On combining static libraries that implement C++/WinRT activatable objects

devblogs.microsoft.com/oldnewthing/20201118-00

November 18, 2020

Raymond Chen

In C++/WinRT 2.0, one of the features is enigmatically documented as

> **Smarter and more efficient `module.g.cpp` for large projects with multiple libs**
>
> The `module.g.cpp` file now also contains two additional composable helpers, named **winrt_can_unload_now**, and **winrt_get_activation_factory**. These have been designed for larger projects where a DLL is composed of a number of libs, each with its own runtime classes. In that situation, you need to manually stitch together the DLL's **DllGetActivationFactory** and **DllCanUnloadNow**. These helpers make it much easier for you to do that, by avoiding spurious origination errors. The `cppwinrt.exe` tool's `-lib` flag may also be used to give each individual lib its own preamble (rather than `winrt_xxx`) so that each lib's functions may be individually named, and thus combined unambiguously.

What does this all mean?

A module makes its Windows Runtime objects available to other components by exporting a function with the well-known name `DllGetActivationFactory`. In its default configuration, C++/WinRT generates code for the simple case that you have a single self-contained project that implements all of its own objects: It autogenerates a `DllGet-ActivationFactory` function that knows how to produce the objects in the .winmd files you said you were implementing.

However, it's possible that you've broken a large project into a bunch of libraries, with each library implementing some objects, and you want to link them all together into a single module. That's where the the `-library` flag comes into play.

What you do is have each library pass the `-library name` flag to `cppwinrt.exe`. This causes the C++/WinRT compiler to generate a function named `name_get_ activation_ factory` rather than the default name `winrt_ get_ activation_ factory`. You can now provide your own implementation of `winrt_ get_ activation_ factory`.

In the simplest case, you are just aggregating the information from the various libraries you are consuming:

```
void* __stdcall winrt_get_activation_factory(
    std::wstring_view const& name)
{
  void* factory = library1_get_activation_factory(name);
  if (!factory) factory = library2_get_activation_factory(name);
  if (!factory) factory = library3_get_activation_factory(name);
  if (!factory) factory = library4_get_activation_factory(name);
  return factory;
}
```

The C++/WinRT library does understand a little bit about WRL (the Windows Runtime C++ Template Library), an older library for implementing Windows Runtime objects. Specifically, it understands enough that if `winrt_get_activation_factory` returns `nullptr`, then it asks WRL if it can produce the activation factory before giving up.

If you're using some other library for implementing Windows Runtime objects, you can hook that other library into your custom `winrt_get_activation_factory` function:

```
void* __stdcall winrt_get_activation_factory(
    std::wstring_view const& name)
{
  void* factory = library1_get_activation_factory(name);
  if (!factory) factory = library2_get_activation_factory(name);
  if (!factory) factory = library3_get_activation_factory(name);
  if (!factory) factory = library4_get_activation_factory(name);
  if (!factory) factory = OtherLibrary_GetActivationFactory(name);
  return factory;
}
```

There's a second function that the `-library` flag renames: `winrt_can_unload_now`. You also need to aggregate the various libraries in your implementation of that function:

```
bool __stdcall winrt_can_unload_now() noexcept
{
  return library1_can_unload_now() &&
    library2_can_unload_now() &&
    library3_can_unload_now() &&
    library4_can_unload_now();
}
```

Again, C++/WinRT knows enough about WRL to invite it to the party: If all C++/WinRT libraries are okay with unloading, then it also checks that WRL is okay with unloading before allowing the unload to proceed.

As with `winrt_get_activation_factory`, you can hook any other library into the custom implementation of `winrt_can_unload_now`.

Next time, we'll look at another trick which this flag enables.

Raymond Chen

Follow