Лабораторная работа №6

Тема: Шаблоны и стили WPF, менеджер пакетов NuGet.

Цели и задачи: Изучить основные принципы создания шаблонов и стилей WPF. Научиться их практическому применению, при создании приложений. Научиться пользоваться менеджером пакетов NuGet.

Краткие сведения

В WPF стилизация и использование шаблонов относятся к набору функций (стилей, шаблонов, триггеров и раскадровок), позволяющих разработчикам и дизайнерам создавать визуально привлекательные эффекты, а также создавать целостный внешний вид своих продуктов. Несмотря на то, что разработчики и дизайнеры могут настроить внешний вид в масштабе приложений, для обслуживания и совместного использования внешнего вида внутри приложений и между приложениями необходима строгая модель стилей и шаблонов.

Другая функция модели стилизации WPF состоит в разделении представления и логики. Это означает, что дизайнеры могут создавать внешний вид приложения, используя только XAML, в то же самое время, когда разработчики работают над логикой программы, используя C# или Visual Basic.

Таким образом шаблон представляет собой макет размещения информации внутри какого либо элемента управления, а стиль отвечает за настройки элемента (за его свойства). Также в стилях используются так называемые

триггеры (Triggers). Они отвечают за изменение параметров, при наступлении какого либо события.

NuGet — это бесплатный продукт, с открытым кодом который облегчает добавление сторонних библиотек в разрабатываемое приложение.

Пример создания приложения

1. Создать проект WPF;

2. Подключить пакет NuGet (вызвать контекстное меню проекта «Управление пакетами NuGet…» Выбрать необходимый пакет (рис. 1) Установить пакет);

3. Добавить в пространство имен «HtmlAgilityPack»;

