How do I show the sharing pane from a Win32 desktop application?

devblogs.microsoft.com/oldnewthing/20170315-00

March 15, 2017

Raymond Chen

A customer wanted to show the sharing pane from their Win32 desktop application. In a UWP application, this would be done by calling `Windows.ApplicationModel.DataTransfer.DataTransferManager.ShowSharingUI()`. Let's do it in a Win32 desktop app by following the basic rules for projection: Static methods are represented as methods on a "Statics" interface on the activation factory.

Start with the scratch program and make these changes. (Remember, Little Programs do little to no error checking.)

```
#include <wrl/client.h>
#include <wrl/wrappers/corewrappers.h>
#include <windows.applicationmodel.datatransfer.h>
#include <tchar.h> // Huh? Why are you still using ANSI?
#include <roapi.h>

namespace WRL = Microsoft::WRL;
namespace dt = ABI::Windows::ApplicationModel::DataTransfer;

using Microsoft::WRL::Wrappers::HStringReference;

void OnChar(HWND hwnd, TCHAR ch, int cRepeat)
{
  switch (ch) {
  case TEXT(' '):
    {
      WRL::ComPtr<dt::IDataTransferManagerStatics> dtmStatics;
      RoGetActivationFactory(HStringReference(
        RuntimeClass_Windows_ApplicationModel_DataTransfer_DataTransferManager)
                             .Get(), IID_PPV_ARGS(&dtmStatics));
      dtmStatics->ShowShareUI();
    }
    break;
  }
}

HANDLE_MSG(hwnd, WM_CHAR, OnChar);
```

Fire up this program, hit the space bar, and… nothing happens.

Okay, so maybe we need to do a tiny bit of error checking after all. The call to `ShowShareUI` fails with `E_NOT_SET`. The reason is that the `ShowShareUI` method has an implicit dependency on the current thread's `CoreWindow`, because it needs to know which window is being shared. But since we are a Win32 desktop program, we don't have a `CoreWindow`.

Oh no, what do we do?

Enter the interop pattern.

To accommodate Win32 desktop programs, there is a parallel universe of `HWND`-based methods. In places where WinRT depends on the current thread's `CoreWindow`, this alternative universe offers a similarly-named method, but with the `ForWindow` suffix, indicating that it operates on classic Win32 `HWND`s rather than fancy-pants `CoreWindow`s.

One component of this parallel universe of `-ForWindow` methods consists of interfaces that end in the name `Interop`. In our case, it's `IDataTransferManagerInterop`. This interface is available on the activation factory, the same as the `IDataTransferManager-Statics` interface. The general pattern is as follows:

XxxStatics	XxxInterop
GetForCurrentView	GetForWindow
DoSomething (implied "for current view")	DoSomethingForWindow

In our case, we have a `ShowSharingUI()` method on the `Statics` interface, so the corresponding interop method is called `ShowSharingForWIndow()`.

```
#include <shlobj.h> // IDataTransferManagerInterop

void OnChar(HWND hwnd, TCHAR ch, int cRepeat)
{
  switch (ch) {
  case TEXT(' '):
    {
      WRL::ComPtr<dt::IDataTransferManagerInterop> dtmInterop;
      RoGetActivationFactory(HStringReference(
        RuntimeClass_Windows_ApplicationModel_DataTransfer_DataTransferManager)
                             .Get(), IID_PPV_ARGS(&dtmInterop));
      dtmInterop->ShowShareUIForWindow(hwnd);
    }
    break;
  }
}
```

Okay, so now we show the share pane, but the pane just offers to share a screen shot. How can we get the pane to offer custom data provided by the program? We'll look at that next time.

Bonus chatter: One of my colleagues noted that "data transfer manager" is a poor name for the class, seeing as transferring data is what computers do most of the time anyway.

Raymond Chen

Follow