# Macoratti.net Xamarin Android - Localizando a sua aplicação - II



Neste artigo vou mostrar com podemos **localizar** uma aplicação Android usando o Xamarin Android, o Visual Studio 2015 e a linguagem C#.



Na **primeira parte do artigo** eu mostrei uma abordagem bem simples de localização para pequenas aplicações Android, onde basta definir arquivos XML com o sufixo da cultura de localização na pasta **Resources**, definir no layout, na propriedade **android:text** o id no formato **@string/id** e usar os métodos **GetText() ou GetString()** para que a localização fosse processada com base no valor definido em em **Custom Locale** do dispositivo. Assim as etapas a serem cumpridas podem ser resumidas da seguinte forma:

- Definir as **pastas de recursos** para conter as strings , imagens e outros recursos que serão localizados;
- A utilização dos métodos **GetText e/ou GetString**, que são usados para recuperar as strings localizadas no código;
- A utilização de um identificador único no formato **@String/id** em arquivos **AXML**, para colocar automaticamente strings localizadas em layouts;

Neste artigo eu apresentar outra abordagem que obtém o mesmo resultado. Nessa abordagem vamos definir arquivos de recursos em uma pasta do projeto.

Para isso vamos criar uma pasta **AppResources** no projeto e definir arquivos de recursos(**resource files**) da plataforma .NET definindo o nome dos arquivos usando o respectivo sufixo para a cultura desejada.

**Nota**: *Um resource file ou arquivo de recurso é um arquivo não executável que é requisitado pela aplicação e que é distribuído junto com ela. Um arquivo de recurso pode conter : mapa de bits (bitmaps) , icones, cursores , textos , etc.*

Vamos então usar a classe **ResourceManager** que representa um gerenciador de recursos que fornece acesso aos recursos de cultura específico em tempo de execução para obter o respectivo arquivo de cultura com os textos traduzidos.

Depois vamos usar o método **CurrentUICulture** que obtém o objeto CultureInfo que representa a cultura da interface do usuário atual usada pelo gerenciador de recursos e o método **CurrentCulture** que define o objeto CultureInfo que representa a cultura usada pela thread atual.

Vamos então implementar essa outra abordagem.

**Recursos usados:**

- **Visual Studio Community 2015 ou Xamarin Studio**
- **Xamarin**
- **Emulador Android virtual ou físico (veja como emular usando o Vysor)**

**Nota: Baixe e use a versão Community 2015 do VS ela é grátis e é equivalente a versão Professional.**

## Criando os arquivos de recursos para localizar os textos dos controles

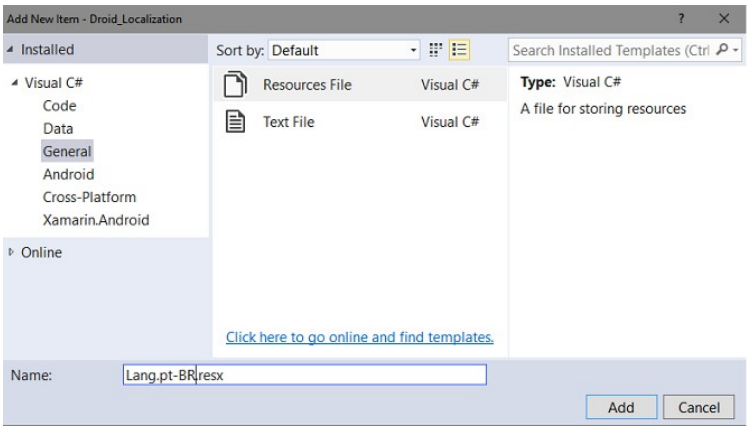
Abra o projeto **Droid_Localizacao,** criado no artigo anterior, no VS 2015 Community;

Clique com o botão direito sobre o projeto e a seguir clique em **Add -> New Folder;**

