

# Основи розробки графічного інтерфейсу користувача засобами WPF

Інструментальні засоби візуального програмування

Тема 03, ЧДБК-2019

# План лекції

- Компонування інтерфейсу за допомогою контейнерних елементів
- Основні елементи управління XAML
- Взаємодія з додатком за допомогою подій
- Команди як інструмент взаємодії з додатком

# Компонування інтерфейсу за допомогою контейнерних елементів

Питання 3.1.

# Компонування за допомогою панелей

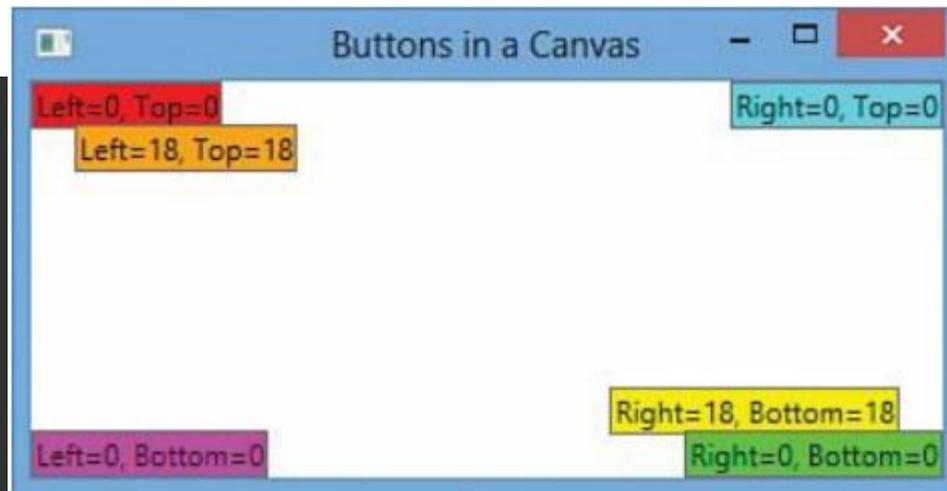
- WPF містить вбудовані панелі, які допомагають уникнути неприємностей з неякісним компонуванням.
- Розглянемо 5 основних вбудованих панелей (у просторі імен `System.Windows.Controls`):
  - Canvas
  - StackPanel
  - WrapPanel
  - DockPanel
  - Grid

# Панель Canvas

- Canvas (холст) – найпростіша панель.
  - Використовувати для організації інтерфейсу користувача не варто.
  - Підтримує тільки позиціонування за координатами. впрочем, координаты хотя бы задаются в независимых от устройства пикселах, в отличие от прежних систем конструирования пользовательских интерфейсов.
  - Дозволяє задавати координати відносно будь-якого кута.
- Позиціонування на холсті відбувається за допомогою приєднаних властивостей: `Left`, `Top`, `Right` и `Bottom`.
  - Визначають, що найближча сторона елемента завжди повинна бути на фіксованій відстані від відповідної сторони холста.
- Вказується кут, до якого «примыкает» кожен елемент.
  - Приєднані властивості виступають в ролі полів (к которым добавляются значения самого свойства `Margin` элемента).
  - Якщо приєднаних властивостей для елемента не задано, він поміщається в лівий верхній кут (`Left = 0`, `Top = 0`).

# Кнопки на панели Canvas

```
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        Title="Buttons in a Canvas">
  <Canvas>
    <Button Background="Red">Left=0, Top=0</Button>
    <Button Canvas.Left="18" Canvas.Top="18"
           Background="Orange">Left=18, Top=18</Button>
    <Button Canvas.Right="18" Canvas.Bottom="18"
           Background="Yellow">Right=18, Bottom=18</Button>
    <Button Canvas.Right="0" Canvas.Bottom="0"
           Background="Lime">Right=0, Bottom=0</Button>
    <Button Canvas.Right="0" Canvas.Top="0"
           Background="Aqua">Right=0, Top=0</Button>
    <Button Canvas.Left="0" Canvas.Bottom="0"
           Background="Magenta">Left=0, Bottom=0</Button>
  </Canvas>
</Window>
```



# Взаимодействие Canvas со свойствами компоновки дочерних элементов

Свойство	Допустимо ли внутри Canvas
Margin	Частично. Для двух сторон, использованных для позиционирования элемента (по умолчанию Top и Left), к значениям присоединенных свойств прибавляются соответствующие значения двух из четырех полей
HorizontalAlignment и VerticalAlignment	Нет. Элементам назначается в точности та величина вертикального выравнивания, которая им необходима
LayoutTransform	Да. Отличается от RenderTransform тем, что при использовании LayoutTransform элементы всегда отстоят на заданное расстояние от выбранного угла Canvas

# Z-порядок

- Визначає порядок розташування елементів один понад іншим.
  - Можна задавати явно за допомогою приєднаної властивості Zindex

```
<Canvas>  
<Button Canvas.ZIndex="1" Background="Red">On Top!</Button>  
<Button Background="Orange">On Bottom with a Default ZIndex=0</Button>  
</Canvas>
```

- Якщо для кількох елементів задано один Zindex, їх взаємне розташування визначається порядком слідування в колекції Children відповідної панелі
- У C#-кодi це можна здійснити так:

```
Panel.SetZIndex(redButton, 0);
```

# Панель Canvas примітивна, проте швидка

- Canvas дуже зручна для точного позиціонування примітивних фігур у векторних зображеннях.
  - Дає змогу максимально точно контролювати розміщення інтерфейсів, добиваючись найвищої продуктивності.

# Панель StackPanel

- Послідовно розміщає свої дочірні елементи у вигляді стопки.
  - Якщо властивість `FlowDirection = RightToLeft`, то для панелі `StackPanel` з горизонтальною орієнтацією збирання стопки відбувається справа наліво.
  - За відсутності приєднаних властивостей єдиний спосіб організувати дочірні елементи – використати властивість `Orientation` (тип `System.Windows.Controls.Orientation`), яка може набувати значень `Horizontal` або `Vertical`.
  - За замовчуванням орієнтація `Vertical`.

