Inside C++/WinRT: Coroutine completions: The oversimplified version

devblogs.microsoft.com/oldnewthing/20230123-00

January 23, 2023

Raymond Chen

C++/WinRT coroutines use the `Completed` delegate property to be notified when an asynchronous operation is complete. There are multiple parts of the completion handler. Today we'll look at an oversimplified version, and then we will gradually build it up.

```
template <typename Async>
struct await_adapter
{
    await_adapter(Async const& async) : async(async) { }

    Async const& async;

    bool await_ready() const noexcept
    {
        return false;
    }

    void await_suspend(std::experimental::coroutine_handle<> handle) const
    {
        async.Completed([handle](auto&& ...)
        {
            handle.resume();
        });
    }

    auto await_resume() const
    {
        return async.GetResults();
    }
};
```

To `co_await` an `IAsyncAction` or `IAsyncOperation`, we register a completion callback that resumes the awaiting coroutine. When the coroutine resumes, it will call `await_resume()` to obtain the result of the `co_await`, and we propagate the result of the `GetResults()` method.

We don't particularly care about the parameters passed to the completion delegate: Those tell us whether the asynchronous work completed successfully, was cancelled, or failed outright, but we don't need to remember that information because `GetResults()` will report the information again: If the asynchronous work did not complete successfully, then `GetResults()` will thrown an exception describing why it was not successful.

As things go, this is a fairly standard implementation of a coroutine awaiter, although there is a race condition we need to fix:

```
void await_suspend(std::experimental::coroutine_handle<> handle) const
{
    auto extend_lifetime = async;
    async.Completed([handle](auto&& ...)
    {
        handle.resume();
    });
}
```

If the completion handler runs before `Completed()` returns, then we end up destroying the `async` while there is still an active call on it. This is a problem I called out at the start of my coroutine series. To fix this, we make a local copy of the `async` to extend its lifetime to the end of the `await_suspend` function.

This is just the starting point for C++/WinRT coroutine completion handlers. Next time, we'll add apartment-preserving behavior.