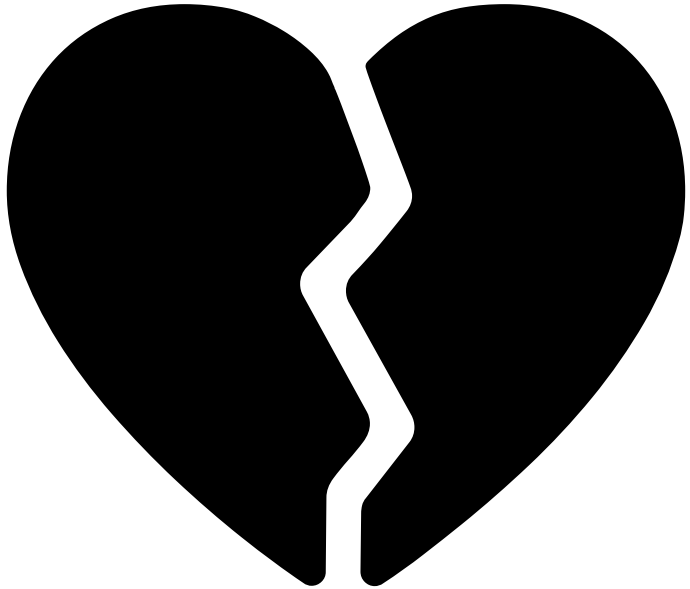
Still building mobile and desktop apps like it's 2006?

Have we been doing XAML wrong all this time?




2022
Developer
Survey
In May 2022 over 70,000 developers told us how they learn and level up, which tools they're using, and what they want.
Read the overview →
Methodology →

“when compared with "dreaded" languages,
loved programming languages are NEW and/or have great tooling”

Blog / Series

Programming Thoughts

The Slow March of Progress in Programming Language Tooling

11 minute read Updated: July 11, 2022

Adam Gordon Bell

The 2022 Stack Overflow developer survey is out!

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns="http://xamarin.com/schemas/2014/forms"
xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
```

```
x:DataType="local:XXXXXXXXXX“

Property="{x:Bind Xxxxx}"
```

RaphMar2021 commented 3 days ago

Seriously, i can't live without the XAML designer.

```
static Grid CreateGrid() => new()
{
            RowSpacing = 1,

            RowDefinitions = Rows.Define(
                        (Row.Title, 20),
                        (Row.Description, 20),
                        (Row.BottomPadding, 1)),

            Children =
            {
                        new Label { LineBreakMode = LineBreakMode.TailTruncation, MaxLines = 1 }
                                    .Row(Row.Title)
                                    .Font(size: 16).DynamicResource(Label.TextColorProperty, nameof(BaseTheme.PrimaryTextColor))
                                    .Top().Padding(10, 0)
                                    .Bind(Label.TextProperty, nameof(StoryModel.Title))
                                    .SemanticHint("The title of the news article."),

                        new Label().Row(Row.Description)
                                    .Font(size: 13).DynamicResource(Label.TextColorProperty, nameof(BaseTheme.SecondaryTextColor))
                                    .Paddings(10, 0, 10, 5)
                                    .Bind(Label.TextProperty, nameof(StoryModel.Description))
                                    .SemanticHint("The description of the news article.")
            }
};
```

