## **METANIT.COM**



Сайт о программировании













### Определение языковой культуры

Последнее обновление: 30.06.2016













В прошлой теме был создан проект и ряд ресурсов, позволяющих локализовать приложение. Однако просто установки в коде нужного ресурса бывает недостаточно. Нужно явным образом указать требуемую языковую культуру для определенной платформы.

Поскольку каждая платформа требует определенного подхода, то нам надо использовать класс

DependencyService, который позволит сопоставить нужные зависимости.

Вначале добавим в главный проект интерфейс ILocalize:

```
using System.Globalization;
public interface ILocalize
{
    CultureInfo GetCurrentCultureInfo();
}
```

Его единственный метод будет определять языковую культуру мобильного устройства.

Tenepь добавим в проект для iOS следующий класс Localize:

```
var pref =
NSLocale.PreferredLanguages[0];
                netLanguage = pref.Replace("_", "-
"); // заменяет pt BR на pt-BR
            System.Globalization.CultureInfo ci =
null;
            try
                ci = new
System.Globalization.CultureInfo(netLanguage);
            catch
                ci = new
System.Globalization.CultureInfo(prefLanguage);
            return ci;
    }
}
```

Peaлизация этого класса использует массив NSLocale.PreferredLanguages для определения языковой культуры.

При получении языка на устройстве iOS нам следует учитывать следующий аспект. Пользователь теоретически может установить в качестве культуры, например, "en-ES". Например, если пользователь живет в Испании и в качестве региона выставил Испанию, однако захотел в качестве языка использовать английский. Культура "en-ES" не

соответствует ни однйо из культур в .NET, поэтому программа может вывалиться в ошибку. Чтобы ее избежать, здесь применяется блок try...catch, где в случае неудачи получить установленную культуры мы получаем предпочтительную культуру - "en-US".

Некоторые системные элементы управления iOS переводит автоматически (например, элемент Picker). Но чтобы указать системе на необходимость перевода, нам надо внести в файл **Info.plist**, который имеется в проекте, соответствующие определения языковых культур. Info.plist представляет обычный xml-файл, поэтому откроем его в каком-нибудь текстовом редакторе. Он выглядит примерно так:

В пределах элемента dict добавим определения языков. В моем случае язык по умолчанию английский (en) и две дополнительных языковых культуры: русскоязычная (ru) и немецкая (de). Поэтому я добавляю следующие строки:

```
<key>CFBundleLocalizations</key>
<array>
```

```
    <string>de</string>
    <string>ru</string>
    </array>
    <key>CFBundleDevelopmentRegion</key>
    <string>en</string>
```

### Настройка проекта для Android

В проект для Androida добавим следующий класс:

```
using Xamarin.Forms;
[assembly:
Dependency(typeof(LocalizeApp.Droid.Localize))]
namespace LocalizeApp.Droid
{
    public class Localize : ILocalize
        public System.Globalization.CultureInfo
GetCurrentCultureInfo()
            var androidLocale =
Java.Util.Locale.Default;
            var netLanguage =
androidLocale.ToString().Replace("_", "-");
            return new
System.Globalization.CultureInfo(netLanguage);
    }
}
```

Опять же это реализация интерфейса ILocalize, которая использует объект Java.Util.Locale.Default для определения культуры.

### Настройка проекта для UWP

Также добавим реалзицию интерфейса ILocalize в проект для UWP:

```
using System.Globalization;
using Xamarin.Forms;
[assembly:
Dependency(typeof(LocalizeApp.UWP.Localize))]
namespace LocalizeApp.UWP
{
    public class Localize : ILocalize
        {
        public System.Globalization.CultureInfo
GetCurrentCultureInfo()
        {
            return CultureInfo.CurrentUICulture;
        }
     }
}
```

Языковая культура приложения должна определяться как можно раньше. В частности, это можно сделать при открытии стартовой страницы приложения. Итак, добавим в конструктор класса **Арр** в главном проекте определения языковой культуры:

И поскольку фреймворк автоматически распознает язык на устройстве Windows Phone, нам нет нужды устанавливать его вручную, поэтому сначала в коде проверяем тип платформы Device.OS != TargetPlatform.WinPhone

## World of Tanks

Только реальная техника. Более 450 точны Регистрируйся и играй! worldoftanks.ru

#### <u>Назад Содержание</u> <u>Вперед</u>















Комментарии Сообщество



Войти

Рекомендовать

**Поделиться** 

Лучшее в начале

Присоединиться к обсуждению...

войти с помощью

или через disqus (?)

Имя



**Антон** • 2 года назад

Стоит заметить, что на WP8.1 System.Threading.Thread.CurrentThread.Currer уже не прокатит.

Достаточно простого return CultureInfo.CurrentUlCulture;

Ответить • Поделиться >



Phenman → Антон • год назад с выходом Win 10 с её UWP под WP8.1 что-то писать не имеет никакого смысла.

◆ Ответить • Поделиться >

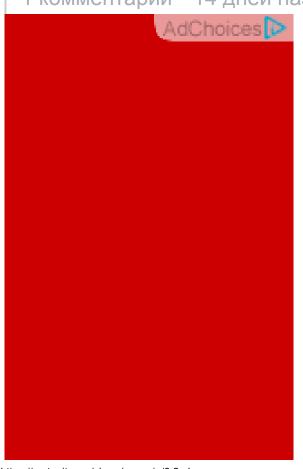
**TAKKE HA METANIT.COM** 

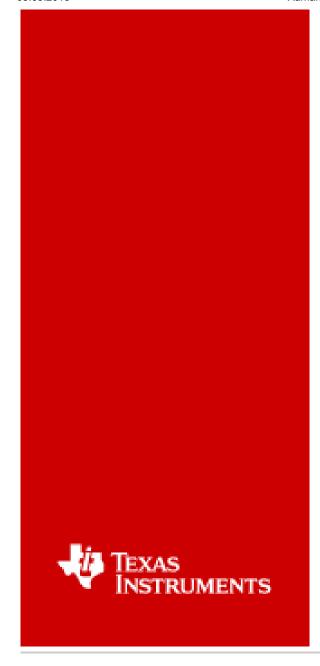
# Django | Статические файлы

1 комментарий • 14 дней назад

#### Введение в язык Go

5 комментариев • 3 месяца назад





<u>Вконтакте</u> | <u>Телеграм</u> | <u>Twitter</u> | <u>Google+</u> | <u>Youtube</u> | <u>Помощь сайту</u>

Контакты для связи: metanit22@mail.ru

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