How to: Localize an application

Article • 02/24/2022 • 6 minutes to read • 2 contributors

In this article

This tutorial explains how to create a localized application by using the LocBaml tool.

Note

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.

Overview

This article gives you a step-by-step approach to localizing an application. First, you prepare your application so that the text that will be translated can be extracted. After the text is translated, you merge the translated text into a new copy of the original application.

Create a sample application

In this step, you prepare your app for localization. In the Windows Presentation Foundation (WPF) samples, a HelloApp sample is supplied that will be used for the code examples in this discussion. If you would like to use this sample, download the Extensible Application Markup Language (XAML) files from the LocBaml Tool Sample .

1. Develop your application to the point where you want to start localization.

2. Specify the development language in the project file so that MSBuild generates a main assembly and a satellite assembly (a file with the .resources.dll extension) to contain the neutral language resources. The project file in the HelloApp sample is HelloApp.csproj. In that file, you will find the development language identified as follows:

   ```
   <UICulture>en-US</UICulture>
   ```

3. Add Uids to your XAML files. Uids are used to keep track of changes to files and to identify items that must be translated. To add Uids to your files, run `updateuid` on your project file:

   ```
   msbuild -t:updateuid helloapp.csproj
   ```

   To verify that you have no missing or duplicate Uids, run `checkuid`:

   ```
   msbuild -t:checkuid helloapp.csproj
   ```

   After running `updateuid`, your files should contain Uids. For example, in the *Pane1.xaml* file of HelloApp, you should find the following:

   XAML Copy

   ```xml
   <StackPanel x:Uid="StackPanel_1">
     <TextBlock x:Uid="TextBlock_1">Hello World</TextBlock>
     <TextBlock x:Uid="TextBlock_2">Goodbye World</TextBlock>
   </StackPanel>
   ```

Create the neutral-language resources satellite assembly

After the application is configured to generate a neutral-language resources satellite assembly, you build the application. This generates the main application assembly as well as the neutral-language resources satellite assembly that's required by LocBaml for localization.

To build the application:

1. Compile HelloApp to create a dynamic-link library (DLL):

   ```
   msbuild helloapp.csproj
   ```

2. The newly created main application assembly, HelloApp.exe, is created in the following folder: *C:\HelloApp\Bin\Debug*

3. The newly created neutral-language resources satellite assembly, HelloApp.resources.dll, is created in the following folder: *C:\HelloApp\Bin\Debug \en-US*

Build the LocBaml tool

1. All the files necessary to build LocBaml are located in the WPF samples. Download the C# files from the LocBaml Tool Sample .

2. From the command line, run the project file (locbaml.csproj) to build the tool:

   ```
   msbuild locbaml.csproj
   ```

3. Go to the *Bin\Release* directory to find the newly created executable file (locbaml.exe). Example: *C:\LocBaml\Bin\Release\locbaml.exe*

4. The options that you can specify when you run LocBaml are as follows.

Option	**Description**
`parse` or `-p`	Parses Baml, resources, or DLL files to generate a .csv or .txt file.
`generate` or `-g`	Generates a localized binary file by using a translated file.
`out` or `-o` {*filedirectory*]	Output file name.
`culture` or `-cul` {*culture*]	Locale of output assemblies.
`translation` or `-trans` {*translation.csv*]	Translated or localized file.
`asmpath` or `-asmpath` {*filedirectory*]	If your XAML code contains custom controls, you must supply the `asmpath` to the custom control assembly.

Option	Description
`nologo`	Displays no logo or copyright information.
`verbose`	Displays verbose mode information.

ⓘ **Note**

If you need a list of the options when you are running the tool, enter `LocBaml.exe` and then press **Enter**.

Use LocBaml to parse a file

Now that you have created the LocBaml tool, you are ready to use it to parse HelloApp.resources.dll to extract the text content that will be localized.

1. Copy LocBaml.exe to your application's bin\debug folder, where the main application assembly was created.

2. To parse the satellite assembly file and store the output as a .csv file, use the following command:

```
LocBaml.exe /parse HelloApp.resources.dll /out:Hello.csv
```

ⓘ **Note**

If the input file, HelloApp.resources.dll, is not in the same directory as LocBaml.exe move one of the files so that both files are in the same directory.

3. When you run LocBaml to parse files, the output consists of seven fields delimited by commas (.csv files) or tabs (.txt files). The following shows the parsed .csv file for the HelloApp.resources.dll:

Parsed .csv file
HelloApp.g.en-US.resources:window1.baml,Stack1:System.Windows.Controls.StackPanel.$Content,Ignore,FALSE,FALSE,,#Text1;#Text2;

Parsed .csv file
HelloApp.g.en-US.resources:window1.baml,Text1:System.Windows.Controls.TextBlock.$Content,None,TRUE,TRUE,,Hello World
HelloApp.g.en-US.resources:window1.baml,Text2:System.Windows.Controls.TextBlock.$Content,None,TRUE,TRUE,,Goodbye World

The seven fields are:

- **BAML Name**. The name of the BAML resource with respect to the source language satellite assembly.
- **Resource Key**. The localized resource identifier.
- **Category**. The value type. See Localization Attributes and Comments.
- **Readability**. Whether the value can be read by a localizer. See Localization Attributes and Comments.
- **Modifiability**. Whether the value can be modified by a localizer. See Localization Attributes and Comments.
- **Comments**. Additional description of the value to help determine how a value is localized. See Localization Attributes and Comments.
- **Value**. The text value to translate to the desired culture.

