A slightly less brief introduction to COM apartments (but it's still brief)

devblogs.microsoft.com/oldnewthing/20191125-00

November 25, 2019

Raymond Chen

The Component Object Model (COM) has this thing called *apartments*. What are they?

I'm going to explain them by going into the history.

In the beginning, we had 16-bit Windows. Each process had only one thread. Code took advantage of this by assuming single-threaded operation.

If you wanted to make a call from one process to another, the call was *marshaled*: The parameters to the call were serialized into a memory buffer, that memory buffer was sent to the receiving process, which deserialized them back into the parameters. This works out fine for parameters that are pass-by-value, like integers and strings.

For parameters that are pass-by-reference, deserializing the object isn't enough. If somebody makes a mutating method call, that mutating method call must go back to the original object, not to a copy. The way COM did this was to create a fake object on the recipient side (a *proxy*). If the recipient of the call invoked a method on the proxy, we recursively marshaled a call back to the original process.

The idea behind marshaling was that you can treat every object as if it were local. If the object happens not to be local, the proxy steps in and forwards the call to the *actual* object in some other process.

Okay, so far we have single-threaded processes and marshaling between processes.

Windows NT introduced multithreading. Now processes could have multiple threads. But there are all these components that assumed single-threaded operation. The solution was to treat each thread as if it were its own process. Communicating between threads was done the same way as communicating between processes: Calls between threads and cross-thread object references are marshaled.

Each "single-threaded pseudo-process" was called an *apartment*.

A little while later, COM introduced the concept of the *multi-threaded apartment* (MTA), and the old-timey "single-threaded pseudo-process" apartments were renamed *single-threaded apartments* (STA).[1] The multi-threaded apartment is one in which all the threads can share objects freely among each other without requiring marshaling.[2] Of course, in order for this to work, the objects themselves must support multi-threaded operation, which usually means lots of mutexes to protect internal state. (And lots of mutexes create the opportunity for lots of deadlocks.)

Every process that uses COM consists of zero or more single-threaded apartments, plus exactly one multi-threaded apartment. The multi-threaded apartment consists of every thread that isn't in a single-threaded apartment. This could happen because the thread explicitly asked to be part of the multi-threaded apartment (by passing the `COINIT_MULTITHREADED` flag to `CoInitializeEx`), or because the thread never expressed any opinion and merely defaulted to the multi-threaded apartment, a situation which is known as being part of the "implicit multi-threaded apartment".

In general, writing single-threaded objects is easier because you don't have to deal with all the race conditions inherent in multi-threaded programming. Multi-threaded objects will typically use locks, and waiting for a lock from a UI thread is a great way to make your UI appear to hang. Furthermore, UI operations are single-threaded, so any component that displays UI or interacts with user interface objects will necessarily be single-threaded.

Windows 2000 introduced the concept of the *neutral-threaded apartment* (NTA). This apartment doesn't have a dedicated thread. Instead, when a call is made into a *neutral-threaded object*, the current thread is temporary commandeered by the neutral apartment for the duration of the call. Neutral-threaded apartments were cool back in the day, but you don't see them much any more, and most people have forgotten that they even exist. Let us not talk about the neutral-threaded apartment again.

Windows 8 introduced a variant of the single-threaded apartment known as the *application single-threaded apartment* (ASTA). It's basically the same as the STA, except that it blocks reentrancy to avoid certain categories of reentrancy bugs and deadlocks.

So that's the quick introduction to apartments. An apartment is a thread or group of threads that can share objects freely among each other. As far as COM is concerned, all threads in an apartment are equivalent. Things get interesting only if you need to communicate between apartments.

Now that I gave you a crash course in apartments, I'm going to confuse matters by introducing contexts. Next time.

[1] There is some optimization opportunity available if the marshaling is within a process. For example, pointers can be marshaled by simply copying them.

[2] The C++ language took a different approach: Objects can be shared freely among threads, *provided* that only one thread performs a non-const operation at a time, and no non-const operation can occur at the same time as a const operation. This basically allows the component to abdicate responsibility for the problem and instead puts the responsibility on the callers.

Applying this rule to the COM world is difficult because it is common for a single component to be referenced by multiple other components, none of which know about each other. For example, there could be multiple clients all with a reference to the same Excel spreadsheet. One add-in is inspecting the data for inconsistencies. Another is cleaning up the data by removing leading and trailing spaces. Another client is monitoring the spreadsheet for changes. And a final client is manipulating data inside the spreadsheet as part of a complex custom calculation. These components have no way of coordinating their actions since they don't know about each other. The only thing they have in common is the Excel spreadsheet itself.

Raymond Chen

Follow