How to support a COM interface conditionally in WRL

devblogs.microsoft.com/oldnewthing/20231027-00

October 27, 2023

Raymond Chen

Last time, we looked at conditionally supporting a COM interface in C++/WinRT. Today, we'll solve the same problem for WRL.

For WRL, you directly override the `QueryInterface` method. and `GetIids` methods.

```
struct Widget : Microsoft::WRL::RuntimeClass<
    Microsoft::WRL::RuntimeClassFlags<Microsoft::WRL::WinRt>,
    ::ABI::Contoso::IWidget, ::Windows::Foundation::IStringable>
{
    STDMETHOD(QueryInterface)(REFIID riid, void** ppvObject)
    {
        // If "Stringable" is not enabled,
        // then don't support IStringable.
        if (riid == __uuidof(::IStringable) &&
            !is_stringable_enabled()) {
            *ppvObject = nullptr;
            return E_NOINTERFACE;
        }
        return RuntimeClass::QueryInterface(riid, ppvObject);
    }

    // Implement IWidget methods
    STDMETHOD(WidgetMethod)();

    // Implement IStringable methods
    STDMETHOD(ToString)(HSTRING* result);
};
```

This is basically the same as the C++/WinRT version: When a query comes in for `IStringable`, we check whether `IStringable` support is enabled. If not, then we set the result to `nullptr` and return `E_NOINTERFACE`. Otherwise, we forward the call to the base class to continue normally.

Bonus chatter: Commenter Euro Micelli points out that you should also read the previous entry on the requirement for stability of `is_stringable_enabled()`.