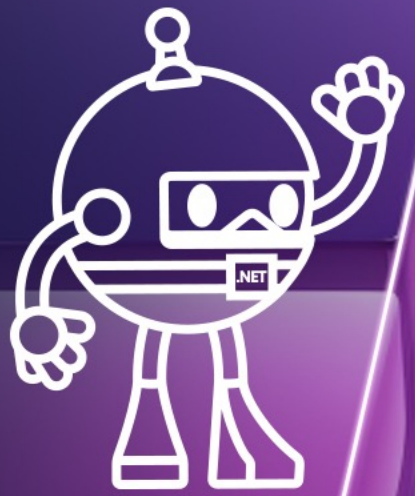
.NET Conf China 2023

2023/12/16
09:30 - 18:00

中国・北京



.NET中文社区

中国 · 北京

.NET Conf China 2023

WPF迁移Avalonia指南

董彬 Dong Bin

rabbitism

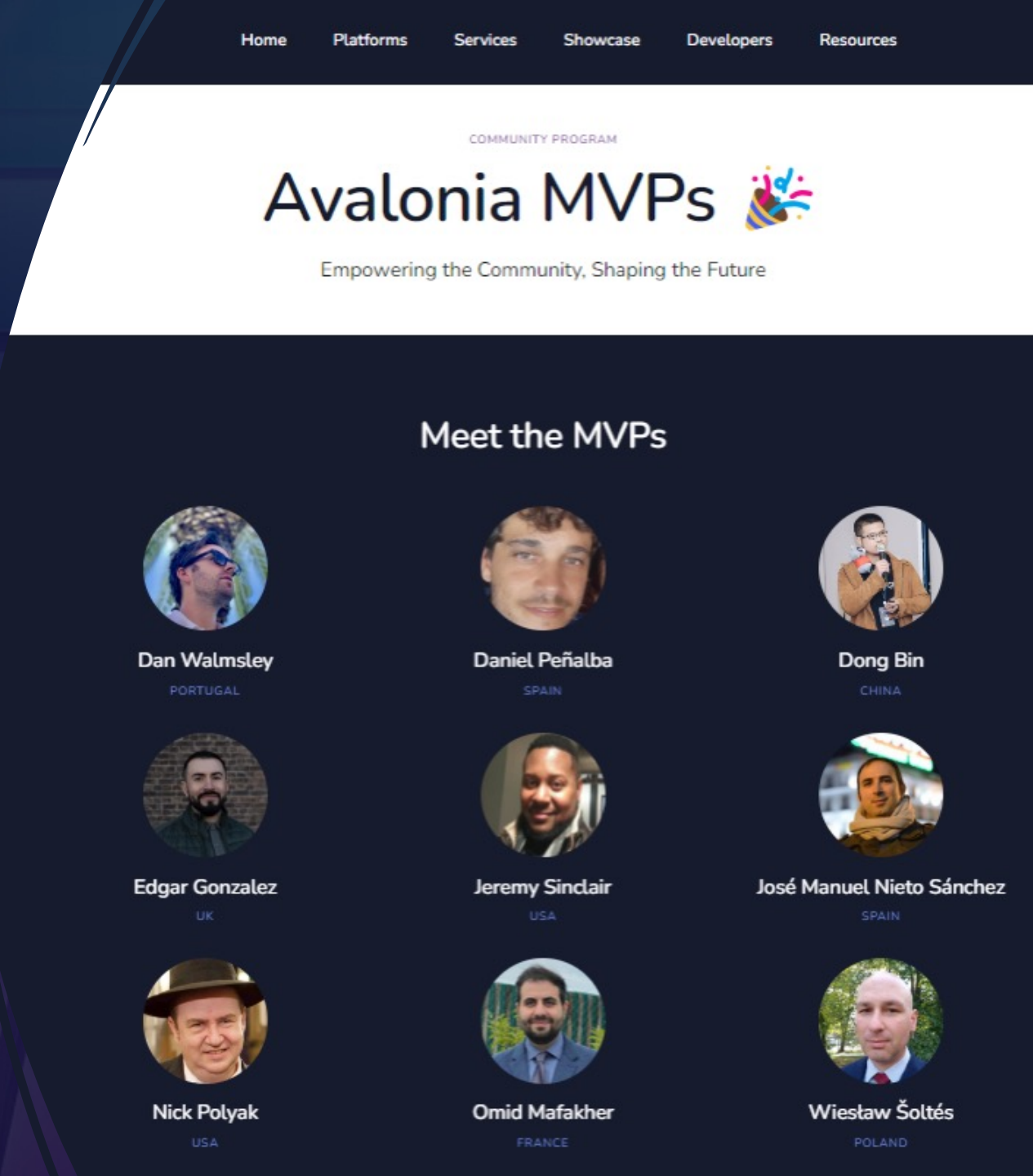
dongbin@irihi.tech

.NET中文社区

董彬

- Avalonia 中文社区发起人
