[04.移动先行之谁主沉浮----XAML的探索](http://www.cnblogs.com/dunitian/p/4869835.html)

如果移动方向有任何问题请参考===> [**异常处理汇总-移动系列（点）**](http://www.cnblogs.com/dunitian/p/4867701.html)

移动先行之谁主沉浮？ 带着你的Net飞奔吧！[**链接======>（点）**](http://www.cnblogs.com/dunitian/p/4868607.html)

**XMAL引入**

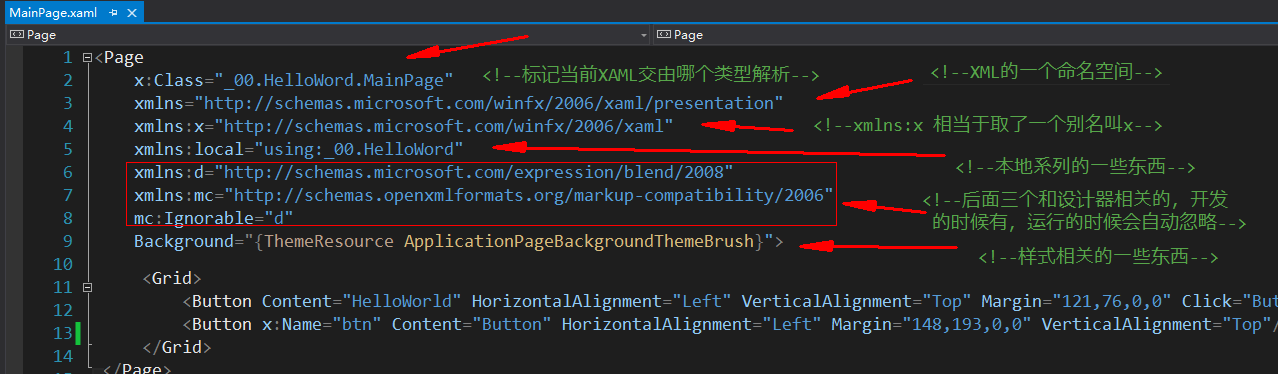
XAML 类似于 HTML，是一种特殊的XML语言

XAML本质上属于一种.NET Programming Language

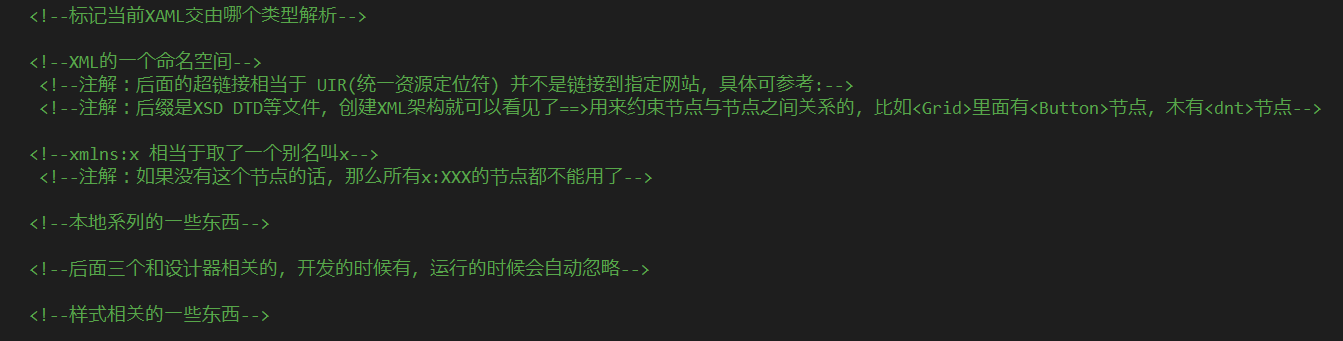
主要应用在 Windows 8、Windows Phone、Silverlight 和 WPF 中

