
Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
AUSTIN
Snorkeling in MAUI
Sam Basu | @samidip
Developer Advocate | Progress
#VSLIVE
NO CODE LIMITS

Building for Mobile

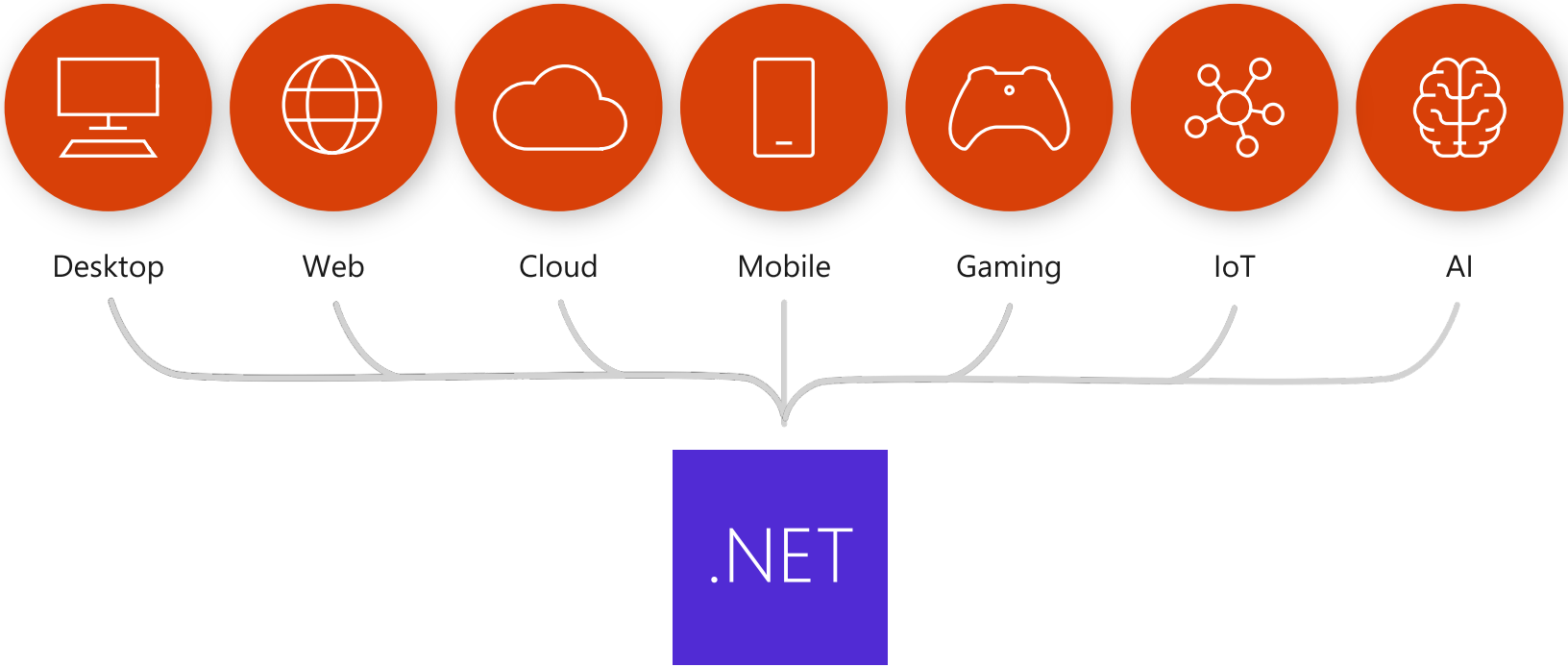
Mobile Your Way

It’s 2022 | Developers have choices

1	Mobile Web / PWA
2	Native
3	Hybrid
4	JS Native
5	X-Compiled

The .NET Stack

Your platform for building anything



.NET – A unified development platform

ECOSYSTEM

NuGet

GitHub

Components, tools, library vendors

TOOLS

Visual Studio

Visual Studio for Mac

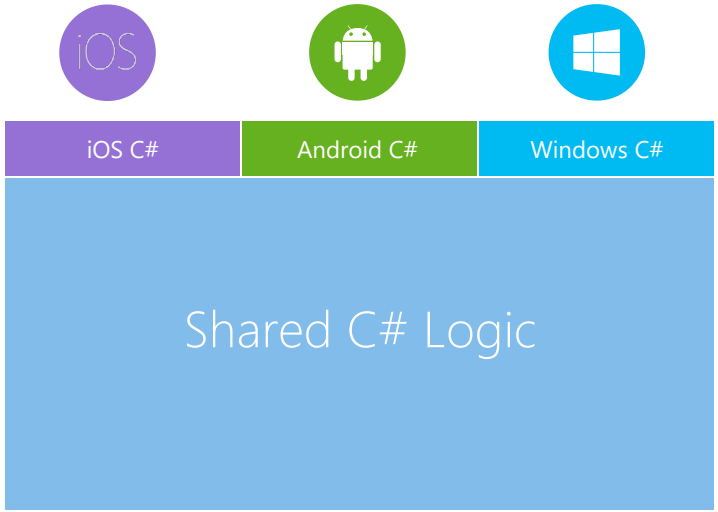
