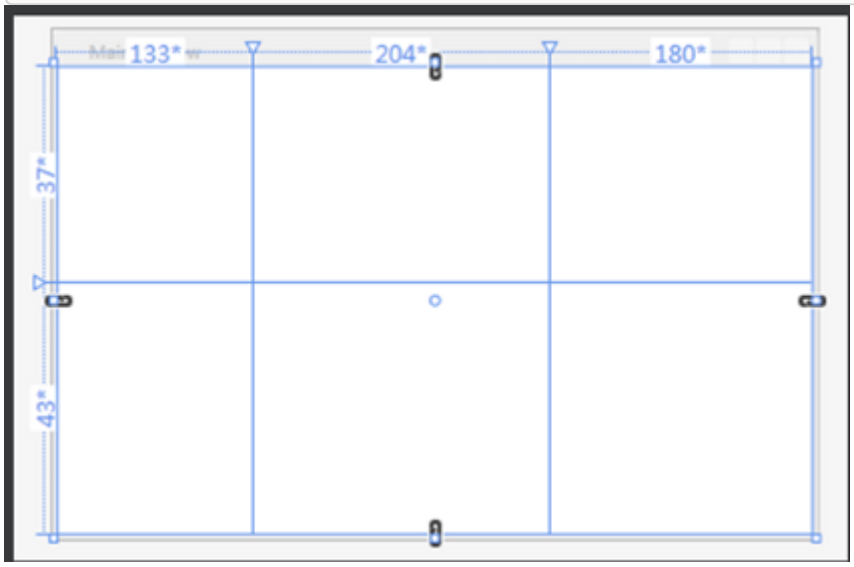
7/17

笔记本: 暑期实习
创建时间: 2021/7/17 16:56 **更新时间:** 2021/7/19 10:48
作者: 134exetj717
URL: http://wiki.suncaper.net/pages/viewpage.action?pageId=41877603

- WPF中的布局元素
    1. Grid
    2. StackPanel
    3. Canvas
    4. DockPanel
    5. WrapPanel
- Grid
    - 特点:
        1. 可以定义任意数量的行和列
    - 使用方法:
        1. 静态调整
        2. 动态调整

```
1.表示了两行三列，同生声明了行的高度，列的宽度（静态）
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="37*"/>
        <RowDefinition Height="43*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="133*"/>
        <ColumnDefinition Width="204*"/>
        <ColumnDefinition Width="180*"/>
    </Grid.ColumnDefinitions>
</Grid>
```



2.动态调整
如果需要动态调整Grid的布局，可以利用后台C#代码进行。假设当前窗体含有一个名为myGrid的Grid元素，通过窗体Loaded事件可以进行如下的行列添加操作。
这种操作特别适合一些动态布局，比如围棋棋盘，扫雷等区域会灵活变化的布局。

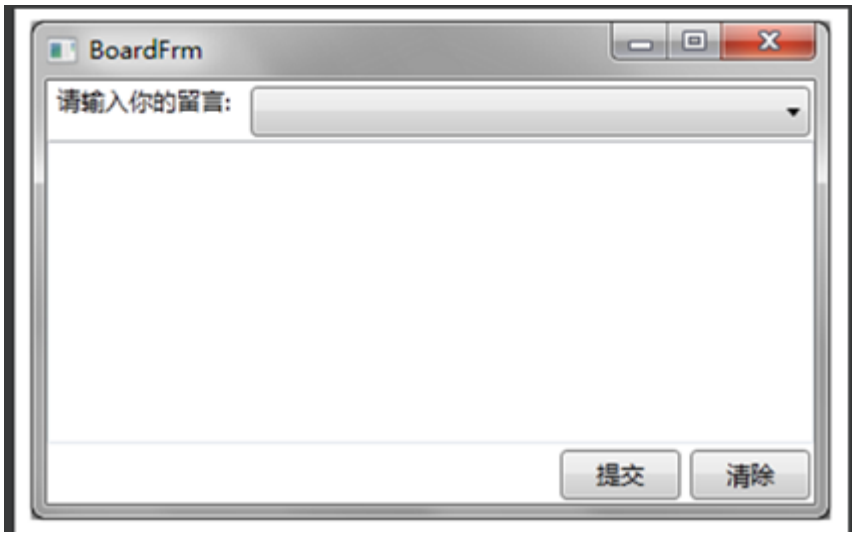
```
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            myGrid.Width = 250;    //定义了Grid的整体长和宽
            myGrid.Height = 100;
            myGrid.HorizontalAlignment = HorizontalAlignment.Left;    //水平排列方式
            myGrid.VerticalAlignment = VerticalAlignment.Top;        //垂直
            myGrid.ShowGridLines = true;    //是否显现
            // Define the Columns
            ColumnDefinition colDef1 = new ColumnDefinition();    //添加了3列
            ColumnDefinition colDef2 = new ColumnDefinition();
            ColumnDefinition colDef3 = new ColumnDefinition();
            myGrid.ColumnDefinitions.Add(colDef1);
            myGrid.ColumnDefinitions.Add(colDef2);
            myGrid.ColumnDefinitions.Add(colDef3);
            // Define the Rows
            RowDefinition rowDef1 = new RowDefinition();        //添加了2行
            RowDefinition rowDef2 = new RowDefinition();
            myGrid.RowDefinitions.Add(rowDef1);
            myGrid.RowDefinitions.Add(rowDef2);
        }
```

实际应用中我们除了定义行列外，最重要的还要设置行的高度和列的宽度才能满足真实项目布局的需要。对行高和列宽我们可以设置三类值：
（1）绝对值：数值后加上单位。
（2）比例值：数值后加上一个星号。前面案例就是这样的。
（3）自动值：Auto。
对于控件的高度和宽度不需要改变，或者是该行或列是专门用于精确间隔的时候，应该选用绝对值。而对于比例值，解析器会把所有的数值作为总和，单项的数值作为分子，然后将当前Grid的具体像素值与之相乘得到的结果作为其确切的值。这种方式最大的好处就是UI的整体尺寸变化时，比例值项目的行或列会保持其固有比例，也就是说能自动适应变化。

- 实战案例

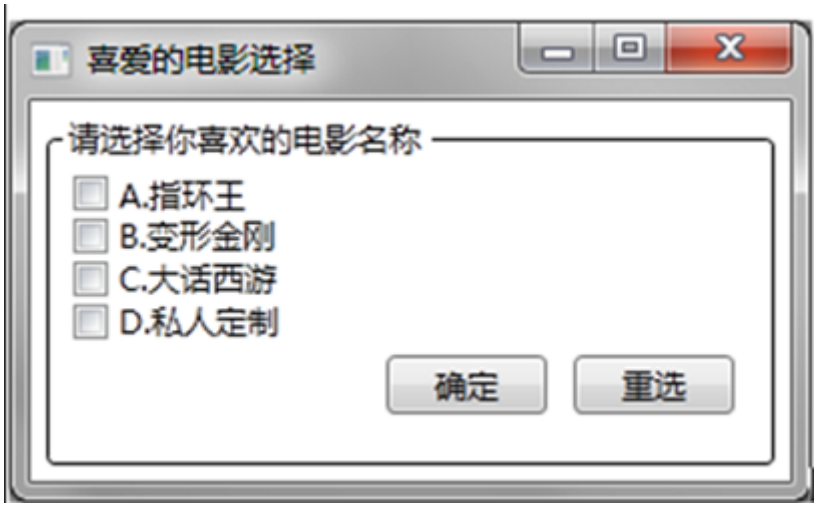
Grid中控件的布局行列归属是通过控件属性中的“Grid.Row”（行号）和“Grid.Column”（列号）来设定的。需要注意的是编号范围都是0~N-1。
如果一个控件需要跨多个行或列，还要结合使用“Grid.RowSpan”（跨行数）
和“Grid.ColumnSpan”（跨列数）。

```
<Grid>
        <Grid.ColumnDefinitions>        //定义了每一列的宽度
            <ColumnDefinition Width="100"/>
            <ColumnDefinition Width="148*"/>
            <ColumnDefinition Width="65"/>
            <ColumnDefinition Width="65"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="30"/>
            <RowDefinition Height="211*"/>
            <RowDefinition Height="30"/>
        </Grid.RowDefinitions>
               <TextBlock Text="请输入你的留言:" Width="90" Height="25"
Margin="5,2" Grid.Row="0" Grid.Column="0"/>
        <ComboBox Grid.Row="0" Grid.Column="1" Height="25" Margin="2"
Grid.ColumnSpan="3"/>
        <TextBox Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="4"/>
        <Button Content="提交" Width="60" Height="25" HorizontalAlignment="Right"
Grid.Row="2" Grid.Column="2" Margin="2"/>
        <Button Content="清除" Width="60" Height="25" HorizontalAlignment="Right"
Grid.Row="2" Grid.Column="3" Margin="2"/>
</Grid>
```



- StackPanel
  - 作用：可以把内部元素纵向或横向紧凑排列，形成栈式布局。通俗地来讲，就是像刨沙子一样，从底部挖的时候，上边的沙子会自动填补空缺。
  - 场合：菜单，列表，动画
  - 三个属性：
    1. Orientation属性，决定内部元素是横向累积还是纵向累积。
    2. HorizontalAlignment属性，决定内部元素水平方向上的对齐方式。
    3. VerticalAlignment属性，决定内部元素垂直方向上的对齐方式。
  - 实战案例：

```
<GroupBox Header="请选择你喜欢的电影名称" Margin="5">
            <StackPanel Margin="5">
                <CheckBox Content="A.指环王"/>
                <CheckBox Content="B.变形金刚"/>
                <CheckBox Content="C.大话西游"/>
                <CheckBox Content="D.私人定制"/>
                <StackPanel Orientation="Horizontal" HorizontalAlignment="Right">
                    <Button Content="确定" Width="60" Margin="5"/>
                    <Button Content="重选" Width="60" Margin="5"/>
                </StackPanel>
            </StackPanel>
</GroupBox>
```



- Canvas
  - 定义：“画布”，与Windows Form窗体上布局一致，WPF的空间中控件没有Left和Top属性，但是把控件放在Canvas中会被附加Canvas.X和Canvas.Y属性，就像放在Grid里会被附加上Grid.Column和Grid.Row
  - 场合：需要坐标定位的图画
  - 实战案例

```
<Canvas>
      <TextBlock Text="用户名：" Canvas.Left="12" Canvas.Top="12"/>
     <TextBox Height="23" Width="200" BorderBrush="Black" Canvas.Left="66"
Canvas.Top="9"/>
     <TextBlock Text="口令：" Canvas.Left="12" Canvas.Top="41"/>
     <TextBox Height="23" Width="200" BorderBrush="Black" Canvas.Left="66"
Canvas.Top="38"/>
     <Button Content="登录" Width="80" Height="22" Canvas.Left="70"
Canvas.Top="67"/>
     <Button Content="关闭" Width="80" Height="22" Canvas.Left="156"
Canvas.Top="67"/>
</Canvas>
```

- DockPanel
  - 定义：将空间切割为4部分。
  - 使用方法：

    1. Dock Panel内的元素会被附加**DockPanel.Dock**这个属性，这个属性的数据类型为Dock枚举。Dock枚举可取Left，Top，Right和Bottom四个值。根据Dock属性值。DockPanel内的元素会向指定方向累积、切分DockPanel内部的剩余可用空间，就像船舶靠岸一样。

    2. DockPanel还有一个重要属性—bool类型的**LastChildFill**，它的默认值是True当LastChildFill属性的值为True时，**DockPanel内最后一个元素的DockPanel.Dock属性值会被忽略，这个元素会把DockPanel内部所有剩余空间充满。**这也正好解释了为什么Dock枚举类型没有Fill这个值。

  - 实战案例：

```
<Grid>
        <DockPanel>
            <TextBox DockPanel.Dock="Top" Height="25" BorderBrush="Black"/>
            <TreeView DockPanel.Dock="Left" Width="100" BorderBrush="Black"/>
            <ListView BorderBrush="Black"/>
        </DockPanel>
```

```
</Grid>
```

- WrapPanel
  - 定义：“流式布局”，在流延伸的方向上排列尽可能多的控件，排不下的就另起一列。 WrapPanel使用Orientation属性来控制流延伸的方向，使用HorizontalAlignment和VerticalAlignment两个属性控制内部控件的对齐。
  - 实战案例：

```
<WrapPanel>
            <Button Content="一" Width="50" Height="50"/>
            <Button Content="二" Width="50" Height="50"/>
            <Button Content="三" Width="50" Height="50"/>
            <Button Content="四" Width="50" Height="50"/>
            <Button Content="五" Width="50" Height="50"/>
            <Button Content="六" Width="50" Height="50"/>
            <Button Content="七" Width="50" Height="50"/>
            <Button Content="八" Width="50" Height="50"/>
            <Button Content="九" Width="50" Height="50"/>
            <Button Content="十" Width="50" Height="50"/>
</WrapPanel>
```