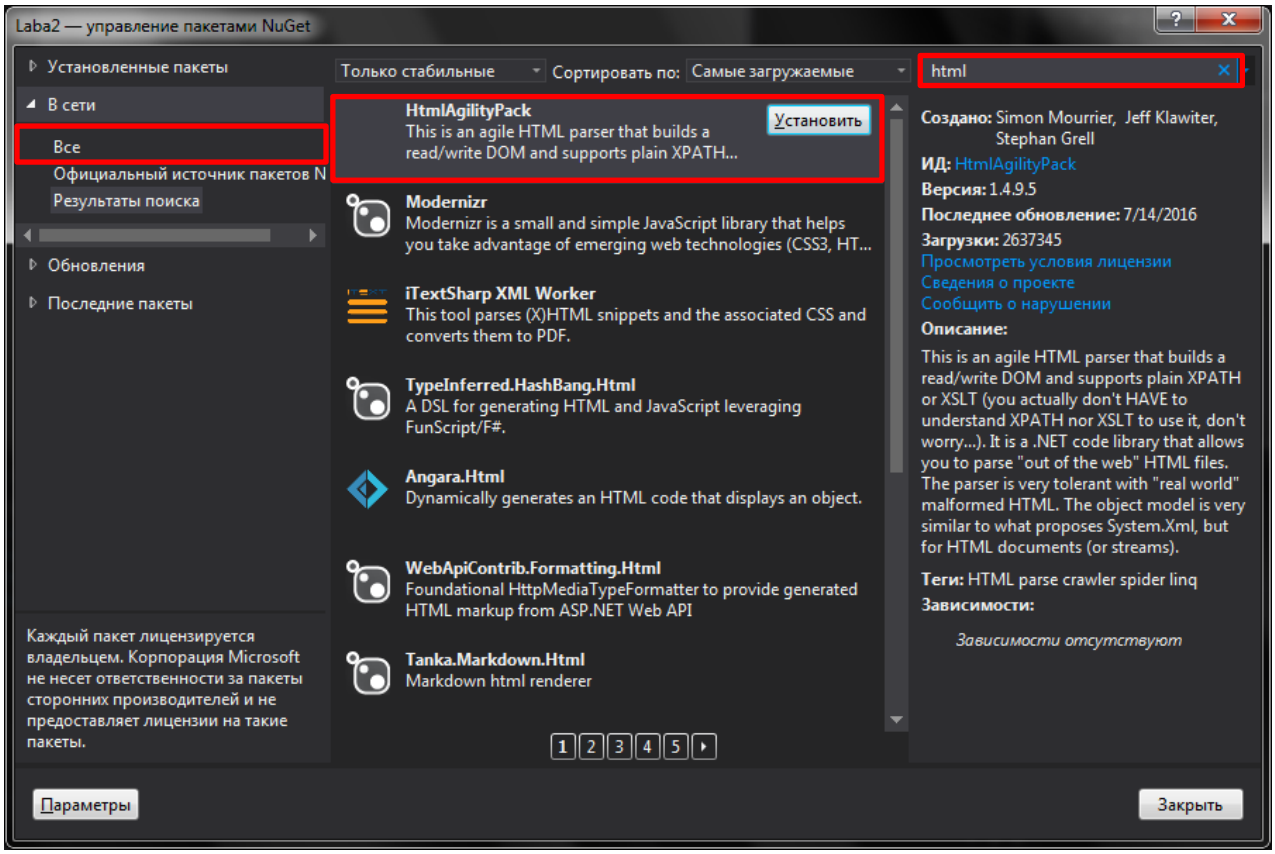


Рисунок 1 – управление пакетами NuGet

4. На форме разместить ListBox;

5. Создать класс для хранения информации о фильме;

```
class Film
    {
        string _name;
        string _description;
        Uri _url;
        Uri _imageUrl;

        public Film() { }

        /// <summary>
        /// Наименование
        /// </summary>
        public string Name
        {
            get { return _name; }
            set { _name = value; }
        }

        /// <summary>
        /// Описание
        /// </summary>
        public string Description
        {
            get { return _description; }
            set { _description = value; }
        }

        /// <summary>
        /// Ссыелка
        /// </summary>
        public Uri Url
        {
            get { return _url; }
            set { _url = value; }
        }

        /// <summary>
        /// Ссылка на изображение
        /// </summary>
        public Uri ImageUrl
        {
            get { return _imageUrl; }
            set { _imageUrl = value; }
        }

    }
```

6. Создать метод загрузки информации о фильме;

```
private void GetFilms()
        {
            string URL = @"http://www.hdkinoteatr.com";

            WebClient wClient = new WebClient();
            wClient.Encoding = Encoding.UTF8;

            HtmlDocument doc = new HtmlDocument(); //Создание документа
HTML
            doc.LoadHtml(wClient.DownloadString(URL)); //Загрузка Html из
интернета

            var elements = doc.GetElementbyId("dle-
content").Elements("div").
```

```
                Where(e => e.Attributes["class"].Value == "base short-
story"); //получение последних добавленных фильмов

            List<Film> Films = new List<Film>();
            foreach (var el in elements)
            {
                Film film = new Film();

                HtmlNode info = el.Elements("div").Where(e =>
e.Attributes["class"].Value == "dpad").ToList()[0]; //получение блока со-
держащего информацию о фильме

                string imgUrl = URL;
                imgUrl += info.Elements("div").
                    Where(e => e.Attributes["class"].Value ==
"img").ToList()[0].

Element("a").Element("img").Attributes["src"].Value; //получение ссылки
на изображение

                film.ImageUrl = new Uri(imgUrl);

                HtmlNode shortInfo = info.Elements("div").
                    Where(e => e.Attributes["class"].Value ==
"story").ToList()[0]; //блок с краткой информацией

                film.Name = shortInfo.Element("h2").InnerText;
                film.Url = new
Uri(shortInfo.Element("h2").Element("a").Attributes["href"].Value);
                film.Description = shortInfo.Elements("div").
                    Where(e => e.Attributes["class"].Value ==
"maincont").ToList()[0].
                    Elements("div").ToList()[1].InnerText;

                Films.Add(film);
            }

            lFilms.ItemsSource = Films; //Вывод информации в ListBox
        }
```

7. Создать шаблон отображения данных в ListBox;

```
<Window.Resources>
      <DataTemplate x:Key="FilmTemplate">
            <Grid Height="125">
                  <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="80"/>
                        <ColumnDefinition Width="274"/>
                  </Grid.ColumnDefinitions>

                  <Border Margin="5" BorderBrush="Black"
         BorderThickness="1" VerticalAlignment="Center">
                        <Image Source="{Binding Path=ImageUrl}"
               Stretch="Fill" Width="80" Height="118"/>
                  </Border>

                  <StackPanel Grid.Column="1" Margin="5">

                        <TextBlock Text="{Binding Path=Name}"
                        ToolTip="{Binding Path=Name}"
                        MaxHeight="15"
                        FontWeight="Bold" FontSize="14"
                        TextWrapping="Wrap" TextAlignment="Center"
                        TextTrimming="WordEllipsis"/>
```

```
                                <TextBlock Text="{Binding Path=Description}"
                                ToolTip="{Binding Path=Description}"
                                MaxHeight="100"
                                TextWrapping="Wrap" TextTrimming="WordEllipsis"
                                TextAlignment="Justify" />

                            </StackPanel>
                        </Grid>
                    </DataTemplate>
                </Window.Resources>
```

