



Internationalizing WPF And Silverlight Applications



Guy Smith-Ferrier
guy@guysmithferrier.com

Blog: <http://www.guysmithferrier.com>

About...

- Author of .NET Internationalization
 - Visit <http://www.dotneti18n.com> to download the complete source code
- The .NET Developer Network
 - <http://www.dotnetdevnet.com>
 - Free user group for .NET developers, architects and IT Pros based in Bristol



Agenda

- Localizing WPF using .resx files
- Localizing WPF using LocBaml
- Localizing WPF using LocBaml and a ResourceDictionary
- Localizing Silverlight using .resx files

Demo

Localizing WPF Using .resx Files

Localizing WPF Using .resx Files

- Using Visual Studio 2008 create a new WPF application
- Add a button

```
<Button Content="Hello world"/>
```

- Add a Resources File (In Solution Explorer, right click WpfApplication1, select Add | New Item, select Resources File) and call it Window1Resources.resx
 - Add a new resource entry called "Button_1" with a value of "Hello World"
 - Set the Access Modifier to Public (using the combo box)

Localizing WPF Using .resx Files (continued)

- In Window1.xaml add a "Resources" namespace and change the button's Content to use a static resource

```
<window x:Class="wpfApplication1.Window1"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:Resources="clr-namespace:wpfApplication1"
  Title="Window1" Height="300" Width="300">
  <Grid>
    <Button Content="{x:Static Resources:Window1Resources.Button_1}"/>
  </Grid>
</window>
```

Localizing WPF Using .resx Files (continued)

- In Visual Studio copy Window1Resources.resx to Window1Resources.fr-FR.resx
 - Change the "Button_1" resource value to "Bonjour Le Monde"
- In App.xaml.cs add a constructor to set the CurrentUICulture

```
public App()  
{  
    Thread.CurrentThread.CurrentUICulture =  
        Thread.CurrentThread.CurrentCulture;  
}
```

Localizing WPF Using .resx Files

Pros And Cons

- Pros:-
 - Good tool support for .resx files
 - Resource granularity is at the resource entry level
- Cons:-
 - Binding to properties requires XAML to be changed
 - Binding can only be applied to dependency properties (in XAML)

LocBaml

- LocBaml is a sample tool for localizing BAML
 - “The LocBaml tool is not a production-ready application. It is presented as a sample that uses some of the localization APIs and illustrates how you might write a localization tool.”
- Download from <http://msdn.microsoft.com/en-us/library/ms771568.aspx>
 - Unzip the download to a folder
 - Open Visual Studio and build the project
- LocBaml is a wrapper around the BamlLocalizer class

Demo

Localizing WPF Using LocBaml

Localizing WPF Using LocBaml

- Using Visual Studio 2008 create a new WPF application
- Add a button

```
<Button Content="Hello world"/>
```

- Open WpfApplication1.csproj using NotePad and add a UICulture element to the first PropertyGroup

```
<UICulture>en-GB</UICulture>
```

Localizing WPF Using LocBaml (continued)

- Open AssemblyInfo.cs, uncomment the NeutralResourcesLanguage attribute and set the culture

```
[assembly: NeutralResourcesLanguage("en-GB",  
    UltimateResourceFallbackLocation.Satellite)]
```

- In App.xaml.cs add a constructor to set the CurrentUICulture

```
public App()  
{  
    Thread.CurrentThread.CurrentUICulture =  
        Thread.CurrentThread.CurrentCulture;  
}
```

Localizing WPF Using LocBaml (continued)

- Open a command prompt in the WpfApplication1 folder and execute this line:-

```
msbuild /t:updateuid wpfApplication1.csproj
```

- In Visual Studio answer "Yes To All" to reload the changed files
- Compile the application

Localizing WPF Using LocBaml (continued)

- In the command prompt change to the bin\Debug folder
- Copy LocBaml.exe from the LocBaml folder to the current folder and execute this line:-

```
locbaml /parse en-GB/WpfApplication1.resources.dll /out:fr-FR.csv
```

- In Windows Explorer double click on the new fr-FR.csv file to load Excel

Localizing WPF Using LocBam1 (continued)

