Inside C++/WinRT: Apartment switching: Bypassing the context callback

devblogs.microsoft.com/oldnewthing/20230126-00

January 26, 2023

Raymond Chen

Last time, we avoided the problem of the synchronous apartment-changing callback.

Our implementation always uses `IContextCallback::ContextCallback()` to switch apartments. This is rather wasteful if we are already in the target apartment: It consumes a good amount of stack space as we call into the COM infrastructure, and then the COM infrastructure calls us back, where we then resume the coroutine, when we could have just resumed the coroutine directly without getting COM involved at all. It is not uncommon to resume a coroutine's execution in the context you are already in (say, because the awaited-on coroutine happens to finish in the same apartment in which it started). The extra stack consumption can become a problem if you have a long chain of coroutines, or are awaiting in a loop.

To short-circuit the `IContextCallback::ContextCallback()`, we can check whether we are already in the correct apartment, in which case we resume the coroutine immediately.

```
inline auto resume_apartment(
    com_ptr<IContextCallback> const& context,
    coroutine_handle<> handle)
{
    WINRT_ASSERT(context.valid());
    if (context ==
        capture<IContextCallback>(WINRT_IMPL_CoGetObjectContext))
    {
        handle();
    }
    else if (is_sta_thread())
    {
        resume_apartment_on_threadpool(context, handle);
    }
    else
    {
        resume_apartment_sync(context, handle);
    }
}
```

If the original context equals the current context, then we are already in the correct apartment, and we can just resume the coroutine immediately. Only if the contexts don't match do we need to go through the whole `IContextCallback::ContextCallback()` ritual.

Next time, we'll try to address another source of unwanted stack build-up.