Peeking inside an IShellItem to see what it's made of

devblogs.microsoft.com/oldnewthing/20160620-00

June 20, 2016

Raymond Chen

Windows XP introduced the `IShellItem` interface which represents an item in the shell namespace. This encapsulates what traditionally is represented by a pair of things, the the `IShellFolder` interface and an `ITEMID_CHILD`. The shell item lets you carry just one object around instead of two.

Another way of representing an item in the shell namespace is in the form of a single `ITEMID_ABSOLUTE`, and you can also create a shell item from that.

Creating a single unit of currency to represent a shell item tries to solve the problem of having to exchange money every time you cross a boundary. The `IShellItem` also gives you some methods which simplifies various operations by wrapping low-level methods on `IShellFolder`. For example, the `IShellItem::BindToHandler` method figures out the right way to get the item you ask for rather than making you puzzle out the arcane rules behind `IShellFolder::BindToObject`, `IShellFolder::BindToStorage`, `IShell-Folder::CreateViewObject`, `IShellFolder::GetUIObjectOf`, and more.

But what if you need something that `IShellItem` doesn't provide a convenience wrapper for? Then you need to peek inside.

If you want to peek inside and get the `IShellFolder` and `ITEMID_CHILD`, you can use the `IParentAndItem` interface, specifically, the `IParentAndItem::GetParentAndItem` method. Once nice thing about the `IParentAndItem::GetParentAndItem` method is that you can pass `nullptr` for the things you aren't interested in.

Alternatively, if you want to peek inside and get the `ITEMIDLIST_ABSOLUTE`, then you can use the `IPersistIDList::GetIDList` method to suck it out. We saw this a while back, but I'll repeat it here just so the information is all in one place.

If you are willing to abandon Windows XP support, you can use the `SHGetIDListFromObject` function which knows how to do this. (It tries other things, too.)

Okay, let's take things out for a spin. We'll get the normal display name for a shell item in four ways:

- By asking the item directly.
- By using the `IShellFolder::GetDisplayName` method.
- By using the `IPersistIDList::GetIDList` method, and then the `SHGetNameFrom-IDList` function.
- By using the `SHGetIDListFromObject` function, and then the `SHGetNameFromID-List` function.

If all goes well, we should get the same string printed each time.

Remember that Little Programs do little to no error checking.

```
#include <windows.h>
#include <shlobj.h>
#include <atlbase.h>
#include <atlalloc.h>
#include <stdio.h>     // horrors! mixing C and C++!

void PrintNameDirectlyFromItem(IShellItem* item)
{
    CComHeapPtr<wchar_t> name;
    item->GetDisplayName(SIGDN_NORMALDISPLAY, &name);
    _putws(name);
}

void PrintNameViaIShellFolder(IShellItem* item)
{
    CComPtr<IShellFolder> folder;
    CComHeapPtr<ITEMID_CHILD> child;
    CComQIPtr<IParentAndItem>(item)->GetParentAndItem(nullptr, &folder, &child);
    STRRET ret;
    folder->GetDisplayNameOf(child, SHGDN_NORMAL, &ret);
    CComHeapPtr<wchar_t> name;
    StrRetToStrW(&ret, child, &name);
    _putws(name);
}

void PrintNameViaAbsoluteIDList(IShellItem* item)
{
    CComHeapPtr<ITEMIDLIST_ABSOLUTE> absolute;
    CComQIPtr<IPersistIDList>(item)->GetIDList(&absolute);
    CComHeapPtr<wchar_t> name;
    SHGetNameFromIDList(absolute, SIGDN_NORMALDISPLAY, &name);
    _putws(name);
}

void PrintNameViaAbsoluteIDList2(IShellItem* item)
{
    CComHeapPtr<ITEMIDLIST_ABSOLUTE> absolute;
    SHGetIDListFromObject(item, &absolute);
    CComHeapPtr<wchar_t> name;
    SHGetNameFromIDList(absolute, SIGDN_NORMALDISPLAY, &name);
    _putws(name);
}

int main(int, char**)
{
    CCoInitialize init;

    CComPtr<IShellItem> item;
    SHGetKnownFolderItem(FOLDERID_Downloads, KF_FLAG_DEFAULT, nullptr,
IID_PPV_ARGS(&item));

    PrintNameDirectlyFromItem(item);
```

```
    PrintNameViaIShellFolder(item);
    PrintNameViaAbsoluteIDList(item);
    PrintNameViaAbsoluteIDList2(item);

    return 0;
}
```

Bonus chatter: When you create a shell item, it takes the things you created it from, and it produces the other equivalent things on demand. For example, if you create a shell item from an absolute item ID list, and then you ask for the folder and child item ID, it will convert the absolute item ID list into a folder and child item ID list. (It also caches the result so that the next time you ask, it'll be able to answer the question more quickly.)

Raymond Chen

Follow