

# Công nghệ .NET

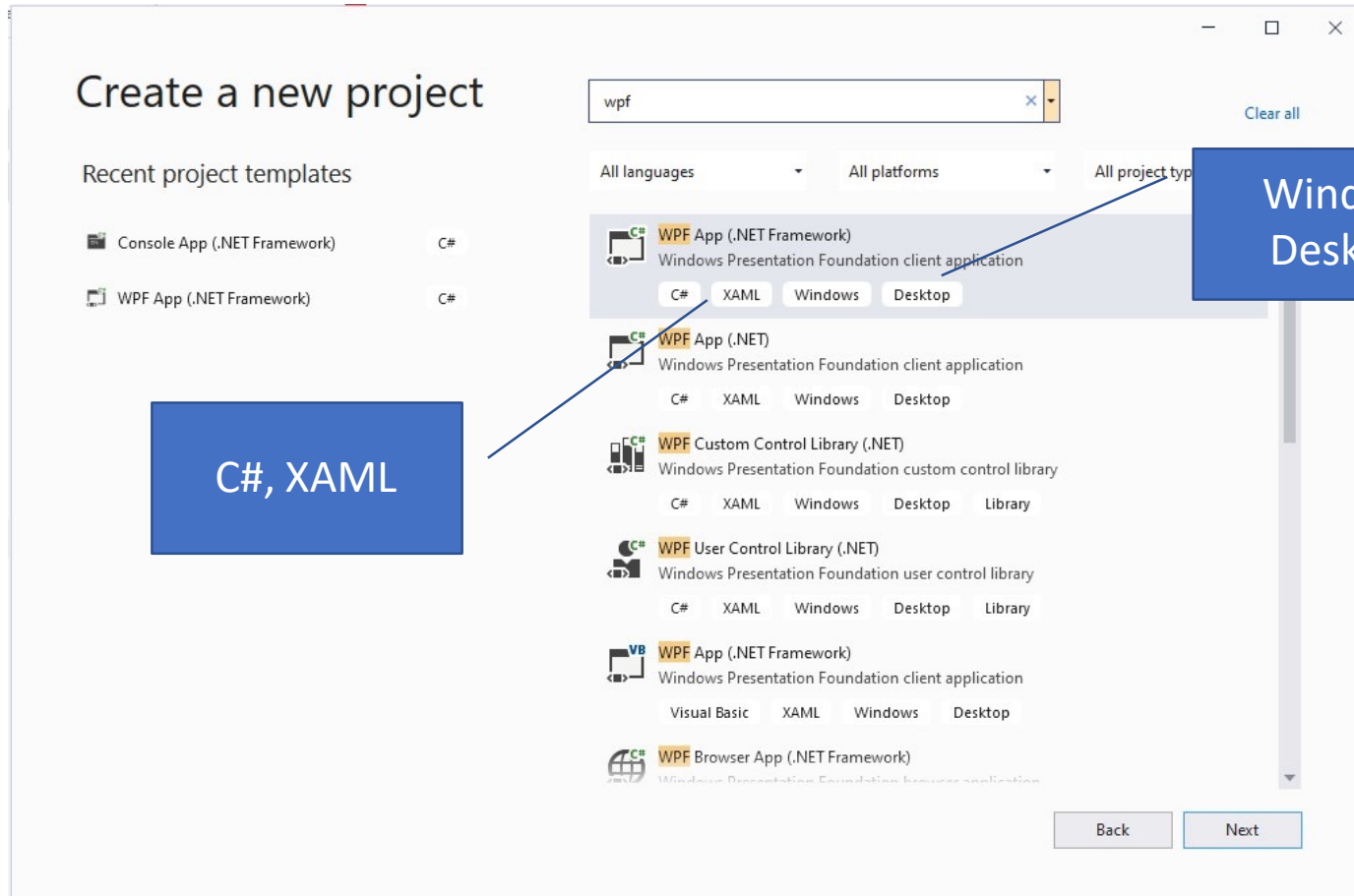
## Bài 7 – Welcome to WPF

Phạm Ngọc Hưng – Khoa CNTT

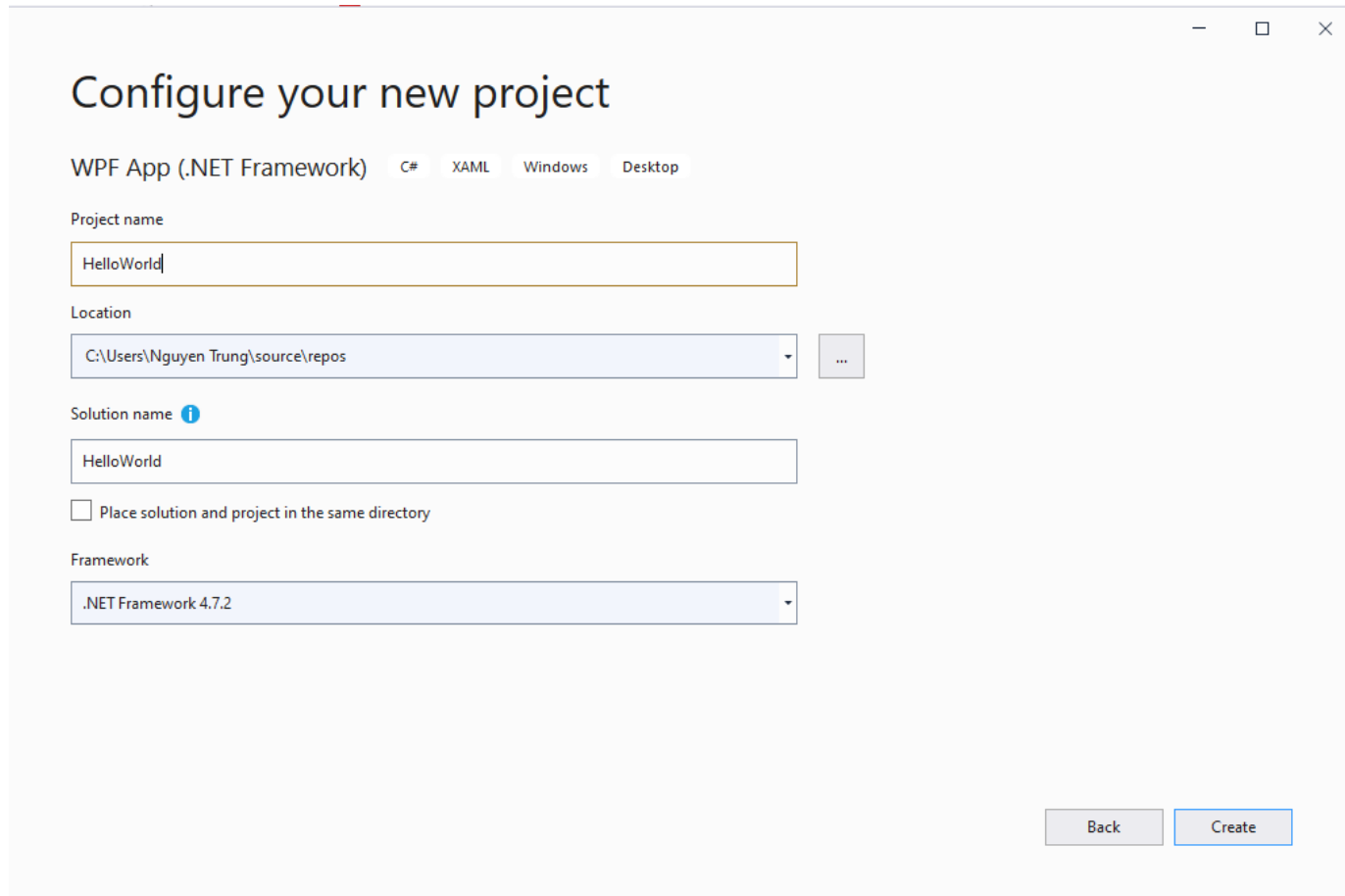
Email: [hung.phamngoc@phenikaa-uni.edu.vn](mailto:hung.phamngoc@phenikaa-uni.edu.vn)

