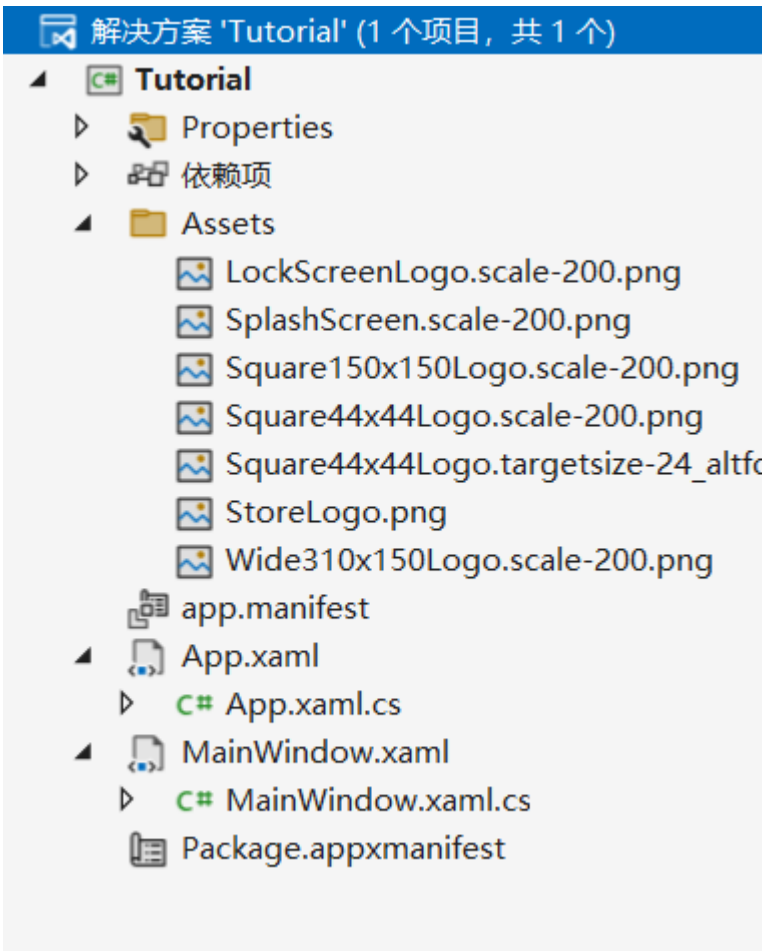
WinUI 3 | Ep2. 认识 WinUI 3 项目 & XAML 基础语法

在上一节中，我们完成了环境的配置，并且创建并运行了第一个 WinUI 3 项目。接下来，让我们来看看它的构成：



App

打开位于 `App.xaml` 下级的 `App.xaml.cs`

```
namespace Tutorial
{
    public partial class App : Application
    {
        public App()
        {
            this.InitializeComponent();
        }

        protected override void
```

```
OnLaunched(Microsoft.UI.Xaml.LaunchActivatedEventArgs args)
        {
            m_window = new MainWindow();
            m_window.Activate();
        }

        private Window? m_window;
    }
}
```

和控制台/ASP.NET Core C#项目不同，WinUI 3 项目没有`Program.cs`和手写的`Main`方法。`App.xaml.cs`里面的`App`类是整个应用的起始点。

`App`类会为我们完成整个应用的初始化，包括它需要的各种资源的准备，初始化完成后会在启动时执行`OnLaunched`方法，创建并展示一个窗口。

> 当然，`Main`方法依旧存在，但是是由项目构建系统根据`App.xaml`自动为我们生成的。

主窗口

WinUI 3 和它的前辈 UWP，WPF 一样，使用了 XAML 作为编写界面的语法。

打开`MainWindow.xaml`，让我们看看 XAML 长什么样：

```
<?xml version="1.0" encoding="utf-8"?>
<Window
    x:Class="Tutorial.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Tutorial"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Title="Tutorial">

    <StackPanel Orientation="Horizontal" HorizontalAlignment="Center"
VerticalAlignment="Center">
        <Button x:Name="myButton" Click="myButton_Click">Click Me</Button>
    </StackPanel>
</Window>
```