Visual Studio Code

CLI

Xamarin Democratizes .NET Mobile Development

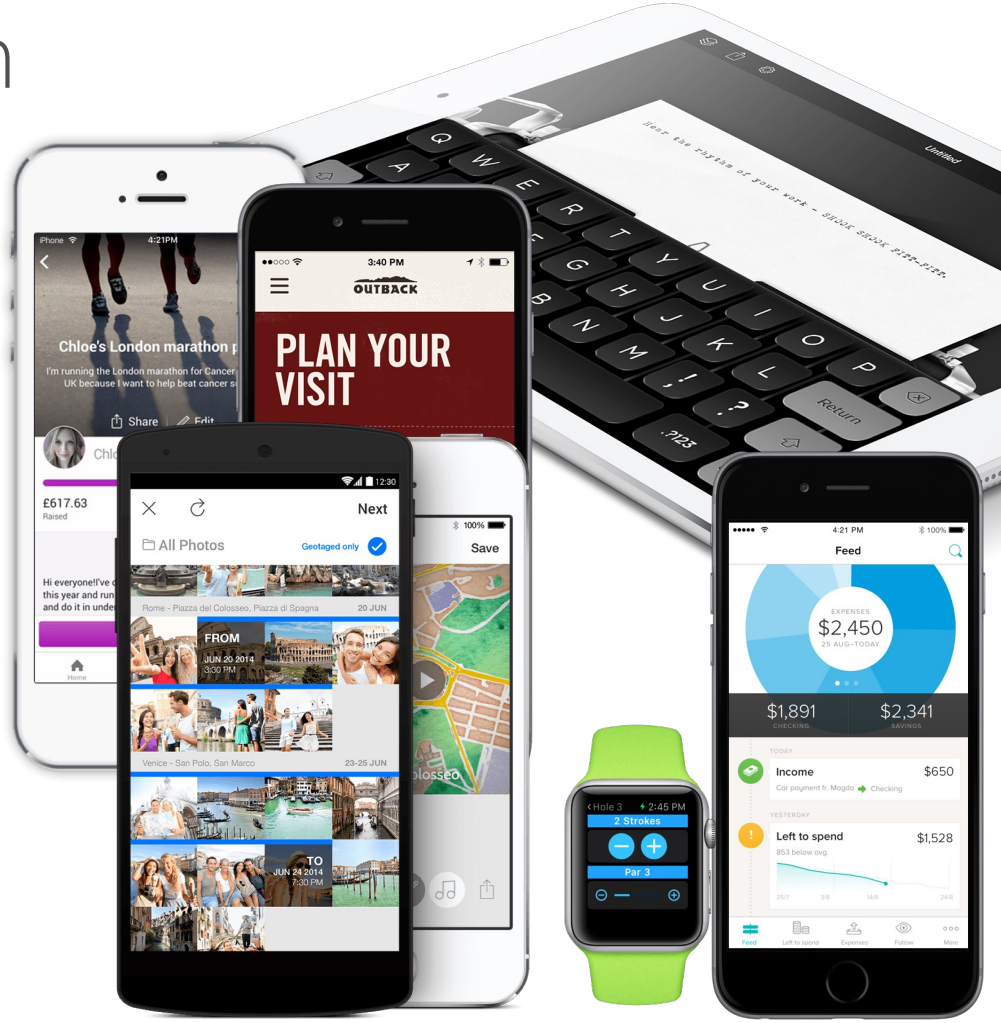
Truly Native | Truly Cross-Platform

Traditional UI approach



3 Native User Interfaces

Shared App Logic



Xamarin.Forms approach

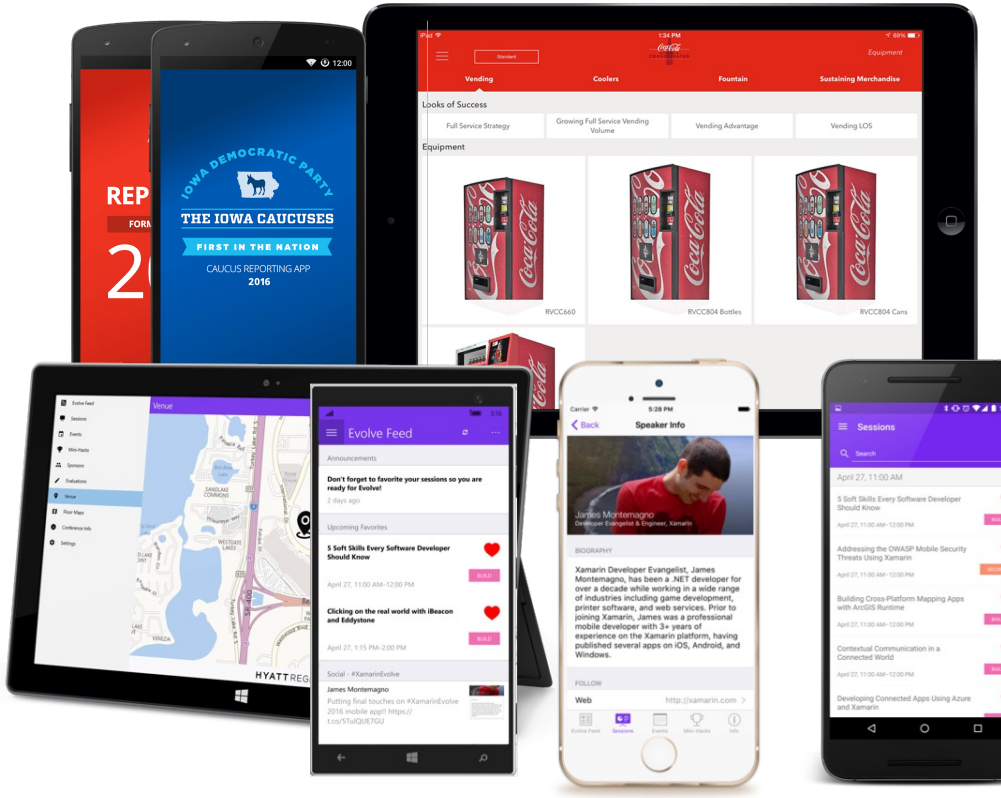
iOS

Shared UI Code

Shared C# Logic

Shared User Interface

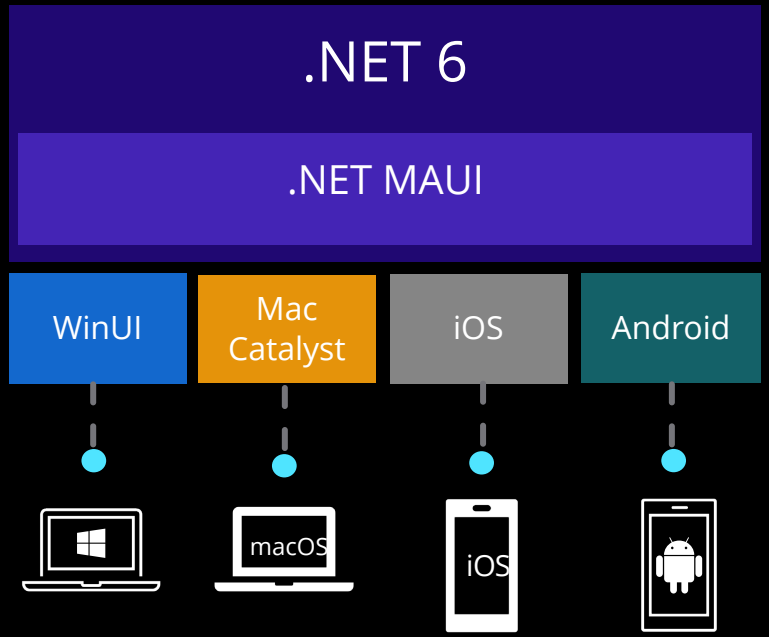
Shared App Logic



So Why Evolve?


.NET
MAUI

.NET MAUI



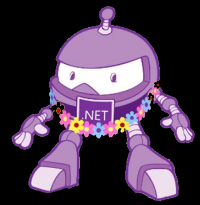
Cross-platform, native UI

Single project system, single codebase

