[1. WPF 3](#_Toc462134483)

[1-1. 版本演进 3](#_Toc462134484)

[1-1-1. WPF3.0 3](#_Toc462134485)

[1-1-2. WPF3.5 3](#_Toc462134486)

[1-1-3. WPF3.5 SP1 3](#_Toc462134487)

[1-1-4. WPF4 3](#_Toc462134488)

[1-1-5. 最新版本WPF4.5 3](#_Toc462134489)

[1-2. 主要来自三个程序集的支持 3](#_Toc462134490)

[1-2-1. PresentationCore 4](#_Toc462134491)

[1-2-2. PresentationFramework 5](#_Toc462134492)

[1-2-3. WindowBase 6](#_Toc462134493)

[2. XAML 6](#_Toc462134494)

[2-1. 为什么WPF要用XAML 6](#_Toc462134495)

[2-1-1. WPF并不必然要使用XAML 6](#_Toc462134496)

[2-2. XAML编译成BAML 7](#_Toc462134497)

[2-3. XAML基础 7](#_Toc462134498)

[2-3-1. XAML文档中的每个元素都映射为.NET类的一个实例 7](#_Toc462134499)

[2-4. XAML名称空间 7](#_Toc462134500)

[2-4-1. XML名称空间的名称和任何特定的.NET名称空间都不匹配 8](#_Toc462134501)

[3. 布局 8](#_Toc462134502)

[3-1. WPF窗口只能包含单个元素 8](#_Toc462134503)

[3-2. 布局容器 9](#_Toc462134504)

[3-2-1. 所有WPF布局容器都派生自System.Windows.Controls.Panel 抽象类 9](#_Toc462134505)

[3-2-2. 核心布局面板 10](#_Toc462134506)

[3-3. StackPanel 12](#_Toc462134507)

[3-4. Border控件 13](#_Toc462134508)

[3-4-1. 继承自Decorator类 13](#_Toc462134509)

[3-5. Grid面板 13](#_Toc462134510)

[3-5-1. ShowGridLines属性可以显示网格线 13](#_Toc462134511)

[3-5-2. 在Visual Studio 中使用Grid面板 14](#_Toc462134512)

[3-5-3. 调整行列高度和跨度 14](#_Toc462134513)

[3-5-4. 设置元素的位置 15](#_Toc462134514)

[3-6. 附着属性 15](#_Toc462134515)

[4. Controls 15](#_Toc462134516)

[4-1. DataGrid 15](#_Toc462134517)

[4-1-1. 可以设置其ItemSource 属性实现列的动态生成 15](#_Toc462134518)

[5. 事件 17](#_Toc462134519)

[5-1. 所有元素的键盘事件及顺序 17](#_Toc462134520)

[6. WPF感悟 17](#_Toc462134521)

[6-1. 解耦和数据关联 17](#_Toc462134522)

[6-1-1. UI和逻辑解耦 17](#_Toc462134523)

[6-1-2. UI层和逻辑层的血管是数据关联 Data Binding 17](#_Toc462134524)

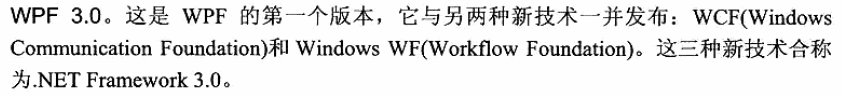
[6-1-3. 尽量不要试图通过UI层操作数据，而是直接去操作数据源 17](#_Toc462134525)

[6-1-4. 数据关联的核心思想是：数据决定UI 17](#_Toc462134526)