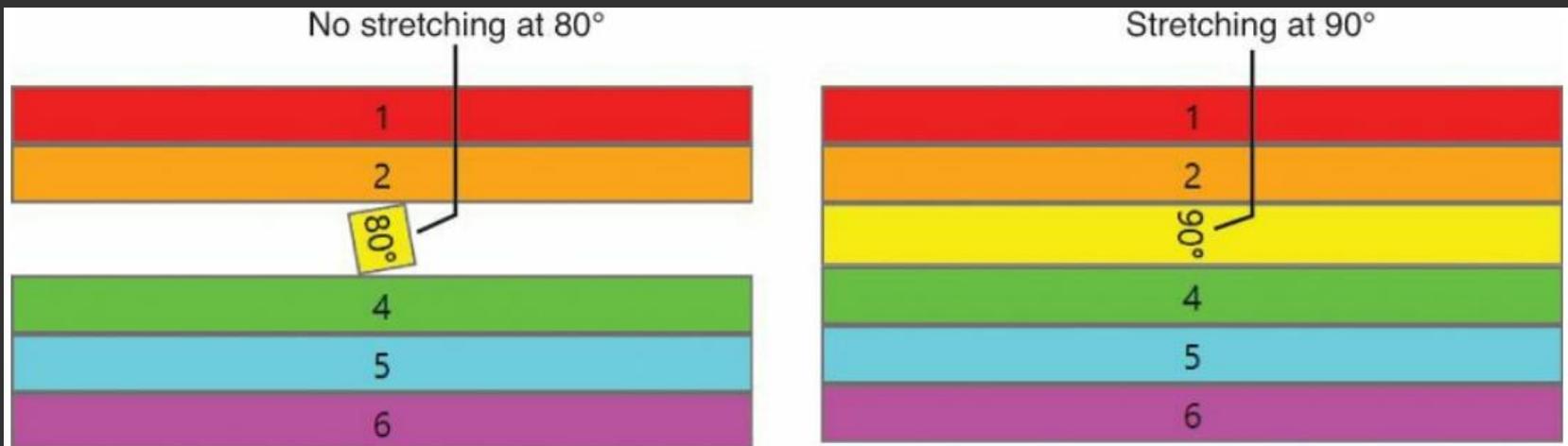


# Деякі властивості панелі StackPanel

Свойство	Допустимо ли внутри StackPanel
Margin	Да. Свойство Margin управляет промежутком между элементом и краями StackPanel, а также промежутком между краями соседних элементов
HorizontalAlignment и VerticalAlignment	Частично, поскольку выравнивание игнорируется в направлении сборки стопки (так как дочерним элементам отводится ровно столько места, сколько им необходимо). Если Orientation="Vertical" ,то игнорируется значение VerticalAlignment. Если Orientation="Horizontal", то игнорируется значение HorizontalAlignment
LayoutTransform	Да. Отличается от RenderTransform тем, что при использовании LayoutTransform оставшиеся в стопке элементы сдвигаются вниз, чтобы освободить место. При комбинировании компоновки Stretch преобразованием RotateTransform или SkewTransform, применяемым в режиме LayoutTransform, растяжение происходит, только если угол кратен

# Огляд елемента управління ToolStrip

- при повороті елемента, який у нормальних умовах був би розтягнутим, розтяг відбувається лише у випадку, коли краї елемента паралельні або перпендикулярні напрямку розтягу.
  - Така поведінка присутня всюди, де елемент розтягується тільки в одному напрямку.
  - Вона проявляється тільки тоді, коли перетворення застосовується в режимі `LayoutTransform`; до `RenderTransform` вона не відноситься.



# Віртуалізуючі панелі

Важной деталью реализации нескольких элементов управления являются панели, производные от абстрактного класса `System.Windows.Controls.VirtualizingPanel`. Наиболее интересная из них - панель `VirtualizingStackPanel`, которая работает, как `StackPanel`, но для повышения производительности временно игнорирует все элементы, не видимые на экране (только во время привязки к данным). Поэтому `VirtualizingStackPanel` является оптимальной панелью, когда требуется привязать к данным по-настоящему много дочерних элементов, и класс `ListBox` использует ее по умолчанию. Эту панель можно использовать также в `TreeView` о чем пойдет речь в главе 10 «Многодетные элементы управления». Еще две виртуализирующие панели – `DataGridCellsPane` и `DataGridRowsPresenter`, они используются в классе `DataGrid` и ассоциированных с ним (см. главу 11 «Изображения, текст и другие элементы управления»).

# Панель WrapPanel

- схожа на StackPanel, проте крім організації дочірніх елементів у стопку вона створює нові рядки або стовпці, коли для однієї стопки не вистачає місця.
  - Корисно для відображення на екрані заздалегідь невідомої кількості елементів, коли компонування повинно відрізнитись від простого списку, як наприклад, у Провіднику Windows.
  - Як і StackPanel, панель WrapPanel не володіє приєднаними властивостями для управління положенням елементів.

