

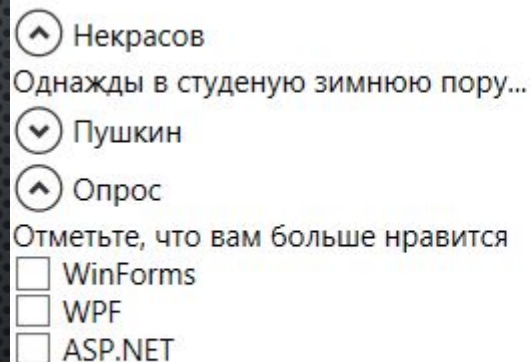
ЭЛЕМЕНТЫ УПРАВЛЕНИЯ



# EXPANDER

Представляет скрытое содержимое, раскрывающееся по нажатию мышкой на указатель в виде стрелки.

```
<StackPanel>
  <Expander Header="Некрасов" IsExpanded="True">
    <TextBlock>Однажды в студеную зимнюю пору...</TextBlock>
  </Expander>
  <Expander Header="Пушкин">
    <TextBlock>Онегин был, по мнению многих, ученый малый, но ...</TextBlock>
  </Expander>
  <Expander Header="Опрос">
    <StackPanel>
      <TextBlock>Отметьте, что вам больше нравится</TextBlock>
      <CheckBox>WinForms</CheckBox>
      <CheckBox>WPF</CheckBox>
      <CheckBox>ASP.NET</CheckBox>
    </StackPanel>
  </Expander>
</StackPanel>
```





# EXPANDER

## Программное создание EXPANDER

```
StackPanel expanderPanel = new StackPanel();
expanderPanel.Children.Add(new CheckBox { Content = "WinForms" });
expanderPanel.Children.Add(new CheckBox { Content = "WPF" });
expanderPanel.Children.Add(new CheckBox { Content = "ASP.NET" });

Expander expander = new Expander();
expander.Header = "Выберите технологию";
expander.Content = expanderPanel;

stackPanel.Children.Add(expander);
```



# ПОЛОСЫ ПРОКРУТКИ

```
<ScrollView>
  <StackPanel>
    <Button MinHeight="60" Background="Red"/>
    <Button MinHeight="60" Background="Orange"/>
    <Button MinHeight="60" Background="Yellow"/>
    <Button MinHeight="60" Background="Green"/>
    <Button MinHeight="60" Background="Blue"/>
  </StackPanel>
</ScrollView>
```



ScrollView поддерживает как вертикальную, так и горизонтальную прокрутку (свойства **HorizontalScrollBarVisibility** и **VerticalScrollBarVisibility**). Эти свойства принимают одно из следующих значений:

- **Auto**: наличие полос прокрутки устанавливается автоматически
- **Visible**: полосы прокрутки отображаются в окне приложения
- **Hidden**: полосы прокрутки не видно, но прокрутка возможна с помощью клавиш клавиатуры
- **Disabled**: полосы прокрутки не используются, а сама прокрутка даже с помощью клавиатуры невозможна

