserted at the end of the class definition in the *LeftView.h* file.

Now that we have the necessary functions, we can react when a node changes in the tree view.  
Using either the ClassWizard, the Messages section of the Properties, or whatever, generate a TVN\_SELCHANGEDmessage for the CLeftView class. Use the suggested function name of OnSelChanged.

Fill out this function as follows:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

void CLeftView::OnSelchanged(NMHDR\* pNMHDR, LRESULT\* pResult)

{

NM\_TREEVIEW\* pNMTreeView = (NM\_TREEVIEW\*)pNMHDR;

*// TODO: Add your control notification handler code here*

\*pResult = 0;

CFTreeBrowserDoc \*pDoc = GetDocument();

*// Get a reference to the tree control*

CTreeCtrl &ctlFiles = this->GetTreeCtrl();

*// Find out what item is selected in the tree*

HTREEITEM nodSelected = ctlFiles.GetSelectedItem();

*// Get the string of the selected node*

CString strSelected = ctlFiles.GetItemText(nodSelected);

HTREEITEM nodParent = nodSelected;

*//Build the full path with wild cards*

do {

nodParent = ctlFiles.GetParentItem(nodParent);

if (nodParent!=NULL)

strSelected = ctlFiles.GetItemText(nodParent) + "\\" + strSelected;

} while (nodParent != NULL);

CString strSearchPath = strSelected + "\\\*.\*";

pDoc->pRightView->DisplayFiles(strSearchPath.GetBuffer(1));

OpenFolder(strSelected);

}

You will need to make sure you have an #include line for *Rightview.h* at the top of the file now, as follows:

Hide   Copy Code

*// LeftView.cpp : implementation of the CLeftView class*

*//*

#include "stdafx.h"

#include "FTreeBrowser.h"

#include "FTreeBrowserDoc.h"

#include "LeftView.h"

#include "RightView.h"

You will notice, if you execute the program at this point, that there are no icons associated with files or folders in the right hand pane yet. The code in the function DisplayFiles is already set to use the third and fifth icons from the icon list associated with the list control (for folders and files respectively. Currently there is no icon list associated with the control. To remedy this, open up CRightView::OnInitialUpdate() function and add the lines as have been added below:

Hide   Copy Code

void CRightView::OnInitialUpdate()

{

CListView::OnInitialUpdate();

*// TODO: You may populate your ListView with items by directly accessing*

*// its list control through a call to GetListCtrl().*

CListCtrl &ctlFiles = this->GetListCtrl();

ctlFiles.ModifyStyle(NULL, LVS\_REPORT);

m\_ListImages.Create(IDB\_TREE\_BMP, 16, 1, RGB(255, 255, 255));

ctlFiles.SetImageList(&m\_ListImages, LVSIL\_SMALL);

}

Before this works, you need to add a class variable to CRightView of type CImageList, name m\_ListImagesand public access.

To make this application behave a little bit more like Microsoft's Windows Explorer, you make a folder clicked on in the right pane become the selected folder in the left pane. To do this, create a function for CRightView to handle the =NM\_DBLCLK message. Fill out the function as follows:

Hide   Copy Code

void CRightView::OnDblclk(NMHDR\* pNMHDR, LRESULT\* pResult)

{

*// TODO: Add your control notification handler code here*

\*pResult = 0;

CExerciseDoc \*pDoc = GetDocument();

CListCtrl &ctlRightView = this->GetListCtrl();

int nItem = ctlRightView.GetSelectionMark();

CString string= ctlRightView.GetItemText(nItem,0);

pDoc->pLeftView->SetFocus();

pDoc->pLeftView->OnRightViewFolderSelected(string, nItem);

}

Next you should add #include "LeftView.h" at the top of the file so the #includes section looks like this:

Hide   Copy Code

*// RightView.cpp : implementation of the CRightView class*

*//*

#include "stdafx.h"

#include "FTreeBrowser.h"

#include "LeftView.h"

#include "FTreeBrowserDoc.h"

#include "RightView.h"