```
<Grid RowSpacing="1" RowDefinitions="20,20,1">
    <Label
        LineBreakMode="TailTruncation" MaxLines="1"
        Grid.Row="0"
        FontSize="16" TextColor="{DynamicResource PrimaryTextColor}"
        VerticalOptions="Start" Padding="10,0"
        Text="{Binding Title}"
        SemanticProperties.Hint="The title of the news article."
        />
    <Label
        Grid.Row="1"
        FontSize="13" TextColor="{DynamicResource SecondaryTextColor}"
        Padding="10,0,10,5"
        Text="{Binding Description}"
        SemanticProperties.Hint="The description of the news article."
        />
</Grid>
```

What can we do about this?

Have we been doing XAML wrong all this time?

What if we treat XAML like a “proper” programming language?

What is a “proper” programming language?

- Easy to read
- Easy to write
- Clear / unambiguous
- Succinct – not unnecessarily verbose
- Consistent
- Easy to understand
- Easy to spot mistakes
- Easy to maintain/modify
- Good tooling

What if we treat XAML like any other text-based file that is compiled into our application?

I want to be wrong about this!


ONE PENNY
1

“I’ve never seen XAML done well before”

“I already do this for other languages (C#, HTML, JS, CSS) but not for XAML”

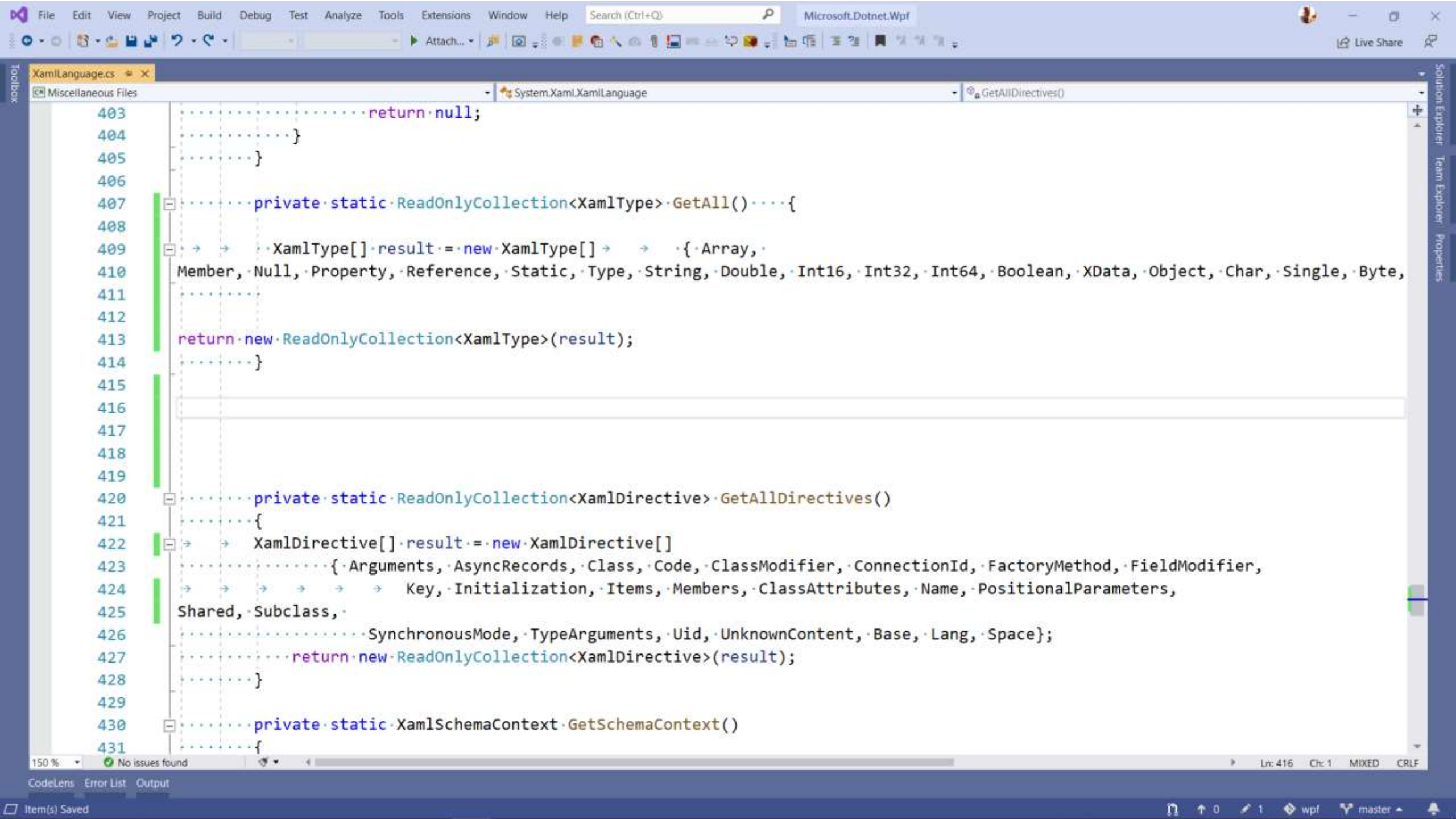
“Why is XAML development like this?”

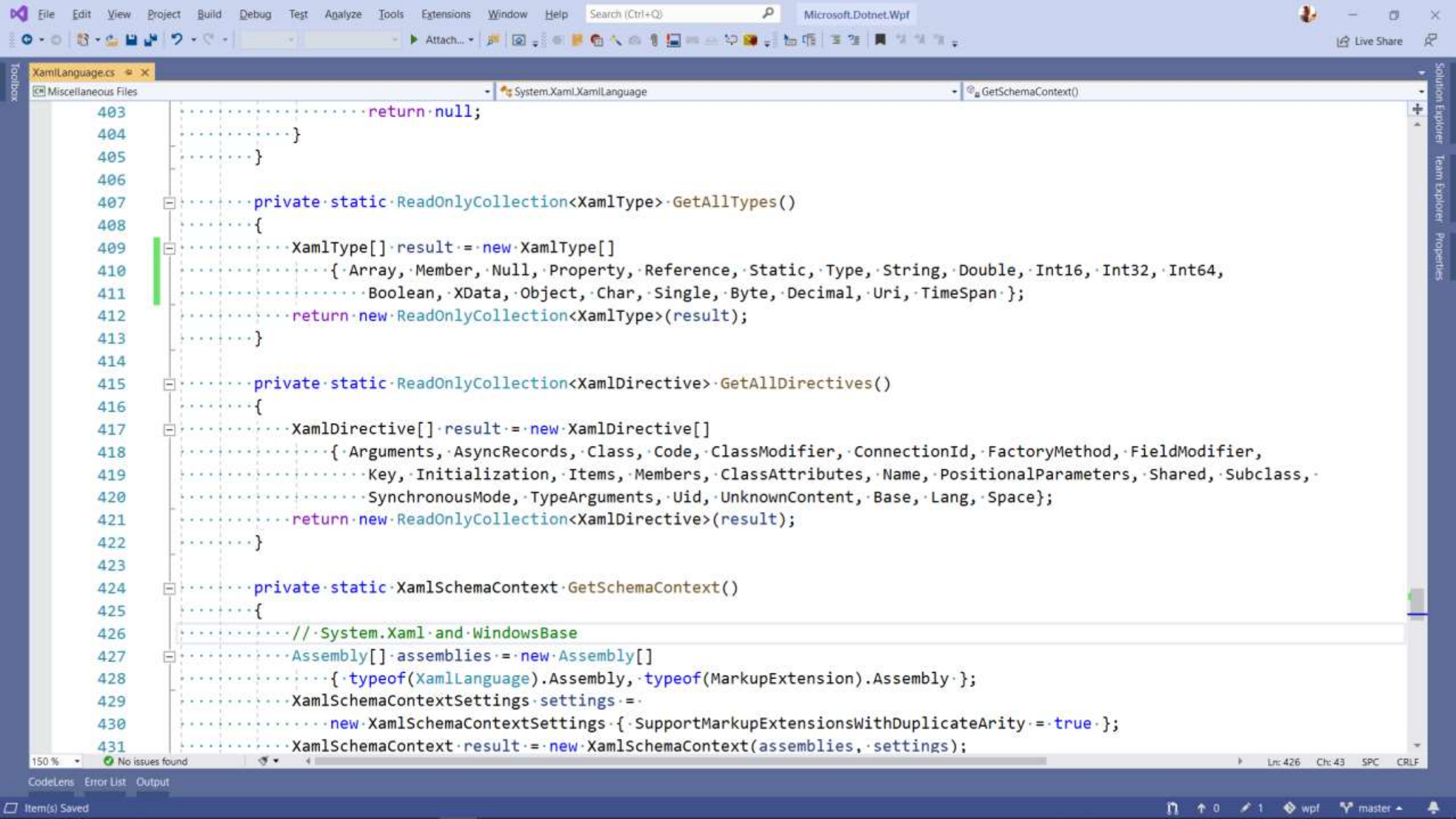

HISTORY

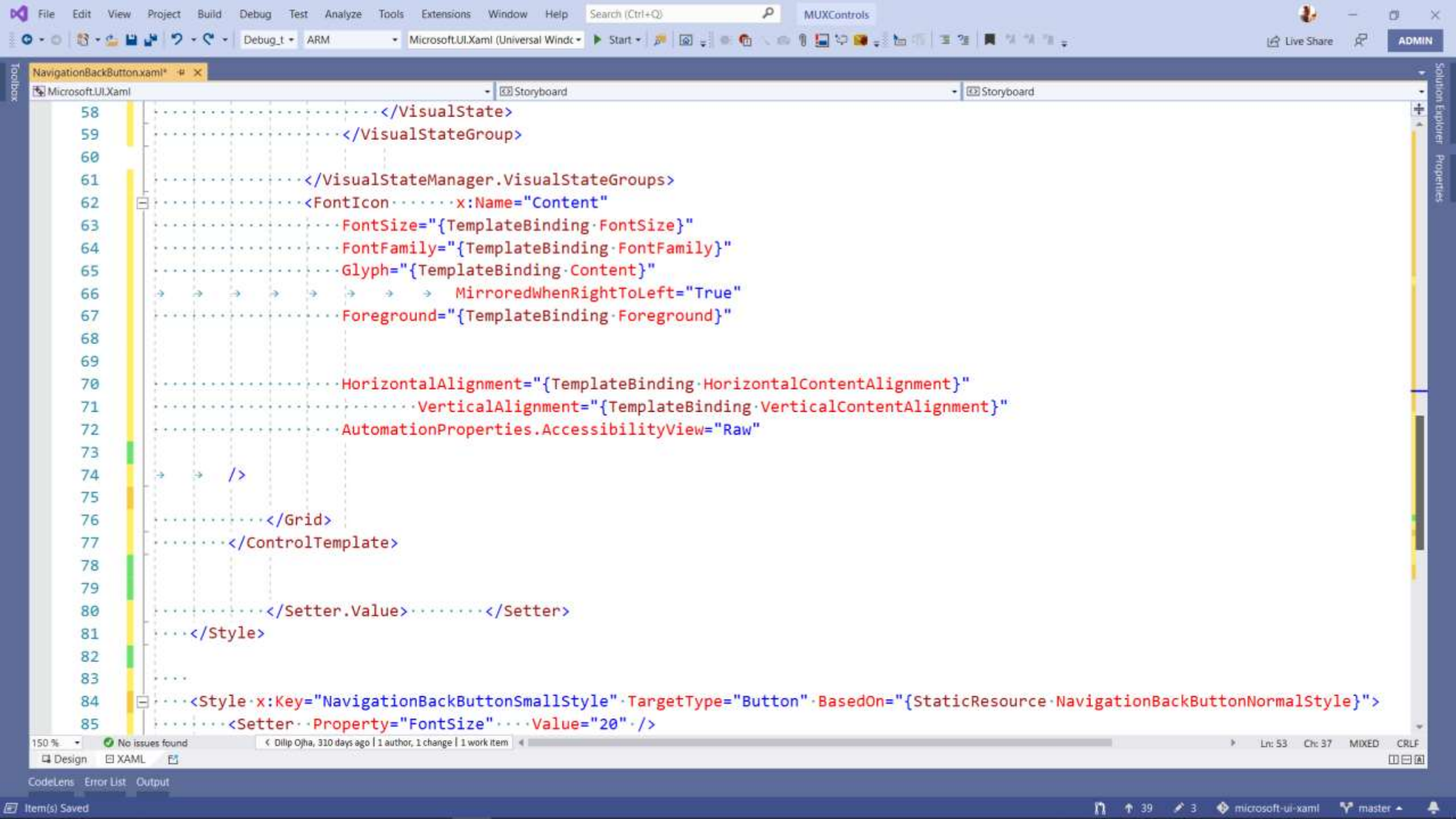
But, times have changed. Sadly, XAML (and how we use it) hasn’t!

Let’s think about

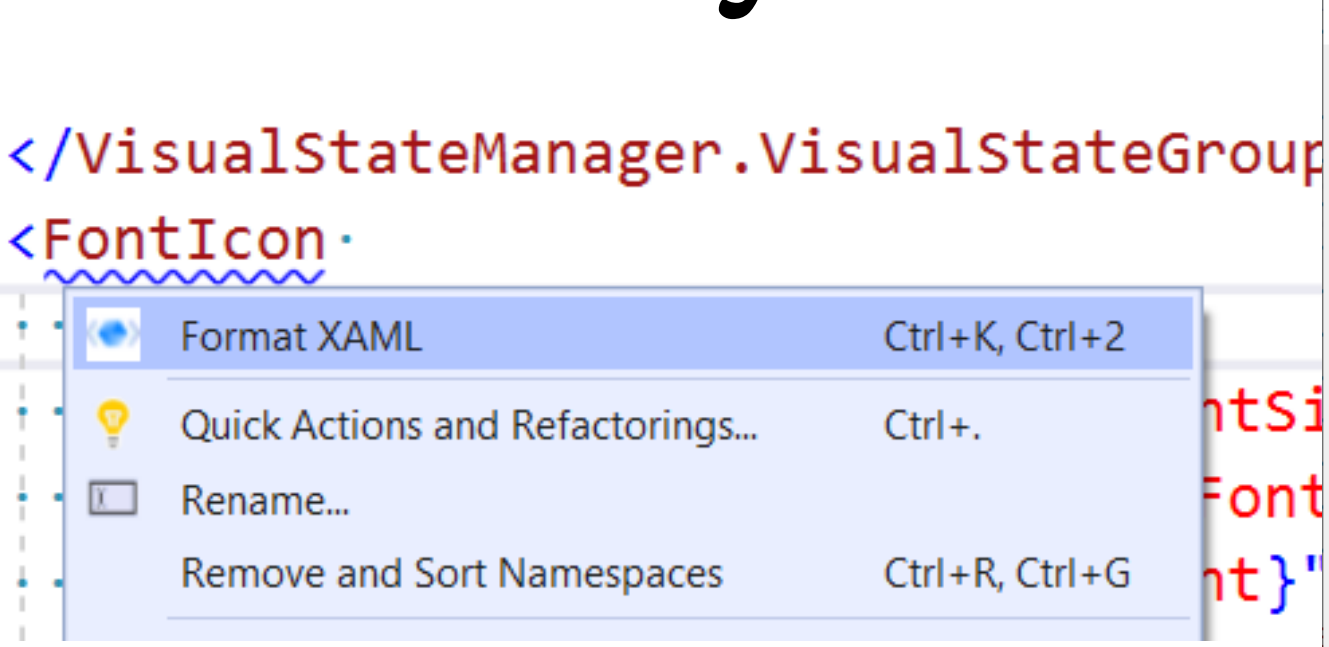
Formatting

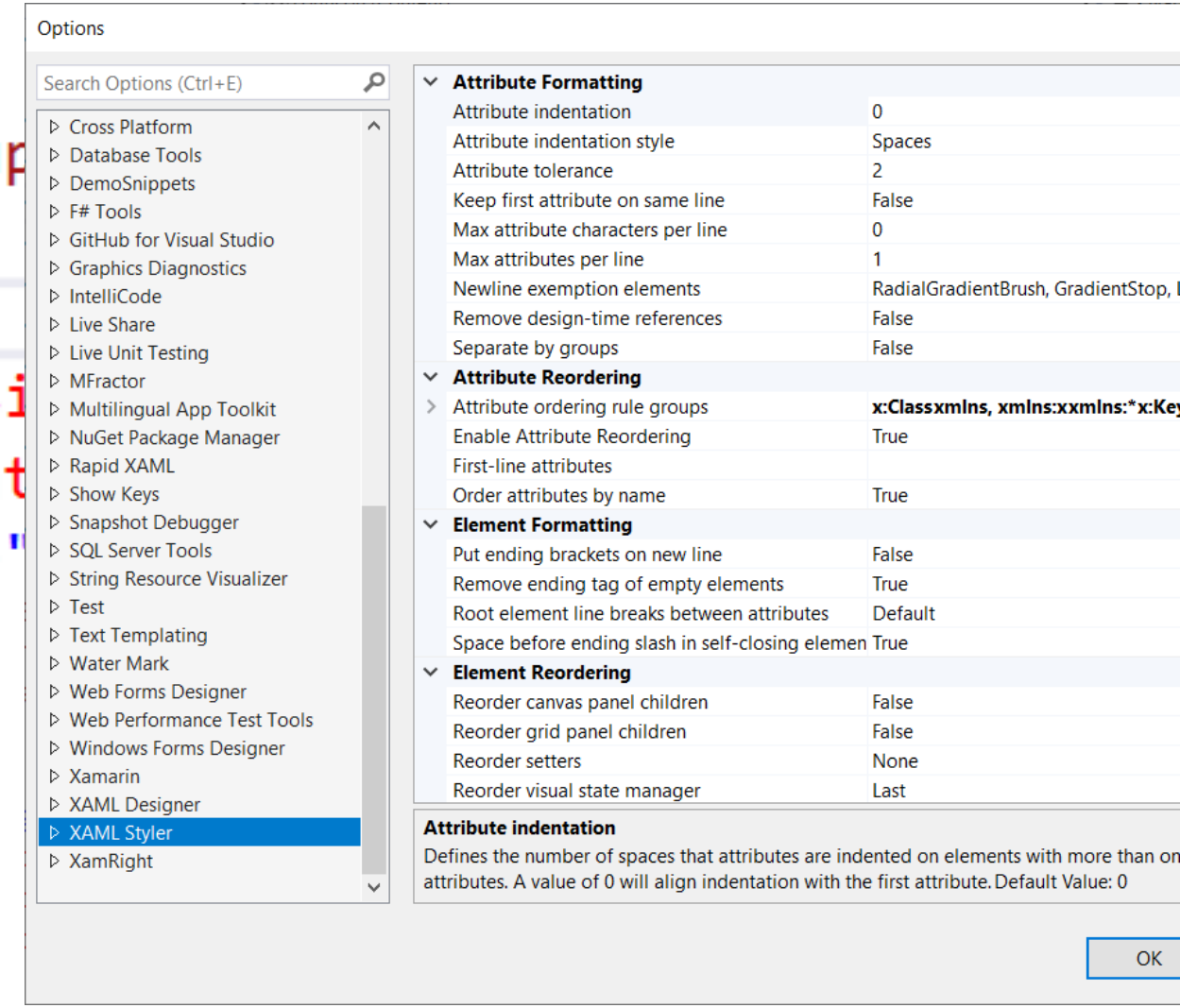

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Microsoft.Dotnet.Wpf
Live Share
XamlLanguage.cs
Miscellaneous Files
System.Xaml.XamlLanguage
GetAllDirectives()
403 return null;
404 }
405 }
406
407 private static ReadOnlyCollection<XamlType> GetAll() {
408
409 XamlType[] result = new XamlType[] { Array,
410 Member, Null, Property, Reference, Static, Type, String, Double, Int16, Int32, Int64, Boolean, XData, Object, Char, Single, Byte,
411
412
413 return new ReadOnlyCollection<XamlType>(result);
414 }
415
416
417
418
419
420 private static ReadOnlyCollection<XamlDirective> GetAllDirectives()
421 {
422 XamlDirective[] result = new XamlDirective[]
423 { Arguments, AsyncRecords, Class, Code, ClassModifier, ConnectionId, FactoryMethod, FieldModifier,
424 Key, Initialization, Items, Members, ClassAttributes, Name, PositionalParameters,
425 Shared, Subclass,
426 SynchronousMode, TypeArguments, Uid, UnknownContent, Base, Lang, Space};
427 return new ReadOnlyCollection<XamlDirective>(result);
428 }
429
430 private static XamlSchemaContext GetSchemaContext()
431 {
150 %
No issues found
Ln: 416 Ch: 1 MIXED CRLF
CodeLens Error List Output
Item(s) Saved
wpf master
Toolbox
Solution Explorer Team Explorer Properties

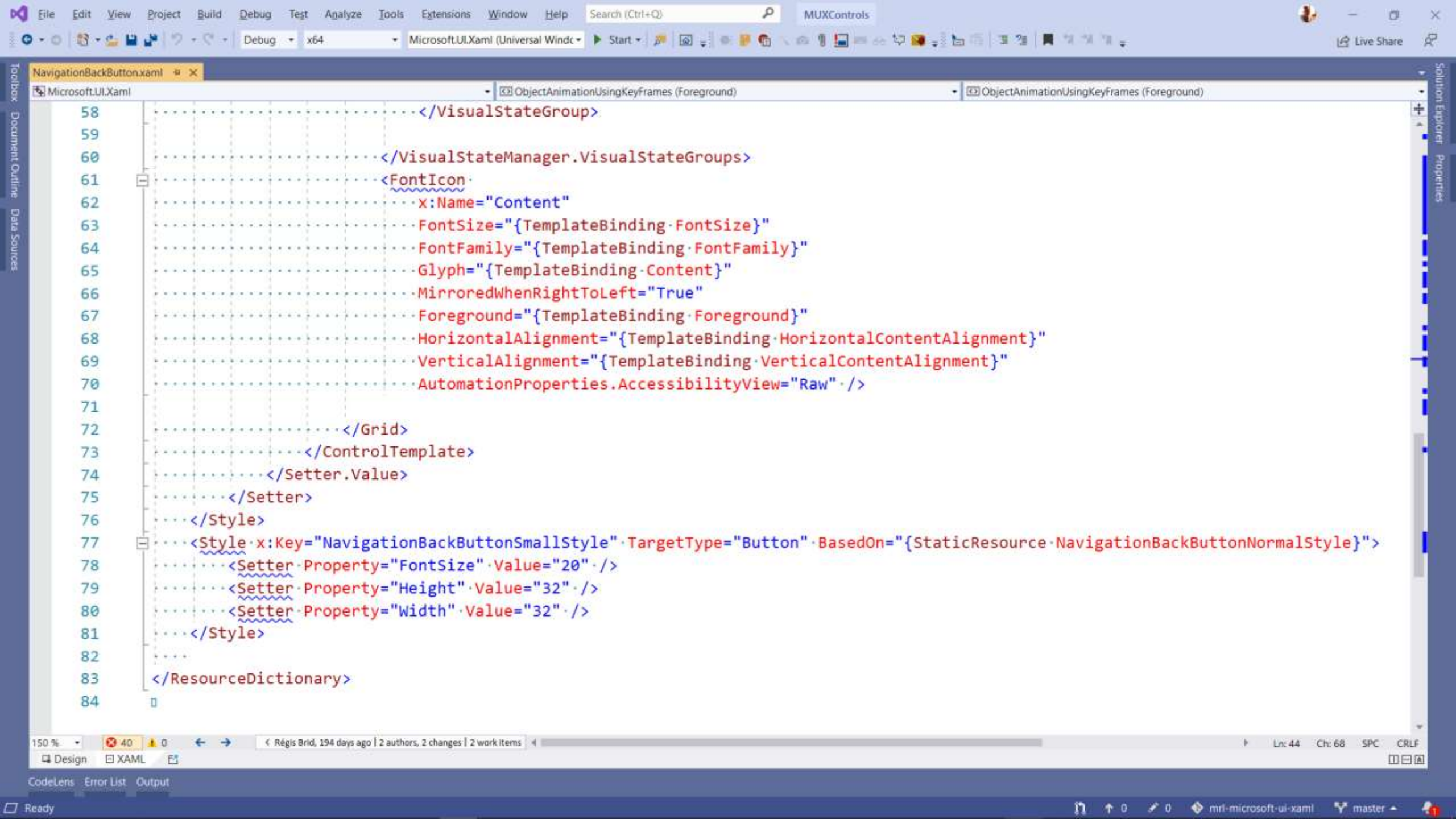

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Microsoft.Dotnet.Wpf
Attach...
Live Share
XamlLanguage.cs
Miscellaneous Files
System.Xaml.XamlLanguage
GetSchemaContext()
403 return null;
404 }
405 }
406
407 private static ReadOnlyCollection<XamlType> GetAllTypes()
408 {
409 XamlType[] result = new XamlType[]
410 { Array, Member, Null, Property, Reference, Static, Type, String, Double, Int16, Int32, Int64,
411 Boolean, XData, Object, Char, Single, Byte, Decimal, Uri, TimeSpan };
412 return new ReadOnlyCollection<XamlType>(result);
413 }
414
415 private static ReadOnlyCollection<XamlDirective> GetAllDirectives()
416 {
417 XamlDirective[] result = new XamlDirective[]
418 { Arguments, AsyncRecords, Class, Code, ClassModifier, ConnectionId, FactoryMethod, FieldModifier,
419 Key, Initialization, Items, Members, ClassAttributes, Name, PositionalParameters, Shared, Subclass,
420 SynchronousMode, TypeArguments, Uid, UnknownContent, Base, Lang, Space};
421 return new ReadOnlyCollection<XamlDirective>(result);
422 }
423
424 private static XamlSchemaContext GetSchemaContext()
425 {
426 // System.Xaml and WindowsBase
427 Assembly[] assemblies = new Assembly[]
428 { typeof(XamlLanguage).Assembly, typeof(MarkupExtension).Assembly };
429 XamlSchemaContextSettings settings =
430 new XamlSchemaContextSettings { SupportMarkupExtensionsWithDuplicateArity = true };
431 XamlSchemaContext result = new XamlSchemaContext(assemblies, settings);
150 %
No issues found
Ln: 426 Ch: 43 SPC CRLF
CodeLens Error List Output