Nguồn: Nguyễn Thành Trung – Khoa CNTT

# Hello World



# Hello World



Configure your new project

WPF App (.NET Framework) C# XAML Windows Desktop

Project name

HelloWorld

Location

C:\Users\Nguyen Trung\source\repos

Solution name ⓘ

HelloWorld

☐ Place solution and project in the same directory

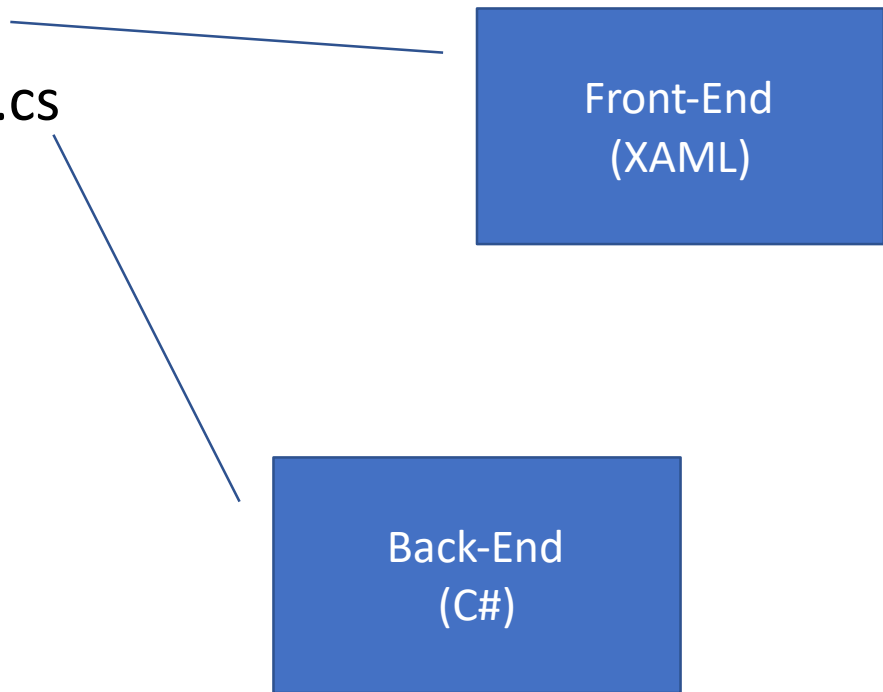
Framework

.NET Framework 4.7.2

Back Create

# Hello World

- Visual Studio tạo một dự án HelloWorld
- Solution Explorer hiển thị các tệp khác nhau
  - MainWindow.xaml
  - MainWindow.xaml.cs



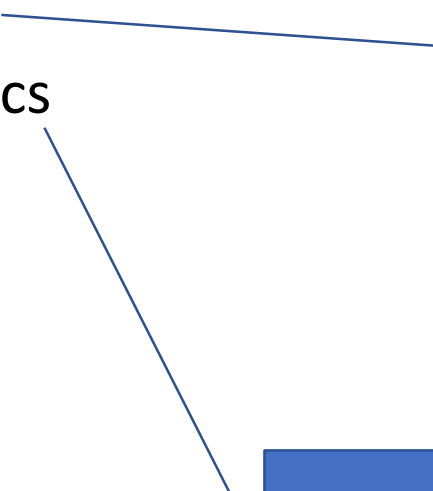
Front-End  
(XAML)

The diagram consists of two blue rectangular boxes. The top box is labeled 'Front-End (XAML)' and the bottom box is labeled 'Back-End (C#)'. A blue line originates from the text 'MainWindow.xaml' in the list above and points to the top box. Another blue line originates from the text 'MainWindow.xaml.cs' in the list above and points to the bottom box.

Back-End  
(C#)

# Hello World

- Visual Studio tạo một dự án HelloWorld
- Solution Explorer hiển thị các tệp khác nhau
  - MainWindow.xaml
  - MainWindow.xaml.cs



Front-End  
(XAML)

The diagram consists of two blue rectangular boxes. The top box is labeled 'Front-End (XAML)' and the bottom box is labeled 'Back-End (C#)'. A blue line originates from the text 'MainWindow.xaml' in the list above and points to the top box. Another blue line originates from the text 'MainWindow.xaml.cs' in the list above and points to the bottom box.

Back-End  
(C#)

# Hello World

- ToolBox
  - Add a TextBlock control
    - Customize the text in the text block
  - Add radio buttons
    - Add display text for each radio button
    - Set a radio button to be checked by default
  - Add the button control

# Hello World

- Add code to the display button

```
if (HelloButton.IsChecked == true)
{
    MessageBox.Show("Hello.");
}
else if (GoodbyeButton.IsChecked == true)
{
    MessageBox.Show("Goodbye.");
}
```

- Debug and test the application

# XAML

- eXtensible Application Markup Language: Ngôn ngữ đánh dấu ứng dụng.
- Giúp đơn giản hóa việc tạo giao diện người dùng cho ứng dụng
- Một thành phần trong XAML luôn bắt đầu bằng dấu < và kết thúc bằng dấu />; hoặc <ten\_object> và kết thúc bằng </ten\_object>
- Ví dụ

```
<StackPanel>  
  <Button Content="Click Me"/>  
</StackPanel>
```



# XAML

- Thuộc tính:
  - Cú pháp thuộc tính: đặt tên thuộc tính đối tượng đang được đặt, theo sau là toán tử gán (=).
  - Giá trị của một thuộc tính luôn được chỉ định là một chuỗi được chứa trong dấu ngoặc kép.
- Ví dụ

```
<Button Background="Blue" Foreground="Red" Content="This is a button"/>
```

# XAML

- Property Element Syntax

- Đối với vài thuộc tính của Object Element, khai báo cú pháp thuộc tính là không thể bởi vì giá trị của thuộc tính không nằm trong phạm vi "chuỗi giá trị"; thay vào đó ta phải khai báo Property Element Syntax

