# Отображение документов нефиксированного формата

**Категории документов в WPF** :

https://msdn.microsoft.com/ru-ru/library/ms748388(v=vs.110).aspx

https://professorweb.ru/my/WPF/documents_WPF/level28/28_1.php

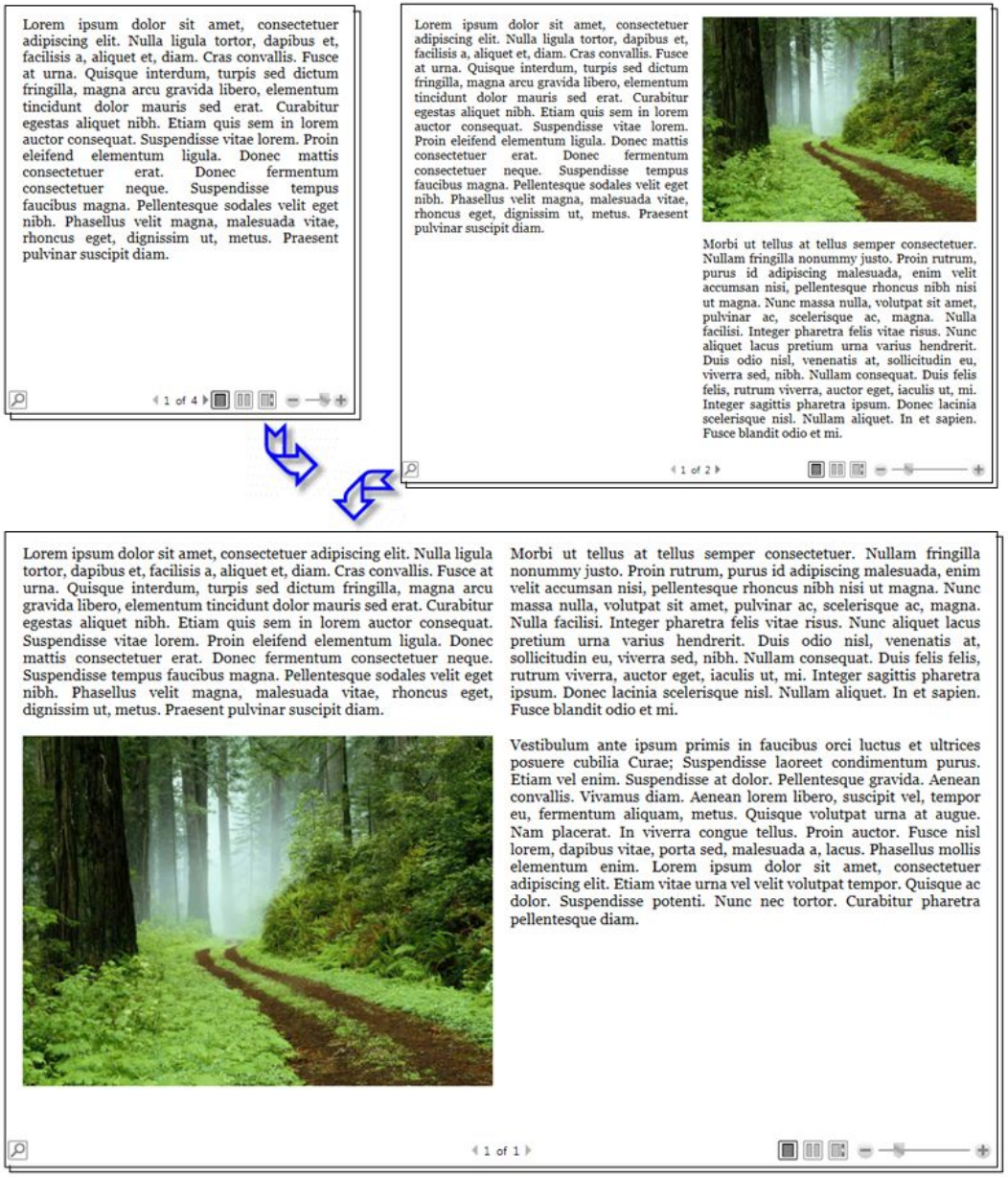
- Документы фиксированного формата (fixed documents)
- Документы нефиксированного формата (Потоковые документы flow documents)

Документы фиксированного формата предназначены для приложений, требующих точного представления в режиме "what you see is what you get" (**WYSIWYG**), независимо от используемого дисплея или принтера. Обычно используются при подготовке публикаций, обработке текста и разметке формы, где строгое соблюдение исходного дизайна страницы является обязательным.

Документы нефиксированного формата предназначены для оптимизации просмотра и удобочитаемости и наиболее удобны в использовании, когда простота чтения является основным требованием.

Документы нефиксированного формата не зависят от предварительно определенного макета, а динамически изменяют и переформатируют свое содержимое на основе переменных времени выполнения (например, размера окна и разрешения устройства) и дополнительных пользовательских настроек. Документы нефиксированного формата предоставляют дополнительные возможности, например, разбиение на страницы и столбцы.

# Пример с MSDN



К фиксированным документам в WPF относятся документы на основе XPS.

**XPS** (англ. *XML Paper Specification*) — открытый графический формат фиксированной разметки на базе XML, разработанный компанией Microsoft.

XPS-файл представляет собой ZIP-архив с использованием Open Packaging Conventions. Внутри файла находится вся необходимая информация для отображения документа в неизменном виде.

Документы фиксированного формата, как часть макета, поддерживают точное *позиционирование* содержимого элементов, независимо от используемых устройств отображения или печати. Например, страница формата фиксированного документа, просматриваемая на экране с разрешением 96 точек на дюйм, будет отображаться точно так же на выводе лазерного принтера с разрешением 600 точек на дюйм или устройстве фотовывода с разрешением 4800 точек на дюйм. *Макет* страницы остается неизменным во всех случаях, в то время как качество документа повышается в соответствии с возможностями каждого устройства.

Отображение содержимого формата фиксированного документа поддерживается с помощью элемента управления **DocumentViewer**.

Элемент управления DocumentViewer предназначен для отображения содержимого в режиме ***только для чтения***; редактирование или изменение содержимого недоступно и не поддерживается.

В качестве содержимого DocumentViewer принимает элемент **FixedDocument**, который как раз и представляет фиксированный документ.

```
<DocumentViewer x:Name="documentViewer">
        <FixedDocument x:Name="fixDoc" >


        </FixedDocument>
</DocumentViewer>
```

Фиксированный документ FixedDocument может содержать различное количество страниц.

Каждая страница представляет элемент **PageContent**.