**1.XMAL 命名空间的声明**

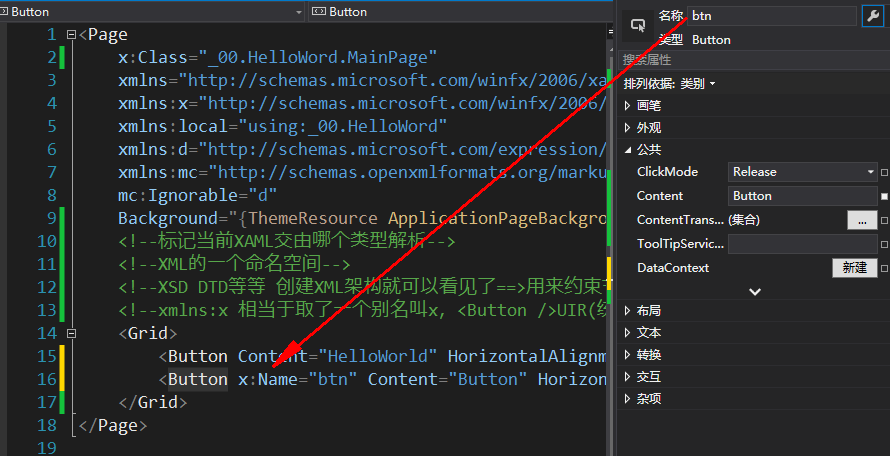
**分析一下头文件**



**详细注解**



**在属性里面改过名字后就会以x:开头**

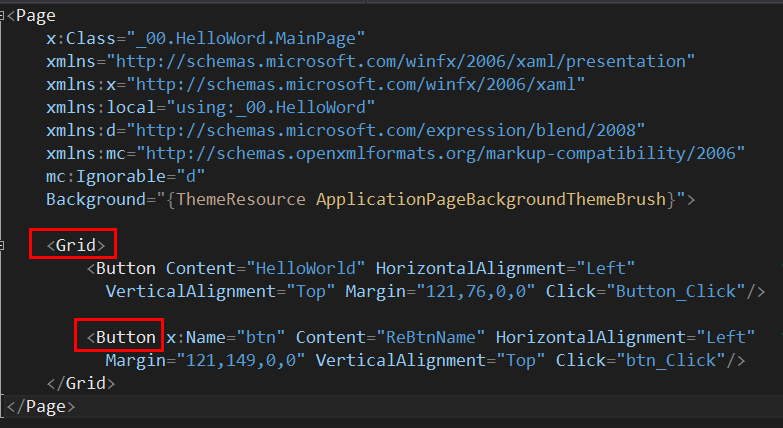


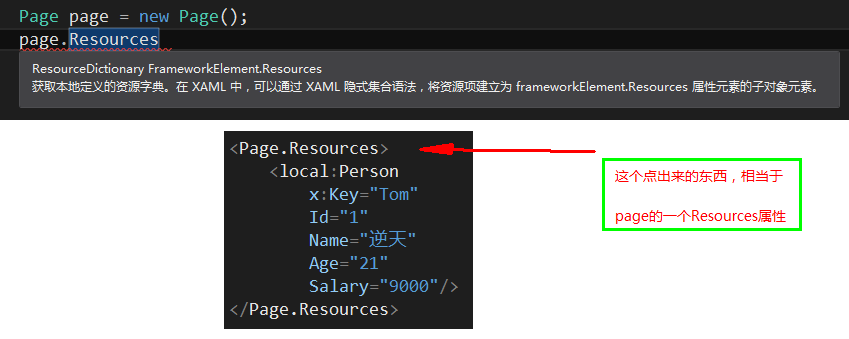
补充：xmlns（XML Namespace）默认命名空间==>xmlns不是XAML中特有或特别定义的，它是XML具备的，目的是为了约束XML的文档结构xmlns:d 和 xmlsn:ms 是为 Visual Studio 设计视图服务的，不会对程序造成任何影响

**2.XAML 创建对象元素**

1.标签名即控件名称

2.一个 XAML 文件只允许有一个根对象元素 Page（类似于xml根节点）

  
3.标签属性或子标签为控件对象属性操作

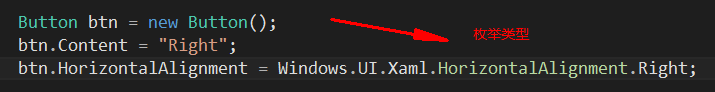
  
4.每个 XAML 标签都会有一个相对应的类型

5.每声明一个 XAML 节点就相当于创建相应类型的对象  
6.在哪个元素结点下添加标签就相当在哪个对象下添加对象

**3.XAML 设置元素对象属性（四种语法）**

**1.键值语法**：依靠内置转换器去实现（可以是文本块，按钮内容，背景图片等）  
　　　　属性键值语法，即 PropName=“PropValue”；  
　　　　绝大多数属性都是使用属性键值的形式来操作；  
　　　　要求属性类型必须为一些“简单的/可以自动转换的”类型（解释转换器带来的效果）

