



## Grid

Последнее обновление: 16.01.2016



Это наиболее мощный и часто используемый контейнер, напоминающий обычную таблицу. Он содержит столбцы и строки, количество которых задает разработчик. Для определения строк используется свойство **RowDefinitions**, а для определения столбцов - свойство **ColumnDefinitions**:

```
1 <Grid.RowDefinitions>
2     <RowDefinition></RowDefinition>
3     <RowDefinition></RowDefinition>
4     <RowDefinition></RowDefinition>
5 </Grid.RowDefinitions>
6 <Grid.ColumnDefinitions>
7     <ColumnDefinition></ColumnDefinition>
8     <ColumnDefinition></ColumnDefinition>
9     <ColumnDefinition></ColumnDefinition>
10 </Grid.ColumnDefinitions>
```

Каждая строка задается с помощью вложенного элемента `RowDefinition`, который имеет открывающий и закрывающий тег. При этом задавать дополнительную информацию необязательно. То есть в данном случае у нас определено в гриде 3 строки.

Каждая столбец задается с помощью вложенного элемента `ColumnDefinition`. Таким образом, здесь мы определили 3 столбца. То есть в итоге у нас получится таблица 3x3.

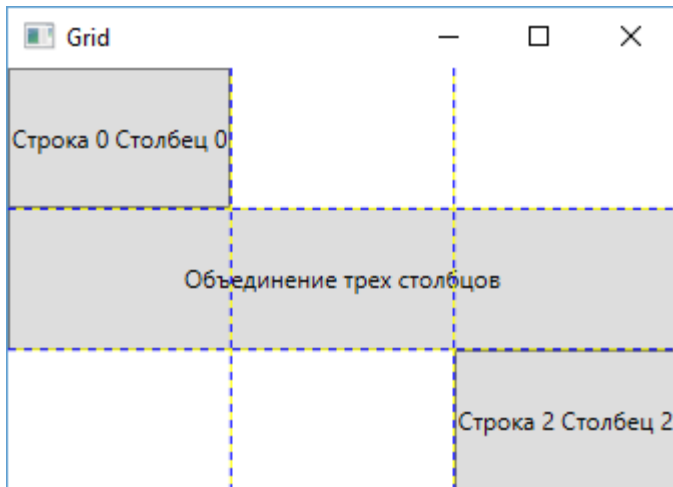
Чтобы задать позицию элемента управления с привязкой к определенной ячейке Grid, в разметке элемента нужно прописать значения свойств **Grid.Column** и **Grid.Row**, тем самым указывая, в каком столбце и строке будет находиться элемент. Кроме того, если мы хотим

растянуть элемент управления на несколько строк или столбцов, то можно указать свойства **Grid.ColumnSpan** и **Grid.RowSpan**, как в следующем примере:

```
1 <Window x:Class="LayoutApp.MainWindow"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6     xmlns:local="clr-namespace:LayoutApp"
7     mc:Ignorable="d"
8     Title="Grid" Height="250" Width="350">
9     <Grid ShowGridLines="True">
10         <Grid.RowDefinitions>
11             <RowDefinition></RowDefinition>
12             <RowDefinition></RowDefinition>
13             <RowDefinition></RowDefinition>
14         </Grid.RowDefinitions>
15         <Grid.ColumnDefinitions>
16             <ColumnDefinition></ColumnDefinition>
17             <ColumnDefinition></ColumnDefinition>
18             <ColumnDefinition></ColumnDefinition>
19         </Grid.ColumnDefinitions>
20         <Button Grid.Column="0" Grid.Row="0" Content="Строка 0 Столбец 0" />
21         <Button Grid.Column="0" Grid.Row="1" Content="Объединение трех столбцов" Grid.ColumnSpan="3" />
```

Атрибут `ShowGridLines="True"` у элемента `Grid` задает видимость сетки, по умолчанию оно равно `False`.

То есть у нас получится следующая картина:



## Установка размеров

Но если в предыдущем случае у нас строки и столбцы были равны друг другу, то теперь попробуем их настроить столбцы по ширине, а строки - по высоте. Есть несколько вариантов настройки размеров.

