**ИИТ БГУИР**

**Факультет повышения квалификации и переподготовки**   
  
**Дисциплина: Средства визуального программирования приложений**

**Лабораторная работа №3**

Привязка данных и валидация

Вариант 3

**Выполнил слушатель гр. 40322-1** Проверила

**Гончаров Максим Евгеньевич Желакович Ирина Миролюбовна**

Минск 2025

*Цели лабораторной работы*:

Ознакомиться с механизмами привязки данных и валидации в WPF.

*Выполнение работы:*

Привязка данных в WPF позволяет автоматически синхронизировать данные между элементами пользовательского интерфейса и источниками данных, а валидация данных проверяет корректность введенной информации, предоставляя обратную связь пользователю при обнаружении ошибок.

1. Привязка

Установлена привязка к соответствующим визуальным элементам окна.

По кнопке Save выведите всю информацию о текущем состоянии объекта, используя MessageBox.

Используя интерфейс INotifyPropertyChanged реализовано, чтобы все визуальные элементы обновлялись при изменении объекта (проверьте по кнопке Load).

public MainWindow()

{

InitializeComponent();

foreach(COLOREYES color in Enum.GetValues(typeof(COLOREYES)))

comboBoxEyes.Items.Add(color);

NewDriver();

grid.DataContext = driver;

}

public class Driver: INotifyPropertyChanged, IDataErrorInfo

Изображение выглядит как текст, Человеческое лицо, снимок экрана, человек

Содержимое, созданное искусственным интеллектом, может быть неверным.

2. Конвертеры значений

Для свойства Gender и RadioButton использован конвертер.

Привязка выбранного значения осуществляется через параметр SelectedValue. Для фото string UriImage привязана к свойству Source элемента Image. Здесь также потребовался конвертер.

internal class EnumGenderConverter : IValueConverter

{

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

{

return value != null && value.Equals(parameter);

}

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

{

return value?.Equals(true) == true ? parameter : Binding.DoNothing;

}

}

internal class ImageConverter : IValueConverter

{

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

{

return new BitmapImage(new Uri(value.ToString(), UriKind.RelativeOrAbsolute));

}

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

{

return Binding.DoNothing;

}

}

<Window.Resources>

<local:EnumGenderConverter x:Key="genderConverter"/>

<local:ImageConverter x:Key="imageConverter"/>

3. Валидация значений

Добавлена валидация значений:

1. Номер удостоверения должен содержать только цифры и английские буквы.

2. Категория прав должна содержать только корректные значения.

3. Дата рождения –на текущий момент владельцу должно быть не менее 18 и не более 100 лет.

4. Дата выдачи – не раньше 18 лет и не позже текущей даты.

5. Срок действия – не раньше текущей даты, не более 10 лет.

Так же добавлена валидация других значений:

public string this[string columnName]

{

get

{

string error = String.Empty;

switch(columnName)

{

case "Number":

if (Number == null || Number < 100000 || Number > 99999999)

error = "Номер должен быть от 100000 до 99999999";

break;

case "Class1":

if (Class1 == null || Class1 < 'A' || Class1 > 'E')

error = "Класс должен быть от A до E";

break;

case "Name":

if (String.IsNullOrWhiteSpace(Name))

error = "Имя не должно быть пустым";

break;

case "Adress":

if (String.IsNullOrWhiteSpace(Adress))

error = "Адрес не должен быть пустым";

break;

case "Hgt":

if (Hgt == null || Hgt < 100.0 || Hgt > 250.0)

error = "Рост должен быть от 100.0 до 250.0 см";

break;

case "Dob":

if (Dob == null || Dob > DateTime.Now.AddYears(-18) || Dob < DateTime.Now.AddYears(-100))

error = "Дата рождения должна быть не менее 18 лет и не более 100 лет";

break;

case "Iss":

if (Iss == null || Iss > DateTime.Now || (Dob != null && Iss < Dob.Value.AddYears(18)))

error = "Дата выдачи должна быть не позже сегодняшней даты и не раньше чем через 18 лет после даты рождения";

break;

case "Exp":

if (Exp == null || Iss == null || Exp < Iss || Exp > Iss.Value.AddYears(10))

error = "Дата окончания должна быть не раньше даты выдачи и не позже чем через 10 лет после даты выдачи";

break;

}

return error;

}

}

В случае ошибки – появляется оранжевая рамка вокруг элемента толщиной 6 пкс. Для этого создан шаблон:

<ControlTemplate x:Key="ErrorTemplate">

<StackPanel Orientation="Horizontal">

<Border BorderBrush="Red" BorderThickness="6">

<AdornedElementPlaceholder/>

</Border>

<TextBlock Text="\*\*\*" Foreground="Red" FontSize="28">

</TextBlock>

</StackPanel>

</ControlTemplate>

Добавлена всплывающая подсказка с сообщением об ошибке (использованы триггеры).

