.NET Framework中直接事件的弊端：

事件的拥有者和事件响应者之间必须能够建立事件订阅，这样以来，事件的宿主必须能够直接访问事件的响应者，如果是内部事件，则必须建立一个事件链。

直接事件和路由事件的区别：

直接事件激发时，发送者直接将消息通过事件订阅交给事件响应者，事件响应者使用其事件处理器对事件的发生做出响应。

路由事件激发时，事件拥有者和事件响应者之间没有直接显式的订阅关系，事件拥有者只负责激发事件，事件的响应者则安装有事件侦听器，针对某类事件进行侦听，当有某类事件传递至此时事件响应者使用事件处理器来响应事件并决定此事件是否继续传递。将事件的Handled设置为true就可以阻止事件的继续传递。

路由事件：可以在元素树中向上冒泡和向下隧道传播，并且沿着传播路径被事件处理程序处理。

自定义路由事件：

1. 声明并注册路由事件。
2. 利用CLR事件包装路由事件。
3. 创建可以激发路由事件的方法。

向上冒泡的路由事件：

例：程序routed\_event\_test1

ReportLocationButton.cs

// Copyright 2016.刘珅珅

// author：刘珅珅

// 路由事件

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

using *System*.*Windows*.*Controls*;

using *System*.*Windows*;

namespace routed\_event\_test1

{

// 路由事件参数

public class ReportedLocationEventArgs:*RoutedEventArgs*

{

public ReportedLocationEventArgs(*RoutedEvent* routedEvent, object source)

: base(routedEvent, source)

{ }

public string Location { get; set; }

}

public class ReportLocationButton:*Button*

{

// 声明并定于路由事件

// 向上冒泡的路由事件

public static readonly *RoutedEvent* ReportLocationEvent = *EventManager*

.*RegisterRoutedEvent*("ReportTime", *RoutingStrategy*.*Bubble*,

typeof(*EventHandler*<ReportedLocationEventArgs>), typeof(ReportLocationButton));

// CLR事件包装器

public event *RoutedEventHandler* ReportTime

{

add { *AddHandler*(ReportLocationEvent, value); }

remove { *RemoveHandler*(ReportLocationEvent, value); }

}

// 激发路由事件

protected override void OnClick()

{

base.OnClick();

ReportedLocationEventArgs args = new ReportedLocationEventArgs(ReportLocationEvent, this);

args.Location = this.*Name*;

this.*RaiseEvent*(args); // 激发路由事件

}

}

}

MainWindow.xaml

<Window x:Class="routed\_event\_test1.MainWindow"

x:Name="Window"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:routed\_event\_test1"

Title="MainWindow" Height="350" Width="255"

local:ReportLocationButton.ReportTime="ReportTimeHandler">

<Grid x:Name="gridRoot" Background="Lime" local:ReportLocationButton.ReportTime="ReportTimeHandler">

<Grid x:Name="gridA" Margin="10" Background="Blue"

local:ReportLocationButton.ReportTime="ReportTimeHandler">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Canvas x:Name="canvasLeft" Grid.Column="0" Background="Red" Margin="10,10,10,165"

local:ReportLocationButton.ReportTime="ReportTimeHandler">

<local:ReportLocationButton x:Name="buttonLeft" Width="65" Height="100" Margin="10" Content="Left"

local:ReportLocationButton.ReportTime="ReportTimeHandler"/>

</Canvas>

<Canvas x:Name="canvasRight" Grid.Column="1" Background="Yellow" Margin="10,10,10,165"

local:ReportLocationButton.ReportTime="ReportTimeHandler">

<local:ReportLocationButton x:Name="buttonRight" Width="65" Height="100" Margin="10" Content="Right"

/>

</Canvas> local:ReportLocationButton.ReportTime="ReportTimeHandler"

<ListBox x:Name="listbox" Grid.ColumnSpan="2" HorizontalAlignment="Left"

Height="134" Margin="10,165,0,0" VerticalAlignment="Top" Width="207"/>

</Grid>

</Grid>

</Window>

MainWindow.xaml.cs:

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

using *System*.*Windows*;

using *System*.*Windows*.*Controls*;

using *System*.*Windows*.*Data*;

using *System*.*Windows*.*Documents*;

using *System*.*Windows*.*Input*;

using *System*.*Windows*.*Media*;

using *System*.*Windows*.*Media*.*Imaging*;

using *System*.*Windows*.*Navigation*;

using *System*.*Windows*.*Shapes*;

namespace routed\_event\_test1

{

/// <summary>

/// MainWindow.xaml 的交互逻辑

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

*InitializeComponent*();

}

// 路由事件的事件处理函数

private void ReportTimeHandler(object sender, ReportedLocationEventArgs e)

{

e.Location = (sender as *FrameworkElement*).*Name*;

string l = e.Location;

string c = "我到达了: " + l;

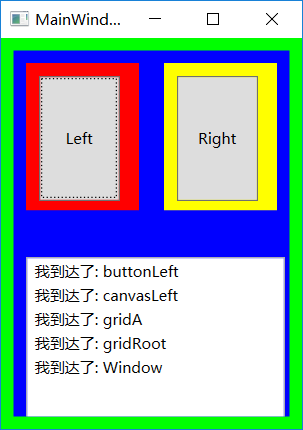
this.listbox.*Items*.*Add*(c);

}

}

}

在XAML文档MainWindow.xaml中，每个控件都注册了路由事件的处理函数，点击Left按钮：



如果某个控件没有注册事件处理函数，则不会处理该路由事件