- Avalonia MVP
- OpenKylin Avalonia SIG 组织者

- Semi.Avalonia作者
- Ursa.Avalonia作者



Agenda

- WPF应用在机制上与Avalonia的差异
- WPF中的关键技术在Avalonia中的替换方案
- WPF的生态在Avalonia中的替换方案

Avalonia UI 是什么

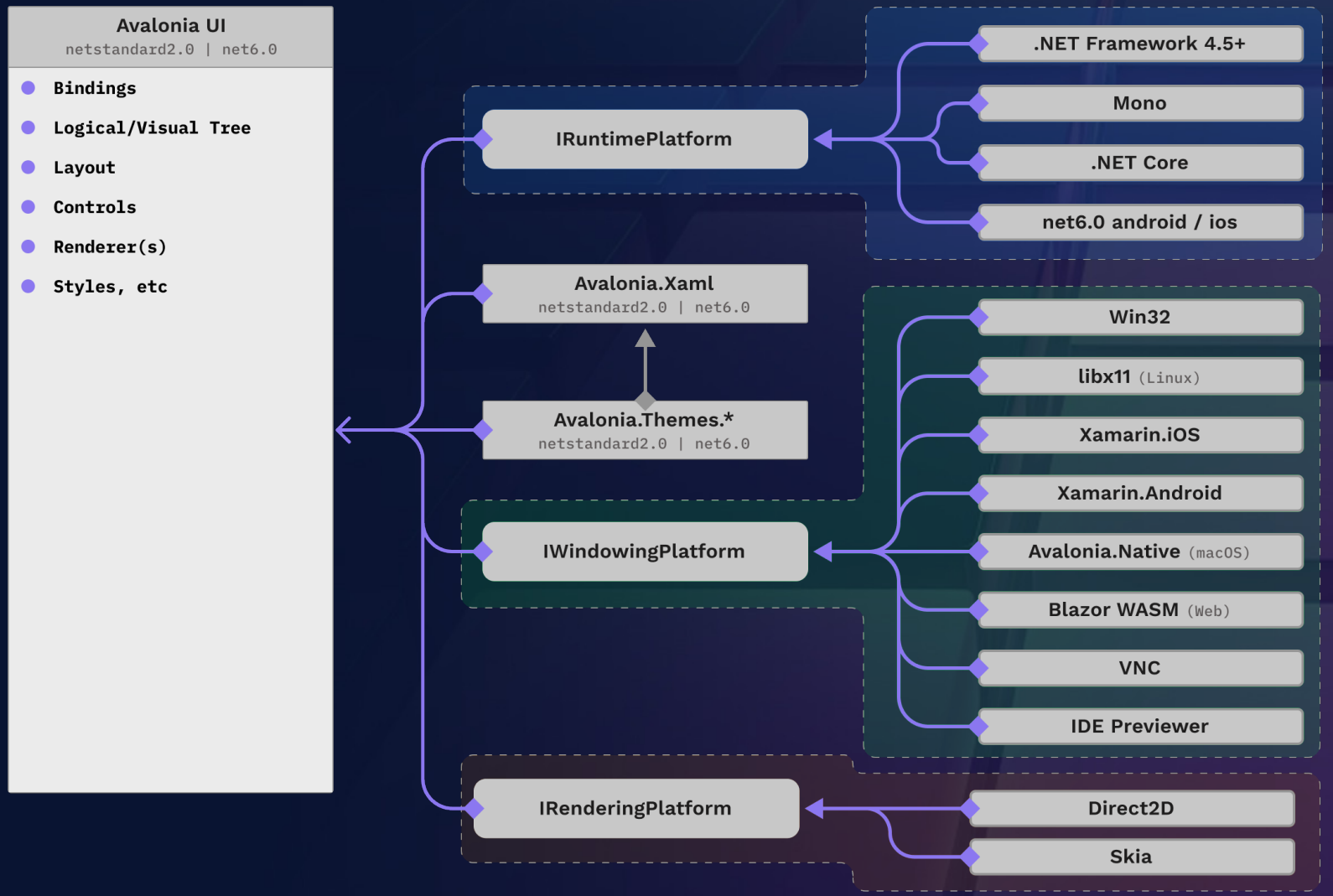
Avalonia UI
是基于 .NET 和 XAML 的
开源跨平台 UI 框架

