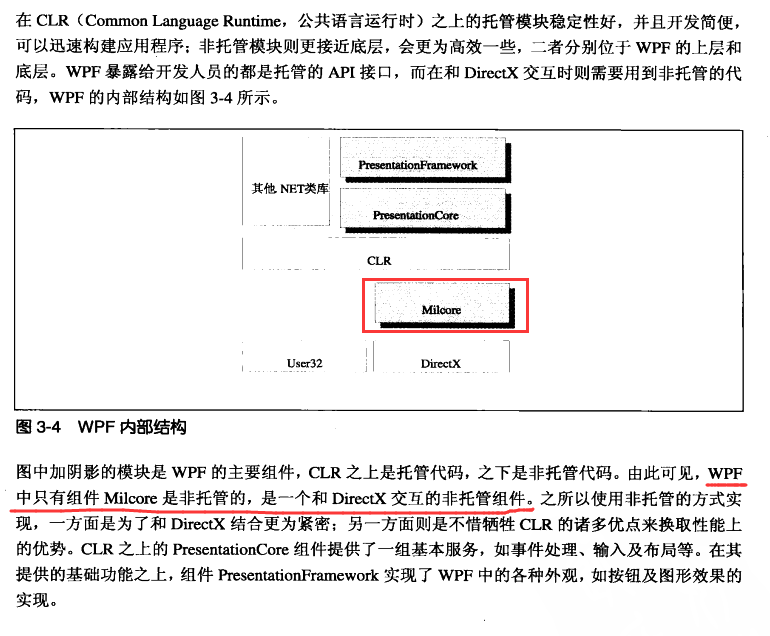
# 一、WPF内部结构

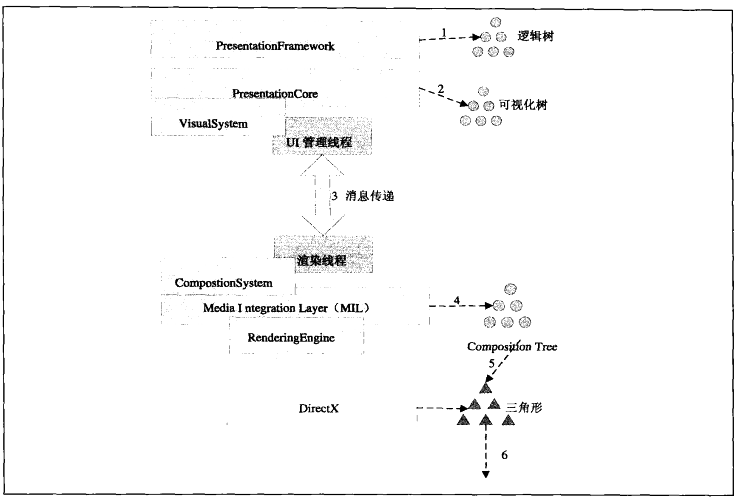


# 二、WPF如何实现绘制

（1）绘制所需数据结构（逻辑树）:WPF将每一个可视化单元都看做是一个Visual，类似早期程序中的窗口，每个Visual也是自顶向下的树（可视化树，Visual Tree），每个节点都会保存绘制自己所需的命令。PS：构成3D图形的基本图元是一个个的三角形

（2）绘制过程

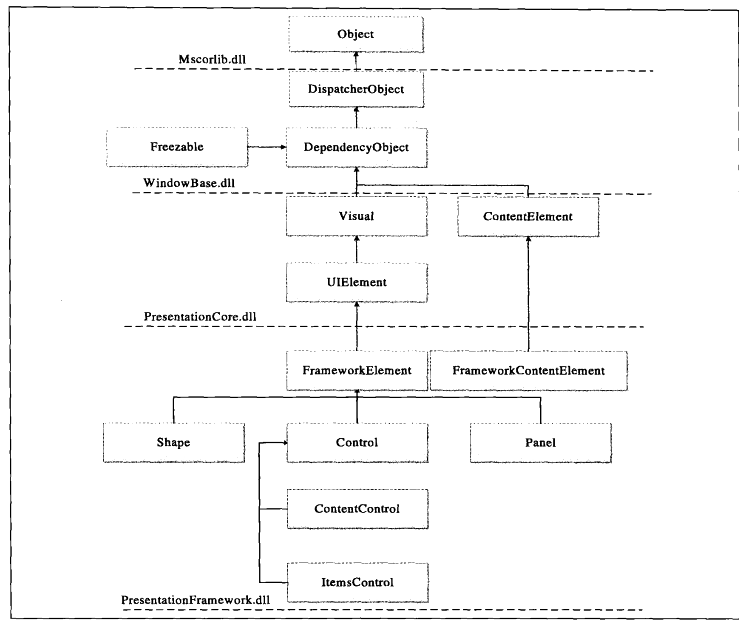
WPF程序从两个线程开始，分别负责UI管理（托管代码，逻辑树、可视化树）和渲染（非托管代码，Composition Tree），二者通过消息传递数据



# 三、层次结构

（1）WPF的类主要集中在WindowsBase.dll、PresentationCore.dll、PresentationFramework.dll

（2）主要类结构



Object

DispatcherObject（System.Treading.DispatcherObject），多线程时可能用到

DependencyObject（System.Windows.DependencyObject），支持依赖属性的基类

Freezable（System.Windows.Freezable），Changed事件，Freeze方法可使自己只读，通常在两个线程共享或提高效率时，需要将对象“冻结”

Visual（System.Windows.Media.Visual），可视化树的一个节点，为WPF提供可视化支持，包括输出显示、透明度、坐标转换及区域剪切等

UIElement（System.Windows.UIElement），增加了布局、输入和事件

FrameworkElement（System.Windows. FrameworkElement），增加了数据绑定、样式及资源等WPF核心功能

Shape（System.Windows.Shapes.Shape），基本形状的基类，继承自FrameworkElement，因此具备一些元素的特性，如响应鼠标和键盘消息等

Control（System.Windows.Controls.Control），控件的基类。WPF的控件可以支持控件模板，使得控件的外观可以随心所欲的变化。WPF中可视化内容被称为“元素”，对控件模板的支持是判别控件和元素一个重要标准

ContentControl（System.Windows.Controls.ContentControl），单一内容控件，文字、图片及视频等任何类型数据

ItemsControl（System.Windows.Control.ItemsControl），多条目显示控件的基类（列表、树形视图），如菜单、工具栏及状态栏

Panel（System.Windows.Controls.Panel），可用作所有布局的容器，是WPF布局系统的基础，合理使用之，是灵活布局界面内容的关键

# 四、XAML

## （1）二要素：元素+属性

## （2）4个根元素：

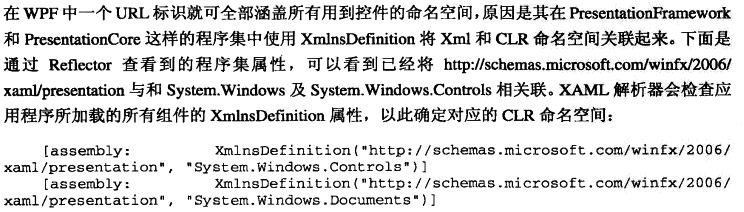
Window（窗口）、Page（页面）、Application（应用程序）、ResourceDictionary（逻辑资源集合），命名空间一般放到根元素中，以便应用到整个文件.

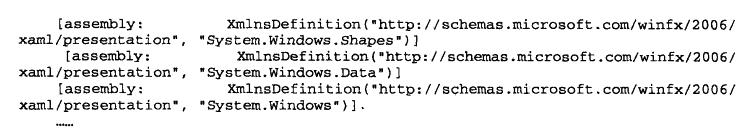
说明：①xaml文件，即便根元素不是以上4种，WPF也会建一个Page对象，然后设置其Content为该xaml对应根元素

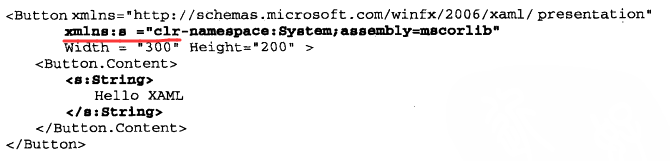
②window对象不能作为元素子元素

## （3）命名空间

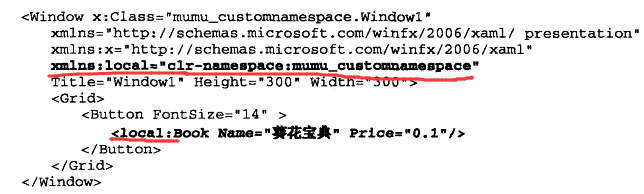
①系统

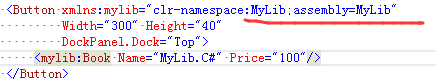






②自定义



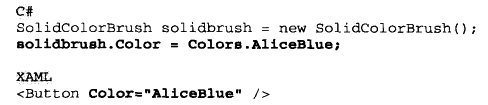


## （4）属性

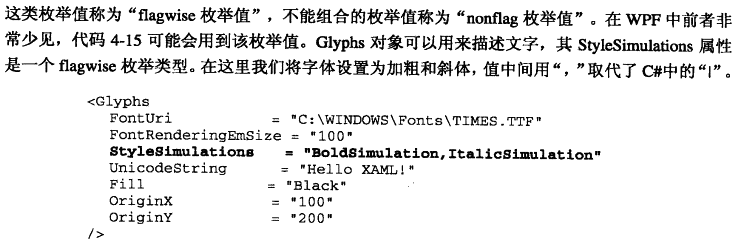
### ①简单属性

如Title、Width，值必须放在引号内

枚举：

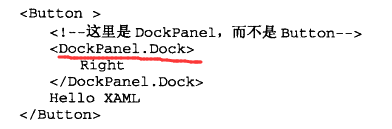


flagwise枚举值



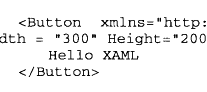
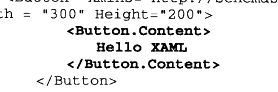
### ②附加属性