<Style TargetType="TextBox">

<Style.Triggers>

<Trigger Property="Validation.HasError" Value="True">

<Setter Property="ToolTip"

Value="{Binding RelativeSource={RelativeSource Self},

Path=(Validation.Errors)[0].ErrorContent}"/>

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="DatePicker">

<Style.Triggers>

<Trigger Property="Validation.HasError" Value="True">

<Setter Property="ToolTip"

Value="{Binding RelativeSource={RelativeSource Self},

Path=(Validation.Errors)[0].ErrorContent}"/>

</Trigger>

</Style.Triggers>

</Style>

…

…

<Image x:Name="image" Source="{Binding UriImage, Converter={StaticResource imageConverter}}" MouseDown="Image\_MouseDown" Grid.RowSpan="6" Margin="20"/>

<TextBox x:Name="textBoxName" Grid.Row="1" Grid.Column="1" VerticalAlignment="Center"

FontSize="16" Margin="12" FontWeight="Bold"

Validation.ErrorTemplate="{StaticResource ErrorTemplate}">

<TextBox.Text>

<Binding Path="Name">

<Binding.ValidationRules>

<DataErrorValidationRule/>

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

<TextBox x:Name="textBoxNumber" Grid.Row="1" Grid.Column="2"

VerticalAlignment="Center" FontSize="16" Margin="12"

Validation.ErrorTemplate="{StaticResource ErrorTemplate}" FontWeight="Bold">

<TextBox.Text>

<Binding Path="Number">

<Binding.ValidationRules>

<DataErrorValidationRule/>

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

<TextBox x:Name="textBoxAdress" Grid.Row="2" Grid.Column="1"

VerticalAlignment="Center" FontSize="16" Margin="12" FontWeight="Bold"

Validation.ErrorTemplate="{StaticResource ErrorTemplate}">

<TextBox.Text>

<Binding Path="Adress">

<Binding.ValidationRules>

<DataErrorValidationRule/>

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

<StackPanel Grid.Row="2" Grid.Column="2" Orientation="Horizontal">

<Label Content="CLASS" VerticalAlignment="Center" Margin="10"/>

<TextBox x:Name="textBoxClass1" VerticalAlignment="Center" Width="40" FontSize="16"

Margin="30 10 10 10" FontWeight="Bold" Validation.ErrorTemplate="{StaticResource ErrorTemplate}">

<TextBox.Text>

<Binding Path="Class1">

<Binding.ValidationRules>

<DataErrorValidationRule/>

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

</StackPanel>

<StackPanel Grid.Row="3" Grid.Column="1" Orientation="Horizontal">

<Label Content="DOB" VerticalAlignment="Center" Margin="10"/>

<DatePicker x:Name="datePickerDOB" VerticalAlignment="Center" Width="160" FontSize="16"

Margin="20 10 10 10" FontWeight="Bold"

Validation.ErrorTemplate="{StaticResource ErrorTemplate}">

<DatePicker.SelectedDate>

<Binding Path="Dob">

<Binding.ValidationRules>

<DataErrorValidationRule/>

</Binding.ValidationRules>

</Binding>

</DatePicker.SelectedDate>

</DatePicker>

</StackPanel>

<StackPanel Grid.Row="3" Grid.Column="2" Orientation="Horizontal">

<Label Content="ISS" VerticalAlignment="Center" Margin="10"/>

<DatePicker x:Name="datePickerISS" VerticalAlignment="Center" Width="160" FontSize="16"

Margin="20 10 10 10" FontWeight="Bold"

Validation.ErrorTemplate="{StaticResource ErrorTemplate}">

<DatePicker.SelectedDate>

<Binding Path="Iss">

<Binding.ValidationRules>

<DataErrorValidationRule/>

</Binding.ValidationRules>

</Binding>

</DatePicker.SelectedDate>

</DatePicker>

</StackPanel>

<StackPanel Grid.Row="4" Grid.Column="2" Orientation="Horizontal">

<Label Content="EXP" VerticalAlignment="Center" Margin="10"/>

<DatePicker x:Name="datePickerEXP" VerticalAlignment="Center" Width="160" FontSize="16"

Margin="20 10 10 10" FontWeight="Bold"

Validation.ErrorTemplate="{StaticResource ErrorTemplate}">

<DatePicker.SelectedDate>

<Binding Path="Exp">

<Binding.ValidationRules>

<DataErrorValidationRule/>

</Binding.ValidationRules>

</Binding>

</DatePicker.SelectedDate>

</DatePicker>

</StackPanel>

Кнопки отрабатываются корректно в любой последовательности.

Отчистить:

Изображение выглядит как текст, Человеческое лицо, снимок экрана, человек

Содержимое, созданное искусственным интеллектом, может быть неверным.

Загрузить:

Изображение выглядит как текст, человек, снимок экрана, Человеческое лицо

Содержимое, созданное искусственным интеллектом, может быть неверным.

Сохранить:

Изображение выглядит как текст, человек, снимок экрана, Человеческое лицо

Содержимое, созданное искусственным интеллектом, может быть неверным.

Код проектов всех лабораторных работ, задания и отчеты находятся в публичном доступе на сайте GitHub: <https://github.com/Maxevgen555/SVPP_LB>. Файл решения SVPP\_LB.sln находится в папке \ЛБ1\LB1\_Calc\.

*Вывод:*

Я ознакомился с механизмами привязки данных и валидации в WPF. Установил привязку к соответствующим визуальным элементам окна, для свойств Gender и RadioButton использовал конвертер, добавил валидацию значений и всплывающую подсказку с сообщением об ошибке.