The dreaded "main" threading model

devblogs.microsoft.com/oldnewthing/20040602-00

June 2, 2004

Raymond Chen

In the absence of an explicit threading model for your COM object, you get the "main" threading model. The "main" threading model is little-known, and that's a good thing. It's a relic from the days before multi-threading.

The first thread in a process to initialize COM becomes declared the "main" thread. (It might be the first thread to initialize COM in apartment model; I forget.) When a "main" threaded object is created, COM marshals the creation call to the main thread, creates the object, then marshals the result back to the creator's thread. Similarly, when you invoke any method on the object, the call is marshalled to the main thread, invoked, then the result is marshalled back.

In other words, a "main" threaded object is like an apartment threaded object, with the additional constraint that the only apartment that can use it is the one that the "main" thread belongs to.

As you can imagine, this is a horrific performance penalty in any multithreaded application, since there is so much marshalling going on. Even worse, it completely bottlenecks the main thread because there are now all these objects that **must** be serviced on that thread and no other thread.

Even worse than worse, all this marshalling creates new opportunities for re-entrancy. While waiting for the main thread to do its thing, the calling thread will likely process messages, which means that you can receive a window message at a time when you didn't expect it.

So why does this awful threading model exist at all?

For backwards compatibility with COM objects written before multithreaded support was added to COM. Back in those days, there was only one thread, so COM objects could be extremely lazy with their synchronization. In fact, they didn't need any! If you have only one thread, then you certainly don't need to coordinate your actions with other threads because there are none.

That's also why "main" threading model is the default. Threading models were invented when multithreading support was added to COM. Before then, there were no threads, so no threading models. All old objects therefore didn't specify a threading model in their registration.

The only reason you should even be aware of this ancient threading model in the first place is that **if you forget to specify a threading model in your object registration, you will get the dreaded "main" threading model by default**.

And then you will wonder why your application's performance is horrible, and why you have all these strange re-entrancy problems.

Raymond Chen

Follow