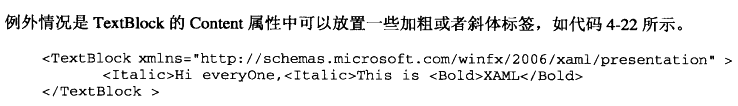
在另外一个类中定义，可用于多个控件



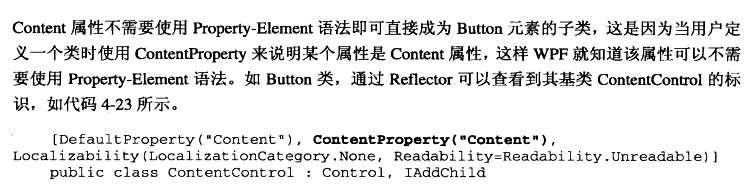
### ③Content属性

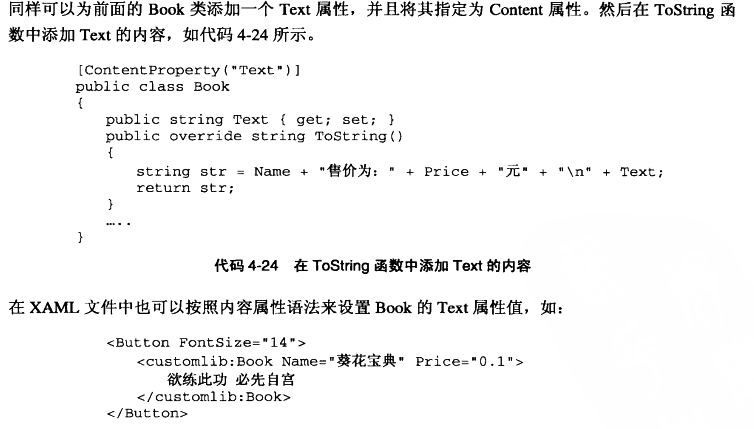
1）写法：





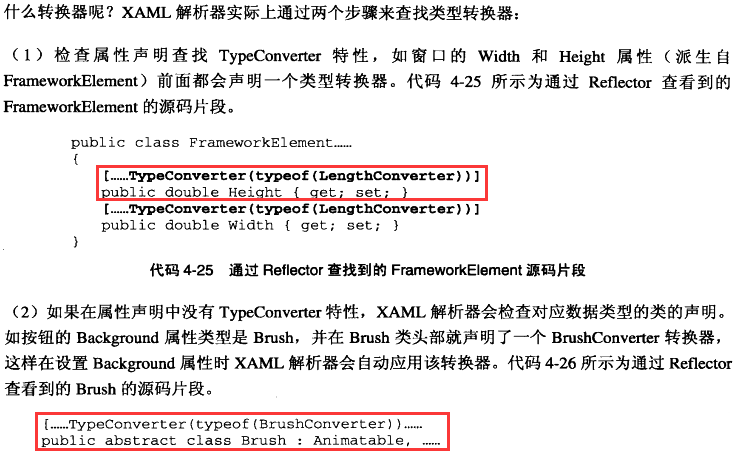
2）



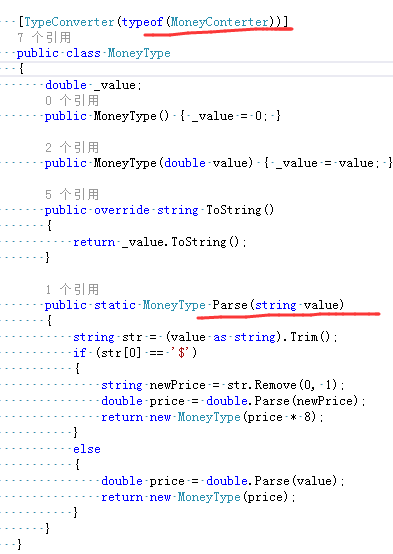


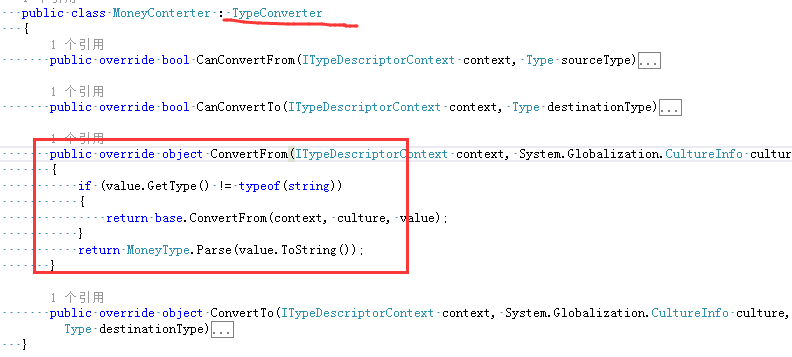
## （5）类型转换器

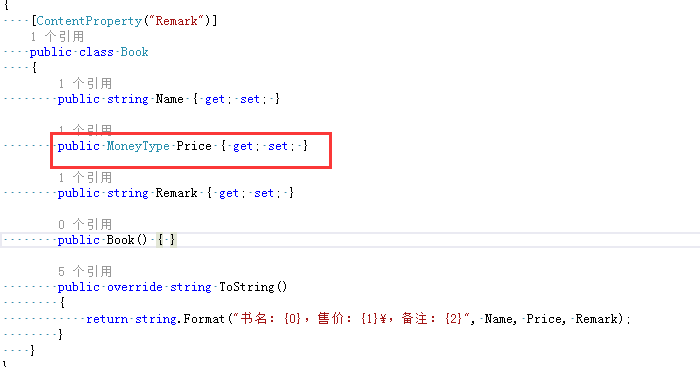
字符串——>类型转换器——>CLR对象

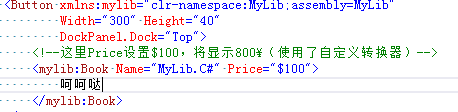


自定义：



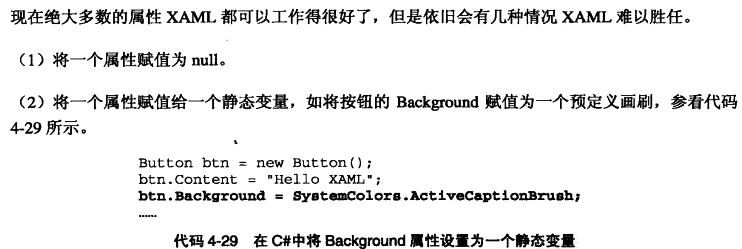


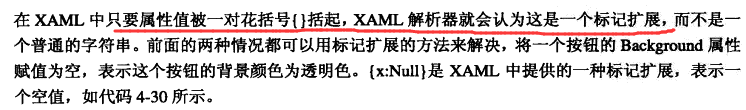




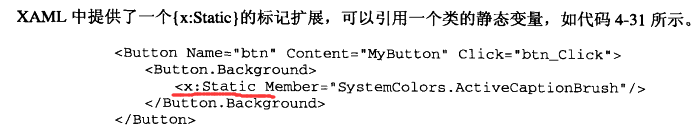
## （6）标记扩展

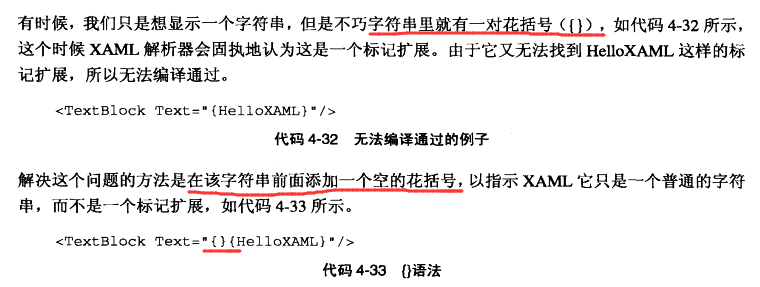
问题：



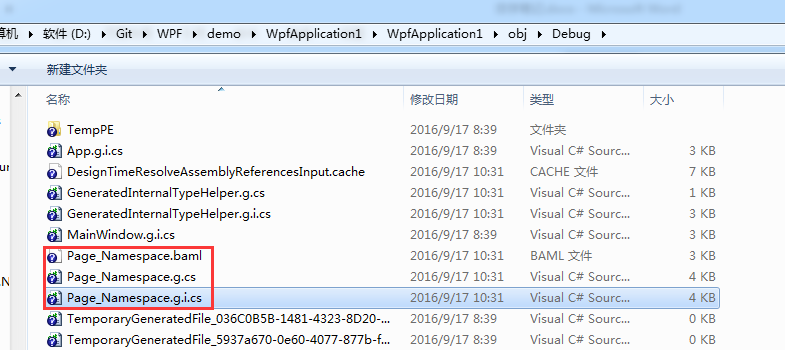








## （7）WPF工作原理（关联XAML和代码文件）

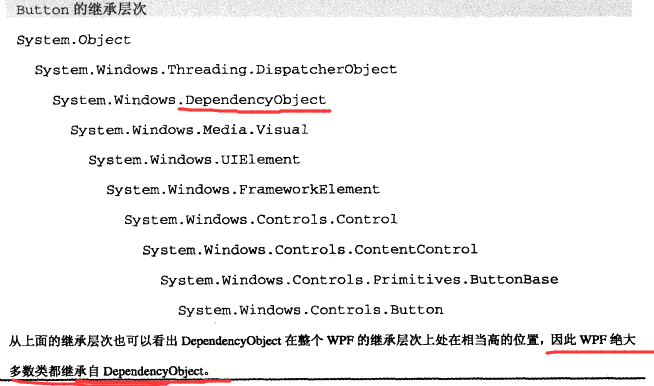


# 五、依赖属性

依赖属性是一种类型为DependencyProperty的属性，其依赖属性标识（Dependency property identifier）则是依赖属性的实例

## （1）DependencyObject

继承该类后才可以注册和拥有依赖属性



## （2）WPF属性系统：

WPF提供一系列的服务（这些服务总称WPF属性系统）扩展了普通的.NET属性

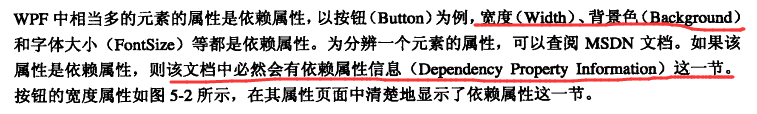
## （3）.NET属性包装器

指属性的get、set实现。

它可以把依赖属性包装成普通的.NET属性暴露给用户使用，这个实现均调用DependencyObject的GetValue和SetValue方法

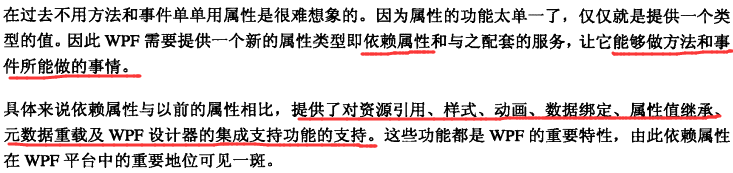
