The empty Windows Runtime string is not just a pretty face

devblogs.microsoft.com/oldnewthing/20220706-00

July 6, 2022

Raymond Chen

As I noted some time ago, the empty Windows Runtime string is represented by a null pointer. This has natural but perhaps surprising consequences: Even though it is a null pointer, the empty Windows Runtime string is a real string, with hopes and dreams. Or at least a length and data.

At the ABI level, `WindowsGetStringLen` reports that a null pointer string has a length of zero, and `WindowsGetStringRawBuffer` gives you a buffer that consists of a single null terminator.

Since an empty string and a null pointer are indistinguishable at the ABI layer, if you operate at the ABI layer (using raw `HSTRING` s) or at a thin projection layer (such as C++/CX and C++/WinRT), you can take advantage of this equivalence.

For starters, you don't need to check for a null pointer before trying to use the string, because a null pointer is a perfectly valid `HSTRING` .

ABI	`if (s != nullptr && WindowsGetStringLen(s) == 1)`	`if (s != nullptr && s == HStringReference(L"hi").Get())`
C++/CX	`if (s != nullptr && s->Length() == 1)`	`if (s != nullptr && s == L"hi")`
C++/WinRT	`if (s != hstring{} && s.size() == 1)`	`if (s != hstring{} && s == L"hi"sv)`

If you are checking for a nonempty string, you can just check for null. C++/WinRT and C++/CX even have special methods that tell you directly.

	Slower way	**Quicker way**

ABI	`if (WindowsGetStringLen(s) != 0)`	`if (s != nullptr)`
C++/CX	`if (s->Length() != 0)`	`if (!s->IsEmpty())`
C++/WinRT	`if (s.size() != 0)`	`if (!s.empty())`

Related: The C++/CX `String^` is not an object, even though it wears a hat.

Raymond Chen

Follow