# WPF

## 版本演进

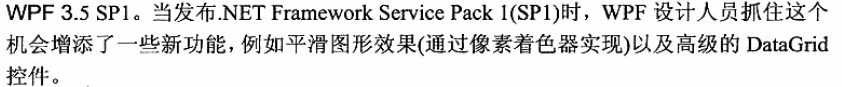
### WPF3.0



### WPF3.5



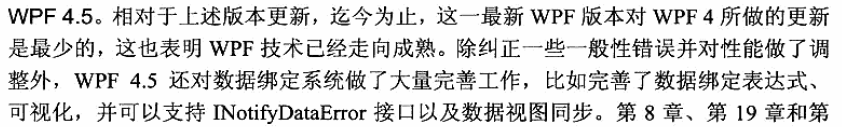
### WPF3.5 SP1



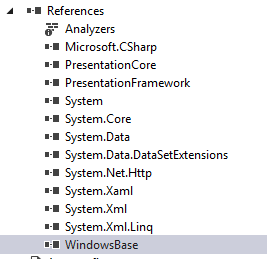
### WPF4



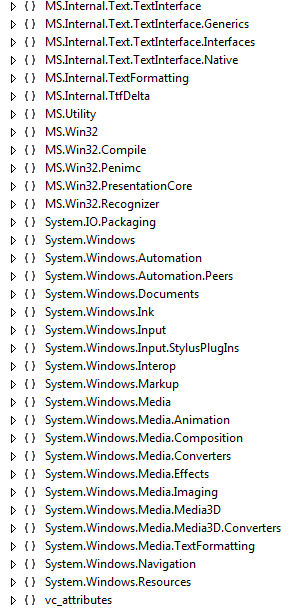
### 最新版本WPF4.5



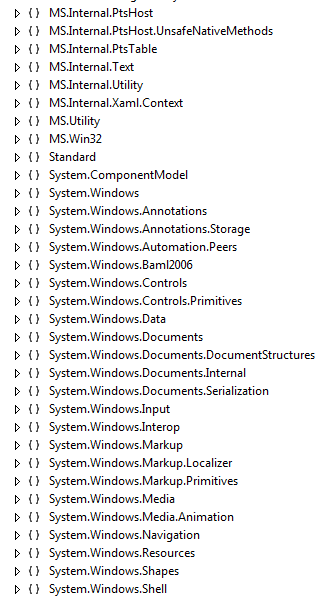
## 主要来自三个程序集的支持



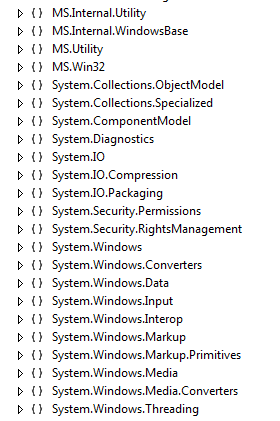
### PresentationCore



### PresentationFramework



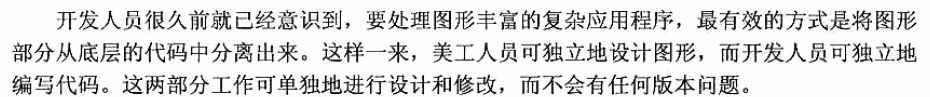