可以看见里面有一个 Window 标签，表示整个窗口，并且其中设置了窗口的标题为"Tutorial"，这个窗口内有一个 StackPanel 面板，其中包含一个内容为"Click Me!"的 Button。正如这个简单的例子所示，整个应用的界面正是由 XAML 来描述的。

XAML

> XAML 是一种用来描述用户界面的语法。它取代了传统的使用代码构造界面，而使用声明式的语法来描述界面，实现代码逻辑和界面声明的分离，让设计师能够不编写代码，只需要通过 XAML 来设计界面。

XAML 基于 XML 的语法。如果你已经熟悉了 XML，那么接下来的部分内容对你而言应该是基础知识。

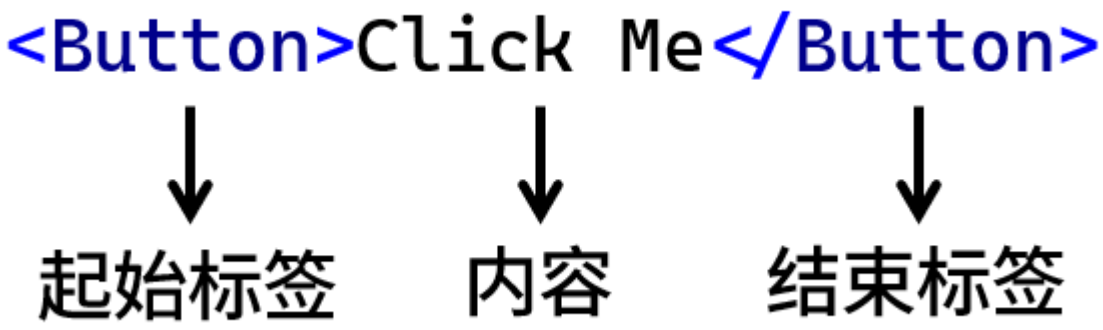
XAML 对象元素

在 C#之中，一切皆对象。XAML 中的对象元素会直接声明 C#中对应类型的实例

```
<Button x:Name="button1">Hello!</Button>
```

简单的一行，我们就声明了一个变量名为 `button1` 的 Button 对象，它大致相当于下面这段 C#代码

```
Button button1 = new Button()
{
    Content = "Hello!"
};
```

> ### 一点 XML 基础知识
>
> **标签：** 在 XML 中，用 `<` 和 `>` 包裹起来的东西（含 `<` 和 `>`）被称作一个**标签**。
>
> **元素：** 一个 XML 元素是由一个**起始标签**和一个**结束标签**，还有它们之间的**内容**构成的：
>
> 
>
>
> 内容并不是必须的。没有内容的元素被称为空元素。

```
<Button></Button>
```

你可以使用自闭合标签来简化空元素的书写：

```
<Button />
```

内容

起始标签和结束标签中间夹着的被称为内容(Content)，在这里它是一段文字"Click Me!"，但也可以是数字，甚至另一个元素：

```
<Button x:Name="button1">
    <Image Source="/Assets/WinUI3.png"></Image>
</Button>
```

这段 XAML 相当于：

```
Button button1 = new Button()
{
    Content = new Image()
    {
        Source = "/Assets/WinUI3.png",
    }
};
```

属性

上面的例子中，`<Image Source="/Assets/WinUI3.png" />` 的 `Source="..."`。
对于一个元素，它对应的类型的可访问属性都能通过这种语法赋值。

内容也是一种属性，叫 `Content`，所以以下两种写法是等价的：

```
<Button>Hello!</Button>
<Button Content="Hello!" />
```

属性元素

如果你希望把一个属性的值设置为一个对象元素，可以使用属性元素语法，以下两种写法是等价的。

属性元素和对象元素一样，也拥有内容。它的内容同样可以是字符串，数字甚至另一个对象元素

```
<Button>
    <Button.FontSize>
        24
    </Button.FontSize>
    Hello!
</Button>

<Button FontSize="24" >
    Hello!
<Button />
```

