Reminder: If you intend to use a C++/WinRT namespace, you must include that namespace’s header file

devblogs.microsoft.com/oldnewthing/20221207-00

December 7, 2022

Raymond Chen

A customer reported that they had a working C++/WinRT XAML app, but when they changed a hard-coded AutomationProperty attached property, to a dynamic binding, the project failed to build.

The working XAML fragment:

```
<ListView x:Name="InstructionList"
    AutomationProperties.Name="Instructions">
```

But this didn’t build:

```
<ListView x:Name="InstructionList"
    AutomationProperties.Name="{x:Bind InstructionsName}">
```

The second version encountered these compiler errors:

```
TutorialPage.xaml.g.hpp(455): C2027: use of undefined type 'winrt::Windows::UI::
Xaml::Automation::AutomationProperties'
TutorialPage.xaml.g.hpp(454): C3861: 'SetName': identifier not found
```

I used my psychic powers to diagnose that in the second version, they forgot to `#include <winrt/Windows.UI.Xaml.Automation.h>` in their precompiled header file. C++/WinRT has the general rule that if you intend to use a type, you must include the corresponding header for the namespace that defines the type.

What happened here is that the `x:Bind` markup extension results in XAML compiler code generation that establishes the bindings programmatically. In this case, the `AutomationProperties.Name="{x:Bind ...}"` caused the XAML compiler to emit a call to `winrt::Windows::UI::Xaml::Automation::AutomationProperties::SetName()`. Which means that the compiler needs to know how to call that `SetName()` method, and the information the compiler needs is in the namespace header file `winrt/Windows.UI.Xaml.Automation.h`.

Another customer ran into a related error message:

```
error C2039: 'Xaml' is not a member of 'winrt::Windows::UI'
```

Again, the reason is that the customer forgot to include the header for the namespace they are trying to use. In this case, they were trying to call the `winrt::Windows::UI::Xaml::Media::Imaging::BitmapImage::DecodePixelWidth()` method, so they header they needed to include was `#include <winrt/Windows.UI.Xaml.Media.Imaging.h>`.

Bonus chatter: The customer who had a problem with `AutomationProperties.Name` also tried this alternative:

```
<ListView x:Name="InstructionList"
    AutomationProperties.Name="{Binding InstructionsName}">
```

This third version built successfully, but the Name property was not set.

The reason for this is that the `{Binding ...}` markup extension uses runtime binding rather than compile-time binding. At runtime, the binding system verifies that the binding source is bindable, and if so, looks for the property in the binding source's metadata. For people migrating from C#, a gotcha is that C# defaults to making all objects bindable, whereas C++/CX and C++/WinRT require you to mark the objects you intend to bind with the `Windows.UI.Xaml.Data.Bindable` attribute.

```
[Windows.UI.Xaml.Data.Bindable]
runtimeclass DataSource
{
    ...
}
```

If the data source is not bindable, or the property is not present, then the binding simply fails silently.

Using `{x:Bind}` results in better runtime performance because all the discovery happens at compile time, and the result is code that sets properties by calling the setters directly. And it means that any typos are caught at compile time, rather than failing silently at runtime.

Exercise: Why did the second customer get an error about the `Xaml` namespace, when the real problem was that they forgot to include the header for the `Imaging` nmamespace?

Raymond Chen

Follow