Item(s) Saved
wpf master
Toolbox
Solution Explorer Team Explorer Properties


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
MUXControls
Debug_t
ARM
Microsoft.UI.Xaml (Universal Windc
Start
Live Share
ADMIN
NavigationBackButton.xaml*
Microsoft.UI.Xaml
Storyboard
Storyboard
58 </VisualState>
59 </VisualStateGroup>
60
61 </VisualStateManager.VisualStateGroups>
62 <FontIcon x:Name="Content"
63 FontSize="{TemplateBinding FontSize}"
64 FontFamily="{TemplateBinding FontFamily}"
65 Glyph="{TemplateBinding Content}"
66 MirroredWhenRightToLeft="True"
67 Foreground="{TemplateBinding Foreground}"
68
69
70 HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
71 VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
72 AutomationProperties.AccessibilityView="Raw"
73
74 />
75
76 </Grid>
77 </ControlTemplate>
78
79
80 </Setter.Value> </Setter>
81 </Style>
82
83
84 <Style x:Key="NavigationBackButtonSmallStyle" TargetType="Button" BasedOn="{StaticResource NavigationBackButtonNormalStyle}">
85 <Setter Property="FontSize" Value="20" />
150 %
No issues found
Dilip Ojha, 310 days ago | 1 author, 1 change | 1 work item
Ln: 53 Ch: 37 MIXED CRLF
Design
XAML
CodeLens
Error List
Output
Item(s) Saved
39
3
microsoft-ui-xaml
master

XAML Styler


/VisualStateManager.VisualStateGroup
<FontIcon
Format XAML Ctrl+K, Ctrl+2
Quick Actions and Refactorings... Ctrl+.
Rename...
Remove and Sort Namespaces Ctrl+R, Ctrl+G


Options
Search Options (Ctrl+E)
Cross Platform
Database Tools
DemoSnippets
F# Tools
GitHub for Visual Studio
Graphics Diagnostics
IntelliCode
Live Share
Live Unit Testing
MFractor
Multilingual App Toolkit
NuGet Package Manager
Rapid XAML
Show Keys
Snapshot Debugger
SQL Server Tools
String Resource Visualizer
Test
Text Templating
Water Mark
Web Forms Designer
Web Performance Test Tools
Windows Forms Designer
Xamarin
XAML Designer
XAML Styler
XamRight
Attribute Formatting
Attribute indentation 0
Attribute indentation style Spaces
Attribute tolerance 2
Keep first attribute on same line False
Max attribute characters per line 0
Max attributes per line 1
Newline exemption elements RadialGradientBrush, GradientStop,
Remove design-time references False
Separate by groups False
Attribute Reordering
Attribute ordering rule groups x:Classxmlns, xmlns:xxmlns:*x:Key
Enable Attribute Reordering True
First-line attributes
Order attributes by name True
Element Formatting
Put ending brackets on new line False
Remove ending tag of empty elements True
Root element line breaks between attributes Default
Space before ending slash in self-closing elemen True
Element Reordering
Reorder canvas panel children False
Reorder grid panel children False
Reorder setters None
Reorder visual state manager Last
Attribute indentation
Defines the number of spaces that attributes are indented on elements with more than on
attributes. A value of 0 will align indentation with the first attribute. Default Value: 0
OK


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
MUXControls
Debug x64 Microsoft.UI.Xaml (Universal Windc Start
Live Share
NavigationBackButton.xaml
Microsoft.UI.Xaml
ObjectAnimationUsingKeyFrames (Foreground)
ObjectAnimationUsingKeyFrames (Foreground)
58 </VisualStateGroup>
59
60 </VisualStateManager.VisualStateGroups>
61 <FontIcon
62 x:Name="Content"
63 FontSize="{TemplateBinding FontSize}"
64 FontFamily="{TemplateBinding FontFamily}"
65 Glyph="{TemplateBinding Content}"
66 MirroredWhenRightToLeft="True"
67 Foreground="{TemplateBinding Foreground}"
68 HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
69 VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
70 AutomationProperties.AccessibilityView="Raw" />
71
72 </Grid>
73 </ControlTemplate>
74 </Setter.Value>
75 </Setter>
76 </Style>
77 <Style x:Key="NavigationBackButtonSmallStyle" TargetType="Button" BasedOn="{StaticResource NavigationBackButtonNormalStyle}">
78 <Setter Property="FontSize" Value="20" />
79 <Setter Property="Height" Value="32" />
80 <Setter Property="Width" Value="32" />
81 </Style>
82
83 </ResourceDictionary>
84
150 %
40
0
Régis Brid, 194 days ago | 2 authors, 2 changes | 2 work items
Ln: 44 Ch: 68 SPC CRLF
Design XAML
CodeLens Error List Output
Ready
mrl-microsoft-ui-xaml master
Toolbox Document Outline Data Sources
Solution Explorer Properties


Microsoft Blend
PREVIEW
Microsoft Visual Studio
2022

Let’s think about

Responsibility

