Data binding

https://learn.microsoft.com/en-us/windows/apps/develop/data-binding/data-binding-overview

Main content

- ❑ Basic Data binding
- ❑ Converter
- ❑ INotifyPropertyChanged
- ❑ List data binding: ComboBox, ListView

Basic Data binding

Old way: DataContext & Binding

```
1 reference
private void Window_Activated(object sender,
{
    var sv = new Student
    {
        ID = "001",
        Fullname = "Nguyen Minh Tam"
    };
    layout.DataContext = sv;
}
```

```
class Student
{
    0 references
    public string ID { get; set; }
    0 references
    public string Fullname { get; set; }
}
```

```
<StackPanel x:Name="layout" Orientation="Vertical"
    <TextBlock Text="{Binding ID}" />
    <TextBlock Text="{Binding Fullname}"/>
</StackPanel>
```

001
Nguyen Minh Tam

New way: x:Bind

```
4 references
public Student ViewModel { get; set; }
1 reference
private void Window_Activated(object send
{
    ViewModel = new Student
    {
        ID = "001",
        Fullname = "Nguyen Minh Tam"
    };
}
```

```
<StackPanel x:Name="layout" Orientation="Vertical" Ho
    <TextBlock Text="{x:Bind ViewModel.ID}" />
    <TextBlock Text="{x:Bind ViewModel.Fullname}"/>
</StackPanel>
```

Binding vs x:Bind

Binding: any type, runtime

x:Bind: type strict, compile

Converter

Expected result

```
public class Student
{
    2 references
    public string ID { get; set; } = "";
    2 references
    public string Fullname { get; set; } = "";
    1 reference
    public string Telephone { get; set; } = "";
}
```

```
private void Window_Activated(object sende
{
    ViewModel = new Student
    {
        ID = "001",
        Fullname = "Nguyen Minh Tam",
        Telephone = "0977128732"
    };
}
```

Converter – Step 1 - StringToTelephoneConverter

```
public string Separator { get; set; } = "-";

0 references
public object Convert(object value, Type targetTy
{
    if (value == null) {
        return "";
    }

    string telephone = (string) value;
    var formatted = Regex.Replace(telephone,
    @"(\d{4,5})(\d{3})(\d{3})",
    $"$1{Separator}$2{Separator}$3"
    );
    return formatted;
}

0 references
public object ConvertBack(object value, Type targ
{
    throw new NotImplementedException();
}
```

Step 2: Declaration as a resource

```
<StackPanel x:Name="layout" Orientation="Vertical" HorizontalAlignment="Cent
    <StackPanel.Resources>
        <local:StringToTelephoneConveter x:Key="converter" Separator="-"/>
    </StackPanel.Resources>
    <TextBlock Text="{x:Bind ViewModel.ID}" />
    <TextBlock Text="{x:Bind ViewModel.Fullname}"/>
    <TextBlock DataContext="{x:Bind ViewModel}" Text="{Binding Telephone, Co
</StackPanel>
```

Step 3: Use the Converter

```
<TextBlock DataContext="{x:Bind ViewModel}"
           Text="{Binding Telephone, Converter={StaticResource converter}}"/>
</StackPanel>
```

Multi value converter

C#

```
public class MyMultiValueConverter : IMultiValueConverter {
  public object Convert(object[] values, Type targetType, object parameter, System.Gl
    int firstValue = (int)values[0];
    double secondValue = (double)values[1];
    string thirdValue = (string)values[2];

    return "You said " + thirdValue + ", but it's rather " + firstValue * secondValue
  }

  public object[] ConvertBack(object value, Type[] targetTypes, object parameter, Sys
    throw new NotImplementedException("Going back to what you had isn't supported.");
  }
}
```

XAML

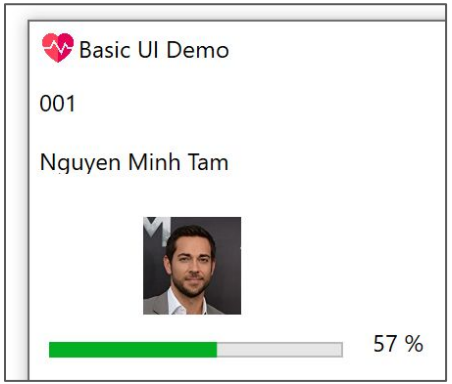
```
<TextBlock.Text>
  <MultiBinding Converter="{StaticResource myNs:MyMultiValueConverter}">
    <Binding Path="FirstValue" />
    <Binding Path="SecondValue" />
    <Binding Path="ThirdValue" />
  </MultiBinding>
</TextBlock.Text>
```

