**Методические указания к лабораторной работе №11**

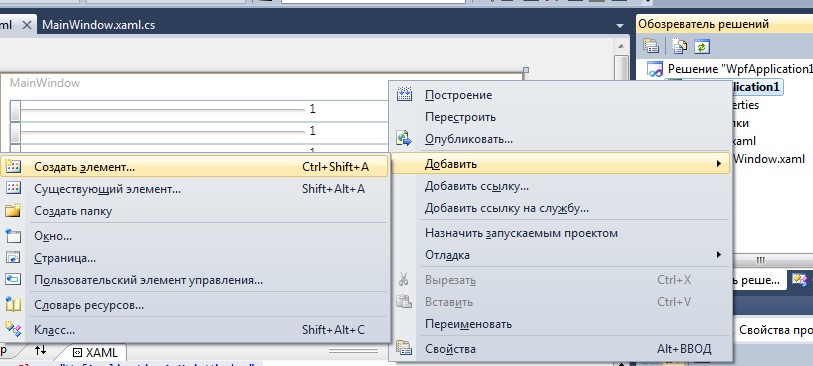
# «Пользовательские элементы в WPF-приложениях»

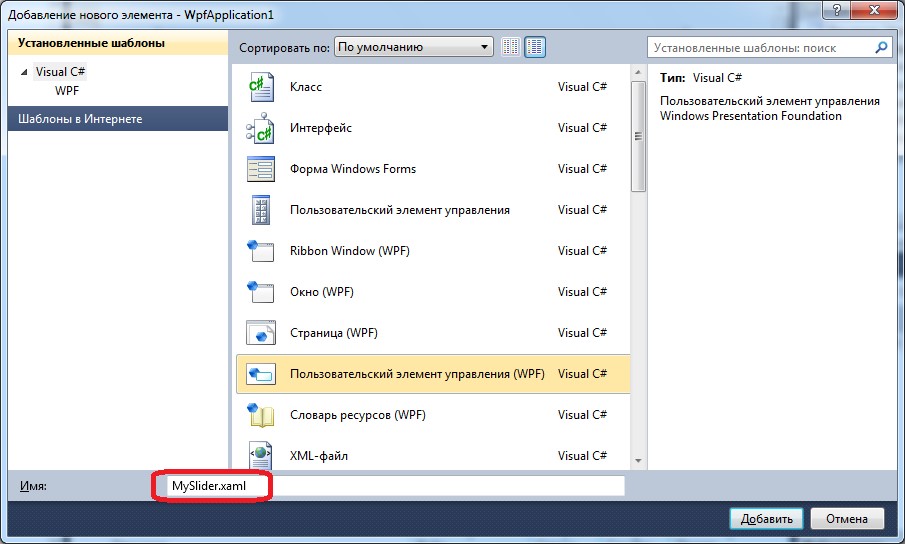
Повторяющиеся участки программного кода необходимо выделять в виде функций, классов. Такое же правило действует и по отношению к WPF-приложениям: повторяющиеся участки XAML-кода с обработчиками выделяются в виде пользовательских элементов. Рассмотрим следующий XAML-код:

|  |
| --- |
| **Код XAML** |
| <Window x:Class="WpfApplication1.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="MainWindow" Height="350" Width="525">  <StackPanel>  <StackPanel Orientation="Horizontal">  <Slider x:Name="sl1" Maximum="100" Minimum="1" Width="300" IsSnapToTickEnabled="True" />  <TextBlock Text="{Binding ElementName=sl1, Path=Value}" />  </StackPanel>  <StackPanel Orientation="Horizontal">  <Slider x:Name="sl2" Maximum="100" Minimum="1" Width="300" IsSnapToTickEnabled="True" />  <TextBlock Text="{Binding ElementName=sl2, Path=Value}" />  </StackPanel>  <StackPanel Orientation="Horizontal">  <Slider x:Name="sl3" Maximum="100" Minimum="1" Width="300" IsSnapToTickEnabled="True" />  <TextBlock Text="{Binding ElementName=sl3, Path=Value}" />  </StackPanel>  </StackPanel>  </Window> |
| **Результат** |
|  |

Для выделения повторяющегося участка кода в виде пользовательского элемента необходимо:

1. Добавить в проект новый элемент «Пользовательский элемент управления (WPF)». При добавлении рекомендуется задать смысловое название для файла, например MySlider.xaml. Указанное название определит имя класса пользовательского элемента.





1. В XAML-коде пользовательского элемента записать повторяющийся участок XAML-кода основного

приложения:

|  |
| --- |
| Файл MySlider.xaml:    <UserControl x:Class="**WpfApplication1**.**MySlider**"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:d="http://schemas.microsoft.com/expression/blend/2008" mc:Ignorable="d"  d:DesignHeight="300" d:DesignWidth="300">  **<StackPanel Orientation="Horizontal">**  **<Slider x:Name="sl" Maximum="100" Minimum="1" Width="300" IsSnapToTickEnabled="True" />**  **<TextBlock Text="{Binding ElementName=sl, Path=Value}" />**  **</StackPanel>** </UserControl> |

В атрибуте x:Class определяются через точку пространство имен, в котором определен пользовательский элемент (в примере – WpfApplication1), и имя класса пользовательского элемента (в примере – MySlider).