# Панель WrapPanel

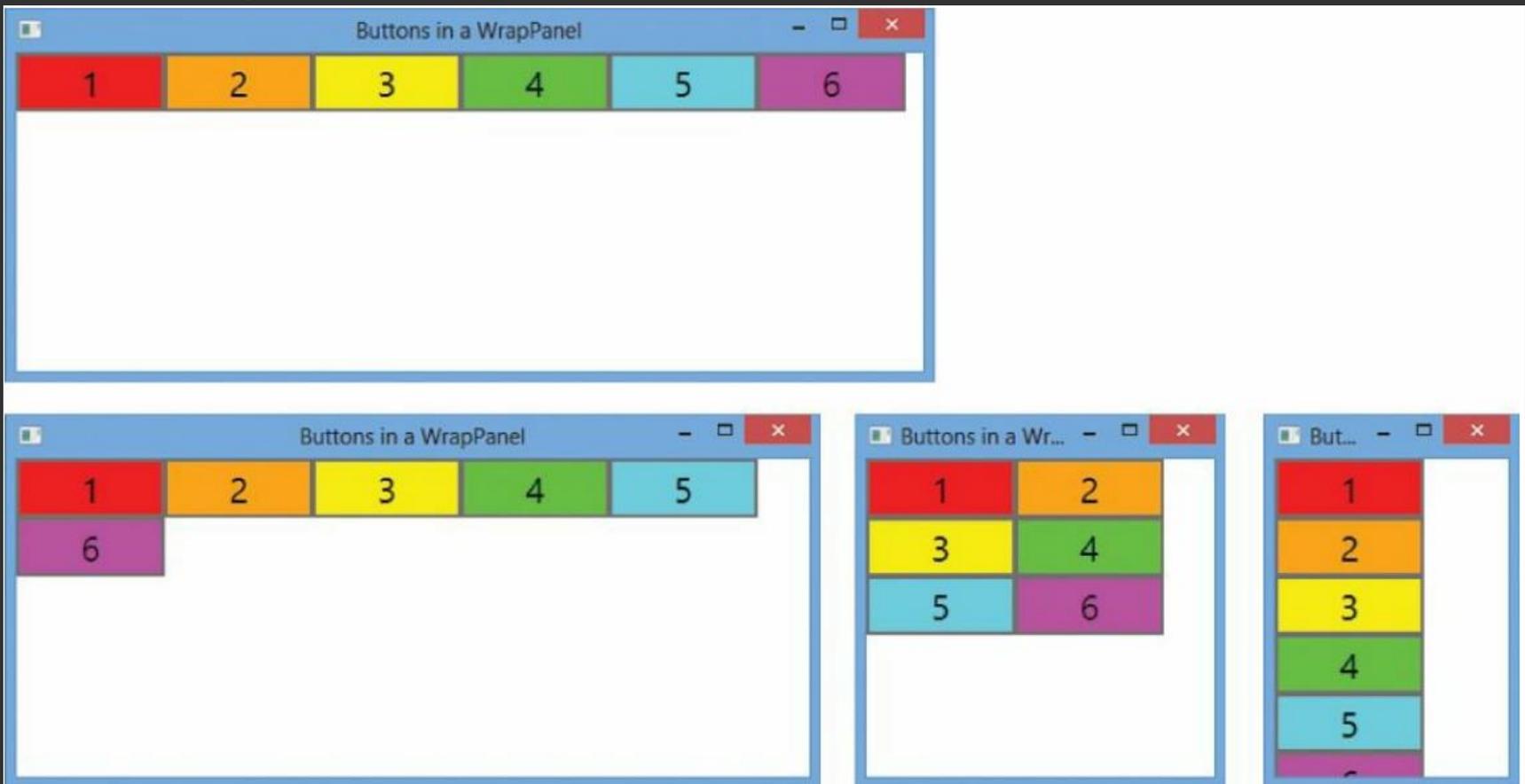
- У класі WrapPanel визначено 3 властивості, що контролюють його поведінку:
  - Orientation – аналогічно властивості StackPanel, проте за замовчуванням – значення Horizontal.
    - Панель с горизонтальною орієнтацією виглядає, як вид Ескизи сторінок в Проводнику Windows: елементи розполагаються один за другим слева направо, а коли місце закінчується, переходять на наступну строку.
    - Панель с вертикальною орієнтацією виглядає, як вид Список в Проводнику Windows: елементи розполагаються один під другим, а коли місце закінчується, починається новий стовбець.
  - ItemHeight – єдина висота для всіх дочірніх елементів. Таким образом кожен потомок розпоряджається цією висотою, залежить від значень його властивостей VerticalAlignment, Height і пр. Елементи, ширина яких перевищує ItemHeight, отсекаються.
  - ItemWidth – єдина ширина для всіх дочірніх елементів. Таким образом кожен потомок розпоряджається цією шириною, залежить від значень його властивостей HorizontalAlignment, Width і пр. Елементи, висота яких перевищує ItemWidth, отсекаються.

- По умолчанию свойства `ItemHeight` и `ItemWidth` не установлены (точнее, имеют значение `Double.NaN`).
- В этом случае панель `WrapPanel` с вертикальной ориентацией отводит каждому столбцу ширину, равную ширине самого широкого элемента в нем, а панель с горизонтальной ориентацией отводит каждой строке высоту, равную высоте самого высокого элемента в ней.
- Поэтому по умолчанию ни в строках, ни в столбцах отсечение не производится.

#### СОВЕТ

Можно заставить панель `WrapPanel` располагать элементы в одну строку или в один столбец. Для этого следует присвоить свойству `Width` (в случае горизонтальной ориентации) или свойству `Height` (в случае вертикальной ориентации) значение `Double.MaxValue` либо `Double.PositiveInfinity`. В XAML это достигается с помощью расширения разметки `x:Static`, поскольку ни то ни другое значение не поддерживается конвертером типа `System.Double`.