### Автоматические размеры

Здесь столбец или строка занимает то место, которое им нужно

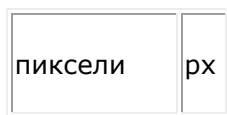
```
1 <ColumnDefinition Width="Auto" />
2 <RowDefinition Height="Auto" />
```

### Абсолютные размеры

В данном случае высота и ширина указываются в единицах, независимых от устройства:

```
1 <ColumnDefinition Width="150" />
2 <RowDefinition Height="150" />
```

Также абсолютные размеры можно задать в пикселях, дюймах, сантиметрах или точках:



дюймы	in
сантиметры	cm
точки	pt

Например,

```
1 <ColumnDefinition Width="1 in" />
2 <RowDefinition Height="10 px" />
```

### Пропорциональные размеры.

Например, ниже задаются два столбца, второй из которых имеет ширину в четверть от ширины первого:

```
1 <ColumnDefinition Width="*" />
2 <ColumnDefinition Width="0.25*" />
```

Если строка или столбец имеет высоту, равную \*, то данная строка или столбец будет занимать все оставшееся место. Если у нас есть несколько сток или столбцов, высота которых равна \*, то все доступное место делится поровну между всеми такими сроками и столбцами. Использование коэффициентов (0.25\*) позволяет уменьшить или увеличить выделенное место на данный коэффициент. При этом все коэффициенты складываются (коэффициент \* аналогичен 1\*) и затем все пространство делится на сумму коэффициентов.

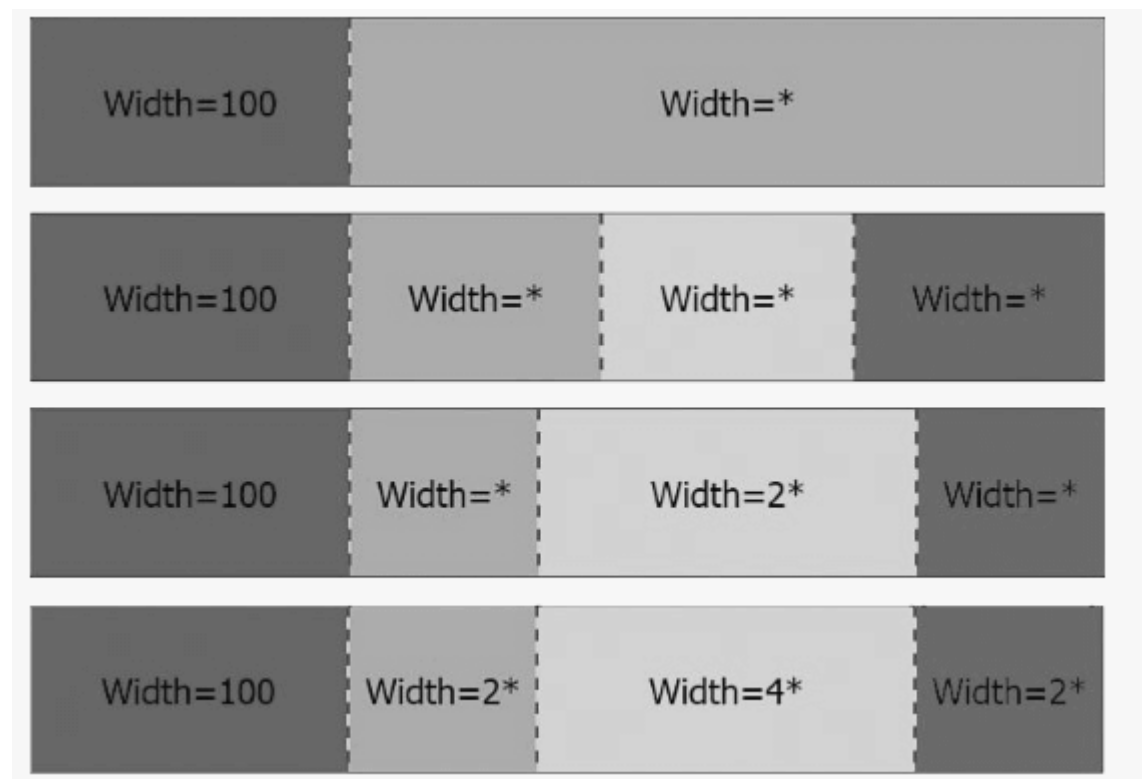
Например, если у нас три столбца:

```
1 <ColumnDefinition Width="*" />
2 <ColumnDefinition Width="0.5*" />
3 <ColumnDefinition Width="1.5*" />
```

В этом случае сумма коэффициентов равна  $1* + 0.5* + 1.5* = 3*$ . Если у нас грид имеет ширину 300 единиц, то для коэффициент  $1*$  будет соответствовать пространству  $300 / 3 = 100$  единиц. Поэтому первый столбец будет иметь ширину в 100 единиц, второй -  $100*0.5=50$  единиц, а третий -  $100 * 1.5 = 150$  единиц.

