

## Задание 2. Photo Gallery

Элементы WPF: DockPanel, WrapPanel, ListBox, TreeView, Menu, Popup



### Рекомендации

Компоновку основных элементов можно организовать с помощью контейнера DockPanel или Grid. В случае DockPanel элементы могут быть привязаны к краям формы с помощью свойства **DockPanel.Dock**: панель Menu – к верхнему краю, панель с кнопками - к нижнему краю, дерево папок TreeView – к левому краю, список файлов ListBox – к правому краю (последний элемент можно не привязывать – в этом случае он займет все оставшееся пространство).

Заполнение дерева папок можно организовать в режиме «по требованию» - информация о подкаталогах подгружается только в случае раскрытия каталога пользователем.

В начале можно заполнить дерево дисками:

```
foreach (string s in Directory.GetLogicalDrives())
{
    // Отдельный элемент дерева (ветка)
    TreeViewItem item = new TreeViewItem();

    // В заголовке храним «короткое имя папки»
    item.Header = s;
    // В тэге храним полный путь (в случае дисков Header и Tag совпадают)
    item.Tag = s;

    // Изначально ветка не содержит подкаталогов
    // Используем «пустышку» - object dummyNode = null;
    item.Items.Add(dummyNode);

    // Обработчик «раскрытия» каталога – нужно загрузить подкаталоги
    item.Expanded += new RoutedEventHandler(folder_Expanded);
}
```

```

        // Добавляем ветку в дерево
        tree.Items.Add(item);
    }

```

В обработчике `folder_Expanded` сначала необходимо установить, выполнялась ли уже загрузка подкаталогов для конкретного элемента `TreeViewItem`. Если каталог уже раскрывался пользователем, то коллекция `Items` содержит подкаталоги (в этом случае ничего делать не нужно). Если каталог еще не раскрывался (`Items` содержит пустышку `dummyNode`), то нужно получить подкаталоги и заполнить коллекцию узла (для получения всех подкаталогов можно использовать метод `Directory.GetDirectories`).

Справа отображается список картинок в выбранной папке. Обновление списка выполняется при изменении активного элемента в дереве (событие `SelectedItemChanged`).

Для получения всех файлов в папке по маске используем - `Directory.GetFiles(folder, "*.jpg")`.

В качестве контейнера можно использовать `ListBox`. Для того чтобы элементы списка отображались не в стандартном режиме для списка, а в более привычной форме (как в Проводнике Windows), необходимо настроить «шаблон отображения» `ItemsPanelTemplate`. В следующей XAML-разметке устанавливаем в качестве шаблона отображения контейнер `WrapPanel`:

```

<ListBox x:Name="pictureBox" SelectionChanged="pictureBox_SelectionChanged"
        ScrollViewer.HorizontalScrollBarVisibility="Disabled">
    <ListBox.ItemsPanel>
        <ItemsPanelTemplate>
            <WrapPanel/>
        </ItemsPanelTemplate>
    </ListBox.ItemsPanel>
</ListBox>

```

В разметке дополнительно отключается возможность горизонтальной прокрутки, чтобы элементы не выходили за пределы `ListBox` вправо.

Каждый элемент контейнера `ListBoxItem` соответствует одному файлу и отображается в виде картинки. Для этого необходимо подготовить объект `Image` и заполнить содержимое элемента `ListBoxItem`:

```

ListBoxItem item = new ListBoxItem();
item.Padding = new Thickness(3, 8, 3, 8);
Image image = new Image();
// Исходный размер
image.Height = 35;
// Сохраняем путь к файлу
item.Tag = s;

// Обработчик двойного щелчка – показываем картинку (н-р, в отдельном окошке)
item.MouseDoubleClick += delegate { ShowPhoto(); };

// Для возможности преобразований картинки (повороты, изменение размера)
// для каждой картинки устанавливаем свойство LayoutTransform
// в случае двух преобразований необходимо использовать «группу»
TransformGroup tg = new TransformGroup();
// В эту группу поместим объект типа ScaleTransform для возможности изменять размер
// каждая картинка сохраняет ссылку на один и тот же объект scaleTransform
// объявленный как элемент класса: var scaleTransform = new ScaleTransform(3, 3);
tg.Children.Add(scaleTransform);
// отдельный объект для возможности поворачивать картинку
tg.Children.Add(new RotateTransform());
item.LayoutTransform = tg;

```

```
// Объект uri относится к типу Uri и формируется по пути к файлу.
image.Source = new BitmapImage(uri);
item.Content = image;

// Свойство ToolTip настраивает «подсказку» при наведении мышкой на элемент
// Можно инициализировать любым объектом
// (н-р, строка или контейнер из нескольких элементов)
item.ToolTip = ..;
```

Для управления размером файлов-картинок можно воспользоваться ползунком **Slider**, который отображается во всплывающем окне **Popup** при нажатии на кнопку:



```
<Popup x:Name="zoomPopup" MouseLeave="zoomPopup_MouseLeave"
PlacementTarget="{Binding ElementName=zoomButton}" Placement="Center"
StaysOpen="False">
    <Grid Background="White">
        <Border BorderBrush="Black" BorderThickness="1">
            <Slider Margin="20" Orientation="Vertical" x:Name="zoomSlider"
                Value="3" Minimum="1" Maximum="10" Height="100"
                ValueChanged="zoomSlider_ValueChanged"/>
        </Border>
    </Grid>
</Popup>
```

Свойство **PlacementTarget** управляет размещением всплывающего окна (рядом с кнопкой **zoomButton**).

Для отображения/скрытия всплывающего окна используем свойство **zoomPopup.IsOpen**.

При изменении положения ползунка необходимо обновить значения объекта **scaleTransform**, на который ссылаются все файлы-картинки. Это приведет к увеличению/уменьшению размеров всех файлов-картинок.

```
void zoomSlider_ValueChanged(object sender, RoutedEventArgs e)
{
    st.ScaleX = zoomSlider.Value;
    st.ScaleY = zoomSlider.Value;
}
```

Для возврата к «стандартным» размерам картинок нужно предусмотреть отдельную кнопку на нижней панели.

В верхнем меню можно реализовать возможность изменения имени файла, удаления файла с запросом, добавления картинки в «Избранное», отправка картинки на печать (с помощью стандартного диалогового окна `PrintDialog` и запуск отдельного приложения для редактирования картинки (`System.Diagnostics.Process.Start("mspaint.exe", filename)`);