Then you need to create the function CLeftView::OnRightViewFolderSelected(...). Make the return type void, access public and declaration OnRightViewFolderSelected(CString strPath, UINT index). You can now fill out the function with the following code:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

void CLeftView::OnRightViewFolderSelected(CString strPath, UINT

index)

{

CFTreeBrowserDoc \*pDoc = GetDocument();

*// Get a reference to the tree control*

CTreeCtrl &ctlFolders = this->GetTreeCtrl();

*// Find out what item is selected in the tree*

HTREEITEM nodSelected = ctlFolders.GetSelectedItem();

*//Open up the branch*

ctlFolders.Expand(nodSelected, TVE\_EXPAND);

int count=0;

HTREEITEM nodChild;

nodChild = ctlFolders.GetChildItem(nodSelected);

if (index > 0){

do{

nodChild = ctlFolders.GetNextItem(nodChild,TVGN\_NEXT);

++count;

}while (count < (int)index);

}

if (nodChild != NULL)

{

ctlFolders.SelectItem(nodChild);

ctlFolders.Expand(nodChild, TVE\_EXPAND);

nodSelected = nodChild;

*// Get the string of the selected node*

CString strSelected = ctlFolders.GetItemText(nodSelected);

HTREEITEM nodParent = nodSelected;

*//Build the full path with wild cards*

do {

nodParent = ctlFolders.GetParentItem(nodParent);

if (nodParent!=NULL)

strSelected = ctlFolders.GetItemText(nodParent) +

"\\" + strSelected;

} while (nodParent != NULL);

CString strSearchPath = strSelected + "\\\*.\*";

pDoc->pRightView->DisplayFiles(strSearchPath.GetBuffer(1));

}

}

Lastly, you need to populate the CLeftView::OpenFolder(...) function as follows...

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

void CLeftView::OpenFolder(CString CStrPath)

{

CTreeCtrl &ctlFolders = this->GetTreeCtrl();

HTREEITEM hRoot;

HTREEITEM hFolder;

hRoot = ctlFolders.GetSelectedItem();

HTREEITEM hChild = ctlFolders.GetChildItem(hRoot);

*// Clear the selected node*

while(hChild != 0){

ctlFolders.DeleteItem(hChild);

hChild = ctlFolders.GetChildItem(hRoot);

}

WIN32\_FIND\_DATA FindFileData;

HANDLE hFind;

CStrPath = CStrPath + "\\\*.\*";

hFind = FindFirstFile(CStrPath, &FindFileData);

if (hFind != INVALID\_HANDLE\_VALUE){

do {

long lFSize = FindFileData.nFileSizeLow;

CString strFSize="";

*//Want folders that aren't . and ..*

if (FindFileData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY

&& FindFileData.cFileName != CString (".")

&& FindFileData.cFileName != CString (".."))

hFolder = ctlFolders.InsertItem

(FindFileData.cFileName,2,3,hRoot);

}while((::WaitForSingleObject(m\_hStopEvent, 0) != WAIT\_OBJECT\_0)

&& (::FindNextFile(hFind, &FindFileData)));

::FindClose(hFind);;

}

}

That's it.

References

* [This article](http://www.functionx.com/visualc/applications/explorer.htm) on the functionx.com web site describes a Windows Explorer application. I found this article very useful in its straightforward style and I learned a lot about how to work with the Windows Explorer Framework from it. The application isn't, however, anything to do with browsing system files. Instead, it deals with browsing hard coded data.
* I learned about how to get directory listings in ['File and Directory Enumeration'](http://www.codeproject.com/cpp/recursedir.asp) by Andreas Saurwein Franci Gonçalves.

Article History

* 2009-05-04: Updated article
  + Hopefully, improved the weak introduction to the article
  + Renamed at least one badly named variable
  + Provided a description of a function CLeftView::OpenFolder(...) that had been omitted
  + Rewritten the application as an SDI instead of an MDI (This involves only a slight change in the article and I include instructions on how to create the application as an MDI.)
  + Moved the references to the end of the article and tidied them up
* 2006-03-09: Initial article

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License (CPOL)](http://www.codeproject.com/info/cpol10.aspx)