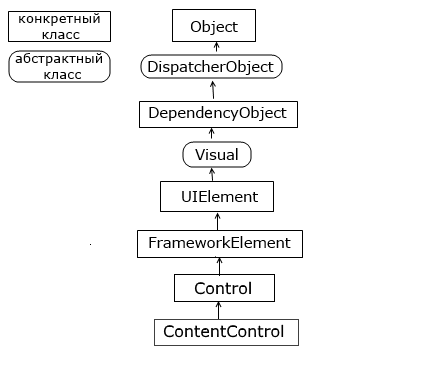
## Элементы управления содержимым

Элементы управления содержимым (content controls) представляют такие элементы управления, которые содержат в себе другой элемент. Все элементы управления содержимым наследуются от класса **ContentControl**, который в свою очередь наследуется от класса System.Window.Controls.Control.



К элементам управления содержимым относятся такие элементы как Button, Label, ToggleButton, ToolTip, RadioButton, CheckBox, GroupBox, TabItem, Expander, ScrollViewer. Также элементом управления содержимым является и главный элемент окна - Window.

Отличительной чертой всех этих элементов является наличие свойства **Content**, которое и устанавливает вложенный элемент. В этом элементы управления содержимым схожи с контейнерами компоновки. Только контейнеры могут иметь множество вложенных элементов, а элементы управления содержимым только один.

Свойство Content может представлять любой объект, который может относиться к одному из двух типов:

* Объект класса, не наследующего от UIElement. Для такого объекта вызывается метод ToString(), который возвращает строковое преставление объекта. Затем эта строка устанавливается в качестве содержимого.
* Объект класса, наследующего от UIElement. Для такого объекта вызывается метод UIElement.OnRender(), который выполняет отрисовку внутри элемента управления содержимым.

Рассмотрим на примере кнопки, которая является элементом управления содержимым:

|  |  |
| --- | --- |
| 1 | <Button Content="Hello World!" /> |

В качестве содержимого устанавливается обычная строка. Этот же пример мы можем в XAML прописать иначе:

|  |  |
| --- | --- |
| 1  2  3  4  5 | <Button>      <Button.Content>          Hello World!      </Button.Content>  </Button> |

Либо мы можем использовать сокращенное неявное определения свойства Content:

|  |  |
| --- | --- |
| 1  2  3 | <Button>      Hello World!  </Button> |

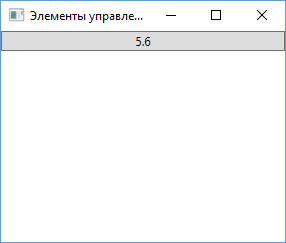
Возьмем другой пример. Определим кнопку с именем button1:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <Window x:Class="ControlsApp.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:ControlsApp"          mc:Ignorable="d"          Title="Элементы управления" Height="250" Width="300">      <StackPanel>          <Button x:Name="button1" />      </StackPanel>  </Window> |

А в файле коде MainWindow.xaml.cs присвоим ее свойству Content какой-либо объект:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | using System;  using System.Windows;    namespace ControlsApp  {      public partial class MainWindow : Window      {          public MainWindow()          {              InitializeComponent();              double d = 5.6;              button1.Content = d;          }      }  } |

В итоге мы получим следующую кнопку:

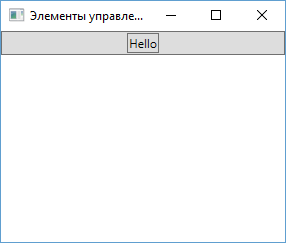


В итоге число конвертируется в строку и устанавливается в качесте содержимого.