1. Добавленный пользовательский элемент MySlider определен в пространстве имен .NET (в примере – WpfApplication1). Чтобы использовать его в основной программе, необходимо в корневом элементе Window указать его пространство имен. Для отображения пространства имен .NET на пространство имен XML в XAML предусмотрен специальный синтаксис:

xmls:ПРЕФИКС=”clr-namespace:ПРОСТРАНСТО\_ИМЕН;assembly=ИМЯ\_СБОРКИ”

Если пользовательский элемент находится в этом же проекте, то параметр assembly не указывается. Для рассматриваемого примера пространство имен элемента – WpfApplication1, в качестве префикса выберем local (можно выбрать любое другое имя):

|  |
| --- |
| Файл MainWindows.xaml:    <Window x:Class="WpfApplication1.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="MainWindow" Height="350" Width="525" **xmlns:local="clr-namespace:WpfApplication1"**  >  … |

1. Заменить в основном коде приложения повторяющиеся участки XAML-кода на пользовательский

элемент:

|  |
| --- |
| **Код XAML** |
| <Window x:Class="WpfApplication1.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="MainWindow" Height="350" Width="525" xmlns:local="clr-namespace:WpfApplication1"  >  <StackPanel>  <local:MySlider /> <local:MySlider />  <local:MySlider />  </StackPanel>  </Window> |
| **Результат** |
|  |

## *Задание 1*

В приложении, разработанном в предыдущей лабораторной работе, выделите повторяющиеся участки XAML-кода в виде пользовательского элемента.

## Эффекты

В WPF реализованы визуальные эффекты (размытие, отбрасывание тени), которые можно применить практически к любому элементу. Все эффекты являются объектами классов, унаследованных от абстрактного класса Effect, объявленного в пространстве имен

System.Windows.Media.Effects. Для одного элемента можно установить только один эффект.

### Эффект размытия (BlurEffect)

Размывает содержимое элемента, как если смотреть на него через расфокусированную линзу. Степень размытия определяется значением свойства Radius (значение по умолчанию 5).

|  |  |
| --- | --- |
| <Button Content="BlurEffect with Radius=2">  <Button.Effect>  <BlurEffect Radius="2" />  </Button.Effect>  </Button>  +<Button Content="BlurEffect with Radius=5  (Default)">  <Button.Effect>  <BlurEffect Radius="5" />  </Button.Effect>  </Button>  <Button Content="BlurEffect with Radius=10">  <Button.Effect>  <BlurEffect Radius="10" />  </Button.Effect>  </Button> |  |

### Эффект отбрасывания тени (DropShadowEffect)

Позволяет добавить за элементом размытую сдвинутую тень с указанным цветом и степенью прозрачности. Свойства данного эффекта:

**Direction** – положение тени относительно элемента в виде значения угла от 0 до 360. Значение 0 – тень находится справа. Значение по умолчанию 315 – тень находится справа внизу;

**ShadowDepth** – расстояние тени от элемента (по умолчанию 5); при значении 0 можно создать эффект ореола вокруг элемента;

**Color** – цвет отбрасываемой тени (по умолчанию черный);

**Opacity** – степень непрозрачности тени (по умолчанию 1 – полностью непрозрачна); **BlurRadius** – степень размытия тени (по умолчанию 5).

|  |  |
| --- | --- |
| <TextBlock Text="Тень по умолчанию">  <TextBlock.Effect>  <DropShadowEffect />  </TextBlock.Effect>  </TextBlock>  <Button Content="Сияющая кнопка">  <Button.Effect>  <DropShadowEffect Color="Red" BlurRadius="20"  ShadowDepth="0" Opacity="0.6" />  </Button.Effect>  </Button> |  |

***Задание 2***

Разработать WPF-

приложение

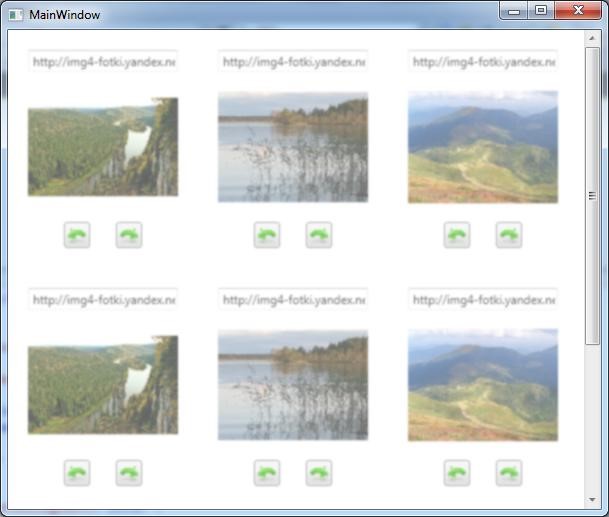
«

Просмотр

изображений

»

