Разработка приложений на платформе .NET

Лекция 18

Ресурсы.
Стили.
Анимация.

Сегодня

- Ресурсы
- Стили
- Анимация

Ресурсы в WPF

- Позволяют определять объекты, доступные другим элементам **WPF** в пользовательском интерфейсе
- Ресурсный объект определяется один раз, а может использоваться несколько раз
- Обычно определяются в **XAML**, но могут определятся и в коде.
- Объект любого типа может быть определен как ресурс.
- Каждый элемент **WPF** определяет коллекцию **Resources**.
- Коллекция **Resources** доступна самому элементу управления, в котором она определена, и элементам управления в его визуальном дереве.
- Чаще используют коллекцию **Windows.Resources**

Ресурсы в WPF

```
<Window.Resources>
    <LinearGradientBrush x:Key="lgBrush" StartPoint="0,0"
                                                    EndPoint="1,1">
      <GradientStop Color="Blue" Offset="0"/>
      <GradientStop Color="Green" Offset="0.5"/>
      <GradientStop Color="Red" Offset="1"/>
    </LinearGradientBrush>
</Window.Resources>
```

x:Key – Задает имя ресурса, на который должны ссылаться желающие использовать этот ресурс.
x:Key – должен быть уникален в той коллекции ресурсов, где определен, а не во всем документе

Использование:

```
<Button Background="{StaticResource lgBrush}“>Hello</Button>
<Button Background="{DynamicResource lgBrush}“>Hello</Button>
```

StaticResource – ресурс определяется (ищется) только один раз
DynamicResource – ресурс определяется каждый раз, когда это необходимо (больше накладных расходов)
Ресурсы ищутся в коллекции ресурсов текущего элемента управления и в коллекциях ресурсов элементов выше в визуальном дереве до первого найденного ресурса с подходящим ключом.

Демонстрация

Ресурсы

Доступ из кода

- Коллекция **Resources** – свойство **FrameworkElement**, а следовательно имеется и у любого его наследника.
- **Resources** – словарь (IDictionary)

- Добавление в словарь элемента. Метод **Add**

```
Resources.Add("StartDate", DateTime.Now);
myButton.Resources.Add("StartDate", DateTime.Now);
```

- Доступ к ресурсу элемента – как к словарю

```
DateTime start = (DateTime)Resources["StartDate"];
DateTime end = (DateTime)grid.Resources[“EndDate"];
```

- Поиск ресурса в словарях вверх по визуальному дереву

```
DateTime dt2 = (DateTime)FindResource("StartDate");
```

- Безопасный поиск ресурса в словарях вверх по визуальному дереву

```
Brush brush = (Brush)TryFindResource("MyBrush");
Brush redBrush = myButton.TryFindResource(“RedBrush") as Brush;
```

- Изменение ресурса

```
grid.Resources["myBrush"] = Brushes.Blue;
```

Место заведения ресурсов

- На уровне элемента
  - Доступно элементу и всем ниже лежащим элементам в его визуальном дереве
- На уровне окна
  - Доступно всем элементам в окне
- На уровне приложения
  - Доступно всем элементам во всех окнах
- В отдельных словарях ресурсов
  - Доступно в том месте, где подключен словарь
  - Может быть доступно нескольким приложениям

Словари ресурсов

Ресурсы могут хранятся в отдельном файле, словаре. **XAML**
Могут использоваться в любом месте приложения
Словарь (**XAML** файл):

```
<ResourceDictionary>
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <LinearGradientBrush x:Key="lgBrush1" StartPoint="0,0" EndPoint="1,1">
    <GradientStop Color="Blue" Offset="0"/>
    <GradientStop Color="Green" Offset="0.5"/>
  </LinearGradientBrush>
  <LinearGradientBrush x:Key="lgBrush2" StartPoint="1,1" EndPoint="0,0">
    <GradientStop Color="LightSkyBlue" Offset="0"/>
    <GradientStop Color="LightCoral" Offset="1"/>
  </LinearGradientBrush>
</ResourceDictionary>
```

Использование словарей

Подключение словарей

```
<Window.Resources>
    <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="MyDictionary.xaml"/>
      </ResourceDictionary.MergedDictionaries>
      <ImageBrush x:Key="imBrush" ImageSource="Chrysanthemum.jpg"/>
    </ResourceDictionary>
  </Window.Resources>
```

Использование ресурсов

```
  <Grid Background="{StaticResource lgBrush}" >
    <Button Foreground="{StaticResource imBrush}"
        Background="{StaticResource revlgBrush}/>
  </Grid>
```

Демонстрация

Словари ресурсов

Сегодня

- Ресурсы
- Стили
- Анимация

Стили

- Стиль – это коллекция свойств, которые могут быть применены к объекту.
- Определяют общий набор характеристик
- Могут определять характеристики не связанные с форматированием. Любое **Dependency Property.**
- Поддерживают триггеры
- Стили устанавливают первоначальные характеристики, которые могут переопределяться.