Avalonia UI 是什么

- 可运行于所有主流平台

- MIT 协议
- 基于.NET和XAML
- 高保真，不依赖浏览器，不依赖原生控件
- 与WPF保持类似的编程范式，支持MVVM

Avalonia的技术架构



Avalonia与WPF的对比

Application Code	Application Code
Avalonia	WPF PresentationCore+PresentationFramework
Skia	MilCore
Mono / .NET CLR	DirectX / .NET CLR
Host OS	Windows

WPF与Avalonia的机制差异

- TargetFramework
  - WPF: net6.0-windows
  - Avalonia: net6.0

WPF与

- TargetFr
  - WPF:
  - Avalo
- Markup 约
  - WPF:
  - Avalo

```
1 usage
partial class MainWindow
{
    internal global::Avalonia.Controls.Button TestButton;

    /// <summary>
    /// Wires up the controls and optionally loads XAML markup and attaches dev tools (if Avalonia.Diagnostics package is referenced).
    /// </summary>
    /// <param name="loadXaml">Should the XAML be loaded into the component.</param>
    /// <param name="attachDevTools">Should the dev tools be attached.</param>

    1 usage
    public void InitializeComponent(bool loadXaml = true, bool attachDevTools = true)
    {
        if (loadXaml)
        {
            AvaloniaXamlLoader.Load( obj: this);
        }

f DEBUG
        if (attachDevTools)
        {
            this.AttachDevTools();
        }
ndif

        TestButton = this.FindNameScope()?.Find<global::Avalonia.Controls.Button>( name: "TestButton");
    }
```

WPF与Avalonia的机制差异

- TargetFramework
  - WPF: net6.0-windows
  - Avalonia: net6.0
- Markup 编译
  - WPF: 编译器两次编译
  - Avalonia: Source Generator
- XAML
  - WPF: BAML
  - Avalonia: IL

WPF与Avalonia的机制差异

- TargetFramework
  - WPF: net6.0-windows
  - Avalonia: net6.0
- Markup 编译
  - WPF: 编译器两次编译
  - Avalonia: Source Generator
- XAML
  - WPF: BAML
  - Avalonia: IL
- 默认样式
  - WPF：自带默认样式
  - Avalonia：样式单独分发

