Lab 06:

LẬP TRÌNH C# VỚI ỨNG DỤNG WPF CỦA VS.NET

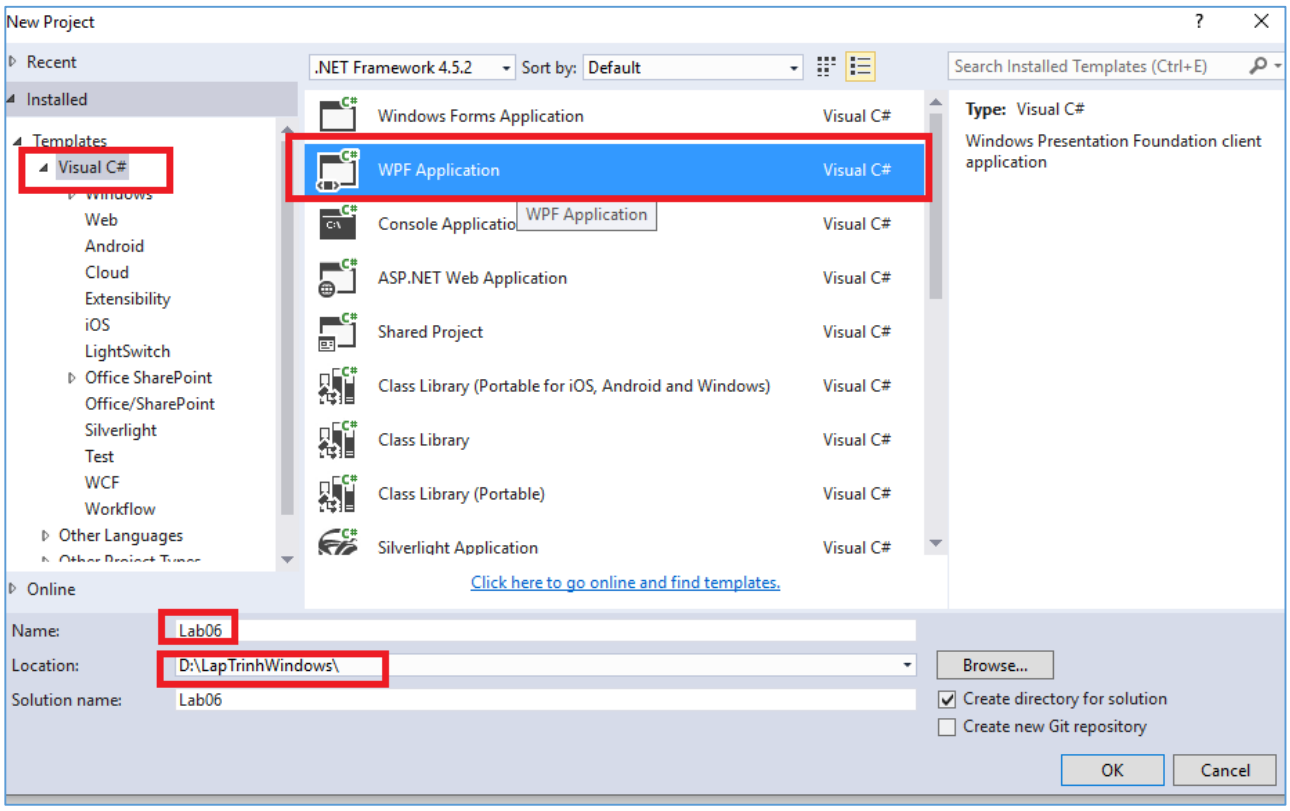
A. MỤC TIÊU:

- ✓ Giới thiệu các công cụ cần thiết để phát triển ứng dụng WPF và giúp học viên làm quen với WPF bằng việc hướng dẫn phát triển một ứng dụng đơn giản cụ thể.
- ✓ Tìm hiểu ngôn ngữ đặc tả XAML cho các control như Button, TextBlock, và các Panel (Grid, StackPanel, WrapPanel, DockPanel)
- ✓ Xử lý sự kiện trên XAML
- ✓ Menu và ListView

B. HƯỚNG DẪN:

✓ WPF, viết tắt của Windows Presentation Foundation, là hệ thống API mới hỗ trợ việc xây dựng giao diện đồ hoạ trên nền Windows

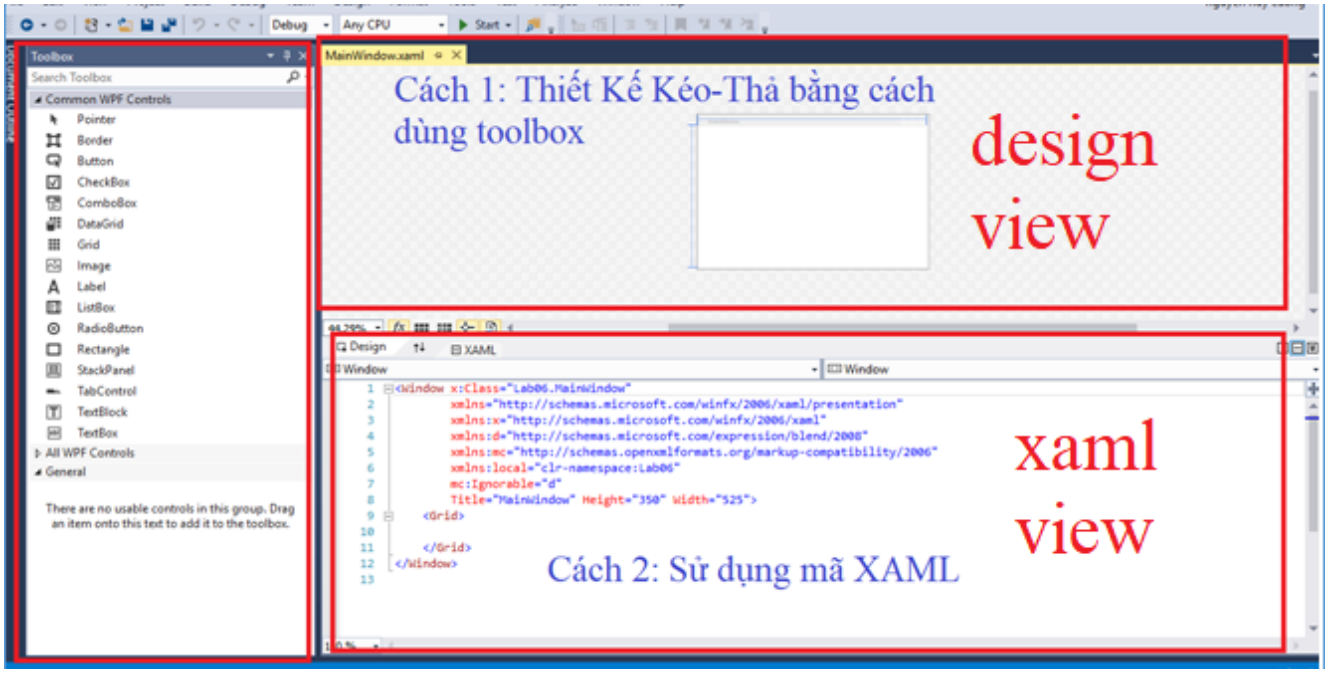
- ✓ Tạo ứng dụng WPF
  - Vào File chọn New > Project. Trong hộp thoại New Project chọn Visual C#, WPF Application, đặt tên dự án là **Lab06** trong ô Name, và chọn vị trí lưu ứng dụng trong Location.



