Bài 1

BỐ TRÍ GIAO DIỆN TRONG ỨNG DỤNG WPF

Bài này giới thiệu cách thức bố trí giao diện trong ứng dụng WPF. Phần đầu sẽ giới thiệu về các dạng *panel*, một sự đổi mới trong phương thức bố trí giao diện của ứng dụng WPF so với MFC, VB Forms hay ngay cả Windows Forms nhằm tăng tính linh hoạt. Sau đó, các dạng panel thông dụng cùng với đặc tính của chúng sẽ được trình bày thông qua các ví dụ đơn giản.

1 Giới thiệu chung

Như đã giới thiệu trong bài mở đầu, WPF sử dụng các dạng *panel* khác nhau để bố trí các phần tử trên giao diện người dùng. Điều này xuất phát từ ý tưởng kết hợp công nghệ giao diện mạnh như Windows Forms, với các kỹ thuật sắp đặt (layout) của trình duyệt nhằm nâng cao tính linh hoạt trong việc bố trí các phần tử trên giao diện.

Các công nghệ xây dựng giao diện như VB6 form, Access forms… dựa trên nguyên tắc bố trí theo *vị trí tuyệt đối*. Nghĩa là, người lập trình phải xác định giá trị tọa độ góc trên bên trái của một control (so với với góc trên bên trái của một form) khi muốn đặt nó lên form. Điều này cho phép lập trình viên điều khiển vị trí của control khá dễ dàng, nhưng lại thường đòi hỏi một lượng lớn mã trình khi cần thay đổi kích thước form. Đây là phương pháp tiếp cận theo hướng *áp đặt* (imperative), trong đó máy tính được chỉ rõ phải làm những bước gì, khi nào và theo trình tự nào. Với cách thức bố trí này, các điều khiển như Label hay Panel không tự động kéo giãn để phù hợp với kích thước phần nội dung chứa trong nó. Và như vậy, nếu phần nội dung của một Label lớn hơn vùng có thể hiển thị của Label đó, thì nội dung này sẽ bị cắt đi hoặc bị che lấp.

Trong khi đó, các phần tử giao diện Web trên trình duyệt được sắp xếp theo phương thức *khai báo* (declarative), trong đó, người lập trình chỉ đưa ra những thứ cần làm, còn máy tính sẽ giải quyết vấn đề làm như thế nào. Với phương thức này, giao diện trên trình duyệt không đòi hỏi mã trình để thay đổi kích thước các *vùng chứa* (containner). HTML cho phép ta định ra một chuỗi các vùng chứa, ví dụ như các phần tử `<div>`, `<table>`, `<tr>` và `<td>`, để bố trí các phần tử UI khác trong đó một cách linh động bên phải hoặc bên trái một đối tượng; hay cũng có thể sắp xếp chúng theo vị trí tuyệt đối trên trang Web. Các phần tử như `<div>` quan tâm tới kích thước bên trong nội dung của nó và sẽ tự động giãn ra để chứa đủ nội dung bên trong.

Tuy nhiên, cả hai cách tiếp cận nêu trên đều khó có thể đạt được cách bố trí như ý, mặc dù cách bố trí trên trình duyệt có giảm lượng code xử lý. Hiện nay, Windows Forms đưa thêm những khái niệm như *Docking* (cập bến) hay *Anchoring* (buông neo), bổ sung một cách tiếp cận kiểu khai báo linh hoạt hơn để phát triển các ứng dụng trên máy trạm. WPF tiếp bước xu hướng này với việc bố trí giao diện dựa trên khái niệm về *panel*.

Phần lớn các phần tử UI trong ứng dụng WPF chỉ có thể chứa duy nhất một phần tử con. Chẳng hạn, đoạn mã XAML sau sẽ mắc lỗi biên dịch sau: "*The 'Button' object already has a child and cannot add 'CheckBox'. 'Button' can accept only one child.*" Nghĩa là, đối tượng nút bấm 'Button' đã chứa một phần tử con (cụ thể là đối tượng 'TextBlock') và do đó, không thể thêm vào một đối tượng 'CheckBox' hay 'ComboBox' nữa.

