Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчет по ТРПО. Лабораторная работа   
**Привязка данных**

Выполнили: Максимов К. Э.

Группа: ПР-31

Преподаватель: Бушуева Е. В.

2024

Задание 3

<Window x:Class="Lab\_4\_WPF.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:Lab\_4\_WPF"

mc:Ignorable="d"

Title="MainWindow" Height="200" Width="400">

<StackPanel>

<TextBox x:Name="t1"/>

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

<Slider x:Name="slider1" Minimum="1" Maximum="100"/>

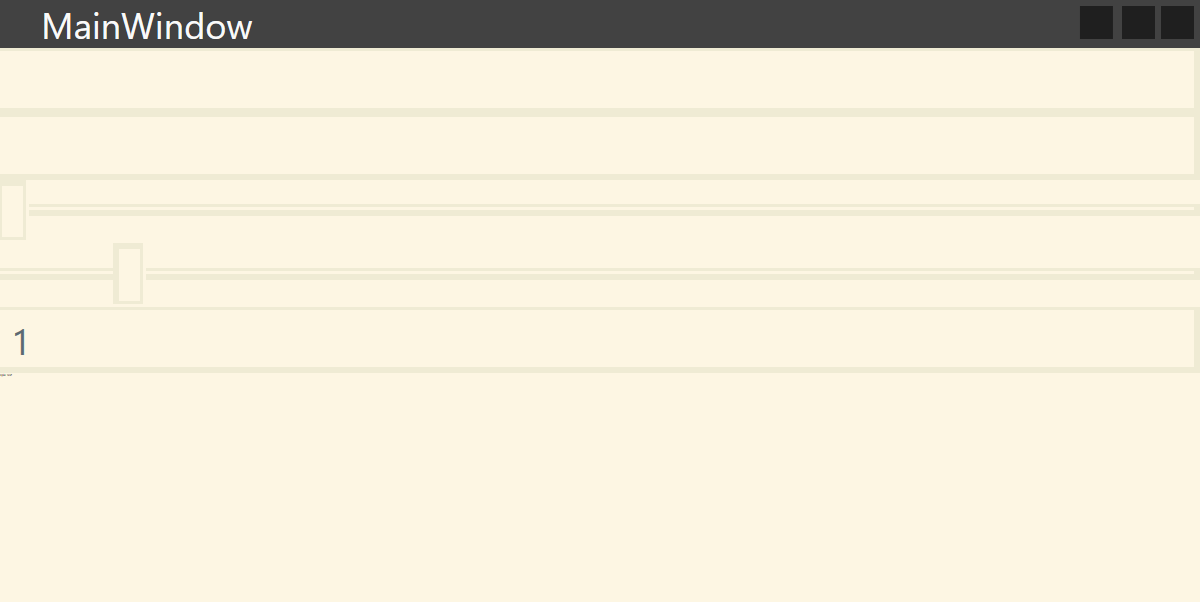
<Slider x:Name="slider2" Value="{Binding ElementName=slider1, Path=Value}"/>

<TextBox x:Name="fontSizeInput" Text="{Binding ElementName=slider1, Path=Value, Mode=TwoWay}"/>

<TextBlock Text="Sample Text" FontSize="{Binding ElementName=slider1, Path=Value}"/>

</StackPanel>

</Window>



Задание 4

<Window x:Class="Lab\_3\_WPF.MainWindow"

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

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

Title="MainWindow" Height="450" Width="800">

<DockPanel>

<Menu DockPanel.Dock="Top">

<MenuItem Header="Меню">

<MenuItem Header="Изменить цвет фона" Command="{Binding ChangeColorCommand}"/>

<MenuItem Header="Получить информацию о разработчике" Command="{Binding AboutDeveloperCommand}"/>

<MenuItem Header="Выход" Command="{Binding ExitCommand}"/>

</MenuItem>

</Menu>

<ToolBar DockPanel.Dock="Top">

<Button Content="Изменить цвет фона" Command="{Binding ChangeColorCommand}" />

<Button Content="Получить информацию о разработчике" Command="{Binding AboutDeveloperCommand}" />