Cross-platform, device API access

Native compilation, native performance

Deploy to multiple devices, mobile & desktop



github.com/dotnet/maui

Productive environment

Maximize code reuse

Native performance & integrations

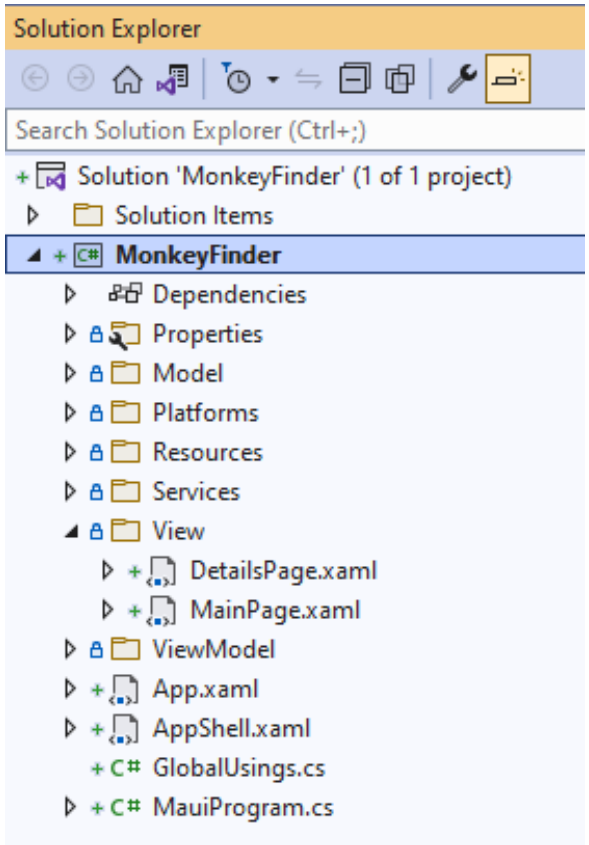
Expansive ecosystem

Cross-Platform APIs

Sms
Secure Settings
Preferences
Flashlight
Clipboard
Gyroscope
Magnetometer
Geolocation
Phone Dialer
Device Display Info
Email
Device Info
Battery
Compass
Accelerometer
Screen Lock
Text to Speech
Data Transfer
Geocoding
Vibration
File System
Connectivity