В этот элемент помещается объект **FixedPage**, в который в свою очередь помещаются другие элементы (текстовые поля, фигуры и др.)

```
<FixedDocument x:Name="fixDoc" >
                <PageContent>
                    <FixedPage> …
                    </FixedPage>
                </PageContent>
                <PageContent>
                    <FixedPage>  …
                    </FixedPage>
                </PageContent>
            </FixedDocument>
```

Например,

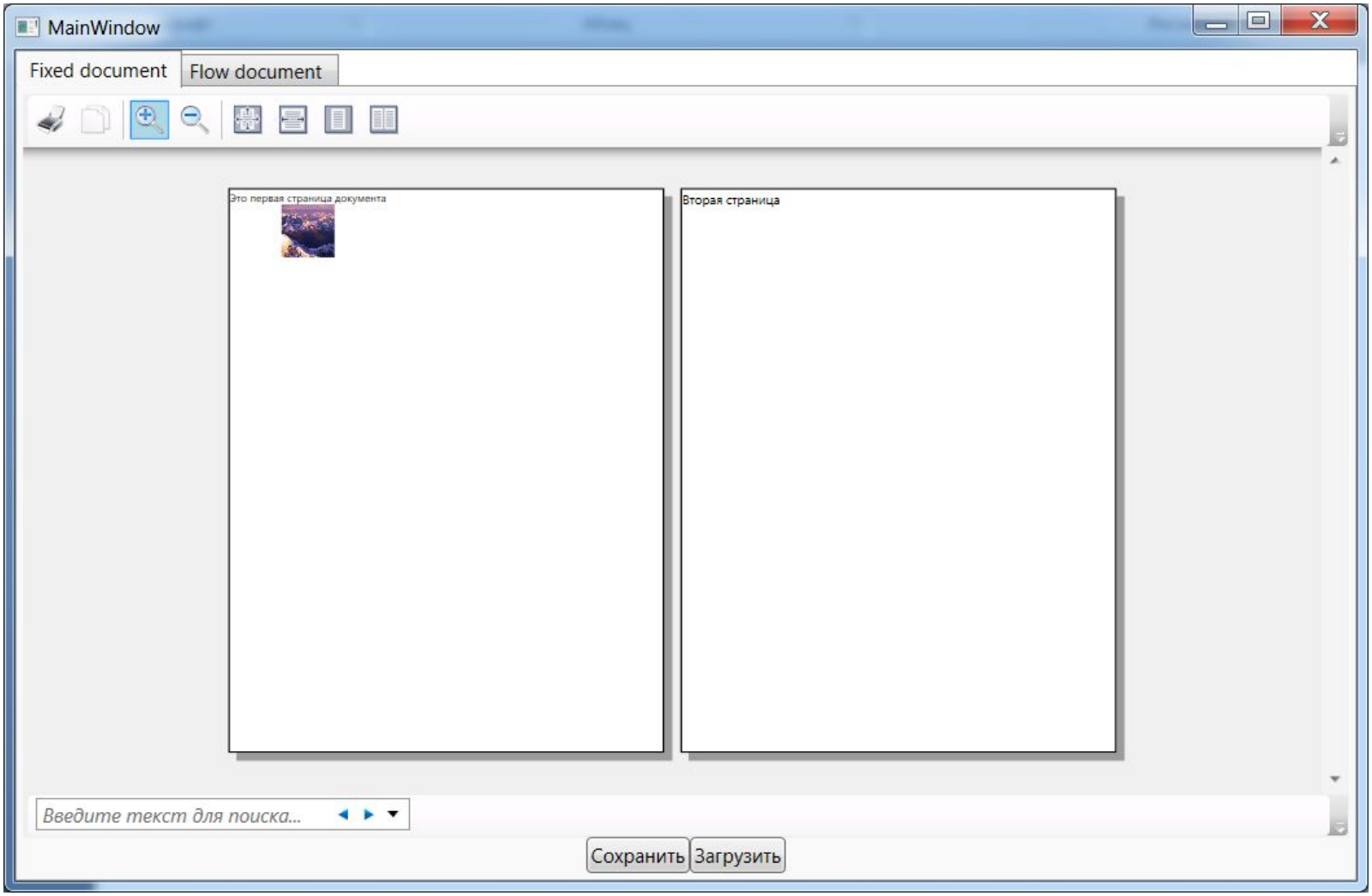
```
<FixedDocument x:Name="fixDoc" >
                    <PageContent>
                        <FixedPage>
                            <Grid Margin="10">
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="50"></RowDefinition>
                                    <RowDefinition   Height="*"></RowDefinition>
                                </Grid.RowDefinitions>
            <TextBlock Grid.Row="0" Text="Первая страница    Привет, Вася" />
            <Ellipse Grid.Row="1" Width="50" Height="50" Fill="Red" />
                            </Grid>
                        </FixedPage>
                    </PageContent>
                    <PageContent >
                        <FixedPage>
                            <StackPanel x:Name="stPanel" Margin="10" >
                                <TextBlock Text="Вторая страница" />
                            </StackPanel>
                        </FixedPage>
                    </PageContent>  </FixedDocument>
```

# Пример создания фиксированного документа в коде

```
FixedPage pg = new FixedPage();
            StackPanel panel = new StackPanel();
            panel.Children.Add(new TextBlock
            {
                Text = "Это первая страница документа",
                FontSize = 20

            });
```

```
ImageSourceConverter converter = new ImageSourceConverter();
 string path =
                                                    string.Format(@"{0}\{1}",
(System.IO.Path.GetDirectoryName(Assembly.GetEntryAssembly().Location))
, "foto.jpg");
            ImageSource imageSource =
 (ImageSource)converter.ConvertFromString(path);
            panel.Children.Add(new Image
{ Source =imageSource ,Height=100,Width=100,Stretch=Stretch.Fill});
            pg.Children.Add(panel);
            PageContent pc = new PageContent { Child = pg };
            fixDoc.Pages.Add(pc);
```

```
pg = new FixedPage();

            pg.Children.Add(new TextBlock
            {
                Text = "Вторая страница",
                FontSize = 24

            });
            pc = new PageContent { Child = pg };
            fixDoc.Pages.Add(pc);
```


MainWindow
Fixed document
Flow document
Это первая страница документа
Вторая страница
Введите текст для поиска...
Сохранить
Загрузить

Для сохранения и открытия документа применяется класс **XpsDocument**.

Для его использования нам надо добавить в проект библиотеки **ReachFramework.dll** и **System.Printing.dll**.

Например, открытие и отображение:

```
XpsDocument doc = new XpsDocument(ofd.FileName, FileAccess.Read);
documentViewer.Document = doc.GetFixedDocumentSequence();
```

