Creating an already-completed asynchronous activity in C++/WinRT, part 9

devblogs.microsoft.com/oldnewthing/20240719-00

July 19, 2024

Raymond Chen

Last time, we created proxy objects that can produce an already-completed `IAsyncAction` by implementing interface directly. But we can do better… by cheating!

The observation here is that in practice, nobody ever asks for the `Completed` and `Progress` handlers, so maybe we can chat and not bother remembering them. In fact, the caller can't even reliably compare the returned value against the value it had originally registered, because the the returned value of the property might be an agile proxy wrapper. So maybe we can cheat and simply never save it. This allows us to remove the mutex from our class entirely, and it also lets us use CRTP to avoid having to know the types of the `Completed` and `Progress` handlers.

```
template<typename D, typename Async>
struct completed_async_base
{
    using Traits = winrt_async_traits<Async>;

    auto outer() { return static_cast<D*>(this); }

    auto Id() { return 1; }
    // Status and Error code implemented by D
    auto Cancel() { }
    auto Close() { }

    template<typename Handler>
    auto Completed(Handler const& handler)
    {
        handler(*outer(), outer()->Status());
    }
    auto Completed() { return nullptr; }

    template<typename...Args>
    auto Progress(Args...&&) { }
};
```

Okay, now that we've explored the studio space, we should step back and ask ourselves whether this version which implements the interfaces explicitly is better than the earlier version that used the coroutine infrastructure.

Maybe, but maybe not. Your program is probably already using the coroutine infrastructure for other things, so the incremental cost of adding another coroutine is relatively small. And it's certainly less typing. And objects which implement a Windows Runtime asynchronous interface typically have a very short lifetime: You await the Windows Runtime asynchronous interface, and then you throw it away. So any size savings is not really going to save you a lot, since you aren't creating thousands of these objects.

But still, it was a handy exercise, and maybe we learned some C++ and Windows Runtime coding techniques.