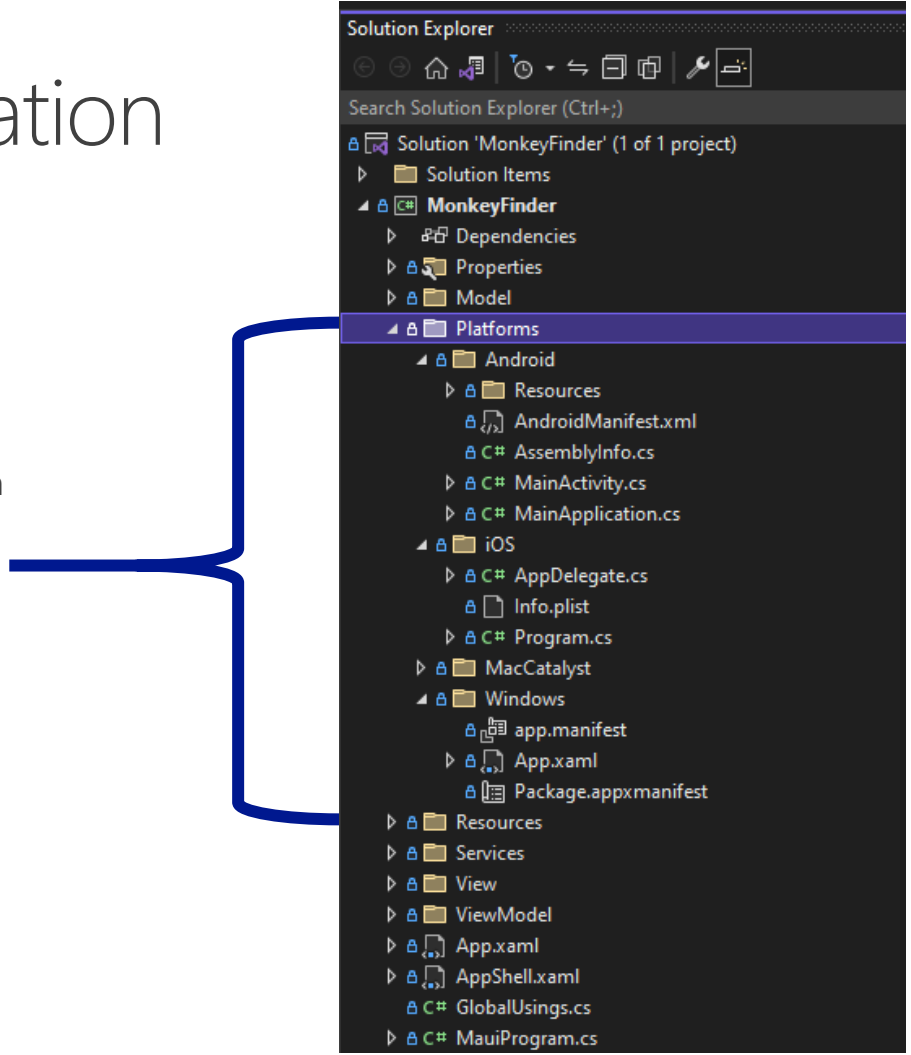
Project Structure

One Single Project that hosts all of the source code for the project including UI, platform code, shared resources and more.



Platform Integration

Platform Specific folders hold platform specific resources, configuration, startup logic, and platform code.



Multi-Targeted

```
<TargetFrameworks>
net6.0-android;net6.0-ios;net6.0-maccatalyst;net6.0-windows10.0.19041
</TargetFrameworks>

<!-- Display name -->
<ApplicationTitle>MonkeyFinder</ApplicationTitle>

App Identifier -->
<ApplicationId>com.companyname.monkeyfinder</ApplicationId>

<!-- Versions -->
<ApplicationVersion>1</ApplicationVersion>
```

Conditionally Compile

```
#if ANDROID
        var button = new Android.Widget.Button(context);
#elif IOS
        var button = new UIKit.UIButton();
#endif
```

.NET MAUI Startup

```
public static MauiApp CreateMauiApp()
{
    var builder = MauiApp.CreateBuilder();
    builder
        .UseMauiApp<App>()
        .ConfigureFonts(fonts =>
        {
            fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
        });


    return builder.Build();
}
```

.NET MAUI Startup

```
public static MauiApp CreateMauiApp()
{
    var builder = MauiApp.CreateBuilder();
    builder
        .UseMauiApp<App>()
        .ConfigureFonts(fonts =>
        {
            fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
        });

    builder.Services.AddSingleton<MonkeyService>();
    builder.Services.AddSingleton<MonkeysViewModel>();
    builder.Services.AddSingleton<MainPage>();

    return builder.Build();
}
```

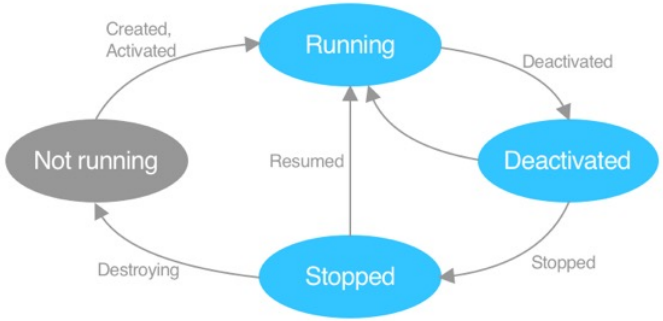
.NET MAUI Application

- Application lifecycle:
  - **OnStart**
  - **OnSleep**
  - **OnResume**

```
public partial class App : Application
{
    0 references
    public App()
    {
        InitializeComponent();

        MainPage = new AppShell();
    }

    0 references
    protected override void OnStart()
    {
        base.OnStart();
    }

    0 references
    protected override void OnSleep()
    {
        base.OnSleep();
    }

    0 references
    protected override void OnResume()
    {
        base.OnResume();
    }
}
```

.NET MAUI Application Lifecycle

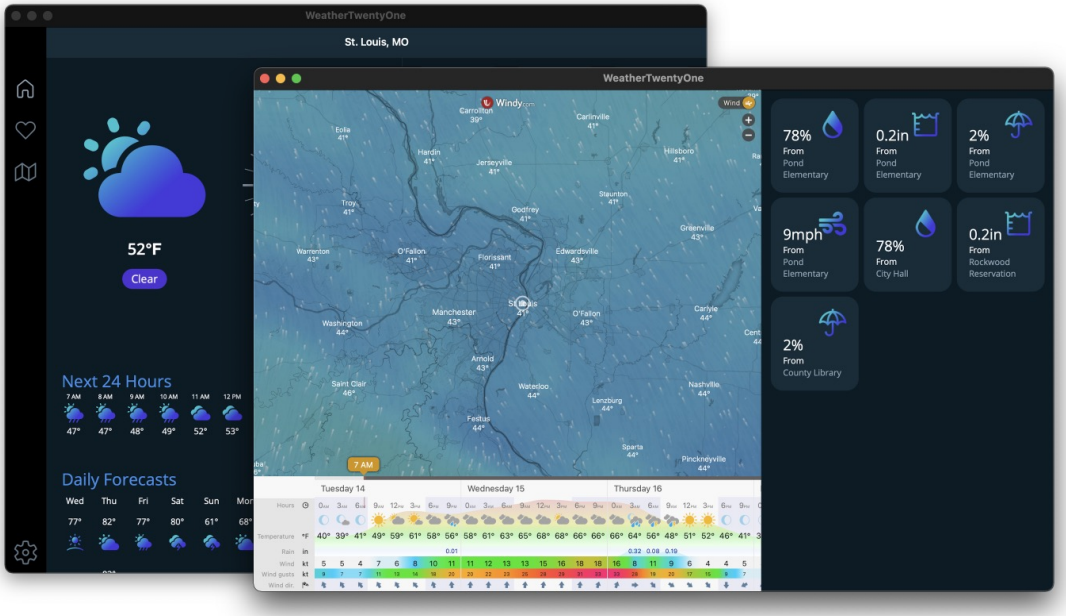
- Windows lifecycle:
  - **Created**
  - **Activated**
  - **Deactivated**
  - **Stopped**
  - **Resumed**
  - **Destroying**




```
protected override Window CreateWindow(IActivationState activationState)
{
    Window window = base.CreateWindow(activationState);

    window.Created += (s, e) =>
    {
        // Custom logic
    };

    window.Activated += (s, e) =>
    {
        // Custom logic
    };

    return window;
}
```