Informe o nome **AppResources.**

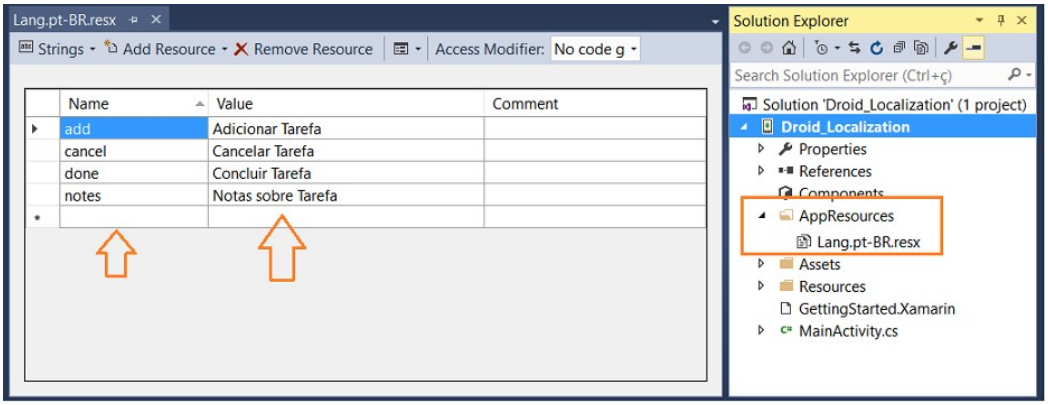
Clique com o botão direito sobre a pasta **AppResources** e a seguir clique em **Add -> New Item;**

A seguir selecione o template **Resources File** e informe o nome **Lang.pt-BR.resx:**



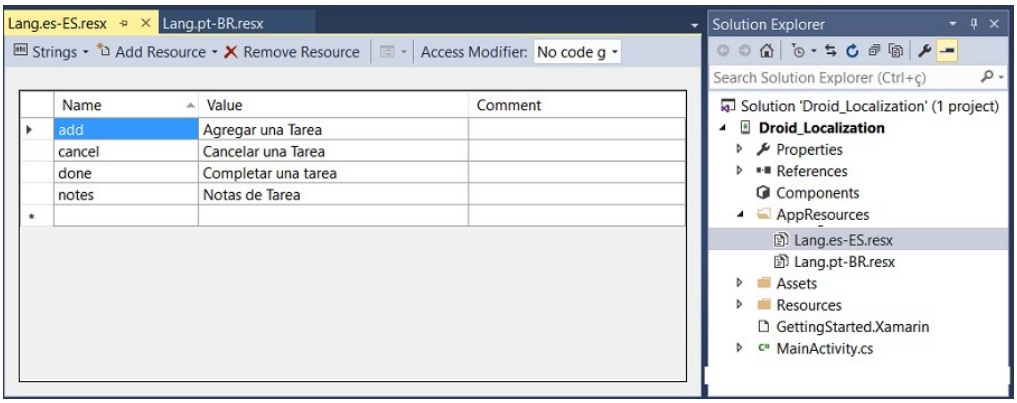

A nomenclatura usada refere-se ao nome do arquivo **Lang** mais o sufixo **pt-BR** que representa a cultura que este arquivo vai tratar.

A seguir abra o arquivo **Lang.pt-BR.resx** e defina nele o seguinte conteúdo :

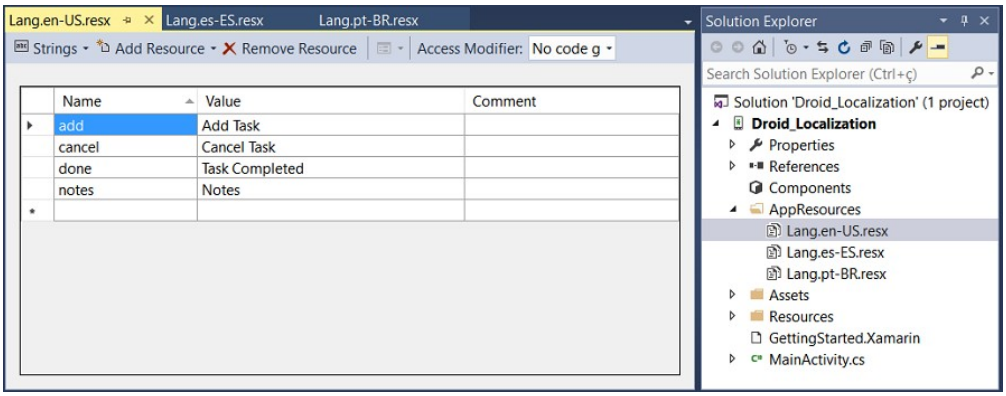



Em **Name** definimos o id referente ao recurso que iremos traduzir e em **Value** temos a respectiva tradução para o português.

