How do I modify the contents of a boxed Windows Runtime value?

devblogs.microsoft.com/oldnewthing/20230405-00

April 5, 2023

Raymond Chen

When you box a value in the Windows Runtime, it's boxed for good. You can't change the value inside a box. Most of the time, it's pretty obvious that you can't do it:

```
// C#
void UpdateX(Nullable<Point> pt)
{
  if (pt != null) {
    // Error: Cannot modify the return value of Point?.Value
    // because it is not a variable
    pt.Value.X = 42.0;
  }
}

// C++/WinRT
void UpdateX(IReference<Point> pt)
{
  if (pt != nullptr) {
    // Error: expression must be a modifiable lvalue
    pt.Value().X = 42.0;
  }
}

// C++/CX
void UpdateX(IBox<Point>^ pt)
{
  if (pt != nullptr) {
    // Error: lvalue required as left operand of assignment
    pt->Value.X = 42.0;
  }
}
```

In the case of C++/WinRT, I think the error is unlikely to occur because you're clearly modifying the result of a function call.

But for C# and C++/CX, the property syntax looks a lot like a member variable access, and you may not realize that the property value is secretly the result of a function call.

Boxed values are read-only. If you look at `IReference<T>` (which is the interface at the ABI layer that is used for Windows Runtime boxed values), you'll see that there is a read-only `Value` property, but no method for setting a new value. Once a value is boxed, you can't change it.

But what if you want to change it?

You'll have to box up a new value and then ask everybody to switch over to it.

```
// C#
Nullable<Point> UpdateX(Nullable<Point> pt)
{
  if (pt != null) {
    var value = pt.Value;
    value.X = 42.0;
    pt = value; // box up a new value
  }
  return pt;
}

// C++/WinRT
IReference<Point> UpdateX(IReference<Point> pt)
{
  if (pt != nullptr) {
    auto value = pt.Value();
    value.X = 42.0;
    pt = winrt::box_value(value); // box up a new value
  }
  return pt;
}

// C++/CX
IBox<Point> UpdateX(IBox<Point>^ pt)
{
  if (pt != nullptr) {
    auto value = pt->Value;
    value.X = 42.0;
    pt = value; // box up a new value
  }
  return pt;
}
```

Of course, you now have to take the updated boxed value and update wherever you got it from.

```
// C#
flyoutShowOptions.Position = UpdateX(flyoutShowOptions.Position);

// C++/WinRT
flyoutShowOptions.Position(UpdateX(flyoutShowOptions.Position()));

// C++/CX
flyoutShowOptions->Position = UpdateX(flyoutShowOptions->Position);
```