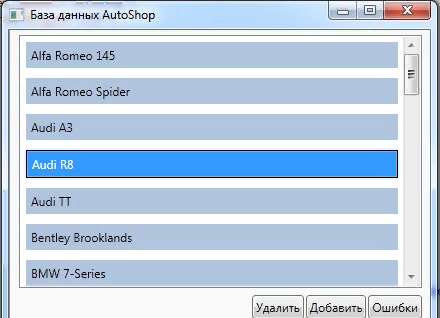
**Практическая работа Стили и шаблоны в WPF**

1. **Разработать стиль для ListBox, который применяет серо-голубой тон к каждому элементу.** Для отделения индивидуальных элементов друг от друга (вместо общего сливающегося фона) стиль также должен добавлять некоторое пространство под поля.

Разработать триггер, изменяющий цвет фона и добавляющий сплошную рамку, когда свойство ListBoxItem.IsSelected получает значение true.



**Ключ к заданию:**

Стили списков — они позволяют применить набор характеристик форматирования к каждому из индивидуальных элементов списка. Это важно, потому что система привязки данных WPF генерирует объекты-элементы списка автоматически.

В результате не так легко применить нужное форматирование к индивидуальным элементам.

Решение обеспечивает свойство ItemsContainerStyle. Если свойство ItemsContainerStyle установлено, списочный элемент управления передаст его каждому своему элементу при его создании. В случае элемента управления ListBox каждый элемент представлен объектом ListBoxItem. (В ComboBox это ComboBoxItem и т.д.) Таким образом, любой стиль, который применяется посредством свойства ListBox.ItemContainerStyle, используется для установки свойств каждого объекта ListBoxItem.

<ListBox Name="lstCars" Margin="5">

<ListBox.ItemContainerStyle>

<Style>

<Setter Property="ListBoxItem.Background" Value="LightSteelBlue"/>

<Setter Property="ListBoxItem.Margin" Value="5"/>

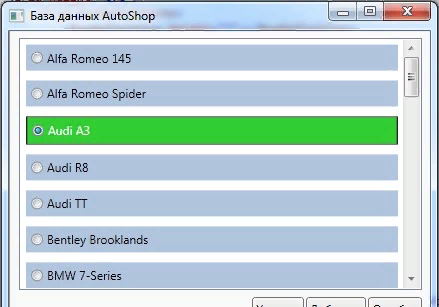
<Setter Property="ListBoxItem.Padding" Value="5"/>

</Style>

</ListBox.ItemContainerStyle>

</ListBox>

1. **Разработать приложение. использующее ListBox с переключателями и флажками**



**Ключ к заданию:**

Базовая техника состоит в замене шаблона элемента управления, используемого в качестве контейнера для каждого элемента списка. Модифицировать свойство ListBox.Template не понадобится, поскольку оно обеспечивает шаблон для ListBox. Вместо этого нужно модифицировать свойство ListBoxItem.Template. Ниже показан шаблон, который необходим для помещения каждого элемента списка в оболочку RadioButton:

<Stylе ...

<Setter Property="Margin" Value="5"/>

<Setter Property="Padding" Value="5"/>

<Setter Property="Background" Value="LightSteelBlue"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type ListBoxItem}">

<Border x:Name="brd" Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}">

<RadioButton Focusable="False" Margin="{TemplateBinding Padding}" Foreground="{TemplateBinding Foreground}" IsChecked="{Binding Path=IsSelected, RelativeSource={RelativeSource TemplatedParent},Mode=TwoWay}">

<ContentPresenter/>

</RadioButton>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsSelected" Value="True">

<Setter Property="Background" Value="LimeGreen"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="BorderBrush" Value="Black"/>

