

CUS

Microsoft Announces Visual Studio 2019



ICommand Interface In MVVM

ASK A QUESTION

CONTRIBUTE

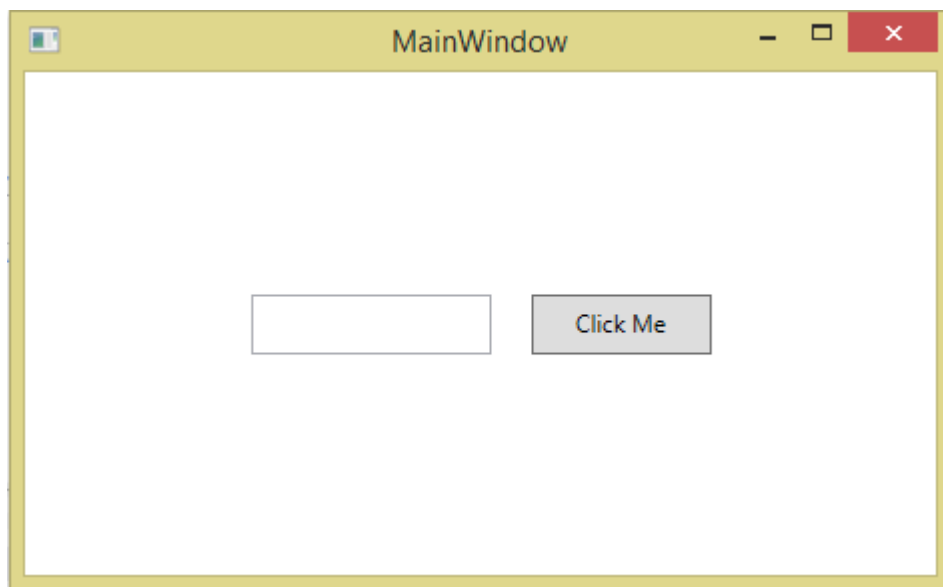
2 2 20.9k

[MVVM ICommand.zip](#)[Download Free Office API](#)

In this article, we will learn about button click event in WPF with MVVM concept. We use ICommand interface for generating the button click event. Let's take one simple example of one simple textbox and button. After writing the text in textbox, when we click on button, it will display a simple message box.

Before starting this article, please read [Explain INotifyPropertyChanged In WPF - MVVM](#) article first.

First, create one WPF application and create a window like the following.



XAML code

```
01. <Window x:Class="MVVM_ICommand.MainWindow" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pre
02.     <StackPanel Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center">
03.         <TextBox Width="120" Height="30" Margin="10"></TextBox>
04.         <Button Width="90" Margin="10" Content="Click Me"></Button>
05.     </StackPanel>
06. </Window>
```

CUS

Microsoft Announces Visual Studio 2019

implementation is given in my previous article.



ASK A QUESTION

CONTRIBUTE

Give Your Model command reference to the ViewModel and define the related resources file. Then, give data

Now, let's start our main code for making a new class RelayCommand. Make it as public.

```
01. public class RelayCommand {}
02.
03. Now implement the interface ICommand
04. public class RelayCommand: ICommand {
05.     public bool CanExecute(object parameter) {
06.
07.     }
08.     public event EventHandler CanExecuteChanged;
09.     public void Execute(object parameter) {
10.
11.     }
12. }
```

It will create two methods and one event. The first method "CanExecute" decides whether we are allowed to fire the command (the button click event) or not. The second method "Execute" method contains the actual logic. If CanExecute method returns true, then Execute method is run.

Now, Create two action properties that we initialize in constructor of the relay command class.

```
01. Action < object > _executeMethod;
02. Func < bool, object > _canExecuteMethod;
03.
04. public RelayCommand(Action < object > executeMethod, Func < bool, object > canExecuteMethod) {
05.     _executeMethod = executeMethod;
06.     _canExecuteMethod = canExecuteMethod;
07. }
```

Here, in CanExecute method, we put null validation to check if _canExecutemethod is initialized or not. If not, then it returns false; otherwise it returns true. Also, write the logic for Execute method. Here, we simply initialize the _execute method because we make this RelayCommand class to be generalized.

```
01. public bool CanExecute(object parameter) {
02.     if (_canExecuteMethod != null) {
03.         return true;
04.     } else {
05.         return false;
06.     }
07. }
08.
09.
10. public void Execute(object parameter) {
11.     _executeMethod(parameter);
12. }
```

