Overview of WPF layout controls:

<https://www.codeproject.com/Articles/30904/WPF-Layouts-A-Visual-Quick-Start>

### WPF Visual Quick Start Articles by Josh Fischer

* Layouts (this article)
* [Controls](https://www.codeproject.com/KB/WPF/WPFControlsQS.aspx)
* [Lists/Views](https://www.codeproject.com/KB/WPF/WPFItemsControlsQS.aspx)

Introduction

Switching from WinForms UI development to WPF can be intimidating. The question I kept asking myself is, "Where do I start?" Hopefully, this article will help you quickly develop a good starting point for WPF as it did for me. The examples are written in XAML, as it can be easily copied and pasted into the Visual Studio (or any other) designer. My goal is to show how the layouts work with screenshots and XAML, instead of using text descriptions.

Background

Layouts are the mechanism that WPF uses to define how visual items will be displayed on the screen for standard 2D applications. This concept is not new, and if you have used any of the Panel controls in WinForms (FlowLayoutPanel, TableLayoutPanel, etc.), you will not have any trouble understanding the basic layouts in WPF. In their simplest form, the layout classes are nothing more than containers for the other controls in your application and handle the positioning of those controls for you.

The Layouts

* **StackPanel** - Arranges controls in a line, either horizontally or vertically.



Hide   Copy Code

<StackPanel Orientation="Vertical"> <!-- Vertical is the default -->

<Label Background="Red">Red 1</Label>

<Label Background="LightGreen">Green 1</Label>

<StackPanel Orientation="Horizontal">

<Label Background="Red">Red 2</Label>

<Label Background="LightGreen">Green 2</Label>

<Label Background="LightBlue">Blue 2</Label>

<Label Background="Yellow">Yellow 2</Label>

<Label Background="Orange">Orange 2</Label>

</StackPanel>

<Label Background="LightBlue">Blue 1</Label>

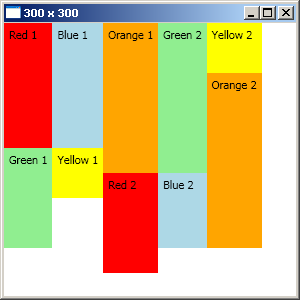
<Label Background="Yellow">Yellow 1</Label>

<Label Background="Orange">Orange 1</Label>

</StackPanel>

**Note:**

* + Items are stretched to fit their contents (text) and to fit the window.
  + By default, items will not fill all the available space (whitespace in screenshot).
  + Layouts can be nested within one another.
* **WrapPanel** - Arranges items in a line until the border is hit, then wraps to the next line.



***Vertical orientation***



***Horizontal orientation***

Hide   Copy Code

<WrapPanel Orientation="Vertical">

<Label Height="125" Background="Red">Red 1</Label>

<Label Height="100" Background="LightGreen">Green 1</Label>

<Label Height="125" Background="LightBlue">Blue 1</Label>

<Label Height="50" Background="Yellow">Yellow 1</Label>

<Label Height="150" Background="Orange">Orange 1</Label>

<Label Height="100" Background="Red">Red 2</Label>

<Label Height="150" Background="LightGreen">Green 2</Label>

<Label Height="75" Background="LightBlue">Blue 2</Label>

<Label Height="50" Background="Yellow">Yellow 2</Label>

<Label Height="175" Background="Orange">Orange 2</Label>

</WrapPanel>

<WrapPanel Orientation="Horizontal"> <!-- Horizontal is the default -->

<Label Width="125" Background="Red">Red 1</Label>

<Label Width="100" Background="LightGreen">Green 1</Label>

<Label Width="125" Background="LightBlue">Blue 1</Label>

<Label Width="50" Background="Yellow">Yellow 1</Label>

<Label Width="150" Background="Orange">Orange 1</Label>

<Label Width="100" Background="Red">Red 2</Label>

<Label Width="150" Background="LightGreen">Green 2</Label>

<Label Width="75" Background="LightBlue">Blue 2</Label>

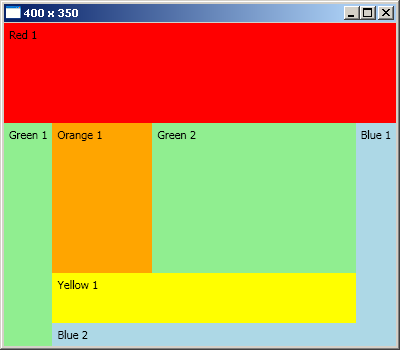
<Label Width="50" Background="Yellow">Yellow 2</Label>

<Label Width="175" Background="Orange">Orange 2</Label>

</WrapPanel>

**Note:**

* + Items are stretched to fit their contents (text).
  + If an item cannot fit entirely in the allotted space, the entire control is moved to the next line.
* **DockPanel** - Places controls in five different regions: top, bottom, left, right, or center (fill).



