**Dependency Property**

In C#, there are Normal properties with Get and Set.

* Dependency Property is wrapped within (is the backing field for) a Normal property
* Help achieve DataBinding, Animation, Styling, Callbacks, etc.

A screen shot of a computer code

Description automatically generated with low confidence

A screen shot of a computer code

Description automatically generated with low confidence

* Dependency Properties are always static & readonly, so it always available to all controls
* “OnSelectedValueChanged” is a callback that will be triggered by SetValue

\*\* Major use: provide change notification

**DependencyProperty vs INotifyPropertyChanged**

* DependencyProperty uses **less memory** and is **faster** than INotifyPropertyChanged
* There are some limitations with DependencyProperty:
  + To create a DependencyProperty your objects needs to inherit from DependencyObject.
  + only the thread that the DependencyObject was created on may access the DependencyObject directly.
  + DependencyObject seals Equals and GetHashCode(),
  + They are not marked as Serializable : (but you can use the XAMLWriter and XAMLReader to do so...
* Don’t use DependencyProperty if you’re inheritting from a base class you can’t control
* Animations: **Using DependencyProperty make the poperties animatable**. If you want to animate a property, there is no simple work-around because, as the MSDN says : In order to be animated, the animation's target property must be a dependency property.
* When you performs **testing** on your object, you will be in trouble if you use DependencyObject: the test are not done on the same thread that created the object and then throws you a "System.InvalidOperationException: The calling thread cannot access this object because a different thread owns it"