In C++/CX, hat pointers are contextually convertible to bool, but you can't always static_cast them to bool

devblogs.microsoft.com/oldnewthing/20191205-00

December 5, 2019

Raymond Chen

C++/CX is a language extension intended to make consuming the Windows Runtime easier. It is, however, no longer the C++ projection of choice. That honor now belongs to C++/WinRT, which allows you to consume the Windows Runtime using standard-conforming C++, no language extensions required.

For those of you stuck with C++/CX, here's a little puzzle: What do these functions do?

```
bool Mystery1(Object^ o)
{
    if (o) {
        return true;
    } else {
        return false;
    }
}

bool Mystery2(Object^ o)
{
    return static_cast<bool>(o);
}

bool Mystery3(Object^ o)
{
    return bool(o);
}

bool Mystery4(Object^ o)
{
    return (bool)o;
}
```

You'd think these would all be equivalent, but they're not.

In the first mystery function, the hat pointer `o` is contextually converted to `bool`, and that's done by treating `nullptr` as falsy and anything else as truthy. In this respect, hat pointers are like star pointers.

The remaining mystery functions take the object that `o` points to and attempt to unbox it to a `bool`, and they all behave the same way:

If `o` is	Then you get
`(Object^)true`	`true`
`(Object^)false`	`false`
`nullptr`	`NullReferenceException` thrown
anything else	`InvalidCastException` thrown

If you just want to know what happens and don't care to understand the deep metaphysical significance of those last two rows, I don't blame you.

But that's probably not why you're here. You want to understand the weird crazy world that led to the strange table above.

What's going on is that a `Object^` is really an `IInspectable*` under the hood. And cast operations on `IInspectable*` are performed by doing a `QueryInterface`. In this case, we are casting to `IBox<bool>*`.

If you have a `nullptr`, then the attempt to call `QueryInterface` results in a null pointer dereference, hence the `NullReferenceException`.

If the object is not a boxed `bool`, then the `QueryInterface` fails with `E_NOINTERFACE`, which is expressed in C++/CX as an `InvalidCastException`.

For me, the weird part is that there are two different categories of results: The contextual conversion is different from the other conversions.

It means that you get weird puzzles like this:

```
Object^ p = false;
Object^ q = false;

if (p)                    std::cout << 1;
if ((bool)p)              std::cout << 2;
if (static_cast<bool>(p)) std::cout << 3;
if (p == q)               std::cout << 4;
if (p == false)           std::cout << 5;
if (!p)                   std::cout << 6;
if ((bool)p == (bool)q)   std::cout << 7;
```

What does this fragment print?

Condition	What's happening	Result
`if (p)`	Tests `p` against `nullptr`.	prints `1`
`if ((bool)p)`	Unboxes `p` to `bool`.	does not print
`if (static_cast<bool>(p))`	Unboxes `p` to `bool`.	does not print
`if (p == q)`	Compares two objects for identity.	does not print
`if (p == false)`	Boxes `false` then compares two objects for identity.	does not print
`if (!p)`	Tests `p` against `nullptr`.	does not print
`if ((bool)p == (bool)q)`	Unboxes `p` and `q` and compares them.	prints 7

Converting hat pointers to `bool` is very strange. Be glad you don't have to deal with it.

Next time, we'll look at C++/WinRT. It'll be a lot less strange.

Raymond Chen

Follow