The following table shows how these fields map to the delimited values of the .csv file:

BAML name	Resource key	Catego
HelloApp.g.en-US.resources:window1.baml	Stack1:System.Windows.Controls.StackPanel.$Content	Ignore
HelloApp.g.en-US.resources:window1.baml	Text1:System.Windows.Controls.TextBlock.$Content	None
HelloApp.g.en-US.resources:window1.baml	Text2:System.Windows.Controls.TextBlock.$Content	None

Notice that all the values for the **Comments** field contain no values; if a field doesn't have a value, it is empty. Also notice that the item in the first row is neither readable nor modifiable, and has "Ignore" as its **Category** value, all of which indicates that the value is not localizable.

4. To facilitate discovery of localizable items in parsed files, particularly in large files, you can sort or filter the items by **Category**, **Readability**, and **Modifiability**. For example, you can filter out unreadable and unmodifiable values.

Translate the localizable content

Use any tool that you have available to translate the extracted content. A good way to do this is to write the resources to a .csv file and view them in Microsoft Excel, making translation changes to the last column (value).

Use LocBaml to generate a new .resources.dll file

The content that was identified by parsing HelloApp.resources.dll with LocBaml has been translated and must be merged back into the original application. Use the `generate` or `-g` option to generate a new .resources.dll file.

1. Use the following syntax to generate a new HelloApp.resources.dll file. Mark the culture as en-US (/cul:en-US).

   ```
   LocBaml.exe /generate HelloApp.resources.dll /trans:Hello.csv /out:c:\
   /cul:en-US
   ```

   Note

   If the input file, Hello.csv, is not in the same directory as the executable, LocBaml.exe, move one of the files so that both files are in the same directory.

2. Replace the old *HelloApp.resources.dll* file in the *C:\HelloApp\Bin\Debug\en-US\HelloApp.resources.dll* directory with your newly created *HelloApp.resources.dll* file.

3. "Hello World" and "Goodbye World" should now be translated in your application.

4. To translate to a different culture, use the culture of the language that you are translating to. The following example shows how to translate to French-Canadian:

```
LocBaml.exe /generate HelloApp.resources.dll /trans:Hellofr-CA.csv /out:c:\
/cul:fr-CA
```

5. In the same assembly as the main application assembly, create a new culture-specific folder to house the new satellite assembly. For French-Canadian, the folder would be fr-CA.

6. Copy the generated satellite assembly to the new folder.

7. To test the new satellite assembly, you need to change the culture under which your application will run. You can do this in one of two ways:

   - Change your operating system's regional settings.

   - In your application, add the following code to App.xaml.cs:

XAML Copy

```xml
<Application
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presenta-
tion"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.App"
    x:Uid="Application_1"
    StartupUri="Window1.xaml">
</Application>
```

C# Copy

```csharp
using System.Windows;
using System.Globalization;
using System.Threading;

namespace SDKSample
{
    public partial class App : Application
    {
        public App()
        {
            // Change culture under which this application runs
```

```
                CultureInfo ci = new CultureInfo("fr-CA");
                Thread.CurrentThread.CurrentCulture = ci;
                Thread.CurrentThread.CurrentUICulture = ci;
            }
        }
    }
```

Tips for Using LocBaml

- All dependent assemblies that define custom controls must be copied into the local directory of LocBaml or installed into the GAC. This is necessary because the localization API must have access to the dependent assemblies when it reads the binary XAML (BAML).
- If the main assembly is signed, the generated resource DLL must also be signed in order for it to be loaded.
- The version of the localized resource DLL needs to be synchronized with the main assembly.

See also

- Globalization for WPF
- Use Automatic Layout Overview

Recommended content

Globalization - WPF .NET Framework

Globalization and localization overview - WPF .NET Framework

Learn about localization and globalization for Windows Presentation Foundation, including automatic layout, satellite assemblies, and localized attributes.

Use design-time sample data with the XAML Designer in Visual Studio - Visual Studio (Windows)

Learn how to use design-time sample data in XAML.

Pack URIs - WPF .NET Framework

Learn about the many ways to use uniform resource identifiers (URIs) to identify and load files in Windows Presentation Foundation (WPF).

ContentControl.ContentTemplateSelector Property (System.Windows.Controls)

Gets or sets a template selector that enables an application writer to provide custom template-selection logic.

Show more