Applying a filter to the contents of an Explorer Browser

devblogs.microsoft.com/oldnewthing/20141103-00

November 3, 2014

Raymond Chen

Today's Little Program hosts an Explorer Browser but filters the contents to remove DLL files. You can, of course, substitute your own filter. (For example, maybe you want to show only files that changed since the last time the user ran your program, or you might want a view of My Computer but filtered to show only removable drives.)

Remember that Little Programs do little to no error checking, and they don't necessarily demonstrate the best programming style. They're just quick demonstrations. Today's smart pointer library is... (rolls dice) ... WRL!

Start with our minimal explorer browser program and make these changes.

```
#include <shlwapi.h> // PathFindExtensionW
#include <wrl\client.h>
#include <wrl\implements.h>
using namespace Microsoft::WRL;
class FolderFilterNoDLLs :
    public RuntimeClass<RuntimeClassFlags<ClassicCom>,
                        IFolderFilter>
{
 // *** IFolderFilter ***
 IFACEMETHODIMP GetEnumFlags(IShellFolder *psf,
    PCIDLIST_ABSOLUTE pidlFolder, HWND *phwnd,
    DWORD *pgrfFlags) { return S_OK; }
 IFACEMETHODIMP ShouldShow(IShellFolder *psf,
    PCIDLIST_ABSOLUTE pidlFolder,
    PCUITEMID_CHILD pidlItem)
 {
  BOOL fShow = TRUE;
  ComPtr<IShellItem> spsi;
  HRESULT hr = SHCreateItemWithParent(pidlFolder, psf, pidlItem,
                                      IID_PPV_ARGS(&spsi));
  if (SUCCEEDED(hr)) {
   SFGAOF sfgaof;
   hr = spsi->GetAttributes(SFGAO_FILESYSTEM | SFGAO_FOLDER,
                            &sfgaof);
   if (SUCCEEDED(hr) && sfgaof == SFGAO_FILESYSTEM) {
    LPWSTR pszName;
    hr = spsi->GetDisplayName(SIGDN_PARENTRELATIVEPARSING,
                              &pszName);
    if (SUCCEEDED(hr))
    {
     fShow = CompareStringOrdinal(
                 PathFindExtensionW(pszName), -1,
                 L".dll", -1, TRUE) != CSTR_EQUAL;
     CoTaskMemFree(pszName);
    }
   }
  }
  if (SUCCEEDED(hr)) hr = fShow ? S_OK : S_FALSE;
  return hr;
 }
};
```

The real work happens in the `ShouldShow` method.

- Create an `IShellItem` because it's more convenient.
- Query the `SFGAO_FILESYSTEM` and `SFGAO_FOLDER` attributes.
- If the attributes say "Yes, it's a file system object, and no, it's not a folder"...
    - Get the display name.
    - If the display name ends in `.dll`, then hide the item.

All that's left is to plug this into the Explorer Browser.

```
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
    BOOL fSuccess = FALSE;
    RECT rc;
    PIDLIST_ABSOLUTE pidl = NULL;
    ComPtr<IFolderFilter> spff;
    ComPtr<IFolderFilterSite> spffs;
    if (SUCCEEDED(CoCreateInstance(CLSID_ExplorerBrowser, NULL,
                        CLSCTX_INPROC, IID_PPV_ARGS(&g_peb))) &&
        GetClientRect(hwnd, &rc) &&
        SUCCEEDED(g_peb->Initialize(hwnd, &rc, NULL)) &&
        SUCCEEDED(g_peb->SetOptions(EBO_NAVIGATEONCE)) &&
        SUCCEEDED(MakeAndInitialize<FolderFilterNoDLLs>(&spff)) &&
        SUCCEEDED(g_peb->QueryInterface(IID_PPV_ARGS(&spffs))) &&
        SUCCEEDED(spffs->SetFilter(spff.Get())) &&
        SUCCEEDED(SHParseDisplayName(
                        L"C:\\Program Files\\Internet Explorer",
                                     NULL, &pidl, 0, NULL)) &&
        SUCCEEDED(g_peb->BrowseToIDList(pidl, SBSP_ABSOLUTE))) {
        fSuccess = TRUE;
    }
    ILFree(pidl);
    return fSuccess;
}
```

We apply the filter to the `IExplorerBrowser` by querying for `IFolderFilterSite` and using `IFolderFilterSite::SetFilter` to attach our "no DLLs" filter.

Bonus reading: Filtering the folders that appear in the Browse for Folder dialog.

Raymond Chen

Follow