Лабораторная работа №1

Тема: Основы создания приложений Windows Presentation Foundation (WPF).

Цели и задачи: Изучить основные принципы создания интерфейса приложений WPF. Научится создавать приложения WPF.

Предварительные требования

Для создания приложений WPF необходимо наличие Visual Studio 2012 и выше.

Краткие сведения

Windows Presentation Foundation (WPF) — это система следующего поколения для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. С WPF можно создавать широкий спектр как автономных приложений, так и приложений, размещенных в веб-обозревателе. В основе WPF лежит векторная система визуализации, не зависящая от разрешения и созданная с расчетом на возможности современного графического оборудования. WPF расширяет базовую систему полным набором функций разработки приложений, в том числе Extensible Application Markup Language (XAML), элементами управления, привязкой данных, макетом, 2-D- и 3-D-графикой, анимацией, стилями, шаблонами, документами, мультимедиа, текстом и оформлением. WPF входит в состав Microsoft .NET Framework и позволяет создавать приложения, включающие другие элементы библиотеки классов .NET Framework. В WPF дополнительно совер-

шенствуется процесс программирования для разработки клиентских приложений Windows.

Одним из очевидных усовершенствований является возможность разрабатывать приложения с помощью разметки и кода программной части.

Разметка XAML обычно используется для реализации внешнего вида приложения.

Для реализации поведения используется один из языков программирования в среде Visual Studio например C#.

Такой подход к реализации приложения имеет большое количество положительных сторон, одной из которых можно назвать –независимость внешнего вида от языка программирования и, соответственно более гибкая схема разработки приложения. Например, дизайнер с помощью XAML создает внешний вид приложения и ему нет особого дела до того, что делает программист, занимаясь разработкой программной части – каждый занимается своим делом. Также, использование XAML упрощает перенос приложения на другой язык, поддерживающий .NET – не требуется полная переработка приложения, надо только сменить программную часть, а интерфейсная останется без изменения.

Пример создания приложения

1. Создать проект (рис. 1);

2. Форма конструктора не сильно отличается от привычного Windows Forms конструктора за исключением разметки XAML, в которой размещаются свойства компонентов. Также при создании размещении элементов на форме им автоматически не присваиваются имена;

3. Размещаем компоненты на форме (4 Label, 2 TextBox и одна кнопка) (рис. 2);

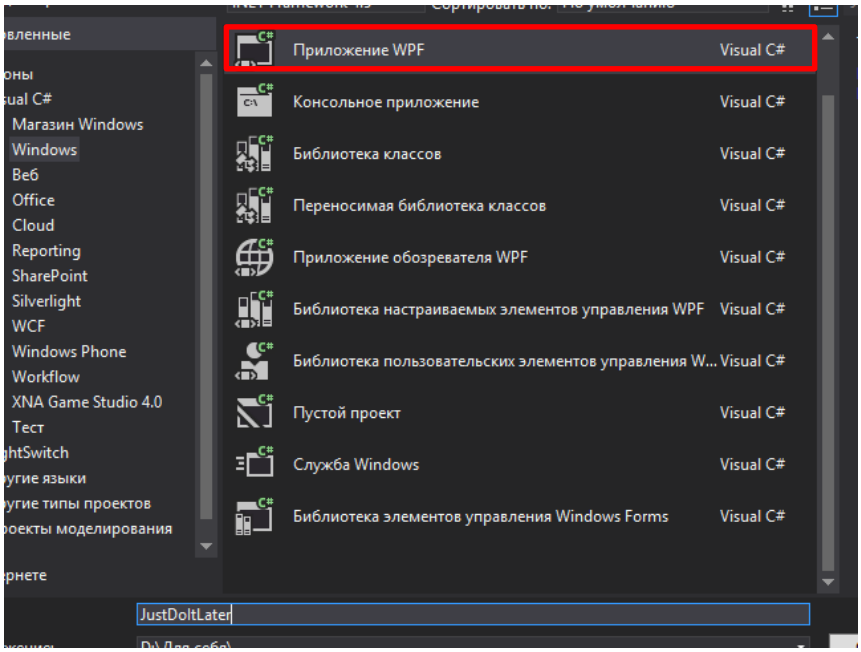


Рисунок 1 – Создание проекта WPF

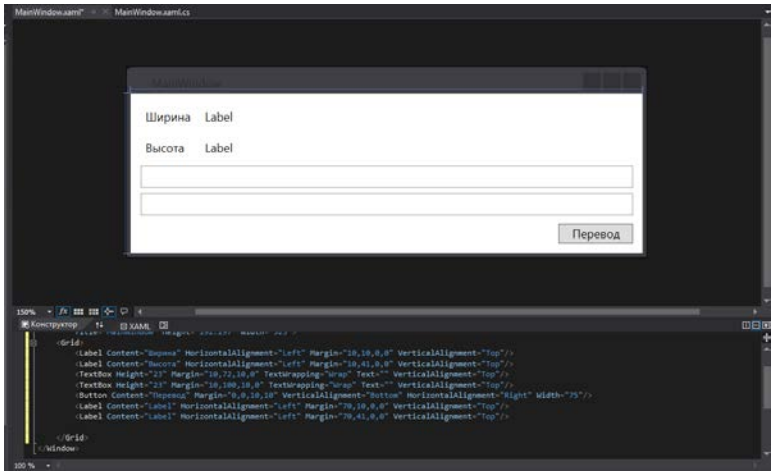
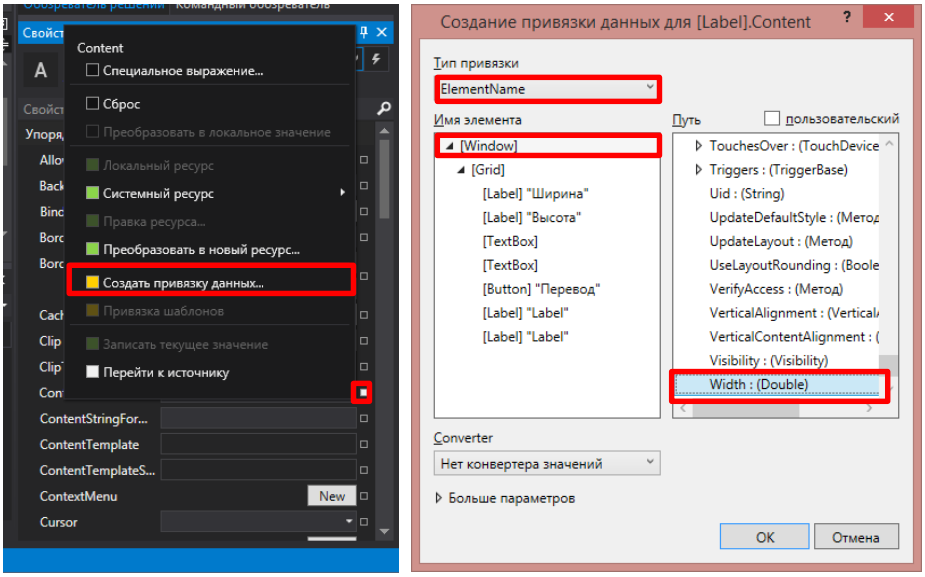


Рисунок 2 – Создание тестового приложения

4. Для отображения ширины и высоты окна удобнее всего создать привязку данных. В свойствах Label на против свойства Caontent из выпадающего списка выбираем создать привязку данных (рис. 3). Аналогично привязка создается и для второго Label. При этом в коде XAML отобразится созданная привязка:

```
    <Label Content="{Binding Width, ElementName=window}" HorizontalAlignment="Left"
Margin="70,10,0,0" VerticalAlignment="Top"/>
    <Label Content="{Binding Height, ElementName=window}" HorizontalAlignment="Left" Mar-
gin="70,41,0,0" VerticalAlignment="Top"/>
```



а б

Рисунок 3 – Создание привязки данных

а) Создание привязки данных

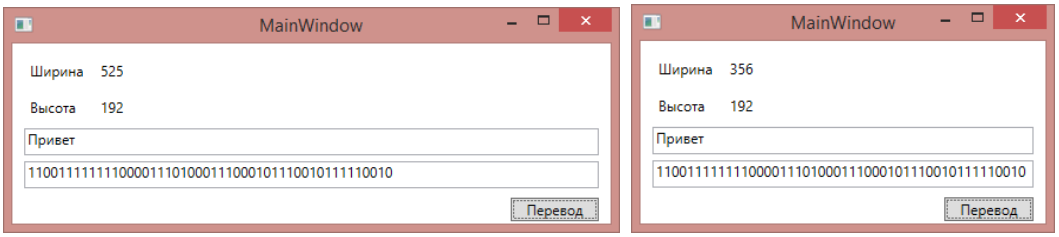
б) Выбор элемента и свойства к которому необходимо привязаться

5. Создаем событие Click кнопки и присваиваем имена для двух TextBox;

6. В созданном событие впишем код, где tInput и tOutput – имена первого и второго TextBox соответственно:

```
    if (tInput.Text != String.Empty)
    {
        byte[] bi = Encoding.Default.GetBytes(tInput.Text); //Перевод строки в байты
        string hex = BitConverter.ToString(bi).Replace("-", ""); //Перевод строки в
16ричную систему счисления
        tOutput.Text = Convert.ToString(Convert.ToInt64(hex, 16), 2); //Вывод результата
    }
```

7. Собираем и запускаем тестовое приложение (при изменении размеров окна данные в Label`ах будут изменяться автоматически) (рис. 4).



а б

Рисунок 4 – Запуск тестового приложения и отображение работы привязки данных

а) Отображение размеров окна до изменения размеров

б) Отображение размеров окна после изменения размеров

Итоговая XAML разметка для тестового приложения

```
<Window x:Name="window" x:Class="JustDoItLater.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="192.297" Width="525">
    <Grid>
        <Label Content="Ширина" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalA-
lignment="Top"/>
        <Label Content="Высота" HorizontalAlignment="Left" Margin="10,41,0,0" VerticalA-
lignment="Top"/>
        <TextBox x:Name="tInput" Height="23" Margin="10,72,10,0" TextWrapping="Wrap"
Text="" VerticalAlignment="Top"/>
        <TextBox x:Name="tOutput" Height="23" Margin="10,100,10,0" TextWrapping="Wrap"
Text="" VerticalAlignment="Top"/>
        <Button Content="Перевод" Margin="0,131,10,0" VerticalAlignment="Top" Horizonta-
lAlignment="Right" Width="75" Click="Button_Click"/>
        <Label Content="{Binding Width, ElementName=window}" HorizontalAlignment="Left"
Margin="70,10,0,0" VerticalAlignment="Top"/>
        <Label Content="{Binding Height, ElementName=window}" HorizontalAlignment="Left"
Margin="70,41,0,0" VerticalAlignment="Top"/>
    </Grid>
</Window>
```

Задания на лабораторную работу

1. Изучить материал, представленный в данных методических указаниях к лабораторной работе;

2. Разработать WPF приложение, согласно заданию преподавателя;

3. Составить отчет. В отчете отобразить:

– цели и задачи лабораторной работы;

– XAML разметку;

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

– скриншоты работы приложения;

– выводы по проделанной лабораторной работе.