依赖属性更像一个计算过程，整个计算过程依赖其他属性和内在外在的多种因素

## （4）分辨依赖属性

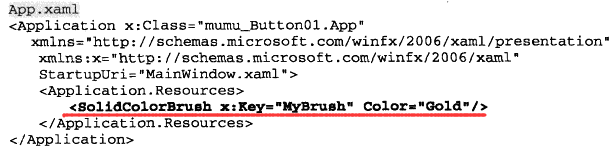


## （5）依赖属性引入原因

WPF设计思路之一：侧重属性胜于方法和事件

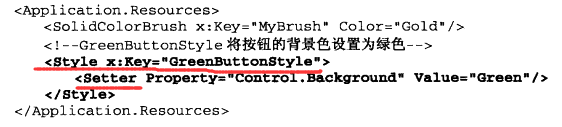


### 依赖属性对资源引用的支持





### 对样式的支持

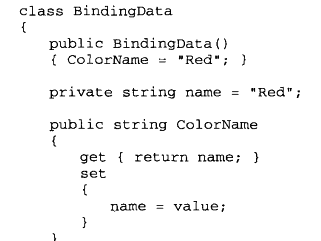




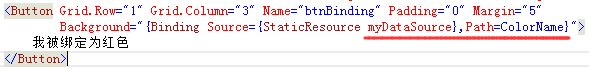
### 对动画的支持？



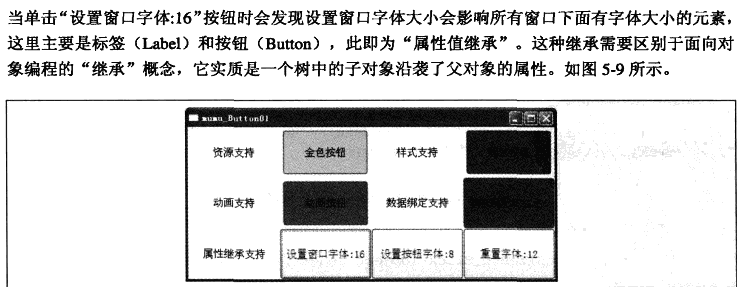
### 对数据绑定的支持







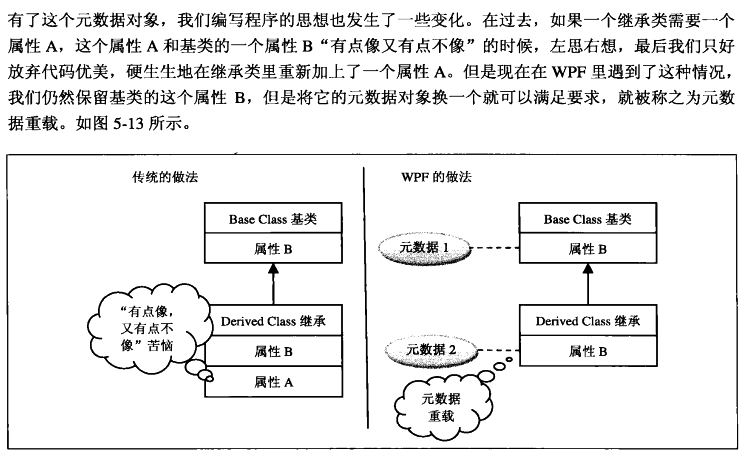
### 对属性值继承的支持



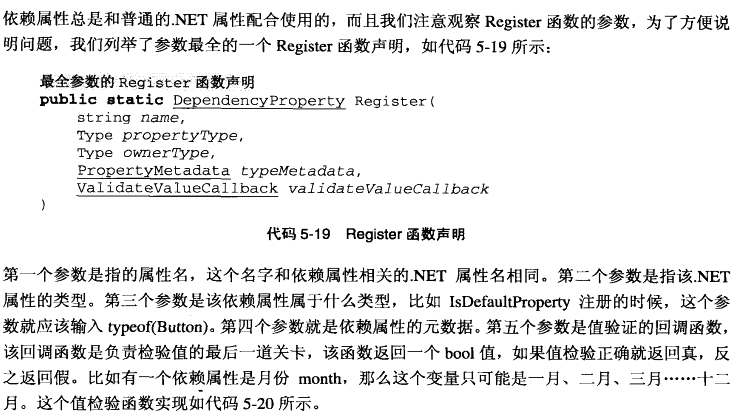
### 对元数据重载的支持

依赖属性和普通.NET属性区别之一是有一个元数据对象，元数据和依赖属性是一对一的关系。

元数据重载：



## （6）依赖属性组成部分



包含：

依赖属性变量

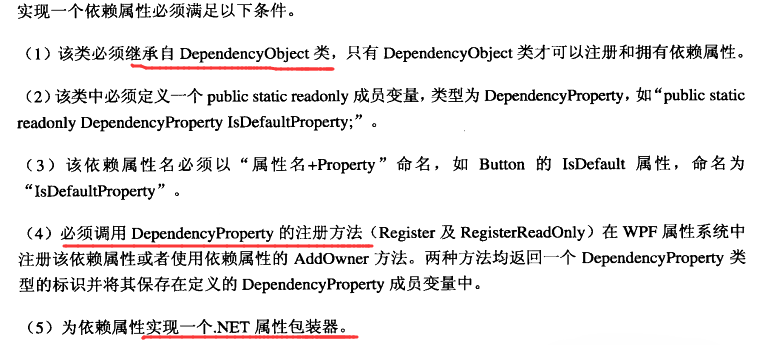
普通的.NET属性

.NET属性包装器

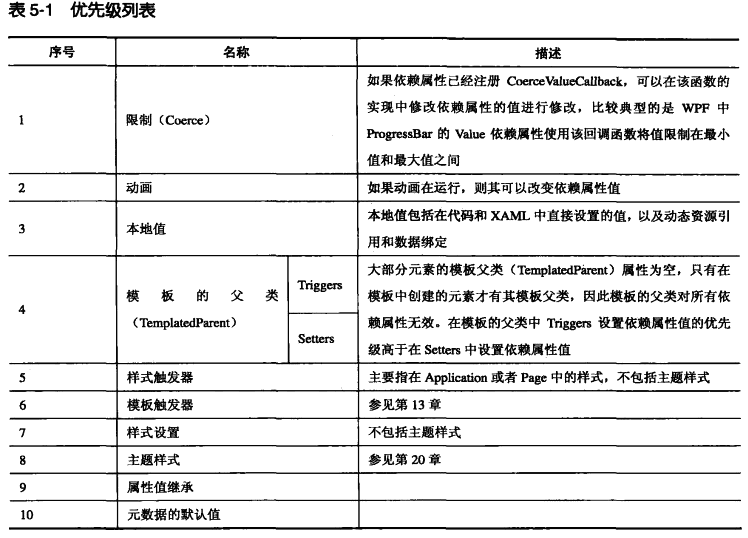
元数据

值验证函数

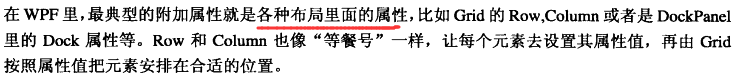
## （7）自定义依赖属性

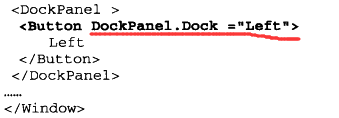


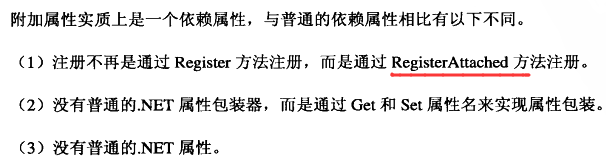
## （8）优先级

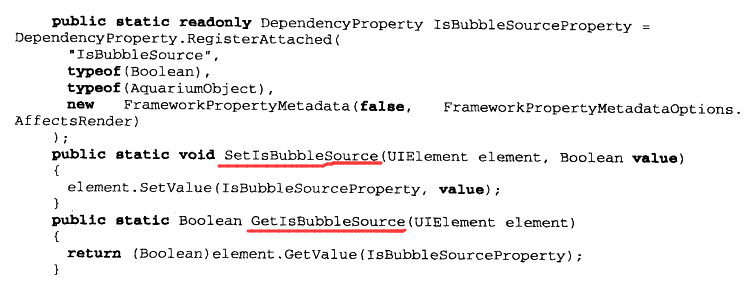


## （9）附加属性







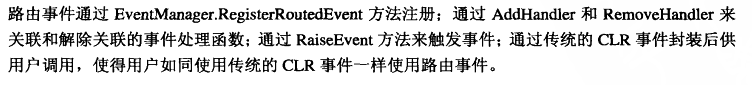


# 六、路由事件

## （1）定义

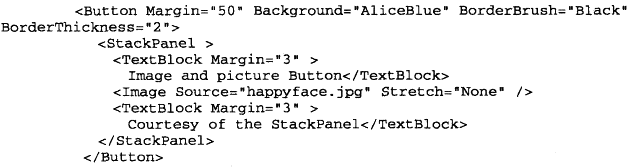


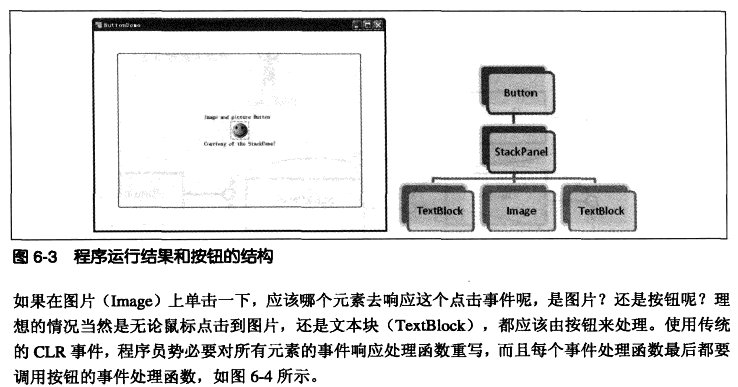


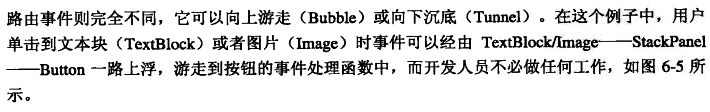


