Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Специальность: 09.02.07 Информационные системы и программирование

Практическая работа №9

по дисциплине «Технология разработки программного обеспечения»

на тему «Привязка данных»

Выполнил:

студент гр. ПР-31 \_\_\_\_\_\_\_\_\_\_/Пинигина Д.А.

проверил: \_\_\_\_\_\_\_\_\_\_/Гариев Д.О.

дата: \_\_\_\_\_\_\_\_\_\_

оценка: \_\_\_\_\_\_\_\_\_\_

2022

# **СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc118833313)

[НЕВЕРНЫЕ ЗНАЧЕНИЯ 4](#_Toc118833314)

[Неверные имена 4](#_Toc118833315)

[Неверные свойства 5](#_Toc118833316)

[Неверные имена и свойства 6](#_Toc118833317)

[ЗАДАНИЕ 2 8](#_Toc118833318)

[ЗАДАНИЕ 3 13](#_Toc118833319)

[ЗАДАНИЕ 4 15](#_Toc118833320)

[ВЫВОД 23](#_Toc118833321)

# **ЦЕЛЬ**

Цель: научиться пользоваться привязкой данных, отказаться от обработчиков событий, узнать о составляющих и режимах привязки.

# **НЕВЕРНЫЕ ЗНАЧЕНИЯ**

Возьмём любой код, где имеются привязки данных, и посмотрим, какие ошибки и предупреждения выводит программа.

Имеется код (см. также Рисунок 1):

<StackPanel>

<TextBox x:Name="t1"/>

<TextBox x:Name="t2" Text="{Binding ElementName=t1, Path=Text}" />

<Slider x:Name="slider1" />

<Slider x:Name="slider2" Value="{Binding ElementName=slider1, Path=Value}" />

</StackPanel>

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 — Относительно правильный код

## **Неверные имена**

Изменим в ElementName существующие имена объектов на тех, которых нет. В итоге у нас получится следующий код (см. также Рисунок 2):

<StackPanel>

<TextBox x:Name="t1"/>

<TextBox x:Name="t2" Text="{Binding ElementName=eeyey, Path=Text}" />

<Slider x:Name="slider1" />

<Slider x:Name="slider2" Value="{Binding ElementName=sssss, Path=Value}" />

</StackPanel>

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 — Код с неправильными именами

Запускаем проект и заходим в Сбои привязки, далее высветится окно Ошибки привязки XAML, где в описании будет написано: «Не удаётся найти источник ElementName». Контекст данных будет null (Рисунок 3).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 — Сбои привязки

В окне Вывод укажет об ошибке (Рисунок 4).

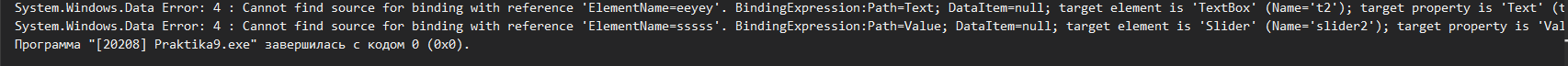


Рисунок 4 — Ошибки в окне Вывод

## **Неверные свойства**

Вернём на место те же названия, что и были до этого, но теперь поменяем на некорректные имена свойств, которых нет у указанных объектов.

Код примерно такой (см. также Рисунок 5):

<StackPanel>

<TextBox x:Name="t1"/>

<TextBox x:Name="t2" Text="{Binding ElementName=t1, Path=Queque}" />

<Slider x:Name="slider1" />

<Slider x:Name="slider2" Value="{Binding ElementName=slider1, Path=Text}" />

</StackPanel>

Изображение выглядит как текст, внутренний, экран, снимок экрана

Автоматически созданное описание

Рисунок 5 — Код с неверными свойствами

Запускаем проект и заходим в Сбои привязки, далее высветится окно Ошибки привязки XAML, в описании будет сказано «Свойство Queque не найдено для объекта типа TextBox». Только в этом случае контекст не будет равен null (Рисунок 6).

