How do I receive ownership of a C-style array in a Windows Runtime component?

devblogs.microsoft.com/oldnewthing/20211202-00

December 2, 2021

Raymond Chen

Last time, we looked at working around the inability to transfer ownership of a C-style array into a Windows Runtime method, and we hit upon the idea of reversing the roles in order to use the *ReceiveArray* pattern. Last time, we looked at how this reversed method would be called. This time, we'll implement it on the provider side.

```
namespace winrt::Sample::implementation
{
    struct Widget : WidgetT<Widget>
    {
        winrt::slim_mutex m_mutex;
        winrt::com_array<int32_t> m_values;

        void SetIndices(IndexProducer const& producer)
        {
            auto values = producer();
            auto lock = std::lock_guard(m_mutex);
            m_values = std::move(values);
        }
    };
}
```

The consumption is a bit tricky due to having to deal with locking.

- We don't want to call into external code while holding the lock.
- Modifying the member variables must be done while holding the lock.

This means that we need to order the operations carefully.

First, we call the delegate outside the lock. That way, the delegate is free to enter any locks it wishes in order to produce the C-style array, which is then transferred to the `com_array`.

Next, we enter the lock to prevent concurrent modification of the `Widget` object.

Once safely inside the lock, we move-assign the C-style array into the `m_values` member, thereby transferring ownership of the data to the member variable. This achieves our goal of getting the data into the `m_values` without ever being copied.

Next time, we'll look at another weird corner case in the Windows Runtime that C-style arrays run up against.

Raymond Chen

Follow