- Change "Hello World" to "Bonjour Le Monde"
- File | Save As, set "Save as type" to "CSV (Comma delimited)"
- Make a directory beneath bin\Debug called "fr-FR"
- In the command prompt execute this line:-

```
LocBam1.exe /generate en-GB/wpflLocBam11.resources.dll  
/trans:fr-FR.csv /out:fr-FR /culture:fr-FR
```

LocBaml CSV Columns

Column	Description
BAML Name	The name of the BAML resource stream
Resource Key	The resource entry identifier
Localization Category	The category
Readable	Should the resource be visible to the localizer
Modifiable	Should the resource be modifiable by the localizer
Localization Comments	Comments from the developer to the localizer
Value	The value of the resource entry

Localization Comments

- Localization comments are added to the XAML

```
<Button x:Uid="Button_1"  
        Content="Hello world"  
        ToolTip="Click to see the world"  
        Localization.Comments="$Content (The text shown in the button)  
                               $ToolTip (The text shown over the button)"/>
```

Localization Comments (continued)

- Localization comments can be extracted/included by adding a `LocalizationDirectivesToLocFile` to the `.csproj`

```
<LocalizationDirectivesToLocFile>All</LocalizationDirectivesToLocFile>
```

- Comments can be extracted to a separate file

```
<LocalizableAssembly>  
<LocalizableFile Name="window1">  
  <LocalizationDirectives Uid="Button_1" Comments="$Content (The text  
shown in the button) $ToolTip (The text shown over the button)" />  
</LocalizableFile>  
</LocalizableAssembly>
```

LocBaml Equivalents For ASP.NET Developers

	ASP.NET	LocBaml
Generate tags in source code	Tools Generate Local Resources	msbuild /t:updateuid locbaml /parse
Tag Identifier	meta:resourcekey	x:Uid
Source Of Localized Resources	.resx file	.csv file
Build Satellite Assemblies	Build (in Visual Studio)	locbaml /generate
'Prevent' Localization	meta:localize	Localization.Attributes

Localizing WPF Using LocBaml

Pros And Cons

- Pros:-

- Localization can be performed without the original source
- Localization does not need to be part of the development process
- Binding to properties requires no change to the XAML
- Bind to any property

- Cons:-

- Poor tool support
- Resource granularity is at the BAML resource level
 - No effective resource fallback
- LocBaml overwrites existing CSV files
- LocBaml does not sign generated assemblies
- LocBaml fails to generate correct CSV files when resources contain commas

Integrating LocBaml With Visual Studio

Integrating "msbuild /t:updateuid"

- In Visual Studio select Tools | External Tools..., click Add
 - Set Title to "msbuild /t:updateuid"
 - Set Command to "C:\Windows\Microsoft.NET\Framework\v3.5\MSBuild.exe"
 - Set Arguments to "/t:updateuid \$(ProjectFileName)"
 - Set Initial directory to "\$(ProjectDir)"
 - Check "Use Output window"
 - Click OK

Integrating LocBaml With Visual Studio

Integrating "locbaml /parse"

- In Visual Studio select Tools | External Tools..., click Add
 - Set Title to "locbaml /parse"
 - Set Command to "c:\CS35Tests\WpfApplication1\bin\debug\locbaml.exe"
 - Set Arguments to "/parse en-GB/\$(TargetName).resources.dll /out:fr-FR.csv"
 - Set Initial directory to "\$(BinDir)"
 - Check "Use Output window"
 - Click OK

Integrating LocBaml With Visual Studio

Integrating "locbaml /generate"

- In Visual Studio select Tools | External Tools..., click Add
 - Set Title to "locbaml /generate"
 - Set Command to "c:\CS35Tests\WpfApplication1\bin\debug\locbaml.exe"
 - Set Arguments to "/generate en-GB/\$(TargetName).resources.dll /trans:fr-FR.csv /culture:fr-FR /out:fr-FR "
 - Set Initial directory to "\$(BinDir)"
 - Check "Use Output window"
 - Click OK

Demo

Localizing WPF Using LocBaml And Resource Dictionaries

Localizing Using LocBaml And Dictionaries

- Add a resource dictionary called Window1Resources

```
<ResourceDictionary
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:system="clr-namespace:System;assembly=mcorlib">
  <system:String x:Key="Button_1Message">Hello</system:String>
</ResourceDictionary>
```

