МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Методические указания к лабораторной работе №2

«Диспетчеры компоновки»

по дисциплине «Технологии разработки пользовательских интерфейсов»

Вичугов В.Н., доцент каф. АиКС

Томск 2012

Окно WPF-приложения обычно представлено корневым элементом Window. Дочерним элементом корневого элемента является диспетчер компоновки, который в свою очередь содержит любое количество элементов (в том числе, вложенных диспетчеров компоновки), определяющих пользовательский интерфейс. Диспетчер компоновки является объектом класса, унаследованного от абстрактного класса System.Windows.Controls.Panel.

Основные панели (диспетчеры компоновки, контейнерные элементы управления) WPF:

Canvas Элементы остаются в точности там, где были размещены во время проектирования

DockPanel Привязывает содержимое к определенной стороне панели (Top (верхняя), Bottom (нижняя), Left (левая) или Right (правая))

Grid Располагает содержимое внутри серии ячеек, расположенных в табличной сетке

StackPanel Выводит содержимое по вертикали или горизонтали, в зависимости от значения свойства Orientation

WrapPanel Позиционирует содержимое слева направо, перенося на следующую строку по достижении границы панели. Последовательность размещения происходит сначала сверху вниз или сначала слева направо, в зависимости от значения свойства Orientation

Диспетчер компоновки Canvas

Панель Canvas поддерживает абсолютное позиционирование содержимого пользовательского интерфейса. Если пользователь изменяет размер окна, делая его меньше, чем компоновка, обслуживаемая панелью Canvas, ее внутреннее содержимое становится невидимым до тех пор, пока контейнер вновь не увеличится до размера, равного или больше начального размера области Canvas.

Панель Canvas обладает следующим недостатком: элементы внутри Canvas не изменяются динамически при применении стилей или шаблонов.

Рассмотрим панель Canvas со следующим содержимым:

```
<Canvas>
        <Label Canvas.Left="10" Canvas.Top="10" Content="Регистрация пользователя" />
        <Label Canvas.Left="10" Canvas.Top="40" Content="ФИО" />
        <TextBox Canvas.Left="70" Canvas.Top="40" Width="200" />
        <Label Canvas.Left="10" Canvas.Top="70" Content="Email" />
        <TextBox Canvas.Left="70" Canvas.Top="70" Width="200" />
        <Button Canvas.Left="50" Canvas.Top="100" Content="Зарегистрироваться" />
</Canvas>
```

У элементов управления Label, TextBox, Button отсутствуют атрибуты Left и Top, поэтому для определения положения элементов на панели используется синтаксис присоединяемых свойств.

Присоединяемые свойства XAML (attached properties)

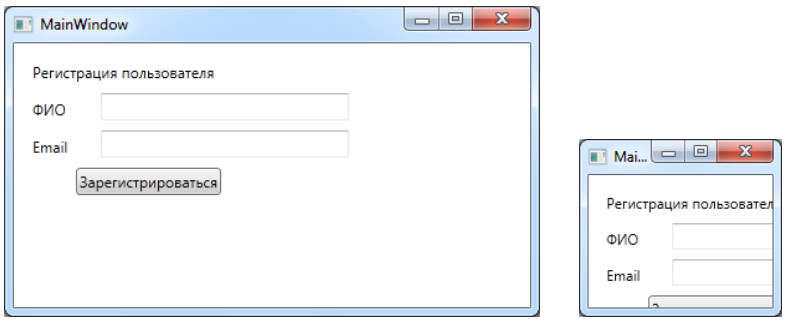
В XAML поддерживается специальный синтаксис, используемый для определения значения присоединяемого свойства. Присоединяемые свойства позволяют дочернему элементу устанавливать значение какого-то свойства, которое в действительности определено в родительском элементе. Общий шаблон:

```
<РодительскийЭлемент>
    <ДочернийЭлемент РодительскийЭлемент.СвойствоРодительскогоЭлемента = "Значение">
</РодительскийЭлемент>
```

С помощью присоединяемых свойств можно определить значения лишь ограниченного набора свойств родительских элементов, которые определены специальным образом в классе родительского элемента.