Hình 1: Tạo ứng dụng WPF

- VS sẽ phát sinh 2 tập tin mặc định hai tập tin ***MainWindow.xaml*** và ***MainWindow.xaml.cs***

✓ Thiết kế giao diện WPF



Hình 2: Thiết kế giao diện trong WPF

✓ Xem xét đoạn mã XAML của ứng dụng (Theo cấu trúc XML- có thẻ mở và đóng)

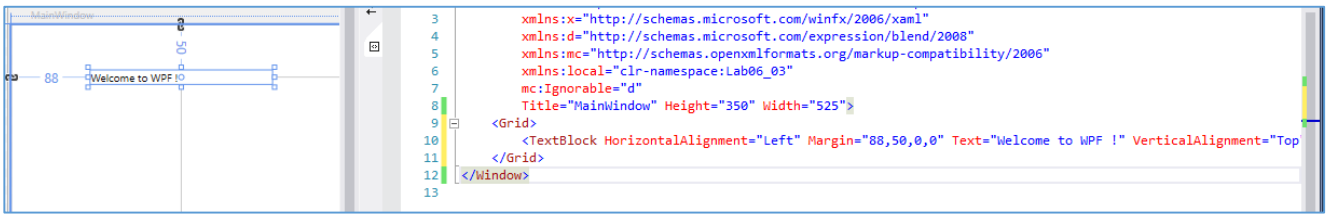
```
<Window x:Class="Lab06.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Lab06"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <Grid>

    </Grid>
</Window>
```

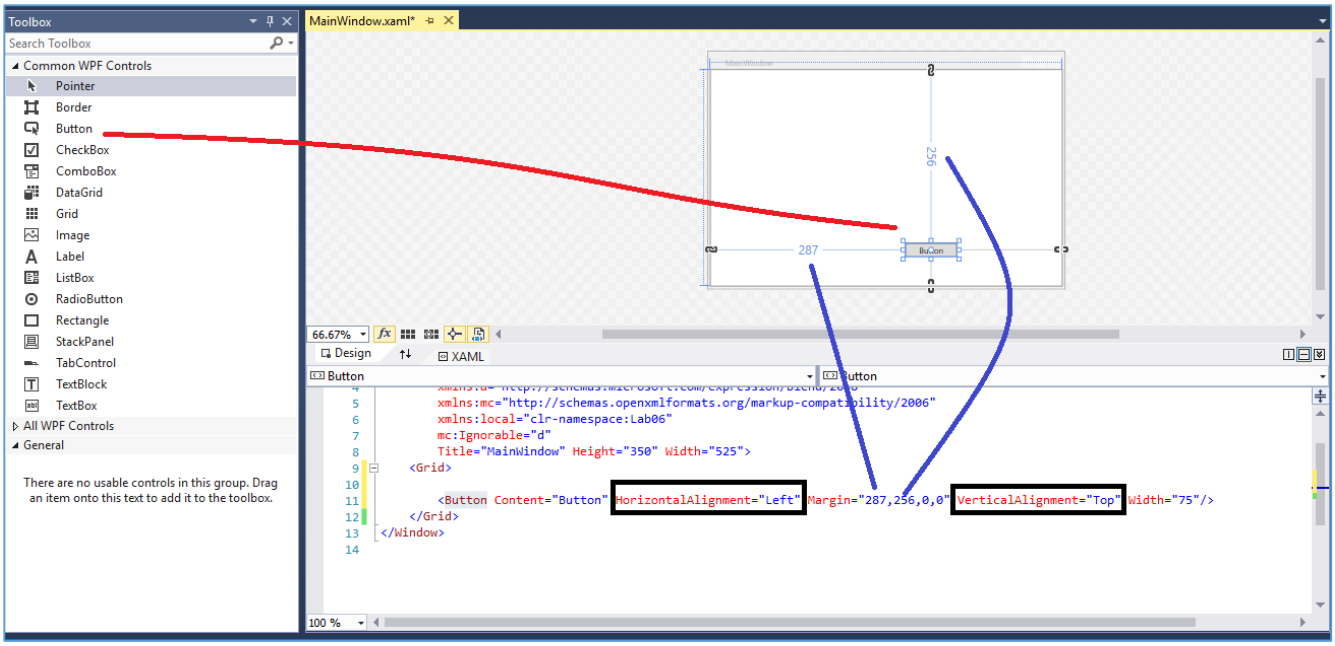
- Giao diện được tạo bởi phần tử **Window** chứa các thuộc tính (***properties***) sau
  - ***Class:*** Xác định lớp thực thi form ứng dụng. Trong trường hợp này là lớp MainWindow được chứa trong namespace Lab06
  - **xmlns:** namespace **XML** để định nghĩa các lược đồ được dùng bởi WPF
  - **Title:** Tiêu đề của form
  - ***Height*** và ***Width***: xác định chiều cao và chiều rộng mặc định cho form.

  Thành phần của giao diện là các thẻ con của windows
  - **Grid** gồm các cột và hàng giúp ta dễ dàng trong việc thiết kế giao diện. Ngoài Grid còn có các kiểu ***WrapPanel / StackPanel / DockPanel / Canvas***.

✓ Tìm hiểu ***TextBlock*** : Giống như Label trong windows forms



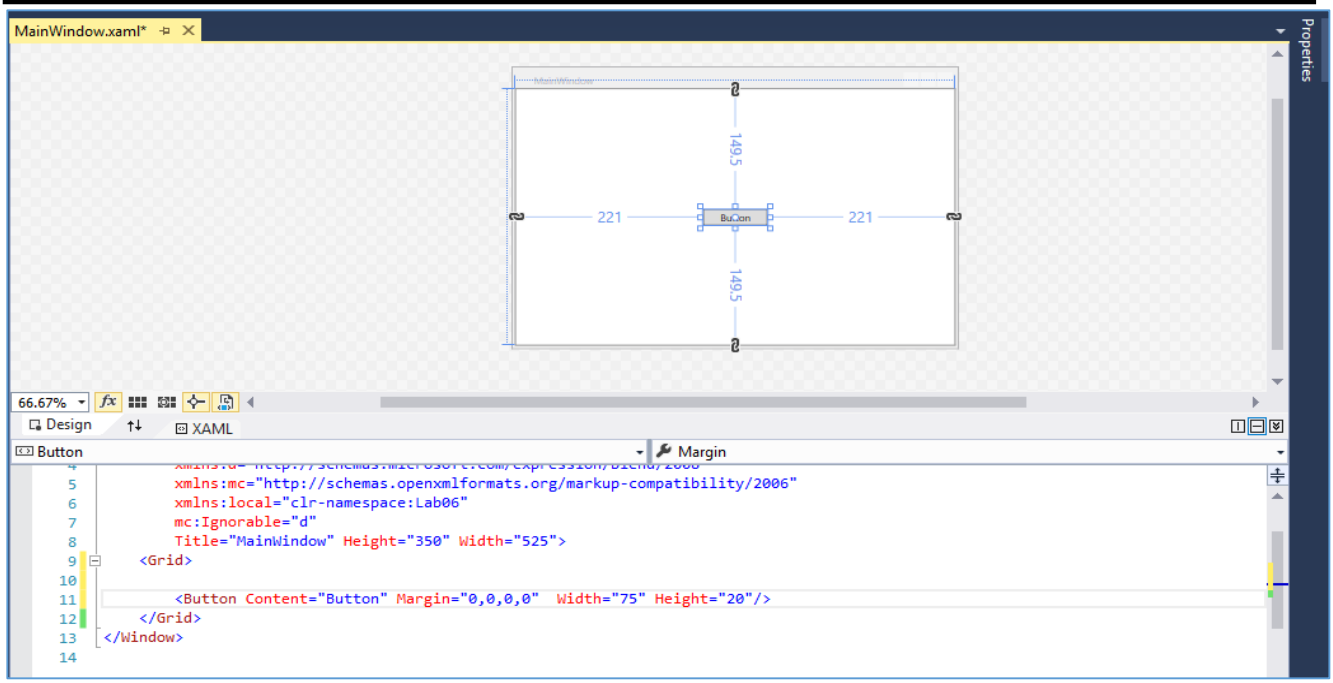
✓ Thêm **Button** vào Forms từ kéo thả ở toolbox



