[C# WPF Tutorial #8 - Data Bindings using INotifyPropertyChanged](https://www.youtube.com/watch?v=56m0H1qhZuw&list=PLih2KERbY1HHOOJ2C6FOrVXIwg4AZ-hk1&index=8)

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

MainWindow.xaml

<Window x:Class="\_08\_WPF\_Tutorial\_v5.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_08\_WPF\_Tutorial\_v5"

mc:Ignorable="d"

Title="MainWindow" Height="250" Width="300">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBox Name="txtInput"

VerticalContentAlignment="Center"

Width="250"

Height="40"

FontSize="16"

Text="{ Binding BoundText,

UpdateSourceTrigger=PropertyChanged,

Mode=OneWayToSource}" />

<Button Grid.Row="1"

Name="btnSet"

Content="Set"

Width="100"

Height="40"

FontSize="16"

FontWeight="Light"

HorizontalAlignment="Center"

Background="CadetBlue"

Foreground="Azure"

Click="btnSet\_Click"/>

<TextBlock Grid.Row="2"

Name="tbResult"

FontSize="16"

Text="{Binding BoundText}"

VerticalAlignment="Center"

HorizontalAlignment="Center" />

</Grid>

</Window>

MainWindow.xaml.cs

using System.ComponentModel;

using System.Runtime.CompilerServices;

using System.Windows;

namespace \_08\_WPF\_Tutorial\_v5

{

public partial class MainWindow : Window, INotifyPropertyChanged

{

public MainWindow()

{

DataContext = this;

InitializeComponent();

}

private string \_boundText;

public event PropertyChangedEventHandler? PropertyChanged;

public string BoundText

{

get { return \_boundText; }

set

{

\_boundText = value;

OnPropertyChanged();

}

}

private void btnSet\_Click(object sender, RoutedEventArgs e)

{

BoundText = "set from code";

}

private void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("BoundText"));

}

}

}

Process:

# Start new WPF solution in Visual Studio.

A black screen with white text

Description automatically generated

A black rectangular object with a black background

Description automatically generated

# Name it:

A screenshot of a computer

Description automatically generated

# Starting point. Build the GUI to look like this:

A screenshot of a computer

Description automatically generated

# Add the following code to the .xaml file to create the GUI.

MainWindow.xaml

<Window x:Class="\_08\_WPF\_Tutorial\_v5.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_08\_WPF\_Tutorial\_v5"

mc:Ignorable="d"

Title="MainWindow" Height="250" Width="300">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBox Name="txtInput"

VerticalContentAlignment="Center"

Width="250"

Height="40"

FontSize="16"

Text="{Binding BoundText}" />

<Button Grid.Row="1"

Name="btnSet"

Content="Set"

Width="100"

Height="40" />

<TextBlock Grid.Row="2"

Name="tbResult"

FontSize="16"

Text="{Binding BoundText}"

VerticalAlignment="Center"

HorizontalAlignment="Center" />

</Grid>

</Window>

# The MainWindow.xaml.cs code behind page will look like this:

MainWindow.xaml.cs

using System.Windows;

namespace \_08\_WPF\_Tutorial\_v6

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

}

}

# Make the following changes to add a property for the text:

Add a Full Property by typing propfull and pressing TAB twice.

MainWindow.xaml.cs

using System.Windows;

namespace \_08\_WPF\_Tutorial\_v6

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private string \_boundText;

public string BoundText

{

get { return \_boundText; }

set

{

\_boundText = value;

}

}

}

}

## Define a DataContext:

MainWindow.xaml.cs

using System.Windows;

namespace \_08\_WPF\_Tutorial\_v6

{

public partial class MainWindow : Window

{

public MainWindow()

{

DataContext = this;

InitializeComponent();

}

private string \_boundText;

public string BoundText

{

get { return \_boundText; }

set

{

\_boundText = value;

}

}

}

}

# To make use of OnPropertyChanged() method we have to implement INotifyPropertyChanged:

MainWindow.xaml.cs

using System.ComponentModel;

using System.Windows;

namespace \_08\_WPF\_Tutorial\_v6

{

public partial class MainWindow : Window, INotifyPropertyChanged

{

public MainWindow()

{

DataContext = this;

InitializeComponent();

}

private string \_boundText;

public event PropertyChangedEventHandler? PropertyChanged;

public string BoundText

{

get { return \_boundText; }

set

{

\_boundText = value;

}

}

}

}

# Invoke the PropertyChanged method:

using System.ComponentModel;

using System.Windows;

namespace \_08\_WPF\_Tutorial\_v6

{

public partial class MainWindow : Window, INotifyPropertyChanged

{

public MainWindow()

{

DataContext = this;

InitializeComponent();

}

private string \_boundText;

public event PropertyChangedEventHandler? PropertyChanged;

public string BoundText

{

get { return \_boundText; }

set

{

\_boundText = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("BoundText"));

}

}

}

}

# Cause the changes to be live in real time:

Add UpdateSourceTrigger in order to make it real time.

