**Создание стиля для элемента управления (WPF .NET)**

В Windows Presentation Foundation (WPF) можно настраивать внешний вид элемента управления, используя собственный перенастраиваемый стиль. Стили можно применять глобально ко всему приложению, отдельным окнам и страницам или непосредственно к элементам управления.

**Создание стиля**

Style обеспечивает удобный способ применения набора значений свойств к одному или нескольким элементам. Стиль можно использовать для любого элемента, производного от FrameworkElement или FrameworkContentElement, например Window или Button.

Чаще всего стиль объявляется как ресурс в разделе Resources файла XAML. Так как стили являются ресурсами, для них действуют те же правила определения области, что и для всех других ресурсов. Проще говоря, то, где вы объявляете стиль, влияет на то, где этот стиль может быть применен. Например, если объявить стиль в корневом элементе файла XAML определения приложения, стиль может использоваться в любом месте приложения.

XAML

<Application x:Class="IntroToStylingAndTemplating.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:IntroToStylingAndTemplating"

StartupUri="WindowExplicitStyle.xaml">

<Application.Resources>

<ResourceDictionary>

<Style x:Key="Header1" TargetType="TextBlock">

<Setter Property="FontSize" Value="15" />

<Setter Property="FontWeight" Value="ExtraBold" />

</Style>

</ResourceDictionary>

</Application.Resources>

</Application>

Стиль, объявленный в одном из файлов XAML приложения, можно использовать только в этом файле XAML.

XAML

<Window x:Class="IntroToStylingAndTemplating.WindowSingleResource"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:IntroToStylingAndTemplating"

mc:Ignorable="d"

Title="WindowSingleResource" Height="450" Width="800">

<Window.Resources>

<Style x:Key="Header1" TargetType="TextBlock">

<Setter Property="FontSize" Value="15" />

<Setter Property="FontWeight" Value="ExtraBold" />

</Style>

</Window.Resources>

<Grid />

</Window>

Стиль состоит из дочерних элементов <Setter>, которые задают свойства для элементов, к которым применяется стиль. Обратите внимание, что в приведенном выше примере стиль настроен на применение к типам TextBlock с использованием атрибута TargetType. Стиль задаст 15 для FontSize и ExtraBold для FontWeight. Добавьте <Setter> для каждого свойства, которое изменяет стиль.

**Неявное применение стиля**

Style обеспечивает удобный способ применения набора значений свойств к нескольким элементам. Рассмотрим, к примеру, следующие элементы TextBlock и их внешний вид по умолчанию в окне.

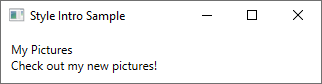
XAMLКопировать

<StackPanel>

<TextBlock>My Pictures</TextBlock>

<TextBlock>Check out my new pictures!</TextBlock>

</StackPanel>



Внешний вид по умолчанию можно изменить, задав такие свойства, как FontSize и FontFamily, непосредственно в каждом элементе TextBlock. Но если вы хотите, чтобы элементы TextBlock могли совместно использовать некоторые свойства, можно создать Style в разделе Resources файла XAML, как показано ниже.

XAMLКопировать

<Window.Resources>

<!--A Style that affects all TextBlocks-->

<Style TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

<Setter Property="FontFamily" Value="Comic Sans MS"/>

<Setter Property="FontSize" Value="14"/>

</Style>

</Window.Resources>

Если вы задаете стилю TargetType для типа TextBlock и опускаете атрибут x:Key, стиль применяется ко всем элементам TextBlock, областью действия которых является стиль, представленный по сути файлом XAML.

Теперь элементы TextBlock выглядят следующим образом.

**Явное применение стиля**

Если вы добавляете в стиль атрибут x:Key со значением, стиль больше не будет неявно применяться ко всем элементам TargetType. Стиль будет применяться только к тем элементам, которые явно ссылаются на него.

Ниже приведен стиль из предыдущего раздела, который объявлен с атрибутом x:Key.

XAMLКопировать

<Window.Resources>

<Style x:Key="TitleText" TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

<Setter Property="FontFamily" Value="Comic Sans MS"/>

<Setter Property="FontSize" Value="14"/>

</Style>

</Window.Resources>

Чтобы применить этот стиль, задайте для свойства элемента Style значение x:Key, используя расширение разметки StaticResource, как показано ниже.

XAMLКопировать

<StackPanel>

<TextBlock Style="{StaticResource TitleText}">My Pictures</TextBlock>

<TextBlock>Check out my new pictures!</TextBlock>

</StackPanel>

Обратите внимание, что стиль применен к первому элементу TextBlock, а второй элемент TextBlock остается неизменным. Неявный стиль из предыдущего раздела был изменен на стиль, объявляющий атрибут x:Key. То есть единственный элемент, затронутый стилем, — это тот, который непосредственно ссылался на стиль.

