Is it acceptable to "borrow" dependency properties from unrelated classes?

Asked 16 years, 2 months ago Modified 16 years, 1 month ago Viewed 209 times

4

I'm writing a class that renders some content in WPF, and I want to give the user control over how the content is rendered. The rendering is mostly stroking lines, so I decided to look to the System.Windows.Forms.Shapes.Line class to get an idea of what properties I might want to implement. This led me to implement most of the `StrokeXXXX` properties, and it's a lot of drudgework since each requires metadata to affect the rendering.

A colleague suggested that I just "borrow" the properties from Shape like this:

```
Shape.StrokeThicknessProperty.AddOwner(typeof(MyType));
```

This seems like a pretty good idea. I thought that by doing this I'd lose the ability to set coercion and property changed callbacks, but it looks like the overload that takes a PropertyMetadata allows this. The only downside I can see is if the implementation of Shape changes, it will affect our class, but I'm uncertain how often I expect the .NET interfaces to change significantly.

What do you think? Is this a decent shortcut to defining properties when a well-known class has the behavior you want and a stable interface, or a sure-fire way to play with fire while in a gasoline bath?

wpf dependency-properties

Share Improve this question Follow

asked Oct 22, 2008 at 18:34

OwenP

25.4k ● 13 ● 70 ● 105

2 Answers

Sorted by: Highest score (default)

4

It is very safe and useful to borrow DPs... Read the following post by Dr WPF about the subject!

Here is a few of the "tips" he provide:

- You should always know what the owner class does with any property you borrow.
- You should pay attention to default values and inheritance. Sometimes you need a bool property with a default value of true... other times you may want a default value of false. Sometimes you need a property that inherits... other times you explicitly don't want inheritance. (Borrowing a property like TextElement.FontSize could really screw up things lower in the tree.)
- The owner class may sometimes define a PropertyChangedCallback that will interfere with your ability to use the property as you wish. Always know what the owner class does with the property.
- The owner class may provide a validation routine for the property that prevents you from entering the value you want to specify. Again, always know what the owner class does with the property. The property may be registered in a manner that makes it costly perf-wise, such as FixedPage.Bottom which invalidates the parent's arrange anytime the property changes on an object. Sometimes you may explicitly want this behavior... other times it will just unnecessarily cause layout passes. Again, always know what the owner class does with the property.
- If you use a property in a scenario where the framework itself is trying to use the property (such as TextSearch.TextPath on an ItemsControl), you are liable to find yourself in contention with the framework.

Share Improve this answer Follow

answered Oct 23, 2008 at 6:15

rudigrobler

17.1k ● 12 ● 64 ● 74

1

I'd just keep creating my own dependency properties, personally. It really isn't much extra work, and then you don't have to worry about any of the possible caveats.

Share Improve this answer Follow

answered Oct 24, 2008 at 19:49

Ed Ball

2,059 ● 2 ● 15 ● 13