```
public int PlaceOrder(Basket basket, User user)
{
    if (basket.Items.Count == 0)
    {
        throw new EmptyBasketException();
    }

    foreach (var item in basket.Items)
    {
        if (!this.IsItemAvailable(item))
        {
            throw new EmptyBasketException();
        }
    }

    var invoice = this.AddInvoice(user, basket.Items);

    this.db.SaveOrder(invoice);

    SmtpClient client = new SmtpClient();
    client.Send(new MailMessage("noreply@onlineshop.example",
                                user.EmailAddress,
                                "Order Confirmation",
                                this.CreateEmailBody(invoice)));

    return ORDER_PLACED;
}
```

```
public int PlaceOrder(Basket basket, User user)
{
    if (basket.Items.Count == 0)
    {
        throw new EmptyBasketException();
    }

    foreach (var item in basket.Items)
    {
        if (!this.IsItemAvailable(item))
        {
            throw new EmptyBasketException();
        }
    }

    var invoice = this.AddInvoice(user, basket.Items);

    this.db.SaveOrder(invoice);

    SmtpClient client = new SmtpClient();
    client.Send(new MailMessage("noreply@onlineshop.example",
                                user.EmailAddress,
                                "Order Confirmation",
                                this.CreateEmailBody(invoice)));

    return ORDER_PLACED;
}
```

A

C

D

```
<Button
    Content="A"
    Grid.Row="5"
    Grid.Column="3"
    Margin="10,10,20,30"
    Click="On_A_Clicked" />
<Button
    Content="C"
    Grid.Row="5"
    Grid.Column="4"
    Margin="0,10,20,30"
    Click="On_C_Clicked" />
<Button
    Content="D"
    Grid.Row="5"
    Grid.Column="5"
    Margin="0,10,40,30"
    Click="On_D_Clicked" />
```

```
<c:RowOfButtons
      Margin="10,10,40,30"
      HorizontalAlignment="Right"
      Grid.Row="5"
      Grid.ColumnSpan="6">

      <Button Content="A" Click="On_A_Clicked" />
      <Button Content="C" Click="On_C_Clicked" />
      <Button Content="D" Click="On_D_Clicked" />

 </c:RowOfButtons>
```

```
public class RowOfButtons : StackPanel
{
    public RowOfButtons()
    {
        Spacing = 20;
        Orientation = Orientation.Horizontal;
    }
}
```

But what about Grids?

But what about Performance?

So, get rid of Attached Properties?

