

## WPF中的命令模式：打造清晰、可重用的代码利器

作者：微信公众号：【架构师老卢】

12-5 8:43

1266

# WPF #2

**概述：**在WPF中，Command是一种优秀的机制，通过它，我们能够将用户界面操作与业务逻辑分离，提高代码的可维护性和可重用性。通过自定义ICommand接口的实现（如RelayCommand），我们能够轻松创建并在XAML中绑定命令，实现清晰的MVVM架构。这种模式使得应用程序的开发更加灵活，同时提高了代码的可测试性。

在WPF (Windows Presentation Foundation) 中，Command (命令) 是一种用于处理用户界面元素交互的机制，它有助于将用户输入（如按钮点击、菜单选择等）与应用程序逻辑分离开来。使用命令模式，可以在MVVM (Model-View-ViewModel) 架构中更好地组织代码，并且有助于实现可重用和可测试的代码。以下是关于WPF中Command的详细讲解：

## 1. Command的作用和功能：

在WPF中，Command主要有以下几个作用和功能：

- 解耦UI和业务逻辑：**使用Command可以将用户界面元素（如按钮）的操作与实际的业务逻辑分离，使代码更易维护和测试。
- 可重用性：**可以在多个界面元素中共享相同的命令，从而提高代码的可重用性。
- 支持异步操作：**Command可以处理异步操作，例如在后台线程中执行某些任务而不阻塞用户界面。
- 状态管理：**命令可以通过CanExecute方法控制是否允许执行，从而实现对命令的状态管理。

## 2. Command的用法：

在WPF中，可以使用ICommand接口来定义自定义命令，也可以使用RoutedCommand和RoutedUICommand类来创建路由命令。以下是使用ICommand接口的示例：

```
1 using System;
2 using System.Windows.Input;
3
4 public class RelayCommand : ICommand
5 {
6     private readonly Action<object> _execute;
7     private readonly Func<object, bool> _canExecute;
8
9     public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)
10    {
11        _execute = execute ?? throw new ArgumentNullException(nameof(execute));
12        _canExecute = canExecute;
13    }
14
15    public event EventHandler CanExecuteChanged
16    {
17        add { CommandManager.RequerySuggested += value; }
18        remove { CommandManager.RequerySuggested -= value; }
19    }
20
21    public bool CanExecute(object parameter)
22    {
23        return _canExecute == null || _canExecute(parameter);
24    }
25
26    public void Execute(object parameter)
27    {
28        _execute(parameter);
29    }
30 }
```