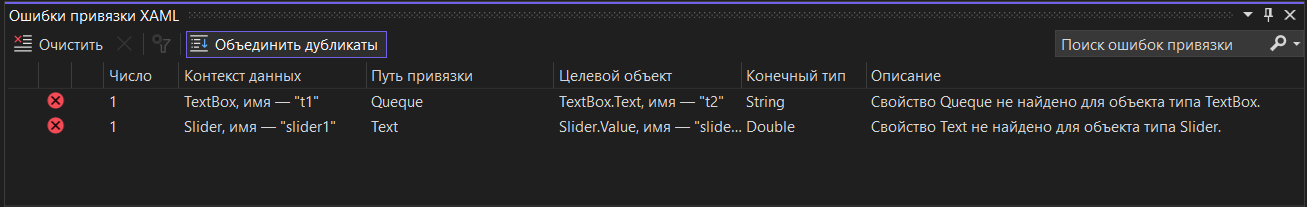


Рисунок 6 — Ошибки в Сбои привязки

В окне Вывод укажет об ошибке (Рисунок 7).

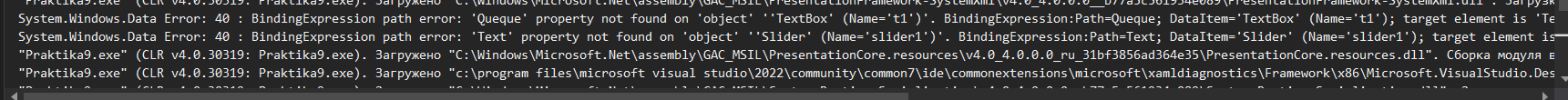


Рисунок 7 — Ошибки в окне Вывод

## **Неверные имена и свойства**

Попробуем посмотреть, что будет если ввести неправильные и имена, и значения (см. также Рисунок 8).

<StackPanel>

<TextBox x:Name="t1"/>

<TextBox x:Name="t2" Text="{Binding ElementName=t, Path=Queque}" />

<Slider x:Name="slider1" />

<Slider x:Name="slider2" Value="{Binding ElementName=sli, Path=Text}" />

</StackPanel>

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 — Код с неверными именами и свойствами

Запускаем проект и заходим в Сбои привязки, должно высветиться окно Ошибки привязки XAML. По итогу выйдет абсолютно идентичное окно, что и при простом неправильном указании имени объекта (Рисунок 9).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 — Ошибки в Сбои привязки

В окне Вывод укажет на ошибку (Рисунок 10).



Рисунок 10 — Ошибки в окне Вывод

# **ЗАДАНИЕ 2**

Сначала введём код, который уже дан к заданию. Выглядит он следующим образом (см. также Рисунок 11):

<StackPanel>

<TextBox x:Name="t1"/>

<TextBox x:Name="t2" Text="{Binding ElementName=t1, Path=Text}" />

<Slider x:Name="slider1" />

<Slider x:Name="slider2" Value="{Binding ElementName=slider1, Path=Value}" />

</StackPanel>

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 — Исходный код

Запустим программу и узнаем, почему данный код некорректный. Перед нами два текстовых поля и два слайдера (Рисунок 12). Оба слайдера при движении имеют ведут себя абсолютно одинаково, хоть первый тяни, хоть второй. С текстовыми полями тоже не лучше ситуация, сначала вводится текст в первом поле, оно сразу появляется во втором. Если же что-то изменить во втором поле, то оно при нажатии на первое поле, в нём сразу появится текст из второго поля (Рисунок 13).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 — Работа исходного кода

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13 — Некорректная работа кода

Задумка кода была в том, что при вводе текста в первое поле, оно отображалось во втором. Даже если что-то напишут во втором, то при переключении на первое поле, эти изменения не будут учтены. Все изменения и ввод будут идти от первого поля. Слайдеры же работают так, что если потянуть первый, то за ним потянется второй. Если же потянуть второй, то за ним не потянется первый слайдер.

На данный момент указан режим привязки Default, из-за чего был указан Two Way к объектам.

