Разработка приложений на платформе .NET

Лекция 20

Персонализация элементов управления

Выбор способа

1. Устраивает функциональность элемента управления и его внешний вид, но не устраивает способ представления данных элементом управления
   - Выбор: **DataTemplate, HierarchicalDataTemplate, ItemsPanelTemplate**
2. Устраивает функциональность какого-либо одного элемента управления, но полностью не устраивает его внешний вид.
   - Выбор: **ControlTemplate**
3. Желаемая функциональность может быть получена путем комбинации нескольких работающих сообща, обособленно элементов управления. Этот блок Элементов управления используется многократно.
   - Выбор: **UserControl**
4. Нет элемента управления или комбинации их, которая бы выполняла необходимую функциональность
   - Выбор: **CustomControl**

Сегодня

- Шаблоны данных
  - DataTemplate
  - HierarchicalDataTemplate
  - ItemsPanelTemplate
- ControlTemplate
- UserControl
- CustomControl

Шаблоны данных

- Когда использовать?
  - Устраивает функциональность элемента управления
  - Устраивает внешний вид
  - Не устраивает способ представления данных
- Например:
  - Отображение списка не UI элементов в списочном элементе управления. Вызывается метод ToString() у каждого элемента и отображается виде простого текста. Нет никакого оформления. Конвертеры и StringFormat несильно помогут.
- Шаблон данных – это фрагмент XAML разметки, который определяет как привязанный объект данных должен быть отображен
- Шаблоны данных
  - DataTemplate – основной шаблон данных
  - HierarchicalDataTemplate – шаблон отображения иерархических данных. Отображение в TreeListView
  - ItemsPanelTemplate – задание контейнера компоновки в списочных элементах управления

Демонстрация

DataTemplate

Шаблоны данных

- Поддерживают:
  - Элементы управления содержимым (наследники от **ContentControl**)
    - Свойство: **ContentTemplate**
    - Определяет отображение того, что помещается в свойство **Content**
  - Элементы управления списками (наследники от **ItemsControl**)
    - Свойство: **ItemTemplate**
    - Определяет отображение каждого элемента списка в коллекции

```
<ListBox ItemsSource="{Binding Path=productList}">
   <ListBox.ItemTemplate>
      <DataTemplate>
         <TextBlock Text="{Binding Path=Name}"/>
      </DataTemplate>
   </ListBox.ItemTemplate>
</ListBox>
```

Шаблон данных как ресурс

В коллекции ресурсов:

```
<DataTemplate  x:Key="dt">
        <Image Source={Binding Path=ImageUri}/>
</DataTemplate>
```

В списочном элементе управления:

….

ItemTemplate="{StaticResource dt}"

Осуществит привязку свойства **Source** рисунка к пути к файлу

Триггеры в шаблонах данных

- Также как и в стилях шаблоны данных поддерживают триггеры
  - Trigger
  - DataTrigger
  - MuliTrigger
  - MultiDataTrigger
  - EventTrigger

```
<DataTemplate>
  <Border Name="border" BorderThickness="1"
            BorderBrush="SteelBlue" Background="LightGray">
  ...
  </Border>
  <DataTemplate.Triggers>
      <DataTrigger Binding="{Binding Path=Cost}" Value="100">
        <Setter  TargetName="border" Property="Background" Value="Red"/>
     </DataTrigger>
   </DataTemplate.Triggers>
 </DataTemplate>
```

Демонстрация

Триггеры в DataTemplate

Автоматически применяемые шаблоны данных

- Как и для стилей, если шаблон данных определён в ресурсах и при этом не задан **x:Key** и задан **DataType**, то такой шаблон автоматически будет применятся (вместо вызова **ToString**()) для отображения данных указанного типа (если конечно явно не задано иное)

```
<Grid.Resources>
        <DataTemplate   DataType="{x:Type DAL:Product}">
                    ....
        </DataTemplate>
</Grid.Resources>
```

Демонстрация

Автоматическое применение DataTemplate

Выбор способа отображения

- ItemsContol выбирает способ отображения каждого элемента данных в следующем порядке:
  1. Если задано свойство DisplayMemberPath в ItemsContol, то используется указанное свойство в привязанном объекте
  2. Если задан шаблон данных, то используется он
  3. Используется селектор шаблонов
     - Класс наследник от DataTemplateSelector
     - С переопределенным методом DataTemplate SelectTemplate(object item, DependencyObject container)

HierarchicalDataTemplate

- Служит для задания отображения иерархических данных
- Содержит свойство **ItemsSource**, задающее коллекцию дочерних объектов
- Свойство **ItemTemplate** задает шаблон данных для отображения следующего нижележащего уровня. Нижележащий уровень может быть задан с помощью **HierarchicalDataTemplate** или с помощью **DataTemplate**. И т.д.

```
<HierarchicalDataTemplate   ItemsSource="{Binding Path=Orders}">
    <TextBlock Text="{Binding Name}"/>
    <HierarchicalDataTemplate.ItemTemplate>
       <DataTemplate ……../>
    </HierarchicalDataTemplate.ItemTemplate>
 </HierarchicalDataTemplate>
```

```
<HierarchicalDataTemplate   ItemsSource="{Binding Path=Orders}">
    <TextBlock Text="{Binding Name}"/>
    <HierarchicalDataTemplate.ItemTemplate>
       <HierarchicalDataTemplate ……../>
    </HierarchicalDataTemplate.ItemTemplate>
 </HierarchicalDataTemplate>
```