<Button Content="Выход" Command="{Binding ExitCommand}" />

</ToolBar>

<StatusBar DockPanel.Dock="Bottom">

<TextBlock Text="{Binding StatusBarText}"/>

</StatusBar>

<Grid Background="{Binding BackgroundColor}" />

</DockPanel>

</Window>

using System.ComponentModel;

using System.Runtime.CompilerServices;

using System.Windows;

using System.Windows.Input;

namespace Lab\_3\_WPF

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

DataContext = new MainViewModel();

}

}

public class MainViewModel : INotifyPropertyChanged

{

public MainViewModel()

{

ChangeColorCommand = new RelayCommand(ChangeColor);

AboutDeveloperCommand = new RelayCommand(AboutDeveloper);

ExitCommand = new RelayCommand(Exit);

}

public ICommand ChangeColorCommand { get; }

public ICommand AboutDeveloperCommand { get; }

public ICommand ExitCommand { get; }

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

private void ChangeColor()

{

BackgroundColor = "LightBlue";

StatusBarText = "Цвет фона изменен";

}

private void AboutDeveloper()

{

MessageBox.Show("Разработчик: Кирилл");

StatusBarText = "Информация о разработчике";

}

private void Exit()

{

Application.Current.MainWindow.Close();

}

private string \_backgroundColor;

public string BackgroundColor

{

get { return \_backgroundColor; }

set

{

\_backgroundColor = value;

OnPropertyChanged();

}

}

private string \_statusBarText;

public string StatusBarText

{

get { return \_statusBarText; }

set

{

\_statusBarText = value;

OnPropertyChanged();

}

}

}

public class RelayCommand : ICommand

{

private readonly Action \_action;

public RelayCommand(Action action)

{

\_action = action;

}

public event EventHandler CanExecuteChanged;

public bool CanExecute(object parameter)

{

return true;

}

public void Execute(object parameter)

{

\_action();

}

}

}



<Window x:Class="WpfApp\_2.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:WpfApp\_2"

mc:Ignorable="d"

Title="Графический редактор" Height="450" Width="800">

<Window.Resources>

<local:ColorConverter x:Key="ColorConverter"/>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal" Grid.Row="0">

<ComboBox Name="colorComboBox" Width="120" Height="20" Margin="10" MaxDropDownHeight="200" ItemsSource="{Binding AvailableColors}" SelectedItem="{Binding SelectedColor}"/>

<Slider Name="sizeSlider" Width="120" Margin="10" Minimum="1" Maximum="10" Value="{Binding SelectedSize}"/>

<RadioButton Content="Рисование" IsChecked="{Binding IsDrawMode}" Margin="10"/>

<RadioButton Content="Редактирование" IsChecked="{Binding IsEditMode}" Margin="10"/>

<RadioButton Content="Удаление" IsChecked="{Binding IsDeleteMode}" Margin="10"/>

</StackPanel>

<Canvas Name="myCanvas" Background="White" MouseDown="Canvas\_MouseDown" MouseMove="Canvas\_MouseMove" MouseUp="myCanvas\_MouseUp" Grid.Row="1">

<Canvas.Resources>

<Style TargetType="Ellipse">

<Setter Property="Stroke" Value="{Binding SelectedColor, Converter={StaticResource ColorConverter}}"/>

