Whoever

— Today at 1:47 AM

I don't really understand the "PropertyMetadata" section. I have seen it used to do crazy things, but I don't really understand it.

1. [1:47 AM]

I know it can be used to set the default data, but other than that it is an enigma

1. [1:49 AM]

I guess more specifically how the "PropertyChangedCallback propertyChangedCallback" works. I vaguely remember from JavaScript that it basically is a pointer to a method that gets called? But I don't understand how or when it would know to call it. If that is even what it is...

1. @Whoever

I know it can be used to set the default data, but other than that it is an enigma

nohopestage — Today at 1:51 AM

Well, that's pretty much all I've used it for. I don't really think there's much else to it

1. @Whoever

I guess more specifically how the "PropertyChangedCallback propertyChangedCallback" works. I vaguely remember from JavaScript that it basically is a pointer to a method that gets called? But I don't understand how or when it would know to call it. If that is even what it is...

nohopestage — Today at 1:53 AM

It gets called every time the property changes. That's all I know about it lol. Don't know exactly how it works

Whoever

— Today at 1:53 AM

I was just reading about it here. I guess you can call a method each time a dependency property is changed. I think I have seen it used this way before using an unamed method syntax "() => {}" but I don't remember why

1. [1:53 AM]

<https://learn.microsoft.com/en-us/dotnet/desktop/wpf/properties/dependency-property-callbacks-and-validation?view=netdesktop-7.0#property-changed-callbacks>

1. [1:55 AM]

Yeah, I sometimes see this datatype get passed to the metadata parameter "new FrameworkPropertyMetadata" which is also really confusing...

1. @Whoever

I was just reading about it here. I guess you can call a method each time a dependency property is changed. I think I have seen it used this way before using an unamed method syntax "() => {}" but I don't remember why

nohopestage — Today at 1:57 AM

You can also pass a method like that to its constructor instead of a method reference:

new PropertyMetadata(default, (s, e) => { })

Whoever

— Today at 1:57 AM

Yeah, I have seen that before also, but I don't understand what the s, and e parameters are.

1. [1:58 AM]

I know they are populated from the binding, but I don't know what data/datatype they hold (edited)

1. @Whoever

I know they are populated from the binding, but I don't know what data/datatype they hold (edited)

nohopestage — Today at 1:59 AM

It's the same thing as here:

private static void SelectItemCommandChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)

1. [1:59 AM]

So maybe it should a d instead of an s, but that's just a matter of preference (edited)

Whoever

— Today at 1:59 AM

Oh, they look similar to event args then I am guessing. But for Commands instead.

nohopestage — Today at 2:00 AM

They don't really have anything to do with commands

Whoever

— Today at 2:01 AM

The DependencyObject is the sender, and the e is the arguments? (edited)

nohopestage — Today at 2:02 AM

Yes

Whoever

— Today at 2:02 AM

Ok, that makes more sense. Similar to an event.

1. [2:05 AM]

Why do you unregister the methods if you get a null value here?

if (e.NewValue is null)

{

listBox.SelectionChanged -= ListBox\_SelectionChanged;

listBox.Unloaded -= ListBox\_Unloaded;

return;

}

1. [2:06 AM]

Oh because if it is null, you don't want the "changedEvent" to fire?

nohopestage — Today at 2:07 AM

Yes

Whoever

— Today at 2:07 AM

Basically ignoreing it. I see, because the DependencyProperty is going to fire all methods it is registered to regardless if the value is null, so you essentially "disable" them so the null doesn't register. I think I get that part, then you re-register, if the value is valid. (edited)

nohopestage — Today at 2:11 AM

Yeah. Basically, if it's null, there's no command to execute when selection changes, so why even bother proceeding further?

Whoever

— Today at 2:11 AM

Gotcha

nohopestage — Today at 2:14 AM

Now I'm thinking maybe this isn't a good idea? If you have multi-selection enabled, how would you know if the user selected all the items they meant to select?

Whoever

— Today at 2:14 AM

I am not sure what you mean.

nohopestage — Today at 2:15 AM

So the command executes each time an item is selected. If the user can select multiple items, how do you know they're not planning to select more?

Whoever

— Today at 2:16 AM

Oh, I think that is how it works normally.