　　比如：**水平对齐方式应该是枚举类型**



**但，键值语法给他赋值的是字符串，同样能达到效果====>可以自动转换（内置属性类型转换器）**

http://images2015.cnblogs.com/blog/658978/201510/658978-20151013100147413-371267132.png

　　补充：

**只有实例化对象才可以设置实例属性**，格式如下：

 　　　　 <objectName propertyName=“propertyValue”/>

  　　　　或者

  　　　　<objectName propertyName=“propertyValue”>

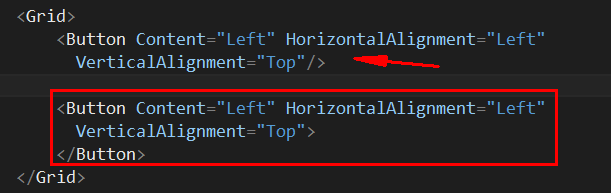
  　　　　</objectName>

　　每个属性对应一个属性值，属性值类型必须与属性匹配

　　一个标记中可以设置对象的多个属性

　　objectName是要实例化的对象，propertyName是需要设置的属性名称，propertyValue是属性的值

　　比如：



　　**2.属性元素语法**；文本块值，按钮内容，背景图片

　　某些属性可以使用属性元素语法来设置，格式为：

  　　<object>

    　　　　<object.property>

       　　　　　<!—元素属性值-->

    　　　　</object.property>

  　　　　</object>