Displaying image using data binding

Using embeded image inside the executable file

```
class Student
{
    1 reference
    public string ID { get; set; }
    1 reference
    public string Fullname { get; set; }
    1 reference
    public int Credits { get; set; }
    0 references
    public string AvatarPath { get; set; }
}
```

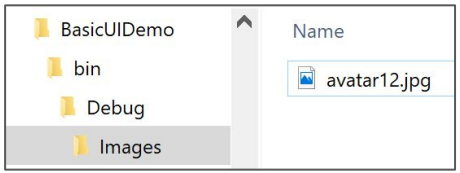
```
<Image Source="{Binding AvatarPath}"
       Width="50" Height="50"
       Canvas.Left="58" Canvas.Top="72"/>
```

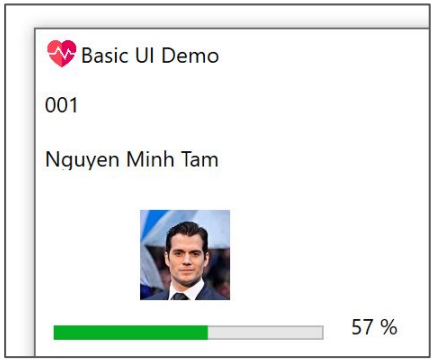


Using absolute path

```
class RelativeToAbsoluteConverter : IValueConverter
{
    1 reference
    public object Convert(object value, Type targetType, object param
    {
        var imageFile = (string)value;
        var currentFolder = AppDomain.CurrentDomain.BaseDirectory;
        return currentFolder + imageFile;
    }
```

```
 <local:RelativeToAbsoluteConverter x:Key="absoluteConverter"/>
indow.Resources>
nvas>
 <Image Source="{Binding AvatarPath,
    Converter={StaticResource absoluteConverter}}"
```





Bonus: Add rounded corner

```
<Border Width="50" Height="50" Canvas.Left="58" Canvas.Top="72"
        CornerRadius="25">
    <Border.Background>
        <ImageBrush ImageSource="{Binding AvatarPath,
            Converter={StaticResource absoluteConverter}}" />
    </Border.Background>
    <Border.Effect>
        <DropShadowEffect />
    </Border.Effect>
</Border>
```

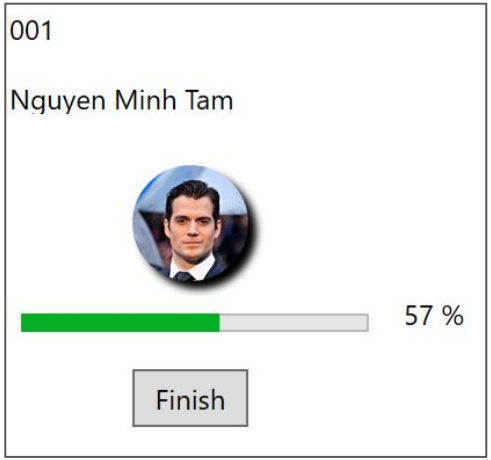
UI Synchronization

Core idea

Underlying *data* change => UI automatically change

Adding button to update

```
<Button Content="Finish" Name="finishButton"
        Width="50" Height="25"
        Click="finishButton_Click"
        Canvas.Left="58" Canvas.Top="160"/>
```



Update the credits when clicked

```
1 reference
private void finishButton_Click(object sender, RoutedEventArgs e)
{
    _sv.Credits = 140;
}
```

We see nothing change in the UI! Why?

Need to implement INotifyPropertyChanged

Underlying code of getter and setter

```
class Student
{
    1 reference
    public string ID { get; set; }
    1 reference
    public string Fullname { get; set; }

    private int _credits; // Backup field
    2 references
    public int Credits {
        get { return _credits; }
        set
        {
            _credits = value;
        }
    }
    1 reference
    public string AvatarPath { get; set; }
}
```

Implementing INotifyPropertyChanged

```
class Student: INotifyPropertyChanged
{
    1 reference
    public string ID { get; set; }
    1 reference
    public string Fullname { get; set; }

    public event PropertyChangedEventHandler PropertyChanged;

    private int _credits; // Backup field
    2 references
    public int Credits {
        get { return _credits; }
        set
        {
            _credits = value;
            PropertyChanged?.Invoke(this,
                new PropertyChangedEventArgs("Credits"));
        }
    }
```