### WindowBase



# XAML

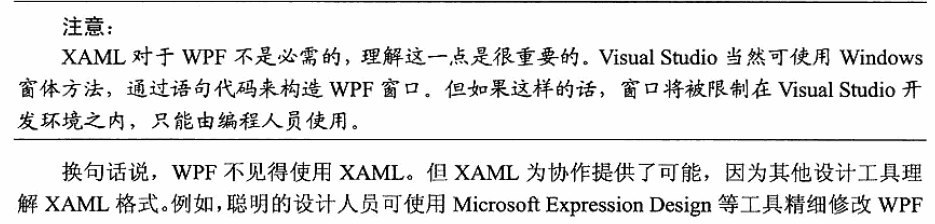
## 为什么WPF要用XAML

主要还是为了分工和效率。所以在界面设计和开发没有分工的公司（往往全部工作都有程序员来完成），使用XAML的理由就没那么充分了

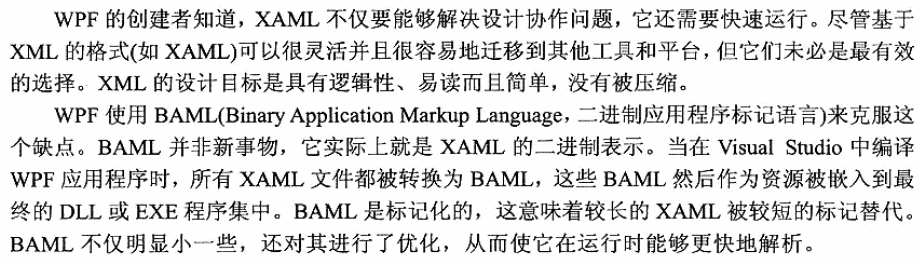


### WPF并不必然要使用XAML

不过如果用的话，微软宣称可以更好的协作。

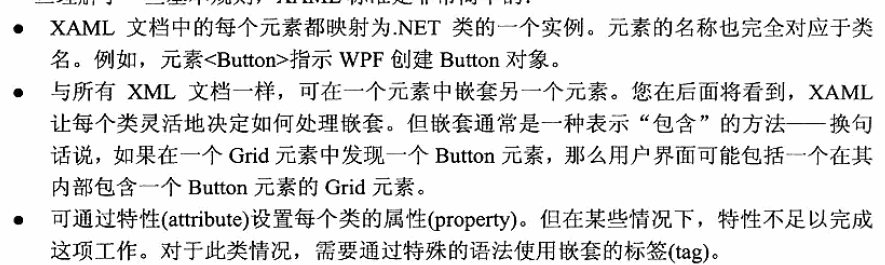


## XAML编译成BAML

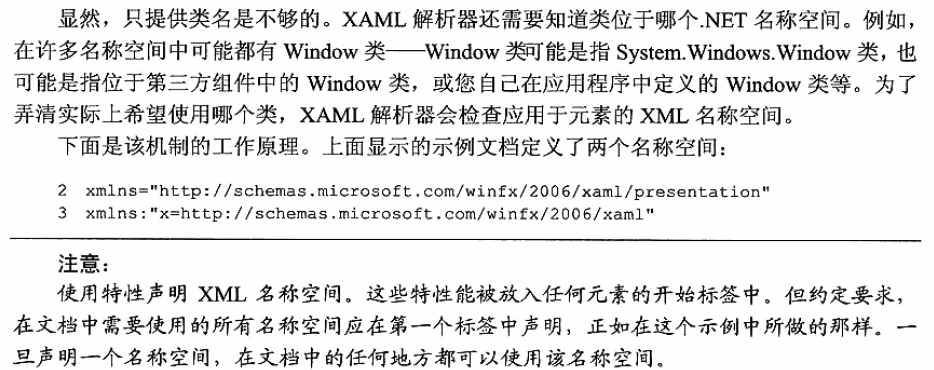


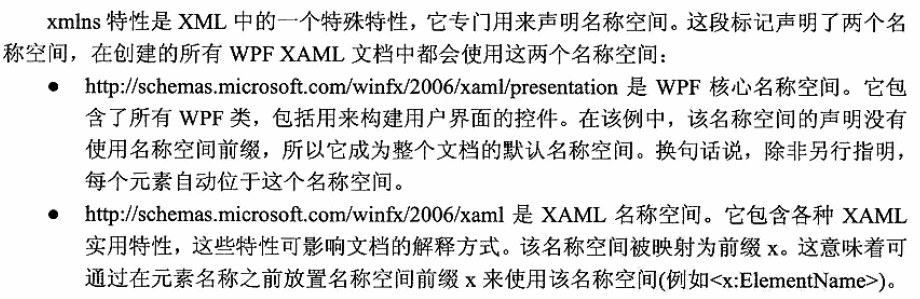
