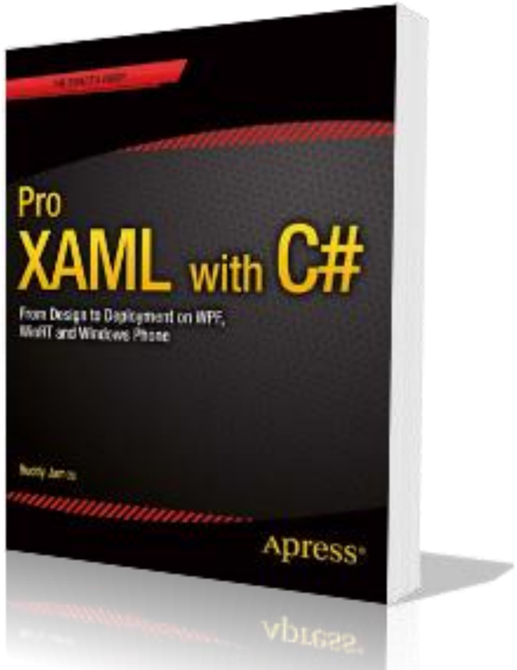
Cross Platform Localization Strategies

Lori Lalonde
Microsoft MVP – Windows Platform Dev
Xamarin Certified Developer

ObjectSharp

Me, Myself and I

LORI LALONDE
Consultant



CTTDNUG

Education
Entertainment
Exposure

MVP
Microsoft®
Most Valuable Professional

Xamarin Certified
Mobile Developer

Twitter: @loriblalonde
Email: loriblalonde@gmail.com
Blog: geekswithblogs.net/lorilalonde
Web: solola.ca
LinkedIn: http://ca.linkedin.com/in/lorilalonde

ObjectSharp

Why?

Appeal to a Broader Market

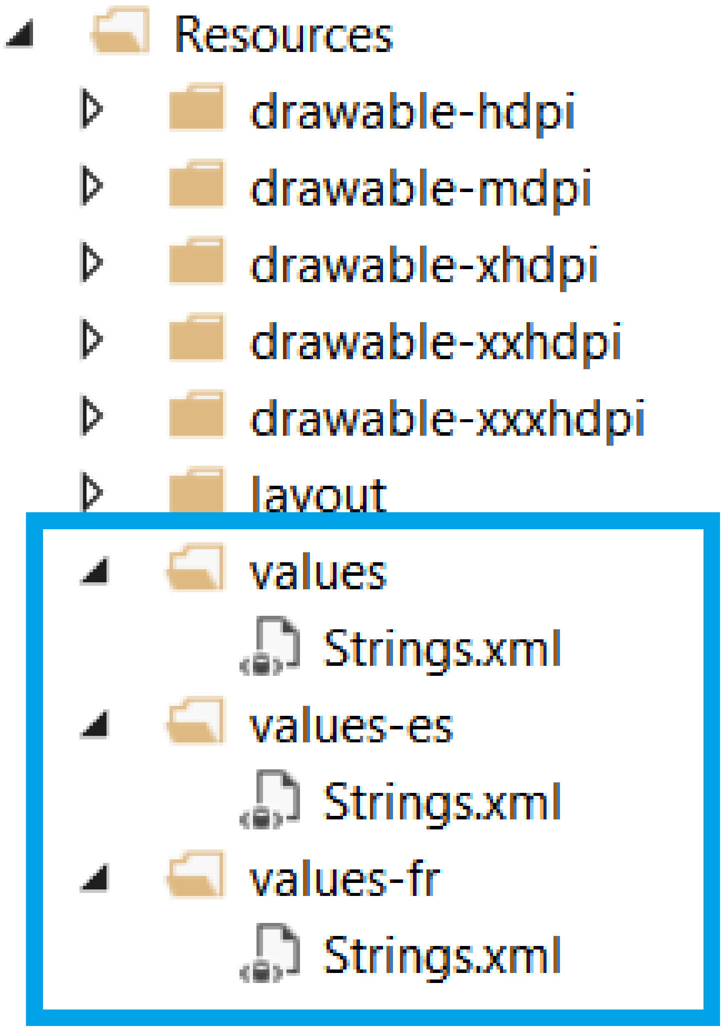
*photo attribution: created by niroworld at fotolia (https://www.fotolia.com/id/66976406)
Royalty-free license was purchased for the use of this image

