# Создание шаблона для элемента управления (WPF.NET)

В Windows Presentation Foundation (WPF) можно настраивать визуальную структуру и функциональные возможности существующего элемента управления с помощью своего собственного шаблона многократного использования. Шаблоны можно применять глобально ко всему приложению, отдельным окнам и страницам или непосредственно к элементам управления. Большинство сценариев, в которых требуется создать новый элемент управления, можно реализовать, создав вместо этого новый шаблон для существующего элемента управления.

В этой статье будет показано, как создать новый шаблон ControlTemplate для элемента управления Button.

## Когда следует создавать ControlTemplate

Элементы управления имеют много свойств, таких, например, как Background, Foreground и FontFamily. Эти свойства управляют различными аспектами внешнего вида элемента управления, но с помощью них можно внести не так много изменений. Например, для элемента управления CheckBox можно задать синий цвет фона с помощью свойства Foreground и курсив с помощью свойства FontStyle. Если вы хотите внести такие изменения внешнего вида элемента управления, которые не предусмотрены его свойствами, можно создать шаблон ControlTemplate.

В большинстве пользовательских интерфейсов кнопка обычно выглядит как прямоугольник с текстом. Если вы хотите создать круглую кнопку, можно создать новый элемент управления, который наследует от кнопки или воссоздает функциональность кнопки. И вдобавок этот новый пользовательский элемент управления будет иметь круглую форму.

Вы можете не создавать новые элементы управления, а просто настроить визуальный макет существующего элемента управления. Например, для круглой кнопки можно создать шаблон ControlTemplate с желаемым визуальным макетом.

С другой стороны, если вам нужен элемент управления с новыми функциями, другими свойствами и новыми параметрами, лучше создать новый UserControl.

## Предварительные требования

Создайте новое приложение WPF и в файле MainWindow. XAML (или в другом окне по своему усмотрению) задайте следующие свойства **< элемента Window >** :

| **Свойство** | **Значение** |
| --- | --- |
| **Title** | Template Intro Sample |
| **SizeToContent** | WidthAndHeight |
| **MinWidth** | 250 |

Задайте для содержимого **< элемента Window >** следующий код XAML:

XAMLКопировать

<StackPanel Margin="10">

<Label>Unstyled Button</Label>

<Button>Button 1</Button>

<Label>Rounded Button</Label>

<Button>Button 2</Button>

</StackPanel>

В итоге файл MainWindow.xaml должен выглядеть следующим образом.

XAML

<Window x:Class="IntroToStylingAndTemplating.Window1"

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="Template Intro Sample" SizeToContent="WidthAndHeight" MinWidth="250">

<StackPanel Margin="10">

<Label>Unstyled Button</Label>

<Button>Button 1</Button>

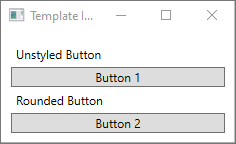
<Label>Rounded Button</Label>

<Button>Button 2</Button>

</StackPanel>

</Window>

Если запустить приложение, оно будет выглядеть так.



## Создание шаблона ControlTemplate

Чаще всего ControlTemplate объявляется как ресурс в разделе Resources файла XAML. Так как шаблоны являются ресурсами, для них действуют те же правила определения области, что и для всех других ресурсов. Проще говоря, то, где вы объявляете шаблон, влияет на то, где этот шаблон может быть применен. Например, если вы объявите шаблон в корневом элементе XAML-файла определения приложения, этот шаблон можно будет использовать в любом месте вашего приложения. Если вы определяете шаблон в окне, его смогут использовать только элементы управления из этого окна.

Для начала добавьте элемент Window.Resources в свой файл MainWindow.xaml:

XAML

<Window x:Class="IntroToStylingAndTemplating.Window2"

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="Template Intro Sample" SizeToContent="WidthAndHeight" MinWidth="250">

<Window.Resources>

</Window.Resources>

<StackPanel Margin="10">

<Label>Unstyled Button</Label>

<Button>Button 1</Button>

<Label>Rounded Button</Label>

<Button>Button 2</Button>

</StackPanel>

</Window>

Создайте новый **< объект ControlTemplate >** со следующими заданными свойствами:

| **Свойство** | **Значение** |
| --- | --- |
| **x:Key** | roundbutton |
| **TargetType** | Button |

Этот шаблон элемента управления будет простым:

* корневой элемент этого элемента управления, Grid;
* Ellipse, чтобы кнопка отображалась круглой;
* ContentPresenter для вывода указанного пользователем содержимого кнопки.