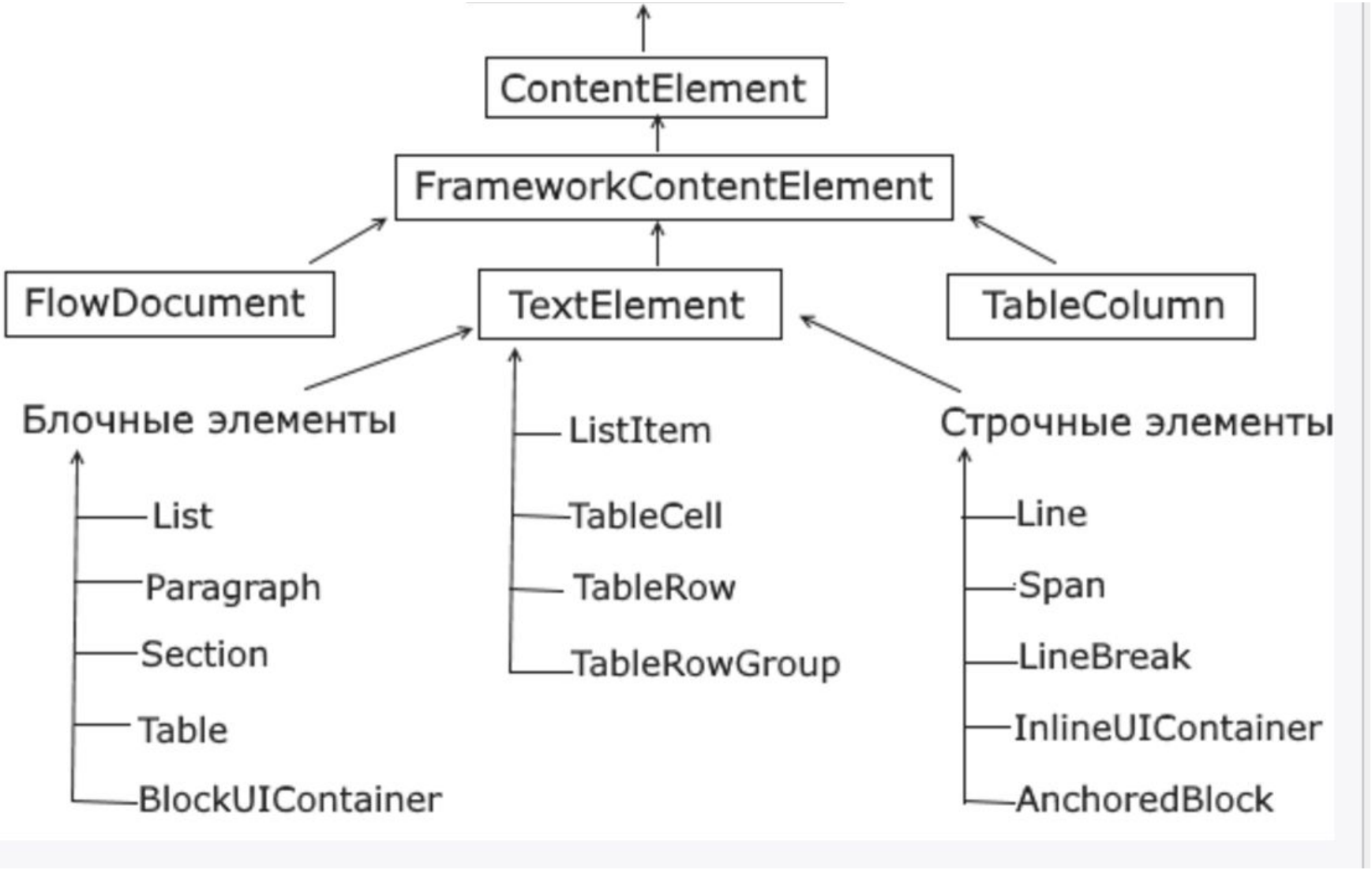
## Потоковые документы

Отображение содержимого документа нефиксированного формат поддерживается с помощью трех различных элементов управления **FlowDocumentReader**, **FlowDocumentPageViewer** и **FlowDocument ScrollViewer (можно еше RichTextBox)**

Потоковые документы в WPF представлены классом **FlowDocument**, который может включать в себя различные потоковые элементы (flow elements)

FlowDocument задает строгую модель содержимого для своего дочернего содержимого. Дочерние элементы верхнего уровня, содержащиеся в FlowDocument, должны быть производными от **Block** и называются ***блочными***.

Иерархию потоковых элементов можно представить следующим образом:


ContentElement
FrameworkContentElement
FlowDocument
TextElement
TableColumn
Блочные элементы
ListItem
Строчные элементы
List
TableCell
Line
Paragraph
TableRow
Span
Section
TableRowGroup
LineBreak
Table
InlineUIContainer
BlockUIContainer
AnchoredBlock

# Допустимые дочерние элементы верхнего уровня:

| Блок | Описание |
|---|---|
| `Paragraph` | Содержит текст, возможно форматированный. |
| `List` | Содержит списки разного типа (нумерованный, маркированный и т. д.). |
| `Table` | Содержит таблицы, похожие на таблицы Microsoft Word или HTML. |
| `BlockUIContainer` | Содержит различные элементы пользовательского интерфейса, становящиеся частью размещения. |
| `Section` | Содержит группу прочих блоков. Разделы удобны для применения общих атрибутов к группам блоков, например, одни и те же атрибуты шрифта для нескольких абзацев. |

Сначала будем использовать контейнер

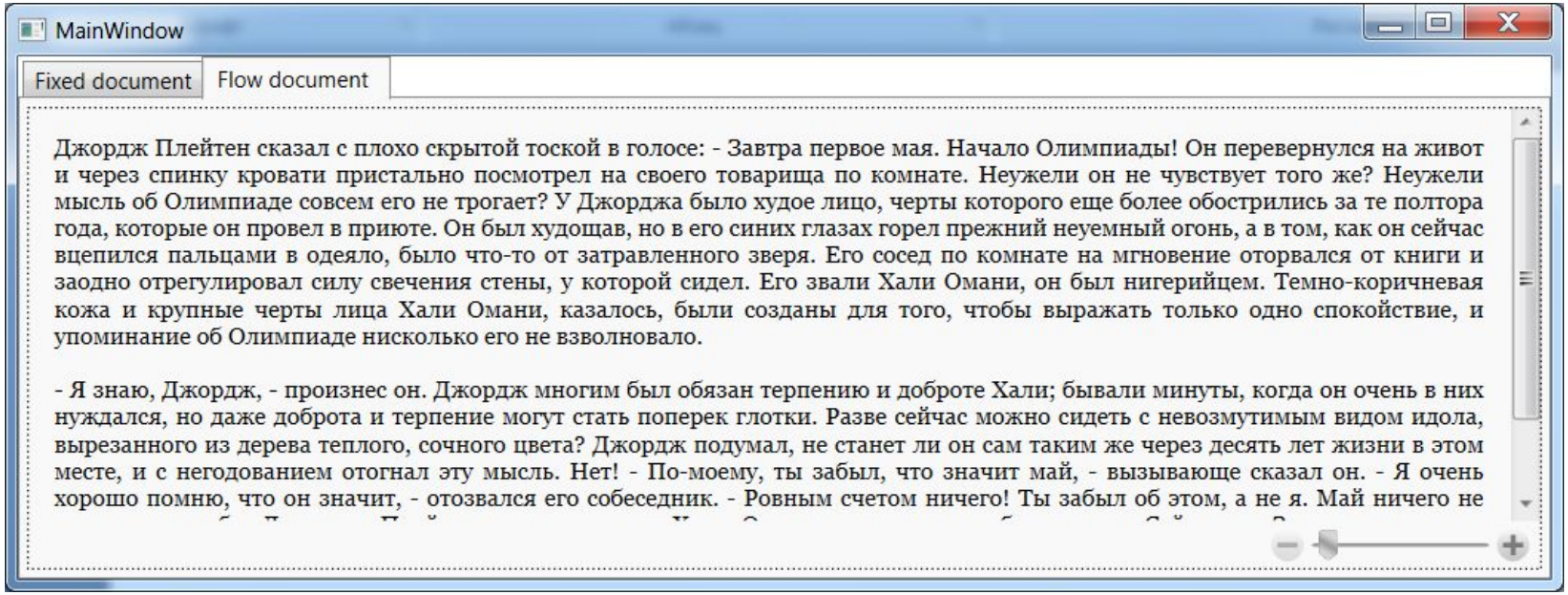
**FlowDocumentScrollViewer**

```
<FlowDocumentScrollViewer>
        <FlowDocument>
            <Paragraph>Привет, Вася!</Paragraph>
            <Paragraph >Ку-ку</Paragraph>
         </FlowDocument>
</FlowDocumentScrollViewer>
```

Данный элемент управления из всех подобных ему меньше всего влияет на производительность системы

У каждого из контейнеров есть свойство **IsToolBarVisible** – переключатель (тип **bool**) добавляет внизу контейнера просмотра маленький ползунок который и используется для увеличения или уменьшения содержимого. По умолчанию установлено значение false.