General Considerations

- ✓ Design your app to support multiple languages (even if you’re supporting one language)
- ✓ Never hardcode strings
- ✓ Store strings in language resource files
- ✓ Avoid string concatenation
- ✓ Ensure your UI is flexible to adapt to variable string length

Xamarin Native Localization

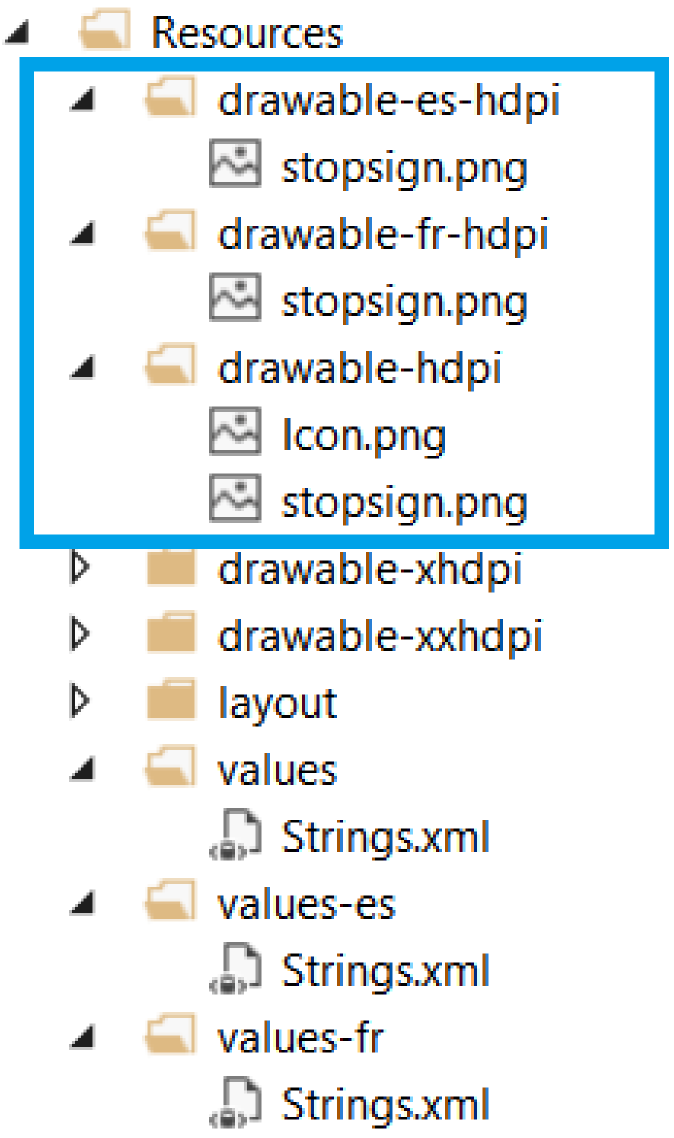
Xamarin Android String Resources



Resources
drawable-hdpi
drawable-mdpi
drawable-xhdpi
drawable-xxhdpi
drawable-xxxhdpi
layout
values
Strings.xml
values-es
Strings.xml
values-fr
Strings.xml

- Strings stored in XML file(s) in values directory
- Place XML files in values directory for each supported language code

Xamarin Android Image Resources



```
◢ Resources
    ◢ drawable-es-hdpi
        stopsign.png
    ◢ drawable-fr-hdpi
        stopsign.png
    ◢ drawable-hdpi
        Icon.png
        stopsign.png
    ▷ drawable-xhdpi
    ▷ drawable-xxhdpi
    ▷ layout
    ◢ values
        Strings.xml
    ◢ values-es
        Strings.xml
    ◢ values-fr
        Strings.xml
```

