Creating a non-agile delegate in C++/WinRT, part 3: The other cases and why they aren't interesting

devblogs.microsoft.com/oldnewthing/20200408-00

April 8, 2020

Raymond Chen

We've been looking at one specific case of a non-agile delegate, namely a delegate that is invoked on a background thread and wants to execute synchronously on a UI thread. What about the other cases?

Yeah, what about them? Let's write out a table.

	Raised on	
Handled on	**Background thread**	**UI thread**
Background thread	Already there	Makes no sense
UI thread	`resume_ synchronous`	Already there
Other UI thread		`resume_ synchronous`

Two of the boxes are labeled *Already there*. Those are the cases where the event is raised in the same apartment that you want to handle it, in which case everything is fine and there's no need to play any apartment-switching games.

Two of the boxes are labeled `resume_ synchronous`. In these cases, you want to run the handler synchronously on a UI thread which is not the one your handler was invoked in. In those cases, you can use the `resume_ synchronous` function we discussed last time. Of course, since you are running on a UI thread, you shouldn't perform long blocking operations. This is particularly true if you're in the bottom right corner, because you are holding *two* UI threads hostage while your event handler is running.

That leaves the last box, labeled *Makes no sense*.

That box makes no sense.

That box represents the case where the event is raised on a UI thread, but you want to handle it synchronously on a background thread. Why would you do that?

The usual reason for moving to a background thread is to allow you to perform long-running operations without affecting the responsiveness of the UI thread. But that doesn't help us here, because this table is about running the code *synchronously* in another apartment, so the UI thread is going to remain unresponsive while the background thread does its work. Switching to the background thread didn't accomplish anything. You may as well just do it on the UI thread.

Remember that this entire discussion is in the context of running the handler synchronously. If you can run the handler asynchronously, perhaps with the assistance of a deferral, then you should just do that.

We'll wrap up this discussion next time by connecting this discussion to C++/WinRT in a different way.

Raymond Chen

Follow