How To Implement MergedDictionaries For ResourceDictionary In Silverlight 2

Jürgen Bäurle

December 2008

Parago Media GmbH & Co. KG

Introduction

Microsoft's Windows Presentation Foundation (WPF) resources support the very handy merged resource dictionary feature. Unfortunately the Silverlight 2 resources do not support this feature. The feature provides a way to define and split resources into separate files.

This feature is also very helpful for developing custom controls in Silverlight 2. Right now all default style keys must be described in "Themes/Generic.xaml". This file can be large and merged resource dictionaries are a great solution to split the style definitions into manageable pieces.

Merged Resource Dictionary in WPF

In WPF resource dictionaries are merged as follows:

```
<ResourceDictionary>
  <ResourceDictionary.MergedDictionaries>
    <ResourceDictionary Source="MyResourceDictionary1.xaml"/>
    <ResourceDictionary Source="MyResourceDictionary2.xaml"/>
  </ResourceDictionary.MergedDictionaries>
</ResourceDictionary>
```

The files MyResourceDictionary1.xaml and MyResourceDictionary2.xaml contain resources that will be merged into the main dictionary. The property Source defines the location by defining an URI to the file containing the resource dictionary. The URI is defined by the "pack URI scheme" (see MSDN documentation).

Merged Resource Dictionary in Silverlight 2

Since there is no counterpart collection named MergedDictionaries in Silverlight a cutom merging functionality must be developed.

But first have a look at the final result.

The following code shows the Generic.xaml file in the sample project. The file itself defines no styles at all, but it will merge keys for the resource dictionaries defined in CustomControl1.xaml and CustomControl3.xaml into the Generic.xaml dictionary. The dictionary CustomControl1.xaml is then again merging with the resource dictionary CustomControl2.xaml. That's the reason why the two keys "Parago.Windows.Controls.CustomControl1" and "Parago.Windows.Controls.CustomControl2" are defined for the first embedded dictionary.

```
<ResourceDictionary
  xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:parago="clr-namespace:Parago.Windows.Controls;assembly=Parago.Windows.Controls">

  <parago:ResourceDictionary.MergedDictionaries>
    <parago:ResourceDictionary
       Keys="Parago.Windows.Controls.CustomControl1,Parago.Windows.Controls.CustomControl2"
       Source="/Parago.Windows.Controls;component/Themes/Controls/CustomControl1.xaml"/>
    <parago:ResourceDictionary
       Keys="Parago.Windows.Controls.CustomControl3"
       Source="/Parago.Windows.Controls;component/Themes/Controls/CustomControl3.xaml"/>
  </parago:ResourceDictionary.MergedDictionaries>

</ResourceDictionary>
```

The XAML file "CustomControl1.xaml" for instance is defined as follows:

```
<ResourceDictionary
  xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:parago="clr-namespace:Parago.Windows.Controls;assembly=Parago.Windows.Controls">

  <parago:ResourceDictionary.MergedDictionaries>
    <parago:ResourceDictionary
      Keys="Parago.Windows.Controls.CustomControl2"
      Source="/Parago.Windows.Controls;component/Themes/Controls/CustomControl2.xaml"/>
  </parago:ResourceDictionary.MergedDictionaries>

  <Style TargetType="parago:CustomControl1">
    <Setter Property="IsTabStop" Value="False"/>
    <Setter Property="Template">
      <Setter.Value>
        <ControlTemplate TargetType="parago:CustomControl1">
          <Grid>
            <TextBlock>CustomControl3</TextBlock>
          </Grid>
        </ControlTemplate>
      </Setter.Value>
    </Setter>
  </Style>

</ResourceDictionary>
```

The custom ResourceDictionary class calls internally the XamlReader.Load method. In order to correctly merge the resource directories the namespace must be defined completely including the assembly part, otherwise an exception will be thrown.