## XAML基础

### XAML文档中的每个元素都映射为.NET类的一个实例

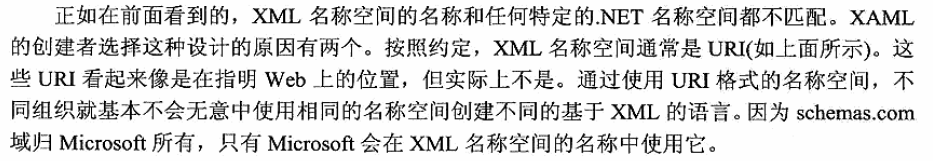


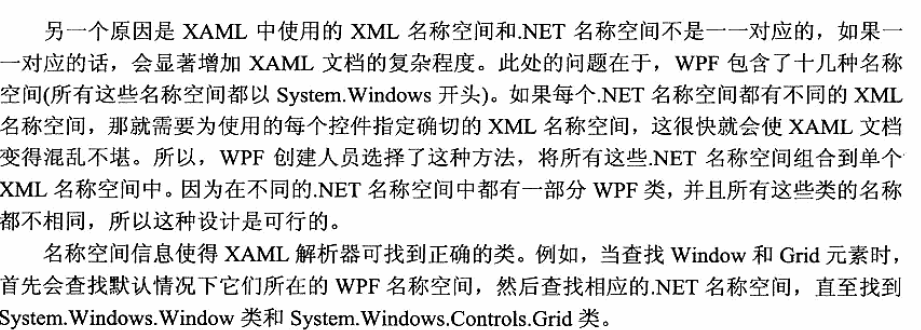
## XAML名称空间





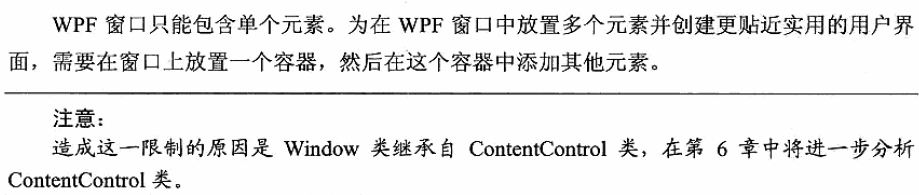
### XML名称空间的名称和任何特定的.NET名称空间都不匹配





# 布局

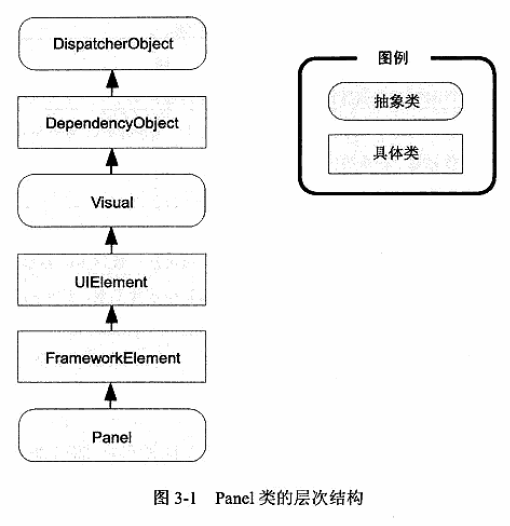
## WPF窗口只能包含单个元素



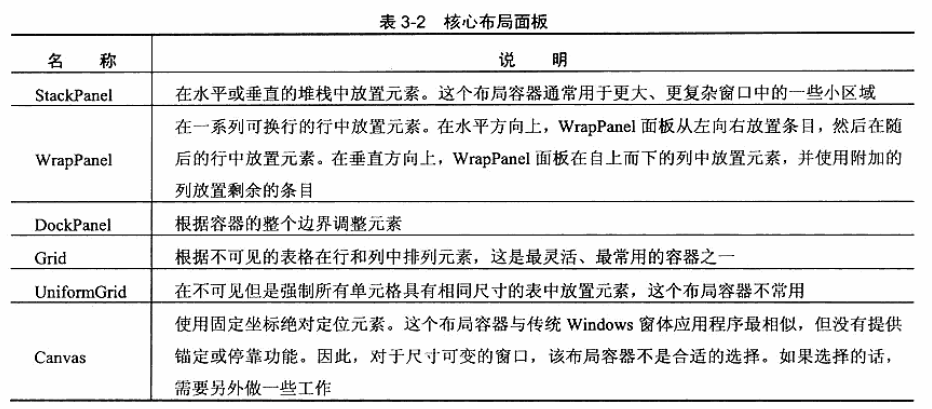
## 布局容器

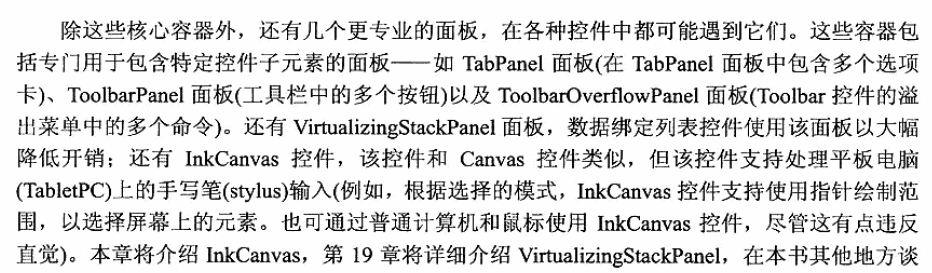
### 所有WPF布局容器都派生自System.Windows.Controls.Panel 抽象类