- **Установите для этого свойства значение true и добавьте в параграфы больше текста.**



***Свойства масштабирования***:

- MinZoom – Минимальное значение ползунка увеличения контента.
- MaxZoom – Максимально значение ползунка увеличения контента.
- Zoom – Текущее значение, обычно устанавливается значение 100. Это сохранит первоначальный размер шрифта.
- ZoomIncrement – Шаг изменения. На ползунке.

Добавление функциональности Zoom для контейнеров сильно бьет по производительности.

Все потоковые элементы являются наследниками класса **TextElement** и могут быть блочными (**block**) и строчными (**inline**).

## Элемент Paragraph

Элемент Paragraph содержит коллекцию **Inlines**, которая в свою очередь включает строковые элементы (не только текст). Чтобы параграф отображал текст, надо использовать строчный элемент **Run.**

```
<Paragraph>
    <Run>Привет, Вася!</Run>
</Paragraph>
```

Если не использовать **Run** и напрямую писать текст в содержимое параграфа, то элемент **Run** все равно будет создан, только неявно.

Чтобы получить *в коде* содержимое параграфа, надо получить текст элемента Run:

```
string s = ((Run)p1.Inlines.FirstInline).Text;
```

**Элемент List**

Представляет собой список. Он содержит коллекцию элементов **ListItem**, которые и представляют элементы списка.

Каждый из элементов **ListItem**, в свою очередь, может включать другие блочные элементы

С помощью свойства **MarkerStyle** можно задать формат списка:

Disk – Сплошной диск, это значение установлено по умолчанию

Box – Сплошной квадрат.

Circle – Окружность без заливки.

Square – Квадрат без заливки.

Decimal – Числа по порядку начинаются из свойства StatringIndex которое по умолчанию установлено на 1.

UpperLatin – Латинская буква в верхнем регистре.

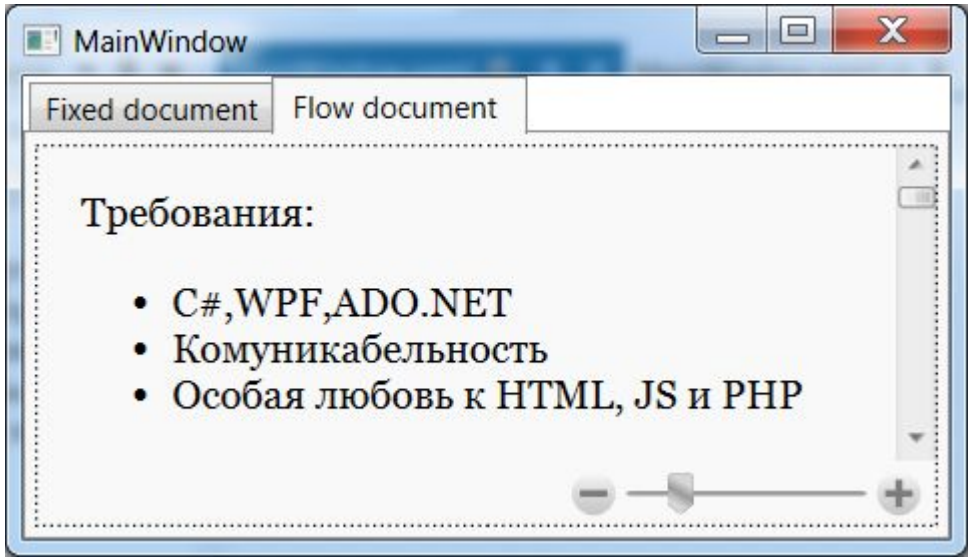
LowerLatin – Латинская буква в нижнем регистре.

UpperRoman – Римская маленькая цифра.

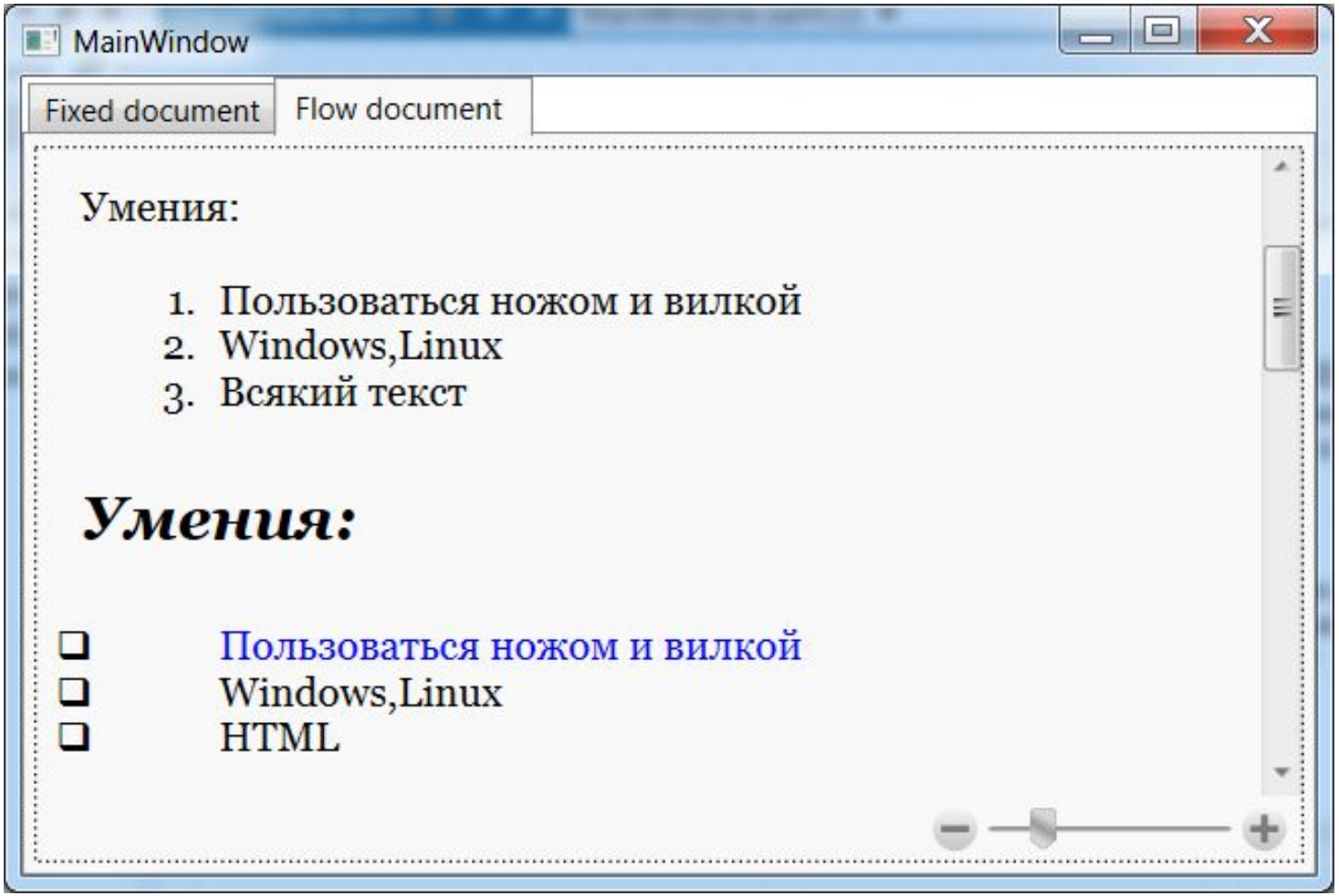
LowerRoman – римская большая цифра.