Tools to use rather than manually writing

PropertyChanged.Fody (Need at least VS 2019)

Weaving: injecting functionality into an existing program

- Source code: inject source code lines before the code is compiled
- IL weaving (for .NET) adds the code as IL instructions in the assembly
- ByteCode weaving (for Java) works on the class file

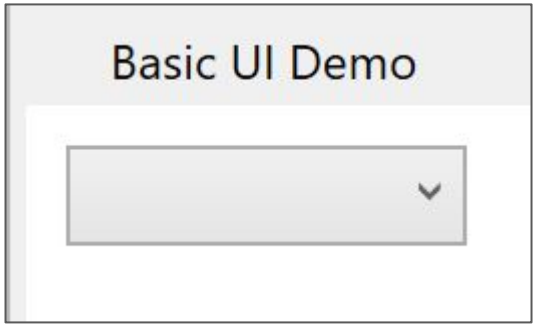
Exercises

1. Do data binding for a **book**: Book's name, Cover's image, Author, Published Year, ISBN
2. Do data binding for a **mobile phone**: Phone's name, Image, Manufacturer, Price
3. Do data binding for an **employee**: Fullname, Email, Address, Telephone number, Avatar's image

Binding a list

Prepare UI using ComboBox

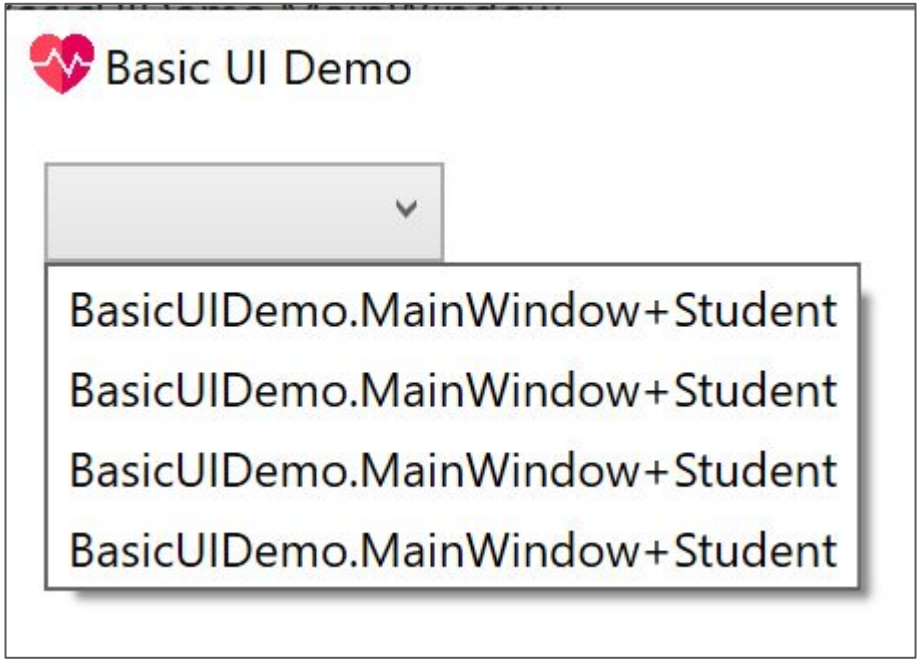
```
<Canvas>
    <ComboBox Width="100" Height="25" Name="studentsComboBox"
              Canvas.Left="10" Canvas.Top="10" >

    </ComboBox>
</Canvas>
```



Sample data

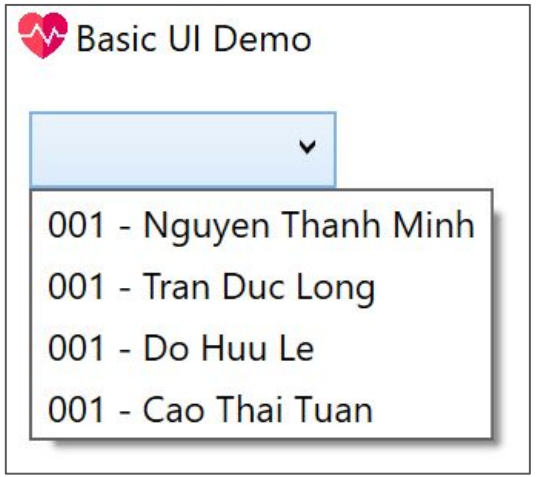
```
List<Student> _list;
1 reference
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    _list = new List<Student>()
    {
        new Student() {ID = "001", Fullname="Nguyen Thanh Minh", Credits=32, AvatarPath="avatar11.jpg"},
        new Student() {ID = "001", Fullname="Tran Duc Long", Credits=12, AvatarPath="avatar12.jpg"},
        new Student() {ID = "001", Fullname="Do Huu Le", Credits=77, AvatarPath="avatar13.jpg"},
        new Student() {ID = "001", Fullname="Cao Thai Tuan", Credits=16, AvatarPath="avatar14.jpg"}
    };
    studentsComboBox.ItemsSource = _list;
}
```

