## Текстовые элементы управления

### TextBlock

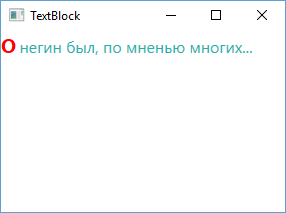
Элемент предназначен для вывода текстовой информации, для создания простых надписей:

|  |  |
| --- | --- |
| 1  2  3  4 | <StackPanel>      <TextBlock>Текст1</TextBlock>      <TextBlock Text="Текст2" />  </StackPanel> |

Ключевым свойством здесь является свойство Text, которое задает текстовое содержимое. Причем в случае <TextBlock>Текст1</TextBlock> данное свойство задается неявно.

С помощью таких свойств, как FontFamily, TextDecorations и др., мы можем настроить отображение текста. Однако мы можем задать и более сложное форматирование, например:

|  |  |
| --- | --- |
| 1  2  3  4 | <TextBlock TextWrapping="Wrap">      <Run FontSize="20" Foreground="Red" FontWeight="Bold">О</Run>      <Run FontSize="16" Foreground="LightSeaGreen">негин был, по мненью многих...</Run>  </TextBlock> |



Элементы Run представляют куски обычного текста, для которых можно задать отдельное форматирование.

Для изменения параметров отображаемого текста данный элемент имеет такие свойства, как **LineHeight**, **TextWrapping** и **TextAlignment**.

Свойство **LineHeight** позволяет указывать высоту строк.

Свойство **TextWrapping** позволяет переносить текст при установке этого свойства TextWrapping="Wrap". По умолчанию это свойство имеет значение NoWrap, поэтому текст не переносится.

Свойство **TextAlignment** выравнивает текст по центру (значение Center), правому (Right) или левому краю (Left): <TextBlock TextAlignment="Right">

Для декорации текста используется свойство **TextDecorations**, например, если TextDecorations="Underline", то текст будет подчеркнут.

Если нам вдруг потребуется перенести текст на другую строку, то тогда мы можем использовать элемент LineBreak:

|  |  |
| --- | --- |
| 1  2  3  4  5 | <TextBlock>      Однажды в студеную зимнюю пору      <LineBreak />      Я из лесу вышел  </TextBlock> |

### TextBox

Если TextBlock просто выводит статический текст, то этот элемент представляет поле для ввода текстовой информации.

Он также, как и TextBlock, имеет свойства **TextWrapping**, **TextAlignment** и **TextDecorations**.

С помощью свойства **MaxLength** можно задать предельное количество вводимых символов.

|  |  |
| --- | --- |
| 1 | <TextBox MaxLength="250" TextChanged="TextBox\_TextChanged">Начальный текст</TextBox> |

В коде C# мы можем обработать событие изменения текста:

|  |  |
| --- | --- |
| 1  2  3  4  5 | private void TextBox\_TextChanged(object sender, TextChangedEventArgs e)  {      TextBox textBox = (TextBox)sender;      MessageBox.Show(textBox.Text);  } |

По умолчанию, если вводимый текст превышает установленные границы поля, то текстовое поле растет, чтобы вместить весь текст. Но визуально это не очень хорошо выглядит. Поэтому, как и в случае с TextBlock, мы можем перенести непомещающийся текст на новую строку, установив свойство TextWrapping="Wrap".

Чобы переводить по нажатию на клавишу Enter курсор на следующую строку, нам надо установить свойство AcceptsReturn="True".

Также мы можем добавить полю возможность создавать табуляцию с помощью клавиши Tab, установив свойство AcceptsTab="True"

Для отображения полос прокрутки TextBox поддерживает свойства **VerticalScrollBarVisibility** и **НоrizontalScrollBarVisibility**:

|  |  |
| --- | --- |
| 1  2 | <TextBox AcceptsReturn="True" Height="100" VerticalScrollBarVisibility="Auto"           HorizontalScrollBarVisibility="Auto">Начальный текст</TextBox> |

Возможно, при создании приложения нам потребуется сделать текстовое поле недоступным для ввода (на время в зависимости от условий или вообще), тогда для этого нам надо установить свойство IsReadOnly="True".

