**27.Компановка и элементы управления в WPF**

Задание №1

В работе 24 «Язык разметки XAML». Измените

интерфейс проекта WPF – XmlTaskWPF следующим образом: Добавьте возможность скрывать функциональность под ЭУ Expander; предложите выбор вывода списка в качестве Компонеты с использованием RadioButtons; добавьте вывод списка элементов в TreeView, ListBox, TextBox.

Листинг программы:

<Window x:Class="z1.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:z1"

mc:Ignorable="d"

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

<Grid x:Name="mainGrid">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="3\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition />

</Grid.RowDefinitions>

<ScrollViewer Grid.Column="0"

Grid.Row="0" Grid.RowSpan="4">

<StackPanel>

<StackPanel Grid.Column="0" Grid.Row="0">

<Button x:Name="buttonOpenFile"

Content="Открыть файл"

Command="{Binding Open}"/>

</StackPanel>

<Expander Grid.Column="0"

Grid.Row="1"

Header="Добавление машины">

<StackPanel>

<StackPanel Margin="10,5">

<Label Content="Марка:"/>

<TextBox Text="{Binding Stamp, UpdateSourceTrigger=PropertyChanged}"/>

</StackPanel>

<StackPanel Margin="10,5">

<Label Content="Год выпуска:"/>

<TextBox Text="{Binding YearRelease, UpdateSourceTrigger=PropertyChanged}"/>

</StackPanel>

<StackPanel Margin="10,5">

<Label Content="Срок аренды стоянки(дни):"/>

<TextBox Text="{Binding ParkingRentalPeriod, UpdateSourceTrigger=PropertyChanged}"

MinWidth="150"/>

</StackPanel>

<Button Content="Добавить"

Command="{Binding Add}"/>

</StackPanel>

</Expander>

<Expander Grid.Column="0"

Grid.Row="2"

Header="Функцирнальность">

<StackPanel>

<StackPanel Margin="10,5">

<Label Content="Id:"/>

<TextBox x:Name="textBoxId" MinWidth="150"/>

</StackPanel>

<Button Content="Найти"

Command="{Binding ToFind}"

CommandParameter="{Binding ElementName=textBoxId, Path=Text}"/>

<Button Content="Удалить"

Command="{Binding Delete}"

CommandParameter="{Binding ElementName=textBoxId, Path=Text}"/>

</StackPanel>

</Expander>

<Expander Grid.Column="0"

Grid.Row="3"

Header="Компоненты">

<StackPanel>

<GroupBox Header="Способ компоновки">

<StackPanel>

<RadioButton x:Name="rabioButtonTextBox" Command="{Binding CheckedComp}"

CommandParameter="{Binding ElementName=rabioButtonTextBox,Path=Content}"

Margin="0,5"

Content="TextBox"/>

<RadioButton x:Name="rabioButtonTreeView" Command="{Binding CheckedComp}"

CommandParameter="{Binding ElementName=rabioButtonTreeView,Path=Content}"

Margin="0,5"

Content="TreeView"/>

<RadioButton x:Name="rabioButtonListBox" Command="{Binding CheckedComp}"

CommandParameter="{Binding ElementName=rabioButtonListBox,Path=Content}"

IsChecked="True"

Margin="0,5"

Content="ListBox"/>

</StackPanel>

</GroupBox>

</StackPanel>

</Expander>

</StackPanel>

</ScrollViewer>

<ScrollViewer Grid.Column="1"

Grid.Row="0"

Grid.RowSpan="4"

x:Name="s"

Content="{Binding ContentComp}">

</ScrollViewer>

</Grid>

</Window>

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using z1.Commands;

using z1.Models;

using z1.Share;

namespace z1.ViewModels

{

internal class MainWindowsViewModel : INotifyPropertyChanged

{

private string \_path;

private XmlDocumentWorker \_xmlDocWorker;

private string \_stamp;

private bool IsListBox = true;

public string Stamp

{

get { return \_stamp; }

set

{

\_stamp = value;

OnPropertyChanged("Stamp");

}

}

private int \_yearRelease;

public string YearRelease

{

get { return \_yearRelease.ToString(); }

set

{

try

{

\_yearRelease = Convert.ToInt32(value);

OnPropertyChanged("YearRelease");

}

catch (FormatException)

{

}

}

}

private int \_parkingRentalPeriod;

public string ParkingRentalPeriod

{

get { return \_parkingRentalPeriod.ToString(); }

set

{

try

{

\_parkingRentalPeriod = Convert.ToInt32(value);

OnPropertyChanged("ParkingRentalPeriod");

}

catch (FormatException)

{

}

}

}

private string \_readData;

public string ReadData

{

get

{

return \_readData;

}

set

{

\_readData = value;

OnPropertyChanged("ReadData");

}

}

private List<Auto> \_autos;

public List<Auto> Autos

{

get

{

return \_autos;

}

set

{

\_autos = value;

OnPropertyChanged("Autos");

}

}

private Control \_contentComp;

public Control ContentComp

{

get

{

return \_contentComp;

}

set

{

\_contentComp = value;

OnPropertyChanged("ContentComp");

}

}

private Command \_open;

public Command Open

{

get

{

return \_open ??

(\_open = new Command(obj => OpenFile()));

}

}

private Command \_add;

public Command Add

{

get

{

return \_add ??

