Avalonia MVVM

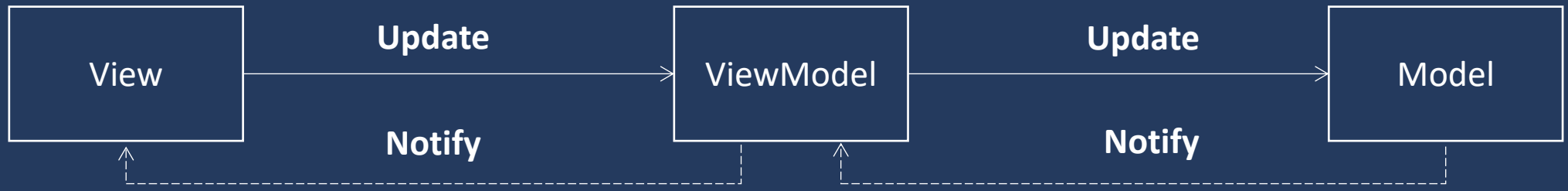
完美的UI逻辑与业务逻辑的分离艺术

WenkeYin（尹文科）
LayUI.WPF|LayUI.Avalonia 作者

本期内容

1. MVVM模式概述
2. 何时使用MVVM模式
3. MVVM与Avalonia的结合
4. 数据绑定讲解
5. 什么是INotifyPropertyChanged
6. 什么是命令（ICommand）

MVVM模式概述



MVVM是一种程序应用结构方式，通常它使用数据绑定系统帮我们在视图与视图模型之间传递数据。这种模式下，我们就可以实现应用逻辑与视图进行分离。

通常我们说的MVVM其实是一种简称，全称为：Model（模型）-View（视图）-ViewModel（视图模型）。

通过MVVM这种模式我们可以对我们的业务进行单独的单元测试，这个测试完全不依赖目标UI平台。

何时使用MVVM模式

与事件驱动的代码后台模式相比，传统的开发模式对于小项目不使用MVVM模式可能容易理解和维护。而MVVM是一种更复杂的编程模式，它采用的是数据驱动的方式进行开发，随着业务的增加，程序往往会更加复杂，这种模式才会显现出来它的优势。

MVVM模式适用场景：

- 复杂的数据交互应用
- 高密度可测试性的项目
- 多平台共享业务逻辑
- 大型团队协作开发

根据上面描述，我们可以得出以下结论考虑转换：

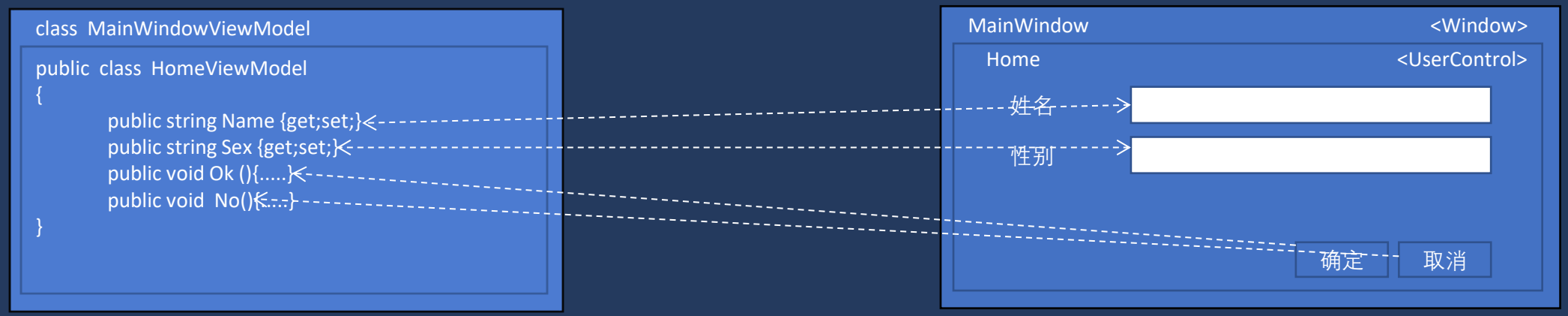
1. 简单的代码后台模式随着业务的增加，程序变得难以维护和复用，这时可以开始考虑转到MVVM模式
2. 从一开始就使用MVVM模式，为后续程序的不断增长以及人员的高复用和项目的可测试性做准备