Дополнительное задание: определите, каким образом присоединяемые свойства объявляются в классе родительского элемента.

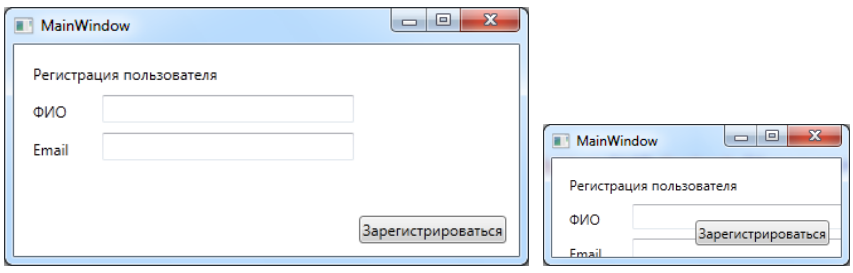
Пример работы приложения:



Для дочернего элемента необходимо указать привязку по вертикали (Canvas.Top или Canvas.Bottom) и привязку по горизонтали (Canvas.Left или Canvas.Right). Также можно (не обязательно) задать ширину и высоту элемента с помощью атрибутов Width и Height.

Таким образом, положение дочернего элемента управления можно задать относительно любого угла окна. Например, в следующем примере положение кнопки задано относительно нижнего правого угла окна:

```
<Canvas>
...
        <Button Canvas.Right="10" Canvas.Bottom="10" Content="Зарегистрироваться" />
</Canvas>
```



Порядок объявления дочерних элементов управления определяет порядок их вывода на экран. В приведенном выше примере кнопка выводится перед текстовыми полями, т.к. она была объявлена в файле XAML последней.

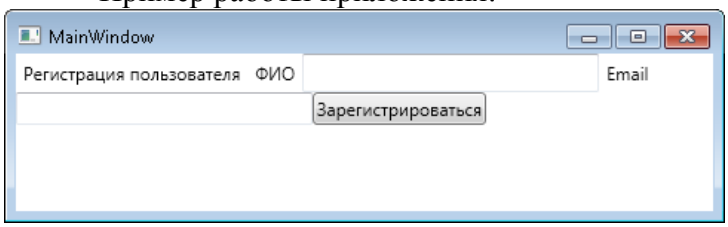
Диспетчер компоновки WrapPanel

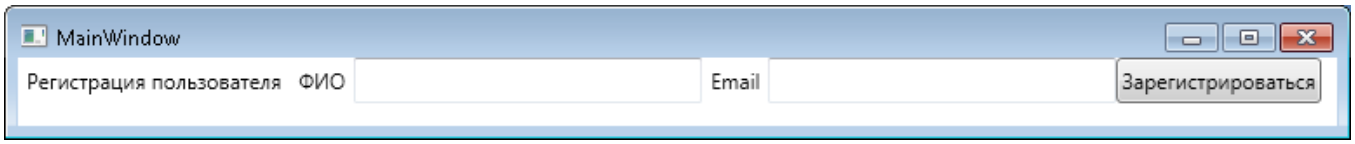
Панель WrapPanel выводит дочерние элементы последовательно слева направо (либо сверху вниз, если для атрибута Orientation установлено значение “Vertical”) и при достижении границы окна переходит на новую строку (столбец). При изменении размеров окна панель перераспределяет компоненты таким образом, чтобы они находились в окне.

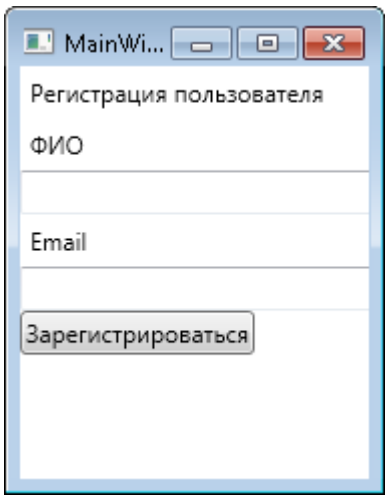
Рассмотрим панель WrapPanel со следующим содержимым:

```
<WrapPanel>
        <Label Content="Регистрация пользователя" />
        <Label Content="ФИО" />
        <TextBox Width="200" />
        <Label Content="Email" />
        <TextBox Width="200" />
        <Button Content="Зарегистрироваться" />
</WrapPanel>
```