# Динамічні скріншоти при зміні розмірів вікна



# Взаимодействие WrapPanel со свойствами компоновки дочерних элементов

Свойство	Допустимо ли внутри WrapPanel
Margin	Да. Поля учитываются, когда WrapPanel вычисляет размеры элементов, чтобы определить подразумеваемую по умолчанию ширину или высоту стопки
HorizontalAlignment и VerticalAlignment	Частично. Выравнивание можно задавать в направлении, противоположном направлению роста стопки, как и в случае StackPanel. Но выравнивание может быть полезно и в направлении роста стопки, если значение ItemHeight или ItemWidth таково, что в элементе имеется дополнительное пространство для выравнивания
LayoutTransform	Да. Отличается от RenderTransform тем, что при использовании LayoutTransform оставшиеся элементы сдвигаются, чтобы освободить место, но только если не установлено свойство ItemHeight или ItemWidth (в зависимости от ориентации). При комбинировании компоновки Stretch с преобразованием RotateTransform или SkewTransform, применяемым в режиме LayoutTransform, растяжение происходит, только если угол кратен 90°, как и в случае StackPanel

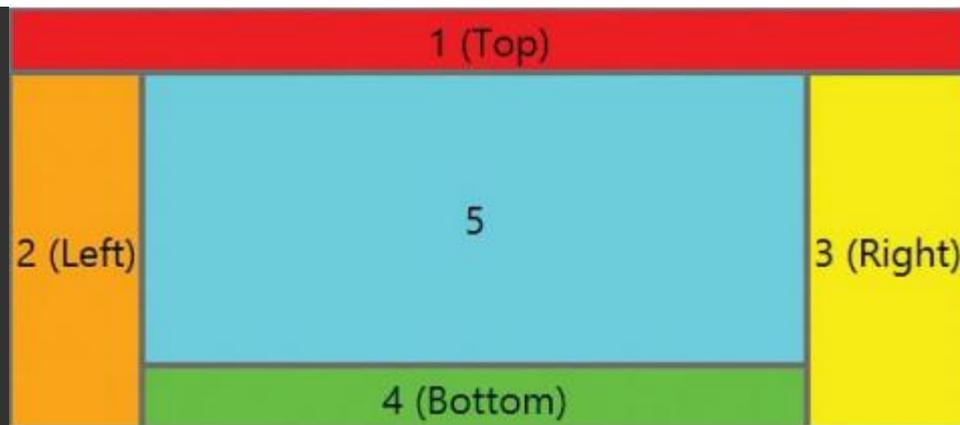
○ Панель WrapPanel обычно используется не для компоновки элементов управления внутри окна Window, а для компоновки внутри вложенных элементов управления.

# Панель DockPanel

- дает простой способ пристыковки элемента к одной из сторон, растягивая его на всю имеющуюся ширину или высоту.
  - Отличие от Canvas заключается в том, что элементы пристыковываются не к одному углу, а ко всей стороне.
  - Кроме того, DockPanel позволяет расположить один элемент так чтобы он занял все место, свободное от пристыкованных элементов.
- В классе DockPanel определено присоединенное свойство Dock, с помощью которого дочерние элементы могут управлять своим положением.
  - Если LastChildFill равно true (по умолчанию), то значение свойства Dock, заданное для последнего добавленного элемента, игнорируется.
  - Если же оно равно false, то последний элемент можно пристыковать к любой стороне (по умолчанию к левой, Left).

# Панель DockPanel

```
<DockPanel>  
  <Button DockPanel.Dock="Top" Background="Red">1 (Top)</Button>  
  <Button DockPanel.Dock="Left" Background="Orange">2 (Left)  
</Button>  
  <Button DockPanel.Dock="Right" Background="Yellow">3 (Right)  
</Button>  
  <Button DockPanel.Dock="Bottom" Background="Lime">4 (Bottom)  
</Button>  
  <Button Background="Aqua">5</Button>  
</DockPanel>
```



# Панель DockPanel

- Как и в случае StackPanel, растяжение элементов определяется подразумеваемым по умолчанию значением свойства HorizontalAlignment или VerticalAlignment.
- Если элемент не будет занимать все пространство, выделенное ему панелью DockPanel, то можно задать другое выравнивание.

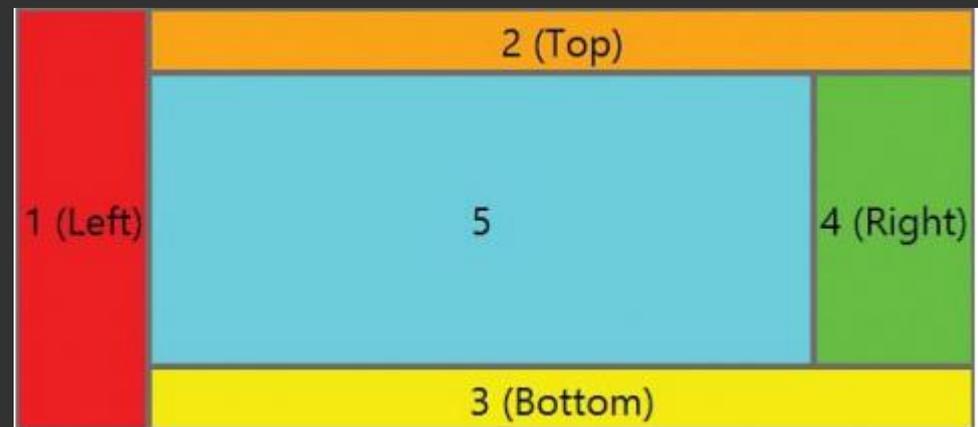
```
<DockPanel>
  <Button DockPanel.Dock="Top" HorizontalAlignment="Right"
    Background="Red">1 (Top, Align=Right)</Button>
  <Button DockPanel.Dock="Left" VerticalAlignment="Bottom"
    Background="Orange">2 (Left, Align=Bottom)</Button>
  <Button DockPanel.Dock="Right" VerticalAlignment="Bottom"
    Background="Yellow">3 (Right, Align=Bottom)</Button>
  <Button DockPanel.Dock="Bottom" HorizontalAlignment="Right"
    Background="Lime">4 (Bottom, Align=Right)</Button>
  <Button Background="Aqua">5</Button>
</DockPanel>
```