**CanContentScroll** – свойство для прокрутки к началу следующего элемента.

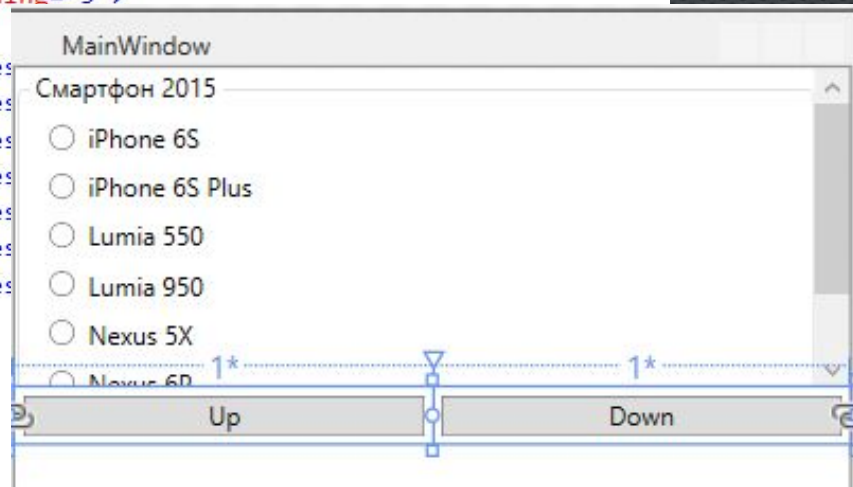


# ПОЛОСЫ ПРОКРУТКИ

Прокрутка программным способом - с помощью следующих методов элемента ScrollViewer:

- `LineUp()`, `LineDown()`, `LineRight()`, `LineLeft()`: прокрутка соответственно вверх, вниз, вправо, влево.
- `ScrollToEnd()`, `ScrollToHome()`: прокрутка в конец окна и в начало.
- `ScrollToRightEnd()`, `ScrollToLeftEnd()`: прокрутка в правый и левый конец окна.

```
<StackPanel>
  <ScrollViewer Name="scroll" CanContentScroll="True" Height="150">
    <GroupBox Header="Смартфон 2015" Padding="5">
      <StackPanel>
        <RadioButton GroupName="Phones" />
        <RadioButton GroupName="Phones" />
        <RadioButton GroupName="Phones" />
        <RadioButton GroupName="Phones" />
        <RadioButton GroupName="Phones" />
        <RadioButton GroupName="Phones" />
      </StackPanel>
    </GroupBox>
  </ScrollViewer>
  <Grid>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="*" />
      <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Button Content="Up" Grid.Column="0" Margin="4" Click="Up_Click" />
    <Button Content="Down" Grid.Column="1" Margin="4" Click="Down_Click" />
  </Grid>
</StackPanel>
```



```
void Up_Click(object sender, RoutedEventArgs e)
{
    scroll.LineUp();
}

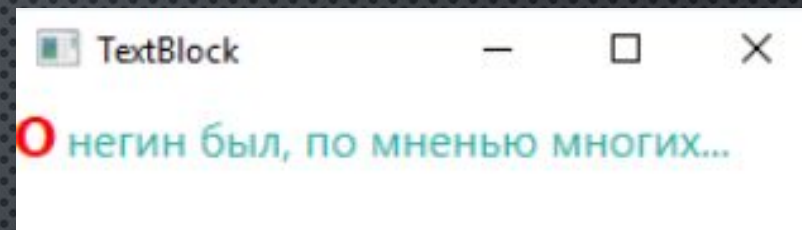
void Down_Click(object sender, RoutedEventArgs e)
{
    scroll.LineDown();
}
```



# TEXTBLOCK

Элемент предназначен для вывода текстовой информации, для создания простых надписей

```
<TextBlock TextWrapping="Wrap">
  <Run FontSize="20" Foreground="Red" FontWeight="Bold">0</Run>
  <Run FontSize="16" Foreground="LightSeaGreen">негин был, по мнению многих...</Run>
</TextBlock>
```



- Свойство **Run** представляют куски обычного текста, для которых можно задать отдельное форматирование.
- Свойство **LineHeight** позволяет указывать высоту строк.
- Свойство **TextWrapping** позволяет переносить текст при `TextWrapping="Wrap"`.
- Свойство **TextAlignment** выравнивает текст по центру, правому или левому краю.
- Свойство **TextDecorations** для декорации текста.
- Свойство **LineBreak** для переноса на новую строку.

```
<TextBlock>
  Однажды в студеную зимнюю пору
  <LineBreak />
  Я из лесу вышел
</TextBlock>
```