Khi chạy ứng dụng nếu chúng ta thay đổi kích cỡ form thì khoảng cách từ button đến cạnh trên và cạnh trái vẫn không thay đổi (Do các thuộc tính được neo theo ***HorizontalAlignment*** (cạnh trái hay phải) và **VerticalAlignment** (trên / dưới). Khi kéo thả nhiều control thì tương đối khó điều chỉnh giao diện

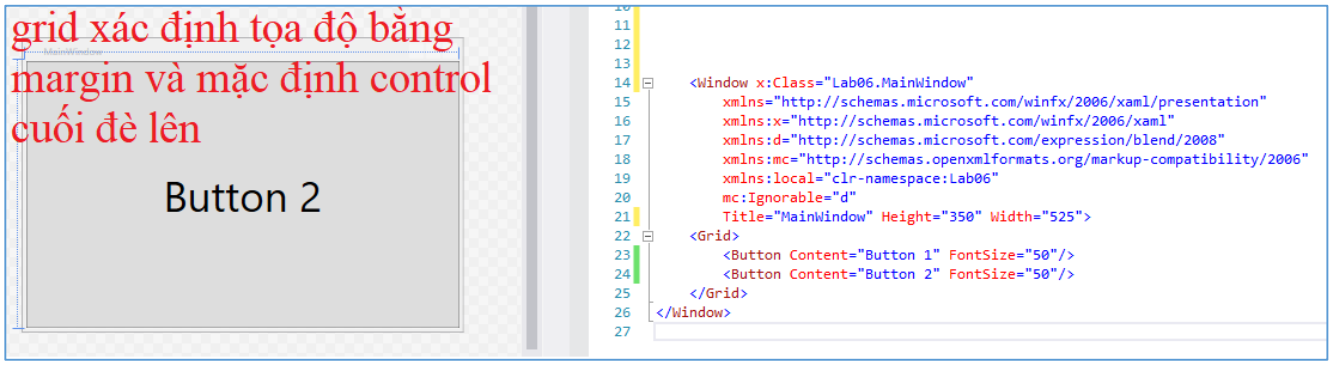
✓ Chỉnh sửa lại Button vào forms từ XAML

```
<Button Content="Button" Margin="0,0,0,0"  Width="75" Height="20"/>
```



Lúc này button luôn nằm giữa form và khi chúng ta thay đổi kích cỡ form button sẽ tự điều chỉnh vị trí của mình một cách tương đối với kích cỡ form sao cho vị trí của button luôn là giữa form:

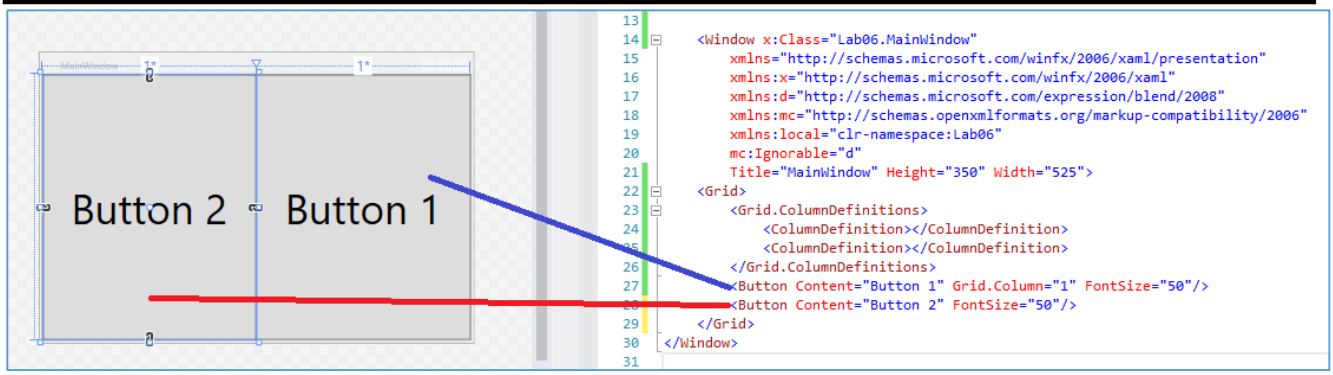
- ✓ **Sử dụng Grid trong WPF**



- Có thể chia Grid theo dòng, cột. Control mặc định luôn ở cột chia đầu tiên

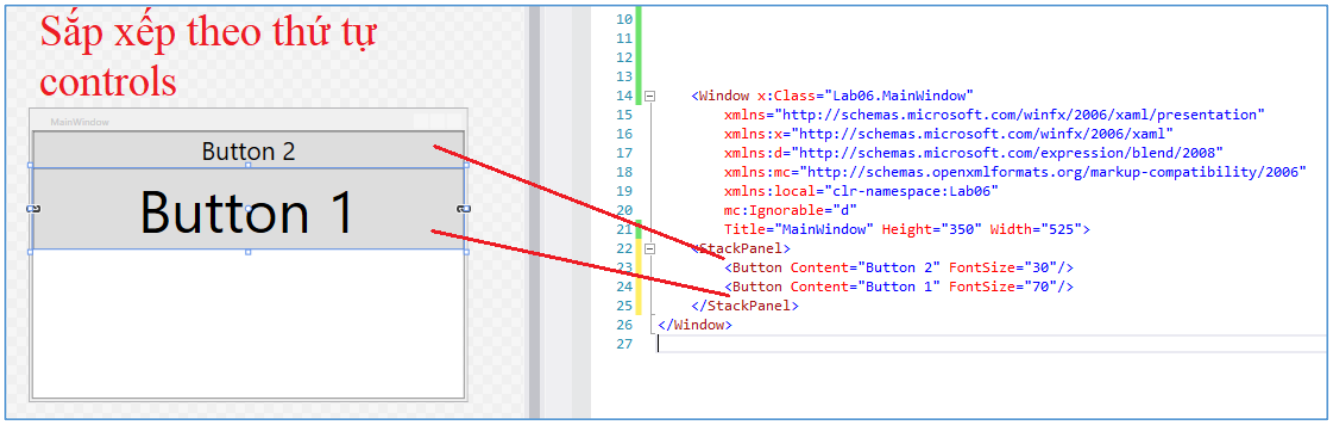
Khi chia Grid ra theo 2 cột, Các controls được mặc định hiện thị trong cột đầu tiên