Демонстрация

Стили

Создание стиля

Стили, обычно создаются в ресурсах

```
<Window.Resources>
   <Style x:Key="MyButtonStyle" TargetType="Button">
         <Setter Property="Height" Value="40"/>
         <Setter Property="Margin" Value="3,0,0,5"/>
         <Setter Property="Background" Value="LightBlue"/>
   </Style>
</Window.Resources>
```

Использование: `<Button Style="{StaticResource MyButtonStyle}/>`

Могут создаваться и напрямую в элементе:

```
<Button.Style>
     <Style>
         <Setter Property="Margin" Value="3"/>
     </Style>
</Button.Style>
```

Установка свойств

- Коллекция **Setters**
- Свойства класса **Setter**:
  - **Property** – какое свойство устанавливаем
  - **Value** – значение

```
<Setter Property="Margin" Value="3,0,0,5"/>
```

- Если сложное значение, то можно устанавливать и так:

```
<Setter Property="Foreground">
    <Setter.Value>
        <ImageBrush ImageSource="a.jpg"/>
    </Setter.Value>
</Setter>
```

Стиль

- Один стиль может применятся для разных типов элементов (не рекомендуется). Необходимо указать для какого типа задается свойство.

```
<Style x:Key="MyStyle">
        <Setter Property="TextBox.Height" Value="40"/>
        <Setter Property=‘‘Button.Height" Value="50"/>
</Style>
  ....
<Button Style="{StaticResource MyStyle}"/>
<TextBox Style="{StaticResource MyStyle}"/>
```

- Если указывается TargetType, то все свойства подразумеваются для этого типа

```
<Style x:Key="MyButtonStyle" TargetType="Button">
```

Автоматическое задание стиля

- Свойство **x:Key** у стиля должно быть задано всегда, чтобы на него можно было сослаться.
- Но. Если свойство **x:Key** не задано, то стиль автоматически применяется для всех элементов типа заданного в свойстве **TargetType**

```
<Style TargetType="Button">
    ……
</Style>
```

- Можно отменить автоматическое применение стиля к элементу задав свое свойство **Style** другим стилем или **{x:Null}**.
- Элемент может переопределить свойство, установленное стилем, явно задав его значение (это возможно благодаря возможностям **DependencyProperty**).

Наследование стилей

- Свойство BaseOn – может указывать на стиль от которого наследуется этот стиль.
- Может дополнять и переопределять свойства заданные в родительском стиле.

```
<Style x:Key="MyButtonStyle" TargetType="Button">
    <Setter Property="Height" Value="40"/>
</Style>
<Style x:Key="MeButtonStyle2" TargetType="Button"
       BasedOn="{StaticResource MyButtonStyle}">
        ..............
</Style>
```

- Наследник должен быть предназначен для того же типа (т.е. иметь такое же значение TargetType)

Итого. Свойства стиля

- Коллекция **Setters**
- Коллекция **Resources**
- Свойство **BasedOn**
- Свойство **TargetType**
- Коллекция **Triggers**

Триггеры

- Внесение простых изменений в зависимости от различных событий
- Декларативно
- Добавляются в коллекцию Style.Triggers

```
<Style>
    <Setter …..>
    <Style.Triggers>
        <Trigger  Property="Button.IsFocused"  Value="true">
                <Setter  Property="Height"  Value="60"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

Демонстрация

Триггеры

Типы триггеров

- **Trigger** – Триггер свойств. Наблюдает за свойством и активизируется, когда значение свойства будет равно **Value**
- **MultiTrigger** – Мультитриггер. Наблюдает за множеством свойств и активизируется, когда значение всех свойств будет равно значениям **Value**
- **DataTrigger** – Триггер данных. Наблюдает за связанным свойством (**Binding**).
- **MultiDataTrigger** – отслеживает множество связанных свойств
- **EventTrigger** – триггер событий. Инициирует серию объектов **Action** при порождении указанного события.

Простой триггер

- Свойство, за которым происходит наблюдение указывается в свойстве **Property=“свойство”**
- Значение свойства, при котором должен активироваться триггер, указывается в свойстве **Value.**
- **Setters** – коллекция установщиков **Setter**, которые устанавливают свойства элемента в случае активации триггера.
- Как только триггер становится неактивным, элементу управления возвращается его исходный вид.