　　如果一个对象的属性是一个复杂类型，不能够由简单的字符串直接表示的时候，这时需要用到属性元素的语法：

　　<标签名>

    　　<标签名.属性名>

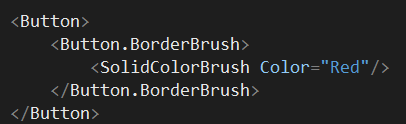
        　　<属性值/>

    　　</标签名.属性名>

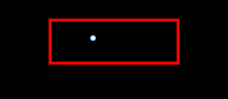
　　</标签名>

　　文本块值，按钮内容，背景图片

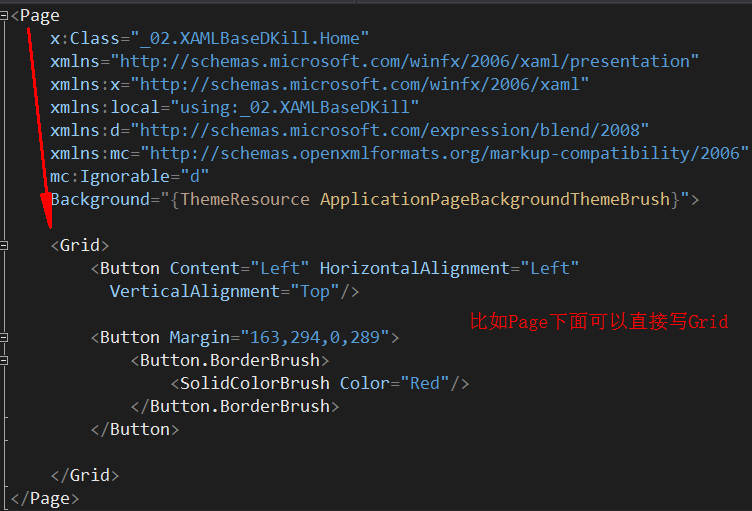
比如：

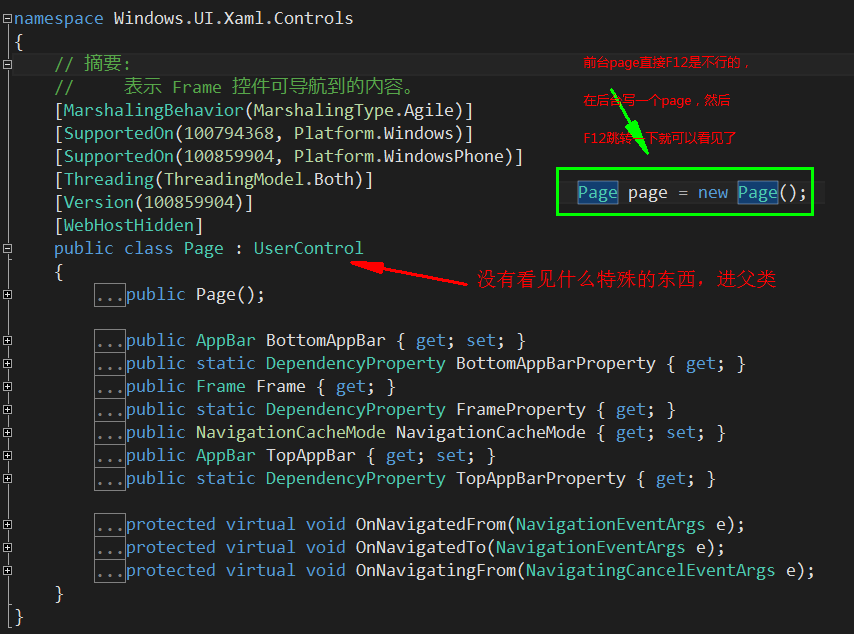


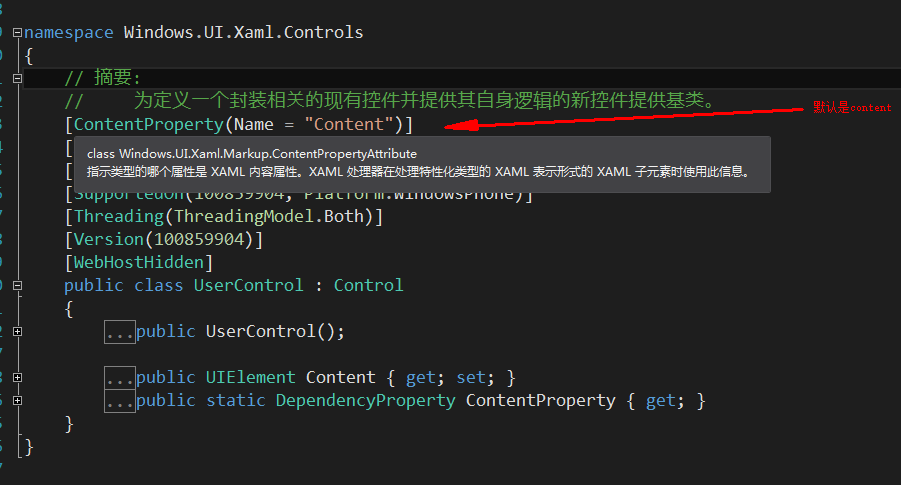
效果：

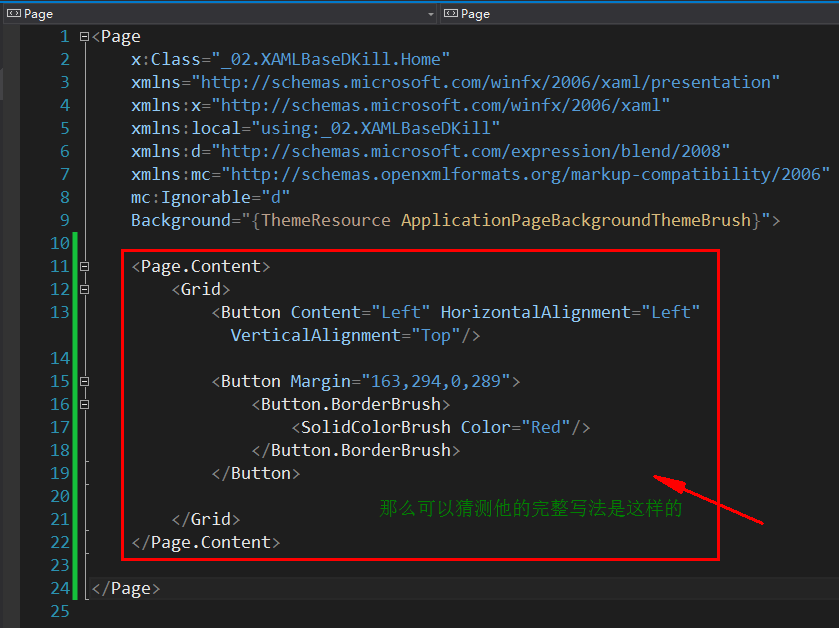


　　**3.默认属性语法**：只是方便，语法糖（看特性标签秒懂）  
　　　　默认属性语法；只是方便语法糖  
　　　　对于大多数对象元素都有一个默认属性；  
　　　　直接将元素的 “InnerText” 设置为属性值：  
　　　　例如：  
　　　　<Button x:Name="btn1" Content="Hello"/>  
　　　　<Button x:Name="btn2">  
　　　　　　World  
　　　　</Button>  
　　　　通过特性标签知道默认属性