事件

我们不仅仅能设置控件的属性，还可以设置事件的处理者。Button拥有Click事件，我们可以这样设置它的处理函数为 `ButtonBase_OnClick(object sender, RoutedEventArgs e)` 方法。

```
<Button Click="ButtonBase_OnClick">
        Hello WinUI 3!
</Button>
```

命名空间

C#中所有类都有命名空间，因此在XAML中使用C#的类型或XAML规范的类型，也需要引用命名空间。要引用命名空间，只需要设置xmlns属性：

```
xmlns:别名="命名空间路径"
// 比如 xmlns:controls="using:Tutorial.Controls" ，就可以引入Tutorial.Controls命名空间中的类型，并且起别名为controls
```

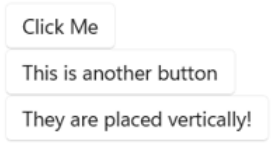
使用时，需要在前面添加别名和一个冒号，表明是哪个命名空间下的类型。

```
<controls:MyCustomControl .../>
```

布局面板

回到 `MainWindow.xaml` ，我们可以试着给界面多添加几个按钮。只需要在 StackPanel 里多添加几个 Button 元素就可以了。

```
<StackPanel
    HorizontalAlignment="Center"
    VerticalAlignment="Center">
    <Button>Click Me</Button>
    <Button>This is another button</Button>
    <Button>They are placed vertically!</Button>
</StackPanel>
```



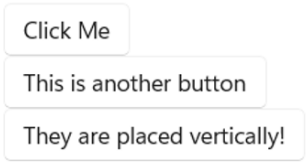
StackPanel

`StackPanel`是一种常见的**布局面板**（layout panel），借助布局面板，你可以自由的排列界面上的元素。布局面板是控件的**容器**，它不具有内容，但是可以有多个子控件，而不是像普通的对象元素的内容只能有一个元素。

```
<StackPanel>
        <Button>Hello!</Button>
        <Button>Hello!</Button>
        <Button>Hello!</Button>
</StackPanel> <!-- 可行 -->

<Button>
        <Button>Hello!</Button>
        <Button>Hello!</Button>
        <Button>Hello!</Button>
</Button> <!-- 错误!  -->
```

位于`StackPanel`里面的控件会垂直或水平排布，可以通过设置其`Orientation`属性来切换。



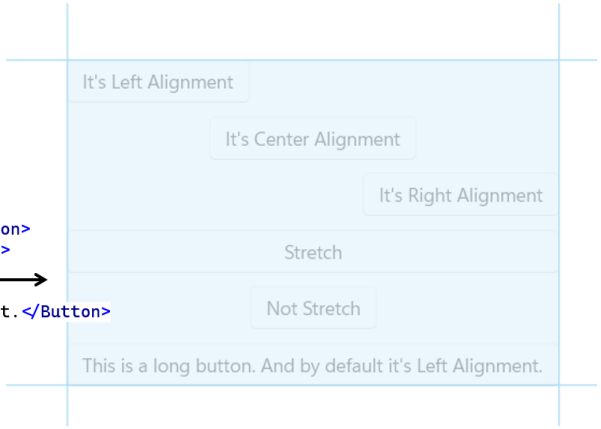
`Orientation="Vertical"`

`Orientation="Horizontal"`

StackPanel还有一些常用的属性，比如Spacing可以设置每一个子元素之间的间距

目前，在这个StackPanel里面的元素都是靠左排列的。我们可以通过HorizontalAlignment和VerticalAlignment修改每个 `Button` 在它所处的容器中的对齐方式

```
<StackPanel
        Spacing="10"
        HorizontalAlignment="Center"
        VerticalAlignment="Center">
        <Button HorizontalAlignment="Left">It's Left Alignment</Button>
        <Button HorizontalAlignment="Center">It's Center Alignment</Button>
        <Button HorizontalAlignment="Right">It's Right Alignment</Button>
        <Button HorizontalAlignment="Stretch">Stretch</Button>
        <Button HorizontalAlignment="Center">Not Stretch</Button>
        <Button>This is a long button. And by default it's Left Alignment.</Button>
    </StackPanel>
```