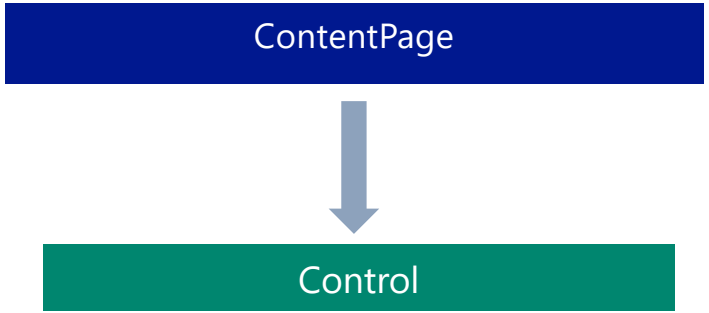
Multi-window Support

```
var secondWindow = new Window { Page = new MySecondPage { // ... } };

Application.Current.OpenWindow(secondWindow);
```



App Pages

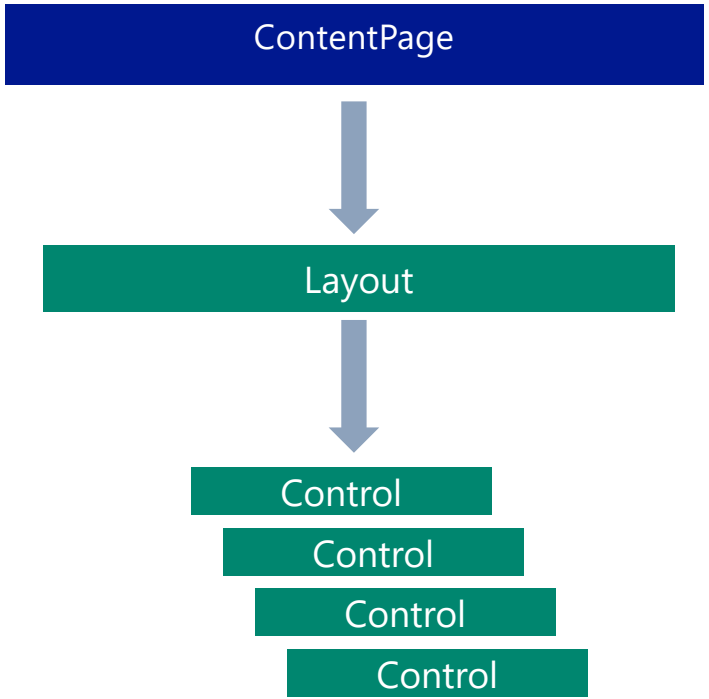
- Single screen of content
- ContentPage holds one visual element



App Layout

- Layouts handle child elements

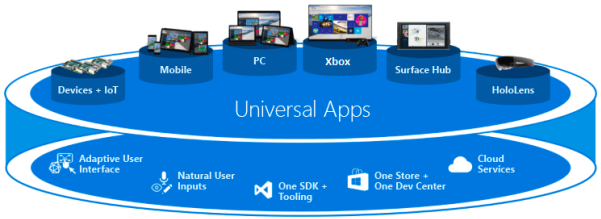
- Layouts include:
  - Grid
  - StackLayout
  - RelativeLayout
  - FlexLayout
  - & more



XAML/C# Ecosystem

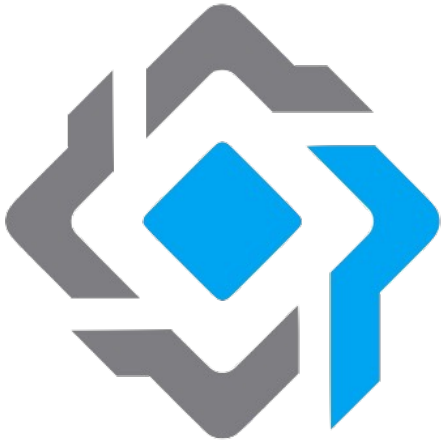
- ✓ Simple but verbose markup
- ✓ Extensibility
- ✓ Solid Tooling
- ✓ Rich Ecosystem

Windows Presentation Foundation



One Windows Platform

MVVM Design Pattern



Model-View-ViewModel

How to display information

What to display
Flow of interaction

Business Logic
Data objects

Data Binding Essentials

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

Data Binding - Properties

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

```
private string firstname = string.Empty;
public string FirstName
{
  get { return firstname; }
  set {
    if (firstname == value)
      return;

    firstname = value;
    OnPropertyChanged ("FirstName");
  }
}
```

Data Binding – XAML

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

```
private string firstname = string.Empty;
public string FirstName
{
  get { return firstname; }
  set {
    if (firstname == value)
      return;

    firstname = value;
    OnPropertyChanged ("FirstName");
  }
}
```

```
<Label Text="{Binding FirstName}"/>

<Entry Placeholder="FirstName"
       Text="{Binding FirstName, Mode=TwoWay}"/>
```

