The macros for declaring COM interfaces, revisited: C++ implementation

devblogs.microsoft.com/oldnewthing/20200911-00

September 11, 2020

Raymond Chen

Last time, we looked at the macros for declaring COM interfaces and how they expand when compiled for C++. Now we'll look at the macros you use for implementing the interface.

```
class Class : public ISomething
{
public:
    // *** IUnknown ***
    IFACEMETHOD(QueryInterface)(REFIID riid, void** ppv);
    IFACEMETHOD_(ULONG, AddRef)();
    IFACEMETHOD_(ULONG, Release)();

    // *** ISomething ***
    IFACEMETHOD(Method1)();
    IFACEMETHOD_(int, Method2)();
    IFACEMETHOD(Method3)(int iParameter);
};
```

The `IFACEMETHOD` and `IFACEMETHOD_` macros are used for declaring that your class implements a particular method. The result of the macro expansion is this:

```
class Class : public ISomething
{
public:
    // *** IUnknown ***
    /* IFACEMETHOD(QueryInterface)(REFIID riid, void** ppv); */
    __override virtual __declspec(nothrow)
    HRESULT QueryInterface(REFIID riid, void** ppv);

    /* IFACEMETHOD_(ULONG, AddRef)(); */
    __override virtual __declspec(nothrow)
    ULONG AddRef();

    /* IFACEMETHOD_(ULONG, Release)(); */
    __override virtual __declspec(nothrow)
    ULONG Release();

    // *** ISomething ***
    /* IFACEMETHOD(Method1)(); */
    __override virtual __declspec(nothrow)
    HRESULT Method1();

    /* IFACEMETHOD_(int, Method2)(); */
    __override virtual __declspec(nothrow)
    int Method2();

    /* IFACEMETHOD(Method3)(int iParameter); */
    __override virtual __declspec(nothrow)
    HRESULT Method3(int iParameter);
};
```

Analogous with the `STDMETHOD` and `STDMETHOD_` macros, you use the underscore version of the `IFACEMETHOD` macro (`IFACEMETHOD_`) if the return value is not `HRESULT` .

The `__override` annotation is understood by static analysis tools like PREfast. The annotation means that the static analysis tool should verify that function declaration overrides an identical method in the base class.

The `__override` annotation was introduced as part of SAL, the Microsoft Standard Annotation Language. It has been made redundant by C++11's `override` keyword, but the macros still generate them just the same.

In practice, therefore, you are probably going to write

```
    IFACEMETHOD(Method1)() override;
```

to inform both the static analysis tool and the C++ compiler of your intention to override a method from the base class.

When it comes time to define the method, you do it with the `IFACEMETHODIMP` macro:

```
IFACEMETHODIMP Class::Method1()
{
    ...
}
```

Use `IFACEMETHODIMP_(T)` if the return type is not `HRESULT`.

The above expands to

```
__override HRESULT __stdcall Class::Method1()
{
    ...
}
```

If you want to implement the method inline, then you can put the implementation directly after the declaration inside the class definition.

```
class Class : public ISomething
{
public:
    ...
    IFACEMETHOD(Method1)() override { ... }
    ...
};
```

Raymond Chen

Follow