Исправим код (см. также Рисунок 14):

<StackPanel>

<TextBox x:Name="t1"/>

<TextBox x:Name="t2" Text="{Binding ElementName=t1, Path=Text, Mode=OneWay}" />

<Slider x:Name="slider1" />

<Slider x:Name="slider2" Value="{Binding ElementName=slider1, Path=Value, Mode=OneWay}" />

</StackPanel>

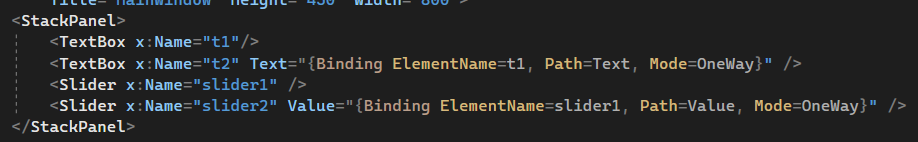


Рисунок 14 — Измененный код

Используя OneWay, мы обновляем целевое свойство при изменении исходного. При запуске проекта, двигая первый слайдер, за ним пойдет и второй слайдер. Если потянуть второй слайдер, то первый за ним не пойдёт (Рисунок 15).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 15 — Работа слайдеров

При вводе текста в первое поле, во втором текст будет абсолютно такой же (Рисунок 16). При вводе текста во второе поле, в первом не произойдёт никаких изменений (Рисунок 17).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16 — Работа текстового поля

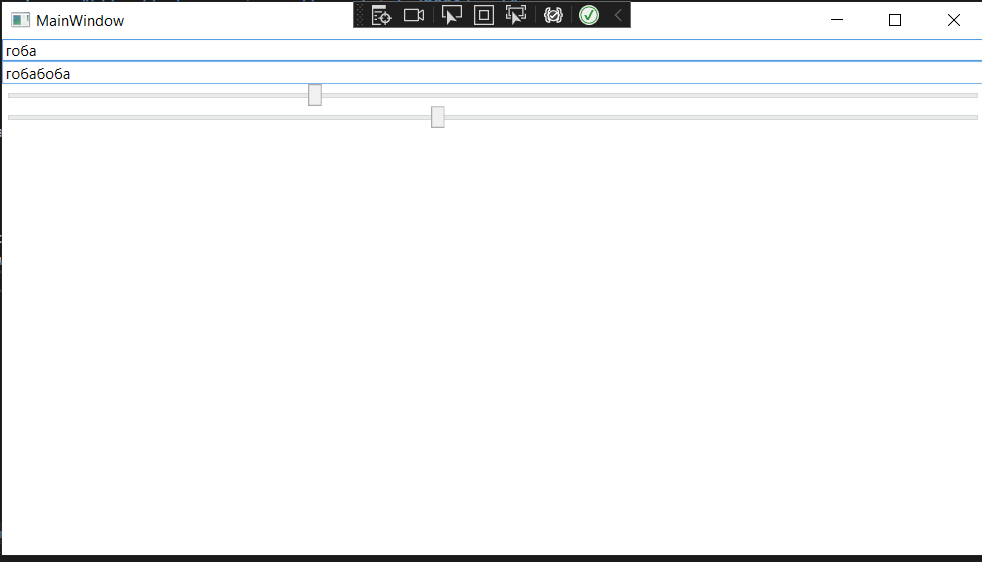


Рисунок 17 — Взаимосвязь текстовых полей

# **ЗАДАНИЕ 3**

Используя исходный код, нам нужно добавить текстовое поле, в которое мы будем вводить размер шрифта. В общей сложности нам надо сделать так, чтобы все элементы были между собой связаны. При перетягивании слайдера менялся сам текст и указывалось текущее значение в текстовом поле. При вводе размера шрифта менялось положение слайдера и размер текста.

