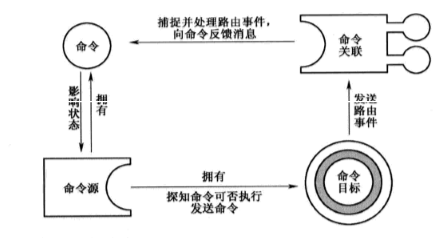
在WPF中，路由事件不具有约束力，而命令具有约束力。

WPF命令系统

1. 命令（Command）：WPF的命令实际上就是ICommand接口的类，平时使用最多的RoutedCommand类。
2. 命令源（Command Source）：即命令的发送者，实现了ICommandSource接口的类。很多界面元素都实现了这个接口，其中包括Button、MenuItem、ListBoxItem等。
3. 命令目标（Command Target）：即命令将发送给谁，或者说命令将作用在谁身上。命令目标必须是实现了IInputElement接口的类。
4. 命令绑定（Command Binding）：负责把一些外围逻辑与命令关联起来，比如执行之前对命令是否可以进行判断、命令执行之后的后续工作。



例：程序command\_test1

MainWindow.xaml

<Window x:Class="command\_test1.MainWindow"

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

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

Title="MainWindow" Height="350" Width="525">

<StackPanel x:Name="stackPanel">

<Button x:Name="button" Content="Send Command" Margin="3" />

<TextBox x:Name="textBox" Margin="5,0" Height="100" />

</StackPanel>

</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 command\_test1

{

/// <summary>

/// MainWindow.xaml 的交互逻辑

/// </summary>

public partial class MainWindow : *Window*

{

public MainWindow()

{

*InitializeComponent*();

InitializedCommand();

}

// 声明并定义命令

private *RoutedCommand* clearCommand = new *RoutedCommand*("Clear", typeof(MainWindow));

private void InitializedCommand()

{

// 把命令赋值给命令源（发送者）

button.*Command* = clearCommand;

// 指定命令目标

button.*CommandTarget* = textBox;

// 创建命令关联

*CommandBinding* cb = new *CommandBinding*();

cb.*Command* = clearCommand;

cb.*CanExecute* += cb\_CanExecute;

cb.*Executed* += cb\_Executed;

// 把命令关联安置在外围控件上

stackPanel.*CommandBindings*.*Add*(cb);

}

// 当探测命令是否可以执行时，此方法被调用

void cb\_CanExecute(object sender, *CanExecuteRoutedEventArgs* e)

{

if (string.*IsNullOrEmpty*(textBox.*Text*))

{

e.*CanExecute* = false;

}

else

{

e.*CanExecute* = true;

}

// 避免继续向上传而降低程序性能

e.*Handled* = true;

}

// 当命令送达目标后，此方法被调用

void cb\_Executed(object sender, *ExecutedRoutedEventArgs* e)

{

textBox.*Clear*();

// 避免向上传递而降低程序性能

e.*Handled* = true;

}

}

}

WPF命令库：

1. ApplicationCommands
2. ComponentCommands
3. NavigationCommands
4. MediaCommands
5. EditingCommands

WPF中的许多常用的命令就用这些命令库类的只读属性以单例模式暴露出来。

命令参数：WPF中的许多命令都是单例模式，可以使用CommandParameter来区别同一命令实现的不同的要求。

例：程序command\_test2

MainWindow.xaml

<Window x:Class="command\_test2.MainWindow"

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

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

Title="MainWindow" Height="240" Width="360">

<Grid Margin="6">

<Grid.RowDefinitions>

<RowDefinition Height="24" />

<RowDefinition Height="4" />

<RowDefinition Height="24" />

<RowDefinition Height="4" />

<RowDefinition Height="24" />

<RowDefinition Height="4" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<TextBlock Text="Name:" VerticalAlignment="Center" HorizontalAlignment="Left" Grid.Row="0"/>

<TextBox x:Name="nameTextBox" Margin="60,0,0,0" Grid.Row="0" />

<Button Content="New Teacher" Command="New" CommandParameter="Teacher" Grid.Row="2" />

<Button Content="New Student" Command="New" CommandParameter="Student" Grid.Row="4" />

<ListBox x:Name="listBoxNewItems" Grid.Row="6" />

</Grid>

<Window.CommandBindings>

<CommandBinding Command="New" CanExecute="New\_CanExecute" Executed="New\_Executed" />

</Window.CommandBindings>

</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 command\_test2

{

/// <summary>

/// MainWindow.xaml 的交互逻辑

/// </summary>

public partial class MainWindow : *Window*

{

public MainWindow()

{

*InitializeComponent*();

}

private void New\_CanExecute(object sender, *CanExecuteRoutedEventArgs* e)

{

if (string.*IsNullOrEmpty*(nameTextBox.*Text*))

{

e.*CanExecute* = false;

}

else

{

e.*CanExecute* = true;

}

}

private void New\_Executed(object sender, *ExecutedRoutedEventArgs* e)

{

string name = nameTextBox.*Text*;

if (e.*Parameter*.*ToString*() == "Teacher")

{

listBoxNewItems.*Items*.*Add*(string.*Format*("New Teacher: {0}，学而不厌，诲人不倦", name));

}

if (e.*Parameter*.*ToString*() == "Student")

{

listBoxNewItems.*Items*.*Add*(string.*Format*("New Student: {0}，好好学习，天天向上", name));

}

}

}

}