SHOpenRegStream does not mix with smart pointers

devblogs.microsoft.com/oldnewthing/20181227-00

December 27, 2018

Raymond Chen

Some time ago, I noted that `CoGetInterfaceAndReleaseStream` does not mix with smart pointers because it performs an `IUnknown:: Release` of its interface parameter, which messes up all the bookkeeping because smart pointers expect that *they* are the ones which will perform the `Release`.

The other half of the problem is functions like `SHOpenRegStream` and `SHOpenRegStream2` which return a COM pointer directly, rather than putting it into an output parameter. When you put them into a smart pointer, the default behavior of the smart pointer is to create a new reference, so it will call `AddRef` upon assignment or construction, and call `Release` upon replacement or destruction.

```
// Code in italics is wrong

Microsoft::WRL::ComPtr<IStream> stream;
stream = SHOpenRegStream(...);

Microsoft::WRL::ComPtr<IStream> stream(SHOpenRegStream(...));

ATL::CComPtr<IStream> stream;
stream = SHOpenRegStream(...);

ATL::CComPtr<IStream> stream(SHOpenRegStream(...));

_com_ptr_t<IStream> stream;
stream = SHOpenRegStream(...);

_com_ptr_t<IStream> stream(SHOpenRegStream(...));
```

All of these operations will take the raw pointer returned by `SHOpenRegStream`, save it in the smart pointer, and increment the reference count. When the smart pointer is destructed, the reference count will be decremented.

But the object started with a reference count of 1. After storing it in the smart pointer, the reference count is 2, even though there is only one object tracking it. When that object (in this case, the smart pointer) releases its reference, there is still one reference remaining, which nobody is tracking.

You have a memory leak.

The solution is to use the `Attach` method to say, “Here is an object that I would like you to adopt responsibility for.” The smart pointer will take the object but will *not* increment the reference count, because you told it, “I want you to take responsibility for the reference count that I am giving you.”

```
Microsoft::WRL::ComPtr<IStream> stream;
stream.Attach(SHOpenRegStream(...));

ATL::CComPtr<IStream> stream;
stream.Attach(SHOpenRegStream(...));

_com_ptr_t<IStream> stream;
stream.Attach(SHOpenRegStream(...));

_com_ptr_t<IStream> stream(SHOpenRegStream(...), false);
```

The `_com_ptr_t` class has a bonus constructor that takes a boolean parameter that indicates whether the smart pointer should perform an `AddRef` on the pointer. In the case where you want to adopt an existing reference, you pass `false`.

This problem is basically the flip side of the `CoGetInterfaceAndReleaseStream` problem. Whereas that one results in an over-release, this results in an under-release.

And the root cause of both of them is that they use a calling pattern that doesn’t conform to COM recommendations.

Raymond Chen

Follow