(\_add = new Command(obj =>

{

AddAuto(\_stamp, \_yearRelease, \_parkingRentalPeriod);

}));

}

}

private Command \_delete;

public Command Delete

{

get

{

return \_delete ??

(\_delete = new Command(obj =>

{

string id = obj as string;

if (id != null)

{

DeleteAuto(id);

}

}));

}

}

private Command \_toFind;

public Command ToFind

{

get

{

return \_toFind ?? (\_toFind = new Command(obj =>

{

if (\_xmlDocWorker.Doc != null)

{

if (obj != null)

{

string id = obj as string;

Auto auto = ToFindAuto(id);

if (auto != null)

{

MessageBox.Show(

$"{auto}",

"Найденный автомобиль",

MessageBoxButton.OK);

}

else

{

MessageBox.Show(

$"Автомобиль под ID:{id} не найден",

"Не найден автомобиль",

MessageBoxButton.OK);

}

}

else

{

MessageBox.Show(

"Не введенно значение ID",

"ID",

MessageBoxButton.OK);

}

}

else

{

MessageBox.Show(

"Для поиска должен быть открат файл.",

"Файл не открат",

MessageBoxButton.OK);

}

}));

}

}

private Command \_checkedComp;

public Command CheckedComp

{

get

{

return \_checkedComp ??

(\_checkedComp = new Command(obj =>

{

string nameElem = obj as string;

ChangeLayout(nameElem);

}));

}

}

public MainWindowsViewModel()

{

\_xmlDocWorker = new XmlDocumentWorker();

Autos = new List<Auto>();

ChangeLayout("");

}

private void OpenFile()

{

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

{

//ReadData = string.Empty;

\_path = openFileDialog.FileName;

\_xmlDocWorker.Load(\_path);

Autos = \_xmlDocWorker.GetAll();

ReadData = GetText();

}

}

private void AddAuto(string stamp,

int yearRelease, int parkingRentalPeriod)

{

if (\_xmlDocWorker.Doc != null)

{

Auto auto = new Auto(0, stamp, yearRelease,

parkingRentalPeriod);

\_xmlDocWorker.Add(auto);

Autos = \_xmlDocWorker.GetAll();

ReadData = GetText();

}

else

{

MessageBox.Show(

"Для начало откройте файл",

"Файл",

MessageBoxButton.OK);

}

}

private void DeleteAuto(string id)

{

if (\_xmlDocWorker.Doc != null)

{

try

{

int idInt = Convert.ToInt32(id);

\_xmlDocWorker.Delete(idInt);

Autos = \_xmlDocWorker.GetAll();

ReadData = GetText();

}

catch (FormatException)

{

MessageBox.Show(

"Неправельный формат поля",

"Формат",

MessageBoxButton.OK);

}

}

else

{

MessageBox.Show(

"Для начало откройте файл",

"Файл",

MessageBoxButton.OK);

}

}

private string GetText()

{

if (\_autos != null)

{

if (\_autos.Count != 0)

{

\_readData = string.Empty;

foreach (Auto a in \_autos)

{

\_readData += a.ToString();

}

return \_readData;

}

}

return string.Empty;

}

private Auto ToFindAuto(string id)

{

return \_xmlDocWorker.DindBy(id);

}

private void ChangeLayout(string nameComp)

{

switch (nameComp)

{

case "TextBox":

ContentComp = null;

Binding bindingTB = new Binding("ReadData")

{

Source = this

};

TextBox tb = new TextBox();

tb.SetBinding(TextBox.TextProperty, bindingTB);

ContentComp = tb;

break;

case "TreeView":

ContentComp = null;

Binding bindingTVAutos = new Binding("Autos")

{

Source = this

};

TreeView tv = new TreeView();

tv.SetBinding(TreeView.ItemsSourceProperty, bindingTVAutos);

ContentComp = tv;

break;

default:

ContentComp = null;

Binding bindingLBAutos = new Binding("Autos")

{

Source = this

};

Binding bindingLBSelectItem = new Binding("SelectedItem")

{

Source = this

};

ListBox lb = new ListBox();

lb.SetBinding(ListBox.ItemsSourceProperty, bindingLBAutos);

ContentComp = lb;

break;

}

}

public event PropertyChangedEventHandler? PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

}

Анализ результатов:

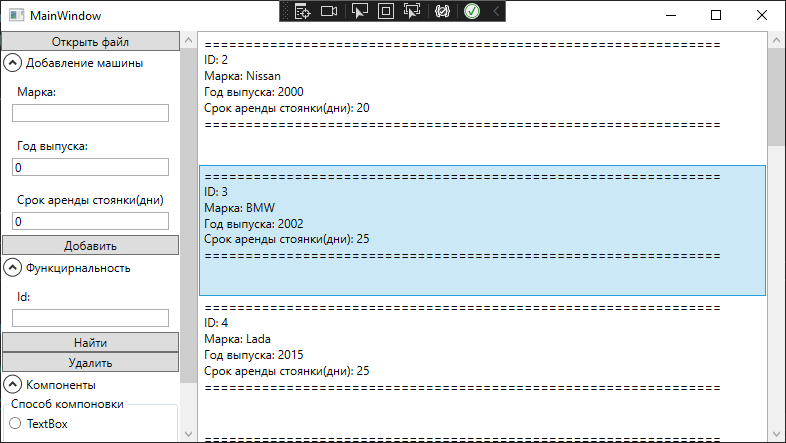


Рисунок 27.1 – Результат выполнения программы задание №1