Лабораторная работа №4

«Форматирование значений привязки данных и конвертеры значений»

Цели и задачи: Научиться создавать, использовать и форматировать значения привязки данных.

Теоретическая часть:

Привязка данных — это процесс установки соединения между логикой приложения и пользовательским интерфейсом. Если для привязки заданы правильные настройки, а изменения значений данных сопровождаются правильными уведомлениями, привязанные к данным элементы автоматически отражают изменения. Привязка данных может также означать, что, если внешнее представление данных в элементе изменяется, то базовые данные могут автоматически обновляться для отражения изменений. Например, если пользователь изменяет значение в TextBox элемент, базовое значение данных автоматически обновляется в соответствии с изменениями.

Привязка к данным обычно используется для того, чтобы поместить серверный или локальные данные конфигурации в формы или другие элементы управления ИП. В WPF эта концепция расширяется и уже включает привязку широкого диапазона свойств к различным источникам данных. В WPF свойства зависимости элементов могут быть привязаны к объектам CLR (включая объекты ADO.NET или объекты, связанные с веб-службами и веб-свойства) и к данным XML.

Конвертеры значений (value converter) также позволяют преобразовать значение из источника привязки к типу, который понятен приемнику привязки. Так как не всегда два связываемых привязкой свойства могут иметь совместимые типы. И в этом случае как раз и нужен конвертер значений.

Пример создания приложения WPF

Создадим примитивное приложение WPF с единственным Label на форме. В данный Label будет выводится сегодняшняя дата:

```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:converters="clr-namespace:WpfApplication1"
        Title="MainWindow" Height="77.778" Width="266.919" >
    <Grid>
        <Label x:Name="labelForDate" />
    </Grid>
</Window>
```

Но так как у DateTime.Now возвращаемый тип данных DateTime, а Label.Content принимает только String, необходимо реализовать конвертер данных, который будет преобразовывать DateTime в String:

```
public class DateConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            DateTime date = (DateTime)value;
            return string.Format("Дата: {0}", date.ToShortDateString());
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            return null;
        }
    }
```

В данном случае конвертер данных представляет собой класс, являющийся наследником интерфейса IValueConverter и содержащий два метода Convert и ConvertBack, производящие преобразования в прямую и обратную сторону соответственно. Каждый из представленных методов обладает одинаковым набором параметров:

1. value – то что переводится;

2. targetType – в какой тип данных;

3. parameter – дополнительные параметры;

4. culture – текущая культура (регион).

Далее необходимо объявить наш конвертер в ресурсах окна и подключить его к созданному Label:

```
<Window.Resources>
        <converters:DateConverter x:Key="dateConverter"/>
    </Window.Resources>
    <Grid>
        <Label x:Name="labelForDate"
            Content="{Binding Path=Date,
            Converter={StaticResource dateConverter}}" />
    </Grid>
```

Также необходимо добавить в код приложения тестовый класс, к которому привязан наш Label:

```
public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            labelForDate.DataContext = new DateForLab();
        }
    }

    public class DateForLab
    {
        public DateTime Date { get { return DateTime.Now; } }
    }
```

Если запустить приложение получится следующий результат, отображенный на рисунке 1.

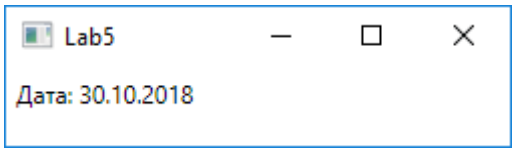

Рисунок 1 – Итоговый результат

Задания на лабораторную работу

1. Изучить материал, представленный в данных методических указаниях к лабораторной работе, а также материал находящийся в сети интернет в свободном доступе по данной теме;
2. Разработать приложение, согласно заданию преподавателя;
3. Составить отчёт. В отчёте отобразить:

– цели и задачи лабораторной работы;

– исходный код приложения (отображающий только логику программы и описание работы);

– код XAML;

– скриншоты работы приложения;

– выводы по данной лабораторной работе.