If you know what interface you want, just pass it directly to CoCreateInstance

devblogs.microsoft.com/oldnewthing/20240520-00

May 20, 2024

Raymond Chen

A pattern I sometimes see is calling `CoCreateInstance` for an interface, and immediately turning around and querying for another interface, and never using the original interface.[1]

```
wil::com_ptr<IWidget> widget;
RETURN_IF_FAILED(
    CoCreateInstance(CLSID_Widget, nullptr,
        CLSCTX_LOCAL_SERVER, IID_PPV_ARGS(&widget)));

wil::com_ptr<IServiceProvider> provider;
RETURN_IF_FAILED(
    widget->QueryInterface(IID_PPV_ARGS(&provider)));

// use the service provider, ignore the widget
```

There's rarely any need to request one interface, then immediately exchange it for another interface. You may as well go directly for the interface you want:

```
wil::com_ptr<IServiceProvider> provider;
RETURN_IF_FAILED(
    CoCreateInstance(CLSID_Widget, nullptr,
        CLSCTX_LOCAL_SERVER, IID_PPV_ARGS(&provider)));

// use the service provider
```

For non-local interfaces, collapsing the request into a single call avoids a round trip to the server.

The only case I can think of where the initial interface is significant is if you want to ensure that the object supports the intermediate interface, even though you aren't going to be using it right now.

```
// Factories must support IServiceProvider.
wil::com_ptr<IServiceProvider> provider;
RETURN_IF_FAILED(
    CoCreateInstance(factoryClassId, nullptr,
        CLSCTX_LOCAL_SERVER, IID_PPV_ARGS(&provider)));

wil::com_ptr<IWidget> widget;
RETURN_IF_FAILED(
    persist->QueryInterface(IID_PPV_ARGS(&widget)));

// use the widget, ignore the IServiceProvider for now
```

In that case, you can still avoid a round trip to the server by using `CoCreateInstanceEx` to request multiple interfaces at once.

```
MULTI_QI mqi[2] = {
    { &__uuidof(IWidget), nullptr, 0 },
    { &__uuidof(IServiceProvider), nullptr, 0 },
};

RETURN_IF_FAILED(CoCreateInstanceEx(
    factoryClassId, nullptr, CLSCTX_LOCAL_SERVER,
    nullptr, ARRAYSIZE(mqi), mqi));

wil::com_ptr<IWidget> widget;
widget.attach(mqi[0].pItf);

wil::com_ptr<IServiceProvider> provider;
provider.attach(mqi[1].pItf);

if (hr != S_OK) {
    // Failed to get at least one interface.
    return E_NOINTERFACE;
}
```

You can refer to my earlier series on `MULTI_QI` for further discussion.

[1] Note that I'm using WIL only for its smart pointer class and still programming to the ABI for the most part. WIL itself contains wrappers for many of the patterns here, but this article is really about the pattern and not the WIL wrappers.