Repita o procedimento acima e crie o arquivo de recurso **Lang.es-ES.resx** que representa a cultura para o idioma Espanhol com o seguinte conteúdo:




Para concluir repita o procedimento acima e crie o arquivo de recurso **Lang.en-US.resx** que representa a cultura para o idioma Inglês com o seguinte conteúdo:




Agora já temos os 3 arquivos de recursos definidos com seus respectivos id representado pela propriedade **Name** e os valores traduzidos para cada idioma.

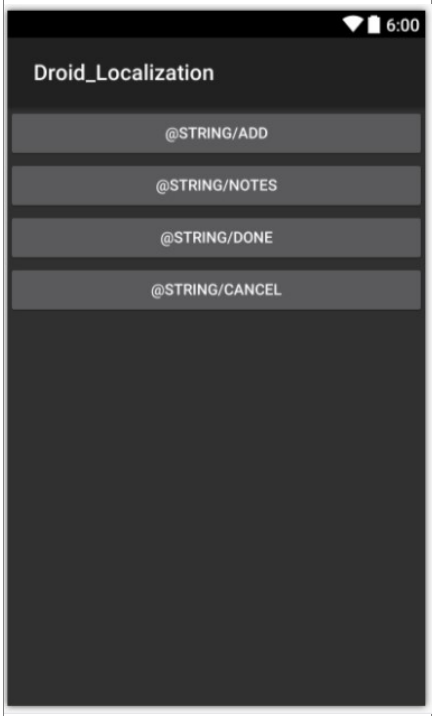
Agora vamos definir o arquivo de layout que será o mesmo que foi definido no artigo anterior.

## Definindo o arquivo de layout da aplicação

Abra o arquivo **Main.axml** na pasta **Resources/layout** no modo **Designer** e a seguir inclua 4 controles Button:

- 4 Button - id => **btnAdd, btnNotes, btnDone e btnCancel**

Abaixo vemos o leiaute no emulador do Xamarin e ao lado o respectivo código XML gerado :




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:minWidth="25px"
  android:minHeight="25px">
  <Button
    android:text="@string/add"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/btnAdd" />
  <Button
    android:text="@string/notes"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/btnNotes" />
  <Button
    android:text="@string/done"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/btnDone" />
  <Button
    android:text="@string/cancel"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/btnCancel" />
</LinearLayout>
```

No arquivo de layout definimos cada propriedade **android:text** dos **Buttons** usando o id correspondente à propriedade **name** do arquivo de recursos definido nas pastas de recursos.

O passo seguinte é definir no arquivo **MainActivity** qual o idioma e cultura deverá ser usado como padrão para exibir o texto correspondente no arquivo de recurso.

## Definindo o código da MainActivity

Abra o arquivo **MainActivity** e inclua o código abaixo :

```
using Android.App;
using Android.Widget;
using Android.OS;
using System.Threading;
using System.Globalization;
using System.Resources;
using System.Reflection;

namespace Droid_Localization
{
  [Activity(Label = "@string/app", MainLauncher = true, Icon = "@drawable/icon")]
  public class MainActivity : Activity
  {
    // carrega o arquivo de Recurso contento a informação da cultura especificada
    string codigoCultura = "es-ES";
    ResourceManager resxManager = new ResourceManager("Droid_Localization.AppResources.Lang", Assembly.GetExecutingAssembly());

    protected override void OnCreate(Bundle bundle)
    {
      base.OnCreate(bundle);

      SetContentView (Resource.Layout.Main);

      // define o idioma e a cultura
      Thread.CurrentThread.CurrentUICulture = new CultureInfo(codigoCultura);
      Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture(codigoCultura);

      var btnCancel = FindViewById<Button>(Resource.Id.btnCancel);
      var btnAdd = FindViewById<Button>(Resource.Id.btnAdd);
      var btnNotes = FindViewById<Button>(Resource.Id.btnNotes);
      var btnDone = FindViewById<Button>(Resource.Id.btnDone);

      if (btnCancel != null)
      {
        btnCancel.Text = resxManager.GetString("cancel");
```

