



Локализация XAML

Последнее обновление: 30.06.2016



В прошлых темах мы рассмотрели локализацию в коде C#, но гораздо проще было бы локализовать элементы интерфейса сразу же в xaml. Для этого возьмем проект из прошлой темы.

Чтобы добавить в приложение возможность локализации элементов xaml, надо добавить класс расширения разметки. Итак, добавим в главный проект следующий класс:

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.Resources;
using System.Globalization;
using System.Reflection;

namespace LocalizeApp
{
    [ContentProperty("Text")]
    public class TranslateExtension : IMarkupExtension
    {
        readonly CultureInfo ci;
        const string ResourceId = "LocalizeApp.Resou

        public TranslateExtension()
        {
            ci = DependencyService.Get<ILocalize>().

        }

        public string Text { get; set; }

        public object ProvideValue(IServiceProvider
        {
            if (Text == null)
                return "";

            ResourceManager resmgr = new ResourceMan
                typeof(TranslateExtension).G

            var translation = resmgr.GetString(Text,

            if (translation == null)
            {
                translation = Text;
            }
        }
    }
}
```

```
        return translation;
    }
}
```

В строке `const string ResourceId = "LocalizeApp.Resource";` устанавливаются ресурсы для локализации. Так как у меня пространство имен проекта называется `LocalizeApp`, а файл ресурсов по умолчанию размещен в корне проекта и называется `Resource.resx`, то полное имя ресурса будет `LocalizeApp.Resource`.

Для извлечения ресурсов используется класс `ResourceManager` и его метод `GetString()`

Теперь применим данный класс в xaml:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/
                xmlns:x="http://schemas.micro
                x:Class="LocalizeApp.MainPage
                xmlns:local="clr-namespace:LocalizeApp;assembly=Lo
<StackLayout>

    <Label x:Name="myLabel" FontSize="Large" Text="{
/>
    <Entry x:Name="myEntry" Placeholder="{local:Tran
    <Button x:Name="myButton" FontSize="Large" Text
SaveButton}" />
</StackLayout>
</ContentPage>
```

Чтобы получить доступ к классу, надо подключить текущее пространство имен проекта: `xmlns:local="clr-namespace:LocalizeApp;assembly=LocalizeApp"`

Затем мы можем использовать привязку у тех свойствам элементов, которые надо локализовать: `Text="{local:Translate Header}"`. В данном случае `local` ссылается на текущее пространство имен, `"Translate"` ссылается на класс `TranslateExtension` - по умолчанию суффикс `Extension` отбрасывается. И затем указывается называется имя ресурса, к которому идет привязка.

В итоге мы можем убрать локализацию из класса приложения `App`, которая была определена в прошлых темах, теперь он будет просто создавать стартовую страницу:

```
public partial class App : Application
{
    public App()
    {
        MainPage = new MainPage();
    }
}
```



[Назад](#) [Содержание](#) [Вперед](#)






ОБРАЗОВАТЕЛЬНЫЙ
ЦЕНТР ПВТ


КУРС **iOS** РАЗРАБОТЧИКА

Высокая востребованность
на рынке IT!



Ищете
тензодатчики?

Тензодатчики от производителя.
Широкий модельный ряд!



Узнать стоимость

[Комментарии](#)[Сообщество](#)[Войти](#) ▼[❤ Рекомендовать](#)[🔗 Поделиться](#)[Лучшее в начале](#) ▼

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS [?](#)

Имя



Юрий Есин • год назад

Можно ещё проще и лаконичнее.

В классе ресурсов у нас уже автоматически реализован функционал для работы с ResourceManager и CultureInfo, в итоге описанный выше класс можно оптимизировать и упростить следующим образом:

```
[ContentProperty("Text")]
public class TranslateExtension :
    IMarkupExtension
```

```
public class TranslateExtension
```

```
{
    static TranslateExtension()
    {
        Resource.Culture =
        DependencyService.Get<ILocalize>
        ().GetCurrentCultureInfo();
    }
}
```

```
public string Text { get; set; }
```

```
public object ProvideValue(IServiceProvider
    serviceProvider)
```

```
{
    if (Text == null) return "";
}
```

```
string translation =
    Resource.ResourceManager.GetString(Text,
    Resource.Culture);
```

```
if (translation == null) translation = Text;
```

```
return translation;
```

```
}
```

```
}
```

^ | v • Ответить • Поделиться ›



Ярослав Орлов • год назад

Для тех кто в танке как я, уточню

resource.Culture = null;

`xmins:local="c1"`

`namespace:LocalizeApp;assembly=LocalizeApp`

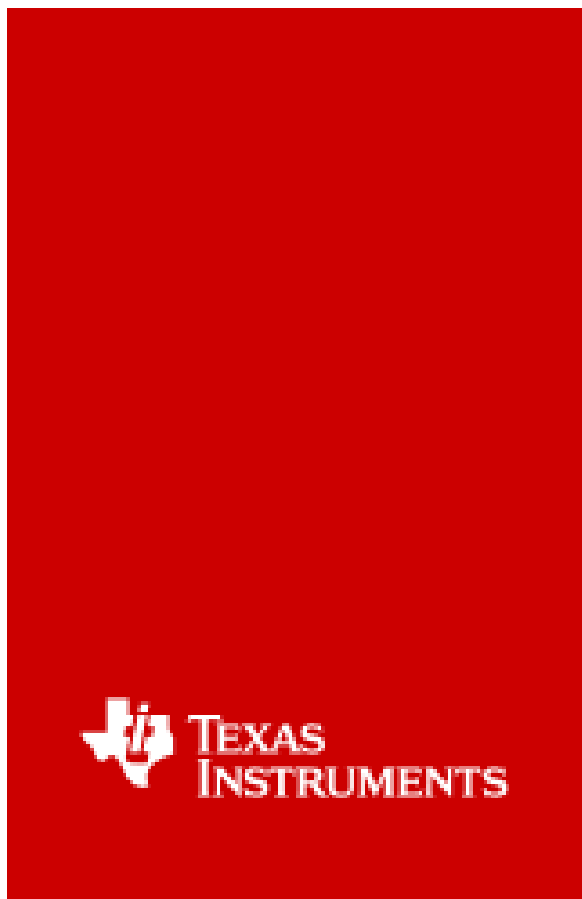
- это пространство имен в котором находится класс `TranslateExtension`, а это

`const string ResourceId =`

`"LocalizeApp.Resource";` - пространство имен в котором расположены ресурсы, на случай если вы располагаете все в разных директориях.

^ | v • Ответить • Поделиться ›

AdChoices



[Вконтакте](#) | [Телеграм](#) | [Twitter](#) | [Google+](#) |
[Youtube](#) | [Помощь сайту](#)

Контакты для связи: metanit22@mail.ru

Copyright © metanit.com, 2012-2018. Все права защищены.