8. У свойства ListBox ItemTemplate установить привязку к созданному шаблону данных (рис. 2);

9. Создать стиль кнопки;

```
        <Style TargetType="{x:Type Button}">
                    <Setter Property="Background" Value="White"/>
                    <Setter Property="Template">
                        <Setter.Value>
                            <ControlTemplate TargetType="Button">
                                <Border x:Name="borderButton"
                                        CornerRadius="5"
                                        Height="{Binding Height,
RelativeSource={RelativeSource TemplatedParent}}"
                                        BorderBrush="#4F9FCB"
                                        BorderThickness="1"
                                        Background="White"
                                        >
                                    <TextBlock VerticalAlignment="Center"
                                               HorizontalAlignment="Center"
                                               TextAlignment="Center"
                                               Text="{Binding Content, Rela-
tiveSource={RelativeSource TemplatedParent}}"/>
                                </Border>

                                <ControlTemplate.Triggers >
                                    <Trigger Property="IsMouseOver" Value="true">
                                        <Setter TargetName="borderButton"
Property="BorderBrush" Value="#00F"/>
                                    </Trigger>

                                    <Trigger Property="IsPressed" Value="true">
                                        <Setter TargetName="borderButton"
Property="BorderBrush" Value="#00F"/>
                                        <Setter TargetName="borderButton"
Property="Background" Value="#66F"/>
                                        <Setter TargetName="borderButton"
Property="Opacity" Value="0.8"/>
                                    </Trigger>
                                </ControlTemplate.Triggers>
                            </ControlTemplate>
                        </Setter.Value>
                    </Setter>
                </Style>
```

10. На событие нажатия кнопки создадим вызов функции обновления содержимого ListBox;

```
GetFilms();
```

11. Запустим приложение и дождемся загрузки информации в ListBox (рис. 3).

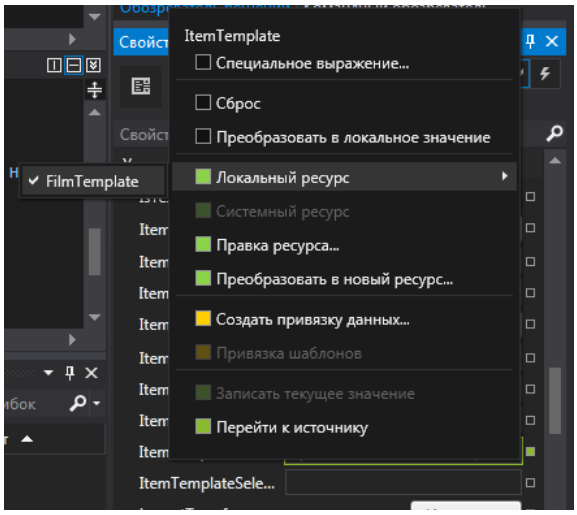
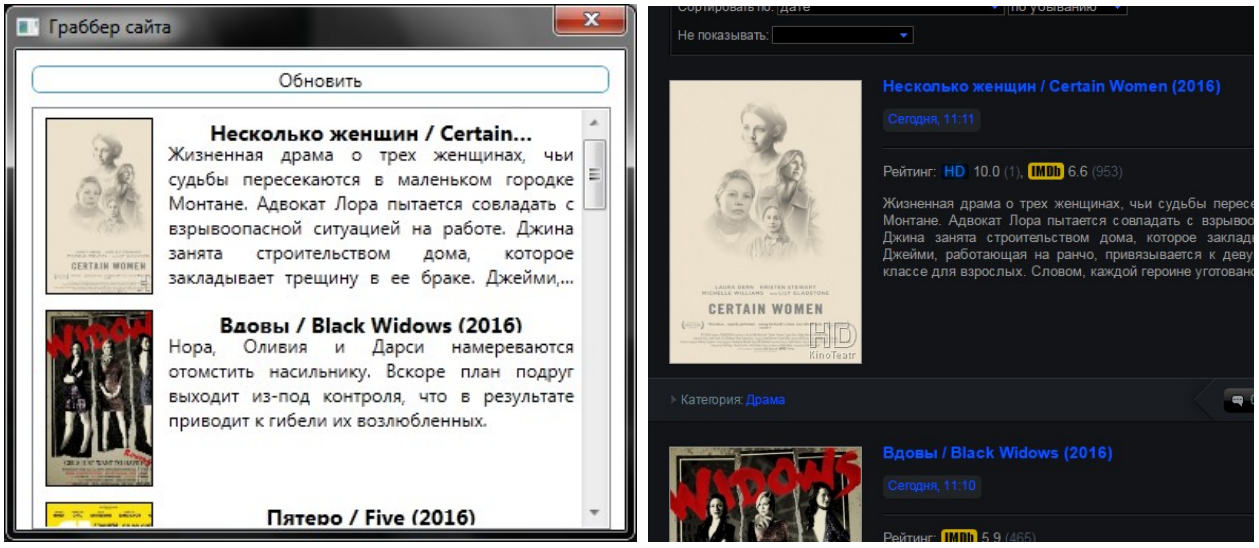