- Ví dụ

```
<Label Content="Title" Margin="6">  
    <Label.Background>  
        <SolidColorBrush Color="Olive"/>  
    </Label.Background>  
</Label>
```

# XAML

- Text Content

- Một số ít các element có khả năng chứa text như là content của nó, để có được khả năng này cho các element ta phải khai báo đúng, Class của Object Element tương ứng phải có khai báo content property có kiểu string hoặc object
- Ví dụ  
`<Button>Hello</Button>`

# XAML

- Events (Attribute Syntax)
  - Events được dùng cho các thành phần như là một sự kiện. Tên của Events là tên của sự kiện.
  - Trong WPF, hiện thực một sự kiện của XAML: tên của sự kiện là tên của phương thức xử lý.

```
<StackPanel Orientation="Horizontal" Grid.Row="5" Grid.ColumnSpan="2" >
  <Button Content="New" Height="23" HorizontalAlignment="Left"
    Margin="6" Name="btnNew" VerticalAlignment="Top"
    Width="50" Click="btnNew_Click" />
  <Button Content="Save" Height="23"
    HorizontalAlignment="Left" Margin="6"
    Name="btnSave" VerticalAlignment="Top"
    Width="50" Click="btnSave_Click" />
  <Button Content="Cancel" Height="23"
    HorizontalAlignment="Left" Margin="6"
    Name="btnCancel" VerticalAlignment="Top"
    Width="50" Click="btnCancel_Click" />
  <Button Content="Delete" Height="23"
```

# XAML

- Root Element và Namespace

- Mỗi tập tin XAML chỉ có 1 root element như : Window , Application , Page , Resource Dictionary
- Root Element chứa các thuộc tính **xmlns** & **xmlns:x** , các thuộc tính này chỉ định bộ xử lý XAML, XAML namespace chứa các định nghĩa kiểu cho việc sao lưu các kiểu đánh dấu như Element.
- Thuộc tính **xmlns** chỉ định namespace XAML mặc định, trong các namespace XAML mặc định các object element có thể được xác định mà không cần đến 1 tiền tố.

# XAML

- Root Element và Namespace
  - Namespace mặc định bao gồm :
    - xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"
    - xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"
  - Note: các thuộc tính xmlns chỉ cần khai báo trên các root element của mỗi tập tin XAML.

# XAML

- Root Element và Namespace

- Tiền tố x:

- được sử dụng để ánh xạ (mapping) với namespace <http://schemas.microsoft.com/winfx/2006/xaml>, đó là namespace hỗ trợ cấu trúc ngôn ngữ XAML.
    - namespace này chứa các cấu trúc lập trình mà sẽ sử dụng rất thường xuyên trong XAML.

# XAML

- Root Element và Namespace

- Tiền tố x:

- x:Key: thiết lập 1 khóa duy nhất cho mỗi resource trong 1 ResourceDictionary. x:Key chiếm khoảng 90% tiền tố x: được sử dụng.
    - x:Class: xác định CLR namespace và class name cho các class cung cấp code-behind cho 1 XAML page.
    - x:Static: tạo ra các giá trị tĩnh. Các giá trị đến từ các thực thể mã loại giá trị không trực tiếp là loại giá trị của thuộc tính mục tiêu, nhưng có thể được đánh giá cho loại đó.
    - x>Type: cung cấp đối tượng Type cho kiểu được đặt tên. Tiền tố này được sử dụng thường xuyên nhất trong các kiểu (styles) và mẫu (templates).



# Các thành phần của WPF

- Giống như các thành phần khác của .NET Framework, WPF tổ chức các chức năng theo một nhóm namespace cùng trực thuộc namespace System.Windows.
- Một ứng dụng WPF điển hình gồm một tập các trang XAML và phần code tương ứng được viết bằng C#/VB (code-behind).
- Tất cả các ứng dụng đều kế thừa từ lớp chuẩn Application của WPF. Lớp này cung cấp những dịch vụ chung cho mọi ứng dụng (biến, phương thức chuẩn).

# Các thành phần của WPF

- Panel:
  - Sắp đặt các thành phần khác nhau trên giao diện.
  - Mỗi panel có thể chứa các thành phần con (control panel,...).
  - Panel khác nhau cho phép sắp xếp thành phần con theo những cách khác nhau.

# Các thành phần của WPF

- Các Panel thông dụng:
  - Grid
  - StackPanel
  - WrapPanel
  - Canvas
  - DockPanel

# Các thành phần của WPF

- Panel
  - StackPanel
    - Bố trí các thành phần con nằm trong nó bằng cách sắp xếp chúng theo thứ tự trước sau
    - Các phần tử sẽ xuất hiện theo thứ tự mà chúng được khai báo trong file XAML theo chiều dọc (ngầm định) hoặc theo chiều ngang (Orientation = "Horizontal")

# Các thành phần của WPF

- Panel
  - StackPanel
    - Ví dụ

```
<!--Sử dụng StackPanel sắp xếp ngàm định theo chiều dọc-->
<StackPanel Background="Beige">
  <!--Thiết lập thuộc tính khung viền bao quanh control 1-->
  <Border Width="200" BorderBrush="DarkSlateBlue" Background="#a9e969" BorderThickness="2">
    <!--Khai báo control 1 dạng checkbox-->
    <CheckBox>Control 1</CheckBox>
  </Border>
  <!--Khai báo control 2 dạng nút bấm-->
  <Button Width="200">Control 2</Button>
  <!--Thiết lập thuộc tính khung viền bao quanh control 3-->
  <Border Width="200" BorderBrush="#feca00" BorderThickness="2">
    <!--Khai báo control 3 dạng text-->
    <TextBlock HorizontalAlignment="Right">Control 3</TextBlock>
  </Border>
</StackPanel>
```

# Các thành phần của WPF

- StackPanel

- Ví dụ

<!--Sử dụng StackPanel sắp xếp theo chiều ngang xác định bằng thuộc tính Orientation="Horizontal"-->

**<StackPanel Background="Beige" Orientation="Horizontal">**

    <!--Thiết lập thuộc tính khung viền bao quanh control 1-->

    <Border Width="90" Height="80" BorderBrush="DarkSlateBlue"  
        Background="#a9e969" BorderThickness="2">