### 核心布局面板



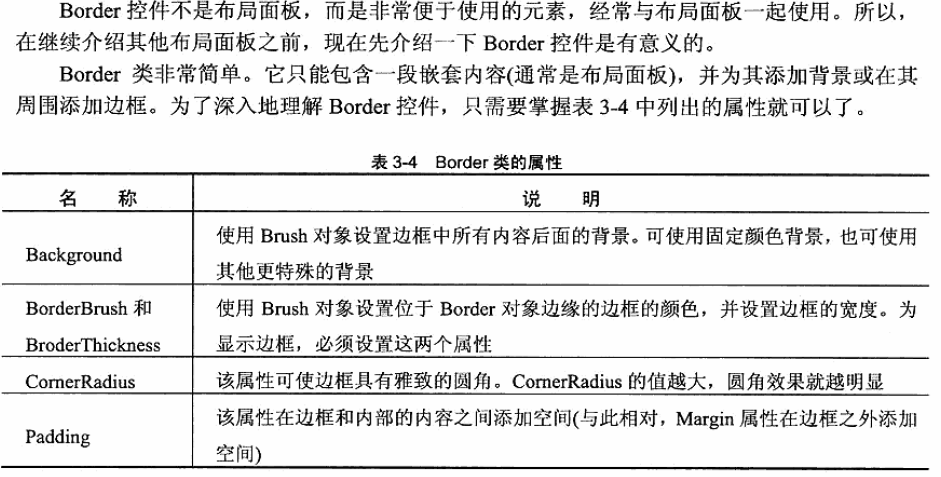


## StackPanel

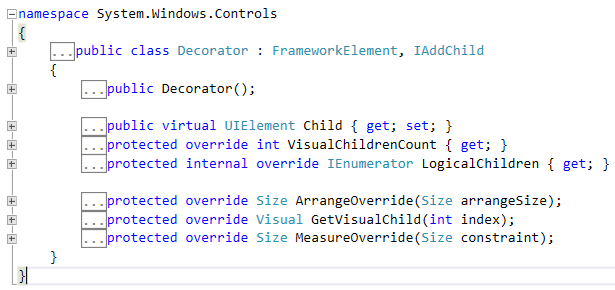
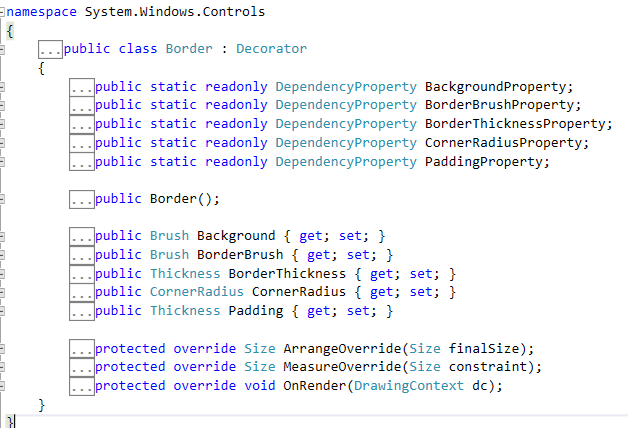