The result


Basic UI Demo
BasicUIDemo.MainWindow+Student
BasicUIDemo.MainWindow+Student
BasicUIDemo.MainWindow+Student
BasicUIDemo.MainWindow+Student

Simple way to modify

```
class Student: INotifyPropertyChanged
{
    0 references
    public override string ToString()
    {
        return $"{ID} - {Fullname}";
    }
```

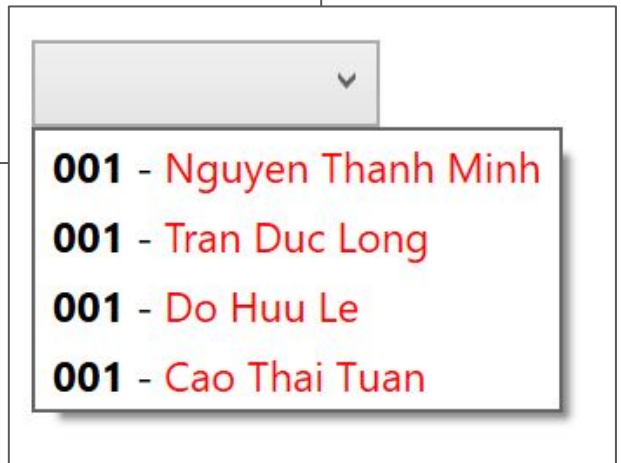


But it should not be this way, why?

Because this class's job is not displaying data

Using ItemTemplate

```
<ComboBox Width="100" Height="25" Name="studentsComboBox"
          Canvas.Left="10" Canvas.Top="10" >
    <ComboBox.ItemTemplate>
        <DataTemplate>
            <StackPanel Orientation="Horizontal">
                <TextBlock Text="{Binding ID}" FontWeight="Bold"/>
                <TextBlock Text=" - "/>
                <TextBlock Text="{Binding Fullname}" Foreground="Red"/>
            </StackPanel>
        </DataTemplate>
    </ComboBox.ItemTemplate>
</ComboBox>
```



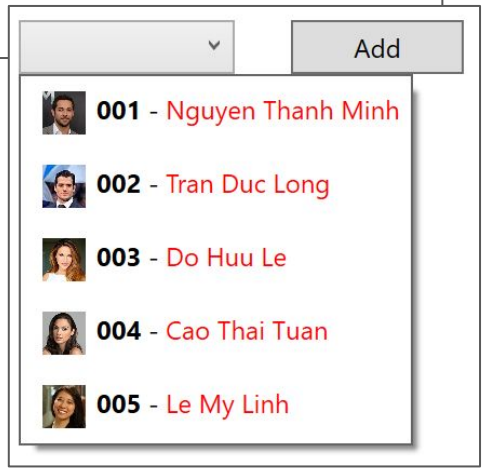
Adding image

```
<DataTemplate>
    <StackPanel Orientation="Horizontal" VerticalAlignment="Center">
        <Image Source="{Binding AvatarPath}" Width="20" Height="20" Margin="5"/>
        <TextBlock Text="{Binding ID}" FontWeight="Bold" Height="20"/>
        <TextBlock Text=" - " Height="20"/>
        <TextBlock Text="{Binding Fullname}" Foreground="Red" Height="20"/>
    </StackPanel>
</DataTemplate>
```

Editing the list

1. Adding a new item

```
private void addStudentButton_Click(object sender, RoutedEventArgs e)
{
    _list.Add(new Student()
    {
        ID = "005",
        Fullname = "Le My Linh", Credits=44, AvatarPath="Images/avatar15.jpg"
    });
}
```



Convert the list to binding list

2. Delete an item

```
private void removeStudentButton_Click(object sender, RoutedEventArgs e)
{
    _list.RemoveAt(1);
}
```

3. Deleting an item - The other way

```
var student = (Student) studentsComboBox.SelectedItem;
_students.Remove(student);
```

4. Update an item

```
private void updateStudentButton_Click(object sender, RoutedEventArgs e)
{
    _list[1].ID = "007";
    _list[1].Fullname = "James Bond";
}
```

