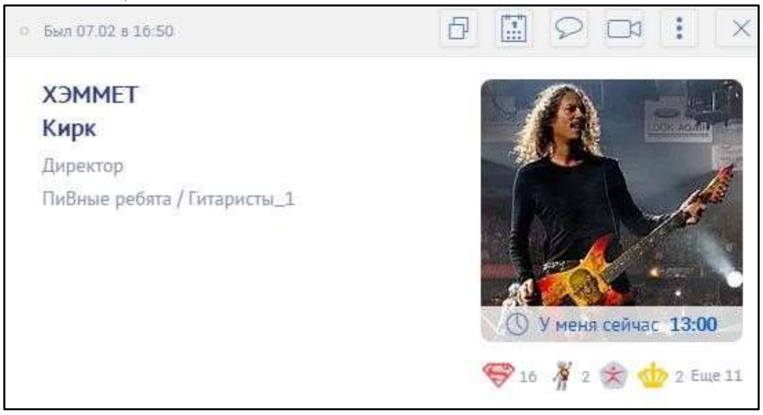
Лабораторная работа №3: Создание пользовательских элементов управления в WPF

Цель работы:

- изучить приемы создания пользовательских элементов управления с использованием технологий WPF;
- познакомиться с примерами использования наследования, полиморфизма и композиции объектов на практике.

Задание:

Разработать WPF-приложение с пользовательскими элементами управления (карточка сотрудника).
Карточка сотрудника отображает основные сведения о сотруднике (имя, отдел, должность, фотография и другие). При наведении на кнопки должна появляться всплывающая подсказка (ToolTip).
Пользователь вводит в текстовое поле имя, во второе текстовое поле – отдел, в третье текстовое поле – должность и так далее. Введенные пользователем данные заносятся в карточку сотрудника.
Один из примеров реализации:



Результат выполнения работы предоставить в виде:

- архив с проектом (если размер архива больше 2 Мбайт, то рекомендуется загрузить проект на https://github.com/ или на другое общедоступное хранилище и предоставить ссылку);
- отчет по лабораторной работе в формате Microsoft Word, который содержит следующие разделы:

1. титульный лист;
2. задание на лабораторную работу;
3. краткое описание разработанных программ и используемых алгоритмов со скриншотами выполнения;
4. вывод с результатами работы.

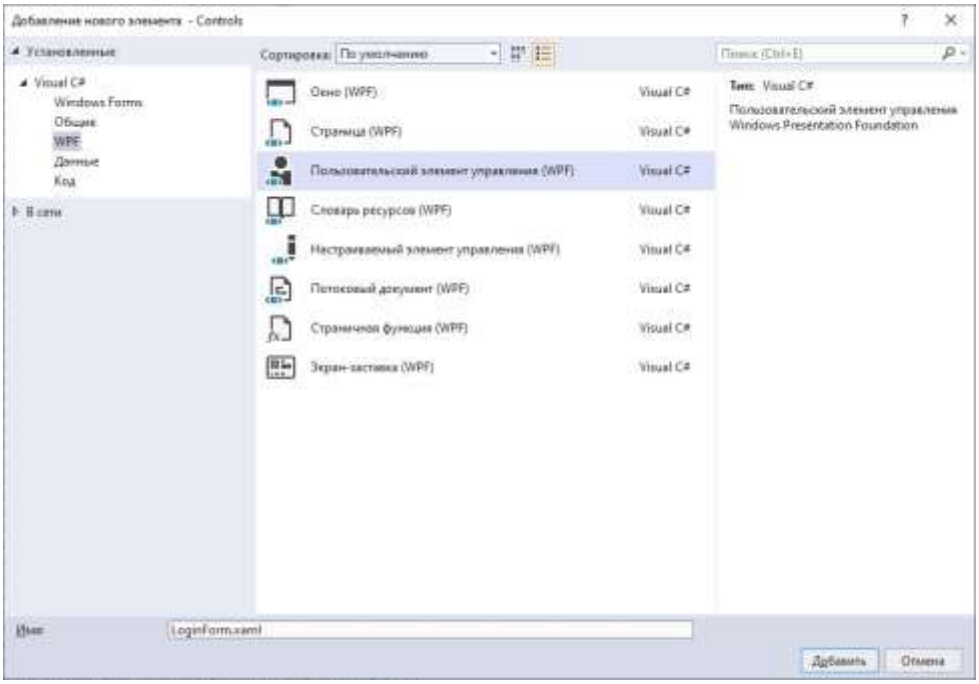
Копирование исходного кода программ не допускается. Проекты с одинаковым исходным кодом засчитываться не будут.

Краткие теоретические сведения:

Создание формы авторизации на WPF:

Пользовательские элементы управления на WPF представляют собой, как правило, композицию нескольких стандартных элементов управления, приведенных к единому стилю.