Рисунок 2 – установка привязки к созданному ранее шаблону данных



а) б)

Рисунок 3 – Результат получения информации с сайта

а) Приложение получающее информацию с сайта

б) Сайт, с которого получается информация

Итоговая XAML разметка для тестового приложения

```
<Window x:Class="Laba2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Граббер сайта" Height="350" Width="412" ResizeMode="NoResize">

    <Window.Resources>
        <DataTemplate x:Key="FilmTemplate">
            <Grid Height="125">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="80"/>
                    <ColumnDefinition Width="274"/>
                </Grid.ColumnDefinitions>
                <Border Margin="5" BorderBrush="Black" BorderThickness="1" Verti-
calAlignment="Center">
                    <Image Source="{Binding Path=ImageUrl}" Stretch="Fill"
Width="80" Height="118"/>
                </Border>
                <StackPanel Grid.Column="1" Margin="5">
                    <TextBlock Text="{Binding Path=Name}"
                               ToolTip="{Binding Path=Name}"
                               MaxHeight="15"
                               FontWeight="Bold" FontSize="14"
                               TextWrapping="Wrap" TextAlignment="Center"
                               TextTrimming="WordEllipsis"/>
                    <TextBlock Text="{Binding Path=Description}"
                               ToolTip="{Binding Path=Description}"
                               MaxHeight="100"
                               TextWrapping="Wrap" TextTrimming="WordEllipsis"
                               TextAlignment="Justify" />
                </StackPanel>
            </Grid>
        </DataTemplate>

        <Style TargetType="{x:Type Button}">
            <Setter Property="Background" Value="White"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border x:Name="borderButton"
                                CornerRadius="5"
                                Height="{Binding Height, RelativeSource={Rela-
tiveSource TemplatedParent}}"
                                BorderBrush="#4F9FCB"
                                BorderThickness="1"
                                Background="White"
                                >
                            <TextBlock VerticalAlignment="Center"
                                       HorizontalAlignment="Center"
                                       TextAlignment="Center"
                                       Text="{Binding Content, RelativeSource={Rel-
ativeSource TemplatedParent}}"/>
                        </Border>

                        <ControlTemplate.Triggers >
                            <Trigger Property="IsMouseOver" Value="true">
                                <Setter TargetName="borderButton" Property="Border-
Brush" Value="#00F"/>
                            </Trigger>

                            <Trigger Property="IsPressed" Value="true">
                                <Setter TargetName="borderButton" Property="Border-
Brush" Value="#00F"/>
                                <Setter TargetName="borderButton" Property="Back-
ground" Value="#66F"/>
```

```
                                <Setter TargetName="borderButton" Property="Opac-
ity" Value="0.8"/>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

    </Window.Resources>

    <Grid>
        <ListBox x:Name="lFilms" Margin="10,37,9,0" ItemTemplate="{DynamicResource
FilmTemplate}"/>
        <Button x:Name="bUpdate" Content="Обновить" Margin="10,10,10,0" Verti-
calAlignment="Top" Click="bUpdate_Click"/>

    </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.