WPF与Avalonia的机制差异

- TopLevel

WPF与Avalonia的机制差异

- TopLevel
- Clipboard

WPF与Avalonia的机制差异

- TopLevel
- Clipboard
- MessageBox

WPF与Avalonia的机制差异

- TopLevel
- Clipboard
- MessageBox
- Pointer

WPF与Avalonia的细节差异

- 依赖对象与依赖属性
  - DependencyObject => AvaloniaObject
  - DependencyPropert => AvaloniaProperty

WPF与Avalonia的细节差异

- 依赖对象与依赖属性
  - DependencyObject => AvaloniaObject
  - DependencyPropert => AvaloniaProperty
- 普通的依赖属性

```
public static readonly DependencyProperty TestProperty = DependencyProperty.Register(
   nameof(Test), typeof(bool), typeof(MainWindow), new PropertyMetadata(default(bool)));

public bool Test
{
   get { return (bool)GetValue(TestProperty); }
   set { SetValue(TestProperty, value); }
}
```

```
public static readonly StyledProperty<bool> TestProperty = AvaloniaProperty.Register<MainWindow, bool>((Test));

public bool Test
{
   get => GetValue(TestProperty);
   set => SetValue(TestProperty, value);
}
```

WPF与Avalonia的细节差异

- 依赖对象与依赖属性
  - DependencyObject => AvaloniaObject
  - DependencyPropert => AvaloniaProperty
- 只读的依赖属性

```
internal static readonly DependencyPropertyKey TestPropertyKey = DependencyProperty.RegisterReadOnly(
   nameof(Test), typeof(bool), typeof(MainWindow), new FrameworkPropertyMetadata(false));

public bool Test => (bool)GetValue(TestPropertyKey.DependencyProperty);
```

```
public static readonly DirectProperty<MainWindow, bool> TestProperty =
   AvaloniaProperty.RegisterDirect<MainWindow, bool>(nameof(Test), o => o.Test);

private bool _test;

public bool Test
{
   get => _test;
   private set => SetAndRaise(TestProperty, ref _test, value);
}
```

WPF与Avalonia的细节差异

- 依赖对象与依赖属性
  - DependencyObject => AvaloniaObject
  - DependencyPropert => AvaloniaProperty
- 附加属性

```
public static readonly DependencyProperty TestProperty = DependencyProperty.RegisterAttached(
   "Test", typeof(bool), typeof(MainWindow), new PropertyMetadata(default(bool)));

public static void SetTest(DependencyObject element, bool value) => element.SetValue(TestProperty, value);

public static bool GetTest(DependencyObject element) => (bool)element.GetValue(TestProperty);
```

```
public static readonly AttachedProperty<bool> TestProperty =
   AvaloniaProperty.RegisterAttached<MainWindow, AvaloniaObject, bool>("Test");

public static void SetTest(AvaloniaObject obj, bool value) => obj.SetValue(TestProperty, value);

public static bool GetTest(AvaloniaObject obj) => obj.GetValue(TestProperty);
```

WPF与Avalonia的细节差异

- 依赖对象与依赖属性
  - DependencyObject => AvaloniaObject
  - DependencyPropert => AvaloniaProperty
    - 在Avalonia中具体实现为StyledProperty, DirectProperty, AttachedProperty.
- 具体差异
  - AvaloniaProperty支持泛型
  - defaultValue, PropertyChangeCallBack不在ProeprtyMetadata中

```
static MainWindow()
{
    BackgroundProperty.Changed.AddClassHandler<MainWindow, IBrush?>(OnBackgroundChange);
}

private static void OnBackgroundChange(MainWindow arg1, AvaloniaPropertyChangedEventArgs<IBrush?> arg2)
{
    // throw new NotImplementedException();
}
```

  - DirectProperty是全新的概念: CLR属性实现通知与绑定

WPF与Avalonia的细节差异

- 绑定Binding
  - UpdateSourceTrigger
  - LogicalTree/VisualTree

```
Background="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=Window, Tree=Logical}, Path=Background}"
```

