What if I want the default value for a XAML dependency property to be a mutable object?

devblogs.microsoft.com/oldnewthing/20191003-00

October 3, 2019

Raymond Chen

We saw last time that the default value for a XAML dependency property should be immutable. One easy case of an immutable object is `null`. But what if you want the default value to be a mutable object?

There's no perfect answer for this, but the common solution is to set an explicit value in your constructor.

```
class Light
{
  public Color Color { get; set; }; // read-write property
}

class Widget
{
 ...

 Widget()
 {
  InitializeComponent();
  FrontLight = new Light() { Color = Colors.Red };
 }

 public static readonly DependencyProperty FrontLightProperty =
    DependencyProperty.Register("FrontLight",
    typeof(Light), typeof(Widget));

 // Provide convenient access to the dependency property.
 public Light FrontLight {
  get => (Light)GetValue(FrontLightProperty);
  set => SetValue(FrontLightProperty, value);
 }
}
```

We define the dependency property with `null` as its default value, taking advantage of the overload that assumes that you're okay with `null` being the default value.

But in our constructor, we explicitly set the value of the `FrontLight` property to a brand new red light. By explicitly setting a value (known in XAML terminology as *setting a local value*), we remove the case where XAML needs to produce a default value.

This works out great for most purposes, but there are some subtleties.

One is that the local value is fairly high in the precedence of value sources, sitting above template properties, implicit styles, style triggers, template triggers, style setters, the default style, and inheritance. Setting a local value means that the dependency property cannot be styled, triggered, or inherited.

Another is that somebody might try to reset the property back to the default by calling `ClearValue`. In that case, the value of the dependency property returns to the default value, which we implicitly declared as `null`, rather than returning to the initial value we set in our constructor (the red `Light`).

Raymond Chen

Follow