Displaying a property sheet for multiple files

devblogs.microsoft.com/oldnewthing/20130617-00

June 17, 2013

Raymond Chen

Today's Little Program will show a property sheet that covers multiple files, just like the one you get from Explorer if you multi-select a bunch of files and right-click them all then select *Properties*.

In fact, that description of how you do the operation interactively maps directly to how you do the operation programmatically!

```
#define UNICODE
#define _UNICODE
#define STRICT_TYPED_ITEMIDS
#include <windows.h>
#include <ole2.h>
#include <shlobj.h>
#include <atlbase.h>
#include <atlalloc.h>


HRESULT GetUIObjectOf(
    IShellFolder *psf,
    HWND hwndOwner,
    UINT cidl,
    PCUITEMID_CHILD_ARRAY apidl, REFIID riid, void **ppv)
{
 return psf->GetUIObjectOf(hwndOwner, cidl, apidl, riid, nullptr, ppv);
}
```

The `GetUIObjectOf` helper function merely wraps the `IShellFolder::GetUIObjectOf` method to insert the pesky `nullptr` parameter between the `riid` and `ppv`. The `riid` and `ppv` parameters by convention go right next to each other, and the `IID_PPV_ARGS` macro assumes that the function you're calling follows that convention. Unfortunately, the people who designed `IShellFolder::GetUIObjectOf` didn't get the memo, and we've been stuck with it ever since.

```
HRESULT InvokeCommandByVerb(
    IContextMenu *pcm,
    HWND hwnd,
    LPCSTR pszVerb)
{
 HMENU hmenu = CreatePopupMenu();
 HRESULT hr = hmenu ? S_OK : E_OUTOFMEMORY;
 if (SUCCEEDED(hr)) {
  hr = pcm->QueryContextMenu(hmenu, 0, 1, 0x7FFF, CMF_NORMAL);
  if (SUCCEEDED(hr)) {
   CMINVOKECOMMANDINFO info = { 0 };
   info.cbSize = sizeof(info);
   info.hwnd = hwnd;
   info.lpVerb = pszVerb;
   hr = pcm->InvokeCommand(&info);
  }
  DestroyMenu(hmenu);
 }
 return hr;
}
```

The `InvokeCommandByVerb` function merely hosts an `IContextMenu` and invokes a single verb.

Okay, those are the only two helper functions we need this week. The rest we can steal from earlier articles.

For the purpose of illustration, the program will display a multi-file property sheet for the first two files in your My Documents folder folder. Remember, Little Programs do little to no error checking.

```
int __cdecl wmain(int, wchar_t **)
{
 CCoInitialize init;
 ProcessReference ref;
 CComPtr<IShellFolder> spsf;
 BindToCsidl(CSIDL_MYDOCUMENTS, IID_PPV_ARGS(&spsf));
 CComPtr<IEnumIDList> speidl;
 spsf->EnumObjects(nullptr, SHCONTF_NONFOLDERS, &speidl);
 if (!speidl) return 0;
 CComHeapPtr<ITEMID_CHILD> spidl1;
 CComHeapPtr<ITEMID_CHILD> spidl2;
 if (speidl->Next(1, &spidl1, nullptr) != S_OK) return 0;
 if (speidl->Next(1, &spidl2, nullptr) != S_OK) return 0;
 PCUITEMID_CHILD rgpidl[2] = { spidl1, spidl2 };
 CComPtr<IContextMenu> spcm;
 GetUIObjectOf(spsf, nullptr, 2, rgpidl, IID_PPV_ARGS(&spcm));
 if (!spcm) return 0;
 InvokeCommandByVerb(spcm, “properties”);
 return 0;
}
```

Because everybody freaks out if I write code that doesn't run on Windows XP, I used the `BindToCSIDL` function instead of one of its more modern equivalents to get access to the My Documents folder.

Once we have My Documents, we ask to enumerate its non-folders. If the enumeration fails or says that there are no items (by returning `S_FALSE`), then we bail immediately.

Next, we enumerate two items from the folder. If we can't get both, then we bail.

We then create a two-item array and get the `IContextMenu` UI object for the collection.

Finally, we invoke the `"properties"` verb on the context menu.

And that's it. If you run this program, you'll see a context menu for the first two files in your My Documents folder.

Raymond Chen

Follow