# LABEL / PASSWORDBOX

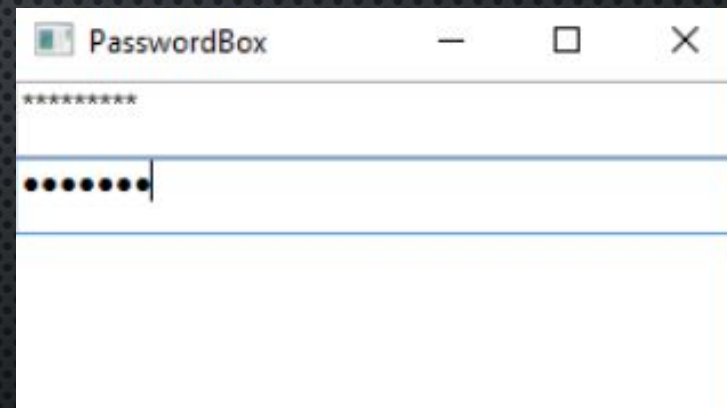
Элемент предназначен для поддержки клавиш быстрого доступа

```
<Label Target="{Binding ElementName=TextBox1}">_привет</Label>  
<TextBox Name="TextBox1" Margin="0 30 0 0" Height="30" Width="100"></TextBox>
```

Alt + первая буква сообщения

Элемент предназначен для ввода парольной информации

```
<StackPanel>  
    <PasswordBox PasswordChar="*" MinHeight="30" />  
    <PasswordBox MinHeight="30" />  
</StackPanel>
```



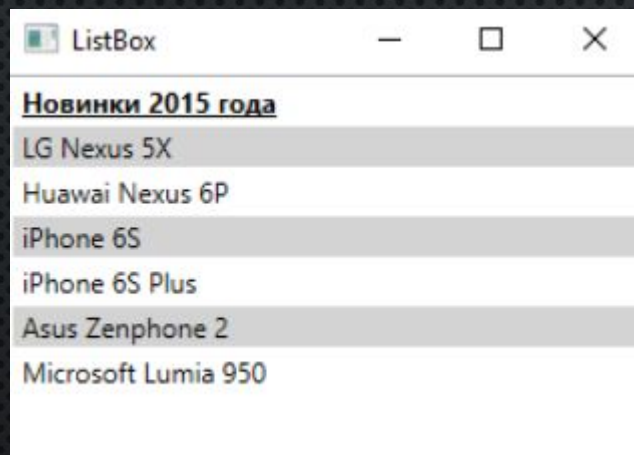


# LISTBOX

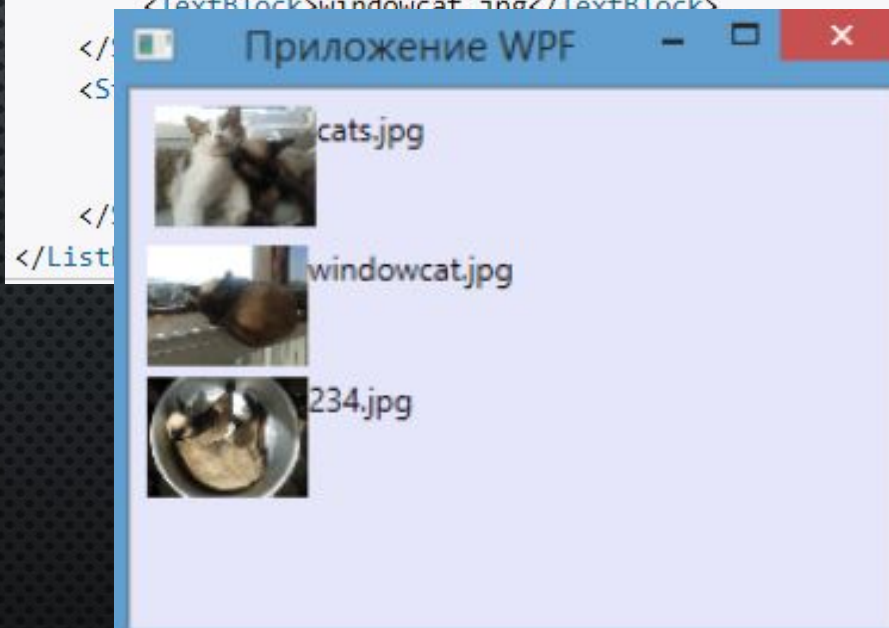
Элемент представляет собой обычный список

```
<ListBox Name="phonesList">  
  <TextBlock FontWeight="Bold" TextDecorations="Underline" Text="Новинки 2015 года" />  
  <ListBoxItem Background="LightGray">LG Nexus 5X</ListBoxItem>  
  <ListBoxItem>Huawei Nexus 6P</ListBoxItem>  
  <ListBoxItem Background="LightGray">iPhone 6S</ListBoxItem>  
  <ListBoxItem>iPhone 6S Plus</ListBoxItem>  
  <ListBoxItem Background="LightGray">Asus Zenphone 2</ListBoxItem>  
  <ListBoxItem>Microsoft Lumia 950</ListBoxItem>  
</ListBox>
```