# Інтерфейс, отриманий з попередніх кодів



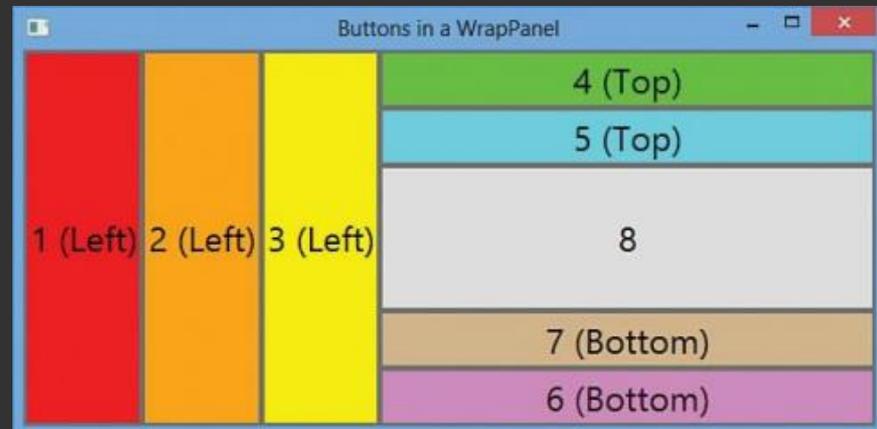
хотя четыре элемента отказались занимать все выделенное им место, нераспределенное пространство не отдается другим элементам.

показаны те же пять кнопок, что и на слайде 104, но добавленные в другом порядке (обозначенном числом и цветом). Обратите внимание, что компоновка изменилась.



- Панель DockPanel полезна для организации верхнего уровня интерфейса внутри элемента Window или Page, когда пристыкованные элементы по большей части представляют собой другие панели, где и находится все самое важное.
  - Так, обычно к верхней стороне пристыковывается меню (Menu), справа и слева находятся какие-то панели, а снизу - строка состояния (StatusBar).
  - Центральную же часть занимают основные данные приложения.

- DockPanel поддерживает произвольное количество дочерних элементов
- показана панель DockPanel с восьмью дочерними элементами — три слева, два сверху, два снизу и один заполняет оставшееся пространство.



# Взаимодействие DockPanel со свойствами компоновки дочерних элементов

Свойство	Допустимо ли внутри DockPanel
Margin	Да. Свойство Margin определяет, сколько места оставлять между элементом и стороной панели, а также промежуток между самими элементами
HorizontalAlignment и VerticalAlignment	Частично. Как и в случае StackPanel, выравнивание в направлении пристыковки игнорируется. Иначе говоря, если Dock равно Left или Right, то не имеет смысла задавать свойство HorizontalAlignment, а если Top или Bottom то свойство VerticalAlignment. Однако для элемента, заполняющего оставшееся пространство, применимы оба свойства, HorizontalAlignment и VerticalAlignment
LayoutTransform	LayoutTransform Да. Отличается от RenderTransform тем, что при использовании LayoutTransform оставшиеся элементы сдвигаются, чтобы освободить место. При комбинировании компоновки Stretch с преобразованием RotateTransform или SkewTransform, применяемым в режиме LayoutTransform, растяжение происходит, только если угол кратен 90°, за исключением элемента, заполняющего оставшееся пространство (потому что он может растягиваться в обоих направлениях)

Таким образом, DockPanel является обобщением StackPanel.

Если свойство LastChildFill равно false, то DockPanel ведет себя, как

- горизонтальная панель StackPanel, когда все потомки пристыковываются к левой стороне,
- вертикальная - когда все потомки пристыковываются к верхней стороне.

# Панель Grid

- позволяет расположить дочерние элементы в несколько строк и несколько столбцов, не полагаясь на режим автоматического переноса (как в WrapPanel).
- Кроме того, предоставляется ряд интересных способов управления строками и столбцами.

```
<Grid Background="LightBlue">  
  <!-- Define four rows: -->  
  <Grid.RowDefinitions>  
    <RowDefinition/>  
    <RowDefinition/>  
    <RowDefinition/>  
    <RowDefinition/>  
  </Grid.RowDefinitions>  
  <!-- Define two columns: -->  
  <Grid.ColumnDefinitions>  
    <ColumnDefinition/>  
    <ColumnDefinition/>  
  </Grid.ColumnDefinitions>
```

## СОВЕТ

В WPF также имеется класс Table(в пространстве имен System.Windows.Documents), предоставляющий примерно такие же возможности, как Grid. Но Table не наследует классу Panel (и даже классу UIElement). Этот класс, производный от FrameworkContentElement, предназначен для отображения содержимого документов

# Продовження коду

```
<!-- Arrange the children: -->
<Label Grid.Row="0" Grid.Column="0" Background="Black"
      Foreground="White" HorizontalContentAlignment =
      "Center">Start Page</Label>
<GroupBox Grid.Row="1" Grid.Column="0" Background="White"
      Header="Start">...</GroupBox>
<GroupBox Grid.Row="2" Grid.Column="0" Background="White"
      Header="Recent">...</GroupBox>
<GroupBox Grid.Row="3" Grid.Column="0" Background="White"
      Header="Options">...</GroupBox>
<GroupBox Grid.Row="1" Grid.Column="1" Background="White"
      Header="Get Started">
  <ListBox>
    <ListBoxItem>Article #1</ListBoxItem>
    <ListBoxItem>Article #2</ListBoxItem>
    <ListBoxItem>Article #3</ListBoxItem>
    <ListBoxItem>Article #4</ListBoxItem>
  </ListBox>
</GroupBox>
</Grid>
```



Сразу бросается в глаза недостаток: список онлайн-статей (Online articles) слишком мал.

Кроме того, было бы лучше, если бы заголовок StartPage занимал всю ширину сетки.

В простейшем случае мы определяем количество строк и столбцов, помещая нужное число элементов RowDefinition и ColumnDefinition внутрь элементов, соответствующих свойствам сетки RowDefinitions и ColumnDefinitions.

Затем позиционируем дочерние элементы с помощью присоединенных свойств Row и Column, принимающих целочисленные значения, начиная с 0.

