Creating an already-completed asynchonous activity in C++/WinRT, part 1

devblogs.microsoft.com/oldnewthing/20240709-00

July 9, 2024

Raymond Chen

When working with asynchronous code, you may need to create an asynchronous activity that has already completed, because you already know the answer. For example, you may be implementing a method whose signature is

```
IAsyncOperation<int> ComputeAsync();
```

but you already have the result and don't need to compute it. How can you return an `IAsyncOperation<int>` that represents the already-computed result?

C# has `Task.FromResult()` and `Task.CompletedTask`. JavaScript has `Promise.resolve()`. The Parallel Patterns Library (PPL) has `task_from_result()`. What about C++/WinRT?

The simplest way is to just `co_return` the result into a coroutine.

```
winrt::Windows::Foundation::IAsyncOperation<int>
    ComputeAsync()
{
    co_return 42;
}
```

Similarly, C# has `Task.FromException()`, JavaScript has `Promise.reject()`, and PPL has `task_from_exception()`. The simple C++/WinRT version is to throw the exception from the coroutine.

But wait, this doesn't do what you think:

```
winrt::Windows::Foundation::IAsyncOperation<int>
    ComputeAsync()
{
    throw winrt::hresult_access_denied();
}
```

There is no `co_await` or `co_return` statement in the function body, so this is not a coroutine: Instead of returning a failed coroutine, this function fails to return a coroutine! When you call it, it throws an exception.

We'll look at ways of making this a coroutine next time.