Đoạn mã XAML sau đây không biên dịch được

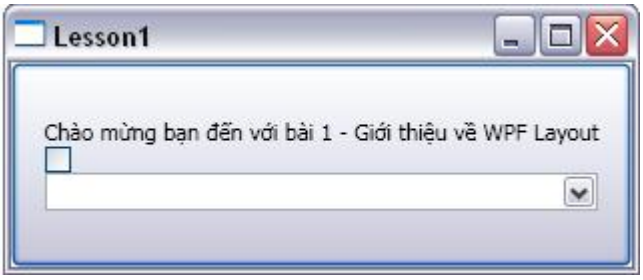
```
<Button>
  <TextBlock> Chào mừng bạn đến với bài 1 - Giới thiệu về WPF Layout
  </TextBlock>
  <CheckBox />
  <ComboBox />
</Button>
```

Để nút bấm này có thể chứa 3 phần tử con bên trong nó, WPF sử dụng panel. Có nhiều dạng panel khác nhau trong WPF và mỗi dạng cho phép một kiểu bố trí giao diện khác nhau. Các panel có thể lồng vào nhau cho phép bố trí các phần tử trên giao diện ở những dạng sắp xếp bất kỳ. Ví dụ, để sửa vấn đề nêu ra ở ví dụ trên, ta có thể lồng một StackPanel bên trong nút bấm, và bố trí các phần tử con bên trong panel đó.

Đoạn mã XAML sau đây cho phép 1 nút bấm chứa được text, checkbox và combobox

```
<Button>
 <StackPanel>
    <TextBlock>Chào mừng bạn đến với bài 1 - Giới thiệu về WPF Layout
    </TextBlock>
    <CheckBox />
    <ComboBox />
 </StackPanel>
</Button>
```

Kết quả biên dịch sẽ là:



Hình 1.1 – Kết quả sửa đổi đoạn mã XAML hiển thị hơn một phần tử giao diện con trong một nút bấm sử dụng StackPanel

2 Các dạng Panel thông dụng

Để bạn đọc thấy được vai trò quan trọng của panel trong việc bố trí giao diện, phần sau đây sẽ lần lượt giới thiệu những dạng panel thường dùng và các đặc tính của chúng thông qua các ví dụ đơn giản.

2.1 StackPanel

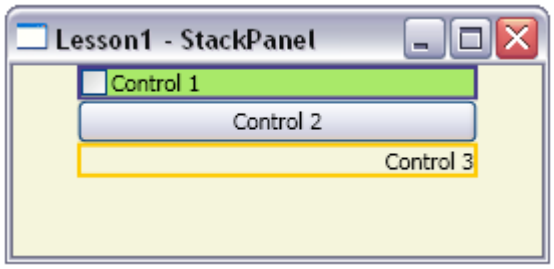
StackPanel bố trí các phần tử con nằm trong nó bằng cách sắp xếp chúng theo thứ tự trước sau. Các phần tử sẽ xuất hiện theo thứ tự mà chúng được khai báo trong file XAML theo chiều dọc (ngầm định) hoặc theo chiều ngang.

2.1.1 Sắp xếp theo chiều dọc

Sau đây là đoạn mã XAML minh họa việc sắp xếp các phần tử UI trong một đối tượng Window bằng StackPanel theo chiều dọc:

```
<!--Khai báo khởi tạo cửa sổ-->
<Window x:Class="Lesson1.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Lesson1 - StackPanel" Height="120" Width="280">
  <!--Sử dụng StackPanel sắp xếp ngầm 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>
</Window>
```

Kết quả là:



Hình 1.2 – Sắp xếp nhiều control theo thứ tự kế tiếp trên xuống dưới sử dụng StackPanel

Trong trường hợp sắp xếp theo chiều dọc, nếu tổng chiều cao của các phần tử con lớn hơn chiều cao của form chứa, thì các phần tử nằm ngoài form sẽ không được nhìn thấy.