Код к этому заданию (см. также Рисунок 18):

<StackPanel>

<TextBox x:Name="textBox1" Text="{Binding ElementName=SliderFontSize, Path=Value, Mode=TwoWay}"></TextBox>

<Slider Minimum="8" Maximum="30" x:Name="SliderFontSize" Value="{Binding ElementName=textBox1, Path=Text, Mode=OneWay}"></Slider>

<TextBlock x:Name="Message" FontSize="{Binding ElementName=SliderFontSize, Path=Value}">Пример WPF-приложения для демонстрации привязки данных</TextBlock>

</StackPanel>

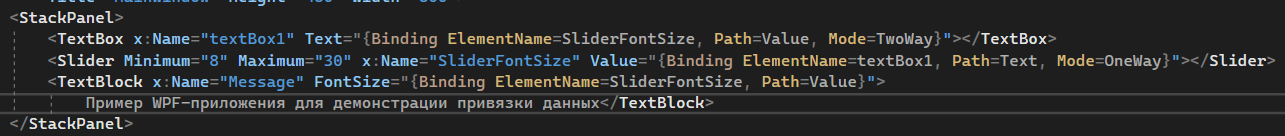


Рисунок 18 — Полученный код

По итогу получился взаимосвязанный интерфейс, где при изменении параметров меняются свойства и значения других объектов (Рисунок 19, Рисунок 20).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 19 — Изменение объектов при использовании слайдера

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 20 — Изменение объектов при вводе размера в поле

# **ЗАДАНИЕ 4**

В данном задании нужно использовать предыдущее задание, где мы делали графический редактор. Только желательно уменьшить количество обработчиков событий.

Код задания выглядит следующим образом (см. также Рисунок 21, Рисунок 22, Рисунок 23, Рисунок 24):

<Canvas>

<Slider x:Name="heights" Height="20" Width="150" Minimum="1" Maximum="60" Canvas.Left="0" Canvas.Top="0" ValueChanged="heights\_ValueChanged"></Slider>

<ComboBox x:Name="colors" SelectedIndex="0" Canvas.Top="0" Canvas.Left="170" SelectionChanged="ComboBox\_SelectionChanged">

<ComboBoxItem Content="Red"></ComboBoxItem>

<ComboBoxItem Content="Blue"></ComboBoxItem>

<ComboBoxItem Content="Yellow"></ComboBoxItem>

<ComboBoxItem Content="Gray"></ComboBoxItem>

<ComboBoxItem Content="Green"></ComboBoxItem>

</ComboBox>

<RadioButton GroupName="ModeDraw" x:Name="option\_one" Canvas.Right="10" Canvas.Top="0" Click="option\_one\_Click">рисование</RadioButton>

<RadioButton GroupName="ModeDraw" x:Name="option\_two" Canvas.Right="10" Canvas.Top="15" Click="option\_two\_Click">редактирование</RadioButton>

<RadioButton GroupName="ModeDraw" x:Name="option\_three" Canvas.Right="10" Canvas.Top="30" Click="option\_three\_Click">удаление</RadioButton>

<InkCanvas EditingMode="Ink" Canvas.Left="0" Canvas.Top="30" x:Name="draw" Height="300" Width="700" >

</InkCanvas>

</Canvas>

DrawingAttributes inkDA = new DrawingAttributes();

public MainWindow()

{

InitializeComponent();

inkDA.Color = Colors.Red;

draw.DefaultDrawingAttributes = inkDA;

}

private void ComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ComboBoxItem comboBox = (ComboBoxItem)colors.SelectedItem;

string res = comboBox.Content.ToString();

switch (res)

{

case "Red":

inkDA.Color = Colors.Red;

break;

case "Blue":

inkDA.Color = Colors.Blue;

break;

case "Green":

inkDA.Color = Colors.SpringGreen;

break;

case "Yellow":

inkDA.Color = Colors.Yellow;

break;

case "Gray":

inkDA.Color = Colors.Gray;

break;

}

}

private void option\_one\_Click(object sender, RoutedEventArgs e)

{

draw.EditingMode = InkCanvasEditingMode.Ink;

}

private void option\_two\_Click(object sender, RoutedEventArgs e)

{

draw.EditingMode = InkCanvasEditingMode.Select;

}

private void option\_three\_Click(object sender, RoutedEventArgs e)

{

draw.EditingMode = InkCanvasEditingMode.EraseByStroke;

}

private void heights\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

inkDA.Height = heights.Value;

}

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 21 — Элементы графического редактора

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 22 — Обработчики

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 23 — Обработчики

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 24 — Обработчики

Таким образом редактор будет работать и менять состояние инструментов, ширину и цвет кисти (Рисунок 25). Также можно переключаться между режимами. Это удаление, рисование и редактирование (Рисунок 26, Рисунок 27, Рисунок 28).