        <!--Khai báo control 1 dạng checkbox-->

        <CheckBox>Control 1</CheckBox>

    </Border>

    <!--Khai báo control 2 dạng nút bấm-->

    <Button Width="90" >Control 2</Button>

    <!--Thiết lập thuộc tính khung viền bao quanh control 3-->

    <Border Width="90" BorderBrush="#feca00" BorderThickness="2">

        <!--Khai báo control 3 dạng text-->

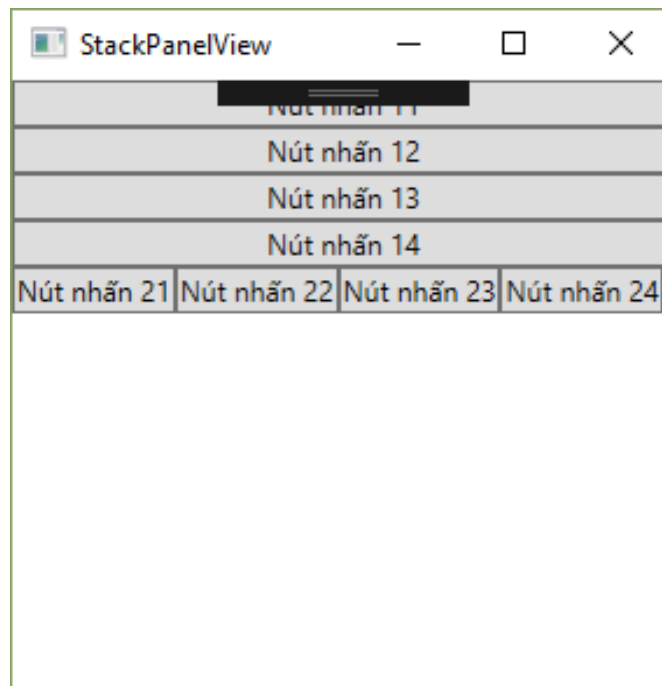
        <TextBlock HorizontalAlignment="Right">Control 3</TextBlock>

    </Border>

**</StackPanel>**

# Các thành phần của WPF

- StackPanel
  - Ví dụ: tạo giao diện như sau



# Các thành phần của WPF

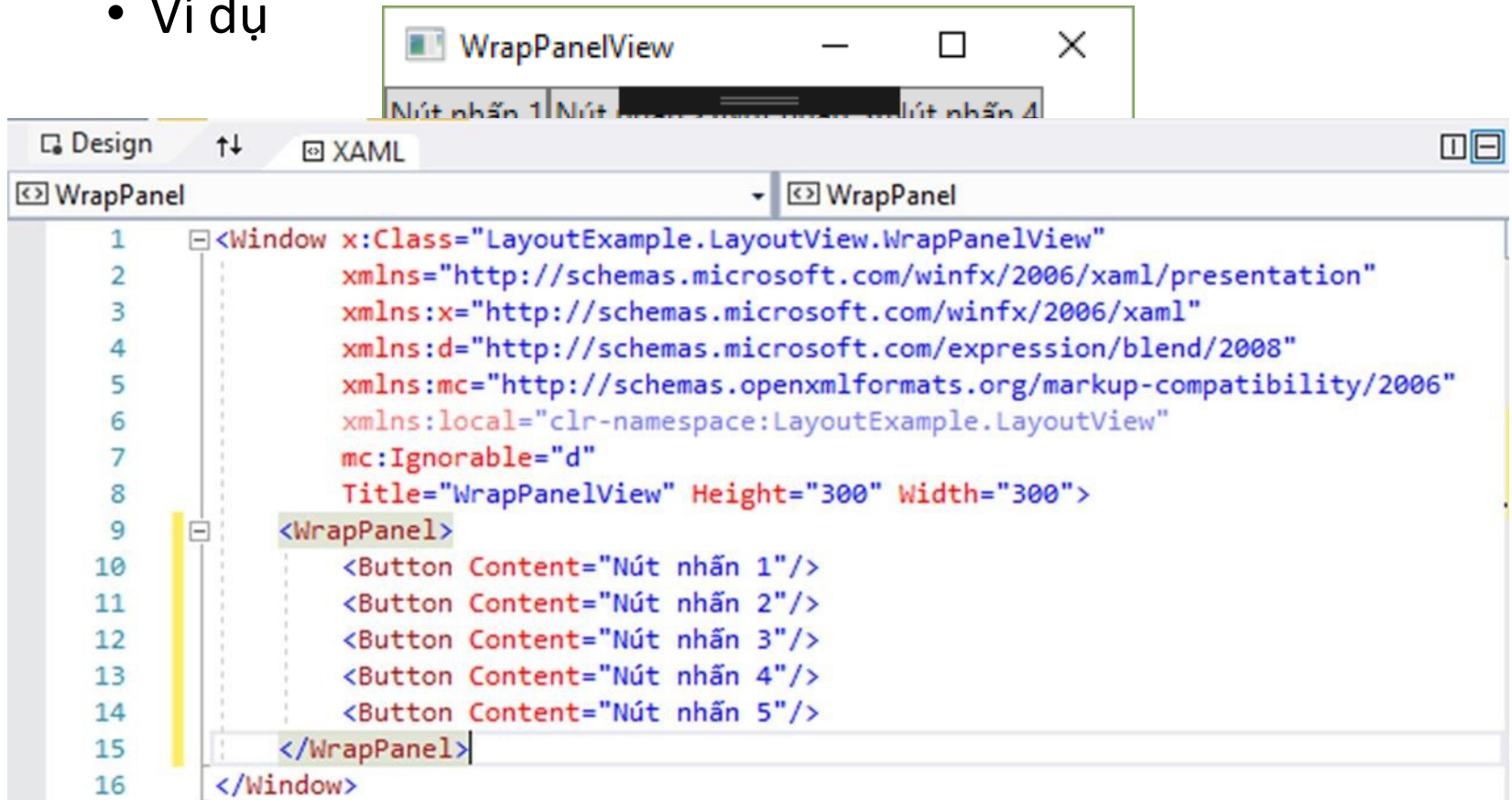
- WarpPanel

- Cho phép sắp xếp các phần tử từ trái sang phải. Khi một dòng phần tử đã điền đầy khoảng không gian cho phép theo chiều ngang, WrapPanel sẽ cuộn phần tử bị tràn khỏi giới hạn kích quy định của nó xuống đầu dòng tiếp theo (tương tự như việc cuộn text).