Avalonia与MVVM的结合

当我们的程序使用MVVM模式，我们就必须建立视图（View）和视图模型（ViewModel）。在这里模型（Model）在MVVM模式中是其他的其他部分，通常用来存储数据，在MVVM模式中侧重的是分离，焦点聚焦在视图与视图模型上。

在Avalonia中视图可以是一个用户控件或是一个窗体的内部元素组成。内部元素可以是普通控件、用户控件或则是自己组合成的高级控件混合而成。

既然我们使用MVVM模式，就一定离不开**数据绑定**，它是程序中将视图与视图模式分离的关键，通常前端AXAML文件的绑定语法为 **{Binding xxxx}** 这种方式进行捆绑，这也是Avalonia的核心技术。我们可以将视图与视图模型之间的这两者关系可是化为数据连接的两个层：



数据绑定讲解

数据绑定是在视图与它显示的数据之间建立连接的过程。 如果绑定具有正确的设置，并且数据提供正确的通知，当数据更改其值时，绑定到数据的元素会自动反映更改。 数据绑定还意味着，如果元素中的数据的外部表示形式发生更改，则可以自动更新基础数据以反映更改。 例如，如果用户编辑元素中的 TextBox 值，基础数据值将自动更新以反映该更改。

在Avalonia中，只有是依赖属性才能进行数据绑定，在这个绑定系统当中有以下几种模式：

1. Default
2. OneWay
3. TwoWay
4. OneTime
5. OneWayToSource

在AXAML中的写法一般法：<TextBox Text="{Binding Name, Mode=TwoWay}" />，如果绑定的数据源是一个对象，则AXAML的绑定语法为： <TextBox Text="{Binding User.Name, Mode=TwoWay}" />

什么是INotifyPropertyChanged

INotifyPropertyChanged它是一个接口，是实现MVVM模式中数据绑定的核心机制，被实现的类是可以允许对象属性发生变化时通知绑定 控件进行更新，如果绑定模式为TwoWay，控件值发生变化同样可以更新模型中的属性值。

由于我们的ViewModel以及Model某些情况下都会实现INotifyPropertyChanged，我们都会对其封装一个基类，直接进行继承使用。

```
public class ViewModelBase : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

注意：这里的模型可以为ViewModel也可以为Model，只要实现目标接口，都可以进行更新。

什么是命令（ICommand）

在Avalonia中命令是按钮的一种行为动作，在按钮中它有一个属性 Command，这个充当我们按钮的点击事件Click，在MVVM模式中，事件处理业务的方式已经不建议继续使用，而是采用Command的方式进行业务代码的操作。在这里我们同样的也需要采用绑定的方式进行处理，这样可以做到我们的视图与业务进行分离。

当我们采用命令的方式进行业务操作，我们的Command是可以直接进行绑定方法的，前提是方法是公开。绑定方法的这种方式这其实是一种简写。

需要注意的是ICommand是一个接口，我们必须去通过一个类去实现完成它，它的实现类请看 **DelegateCommand.cs**文件。

DelegateCommand.cs

命令（Command）捆绑关系图

Home

```
<Button Content="Cancel" Command="{Binding CancelCommand}"/>
<Button Content="OK" Command="{Binding Ok}"/>
```

HomeViewModel

```
private DelegateCommand _CancelCommand;

public DelegateCommand CancelCommand
    => _CancelCommand ?? (_CancelCommand    = new
    DelegateCommand(ExecuteCancelCommand));

void ExecuteCancelCommand() => Cancel();

private void Cancel()
{
}

public  void Ok()
{
}
```

注意：箭头的指向方向即为绑定模式

Thank you