- Sử dụng các ***Property*** để điều chỉnh vị trí thích hợp. Ví dụ để Button1 sang cột 2



Tương tự cho dòng `<Grid.RowDefinitions>`

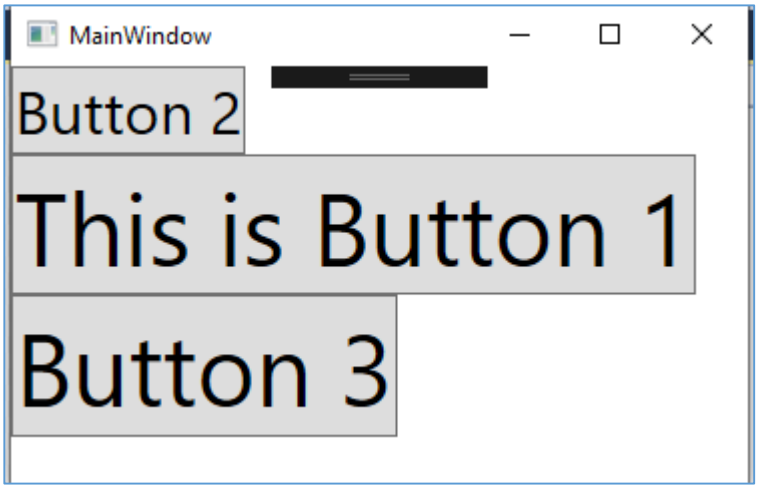
- ✓ Sử dụng ***StackPanel*** : Chia theo thứ tự các controls vừa đủ kích thước và mặc định trên chiều dọc (`Orientation="Vertical"`)



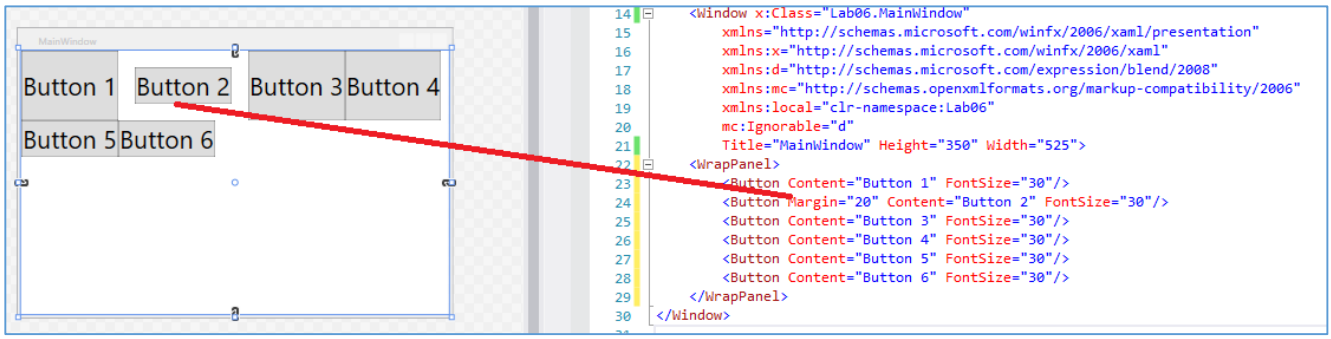
- Để chia theo chiều ngang sử dụng `Orientation="Horizontal"`

- ✓ Sử dụng ***WrapPanel*** : Khi thực thi nếu controls hiện thị không đủ chỗ sẽ tự động wrap xuống dòng. (Mặc định theo chiều ngang `Orientation="Horizontal"`)

