How do I prevent my C++/WinRT implementation class from participating in COM aggregation?

devblogs.microsoft.com/oldnewthing/20240108-00

January 8, 2024

Raymond Chen

In the Windows Runtime there are these things which are called “composable classes”. Composable classes allow the Windows Runtime to expose classes which applications can derive from, or at least do something that *looks like* class derivation, even though under the covers it’s COM aggregation.

In C++/WinRT, you can give your implementation class the `winrt::composable` tag to make it composable. And if you apply the `unsealed` keyword to your `runtimeclass` in the IDL file, then the C++/WinRT autogenerated implementation templates will specify `winrt::composable` automatically.

But what if you don’t want to be composable? How can you force a build break if somebody tries to enable composability?

From looking at the C++/WinRT code, it appears that you can detect whether the class is composable by snooping at the `outer()` method. For non-composing classes, `outer()` is defined as

```
static constexpr inspectable_abi* outer() noexcept { return nullptr; }
```

Whereas for composing classes, `outer()` is defined as

```
inspectable_abi* outer() noexcept { return m_outer; }
```

Therefore, we can distinguish them by seeing whether `outer()` is a constexpr nullptr.

```
struct Widget : WidgetT<Widget>
{
  Widget()
  {
    static_assert(!outer(),
                  "Widget must not be composable.");
  }

  ⟦ ... ⟧
};
```

If we are not composing, then `outer()` is a constexpr function that indeed returns `nullptr`, so everything passes.

If we are composing, then `outer()` is a non-constexpr function, and the `static_assert` fails because you are trying to assert a runtime expression at compile time.

Unfortunately, the error message doesn't use the static assertion text, because the problem occurred even before the compiler could decide whether the expression passed the assertion.

```
error C2131: expression did not evaluate to a constant
```

But at least the line number points to the `static_assert`, so the developer will be taken to your assertion string.