- Add the dictionary to Window1.xaml:-

```
<Window.Resources>
  <ResourceDictionary>
    <ResourceDictionary.MergedDictionaries>
      <ResourceDictionary Source="window1Resources.xaml"/>
    </ResourceDictionary.MergedDictionaries>
  </ResourceDictionary>
</Window.Resources>
```

Localizing Using LocBaml And Dictionaries (continued)

- Add a click event to the button to show the resource

```
MessageBox.Show(this.FindResource("Button_1Message").ToString());
```

- Follow the previous steps to add uids, parse out the resources, modify the CSV and rebuild the assembly

Right To Left Support

- Right to left behaviour is supported through the `FrameworkElement.FlowDirection` property
 - Values can be `LeftToRight` or `RightToLeft`
 - Elements inherit their `FlowDirection` from their container
 - Images never inherit their `FlowDirection`
- To set the `FlowDirection` programmatically:-

```
public Window1()
{
    InitializeComponent();
    if (Thread.CurrentThread.CurrentUICulture.TextInfo.IsRightToLeft)
    {
        Window.SetFlowDirection(this, FlowDirection.RightToLeft);
    }
}
```

Demo

Localizing Silverlight Using .resx Files

Localizing Silverlight Using .resx Files

- Using Visual Studio 2008 create a new Silverlight app
- Add a button

```
<Button Content="Hello world"/>
```

- Add a Resources File (In Solution Explorer, right click WpfApplication1, select Add | New Item, select Resources File) and call it PageResources.resx
 - Add a new resource entry called "Button_1" with a value of "Hello World"
 - Set the Access Modifier to Public (using the combo box)
 - Change the constructor to public in PageResources.Designer.cs

Localizing Silverlight Using .resx Files (continued)

- In Page.xaml add a "Resources" namespace, a static resource and change the button to use a static resource

```
<UserControl x:Class="SilverlightApplication1.Page"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:Resources="clr-namespace:SilverlightApplication1"
  width="400" height="300">
  <UserControl.Resources>
    <Resources:PageResources x:Name="PageResources"/>
  </UserControl.Resources>
  <Grid x:Name="LayoutRoot" Background="white">
    <Button Content="{Binding Button_1, Source={StaticResource PageResources}}"/>
  </Grid>
</UserControl>
```

Localizing Silverlight Using .resx Files (continued)

- In Visual Studio copy PageResources.resx to PageResources.fr-FR.resx
 - Change the "Button_1" resource value to "Bonjour Le Monde"
- Open SilverlightApplication1.csproj using NotePad, locate the SupportedCultures element and set it to fr-FR

```
<SupportedCultures>fr-FR</SupportedCultures>
```

Localizing Silverlight Using .resx Files (continued)

- Open SilverlightApplication1TestPage.aspx and add InitParameters to the Silverlight control

```
<asp:Silverlight ID="Xaml1" runat="server"  
Source="~/ClientBin/SilverlightApplication1.xap"  
MinimumVersion="2.0.30523" width="100%" Height="100%"  
InitParameters="UICulture=fr-FR" />
```

Localizing Silverlight Using .resx Files (continued)

- In App.xaml.cs change the Application_Startup method

```
private void Application_Startup(  
    object sender, StartupEventArgs e)  
{  
    string cultureName = e.InitParams["UICulture"].ToString();  
    if (!String.IsNullOrEmpty(cultureName))  
        Thread.CurrentThread.CurrentUICulture =  
            new CultureInfo(cultureName);  
  
    this.RootVisual = new Page();  
}
```

- Run the application

WPF I18N Features Not Available In Silverlight

- Static markup extension is not supported
- FrameworkElement.FlowDirection is not supported
- Custom cultures and Windows-only cultures **are** supported
 - The cultures must exist on the client (not the server)
 - CultureAndRegionInfoBuilder not supported
- Numerous properties and methods missing from System.Globalization classes (CultureInfo, RegionInfo, TextInfo etc.)

Summary

- There is no 'one true path' for localizing WPF/Silverlight
- The choices are:-
 - .resx files
 - XAML (LocBaml)
 - custom solution
- Tool support for post-build localization (i.e. LocBaml) will improve
- There is no resource provider model in WPF/Silverlight
- Globalization support in WPF is excellent