Hide   Copy Code

<DockPanel>

<Label DockPanel.Dock="Top" Height="100" Background="Red">Red 1</Label>

<Label DockPanel.Dock="Left" Background="LightGreen">Green 1</Label>

<Label DockPanel.Dock="Right" Background="LightBlue">Blue 1</Label>

<Label DockPanel.Dock="Bottom" Background="LightBlue">Blue 2</Label>

<Label DockPanel.Dock="Bottom" Height="50" Background="Yellow">Yellow 1</Label>

<Label Width="100" Background="Orange">

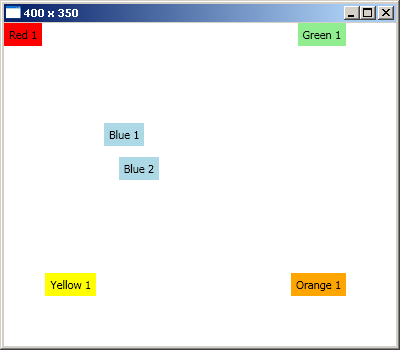
Orange 1</Label> <!-- default is to fill -->

<Label Background="LightGreen">Green 2</Label>

</DockPanel>

**Note:**

* + Child elements tell the parent DockPanel where they should be placed (DockPanel.Dock="Top").
  + You can place more than one control in a region (Yellow 1 and Blue 2).
  + If you do not specify a region, then the item will be set to fill (default behavior).
  + Items are stretched to fit their contents (text) and to fit the region they are in.
  + The size of any non-fill region is dictated by its contents (Red 1 has a height specified so it is larger).
* **Canvas** - Explicitly position controls based on coordinates.



Hide   Copy Code

<Canvas>

<!-- default coordinates 0,0 from top left; like WinForms -->

<Label Background="Red">Red 1</Label>

<Label Canvas.Right="50" Background="LightGreen" >Green 1</Label>

<Label Canvas.Top="100" Canvas.Left="100" Background="LightBlue" >Blue 1</Label>

<Label Canvas.Bottom="166" Canvas.Right="237" Background="LightBlue" >

Blue 2</Label>

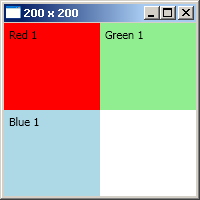
<Label Canvas.Right="300" Canvas.Top="250" Background="Yellow" >Yellow 1</Label>

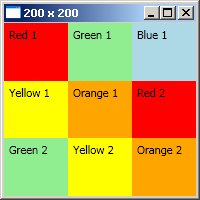
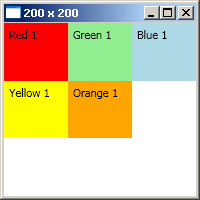
<Label Canvas.Right="50" Canvas.Bottom="50" Background="Orange" >Orange 1</Label>

</Canvas>

**Note:**

* + Child elements tell the parent Canvas where they should be placed (Canvas.Right="50").
  + When no dimensions are specified, the labels are stretched to fit their contents (text).
  + Controls can be placed relative to any side of the window (Blue 1 vs. Blue 2).
  + Top overrides Bottom and Left overrides Right.
* **UniformGrid** - Maintains a series of grid cells of equal size.





Hide   Copy Code

<UniformGrid>

<Label Background="Red">Red 1</Label>

<Label Background="LightGreen">Green 1</Label>

<Label Background="LightBlue">Blue 1</Label>

<Label Background="Yellow">Yellow 1</Label>

<Label Background="Orange">Orange 1</Label>

<Label Background="Red">Red 2</Label>

<Label Background="LightGreen">Green 2</Label>

<Label Background="Yellow">Yellow 2</Label>

<Label Background="Orange">Orange 2</Label>

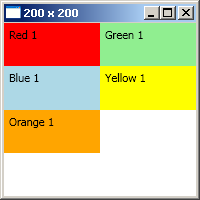
</UniformGrid>

**Note:**

* + Items are stretched to fit the cell.
  + The total number of cells increases by square integers.

| **# items** | **# cells** | **square of** |
| --- | --- | --- |
| 1 | 1 | 1 |
| 2-4 | 4 | 2 |
| 5-9 | 9 | 3 |
| 10-16 | 16 | 4 |
| 17-25 | 25 | 5 |
| ... | ... | ... |

* **Grid** - Defines static columns and rows; like HTML tables.



Hide   Copy Code

<!-- First screenshot -->

<Grid>

<!-- Using default column and row configurations -->

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<Label Grid.Column="0" Grid.Row="0" Background="Red">Red 1</Label>

<Label Grid.Column="1" Grid.Row="0" Background="LightGreen">Green 1</Label>

<Label Grid.Column="0" Grid.Row="1" Background="LightBlue">Blue 1</Label>

<Label Grid.Column="1" Grid.Row="1" Background="Yellow">Yellow 1</Label>

<Label Grid.Column="0" Grid.Row="2" Background="Orange">Orange 1</Label>

</Grid>

<!-- 2nd screenshot uses ColumnSpan and RowSpan -->

<Label Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="2"

Background="Red">Red 1</Label>

<Label Grid.Column="0" Grid.Row="1" Grid.RowSpan="3"

Background="LightGreen">Green 1</Label>

**Note:**

* + Child elements tell the parent grid what cell they should be in.
  + Items are stretched to fill the cell.
  + Each row or column can be given a specific size (Height/Width).
  + Cell contents can be told to span multiple rows (RowSpan) or columns (ColumnSpan).

Points of Interest

Please note this article is intended as a -visual- quick start for beginners. For a more in depth discussion, please see Sacha Barber's [excellent article](https://www.codeproject.com/KB/WPF/BeginWPF1.aspx), or refer to the MSDN documentation.