<Setter Property="StrokeThickness" Value="{Binding SelectedSize}"/>

</Style>

</Canvas.Resources>

</Canvas>

</Grid>

</Window>

public partial class MainWindow : Window, INotifyPropertyChanged

{

private bool \_isDrawMode = true;

private bool \_isEditMode = false;

private bool \_isDeleteMode = false;

private Shape \_currentShape;

private Point \_startPoint;

private string \_selectedColor;

private double \_selectedSize;

public event PropertyChangedEventHandler PropertyChanged;

public bool IsDrawMode

{

get { return \_isDrawMode; }

set { \_isDrawMode = value; OnPropertyChanged(); }

}

public bool IsEditMode

{

get { return \_isEditMode; }

set { \_isEditMode = value; OnPropertyChanged(); }

}

public bool IsDeleteMode

{

get { return \_isDeleteMode; }

set { \_isDeleteMode = value; OnPropertyChanged(); }

}

public string SelectedColor

{

get { return \_selectedColor; }

set { \_selectedColor = value; OnPropertyChanged(); }

}

public double SelectedSize

{

get { return \_selectedSize; }

set { \_selectedSize = value; OnPropertyChanged(); }

}

public List<string> AvailableColors { get; } = new List<string>();

public MainWindow()

{

InitializeComponent();

DataContext = this;

foreach (var prop in typeof(System.Windows.Media.Colors).GetProperties())

{

AvailableColors.Add(prop.Name);

}

SelectedColor = "Black";

SelectedSize = 1;

}

private void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

private void Canvas\_MouseDown(object sender, MouseButtonEventArgs e)

{

\_startPoint = e.GetPosition(myCanvas);

if (IsDrawMode)

{

if (SelectedColor != null)

{

\_currentShape = new Ellipse

{

Width = 0,

Height = 0,

};

Canvas.SetLeft(\_currentShape, \_startPoint.X);

Canvas.SetTop(\_currentShape, \_startPoint.Y);

myCanvas.Children.Add(\_currentShape);

}

else

{

MessageBox.Show("Пожалуйста, выберите цвет");

}

}

else if (IsEditMode || IsDeleteMode)

{

\_currentShape = null;

foreach (var child in myCanvas.Children)

{

if (child is Ellipse ellipse)

{

var mousePos = e.GetPosition(myCanvas);

var left = Canvas.GetLeft(ellipse);

var top = Canvas.GetTop(ellipse);

var right = left + ellipse.Width;

var bottom = top + ellipse.Height;

if (mousePos.X >= left && mousePos.X <= right && mousePos.Y >= top && mousePos.Y <= bottom)

{

\_currentShape = ellipse;

break;

}

}

}

if (\_currentShape != null)

{

if (IsEditMode)

{

}

else if (IsDeleteMode)

{

myCanvas.Children.Remove(\_currentShape);

}

}

}

}

private void Canvas\_MouseMove(object sender, MouseEventArgs e)

{

if (e.LeftButton == MouseButtonState.Pressed && \_currentShape != null)

{

if (IsDrawMode)

{

var pos = e.GetPosition(myCanvas);

var width = pos.X - \_startPoint.X;

var height = pos.Y - \_startPoint.Y;

if (Keyboard.IsKeyDown(Key.LeftShift) || Keyboard.IsKeyDown(Key.LeftCtrl))

{

var size = System.Math.Max(width, height);

\_currentShape.Width = size < 0 ? 0 : size;

\_currentShape.Height = size < 0 ? 0 : size;

}

else

{

\_currentShape.Width = width < 0 ? 0 : width;

\_currentShape.Height = height < 0 ? 0 : height;

}

}

else if (IsEditMode)

{

var pos = e.GetPosition(myCanvas);

var offsetX = pos.X - \_startPoint.X;

var offsetY = pos.Y - \_startPoint.Y;

Canvas.SetLeft(\_currentShape, Canvas.GetLeft(\_currentShape) + offsetX);

Canvas.SetTop(\_currentShape, Canvas.GetTop(\_currentShape) + offsetY);

\_startPoint = pos;

}

}

}

private void myCanvas\_MouseUp(object sender, MouseButtonEventArgs e)

{

if (IsEditMode)

{

\_currentShape = null;

}

}

}

public class ColorConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is string colorName)

{

try

{

Color color = (Color)System.Windows.Media.ColorConverter.ConvertFromString(colorName);

return new SolidColorBrush(color);

}

catch (FormatException)

{

}

}

return new SolidColorBrush(Colors.Black);

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

