When should I use delayed-marshaling when creating an agile reference?

devblogs.microsoft.com/oldnewthing/20191202-00

December 2, 2019

Raymond Chen

The `RoGetAgileReference` function lets you specify whether you want the marshaling of the wrapped object to take place eagerly or lazily.

Flag	Behavior
`AGILEREFERENCE_ DEFAULT`	Eager marshaling
`AGILEREFERENCE_ DELAYEDMARSHAL`	Lazy marshaling

Why should you choose one over the other?

It's a question of whether you want to do a little work now in the hope of saving more work later.

If you marshal eagerly, then at the point that the agile reference is created, it also gathers the information necessary to create a proxy later. Later, if you use the agile reference from another thread, the agile reference uses that captured information to produce a proxy right then and there.

If you marshal lazily, then at the point that the agile reference is created, it merely remembers the COM context that the agile reference was created in, which is a relatively fast operation. Later, if you use the agile reference from another thread, the agile reference first goes back to the COM context to capture the information necessary to create the proxy, and then it returns to the requesting thread and generates the proxy from that information.

Action	Eager marshal	Lazy marshal
Create agile reference	Create information for proxy	Capture current COM context (fast)
Use from same context	Use original object (fast)	Use original object (fast)

Use from other context (first time)	Create proxy	Call into captured context Create information for proxy Return to original context Create proxy
Use from other context (second and subsequent times)	Create proxy	Create proxy

Observe that if your intended operations are limited to the first two rows, then you're better off doing lazy marshaling, since you avoid the *Create information for proxy* step. But the penalty for guessing wrong is that when you use the agile reference from another context for the first time, you need to do extra work to get back to the original context in order to perform the *Create information for proxy* step.

On the other hand, if you know that you're going to explore rows three and four, then you should do eager marshaling, because it's less expensive to create the information for the proxy up front than on demand. The penalty for guessing wrong is that you went through the *Create information for proxy* step unnecessarily.

Notice that in both cases, it's okay if you guess wrong. The operations will all still succeed. It'll just be less efficient.

Raymond Chen

Follow