How do I produce a Windows Runtime asynchronous activity from C++/WinRT?

devblogs.microsoft.com/oldnewthing/20240705-00

July 5, 2024

Raymond Chen

We've looked at how to produce a Windows Runtime asynchronous activity in C++/CX and in C#, and they both involved passing a lambda to a helper function, where the lambda received special parameters for detecting cancellation and report progress. Fortunately, C++/WinRT integrates Windows Runtime asynchrony natively. You just write a function or method that returns an `IAsyncSomething`, and inside the body, you can `co_await` and `co_return`. If you want to check for cancellation, you can request a cancellation token, and if you want to report progress, you can request a progress token.

```
winrt::IAsyncOperation<winrt::Widget> GetWidgetAsync(winrt::hstring);
winrt::IAsyncAction EnableWidgetAsync(winrt::Widget, bool);

winrt::IAsyncActionWithProgress<int>
    EnableWidgetByIdAsync(winrt::hstring id, bool enable)
{
    // If this is an instance member function
    auto lifetime = get_strong();
    // If this is a global function or static member function
    auto lifetime = wil::winrt_module_reference();

    auto cancel = co_await winrt::get_cancellation_token();
    auto progress = co_await winrt::get_progress_token();

    progress(0);
    auto widget = co_await GetWidgetAsync(id);
    progress(1);
    if (cancel()) throw winrt::hresult_canceled();
    if (!widget) co_return false;
    co_await EnableWidgetAsync(widget, enable);
    co_return true;
}
```

C++/WinRT automatically checks for cancellation after every `co_await`,[1] so we didn't really need a cancellation token in this example, but I showed it for completeness.

[1] Except the special awaitables `co_await winrt::get_cancellation_token()` and `co_await winrt::get_progress_token()`.