УО «Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

По дисциплине: “Технологии разработки интерфейса программных средств„

Тема: Привязка данных.

Вариант 6

Выполнил: Комиссаров А. Е.

Брест 2021 Группа ПО-7, 2 курс.

Проверила: Войцехович О.Ю.

Цель: Ознакомиться с принципами использования привязки данных в WPF.

Ход работы:

**Задание 1.**  Проверьте реакцию среды разработки на неверные значения параметров ElementName и Path. Проанализируйте сообщения, которые выводятся в окне вывода (Вид → Вывод) при построении и при запуске приложения.

После смены значений ElementName и Path со стандартных на свои (в конце добавлены три единицы), ползунок перестаёт менять размер шрифта, а при запуске программы в окне вывода выводит сообщение, которое гласит:

«*System.Windows.Data Error: 4 : Cannot find source for binding with reference 'ElementName=****SliderFontSize111****'. BindingExpression:Path=****Value111****; DataItem=null; target element is 'TextBlock' (Name='Message'); target property is 'FontSize' (type 'Double')*»

Что в переводе с английского означает «невозможно найти источник для привязки с названием SliderFontSize111». После смены с заданного обратно на стандартное значение программа работает как следует (ползунок меняет размер шрифта).

**Задание 2.**  Запустите приложение со следующим XAML-кодом:

<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}" />

Определите различие в поведении полей t1 и t2 и модифицируйте код, чтобы устранить это различие.

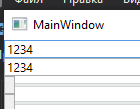
Различие состоит в том, что обновление значения в паре текстов происходит касательно только одно из них, например при вводе текста в поле1, поле2 обновляется сразу же:

Рис. 1 – поле1

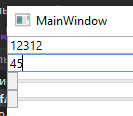
Но при вводе текста в поле2, поле1 остаётся таким же пока не закончится ввод в поле2:

Рис. 2 – поле2

Решением данной проблемы является привязка текстов с обоих сторон:

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

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

<Slider x:Name="slider1" />

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

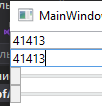
Теперь ввод текста синхронизирован с обеих сторон как в поле1, так и в поле2:

Рис. 3 – поле2 (теперь ввод одновременный)

**Задание 3.** Дополните пример №2 текстовым полем ввода TextBox, в котором пользователь может ввести размер шрифта, и задайте выражения привязки таким образом, чтобы значение ползунка, текст текстового поля и размер шрифта текстового блока соответствовали друг другу.

Текст программы:

<Slider Minimum="8" Maximum="30" x:Name="SliderFontSize"></Slider>

<TextBlock x:Name="Message" FontSize="{Binding ElementName=SliderFontSize, Path=Value}">

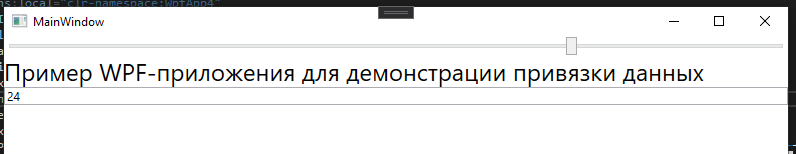
Пример WPF-приложения для демонстрации привязки данных

</TextBlock>

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

Результат выполнения программы:

Рис. 4 – Задание 3 (стандартное значение)

Рис. 5 – Смена размера текста путём ввода числа 24 в текстовое поле

**Задание 4.** Модифицируйте приложения, разработанные в предыдущей лабораторной работе: удалите как можно больше обработчиков событий и реализуйте ту же функциональность приложения с помощью привязки данных.

Код программы:

Содержимое файла MainWindow.XAML:

<Window x:Class="WpfApp3.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:WpfApp3"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="600" ResizeMode="NoResize">

**<Window.Resources>**

<x:Array x:Key="MyEditingModes" Type="{x:Type InkCanvasEditingMode}">

<x:Static Member="InkCanvasEditingMode.Ink"/>

<x:Static Member="InkCanvasEditingMode.EraseByPoint"/>

<x:Static Member="InkCanvasEditingMode.EraseByStroke"/>

</x:Array>

<x:Array x:Key="MyDrawingAttributes" Type="{x:Type DrawingAttributes}">

<DrawingAttributes Color="Red" Width="2"/>

<DrawingAttributes Color="Green" Width="2"/>

<DrawingAttributes Color="Blue" Width="2"/>

</x:Array>

**</Window.Resources>**

<Grid>

<InkCanvas Height="420" x:Name="Drawboard" DefaultDrawingAttributes="{Binding ElementName=lbColors, Path=SelectedValue}" EditingMode="{Binding ElementName=lbEditingModes, Path=SelectedValue}"/>

<Menu Background="White" Width="600" Height="20" Margin="0 -390 0 0">

<MenuItem Header="\_Цвет кисти" ToolTip="Выбор цвета.">

<ListBox x:Name="lbColors" ItemsSource**="{StaticResource MyDrawingAttributes}**">

<ListBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="**{Binding Path=Color}**"></TextBlock>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</MenuItem>

<MenuItem Header="\_Толщина кисти">

<Slider Name="Slider111" Height="25" Width="100" Minimum="1" Maximum="15" Value="2" ValueChanged="Slider111\_ValueChanged" />

</MenuItem>

<MenuItem Header="\_Режим работы">

<ListBox x:Name="lbEditingModes" ItemsSource="**{StaticResource MyEditingModes}**" />

</MenuItem>

</Menu>

</Grid></Window>

Функции файла MainWindow.xaml.CS:

private void Slider111\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

Drawboard.DefaultDrawingAttributes.Width = Slider111.Value;

Drawboard.DefaultDrawingAttributes.Height = Slider111.Value;

}

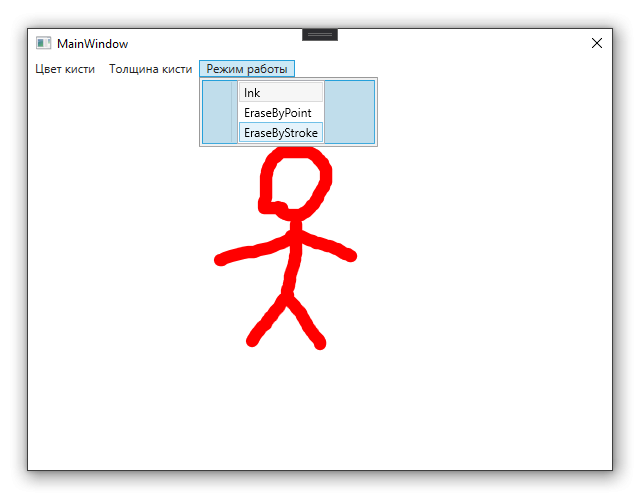
Результат работы программы:

Рис. 6 – Программа работает как и в предыдущей лабораторной, но уже на принципах использования привязки данных WPF.

Вывод: Я ознакомился с принципами использования привязки данных в WPF.