## 3. 使用Command的步骤：

步骤如下：

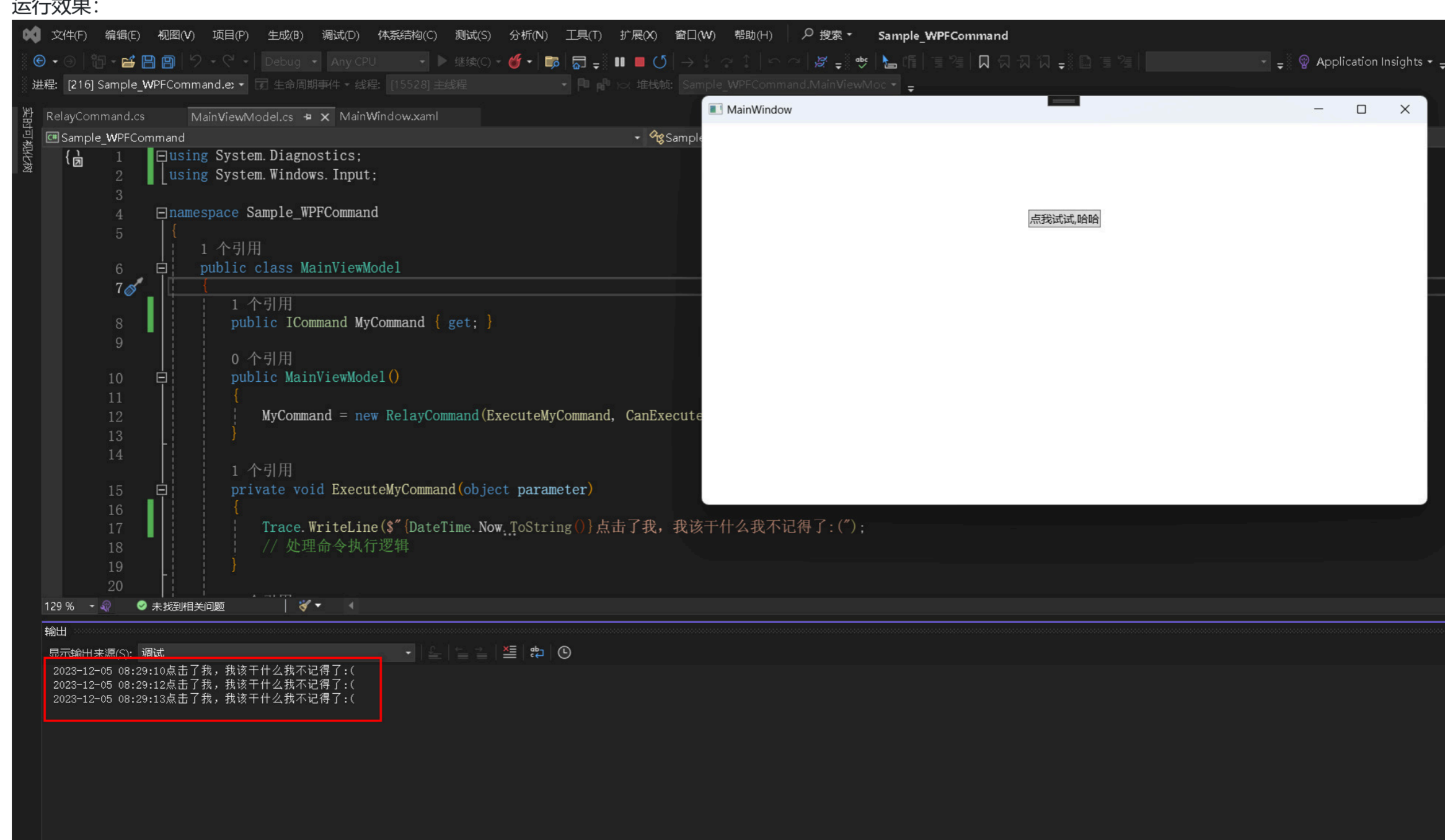
### 步骤 1：创建ViewModel并定义命令

```
1 using System.Diagnostics;
2 using System.Windows.Input;
3
4 namespace Sample_WPFCommand
5 {
6     public class MainViewModel
7     {
8         public ICommand MyCommand { get; }
9
10        public MainViewModel()
11        {
12            MyCommand = new RelayCommand(ExecuteMyCommand, CanExecuteMyCommand);
13        }
14
15        private void ExecuteMyCommand(object parameter)
16        {
17            Trace.WriteLine($"{DateTime.Now.ToString()}点击了我，我该干什么我不记得了：(");
18            // 处理命令执行逻辑
19        }
20
21        private bool CanExecuteMyCommand(object parameter)
22        {
23            // 定义命令是否可执行的逻辑
24            return true;
25        }
26    }
27
28 }
```

### 步骤 2：在XAML中绑定命令

```
1 <Window x:Class="Sample_WPFCommand.MainWindow"
2       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4       xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5       xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6       xmlns:local="clr-namespace:Sample_WPFCommand"
7       mc:Ignorable="d"
8       Title="MainWindow" Height="450" Width="800">
9
10    <Window.DataContext>
11        <local:MainViewModel />
12    </Window.DataContext>
13
14    <Grid>
15        <Grid.RowDefinitions>
16            <RowDefinition/>
17            <RowDefinition/>
18        </Grid.RowDefinitions>
19
20        <Button Grid.Row="0" Content="点我试试,哈哈" Command="{Binding MyCommand}" HorizontalAlignment="Center" VerticalAlignment="Center" />
21    </Grid>
22 </Window>
```

运行效果：



## 4. 实例源代码：

上述步骤中的源代码已经涵盖了一个简单的WPF应用程序中如何使用Command。请根据实际需求修改ExecuteMyCommand和CanExecuteMyCommand方法中的逻辑。

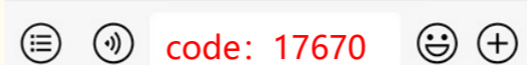
源代码获取：公众号回复消息【code: 17670】

## 相关代码下载地址



**重要提示!**：取消关注公众号后将无法再启用回复功能，不支持解封!

**第一步:** 微信扫码关注公众号“架构师老卢”

**第二步:** 在公众号聊天框发送 **code: 17670**，如： 获取下载地址

**第三步:** 恭喜你，快去下载你想要的资源吧