- Ячейки сетки можно оставлять пустыми, и, наоборот, в одной ячейке может находиться несколько элементов.

# Вирішення проблем

- обе проблемы легко решить с помощью дополнительных присоединенных свойств класса Grid: RowSpan и ColumnSpan.
- По умолчанию свойства RowSpan и ColumnSpan равны 1, но могут принимать любое значение, большее или равное 1, - соответственно количество строк столбцов, занимаемых данной ячейкой.

если добавить в последний элемент GroupBox в листинге атрибут:  
**Grid.RowSpan= '3 '**

а в элемент Label- атрибут  
**Grid.ColumnSpan= '2 '**



# Вирішення проблем

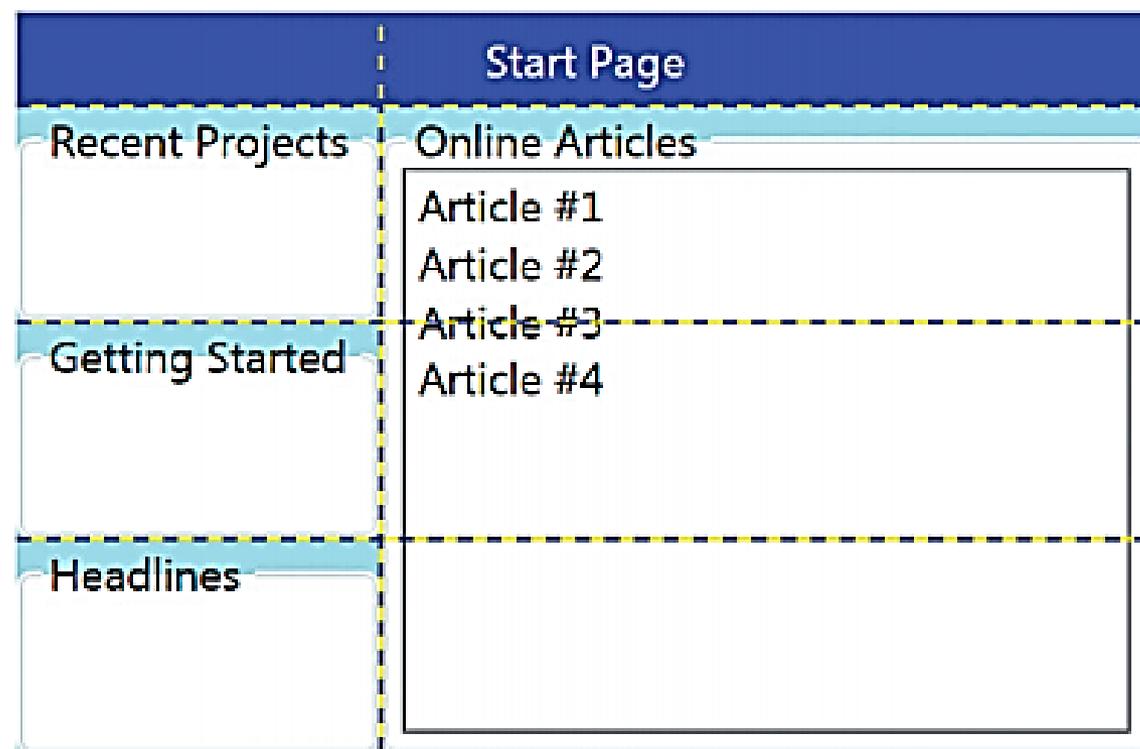
- Сетка на рисунке все еще выглядит не вполне удовлетворительно, потому по умолчанию высоты всех строк и ширины всех столбцов равны.
- В идеале нужно было бы оставить больше места для списка онлайн-статей, а затем указав, что размеры первой строки и первого столбца должны соответствовать содержимому.
  - Такой автоматический выбор размера достигается присваивания свойствам `Height` и `Width` соответственно в элементах `RowDefinition` и `ColumnDefinition` специального значения `Auto`, нечувствительного к регистру букв.

```
<!-- Define four rows: -->
<Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition/>
    <RowDefinition/>
    <RowDefinition/>
</Grid.RowDefinitions>
<!-- Define two columns: -->
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition/>
</Grid.ColumnDefinitions>
```



## СОВЕТ

У панели Grid имеется простое свойство ShowGridLines; если оно равно true, то ячейки будут разграничены желто-синими пунктирными линиями. В промышленном приложении это не к чему, зато полезно для «отладки» компоновки. На рис.5.13 показан результат установки свойства ShowGridLines="True" для сетки, изображенной на рис. 5.12.



# Задание размеров строк и столбцов

- В отличие от элементов типа `FrameworkElement`, свойства `Height` и `Width` элементов `RowDefinition` и `ColumnDefinition` по умолчанию не равны `Auto`.
  - они имеют тип `System.Windows.GridLength`, а не `double`.
- панель `Grid` поддерживает три способа задания размера в элементах `RowDefinition` и `ColumnDefinition`:
  - **Абсолютный размер** - числовое значение `Height` или `Width` означает, что размер задан в независимых от устройства пикселах. Абсолютные значения не позволяют строкам и столбцам увеличиваться или сжиматься при изменении размера самой сетки `Grid` или находящихся внутри нее элементов.
  - Автоматический выбор размера
  - Пропорциональное изменение размера