Grid

还有另一种常见的布局面板网格Grid，能以类似于表格一样的方式完成布局。

使用Grid，首先要确定行与列。

```
<Grid>
        <Grid.ColumnDefinitions>
                <ColumnDefinition Width="250" />
                <ColumnDefinition Width="150" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="3*" />
                <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <Rectangle Grid.Row="0" Fill="Green" Height="100" />
```

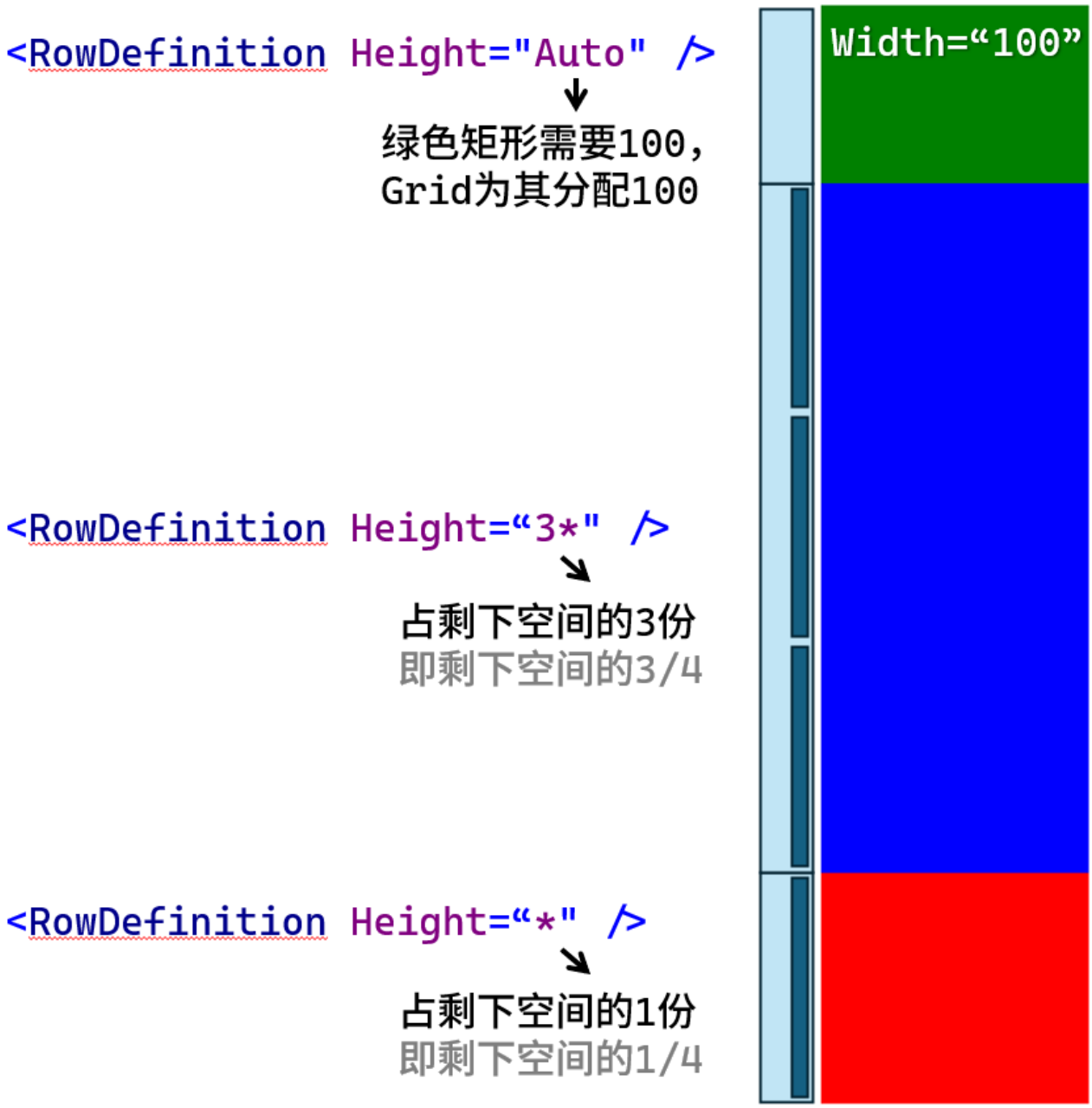
```
        <Rectangle Grid.Row="1" Fill="Blue"  />
        <Rectangle Grid.Row="2" Fill="Red" />
</Grid>
```