通过Orientation 属性来设置元素方向，默认情况下自上而下排列元素

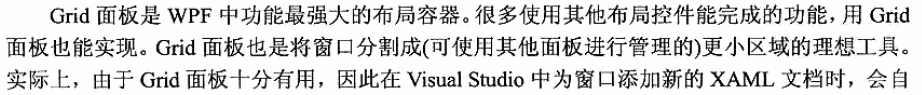
## Border控件

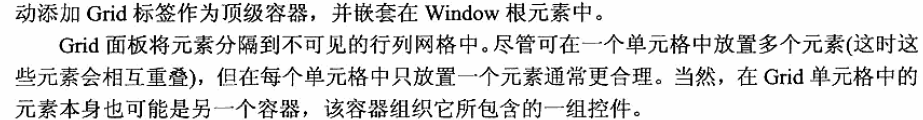


### 继承自Decorator类



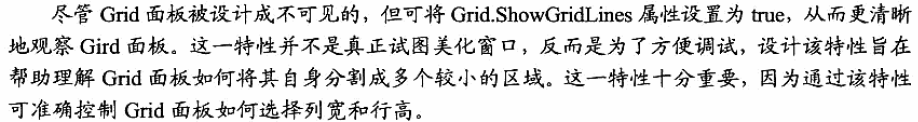
## Grid面板



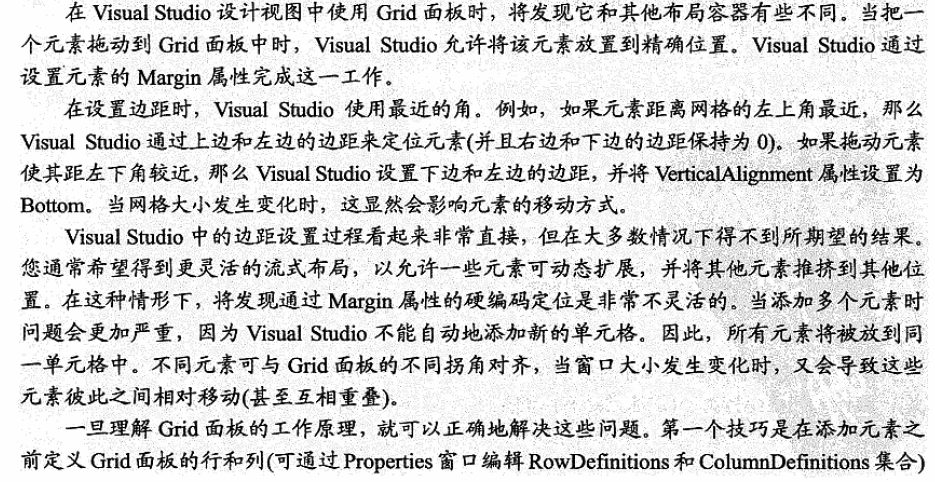


### ShowGridLines属性可以显示网格线

可以在调试时使用

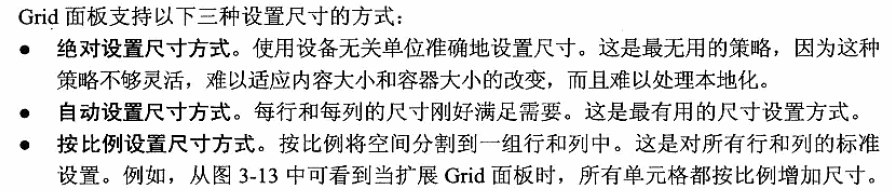