Data Binding – C#

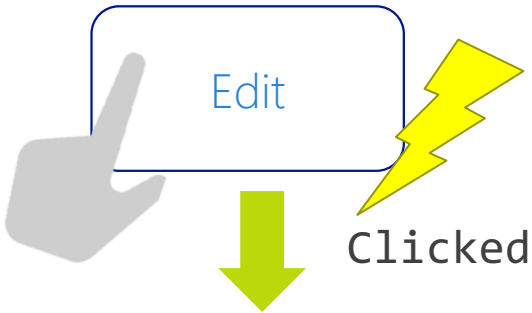
```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

```
private string firstname = string.Empty;
public string FirstName
{
  get { return firstname; }
  set {
    if (firstname == value)
      return;

    firstname = value;
    OnPropertyChanged ("FirstName");
  }
}
```

```
Label firstName = new Label ();
firstName.SetBinding (Label.TextProperty, "FirstName");
```

Event Handling

- UI raises events to notify code about user activity
  - `Clicked`
  - `ItemSelected`
  - …
- The downside is that these events **must be handled** in the code behind file



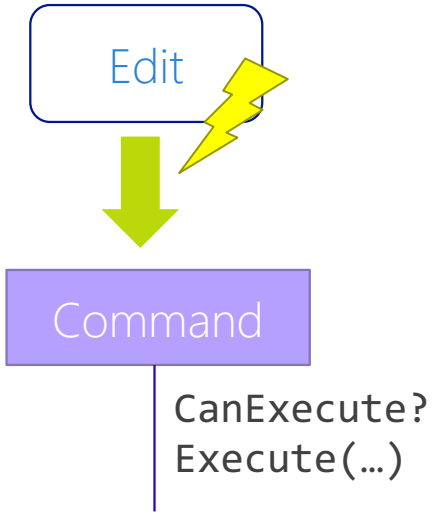

```
public MainPage()
{
    ...
    Button editButton = ...;
    editButton.Clicked += OnClick;
}

void OnClick (object sender, EventArgs e)
{
     ...
}
```

Commands

- Microsoft defined the **`ICommand`** interface to provide a commanding abstraction for their XAML frameworks

```
public interface ICommand
{
    event EventHandler CanExecuteChanged;
    bool CanExecute(object parameter);
    void Execute(object parameter);
}
```

Can provide an optional parameter (often **`null`**) for the command to work with for context



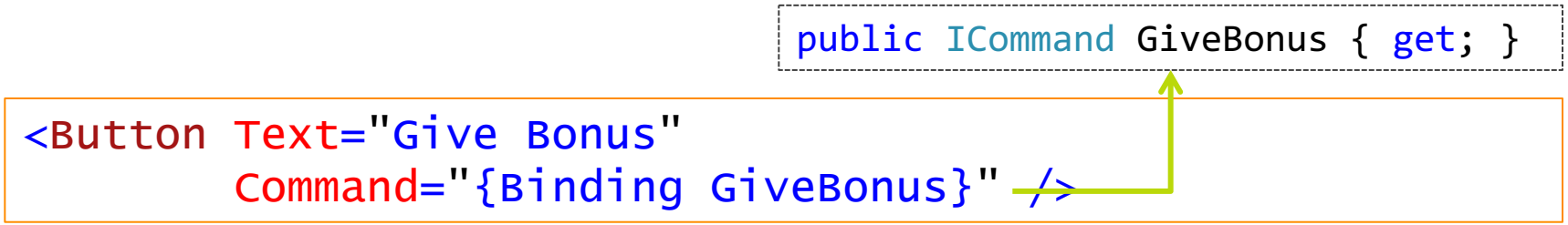
Commands in .NET MAUI

.NET MAUI controls expose a **`Command`** property for the main action of a control

`Menu` `Button` `ToolbarItem` `TextCell`

Commands in .NET MAUI

.NET MAUI controls expose a **Command** property for the main action of a control



Compiled Bindings

.NET MAUI controls expose a **`x:DataType`** property that enables compile time checks and optimizations.

```
<ContentPage x:DataType="viewmodel:EmployeeViewModel">

    <Button Text="Give Bonus"
            Command="{Binding GiveBonus}" />

</ContentPage  >
```

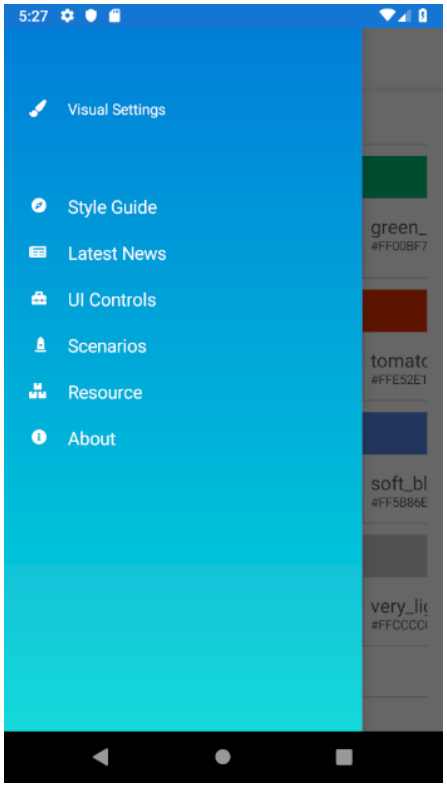
Navigation Basics

- Root Page:
  - NavigationPage – Gives each page an INavigation

- Standard Navigation
  - Navigation.PushAsync(page: nextPage);
  - Navigation.PopAsync();

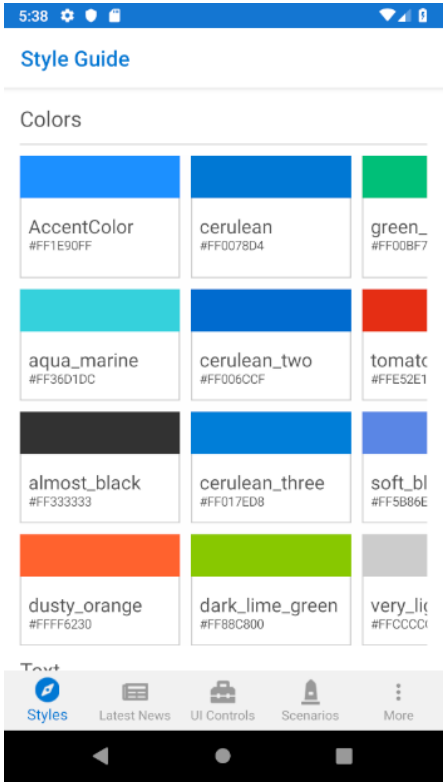
- Modal Navigation
  - Navigation.PushModalAsync(page: modalPage);
  - Navigation.PopModalAsync();

The Shell