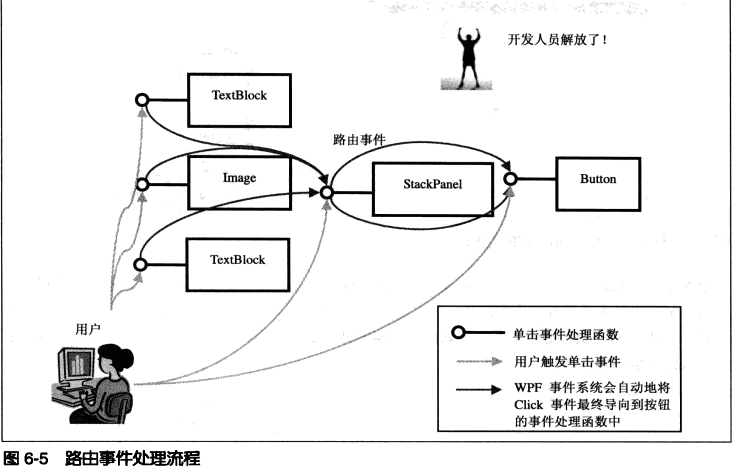
## （2）作用

WPF中的几乎任何一个界面元素都可以任意嵌套，且装配简单



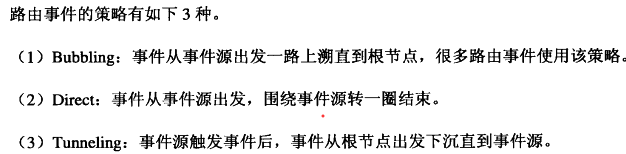


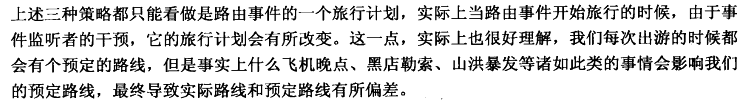




## （3）路由策略

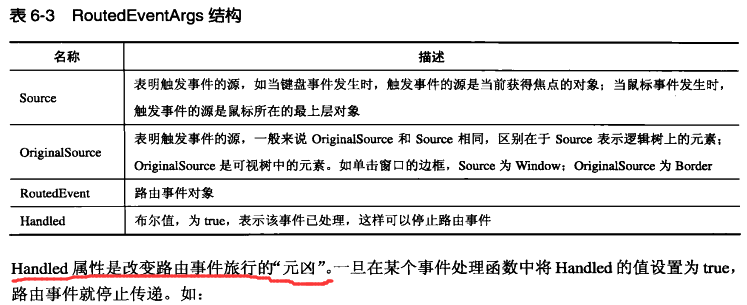
### 三种策略



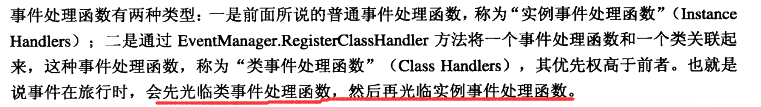


### 策略影响因素——事件处理函数



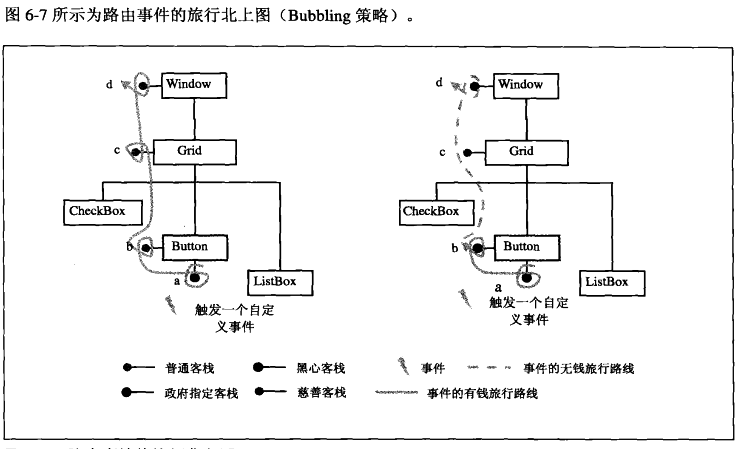


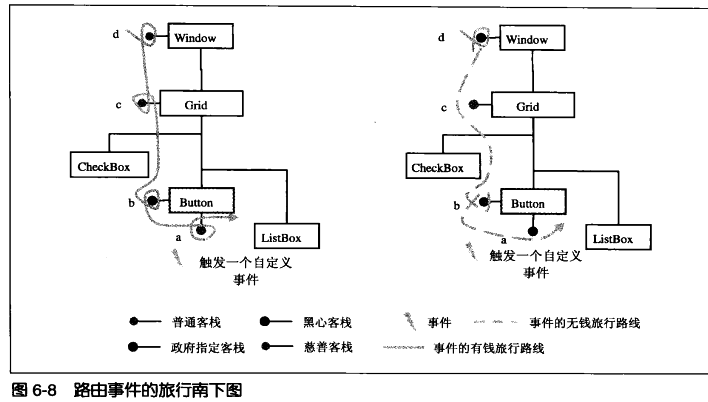
### 策略影响因素——类和实例事件处理函数

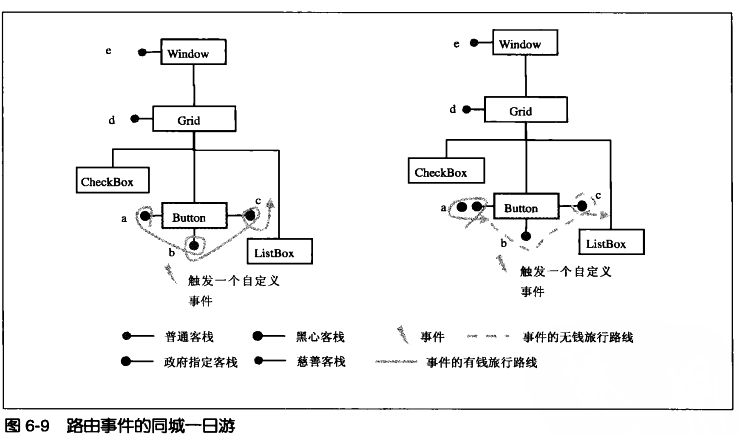


### 实际策略

事件在旅行时有两种状态：未处理、已处理





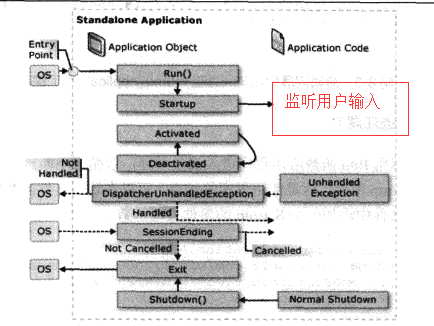


# 七、WPF命令

# 八、应用程序

## 1、应用程序Application

在应用程序中全局维一，通过Application.Current获取



### ①Startup事件

应用程序的第一个事件，可以在这里创建窗口（也可在xaml中给Application的StartupUri属性赋值）

### ②两种状态：激活、非激活

一般，桌面上只有一个窗口处于激活状态，可以监视Application的Activated和Deactivated（应用程序关闭时也会触发）事件

### ③DispatcherUnhandledException

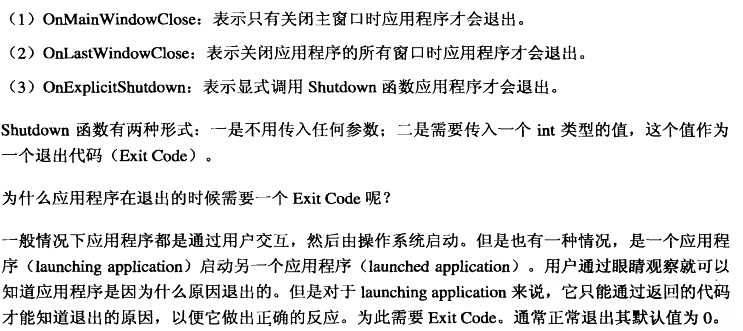
WPF在程序出错时会触发一个DispatcherUnhandledException事件

### ④非正常退出

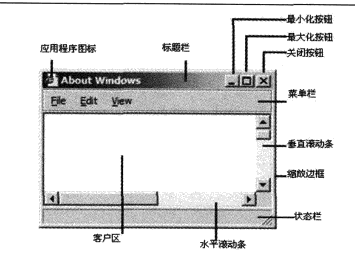
关机、注销或重启，触发SessionEnding事件，设置e.Cancel=true可阻止操作系统的操作

### ⑤正常退出

退出模式Application的ShutdownMode属性， 有三个值



## 2、窗口Window



### ①生命周期

SourceInitiated，窗口第一个事件

