Windows Runtime delegates and object lifetime in C++/CX, redux

devblogs.microsoft.com/oldnewthing/20200123-00

January 23, 2020

Raymond Chen

One thing to watch out for when using delegates in C++/CX is that invoking a delegate can raise `Platform::DisconnectedException`. If the delegate is inside a C++/CX event, then the runtime will do the work of catching the `Platform:: DisconnectedException` exception, but if you are invoking the delegate manually, then it falls to you to deal with the possibility that the delegate's object no longer exists.

```
public delegate void MenuItemInvoked();
ref class CustomMenuItem
{
public:
  CustomMenuItem(MenuItemInvoked^ handler) :
    m_handler(handler) { }

private:
  MenuItemInvoked^ m_handler;

  void NotifyClientThatItemWasInvoked()
  {
     if (m_handler) m_handler();
  }
}
```

When the item is invoked, we invoke the handler, but it's possible that the object that was supposed to handle the event has already been destroyed. In that case, the runtime will fail to resolve the weak reference to a strong reference, and it will raise the `Platform:: DisconnectedException`. The above code doesn't handle that exception, so it will crash.

What you should do is catch the `Platform:: DisconnectedException` and use that as a signal that the handler is no longer any good and shouldn't be invoked any more.

```
void NotifyClientThatItemWasInvoked()
{
   try
   {
     if (m_handler) m_handler();
   }
   catch (Platform::DisconnectedException^)
   {
     // Handler is no good.
     // Don't bother invoking it any more.
     m_handler = nullptr;
   }
}
```

Raymond Chen

Follow