1. [2:16 AM]

I mean, you have to have some other method of executing the selection

1. [2:16 AM]

Like a button or something

nohopestage — Today at 2:17 AM

Yeah, makes more sense to have a button like that

Whoever

— Today at 2:18 AM

I can't think of a situation where the user would be expected to make all selections at once instantly. You have to allow the user to make the selection through multiple clicks, then have some other action to confirm the selection.

1. [2:18 AM]

I think only in a very unique situtation would you need to fire the event off right as soon as the user makes their selection.

1. [2:19 AM]

To "finalize" their selection I mean

1. [2:19 AM]

Except if it is a single selection. That is the only time I could really see it

nohopestage — Today at 2:19 AM

Yep, that's what I'm talking about

1. [2:20 AM]

So it isn't a problem in your case?

Whoever

— Today at 2:20 AM

No. Not at all.

1. [2:20 AM]

Even if it is a single selection situation, I just get the one item in the list, then execute whatever I need to in the event then and there.

1. [2:21 AM]

I have the user select the different tags they wish to filter on, then they click the "Accept" button that locks in their selection. So it works just fine. (edited)

1. [2:22 AM]

But, because I can't directly bind to the "SelectedItems" on the ListBox, I have to create an event each selection and have it set a variable with the selected items.

nohopestage — Today at 2:23 AM

Oh, got it

Whoever

— Today at 2:23 AM

Then I use the stored selections in those variables to process the filtering

nohopestage — Today at 2:26 AM

Hmm, then you should probably remove the if statement inside ListBox\_SelectionChanged

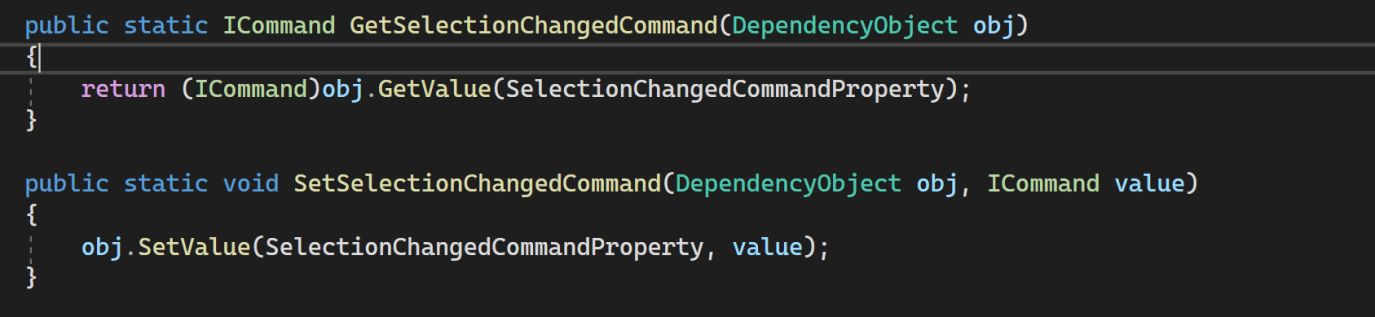
Whoever

— Today at 2:26 AM

Ah, yes. Good point

1. [2:28 AM]

I am a little confused on where these come in:



1. [2:29 AM]

I don't see them used anywhere in the class, though I know they must be since the naming convention caused an error.

nohopestage — Today at 2:30 AM

They're just required for the attached property to work, and the getter is actually used inside ListBox\_SelectionChanged

1. [2:30 AM]

Oh, nvm it isn't

1. [2:30 AM]

But it could be used instead

1. [2:31 AM]

Like this:

ICommand command = GetSelectionChangedCommand(listBox);