# Các thành phần của WPF

- WrapPanel
  - Ví dụ



# Các thành phần của WPF

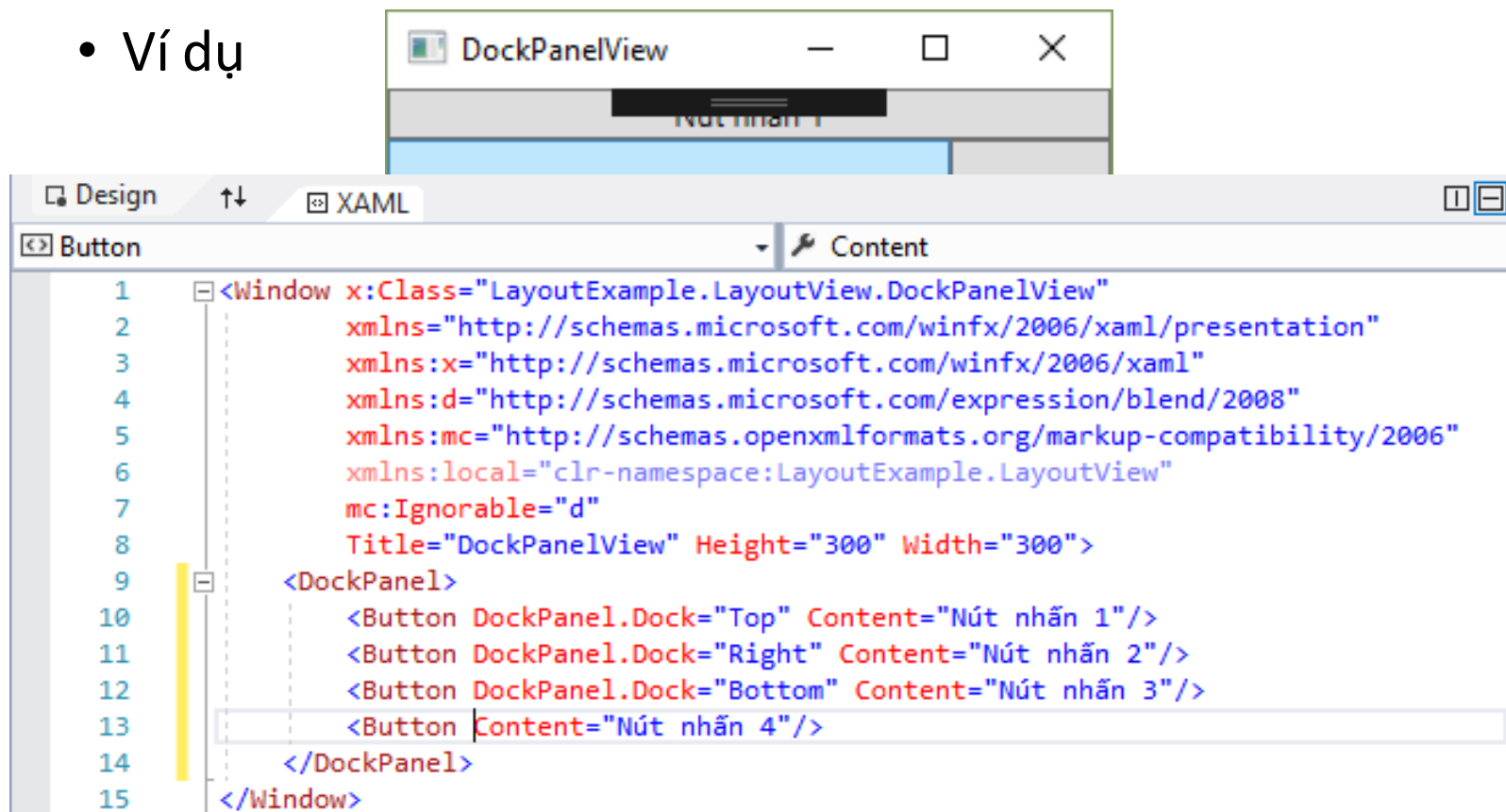
- DockPanel

- Cho phép các phần tử bám lên các cạnh của panel DockPanel bao chứa chúng. Nếu như có nhiều phần tử cùng bám về một cạnh, chúng sẽ tuân theo thứ tự mà chúng được khai báo trong file XAML

# Các thành phần của WPF

- DockPanel

- Ví dụ



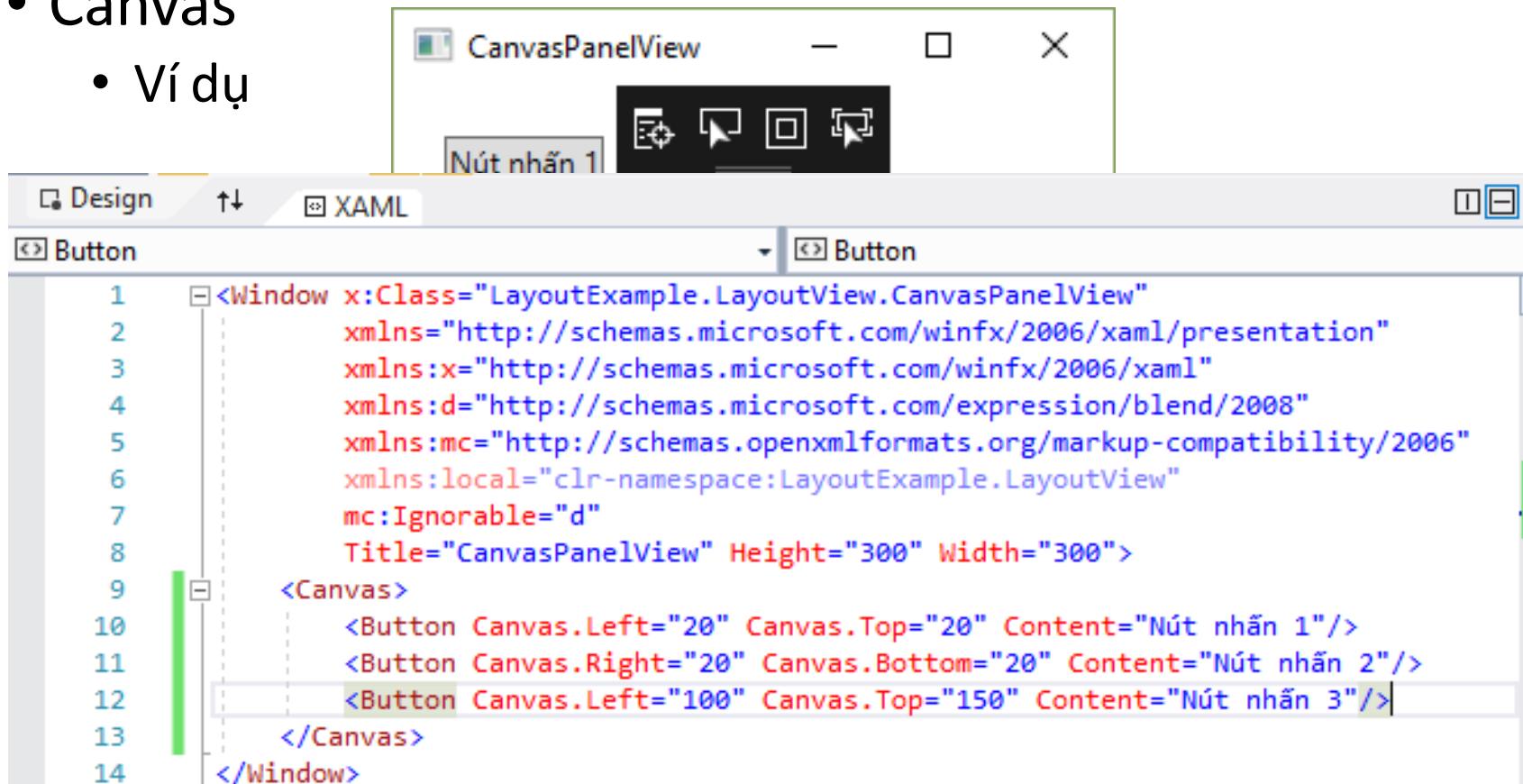
# Các thành phần của WPF

- Canvas

- Sử dụng phương thức sắp xếp các phần tử UI theo vị trí tuyệt đối bằng cách đặt thuộc tính Top (đỉnh) và Left (bên trái) của chúng.
- Thêm vào đó, thay vì đặt thuộc tính Top, Left, ta có thể đặt thuộc tính Bottom (đáy), Right (bên phải). Nếu đặt đồng thời thuộc tính Left và Right, thuộc tính Right sẽ bị bỏ qua. Tương tự thuộc tính Top sẽ được ưu tiên hơn thuộc tính Bottom.
- Phần tử UI sẽ không thay đổi kích thước để thỏa mãn 2 thuộc tính trên cùng một lúc. Các phần tử được khai báo sớm hơn trong file XAML sẽ có thể bị che khuất phía dưới các phần tử được khai báo muộn hơn nếu vị trí của chúng xếp chồng lên nhau.

