Customizing item enumeration with IShellItem, the old-fashioned way

devblogs.microsoft.com/oldnewthing/20150202-00

February 2, 2015

Raymond Chen

If you are targeting Windows 8 or higher, you can use `STR_ENUM_ITEMS_FLAGS` to customize how shell items are enumerated. But what if you need to run on older systems, too?

In that case, you will need to drop to the lower-level `IShellFolder::EnumObjects` function, like we did before, and then reconstructe shell items from the low-level `IShell-Folder` and `ITEMID_CHILD`. (Note that the term “low-level” is used here only in a relative sense; it’s lower level than `IShellItem`.)

We can wrap that inside a helper class.

```
#define UNICODE
#define _UNICODE
#define STRICT
#define STRICT_TYPED_ITEMIDS
#include <windows.h>
#include <shlobj.h>
#include <atlbase.h>
CComModule _Module;
#include <atlcom.h>
#include <atlalloc.h>
#include <stdio.h>
#include <tchar.h>

class CEnumItemsWithSHCONTF :
    public CComObjectRoot,
    public IEnumShellItems
{
public:
  BEGIN_COM_MAP(CEnumItemsWithSHCONTF)
    COM_INTERFACE_ENTRY(IEnumShellItems)
  END_COM_MAP()

  static HRESULT Create(HWND hwnd, SHCONTF shcontf,
                  IShellItem *psiFolder, REFIID riid, void **ppv);

  STDMETHOD(Next)(ULONG celt, IShellItem **ppsi, ULONG *pceltFetched);
  STDMETHOD(Skip)(ULONG celt);
  STDMETHOD(Reset)();
  STDMETHOD(Clone)(IEnumShellItems **ppesiClone);

private:
  static HRESULT CreateRef1(CComObject<CEnumItemsWithSHCONTF> **ppObj);
  HRESULT Initialize(HWND hwnd, SHCONTF shcontf, IShellItem *psiFolder);
  HRESULT CloneFrom(CEnumItemsWithSHCONTF *pSource);
private:
  CComPtr<IShellFolder> m_spsfParent;
  CComPtr<IEnumIDList> m_speidl;
};

HRESULT CEnumItemsWithSHCONTF::CreateRef1(
    CComObject<CEnumItemsWithSHCONTF> **ppObj)
{
  CComObject<CEnumItemsWithSHCONTF> *pObj;
  HRESULT hr = CComObject<CEnumItemsWithSHCONTF>::
                      CreateInstance(&pObj);
  *ppObj = CComPtr<CComObject<CEnumItemsWithSHCONTF> >(pObj).Detach();
  return hr;
}

HRESULT CEnumItemsWithSHCONTF::Initialize(
  HWND hwnd, SHCONTF shcontf, IShellItem *psiFolder)
{
  HRESULT hr = psiFolder->BindToHandler(
    nullptr, BHID_SFObject, IID_PPV_ARGS(&m_spsfParent));
  if (SUCCEEDED(hr)) {
    hr = m_spsfParent->EnumObjects(hwnd, shcontf, &m_speidl);
  }
  return hr;
}

HRESULT CEnumItemsWithSHCONTF::CloneFrom(
    CEnumItemsWithSHCONTF *pSource)
{
  HRESULT hr = pSource->m_speidl->Clone(&m_speidl);
  if (SUCCEEDED(hr)) {
    m_spsfParent = pSource->m_spsfParent;
  }
  return hr;
}

HRESULT CEnumItemsWithSHCONTF::Create(
    HWND hwnd, SHCONTF shcontf,
    IShellItem *psiFolder, REFIID riid, void **ppv)
{
  *ppv = nullptr;

  CComPtr<CComObject<CEnumItemsWithSHCONTF>> spObj;
  HRESULT hr = CreateRef1(&spObj);

  if (SUCCEEDED(hr)) {
    hr = spObj->Initialize(hwnd, shcontf, psiFolder);
    if (SUCCEEDED(hr)) {
      hr = spObj->QueryInterface(riid, ppv);
    }
  }
  return hr;
}

HRESULT CEnumItemsWithSHCONTF::Next(
    ULONG celt, IShellItem **ppsi, ULONG *pceltFetched)
{
  if (celt != 1 && pceltFetched == nullptr) {
    return E_INVALIDARG;
  }

  for (ULONG i = 0; i < celt; i++) ppsi[i] = nullptr;

  ULONG celtFetched = 0;
  HRESULT hr = S_OK;
  while (hr == S_OK && celtFetched < celt) {
    CComHeapPtr<ITEMID_CHILD> spidlChild;
    hr = m_speidl->Next(1, &spidlChild, nullptr);
    if (hr == S_OK) {
      hr = SHCreateItemWithParent(nullptr, m_spsfParent,
        spidlChild, IID_PPV_ARGS(&ppsi[celtFetched]));
      if (SUCCEEDED(hr)) celtFetched++;
    }
  }

  if (pceltFetched != nullptr) *pceltFetched = celtFetched;
  if (SUCCEEDED(hr)) {
    hr = (celtFetched == celt) ? S_OK : S_FALSE;
  }
  return hr;
}

HRESULT CEnumItemsWithSHCONTF::Skip(ULONG celt)
{
  return m_speidl->Skip(celt);
}

HRESULT CEnumItemsWithSHCONTF::Reset()
{
  return m_speidl->Reset();
}

HRESULT CEnumItemsWithSHCONTF::Clone(
    IEnumShellItems **ppesiClone)
{
  *ppesiClone = nullptr;

  CComPtr<CComObject<CEnumItemsWithSHCONTF>> spClone;
  HRESULT hr = CreateRef1(&spClone);

  if (SUCCEEDED(hr)) {
    hr = spClone->CloneFrom(this);
    if (SUCCEEDED(hr)) {
        *ppesiClone = spClone.Detach();
    }
  }
  return hr;
}
```