Kết quả khi thực thi

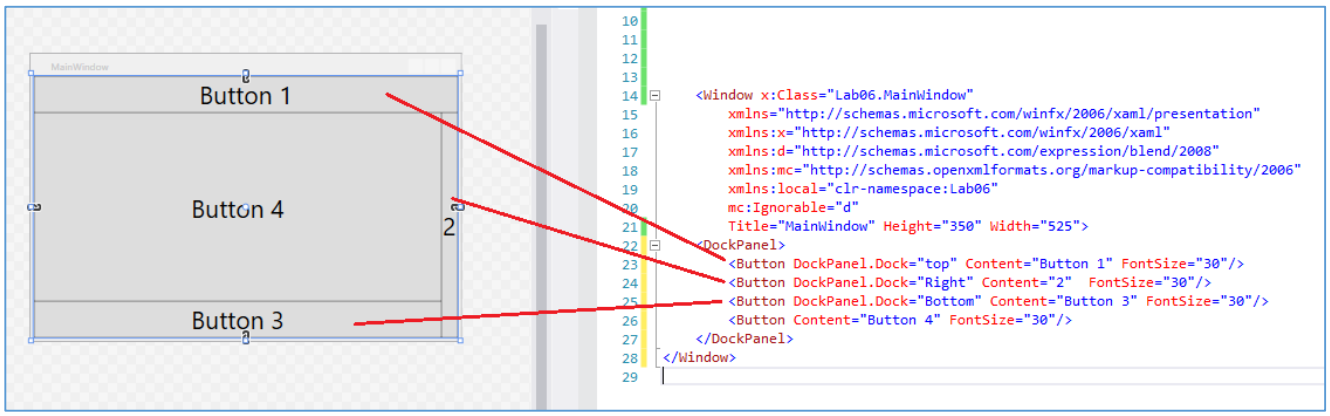


- Để sử dụng giãn cách giữa các button sử dụng ***Margin***

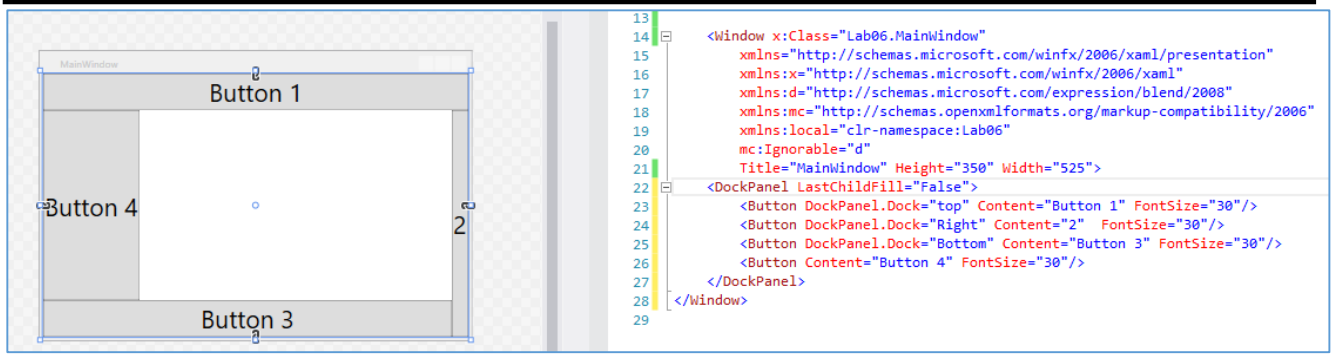


- Tương tự `Orientation="Vertical"` để chia theo chiều dọc

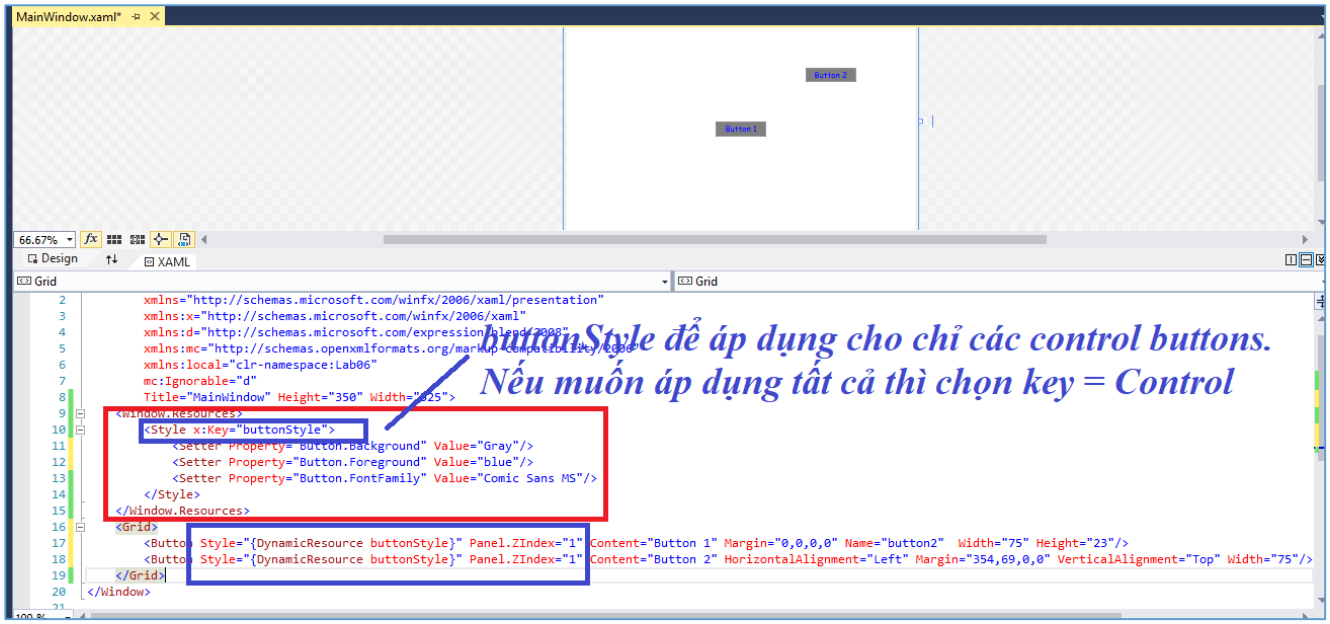
✓ Sử dụng DockPanel: Mặc định Dock là left,không nằm đè lên nhau



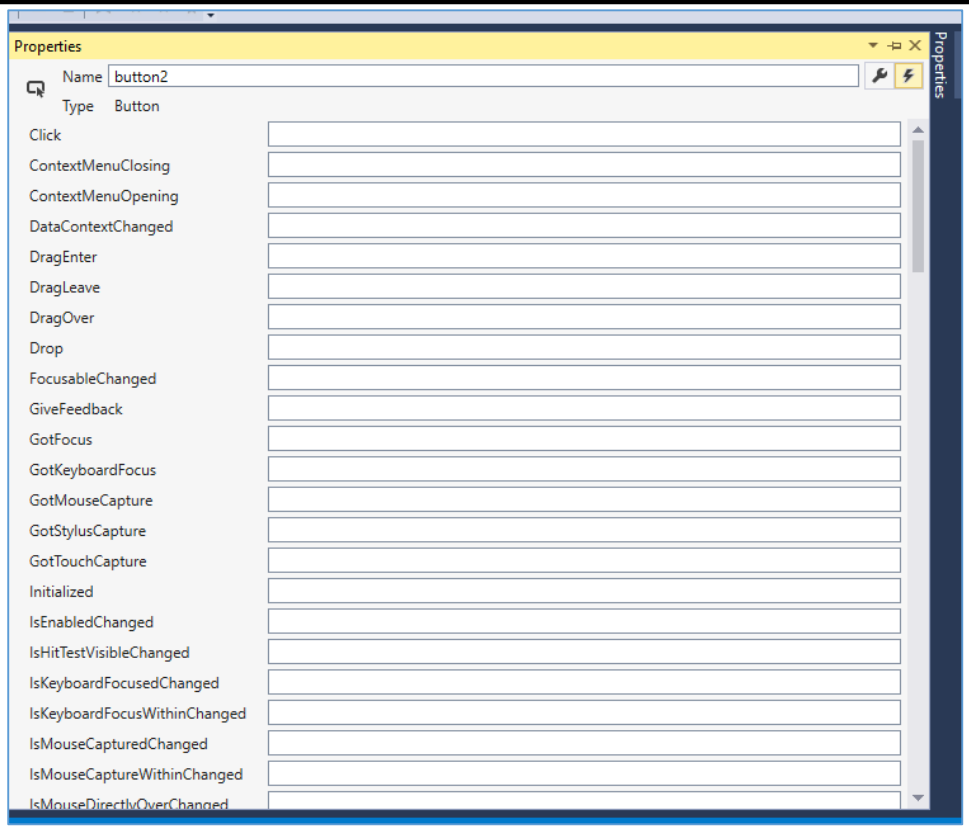
- Thuộc tính `LastChildFill="False"` Để không tự động làm đầy control cuối cùng vào Panel



✓ Sử dụng ***Style*** trong WPF: Khi các controls có cùng style sheet (như css ở web) chúng ta nên sử dụng ***Window.Resources*** phía trên phần tử **Grid**