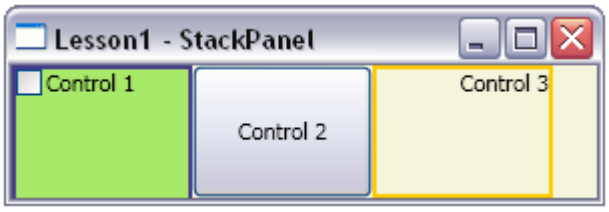
2.1.2 Sắp xếp theo chiều ngang

Sau đây là đoạn mã XAML minh họa việc sử dụng StackPanel để sắp xếp các phần tử UI cùng ví dụ ở trên theo chiều ngang. Điểm khác biệt duy nhất ở đây là thiết lập thêm thuộc tính `Orientation="Horizontal"` của đối tượng StackPanel được sử dụng.

```
<!--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>
```

Và kết quả sẽ là:



Hình 1.3 – Sắp xếp nhiều control theo thứ tự kế tiếp trái sang phải sử dụng StackPanel

Trong trường hợp sắp xếp theo chiều ngang, nếu tổng chiều rộng của các phần tử con lớn hơn chiều rộng của form chứa, thì các phần tử nằm ngoài form sẽ không được nhìn thấy.

2.2 WrapPanel

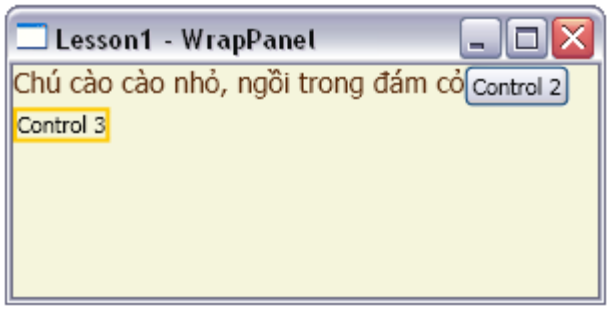
WrapPanel cho phép sắp xếp các phần tử từ trái sang phải. Khi một dòng phần tử đã điền đầy khoảng không gian cho phép theo chiều ngang, WrapPanel sẽ cuốn phần tử tiếp theo xuống đầu dòng tiếp theo (tương tự như việc cuốn text).

Dưới đây là một ví dụ đơn giản về việc sử dụng WrapPanel:

```
<Window x:Class="Lesson1.Window3"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Lesson1 - WrapPanel" Height="150" Width="300">
  <!--Sử dụng WrapPanel-->
  <WrapPanel Background="Beige">
    <TextBlock FontSize="14"
               Foreground="#58290A">
      Chú cào cào nhỏ, ngồi trong đám cỏ
    </TextBlock>
    <Button>Control 2</Button>
    <Border BorderBrush="#feca00" BorderThickness="2">
      <TextBlock>Control 3</TextBlock>
    </Border>
  </WrapPanel>

</Window>
```

Do chiều dài tổng cộng của 3 control lớn hơn chiều dài của Window, đồng thời, chiều dài của 2 control đầu (TextBlock và Button) nhỏ hơn chiều dài Window, WrapPanel sẽ xếp TextBlock cuối cùng xuống hàng dưới. Kết quả là:



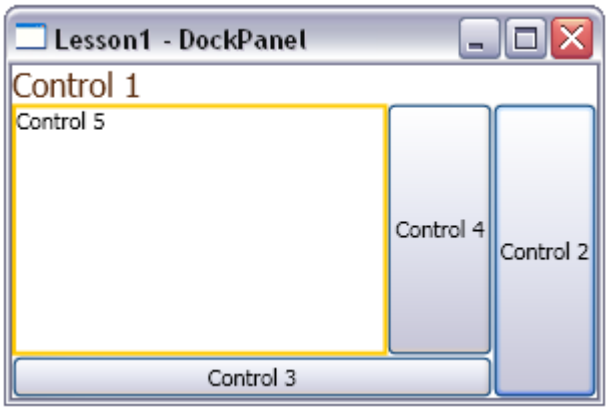
Hình 1.4 – Sử dụng WrapPanel

2.3 DockPanel

DockPanel cho phép các phần tử bám lên các cạnh của panel DockPanel bao chứa chúng, tương tự như khái niệm Docking trong Windows Forms. 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. Sau đây là đoạn mã XAML minh họa việc sử dụng DockPanel:

```
<DockPanel>
    <TextBlock FontSize="16" DockPanel.Dock="Top"
           Foreground="#58290A">
      Control 1
    </TextBlock>
    <Button DockPanel.Dock="Right">Control 2</Button>
    <Button DockPanel.Dock="Bottom">Control 3</Button>
    <Button DockPanel.Dock="Right">Control 4</Button>
    <Border BorderBrush="#feca00" BorderThickness="2">
      <TextBlock>Control 5</TextBlock>
    </Border>
</DockPanel>
```

Phần tử Border cuối cùng sẽ điền vào phần không gian còn lại vì thuộc tính `DockPanel.Dock` không xác định. Kết quả là:



Hình 1.5 – Sử dụng DockPanel – Control 1 bám vào cạnh trên DockPanel,
Control 2 và 4 nằm bên phải, Control 3 nằm dưới đáy; Control 5 chiếm toàn bộ không gian còn lại.

2.4 Canvas

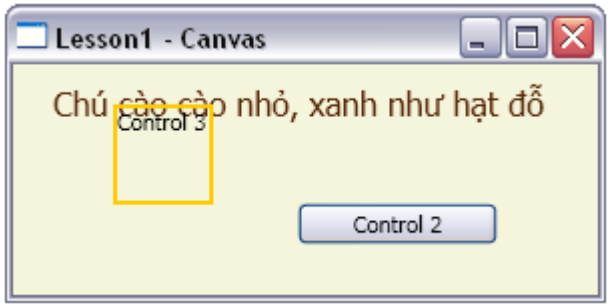
Panel dạng **Canvas** sử dụng phương thức sắp xếp các phần tử UI theo vị trí tuyệt đối bằng cách đặt thuộc tính Top (đỉnh) và Left (bên trái) của chúng. Thêm vào đó, thay vì đặt thuộc tính Top, Left, ta có thể đặt

thuộc tính Bottom (đáy), Right (bên phải). Nếu ta đặt đồng thời thuộc tính Left và Right, thuộc tính Right sẽ bị bỏ qua. Phần tử UI sẽ không thay đổi kích thước để thỏa mãn 2 thuộc tính trên cùng một lúc. Tương tự thuộc tính Top sẽ được ưu tiên hơn thuộc tính Bottom. 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.

Sau đây là một ví dụ minh họa việc sử dụng Canvas để sắp xếp các phần tử UI.

```
<!--Sử dụng Canvas-->
  <Canvas Background="Beige">
    <TextBlock FontSize="16" Canvas.Top="10" Canvas.Left="20"
               Foreground="#58290A">
      Chú cào cào nhỏ, xanh như hạt đỗ
    </TextBlock>
    <Button Canvas.Bottom="25" Canvas.Right="50" Width="100">Control 2</Button>
    <Border BorderBrush="#feca00" BorderThickness="2" Height="50" Width="50"
            Canvas.Top="20" Canvas.Left="50">
      <TextBlock>Control 3</TextBlock>
    </Border>
  </Canvas>
```

Vị trí của phần tử TextBlock đầu tiên và phần tử Border được đặt theo thuộc tính Top, Left, trong khi đó, phần tử Button được sắp vị trí theo thuộc tính Bottom, Right. Border sẽ nằm chồng lên TextBlock đầu tiên vì có sự xếp chồng về vị trí của hai phần tử này. Thêm vào đó, TextBlock đầu được khai báo trước Border trong đoạn mã XAML. Kết quả là:



Hình 1.6 – Sử dụng Canvas để sắp xếp các phần tử UI

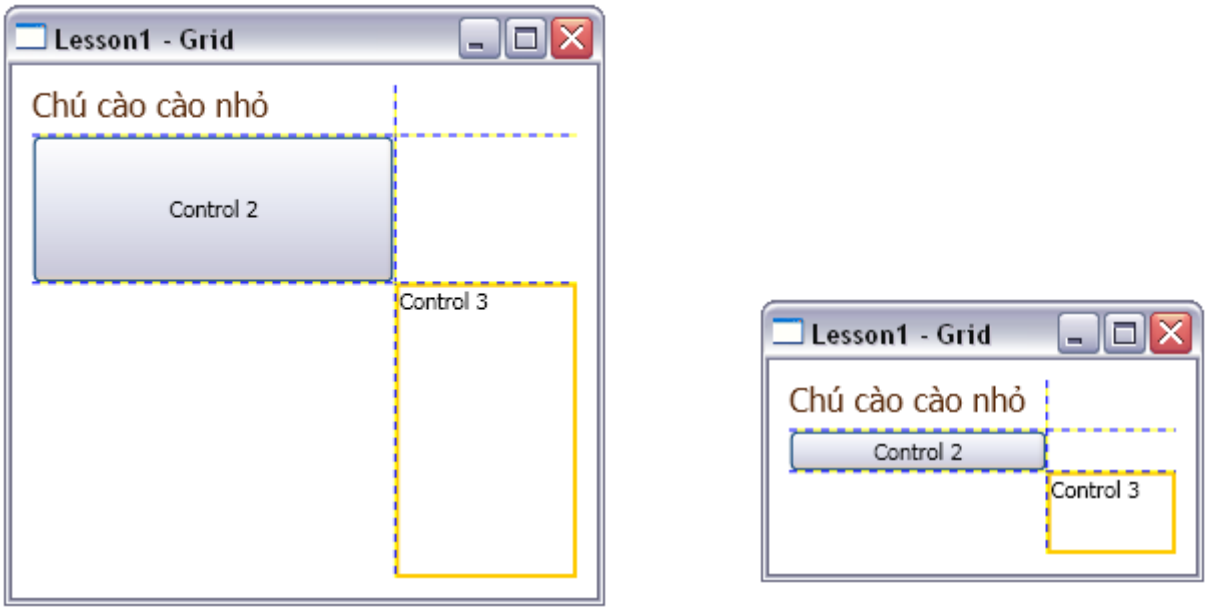
2.5 Grid

Panel dạng **Grid** là dạng panel hết sức linh hoạt, và có thể sử dụng để đạt được gần như tất cả khả năng mà các dạng panel khác có thể làm được, mặc dù mức độ khó dễ không giống nhau. Grid cho phép ta phân định các dòng và cột theo dạng một lưới kẻ ô, và sau đó sẽ sắp đặt các phần tử UI vào các ô tùy ý. Grid sẽ tự động chia đều các dòng và cột (dựa trên kích thước của phần nội dung). Tuy nhiên, ta có thể 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. Ta có thể nhận biết sự khác biệt của 2 dạng phân định kích thước nêu trên bằng cách thay đổi kích thước của form chứa panel Grid. Thêm vào đó, thuộc tính ShowGridLines được đặt bằng True cho phép hiển thị các đường kẻ ô. Sau đây là một ví dụ minh họa về việc sử dụng Grid với hai dạng phân định:

```
<!---Sử dụng panel Grid có các cạnh cách lề 10 pixel và có hiển thị các đường kẻ ô-->
  <Grid Margin="10" ShowGridLines="True">
    <!--Định nghĩa thuộc tính cột - Có tổng cộng 2 cột-->
    <Grid.ColumnDefinitions>
      <!--Khai báo cột 0 - có chiều rộng tỉ lệ gấp đôi cột kế tiếp-->
```

```
      <ColumnDefinition Width="2*" />
      <!--Khai báo cột 1 - Thuộc tính ngầm định-->
      <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <!--Định nghĩa thuộc tính hàng - Có tổng cộng 3 hàng -->
    <Grid.RowDefinitions>
      <!--Khai báo hàng 0 có chiều cao bằng 25 pixel-->
      <RowDefinition Height="25" />
      <!--Khai báo hàng 1 - Thuộc tính ngầm định -->
      <RowDefinition />
      <!--Khai báo hàng 2 - Đặt chiều cao gấp đôi hàng trước (hàng 1)-->
      <RowDefinition  Height="2*"/>
    </Grid.RowDefinitions>
    <!--Đặt TextBlock 1 vào cột 0 hàng 0-->
    <TextBlock FontSize="16"
               Foreground="#58290A"
               Grid.Column="0" Grid.Row="0">
      Chú cào cào nhỏ
    </TextBlock>
    <!--Đặt Button vào vị trí cột 0 hàng 1-->
    <Button Grid.Column="0" Grid.Row="1">
      Control 2
    </Button>
    <!--Đặt Border vào vị trí cột 1 hàng 2-->
    <Border BorderBrush="#feca00" BorderThickness="2"
            Grid.Column="1" Grid.Row="2">
      <TextBlock>Control 3</TextBlock>
    </Border>
  </Grid>
```

Sau đây là kết quả:



a) Trạng thái khởi tạo

b) Sau khi thay đổi kích thước form chứa Grid

Hình 1.7 – Sử dụng Grid để sắp xếp các phần tử UI – Khi kích thước form chứa Grid thay đổi, chiều cao của hàng 1 luôn cố định (25pixel), trong khi đó, chiều cao của hàng 2 luôn gấp đôi hàng 1; chiều rộng của cột 0 luôn gấp đôi chiều rộng của cột 1.

Câu hỏi ôn tập

1. ***Ứng dụng WPF sử dụng cách tiếp cận nào để bố trí các phần tử trên giao diện người dùng?***
   A. Phương thức áp đặt (imperative)
   B. Phương thức khai báo (declarative)
   C. Cả hai phương thức trên

   Câu trả lời: C

2. ***Trừ các dạng panel, các phần tử UI trong WPF như Label, Button cho phép chứa tối đa bao nhiêu phần tử con?***
   A. 0
   B. 1
   C. 2
   D. Không giới hạn số lượng

   Câu trả lời: B

3. ***Viết mã trình bằng C# để đạt được kết quả tương tự như đoạn mã XAML trong mục 2.1.2.***

Mã trình C# ví dụ:

```
        public Window7()
        {
            InitializeComponent();
            //
            //Đặt thuộc tính cửa sổ chính
            this.Height = 100; //chiều cao
            this.Width = 300;  //chiều rộng
            this.Title = "Lesson 1 - StackPanel"; //nhan đề
            //
            //Tạo StackPanel
            StackPanel stackPanel = new StackPanel();
            stackPanel.Background = new SolidColorBrush(Colors.Beige); //đặt thuộc
tính màu nền
            stackPanel.Orientation = Orientation.Horizontal;  //Đặt hướng sắp đặt
            //
            //Tạo đối tượng Border 1
            Border border1 = new Border();
            border1.Width = 90;
            border1.Height = 90;
            border1.BorderBrush = new SolidColorBrush(Colors.DarkSlateBlue); //đặt màu
đường viền
            border1.Background = new SolidColorBrush(Color.FromRgb(169, 233, 105));
            border1.BorderThickness = new Thickness(2); //đặt độ dày đường viền
            //
            //Tạo đối tượng CheckBox
            CheckBox checkbox = new CheckBox();
            checkbox.Content = "Control 1"; //đặt nội dung text của CheckBox
            //
            //Đặt CheckBox vào trong Border 1
            border1.Child = checkbox;
            //
            //Tạo đối tượng Button
            Button button = new Button();
            button.Width = 90;
            button.Height = 90;
            button.Content = "Control 2";
            //
            //Tạo đối tượng Border 2
            Border border2 = new Border();
```

```
            border2.Width = 90;
            border2.Height = 90;
            border2.BorderBrush = new SolidColorBrush(Color.FromRgb(254, 202, 0));
            border2.BorderThickness = new Thickness(2);
            //
            //Tạo đối tượng TextBlock
            TextBlock textBlock = new TextBlock();
            textBlock.Text = "Control 3";
            //
            //Đặt TextBlock vào trong Border 2
            border2.Child = textBlock;
            //
            //Đặt các đối tượng đã tạo vào trong StackPanel
            stackPanel.Children.Add(border1);
            stackPanel.Children.Add(button);
            stackPanel.Children.Add(border2);
            //
            //Đặt StackPanel vào cửa sổ chính
            this.Content = stackPanel;
          }
```

4. Viết mã XAML sử dụng Grid để đạt kết quả tương tự như đoạn mã trong mục 2.3.

Gợi ý: Sử dụng nhiều Grid lồng nhau.

Mã XAML vi dụ:

```
<Window x:Class="Lesson1.Window8"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Lesson1 - Cau hoi 4" Height="200" Width="300">
  <!--Tạo Grid 1 gồm 1 cột và 2 hàng-->
    <Grid Margin="0" ShowGridLines="False">
      <Grid.ColumnDefinitions>
        <ColumnDefinition/>
      </Grid.ColumnDefinitions>
      <Grid.RowDefinitions>
        <RowDefinition Height="25"/>
        <RowDefinition/>
      </Grid.RowDefinitions>
      <!--Tạo đối tượng TextBlock nằm ở ô [0,0] của Grid 1-->
      <TextBlock
        Grid.Column="0" Grid.Row="0" FontSize="16" DockPanel.Dock="Top"
               Foreground="#58290A">
        Control 1
      </TextBlock>

      <!--Tạo Grid 2 gồm 2 cột, 1 hàng, nằm trong cột 0 hàng 1 của Grid 1-->
      <Grid Margin="0" ShowGridLines="False" Grid.Column="0" Grid.Row="1">
        <Grid.ColumnDefinitions>
          <ColumnDefinition/>
          <ColumnDefinition Width="60"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
          <RowDefinition/>
        </Grid.RowDefinitions>

        <!--Tạo Grid 3 gồm 2 hàng, 1 cột nằm trong ô [0,0] của Grid 2-->
        <Grid Margin="0" ShowGridLines="False" Grid.Column="0" Grid.Row="0">
          <Grid.ColumnDefinitions>
            <ColumnDefinition/>
          </Grid.ColumnDefinitions>
          <Grid.RowDefinitions>
```

```
            <RowDefinition/>
            <RowDefinition Height="25" />
          </Grid.RowDefinitions>

          <!--Tạo Grid 4 gồm 2 cột, 1 hàng, nằm trong ô [0,0] của Grid 3-->
          <Grid Margin="0" Grid.Column="0" Grid.Row="0">
            <Grid.ColumnDefinitions>
              <ColumnDefinition/>
              <ColumnDefinition Width="60"></ColumnDefinition>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
              <RowDefinition/>
            </Grid.RowDefinitions>

            <!--Tạo đối tượng Border nằm trong ô [0, 0] của Grid 4-->
            <Border Grid.Column="0" Grid.Row="0" BorderBrush="#feca00"
BorderThickness="2">
              <TextBlock>Control 5</TextBlock>
            </Border>
            <!--Tạo đối tượng Button Control 4 nằm trong ô [0, 1] của Grid 4-->
            <Button Grid.Column="1" Grid.Row="0">Control 4</Button>
          </Grid>

          <!---Tạo đối tượng Button Control 3 tại ô [0,1] của Grid 3-->
          <Button Grid.Column="0" Grid.Row="1">Control 3</Button>
        </Grid>
        <!--Tạo đối tượng Button Control 2 nằm trong ô [0,1] của Grid 2-->
        <Button Grid.Column="1" Grid.Row="0">Control 2</Button>
      </Grid>
    </Grid>
</Window>
```

Tài liệu tham khảo

1. MSDN – Windows Presentation Foundation - The Layout System. http://msdn.microsoft.com/en-us/library/ms745058.aspx

2. How can I control the layout of items in WPF? URL: http://learnwpf.com/Posts/Post.aspx?postId=c76411d6-5350-4a10-b6bb-f1481c167ecf
3. Why WPF Rocks (Custom Layout Panel Showcase). http://dotnet.org.za/rudi/archive/2008/04/15/why-wpf-rocks-custom-layout-panel-showcase.aspx