# Các thành phần của WPF

- Canvas
  - Ví dụ



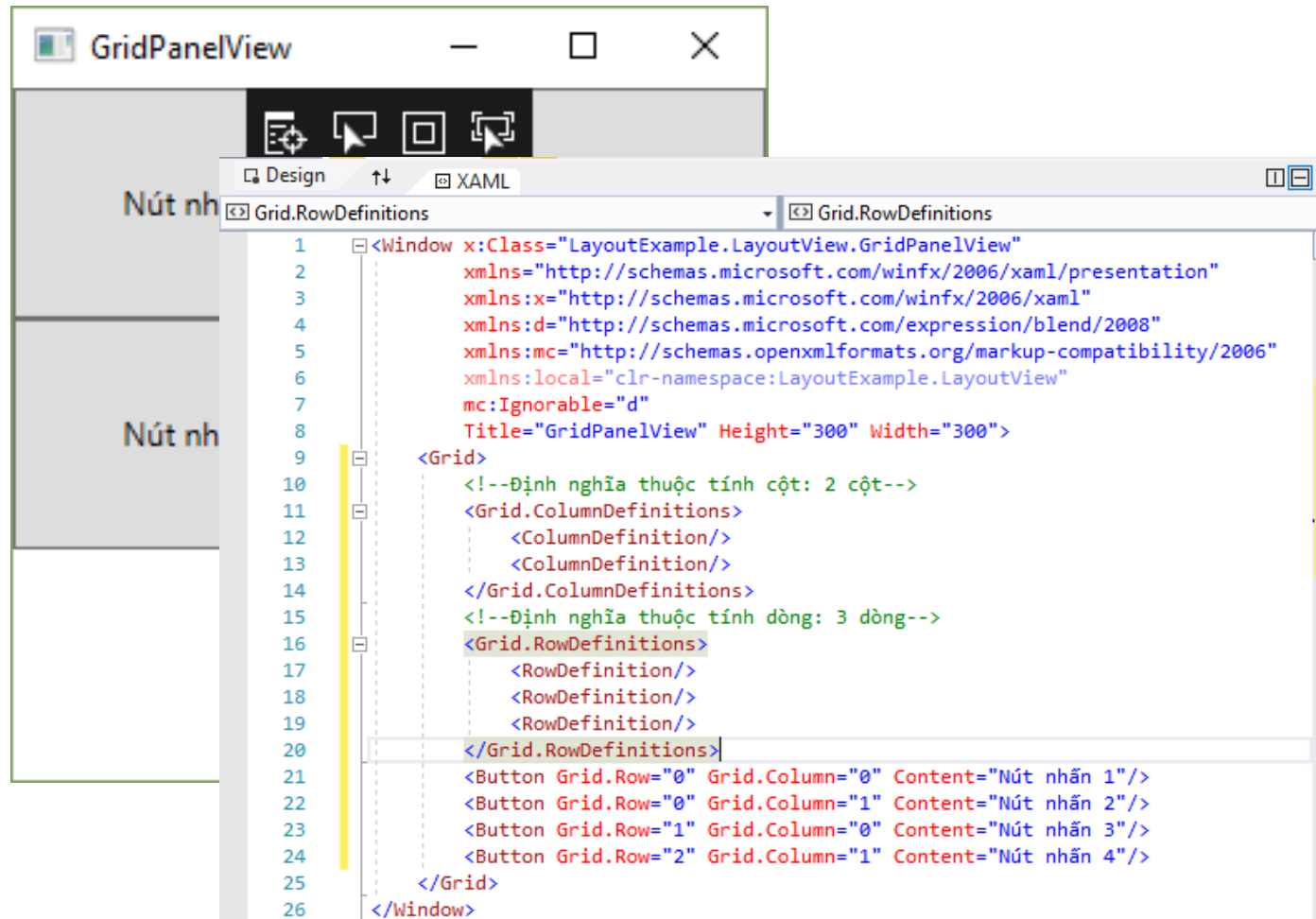
# Các thành phần của WPF

- Grid

- Cho phép sắp đặt các thành phần con lên một lưới tọa độ
- Grid sẽ tự động chia đều các dòng và cột (dựa trên kích thước của phần nội dung)
- Sử dụng dấu sao (\*) để phân định kích thước theo tỉ lệ hoặc phân định giá trị tuyệt đối về chiều cao hoặc chiều rộng cho hàng và cột.