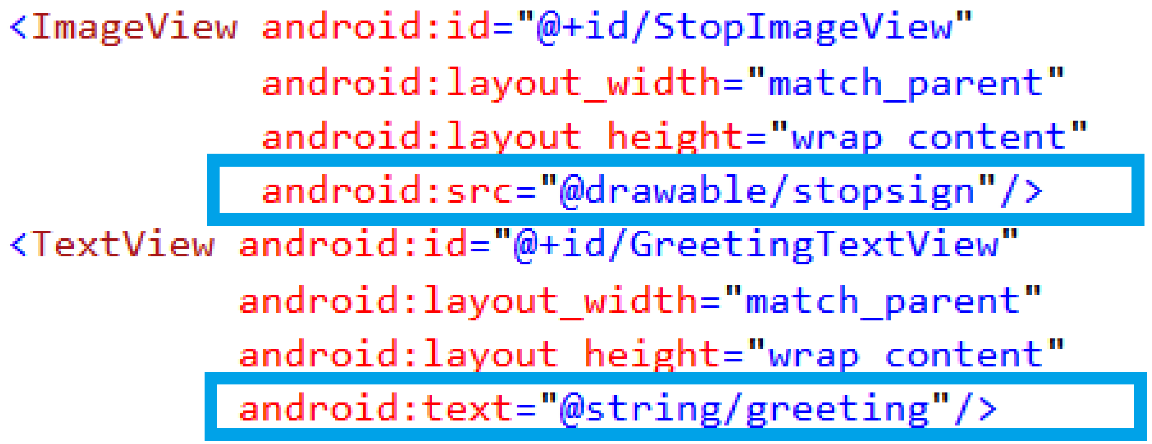
- Images with text should be localized
- Set Image BuildAction as AndroidResource
- Place localized images in drawable directory for each supported language code and device resolution

ObjectSharp

Xamarin.Android: Accessing Resources

- In layout file:
  Strings - @string/[id]

  Images - @drawable/[id]



```
<ImageView android:id="@+id/StopImageView"
           android:layout_width="match_parent"
           android:layout height="wrap content"
           android:src="@drawable/stopsign"/>
<TextView android:id="@+id/GreetingTextView"
          android:layout_width="match_parent"
          android:layout height="wrap content"
          android:text="@string/greeting"/>
```

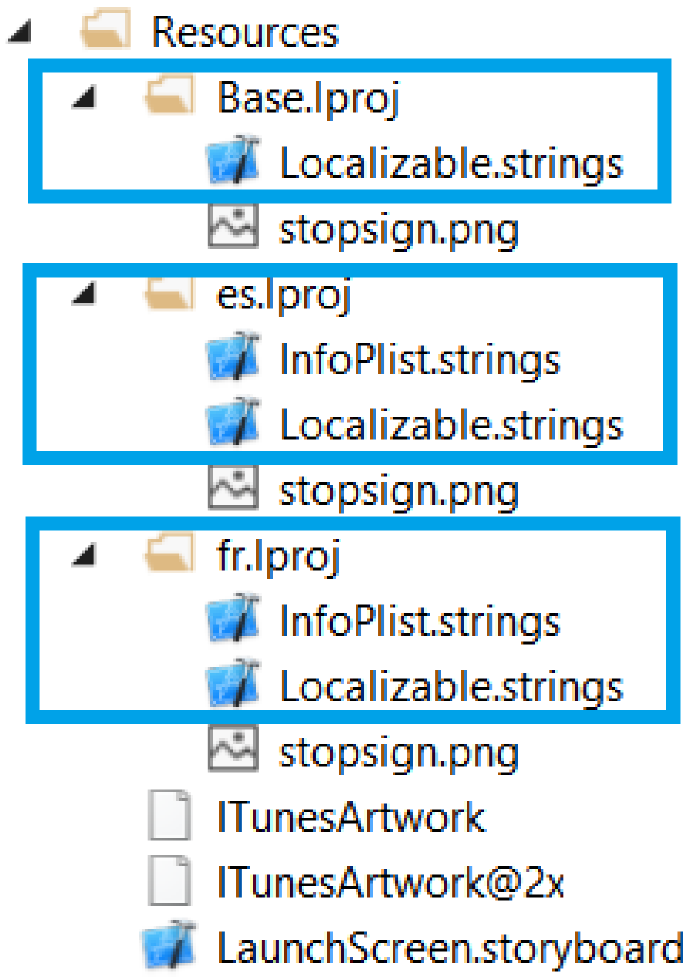
- In code:

```
var greeting = Resources.GetString(Resource.String.greeting);

var myImage = Resources.GetDrawable(Resource.Drawable.stopsign);
```