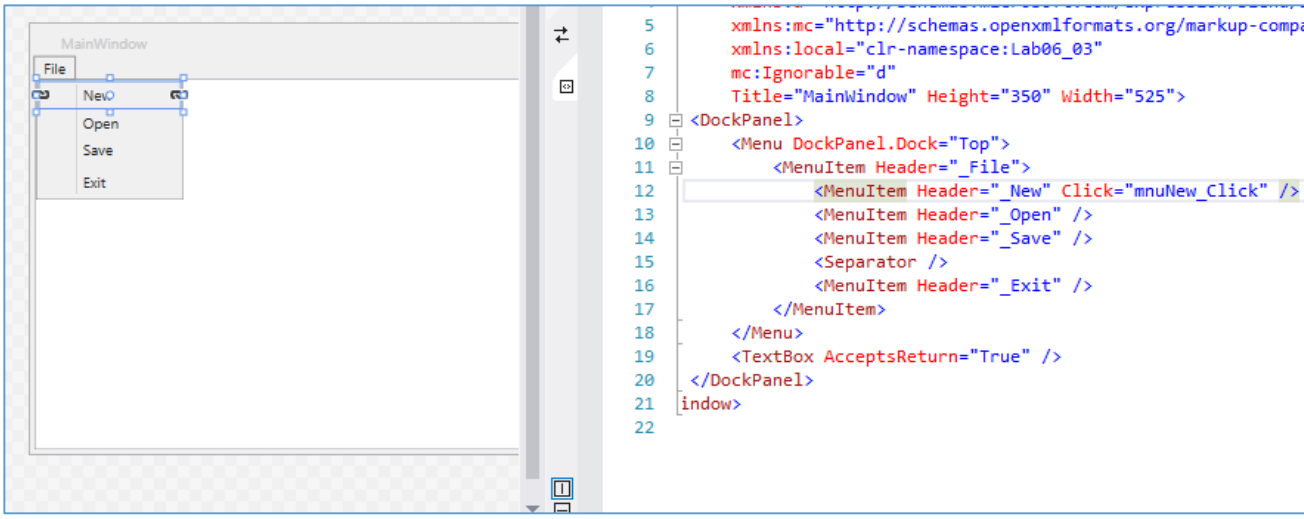


✓ Xử lý sự kiện với WPF: WPF cung cấp các sự kiện được định nghĩa sẵn như Click, MouseOver, v.v. cho các form và điều khiển. chúng ta có thể xem danh sách các sự kiện bằng cách vào cửa sổ Properties và chọn biểu tượng tia chớp:



Chúng ta double click vào sự kiện cần viết (hoặc với button thì double click) để VS tạo ra Events

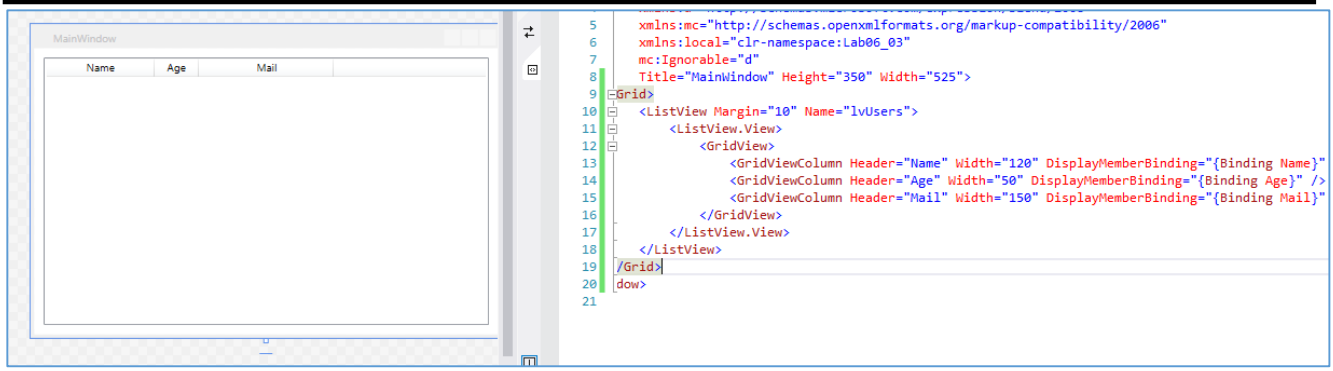
- ✓ Menu trên WPF (MenuStrip trên windows form): Sử dụng <Menu>



Event cho sự kiện New

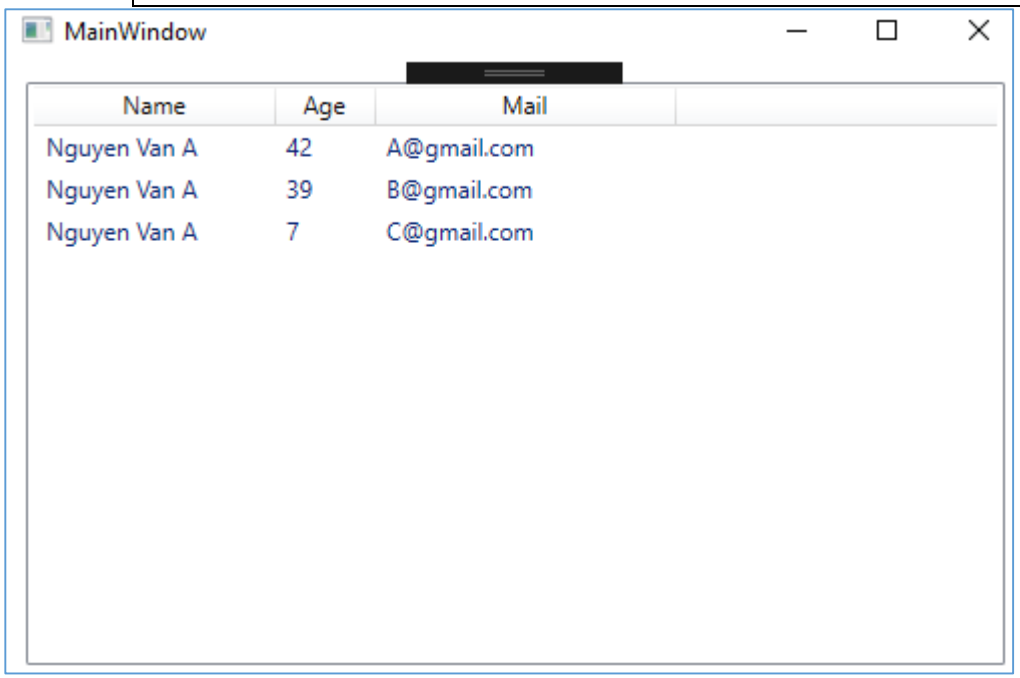
```
private void mnuNew_Click(object sender, RoutedEventArgs e)
{
        MessageBox.Show("New");
}
```

- ✓ ListView trên WPF
  - Sử dụng <***ListView***>



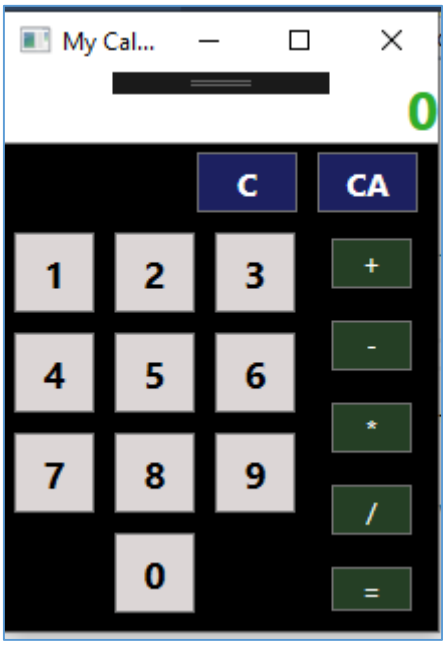
- Đổ dữ liệu vào ListView : Sử dụng ***ItemSource***

```
            List<User> items = new List<User>();
            items.Add(new User() { Name = "Nguyen Van A", Age = 42, Mail =
"A@gmail.com" });
            items.Add(new User() { Name = "Nguyen Van A", Age = 39, Mail =
"B@gmail.com" });
            items.Add(new User() { Name = "Nguyen Van A", Age = 7, Mail =
"C@gmail.com" });
        lvUsers.ItemsSource = items;
```