WPF与Avalonia的细节差异

- 绑定Binding
  - UpdateSourceTrigger
  - LogicalTree/VisualTree

```
Background="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=Window, Tree=Logical}, Path=Background}"
```

  - 简化的绑定语法
    - XAML

```
Background="{Binding $parent[Window].Background}"
```

    - C#

```
button[!BackgroundProperty] = this[!BackgroundProperty];
```

WPF与Avalonia的细节差异

- 绑定Binding
  - UpdateSourceTrigger
  - LogicalTree/VisualTree

```
Background="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=Window, Tree=Logical}, Path=Background}"
```

  - 简化的绑定语法
    - XAML

```
Background="{Binding $parent[Window].Background}"
```

    - C#

```
button[!BackgroundProperty] = this[!BackgroundProperty];
```

  - IValueConverter: Avalonia 自己的命名空间

WPF与Avalonia的细节差异

- 样式
  - Avalonia: WPF 与 CSS 的结合体
  - Avalonia ControlTheme 与 WPF Style等价

WPF与Avalonia的细节差异

- 样式
  - Avalonia: WPF 与 CSS 的结合体
  - Avalonia ControlTheme 与 WPF Style等价
  - StyleKey:
    - WPF: `DefaultStyleKeyProperty.OverrideMetadata(typeof(CustomControl), new FrameworkPropertyMetadata(typeof(CustomControl)));`
    - Avalonia: `protected override Type StyleKeyOverride => typeof(CustomControl);`

WPF与Avalonia的细节差异

- 样式
  - Avalonia: WPF 与 CSS 的结合体
  - Avalonia ControlTheme 与 WPF Style等价
  - StyleKey:
    - WPF: `DefaultStyleKeyProperty.OverrideMetadata(typeof(CustomControl), new FrameworkPropertyMetadata(typeof(CustomControl)));`
    - Avalonia: `protected override Type StyleKeyOverride => typeof(CustomControl);`
  - Avalonia中没有DataTrigger和EventTrigger
  - WPF中的VisualStateManager可以用Avalonia的PseudoClass替换
  - WPF中的Storyboard可以用Avalonia的KeyframeAnimation替换
  - Avalonia可以方便地对TemplatePart赋值

WPF与Avalonia的细节差异

- 数据模板
  - Avalonia的数据模板可以指定DataType为接口
  - Avalonia的IDataTemplate接口为DataTemplate和DataTemplateSelector的结合体

```
public class ViewLocator : IDataTemplate
{
    public Control Build(object data)
    {
        return new TextBlock { Text = data.ToString() };
    }

    public bool Match(object data)
    {
        return data is ViewModelBase;
    }
}
```

WPF与Avalonia的细节差异

- 隧道传播事件 Tunnelling Event
  - WPF中的PreviewKeyDown与KeyDown
  - Avalonia:

```
public MainWindow()
{
    InitializeComponent();
    button.AddHandler(Button.KeyDownEvent, OnButtonKeyDown, RoutingStrategies.Tunnel);
}

private void OnButtonKeyDown(object? sender, KeyEventArgs e)
{
    throw new NotImplementedException();
}
```

WPF与Avalonia的细节差异

- Logical Tree / Visual Tree 查询
  - WPF：VisualTreeHelper/LogicalTreeHelper
  - Avalonia: ILogical和Visual的拓展方法

WPF与Avalonia的细节差异

- PointerEventArgs
  - WPF: `bool b = e.LeftButton == MouseButtonState.Pressed;`
  - Avalonia: `bool b = e.GetCurrentPoint(this).Properties.IsLeftButtonPressed;`

WPF生态在Avalonia的平替

- MVVM框架
  - ReactiveUI 和 CommunityToolkit 可通用
  - Prism
- Charts
  - LiveCharts2
  - ScottPlot
- 控件库
  - 暂无DevExpress, Syncfusion, Telerik
  - ActiPro开始布局
- 随时关注awesome-avalonia仓库

Thanks
Q&A