Доступ по индексу: `phoneList.Items[0]`



```
<ListBox Name="Photos" Background="Lavender">  
  <ListBoxItem Margin="3">  
    <StackPanel Orientation="Horizontal">  
      <Image Source="cats.jpg" Width="60" />  
      <TextBlock>cats.jpg</TextBlock>  
    </StackPanel>  
  </ListBoxItem>  
  <StackPanel Orientation="Horizontal">  
    <Image Source="windowcat.jpg" Width="60" />  
    <TextBlock>windowcat.jpg</TextBlock>  
  </StackPanel>  
</ListBox>
```

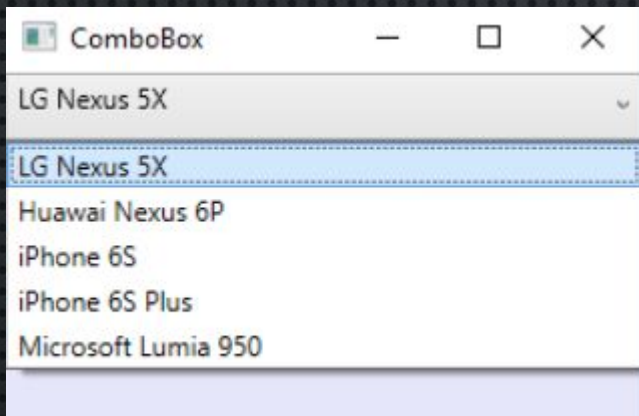




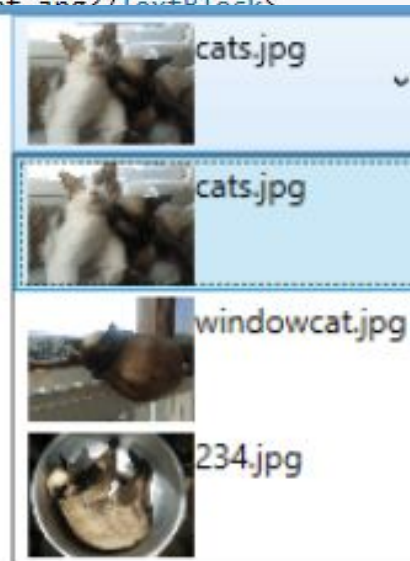
# COMBOBOX

ComboBox содержит коллекцию элементов и образует выпадающий список

```
<ComboBox Name="phonesList" Height="30" VerticalAlignment="Top">  
  <TextBlock>LG Nexus 5X</TextBlock>  
  <TextBlock>Huawei Nexus 6P</TextBlock>  
  <TextBlock>iPhone 6S</TextBlock>  
  <TextBlock>iPhone 6S Plus</TextBlock>  
  <TextBlock>Microsoft Lumia 950</TextBlock>  
</ComboBox>
```



```
<ComboBox Height="50" Width="150" VerticalAlignment="Top">  
  <ComboBoxItem IsSelected="True">  
    <StackPanel Orientation="Horizontal">  
      <Image Source="cats.jpg" Width="60" />  
      <TextBlock>cats.jpg</TextBlock>  
    </StackPanel>  
  </ComboBoxItem>  
  <StackPanel Orientation="Horizontal">  
    <Image Source="windowcat.jpg" Width="60" />  
    <TextBlock>windowcat.jpg</TextBlock>  
  </StackPanel>  
  <StackPanel Orientation="Horizontal">  
    <Image Source="234.jpg" Width="60" />  
    <TextBlock>234.jpg</TextBlock>  
  </StackPanel>  
</ComboBox>
```





