Why does my C++/WinRT project get unresolved externals for constructors?

devblogs.microsoft.com/oldnewthing/20230920-00

September 20, 2023

Raymond Chen

A customer was trying to build a C++/WinRT project, and they got this linker error:

```
LNK2019 unresolved external symbol "public: __cdecl winrt::Windows::Foundation::
EventHandler<struct winrt::Windows::Foundation::IInspectable>::EventHandler<struct
winrt::Windows::Foundation::IInspectable><class <lambda_5bea4cc895695129
4713c74881f70682> >(class <lambda_5bea4cc8956951294713c74881f70682>)"
```

Okay, the first order of business is trying to parse this thing.

```
winrt::Windows::Foundation::EventHandler<struct winrt::Windows::Foundation::
IInspectable>::
                            EventHandler<struct winrt::Windows::Foundation::
IInspectable>
    <class <lambda_5bea4cc8956951294713c74881f70682> >
    (class <lambda_5bea4cc8956951294713c74881f70682>)
```

After breaking it down this way, we see that this is the `EventHandler<struct winrt::Windows::Foundation::IInspectable>` constructor, specifically a templated constructor whose template parameters are `<class <lambda_5bea4cc8956951294713c74881f70682> >`.

Okay, so the missing external is an `EventHandler` constructor.

Some time ago, we investigated why a C++/WinRT project gets errors of the form "unresolved external symbol … consume_Something" and concluded that it was due to failure to include the header file for the namespace that being used. We then learned that the C++/WinRT library did some error message metaprogramming so that the error is caught at compile time rather than link time. The trick is to have the method return `auto`: If you try to call an `auto`-returning function before it is defined, then the compiler complains that it doesn't yet know what the function returns.

Unfortunately, constructors don't "return" anything, so the `auto` trick is not available for constructors. The code compiles, and then you get a linker error because the constructor definition is missing.

The solution is the same as last time: Include the header file for the namespace you are consuming.

In this case, you are trying to construct an `EventHandler`, which is part of the `winrt::Windows::Foundation` namespace, so the header file you need to include is `#include <winrt/Windows.Foundation.h>`.

Bonus chatter: I was able to come up with some error message metaprogramming to detect the missing header file at the point of construction, but it's rather ugly and (worse) introduces another template for each runtime class and a template specialization for each namespace. Templates are responsible for a large amount of the compilation time of C++/WinRT code, so introducing lots of new templates is something we try to avoid.

But here it is anyway. Maybe you can find something cheaper.

```
// base.h

namespace winrt::impl
{
  template<typename Header>
  struct include_check
  {
    static auto require()
    {
      static_assert(std::is_same_v<Header, void>,
          "Required header file missing (see error details for missing header
file)");
    }
  };
}

// Contoso.0.h

namespace winrt::impl::headers
{
    struct winrt_slash_contoso_dot_h;
}

// Contoso.2.h

namespace winrt::Contoso
{
  struct Widget : winrt::Contoso::IWidget
  {
    template<typename Header = impl::headers::winrt_slash_contoso_dot_h,
             typename = decltype(impl::include_check<Header>
                                 ::require())>
    Widget();
  };
}

// Contoso.h

namespace winrt::impl
{
  template<>
  struct include_check<headers::winrt_slash_contoso_dot_h>
  {
    static void require();
  };
}

namespace winrt::Contoso
{
    template<typename Header, typename>
    inline Widget::Widget()
    {
```

```
        /* body */
    }
}
```

The resulting error is

```
// msvc
error C2338: Required header file missing (see error details for missing header file)
note: while compiling class template member function 'auto winrt::impl::
include_check<Header>::require(void)'
with
[
    Header=winrt::impl::headers::winrt_slash_contoso_dot_h
]
note: see reference to class template instantiation 'winrt::impl::
include_check<Header>' being compiled
with
[
    Header=winrt::impl::headers::winrt_slash_contoso_dot_h
]

// gcc
In instantiation of 'static auto winrt::impl::include_check<Header>::require() [with
Header = winrt::impl::headers::winrt_slash_contoso_dot_h]':
required by substitution of 'template<class Header, class> winrt::Contoso::
Widget::Blah() [with Header = winrt::impl::headers::winrt_slash_contoso_dot_h;
<template-parameter-1-2> = <missing>]'
required from here
error: static assertion failed: Required header file missing (see error details for
missing header file)
|         static_assert(std::is_same_v<Header, void>, "Required header file missing
(see error details for missing header file)");
|                       ~~~~~^~~~~~~~~~~~~~~~~~~~~
note: 'std::is_same_v<winrt::impl::headers::winrt_slash_contoso_dot_h, void>'
evaluates to false

// clang
error: static assertion failed due to requirement 'std::is_same_v<winrt::impl::
headers::winrt_slash_contoso_dot_h, void>': Required header file missing (see error
details for missing header file)
        static_assert(std::is_same_v<Header, void>, "Required header file missing
(see error details for missing header file)");
        ^            ~~~~~~~~~~~~~~~~~~~~~~~~~~~
note: in instantiation of member function 'winrt::impl::include_check<winrt::impl::
headers::winrt_slash_contoso_dot_h>::require' requested here
typename = decltype(impl::include_check<Header>::require())
                                                ^
note: in instantiation of default argument for 'Widget<impl::headers::
winrt_slash_contoso_dot_h>' required here
Widget();
^~~~~~
note: while substituting deduced template arguments into function template 'Widget'
[with Header = (no value), $1 = (no value)]
    winrt::Contoso::Widget w;
```