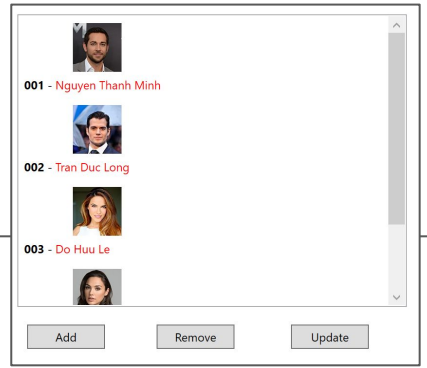
Using ListView

Convert to using ListView

```
<ListView Width="200" Height="300" Name="studentsComboBox"
          Canvas.Left="10" Canvas.Top="10" >
    <ListView.ItemTemplate>
        <DataTemplate>
            <StackPanel Orientation="Horizontal" VerticalAlignment="Center">
                <Image Source="{Binding AvatarPath}" Width="20" Height="20" Margin="5"/>
                <TextBlock Text="{Binding ID}" FontWeight="Bold" Height="20"/>
                <TextBlock Text=" - " Height="20"/>
                <TextBlock Text="{Binding Fullname}" Foreground="Red" Height="20"/>
            </StackPanel>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

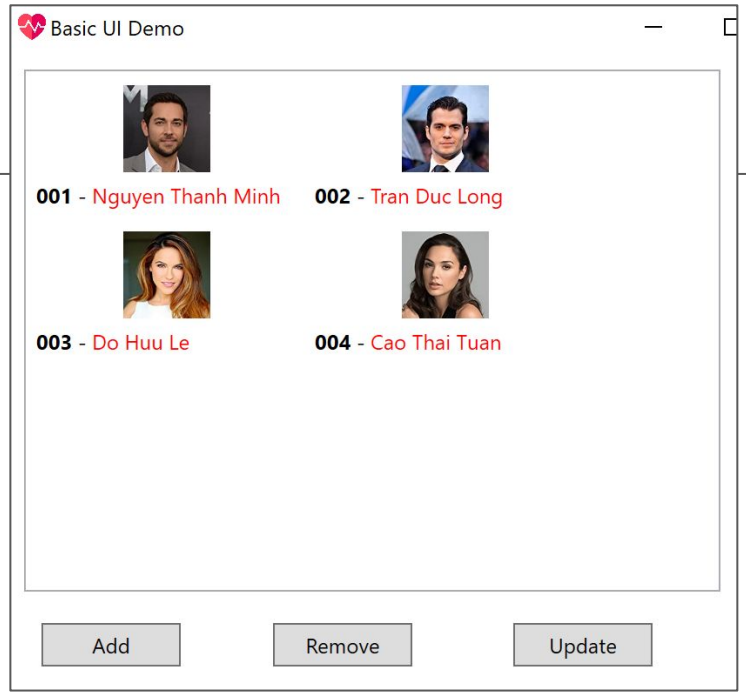
Minor change in displaying

```
<ListView Width="400" Height="300" Name="studentsComboBox"
          Canvas.Left="10" Canvas.Top="10" >
    <ListView.ItemTemplate>
        <DataTemplate>
            <StackPanel VerticalAlignment="Center" Width="150" Height="80">
                <Image Source="{Binding AvatarPath}" Width="50" Height="50" Margin="5"/>
                <StackPanel Orientation="Horizontal" >
                    <TextBlock Text="{Binding ID}" FontWeight="Bold" Height="20"/>
                    <TextBlock Text=" - " Height="20"/>
                    <TextBlock Text="{Binding Fullname}" Foreground="Red" Height="20"/>
                </StackPanel>
            </StackPanel>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```



Using WrapPanel

```
<ListView Width="400" Height="300" Name="studentsComboBox"
          Canvas.Left="10" Canvas.Top="10" ScrollViewer.HorizontalScrollBarVisibility="Disabled">
    <ListView.ItemsPanel>
        <ItemsPanelTemplate>
            <WrapPanel />
        </ItemsPanelTemplate>
    </ListView.ItemsPanel>
```



In class exercises (choose 1 only)

