**Цель работы:** научиться применять шаблон MVVM для создания программной системы.

**Вариант 11**

**Выполнение заданий:**

*Задание 1*: Изучить примеры реализации шаблона MVVM. Применить шаблон MVVM для реализации проекта.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace MVVM

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

DataContext = new ApplicationViewModel();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel;

using System.Runtime.CompilerServices;

namespace MVVM

{

public class Phone : INotifyPropertyChanged

{

private string title;

private string company;

private int price;

public string Title

{

get { return title; }

set

{

title = value;

OnPropertyChanged("Title");

}

}

public string Company

{

get { return company; }

set

{

company = value;

OnPropertyChanged("Company");

}

}

public int Price

{

get { return price; }

set

{

price = value;

OnPropertyChanged("Price");

}

}

public event PropertyChangedEventHandler PropertyChanged;

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

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using System.Collections.ObjectModel;

namespace MVVM

{

public class ApplicationViewModel : INotifyPropertyChanged

{

private Phone selectedPhone;

public ObservableCollection<Phone> Phones { get; set; }

public Phone SelectedPhone

{

get { return selectedPhone; }

set

{

selectedPhone = value;

OnPropertyChanged("SelectedPhone");

}

}

public ApplicationViewModel()

{

Phones = new ObservableCollection<Phone>

{

new Phone { Title="iPhone 7", Company="Apple", Price=56000 },

new Phone {Title="Galaxy S7 Edge", Company="Samsung", Price =60000 },

new Phone {Title="Elite x3", Company="HP", Price=56000 },

new Phone {Title="Mi5S", Company="Xiaomi", Price=35000 }

};

}

public event PropertyChangedEventHandler PropertyChanged;

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

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

}

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

mc:Ignorable="d"

Title="MainWindow" Height="350" Width="525">

<Window.Resources>

<Style TargetType="TextBlock">

<Setter Property="FontSize" Value="14" />

</Style>

<Style TargetType="TextBox">

<Setter Property="FontSize" Value="14" />

</Style>

</Window.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="0.8\*" />

</Grid.ColumnDefinitions>

<ListBox Grid.Column="0" ItemsSource="{Binding Phones}"

SelectedItem="{Binding SelectedPhone}">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Margin="5">

<TextBlock FontSize="18" Text="{Binding Path=Title}" />

<TextBlock Text="{Binding Path=Company}" />

<TextBlock Text="{Binding Path=Price}" />

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<StackPanel Grid.Column="1" DataContext="{Binding SelectedPhone}">

<TextBlock Text="Выбранный элемент" />

<TextBlock Text="Модель" />

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

<TextBlock Text="Производитель" />

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

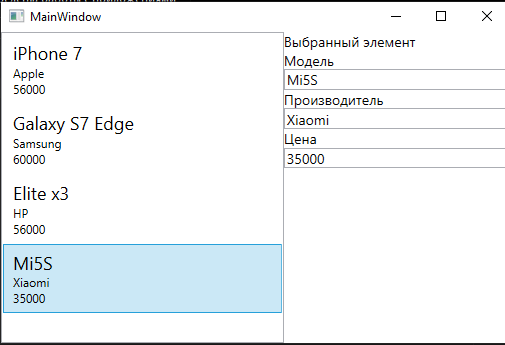
<TextBlock Text="Цена" />

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

</StackPanel>

</Grid>

</Window>



[Ссылка на файл](MVVM/MVVM/bin/Debug/MVVM.exe)

*Ответы на контрольные вопросы:*

1. Каково назначение шаблона MVVM?

MVVM позволяет связывать элементы View со свойствами и событиями ViewModel. При этом ViewModel — абстракция представления. Свойства View совпадают со свойствами ViewModel/Model. При этом ViewModel не имеет ссылки на интерфейс представления. Изменение состояния ViewModel автоматически изменяет View, и наоборот. Для этого используется механизм связывания данных. Также характерная черта MVVM — двусторонняя коммуникация с View (рисунок 26.1).

2. Из каких элементов состоит шаблон MVVM?

* View — содержит поля, соответствующие интерфейсу пользователя.
* ViewModel — содержит такие же поля, но в предметной области.
* Собственно, Model.

3. Принцип работы шаблона MVVM?

MVVM позволяет связывать элементы View со свойствами и событиями ViewModel. При этом ViewModel — абстракция представления. В MVVM есть:

* Model - представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения
* View — содержит поля, соответствующие интерфейсу пользователя, графический интерфейс (окна, списки, кнопки и т. п.)
* ViewModel — содержит такие же поля, но в предметной области, т. е. она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.