None – без маркера.

```
<Paragraph>Требования:</Paragraph>
<List>
     <ListItem>
        <Paragraph>C#,WPF,ADO.NET</Paragraph>
     </ListItem>
     <ListItem>
         <Paragraph>Комуникабельность</Paragraph>
      </ListItem>
      <ListItem>
         <Paragraph>Особая любовь к HTML, JS и PHP</Paragraph>
      </ListItem>
</List>
```



# Создайте списки



```
<Paragraph>Умения:</Paragraph>
        <List MarkerStyle="Decimal">
            <ListItem>
          <Paragraph>Пользоваться ножом и вилкой</Paragraph>
            </ListItem>
         <ListItem>
            <Paragraph>Windows,Linux</Paragraph>
        </ListItem>
        <ListItem>
            <Paragraph>Всякий текст</Paragraph>
        </ListItem>
    </List>
```

```
<Paragraph FontSize="24"
           FontStyle="Italic"
           FontWeight="Bold">Умения:</Paragraph>
     <List MarkerStyle="Square" MarkerOffset="50">
     <ListItem>
        <Paragraph
     Foreground="Blue">Пользоваться ножом и вилкой</Paragraph>
     </ListItem>
      <ListItem>
             <Paragraph>Windows,Linux</Paragraph>
      </ListItem>
                      <ListItem>
                          <Paragraph>HTML</Paragraph>
                      </ListItem>
                  </List>
```

Аналог **<BR\>** в WPF – тег :

**<LineBreak></LineBreak>**

Он используется в параграфе, а не в Run

Wpf поддерживает полезную функцию автоматического выравнивания текста.

Для этого еужно задать свойство

```
<FlowDocument IsOptimalParagraphEnabled="True">
```

***Элемент Table***

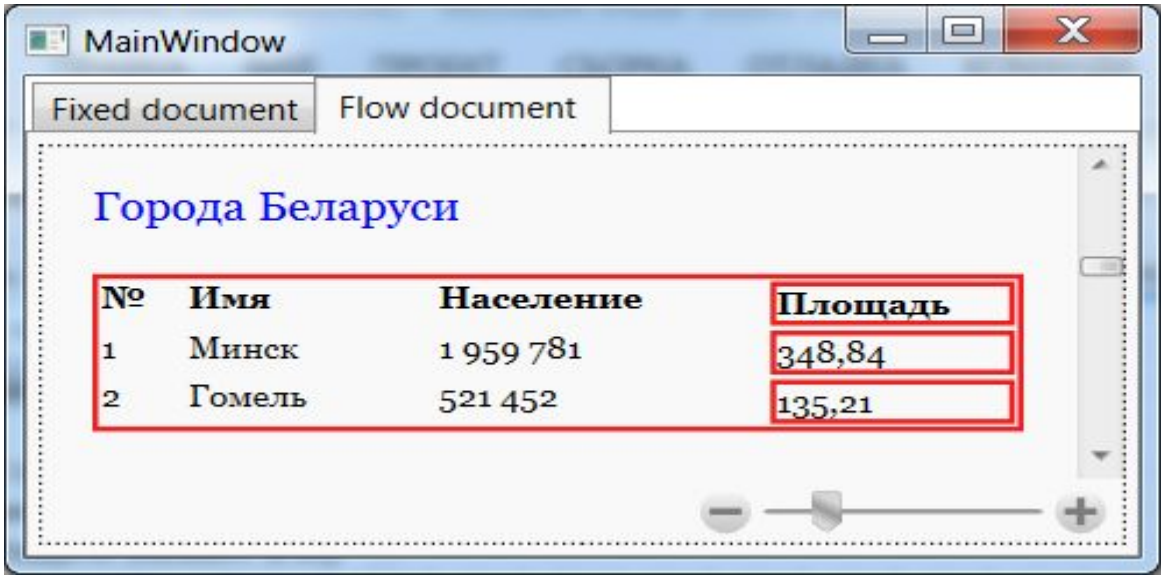
Организует вывод содержимого в виде таблицы.

Он имеет вложенный элемент **TableRowGroup**. Этот элемент позволяет задать однообразный вид таблицы и содержит коллекцию строк - элементов **TableRow** (строку таблицы).

А каждый элемент TableRow содержит несколько элементов **TableCell** (ячейка таблицы).

Если не указать явным образом ширину столбцов, WPF равномерно распределит пространство между всеми столбцами. Это поведение можно изменить, присвоив свойству Table.Rows набор объектов TableColumn и определив для каждого из них ширину Width

Создайте в документе такую таблицу

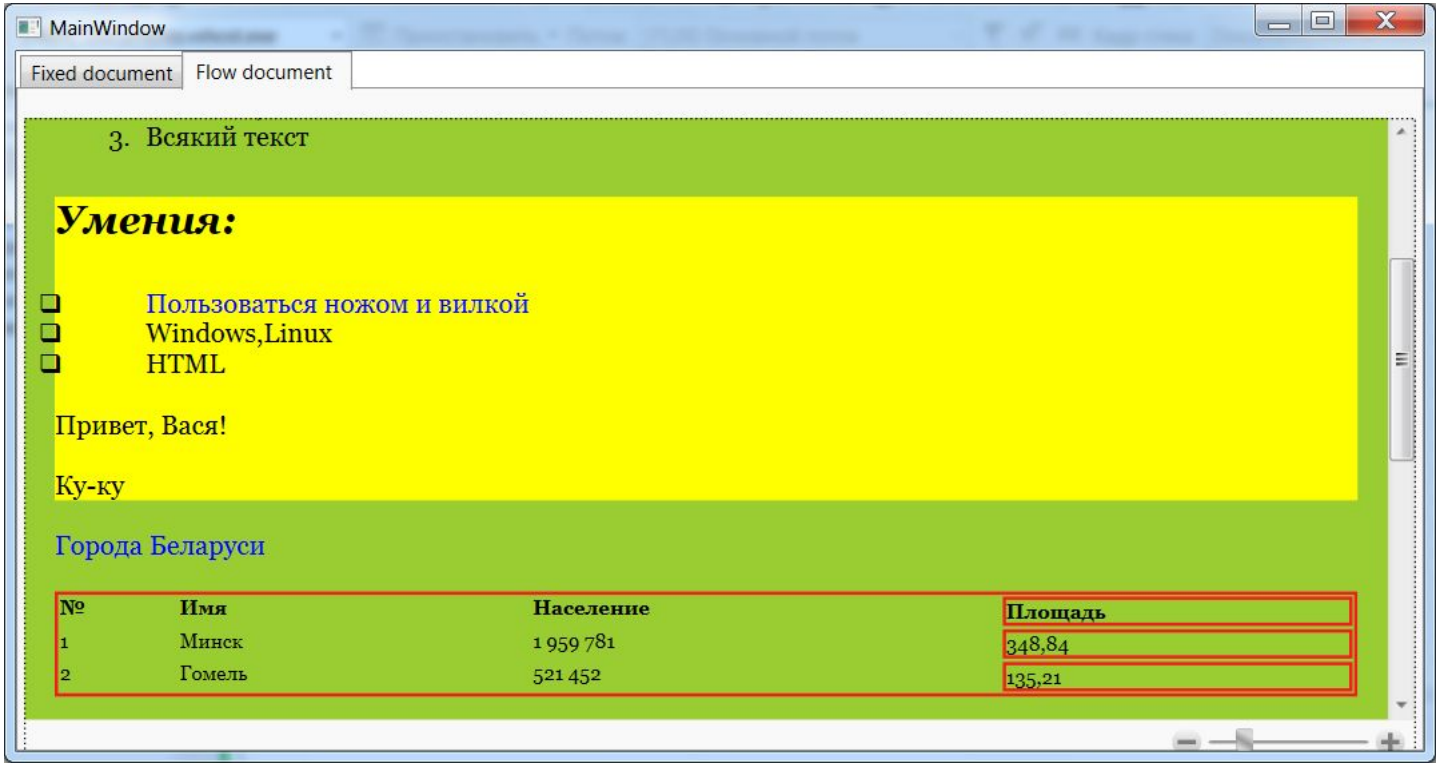



## Элемент Section

Предназначен для группировки других блочных элементов и прежде всего для однообразной стилизации этих элементов

Блок Section напоминает **Div** из HTML, задав ему свойство , можно задать это же свойство всем дочерним элементам.