Явно или неявно примененный стиль становится запечатанным и не подлежит изменению. Чтобы динамически изменить стиль, который уже был применен, нужно создать новый стиль, который заменит существующий. Дополнительные сведения см. в описании свойства IsSealed.

Можно создать объект, который выбирает стиль, который нужно применить, основываясь на собственной логике. Ознакомьтесь с примером для класса StyleSelector.

**Применение стиля программным способом**

Чтобы назначить именованный стиль элементу программным способом, возьмите стиль из коллекции ресурсов и присвойте его свойству Style этого элемента. Обратите внимание, что элементы в коллекции ресурсов имеют тип Object. Поэтому полученный стиль необходимо привести к System.Windows.Style, прежде чем назначать его свойству Style. Например, следующий код задает стиль TextBlock с именем textblock1 определенному стилю TitleText.

C#Копировать

textblock1.Style = (Style)Resources["TitleText"];

**Расширение стиля**

Возможно, вы хотите, чтобы два элемента TextBlock совместно использовали некоторые значения свойств, такие как FontFamily и HorizontalAlignment с выравниванием по центру. При этом текст **Мои рисунки** должен иметь некоторые дополнительные свойства. Это можно сделать, создав новый стиль на основе первого стиля, как показано ниже.

XAMLКопировать

<Window.Resources>

<!-- .... other resources .... -->

<!--A Style that affects all TextBlocks-->

<Style TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

<Setter Property="FontFamily" Value="Comic Sans MS"/>

<Setter Property="FontSize" Value="14"/>

</Style>

<!--A Style that extends the previous TextBlock Style with an x:Key of TitleText-->

<Style BasedOn="{StaticResource {x:Type TextBlock}}"

TargetType="TextBlock"

x:Key="TitleText">

<Setter Property="FontSize" Value="26"/>

<Setter Property="Foreground">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">

<LinearGradientBrush.GradientStops>

<GradientStop Offset="0.0" Color="#90DDDD" />

<GradientStop Offset="1.0" Color="#5BFFFF" />

</LinearGradientBrush.GradientStops>

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

XAMLКопировать

<StackPanel>

<TextBlock Style="{StaticResource TitleText}" Name="textblock1">My Pictures</TextBlock>

<TextBlock>Check out my new pictures!</TextBlock>

</StackPanel>

Теперь этот стиль TextBlock выравнивается по центру, а также использует шрифт Comic Sans MS с размером 26 и цвет переднего плана LinearGradientBrush, как показано в примере. Обратите внимание, что значение FontSize базового стиля переопределено. Если несколько Setter указывают на одно и то же свойство в Style, приоритет имеет свойство Setter, объявленное последним.

Ниже показано, как будут выглядеть элементы TextBlock.

Стиль TitleText расширяет стиль, который был создан для типа TextBlock, на который ссылается BasedOn="{StaticResource {x:Type TextBlock}}". Вы также можете расширить стиль с x:Key, используя x:Key стиля. Например, если существовал стиль с именем Header1, который вы хотите расширить, используйте BasedOn="{StaticResource Header1}".

**Связь между свойством TargetType и атрибутом x:Key**

Как было показано выше, присвоив свойству TargetType значения TextBlock без назначения стиля x:Key, вы примените стиль ко всем элементам TextBlock. В этом случае для x:Key неявно устанавливается значение {x:Type TextBlock}. Это означает, что при явном присвоении x:Key значения, отличающегося от {x:Type TextBlock}, Style не применяется ко всем элементам TextBlock автоматически. Вместо этого необходимо явным образом применить стиль (используя значение x:Key) ко всем элементам TextBlock. Если стиль находится в разделе ресурсов и вы не задаете для стиля свойство TargetType, задайте атрибут x:Key.

Свойство TargetType не только предоставляет значение по умолчанию для x:Key, но и указывает тип, к которому применяются свойства метода задания. Если вы не указываете TargetType, необходимо определите свойства в объектах Setter, используя имя класса и синтаксис Property="ClassName.Property". Например, вместо установки Property="FontSize" необходимо задать для Property значение "TextBlock.FontSize" или "Control.FontSize".

Также обратите внимание, что многие элементы управления WPF состоят из других элементов управления WPF. Если создать стиль, который применяется ко всем элементам управления типа, можно получить непредвиденные результаты. Например, если вы создаете стиль, предназначенный для типа TextBlock в Window, этот стиль будет применен ко всем элементам управления TextBlock в окне, даже если TextBlock является частью другого элемента управления, например ListBox.