Для выделения текста есть свойства **SelectionStart**, **SelectionLength** и **SelectionText**. Например, выделим программно текст по нажатию кнопки:

|  |  |
| --- | --- |
| 1  2  3  4 | <StackPanel>      <TextBox x:Name="textBox1" Height="100" SelectionBrush="Blue" />      <Button Content="Выделить текст" Height="30" Width="100" Click="Button\_Click" Margin="10" />  </StackPanel> |

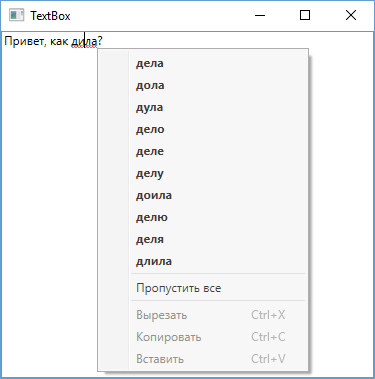
Обработчик нажатия кнопки:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | private void Button\_Click(object sender, RoutedEventArgs e)  {      textBox1.SelectionStart = 5;      textBox1.SelectionLength = 10;      textBox1.Focus();      // данное выражение эквивалентно      //textBox1.Select(5, 10);  } |

#### Проверка орфографии

TextBox обладает встроенной поддержкой орфографии. Чтобы ее задействовать, надо установить свойство SpellCheck.IsEnabled="True". Кроме того, по умолчанию проверка орфографии распространяется только на английский язык, поэтому, если приложение заточено под другой язык, нам надо его явным образом указать через свойство **Language**:

|  |  |
| --- | --- |
| 1  2  3 | <DockPanel>      <TextBox SpellCheck.IsEnabled="True" Language="ru-ru">Привет, как дила?</TextBox>  </DockPanel> |



### Метка (Label)

Главной особенностью меток является поддержка мнемонических команд-клавиш быстрого доступа, которые передают фокус связанному элементу. Например,

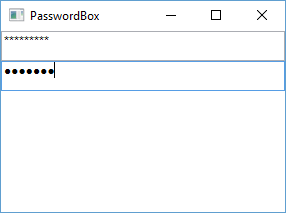
|  |  |
| --- | --- |
| 1  2 | <Label Target="{Binding ElementName=TextBox1}">\_привет</Label>  <TextBox Name="TextBox1" Margin="0 30 0 0" Height="30" Width="100"></TextBox> |

Теперь, нажав на клавишу "п", мы переведем фокус на связанное текстовое поле. При вызове приложения подчеркивание не отображается, чтобы отображать подчеркивание, надо нажать на клавишу Alt. Тогда чтобы перевести фокус на связанное текстовое поле необходимо будет нажать сочетание Alt + "п". Если не предполагается использование клавиш быстрого доступа, то для вывода обычной текста вместо меток лучше использовать элемент TextBlock.

### PasswordBox

Элемент предназначен для ввода парольной информации. По сути это тоже текстовое поле, только для ввода символов используется маска. Свойство **PasswordChar** устанавливает символ маски, отображаемый при вводе пароля. Если это свойство не задано, то по умолчанию для маски символа используется черная точка. Свойство **Password** устанавливает парольную строку, отображаемую по умолчанию при загрузке окна приложения.

|  |  |
| --- | --- |
| 1  2  3  4 | <StackPanel>      <PasswordBox PasswordChar="\*" MinHeight="30" />      <PasswordBox MinHeight="30" />  </StackPanel> |



### RichTextBox

Для вывода текстового содержимого, насыщенного форматированием, графикой, предназначен RichTextBox. Можно даже сказать, что он выводит не просто текст, а документы с более сложным форматированием, чем обычный TextBox. Более подробно о нем, а также как на его основе создать простой текстовый редактор, мы поговорим в главе, посвященной документам.