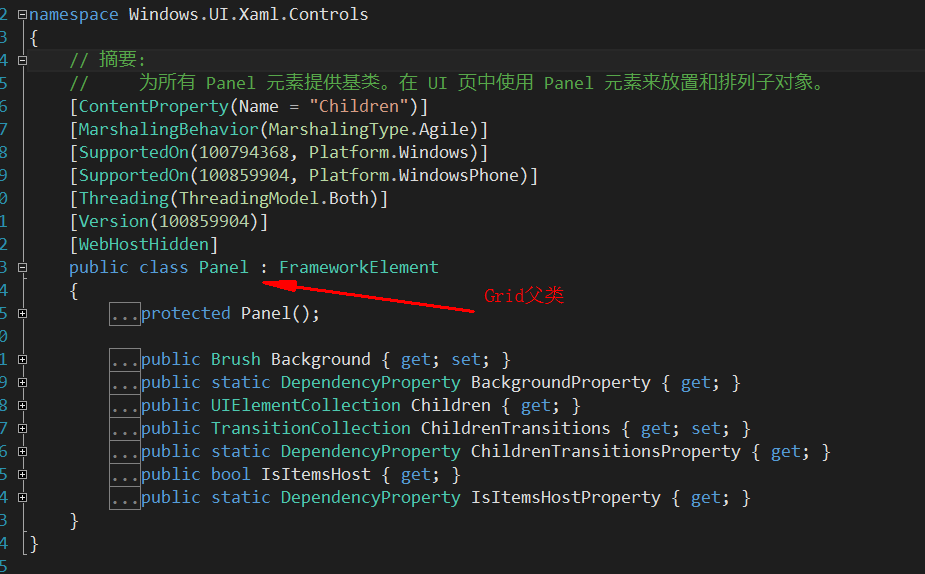




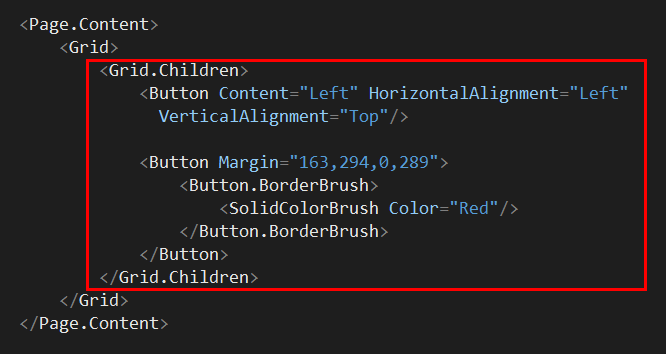




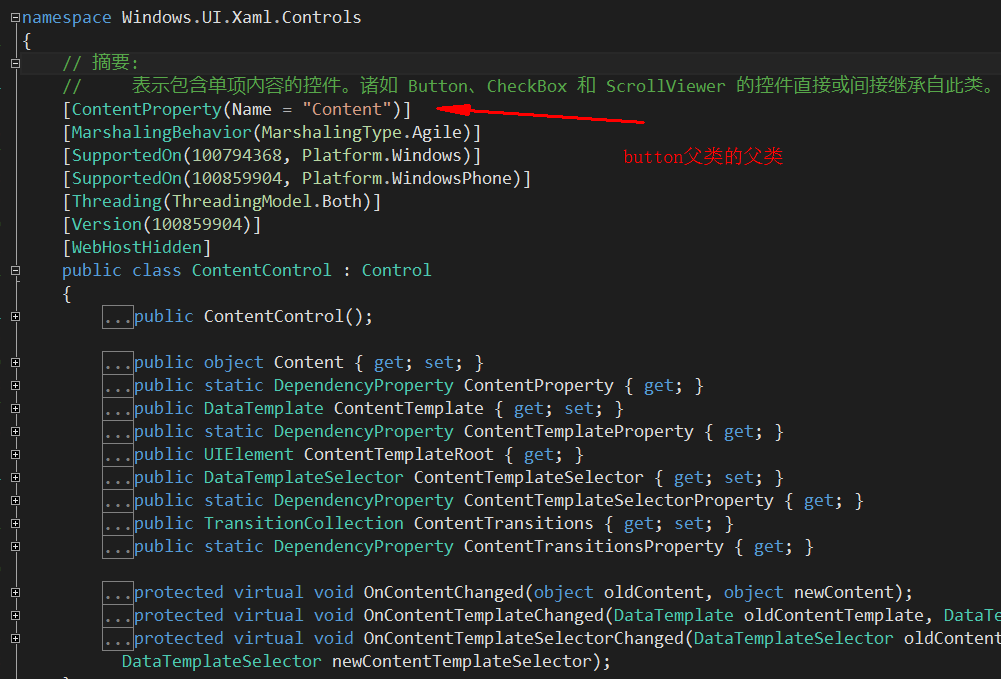
同样的可以看一下**Grid（类似于表格）**的默认属性（进父类）==>**Children**