Добейтесь в своем документе такого эффекта:

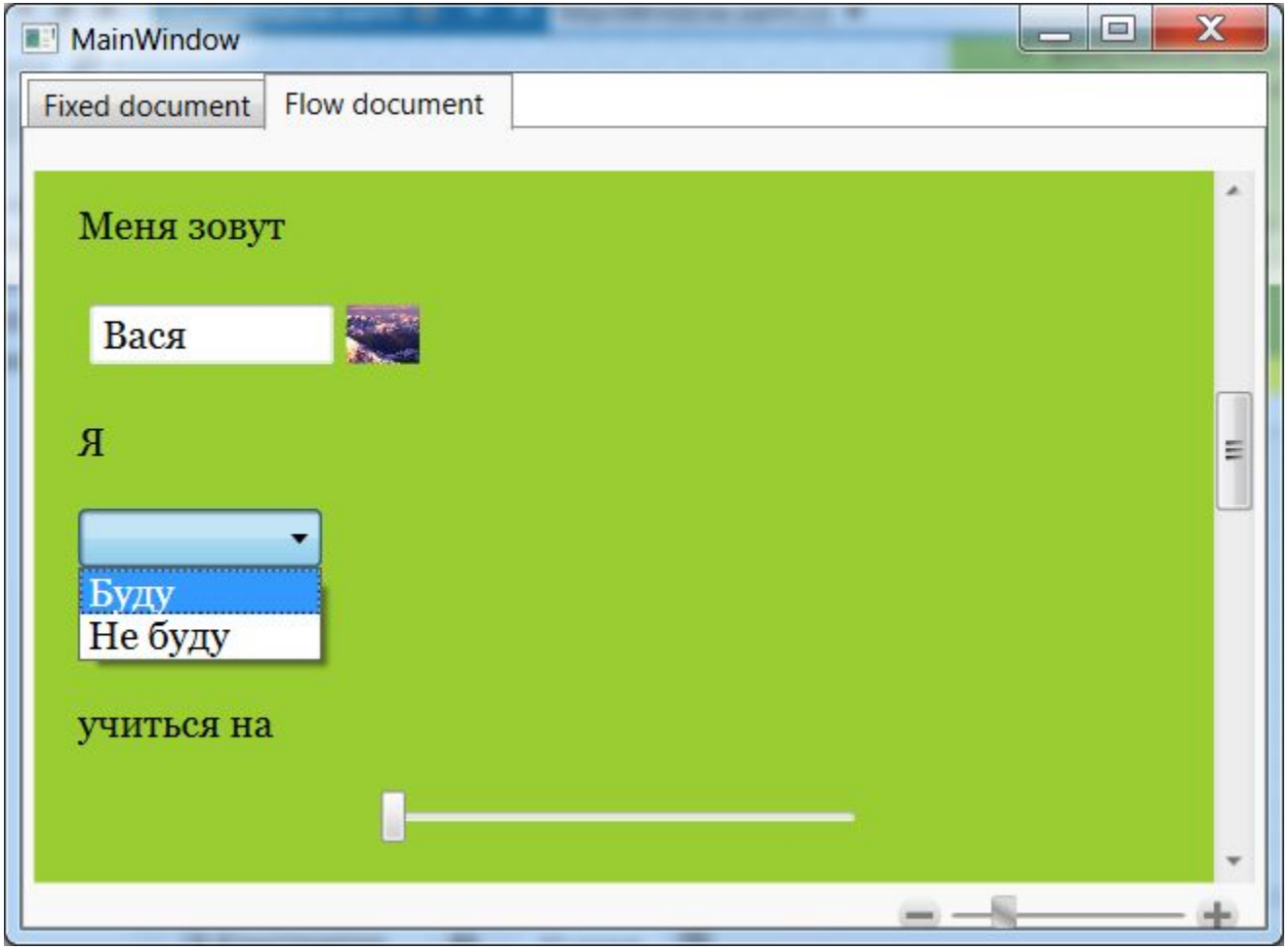


## Элемент BlockUIContainer

Позволяет добавить в документ различные элементы управления, которые не являются блочными или строчными элементами. Например, можно добавить кнопку или картинку к документу.

```
<BlockUIContainer FontSize="13">
        <StackPanel Orientation="Vertical">
            <TextBlock Height="40" Padding="10">
               Click the Button to see TIOBE Rate
           </TextBlock>
            <Button Width="60">Click Me</Button>
        </StackPanel>
    </BlockUIContainer>
```

Добавьте в документ элементы управления:



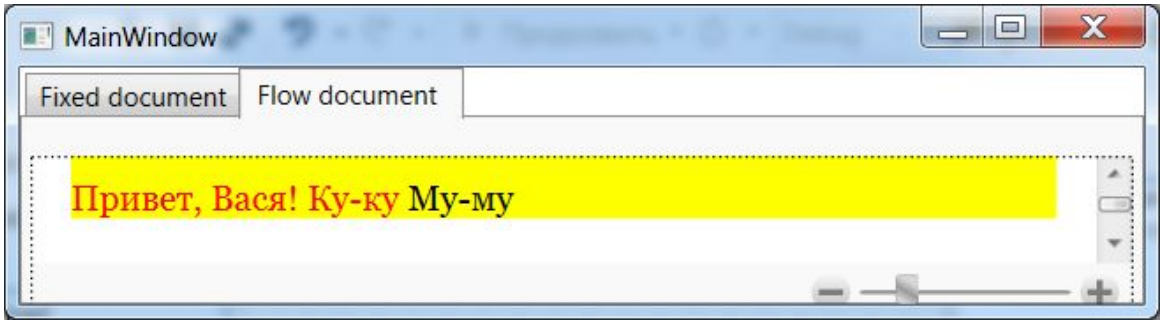
## Строчные элементы

**Run** - хранит некоторый текст, выводимый в блочном элементе

### Span

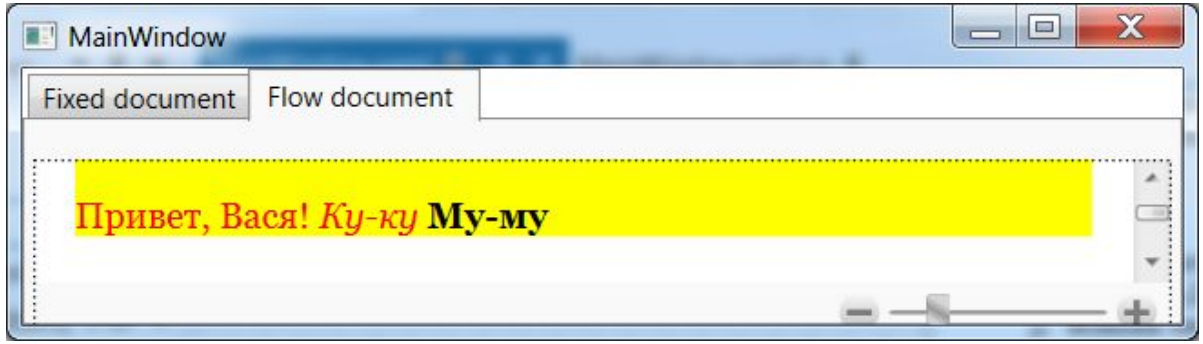
Объединяет другие строчные элементы и применяет к ним определенное форматирование

```
<Paragraph x:Name="p1">
    <Span Foreground="Red">
        <Run>Привет, Вася!</Run>
        <Run>Ку-ку</Run>
    </Span>
    <Run>Му-му</Run>
</Paragraph>
```