```
public class RowOfButtons : StackPanel
{
    public RowOfButtons()
    {
        Spacing = 20;
        Orientation = Orientation.Horizontal;
    }
}
```

Let’s think about

Naming

```
public void DoStuff()...

public double Add20Percent(double price)...

public double AddTax(double price)...
```

```
Margin="{StaticResource MediumLeftRightMargin}"
```

```
public class ActionRedButton : Button {
```

Let’s think about

Magic values

```
price = price * 1.2;
```

It's a kitten

That's unexpected

Oh, look at the fluffy kitty.

```
<DataTemplate>
  <Border Margin="10" StrokeThickness="2" BackgroundColor="SeaGreen">
    <Grid Margin="10" ColumnDefinitions="Auto,*">

      <Image Source="{Binding Image}" />

      <VerticalStackLayout Spacing="4" Margin="8,8,8,8" Grid.Column="1">
        <Label Text="{Binding Title}" FontSize="30" FontAttributes="Bold" />
        <Label Text="{Binding Subtitle}" FontSize="24" />
        <Label Text="{Binding Description}" FontSize="20" Opacity="0.7" />
      </VerticalStackLayout>
    </Grid>
  </Border>
</DataTemplate>
```

```
<DataTemplate>
  <Border Margin="10"
          StrokeThickness="2"
          BackgroundColor="{StaticResource HighlightColor}">
    <Grid Margin="10" ColumnDefinitions="Auto,*">
      <Image Source="{Binding Image}" />

      <VerticalStackLayout Spacing="{StaticResource IntraItemSpacing}"
                           Margin="{StaticResource ItemMargin}"
                           Grid.Column="1">
        <Label Text="{Binding Title}" FontSize="30" FontAttributes="Bold" />
        <Label Text="{Binding Subtitle}" FontSize="24" />
        <Label Text="{Binding Description}" FontSize="20" Opacity="0.7" />
      </VerticalStackLayout>
    </Grid>
  </Border>
</DataTemplate>
```

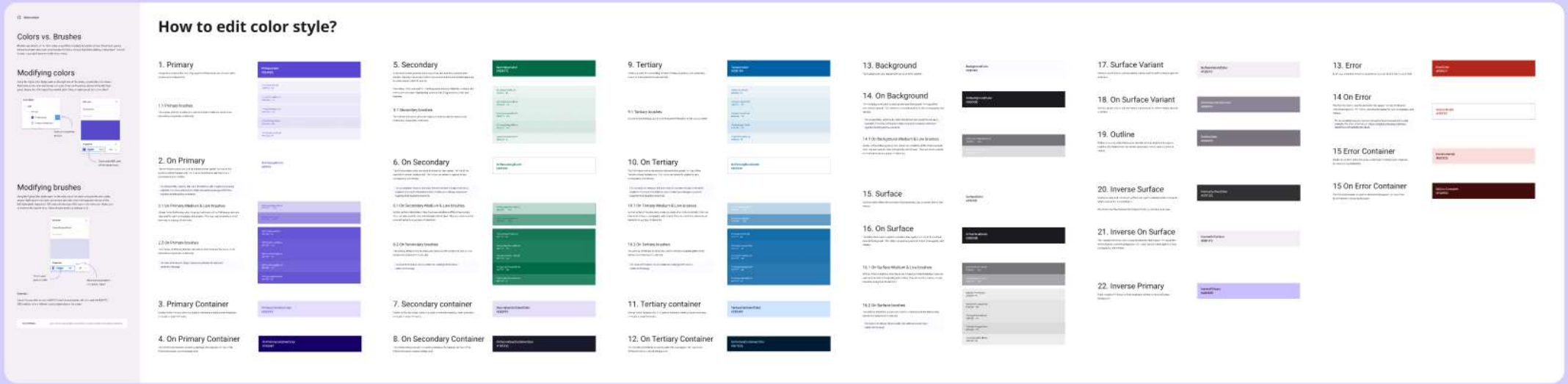
```
<x:Double x:Key="StandardItemSpacing">8</x:Double>

<Thickness x:Key="ItemMargin">
    {StaticResource StandardItemSpacing}
</Thickness>
```

```
<x:Double x:Key="StandardItemSpacing">8</x:Double>

<rxt:XamlThickness
    x:Key="ItemMargin"
    Bottom="{StaticResource StandardItemSpacing}"
    Left="{StaticResource StandardItemSpacing}"
    Right="{StaticResource StandardItemSpacing}"
    Top="{StaticResource StandardItemSpacing}" />
```

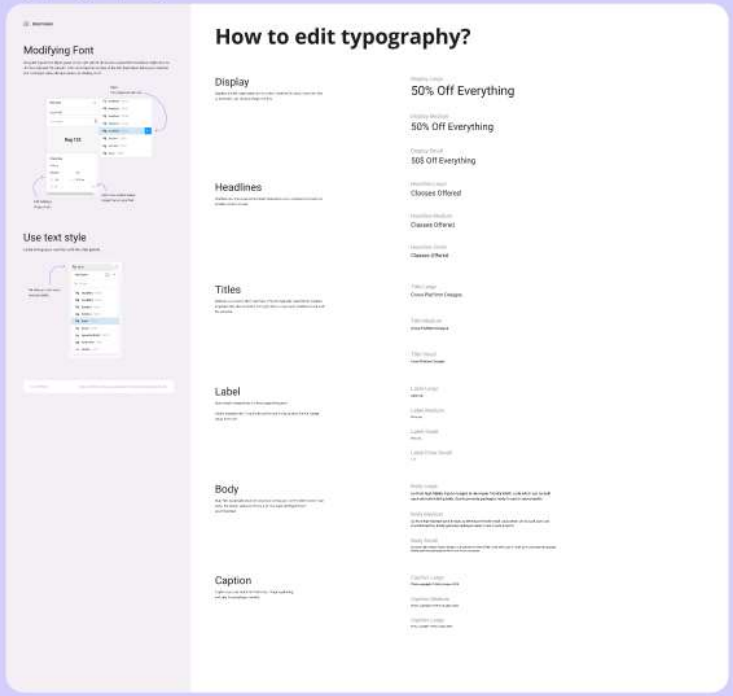
```
5 references
public class XamlThickness : DependencyObject
{
    2 references
    public double Left...
    2 references
    public double Top...
    2 references
    public double Right...
    2 references
    public double Bottom
    {
        get => (double)GetValue(BottomProperty);
        set => SetValue(BottomProperty, value);
    }

    public static readonly DependencyProperty LeftProperty =
        DependencyProperty.Register(nameof(Left), typeof(double), typeof(XamlThickness), new PropertyMetadata(0.0));
    public static readonly DependencyProperty TopProperty =
        DependencyProperty.Register(nameof(Top), typeof(double), typeof(XamlThickness), new PropertyMetadata(0.0));

    public static readonly DependencyProperty RightProperty =
        DependencyProperty.Register(nameof(Right), typeof(double), typeof(XamlThickness), new PropertyMetadata(0.0));

    public static readonly DependencyProperty BottomProperty =
        DependencyProperty.Register(nameof(Bottom), typeof(double), typeof(XamlThickness), new PropertyMetadata(0.0));

    public static implicit operator Thickness(XamlThickness xt)
    {
        return new Thickness(xt.Left, xt.Top, xt.Right, xt.Bottom);
    }
}
```

```
<DataTemplate>
  <Border Margin="10"
          StrokeThickness="2"
          BackgroundColor="{StaticResource HighlightColor}">
    <Grid Margin="10" ColumnDefinitions="Auto,*">
      <Image Source="{Binding Image}" />