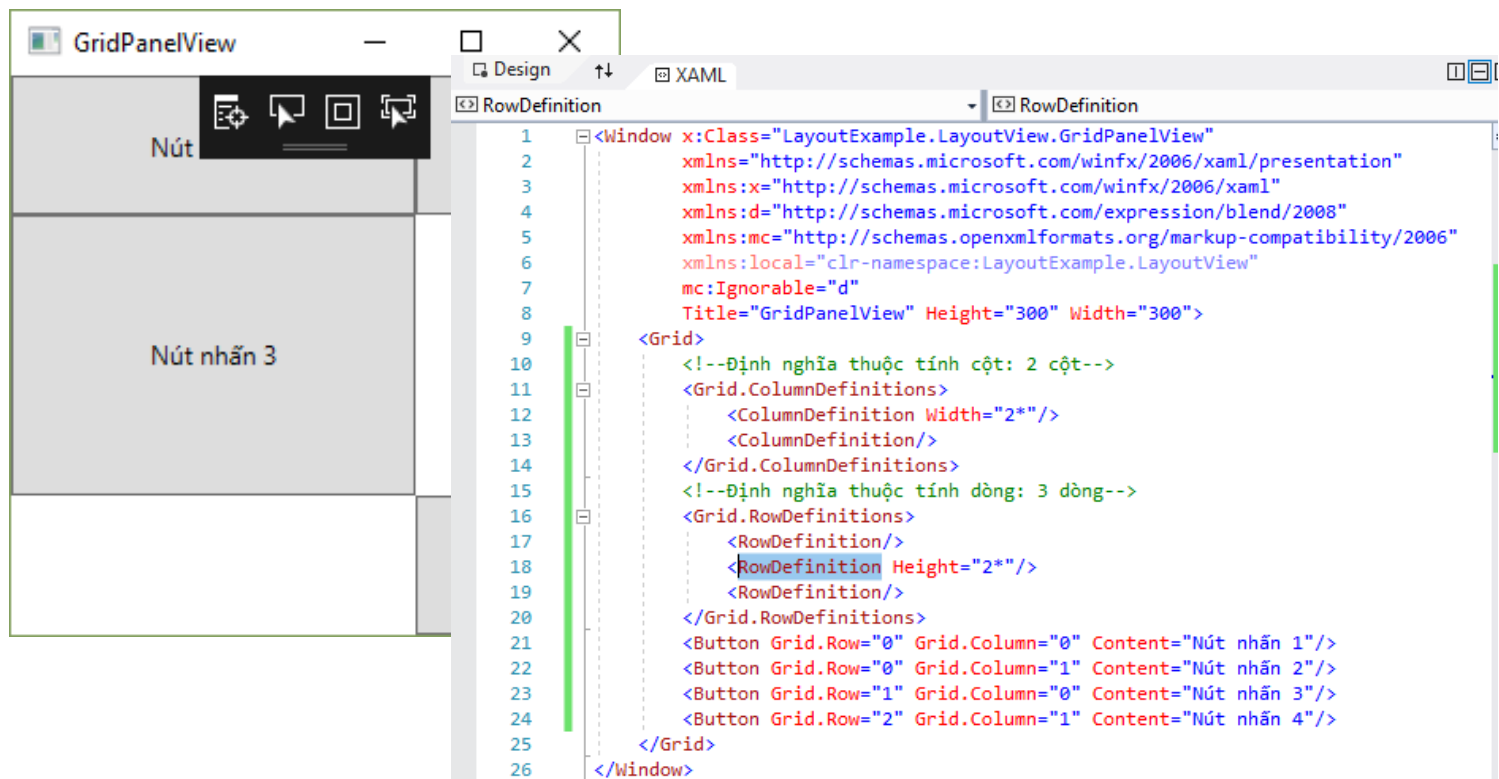
# Các thành phần của WPF

- Grid
  - Ví dụ



# Các thành phần của WPF

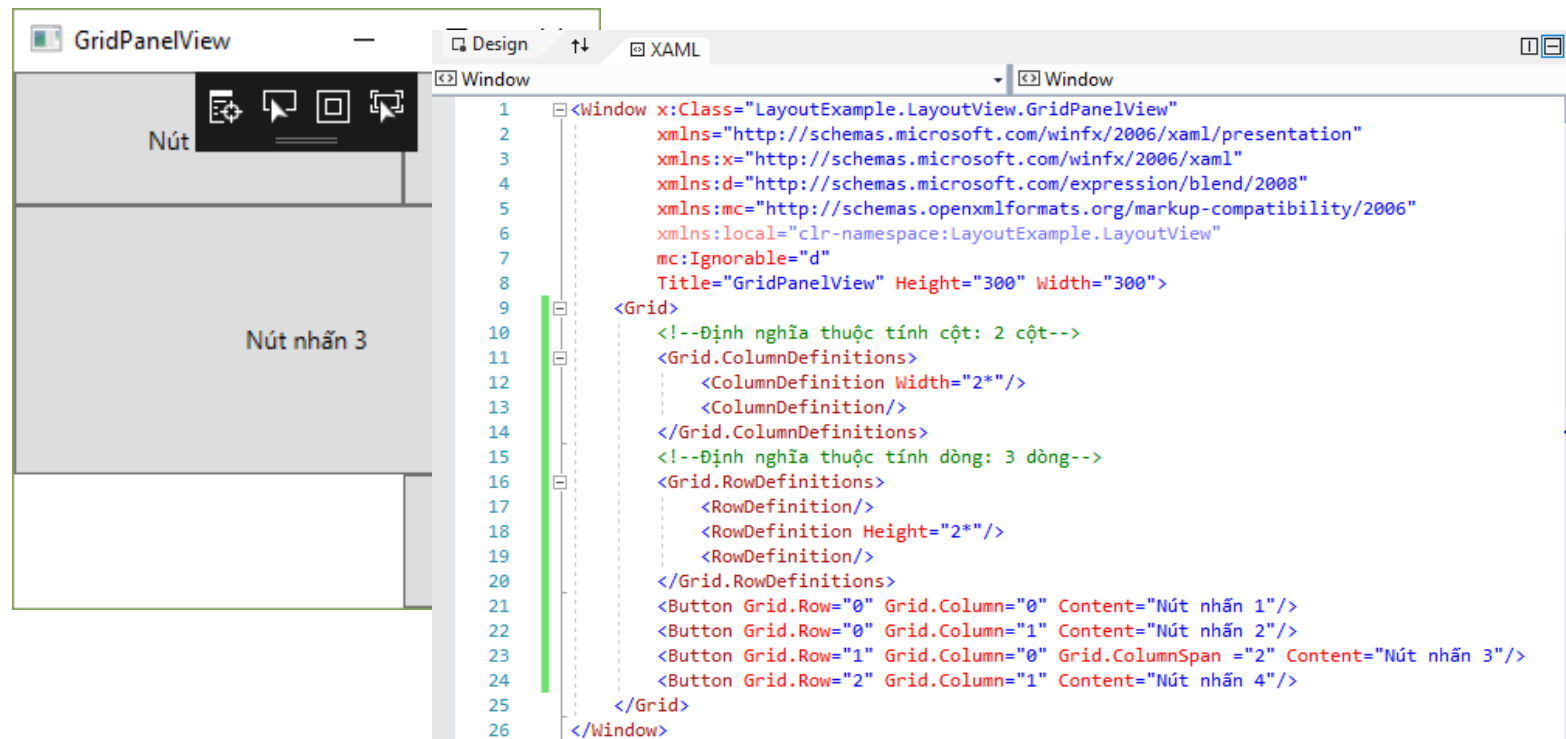
- Grid
  - Khi thay đổi kích thước dòng/cột