со следующим интерфейсом:

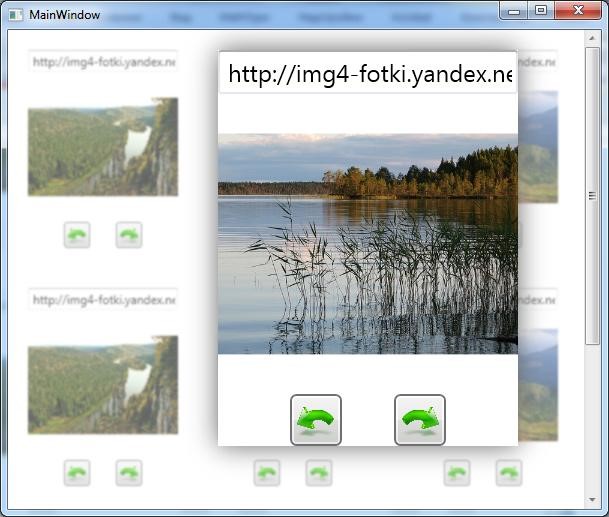


При наведении на

изображение

соответствующий элемент плавно увеличивается, исчезает размытие,

появляется тень:

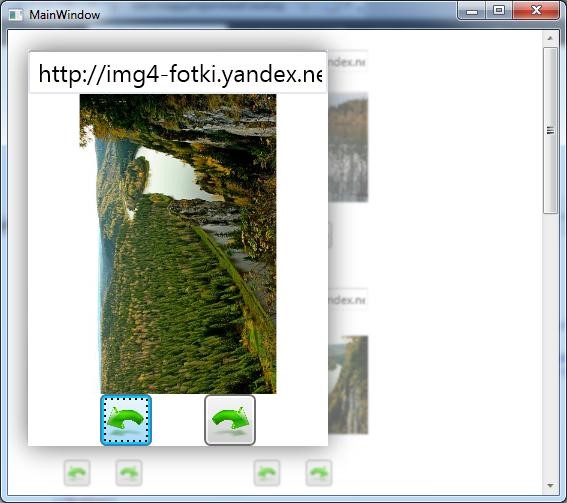


При нажатии на кнопки

изображение

пов

орачивается на 90 градусов влево или вправо:



|  |
| --- |
| **Подсказки:**   1. Различные изображения и URL-адреса можно устанавливать в CS-файле в конструкторе окна после вызова метода инициализации элементов:     public MainWindow()  {  InitializeComponent();  control1.image.Source = new BitmapImage(new Uri("http://img4-fotki.yandex.net/get/6614/39108201.e/0\_STATIC89c24\_345827ea\_L")); control1.textbox.Text = "http://img4-fotki.yandex.net/get/6614/39108201.e/0\_STATIC89c24\_345827ea\_L"; control2.image.Source = ...  Для ускорения работы приложения можно загрузить изображения на диск, подключить их к проекту и загрузить их в элементы Image по относительному адресу:    control1.image.Source = new BitmapImage(new Uri("1.jpg", UriKind.Relative));    Вся остальная функциональность приложения определяется только в XAML-коде.     1. В диспетчере компоновки WrapPanel последующий элемент перекрывает предыдущий при увеличении размеров предыдущего. Чтобы выделенный элемент не перекрывался последующим элементом, необходимо в **основном** XAML-файле добавить триггер, который при наведении курсора на элемент управления записывает в его свойство ZIndex большое значение. Таким образом, элемент отображается над другими элементами:     <Trigger Property="IsMouseOver" Value="True">  <Setter Property="Panel.ZIndex" Value="99" />  </Trigger>     1. Для одного элемента можно применить только один эффект. Чтобы применить два эффекта к выделенному элементу (снятие размытия и появление тени), необходимо использовать два вложенных диспетчера компоновки: для одного будет применен один эффект, для другого – другой эффект:     <StackPanel ...> ...  <StackPanel ...> ...  </StackPanel> </StackPanel>     1. Свойства для анимации эффектов:   Storyboard.TargetProperty="Effect.(BlurEffect.Radius)"  Storyboard.TargetProperty="Effect.(DropShadowEffect.Opacity)"   1. События для триггеров события: MouseEnter, MouseLeave   Свойства для триггеров: IsMouseOver, IsPressed     1. Обработка нажатия на кнопки осуществляется с помощью триггеров привязки. Связанным свойством является свойство «Кнопка нажата». Триггер запускает анимацию изменения угла вращения изображения на 90 градусов:     <DataTrigger Binding="{Binding ElementName=btnRight, Path=IsPressed}" Value="True">  <DataTrigger.EnterActions> <BeginStoryboard> ...  </BeginStoryboard>  </DataTrigger.EnterActions>  </DataTrigger> |