      <VerticalStackLayout Spacing="{StaticResource IntraItemSpacing}"
                           Margin="{StaticResource ItemMargin}"
                           Grid.Column="1">
        <Label Text="{Binding Title}" FontSize="30" FontAttributes="Bold" />
        <Label Text="{Binding Subtitle}" FontSize="24" />
        <Label Text="{Binding Description}" FontSize="20" Opacity="0.7" />
      </VerticalStackLayout>
    </Grid>
  </Border>
</DataTemplate>
```

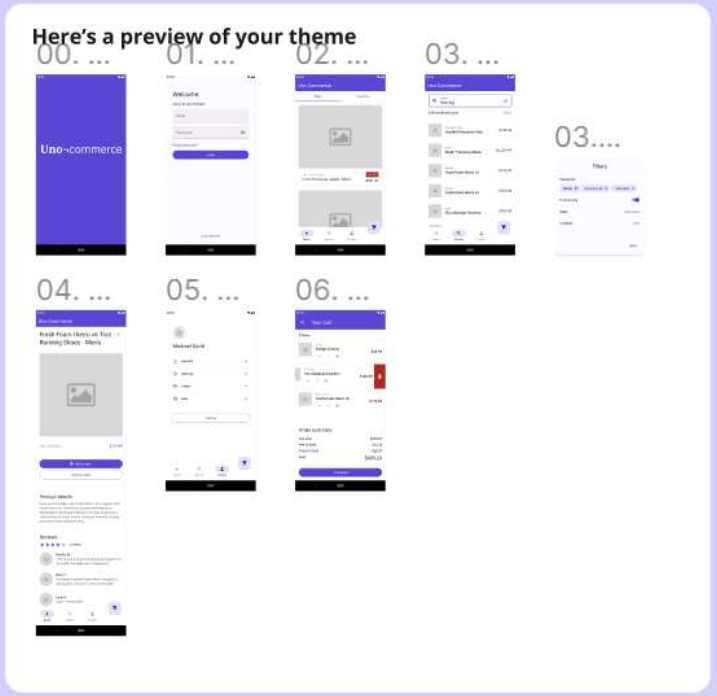
Edit color styles



Typography



Preview



```
<DataTemplate>
  <Border Margin="10"
          StrokeThickness="2"
          BackgroundColor="{StaticResource HighlightColor}">
    <Grid Margin="10" ColumnDefinitions="Auto,*">

      <Image Source="{Binding Image}" />

      <VerticalStackLayout Spacing="{StaticResource IntraItemSpacing}"
                           Margin="{StaticResource ItemMargin}"
                           Grid.Column="1">
        <Label Text="{Binding Title}" Style="{StaticResource TitleText}" />
        <Label Text="{Binding Subtitle}" Style="{StaticResource SubtitleText}" />
        <Label Text="{Binding Description}" Style="{StaticResource DefaultText}" />
      </VerticalStackLayout>
    </Grid>
  </Border>
</DataTemplate>
```

Let’s think about

Custom Types

```
class Person { }

class Student : Person { }

void DoSomethingWithAStudent(Person person)
{
    var student = (Student)person;

    // ....
}

void DoSomethingWithAStudent(Student student)
{
    // ....
}
```

```
<DataTemplate>
  <Border Margin="10"
          StrokeThickness="2"
          BackgroundColor="{StaticResource HighlightColor}">
    <Grid Margin="10" ColumnDefinitions="Auto,*">

      <Image Source="{Binding Image}" />

      <VerticalStackLayout Spacing="{StaticResource IntraItemSpacing}"
                           Margin="{StaticResource ItemMargin}"
                           Grid.Column="1">
        <Label Text="{Binding Title}" Style="{StaticResource TitleText}" />
        <Label Text="{Binding Subtitle}" Style="{StaticResource SubtitleText}" />
        <Label Text="{Binding Description}" Style="{StaticResource DefaultText}" />
      </VerticalStackLayout>
    </Grid>
  </Border>
</DataTemplate>
```

```
<DataTemplate>
  <c:StandardItemBorder>
    <Grid Margin="{StaticResource InnerMargin}" ColumnDefinitions="Auto,*">

      <Image Source="{Binding Image}" />

      <VerticalStackLayout Grid.Column="1"
                           Spacing="{StaticResource IntraItemSpacing}"
                           Margin="{StaticResource ItemMargin}">
        <c:Title Text="{Binding Title}" />
        <c:Subtitle Text="{Binding Subtitle}" />
        <c:DefaultText Text="{Binding Description}" />

      </VerticalStackLayout>
    </Grid>
  </c:StandardItemBorder>
</DataTemplate>
```

```
internal class StandardItemBorder : Border
{
    public StandardItemBorder()
    {
        Style = (Style)App.Current.Resources["StandardItemBorder"];
    }
}


internal class Title : Label
{
    public Title()
    {
        Style = (Style)App.Current.Resources["TitleText"];
    }
}
```

But I don't want to have to create all those types myself!

CommonStyles.xaml

ResourceDictionary | ResourceDictionary

```xaml
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/present
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">


    <Color x:Key="AppAccent">#236821</Color>
    <Color x:Key="SecondaryCOlor">#87ff56</Color>

    <SolidColorBrush x:Key="AppAccentBrush"
                     Color="{StaticResource AppAccent}" />


    <Style TargetType="Button" x:Key="StandardButton">
        <Setter Property="Background" Value="Azure" />
        <Setter Property="FontSize" Value="40"/>
    </Style>


    <Style TargetType="Button" x:Key="SuperButton">
        <Setter Property="Background" Value="Green" />
        <Setter Property="FontSize" Value="24"/>
    </Style>

    <Style TargetType="Button" x:Key="ErrorButton">
        <Setter Property="Background" Value="Red" />
        <Setter Property="FontSize" Value="40"/>
        <Setter Property="FontWeight" Value="Bold"/>
    </Style>

    <Style TargetType="Button" >
        <Setter Property="Background" Value="Blue" />
    </Style>


</ResourceDictionary>

```

CommonStyles.cs

App382 | App382.AppColors | AppAccent

```cs
/// <auto-generated> ...
using ...

namespace App382
{
    0 references | 0 changes | 0 authors, 0 changes
    public partial class AppColors
    {
        public const string AppAccent = "AppAccent";
        public const string SecondaryCOlor = "SecondaryCOlor";
    }

    0 references | 0 changes | 0 authors, 0 changes
    public partial class AppBrushes
    {
        public const string AppAccentBrush = "AppAccentBrush";
    }

    1 reference | 0 changes | 0 authors, 0 changes
    public class StandardButton : Button
    {
        0 references | 0 changes | 0 authors, 0 changes
        public StandardButton()
        {
            this.Style = (Style)App.Current.Resources["StandardButton"];
        }
    }
    1 reference | 0 changes | 0 authors, 0 changes
    public class SuperButton : Button
    {
        0 references | 0 changes | 0 authors, 0 changes
        public SuperButton()
        {
            this.Style = (Style)App.Current.Resources["SuperButton"];
        }
    }
    1 reference | 0 changes | 0 authors, 0 changes
    public class ErrorButton : Button
    {
        0 references | 0 changes | 0 authors, 0 changes
        public ErrorButton()
        {
            this.Style = (Style)App.Current.Resources["ErrorButton"];
        }
    }
}
```

CommonStyles.xaml

ResourceDictionary

```
41
42      <Style TargetType="Button" x:Key="ErrorButton">
43          <Setter Property="Background" Value="Red" />
44          <Setter Property="FontSize" Value="40"/>
45          <Setter Property="FontWeight" Value="Bold"/>
46      </Style>
47
```

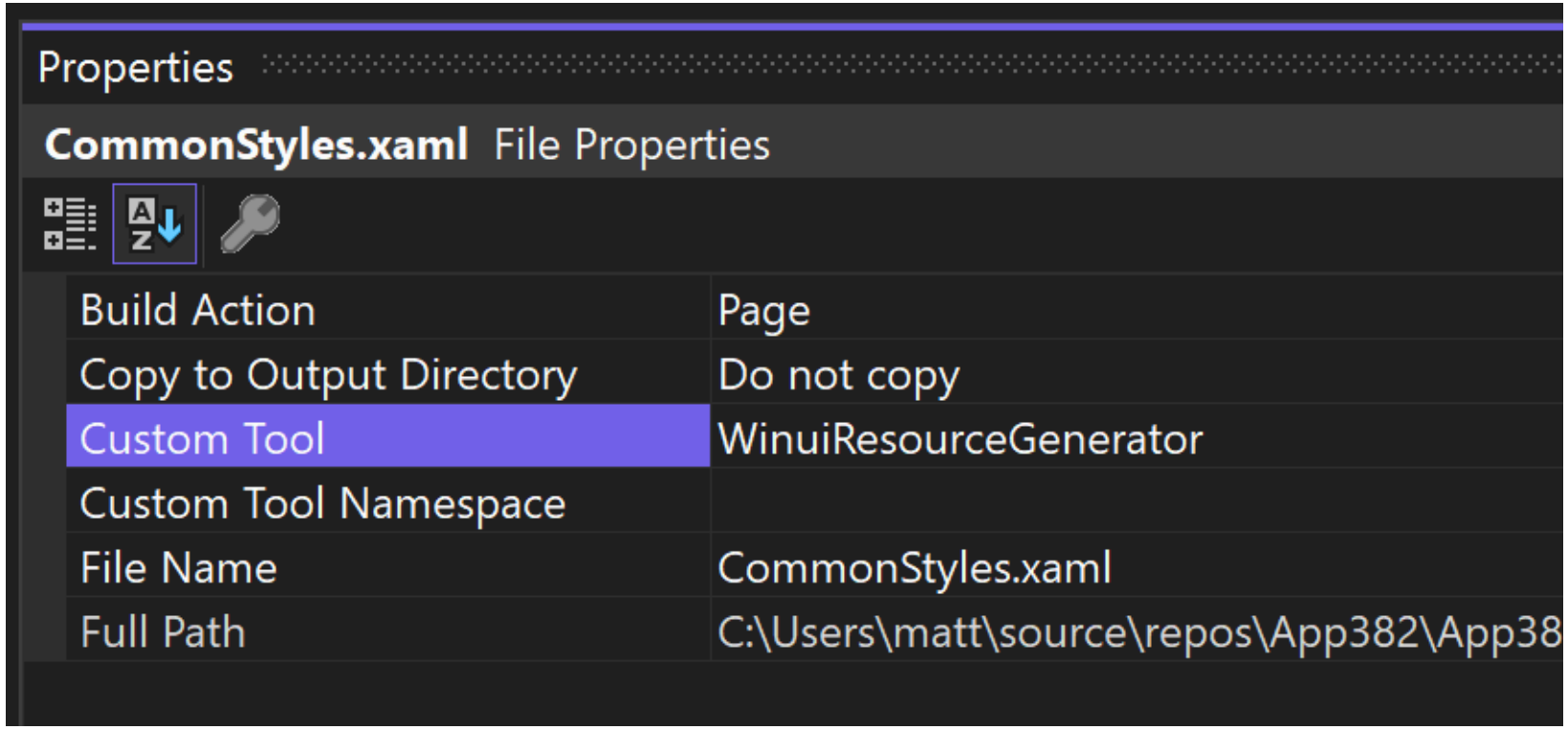
200 % 4 0 0 changes | 0 authors, 0 changes Ln: 1

XAML Design

CommonStyles.cs

App382 App382.GenStyles.SuperButton SuperButton()