</Trigger>

</ControlTemplate.Triggers>

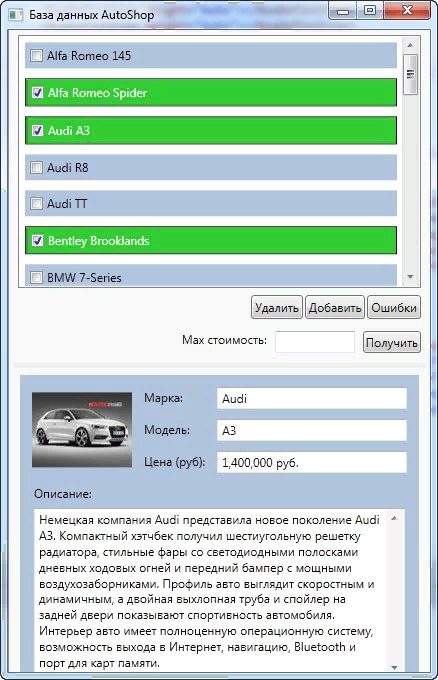
</ControlTemplate>

</Setter.Value>

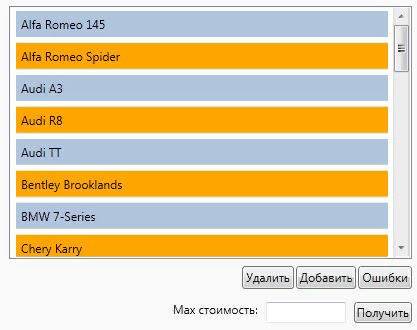
</Setter>

</Style>

Самостоятельно разработать для checkbox



1. **Разработать ListBox с чередующимися строками. Использовать словарь ресурсов.**



**Ключ к заданию:**

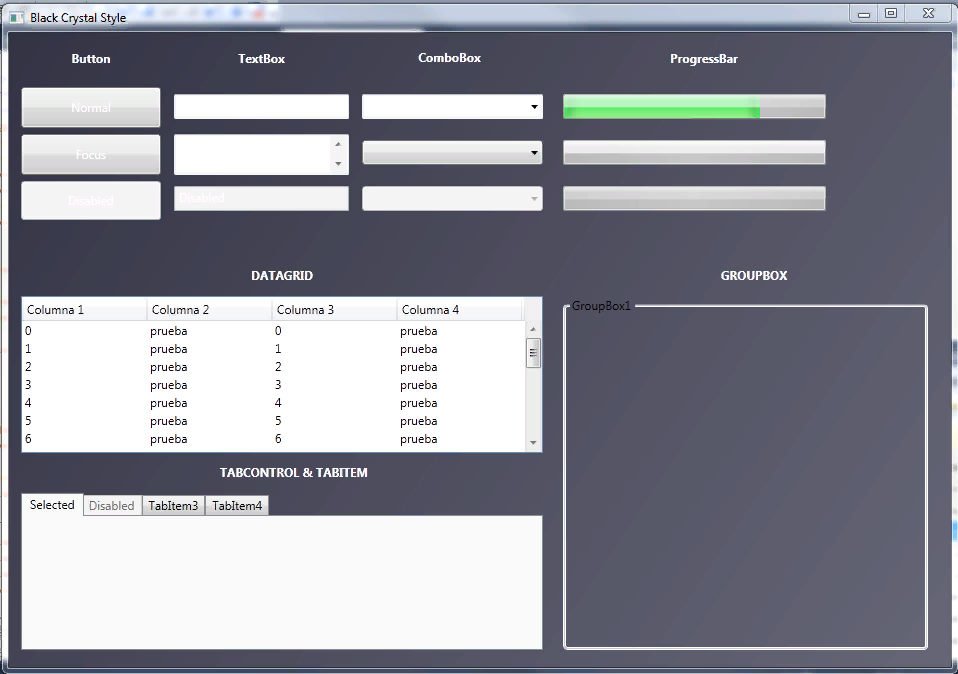
WPF предлагает встроенную поддержку чередующихся элементов через два свойства: AlternationCount и AlternationIndex. Свойство AlternationCount — это количество элементов, формирующих последовательность, после которой стиль переключается. По умолчанию свойство AlternationCount установлено в 0, и чередующееся форматирование не используется.

Если вы установите AlternationCount в 1, стиль будет меняться после каждой строки, что позволит применить шаблон форматирования "четный-нечетный".

Если свойство AlternationCount установлено в 2, то первый элемент ListBoxItem получает AlternationIndex, равный 0, второй — 1, третий — снова 0, четвертый — 1, и т.д. Трюк состоит в использовании в стиле ItemContainerStyle триггера, который проверяет значение AlternationIndex и соответственно варьирует форматирование (вернем стандартный шаблон для ListBoxItem)

1. Разработать интерфейс по образцу:

Исходный вид:



Конечный вид:

