Data binding

Nội dung chính

- Data binding cơ bản
- Converter
- INotifyPropertyChanged
- Binding danh sách

Data binding cơ bản

Binding đối tượng vào màn hình

Giả sử muốn data binding một đối tượng thuộc lớp Sinh viên (Student) với các thuộc tính

- ID: Mã số sinh viên
- Fullname: Họ và tên

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

Kết nối dữ liệu và màn hình

Thuộc tính DataContext (Ngữ cảnh dữ liệu)

```
private void Window_Loaded(object senc
{
    var sv = new Student {
        ID = "001",
        Fullname = "Nguyen Minh Tam"
    };
    this.DataContext = sv;
}
```

```
<Label Content="{Binding ID}" Width="
<Label Content="{Binding Fullname}" W
```

Bổ sung

Giả định trường qui định sinh viên phải tích lũy 140 tín chỉ thì sẽ được tốt nghiệp

Thông tin sinh viên cần bổ sung thêm số tín chỉ hiện tại (currentCredits - int)

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

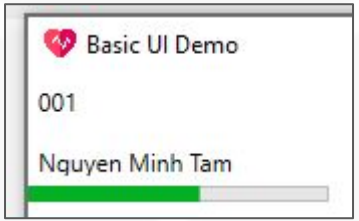
Hiệu chỉnh lại binding để hiển thị

```
private void Window_Loaded(object send
{
    var sv = new Student {
        ID = "001",
        Fullname = "Nguyen Minh Tam",
        Credits = 80
    };
    this.DataContext = sv;
}
```

```
<Label Content="{Binding ID}" Width=
<Label Content="{Binding Fullname}"
<Label Content="{Binding Credits}" W
```

Có cách hiển thị nào tốt hơn không? ProgressBar

```
<Label Content="{Binding ID}" Width="15
<Label Content="{Binding Fullname}" Wid
<ProgressBar Value="{Binding Credits}"
             Minimum="0" Maximum="140"
             Width="150" Height="8"
             Canvas.Top="55"/>
```



Tốt hơn được không? Thêm tỉ lệ phần trăm?

Converter

Tạm thời chỉ quan tâm chiều đi, chuyển số tín chỉ sang %

```
class CreditsToPercentageConverter : IValueConverter
{
    0 references
    public object Convert(object value, Type targetTyp
    {
        int credits = (int) value ;
        double percentage = credits * 100 / 140;
        int number = (int)percentage;
        return $"{number} %";
    }
```

Khai báo converter như một tài nguyên

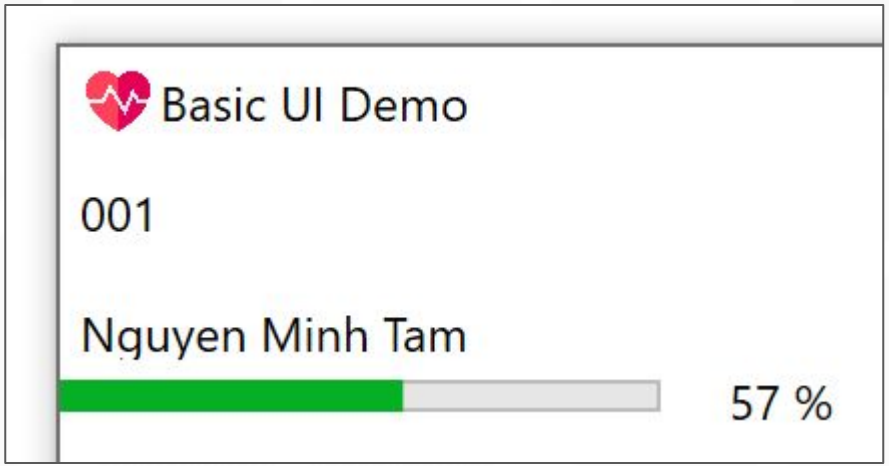
```
<Window.Resources>
    <local:CreditsToPercentageConverter x:Key="percentConverter" />
</Window.Resources>
```

Có thể hiểu cách viết trên tương đương khai báo một biến tên là ***percentConverter*** kiểu CreditsToPercentageConverter

```
var percentConverter = new CreditsToPercentageConverter();
```

Sử dụng tài nguyên Converter

```
<TextBlock Text="{Binding Credits,
                 Converter={StaticResource percentConverter}}"/>
```



Tốt hơn nữa? Biến số tín chỉ thành tham số

```
0 references
class CreditsToPercentageConverter : IValueConverter
{
    1 reference
    public int MaxCredits { get; set; }

    0 references
    public object Convert(object value, Type targetType,
    {
        int credits = (int) value ;
        double percentage = credits * 100 / MaxCredits;
        int number = (int)percentage;
        return $"{number} %";
    }
```

```
<Window.Resources>
    <local:CreditsToPercentageConverter
        x:Key="percentConverter" MaxCredits="140" />
</Window.Resources>
```

Bài tập vận dụng

Biết một sinh viên có **thêm** thông tin địa chỉ, email, số điện thoại.

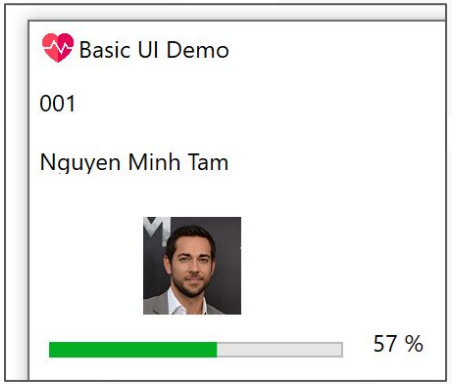
Hãy xây dựng 1 màn hình hiển thị thông tin sinh viên với đầy đủ các thông tin trên sử dụng data binding.

Hiển thị hình ảnh lấy từ project

Sử dụng data binding để hiển thị ảnh nhúng trong tập tin nhị phân của chương trình

```
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"/>
```

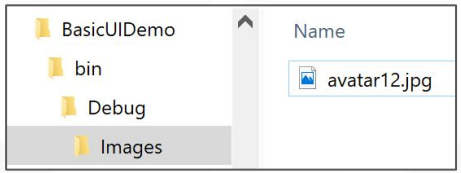


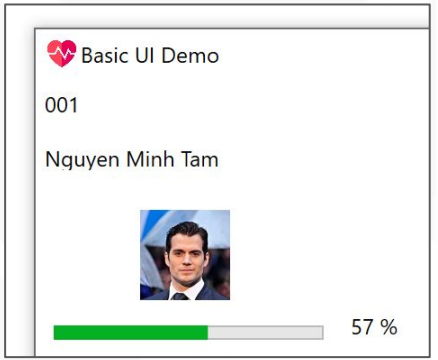
Hiển thị hình ảnh từ đường dẫn bên ngoài

Tự viết converter chuyển đổi đường dẫn

```
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}}"
```





Mở rộng - Thêm biên bo tròn cho avatar

Thêm bo tròn cho ảnh

```
<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>
```

Đồng bộ UI và dữ liệu

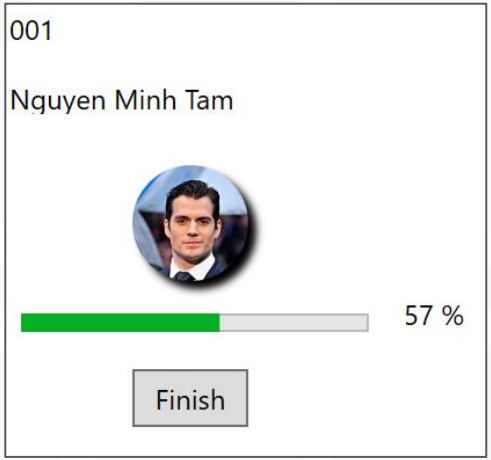
Ý tưởng chính

Khi data bên dưới thay đổi thì giao diện tự động thay đổi

Thêm chức năng

Thêm nút bấm cho biết sinh viên đã hoàn thành số tín chỉ học được của năm học

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



Thay đổi sinh viên hiện tại thành private member

```
Student _sv;
1 reference
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    _sv = new Student {
        ID = "001",
        Fullname = "Nguyen Minh Tam",
        Credits = 80,
        AvatarPath = "Images/avatar12.jpg"
    };
    this.DataContext = _sv;
}
```

Khi bấm nút Finish thì cập nhật lại biến _sv

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

Tuy nhiên giao diện vẫn không thay đổi

Cần cài đặt giao diện INotifyPropertyChanged

Bản chất getter và 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; }
}
```

Cài đặt giao diện 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"));
        }
    }
```

Bài tập vận dụng

Tự viết tiếp các hàm setter để cập nhật cho các thuộc tính còn lại

Cảm thấy chán vì công việc lặp lại tẻ nhạt?

Cấp 1: Tách thành hàm để tái sử dụng

Cấp 2: Dùng CallerMemberName

Cấp 3: Tự động chèn thông tin với Fody.PropertyChanged (cần VS 2019 trở lên)

Bài tập vận dụng

1. Thực hiện binding hiển thị một quyển sách với các thông tin: Tên sách, ảnh bìa sách, tác giả, năm xuất bản
2. Thực hiện binding hiển thị một điện thoại với các thông tin: Tên điện thoại, Hình ảnh, Tên hãng, Giá bán
3. Thực hiện binding hiển thị một nhân viên với các thông tin: Họ và tên, Email, Địa chỉ, số điện thoại, Avatar

Mở rộng - Tự binding với lớp Window hiện tại

Đôi khi ta muốn binding một thuộc tính nào đó bên trong lớp cửa sổ hiện tại

Cú pháp này nhìn sẽ hơi lạ

this.**DataContext** = this

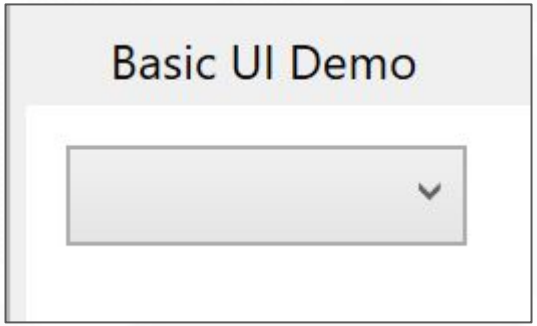
Binding danh sách

Chuẩn bị giao diện để binding

Tạo ra ComboBox

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

    </ComboBox>
</Canvas>
```

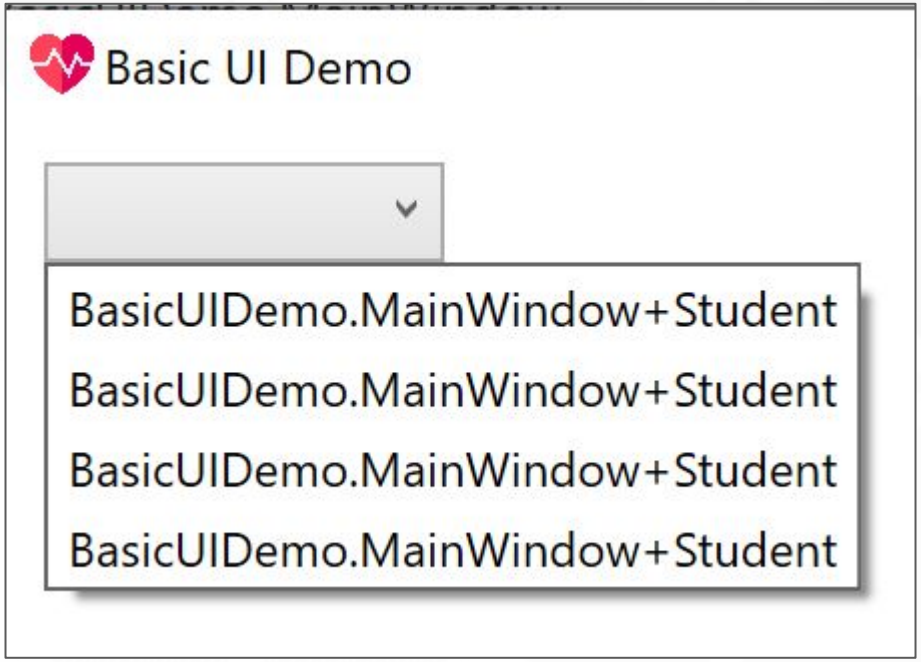


Chuẩn bị dữ liệu mẫu danh sách sinh viên

Chuẩn bị mảng sinh viên gồm 4 sinh viên và 4 avatar

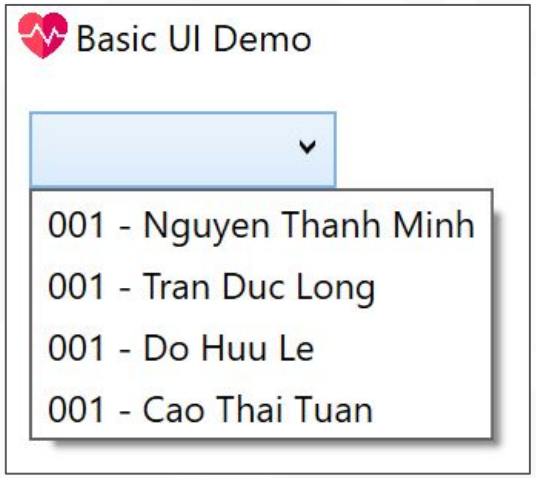
```
List<Student> _list;
1 reference
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    _list = new List<Student>()
    {
        new Student() {ID = "001", Fullname="Nguyen Thanh Minh", Credits=32, AvatarPath="avatar11.jpg"},
        new Student() {ID = "001", Fullname="Tran Duc Long", 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;
}
```

Xem kết quả


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

Chỉnh sửa kết quả - Cách đơn giản nhất

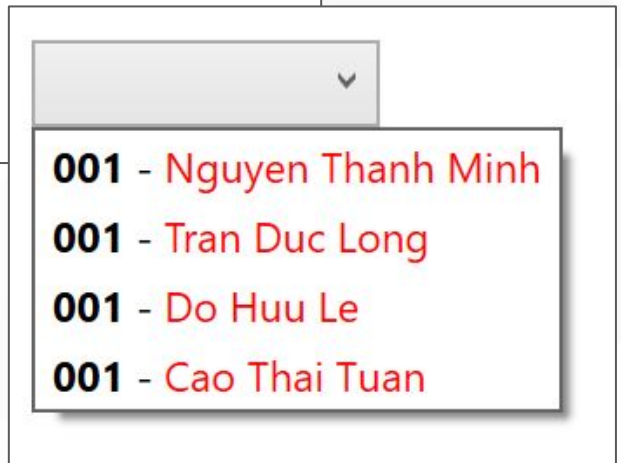
```
class Student: INotifyPropertyChanged
{
    0 references
    public override string ToString()
    {
        return $"{ID} - {Fullname}";
    }
```



Tuy nhiên lớp Student chỉ là lớp chứa dữ liệu, việc hiển thị như thế nào nên nằm ở code chịu trách nhiệm giao diện, là file xaml

Tạo ra ItemTemplate cho mỗi Item của ComboBox