The `CEnumItemsWithSHCONTF` class does the work of enumerating child items the old-fashioned way, then constructing shell items from the result. Most of this code is boilerplate (including the part to avoid having a live COM object with reference count zero).

The object has two members, the source folder from which the items are being enumerated and the low-level enumerator itself. We initialize the object by asking for the low-level `IEnumIDList` handler and calling `IEnumIDList::EnumObjects` with the specific flags we want. When it is time to generate items, we ask the inner enumerator for the next ID list, and construct a shell item around it by comining the ID list with the parent folder.

The rest is bookkeeping: We keep track of the number of elements fetched so far in order to return it to the caller if requested, and also in order to decide what the return value should be. If all items were retrieved successfully, then return `S_OK`. If we ran out of items, then return `S_FALSE`. If something went wrong, we return the error code, possibly with partially-fetched results.

The other enumerator operations like `Reset` and `Clone` are delegated to the inner enumerator. Cloning is a little tricky because we also have to clone ourselves!

Now we can adapt our program from last time to use this class instead of `BHID_EnumItems`.

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

 if (argc < 2) return 0;
 CComPtr<IShellItem> spsiFolder;
 SHCreateItemFromParsingName(argv[1], nullptr,
                             IID_PPV_ARGS(&spsiFolder));

 CComPtr<IEnumShellItems> spesi;
 CEnumItemsWithSHCONTF::Create(nullptr, SHCONTF_FOLDERS,
    spsiFolder, IID_PPV_ARGS(&spesi));
 for (CComPtr<IShellItem> spsi;
      spesi->Next(1, &spsi, nullptr) == S_OK;
      spsi.Release()) {
  PrintDisplayName(spsi, SIGDN_NORMALDISPLAY, L"Display Name");
  PrintDisplayName(spsi, SIGDN_FILESYSPATH, L"File system path");
  wprintf(L"\n");
 }
 return 0;
}
```

Raymond Chen

Follow