# ВКЛАДКИ TABCONTROL

## Элементарная структура вкладки

```
<TabControl>
  <TabItem Header="Вкладка 1">Первая вкладка</TabItem>
  <TabItem Header="Вкладка 2">Вторая вкладка</TabItem>
</TabControl>
```

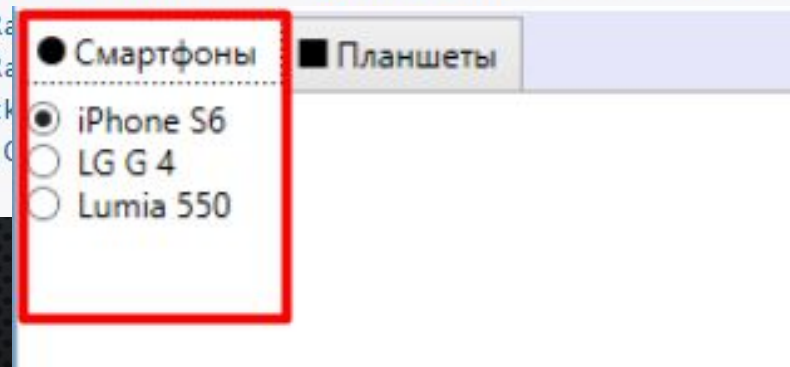
## Программное добавление вкладок

```
<TabControl x:Name="products">
</TabControl>
```

```
// формируем содержимое вкладки в виде списка
ListBox notesList = new ListBox();
notesList.Items.Add("Macbook Pro");
notesList.Items.Add("HP Pavilion 5478");
notesList.Items.Add("Acer LK-08");
// добавление вкладки
products.Items.Add(new TabItem
{
  Header = new TextBlock { Text = "Ноутбуки" }, // установка заголовка
  Content = notesList // установка содержимого вкладки
});
```

## Вкладка со сложной структурой

```
<TabControl x:Name="products">
  <TabItem x:Name="smartphonesTab">
    <TabItem.Header>
      <StackPanel Orientation="Horizontal">
        <Ellipse Height="10" Width="10" Fill="Black" />
        <TextBlock Margin="3">Смартфоны</TextBlock>
      </StackPanel>
    </TabItem.Header>
    <TabItem.Content>
      <StackPanel>
        <RadioButton IsChecked="True">iPhone S6</RadioButton>
        <RadioButton />
        <RadioButton />
      </StackPanel>
    </TabItem.Content>
  </TabItem>
</TabControl>
```