# Автоматичний вибір та пропорційна зміна розміру

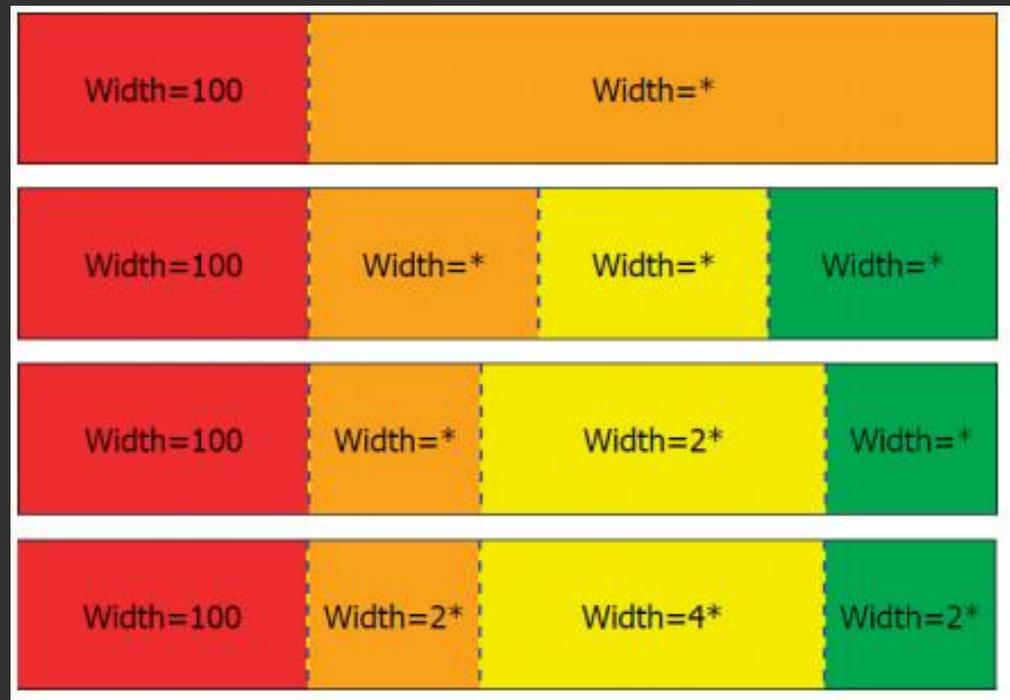
- Автоматический выбор размера – если Height или Width равно Auto, то дочерним элементам выделяется столько места, сколько необходимо, но не больше (для свойств Height и Width во всех остальных классах WPF это режим по умолчанию).
  - Для строки эта величина равна высоте самого высокого элемента, а для столбца - ширине самого широкого элемента.
  - Когда речь идет о тексте, режим лучше задания абсолютных размеров, так как можно не опасаться отсечения из-за выбора другого шрифта или локализации.
- Пропорциональное изменение размера («звездочка») - предусмотрен специальный синтаксис задания свойств Height и Width, позволяющий распределить имеющееся пространство поровну или в соответствии с заданными пропорциями.
  - Если задано пропорциональное изменение размера, строка и столбец увеличиваются или сжимаются при изменении размера сетки.

# Режим \*

- Если высота строки или ширина столбца равна \*, то соответствующему структурному элементу выделяется все оставшееся место.
- Если размер \* задан для нескольких строк или столбцов, то все оставшееся место делится между ними поровну.
- Перед символом \* можно указывать коэффициент (например, 2\* или 5.5\*), тогда соответствующей строке или столбцу будет выделено пропорционально больше места, чем остальным строкам или столбцам, в размере которых присутствует символ \*.
  - Столбец шириной 2\* всегда в два раза шире столбца шириной \* в той же самой сетке.
  - Столбец шириной 5.5\* в два раза шире столбца шириной 2.75\* в той же самой сетке.

# Пропорционально измеренные столбцы сетки

- Под словами «оставшееся пространство» понимается высота или ширина сетки за вычетом всех строк или столбцов, для которых задан абсолютный размер или его автоматический выбор.



# Интерактивное задание размера с помощью GridSplitter

- В сетку Grid можно добавить произвольное число дочерних элементов GridSplitter, указав для них присоединенные свойства Grid.Row, Grid.Column, Grid.RowSpan и/или Grid.ColumnSpan, как для любых других потомков.
- Буксировка GridSplitter изменяет размер по меньшей мере одной ячейки.
- Что происходит с остальными - изменение размера или просто перемещение - зависит от заданного способа изменения размера: пропорционально или как-то иначе.
  - По умолчанию ячейки, на которых изменение размера отражается непосредственно, определяются свойствами выравнивания GridSplitter.

# Ячейки, изменяемые непосредственно при буксировке GridSplitter с различным выравниванием

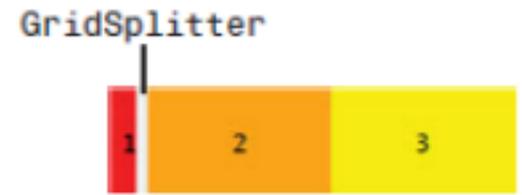
		HorizontalAlignment			
		Left	Right	Center	Stretch
VerticalAlignment	Top	Current cell and cell to the left	Current cell and cell to the right	Cells to the left and right	Current cell and cell above
	Bottom	Current cell and cell to the left	Current cell and cell to the right	Cells to the left and right	Current cell and cell below
	Center	Current cell and cell to the left	Current cell and cell to the right	Cells to the left and right	Cells above and below
	Stretch	Current cell and cell to the left	Current cell and cell to the right	Cells to the left and right	Cells to the left and right if GridSplitter is taller than it is wide, or cells to the top and bottom if GridSplitter is wider than it is tall

# Задание общего размера для строк и столбцов

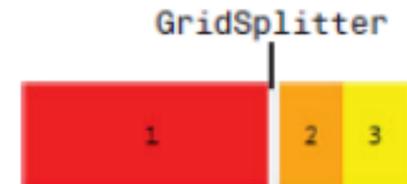
- В классах `RowDefinitions` и `ColumnDefinitions` имеется свойство `SharedSizeGroup` позволяющее задать режим, при котором линейные размеры нескольких строк и/или столбцов будут оставаться одинаковыми даже в случае, когда размер любой из них изменяется в процессе выполнения программы (например, с помощью `GridSplitter`).
  - Свойству `SharedSizeGroup` можно присвоить произвольное строковое значение (чувствительное к регистру); оно интерпретируется как имя группы.
  - Размеры всех строк или столбцов, находящихся в одной группе, изменяются синхронно.