```
<Style x:Key="myStyle" TargetType="Button">
    <Style.Setters>
        <Setter Property="Background" Value="LightBlue"/>
    </Style.Setters>
    <Style.Triggers>
        <Trigger Property="Button.IsFocused" Value="true">
            <Setter Property="Height" Value="60"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

MultiTrigger

- Аналогично **Trigger**, но отслеживается несколько свойств одновременно.
- Отслеживаемые свойств задаются в коллекции **MultiTrigger.Conditions** объектами **Condition**
- Триггер срабатывает только когда выполнены все условия в **Conditions**

```
<Style.Triggers>
    <MultiTrigger>
      <MultiTrigger.Conditions>
        <Condition Property="IsFocused" Value="True"/>
        <Condition Property="IsMouseOver" Value="True"/>
      </MultiTrigger.Conditions>
     <MultiTrigger.Setters>
      <Setter Property="Width" Value="60"/>
     </MultiTrigger.Setters>
   </MultiTrigger>
</Style.Triggers>
```

Триггеры данных

- DataTrigger и MultiDataTrigger

- Такие же как и Trigger и MultiTrigger, но отслеживают связанные данные.
- Вместо свойства Property имеется свойство Binding

```
<DataTrigger Binding="{Binding CustomerName}“
                Value="Иван">
    <Setter Property="Background" Value="Green"/>
</DataTrigger>
```

Триггер события

- **EventTrigger**
- В отличии от других триггеров, которые отслеживают значение свойства и сопоставляют его с указанным значением, триггеры событий указывают события и активизируются при его порождении.
- Содержит коллекцию **Actions** вместо коллекции **Setters**. Коллекция **Actions** содержит набор действий, которые должны быть выполнены при активации триггера. Анимация
- Событие задается в свойстве **RoutedEvent** тригера
- Действия задаются в коллекции **Actions**

```
<Style.Triggers>
    <EventTrigger RoutedEvent="Button.MouseEnter">
       <EventTrigger.Actions>
          <BeginStoryboard>
              <Storyboard>
               <DoubleAnimation Storyboard.TargetProperty="Width"
                                From="60" To="80" Duration="0:0:0.2"/>
            </Storyboard>
          </BeginStoryboard>
        </EventTrigger.Actions>
    </EventTrigger>
</Style.Triggers>
```

Сегодня

- Ресурсы
- Стили
- Анимация

Анимация

- Изменение свойств в течении промежутка времени
- Анимировать можно практически все свойства элемента управления
- Для анимации используются классы **Animation** (более 40 классов)
- Классы **Animation**:
  - Линейные анимации
  - Анимации по ключевым кадрам
  - Анимация на основе путей

Анимация

- Линейная. Классы имеют формат имени
- <Имя_типа>**Animation**
  - Например: **DoubleAnimation**, **ColorAnimation**, **Int32Animation**

- Анимация на основе ключевых кадров изменяет свойство, используя несколько точек маршрута – ключевых кадров. Поток анимации запускается и проходит через каждый ключевой кадр.
  <Имя_типа>**AnimationUsingKeyFrames**
  - Например: **DoubleAnimationUsingKeyFrames**

- Анимация на основе путей использует объект **Path**. Используется для анимации свойств, связанных с перемещение объектов.
  <Имя_типа>**AnimationUsingPath**
  - Например: **DoubleAnimationUsingPath**

Демонстрация

Анимация

Класс Storyboard

- Организует анимацию в пользовательском интерфейсе и управляет ею.
- Содержит коллекцию **Children**, содержащую объекты **Animation.**
- Все объекты **Animation**, создаваемые в **XAML**, должны содержаться в объекте **Storyboard.**
- Если какое-либо свойство в **Animation** не заполнено, то используется свойство из **Storyboard**
- Все дочерние классы **Animation** запускаются одновременно
- **Storyboard**
  - Набор анимаций для одного элемента
  - **TargetName** = "button1" – Конечный элемент для анимации.
  - **TargetProperty** = "Button.Width" – Анимируемое свойство
- Свойства **TargetName** и **TargetProperty** являются прикрепленными, поэтому могут быть установлены в классе **Animation**

Свойства Animation и Storyboard

- **Duration** – длительность анимации
- **AutoReverse** – воспроизведение в обратном порядке после завершения
- **BeginTime** – время начала запуска относительно времени анимации.
  - 0:0:5 – 5 секунд
- **AccelerationRatio** – задает ускорение в начале анимации
- **DecelerationRatio** – задает замедление в конце анимации
- **RepeateBehavior** – задает повторяемость анимации

- Свойства линейной анимации:
- **From** - начальное значение свойства для анимации. Если не указано, то используется текущее свойство элемента
- **To** – конечное значение свойства
- **By** – значение приращение. Используется, если **To** не задано.