XAMLКопировать

<ControlTemplate x:Key="roundbutton" TargetType="Button">

<Grid>

<Ellipse Fill="{TemplateBinding Background}" Stroke="{TemplateBinding Foreground}" />

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

### TemplateBinding

Создавая новый шаблон ControlTemplate, вы можете захотеть использовать общие свойства для изменения внешнего вида элемента управления. Расширение разметки TemplateBinding привязывает свойство элемента из шаблона ControlTemplate к общему свойству, которое задается элементом управления. При использовании расширения TemplateBinding свойства элемента управления могут действовать в качестве параметров шаблона. Это означает, что при задании свойства элемента управления соответствующее значение передается в элемент, который содержит TemplateBinding.

### Ellipse

Обратите внимание, что **Fill** свойства **< и элемента Ellipse >** привязаны к свойствам элемента управления Foreground и Background . **Stroke**

### ContentPresenter

< Элемент ContentPresenter > также добавляется в шаблон. Так как этот шаблон предназначен для кнопки, необходимо учитывать, что эта кнопка наследует от ContentControl. Кнопка представляет содержимое элемента. Можно задать что-либо внутри кнопки, например обычный текст или даже другой элемент управления. Оба следующих варианта — правильные кнопки:

XAML

<Button>My Text</Button>

<!-- and -->

<Button>

<CheckBox>Checkbox in a button</CheckBox>

</Button>

В обоих приведенных выше примерах текст и флажок задаются как свойство Button.Content. Что угодно, так как содержимое может быть представлено с **< помощью ContentPresenter >**, это именно то, что делает шаблон.

Если ControlTemplate применяется к типу ContentControl, такому как Button, выполняется поиск ContentPresenter в дереве элементов. Если ContentPresenter обнаруживается, шаблон автоматически привязывает свойство Content элемента управления к элементу ContentPresenter.

## Использовать шаблон

Найдите кнопки, которые были объявлены в начале этой статьи.

XAML

<StackPanel Margin="10">

<Label>Unstyled Button</Label>

<Button>Button 1</Button>

<Label>Rounded Button</Label>

<Button>Button 2</Button>

</StackPanel>

Задайте в свойстве Template второй кнопки ресурс roundbutton:

XAML

<StackPanel Margin="10">

<Label>Unstyled Button</Label>

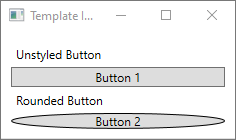
<Button>Button 1</Button>

<Label>Rounded Button</Label>

<Button Template="{StaticResource roundbutton}">Button 2</Button>

</StackPanel>

Если теперь вы запустите проект и посмотрите на результат, то фон кнопки будет иметь овальную форму.



Вы, конечно, заметили, что кнопка не круглая, а вытянутая. Из-за того, как элемент **Ellipse > работает, он всегда расширяется, чтобы заполнить доступное пространство. <** Сделайте круг единообразным, изменив свойства кнопки **width** и **height** на то же значение:

XAMLКопировать

<StackPanel Margin="10">

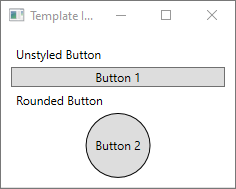
<Label>Unstyled Button</Label>

<Button>Button 1</Button>

<Label>Rounded Button</Label>

<Button Template="{StaticResource roundbutton}" Width="65" Height="65">Button 2</Button>

</StackPanel>



## Добавление триггера

Хотя кнопка с примененным шаблоном выглядит иначе, она ведет себя так же, как и любая другая кнопка. При нажатии кнопки срабатывает событие Click. Однако, как вы могли заметить, при наведении указателя мыши на кнопку визуально ничего не меняется. Все визуальные взаимодействия определяются шаблоном.

В системах динамических событий и свойств, предоставляемых WPF, можно отслеживать значение конкретного свойства, а затем при необходимости изменять стиль шаблона. В данном примере вы будете отслеживать свойство IsMouseOver кнопки. Когда указатель мыши находится над элементом управления, отменяйте стиль **< эллипса >** с помощью нового цвета. Триггер такого типа называется PropertyTrigger.

Чтобы это работало, необходимо добавить в **эллипс > имя <** , на которое вы можете ссылаться. Задайте имя **backgroundElement**.

XAML

<Ellipse x:Name="backgroundElement" Fill="{TemplateBinding Background}" Stroke="{TemplateBinding Foreground}" />

Затем добавьте новый Trigger в коллекцию ControlTemplate.Triggers. Этот триггер будет отслеживать, когда событие IsMouseOver принимает значение true.