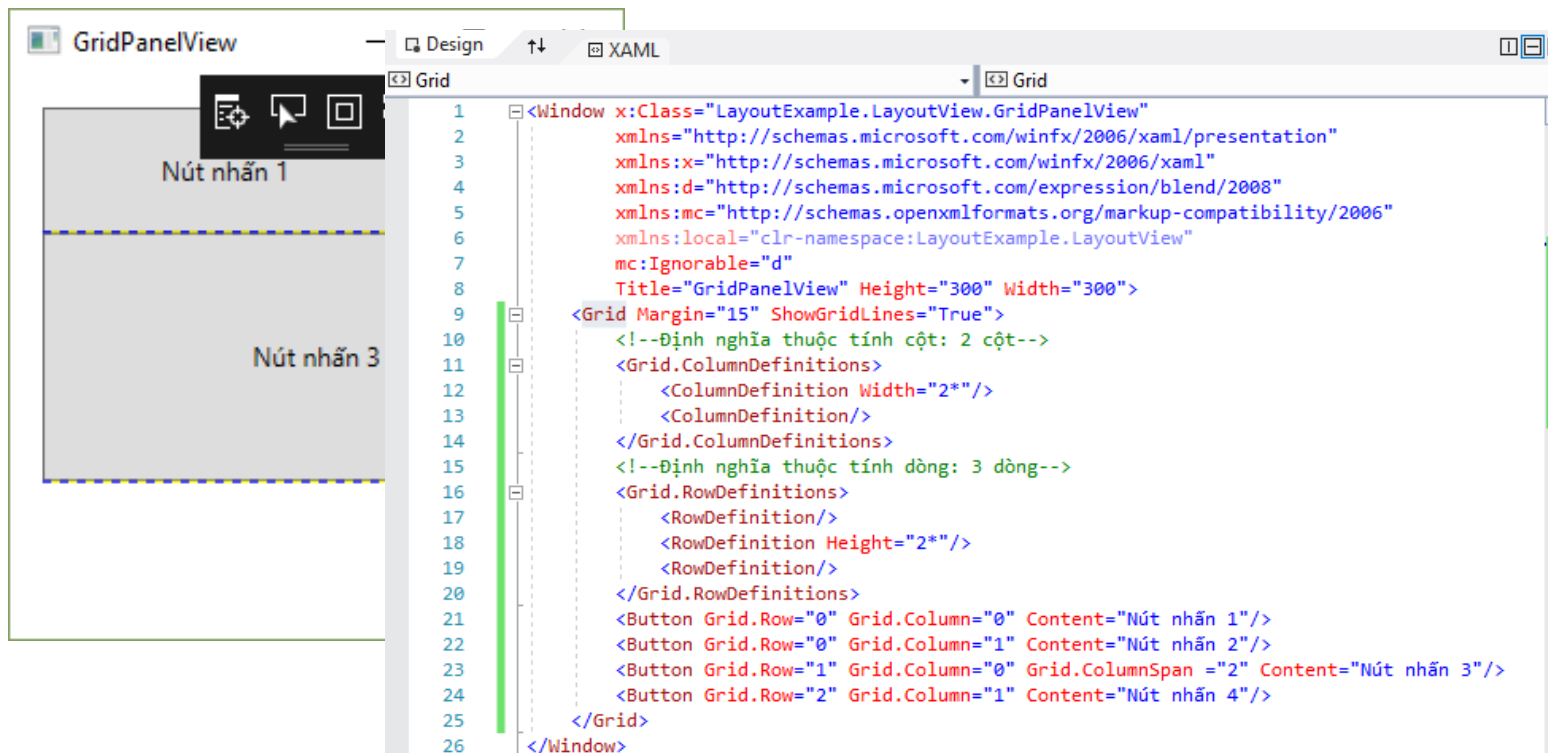
# Các thành phần của WPF

- Grid
  - Khi trộn dòng/cột: Grid.ColumnSpan / Grid.RowSpan



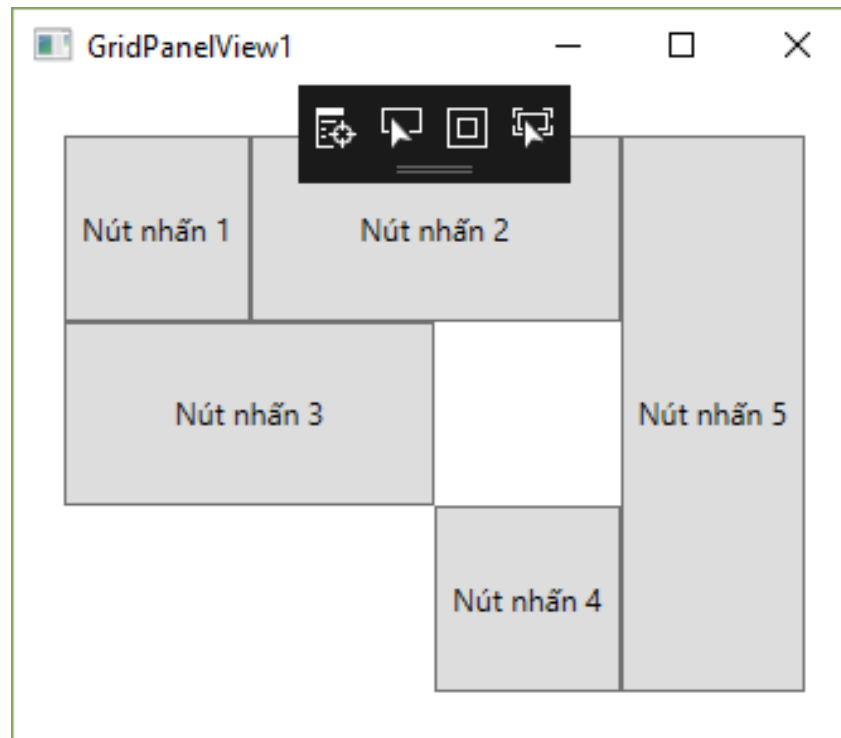
# Các thành phần của WPF

- Grid
  - Canh lề



# Các thành phần của WPF

- Ví dụ: Tạo giao diện như hình dưới với khoảng cách biên cửa sổ là 30px



# Các thành phần của WPF

- Control:

- WPF cung cấp một số lượng lớn các control (Button, Label, TextBox, ListBox, Menu, Slider, SpellCheck, PasswordBox... Ngoài ra, người dùng có thể tùy ý định nghĩa các control theo ý mình.
- Các sự kiện do người dùng tạo ra, như di chuyển chuột hay ấn phím, có thể được các control nắm bắt và xử lý.
- Trong khi các control và các thành phần giao diện khác có thể được đặc tả đầy đủ bằng XAML, các sự kiện bắt buộc phải được xử lý bằng code-behind (C#/VB).

# Các thành phần của WPF

- Style:

- Tương tự CSS trong HTML
- Một Style có thể được dẫn xuất từ một Style khác, thừa kế hoặc chồng lên những thuộc tính đã thiết lập.
- Ví dụ, kiểu ButtonStyle có thể được định nghĩa như sau:

```
<Style x:Key="ButtonStyle">  
  <Setter Property="Control.Background" Value="Red"/>  
  <Setter Property="Control.FontSize" Value="16"/>  
</Style>
```

Bất kỳ nút bấm nào sử dụng kiểu này sẽ có nền màu đỏ và sử dụng font chữ kích thước 16. Ví dụ:

```
<Button Style="{StaticResource ButtonStyle}">  
Click Here  
</Button>
```

# Các thành phần của WPF

- Control:
  - Label: Nhãn
  - TextBox: Hộp soạn thảo
  - Button: Nút bấm
  - CheckBox: Hộp chọn
  - RadioButton: Hộp chọn radio (chỉ được phép chọn 1 mục trong mỗi nhóm)
  - ListBox: Hộp danh sách
  - ComboBox: Hộp danh sách thả xuống