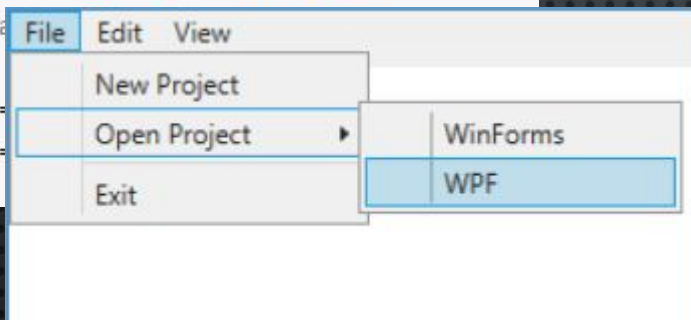




# МЕНЮ

## Элемент для создания стандартного меню

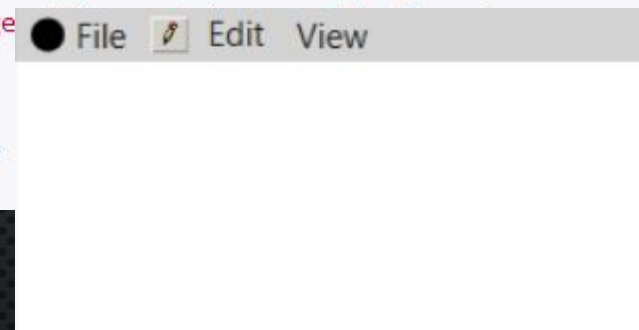
```
<Menu Height="25" VerticalAlignment="Top">
  <MenuItem Header="File">
    <MenuItem Header="New Project" ></MenuItem>
    <MenuItem Header="Open Project" >
      <MenuItem Header="WinForms"></MenuItem>
      <MenuItem Header="WPF" ></MenuItem>
    </MenuItem>
    <Separator />
    <MenuItem Header="Edit" ></MenuItem>
  </MenuItem>
  <MenuItem Header="View" ></MenuItem>
</Menu>
```



```
<MenuItem Header="View" Click="MenuItem_Click"></MenuItem>
```

```
private void MenuItem_Click(object sender, RoutedEventArgs e)
{
    MenuItem menuItem = (MenuItem)sender;
    MessageBox.Show(menuItem.Header.ToString());
}
```

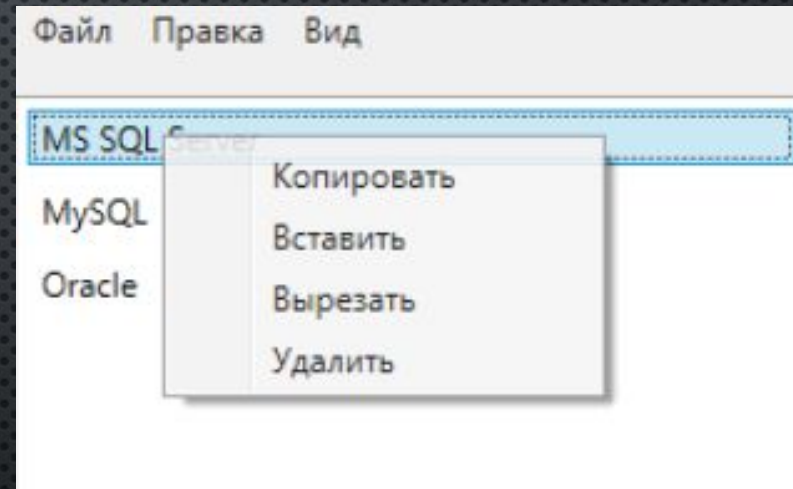
```
<Menu Height="25" VerticalAlignment="Top" Background="LightGray">
  <MenuItem>
    <MenuItem.Header>
      <StackPanel Orientation="Horizontal">
        <Ellipse Height="10" Width="10" Fill="Black" Margin="0 0 5 0" />
        <TextBlock>File</TextBlock>
      </StackPanel>
    </MenuItem.Header>
  </MenuItem>
  <MenuItem Header="Edit">
    <MenuItem.Icon>
      <Image Source="C:\Users\Euge" />
    </MenuItem.Icon>
  </MenuItem>
  <MenuItem Header="View"></MenuItem>
</Menu>
```





# КОНТЕКСТНОЕ МЕНЮ

```
<ListBox Name="list" Height="145">
  <ListBoxItem Margin="3">MS SQL Server</ListBoxItem>
  <ListBoxItem Margin="3">MySQL</ListBoxItem>
  <ListBoxItem Margin="3">Oracle</ListBoxItem>
  <ListBox.ContextMenu>
    <ContextMenu>
      <MenuItem Header="Копировать"></MenuItem>
      <MenuItem Header="Вставить"></MenuItem>
      <MenuItem Header="Вырезать"></MenuItem>
      <MenuItem Header="Удалить"></MenuItem>
    </ContextMenu>
  </ListBox.ContextMenu>
</ListBox>
```