Пример работы приложения:







Диспетчер компоновки StackPanel

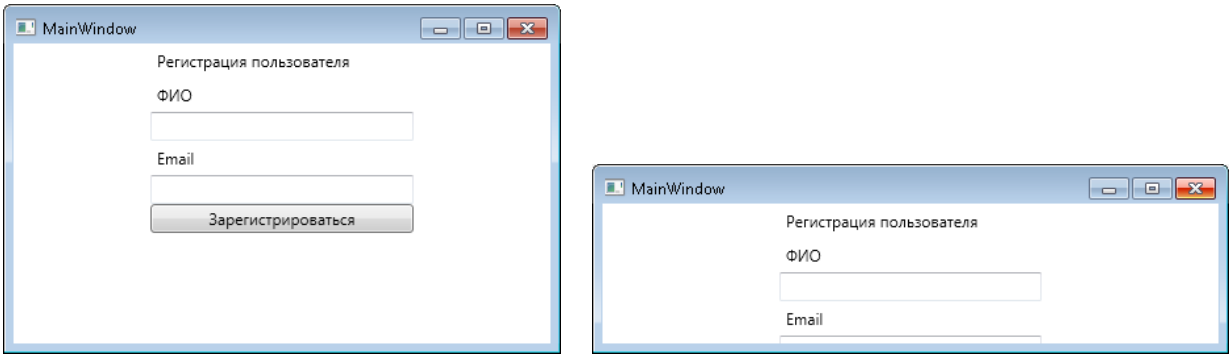
Панель StackPanel располагает содержащиеся в нем элементы управления либо в вертикальном столбце (по умолчанию), либо в горизонтальной строке (если в атрибут Orientation записано значение “Vertical”). Если в панель StackPanel добавлено больше элементов управления, чем может быть отображено по ширине/высоте StackPanel, лишние элементы обрезаются и не отображаются.

При выводе элементов сверху вниз элементы по умолчанию растягиваются по горизонтали. Это поведение можно изменить с помощью свойств HorizontalAlignment и VerticalAlignment.

Рассмотрим панель StackPanel со следующим содержимым:

```
<StackPanel HorizontalAlignment="Center">
        <Label Content="Регистрация пользователя" />
        <Label Content="ФИО" />
        <TextBox Width="200" />
        <Label Content="Email" />
        <TextBox Width="200" />
        <Button Content="Зарегистрироваться" />
 </StackPanel>
```

Пример работы приложения:



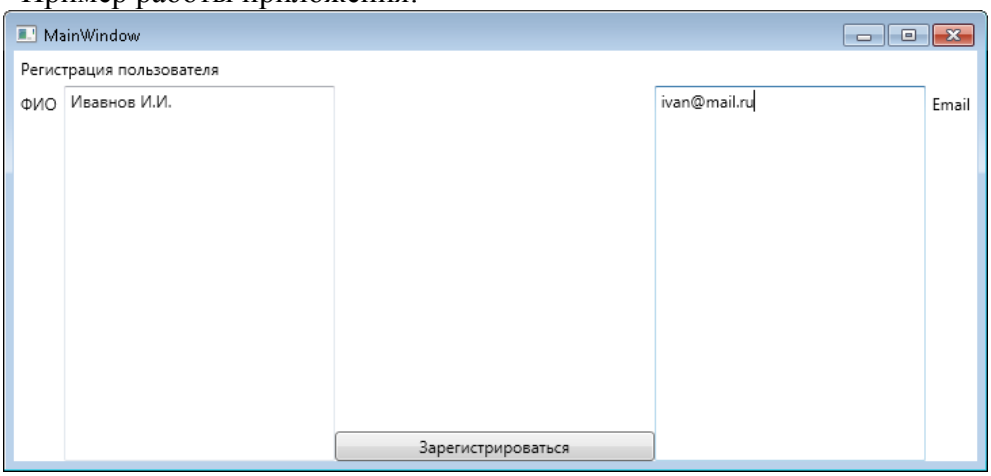
Диспетчер компоновки DockPanel

Панель DockPanel пристыковывает дочерние элементы к различным сторонам панели: Top, Bottom, Left, Right. Атрибут LastChildFill по умолчанию имеет значение True, что означает, что последний дочерний элемент управления будет занимать всё оставшееся пространство панели.