Activated：

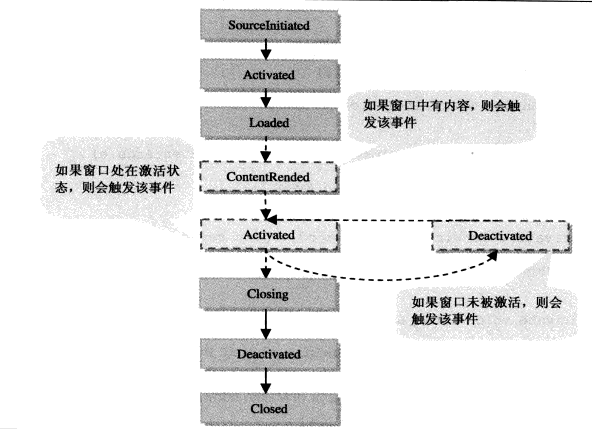
Loaded：已初始化完毕

ContentRended：如果窗口Content属性为空或客户区没有内容，该事件不触发

Deactivated：处于非激活状态时触发

Closing：窗口关闭前，可以阻止关闭

Closed：窗口生命周期已结束



### ②窗口属性

#### 位置和尺寸相关

WindowStartupLocation

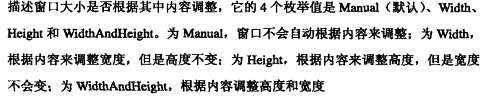
WindowState（最大化等）

MinWidth、MinHeight、MaxWidth、MaxHeight

Width、Height

ActualWidth、ActualHeight：窗口实际尺寸，只读

SizeToContent：窗口大小是否随内容调整



窗口高度、宽度优先级：



#### 外观和样式相关

##### ResizeMode：

NoResize，不能调大小，没有最大化、最小化按钮

CanMinimize，可最小化，不能最大化，不能调大小

CanResize，无限制

CanResizeWithGrip，类CanResize，只是右下角多了个这

##### WindowStyle

None：没有任何边框， 包括最大化最小化关闭按钮

ToolWindow：只有关闭按钮，无最大化、最小化

注：都能调大小

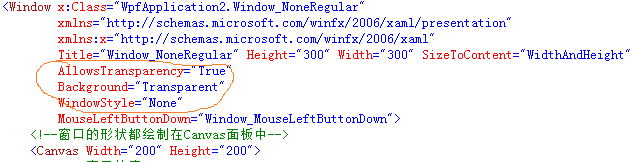
##### ShownInTask

##### Topmost

### ③非规则窗口

非规则窗口，还是矩形窗口，只不过背景设置为了透明

透明的背景色，不接受鼠标的检测，牛逼



# 九、页面和导航

## 1、两种形式：

A、将导航内容寄宿于窗口中，入词典、MSDN或文件资源管理器等

B、XAML浏览器应用程序（XAML Browser Applications，XBAP），扩展名为.xbap，可直接在浏览器运行

## 2、4个核心

### （1）Page

WPF将导航的内容封装为多个页面

Page比Window精简，没有Show、Hide等方法，需要通过导航实现切换

#### 尺寸

一般不设置自身尺寸，由宿主窗口（普通窗口或浏览器）决定，Page可通过设置后WindowWidth、WindowHeight和WindowTitle改变宿主窗口的对应属性