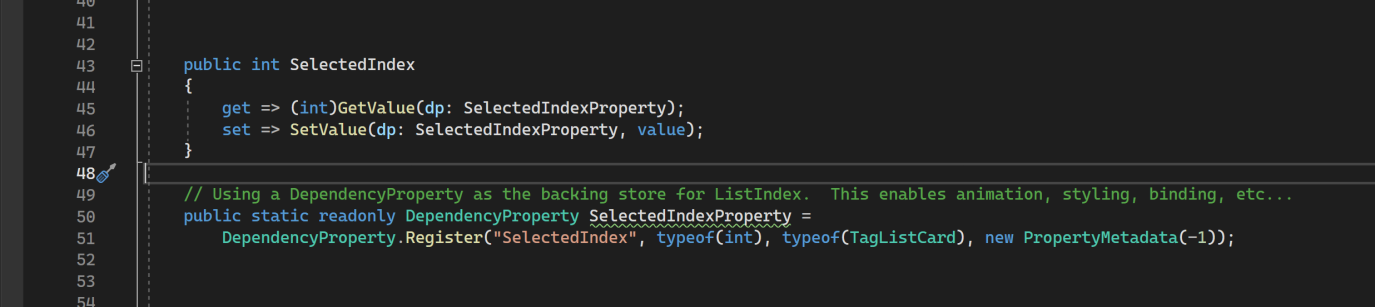
Whoever

— Today at 2:32 AM

Oh, I see what you mean about it could be used there.

1. [2:34 AM]

In the past I haven't needed to add a get/set method with the DependencyProperties. What makes the one you wrote different from say something like this?



1. @Whoever

In the past I haven't needed to add a get/set method with the DependencyProperties. What makes the one you wrote different from say something like this?

nohopestage — Today at 2:34 AM

This is a DependencyProperty

Whoever

— Today at 2:35 AM

Isn't that what you have defined?

nohopestage — Today at 2:35 AM

No, it's an AttachedProperty. They are different

1. [2:36 AM]

Nvm I just made up AttachedProperty lol

Whoever

— Today at 2:36 AM

LOL

nohopestage — Today at 2:36 AM

But they are different, though they are the same class

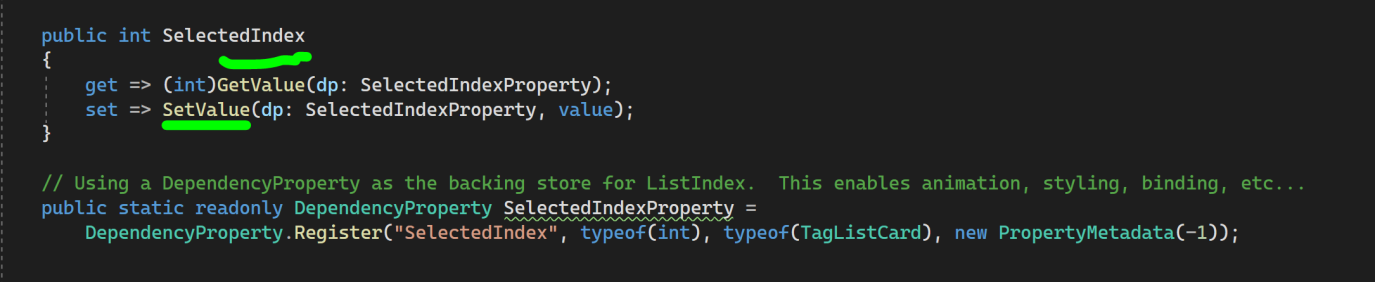
1. [2:37 AM]

Dependency properties are attached to a specific DependencyObject, whereas attached properties can be set on any DependencyObject

Whoever

— Today at 2:40 AM

I am trying to figure out the specification of one vs the other. I just noticed in mine I have these methods. I think maybe they are shorthand for the class somehow? It looks like kind of what you did in your methods? I am trying to figure out what distinguishes one from the other. Somehow yours has to have the get and set methods, but mine doesn't seem to need to? I am assuming the "GetValue" and "SetValue" are in replace of the methods you wrote out? Still trying to piece it together.



1. [2:41 AM]

I am thinking those are some kind of "default" methods, that replace the ones you wrote?

1. [2:42 AM]

Oh wait. You don't specify a property

1. @Whoever

I am thinking those are some kind of "default" methods, that replace the ones you wrote?

nohopestage — Today at 2:43 AM

No, not really. The get/set methods replace them

Whoever

— Today at 2:43 AM

I don't see something similar to "public static readonly ICommand SelectionChangedCommand {get; set'}" Interesting

nohopestage — Today at 2:44 AM

With an attached property you wouldn't be able to just have a property like that, because you need to get the DependencyObject the attached property is set on (edited)

1. [2:47 AM]

I'm not that good at explaining stuff, so sorry if I'm not being helpful at all haha