C. BÀI TẬP

Bài tập 1: Thiết kế giao diện MyCaculator trên ứng dụng WPF như dưới đây



- ✓ Các button số ***1,2,3,4,5,6,7,8,9,0*** cùng 1 buttonSheet
- ✓ Các button + - * / = có cùng 1 buttonSheet
- ✓ 2 Button ***C*** (xóa 1 kí tự cuối cùng của dãy nhập) và ***CA*** (Xóa dãy đã nhập về 0) có cùng 1 buttonSheet

Hướng Dẫn: SV Thử thiết kế bằng WPF

- ***Thiết kê buttonSheet***

```
<Window.Resources>
        <Style x:Key="ButtonMiscStyle" TargetType="{x:Type Button}">
            <Setter Property="Margin" Value="5"/>
           <Setter Property="Background" Value="#FF1C2060"/>
            <Setter Property="Foreground" Value="White"/>
           <Setter Property="Content" Value="C"/>
            <Setter Property="Width" Value="50"/>
            <Setter Property="Height" Value="30"/>
            <Setter Property="FontSize" Value="16"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="VerticalAlignment" Value="Stretch"/>
            <Setter Property="HorizontalAlignment" Value="Stretch"/>
            <Setter Property="FontStretch" Value="Normal"/>
            <Setter Property="Focusable" Value="False"/>
        </Style>
        <Style x:Key="NumberButtonStyle" TargetType="{x:Type Button}">
            <Setter Property="Width" Value="40"/>
            <Setter Property="Height" Value="40"/>
            <Setter Property="Background" Value="#FFDCD6D6"/>
            <Setter Property="Content" Value="1"/>
           <Setter Property="Focusable" Value="False"/>
            <Setter Property="FontSize" Value="18.667"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="HorizontalAlignment" Value="Center"/>
            <Setter Property="VerticalAlignment" Value="Center"/>
```

```
            <Setter Property="Margin" Value="5"/>
        </Style>
        <Style x:Key="ButtonOperatorStyle" TargetType="{x:Type Button}">
            <Setter Property="Background" Value="#FF253F25"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="Width" Value="40"/>
            <Setter Property="Height" Value="25"/>
            <Setter Property="Margin" Value="5,8"/>
            <Setter Property="Focusable" Value="False"/>
            <Setter Property="FontSize" Value="13.333"/>
            <Setter Property="FontWeight" Value="Normal"/>
           <Setter Property="Content" Value="+"/>
        </Style>
       </Window.Resources>
```

- ***Sử dụng Grid chia cột dòng phù hợp (ở ví dụ này cố ý làm Grid để thấy sự phức tạp)***

```
<Grid x:Name="LayoutRoot" Background="Black">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <TextBlock x:Name="txtNumberDisplay" Text="0" d:LayoutOverrides="Height"
TextAlignment="Right" FontSize="26.667" FontWeight="Bold" Background="White"
Foreground="#FF2EAD2A" Grid.ColumnSpan="2"/>
        <DockPanel x:Name="MiscControlsPanel" Background="Black" Grid.Row="1" Margin="5"
Grid.ColumnSpan="2" >
            <Button x:Name="ButtonClearAll" Content="CA" Style="{DynamicResource
ButtonMiscStyle}" Margin="5,0" DockPanel.Dock="Right" Click="ButtonClearAll_Click"/>
            <Button x:Name="ButtonClear" Style="{DynamicResource ButtonMiscStyle}"
Click="ButtonClear_Click" Margin="5,0" DockPanel.Dock="Right" HorizontalAlignment="Right"
/>
        </DockPanel>
        <Grid x:Name="GridNumberButtons" HorizontalAlignment="Left" Grid.Row="2"
Background="Black">
            <Grid.ColumnDefinitions>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
            </Grid.RowDefinitions>
            <Button x:Name="Button1" Style="{DynamicResource NumberButtonStyle}"
Click="Button_Click"/>
            <Button x:Name="Button2" Content="2" Grid.Column="1"
HorizontalAlignment="Left" Style="{DynamicResource NumberButtonStyle}"
Click="Button_Click"/>
            <Button x:Name="Button3" Content="3" Grid.Column="2" Style="{DynamicResource
NumberButtonStyle}" Click=" ButtonNumber_Click"/>
            <Button x:Name="Button4" Content="4" Grid.Row="1" Style="{DynamicResource
NumberButtonStyle}" Click="ButtonNumber_Click"/>
            <Button x:Name="Button5" Content="5" Grid.Row="1" Grid.Column="1"
Style="{DynamicResource NumberButtonStyle}" Click="ButtonNumber_Click"/>
            <Button x:Name="Button6" Content="6" Grid.Row="1" Grid.Column="2"
Style="{DynamicResource NumberButtonStyle}"/>
```

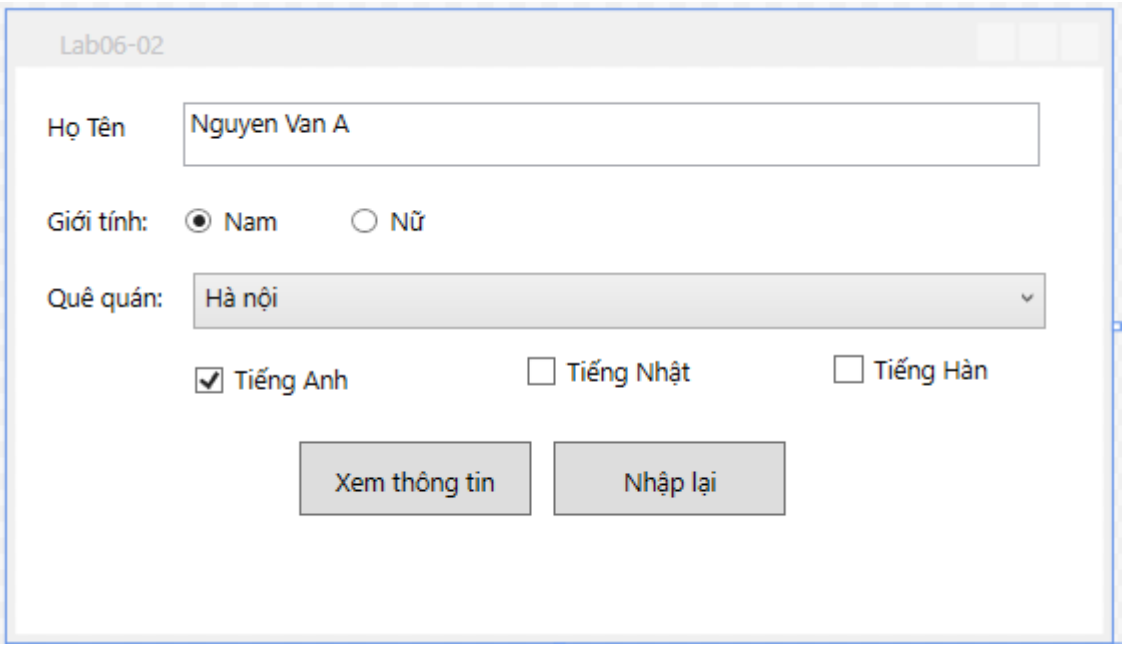
```
            <Button x:Name="Button7" Content="7" Grid.Row="2" Style="{DynamicResource
NumberButtonStyle}"/>
            <Button x:Name="Button8" Content="8" Grid.Row="2" Grid.Column="1"
Style="{DynamicResource NumberButtonStyle}"/>
            <Button x:Name="Button9" Content="9" Grid.Row="2" Grid.Column="2"
Grid.ColumnSpan="5" Style="{DynamicResource NumberButtonStyle}" />
            <Button x:Name="Button0" Content="0" Grid.Row="3" Grid.Column="1"
Style="{DynamicResource NumberButtonStyle}"/>
        </Grid>
        <Grid x:Name="GridOperators" Grid.Row="2" Grid.Column="1"
HorizontalAlignment="Center">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto"/>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
               <RowDefinition Height="Auto"/>
               <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
            </Grid.RowDefinitions>
            <Button x:Name="ButtonPlus" Style="{DynamicResource ButtonOperatorStyle}"/>
            <Button x:Name="ButtonMinus" Style="{DynamicResource ButtonOperatorStyle}"
Content="-" Grid.Row="1"/>
            <Button x:Name="ButtonTimes" Style="{DynamicResource ButtonOperatorStyle}"
Content="*" Grid.Row="2"/>
             <Button x:Name="ButtonDivide" Style="{DynamicResource ButtonOperatorStyle}"
Content="/" Grid.Row="3"/>
            <Button x:Name="ButtonEquals" Style="{DynamicResource ButtonOperatorStyle}"
Content="=" Grid.Row="4" Margin="5,8,5,1" Height="Auto" Click="ButtonEquals_Click"/>
        </Grid>
    </Grid>
```