如果设置了页面的Height、Width，则页面有相应的大小，如果宿主窗口小，页面内容将被裁减，如果宿主窗口大，页面将居中显示

#### 宿主窗口

浏览器、NavigationWindow和Frame，后两者提供了从一个Page导航到另一个Page的功能，并能够记录导航历史，和一系列的导航事件。

##### NavigationWindow

多了一个导航工具栏，可以通过ShowsNavigationUI控制其是否可见

顶层窗口，不允许嵌套到其他控件

##### Frame

轻量级，可嵌入其他控件，包括Frame，默认无导航工具栏，可通过NavigationUIVisibility设置

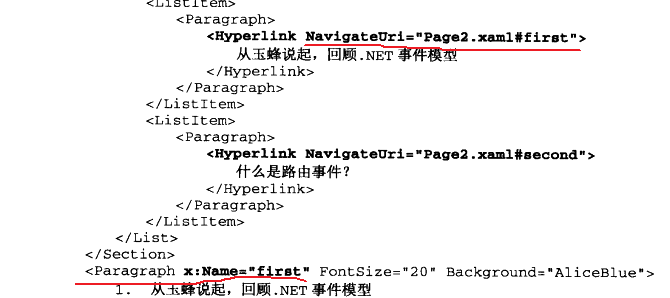
### （2）超链接（Hyperlink）

页面之间通过超链接连接，派生自Span

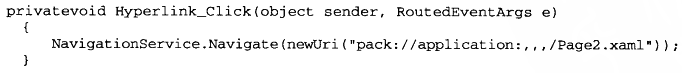
#### 页面间导航



#### 同页面间导航



#### 编码导航



适用场景：

导航前传递参数

导航前设置属性

动态导航

备注：

1、若页面寄宿的不是NavigationWindow，而是一个普通Window，则超链接无法导航

2、Pack URI：



### （3）NavigationServices

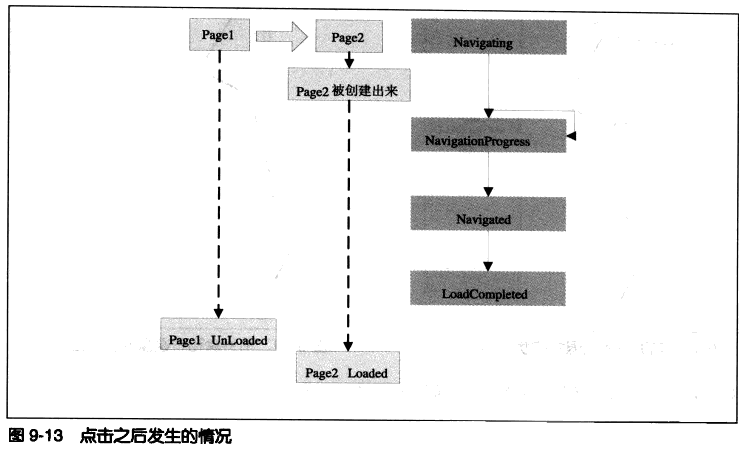
页面之间连接的另一种方式，编程实现，其实是超链接的底层实现

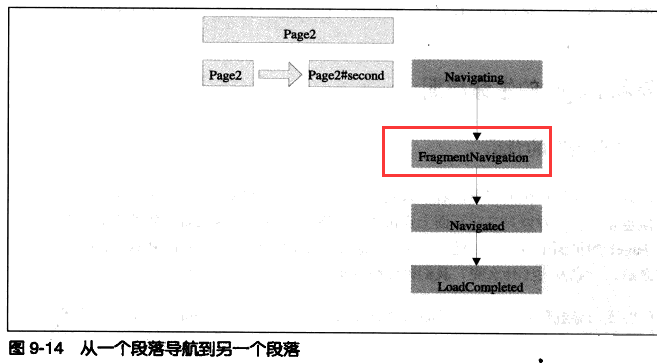
### （4）Journal

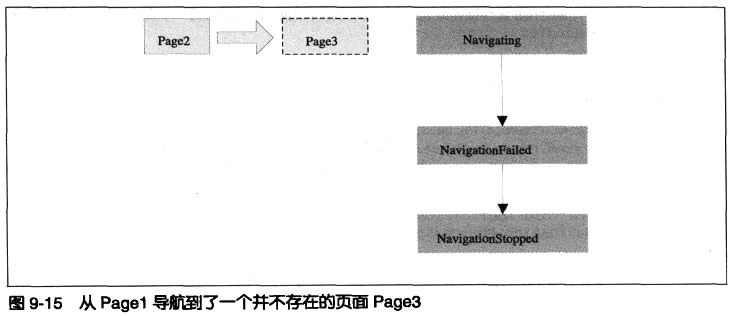
历史记录

## 3、导航和Page的生命周期

### （1）导航过程



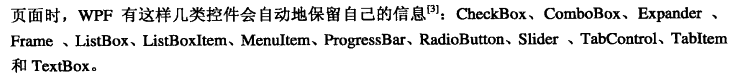




### （2）页面状态保留和数据传递？？

保存全局信息的最好地方是App类，因为一个应用程序对应的App是唯一的，并且所有页面可以通过App.Current访问

1）WPF固有的页面状态保留机制（未验证？）



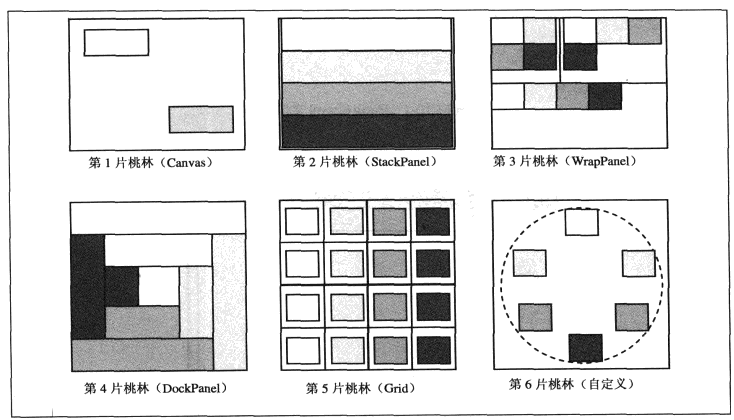
2）使用依赖属性保留简单页面状态信息

WPF页面状态保留机制：从Cookie或Journal查找状态存储

3）使用PageFunction

4）使用IProvideCustomContentState接口保留复杂页面状态信息

# 十、布局



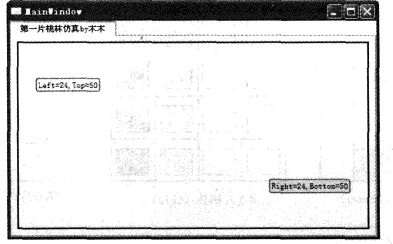
## 1、Canvas

基本面板，仅仅支持用与设备无关的坐标来定位元素。

用4个附加属性Left、Top、Right和Bottom来定位元素

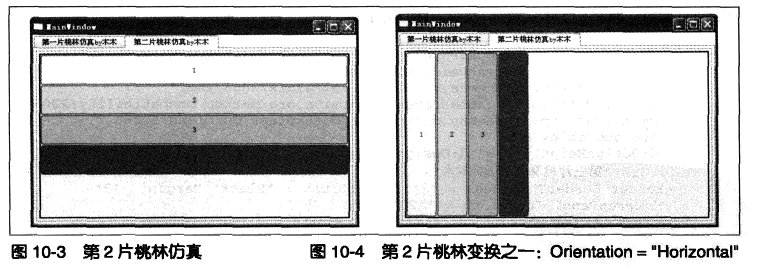
只允许设置一个横向和纵向成对的坐标如Left和Top，以及Right和Bottom等，如果同时设置Left和Right或Top和Bottom，则忽略一个值

特点：原始，效率高，适合矢量绘图



## 2、StackPanel

顺序垂直或水平排列子元素，通过Orientation控制水平还是垂直，默认纵向

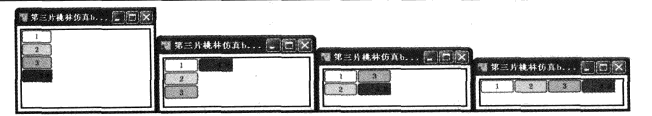


## 3、WrapPanel

除了具有StackPanel的功能外，当没有空间放置子元素时，会自动换行（列），随窗口大小的改变，子元素会自动变换位置

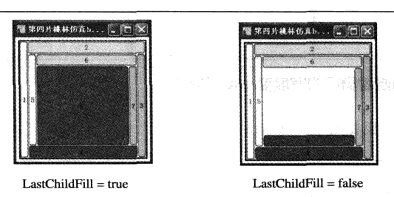
适用于元素个数不确定的情况

属性：Orientation、ItemHeight（允许子元素最大高度，超出即被截断）、ItemWidth

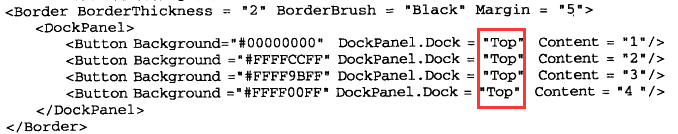


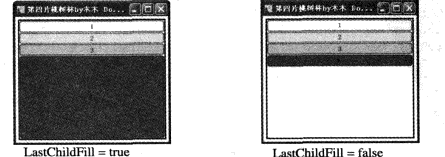
## 4、DockPanel

Dock附加属性，Left、Top、Right、Bottom，没有Fill，默认最后一个添加的子元素会填满所有剩余空间，除非LastChildFill设置false



DockPanel模拟StackPanel





## 5、Grid

### （1）常用附加属性：

Row、Column、RowSpan、ColumnSpan

### （2）行列尺寸3种单位：

绝对尺寸：设置Height、Width一个设备无关的值，当Grid尺寸改变，行列大小不变

自动尺寸：设置Height、Width为Auto，行高为最高元素的高度，列宽为最宽元素的宽度

比例尺寸：设置Height、Width为“3\*”（30%）的特殊语法，行列会根据Grid按比例缩放

①一个行的Height或列的Width设置为\*时，将占据所有剩余空间

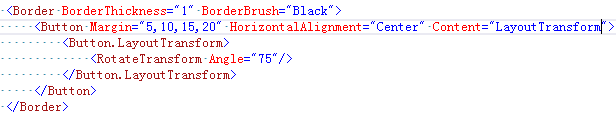
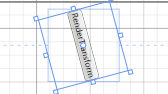
②多行或多列使用\*时剩余空间等分

③行或列在\*之前加一个系数，如2\*、6\*，表示所占比例

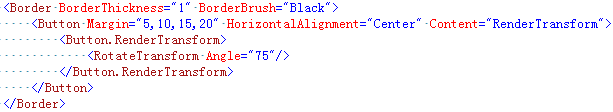
### （3）通用布局

LayoutTransform参与布局，RenderTransform不参与布局，应用RenderTransform的元素会扩展到面板的边界

LayoutTransform：

RenderTransform：

### （4）自定义布局

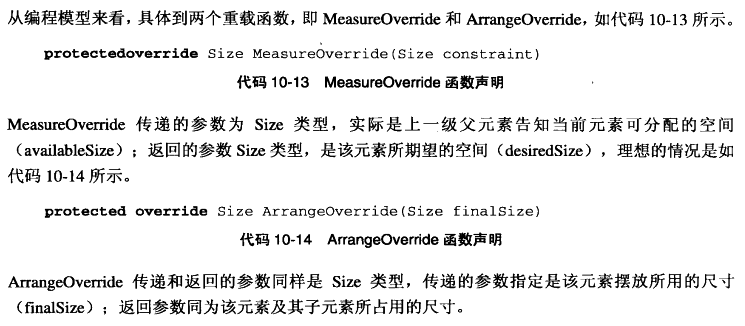
#### 布局两个阶段

##### 测量阶段

父元素询问子元素所期望的尺寸，从而确定自身尺寸

##### 布置阶段

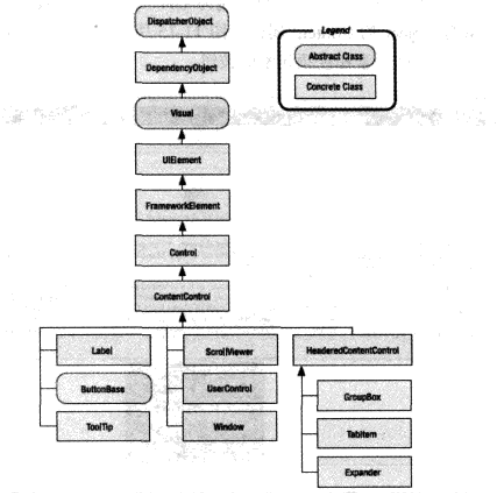
父元素告知子元素的尺寸和位置



# 十一、控件与Content

## 1、Content模型

从ContentControl继承的类均包含一个Content属性



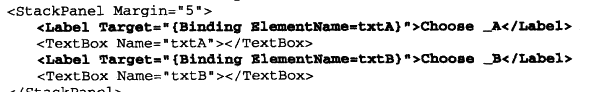
TextBlock是一个轻量级的文本控件，比Label更轻量

## 2、经典控件

### （1）Content控件

#### ①Label

支持以键盘快捷键的方式获得焦点，可以使得与其有紧密关系的控件获得焦点

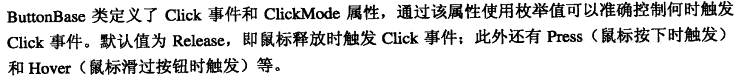


WPF下通过下划线\_设置快捷键的字母，如“\_A”