Рассмотрим панель DockPanel со следующим содержимым:

```
<DockPanel LastChildFill="False">
        <Label DockPanel.Dock="Top" Content="Регистрация пользователя" />
        <Label DockPanel.Dock="Left" Content="ФИО" />
        <TextBox DockPanel.Dock="Left" Width="200" />
        <Label DockPanel.Dock="Right" Content="Email" />
        <TextBox DockPanel.Dock="Right" Width="200" />
        <Button DockPanel.Dock="Bottom" Content="Зарегистрироваться" />
</DockPanel>
```

Пример работы приложения:



Диспетчер компоновки Grid

Подобно HTML-таблице, панель Grid может состоять из набора ячеек, каждая из которых имеет свое содержимое. При определении панели Grid выполняются следующие шаги:

1. Определение и конфигурирование каждого столбца.
2. Определение и конфигурирование каждой строки.
3. Назначение содержимого каждой ячейке сетки с использованием синтаксиса присоединяемых свойств.

Если не определить никаких строк и столбцов, то по умолчанию панель Grid будет состоять из одной ячейки, занимающей всю поверхность окна. Кроме того, если не указать ячейку для дочернего элемента, то он разместится в столбце 0 и строке 0.

Определение столбцов и строк выполняются за счет использования элементов <Grid.ColumnDefinitions> и <Grid.RowDefinitions>, которые содержат коллекции элементов <ColumnDefinition> и <RowDefinition>, соответственно.

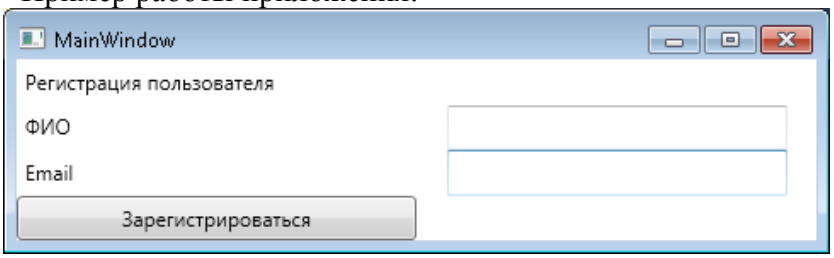
Каждый дочерний элемент прикрепляется к ячейке сетки, используя присоединяемые свойства Grid.Row и Grid.Column. Левая верхняя ячейка определяется с помощью Grid.Column="0" и Grid.Row="0".

