# «Использование стилей в WPF-приложениях»

Стили WPF позволяют определять внешний вид и поведение для группы элементов управления. Рассмотрим пример WPF-приложения:

|  |
| --- |
| **Пример 1 Код XAML** |
| <StackPanel Orientation="Horizontal" VerticalAlignment="Top">  <Button Background="DarkBlue" Foreground="White" FontFamily="Verdana" Padding="5"  Margin="5">Открыть</Button>  <Button Background="DarkBlue" Foreground="White" FontFamily="Verdana" Padding="5" Margin="5">Обработать</Button>  <Button Background="DarkBlue" Foreground="White" FontFamily="Verdana" Padding="5" Margin="5">Сохранить</Button>  <Button Padding="5" Margin="5">Закрыть</Button>  </StackPanel> |
| **Результат** |
|  |

Для трех элементов управления повторяются одни и те же атрибуты с одинаковыми значениями. Как при программировании наличие повторяющегося кода является плохим стилем, так и при разработке WPFинтерфейса повторение участков XAML-кода не приветствуется. В данном случае правильным решением является определение внешнего вида кнопки с помощью стиля и задание этого стиля для трех кнопок. По функциональности стили похожи на каскадные таблицы стилей CSS для HTML-файлов. Обычно стили, как и другие ресурсы приложения, определяются в ресурсах окна:

<Window.Resources>

...

<Style>

</Style>

...

</Window.Resources>

Свойство Resources объявлено для класса FrameworkElement, поэтому ресурсы можно объявить

практически для любого элемента управления:

<StackPanel.Resources>

...

<Style>

</Style>

...

</StackPanel.Resources>

<Button.Resources>

...

<Style>

</Style>

...

</Button.Resources>

Область действия стиля, объявленного в ресурсах какого-либо элемента управления, распространяется только на этот элемент управления и его дочерние элементы управления. Следует учесть, что статический ресурс должен быть определен в коде разметки **перед** ссылкой на него.

Стиль определяется с помощью элемента Style и может использоваться для определения:

* значений атрибутов;
* обработчиков событий;
* **триггеров**, меняющих атрибуты элемента управления при возникновении каких-либо событий или при изменении каких-либо свойств;
* **шаблонов**, переопределяющих внешний вид элементов управления.

Ключевые свойства, определенные в классе Style:

* TargetType – тип элемента, для которого определяется данный стиль;
* BasedOn – родительский стиль (позволяет задавать иерархические стили);
* Setters – коллекция объектов Setter или EventSetter, которые предназначены для установления значений свойств и обработчиков событий;
* Triggers – коллекция триггеров;
* Resourses – коллекция ресурсов, которые необходимо использовать с ресурсами.

Объект Setter определяет значение одного свойства элемента управления:

<Setter Property="НАЗВАНИЕ\_СВОЙСТВА" Value="ЗНАЧЕНИЕ" />

Рассмотрим модифицированную версию примера 1 с использованием стилей:

|  |
| --- |
| **Пример 2 Код XAML** |
| <Window.Resources>  <Style TargetType="Button">  <Style.Setters>  <Setter Property="Background" Value="DarkBlue" />  <Setter Property="Foreground" Value="White" />  <Setter Property="FontFamily" Value="Verdana" />  <Setter Property="Padding" Value="5" />  <Setter Property="Margin" Value="5" />  </Style.Setters>  </Style>  </Window.Resources>  <StackPanel Orientation="Horizontal" VerticalAlignment="Top">  <Button>Открыть</Button>  <Button>Обработать</Button>  <Button>Сохранить</Button>  <Button Padding="5" Margin="5">Закрыть</Button>  </StackPanel> |
| **Результат** |
|  |

Допускается не указывать элемент Style.Setters:

<Style TargetType="Button">

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

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

<Setter Property="FontFamily" Value="Verdana" />

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

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

</Style>

В данном примере стиль был применен ко всем кнопкам окна (тип элементов управления определен в атрибуте TargetType). Если для элемента Style определить атрибут x:Key с именем стиля, то данный стиль будет определен только к тем кнопкам, для которых указано имя стиля в атрибуте Style с помощью расширения разметки StaticResource:

Style="{StaticResource ResourceKey=DocButton}"

Пример стиля, который применяется к трем кнопкам и не применяется к кнопке «Закрыть».