那么完整写法就是：（当然了，我们平时写是没有必要的，这边只是探索一下本质）



**同样的，我们可以反推==>（刚才正推，现在反推）因为Button有一个默认属性是Content，**



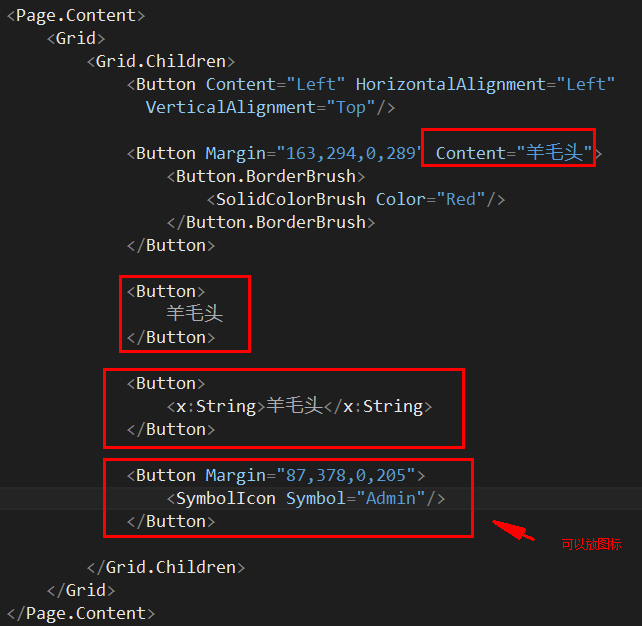
**那么，我们是不是可以用下面这个写法呢？（事实证明正确）**

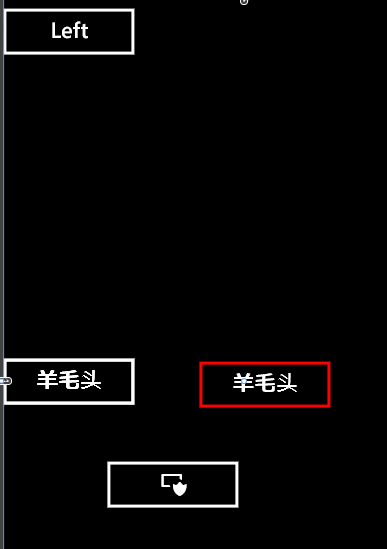
**写法一：<Button><x:String>羊毛头</x:String></Button>**

**写法二：<Button>羊毛头</Button>**



其实里面还是可以放图标之类的，这里就不详解了，来两个图，你们自己练练吧





　　**4.隐式集合语法**；  
　　元素支持一个属性元素的集合，才使用集合语法进行设置属性

　　使用托管代码的Add方法来增加更多的集合元素

　　本质是向对象的集合中添加属性项

　　在此之前我们考虑的都是非集合性质的属性；

　　对于一个集合类的属性可以用重复子元素的方式实现设置值：

　　　　<Grid>  
　　　　　　<Grid.Children>  
　　　　　　　　<TextBlock>Hello1</TextBlock>  
　　　　　　　　<TextBlock>Hello2</TextBlock>  
　　　　　　　　<TextBlock>Hello3</TextBlock>  
　　　　　　　　<TextBlock>Hello4</TextBlock>  
　　　　　　　　<TextBlock>Hello5</TextBlock>  
　　　　　　</Grid.Children>  
　　　　</Grid>