Компоновка ItemsControl

- Свойство **ItemsPanel** задает контейнер компоновки, который используется для расположения элементов списка
- Обычно по умолчанию используется **VirtualizingStackPanel**
- Можно изменить контейнер компоновки по умолчанию задав свойство **ItemsPanel** и указав другой шаблон **ItemsPanelTemplate**

```
<ListBox ScrollViewer.HorizontalScrollBarVisibility="Disabled">
<ListBox.ItemsPanel>
   <ItemsPanelTemplate>
      <WrapPanel/>
   </ItemsPanelTemplate>
 </ListBox.ItemsPanel>
</ListBox>
```

Демонстрация

ItemsPanelTemplate

Сегодня

- Шаблоны данных
  - DataTemplate
  - HierarchicalDataTemplate
  - ItemsPanelTemplate
- ControlTemplate
- UserControl
- CustomControl

Шаблон элемента управления

- Когда использовать?
- Устраивает функциональность какого-либо одного элемента управления, но полностью не устраивает его внешний вид.
  - Выбор: **ControlTemplate**

- Элементы управления **WPF** спроектированы таким образом, чтобы полностью отделить функциональность элемента от его визуального оформления
- Чтобы по новому оформить пользовательский интерфейс элемента **WPF**, нужно создать новый шаблон элемента управления
- Шаблон элемента – **XAML**, описывающий оформление элемента управления
- Шаблон элемента управления определяется в элементе **<ControlTemplate>**
- Любой элемент управления имеет свойство **Template**, при задании которого элемент управления перестает использовать свой прежний вид и отображается как задано в **ControlTemplate** (**Template** – свойство зависимости задаваемое стилями темы по умолчанию. Т.е. темы задают автоматические стили, в которых определены шаблоны по умолчанию для элементов управления)

```
<Button Content="Button" Width="100" Height="30">
    <Button.Template>
        <ControlTemplate>
            <Rectangle Fill="RoyalBlue"/>
        </ControlTemplate>
    </Button.Template>
</Button>
```

Демонстрация

ControlTemplate

ControlTemplate

- Для отображения содержимого элемента управления в шаблоне элемента управления используются элементы
  - <**ContentPresenter**> для элементов управления содержимым (для отображения свойства Content)
  - <**ItemsPresenter**> для элементов управлени элементами (для отображения свойства Items)
- Для правильной работы <…Presenter> необходимо указать атрибут **TargetType** у ControlTemplate

```
<ControlTemplate TargetType="{x:Type Button}">
 <Border BorderBrush="Chocolate" BorderThickness="3">
  <Grid>
   <Rectangle Fill="RoyalBlue"/>
   <ContentPresenter VerticalAlignment="Center"
                     HorizontalAlignment="Center"/>
  </Grid>
 </Border>
</ControlTemplate>
```

Шаблон как ресурс

- Обычно шаблоны определяются в ресурсах, что их можно было использовать многократно
- Часто шаблоны используются в автоматически применяемых стилях, чтобы применить шаблон для всех элементов управления определенного типа

```
<Window.Resources>
    <ControlTemplate x:Key="ButtonTemplate"
                                TargetType="{x:Type Button}">
            …..
    </ControlTemplate>
</Window.Resources>

<Button Template="{StaticResource ButtonTemplate}"
        Content="Button" Width="100" Height="30"/>
```

Триггеры в шаблоне

- Триггеры задаются в коллекции **ControlTemplate.Triggers**
  - Trigger
  - DataTrigger
  - MuliTrigger
  - MultiDataTrigger
  - EventTrigger
- Триггеры необходимо задавать в конце **ControlTempplate**, чтобы они имели доступ к объектам шаблона
- В объектах **Triggers** можно определять объекты **Animation**

Демонстрация

Триггеры в ControlTemplate

Свойства родителя шаблона

- Шаблон можно настраивать, задавая свойства элемента управления, использующего этот шаблон.
- Для настройки шаблона используется **TemplateBinding**
- **TemplateBinding** – облегченный объект **Binding**, для привязки свойств в шаблоне. Если его функциональности не хватает можно использовать полноценный объект **Binding**

```
<ControlTemplate x:Key="ButtonTemplate“  TargetType="{x:Type Button}">
  <Border BorderThickness="{TemplateBinding BorderThickness}">
                                  …
<Button Template="{StaticResource ButtonTemplate}"
            BorderThickness="3">
```

Другой способ, использование **Binding**

```
Fill="{Binding RelativeSource={RelativeSource TemplatedParent},
                  Path=Background}“
```

Демонстрация

Привязки в ControlTemplate

Применение шаблонов в стиле

- Можно использовать стили для автоматического применения шаблона всем элементам данного типа

```
<Style TargetType="{x:Type Button}">
   <Setter Property="Template"
           Value="{StaticResource
                   ButtonTemplate}"/>
</Style>
```

Части шаблона

- Не всегда возможно полностью отделить код от визуального представления.
- Код иногда должен обращаться к части визуального представления
- Такие части обозначаются специальными именами **PART_xxx**
- При переопределении шаблонов необходимо для поддержки работоспособности задавать такие же имена своим частям шаблона

```
<ControlTemplate>
        <Slider Name="PART_RedSlider" …/>
</ControlTemplate>
```

Сегодня

- Шаблоны данных
  - DataTemplate
  - HierarchicalDataTemplate
  - ItemsPanelTemplate
- ControlTemplate
- UserControl
- CustomControl

User и Custom Controls

- Пользовательские элементы управления
  - Состоят из связанных вместе элементов управления с общей функциональностью в общем пользовательском интерфейсе
  - Наследуют класс **UserControl**
  - Шаблон проекта в **VS WPF User Control**
- Настраиваемые (**Custom**) элементы управления
  - Определяют собственное визуальное оформления и функциональность (код и **XAML**)
  - Обычно наследуют класс **Control**, **ContentControl**, **ItemControl**, **Panel**. Могут быть унаследованы от конкретного класса, например. **Button**
  - Шаблон проекта в **VS WPF Custom Control Library**

Демонстрация

UserControl