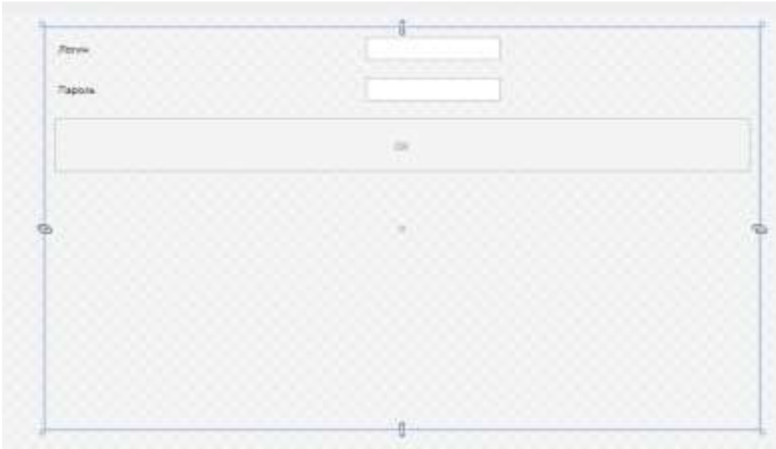
Создайте проект «Приложение WPF (.NET Framework)». В контекстном меню проекта выберите
«Добавить – Создать элемент», а затем, в диалоговом окне, «Пользовательский элемент управления». Введите название и нажмите «Добавить».



После этого в обозревателе решений будут доступны 2 файла: .xaml-файл для создания макета и .csфайл для определения логики пользовательского элемента управления.

Для создания разметки воспользуйтесь визуальным конструктором или отредактируйте .xaml-файл вручную. В качестве корневого элемента можно использовать **Grid** или **StackPanel**.

Добавьте в макет поле ввода логина **TextBox** и поле ввода пароля **PasswordBox**, а так же другие вспомогательные элементы по необходимости: кнопку подтверждения **Button** и заголовки **Label.** Один из вариантов макета:



Для использования элементов макета в .cs-файле необходимо задать им имена: например, для поля ввода логина **Name=«Login»**.

После сборки проекта элемент управления также будет доступен в панели элементов.

Передача параметров элемента управления из разметки:

Некоторые параметры элемента управления удобно задавать из разметки, например, визуальные параметры (цвет фон, параметры шрифта и др.) или значения по умолчанию. В данном случае, требуется задавать значение по умолчанию в поле ввода логина.

Данный функционал реализуется с помощью служебного класса `DependencyProperty`. В .cs-файле создайте объект класса `DependencyProperty`:

```
public static readonly DependencyProperty LoginProperty = DependencyProperty.Register(
            "Login", // имя параметра в разметке
            typeof(string), // тип данных параметра
            typeof(LoginForm), // тип данных элемента управления
            new PropertyMetadata(string.Empty, LoginChanged)); // метаданные - значение параметра по умолчанию и обработчик изменения параметра
```

Для удобства использования значения параметра создайте свойство, делегирующее работу с объектом `LoginProperty`:

```
public string Login
{
       get { return (string)GetValue(LoginProperty); }
       set { SetValue(LoginProperty, value); }
}
```

В обработчике изменения параметра `LoginChanged` необходимо добавить код, который реализует добавление текста в поле ввода логина:

```
private static void LoginChanged(DependencyObject obj, DependencyPropertyChangedEventArgs args)
{
       var loginForm = obj as LoginForm;
       loginForm.LoginTextBox.Text = loginForm.Login;
}
```

После этого можно передать значение по умолчанию через разметку. Для этого в разметку поля ввода логина необходимо добавить **Login=«admin»**.

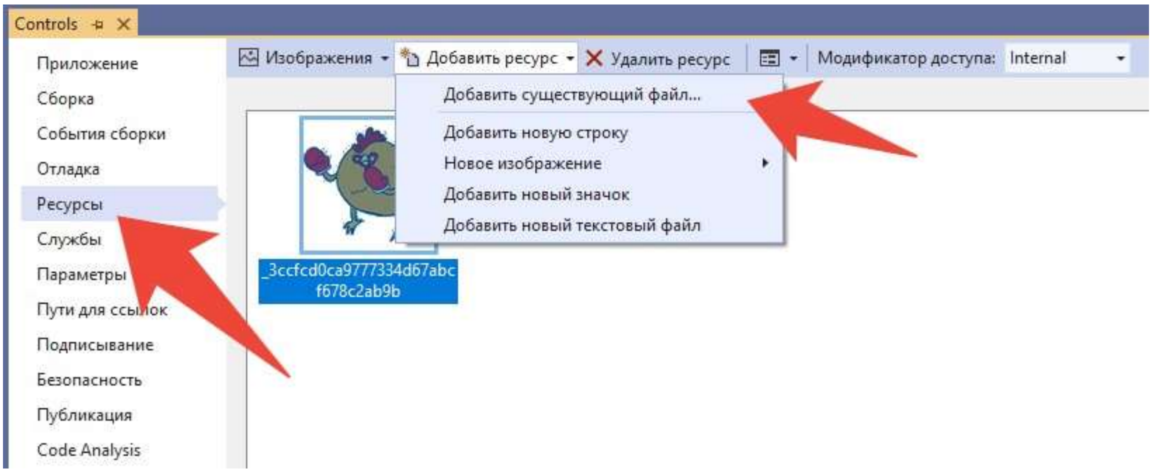
Также необходимо реализовать логику работы элемента управления, например, осуществлять валидацию введенных значений логина и пароля, а по нажатию на кнопку подтверждения осуществлять некоторое действие. На каждое действие необходимо создать и назначить свой обработчик:

```
LoginTextBox.TextChanged += LoginTextBox_TextChanged; // назначение обработчика изменения логина
PasswordBox.PasswordChanged += PasswordBox_PasswordChanged; // назначение обработчика изменения пароля
OkButton.Click += OkButton_Click; // назначение обработчика нажатия кнопки подтверждения
```

