COM asynchronous interfaces, part 2: Abandoning the operation

devblogs.microsoft.com/oldnewthing/20220215-00

February 15, 2022

Raymond Chen

Last time, we ran a very basic demonstration of COM asynchronous interfaces. Now we'll start to get fancy.

Our first fancy thing is creating a fire-and-forget asynchronous call. To do this, you simply abandon the call object after calling the `Begin_` method. It means that you never find out the answer, or even learn that it failed. But if you were going to ignore the result anyway, then it doesn't matter.

```
int main(int, char**)
{
  winrt::init_apartment(winrt::apartment_type::multi_threaded);

  auto pipe = CreateSlowPipeOnOtherThread();

  winrt::com_ptr<::AsyncIPipeByte> call;
  auto factory = pipe.as<ICallFactory>();
  winrt::check_hresult(factory->CreateCall(
    __uuidof(::AsyncIPipeByte), nullptr,
    __uuidof(::AsyncIPipeByte),
    reinterpret_cast<::IUnknown**>(call.put())));

  printf("Beginning the Push\n");
  BYTE buffer[15] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                      11, 12, 13, 14, 15 };
  winrt::check_hresult(call->Begin_Push(buffer, 15));

  call = nullptr; // throw it away!
  printf("Abandoned the call\n");

  Sleep(5000); // just so you can see the other thread
  return 0;
}
```

After initiating a `Push` with `Begin_Push`, we simply release the call object (which happens when we set it to `nullptr`) and go on with our lives. The call is still running, but we have lost our ability to access its results.

If you run this program, you can see that the call to `Push` continues running to completion. Depending on the timing, you may even observe the call become abandoned *before* the `Push` method even starts!

Next time, we'll abandon the operation only if it takes too long.

Raymond Chen

Follow