COM asynchronous interfaces, part 4: Doing work while waiting for the asynchronous operation

devblogs.microsoft.com/oldnewthing/20220217-00

February 17, 2022

Raymond Chen

Last time, we learned how to abandon an asynchronous operation after a timeout. But maybe we want to do some useful work while waiting for the operation. (For example, maybe you want to initiate multiple asynchronous operations and let them run in parallel, and then collect the results later.)

As before, we check on the completion state of the asynchronous call by using the `ISynchronize` interface on the call object. But this time, we call `ISynchronize:;Wait` with a timeout of zero, which means that it doesn't actually wait. It just reports on whether the operation has completed. In other words, passing a wait time of zero lets you poll. (Note that cancellation counts as completion.)

Let's make these changes to our original program.

```
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 Pull\n");
  winrt::check_hresult(call->Begin_Pull(100));

  printf("Doing something else for a while...\n");
  auto sync = call.as<::ISynchronize>();
  while (sync->Wait(COWAIT_DEFAULT, 0) == RPC_S_CALLPENDING) {
    printf("Doing important stuff...\n");
    Sleep(250);
  }

  printf("Getting the answer\n");
  BYTE buffer[100];
  ULONG actual;
  winrt::check_hresult(call->Finish_Pull(buffer, &actual));
  printf("Pulled %lu bytes\n", actual);

  return 0;
}
```

The previous program did some fixed amount of work before waiting for the answer. But this version busies itself with important work as long as the operation is not yet complete.

So far, all of our calls to the slow pipe object have been to a thread that is still responsive. Next time, we'll look at how to initiate asynchronous operations and avoid blocking if the server thread is hung.

Raymond Chen

Follow