Customizing the window handle for item enumeration in IShellItem

devblogs.microsoft.com/oldnewthing/20170713-00

July 13, 2017

Raymond Chen

Some time ago, I showed how to customize the enumeration flags used when enumerating items with `IShellItem`. This controls the `grfFlags` parameter passed to the `IShellFolder::EnumObjects` method, but what about the `hwndOwner` parmaeter? How to do you customize the window handle?

The window handle for the enumeration comes from the site of the enumerator.

There's no real reason you were expected to know this.

Here's a Little Program that demonstrates. It is basically the program we used last time, but translated from ATL to WRL (because that lets me use the `RuntimeClass` template.)

```
#define STRICT
#include <windows.h>
#include <shlobj.h>
#include <shlwapi.h>
#include <knownfolders.h>
#include <wrl/client.h>
#include <wrl/implements.h>
#include <stdio.h>    // Horrors! Mixing stdio and C++!

namespace wrl = Microsoft::WRL;

class COleWindow : public wrl::RuntimeClass<
    wrl::RuntimeClassFlags<wrl::ClassicCom>, IOleWindow>
{
public:
  HRESULT RuntimeClassInitialize(HWND hwnd)
  {
    m_hwnd = hwnd;
    return S_OK;
  }

  STDMETHODIMP GetWindow(_Out_ HWND* phwnd)
  {
    *phwnd = m_hwnd;
    return S_OK;
  }

  STDMETHODIMP ContextSensitiveHelp(BOOL /* fEnterMode */)
  {
    return E_NOTIMPL;
  }

private:
  HWND m_hwnd;
};
```

The `COleWindow` class is a simple object which implements the `IOleWindow` interface. It coughs up the window handle you gave it at initialization.

We can use this object to provide a window for enumeration. Remember that Little Programs do little to no error checking.

```
int __cdecl wmain(int argc, wchar_t** argv)
{
  CCoInitialize init;

  if (argc < 2) return 0;

  HWND hwnd = CreateWindowW(L"static", L"Title",
      WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
      CW_USEDEFAULT, CW_USEDEFAULT,
      nullptr, nullptr, HINST_THISCOMPONENT, 0);

  wrl::ComPtr<IShellItem> folder;
  SHCreateItemFromParsingName(argv[1], nullptr,
                              IID_PPV_ARGS(&folder));

  wrl::ComPtr<IEnumShellItems> enumerator;
  folder->BindToHandler(nullptr, BHID_EnumItems,
                        IID_PPV_ARGS(&enumerator));

  wrl::ComPtr<IUnknown> site;
  wrl::MakeAndInitialize<COleWindow>(&site, hwnd);
  IUnknown_SetSite(enumerator.Get(), site.Get());

  wrl::ComPtr<IShellItem> item;
  while (enumerator->Next(1, item.ReleaseAndGetAddressOf(),
                          nullptr) == S_OK) {
    PWSTR name;
    item->GetDisplayName(SIGDN_NORMALDISPLAY, &name);
    wprintf(L"%s\n", name);
    CoTaskMemFree(name);
  }

  return 0;
}
```

First, we create a window so we have something to pass to `IShellFolder::EnumObjects`. In real life, this is the window you want to use for any UI that is displayed as part of the enumeration.

Next, we take the path from the command line and convert it to an `IShellItem`. This is not new.

Once we have the folder as an `IShellItem`, we ask for its enumerator. If you wanted to customize the flags passed to the `IShellFolder::EnumObjects` method, here's where you would pass a customizing `IBindCtx`.

And then the new part: Before calling any enumeration methods, we create a `COleWindow` object and set it as the enumerator's site. This tells the enumerator where to get its window from.

We have nothing else interesting in our site, but in a real program, your site would probably implement `IServiceProvider` in order to be a full-fledged site chain.

Finally, we use the enumerator in the usual manner and (for demonstration purposes) print out the names of the resulting objects.

Raymond Chen

Follow