XAML

<ControlTemplate x:Key="roundbutton" TargetType="Button">

<Grid>

<Ellipse x:Name="backgroundElement" Fill="{TemplateBinding Background}" Stroke="{TemplateBinding Foreground}" />

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="true">

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

Затем добавьте **< в триггер метод задания >** , который изменяет свойство **< Fill эллипса >** на новый цвет. **><**

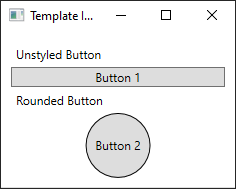
XAML

<Trigger Property="IsMouseOver" Value="true">

<Setter Property="Fill" TargetName="backgroundElement" Value="AliceBlue"/>

</Trigger>

Запустите проект. Обратите внимание, что при наведении указателя мыши на кнопку Цвет **< эллипса >** изменяется.



## Использование VisualState

Визуальные состояния определяются и активируются элементом управления. Например, при наведении указателя мыши на элемент управления активируется состояние CommonStates.MouseOver. Изменения свойств можно анимировать на основе текущего состояния элемента управления. В предыдущем разделе **< пропертитригжер >** использовался для изменения переднего плана кнопки на AliceBlue , когда IsMouseOver свойство имело true значение. Теперь вместо этого создайте визуальное состояние, которое анимирует изменение этого цвета, обеспечивая плавный переход.

Чтобы преобразовать **< пропертитригжер >** в анимированное визуальное состояние, сначала удалите **< элемент ControlTemplate. triggers >** из шаблона.

XAMLКопировать

<ControlTemplate x:Key="roundbutton" TargetType="Button">

<Grid>

<Ellipse x:Name="backgroundElement" Fill="{TemplateBinding Background}" Stroke="{TemplateBinding Foreground}" />

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

Затем в **< корне сетки >** шаблона элемента управления добавьте **< элемент VisualStateManager. VisualStateGroups >** с **< VisualStateGroup >** для CommonStates . Определите два состояния — Normal и MouseOver.

XAML

<ControlTemplate x:Key="roundbutton" TargetType="Button">

<Grid>

<VisualStateManager.VisualStateGroups>

<VisualStateGroup Name="CommonStates">

<VisualState Name="Normal">

</VisualState>

<VisualState Name="MouseOver">

</VisualState>

</VisualStateGroup>

</VisualStateManager.VisualStateGroups>

<Ellipse x:Name="backgroundElement" Fill="{TemplateBinding Background}" Stroke="{TemplateBinding Foreground}" />

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

Любые анимации, определенные в **< VisualState >** , применяются при активации этого состояния. Создайте анимации для каждого состояния. Анимации помещаются внутри **< элемента Storyboard >** .

Это состояние анимирует заливку эллипса, восстанавливающую цвет Background элемента управления.

XAML

<Storyboard>

<ColorAnimation Storyboard.TargetName="backgroundElement"

Storyboard.TargetProperty="(Shape.Fill).(SolidColorBrush.Color)"

To="{TemplateBinding Background}"

Duration="0:0:0.3"/>

</Storyboard>

* MouseOver

Это состояние анимирует изменение цвета Background эллипса на новый цвет: Yellow.

XAML

<Storyboard>

<ColorAnimation Storyboard.TargetName="backgroundElement"

Storyboard.TargetProperty="(Shape.Fill).(SolidColorBrush.Color)"

To="Yellow"

Duration="0:0:0.3"/>

</Storyboard>

Теперь объект **< ControlTemplate >** должен выглядеть следующим образом.

XAML

<ControlTemplate x:Key="roundbutton" TargetType="Button">

<Grid>

<VisualStateManager.VisualStateGroups>

<VisualStateGroup Name="CommonStates">

<VisualState Name="Normal">

<Storyboard>

<ColorAnimation Storyboard.TargetName="backgroundElement"

Storyboard.TargetProperty="(Shape.Fill).(SolidColorBrush.Color)"

To="{TemplateBinding Background}"

Duration="0:0:0.3"/>

</Storyboard>

</VisualState>

<VisualState Name="MouseOver">

<Storyboard>

<ColorAnimation Storyboard.TargetName="backgroundElement"

Storyboard.TargetProperty="(Shape.Fill).(SolidColorBrush.Color)"

To="Yellow"

Duration="0:0:0.3"/>

</Storyboard>

</VisualState>

</VisualStateGroup>

</VisualStateManager.VisualStateGroups>

<Ellipse Name="backgroundElement" Fill="{TemplateBinding Background}" Stroke="{TemplateBinding Foreground}" />

<ContentPresenter x:Name="contentPresenter" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

Запустите проект. Обратите внимание, что при наведении указателя мыши на кнопку Цвет **< эллипса >** анимируется.

