Filtering the folders that appear in the Browse for Folder dialog

devblogs.microsoft.com/oldnewthing/20131014-00

October 14, 2013

Raymond Chen

Today's Little Program applies an arbitrary filter to the *Browse for Folder* dialog: We will filter out drives smaller than 512GB. Hey, remember, I said that Little Programs do not need motivation.

Today's smart pointer class library is... (rolls dice)... WRL!

```
#define STRICT
#define STRICT_TYPED_ITEMIDS
#include <windows.h>
#include <shlobj.h>
#include <propkey.h>
#include <propvarutil.h>
#include <wrl/implements.h>
using namespace Microsoft::WRL;

struct ComVariant : public VARIANT
{
    ComVariant() { VariantInit(this); }
    ~ComVariant() { VariantClear(this); }
};
```

WRL does not come with a "smart `VARIANT`" class analogous to ATL's `CComVariant`, so we provide our own very simple one.

```
class CFunnyFilter :
  public RuntimeClass<
                RuntimeClassFlags<RuntimeClassType::ClassicCom>,
                IFolderFilter>
{
public:
  // *** IFolderFilter ***
  IFACEMETHODIMP ShouldShow(
        IShellFolder* psf,
        PCIDLIST_ABSOLUTE pidlFolder,
        PCUITEMID_CHILD pidlItem)
  {
    if (!IsMyComputerFolder(psf)) return S_OK;

    ComPtr<IShellFolder2> spsf2;
    if (FAILED(ComPtr<IUnknown>(psf).As(&spsf2))) {
      return S_OK;
    }

    ComVariant svt;
    if (FAILED(spsf2->GetDetailsEx(pidlItem,
                                   &PKEY_Capacity, &svt))) {
      return S_OK;
    }

    if (VariantToUInt64WithDefault(svt, 0) >=
                             512ULL*1024ULL*1024ULL*1024ULL) {
      return S_OK;
    }

    return S_FALSE;
  }

  IFACEMETHODIMP GetEnumFlags(
        IShellFolder* psf,
        PCIDLIST_ABSOLUTE pidlFolder,
        HWND *phwnd,
        DWORD *pgrfFlags) { return S_OK; }
};
```

Our custom `ShouldShow` method first checks if we are showing children of *My Computer*. If not, then we allow the item to pass through the filter.

Next, we convert the folder to `IShellFolder2`. If we can't, then we allow the item to pass through the filter. (Arbitrary choice.)

Next, we ask for the capacity of the item. If we can't (no media in drive, or it's not a drive in the first place), then we allow the item to pass through the filter. (Arbitrary choice.)

Next, we look at the capacity, and if it's at least 512GB, then we allow the item to pass through the filter.

Otherwise, we have a drive smaller than 512GB, so we filter it out.

That's it! Let's install this filter in the callback function:

```
int CALLBACK BrowseCallbackProc(HWND hwnd,
    UINT uMsg, LPARAM lParam, LPARAM lpData)
{
  switch (uMsg) {
  case BFFM_IUNKNOWN:
    if (lParam) {
      IUnknown *punk = reinterpret_cast<IUnknown*>(lParam);
      ComPtr<IFolderFilterSite> spFilterSite;
      if (SUCCEEDED(ComPtr<IUnknown>(punk).As(&spFilterSite))) {
        spFilterSite->SetFilter(Make<CFunnyFilter>().Get());
      }
    }
    break;
  }
  return 0;
}
```

When we get the `BFFM_IUNKNOWN` message, we convert the `IUnknown` (cast to `LPARAM`) to a `IFolderFilterSite` and tell it to apply our custom filter.

Okay, let's plug it in and see if smoke comes out.

```
int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   LPSTR lpCmdLine, int nShowCmd)
{
  CCoInitialize init;
  BROWSEINFO bi = { };
  TCHAR szDisplayName[MAX_PATH];
  bi.pszDisplayName = szDisplayName;
  bi.lpfn = BrowseCallbackProc;
  bi.ulFlags = BIF_NEWDIALOGSTYLE;
  PIDLIST_ABSOLUTE pidl = SHBrowseForFolder(&bi);
  CoTaskMemFree(pidl);
  return 0;
}
```

The tricky part here is that we have to pass the `BIF_NEWDIALOGSTYLE` flag because it is the new *Browse for Folder* dialog that sends the `BFFM_IUNKNOWN` message.

Raymond Chen

Follow