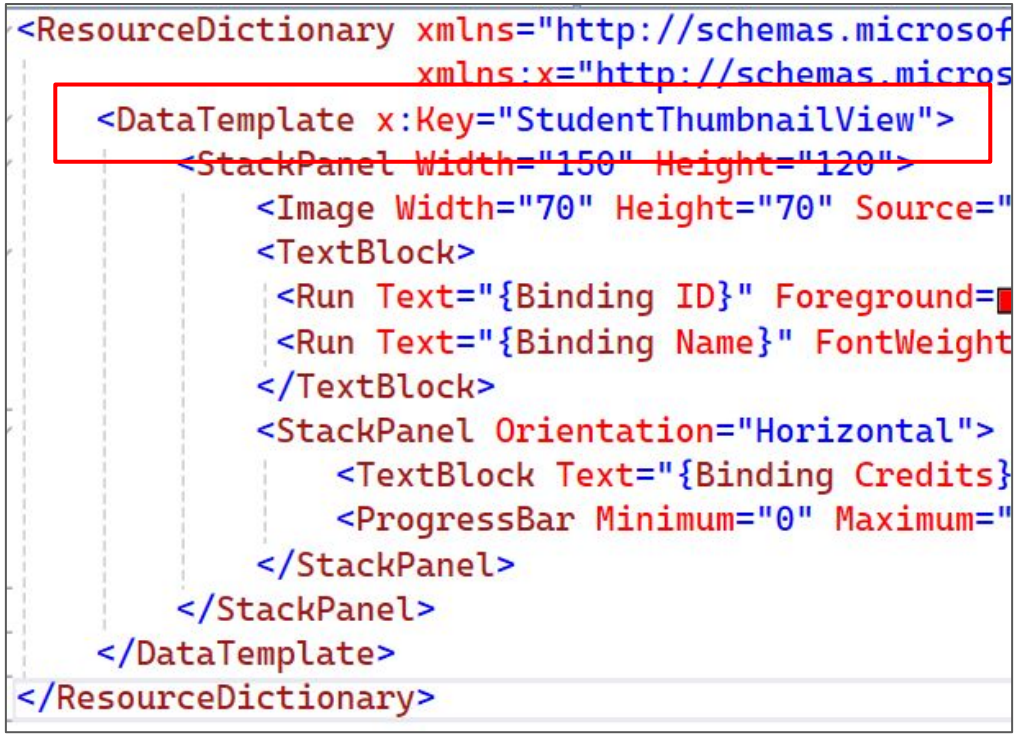
1. Do data binding for a list of **books**: Book's name, Cover's image, Author, Published Year
2. Do data binding for a list of **mobile phones**: Phone's name, Image, Manufacturer, Price
3. Do data binding for a list of **employees**: Fullname, Email, Address, Telephone number, Avatar's image

Prepare at least 22 items

Add, Delete, Update (hard code data)

Saving template for reuse - Step 01

- Extract data template to a Dictionary.xaml file



Extracting template - Step 02

Merge resource into our Window

```
<Window.Resources>
    <ResourceDictionary Source="StudentDictionary.xaml"/>
</Window.Resources>
```

Extracting template - Step 03

Specify the template we want to use

```
<ListView Name="studentsComboBox"
          ItemTemplate="{StaticResource StudentThumbnailView}"
```

Handling context menu

```
<ListView.Resources>
    <ContextMenu x:Key="TenCuaContextMenu">
        <MenuItem Header="Delete" Click="MenuItem_Click"/>
        <MenuItem Header="Edit"/>
    </ContextMenu>
</ListView.Resources>
<ListView.ItemContainerStyle>
    <Style TargetType="{x:Type ListViewItem}" >
        <Setter Property="ContextMenu" Value="{StaticResource
TenCuaContextMenu}" />
    </Style>
</ListView.ItemContainerStyle>
```

Handling double click

Does not work when clicking on empty space, must set event handler using Style

```
<ListView.ItemContainerStyle>
    <Style TargetType="{x:Type ListViewItem}">
        <Setter Property="ContextMenu"
                Value="{StaticResource listViewContextMenu}"/>
        <EventSetter Event="MouseDoubleClick"
                     Handler="listViewItem_DoubleClick"/>
    </Style>
</ListView.ItemContainerStyle>
```

Master-Detail data binding

```
<ComboBox Name="studentsComboBox" Width="200" Height="35" Canvas.Left="38" Canvas.To
    <ComboBox.ItemTemplate...>
</ComboBox>
<StackPanel Canvas.Left="290" Canvas.Top="89"
            DataContext="{Binding ElementName=studentsComboBox,Path=SelectedItem}">
    <TextBlock Text="{Binding ID}" d:Text="001"/>
    <TextBlock Text="{Binding Name}" d:Text="Tran Huy Hoang"/>
</StackPanel>
```