Можно комбинировать все типы размеров. В этом случае от ширины/высоты грида отнимается ширина/высота столбцов/строк с абсолютными или автоматическими размерами, и затем оставшееся место распределяется между столбцами/строками с

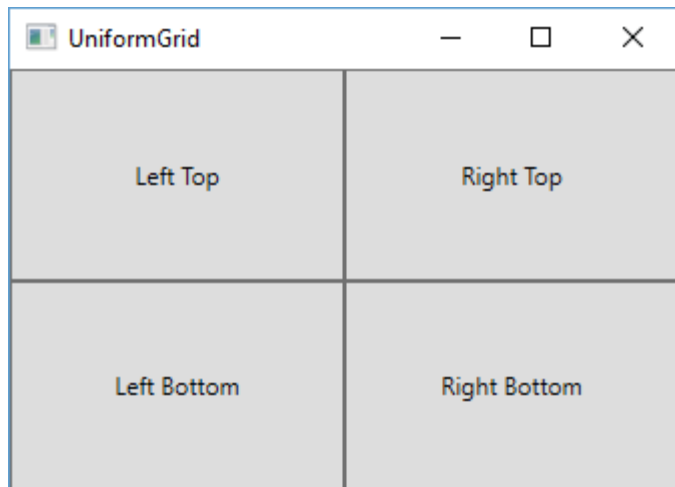
пропорциональными размерами:



## UniformGrid

Аналогичен контейнеру Grid контейнер **UniformGrid**, только в этом случае все столбцы и строки одинакового размера и используется упрощенный синтаксис для их определения:

```
1 <UniformGrid Rows="2" Columns="2">
2     <Button Content="Left Top" />
3     <Button Content="Right Top" />
4     <Button Content="Left Bottom" />
5     <Button Content="Right Bottom" />
6 </UniformGrid>
```



[Назад](#) [Содержание](#) [Вперед](#)



ТАКЖЕ НА METANIT.COM

### Google представил новую ОС - Fuchsia

3 месяца назад • 2 коммент...

Google представил новую ОС - Fuchsia для устройства Google ...

### Добавление данных в MySQLi

2 месяца назад • 1 коммент...

Добавление данных в БД MySQL в языке PHP с помощью библиотеки ...

### Удаление данных в MySQLi

2 месяца назад • 1 коммент...

Удаление данных в БД MySQL в языке PHP с помощью библиотеки ...

### Первое графическое приложение на ...

2 месяца назад • 1 коммент...

Первое графическое приложение на Rust для Windows 10, проект Rust ...

### Введение в ко

2 месяца назад • 1 к

Введение в коруты языка программирования Kotlin, асинхронно

10 Комментариев metanit.com  Политика конфиденциальности Disqus

 Войти

 Рекомендовать 10  Твитнуть  Поделиться

Лучшее в начале



Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS ?



Имя



**Семён Новиков** • 3 года назад

Огромное спасибо за уроки) Теперь это мой любимый сайт.

9 ^ | v • Ответить • Поделиться ›



**Лиля** • 5 лет назад

здравствуйте!

мне тут дали один совет (человек имеет большой опыт в xaml), что в grid лучше использовать или столбцы, или строки, но не вместе. Просто игратья вложением, т.е., допустим, в строку вложить еще один grid уже со столбцами... Что Вы скажете о таком подходе?

2 ^ | v 1 • Ответить • Поделиться ›



**Роман Тимохов** ➔ Лиля • 5 лет назад

Мне сдается ваш товарищ поступает так по причине того что ранее верстал в HTML. и таким образом создает аналогию блока div.

6 ^ | v • Ответить • Поделиться ›



**Metanit** Модератор ➔ Лиля • 5 лет назад

Ну нередко так и происходит. В основном в силу упрощения интерфейсов и, возможно, так действительно проще иной раз управлять расположением элементов. Однако причин совсем уж не использовать в одном гриде по несколько столбцов и строк я не вижу.

А чем это аргументировалось что такой подход более оптимальный?

4 ^ | v • Ответить • Поделиться ›