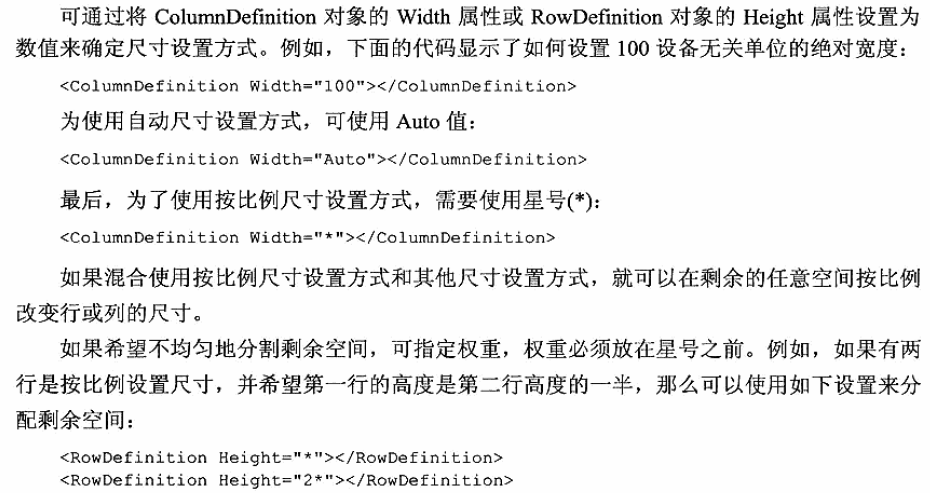
### 在Visual Studio 中使用Grid面板





### 调整行列高度和跨度

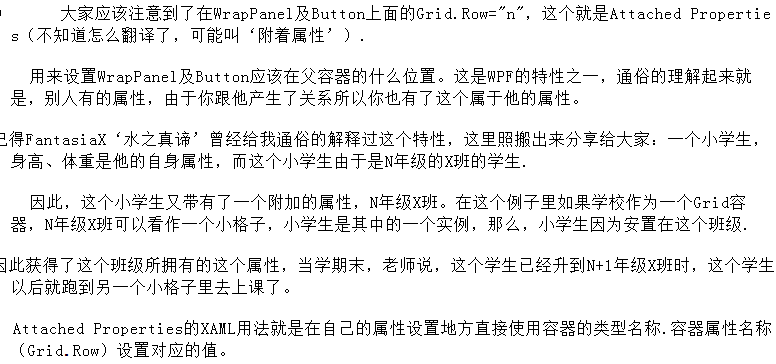




### 设置元素的位置

Grid.Row 和 Grid.Column

## 附着属性



# Controls

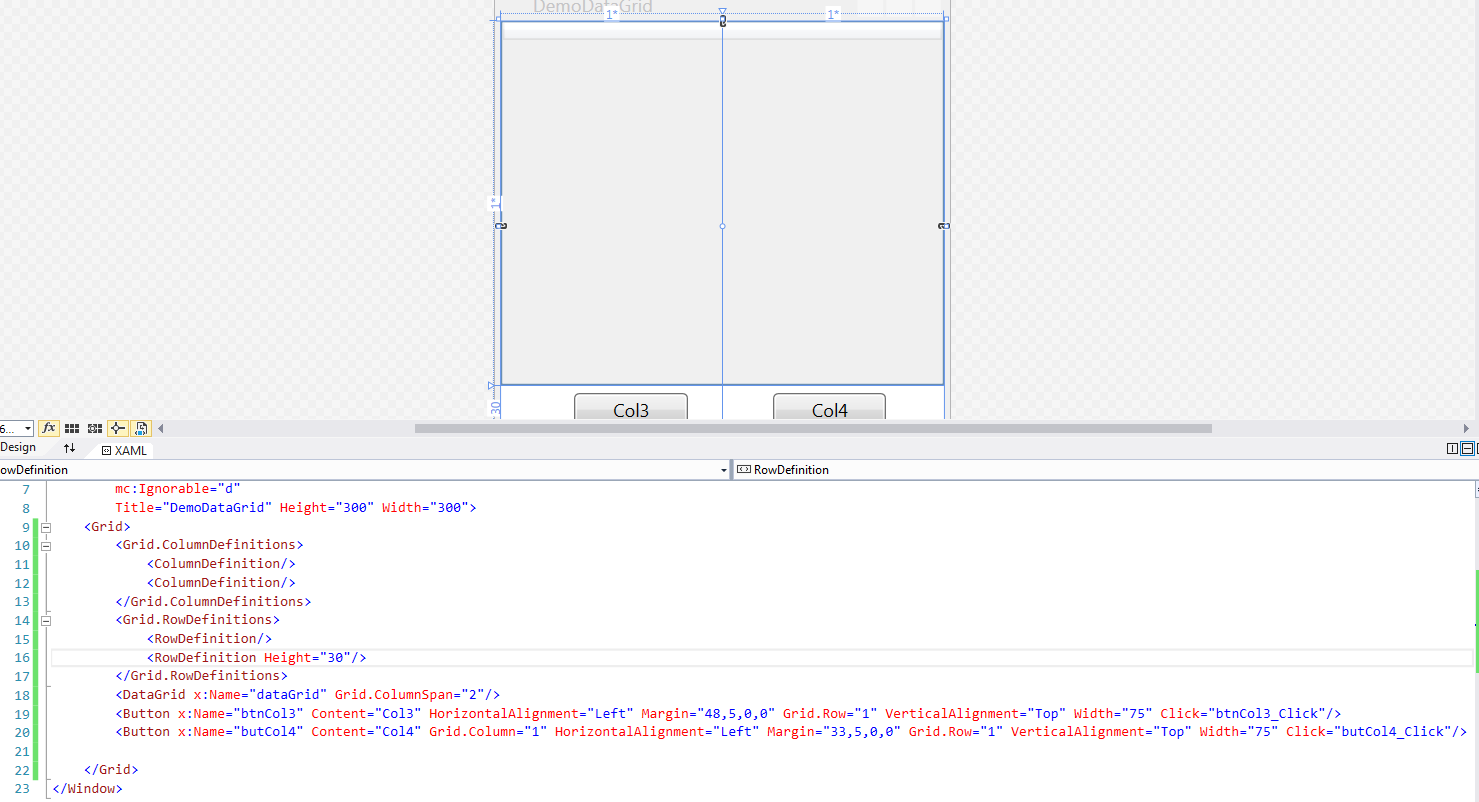
## DataGrid

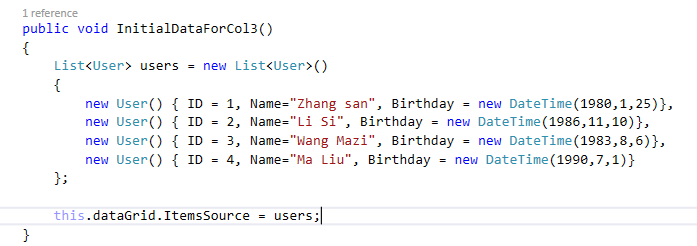
### 可以设置其ItemSource 属性实现列的动态生成

只需要设置其ItemSource即可。而且不同列刷新时没有感觉有闪屏的问题。