```
<Shell>
    <FlyoutItem
        Title="Style Guide"
        Icon="Compass.png">
        <ShellContent
            ContentTemplate="{DataTemplate p:StyleGuidePage}"/>
    </FlyoutItem>
</Shell>
```

The Shell




```
<Shell>
    <TabBar>
        <Tab Title="Styles" Icon="Compass.png">
            <ShellContent
                ContentTemplate="{DataTemplate p:StyleGuidePage}"/>
        </Tab>
    </TabBar>
</Shell>
```

.NET MAUI Page Lifecycle

Legacy Page lifecycle events:

- OnAppearing
- OnDisappearing

Modern Navigation lifecycle events:

- OnNavigatedTo
- OnNavigatingFrom
- OnNavigatedFrom

```
0 references
protected override void OnAppearing()
{
    base.OnAppearing();
}

0 references
protected override void OnDisappearing()
{
    base.OnDisappearing();
}

0 references
protected override void OnNavigatedFrom(NavigatedFromEventArgs args)
{
    base.OnNavigatedFrom(args);
}

0 references
protected override void OnNavigatedTo(NavigatedToEventArgs args)
{
    base.OnNavigatedTo(args);
}

0 references
protected override void OnNavigatingFrom(NavigatingFromEventArgs args)
{
    base.OnNavigatingFrom(args);
}
```

App Themes – Light or Dark

The `AppThemeBinding` markup extension automatically adjusts the value based on the detected theme of the device

```
<ResourceDictionary>
  <Color x:Key="bgLight">White</Color>
  <Color x:Key="bgDark">Black</Color>
</ResourceDictionary>

<Button BackgroundColor="{AppThemeBinding
      Light={StaticResource bgLight},
      Dark={StaticResource bgDark}}"/>
  ...
```

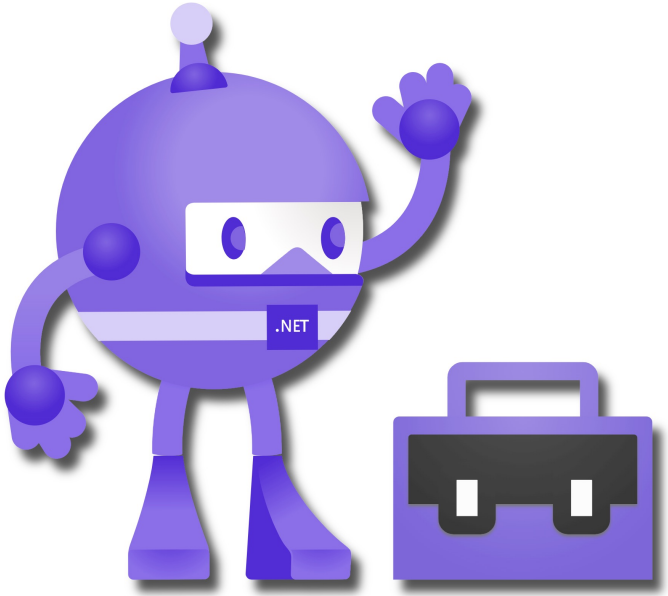
Using AppThemeBinding & Styles

You can also leverage Styles with AppThemeBinding

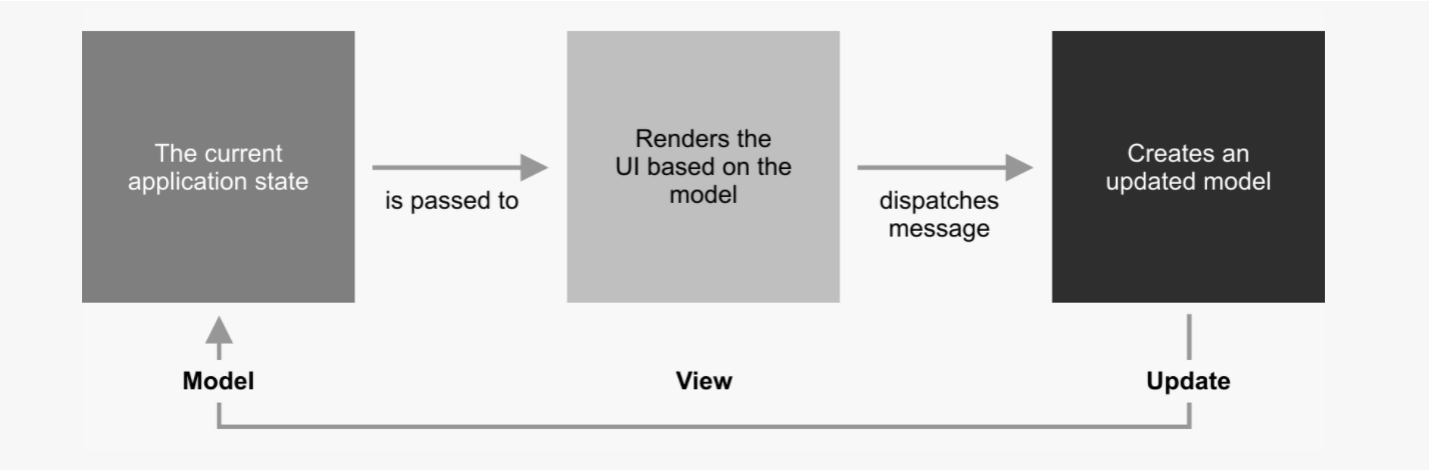
```
<ResourceDictionary>
  <Color x:Key="bgLight">White</Color>
  <Color x:Key="bgDark">Black</Color>