Анимация элемента

- Анимация может быть добавлена непосредственно в коллекцию **Triggers** элемента управления

```
<Button>
  <Button.Triggers>
     <EventTrigger RoutedEvent="Button.MouseEnter"
                                          SourceName="button1">
       <EventTrigger.Actions
        <BeginStoryboard>
         <Storyboard>
           <DoubleAnimation  Storyboard.TargetName="button1"
                                          Storyboard.TargetProperty="Width"
                          From="60" To="80" Duration="0:0:0.2"/>
        </Storyboard>
        </BeginStoryboard>
       </EventTrigger.Actions>
    </EventTrigger>
  </Button.Triggers>
</Button>
```

Управление анимацией

- Для того, чтобы управлять **Storyboard**, необходимо задать ему свойство **Name**
- Использовать классы
  - **PauseStoryboard** - для приостановки **Storyboard**
  - **ResumeStoryboard** - для возобновления **Storyboard**
  - **StopStoryboard** - для остановки **Storyboard**
  - **SeekStoryboard** – для перемотки
  - **SkipStoryboardToFill** – для перемотки в конец
  - **SetStoryboardSpeedRatio** – для ускорения или замедления анимации
- Для запуска анимации используется **BeginStoryboard**

Управление анимацией

```
<Rectangle Name="myRectangle“ Width="100" Height="20"
Margin="12,0,0,5" Fill="#AA3333FF" HorizontalAlignment="Left" />
<StackPanel Orientation="Horizontal" Margin="0,30,0,0">
   <Button Name="BeginButton">Begin</Button>
   <Button Name="PauseButton">Pause</Button>
   <Button Name="ResumeButton">Resume</Button>
   <Button Name="SeekButton">Seek</Button>
   <Button Name="SkipToFillButton">Skip To Fill</Button>
   <Button Name="SetSpeedRatioButton">Triple Speed</Button>
   <Button Name="StopButton">Stop</Button>

   <StackPanel.Triggers>
     <EventTrigger RoutedEvent="Button.Click"
                   SourceName="BeginButton">
       <BeginStoryboard Name="BeginSB">
         <Storyboard >
           <DoubleAnimation
                   Storyboard.TargetName="myRectangle"
                   Storyboard.TargetProperty="Width"
                   Duration="0:0:5" From="100" To="500" />
         </Storyboard>
       </BeginStoryboard>
     </EventTrigger>

     <EventTrigger RoutedEvent="Button.Click"
                   SourceName="PauseButton">
       <PauseStoryboard BeginStoryboardName=" BeginSB " />
     </EventTrigger>

     <EventTrigger RoutedEvent="Button.Click" SourceName="ResumeButton">
       <ResumeStoryboard BeginStoryboardName="BeginSB" />
     </EventTrigger>

     <EventTrigger RoutedEvent="Button.Click" SourceName="SeekButton">
       <SeekStoryboard BeginStoryboardName="BeginSB" Offset="0:0:1"
                       Origin="BeginTime" />
     </EventTrigger>

     <EventTrigger RoutedEvent="Button.Click" SourceName="SkipToFillButton">
       <SkipStoryboardToFill BeginStoryboardName="BeginSB" />
     </EventTrigger>

     <EventTrigger RoutedEvent="Button.Click" SourceName="StopButton">
       <StopStoryboard BeginStoryboardName="BeginSB" />
     </EventTrigger>

     <EventTrigger RoutedEvent="Button.Click" SourceName="SetSpeedRatioButton">
      <SetStoryboardSpeedRatio SpeedRatio="3" BeginStoryboardName="BeginSB" />
     </EventTrigger>

   </StackPanel.Triggers>
</StackPanel>
```

https://msdn.microsoft.com/en-us/library/vstudio/ms744905(v=vs.110).aspx

Демонстрация

Анимация элемента

Демонстрация

Какую анимацию не стоит делать

Лекция Романа Здебского “Лучшие практики разработки производительных и интерактивных приложений на WPF”
https://channel9.msdn.com/Blogs/TechDays-Russia/-WPF-20081128110400