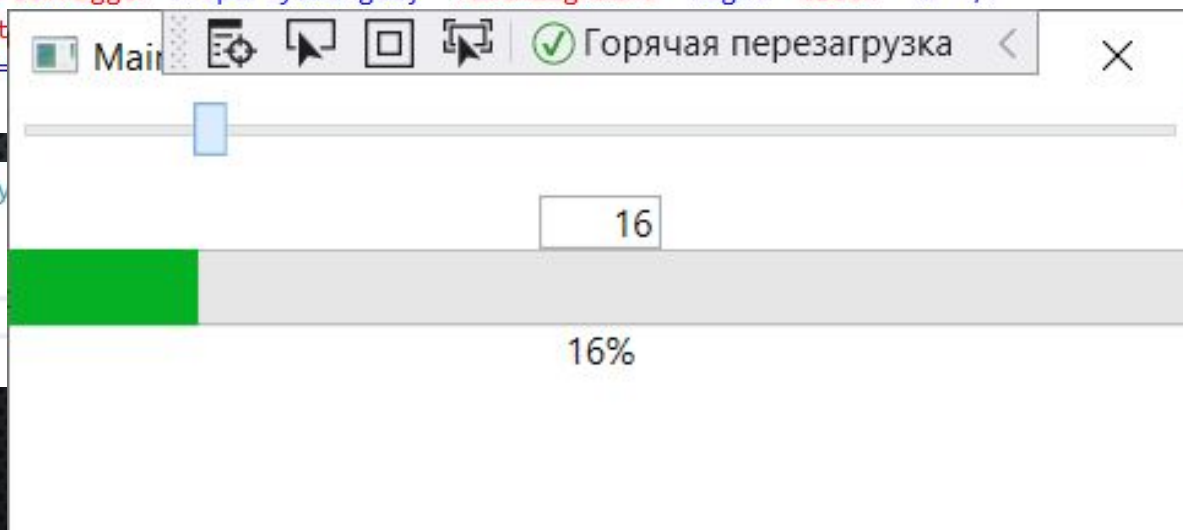
# SLIDER / PROGRESSBAR

Представляет собой обычный ползунок.

- **Orientation**: ориентация ползунка - горизонтальная (Horizontal) или вертикальная (Vertical)
- **Delay**: время в миллисекундах, по истечении которого ползунок переместится на одну единицу после щелчка.
- **Interval**: время в миллисекундах, по истечении которого ползунок может перемещаться
- **TickFrequency**: указывает частоту появления отметок на шкале ползунка.

```
<StackPanel>  
  <Slider x:Name="slValue" Value="0" Minimum="0" Maximum="100" Height="30" TickFrequency="1" IsSnapToTickEnabled="True" ValueChanged="Slider_ValueChanged"/>  
  <TextBox Text="{Binding ElementName=slValue, Path=Value, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Right" Width="40" />  
  <ProgressBar x:Name="pb" Minimum="0" Maximum="100" Value="0" Height="20" />  
  <TextBlock Text="{Binding ElementName=pb, Path=Value, StringFormat="0.00%"}" />  
</StackPanel>
```

```
private void Slider_ValueChanged(object sender, RoutedProperty  
{  
    ((Slider)sender).SelectionEnd = e.OldValue + 1;  
    pb.Value = e.OldValue + 1;  
}
```



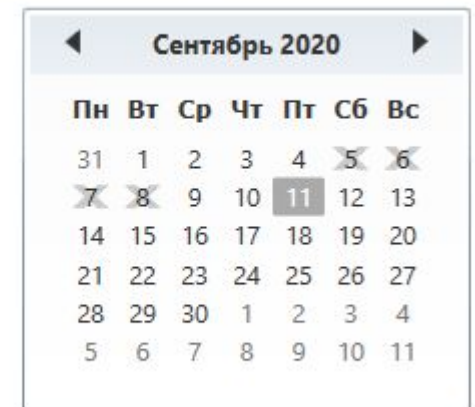


# CALENDAR / DATEPICKER

**Calendar** представляет собой элемент в виде календаря, тогда как **DatePicker** - текстовое поле для ввода даты с выпадающим календарем после ввода.