```
            }
            if (btnAdd != null)
            {
                btnAdd.Text = resxManager.GetString("add");
            }
            if (btnNotes != null)
            {
                btnNotes.Text = resxManager.GetString("notes");
            }
            if (btnDone != null)
            {
                btnDone.Text = resxManager.GetString("done");
            }
        }
    }
}
```

Este código inicia definindo qual a cultura padrão deverá ser usada para exibir os textos dos controles **Buttons** traduzidos para o respectivo idioma.

```
string codigoCultura = "es-ES";
```

A seguir usamos a classe **ResourceManager** que fornece acesso aos recursos de cultura específico em tempo de execução para obter o respectivo arquivo de cultura com os textos traduzidos.

```
ResourceManager resxManager = new ResourceManager("Droid_Localization.AppResources.Lang",
Assembly.GetExecutingAssembly());
```

Aqui é importante destacar que a string **"Droid_Localization.AppResources.Lang"** representa a localização dos arquivos de recursos definidos no projeto.

Depois referenciamos o arquivo de Layout **Main.axml** e definimos o idioma e a cultura padrão que deverão ser usados na aplicação:

```
// define o idioma e a cultura
Thread.CurrentThread.CurrentUICulture = new CultureInfo(codigoCultura);
Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture(codigoCultura);
```

Depois criamos as instâncias de cada um dos Buttons.

Para concluir verificamos se a instância de cada Button não é **null** e atribuímos à sua propriedade **Text** o valor obtido do arquivo de recurso que criamos na pasta **AppResources** usando o gerenciador de recursos e o método **GetString**() e o id do recurso definido.

```
if (btnCancel != null)
{
    btnCancel.Text = resxManager.GetString("cancel");
}
```

Executando o projeto usando o emulador **Genymotion** iremos obter o seguinte resultado:


Essa abordagem define uma cultura inicial padrão que será usada para exibir a localização dos textos e também pode ser usada para realizar a localização em pequenas aplicações.

Pegue o projeto aqui : **Droid_Localization2.zip** (sem as referências)

***(Disse Jesus)*Este povo se aproxima de mim com a sua boca e me honra com os seus lábios, mas o seu coração está longe de mim. Mas, em vão me adoram, ensinando doutrinas que são preceitos dos homens.**

**Mateus 15:8,9**



**Gostou ? Compartilhe no Facebook Compartilhe no Twitter**

Referências:

- Seção VB .NET do Site Macoratti.net
- Super DVD .NET - A sua porta de entrada na plataforma .NET

- Super DVD Vídeo Aulas - Vídeo Aula sobre VB .NET, ASP .NET e C#
- Super DVD C# - Recursos de aprendizagens e vídeo aulas para C#
- Seção C# do site Macoratti.net
- Seção ASP .NET do site Macoratti .net
- Curso Básico VB .NET - Vídeo Aulas
- Curso C# Básico - Vídeo Aulas
- Curso Fundamentos da Programação Orientada a Objetos com VB .NET NEW
- **Macoratti .net | Facebook**
- **macoratti - YouTube**
- **Jose C Macoratti (@macorati) | Twitter**
- VB.NET 2005 - Controles - Macoratti.net
- Seção de Jogos do site Macoratti .net
- Xamarim - Desenvolvimento Multiplataforma com C# ... - Macoratti.net
- Xamarin - Apresentando Xamarin.Forms - Macoratti.net
- Xamarin.Forms - Olá Mundo - Criando sua primeira ... - Macoratti.net
- Xamarin.Forms - Olá Mundo - Anatomia da aplicação - Macoratti.net
- https://developer.xamarin.com/api/type/Android.App.AlertDialog/
- Xamarin Android - Tratando eventos de forma declarativa
- Seção Mobile/Xamarin do site Macoratti .net
- https://developer.android.com/reference/android/app/Activity.html
- https://developer.xamarin.com/api/type/Android.Widget.ProgressBar/

---

José Carlos Macoratti