WAF Localization Sample

WPF Application Framework (WAF)

# Note: This document is outdated!

# Introduction

The LocalizationSample demonstrates a way to localize WPF applications. It uses the “old” ResX files for this task because they are well supported by the .NET Framework and Visual Studio.

**Remark:** Microsoft encourages you to use their tool “LocBaml” for localizing WPF applications (see [MSDN](http://msdn.microsoft.com/en-us/library/ms746621.aspx)). Unfortunately, this tool is not production-ready and you don’t get the same comfortable support from Visual Studio.

The LocalizationSample is part of the WPF Application Framework (WAF) [download](http://waf.codeplex.com/).

# Run the sample

1. Open the WpfApplicationFramework solution.
2. Set the LocalizationSample project as StartUp project and start it in Debug mode.
3. The application shows a window with two text boxes and a button. The text shown in the application is in English or German depending on the Windows language settings.
4. The birthday is formatted as short date time format. This formatting is defined in the Windows Regional and Language Options.
5. Close the application.
6. Open the App.config file. In this file you can define the cultures that should be used in the application. You can use one of these settings:  
     
   <value/> Use the culture defined in Windows  
   <value>en-US</value> Define the culture English - US  
   <value>de-DE</value> Define the culture German - Germany  
     
   Just uncomment the culture you like to use and comment the other two culture settings. The xml example below shows the configuration for the German - Germany culture.

<WpfLocalizationDemo.Properties.Settings>

<setting name="Culture" serializeAs="String">

<!--<value>en-US</value>-->

<value>de-DE</value>

<!--<value />-->

</setting>

<setting name="UICulture" serializeAs="String">

<!--<value>en-US</value>-->

<value>de-DE</value>

<!--<value />-->

</setting>

</WpfLocalizationDemo.Properties.Settings>

1. Run the application again and see how the changes affect its appearance.

# CurrentCulture and CurrentUICulture

The difference between CurrentCulture and CurrentUICulture is not that obvious. These two properties represent the “Regional and Language Options” which can be found in the Windows Control Panel.

* CurrentCulture represents the “Format” tab of the “Regional and Language Options”. This setting is responsible for formatting and parsing of values.
* CurrentUICulture represents the second part of the “Keyboards and Languages” tab, the “Display Language”. This property controls which language of the resources is loaded and shown to the user.  
    
  **Remark:** Not every version of Microsoft Windows supports the “Display Language” option. When you don’t see this field then your Windows doesn’t support this option.

# Code Walkthrough

1. Open the WpfApplicationFramework solution.
2. Expand the LocalizationSample project.
3. Open the App.xaml.cs file.
   1. In the constructor we call the InitializeCulture method.
   2. In the InitializeCulture method we try to retrieve the Culture and UICulture property from the application settings file. When they exist then we set the Culture on the CurrentThread.
   3. In the last line of the InitializeCulture method we set the WPF XmlLanguage with the CurrentCulture. This line is necessary that WPF uses the CultureInfo.CurrentCulture settings.

FrameworkElement.LanguageProperty.OverrideMetadata(  
 typeof(FrameworkElement), new FrameworkPropertyMetadata(   
 XmlLanguage.GetLanguage(  
 CultureInfo.CurrentCulture.IetfLanguageTag)));

**Remark:** Windows Forms and Console applications use the CultureInfo.CurrentCulture setting by default. In my opinion, WPF should do the same so that we don’t have to write the previous line in our applications.

1. Open the Properties/Resources.resx file
   1. This file contains the string resources used in the user interface.
   2. Note that the “Access Modifier” was changed from internal to **public**. You find this setting in the Visual Studio Resources Designer toolbar. This is necessary so that you are able to access the resources within XAML.
   3. Besides the Resources.resx file you find the Resources.de.resx file in the same folder. The second resource file contains the resources for the German language.
2. Open the Presentation/ShellWindow.xaml file
   1. The Title property of the Window uses the x:Static markup extension to access one of the Resources properties which are static.

Title="{x:Static p:Resources.WpfLocalizationDemo}"

* 1. The Window defines the SizeToContent property with the “WidthAndHeight” value. This allows the Window to automatically define its Width and Height in respect to the content.   
       
     By example the words in the German language are longer than in English. With the SizeToContent setting you allow WPF to create a larger Window when it has to show its resources in the German language so that none of the text gets cut.
  2. The Grid in the MainWindow defines its ColumnDefinition.Width and RowDefinition.Height properties with the “Auto” value. This is done because of the same reason as the previous point defines the SizeToContent property.  
       
     **Remark:** Try to avoid defining the Width and Height properties with concrete values. Let WPF calculate those values itself.