　　不同的属性类型我们可以选择不同的设置方式；  
　　　　简单类型数据，比如字符串、数字之类的使用键值形式；  
　　　　复杂类型数据，比如背景图片使用属性元素形式；  
　　　　集合性质使用集合语法；

**4.名称范围 (x:) 属性**

****

**X:Class**

配置 XAML 编译，在标记和代码隐藏之间连接分部类。代码分部类在一个独立的代码文件中定义，标记分部类由代码是由 XAML 在编译过程间创建。  
<object   
x:Class="namespace.classname"...>  
...  
</object>

**x:FieldModifier**

 修改 XAML 编译行为，使指定对象引用的字段被定义有 public 访问权限而不是默认的 private 行为。

*语法：*<object   
x:FieldModifier="public"  
... />

**x:Key**  
唯一标识作为资源被创建和引用的元素，这些资源存在于一个 ResourceDictionary 中。  
XAML 属性使用方法：  
<ResourceDictionary>  
<object x:Key="stringKeyValue".../>  
</ResourceDictionary>  
隐式 ResourceDictionary 中使用：  
<object.Resources>  
<object x:Key="stringKeyValue".../>  
</object.Resources>

**x:Name**  
唯一标识对象元素，可方便从代码隐藏或一般代码中访问已实例化的对象。应用于支持的编程模型之后，  
x:Name 可视为等效于持有一个对象引用（由一个构造函数返回）的变量。  
就相当于给对象栓条绳子，方便代码访问  
x:Key 和 x:Name 不是相同的概念。  
x:Key 仅用于资源词典中。  
x:Name 用于 XAML 的所有区域。一个使用键值的 FindName 调用不会检索键资源。

**x:Uid**

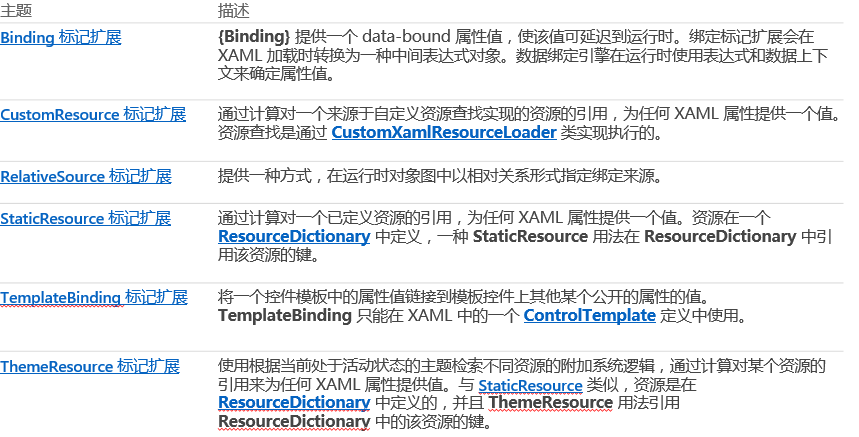
为标记元素提供一个唯一标识符。对于 Windows 运行时 XAML，这个唯一标识符供 XAML 本地化过程和工具使用。  
例如，使用 .resw 资源文件中的资源。  
<Button   
x:Uid="ResourceKey"   
Content="被资源中键为“ResourceKey.Content的值替换”"/>  
开发多语言化系统时会经常用到

