Microsoft

JETBRAINS Microsoft Reactor

MVP Microsoft Most Valuable Professional

Binding Mode

Juster zhu（朱震）
Microsoft MVP | Huawei HCDE | GeneralUpdate Owner

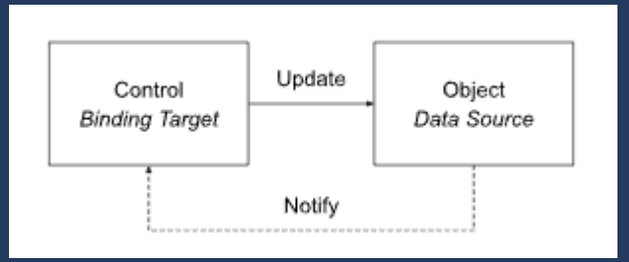
Microsoft

本期内容

1. 绑定模式

Microsoft

数据流的方向（绑定模式）



回顾绑定 (Binding) 的本质：

连接 View (UI 元素属性) 和 ViewModel (数据属性/命令) 的桥梁。

目的：实现 UI 自动更新反映数据变化，用户输入自动更新数据（双向时）。

为什么需要“方向”？

并非所有场景都需要双向同步。 <TextBlock Text="{Binding xxx, Mode=xxxWay}"/>

示例：

显示只读信息（如用户名标签） -> 只需要从 VM -> View (单向)。

用户输入表单（如文本框） -> 需要 View -> VM (用户输入) 和 VM -> View (初始值/验证反馈) (双向)。

显示一次初始化数据后不再改变 -> 只需要初始设置 (一次性)。

核心问题： 数据变化的源头在哪里？需要同步到哪里？数据流方向决定了这个路径。

Microsoft

数据流的方向

模式 (Mode)	数据流向	VM 变 -> View 更新	View 变 (用户交互) -> VM 更新	典型应用
OneWay	Source (VM) -> Target	**是**	**否**	显示只读数据
TwoWay	Source <-> Target	**是**	**是**	用户输入控件 (文本框, 复选框等)
OneTime	Source -> Target (一次)	否 (仅初始)	**否**	初始化后不变的数据 (少用)
OneWayToSource	Target -> Source	**否**	**是**	收集 UI 状态 (相对少用)
Default	取决于目标属性	依属性而定	依属性而定	了解属性默认行为

如果未指定模式，则将始终使用默认模式。对于不因用户交互而改变值的控件属性，默认模式通常是OneWay。对于因用户输入而改变值的控件属性，默认模式通常是TwoWay。
例如，TextBlock.Text属性的默认模式是OneWay，而TextBox.Text属性的默认模式是TwoWay。

Microsoft

默认模式 (Default)

行为： 使用目标依赖属性 (Target property) 的元数据中定义的默认模式。

关键点：
在 Avalonia 中，大多数用户输入控件（TextBox.Text, CheckBox.IsChecked, Slider.Value 等）的默认模式是 TwoWay。
大多数只用于显示的控件（TextBlock.Text, Image.Source 等）的默认模式是 OneWay。

Microsoft

TwoWay (双向：源 <-> 目标)

方向： ViewModel (Source) <-> View (Target)。

行为：
View 属性初始值设置为 VM 属性的值。
当 VM 属性值改变时，View 属性 自动更新。
当 View 属性因用户交互改变时（如输入文本、勾选复选框、移动滑块），VM 属性值 自动更新。

典型应用场景：
所有需要用户输入并同步到 VM 的表单控件 (TextBox.Text, CheckBox.IsChecked, Slider.Value, ComboBox.SelectedItem, RadioButton.IsChecked)。
注意： 在 Avalonia 中，很多用户输入控件的 默认绑定模式就是 TwoWay (这是与 WPF 的重要区别之一！)。

XAML 示例:

```
<TextBox Text="{Binding Email, Mode=TwoWay}"/> <!-- 用户可输入，输入内容实时更新到 VM -->
<!-- 因为 TextBox 默认是 TwoWay，通常可以省略 Mode -->
<CheckBox IsChecked="{Binding IsSubscribed, Mode=TwoWay}"/>
```

Microsoft

OneWay (单向：源 -> 目标)

方向： ViewModel (Source) -> View (Target)。

行为：
View 属性初始值设置为 VM 属性的值。
当 VM 属性值改变时，View 属性 自动更新 以反映新值。
当 View 属性因用户交互改变时（如直接在文本框输入），VM 属性值 不会更新！

典型应用场景：
显示只读数据 (TextBlock.Text, Image.Source, 进度条 Value 仅用于显示进度)。
基于 VM 状态改变 UI 外观 (IsVisible, IsEnabled, 样式触发器源)。

XAML 示例:

```
<TextBlock Text="{Binding UserName, Mode=OneWay}"/> <!-- 显示用户名，用户不能改 -->
<ProgressBar Value="{Binding DownloadProgress, Mode=OneWay}" Maximum="100"/>
```

Microsoft

OneTime (一次性：源 -> 目标，仅初始)

方向： ViewModel (Source) -> View (Target) (仅一次)。

行为：
View 属性 仅在绑定初始化时 设置一次为 VM 属性的值。
之后，无论 VM 属性如何改变，View 属性都不会更新。
View 属性的改变也 永远不会 更新回 VM 属性。

典型应用场景：
显示初始化后 绝对不会改变 的数据（虽然实践中较少见，因为数据常变）。
性能优化：在极少数确定数据只初始化一次且不变的情况下，避免绑定系统持续监听的开销。

XAML 示例:
<TextBlock Text="{Binding ApplicationStartTime, Mode=OneTime}"/> <!-- 显示启动时间，启动后不会变 -->

Microsoft

OneWayToSource (单向：目标 -> 源)

方向： View (Target) -> ViewModel (Source)。

行为：
VM 属性初始值被忽略。
当 View 属性因用户交互改变时，VM 属性值 自动更新。
当 VM 属性值改变时，View 属性 不会更新。

典型应用场景 (相对较少)：
需要从 UI 收集信息，但 UI 的初始值不由这个 VM 属性决定，或者 UI 状态是主要驱动源。
与只写属性配合使用。
自定义控件或行为，需要将 UI 状态反向设置到 VM。

XAML 示例 (假设场景)：

```
<!-- 假设有一个自定义绘图控件，我们只关心用户最后绘制的点坐标，不关心初始值 -->
<MyCustomCanvas LastDrawnPoint="{Binding UserDrawnPoint, Mode=OneWayToSource}"/>
```

Microsoft

小结

1.数据流的方向（绑定模式）概念
2.模式的几种类型、作用