Implementation

The custom and implementing class, also named “ResourceDictionary”, must first register an new attached property called “MergedDictionaries”. Defining and registering an attached property is a straightforward task:

```
...

public static ResourceDictionary GetMergedDictionaries(DependencyObject d)
{
  if(d == null)
    throw new ArgumentNullException("d");

  return (ResourceDictionary)d.GetValue(MergedDictionariesProperty);
}

public static void SetMergedDictionaries(DependencyObject d, ResourceDictionary dictionary)
{
  if(d == null)
    throw new ArgumentNullException("d");

  d.SetValue(MergedDictionariesProperty, dictionary);
}

public static readonly DependencyProperty MergedDictionariesProperty =
  DependencyProperty.RegisterAttached(
    "MergedDictionaries",
    typeof(ResourceDictionary),
    typeof(ResourceDictionary),
    new PropertyMetadata(new PropertyChangedCallback(OnMergedDictionariesPropertyChanged)));

static void OnMergedDictionariesPropertyChanged(DependencyObject d,
  DependencyPropertyChangedEventArgs e)
{
  ResourceDictionary dictionaryToMerge = e.NewValue as ResourceDictionary;

  if(d is System.Windows.ResourceDictionary)
    dictionaryToMerge.OnMergedDictionariesChanged((d as System.Windows.ResourceDictionary));
}

protected virtual void OnMergedDictionariesChanged(System.Windows.ResourceDictionary
  targetDictionary)
{
  if(targetDictionary == null)
    return;
  if(string.IsNullOrEmpty(Keys))
```

```
    throw new Exception("Keys property is not defined");

  System.Windows.ResourceDictionary dictionaryToMerge = GetResourceDictionary();

  // NOTE: Silverlight 2 does not provide an enumerator or iteration option
  // for resource dictionaries

  foreach(string key in Keys.Split(",".ToCharArray()))
  {
    string kv = key.Trim();

    if(!string.IsNullOrEmpty(kv))
    {
      if(!dictionaryToMerge.Contains(kv))
        throw new Exception(string.Format(
           "Key '{0}' does not exist in resource dictionary '{1}'", kv, Source));

      if(!targetDictionary.Contains(kv))
        targetDictionary.Add(kv, dictionaryToMerge[kv]);
    }
  }
}

...
```

For more information about attached properties in Silverlight see the MSDN documentation or this great article on the SilverlightShow.net website: http://www.silverlightshow.net/items/Attached-Properties-in-Silverlight.aspx.

The Silverlight ResourceDictionary class implementation in namespace System.Windows.Controls does not offer an enumerator. All of those methods will throw exceptions. So in addition to the "Source" property (see above) there is also a need to define a new property called "Keys" of type String. The value is a comma-separated list of keys that shall be merged into the main resource dictionary.

If no key name is defined with x:Key in the merging resource dictionary (e.g. CustomControl1.xaml) the type name including the namepace is used. For type CustomControl1 the key is "Parago.Windows.Controls.CustomControl1".

The method OnMergedDictionariesChanged (see listing above) is called for each new defined resource dictionary added to the MergedDictionaries collection. The method will be executed with a given standard ResourceDictionary object instance into which the keys of the current dictionary definition have to be merged.

The method is calling the helper method GetResourceDictionary (see listing below) to parse and transform the XAML file defined in the Source property into a standard Silverlight ResourceDictionary. Then all defined keys will be merged (added) into the given dictionary called "targetDictionary".

The following listing shows the definition of the GetResourceDictionary method:

```
...

protected virtual System.Windows.ResourceDictionary GetResourceDictionary()
{
  if(Source == null)
    throw new Exception("Source property is not defined");

  StreamResourceInfo resourceInfo = Application.GetResourceStream(Source);

  if(resourceInfo != null && resourceInfo.Stream != null)
  {
    using(StreamReader reader = new StreamReader(resourceInfo.Stream))
    {
      string xaml = reader.ReadToEnd();

      if(!string.IsNullOrEmpty(xaml))
        return XamlReader.Load(xaml) as System.Windows.ResourceDictionary;
    }
  }

  throw new Exception(string.Format("Resource dictionary '{0}' does not exist", Source));
}

...
```

That's it.

Summary

In overall it is simple to implement the feature merged resource dictionary for the current version 2 of Silverlight. Especially for custom control development this feature can be very helpful. The sample project for this article shows how to split the Generic.xaml, needed in custom control development, into several XAML files (dictionaries).

Contact Information

If you have any feedback or suggestions, please feel free to contact me:

Jürgen Bäurle
jbaurle@parago.de
http://www.parago.de/jbaurle

Parago Media GmbH & Co. KG
Im Wengert 3 | 71336 Waiblingen, Germany | Phone +49.7146.861803 | Internet http://www.parago.de