ObjectSharp

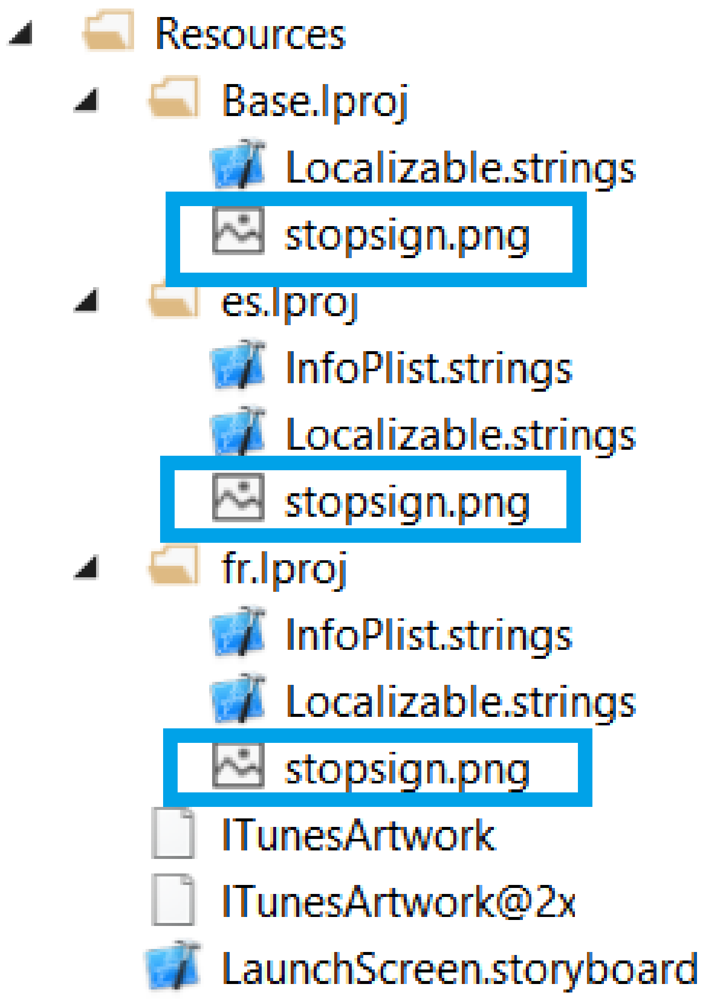
Xamarin iOS String Resources



```
Resources
    Base.lproj
        Localizable.strings
        stopsign.png
    es.lproj
        InfoPlist.strings
        Localizable.strings
        stopsign.png
    fr.lproj
        InfoPlist.strings
        Localizable.strings
        stopsign.png
    ITunesArtwork
    ITunesArtwork@2x
    LaunchScreen.storyboard
```

- Strings stored in .strings files in .lproj directories
- Base.lproj – default language
- <languagecode>.lproj directory – supported language
- .strings file format: “key” = “value”;

Xamarin iOS Image Resources



```
Resources
  Base.lproj
    Localizable.strings
    stopsign.png
  es.lproj
    InfoPlist.strings
    Localizable.strings
    stopsign.png
  fr.lproj
    InfoPlist.strings
    Localizable.strings
    stopsign.png
  ITunesArtwork
  ITunesArtwork@2x
  LaunchScreen.storyboard
```

- Place localized images in respective .lproj directories
- Set Image BuildAction as BundleResource

Xamarin.iOS: Accessing Resources

```
//app display name
var title = NSBundle.MainBundle
            .ObjectForInfoDictionary("CFBundleDisplayName");
AppTitleLabel.Text = title.ToString();

//how to get the localized string in code
HelloLabel.Text = NSBundle.MainBundle.LocalizedString("hello", "comment");

//how to get localized image in code
StopImage.Image = UIImage.FromBundle("stopsign");
```

Escape Characters

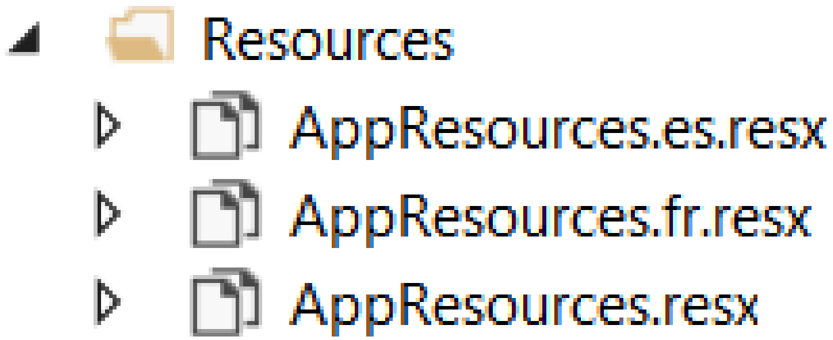
Android and iOS string resources must use the following escape characters:

- \" quote
- \\ backslash
- \n newline

Drawbacks

- Maintain separate set of resource files for each platform
- Platform-specific code needed to access localized strings and images

Windows Phone 8 String Resources



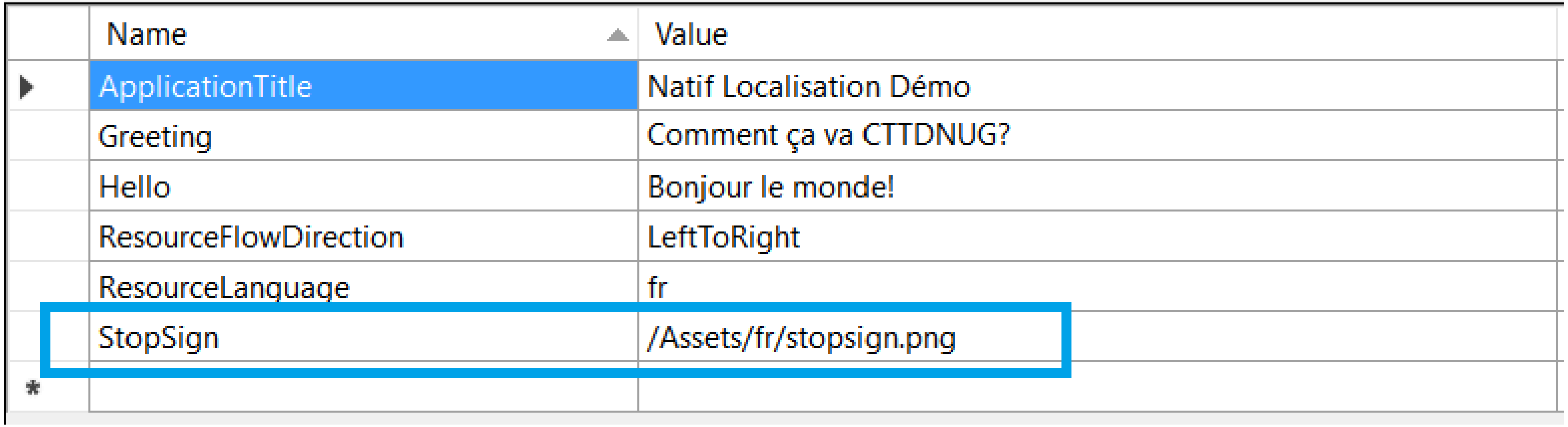
- Resources
  - AppResources.es.resx
  - AppResources.fr.resx
  - AppResources.resx

- Resource file (.resx) localization
- Default language does not include a language code in the name
- Language codes are included in resource file names

Windows Phone 8 Image Resources

- Add image files to project with your defined naming or directory structure convention

- Include Image Uri path as a string resource in Resources files



Name	Value
ApplicationTitle	Natif Localisation Démo
Greeting	Comment ça va CTTDNUG?
Hello	Bonjour le monde!
ResourceFlowDirection	LeftToRight
ResourceLanguage	fr
StopSign	/Assets/fr/stopsign.png
*	

Demo

Leveraging .NET Localization in Xamarin Native apps

Approach

- ✓ Create a Portable Class Library that will contain the resource (.resx) files
- ✓ Include a Translation Helper class
- ✓ Leverage ResourceManager to load the strings

Drawbacks

- ✓ Windows 8.1 and Windows Phone 8.1 (XAML) do not support resx files
- ✓ Application name will not be localized
- ✓ Workaround: include native strings file in each language for the application name only (iOS and Android)

Demo

Xamarin.Forms Localization

Using RESX Files

- Xamarin.Forms supports .NET Localization
- Enables use of one set of resource files (.resx) for string translations
- Data binding support in Xamarin.Forms
- Localized images should still be handled using native approach

Demo

Questions?

ObjectSharp

Additional Resources

Xamarin Localization Documentation

iOS: http://bitly.com/XamiOSLocalization
Android: http://bitly.com/XamAndLocalization
Xamarin.Forms: http://bitly.com/XamFormsLocalization

MSDN Windows Phone 8 Localization: http://bitly.com/WP8Localization

Thanks!

Twitter: @loriblalonde
Email: loriblalonde@gmail.com
Blog: geekswithblogs.net/lorilalonde
Web: solola.ca
LinkedIn: http://ca.linkedin.com/in/lorilalonde

ObjectSharp