Иначе все будет работать, если мы в качестве содержимого используем объект, унаследованный от UIElement:

|  |  |
| --- | --- |
| 1  2  3 | <Button x:Name="button1">      <Button Content="Hello" />  </Button> |

Теперь в качестве содержимого будет использоваться другая кнопка, для которой при визуализации будет вызываться меод OnRender():

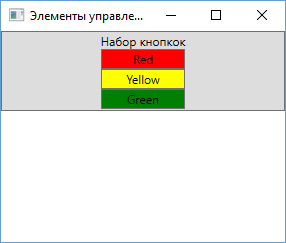


Для создания той же кнопки через код C# мы бы могли прописать следующее выражение:

|  |  |
| --- | --- |
| 1 | button1.Content = new Button { Content = "Hello" }; |

В отличие от контейнеров компоновки для элементов управления содержимым мы можем задать только один вложенный элемент. Если же нам надо вложить в элемент управления содержимым несколько элементов, то мы можем использовать те же контейнеры компоновки:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <Button x:Name="button1">      <StackPanel>          <TextBlock Text="Набор кнопкок" />          <Button Background="Red" Height="20" Content="Red" />          <Button Background="Yellow" Height="20" Content="Yellow" />          <Button Background="Green" Height="20" Content="Green" />      </StackPanel>  </Button> |



То же самое мы могли бы прописать через код C#:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | StackPanel stackPanel = new StackPanel();  stackPanel.Children.Add(new TextBlock { Text = "Набор кнопок" });  stackPanel.Children.Add(new Button { Content = "Red", Height = 20, Background = new SolidColorBrush(Colors.Red) });  stackPanel.Children.Add(new Button { Content = "Yellow", Height = 20, Background = new SolidColorBrush(Colors.Yellow) });  stackPanel.Children.Add(new Button { Content = "Green", Height = 20, Background = new SolidColorBrush(Colors.Green) });  button1.Content = stackPanel; |

### Позиционирование контента

#### Content Alignment

Выравнивание содержимого внутри элемента задается свойствами **HorizontalContentAlignment** (выравнивание по горизонтали) и **VerticalContentAlignment** (выравнивание по вертикали), аналогичны свойствам VerticalAlignment/HorizontalAlignment. Свойство HorizontalContentAlignment принимает значения Left, Right, Center (положение по центру), Stretch (растяжение по всей ширине). Например:

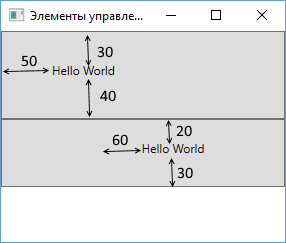
|  |  |
| --- | --- |
| 1  2  3  4  5 | <StackPanel>      <Button Margin="5" HorizontalContentAlignment="Left" Content="Left" Height="90" Width="500" />      <Button Margin="5" HorizontalContentAlignment="Right" Content="Right" Height="90" Width="500" />      <Button Margin="5" HorizontalContentAlignment="Center" Content="Center" Height="90" Width="500" />  </StackPanel> |

VerticalContentAlignment принимает значения Top (положение в верху), Bottom (положение внизу), Center (положение по центру), Stretch (растяжение по всей высоте)

#### Padding

С помощью свойства Padding мы можем установить отступ содержимого элемента:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <StackPanel>      <Button x:Name="button1" Padding="50 30 0 40" HorizontalContentAlignment="Left">          Hello World      </Button>      <Button x:Name="button2" Padding="60 20 0 30" HorizontalContentAlignment="Center">          Hello World      </Button>  </StackPanel> |



Свойство Padding задается в формате Padding="отступ\_слева отступ\_сверху отступ\_справа отступ\_снизу".

Если со всех четырех сторон предполагается один и тот же отступ, то, как и в случае с Margin, мы можем задать одно число:

|  |  |
| --- | --- |
| 1 | <Button x:Name="button2" Padding="20"  Content="Hello World" /> |

Важно понимать, от какой точки задается отступ. В случае с первой кнопкой в ней контект выравнивается по левому краю, поэтому отступ слева будет предполагать отступ от левого края элемента Button. А вторая кнопка располагается по центру. Поэтому для нее отступ слева предполагает отступ от той точки, в которой содержимое бы находилось при центрировании без применения Padding.

Комбинация значений свойств HorizontalContentAlignment/VerticalContentAlignment и Padding позволяет оптимальным образом задать расположение содержимого.