How do I get the user-customed name of My Computer or Recycle Bin?

devblogs.microsoft.com/oldnewthing/20151012-00

October 12, 2015

Raymond Chen

Today's Little Program displays the user-customized name of This PC (the folder formerly known as My Computer) or Recycle Bin. The basic principle here is that if you want to display something the same way that Explorer does, then ask the same data source that Explorer asks: The shell namespace.

Today's smart pointer library is (rolls dice) *nothing*. We're going to use raw pointers.

Remember, Little Programs do little to no error checking.

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

void PrintDisplayName(IShellItem* item, SIGDN sigdn, PCWSTR label)
{
  PWSTR name;
  item->GetDisplayName(sigdn, &name);
  printf("%ls = %ls\n", label, name);
  CoTaskMemFree(name);
}

void PrintKnownFolderDisplayName(
    REFKNOWNFOLDERID rfid, SIGDN sigdn, PCWSTR label)
{
  IShellItem* item;
  SHGetKnownFolderItem(rfid, KF_FLAG_DONT_VERIFY,
                       nullptr, IID_PPV_ARGS(&item));
  PrintDisplayName(item, sigdn, label);
  item->Release();
}

int __cdecl wmain(int argc, wchar_t **argv)
{
  CoInitialize(0);
  PrintKnownFolderDisplayName(FOLDERID_ComputerFolder,
    SIGDN_NORMALDISPLAY, L"name");
  PrintKnownFolderDisplayName(FOLDERID_RecycleBinFolder,
    SIGDN_NORMALDISPLAY, L"name");
  CoUninitialize();
  return 0;
}
```

The `PrintDisplayName` function obtains the display name of a shell item and prints it. The `PrintKnownFolderDisplayName` function gets the item for a known folder and prints its display name. And our main program grabs the This PC and Recycle Bin folders and prints their display names.

Really not that complicated.

If you are old-school and prefer to work with shell folders and item ID lists, you can do it the old school way:

```
#define STRICT_TYPED_ITEMIDS
#define UNICODE
#define _UNICODE
#include <windows.h>
#include <ole2.h>
#include <oleauto.h>
#include <shlobj.h>
#include <shlwapi.h>
#include <stdio.h>
#include <tchar.h>

void PrintDisplayName(PCIDLIST_ABSOLUTE absolute, SHGDNF shgdn, PCWSTR label)
{
  IShellFolder* parentFolder;
  PCUITEMID_CHILD child;
  SHBindToParent(absolute, IID_PPV_ARGS(&parentFolder), &child);
  PrintDisplayName(parentFolder, child, shgdn, label);
  parentFolder->Release();
}

void PrintCsidlDisplayName(int csidl, SHGDNF shgdn, PCWSTR label)
{
  PIDLIST_ABSOLUTE absolute;
  SHGetFolderLocation(nullptr, csidl, nullptr, 0, &absolute);
  PrintDisplayName(absolute, shgdn, label);
  CoTaskMemFree(absolute);
}

int __cdecl wmain(int argc, wchar_t **argv)
{
  CoInitialize(0);
  PrintCsidlDisplayName(CSIDL_DRIVES, SHGDN_NORMAL, L"name");
  PrintCsidlDisplayName(CSIDL_BITBUCKET, SHGDN_NORMAL, L"name");
  CoUninitialize();
  return 0;
}
```

The idea is the same: We bind to the special folder and print its name. Getting the name of an object is done by asking the parent for the name of the child.

Okay, this isn't all that exciting, but it still shows that the way to get the name of something the way Explorer shows it is to get the name the way Explorer gets it.

I'll play with this a little more next week.

Raymond Chen

Follow