Why is there a CSIDL_DESKTOP value if you need the desktop in order to get it anyway?

devblogs.microsoft.com/oldnewthing/20111017-00

October 17, 2011

Raymond Chen

John asks why there is a special constant `CSIDL_DESKTOP` defined for the desktop. After all, in order to use `CSIDL_DESKTOP`, you need to call `SHGetDesktopFolder` and then bind to it. What's the point of having an `ITEMIDLIST` that represents the desktop if, in order to use it, you first need to get the desktop?

It's like asking why the file system uses `.` (dot) to refer to the current directory. You're already *in* the current directory. In order to resolve `.` (dot), you already need to have the current directory, so why bother with the dot at all?

Because it is often convenient to give a name to your starting point.

Suppose somebody wants to save a file to the desktop. How would you represent this as an `ITEMIDLIST`? If the only thing you can do is fill in the blank in the sentence, "Start with the desktop folder, then go to _____, then save the file there," then you need a way to say "where you are now."

And that's what `CSIDL_DESKTOP` gives you. An `ITEMIDLIST` that says "Where you are now."

Besides, if `CSIDL_DESKTOP` weren't defined, somebody would have invented it. Say your program has a list of directories it wants to operate on, say, the Documents folder, the Music folder, and the Shared Documents folder. Great, so let me write a function:

```
void DoItIn(HWND hwnd, int csidl)
{
 PIDLIST_ABSOLUTE pidl;
 if (SUCCEEDED(SHGetSpecialFolderLocation(
                     hwnd, csidl, &pidl))) {
  IShellFolder *psf;
  if (SUCCEEDED(SHBindToObject(NULL, pidl, NULL,
                   IID_PPV_ARGS(&psf)))) {
   ...
   psf->Release();
  }
  CoTaskMemFree(pidl);
 }
}
void DoItInStandardPlaces(HWND hwnd)
{
 const static int csidls[] = {
  CSIDL_MYDOCUMENTS,
  CSIDL_MYMUSIC,
  CSIDL_COMMON_DOCUMENTS,
 };
 for (int i = 0; i < ARRAYSIZE(csidls); i++) {
  DoItIn(hwnd, csidls[i]);
 }
}
```

Now you want to add the desktop folder. Oh wait, there is no `CSIDL` value for the desktop, so I'll have to make one up.

```
// Our custom CSIDLs use the high word. None of the CSIDLs we use
// set any bits in the high word, so we can use the high word to
// detect whether we have a standard CSIDL or a custom CSIDL.
#define CUSTOMCSIDL_DESKTOP 0x00010000
#define ISCUSTOMCSIDL(csidl) HIWORD(csidl)
#define STANDARDCSIDLOF(csidl) LOWORD(csidl)
HRESULT MyGetSpecialFolderLocation(
    HWND hwnd, int csidl, PIDLIST_ABSOLUTE *ppidl)
{
 HRESULT hr;
 if (ISCUSTOMCSIDL(csidl)) {
  *ppidl = (PIDLIST_ABSOLUTE)CoTaskMemAlloc(sizeof(WORD));
  if (*ppidl) {
   ppidl->mkid.cb = 0;
   hr = S_OK;
  } else {
   hr = E_OUTOFMEMORY;
  }
 } else {
  hr = SHGetSpecialFolderLocation(hwnd, STANDARDCSIDLOF(csidl), ppidl);
 }
 return hr;
}
```

Okay, cool, now I can add

```
const static int csidls[] = {
 CSIDL_MYDOCUMENTS,
 CSIDL_MYMUSIC,
 CSIDL_COMMON_DOCUMENTS,
 CUSTOMCSIDL_DESKTOP,
};
```

Oh wait, I also have to have a custom version of `SHBindToObject` that knows how to bind to this special new type of pidl that means "where you are now."

```
HRESULT MyBindToObject(IShellFolder *psf, PCUIDLIST_RELATIVE pidl,
  IBindCtx *pbc, REFIID riid, void **ppv)
{
 HRESULT hr;
 if (pidl->mkid.cb == 0) {
  *ppv = NULL;
  if (psf == NULL) {
   hr = SHGetDesktopFolder(&psf);
   if (SUCCEEDED(hr)) {
    hr = psf->QueryInterface(riid, ppv);
    psf->Release();
   }
  } else {
   hr = psf->QueryInterface(riid, ppv);
  }
 } else {
  hr = SHBindToObject(psf, pidl, pbc, riid, ppv);
 }
 return hr;
}
```

Congratulations, you just reinvented `CSIDL_DESKTOP`.

It can be very convenient to have a name for the null action.

Raymond Chen

Follow