```
44      public class ErrorButton : Button
45      {
            0 references | 0 changes | 0 authors, 0 changes
46          public ErrorButton()
47          {
48              this.Style = (Style)App.Current.Resources["ErrorButton"];
49          }
50      }
```


Properties
CommonStyles.xaml File Properties
Build Action
Page
Copy to Output Directory
Do not copy
Custom Tool
WinuiResourceGenerator
Custom Tool Namespace
File Name
CommonStyles.xaml
Full Path
C:\Users\matt\source\repos\App382\App38

But what about the sealed types in WinUI3?

```xaml
<Style TargetType="TextBlock" x:Key="MyCoolText">
    <Setter Property="FontSize" Value="24" />
    <Setter Property="Foreground" Value="Green" />
    <Setter Property="FontWeight" Value="ExtraBlack" />
</Style>
```

```
7 references
public class MyCoolText : RxTextBlock
{
    1 reference
    public MyCoolText()
    {
        this.SetStyle(App.Current.Resources["MyCoolText"] as Style);
    }
}
```

But what about referencing MergedDictionaries in a MAUI app?

```xaml
<Style TargetType="Label" x:Key="PageTitle">
    <Setter Property="FontSize" Value="30" />
</Style>
```

```csharp
public class PageTitle : Label
{
    public PageTitle()
    {
        const string resKey = "PageTitle";
        Style style = null;

        if (App.Current.Resources.ContainsKey(resKey))
        {
            style = App.Current.Resources[resKey] as Style;
        }
        else
        {
            foreach (var mergeDict in App.Current.Resources.MergedDictionaries)
            {
                if (mergeDict.ContainsKey(resKey))
                {
                    style = mergeDict[resKey] as Style;
                    break;
                }
            }
        }

        if (style != null)
        {
            this.Style = style;
        }
    }
}
```

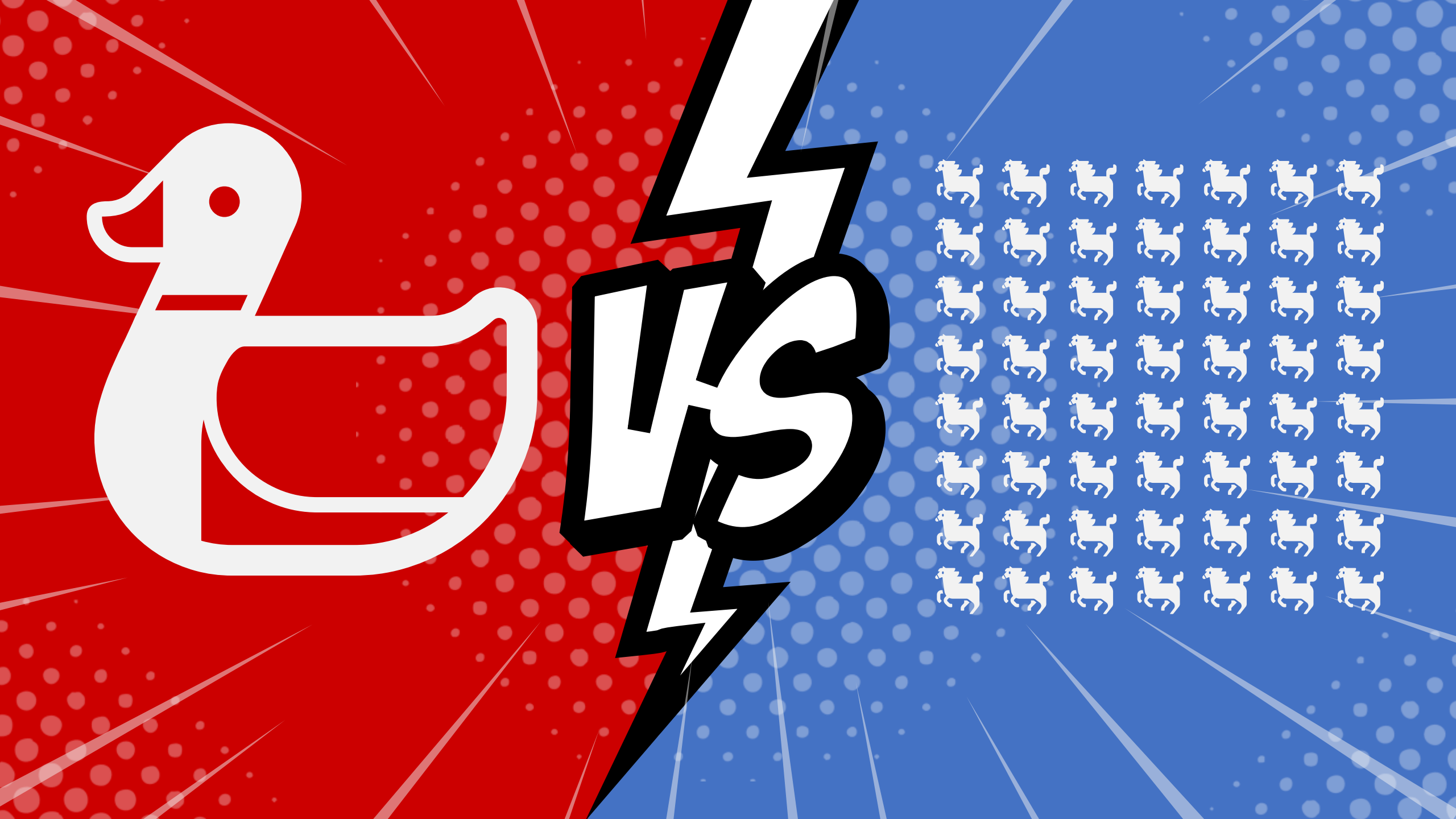
```
1 reference
public class GridForPageWithNavBar : Grid
{
    0 references
    public GridForPageWithNavBar()
    {
        this.Margin = 0;
        this.RowSpacing = 0;

        this.RowDefinitions = new RowDefinitionCollection {
            new RowDefinition {
                Height = new GridLength(
                    DeviceInfo.Platform == DevicePlatform.Android
                                        ? 76
                                        : 92,
                    GridUnitType.Absolute) },
            new RowDefinition { Height = new GridLength(1, GridUnitType.Star) },
            };
    }
}
```

```
<ContentPage ... >
    <c:GridForPageWithNavBar>

        <ScrollView Grid.Row="1" ... >
            ...
        </ScrollView >
        <c:TopNavBar>
            <c:TopBarIconContainer>
                <c:TopBarIcon ... />
                <c:TopBarIcon ... />
                <c:TopBarIcon ... />
            </c: TopBarIconContainer>
        </c:TopNavBar>

    </c:GridForPageWithNavBar>
</ContentPage>
```

But what about managing all those types?

Let’s put it

all together


VS

Android Emulator - pixel_5_-_api_30:5554

4:33

Monkey Finder

Baboon
Africa & Asia

Capuchin Monkey
Central & South America

Blue Monkey
Central and East Africa

Squirrel Monkey
Central & South America

Golden Lion Tamarin
Brazil

Get Monkeys

Find Closest

Android Emulator - pixel_5_-_api_30:5554

4:34

Capuchin Monkey

Show on Map

The capuchin monkeys are New World monkeys of the subfamily Cebinae. Prior to 2011, the subfamily contained only a single genus, Cebus.

Location: Central & South America

Population: 23000

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MonkeyFinder.DetailsPage"
    xmlns:viewmodel="clr-namespace:MonkeyFinder.ViewModel"
    x:DataType="viewmodel:MonkeyDetailsViewModel"
    Title="{Binding Monkey.Name}">

    <ScrollView>
        <VerticalStackLayout>
            <Grid ColumnDefinitions="*,Auto,*" RowDefinitions="160, Auto">

                <!-- Background and Image of Monkey -->
                <BoxView
    Grid.ColumnSpan="3"
    Background="{StaticResource Primary}"
    HeightRequest="160"
    HorizontalOptions="FillAndExpand" />
                <Frame
    Grid.RowSpan="2"
    Grid.Column="1"
    Margin="0,80,0,0"
    HeightRequest="160"
    WidthRequest="160"
    HorizontalOptions="Center"
    Padding="0"
    IsClippedToBounds="True"
    CornerRadius="80">
                    <Image
        Aspect="AspectFill"
        HeightRequest="160"
        HorizontalOptions="Center"
        VerticalOptions="Center"
        Source="{Binding Monkey.Image}"
        WidthRequest="160"/>
                </Frame>
            </Grid>
```