Whoever

— Today at 2:47 AM

Your right, I am seeing that now. But I am still missing how the compiler knows that this:

public static readonly DependencyProperty SelectionChangedCommandProperty =

DependencyProperty.RegisterAttached(

"SelectionChangedCommand",

typeof(ICommand),

typeof(ListBoxExtensions),

new PropertyMetadata(default, SelectItemCommandChanged));

is different than this

public static readonly DependencyProperty SelectedIndexProperty =

DependencyProperty.Register("SelectedIndex", typeof(int), typeof(TagListCard), new PropertyMetadata(-1));

1. [2:47 AM]

Oh wait

1. [2:47 AM]

I see it as soon as I typed that out now

1. [2:47 AM]

DependencyProperty.RegisterAttached VS DependencyProperty.Register

nohopestage — Today at 2:47 AM

Yeah, I was about to bring that up

Whoever

— Today at 2:49 AM

Ok. It makes a lot more sense now.

1. [2:51 AM]

For a min they looked identical to me (edited)

nohopestage — Today at 2:51 AM

Btw, you can use propa for attached properties the same way you use propd for dependency properties

1. [2:51 AM]

It will generate all the stuff you need

Whoever

— Today at 2:51 AM

Oh nice. That is good to know

1. [2:52 AM]

What is the main purpose of a dependency attachment? To gain access to properties through commands you normally wouldn't have access to, or is there another functionality they are intended for?

nohopestage — Today at 2:54 AM

You mean dependency/attached properties?

Whoever

— Today at 2:55 AM

Yeah

nohopestage — Today at 2:58 AM

They can be used for anything you want really. Say you have a custom button style and you want to be able to set CornerRadius without having to change the style every time. You could define an attached property the style would bind to. This way you would be able to set CornerRadius the same way you set SelectionChangedCommand

1. [2:59 AM]

You could also create a custom button class derived from Button and define a dependency property for CornerRadius. Works the same way

1. [3:00 AM]

It would just be <controls:CustomButton CornerRadius="4" /> (edited)

1. [3:03 AM]

The binding in the style would differ though. With an attached property you would do:

`CornerRadius="{Binding extensions:ButtonExtensions.CornerRadius, RelativeSource={RelativeSource TemplatedParent}}"

And with a dependency property:

CornerRadius="{TemplateBinding CornerRadius}"

(edited)

nohopestage — Today at 3:05 AM

You can't use TemplateBinding for attached properties, because it only allows binding to properties that are defined in the control the template is targeting

1. [3:06 AM]

Hope that makes sense

Whoever

— Today at 3:06 AM

Going through it atm

1. [3:06 AM]

Is CornerRadius not bindable?

1. [3:09 AM]

So they allow you to bind to properties that you couldn't normally bind to?

1. [3:10 AM]

I also don't understand what a "{TemplateBinding}" does vs using "{Binding}"

1. @Whoever

So they allow you to bind to properties that you couldn't normally bind to?

nohopestage — Today at 3:13 AM

No, it's just that CornerRadius doesn't exist in Button (edited)

Whoever

— Today at 3:14 AM

Oh, like if you had a border around it, and you wanted to access the border's cornerradius on the custom control, you could add it that way?

1. @Whoever

I also don't understand what a "{TemplateBinding}" does vs using "{Binding}"

nohopestage — Today at 3:14 AM

TemplateBinding is only used inside a ControlTemplate

Whoever

— Today at 3:15 AM

You're going to hate me... but what is a ControlTemplate and when would you use it?

1. [3:16 AM]

Oh nvm I looked it up. Looks like if you want to change the graphics of the control if I understand correctly

nohopestage — Today at 3:17 AM

Yes

1. @Whoever

Oh, like if you had a border around it, and you wanted to access the border's cornerradius on the custom control, you could add it that way?

nohopestage — Today at 3:17 AM

Correct

Whoever

— Today at 3:18 AM

Was just looking at this here about the templates: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/controls/how-to-create-apply-template?view=netdesktop-7.0> Looks really cool for adding flair to the app. Will have to look into this a different day lol.

**1**

1. [3:21 AM]

Thanks for all your help NoHopeStage. I think I am going to play around with the style a little bit more then head to bed. You were instrumental in helping me figure out how to manage a style with a changing binding. Super happy with it. Thank you so much!

**NEW**

nohopestage — Today at 3:23 AM

Glad to have helped!