<b>BlackoutDates</b>	Принимает в качестве значения объект <b>CalendarDateRange</b> , задающий с помощью свойств <b>Start</b> и <b>End</b> диапазон дат, которые будут зачеркнуты в календаре.
<b>DisplayDateStart</b> и <b>DisplayDateEnd</b>	Задают соответственно начальную и конечную дату диапазона, который будет отображаться в календаре.
<b>IsTodayHighlighted</b>	Отмечает, будет ли выделена текущая дата
<b>SelectedDate(SelectedDates)</b>	Задаёт выделённую дату (диапазон выделённых дат)
<b>FirstDayOfWeek</b>	Задаёт первый день недели

```
<StackPanel>
  <Calendar x:Name="calendar1" FirstDayOfWeek="Monday"
SelectedDatesChanged="calendar_SelectedDatesChanged">
    <Calendar.BlackoutDates>
      <CalendarDateRange Start="09/5/2020" End="09/8/2020"></CalendarDateRange>
    </Calendar.BlackoutDates>
  </Calendar>
</StackPanel>
```





# CALENDAR (ВЫДЕЛЕННЫЕ ДАТЫ)

**DisplayMode** (формат отображения дат) и **SelectionMode** (способ выделения)

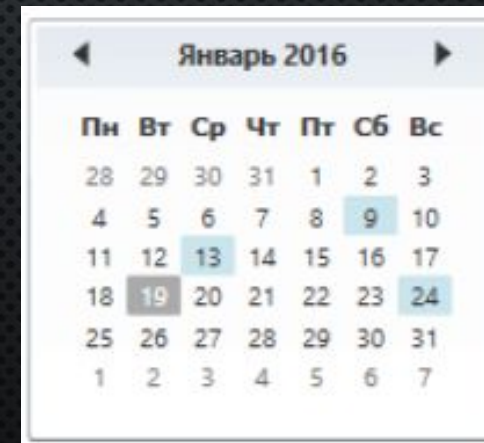
**DisplayMode** может принимать одно из следующих значений:

- **Month** (по умолчанию) отображает все дни текущего месяца
- **Decade** отображает все года текущего десятилетия
- **Year** отображает все месяцы текущего года

**SelectionMode** может принимать одно из следующих значений:

- **SingleDate** (по умолчанию) выделяет только одну дату
- **None** запрещает выделение
- **SingleRange** по нажатию на **Ctrl** выделяет несколько последовательно идущих дат
- **MultipleRange** по нажатию на **Ctrl** выделяет несколько не последовательно идущих диапазонов дат

```
<Calendar x:Name="calendar1" FirstDayOfWeek="Monday" SelectionMode="MultipleRange"
  xmlns:sys="clr-namespace:System;assembly=mscorlib">
  <Calendar.SelectedDates>
    <sys:DateTime>01/9/2016</sys:DateTime>
    <sys:DateTime>01/13/2016</sys:DateTime>
    <sys:DateTime>01/24/2016</sys:DateTime>
  </Calendar.SelectedDates>
</Calendar>
```



The screenshot shows a calendar for January 2016. The days of the week are labeled in Russian: Пн (Monday), Вт (Tuesday), Ср (Wednesday), Чт (Thursday), Пт (Friday), Сб (Saturday), and Вс (Sunday). The dates are arranged in a grid. The dates 9, 13, and 24 are highlighted in blue, indicating they are selected. The date 19 is highlighted in grey, indicating it is the current date. The calendar has navigation arrows at the top.

Пн	Вт	Ср	Чт	Пт	Сб	Вс
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7



# DATEPICKER

- **IsDropDownOpen** (показывает, будет ли связанный с элементом всплывающий календарь оставаться открытым после выбора даты)
- **SelectedDateFormat** (формат даты, принимает значение Short - формат в виде дд.мм.гггг или мм.дд.гггг и значение Long - с полными названиями месяцев)

```
<DatePicker Name="datePicker1" IsDropDownOpen="True" SelectedDateFormat="Short" />
```