Чтобы задать для текста отдельные способы форматирования, применяются элементы **Bold**, **Italic** и **Underline**, которые позволяют создать текст полужирным шрифтом, курсивом и подчеркнутый текст соответственно.

```
<Paragraph x:Name="p1">
 <Span Foreground="Red">
   <Run>Привет, Вася!</Run>
     <Italic>
         <Run>Ку-ку</Run>
     </Italic>
 </Span>
 <Bold>
    <Run>Му-му</Run>
  </Bold>
</Paragraph>
```



Элемент **Hyperlink** позволяет вставить в документ ссылку для перехода

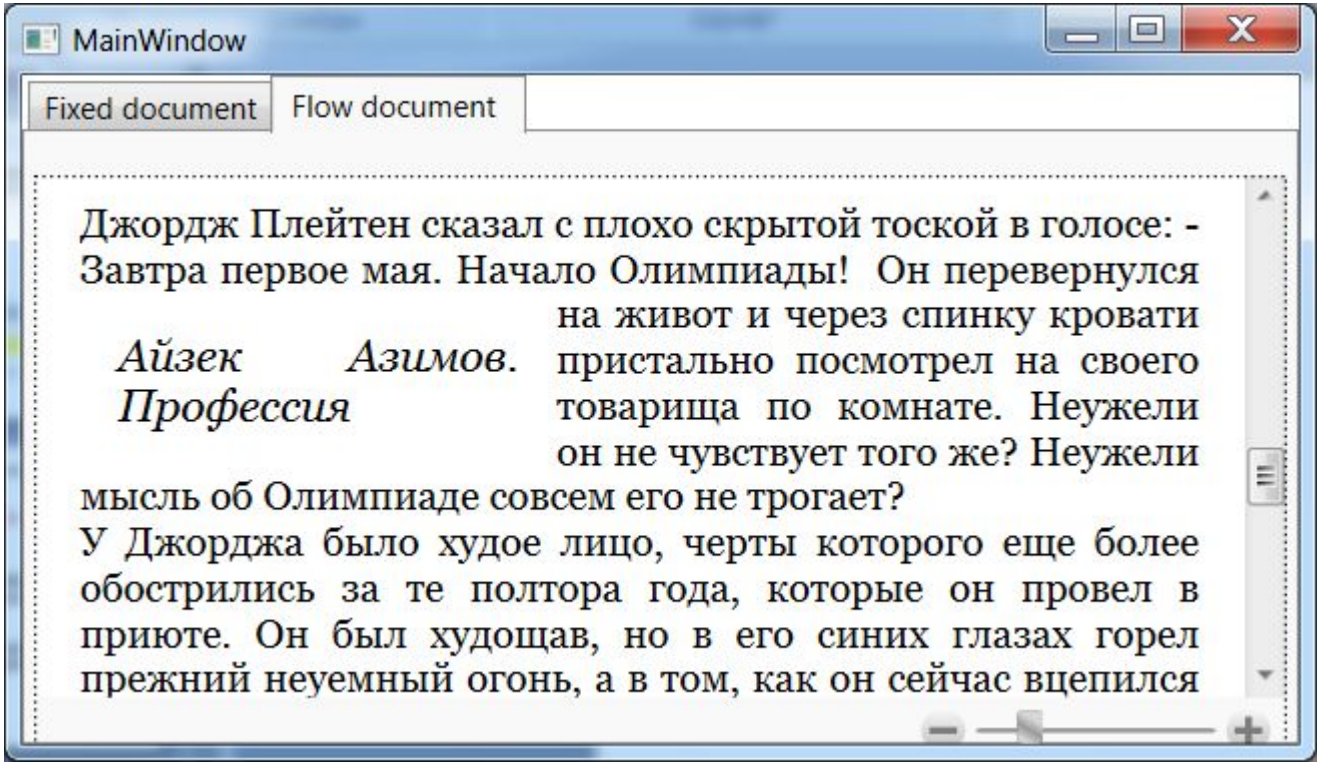
```
<Hyperlink NavigateUri="http:\\microsoft.com">
        Microsoft
</Hyperlink>
```

**InlineUIContainer** подобен **BlockUIContainer** (позволяет помещать другие элементы управления), только является строковым.

Элементы **Floater** и **Figure** позволяют вывести плавающее окно с некоторой информацией, текстом, картинками и прочим:

```
<Paragraph>Джордж Плейтен сказал с плохо скрытой тоской в голосе:
     - Завтра первое мая. Начало Олимпиады!
 <Floater Width="170“
          Padding="5"
          HorizontalAlignment="Left"
          FontSize="18" FontStyle="Italic">
      <Paragraph> Айзек Азимов. Профессия</Paragraph>
 </Floater>
```

Он перевернулся на живот и через спинку кровати пристально посмотрел на своего товарища по комнате. Неужели он не чувствует того же? Неужели мысль об Олимпиаде совсем его не трогает?...



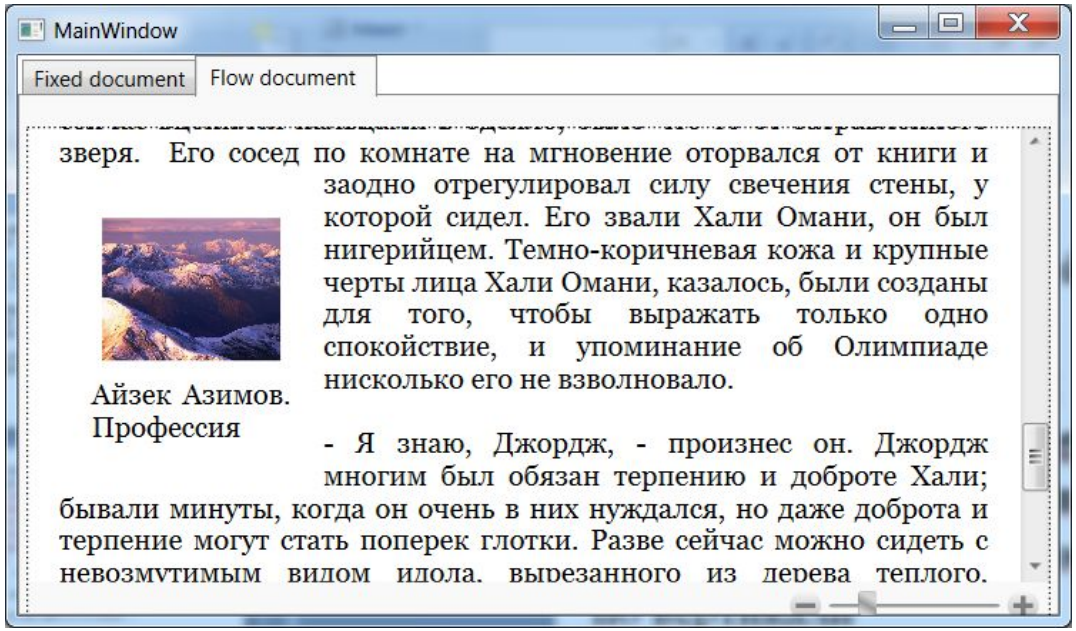

Остальной текст будет обтекать элемент Floater, заданный таким образом, справа.