Now, let's define "CanExecuteChanged" the event logic, which will run the canExecute method continuously.

```
01. public event EventHandler CanExecuteChanged
02. {
03.     add { CommandManager.RequerySuggested += value; }
04.     remove { CommandManager.RequerySuggested -= value; }
05. }
```

CUS

soft Announces Visual Studio 2019



```
01. public class RelayCommand: ICommand
02. {
03.     Action<object> _executeMethod;
04.     Func<bool, object> _canexecuteMethod;
05. }
```

ASK A QUESTION

CONTRIBUTE

```
07.     _canexecuteMethod = canexecuteMethod;
08. }
09.
10. public bool CanExecute(object parameter) {
11.     if (_canexecuteMethod != null) {
12.         return _canexecuteMethod(parameter);
13.     } else {
14.         return false;
15.     }
16. }
17.
18. public event EventHandler CanExecuteChanged {
19.     add {
20.         CommandManager.RequerySuggested += value;
21.     }
22.     remove {
23.         CommandManager.RequerySuggested -= value;
24.     }
25. }
26.
27. public void Execute(object parameter) {
28.     _executeMethod(parameter);
29. }
30. }
```

Now, go for the View-Model class and create the ICommand property, as shown below.

```
public ICommand MyCommand { get; set; }
```

And, also create two methods like,

```
01. private bool CanExecuteMyMethod(object parameter) {
02.     if (string.IsNullOrEmpty(Name)) {
03.         return false;
04.     } else {
05.         if (Name < > "") {
06.             return true;
07.         } else {
08.             return false;
09.         }
10.     }
11. }
12.
13. private void ExecuteMyMethod(object parameter) {
14.     MessageBox.Show("Hello... " + Name);
15. }
16. }
```

Now, create constructor for View-Model and pass both these methods to it.

```
01. public ViewModel()
02. {
03.     MyCommand = new RelayCommand(ExecuteMyMethod, CanExecuteMyMethod);
04. }
```

Full View-Model Code

CUS

Microsoft Announces Visual Studio 2019



```

04.     get;
05.     set;
06. }

```

ASK A QUESTION

CONTRIBUTE

```

10.     public string Name {
11.         get {
12.             return _name;
13.         }
14.         set {
15.             _name = value;
16.         }
17.     }
18.
19.     public ViewModel() {
20.         MyCommand = new RelayCommand(ExecuteMyMethod, CanExecuteMyMethod);
21.     }
22.
23.
24.     private bool CanExecuteMyMethod(object parameter) {
25.         if (string.IsNullOrEmpty(Name)) {
26.             return false;
27.         } else {
28.             if (Name < > "") {
29.                 return true;
30.             } else {
31.                 return false;
32.             }
33.         }
34.     }
35.
36.
37.     private void ExecuteMyMethod(object parameter) {
38.         MessageBox.Show("Hello... " + Name);
39.     }
40.
41. }

```

Now, go for the design View. Build the Solution and open the View's XAML code. Set the command property and give command a name.

```
<Button Width="90" Margin="10" Content="Click Me" Command="{Binding MyCommand}"></Button>
```