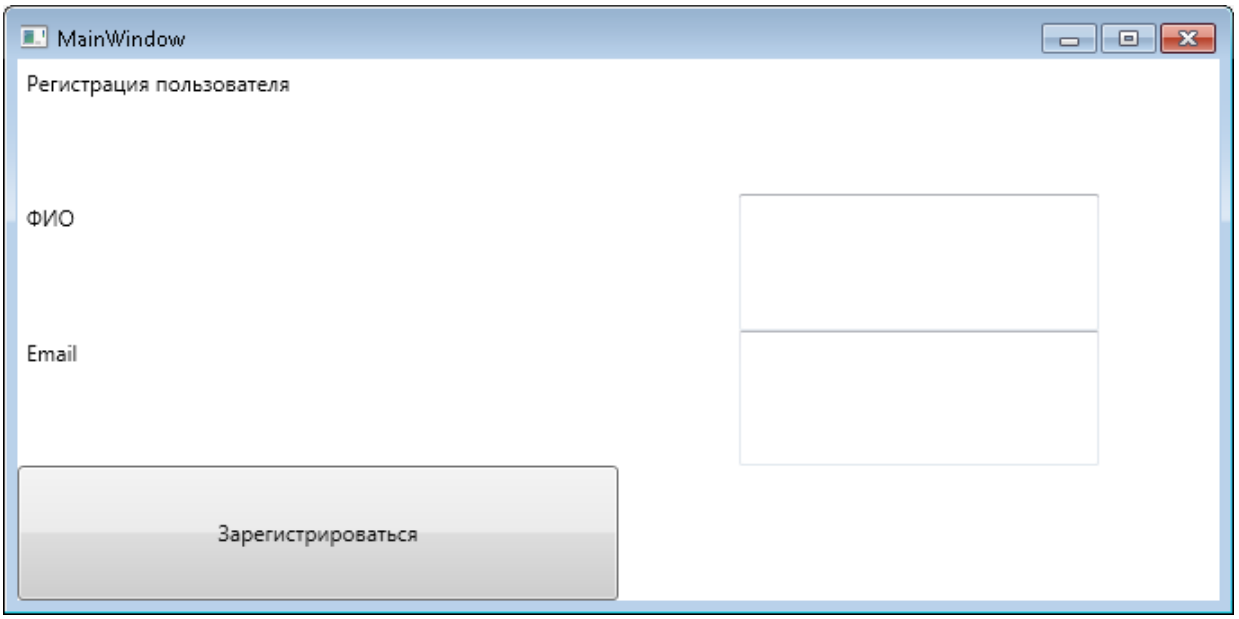
Рассмотрим панель Grid со следующим содержимым:

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

```
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Label Grid.Row="0" Grid.Column="0" Content="Регистрация пользователя" />
        <Label Grid.Row="1" Grid.Column="0" Content="ФИО" />
        <TextBox Grid.Row="1" Grid.Column="1" Width="200" />
        <Label Grid.Row="2" Grid.Column="0" Content="Email" />
        <TextBox Grid.Row="2" Grid.Column="1" Width="200" />
        <Button Grid.Row="3" Grid.Column="0" Content="Зарегистрироваться" />
 </Grid>
```

Пример работы приложения:



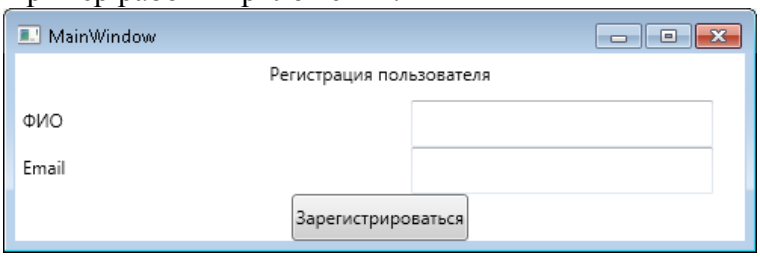


Объединение ячеек осуществляется с помощью присоединяемых свойств Grid.ColumnSpan и Grid.RowSpan аналогично объединению ячеек в HTML-таблицах.

Рассмотрим панель Grid со следующим содержимым:

```
<Grid>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Label Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2" HorizontalAlignment="Center"
Content="Регистрация пользователя" />
        <Label Grid.Row="1" Grid.Column="0" Content="ФИО" />
        <TextBox Grid.Row="1" Grid.Column="1" Width="200" />
        <Label Grid.Row="2" Grid.Column="0" Content="Email" />
        <TextBox Grid.Row="2" Grid.Column="1" Width="200" />
        <Button Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="2" HorizontalAlignment="Center"
Content="Зарегистрироваться" />
 </Grid>
```

Пример работы приложения:



При определении ряда можно задать его высоту с помощью атрибута Height, а при определении столбца можно задать его ширину с помощью атрибута Width. Значение этих атрибутов может быть следующим:

- "Auto" - высота строчки (или ширина колонки) определяется её содержимым;
- "Число" - высота строчки (или ширина колонки) равна указанному числу точек;
- "*" - высота строчки (или ширина колонки) занимает всё свободное пространство. Если строчек (колонок) с таким значением атрибута несколько, то свободное пространство перераспределяется между ними.

Задание

Разработать приложение WPF со следующим графическим интерфейсом: