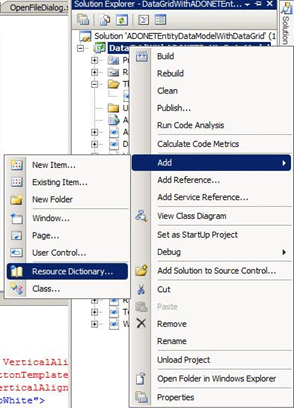
# ResourceDictionary in WPF

The items in a ResourceDictionary are not immediately processed when application code is loaded by a XAML loader. Instead, the ResourceDictionary persists as an object, and the individual values are processed only when they are specifically requested.  
  
The ResourceDictionary class is not derived from DictionaryBase. Instead, the ResourceDictionary class implements IDictionary but relies on a Hashtable internally.  
  
In Extensible Application Markup Language (XAML), the ResourceDictionary class is typically an implicit collection element that is the object element value of several Resources properties, when given in property element syntax. For details on implicit collections in XAML, see XAML Syntax Terminology. An exception is when you want to specify a merged dictionary; for details, see Merged Resource Dictionaries.  
  
Another possible XAML usage is to declare a resource dictionary as a discrete XAML file, and either load it at run time with Load or include it in a (full-trust) project as a resource or loose file. In this case, ResourceDictionary can be declared as an object element, serving as the root element of the XAML. You must map the appropriate XML namespace values (default for the WPF namespace and typically x: for the XAML namespace) onto the Resource Dictionary element if you plan to use it as the root element. Then you can add child elements that define the resources, each with an x:Key value.  
  
**Getting Started:**   
  
First of all make a new WPF Application and add a new ResourceDictionary file. I am putting my ResourceDictionary File in the Themes directory. Like figure1.  
  
  
  
Figure1.  
  
Here is my ResourceDictionary file XML code:  
  
<ResourceDictionary xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"   
                    xmlns:col="clr-namespace:System.Collections;assembly=mscorlib"   
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">   
   <!--This is for gradient button style-->   
    <ControlTemplate x:Key="buttonTemplate" TargetType="{x:Type Button}">   
        <Grid>   
            <Ellipse x:Name="outerCircle">   
                <Ellipse.Fill>   
                    <LinearGradientBrush StartPoint="0,0" EndPoint="0,1">   
                        <GradientStop Offset="0" Color="Blue" />   
                        <GradientStop Offset="1" Color="Red" />   
                    </LinearGradientBrush>   
                </Ellipse.Fill>   
            </Ellipse>   
            <Ellipse x:Name="innerCircle" RenderTransformOrigin=".5,.5">   
                <Ellipse.RenderTransform>   
                    <ScaleTransform ScaleX=".8" ScaleY=".8" />   
                </Ellipse.RenderTransform>   
                <Ellipse.Fill>   
                    <LinearGradientBrush StartPoint="0,0" EndPoint="0,1">   
                        <GradientStop Offset="0" Color="Red" />   
                        <GradientStop Offset="1" Color="Blue" />   
                    </LinearGradientBrush>   
                </Ellipse.Fill>   
            </Ellipse>   
            <Viewbox>   
                <ContentPresenter Margin="{TemplateBinding Padding}" />   
            </Viewbox>   
        </Grid>   
        <ControlTemplate.Triggers>   
            <Trigger Property="IsMouseOver" Value="True">   
                <Setter TargetName="outerCircle" Property="Fill" Value="Orange" />   
            </Trigger>   
            <Trigger Property="IsPressed" Value="True">   
                <Setter Property="RenderTransform" >   
                    <Setter.Value>   
                        <ScaleTransform ScaleX=".9" ScaleY=".9" />   
                    </Setter.Value>   
                </Setter>   
                <Setter Property="RenderTransformOrigin" Value=".5,.5"/>   
            </Trigger>   
        </ControlTemplate.Triggers>   
    </ControlTemplate>

     <!--This style used for normal button style-->   
    <Style x:Key="buttonStyle" TargetType="{x:Type Button}">   
        <Setter Property="Foreground" Value="White" />   
        <Setter Property="Background" Value="Green" />   
        <Setter Property="FontSize" Value="14" />   
    </Style>

     <!--This style used for combo box collection style and list items-->   
    <CollectionViewSource x:Key="myCol">          
        <CollectionViewSource.Source>   
            <col:ArrayList>                  
                <ListBoxItem>Raj Beniwal</ListBoxItem>   
                <ListBoxItem>Vikash Nanda</ListBoxItem>   
                <ListBoxItem>Amit Mishra</ListBoxItem>   
                <ListBoxItem>Ketan Puri</ListBoxItem>   
            </col:ArrayList>   
        </CollectionViewSource.Source>   
   </CollectionViewSource>

     <!--This style used for tab item styles-->   
    <Style x:Key="tabItemStyle" TargetType="{x:Type TabItem}">   
        <Setter Property="BorderBrush" Value="Blue" />   
        <Setter Property="Background" Value="LightPink" />   
        <Setter Property="FontSize" Value="14" />   
        <Setter Property="Foreground" Value="IndianRed" />   
    </Style>      
</ResourceDictionary>

You can initialise the path of Resource Dictionary in App.xaml file.  
  
<Application x:Class="DataGridWithADONETEntityDataModel.App"   
    xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"   
    xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"   
    StartupUri="ResourceDictionary.xaml">   
    <Application.Resources>   
        <ResourceDictionary>   
            <ResourceDictionary.MergedDictionaries>   
                <ResourceDictionary Source="Themes/Dictionary1.xaml" />   
            </ResourceDictionary.MergedDictionaries>   
        </ResourceDictionary>   
    </Application.Resources>   
</Application>

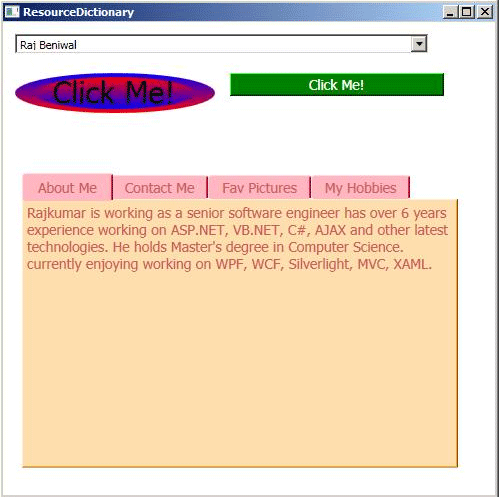
Here is my .xaml code.  
  
<Window x:Class="DataGridWithADONETEntityDataModel.ResourceDictionary"   
    xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"   
    xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"   
         xmlns:res="clr-namespace:DataGridWithADONETEntityDataModel"   
        xmlns:col="clr-namespace:System.Collections;assembly=mscorlib"   
    Title="ResourceDictionary" Height="500" Width="500">      
    <Grid>

 <ComboBox ItemsSource="{Binding Source={StaticResource myCol}}" Height="20"Margin="12,12,66,0" VerticalAlignment="Top"></ComboBox>   
  
<Button Margin="12,51,0,0" Height="40" Width="200" Name="button1"  Template="{StaticResourcebuttonTemplate}" HorizontalAlignment="Left" VerticalAlignment="Top">Click Me!</Button>

 <Button Margin="227,51,51,0" Name="button2"  Style="{StaticResource buttonStyle}" Height="23"VerticalAlignment="Top">Click Me!</Button>   
          
<TabControl Height="295" Margin="12,145,30,22" Name="tabControl1" Width="436"Background="NavajoWhite">

 <TabItem Header="About Me" Name="tabItem1" Style="{StaticResource tabItemStyle}">   
<Grid>   
<TextBlock TextWrapping="Wrap">Rajkumar is working as a senior software engineer has over 6 years experience working on ASP.NET, VB.NET, C#, AJAX and other latest technologies. He holds Master's degree in Computer Science. currently enjoying working on WPF, WCF, Silverlight, MVC, XAML.</TextBlock>   
                </Grid>   
            </TabItem>

<TabItem Header="Contact Me" Name="tabItem2"  Style="{StaticResource tabItemStyle}">   
                <Grid>   
                    <TextBlock TextWrapping="Wrap">I can be reached on at raj2511984 at yahoo.com OR raj2511984@gmail.com</TextBlock>   
                </Grid>   
            </TabItem>   
  
<TabItem Header="Fav Pictures" Name="tabItem3"  Style="{StaticResource tabItemStyle}">   
                <Grid>                      
                    <Image Source="Neha-Dhupia-20091231-001.jpg" Margin="0,0,200,0"></Image>   
                    <Image Source="Neha-Dhupia-20091231-003.jpg"  Margin="180,0,0,0"></Image>                                         
                </Grid>   
            </TabItem>   
              
<TabItem Header="My Hobbies" Name="tabItem4"  Style="{StaticResource tabItemStyle}">   
                <Grid>   
                    <TextBlock TextWrapping="Wrap">MY PASSION'S MUSIC TRANSCENDS BETWEEN HEIGHTS OF HEAVEN AND DEPTHS OF HELL! ONE MINUTE YOU'RE AT THE GREATEST PARTY ON EARTH AND YOUR WORLD IS FILLED WITH JOY, THE NEXT YOURE PLUNGED INTO A DARK DIRTY SCENE FROM A TIM BURTON FILM AND A NIGHTMARE THAT YOU DON'T WANT TO END! IT'S TIME TO START LIVING.</TextBlock>

                </Grid>   
            </TabItem>   
        </TabControl>   
    </Grid>   
</Window>  
  
Now execute the application.  
  


# Using a Resource Dictionary in WPF

By [**Rahman Masudur**](http://www.codeproject.com/script/Membership/View.aspx?mid=2342471), 11 Apr 2009

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | http://s.codeproject.com/script/Ratings/Images/stars-fill-lg.png  http://s.codeproject.com/script/Ratings/Images/stars-empty-lg.png | 4.65 (40 votes) | |  |

Top of Form

## Introduction

In this short article, we are going to see a few tricks about using a **resource** **dictionary** in **WPF**, how we can merge it in XAML, and how we can use it in C#.

## Adding a WPF Resource Dictionary

Since **WPF** applications have rich media and graphics support, reusable styles need to be utilized and in a managed way. We can define the styles in **WPF** XAML files, or perhaps we can manage to accumulate all our useful styles for a particular application in a **resource** **dictionary** file. Adding a **resource** **dictionary** is pretty simple. We have to select the project or folder in Solution Explorer and then right click and select “Add”. We will get a menu item called “**Resource** **Dictionary**”. Clicking on that menu item will popup up the Add New Item wizard with the **ResourceDictionary** Item template selected. Rename the item as you wish.

In a Resouce**Dictionary**, we can keep our custom styles, DataTemplates, ControlTemplates, even custom definitions for Brush, Color, Background and a lot of other stuff. But, the important thing is that we have to assign a key to each of them since it is a **Dictionary**. Or perhaps, we can give names to the styles.

## Using Resource Files in XAML

In this section, we are going to see how we can import a **resource** file to a XAML file for a user control or a Window or a page. Below is provided a simple code listing for demonstration. Since we can have a **resource** **dictionary** for each control, we are going to merge the other **resource** files to the existing **resource** **dictionary**.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/35346/Using-a-Resource-Dictionary-in-WPF)

<Window x:Class="**WPF**Demo.Window1"

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

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

Title="Window1" Height="300" Width="300">

<Window.**Resource**s>

<**ResourceDictionary**>

<**ResourceDictionary**.MergedDictionaries>

<**ResourceDictionary**

Source="**Resource**s/My**ResourceDictionary**.xaml">

</**ResourceDictionary**>

<**ResourceDictionary**

Source="**Resource**s/OthersStyle.xaml">

</**ResourceDictionary**>

</**ResourceDictionary**.MergedDictionaries>

</**ResourceDictionary**>

</Window.**Resource**s>

<Grid>

<Image Source="/**WPF**Demo;component/Images/Add**ResourceDictionary**.jpg"></Image>

</Grid>

</Window>

## Using Resource Files in C#

There can be cases where we need to access a **resource** **dictionary** that we have defined in our project, from C# code. If we have already merged our **resource** **dictionary** in XAML, it is easy to access the inner **resource**s using thecontrol.Find**Resource**("KeyWillGoHere"); method. But ,if we haven’t merged the **resource**s in XAML and we still need to use the **resource** **dictionary**, we have options to use the stuff directly in C# code. Here is a simple code snippet given for better understanding:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/35346/Using-a-Resource-Dictionary-in-WPF)

public partial class Window1 : Window

{

private **ResourceDictionary** my**resourcedictionary**;

private **ResourceDictionary** mystyles;

public Window1()

{

InitializeComponent();

my**resourcedictionary** = new **ResourceDictionary**();

my**resourcedictionary**.Source =

new Uri("/**WPF**Demo;component/**Resource**s/My**ResourceDictionary**.xaml",

UriKind.RelativeOrAbsolute);

mystyles = new **ResourceDictionary**();

mystyles.Source = new Uri("/**WPF**Demo;component/**Resource**s/OthersStyle.xaml",

UriKind.RelativeOrAbsolute);

}

public void ApplyStyle()

{

Style mybuttonstyle = mystyles["MyStyle"] as Style;

Button mybutton = new Button();

mybutton.Style = mybuttonstyle;

}

}

We have used a URI to get hold of our **resource** **dictionary** content. I must mention one thing here that, while defining the URI, the project name goes first, then the relative path. The UriKind option is very important. If we don’t mention the UriKind, it will be unable to parse the URI and find the **resource**. Since this is a **resource** **dictionary**, we have to access the styles using keys, just like in a normal **dictionary**.

Bottom of Form