```
<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>
```



Cải tiến thêm hình ảnh

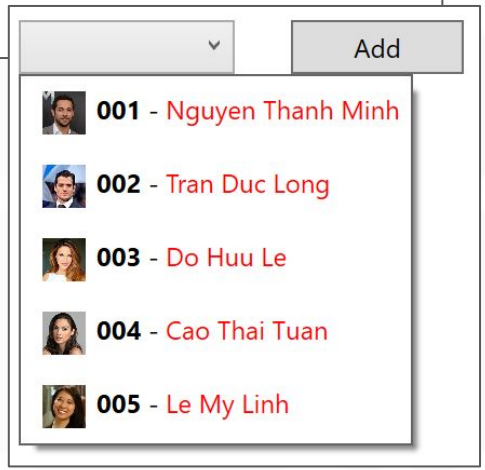
```
<DataTemplate>
    <StackPanel Orientation="Horizontal" VerticalAlignment="Center">
        <Image Source="{Binding AvatarPath}" Width="20" Height="20" Margin="5"/>
        <TextBlock Text="{Binding ID}" FontWeight="Bold" Height="20"/>
        <TextBlock Text=" - " Height="20"/>
        <TextBlock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

Thay đổi danh sách

Thêm phần tử vào danh sách

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

Chuyển danh sách thành binding list



Xóa một phần tử

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

Cập nhật thông tin một phần tử

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

Bài tập vận dụng

1. Thực hiện binding hiển thị ds các quyển sách với các thông tin: Tên sách, ảnh bìa sách, tác giả, năm xuất bản
2. Thực hiện binding hiển thị ds các điện thoại với các thông tin: Tên điện thoại, Hình ảnh, Tên hãng, Giá bán
3. Thực hiện binding hiển thị ds các nhân viên với các thông tin: Họ và tên, Email, Địa chỉ, số điện thoại, Avatar

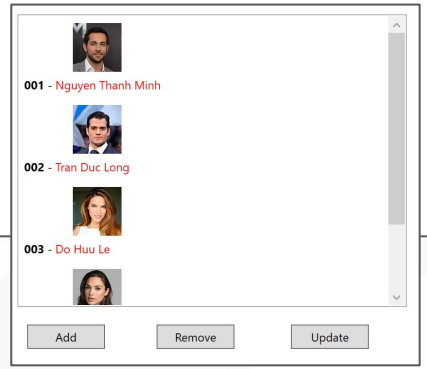
Sử dụng ListView

Chuyển từ ComboBox sang 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>
```

Thay đổi nhẹ cách hiển thị

```
<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>
```



Sử dụng WrapPanel

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