- ***Xử lý các sự kiện***

```
//Xoa ki tu cuoi cung trong du lieu nhap, neu chi co 1 ki tu dua ve 0
        private void ButtonClear_Click(object sender, RoutedEventArgs e)
        {
            if (this.txtNumberDisplay.Text.Length == 1)
                this.txtNumberDisplay.Text = "0";
            else
               this.txtNumberDisplay.Text =
txtNumberDisplay.Text.Remove(this.txtNumberDisplay.Text.Length - 1, 1);
        }
// Khi click button 1,2,3,4,5,6,7,8,9,0
        private void ButtonNumber_Click(object sender, RoutedEventArgs e)
        {
            Button btn = sender as Button;
            SetNumberDisplay(btn.Content.ToString());
        }
        private void SetNumberDisplay(string textNumber)
        {
            if(this.txtNumberDisplay.Text == "0" || this.txtNumberDisplay.Text ==
"+")
            {
                this.txtNumberDisplay.Text = textNumber;
            }
            else
                this.txtNumberDisplay.Text += textNumber;
        }
// Xu ly su kien click operator, Equal
```

Bài tập 2: Thêm project **Lab06-02** vào Lab06 Solution.

Thiết kế giao diện sau trong WPF

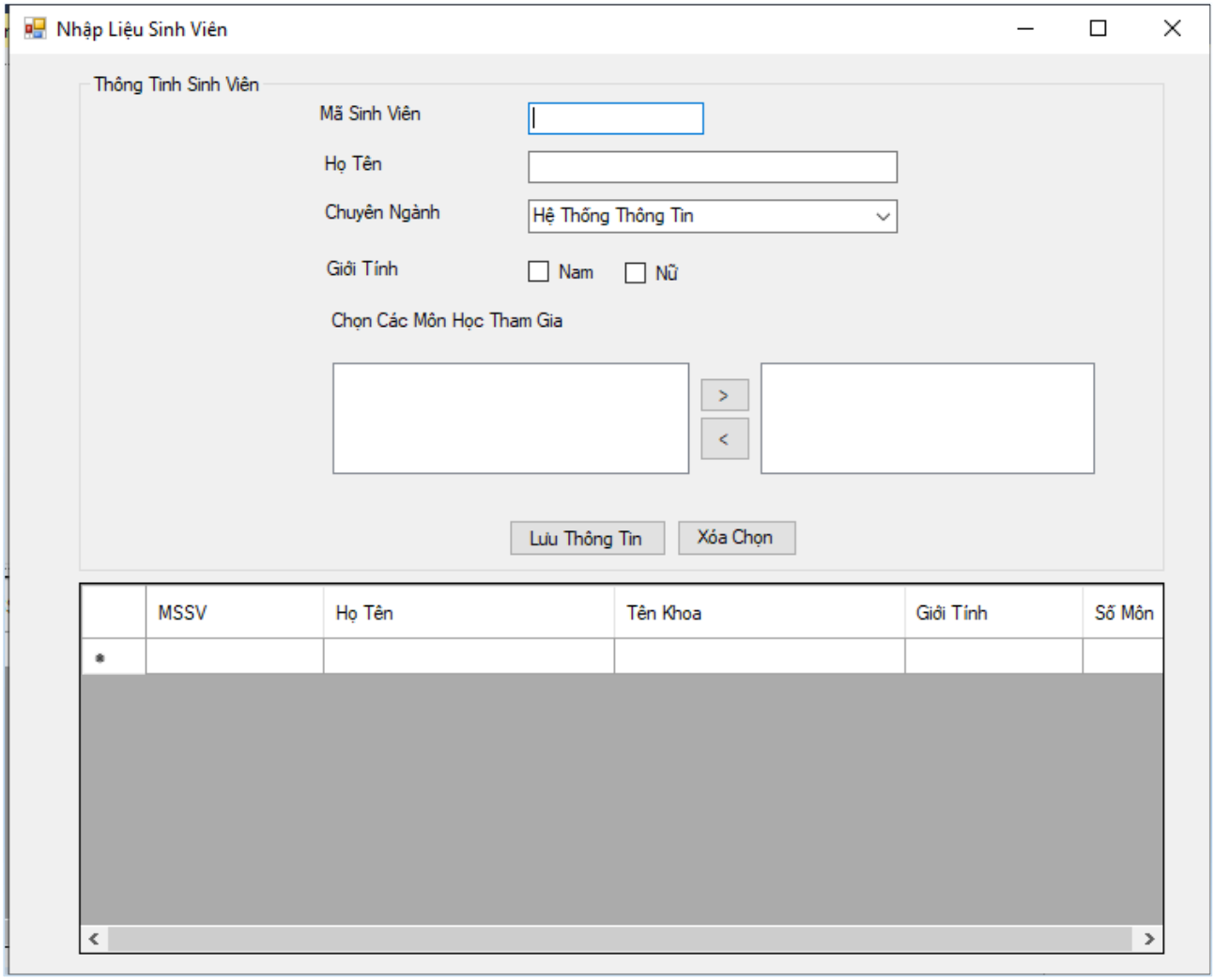


Xử lý sự kiện

- Nhập Lại: Xóa các giá trị đã nhập liệu trên form
- Xem Thông tin: Sử dụng message box để đưa thông tin vừa chọn trên form ra

Bài tập 3: Add thêm project ***Lab06-03*** vào Lab06 Solution: Xem nội dung lại ở bài **Lab02-02**

Thiết kế giao diện trong WPF tương tự như sau



Bài Tập 4: Thêm Project ***Lab06-04*** vào **Lab06** Solution: Xem lại bài tập **Lab03** mô phỏng điều khiển Windows media Player để phát các tập tin Audio và Video

----------Hết Lab 06--------