#### ②Button

可单击，不可双击

##### ButtonBase



##### Button

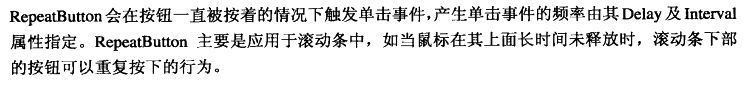
IsDefault，如果true，即使无焦点，按下回车也会触发其Click事件

IsCancel，如果true，则安息ESC将触发Click事件

##### RepeatButton

属于System.Windows.Controls.Primitives命名空间，其提供控件的基类或组成更复杂控件的基本控件

RepeatButton一般不单独使用，用于组成更复杂的控件



##### ToggleButton

单击时可保持状态

IsChecked，第1次单击，true，再次单击，false

若IsTreeState为true，IsChecked将有true、false、null三个值，分别对应Checked事件、UnChecked事件、Indeterminate事件

##### CheckBox

除了外观，其它同ToggleButton

##### RadioButton

继承自ToggleButton，支持互斥性

两种分组方式：

①放在一个StackPanel中即为一组

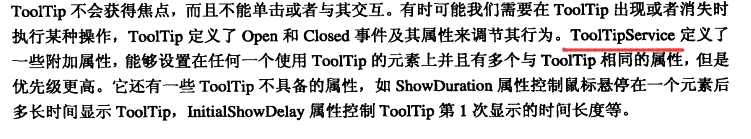
②指定GroupName

##### ToolTip

不能直接放在UI元素树中，必须被赋给另一个元素的ToolTip属性

不能获取焦点、单击或与之交互

定义了Open、Close事件

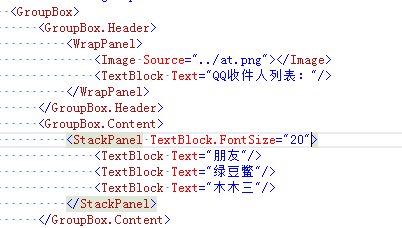


### （2）HeaderedContent控件

#### ①GroupBox

只能直接包含一项子元素（以为是内容控件）

其Header和Content属性可以被设置为任意类型的对象



#### ②Expander

类似GroupBox，包含一个按钮，可展开或折叠

IsExpandered属性

Expanded、Collapsed事件

ExpandDirection，控制方向

### （3）Items控件

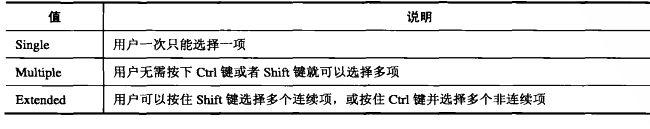
Items属性

ItemsSource属性，可将任意类型的集合赋给Items集合，多用于数据绑定

#### ①ListBox

可动态绑定

SelectionMode



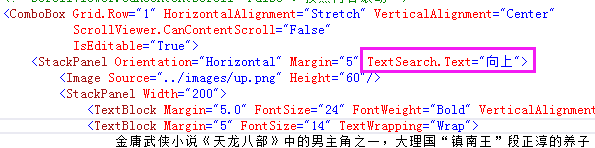
#### ②ComboBox

DropDownOpened、DropDownClosed事件

IsEditable，设置是否输入

ScrollViewer.CanContentScroll="False"：按照内容滚动

TextSearch.Text="向上"，可指定输入定位和定位后显示



#### ③Menu

由于TextBox控件内置对Command的支持，子菜单项只需要设置相应的Command命令即可

IsChecked属性，可以让子菜单具有CheckBox的特性，菜单相应提供Checked和UnChecked事件

右键菜单，ContextMenu

#### ④Range

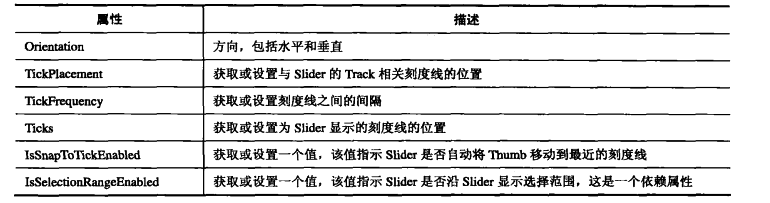
##### ProgressBar

Minimum、Maximum、Value

IsInDeterminate为true，进度条将以一定周期不停运转，这时不必设置Minimum、Maximum、Value属性

##### Slider

音量调节等场景

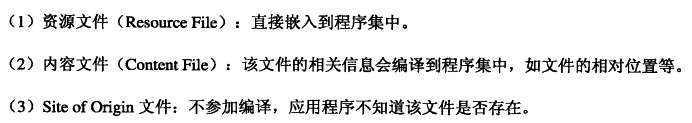


# 十二、资源

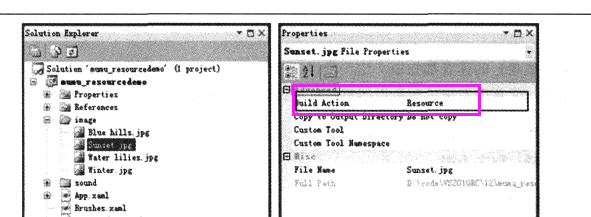
## 1、程序集资源

Xaml、图片、音频、视频等文件

3种打包方式



### （1）资源文件



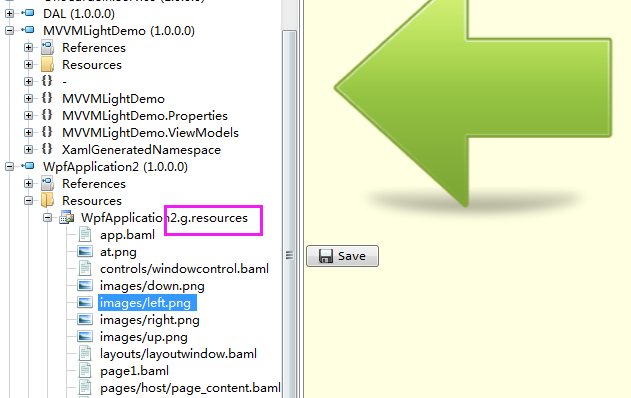


①嵌入的资源和资源

前者用于在Winform项目中嵌入程序集资源，WPF中选择后者

②资源文件可以通过URI规范访问

③查看资源文件内容（反编译工具）

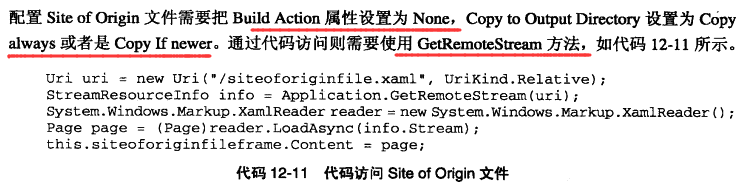