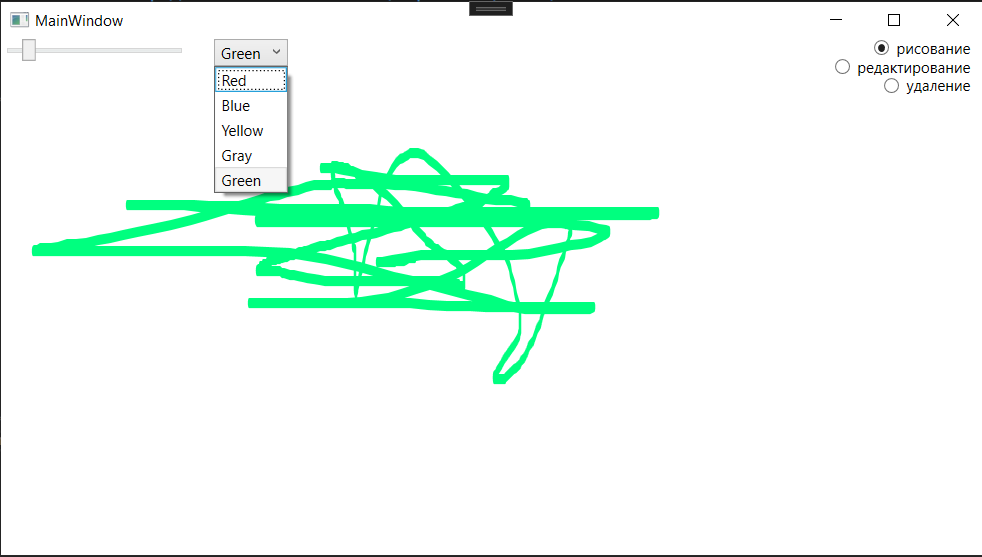


Рисунок 25 — Графический редактор

Изображение выглядит как стрела

Автоматически созданное описание

Рисунок 26 — Режим рисования

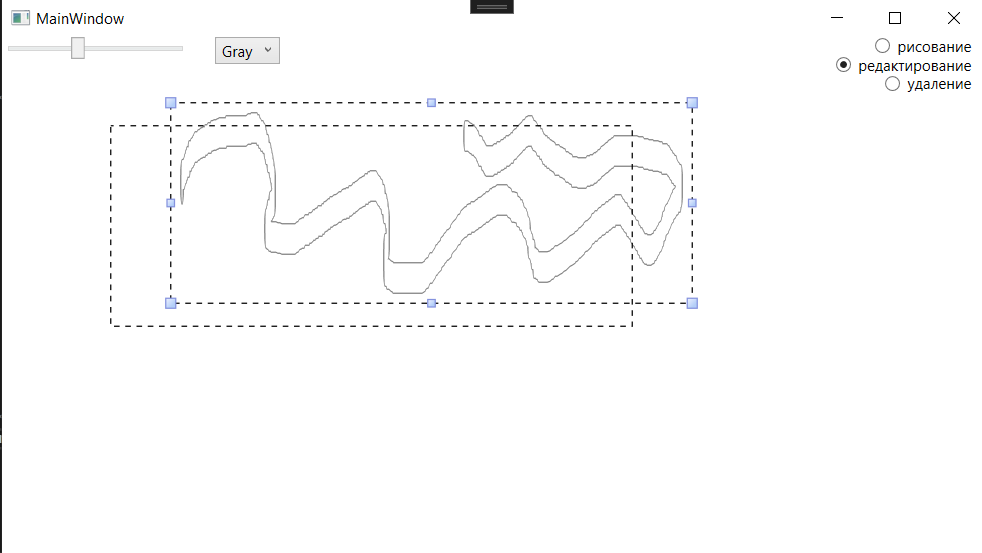


Рисунок 27 — Режим редактирования



Рисунок 28 — Режим удаления

# **ВЫВОД**

Вывод: мы научились пользоваться привязкой данных, отказались от обработчиков событий, узнали о составляющих и режимах привязки.