MainWindow.xaml

<Window x:Class="\_08\_WPF\_Tutorial\_v6.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_08\_WPF\_Tutorial\_v6"

mc:Ignorable="d"

Title="MainWindow" Height="250" Width="300">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBox Name="txtInput"

VerticalContentAlignment="Center"

Width="250"

Height="40"

FontSize="16"

Text="{ Binding BoundText,

UpdateSourceTrigger=PropertyChanged}" />

<Button Grid.Row="1"

Name="btnSet"

Content="Set"

Width="100"

Height="40" />

<TextBlock Grid.Row="2"

Name="tbResult"

FontSize="16"

Text="{Binding BoundText}"

VerticalAlignment="Center"

HorizontalAlignment="Center" />

</Grid>

</Window>

# Add a Click Handler to the button:

For some reason the button already works but this should be explicitly coded as below, then at least other attributes can be assigned.

MainWindow.xaml

<Window x:Class="\_08\_WPF\_Tutorial\_v6.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_08\_WPF\_Tutorial\_v6"

mc:Ignorable="d"

Title="MainWindow" Height="250" Width="300">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBox Name="txtInput"

VerticalContentAlignment="Center"

Width="250"

Height="40"

FontSize="16"

Text="{ Binding BoundText,

UpdateSourceTrigger=PropertyChanged}" />

<Button Grid.Row="1"

Name="btnSet"

Content="Set"

Width="100"

Height="40"

Click="btnSet\_Click" />

<TextBlock Grid.Row="2"

Name="tbResult"

FontSize="16"

Text="{Binding BoundText}"

VerticalAlignment="Center"

HorizontalAlignment="Center" />

</Grid>

</Window>

# AND populate the new Click Handler in the code behind:

MainWindow.xaml.cs

using System.ComponentModel;

using System.Windows;

namespace \_08\_WPF\_Tutorial\_v6

{

public partial class MainWindow : Window, INotifyPropertyChanged

{

public MainWindow()

{

DataContext = this;

InitializeComponent();

}

private string \_boundText;

public event PropertyChangedEventHandler? PropertyChanged;

public string BoundText

{

get { return \_boundText; }

set

{

\_boundText = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("BoundText"));

}

}

private void btnSet\_Click(object sender, RoutedEventArgs e)

{

BoundText = "set from code";

}

}

}

# Add data changes DIRECTION.

At this point both TextBox and TextBlock are populated by the BoundText change from the button. Now make the TextBox one way only.

MainWindow.xaml

<Window x:Class="\_08\_WPF\_Tutorial\_v6.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_08\_WPF\_Tutorial\_v6"

mc:Ignorable="d"

Title="MainWindow" Height="250" Width="300">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBox Name="txtInput"

VerticalContentAlignment="Center"

Width="250"

Height="40"

FontSize="16"

Text="{ Binding BoundText,

UpdateSourceTrigger=PropertyChanged,

Mode=OneWayToSource}" />

<Button Grid.Row="1"

Name="btnSet"

Content="Set"

Width="100"

Height="40"

Click="btnSet\_Click" />

<TextBlock Grid.Row="2"

Name="tbResult"

FontSize="16"

Text="{Binding BoundText}"

VerticalAlignment="Center"

HorizontalAlignment="Center" />

</Grid>

</Window>

# A bit of cleanup

This works fine but now cleanup…create method for OnPropertyChanged and move event invocation…

MainWindow.xaml.cs

using System.ComponentModel;

using System.Windows;

namespace \_08\_WPF\_Tutorial\_v6

{

public partial class MainWindow : Window, INotifyPropertyChanged

{

public MainWindow()

{

DataContext = this;

InitializeComponent();

}

private string \_boundText;

public event PropertyChangedEventHandler? PropertyChanged;

public string BoundText

{

get { return \_boundText; }

set

{

\_boundText = value;

OnPropertyChanged("BoundText");

}

}

private void btnSet\_Click(object sender, RoutedEventArgs e)

{

BoundText = "set from code";

}

private void OnPropertyChanged(string propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

}

# Another tweak

using System.ComponentModel;

using System.Runtime.CompilerServices;

using System.Windows;

namespace \_08\_WPF\_Tutorial\_v6

{

public partial class MainWindow : Window, INotifyPropertyChanged

{

public MainWindow()

{

DataContext = this;

InitializeComponent();

}

private string \_boundText;

public event PropertyChangedEventHandler? PropertyChanged;

public string BoundText

{

get { return \_boundText; }

set

{

\_boundText = value;

OnPropertyChanged();

}

}

private void btnSet\_Click(object sender, RoutedEventArgs e)

{

BoundText = "set from code";

}

private void OnPropertyChanged([CallerMemberName] string? propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

}