## Создание элемента управления RichTextBox

В следующем примере кода показано, как создать элемент управления RichTextBox, в котором пользователь может редактировать форматированное содержимое.

XAML

<Page xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<!-- A RichTextBox with no initial content in it. -->

<RichTextBox />

</Page>

В частности, содержимое, редактируемое в элементе управления RichTextBox, имеет нефиксированный формат. Содержимое нефиксированного формата может включать множество типов элементов, в том числе форматированный текст, изображения, списки и таблицы. Чтобы в элементе управления RichTextBox можно было размещать содержимое нефиксированного формата, он содержит объект FlowDocument, который в свою очередь содержит редактируемое содержимое. Чтобы продемонстрировать содержимое нефиксированного формата в элементе управления RichTextBox, воспользуйтесь следующим примером кода, в котором показано, как создать элемент управления RichTextBox с абзацем, часть текста в котором выделена полужирным.

XAML

<Page xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<StackPanel>

<RichTextBox>

<FlowDocument>

<Paragraph>

This is flow content and you can <Bold>edit me!</Bold>

</Paragraph>

</FlowDocument>

</RichTextBox>

</StackPanel>

</Page>

C#Копировать

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Windows.Documents;

namespace SDKSample

{

public partial class BasicRichTextBoxWithContentExample : Page

{

public BasicRichTextBoxWithContentExample()

{

StackPanel myStackPanel = new StackPanel();

// Create a FlowDocument to contain content for the RichTextBox.

FlowDocument myFlowDoc = new FlowDocument();

// Create a Run of plain text and some bold text.

Run myRun = new Run("This is flow content and you can ");

Bold myBold = new Bold(new Run("edit me!"));

// Create a paragraph and add the Run and Bold to it.

Paragraph myParagraph = new Paragraph();

myParagraph.Inlines.Add(myRun);

myParagraph.Inlines.Add(myBold);

// Add the paragraph to the FlowDocument.

myFlowDoc.Blocks.Add(myParagraph);

RichTextBox myRichTextBox = new RichTextBox();

// Add initial content to the RichTextBox.

myRichTextBox.Document = myFlowDoc;

myStackPanel.Children.Add(myRichTextBox);

this.Content = myStackPanel;

}

}

}

На рисунке ниже показано, как будет выглядеть этот пример.

RichTextBox with content

Такие элементы как Paragraph и Bold определяют, как отображается содержимое внутри элемента управления RichTextBox. По мере того, как пользователь редактирует содержимое RichTextBox, он изменяет это содержимое нефиксированного формата.

**Примечание**

Содержимое нефиксированного формата внутри элемента управления **RichTextBox** ведет себя не так, как в других элементах управления. Например, в элементе управления **RichTextBox** отсутствуют столбцы, и поэтому автоматическое изменение размеров не выполняется. Кроме того, встроенные функции, такие как поиск, режим просмотра, навигация по страницам и увеличение, недоступны в элементе управления **RichTextBox**.

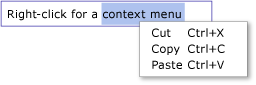
## Проверка орфографии в режиме реального времени

Вы можете включить проверку орфографии в реальном времени для элементов управления TextBox и RichTextBox. При включенной проверке орфографии все слова с ошибками подчеркиваются красной линией (см. рисунок ниже).

Textbox with spell-checking

## Контекстное меню

По умолчанию элементы управления TextBox и RichTextBox имеют контекстное меню, которое отображается при щелчке пользователя правой кнопкой мыши внутри элемента управления. Контекстное меню дает пользователю возможность вырезания, копирования и вставки (см. рисунок ниже).



Можно создать собственное пользовательское контекстное меню, чтобы переопределить меню по умолчанию..