# Приклад

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition/>
    <ColumnDefinition/>
  </Grid.ColumnDefinitions>
  <Label Grid.Column="0" Background="Red"
    HorizontalContentAlignment="Center"
    VerticalContentAlignment="Center">1</Label>
  <GridSplitter Grid.Column="0" Width="5"/>
  <Label Grid.Column="1" Background="Orange"
    HorizontalContentAlignment="Center"
    VerticalContentAlignment="Center">2</Label>
  <Label Grid.Column="2" Background="Yellow"
    HorizontalContentAlignment="Center"
    VerticalContentAlignment="Center">3</Label>
</Grid>
```



Компоновка по умолчанию



Компоновка после буксировки  
SharedSizeGroup вправо

Чтобы элемент GridSplitter был виден и доступен для использования, его ширина Width(или высота Height - в зависимости от ориентации) должна быть задана явно.



```
<Grid IsSharedSizeScope="True">
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" SharedSizeGroup="myGroup"/>
    <ColumnDefinition/>
    <ColumnDefinition SharedSizeGroup="myGroup"/>
  </Grid.ColumnDefinitions>
  <Label Grid.Column="0" Background="Red"
    HorizontalContentAlignment="Center"
    VerticalContentAlignment="Center">1</Label>
  <GridSplitter Grid.Column="0" Width="5"/>
  <Label Grid.Column="1" Background="Orange"
    HorizontalContentAlignment="Center"
    VerticalContentAlignment="Center">2</Label>
  <Label Grid.Column="2" Background="Yellow"
    HorizontalContentAlignment="Center"
    VerticalContentAlignment="Center">3</Label>
</Grid>
```

показано, что происходит с той же сеткой, когда для первого и последнего столбцов задано одинаковое значение SharedSizeGroup.

- Свойство `IsSharedSizeScope` следует установить потому, что группы размеров могут применяться сразу к нескольким сеткам!
  - Чтобы избежать потенциального конфликта имен (и сократить расходы на необходимый в этом случае ход логических деревьев), все сетки, к которым применяется одно и то же значение свойства `SharedSizeGroup`, должны находиться под общим родителем, а свойство `IsSharedSizeScope` для них должно быть равно `true`.

```
<StackPanel Grid.IsSharedSizeScope="True">
  <Grid>...can use SharedSizeGroup...</Grid>
  <Grid>...can use SharedSizeGroup...</Grid>
  <WrapPanel>
    <Grid>...can use SharedSizeGroup...</Grid>
  </WrapPanel>
</StackPanel>
```

Это не просто свойство зависимости в классе `Grid`, но еще и присоединенное свойство, которое можно задавать для родителей, не являющихся сетками

# Сравнение Grid с другими панелями

- Панель Grid - лучший выбор для особо сложной компоновки, потому что она умеет делать все, на что способны другие панели, а также многое другое.
  - Единственное, чего ей не хватает, так это умения динамически генерировать новые строки и столбцы, как WrapPanel.

## **Моделирование Canvas с помощью Grid**

Если взять сетку с одной строкой и одним столбцом и для всех дочерних элементов задавать любое значение `HorizontalAlignment` и `VerticalAlignment`, кроме `Stretch`, то добавляемые в единственную ячейку элементы будут вести себя так же, как при добавлении на панель `Canvas`. Задание `HorizontalAlignment` равным `Left` и `VerticalAlignment` равным `Top` эквивалентно установке для `Canvas.Left` и `Canvas.Top` значения 0. Задание `HorizontalAlignment` равным `Right` и `VerticalAlignment` равным `Bottom` эквивалентно установке для `Canvas.Right` и `Canvas.Bottom` значения 0. Кроме того, задание для каждого элемента поля `Margin` может дать такой же эффект, как присвоение таких же значений присоединенным свойствам `Canvas`. Именно так поступает конструктор `VisualStudio`, когда пользователь помещает или передвигает элементы на поверхности конструктора.

## **Моделирование StackPanel с помощью Grid**

Сетка с одним столбцом и автоматически измеряемыми строками выглядит, так же, как вертикальная стопка StackPanel, если каждый элемент вручи помещать в последовательно нумеруемые строки. Аналогично сетка со строкой и автоматически изменяемыми столбцами выглядит так же, как горизонтальная стопка, если каждый элемент вручную помещать в последовательно нумеруемые столбцы.

## **Моделирование DockPanel с помощью Grid**

С помощью свойств RowSpan и ColumnSpan легко сделать так, чтобы внешние элементы пристыковывались к сторонам сетки и автоматически растягивались как на панели DockPanel. На рис. 5.12 метка Label, по сути, пристыкована к верхней стороне.

В табл. 5.6 показано, как некоторые из свойств компоновки дочерних элементов применяются к элементам, расположенным на панели Grid.

# Таблица 5.6. Взаимодействие Grid со свойствами компоновки дочерних элементов

Свойство	Допустимо ли внутри DockPanel
Margin	Да. Свойство Margin определяет, сколько места оставлять между элементом и сторонами объемлющей его ячейки
HorizontalAlignment и VerticalAlignment	Да. В отличие от остальных панелей, можно в полной мере использовать оба направления, если только не окажется, ячейка с автоматическим изменением размеров вообще не оставила элементу дополнительного места. Поэтому по умолчанию большинство элементов растягиваются, заполняя свои ячейке целиком
LayoutTransform	Да. Отличается от RenderTransform тем, что при использовании LayoutTransform элементы остаются внутри ячеек (если это возможно) и учитывается величина поля Margin. В отличие от RenderTransform, элемент, вышедший в результате масштабирования за пределы ячейки, отсекается

**Дякую за увагу!**