  <Style x:Key="MyButtonStyle" TargetType="Button">
      <Setter Property="BackgroundColor"
              Value="{AppThemeBinding Light={StaticResource bgLight},
                                      Dark={StaticResource bgDark}}"/>
  </Style>
</ResourceDictionary>

<Button Style="{StaticResource MyButtonStyle}">
```

.NET MAUI Community Toolkits

- ✓ CommunityToolkit.Maui
- ✓ CommunityToolkit.Maui.Markup



MVU Design Pattern



Comet

dev-build passing Clancey.Comet on fuget.org Chat on Discord

What is Comet? Comet is a modern way of writing cross-platform UIs. Based on .NET MAUI, it follows the Model View Update (MVU) pattern and magically databinds for you!

Building for Desktop

Targeting Windows

- ❖ Windows Forms (WinForms)
- ❖ Windows Presentation Foundation (WPF)
- ❖ Universal Windows Platform (UWP)
- ❖ Windows UI (WinUI)
- ❖ Progressive Web Apps (PWAs)
- ❖ Electron Shell
- ❖ WPF Renderers for Xamarin.Forms
- ❖ Uno Platform | Avalonia
- ❖ Blazor Mobile Bindings or Blazor Hybrid 🤘
- ❖ .NET MAUI through WinUI 🎉

Reaching Mac OS/Linux

- ❖ Electron Shell
- ❖ Xamarin.Mac
- ❖ Mac OS Renderers for Xamarin.Forms
- ❖ Uno Platform | Avalonia
- ❖ Blazor Mobile Bindings or Blazor Hybrid 👊
- ❖ .NET MAUI through Mac OS Catalyst 👏

- ❖ GTK# Linux Renderers for Xamarin.Forms
- ❖ Uno Platform | Avalonia

Web All the Things

ASP.NET Core and the Modern Web


Totally Modular


Faster Development Cycle


Seamless transition
from on-premises to cloud

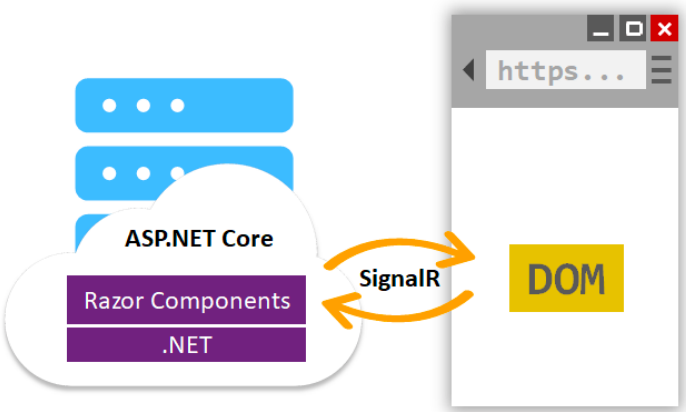

Choose your Editors
and Tools

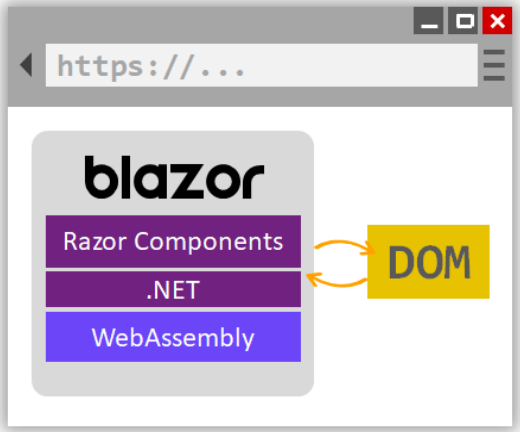
Open Source
with Contributions

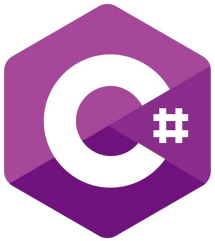

Cross-Platform

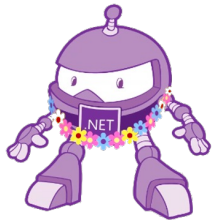

Fast

Blazing a trail ..









Blazor Hybrid: + .NET MAUI

Migration & Modernization

Microsoft .NET Windows Forms

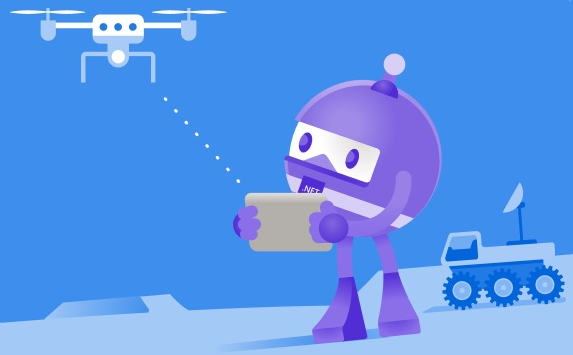
Windows Presentation Foundation

Universal Windows Platform

Web API

jQuery

Time for some Demos ..



The world runs on software

Thank You!

Snorkeling in MAUI

Let's keep chatting | @samidip