По умолчанию плавающее окошко, используемое для элемента Floater, является невидимым. Но для него можно задать текстурированный фон (с помощью свойства Background) или рамку (с помощью свойств BorderBrush и BorderThickness). Можно также использовать свойство Margin, чтобы добавить промежуток между плавающим окошком и документом, и свойство Padding, чтобы добавить промежуток между краями окошка и его содержимым.

**Figure** аналогичен элементу **Floater**, но дает больше возможностей по контролю за позиционированием.

С помощью свойства **HorizontalAnchor** можно управлять позиционированием элемента по горизонтали. Так, значение **ContentLeft** позволяет выровнять текст по левой стороне, **ContentRight** - по правой стороне, а значение **ContentCenter** - по центру, свойство **VerticalAnchor** позволяет выровнять содержимое Figure по вертикали

Сделайте так, используя Floater, а потом Figure



**FlowDocumentPageViewer**

Разбивает документ на страницы в зависимости от размеров текста и размеров окна и отображает их для чтения. когда текст имеет слишком длинные строки, если сказать по другому то этот контейнер при больших размерах окна показывает сразу несколько страниц.

Данный элемент управления не позволяет отключать возможность Zoom

Для управления столбцами у элемента FlowDocument можно настроить следующие свойства:

**ColumnWidth**: устанавливает ширину столбца

**ColumnGap**: устанавливает расстояние между столбцами

**IsColumnWidthFlexible**: при значении True контейнер сам корректирует ширину столбца

**ColumnRuleWidth** и **ColumnRuleBrush**: устанавливает соответственно ширину границы между столбцаи и ее цвет

MainWindow

Fixed document | Flow document | FlowDocumentPageViewer

*Айзек Азимов. Профессия*

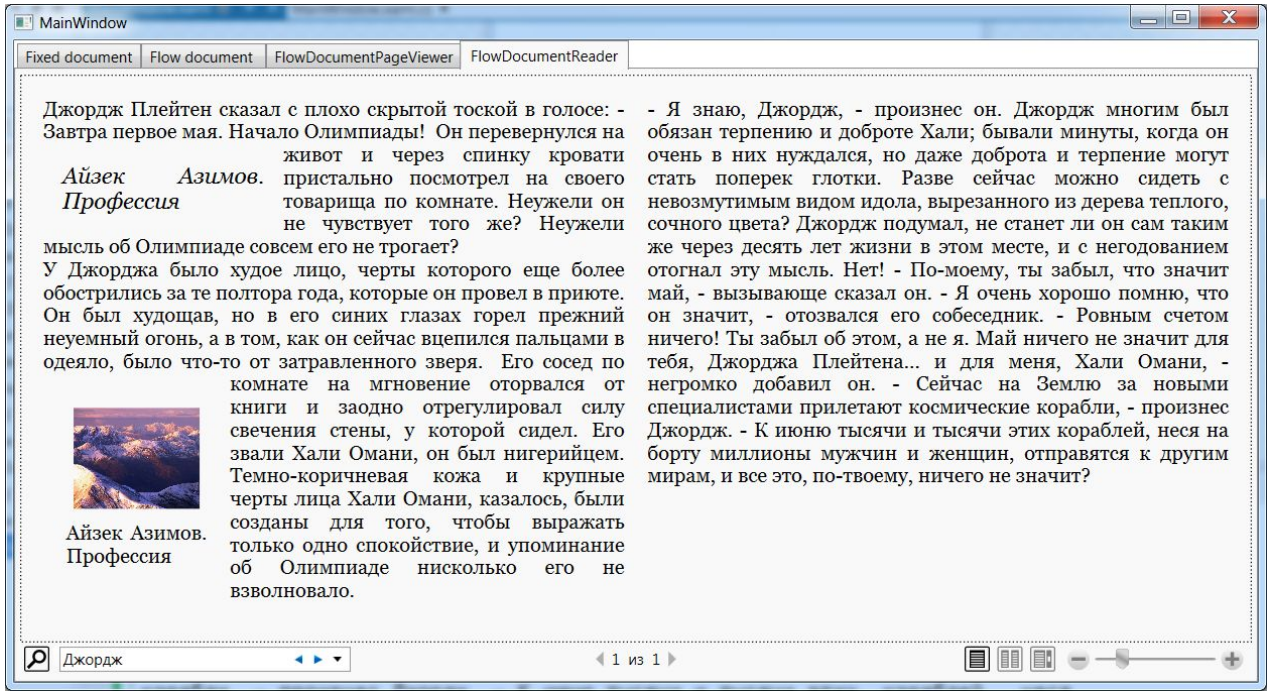
Джордж Плейтен сказал с плохо скрытой тоской в голосе: - Завтра первое мая. Начало Олимпиады!

Он перевернулся на живот и через спинку кровати пристально посмотрел на своего товарища по комнате. Неужели он не чувствует того же? Неужели мысль об Олимпиаде совсем его не трогает?

У Джорджа было худое лицо, черты которого еще более обострились за те полтора года, которые он провел в приюте. Он был худощав, но в его синих глазах горел прежний неуемный огонь, а в том, как он сейчас вцепился пальцами в одеяло, было что-то от затравленного зверя. Его сосед по комнате на мгновение оторвался от книги и заодно отрегулировал силу свечения стены, у которой сидел. Его звали Хали Омани, он был нигерийцем. Темно-коричневая кожа и крупные черты лица Хали Омани, казалось, были созданы для того, чтобы выражать только одно спокойствие, и упоминание об Олимпиаде нисколько его не взволновало.

Айзек Азимов.

- Я знаю, Джордж, - произнес он. Джордж многим был обязан терпению и доброте Хали; бывали минуты, когда он очень в них нуждался, но даже доброта и терпение могут стать поперек глотки. Разве сейчас можно сидеть с невозмутимым видом идола, вырезанного из дерева теплого, сочного цвета? Джордж подумал, не станет ли он сам таким же через десять лет жизни в этом месте, и с негодованием отогнал эту мысль. Нет! - По-моему, ты забыл, что значит май, - вызывающе сказал он. - Я очень хорошо помню, что он значит, - отозвался его собеседник. - Ровным счетом ничего! Ты забыл об этом, а не я. Май ничего не значит для тебя, Джорджа Плейтена... и для меня, Хали Омани, - негромко добавил он. - Сейчас на Землю за новыми специалистами прилетают космические корабли, - произнес Джордж. - К июню тысячи и тысячи этих кораблей, неся на борту миллионы

# FlowDocumentReader

Вобрал в себя функциональность и SckrollViewer и PageViewer и позволяет пользователю самому выбрать вариант просмотра. Данный контейнер отображения является очень «прожорливым» в плане ресурсов системы