WPF：数据驱动UI。

控件（Control）：展示数据、响应用户操作的UI元素。

控件展示的数据称为“数据内容”，控件响应用户的操作后会执行一些方法或以事件（Event）的形式通知应用程序称为控件的“行为”或“算法内容”。

Control是数据和行为的载体。

布局元素Panel：

包括：Canvas DockPanel Grid TabPanel

ToolBarOverflowPanel StackPanel ToolBarPanel UniformGrid

VirtualizingPanel VirtualizingStackPanel WrapPanel

Grid布局：

Grid布局中的比例分配：

例：程序xaml\_test8

<Window x:Class="xaml\_test8.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="MainWindow" Height="350" Width="525">

<Grid ShowGridLines="True">

<Grid.RowDefinitions>

<RowDefinition Height="25" />

<RowDefinition Height="1\*" />

<RowDefinition Height="2\*" />

<RowDefinition Height="2\*" />

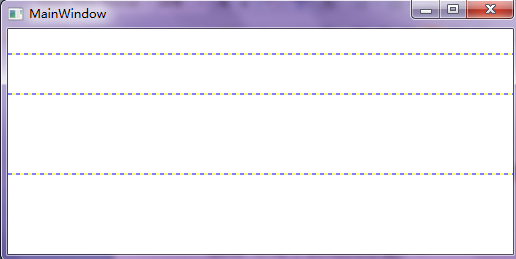
</Grid.RowDefinitions>

</Grid>

</Window>

上例中的Grid定义了4行，其中第一行高度为绝对值25px（像素），第二、三、四行的高度为相对值，Height值的后面加了“\*”，XAML解析器会把所有比例值相加作为分母，把每个比例值做为分子，用这个比例值乘以未被占用的空间的像素数，得到的结果为分配给这个比例值的最终像素值。

输出结果如下图所示：



当窗体大小变化时，第一行保持不变，而第二、三、四行的高度会随着窗体大小的变化而变化。如果不显示指定Height或Width，默认为比例值“1\*”，可以简写为“\*”

Grid布局例子：

例：程序xaml\_test9

<Window x:Class="xaml\_test9.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="留言板" Height="240" Width="400"

MinHeight="200" MinWidth="340" MaxHeight="400" MaxWidth="600">

<Grid Margin="10">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="80" />

<ColumnDefinition Width="4" />

<ColumnDefinition Width="80" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="25" />

<RowDefinition Height="4" />

<RowDefinition Height="\*" />

<RowDefinition Height="4" />

<RowDefinition Height="25" />

</Grid.RowDefinitions>

<TextBlock Text="请选择您的部门并留言：" Grid.Column="0" Grid.Row="0" VerticalAlignment="Center" />

<ComboBox Grid.Column="1" Grid.Row="0" Grid.ColumnSpan="4" />

<TextBox Grid.Column="0" Grid.Row="2" Grid.ColumnSpan="5" BorderBrush="Black" />

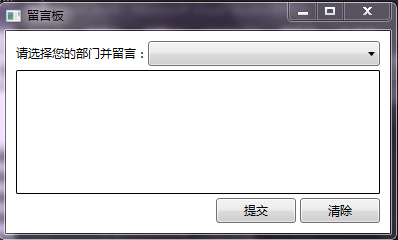
<Button Content="提交" Grid.Column="2" Grid.Row="4" />

<Button Content="清除" Grid.Column="4" Grid.Row="4" />

</Grid>

</Window>

输出结果为：



在Grid中为控件指定行和列遵循以下规则：

1. 行和列都是从0开始计数
2. 指定一个控件在某行，就为这个控件的标签添加Grid.Row=“行编号”这个Attribute；指定一个控件在某列，就为这个控件添加Grid.Column=“列编号”这个Attribute。
3. 若控件需要跨多个行或列，使用Grid.RowSpan=“行数”和Grid.ColumnSpan=“列数”。

StackPanel：栈式布局

适合：

1. 同类元素需要紧凑排列（如制作菜单或者列表）。
2. 移除其中的元素后能够自动补缺的布局或者动画。

Canvas：画布

放置在Canvas中的控件会附加Canvas.X和Canvas.Y属性

适合：

1. 一经设计基本上不会再有改动的小型布局（如图标）。
2. 艺术性比较强的布局。
3. 需要大量使用横纵坐标进行绝对点定位的布局。
4. 依赖于横纵坐标的动画。

WrapPanel：流式布局，在流延伸的方向上，WrapPanel会排列尽可能多的控件，排不下的控件会在新的一行或一列排列。

例：程序xaml\_test10

<Window x:Class="xaml\_test10.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="MainWindow" Height="350" Width="525">

<WrapPanel>

<Button Width="50" Height="50" Content="OK" />

<Button Width="50" Height="50" Content="OK" />

<Button Width="50" Height="50" Content="OK" />

<Button Width="50" Height="50" Content="OK" />

<Button Width="50" Height="50" Content="OK" />

<Button Width="50" Height="50" Content="OK" />

<Button Width="50" Height="50" Content="OK" />

<Button Width="50" Height="50" Content="OK" />

<Button Width="50" Height="50" Content="OK" />

<Button Width="50" Height="50" Content="OK" />

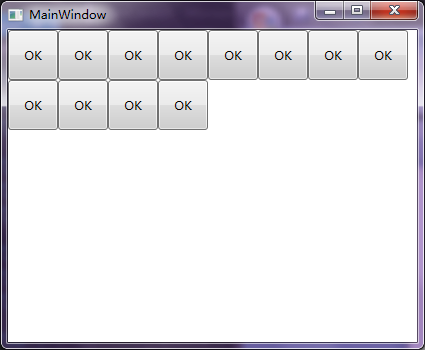
<Button Width="50" Height="50" Content="OK" />

<Button Width="50" Height="50" Content="OK" />

</WrapPanel>

</Window>

输出结果为：



当改变窗体大小时，按钮的排列会随之改变。