|  |
| --- |
| **Пример 3 Код XAML** |
| <Window.Resources>  <Style TargetType="Button" x:Key="DocButton">  <Setter Property="Background" Value="DarkBlue" />  <Setter Property="Foreground" Value="White" />  <Setter Property="FontFamily" Value="Verdana" />  <Setter Property="Padding" Value="5" />  <Setter Property="Margin" Value="5" />  </Style>  </Window.Resources>  <StackPanel Orientation="Horizontal" VerticalAlignment="Top">  <Button Style="{StaticResource ResourceKey=DocButton}">Открыть</Button>  <Button Style="{StaticResource ResourceKey=DocButton}">Обработать</Button>  <Button Style="{StaticResource ResourceKey=DocButton}">Сохранить</Button>  <Button Padding="5" Margin="5">Закрыть</Button>  </StackPanel> |
| **Результат** |
|  |

Допускается вместо {StaticResource ResourceKey=DocButton} указывать {StaticResource DocButton}, т.к. ResourceKey является единственным параметром для расширения разметки StaticResource.

## Задание 1

Проверьте, какое значение имеет больший приоритет: значение свойства, указанное в стиле, или значение атрибута элемента.

Свойства BasedOn класса Style позволяет определять иерархические стили. В этом свойстве с помощью расширения разметки StaticResource указывается родительский стиль. Дочерний стиль наследует все свойства родительского стиля, которые он может дополнить или переопределить. Пример определения дочернего стиля ActiveDocButton на основе родительского стиля DocButton.

|  |
| --- |
| **Пример 4 Код XAML** |
| <Window.Resources>  <Style TargetType="Button" x:Key="DocButton">  <Setter Property="Background" Value="DarkBlue" />  <Setter Property="Foreground" Value="White" />  <Setter Property="FontFamily" Value="Verdana" />  <Setter Property="Padding" Value="5" />  <Setter Property="Margin" Value="5" /> </Style>  <Style BasedOn="{StaticResource DocButton}" TargetType="Button" x:Key="ActiveDocButton"> <Setter Property="Background" Value="DarkRed" /> |
| </Style>  </Window.Resources>  <StackPanel Orientation="Horizontal" VerticalAlignment="Top">  <Button Style="{StaticResource ActiveDocButton}">Открыть</Button>  <Button Style="{StaticResource DocButton}">Обработать</Button>  <Button Style="{StaticResource DocButton}">Сохранить</Button>  <Button Padding="5" Margin="5">Закрыть</Button>  </StackPanel> |
| **Результат** |
|  |

Объект EventSetter определяет имя функции-обработчика для события:

<EventSetter Event="НАЗВАНИЕ\_СОБЫТИЯ" Handler="ИМЯ\_ФУНКЦИИ" />

Пример задания одного обработчика для всех кнопок окна:

|  |
| --- |
| **Пример 5 Код XAML** |
| <Window.Resources>  <Style TargetType="Button">  <Style.Setters>  <Setter Property="Margin" Value="5" />  <EventSetter Event="Click" Handler="Button\_Click" />  </Style.Setters>  </Style>  </Window.Resources>  <StackPanel Orientation="Horizontal" VerticalAlignment="Top">  <Button>Открыть</Button>  <Button>Обработать</Button>  <Button>Сохранить</Button>  </StackPanel> |
| **Код C#** |
| private void Button\_Click(object sender, RoutedEventArgs e) {  MessageBox.Show("Button is clicked");  } |
| **Результат** |
|  |

## Задание 2

Модифицируйте WPF-приложение, разработанное в 3-ей лабораторной работе: используйте стили для однотипных элементов управления.

## Задание 3

Разработайте приложение MultiEdit для одновременной работы с несколькими текстами. Окно должно быть разделено на две части с одинаковыми градиентами. В каждой части окна должно быть несколько многострочных текстовых полей: одно из них большого размера с крупным шрифтом, а остальные маленького размера с мелким шрифтом. То текстовое окно, в котором пользователь набирает текст, должно быть большим, остальные текстовые поля должны быть маленькими. Внешний вид однотипных элементов управления должен определяться с помощью стилей.

Изменить стиль элемента управления в коде можно следующим образом:

(sender as FrameworkElement).Style = (Style)Resources["ИМЯ\_СТИЛЯ"];