**其他 XAML 固有数据类型（就是常见的简单数据类型的声明）**

****

**5.XAML 标记扩展**

可扩展基本 XAML 语言特性的 Windows 运行时 XAML 结构



**1.Binding：**

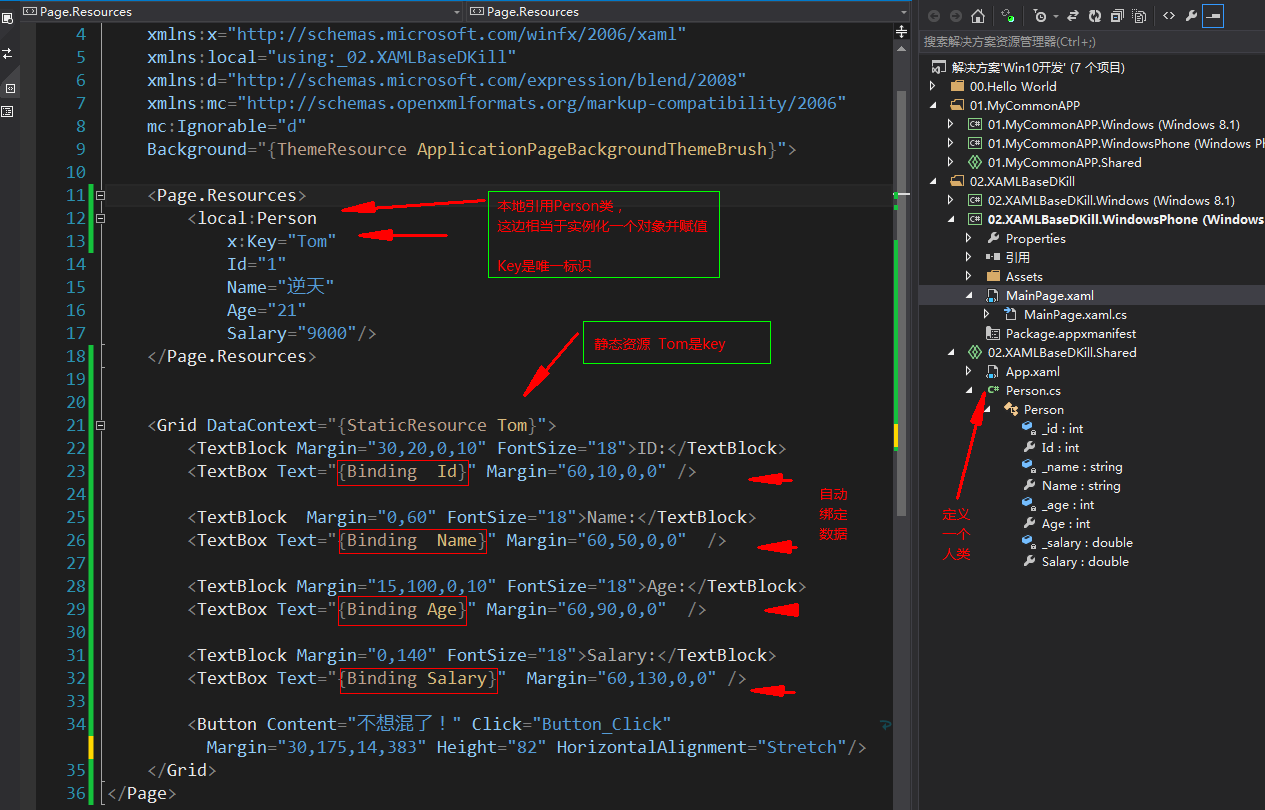
**像我们平时经常用一些自定义类，在XAML里面怎么弄呢？ 这里就用到了Binding了**

**先看一个网络插图：（不要懂，你看箭头部分就好）**



============================

写法如下：详细看图

****

效果



**2.我们再把下面两个说一下，其他的下面的文章里面会慢慢带入的**

**StaticResource 不会实时监视系统主题而跟着变化**  
**ThemeResource 会实时监视系统主题而跟着变化（一般都是一些跟颜色相关的用这个，像字符串之类的就没必要了）**

举个例子：

demo搭建ing

**===================================================================**

**扩展：**

**XAML解析案例：**

**OnNavigatedTo**：**新页面准备的时候**(重写 OnNavigatedTo 方法以检查导航请求并且准备供显示的页面。这个方法就像是初始化(Ini) ,它先于Loaded事件之前被执行,所以在这里可以控制一些初始化前的操作，或初始化的操作。OnNavigatedTo 方法，是在每次页面成为活动(第一次打开时)页面时调用该方法。Silverlight 框架在每次将元素添加到可视化树时引发 Loaded 事件，在激活某一页面时该事件可能会多次发生。)

**OnNavigatedFrom**：**当页面成为非活动的时候 的事件**

**Grid**：**定义由行和列组成的灵活的网格区域**。Grid 的子元素根据其行/列分配（使用 Grid.Row 和 Grid.Column 附加属性来设置）和其他逻辑进行测量和排列。

参考:

<http://www.bubuko.com/infodetail-903282.html>

<https://msdn.microsoft.com/zh-cn/library/hh758292.aspx>

<http://www.cnblogs.com/Jax/archive/2008/04/15/1155135.html>

<http://blog.csdn.net/c979170768/article/details/7017014>

<https://msdn.microsoft.com/zh-cn/library/windows/apps/hh700354.aspx>