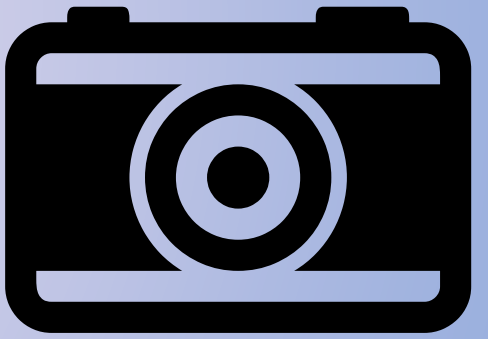
```
<ContentPage
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MonkeyFinder.DetailsPage“ xmlns:viewmodel="clr-namespace:MonkeyFinder.ViewModel"
    x:DataType="viewmodel:MonkeyDetailsViewModel" Title="{Binding Monkey.Name}">
    <ScrollView>
        <VerticalStackLayout>
            <Grid ColumnDefinitions="*,Auto,*" RowDefinitions="160, Auto">
                <!-- Background and Image of Monkey -->
                <BoxView Grid.ColumnSpan="3" Background="{StaticResource Primary}"
                         HeightRequest="160" HorizontalOptions="FillAndExpand" />
                <Frame Grid.RowSpan="2" Grid.Column="1" Margin="0,80,0,0" HeightRequest="160"
                       WidthRequest="160" HorizontalOptions="Center" Padding="0"
                       IsClippedToBounds="True" CornerRadius="80">
                    <Image Aspect="AspectFill" HeightRequest="160" HorizontalOptions="Center"
                           VerticalOptions="Center" Source="{Binding Monkey.Image}" WidthRequest="160"/>
                </Frame>
            </Grid>
            <!-- Details of Monkey -->
            <VerticalStackLayout Padding="10" Spacing="10">
                <Button Text="Show on Map" Command="{Binding OpenMapCommand}"
                        HorizontalOptions="Center" WidthRequest="200" Margin="8"
                        Style="{StaticResource ButtonOutline}"/>
                <Label Style="{StaticResource MediumLabel}" Text="{Binding Monkey.Details}" />
                <Label Style="{StaticResource MicroLabel}" Text="{Binding Monkey.Location, StringFormat='Location: {0}'}" />
                <Label Style="{StaticResource MicroLabel}" Text="{Binding Monkey.Population, StringFormat='Population: {0}'}" />
            </VerticalStackLayout>
        </VerticalStackLayout>
    </ScrollView>
</ContentPage>
```

```
<res:VertScrolllingPage
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:viewmodel="clr-namespace:MonkeyFinder.ViewModel"
    xmlns:res="clr-namespace:MonkeyFinder.Resources"
    x:Class="MonkeyFinder.Details2Page"
    x:DataType="viewmodel:MonkeyDetailsViewModel"
    Title="{Binding Monkey.Name}">

    <res:HeaderImage BgColor="{StaticResource Primary}"
                     Source="{Binding Monkey.Image}"  />

    <res:PageDetails>
        <res:TextButton Text="Show on Map" Command="{Binding OpenMapCommand}" />
        <res:DefaultText Text="{Binding Monkey.Details}" />
        <res:SmallText Text="{Binding Monkey.Location, StringFormat='Location: {0}'}" />
        <res:SmallText Text="{Binding Monkey.Population, StringFormat='Population: {0}'}" />
    </res:PageDetails>

</res:VertScrolllingPage>
```

But I don't want to have to rewrite all my XAML code?

You don’t need to do everything all at once!

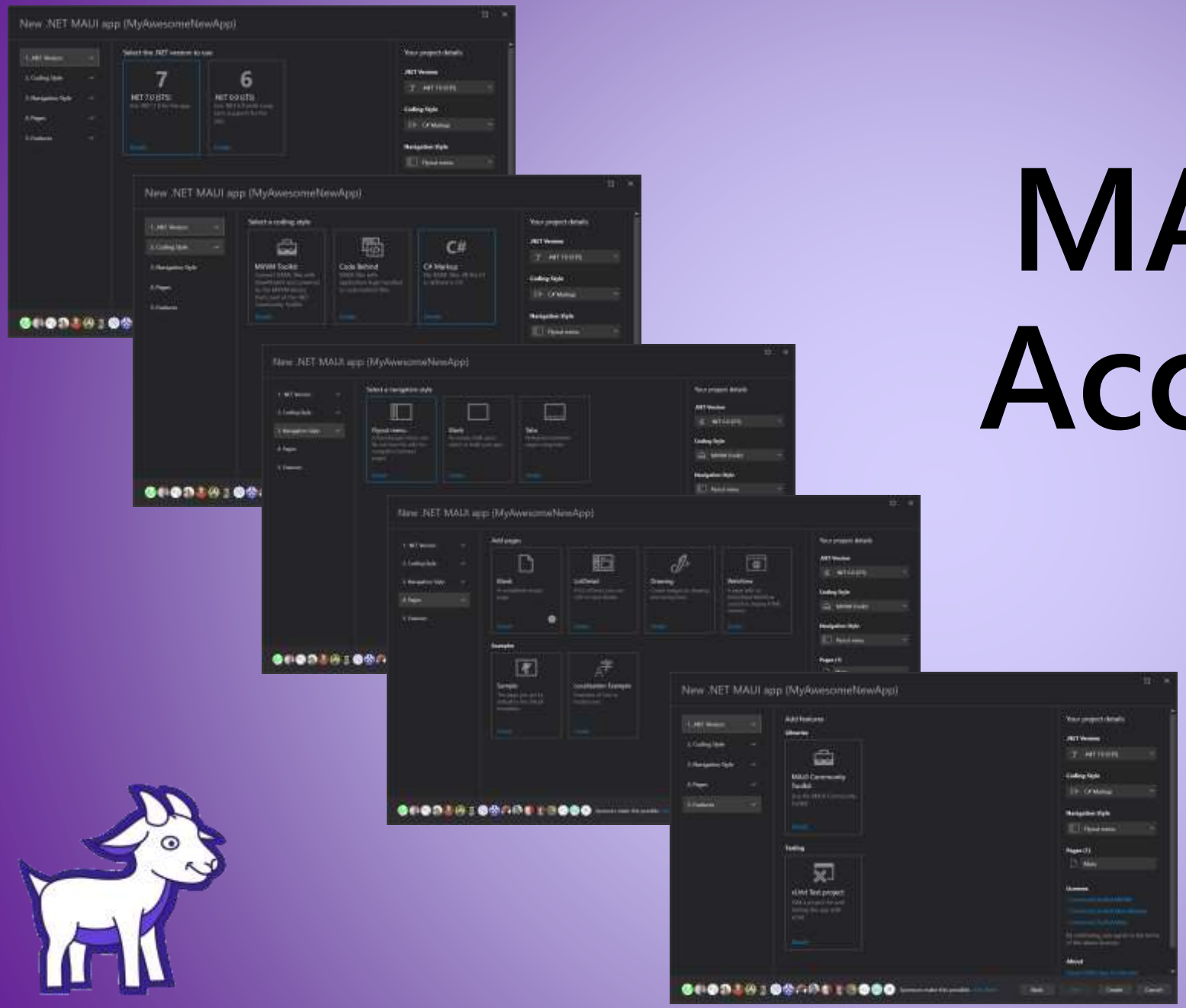
- Format consistently – have coding standards
- Single responsibilities – looks vs layout/positioning
- Naming matters
- No magic values (or numbers)
- Avoid unnecessary duplication
- Avoid unnecessary duplication
- Don’t fear creating custom types



- Don’t treat it too differently from C#

@MRLacey

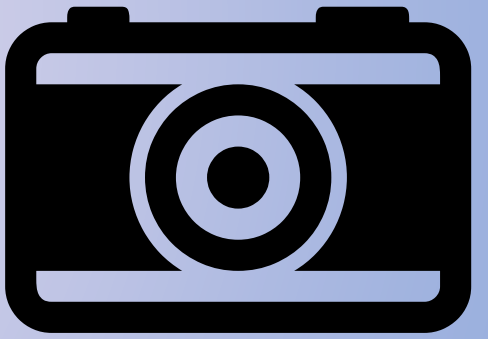
Available from Amazon
or
ask me for the PDF



MAUI App Accelerator

You don’t need to do everything all at once!

- Format consistently
- Single responsibilities – looks vs layout/positioning
- Naming matters
- No magic values (or numbers)
- Avoid unnecessary duplication
- Avoid unnecessary duplication
- Don’t fear creating custom types



- Don’t treat it too differently from C#

@MRLacey