Obtaining the parsing name (and pidl) for a random shell object

devblogs.microsoft.com/oldnewthing/20130204-00

February 4, 2013

Raymond Chen

The parsing name for a shell item is handy, because it lets you regenerate the item later. Actually, the pidl for the shell item is even better, because that is the official way of saving and restoring objects. It's the pidl that gets saved in a shortcut, and since shortcuts can be copied around from machine to machine, pidls must be transportable and forward compatible. (A shortcut file created on Windows XP needs to keep working on all future versions of Windows.)

Here's a handy little tool for grabbing the parsing name and pidl for a random shell object. Start with our scratch program, and add in the `SimpleDropTarget` class, with the following tweaks:

```
public:
 SimpleDropTarget() : m_cRef(1) { ~~/* g_ppr->AddRef(); */~~ }
 ~SimpleDropTarget() { ~~g_ppr->Release();~~ }


…
 // *** IDropTarget ***
 STDMETHODIMP DragEnter(IDataObject *pdto,
    DWORD grfKeyState, POINTL ptl, DWORD *pdwEffect)
 {
  *pdwEffect &= DROPEFFECT_LINK;
  return S_OK;
 }


 STDMETHODIMP DragOver(DWORD grfKeyState,
   POINTL ptl, DWORD *pdwEffect)
 {
  *pdwEffect &= DROPEFFECT_LINK;
  return S_OK;
 }
…
};
```

We are not a COM local server, so we won't worry about managing our process reference. And we will accept anything that has a pidl, so we say that we will accept objects via linking. (The original code accepted by copying, which would have made us reject non-copyable objects.)

Now we can hook these up to our scratch program.

```
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
  g_hwndChild = CreateWindow(
      TEXT("edit"), nullptr, ES_MULTILINE |
      WS_CHILD | WS_VISIBLE | WS_TABSTOP,
      0, 0, 0,0, hwnd, (HMENU)1, g_hinst, 0);
  SimpleDropTarget *psdt = new(std::nothrow) SimpleDropTarget();
  if (psdt) {
    RegisterDragDrop(hwnd, psdt);
    psdt->Release();
  }
  return TRUE;
}


void
OnDestroy(HWND hwnd)
{
  RevokeDragDrop(hwnd);
  PostQuitMessage(0);
}


…
    // Change CoInitialize and CoUninitialize to Ole
    if (SUCCEEDED(OleInitialize(NULL))) {
…
        OleUninitialize();
```

Finally, we need to say what to do when the drop occurs.

```
void AppendText(LPCWSTR psz)
{
  SendMessageW(g_hwndChild, EM_REPLACESEL, 0, (LPARAM)psz);
}


void OpenFilesFromDataObject(IDataObject *pdto)
{
  CComPtr<IShellItemArray> spsia;
  if (SUCCEEDED(SHCreateShellItemArrayFromDataObject(
                               pdto, IID_PPV_ARGS(&spsia)))) {
    CComPtr<IEnumShellItems> spenum;
    spsia->EnumItems(&spenum);
    if (spenum) {
      for (CComPtr<IShellItem> spsi;
           spenum->Next(1, &spsi, nullptr) == S_OK;
           spsi.Release()) {
        CComHeapPtr<wchar_t> spszName;
        if (SUCCEEDED(spsi->GetDisplayName(
                     SIGDN_DESKTOPABSOLUTEPARSING, &spszName))) {
          AppendText(spszName);
          AppendText(L”\r\n”);
        }
        CComHeapPtr<ITEMIDLIST_ABSOLUTE> spidl;
        if (SUCCEEDED(CComQIPtr<IPersistIDList>(spsi)->
                                         GetIDList(&spidl))) {
          UINT cb = ILGetSize(spidl);
          BYTE *pb = reinterpret_cast<BYTE *>
                          (static_cast<PIDLIST_ABSOLUTE>(spidl));
          for (UINT i = 0; i < cb; i++) {
            WCHAR szHex[4];
            StringCchPrintf(szHex, ARRAYSIZE(szHex),
                            L”%02X “, pb[i]);
            AppendText(szHex);
          }
          AppendText(L”\r\n”);
        }
      }
    }
  }
}
```

When the drop occurs, we convert the data object into a shell item array, enumerate the items, and print the parsing name for the item as well as a hex dump of the pidl associated with the item.

I guess we need some header files.

```
#include <shlobj.h>
#include <strsafe.h>
#include <atlbase.h>
#include <atlalloc.h>
```

Run this program and drop the Recycle Bin onto it, say.

```
::{645FF040-5081-101B-9F08-00AA002F954E}
14 00 1F 78 40 F0 5F 64 81 50 1B 10 9F 08 00 AA 00 2F 95 4E 00 00
```

This tells you two things. First, that if you want to generate the Recycle Bin from a parsing name, you can use that string that starts with two colons.

```
var shell = new ActiveXObject("Shell.Application");
var recycleBin = shell.Namespace(
      "::{645FF040-5081-101B-9F08-00AA002F954E}");
var items = recycleBin.Items();
for (var i = 0; i < items.Count; i++) {
 WScript.StdOut.WriteLine(items.Item(i));
}
```

Of course, there is a predefined enumeration for the Recycle Bin, so this was a bit of a waste. You could've just written

```
var recycleBin = shell.Namespace(10);
```

But this technique generalizes to other locations in the shell namespace that do not have a special shorthand value.

The second thing the program tells you is that if you want to generate the Recycle Bin from a pidl, you can just use that chunk of bytes. Okay, that's not quite so interesting from a scripting point of view, but if you're manipulating pidls, this can be quite handy.

We'll use this program a little bit in a few weeks, but at this point, it's just a "Little Program" for today.

Raymond Chen

Follow