## Команды изменения

Команды редактирования дают пользователям возможность форматировать редактируемое содержимое в RichTextBox. Наряду с базовыми командами редактирования RichTextBox включает команды форматирования, не поддерживаемые элементом управления TextBox. Например, при редактировании в RichTextBox пользователь может нажать CTRL+B, чтобы сделать текст полужирным или превратить текст из полужирного в обычный. Полный список команд см. в разделе EditingCommands. Помимо использования сочетания клавиш, можно подключать команды к другим элементам управления, таким как кнопки. В следующем примере показано, как создать простую панель инструментов, содержащую кнопки, с помощью которых пользователь может изменять форматирование текста.

XAML

<Window x:Class="RichTextBoxInputPanelDemo.Window1"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Height="400" Width="600"

>

<Grid>

<!-- Set the styles for the tool bar. -->

<Grid.Resources>

<Style TargetType="{x:Type Button}" x:Key="formatTextStyle">

<Setter Property="FontFamily" Value="Palatino Linotype"></Setter>

<Setter Property="Width" Value="30"></Setter>

<Setter Property="FontSize" Value ="14"></Setter>

<Setter Property="CommandTarget" Value="{Binding ElementName=mainRTB}"></Setter>

</Style>

<Style TargetType="{x:Type Button}" x:Key="formatImageStyle">

<Setter Property="Width" Value="30"></Setter>

<Setter Property="CommandTarget" Value="{Binding ElementName=mainRTB}"></Setter>

</Style>

</Grid.Resources>

<DockPanel Name="mainPanel">

<!-- This tool bar contains all the editing buttons. -->

<ToolBar Name="mainToolBar" Height="30" DockPanel.Dock="Top">

<Button Style="{StaticResource formatImageStyle}" Command="ApplicationCommands.Cut" ToolTip="Cut">

<Image Source="Images\EditCut.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="ApplicationCommands.Copy" ToolTip="Copy">

<Image Source="Images\EditCopy.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="ApplicationCommands.Paste" ToolTip="Paste">

<Image Source="Images\EditPaste.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="ApplicationCommands.Undo" ToolTip="Undo">

<Image Source="Images\EditUndo.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="ApplicationCommands.Redo" ToolTip="Redo">

<Image Source="Images\EditRedo.png"></Image>

</Button>

<Button Style="{StaticResource formatTextStyle}" Command="EditingCommands.ToggleBold" ToolTip="Bold">

<TextBlock FontWeight="Bold">B</TextBlock>

</Button>

<Button Style="{StaticResource formatTextStyle}" Command="EditingCommands.ToggleItalic" ToolTip="Italic">

<TextBlock FontStyle="Italic" FontWeight="Bold">I</TextBlock>

</Button>

<Button Style="{StaticResource formatTextStyle}" Command="EditingCommands.ToggleUnderline" ToolTip="Underline">

<TextBlock TextDecorations="Underline" FontWeight="Bold">U</TextBlock>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="EditingCommands.IncreaseFontSize" ToolTip="Grow Font">

<Image Source="Images\CharacterGrowFont.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="EditingCommands.DecreaseFontSize" ToolTip="Shrink Font">

<Image Source="Images\CharacterShrinkFont.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="EditingCommands.ToggleBullets" ToolTip="Bullets">

<Image Source="Images\ListBullets.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="EditingCommands.ToggleNumbering" ToolTip="Numbering">

<Image Source="Images/ListNumbering.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="EditingCommands.AlignLeft" ToolTip="Align Left">

<Image Source="Images\ParagraphLeftJustify.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="EditingCommands.AlignCenter" ToolTip="Align Center">

<Image Source="Images\ParagraphCenterJustify.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="EditingCommands.AlignRight" ToolTip="Align Right">

<Image Source="Images\ParagraphRightJustify.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="EditingCommands.AlignJustify" ToolTip="Align Justify">

<Image Source="Images\ParagraphFullJustify.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="EditingCommands.IncreaseIndentation" ToolTip="Increase Indent">

<Image Source="Images\ParagraphIncreaseIndentation.png"></Image>

</Button>

<Button Style="{StaticResource formatImageStyle}" Command="EditingCommands.DecreaseIndentation" ToolTip="Decrease Indent">

