[C# WPF Tutorial #12 – ListView](https://www.youtube.com/watch?v=yTzwimqLrIM&list=PLih2KERbY1HHOOJ2C6FOrVXIwg4AZ-hk1&index=12)

A screenshot of a computer

Description automatically generated

Finished code for the multiselect version is here:

MainWindow.xaml

<Window x:Class="\_12\_WPF\_Tutorial\_v1.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:\_12\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="400" Width="400">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListView Name="lvEntries"

Grid.Row="1"

Grid.ColumnSpan="2"

SelectionMode="Extended"

/>

<StackPanel Orientation="Horizontal"

Grid.ColumnSpan="2"

>

<TextBox Name="txtEntry"

Width="190"

Height="35"

FontSize="16"

Margin="5"

VerticalContentAlignment="Center"

/>

<Button Grid.Column="1"

Name="btnAdd"

Content="Add"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnAdd\_Click"

/>

<Button Grid.Column="1"

Name="btnDel"

Content="Delete"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnDel\_Click"

/>

<Button Grid.Column="1"

Name="btnClear"

Content="Clear"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnClear\_Click"

/>

</StackPanel>

</Grid>

</Window>

MainWindow.xaml.cs

using System.Collections;

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

lvEntries.Items.Add(txtEntry.Text);

txtEntry.Clear();

txtEntry.Focus();

}

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

{

//object item = lvEntries.SelectedItem;

var items = lvEntries.SelectedItems;

var result = MessageBox.Show($"Are you sure you want to delete {items.Count} items?", "Sure", MessageBoxButton.YesNo);

if (result == MessageBoxResult.Yes)

{

var itemsList = new ArrayList(items);

foreach (var item in itemsList)

{

lvEntries.Items.Remove(item);

}

}

else

{

}

}

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

{

lvEntries.Items.Clear();

}

}

}

# 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. Code looks like this initially.

MainWindow.xaml

<Window x:Class="\_12\_WPF\_Tutorial\_v1.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:\_12\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid>

</Grid>

</Window>

MainWindow.xaml.cs

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

}

}

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

A screenshot of a computer

Description automatically generated

MainWindow.xaml

<Window x:Class="\_12\_WPF\_Tutorial\_v1.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:\_12\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="400" Width="400">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

</Grid>

</Window>

# Add a ListView to the 2nd Row that spans 2 columns

MainWindow.xaml

<Window x:Class="\_12\_WPF\_Tutorial\_v1.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:\_12\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="400" Width="400">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListView Name="lvEntries"

Grid.Row="1"

Grid.ColumnSpan="2"

/>

</Grid>

</Window>

# Populate the ListView

ListView can be populated either from the xaml file or the xaml.cs file. We will do it from the xaml.cs file.

Populate the Items collection using .Add().

MainWindow.xaml.cs

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

lvEntries.Items.Add("A");

lvEntries.Items.Add("B");

lvEntries.Items.Add("C");

}

}

}

# Populate the ListView using a button to add text from a TextBox

Add Button and TextBox.

MainWindow.xaml

<Window x:Class="\_12\_WPF\_Tutorial\_v1.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:\_12\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="400" Width="400">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListView Name="lvEntries"

Grid.Row="1"

Grid.ColumnSpan="2"

/>

<TextBox Name="txtEntry"

Height="35"

FontSize="16"

Margin="5"

VerticalContentAlignment="Center"

/>

<Button Grid.Column="1"

Name="btnAdd"

Content="Add"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

/>

</Grid>

</Window>

# Add click event handler to Button

MainWindow.xaml

<Window x:Class="\_12\_WPF\_Tutorial\_v1.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:\_12\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="400" Width="400">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListView Name="lvEntries"

Grid.Row="1"

Grid.ColumnSpan="2"

/>

<TextBox Name="txtEntry"

Height="35"

FontSize="16"

Margin="5"

VerticalContentAlignment="Center"

/>

<Button Grid.Column="1"

Name="btnAdd"

Content="Add"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnAdd\_Click"

/>

</Grid>

</Window>

MainWindow.xaml.cs

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

~~lvEntries.Items.Add("A");~~

~~lvEntries.Items.Add("B");~~

~~lvEntries.Items.Add("C");~~

}

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

{

lvEntries.Items.Add(txtEntry.Text);

}

}

}

# Add Buttons Delete and Clear List

Make a few changes here in order to position the buttons neatly.

Add a <StackPanel> with Horizontal orientation and put the TextBox and the 3 Buttons in it.

Add click event handlers to last two buttons.

