How do I show the contents of a directory while respecting the user's preferences for hidden and super-hidden files as well as the user's language preferences?

devblogs.microsoft.com/oldnewthing/20140317-00

March 17, 2014

Raymond Chen

A customer was writing a program in (and this is what they said) “32 bit C++ .Net 4.0” which displayed the contents of a directory, and they wanted to filter out items such as hidden files and protected operating system files (also known as *super-hidden* files) based on the user’s current Explorer preferences. Furthermore, they wanted to show localized folder names, such as *Usarios* instead of *Users*, again, the same way Explorer does. They are currently using `Directory.GetDirectories()`.

The way to do this is to use `IShellFolder::EnumObject`, the same way Explorer does. Don’t pass `SHCONTF_INCLUDEHIDDEN` or `SHCONTF_INCLUDESUPERHIDDEN`, and you will get the default enumeration that filters out hidden items based on the user’s preferences. (You pass the flag to force the items to be included, overriding the user’s preferences.) and the names of the items that come out of the enumeration will be the localized names. You can ask for the parsing name to get the physical file name.

```
#define UNICODE
#define _UNICODE
#define STRICT
#define STRICT_TYPED_ITEMIDS
#include <windows.h>
#include <shlobj.h>
#include <atlbase.h>
#include <atlalloc.h>
int __cdecl wmain(int argc, wchar_t **argv)
{
 CCoInitialize init;
 if (argc < 2) return 0;
 CComHeapPtr<ITEMIDLIST_ABSOLUTE> sppidl;
 CComPtr<IShellFolder> spsf;
 CComPtr<IEnumIDList> speidl;
 if (FAILED(SHParseDisplayName(argv[1], nullptr,
                               &sppidl, 0, nullptr)) ||
     FAILED(SHBindToObject(nullptr, sppidl,
                           nullptr, IID_PPV_ARGS(&spsf))) ||
     FAILED(spsf->EnumObjects(nullptr,
               SHCONTF_FOLDERS | SHCONTF_NONFOLDERS, &speidl)) ||
     speidl == nullptr) return 0;
 for (CComHeapPtr<ITEMID_CHILD> sppidlItem;
      speidl->Next(1, &sppidlItem, nullptr) == S_OK;
      sppidlItem.Free()) {
  PrintDisplayName(spsf, sppidlItem, SHGDN_NORMAL, L"Display Name");
  PrintDisplayName(spsf, sppidlItem, SHGDN_FORPARSING, L"For Parsing");
  wprintf(L"\n");
 }
}
```

The program takes a fully-qualified path on the command line and displays its contents (both in localized display name and in raw file system paths) while respecting the user's preferences for hidden and super-hidden files.

It appears that the customer is writing their program in C#, despite their claim that they were using C++ (or maybe they meant MC++ or C++/CLI). In that case, they can use the *Windows 7 API CodePack for Microsoft® .NET Framework* (gotta love that catchy name).

Raymond Chen

Follow