即使加载100，000行数据，也没感觉有卡的现象

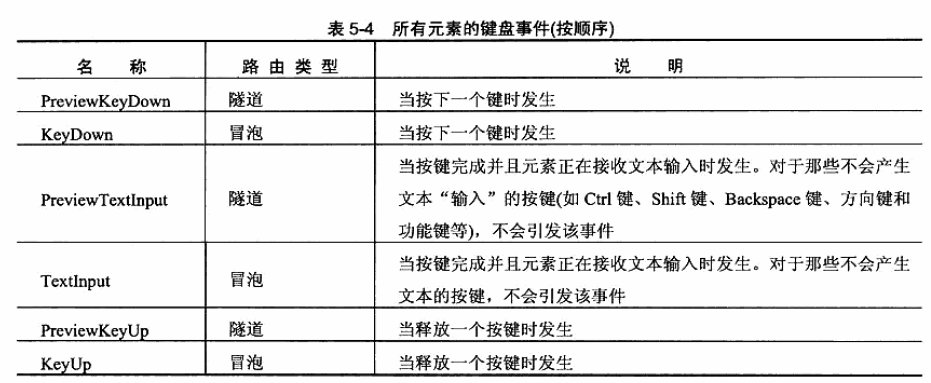
[Demo\DataGridDynamicColumns](Demo/DataGridDynamicColumns)





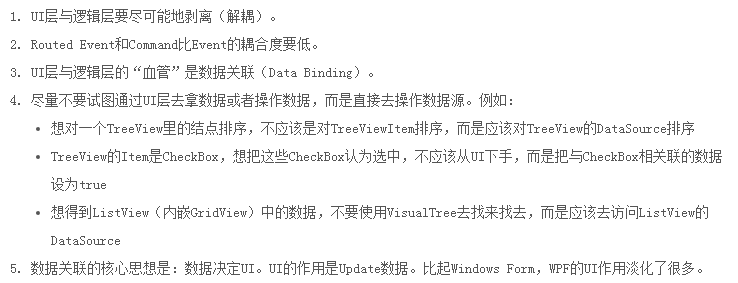
# 事件

## 所有元素的键盘事件及顺序



# WPF感悟

## 解耦和数据关联



### UI和逻辑解耦

### UI层和逻辑层的血管是数据关联 Data Binding

### 尽量不要试图通过UI层操作数据，而是直接去操作数据源

### 数据关联的核心思想是：数据决定UI