### （2）内容文件



### （3）site of origin文件

该类文件不参加编译，只有运行时才知道文件的存在



## 2、URI语法

### （1）应用场景：

1）App启动时设置要显示的用户界面

2）装载图像

3）页面间导航

4）装在资源、内容和Site Of Origin文件

### （2）组成部分

协议：pack

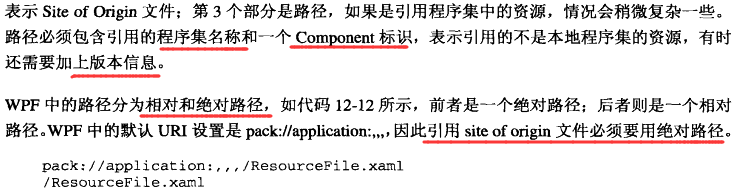
authority：

application:///表示编译时知道的文件，主要是资源和内容文件

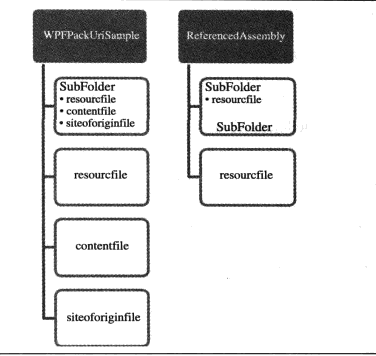
siteoforigin:///表示 Site Of Origin文件

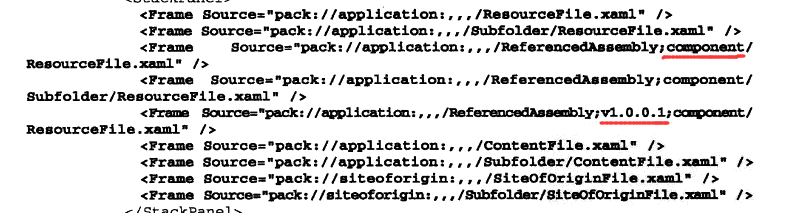
WPF中，”///”由“,,,”代替

路径：



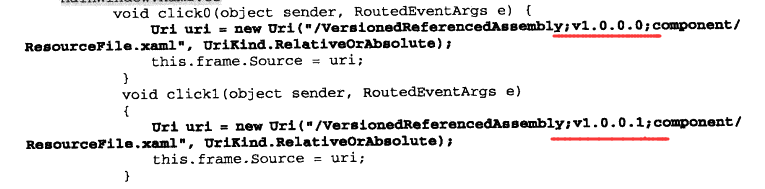
### （3）示例







针对同一程序集不同版本的URI设置：



### （4）URI的处理顺序

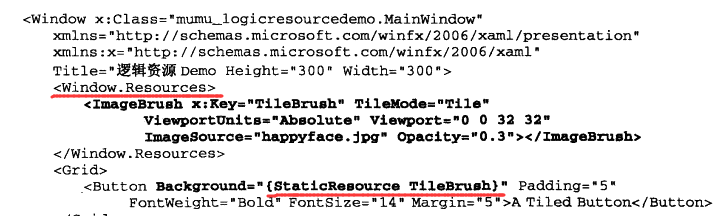
①在程序集的AssemblyAssociatedContentFileAttribute属性的元数据中查找，匹配则为内容文件

②在程序集的资源中查找，找到则为资源文件

③未找到，资源无效

## 3、逻辑资源

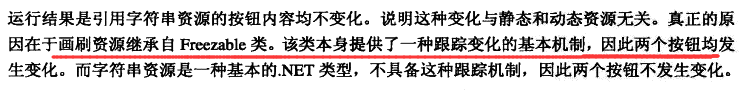
一些保存在元素Resources属性中的。NET对象，通常需要共享给多个子元素



### （1）静态资源和动态资源

区别：

①前者只从资源字典中查找一次资源，后者在应用程序需要时查找资源



②动态资源需要占用更多资源，另一方面可以改善加载时间，因为仅在需要时加载

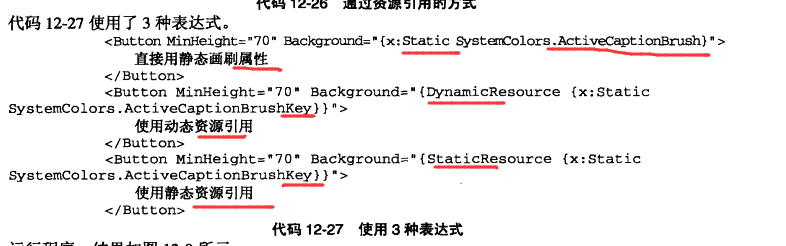
③静态资源必须先声明，动态资源没有这限制；所以Window 的Title属性只能使用动态资源

④多数使用静态资源，一般只有使用系统资源时才使用动态资源

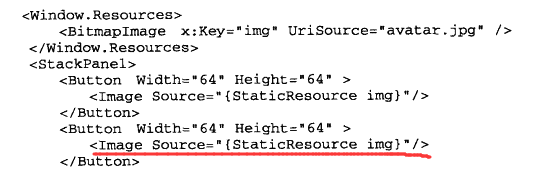
### （2）系统资源

主要指SystemColors、SystemFonts和SystemParameters类

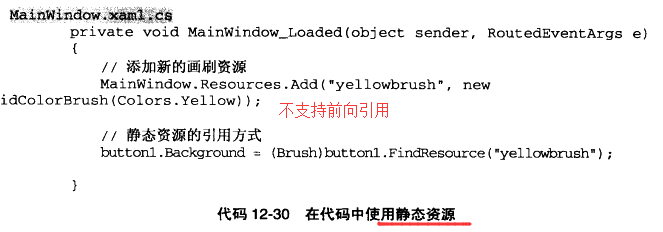




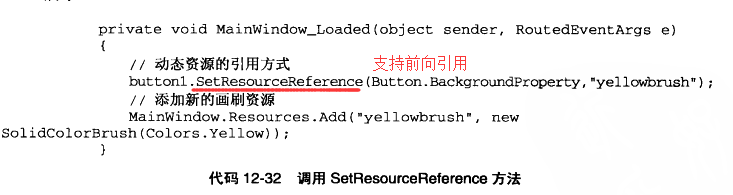
### （3）共享资源



### （4）代码定义和访问资源



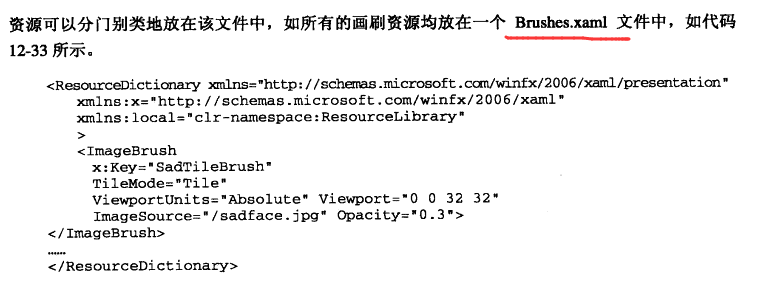


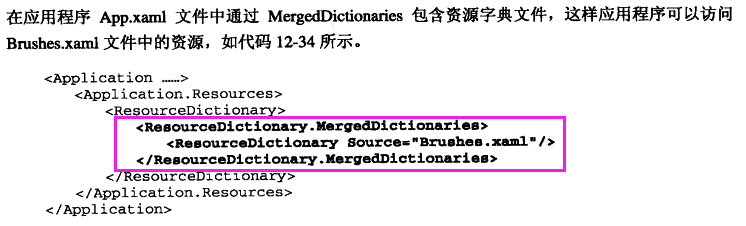


（5）ResourceDirectory

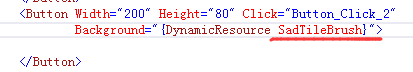
Xaml文件，组织大量资源

新建ResourceDictionary类型的文件





引用：

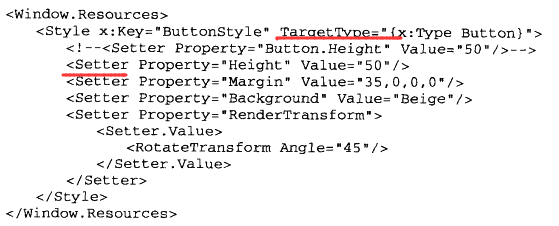


### （5）程序集之间共享资源

将所有资源放在单独程序集中

# 十三、样式和控件模板

## 1、样式



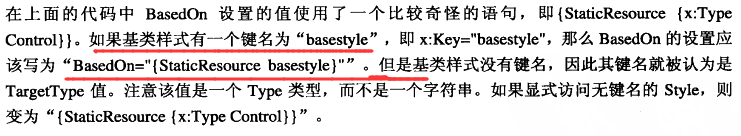
### （1）应用范围：

Style在窗口中定义，应用范围为该窗口，Style在应用程序中定义，应用范围为整个应用程序

### （2）样式继承

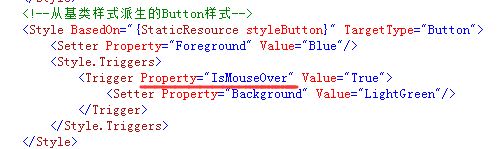
通过BasedOn实现





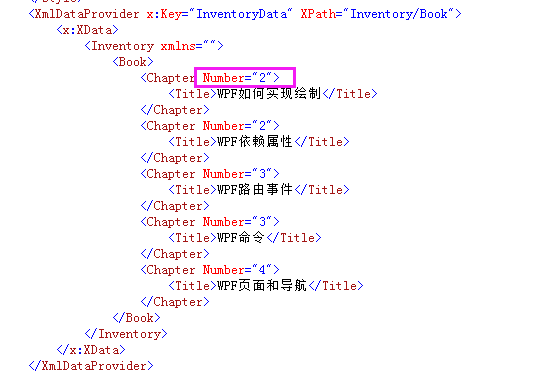
### （3）触发器

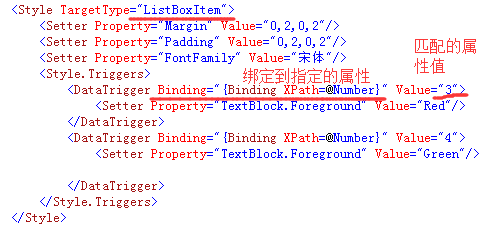
①属性触发器（Trigger）



②数据触发器（DataTrigger）

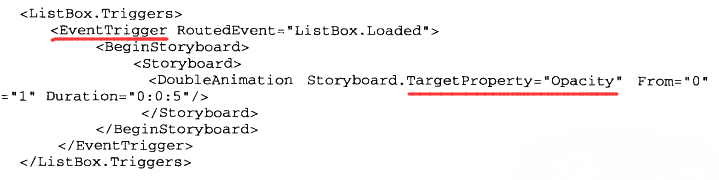
普通.NET属性，不仅仅依赖属性值改变时触发





③事件触发器（EventTrigger）

触发路由事件时会被调用



## 2、模板

4类模板，通过改变控件的可视化树来彻底改变其外观

