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**:

```
<Grid.RowDefinitions>
    <RowDefinition></RowDefinition>
    <RowDefinition></RowDefinition>
    <RowDefinition></RowDefinition>
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
    <ColumnDefinition></ColumnDefinition>
    <ColumnDefinition></ColumnDefinition>
    <ColumnDefinition></ColumnDefinition>
</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.

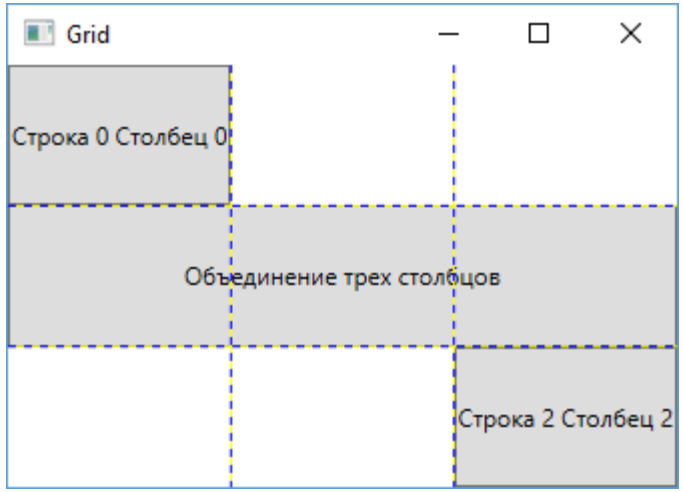
Чтобы задать позицию элемента управления с привязкой к определенной ячейке Grid, в разметке элемента нужно прописать значения свойств **Grid.Column** и **Grid.Row**, тем самым указывая, в каком столбце и строке будет находиться элемент. Кроме того, если мы хотим

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

```
<Window x:Class="LayoutApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:LayoutApp"
        mc:Ignorable="d"
        Title="Grid" Height="250" Width="350">
    <Grid ShowGridLines="True">
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Button Grid.Column="0" Grid.Row="0" Content="Строка 0 Столбец 0"  />
        <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

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

Абсолютные размеры

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

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

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

пиксели	px

дюймы	in
сантиметры	cm
точки	pt

Например,

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

Пропорциональные размеры.

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

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

Если строка или столбец имеет высоту, равную *, то данная строка или столбце 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:

```
<ColumnDefinition Width="*" />
<ColumnDefinition Width="0.5*" />
<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 коэфициент 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.

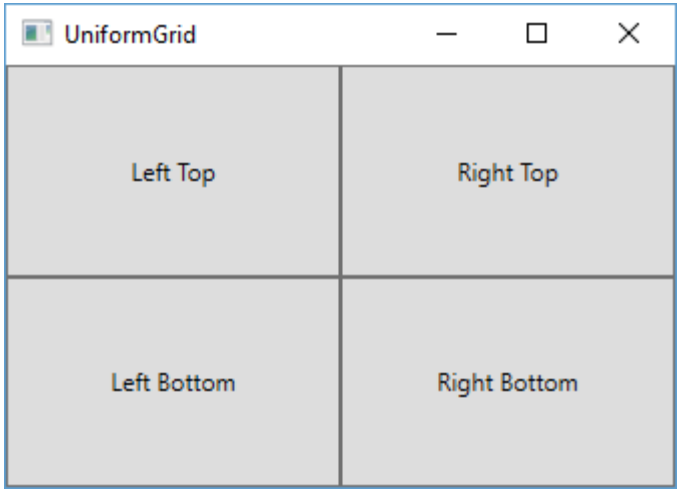
Можно комбинировать все типы размеров. В этом случае от ширины/высоты грида отнимается ширина/высота столбцов/строк с абсолютными или автоматическими размерами, и затем оставшееся место распределяется между столбцами/строками с

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:

```
<UniformGrid Rows="2" Columns="2">
    <Button Content="Left Top" />
    <Button Content="Right Top" />
    <Button Content="Left Bottom" />
    <Button Content="Right Bottom" />
</UniformGrid>
```



Назад Содержание Вперед