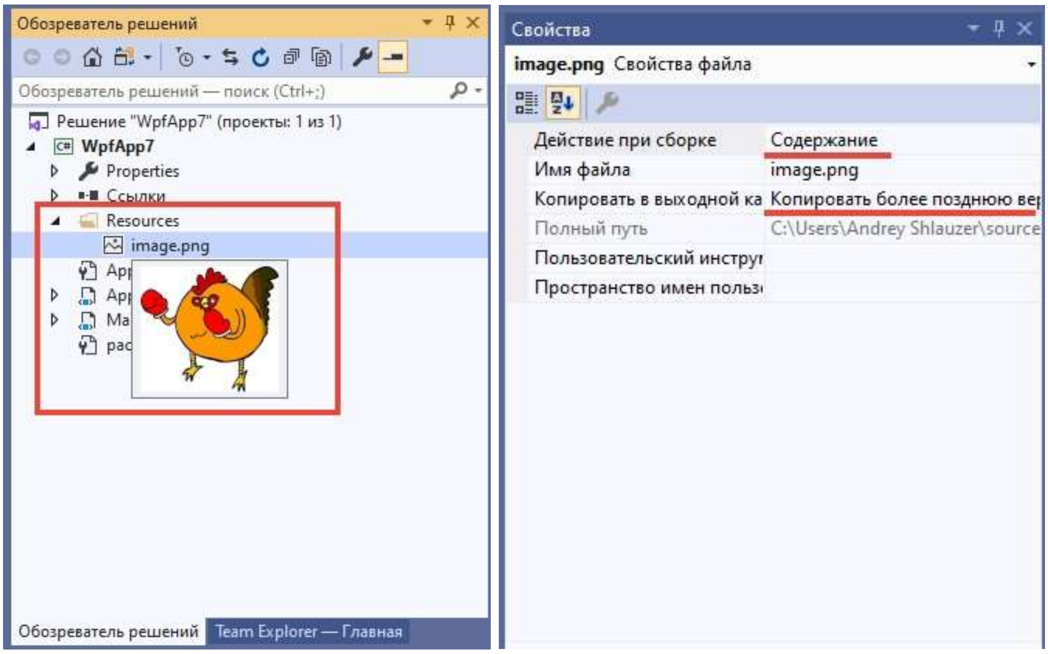
```
private void OkButton_Click(object sender, RoutedEventArgs e) {
      // TODO : need action
}
private void PasswordBox_PasswordChanged(object sender, RoutedEventArgs e) {
      // TODO : need action
}
private void LoginTextBox_TextChanged(object sender, TextChangedEventArgs e) {
      // TODO : need action
}
 private void validateData() {
      // длина логина должна быть не менее 4 символов,
      // а длина пароля - не менее 8 символов
      bool isDataValid = Login.Length >= 4 && Password.Length >= 8;
      if (isDataValid)
            OkButton.IsEnabled = true;
      else
            OkButton.IsEnabled = false;
}
```

Работа с изображениями:

В приложениях Windows Forms и WPF с изображением удобно работать как с ресурсом. Чтобы загрузить изображение, перейдите в свойства проекта (вкладка «Ресурсы») и добавьте файл с изображением в ресурсы.



После добавления изображения в ресурсы оно появится в обозревателе решений. Выделите его и установите в панели свойств такие же значения атрибутов, как на скриншоте (это нужно для того, чтобы изображение копировалось в выходную директорию с .exe-файлом в процессе сборки).



Затем можно обращаться к изображению из кода. В стандартном классе `Resources` будет сгенерировано одноименное поле для получения картинки.

В WPF достаточно прописать путь к картинке по следующему шаблону:

```
image1.Source = new BitmapImage(new Uri("pack://application:,,,/Resources/image.jpg"));
```

В данном примере картинка будет отображена в стандартном компоненте `Image (image1 - имя конкретного компонента в разметке)`.

Список литературы:

1) Герберт Шилдт "C# 4.0: полное руководство"
2) Эндрю Троелсен "Язык программирования C# 5.0 и платформа .NET 4.5"
3) Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7 https://metanit.com/sharp/tutorial/
4) Руководство по WPF https://metanit.com/sharp/wpf/

5) C# 5.0 и платформа .NET 4.5
http://professorweb.ru/my/csharp/charp_theory/level1/infocsharp.php
6) https://github.com/Microsoft/WPF-Samples