Cancelling a Windows Runtime asynchronous operation, part 7: WRL-generated asynchronous operations

devblogs.microsoft.com/oldnewthing/20200709-00

July 9, 2020

Raymond Chen

Last time, we saw that if a C++/WinRT asynchronous operation is cancelled, the result is an `hresult_canceled`. Today, we'll look at another implementation library for asynchronous operations: The Windows Runtime Template Library (WRL).

WRL is a bit out of fashion nowadays, but back in its heyday, it was the stuff everybody used for creating Windows Runtime objects and asynchronous operations.

```
class AsyncBase< ... > : ...
{
    inline HRESULT CheckValidStateForResultsCall()
    {
        Details::AsyncStatusInternal current = Details::_Undefined;
        CurrentStatus(&current);

        if (current == Details::_Error)
        {
            ...
        }

        if (resultType == SingleResult)
        {
            if (current != Details::_Completed)
            {
                ::RoOriginateError(E_ILLEGAL_METHOD_CALL, nullptr);
                return E_ILLEGAL_METHOD_CALL;
            }
        }
        ...
    }
};
```

The `CheckValidStateForResultsCall` method does what is says: It validates that the operation is in a correct state for a call to `GetResults()`. In the case where the operation has been canceled, we fail the first test (which propagates any explicit error code), but make it into the second part. This is a single-result operation[1] and a cancelled operation is not complete, so it returns `E_ILLEGAL_METHOD_CALL`, which propagates back into C++/WinRT as `hresult_illegal_method_call`.

And that solves the mystery of why some cancelled operations throw an `hresult_illegal_method_call` exception. It took us a week to get here, but we finally made it. The propagation of cancellation is a delicate dance between the the operation and the awaiter, and it's easy to stumble.

[1] Nearly all Windows Runtime asynchronous operations are single-result. There's another rarely-seen variant called the multiple-result asynchronous operation, which lets you call `GetResults` before the operation has completed, and it gives you the result-so-far. I am not aware of any operations that do this, but the code supports.

Raymond Chen

Follow