MainWindow.xaml

<Window x:Class="\_12\_WPF\_Tutorial\_v1.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:\_12\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="400" Width="400">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListView Name="lvEntries"

Grid.Row="1"

Grid.ColumnSpan="2"

/>

<StackPanel Orientation="Horizontal"

Grid.ColumnSpan="2"

>

<TextBox Name="txtEntry"

Width="190"

Height="35"

FontSize="16"

Margin="5"

VerticalContentAlignment="Center"

/>

<Button Grid.Column="1"

Name="btnAdd"

Content="Add"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnAdd\_Click"

/>

<Button Grid.Column="1"

Name="btnDel"

Content="Delete"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnDel\_Click"

/>

<Button Grid.Column="1"

Name="btnClear"

Content="Clear"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnClear\_Click"

/>

</StackPanel>

</Grid>

</Window>

MainWindow.xaml.cs

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

lvEntries.Items.Add(txtEntry.Text);

}

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

{

}

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

{

}

}

}

# Add Clear List Button functionality

MainWindow.xaml.cs

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

lvEntries.Items.Add(txtEntry.Text);

}

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

{

}

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

{

lvEntries.Items.Clear();

}

}

}

# Add Clear TextBox functionality when pressing Add Button

MainWindow.xaml.cs

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

lvEntries.Items.Add(txtEntry.Text);

txtEntry.Clear();

}

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

{

}

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

{

lvEntries.Items.Clear();

}

}

}

# Keep focus in TextBox when pressing Add Button

MainWindow.xaml.cs

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

lvEntries.Items.Add(txtEntry.Text);

txtEntry.Clear();

txtEntry.Focus();

}

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

{

}

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

{

lvEntries.Items.Clear();

}

}

}

# Creating the Delete functionality using RemoveAt(index)

(Note this is the RemoveAt Index method)

NOTE: If nothing is selected you will get an Exception.

Firstly ensure that the SelectionMode of the ListView is set to Single.

Then add the code to the btnDel click event handler to RemoveAt the particular Index selected.

MainWindow.xaml

<Window x:Class="\_12\_WPF\_Tutorial\_v1.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:\_12\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="400" Width="400">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListView Name="lvEntries"

Grid.Row="1"

Grid.ColumnSpan="2"

SelectionMode="Single"

/>

<StackPanel Orientation="Horizontal"

Grid.ColumnSpan="2"

>

<TextBox Name="txtEntry"

Width="190"

Height="35"

FontSize="16"

Margin="5"

VerticalContentAlignment="Center"

/>

<Button Grid.Column="1"

Name="btnAdd"

Content="Add"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnAdd\_Click"

/>

<Button Grid.Column="1"

Name="btnDel"

Content="Delete"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnDel\_Click"

/>

<Button Grid.Column="1"

Name="btnClear"

Content="Clear"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnClear\_Click"

/>

</StackPanel>

</Grid>

</Window>

MainWindow.xaml.cs

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

lvEntries.Items.Add(txtEntry.Text);

txtEntry.Clear();

txtEntry.Focus();

}

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

{

int index = lvEntries.SelectedIndex;

lvEntries.Items.RemoveAt(index);

}

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

{

lvEntries.Items.Clear();

}

}

}

# Creating the Delete functionality using Remove(item)

This way of doing the delete is much better as it allows you to work with the actual objects and works for databases etc.

MainWindow.xaml.cs

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

lvEntries.Items.Add(txtEntry.Text);

txtEntry.Clear();

txtEntry.Focus();

}

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

{

~~int index = lvEntries.SelectedIndex;~~

~~lvEntries.Items.RemoveAt(index);~~

object item = lvEntries.SelectedItem;

lvEntries.Items.Remove(item);

}

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

{

lvEntries.Items.Clear();

}

}

}

# Add a response function to give you options when deleting

This works for selecting a single item to delete.

MainWindow.xaml.cs

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

lvEntries.Items.Add(txtEntry.Text);

txtEntry.Clear();

txtEntry.Focus();

}

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

{

object item = lvEntries.SelectedItem;

var result = MessageBox.Show($"Are you sure you want to delete: {(string)item}?", "Sure", MessageBoxButton.YesNo);

if (result == MessageBoxResult.Yes)

{

lvEntries.Items.Remove(item);

}

else

{

}

}

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

{

lvEntries.Items.Clear();

}

}

}

# Converting delete function to multiselect

MainWindow.xaml

<Window x:Class="\_12\_WPF\_Tutorial\_v1.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:\_12\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="400" Width="400">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListView Name="lvEntries"

Grid.Row="1"

Grid.ColumnSpan="2"

SelectionMode="Extended"

/>

<StackPanel Orientation="Horizontal"

Grid.ColumnSpan="2"

>

<TextBox Name="txtEntry"

Width="190"

Height="35"

FontSize="16"

Margin="5"

VerticalContentAlignment="Center"

/>

<Button Grid.Column="1"

Name="btnAdd"

Content="Add"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnAdd\_Click"

/>

<Button Grid.Column="1"

Name="btnDel"

Content="Delete"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnDel\_Click"

/>

<Button Grid.Column="1"

Name="btnClear"

Content="Clear"

Width="50"

Height="50"

HorizontalAlignment="Left"

Margin="5"

Click="btnClear\_Click"

/>

</StackPanel>

</Grid>

</Window>

MainWindow.xaml.cs

using System.Collections;

using System.Windows;

namespace \_12\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

lvEntries.Items.Add(txtEntry.Text);

txtEntry.Clear();

txtEntry.Focus();

}

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

{

~~object item = lvEntries.SelectedItem;~~

var items = lvEntries.SelectedItems;

var result = MessageBox.Show($"Are you sure you want to delete {items.Count} items?", "Sure", MessageBoxButton.YesNo);

if (result == MessageBoxResult.Yes)

{

var itemsList = new ArrayList(items);

foreach (var item in itemsList)

{

lvEntries.Items.Remove(item);

}

}

else

{

}

}

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

{

lvEntries.Items.Clear();

}

}

}