В WPF важное место занимают **ресурсы**. В данном случае под ресурсами подразумеваются не дополнительные файлы (или **физические ресурсы**), как, например, аудиофайлы, файлы с изображениями, которые добавляются в проект. Здесь речь идет о **логических ресурсах**, которые могут представлять различные объекты - элементы управления, кисти, коллекции объектов и т.д. Логические ресурсы можно установить в коде XAML или в коде C# с помощью свойства Resources. Данное свойство опредлено в базовом классе **FrameworkElement**, поэтому его имеют большинство классов WPF.

В чем смысл использования ресурсов? Они повышают эффективность: мы можем определить один раз какой-либо ресурс и затем многократно использовать его в различных местах приложения. В связи с этим улучшается поддержка - если возникнет необходимость изменить ресурс, достаточно это сделать в одном месте, и изменения произойдут глобально в приложении.

Свойство **Resources** представляет объект **ResourceDictionary** или словарь ресурсов, где каждый хранящийся ресурс имеет определенный ключ.

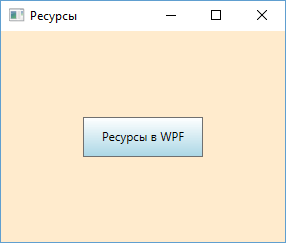
### Определение ресурсов

Определим ресурс окна и ресурс кнопки:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | <Window x:Class="ResourcesApp.MainWindow"          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:ResourcesApp"          mc:Ignorable="d"          Title="Ресурсы" Height="250" Width="300">      <Window.Resources>          <SolidColorBrush x:Key="redStyle" Color="BlanchedAlmond" />            <LinearGradientBrush x:Key="gradientStyle" StartPoint="0.5,1" EndPoint="0.5,0">              <GradientStop Color="LightBlue" Offset="0" />              <GradientStop Color="White" Offset="1" />          </LinearGradientBrush>      </Window.Resources>      <Grid Background="{StaticResource redStyle}">          <Button x:Name="button1" MaxHeight="40" MaxWidth="120" Content="Ресурсы в WPF" Background="{StaticResource gradientStyle}">              <Button.Resources>                  <SolidColorBrush x:Key="darkStyle" Color="Gray" />              </Button.Resources>          </Button>      </Grid>  </Window> |

Здесь у окна определяются два ресурса: redStyle, который представляет объект SolidColorBrush, и gradientStyle, который представляет кисть с линейным градиентом. У кнопки определен один ресурс darkStyle, представляющий кисть SolidColorBrush. Причем каждый ресурс обязательно имеет свойство **x:Key**, которое и определяе ключ в словаре.

А в свойствах Background соответственно у грида и кнопки мы можем применить эти ресурсы: Background="{StaticResource gradientStyle}" - здесь после выражения StaticResource идет ключ применяемого ресурса.



### **Управление ресурсами в коде C#**

Добавим в словарь ресурсов окна градиентную кисть и установим ее для кнопки:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | // определение объекта-ресурса  LinearGradientBrush gradientBrush = new LinearGradientBrush();  gradientBrush.GradientStops.Add(new GradientStop(Colors.LightGray, 0));  gradientBrush.GradientStops.Add(new GradientStop(Colors.White, 1));    // добавление ресурса в словарь ресурсов окна  this.Resources.Add("buttonGradientBrush", gradientBrush);    // установка ресурса у кнопки  button1.Background = (Brush)this.TryFindResource("buttonGradientBrush");  // или так  //button1.Background = (Brush)this.Resources["buttonGradientBrush"]; |

С помощью свойства Add() объект кисти и его произвольный ключ добавляются в словарь. Далее с помощью метода TryFindResource() мы пытаемся найти ресурс в словаре и установить его в качестве фона. Причем, так как этот метод возвращает object, необходимо выполнить приведение типов.

Всего у ResourceDictionary можно выделить следующие методы и свойства:

* Метод **Add(string key, object resource)** добавляет объект с ключом key в словарь, причем в словарь можно добавить любой объект, главное ему сопоставить ключ
* Метод **Remove(string key)** удаляет из словаря ресурс с ключом key
* Свойство **Uri** устанавливает источник словаря
* Свойство **Keys** возвращает все имеющиеся в словаре ключи
* Свойство **Values** возвращает все имеющиеся в словаре объекты

Для поиска нужного ресурса в коллекции ресурсов у каждого элемента определены методы FindResource() и TryFindResource(). Она оба возвращают ресурс, соответствующий определенному ключу. Единственное различие между ними состоит в том, что FindResource() генерирует исключение, если ресурс с нужным ключом не был найден. А метод TryFindResource() в этом случае просто возвращает null.

### **Разделяемые ресурсы**

Когда один и тот же ресурс используется в разных местах, то фактически мы используем один и тот же объект. Однако это не всегда желательно. Иногда необходимо, чтобы примение ресурса к разным объектам различалось. То есть нам необходимо, чтобы при каждом применении создавался отдельный объект ресурса. В этом случае мы можем изспользовать выражение x:Shared="False":

|  |  |
| --- | --- |
| 1 | <SolidColorBrush x:Shared="False" x:Key="redStyle" Color="BlanchedAlmond" /> |

### Примеры использования ресурсов

Рассмотрим еще пару примеров применения ресурсов. К примеру, если мы хотим, чтобы ряд кнопок обладал одинаковыми свойствами, то мы можем определить одну общую кнопку в качестве ресурса:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | <Window x:Class="ResourcesApp.MainWindow"          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:ResourcesApp"          mc:Ignorable="d"          Title="Ресурсы" Height="250" Width="300">      <Window.Resources>          <SolidColorBrush x:Key="redStyle" Color="BlanchedAlmond" />            <LinearGradientBrush x:Key="gradientStyle" StartPoint="0.5,1" EndPoint="0.5,0">              <GradientStop Color="LightBlue" Offset="0" />              <GradientStop Color="White" Offset="1" />          </LinearGradientBrush>          <Button x:Key="resButton" Background="{StaticResource gradientStyle}">              <TextBlock Text="OK" FontSize="16" />          </Button>      </Window.Resources>      <Grid Background="{StaticResource redStyle}">          <Button Width="80" Padding="0" Height="40" HorizontalContentAlignment="Stretch" VerticalContentAlignment="Stretch" Content="{StaticResource resButton}" />      </Grid>  </Window> |

Другой пример - определение списка объектов для списковых элементов:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | <Window x:Class="ResourcesApp.MainWindow"          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:ResourcesApp"            xmlns:sys="clr-namespace:System;assembly=mscorlib"          xmlns:col="clr-namespace:System.Collections;assembly=mscorlib"            mc:Ignorable="d"          Title="Ресурсы" Height="250" Width="300">      <Window.Resources>          <col:ArrayList x:Key="phones">              <sys:String>iPhone 6S</sys:String>              <sys:String>Nexus 6P</sys:String>              <sys:String>Lumia 950</sys:String>              <sys:String>Xiaomi MI5</sys:String>          </col:ArrayList>      </Window.Resources>      <Grid>          <ListBox ItemsSource="{StaticResource phones}" />      </Grid>  </Window> |