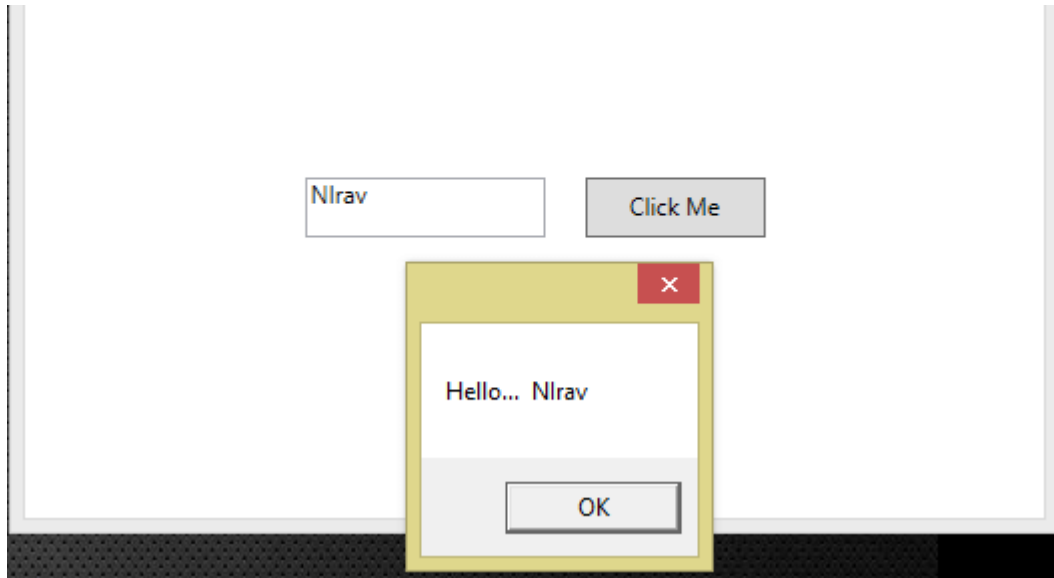
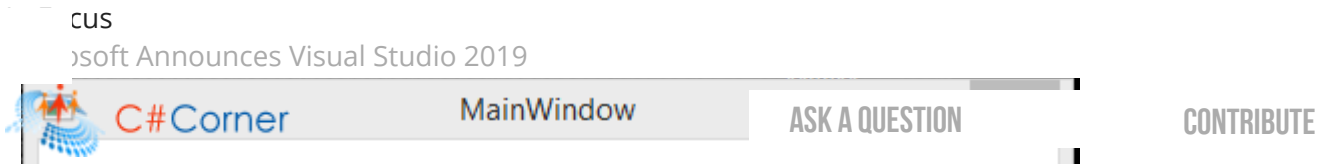
Full XAML Code

```

01. <Window x:Class="MVVM_ICommand.MainWindow" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pre
    namespace:MVVM_ICommand.ViewModel" Title="MainWindow" Height="350" Width="525">
02.     <Window.Resources>
03.         <viewModel:ViewModel x:Key="vm"></viewModel:ViewModel>
04.     </Window.Resources>
05.     <StackPanel Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center" D
        {Binding Source={StaticResource vm}}">
06.         <TextBox Width="120" Height="30" Margin="10" Text="
        {Binding Path=Name, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"></TextBox>
07.         <Button Width="90" Margin="10" Content="Click Me" Command="{Binding MyCommand}">
08.     </StackPanel>
09. </Window>

```

Run the application.



This is how this command interface works in MVVM . If you have any confusion, you can download the attached source code and debug it line by line.

[ICommand](#)
[MVVM](#)
[WPF](#)


Nirav Daraniya *TOP 500*

I have experienced in Microsoft technology like Windows application, Windows services, WPF, MVVM, WCF services and also in IOT (Internet of things). And I have also small experienced in developing and customization of Microsoft... [Read more](#)

<http://www.codeoverflow.net>

372

668.5k

1

2

2



Type your comment here and press Enter Key (Minimum 10 characters)



Nice post..

Prasanna Murali

192 8.3k 400.4k

Oct 09, 2016

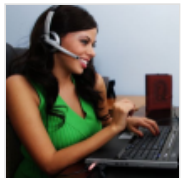
1

0 Reply

Good..

ketan borsdiya

Oct 06, 2016

[ASK A QUESTION](#)[CONTRIBUTE](#)

Hire a remote developer

Looking to add more bandwidth to your software team? Web Developers, designers, testers are now available on demand. Flexible hours, very competitive rates, expert team and High Quality work.

TRENDING UP

- 01 ASP.NET Web API Using MVC And jQuery To Upload And Download Files - Part Seven
- 02 ASP.NET Web API Using MVC And HttpClient To Upload And Download Files - Part Eight
- 03 Building A Blockchain In .NET Core - Basic Blockchain
- 04 Create Your First Azure Machine Learning Workspace
- 05 Introduction To Blockchain, Ethereum And Smart Contracts
- 06 ASP.NET Core Blazor CRUD Using Entity Framework And Web API
- 07 What Is ML.NET
- 08 Best Practices Of Writing C# Code
- 09 Comparison Between React And Angular
- 10 Azure Machine Learning Client And Server Applications

[View All](#) 

CUS

Microsoft Announces Visual Studio 2019

[ASK A QUESTION](#)[CONTRIBUTE](#)[Philadelphia](#)[New York](#)[London](#)[Delhi](#)

JOIN C# CORNER

and millions of developer friends worldwide.

[Sign Up](#)

[Learn ASP.NET MVC](#) [Learn ASP.NET Core](#) [Learn Python](#) [Learn JavaScript](#) [Learn Xamarin](#)
[Learn Oracle](#) [More...](#)

[Home](#) [Events](#) [Consultants](#) [Jobs](#) [Career Advice](#) [Stories](#) [Partners](#)

CUS

Microsoft Announces Visual Studio 2019



C# Corner

[About Us](#)

[Contact Us](#)

[Privacy Policy](#)

[Terms](#)

[ASK A QUESTION](#)

[or](#)

[CONTRIBUTE](#)

©2018 C# Corner. All contents are copyright of their authors.