<Image Source="Images\ParagraphDecreaseIndentation.png"></Image>

</Button>

</ToolBar>

<!-- By default pressing tab moves focus to the next control. Setting AcceptsTab to true allows the

RichTextBox to accept tab characters. -->

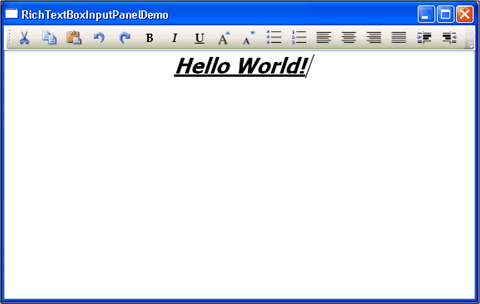
<RichTextBox Name="mainRTB" AcceptsTab="True"></RichTextBox>

</DockPanel>

</Grid>

</Window>

На следующем рисунке показано, как будет выглядеть этот пример.



## Определение изменения содержимого

Обычно для того, чтобы определить изменение текста в элементах управления TextBox и RichTextBox используется событие TextChanged, а не KeyDown, как можно было ожидать.

## Сохранение, загрузка и печать содержимого RichTextBox

В следующем примере показано, как сохранить содержимое RichTextBox в файл, как загрузить это содержимое обратно в RichTextBox и как распечатать содержимое. Ниже для примера приведена разметка.

XAML

<Page xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

x:Class="SDKSample.SaveLoadPrintRTB" >

<StackPanel>

<RichTextBox Name="richTB">

<FlowDocument>

<Paragraph>

<Run>Paragraph 1</Run>

</Paragraph>

</FlowDocument>

</RichTextBox>

<Button Click="SaveRTBContent">Save RTB Content</Button>

<Button Click="LoadRTBContent">Load RTB Content</Button>

<Button Click="PrintRTBContent">Print RTB Content</Button>

</StackPanel>

</Page>

Ниже для примера приведен код.

C#

using System;

using System.IO;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Media;

namespace SDKSample

{

public partial class SaveLoadPrintRTB : Page

{

// Handle "Save RichTextBox Content" button click.

void SaveRTBContent(Object sender, RoutedEventArgs args)

{

// Send an arbitrary URL and file name string specifying

// the location to save the XAML in.

SaveXamlPackage("C:\\test.xaml");

}

// Handle "Load RichTextBox Content" button click.

void LoadRTBContent(Object sender, RoutedEventArgs args)

{

// Send URL string specifying what file to retrieve XAML

// from to load into the RichTextBox.

LoadXamlPackage("C:\\test.xaml");

}

// Handle "Print RichTextBox Content" button click.

void PrintRTBContent(Object sender, RoutedEventArgs args)

{

PrintCommand();

}

// Save XAML in RichTextBox to a file specified by \_fileName

void SaveXamlPackage(string \_fileName)

{

TextRange range;

FileStream fStream;

range = new TextRange(richTB.Document.ContentStart, richTB.Document.ContentEnd);

fStream = new FileStream(\_fileName, FileMode.Create);

range.Save(fStream, DataFormats.XamlPackage);

fStream.Close();

}

// Load XAML into RichTextBox from a file specified by \_fileName

void LoadXamlPackage(string \_fileName)

{

TextRange range;

FileStream fStream;

if (File.Exists(\_fileName))

{

range = new TextRange(richTB.Document.ContentStart, richTB.Document.ContentEnd);

fStream = new FileStream(\_fileName, FileMode.OpenOrCreate);

range.Load(fStream, DataFormats.XamlPackage);

fStream.Close();

}

}

// Print RichTextBox content

private void PrintCommand()

{

PrintDialog pd = new PrintDialog();

if ((pd.ShowDialog() == true))

{

//use either one of the below

pd.PrintVisual(richTB as Visual, "printing as visual");

pd.PrintDocument((((IDocumentPaginatorSource)richTB.Document).DocumentPaginator), "printing as paginator");

}

}

}

}