在ColumnDefinitions中定义列，RowDefinitions中定义行。

完成行和列的定义后，在Grid的子元素中可以设置属性`Grid.Row`和`Grid.Column`的值来指定该元素将位于哪一行和哪一列。如果不设置，那么默认值为零，即最左上角的方格。

每一列（行）的宽度（高度）可以设置成一个具体的数值，也有两种指定相对宽度（高度）的方法：

- 自动设置为控件需要的大小：`Auto`
- 将剩余空间按比例分配：`*`，`*`前可以带一个数字，表示这一列分配比例。比如有两行`3*`和`*`，那么一列会占剩余空间的两份，另一行会占剩余空间的一份，即3/4和1/4。



控件

WinUI 3的界面由布局和控件构成。控件用来显示数据以及供用户交互，而布局则是描述控件如何排布在窗口上。我们已经接触过了Button控件了，它是一个按钮，可以点击，也可以显示一段文字。在WinUI 3 Gallery应用中， 可以看到WinUI 3内置的绝大多数控件及他们的用法。

不少控件继承自`ContentControl`，拥有内容（`Content`）属性。

后台代码

在新创建的项目模板中，主窗口内有一个按钮。点击这个按钮，上面的文字会发生变化。那么，响应点击事件，并且修改按钮的内容的代码在哪里呢？在传统的UI编程中，UI界面的声明和代码

逻辑混杂在一起，而WinUI 3并不是这样。借助XAML，UI的声明得以分离开，而代码则被放在了单独的一个文件`*.xaml.cs`，即后台代码（code-behind）里面。

打开主窗口的后台代码文件，可以发现后台代码相当简单。

```
using Microsoft.UI.Xaml;

// To learn more about WinUI, the WinUI project structure,
// and more about our project templates, see: http://aka.ms/winui-project-
info.

namespace Tutorial;

/// <summary>
/// An empty window that can be used on its own or navigated to within a
Frame.
/// </summary>
public sealed partial class MainWindow : Window
{
    public MainWindow ( )
    {
        AppWindow.DefaultTitleBarShouldMatchAppModeTheme = true;
        this.InitializeComponent();
    }

    private void Button1_OnClick (object sender, RoutedEventArgs e)
    {
        Button1.Content = "WinUI 3 is Awesome!";
    }
}
```

其中只有`MainWindow`类的声明。注意到`MainWindow`类是`partial`的，这是因为XAML在编译时会生成界面相关的代码。比如构造函数里面的`this.InitializeComponent()`方法，这个方法的定义就是放在`MainWindow`的另一个分部类里面的；除此之外，还会根据设置的`x:Name`属性在`MainWindow`分部类中生成对应的控件的属性。

在XAML里有一个`Button`元素，

- 其`x:Name`为`Button1`，因此会自动在`MainWindow`的分部类里生成一个`private Button Button1;`，因此我们可以在代码中直接访问这个控件。
- `Click`事件的处理函数设置为后台代码中的`Button1_OnClick`。当点击按钮时，就会执行在XAML里设置的这个函数，进而改变按钮的内容。

前面提到了XAML会生成对应的分部类。因此每一个XAML文件都会对应一个类型，这个类型通过在 XAML 的根元素上的 `x:Class` 属性指定。在这里XAML对应的是 `MainWindow` 这个类型，编译时会为 `MainWindow` 生成分部类。