Бесплатная электронная книга

УЧУСЬ

Xamarin.Forms

Free unaffiliated eBook created from
Stack Overflow contributors.

#xamarin.fo

rms

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: xamarin-forms

It is an unofficial and free Xamarin.Forms ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Xamarin.Forms.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с Xamarin.Forms

замечания

Xamarin.Forms позволяет создавать приложения iOS, Android и Windows с большим количеством общего кода, включая код пользовательского интерфейса или разметку XAML UI. Страницы приложений и представления сопоставляются с собственными элементами управления на каждой платформе, но могут быть настроены для обеспечения пользовательского интерфейса, специфичного для платформы, или для доступа к специфичным для платформы функциям.

Версии

Версия	Дата выхода
2.3.1	2016-08-03
2.3.0-Hotfix1	2016-06-29
2.3.0	2016-06-16
2.2.0-Hotfix1	2016-05-30
2.2.0	2016-04-27
2.1.0	2016-03-13
2.0.1	2016-01-20
2.0.0	2015-11-17
1.5.1	2016-10-20
1.5.0	2016-09-25
1.4.4	2015-07-27
1.4.3	2015-06-30
1.4.2	2015-04-21
1.4.1	2015-03-30
1.4.0	2015-03-09
1.3.5	2015-03-02

Версия	Дата выхода
1.3.4	2015-02-17
1.3.3	2015-02-09
1.3.2	2015-02-03
1.3.1	2015-01-04
1.3.0	2014-12-24
1.2.3	2014-10-02
1.2.2	2014-07-30
1.2.1	2014-07-14
1.2.0	2014-07-11
1.1.1	2014-06-19
1.1.0	2014-06-12
1.0.1	2014-06-04

Examples

Установка (Visual Studio)

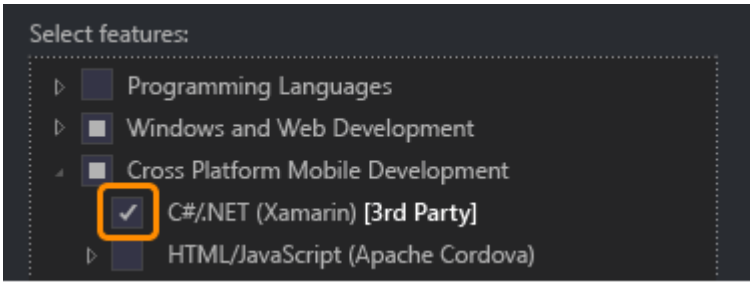
Xamarin.Forms - это кросс-платформенная абстрактная абстракция пользовательского интерфейса, которая позволяет разработчикам легко создавать пользовательские интерфейсы, которые могут совместно использоваться в Android, iOS, Windows и Windows Phone. Пользовательские интерфейсы визуализируются с использованием собственных элементов управления целевой платформы, что позволяет приложениям Xamarin.Forms сохранять соответствующий внешний вид для каждой платформы.

Плагин Xamarin для Visual Studio

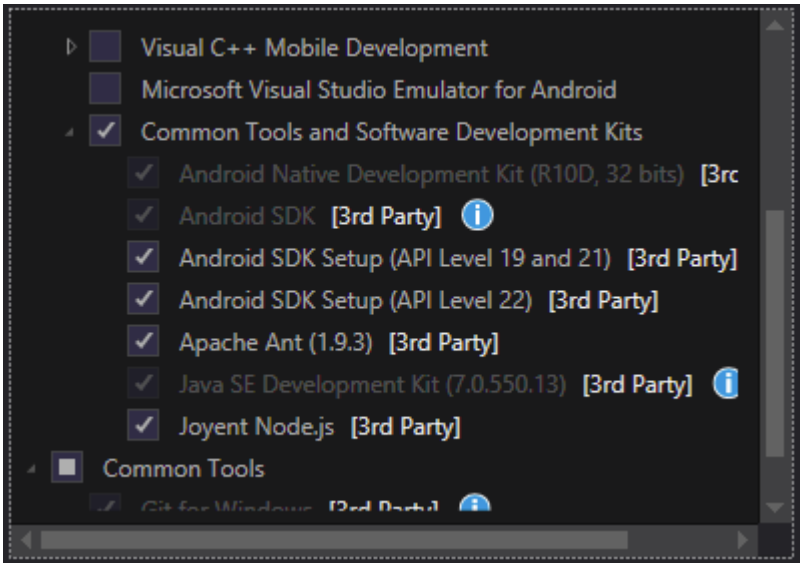
Чтобы начать работу с Xamarin.Forms для Visual Studio, вам нужно иметь сам плагин Xamarin. Самый простой способ установить его - загрузить и установить последнюю версию Visual Studio.

Если у вас уже установлена последняя версия Visual Studio, откройте «Панель управления»> «Программы и компоненты», щелкните правой кнопкой мыши на Visual Studio и нажмите «Изменить». Когда откроется программа установки, нажмите «Изменить» и

выберите инструменты для совместной разработки кросс-платформенной платформы:



Вы также можете выбрать установку Android SDK:



Снимите этот флажок, если у вас уже установлен SDK. Вы сможете настроить Xamarin для использования существующего Android SDK позже.

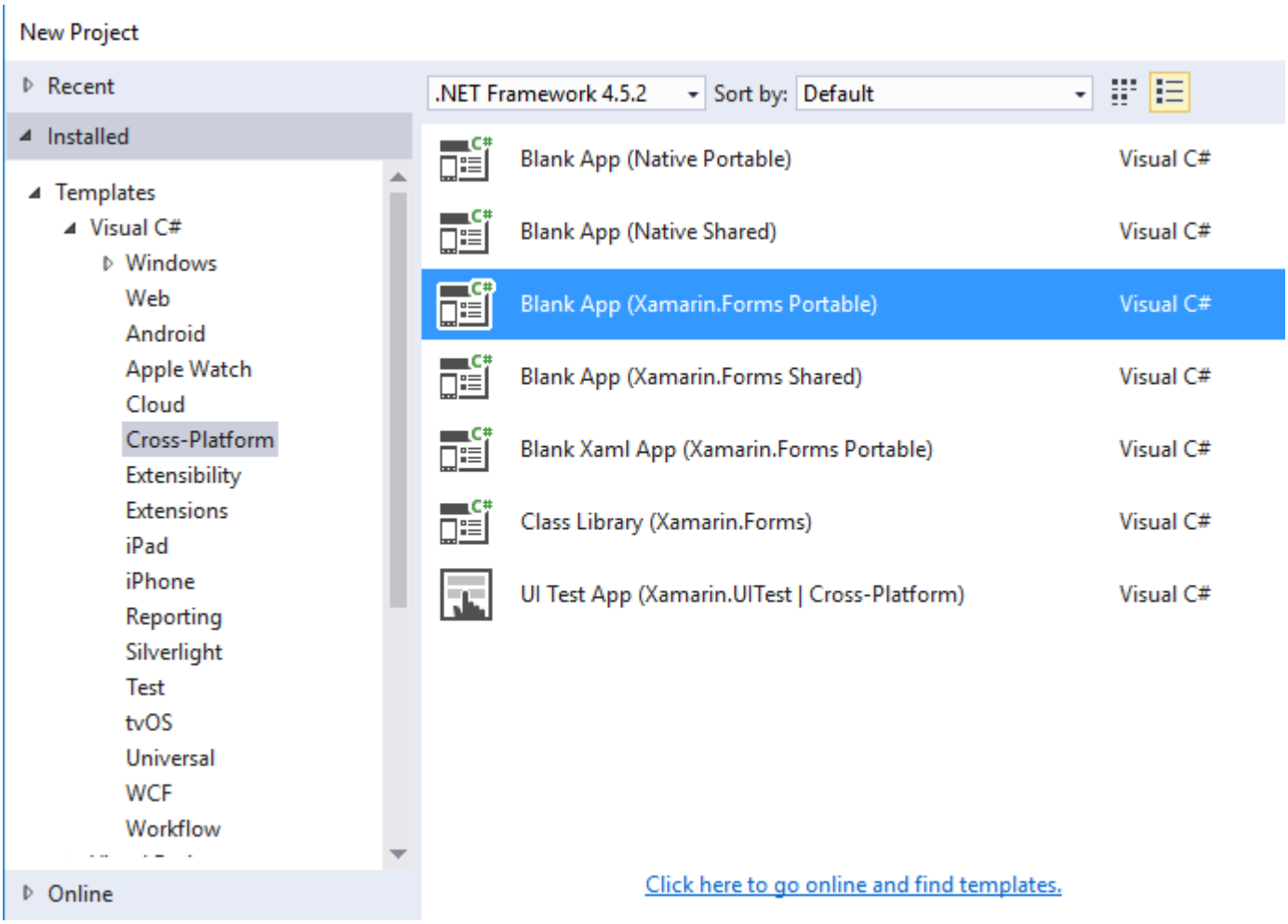
Xamarin.Forms

Xamarin.Forms - это набор библиотек для вашей библиотеки Portable Class и собственных сборок. Сама библиотека Xamarin.Forms доступна как пакет NuGet. Чтобы добавить его в свой проект, просто используйте обычную команду `Install-Package` консоли диспетчера пакетов:

```
Install-Package Xamarin.Forms
```

для всех ваших исходных сборок (например, MyProject, MyProject.Droid и MyProject.iOS).

Самый простой способ начать работу с Xamarin.Forms - создать пустой проект в Visual Studio:



Как вы можете видеть, есть два доступных варианта создания пустого приложения - Portable и Shared. Я рекомендую вам начать работу с Portable one, потому что он наиболее часто используется в реальном мире (различия и дополнительное объяснение добавляются).

После создания проекта убедитесь, что вы используете последнюю версию Xamarin.Forms, поскольку исходный шаблон может содержать старый. Используйте консоль диспетчера пакетов или «Управление пакетами NuGet» для обновления до последних версий Xamarin.Forms (помните, что это всего лишь пакет NuGet).

Хотя шаблоны Visual Studio Xamarin.Forms создадут для вас проект платформы iOS, вам нужно будет подключить Xamarin к хосту сборки Mac, чтобы иметь возможность запускать эти проекты на симуляторе iOS или на физических устройствах.

Hello World Xamarin Forms: Visual Studio

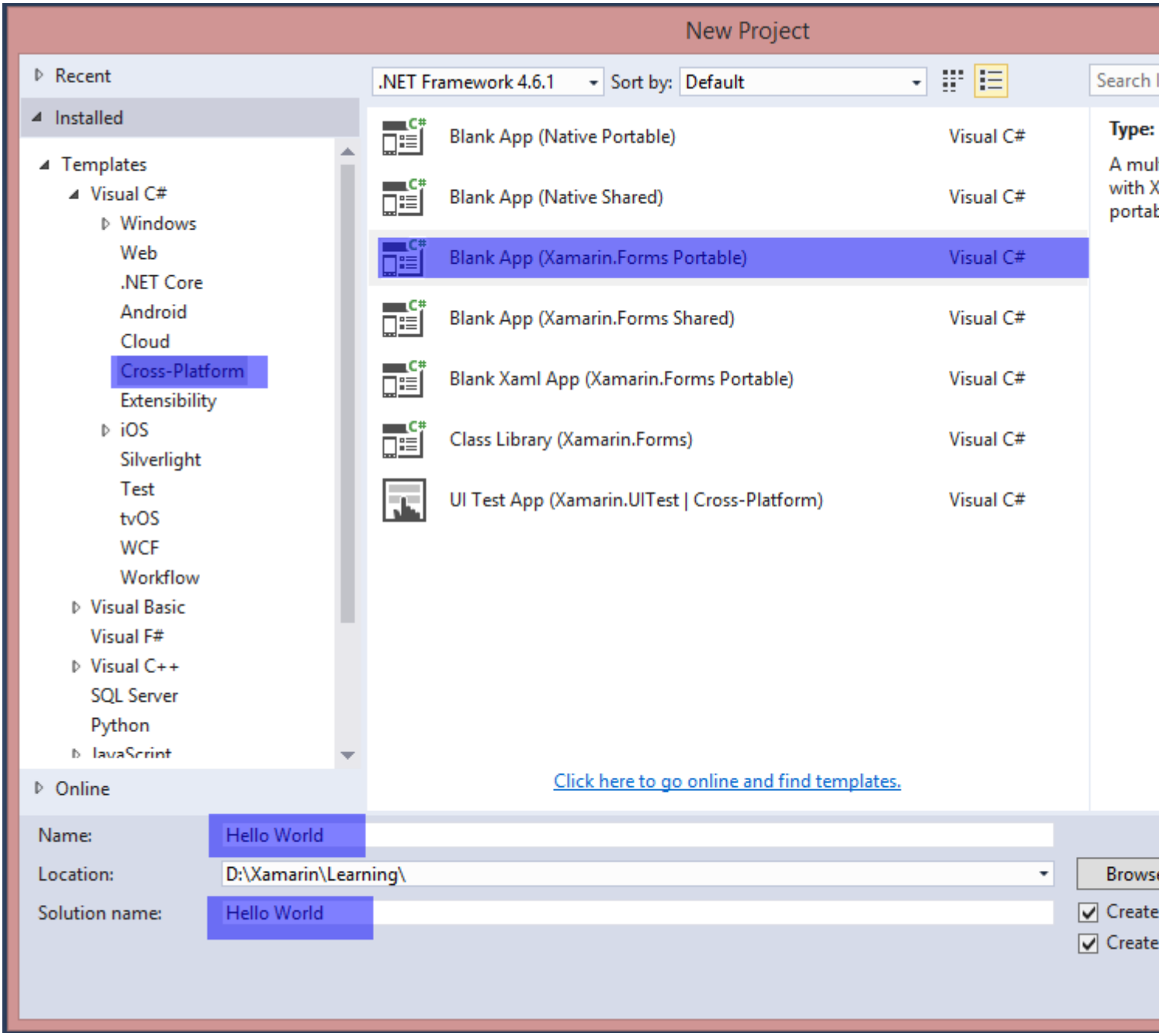
После успешной установки Xamarin, как описано в первом примере, пришло время запустить первое примерное приложение.

Шаг 1. Создание нового проекта.

В Visual Studio выберите New -> Project -> Visual C # -> Cross-Platform -> Blank App (Xamarin.Forms Portable)

Назовите приложение «Hello World» и выберите место для создания проекта и нажмите «OK». Это создаст для вас решение, которое содержит три проекта:

1. HelloWorld (здесь размещается ваша логика и представления, то есть переносимый проект)
2. HelloWorld.Droid (проект Android)
3. HelloWorld.iOS (проект iOS)



Шаг 2: Исследование образца

Создав решение, образец приложения будет готов к развертыванию. Откройте `App.cs` расположенную в корне переносного проекта и исследуйте код. Как видно ниже, `Content` s образца представляет собой `StackLayout` котором содержится `Label` :

```
using Xamarin.Forms;

namespace Hello_World
{
    public class App : Application
    {
        public App()
        {
            // The root page of your application
            MainPage = new ContentPage
            {
                Content = new StackLayout
                {
                    VerticalOptions = LayoutOptions.Center,
                    Children = {
                        new Label {
                            HorizontalTextAlignment = TextAlignment.Center,
                            Text = "Welcome to Xamarin Forms!"
                        }
                    }
                }
            };
        }
        protected override void OnStart()
        {
            // Handle when your app starts
        }
        protected override void OnSleep()
        {
            // Handle when your app sleeps
        }
        protected override void OnResume()
        {
            // Handle when your app resumes
        }
    }
}
```

Шаг 3: Запуск приложения

Теперь просто щелкните правой кнопкой мыши проект, который вы хотите запустить (`HelloWorld.Droid` или `HelloWorld.iOS`), и нажмите « `Set as StartUp Project` . Затем на панели инструментов Visual Studio нажмите кнопку « `Start` (зеленая треугольная кнопка, напоминающая кнопку «Воспроизведение»), чтобы запустить приложение на целевом эмуляторе / эмуляторе.

Прочитайте Начало работы с Xamarin.Forms онлайн: https://riptutorial.com/ru/xamarin-forms/topic/908/начало-работы-с-xamarin-forms

глава 2: CarouselView - предварительная версия

замечания

CarouselView - это Xamarin Control, который может содержать любой вид View. Этот предварительный релиз можно использовать только в проектах Xamarin Forms.

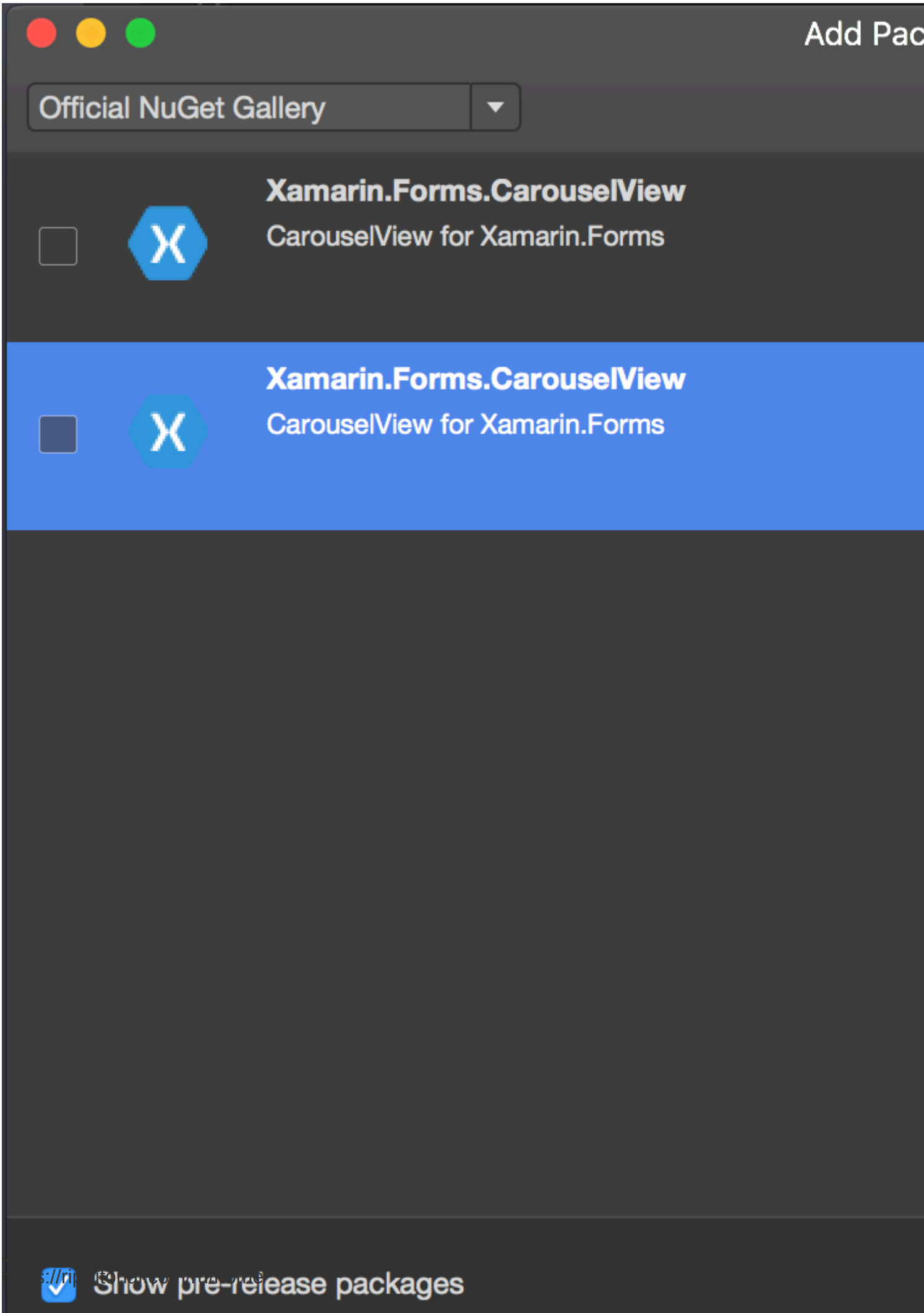
В примере, представленном Джеймсом Монтеманьо , в блоге Xamarin, CarouselView используется для отображения изображений.

В настоящий момент CarouselView не интегрирован в Xamarin.Forms. Чтобы использовать это в своих проектах, вам нужно будет добавить пакет NuGet (см. Пример выше).

Examples

Импортировать CarouselView

Самый простой способ импортировать CarouselView - использовать диспетчер NuGet-Packages в студии Xamarin / Visual:


Add Pac
Official NuGet Gallery
Xamarin.Forms.CarouselView
CarouselView for Xamarin.Forms
Xamarin.Forms.CarouselView
CarouselView for Xamarin.Forms
Show pre-release packages

за файлом. Это основа, которую вам нужно сделать для интеграции CarouselView в представление. Приведенные примеры не покажут вам ничего, потому что CarouselView пуст.

Создание связующего источника

В качестве примера ItemSource я буду использовать ObservableCollection строк.

```
public ObservableCollection<TechGiant> TechGiants { get; set; }
```

TechGiant - это класс, в котором будут представлены имена Технологических гигантов

```
public class TechGiant
{
    public string Name { get; set; }

    public TechGiant(string Name)
    {
        this.Name = Name;
    }
}
```

После InitializeComponent вашей страницы создайте и заполните ObservableCollection

```
TechGiants = new ObservableCollection<TechGiant>();
TechGiants.Add(new TechGiant("Xamarin"));
TechGiants.Add(new TechGiant("Microsoft"));
TechGiants.Add(new TechGiant("Apple"));
TechGiants.Add(new TechGiant("Google"));
```

Наконец, установите TechGiants как ItemSource DemoCarouselView

```
DemoCarouselView.ItemsSource = TechGiants;
```

DataTemplates

В XAML-файле дайте CarouselView DataTemplate:

```
<cv:CarouselView.ItemTemplate>
</cv:CarouselView.ItemTemplate>
```

Определите DataTemplate. В этом случае это будет метка с привязкой текста к источнику элементов и зеленым фоном:

```
<DataTemplate>
    <Label Text="{Binding Name}" BackgroundColor="Green"/>
</DataTemplate>
```

Это оно! Запустите программу и посмотрите результат!

Прочитайте CarouselView - предварительная версия онлайн: https://riptutorial.com/ru/xamarin-forms/topic/6094/carouselview---предварительная-версия

глава 3: Contact Picker - Xamarin Forms (Android и iOS)

замечания

Контакты Picker XF (Android и iOS)

Examples

contact_picker.cs

```
using System;

using Xamarin.Forms;

namespace contact_picker
{
    public class App : Application
    {
        public App ()
        {
            // The root page of your application
            MainPage = new MyPage();
        }

        protected override void OnStart ()
        {
            // Handle when your app starts
        }

        protected override void OnSleep ()
        {
            // Handle when your app sleeps
        }

        protected override void OnResume ()
        {
            // Handle when your app resumes
        }
    }
}
```

MyPage.cs

```
using System;

using Xamarin.Forms;

namespace contact_picker
{
```

```
    public class MyPage : ContentPage
    {
        Button button;
        public MyPage ()
        {
            button = new Button {
                Text = "choose contact"
            };

            button.Clicked += async (object sender, EventArgs e) => {

                    if (Device.OS == TargetPlatform.iOS) {
                        await Navigation.PushModalAsync (new ChooseContactPage ());
                        }
                    else if (Device.OS == TargetPlatform.Android)
                    {
                        MessagingCenter.Send (this, "android_choose_contact", "number1");
                    }


            };

            Content = new StackLayout {
                Children = {
                    new Label { Text = "Hello ContentPage" },
                    button
                }
            };
        }

        protected override void OnSizeAllocated (double width, double height)
        {
            base.OnSizeAllocated (width, height);

            MessagingCenter.Subscribe<MyPage, string> (this, "num_select", (sender, arg) => {
                DisplayAlert ("contact", arg, "OK");
            });

        }
    }
}
```

ChooseContactPicker.cs

```
using System;
using Xamarin.Forms;

namespace contact_picker
{
    public class ChooseContactPage : ContentPage
    {
        public ChooseContactPage ()
        {

        }
    }
}
```

ChooseContactActivity.cs

```
using Android.App;
using Android.OS;
using Android.Content;
using Android.Database;
using Xamarin.Forms;

namespace contact_picker.Droid
{

    [Activity (Label = "ChooseContactActivity")]

    public class ChooseContactActivity : Activity
    {
        public string type_number = "";
        protected override void OnCreate (Bundle savedInstanceState)
        {

            base.OnCreate (savedInstanceState);

            Intent intent = new Intent(Intent.ActionPick,
Android.Provider.ContactsContract.CommonDataKinds.Phone.ContentUri);
            StartActivityForResult(intent, 1);
        }

        protected override void OnActivityResult (int requestCode, Result resultCode, Intent
data)
        {
            // TODO Auto-generated method stub

            base.OnActivityResult (requestCode, resultCode, data);
            if (requestCode == 1) {
                if (resultCode == Result.Ok) {

                    Android.Net.Uri contactData = data.Data;
                    ICursor cursor = ContentResolver.Query(contactData, null, null, null,
null);

                    cursor.MoveToFirst();

                    string number =
cursor.GetString(cursor.GetColumnIndexOrThrow(Android.Provider.ContactsContract.CommonDataKinds.Phone.N



                    var twopage_renderer = new MyPage();
                    MessagingCenter.Send<MyPage, string> (twopage_renderer, "num_select",
number);
                    Finish ();
                    Xamarin.Forms.Application.Current.MainPage.Navigation.PopModalAsync ();


                }
                else if (resultCode == Result.Canceled)
                {
                    Finish ();
                }
            }
        }
```

```
    }
}
```

MainActivity.cs

```
using System;

using Android.App;
using Android.Content;
using Android.Content.PM;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using Xamarin.Forms;

namespace contact_picker.Droid
{
    [Activity (Label = "contact_picker.Droid", Icon = "@drawable/icon", MainLauncher = true,
ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation)]
    public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsApplicationActivity
    {
        protected override void OnCreate (Bundle bundle)
        {
            base.OnCreate (bundle);

            global::Xamarin.Forms.Forms.Init (this, bundle);

            LoadApplication (new App ());


            MessagingCenter.Subscribe<MyPage, string>(this, "android_choose_contact", (sender,
args) => {
                Intent i = new Intent (Android.App.Application.Context,
typeof(ChooseContactActivity));
                i.PutExtra ("number1", args);
                StartActivity (i);
            });
        }
    }
}
```

ChooseContactRenderer.cs

```
using UIKit;
using AddressBookUI;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;
using contact_picker;
using contact_picker.iOS;


[assembly: ExportRenderer (typeof(ChooseContactPage), typeof(ChooseContactRenderer))]

namespace contact_picker.iOS
{
```

```
    public partial class ChooseContactRenderer : PageRenderer
    {
        ABPeoplePickerNavigationController _contactController;

        public string type_number;

        protected override void OnElementChanged (VisualElementChangedEventArgs e)
        {
            base.OnElementChanged (e);

            var page = e.NewElement as ChooseContactPage;

            if (e.OldElement != null || Element == null) {
                return;

            }

        }

        public override void ViewDidLoad ()
        {
            base.ViewDidLoad ();

            _contactController = new ABPeoplePickerNavigationController ();

            this.PresentModalViewController (_contactController, true); //display contact
chooser


            _contactController.Cancelled += delegate {
                Xamarin.Forms.Application.Current.MainPage.Navigation.PopModalAsync ();

                this.DismissModalViewController (true); };

            _contactController.SelectPerson2 += delegate(object sender,
ABPeoplePickerSelectPerson2EventArgs e) {

                var getphones = e.Person.GetPhones();
                string number = "";

                if (getphones == null)
                {
                    number = "Nothing";
                }
                else if (getphones.Count > 1)
                {
                    //il ya plus de 2 num de telephone
                    foreach(var t in getphones)
                    {
                        number = t.Value + "/" + number;
                    }
                }
                else if (getphones.Count == 1)
                {
                    //il ya 1 num de telephone
                    foreach(var t in getphones)
                    {
                        number = t.Value;
                    }
                }
```

```
                Xamarin.Forms.Application.Current.MainPage.Navigation.PopModalAsync ();


                var twopage_renderer = new MyPage();
                MessagingCenter.Send<MyPage, string> (twopage_renderer, "num_select", number);
                this.DismissModalViewController (true);



            };
        }

        public override void ViewDidUnload ()
        {
            base.ViewDidUnload ();

            // Clear any references to subviews of the main view in order to
            // allow the Garbage Collector to collect them sooner.
            //
            // e.g. myOutlet.Dispose (); myOutlet = null;

            this.DismissModalViewController (true);
        }

        public override bool ShouldAutorotateToInterfaceOrientation (UIInterfaceOrientation
toInterfaceOrientation)
        {
            // Return true for supported orientations
            return (toInterfaceOrientation != UIInterfaceOrientation.PortraitUpsideDown);
        }
    }
}
```

Прочитайте Contact Picker - Xamarin Forms (Android и iOS) онлайн:
https://riptutorial.com/ru/xamarin-forms/topic/6659/contact-picker---xamarin-forms--android-и-ios-

глава 4: DependencyService

замечания

При использовании `DependencyService` вам обычно требуется 3 части:

- **Интерфейс** - определяет функции, которые вы хотите абстрагировать.
- **Реализация платформы** - класс в рамках каждого конкретного проекта, реализующего ранее определенный интерфейс.
- **Регистрация.** Каждый класс реализации конкретной платформы должен быть зарегистрирован в `DependencyService` через атрибут метаданных. Это позволяет `DependencyService` находить вашу реализацию во время выполнения.

При использовании `DependencyService` вам необходимо предоставить реализацию для каждой целевой платформы. Когда реализация не предоставляется, приложение будет работать во время выполнения.

Examples

Интерфейс

Интерфейс определяет поведение, которое вы хотите открыть через `DependencyService` . Одним из примеров использования `DependencyService` является служба Text-To-Speech. В настоящее время абстракции для этой функции в Xamarin.Forms нет, поэтому вам нужно создать свой собственный. Начните с определения интерфейса для поведения:

```
public interface ITextToSpeech
{
       void Speak (string whatToSay);
}
```

Поскольку мы определяем наш интерфейс, мы можем сопоставить его с нашим общим кодом.

Примечание. Классы, реализующие интерфейс, должны иметь конструктор без параметров для работы с `DependencyService` .

реализация iOS

Определенный вами интерфейс должен быть реализован на каждой целевой платформе. Для iOS это выполняется через структуру `AVFoundation` . Следующая реализация интерфейса `ITextToSpeech` обрабатывает данный текст на английском языке.

```
using AVFoundation;

public class TextToSpeechiOS : ITextToSpeech
{
    public TextToSpeechiOS () {}

    public void Speak (string whatToSay)
    {
        var speechSynthesizer = new AVSpeechSynthesizer ();

        var speechUtterance = new AVSpeechUtterance (whatToSay) {
            Rate = AVSpeechUtterance.MaximumSpeechRate/4,
            Voice = AVSpeechSynthesisVoice.FromLanguage ("en-US"),
            Volume = 0.5f,
            PitchMultiplier = 1.0f
        };

        speechSynthesizer.SpeakUtterance (speechUtterance);
    }
}
```

Когда вы создали свой класс, вам нужно включить `DependencyService` для его обнаружения во время выполнения. Это делается добавлением атрибута `[assembly]` выше определения класса и вне любых определений пространства имен.

```
using AVFoundation;
using DependencyServiceSample.iOS;

[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechiOS))]
namespace DependencyServiceSample.iOS {
    public class TextToSpeechiOS : ITextToSpeech
...
```

Этот атрибут регистрирует класс с помощью `DependencyService` поэтому его можно использовать, когда необходим экземпляр интерфейса `ITextToSpeech` .

Общий код

После того, как вы создали и зарегистрировали свои классы для платформы, вы можете начать подключать их к своему коду. Следующая страница содержит кнопку, которая запускает функции преобразования текста в речь с использованием заранее определенного предложения. Он использует `DependencyService` для извлечения реализации `ITextToSpeech` на `ITextToSpeech` во время выполнения с использованием собственных SDK.

```
public MainPage ()
{
    var speakButton = new Button {
        Text = "Talk to me baby!",
        VerticalOptions = LayoutOptions.CenterAndExpand,
        HorizontalOptions = LayoutOptions.CenterAndExpand,
    };

    speakButton.Clicked += (sender, e) => {
        DependencyService.Get<ITextToSpeech>().Speak("Xamarin Forms likes eating cake by the
```

```
ocean.");
    };

    Content = speakButton;
}
```

Когда вы запускаете это приложение на устройстве iOS или Android и нажимаете кнопку, вы услышите, как приложение произносит данное предложение.

Реализация Android

Специфическая реализация Android немного сложнее, потому что она заставляет вас наследовать от родного `Java.Lang.Object` и заставляет вас реализовать интерфейс `IOnInitListener` . Android требует, чтобы вы предоставили действительный контекст Android для многих методов SDK, которые он предоставляет. Xamarin.Forms предоставляет объект `Forms.Context` который предоставляет вам контекст Android, который вы можете использовать в таких случаях.

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;

public class TextToSpeechAndroid : Java.Lang.Object, ITextToSpeech,
TextToSpeech.IOnInitListener
{
    TextToSpeech _speaker;

    public TextToSpeechAndroid () {}

    public void Speak (string whatToSay)
    {
        var ctx = Forms.Context;

        if (_speaker == null)
        {
            _speaker = new TextToSpeech (ctx, this);
        }
        else
        {
            var p = new Dictionary<string,string> ();
            _speaker.Speak (whatToSay, QueueMode.Flush, p);
        }
    }

    #region IOnInitListener implementation

    public void OnInit (OperationResult status)
    {
        if (status.Equals (OperationResult.Success))
        {
            var p = new Dictionary<string,string> ();
            _speaker.Speak (toSpeak, QueueMode.Flush, p);
        }
    }
```

```
    #endregion
}
```

Когда вы создали свой класс, вам нужно включить `DependencyService` для его обнаружения во время выполнения. Это делается добавлением атрибута `[assembly]` выше определения класса и вне любых определений пространства имен.

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;


[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechAndroid))]
namespace DependencyServiceSample.Droid {
    ...
```

Этот атрибут регистрирует класс с помощью `DependencyService` поэтому его можно использовать, когда необходим экземпляр интерфейса `ITextToSpeech` .

Прочитайте DependencyService онлайн: https://riptutorial.com/ru/xamarin-forms/topic/2508/dependencyservice

глава 5: MessagingCenter

Вступление

Xamarin.Forms имеет встроенный механизм обмена сообщениями для продвижения развязанного кода. Таким образом, модели просмотра и другие компоненты не должны знать друг друга. Они могут общаться по простому договору обмена сообщениями.

Там в основном два основных компонента для использования `MessagingCenter` .

Подписаться ; прослушивать сообщения с определенной подписью (договор) и выполнять код при получении сообщения. Сообщение может иметь несколько подписчиков.

Отправить ; отправка сообщения для подписчиков.

Examples

Простой пример

Здесь мы увидим простой пример использования MessagingCenter в Xamarin.Forms.

Во-первых, давайте посмотрим на подписку на сообщение. В модели `FooMessaging` мы подписываемся на сообщение, поступающее из `MainPage` . Сообщение должно быть «Привет», и когда мы его получим, мы регистрируем обработчик, который устанавливает свойство `Greeting` . Наконец, `this` означает, что текущий экземпляр `FooMessaging` регистрируется для этого сообщения.

```
public class FooMessaging
{
    public string Greeting { get; set; }

    public FooMessaging()
    {
        MessagingCenter.Subscribe<MainPage> (this, "Hi", (sender) => {
            this.Greeting = "Hi there!";
        });
    }
}
```

Чтобы отправить сообщение, запускающее эту функцию, нам нужно иметь страницу под названием `MainPage` и реализовать код, как показано ниже.

```
public class MainPage : Page
{
    private void OnButtonClick(object sender, EventArgs args)
    {
        MessagingCenter.Send<MainPage> (this, "Hi");
```

```
    }
}
```

В нашей `MainPage` у нас есть кнопка с обработчиком, который отправляет сообщение. `this` должен быть экземпляр `MainPage` .

Передача аргументов

Вы также можете передавать аргументы с сообщением для работы.

Мы будем использовать классифицированные из нашего предыдущего примера и расширять их. В принимающей части, прямо за вызовом метода `Subscribe` добавьте тип аргумента, который вы ожидаете. Также убедитесь, что вы также объявляете аргументы в подписи обработчика.

```
public class FooMessaging
{
    public string Greeting { get; set; }

    public FooMessaging()
    {
        MessagingCenter.Subscribe<MainPage, string> (this, "Hi", (sender, arg) => {
            this.Greeting = arg;
        });
    }
}
```

При отправке сообщения обязательно укажите значение аргумента. Кроме того, здесь вы добавляете тип прямо за методом `Send` и добавляете значение аргумента.

```
public class MainPage : Page
{
    private void OnButtonClick(object sender, EventArgs args)
    {
        MessagingCenter.Send<MainPage, string> (this, "Hi", "Hi there!");
    }
}
```

В этом примере используется простая строка, но вы также можете использовать любые другие (сложные) объекты.

Отказ от подписки

Когда вам больше не нужно получать сообщения, вы можете просто отказаться от подписки. Вы можете сделать это следующим образом:

```
MessagingCenter.Unsubscribe<MainPage> (this, "Hi");
```

Когда вы предоставляете аргументы, вы должны отказаться от подписки на полную подпись, например:

```
MessagingCenter.Unsubscribe<MainPage, string> (this, "Hi");
```

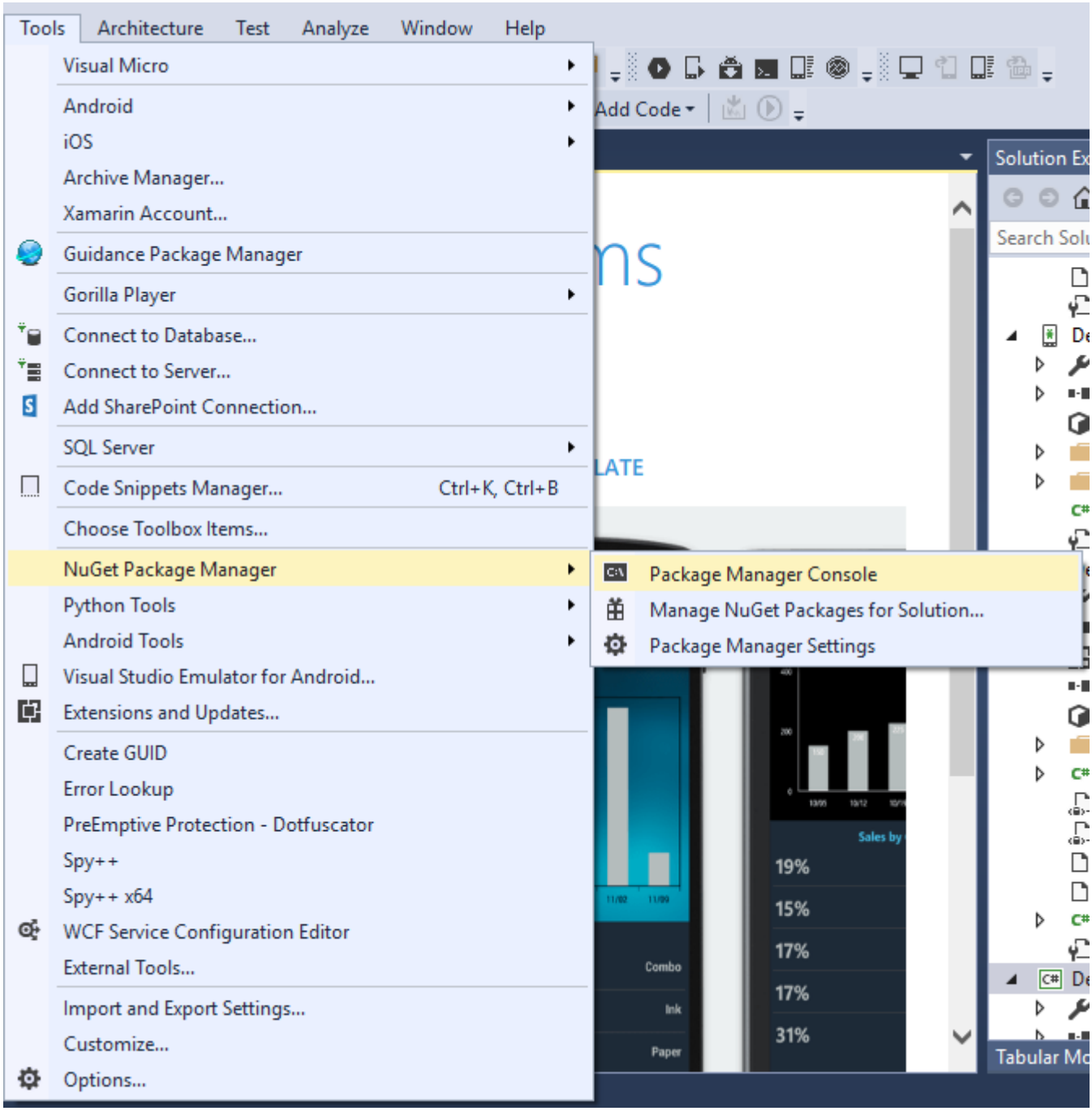
Прочитайте MessagingCenter онлайн: https://riptutorial.com/ru/xamarin-forms/topic/9672/messagingcenter

глава 6: OAuth2

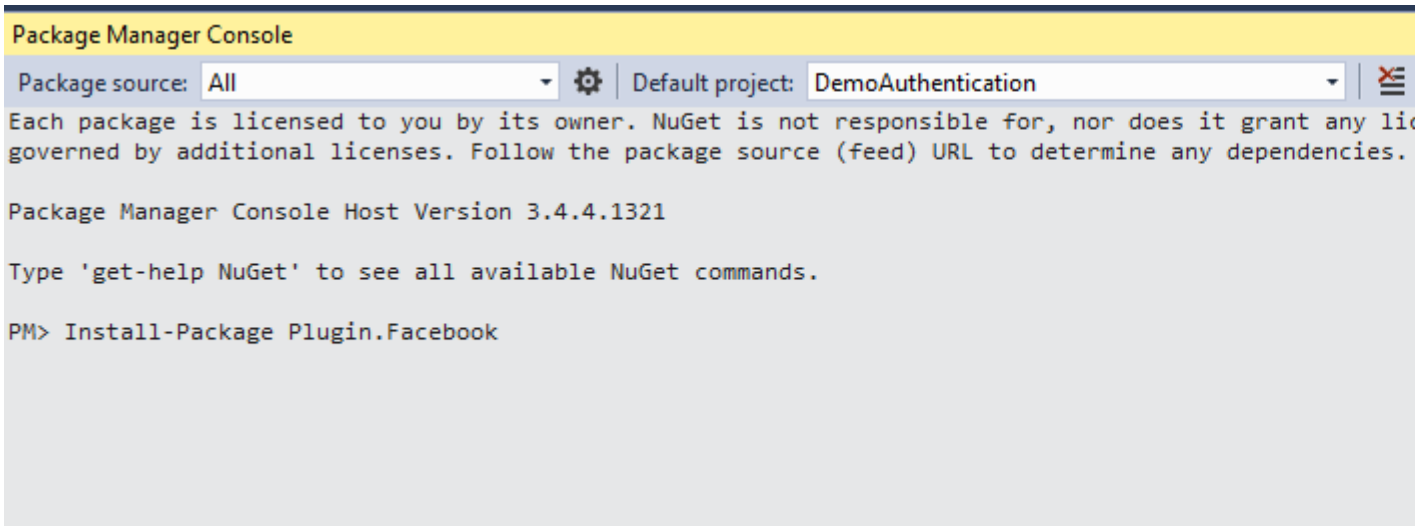
Examples

Аутентификация с помощью плагина

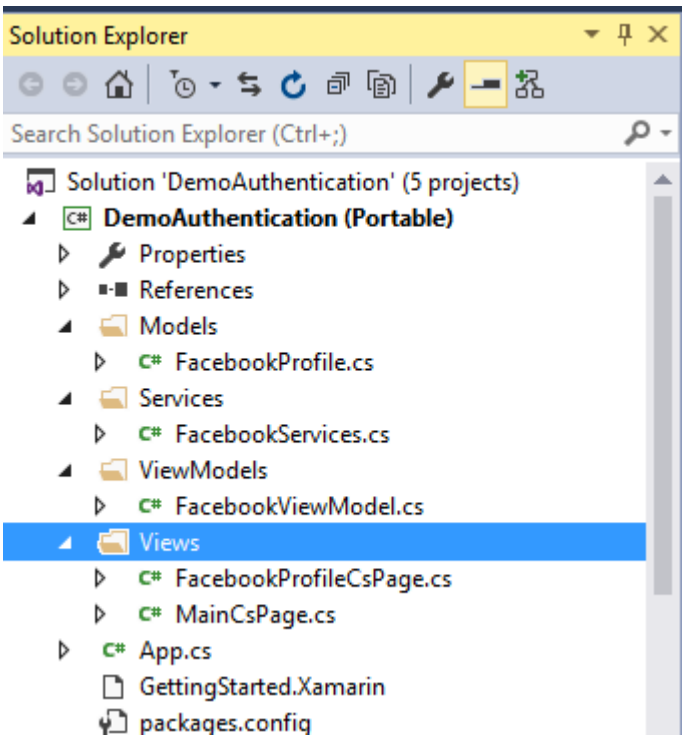
1. Сначала перейдите в меню « **Инструменты»** > « **Диспетчер пакетов NuGet»** > « **Диспетчер пакетов»** .



2. Введите эту команду « **Install-Package Plugin.Facebook** » в консоли **управления пакетами** .



3. Теперь все файлы создаются автоматически.



Видео : Войти с Facebook в Xamarin Forms

Другая аутентификация с помощью плагина. Поместите команду в консоль диспетчера пакетов, как показано на шаге 2.

1. **Youtube** : Install-Package Plugin.Youtube
2. **Twitter** : установочный пакет Plugin.Twitter
3. **Foursquare** : Install-Package Plugin.Foursquare
4. **Google** : установить пакет. Plugin.Google
5. **Instagram** : Install-Package Plugin.Instagram
6. **Eventbrite** : Install-Package Plugin.Eventbrite

Прочитайте OAuth2 онлайн: https://riptutorial.com/ru/xamarin-forms/topic/8828/oauth2

глава 7: SQL Database и API в форматах Xamarin.

замечания

Создайте свой собственный api с базой данных Microsoft SQL и реализуйте их в приложении форм Xamarin.

Examples

Создайте API с использованием базы данных SQL и реализуйте в форматах Xamarin,

Блог исходного кода

Прочитайте SQL Database и API в форматах Xamarin. онлайн: https://riptutorial.com/ru/xamarin-forms/topic/6513/sql-database-и-api-в-форматах-xamarin-

глава 8: Xamarin.Forms Просмотров

Examples

кнопка

Кнопка , вероятно, является наиболее распространенным элементом управления не только в мобильных приложениях, но и в любых приложениях, имеющих пользовательский интерфейс. Концепция кнопки имеет слишком много целей, чтобы перечислить здесь. Вообще говоря, вы будете использовать кнопку, чтобы пользователи могли инициировать какие-то действия или действия в вашем приложении. Эта операция может включать в себя что угодно: от базовой навигации в вашем приложении до отправки данных на веб-службу где-то в Интернете.

XAML

```
<Button
    x:Name="MyButton"
    Text="Click Me!"
    TextColor="Red"
    BorderColor="Blue"
    VerticalOptions="Center"
    HorizontalOptions="Center"
    Clicked="Button_Clicked"/>
```

Код XAML позади

```
public void Button_Clicked( object sender, EventArgs args )
{
    MyButton.Text = "I've been clicked!";
}
```

Код

```
var button = new Button( )
{
    Text = "Hello, Forms !",
    VerticalOptions = LayoutOptions.CenterAndExpand,
    HorizontalOptions = LayoutOptions.CenterAndExpand,
    TextColor = Color.Red,
    BorderColor = Color.Blue,
};

button.Clicked += ( sender, args ) =>
{
    var b = (Button) sender;
    b.Text = "I've been clicked!";
};
```

DatePicker

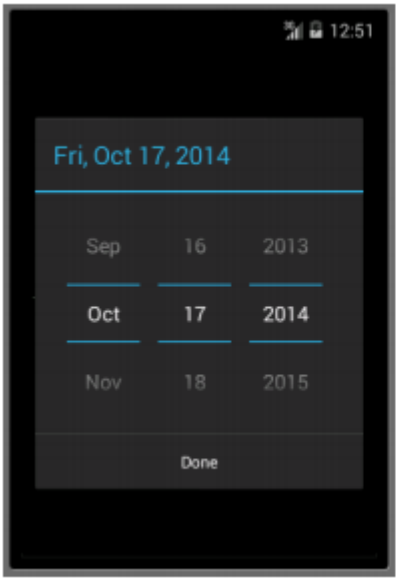
Довольно часто в мобильных приложениях будет причина иметь дело с датами. При работе с датами вам, вероятно, потребуется какой-то пользовательский ввод для выбора даты. Это может произойти при работе с графическим или календарным приложением. В этом случае лучше всего предоставить пользователям специализированный элемент управления, который позволяет им интерактивно выбирать дату, а не требовать от пользователей вручную вводить дату. Это то, где элемент управления DatePicker действительно полезен.

XAML

```
<DatePicker Date="09/12/2014" Format="d" />
```

Код

```
var datePicker = new DatePicker{
Date = DateTime.Now,
Format = "d"
};
```



запись

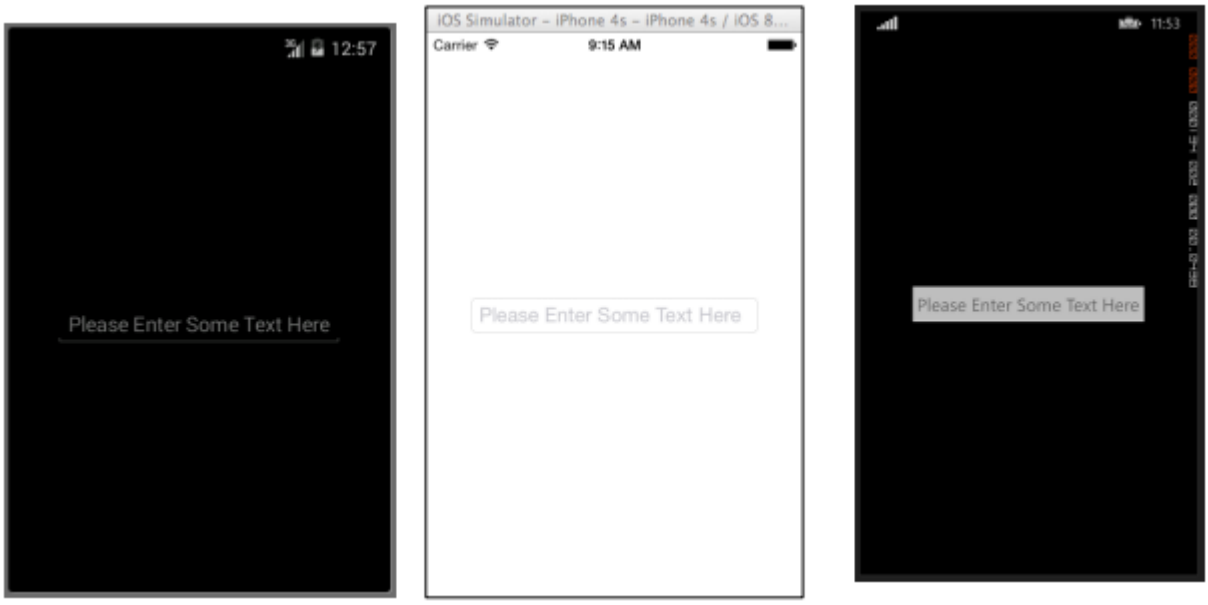
Вид входа используется, чтобы позволить пользователям вводить одну строку текста. Эта единственная строка текста может использоваться для нескольких целей, включая ввод основных заметок, учетных данных, URL-адресов и т. Д. Этот вид является многоцелевым представлением, что означает, что если вам нужно набрать обычный текст или хотите скрыть пароль, все это делается через этот единственный элемент управления.

XAML

```
<Entry Placeholder="Please Enter Some Text Here"
HorizontalOptions="Center"
VerticalOptions="Center"
Keyboard="Email"/>
```

Код

```
var entry = new Entry {
Placeholder = "Please Enter Some Text Here",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center,
Keyboard = Keyboard.Email
};
```



редактор

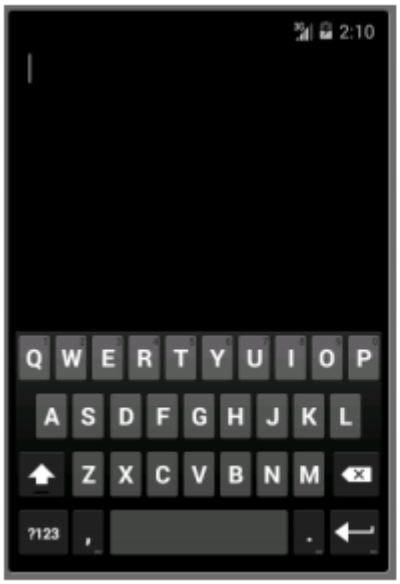
Редактор очень похож на Entry, поскольку он позволяет пользователям вводить некоторый текст свободной формы. Разница в том, что редактор допускает многострочный ввод, тогда как Entry используется только для однострочного ввода. Запись также предоставляет еще несколько свойств, чем редактор, чтобы разрешить дальнейшую настройку представления.

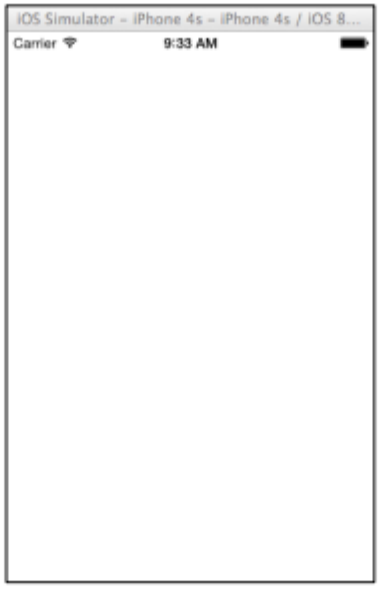
XAML

```
<Editor HorizontalOptions="Fill"
VerticalOptions="Fill"
 Keyboard="Chat"/>
```

Код

```
var editor = new Editor {
HorizontalOptions = LayoutOptions.Fill,
VerticalOptions = LayoutOptions.Fill,
Keyboard = Keyboard.Chat
};
```





Образ

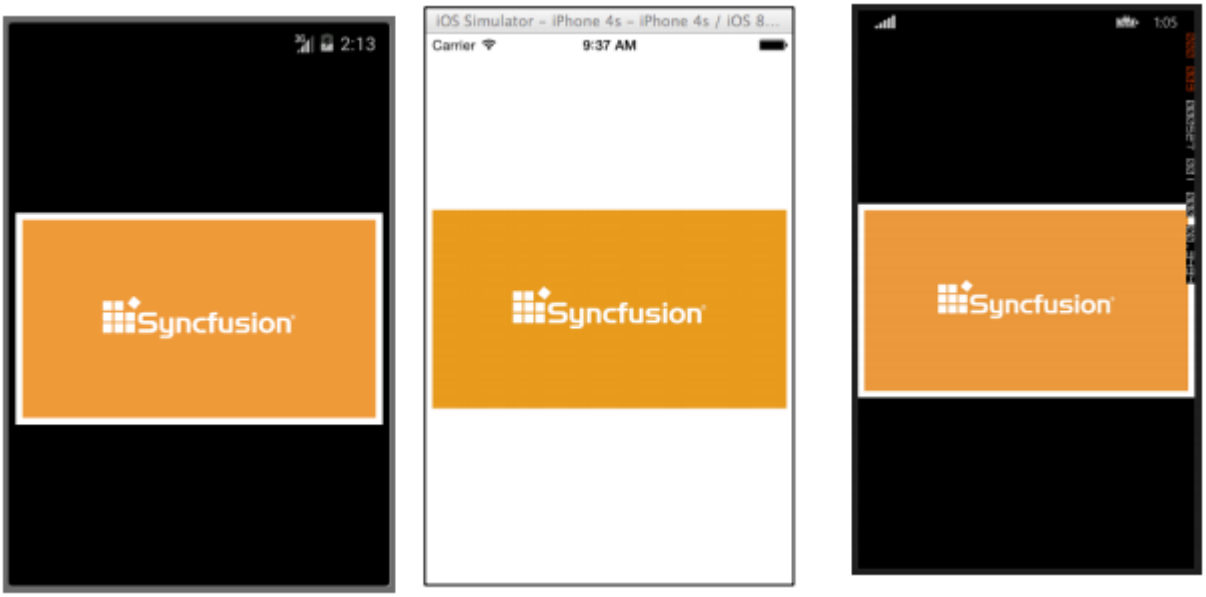
Изображения являются очень важными частями любого приложения. Они предоставляют возможность добавлять дополнительные визуальные элементы, а также брендинг в ваше приложение. Не говоря уже о том, что изображения обычно более интересны, чем текст или кнопки. Вы можете использовать изображение как отдельный элемент в своем приложении, но элемент изображения также можно добавить к другим элементам View, таким как кнопка.

XAML

```
<Image Aspect="AspectFit" Source="http://d2g29cya9iq7ip.cloudfront.net/co
ntent/images/company/aboutus-video-bg.png?v=25072014072745"/>
```

Код

```
var image = new Image {
Aspect = Aspect.AspectFit,
Source = ImageSource.FromUri(new Uri("http://d2g29cya9iq7ip.cloudfron
t.net/content/images/company/aboutus-video-bg.png?v=25072014072745"))
};
```



этикетка

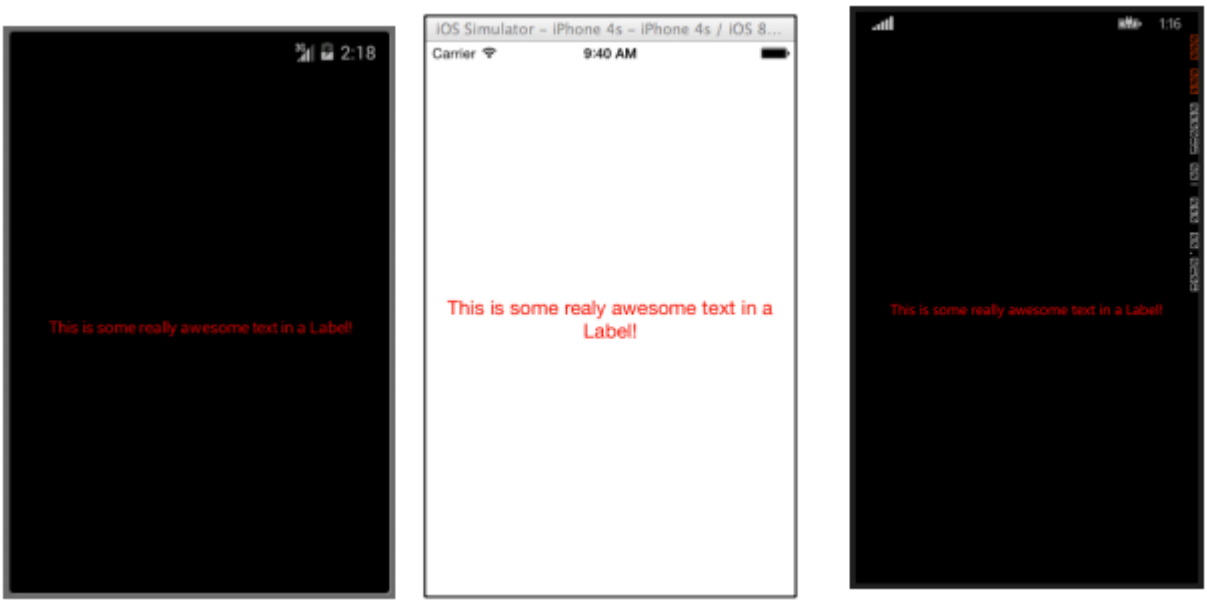
Верьте или нет, лейбл является одним из самых важных, но недооцениваемых классов View не только в Xamarin.Forms, но и в разработке пользовательского интерфейса в целом. Это рассматривается как довольно скучная строка текста, но без этой строки текста было бы очень сложно передать определенные идеи пользователю. Элементы управления метками могут использоваться для описания того, что пользователь должен ввести в редактор или элемент управления Entry. Они могут описывать раздел пользовательского интерфейса и давать ему контекст. Они могут использоваться, чтобы показать общее количество в приложении калькулятора. Да, ярлык - действительно самый универсальный элемент управления в сумке для инструмента, который может не всегда привлекать много внимания, но это первое, что заметили, если его там нет.

XAML

```
<Label Text="This is some really awesome text in a Label!"
TextColor="Red"
XAlign="Center"
YAlign="Center"/>
```

Код

```
var label = new Label {
Text = "This is some really awesome text in a Label!",
TextColor = Color.Red,
XAlign = TextAlignment.Center,
YAlign = TextAlignment.Center
};
```



Прочитайте Xamarin.Forms Просмотров онлайн: https://riptutorial.com/ru/xamarin-forms/topic/7369/xamarin-forms-просмотров

глава 9: Всплывающие уведомления

замечания

Единого способа обработки push-уведомлений в Xamarin Forms нет, поскольку реализация в значительной степени зависит от специфических функций и событий платформы. Для этого обязательно потребуется конкретный код платформы.

Однако, используя `DependencyService` вы можете использовать как можно больше кода. Также есть плагин, разработанный для этого rdelrosario, который можно найти на его GitHub .

Код и скриншоты взяты из серии блога Джеральда Верджюиса, которая объясняет процесс более подробно.

Examples

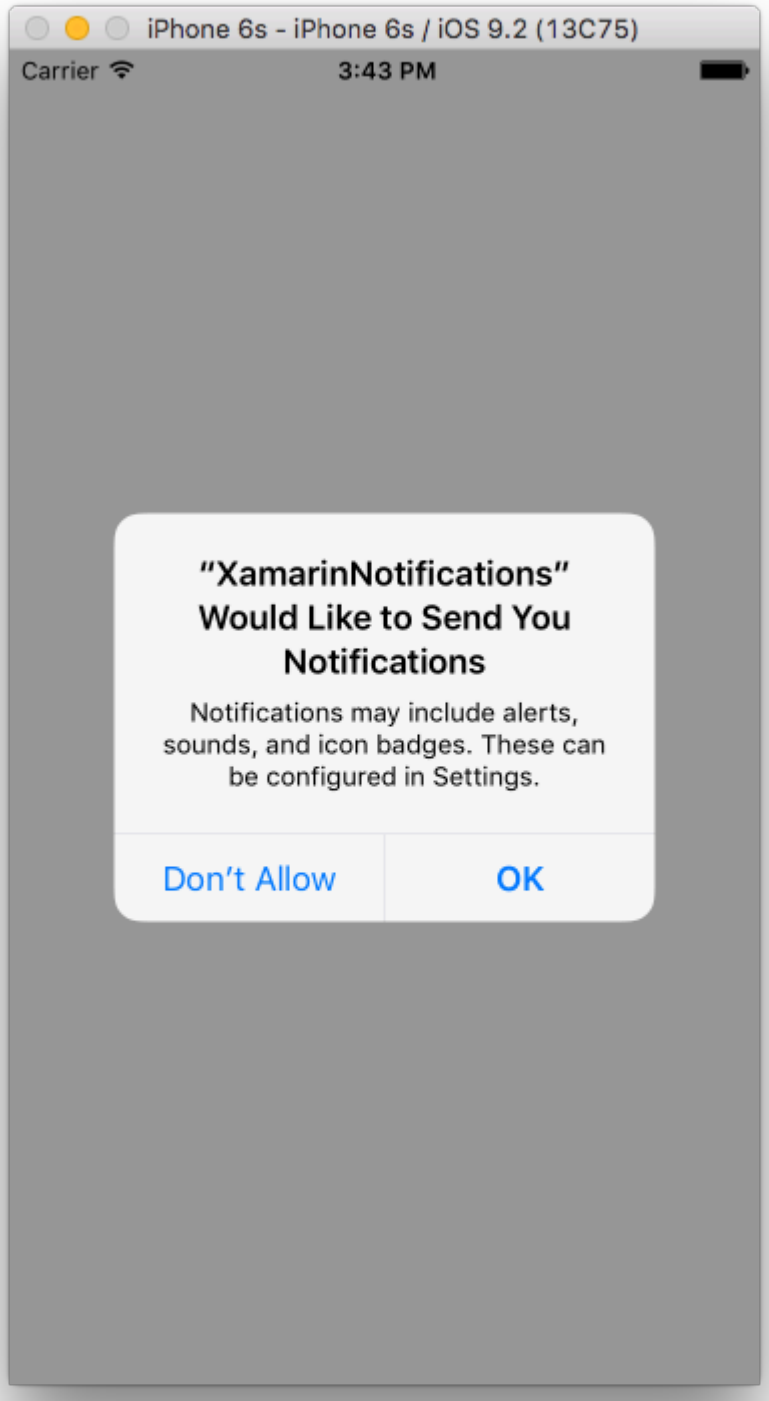
Push-уведомления для iOS с Azure

Чтобы начать регистрацию для push-уведомлений, вам необходимо выполнить приведенный ниже код.

```
// registers for push
var settings = UIUserNotificationSettings.GetSettingsForTypes(
    UIUserNotificationType.Alert
    | UIUserNotificationType.Badge
    | UIUserNotificationType.Sound,
    new NSSet());

UIApplication.SharedApplication.RegisterUserNotificationSettings(settings);
UIApplication.SharedApplication.RegisterForRemoteNotifications();
```

Этот код можно `AppDelegate.cs` сразу, когда приложение запускается в `FinishedLaunching` в файле `AppDelegate.cs` . Или вы можете делать это каждый раз, когда пользователь решает, что они хотят включить push-уведомления.

Запуск этого кода вызовет предупреждение, чтобы вызвать пользователя, если они согласятся, что приложение может отправлять им уведомления. Так же реализуйте сценарий, когда пользователь это отрицает!

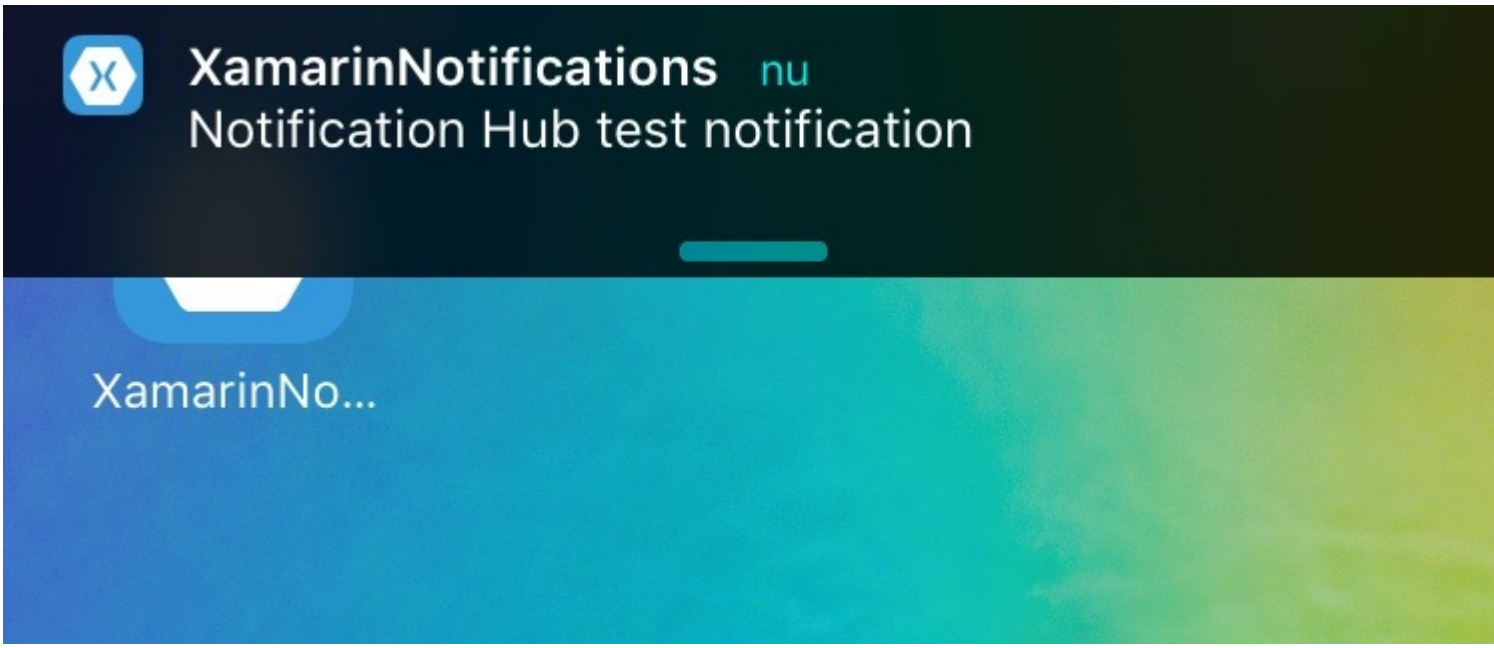


Это события, которые нуждаются в реализации для внедрения push-уведомлений в iOS. Их можно найти в файле `AppDelegate.cs` .

```
// We've successfully registered with the Apple notification service, or in our case Azure
public override void RegisteredForRemoteNotifications(UIApplication application, NSData
deviceToken)
{
   // Modify device token for compatibility Azure
    var token = deviceToken.Description;
```

```
    token = token.Trim('<', '>').Replace(" ", "");

    // You need the Settings plugin for this!
    Settings.DeviceToken = token;

    var hub = new SBNotificationHub("Endpoint=sb://xamarinnotifications-
ns.servicebus.windows.net/;SharedAccessKeyName=DefaultListenSharedAccessSignature;SharedAccessKey=<your
own key>", "xamarinnotifications");

    NSSet tags = null; // create tags if you want, not covered for now
    hub.RegisterNativeAsync(deviceToken, tags, (errorCallback) =>
    {
        if (errorCallback != null)
        {
            var alert = new UIAlertView("ERROR!", errorCallback.ToString(), null, "OK", null);
            alert.Show();
        }
    });
}

// We've received a notification, yay!
public override void ReceivedRemoteNotification(UIApplication application, NSDictionary
userInfo)
{
    NSObject inAppMessage;

    var success = userInfo.TryGetValue(new NSString("inAppMessage"), out inAppMessage);

    if (success)
    {
        var alert = new UIAlertView("Notification!", inAppMessage.ToString(), null, "OK",
null);
        alert.Show();
    }
}

// Something went wrong while registering!
public override void FailedToRegisterForRemoteNotifications(UIApplication application, NSError
error)
{
   var alert = new UIAlertView("Computer says no", "Notification registration failed! Try
again!", null, "OK", null);

   alert.Show();
}
```

Когда получено уведомление, это выглядит так.



Push-уведомления для Android с Azure

Реализация на Android - это немного больше работы и требует, чтобы определенная `Service` была реализована.

Сначала давайте посмотрим, может ли наше устройство получать push-уведомления, и если да, зарегистрируйте его в Google. Это можно сделать с помощью этого кода в нашем файле `MainActivity.cs` .

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

    global::Xamarin.Forms.Forms.Init(this, bundle);

    // Check to ensure everything's setup right for push
    GcmClient.CheckDevice(this);
    GcmClient.CheckManifest(this);
    GcmClient.Register(this, NotificationsBroadcastReceiver.SenderIDs);

    LoadApplication(new App());
}
```

Идентификаторы отправителя можно найти в коде под ним и номер проекта, который вы получаете с панели инструментов разработчика Google, чтобы отправлять push-сообщения.

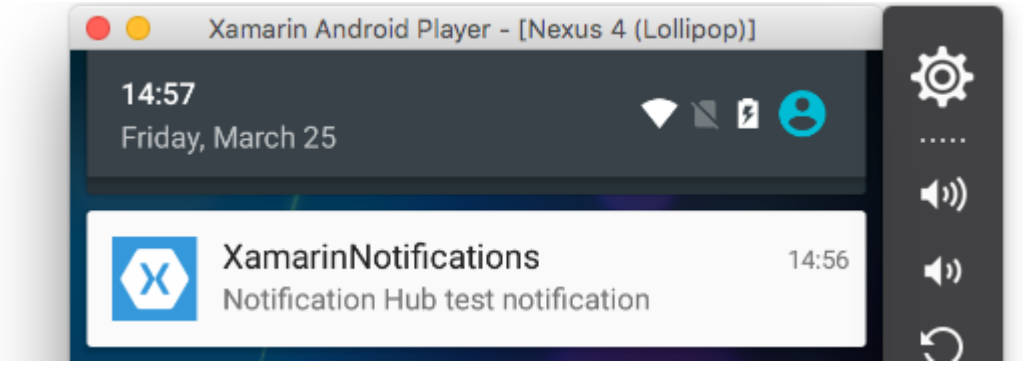
```
using Android.App;
using Android.Content;
using Gcm.Client;
using Java.Lang;
using System;
using WindowsAzure.Messaging;
using XamarinNotifications.Helpers;

// These attributes are to register the right permissions for our app concerning push messages
```

```
[assembly: Permission(Name = "com.versluisit.xamarinnotifications.permission.C2D_MESSAGE")]
[assembly: UsesPermission(Name =
"com.versluisit.xamarinnotifications.permission.C2D_MESSAGE")]
[assembly: UsesPermission(Name = "com.google.android.c2dm.permission.RECEIVE")]

//GET_ACCOUNTS is only needed for android versions 4.0.3 and below
[assembly: UsesPermission(Name = "android.permission.GET_ACCOUNTS")]
[assembly: UsesPermission(Name = "android.permission.INTERNET")]
[assembly: UsesPermission(Name = "android.permission.WAKE_LOCK")]

namespace XamarinNotifications.Droid.PlatformSpecifics
{
    // These attributes belong to the BroadcastReceiver, they register for the right intents
    [BroadcastReceiver(Permission = Constants.PERMISSION_GCM_INTENTS)]
    [IntentFilter(new[] { Constants.INTENT_FROM_GCM_MESSAGE },
    Categories = new[] { "com.versluisit.xamarinnotifications" })]
    [IntentFilter(new[] { Constants.INTENT_FROM_GCM_REGISTRATION_CALLBACK },
    Categories = new[] { "com.versluisit.xamarinnotifications" })]
    [IntentFilter(new[] { Constants.INTENT_FROM_GCM_LIBRARY_RETRY },
    Categories = new[] { "com.versluisit.xamarinnotifications" })]

    // This is the bradcast reciever
    public class NotificationsBroadcastReceiver : GcmBroadcastReceiverBase<PushHandlerService>
    {
        // TODO add your project number here
        public static string[] SenderIDs = { "96688------" };
    }

    [Service] // Don't forget this one! This tells Xamarin that this class is a Android
Service
    public class PushHandlerService : GcmServiceBase
    {
        // TODO add your own access key
        private string _connectionString =
ConnectionString.CreateUsingSharedAccessKeyWithListenAccess(
            new Java.Net.URI("sb://xamarinnotifications-ns.servicebus.windows.net/"), "<your
key here>");

        // TODO add your own hub name
        private string _hubName = "xamarinnotifications";

        public static string RegistrationID { get; private set; }

        public PushHandlerService() : base(NotificationsBroadcastReceiver.SenderIDs)
        {
        }

        // This is the entry point for when a notification is received
        protected override void OnMessage(Context context, Intent intent)
        {
            var title = "XamarinNotifications";

            if (intent.Extras.ContainsKey("title"))
                title = intent.Extras.GetString("title");

            var messageText = intent.Extras.GetString("message");

            if (!string.IsNullOrEmpty(messageText))
                CreateNotification(title, messageText);
        }
```

```
        // The method we use to compose our notification
        private void CreateNotification(string title, string desc)
        {
            // First we make sure our app will start when the notification is pressed
            const int pendingIntentId = 0;
            const int notificationId = 0;

            var startupIntent = new Intent(this, typeof(MainActivity));
            var stackBuilder = TaskStackBuilder.Create(this);

            stackBuilder.AddParentStack(Class.FromType(typeof(MainActivity)));
            stackBuilder.AddNextIntent(startupIntent);

            var pendingIntent =
                stackBuilder.GetPendingIntent(pendingIntentId, PendingIntentFlags.OneShot);

            // Here we start building our actual notification, this has some more
            // interesting customization options!
            var builder = new Notification.Builder(this)
                .SetContentIntent(pendingIntent)
                .SetContentTitle(title)
                .SetContentText(desc)
                .SetSmallIcon(Resource.Drawable.icon);

            // Build the notification
            var notification = builder.Build();
            notification.Flags = NotificationFlags.AutoCancel;

            // Get the notification manager
            var notificationManager =
                GetSystemService(NotificationService) as NotificationManager;

            // Publish the notification to the notification manager
            notificationManager.Notify(notificationId, notification);
        }

        // Whenever an error occurs in regard to push registering, this fires
        protected override void OnError(Context context, string errorId)
        {
            Console.Out.WriteLine(errorId);
        }

        // This handles the successful registration of our device to Google
        // We need to register with Azure here ourselves
        protected override void OnRegistered(Context context, string registrationId)
        {
            var hub = new NotificationHub(_hubName, _connectionString, context);

            Settings.DeviceToken = registrationId;

            // TODO set some tags here if you want and supply them to the Register method
            var tags = new string[] { };

            hub.Register(registrationId, tags);
        }

        // This handles when our device unregisters at Google
        // We need to unregister with Azure
        protected override void OnUnRegistered(Context context, string registrationId)
        {
            var hub = new NotificationHub(_hubName, _connectionString, context);
```

```
            hub.UnregisterAll(registrationId);
        }
    }
}
```

Примерное уведомление на Android выглядит так.

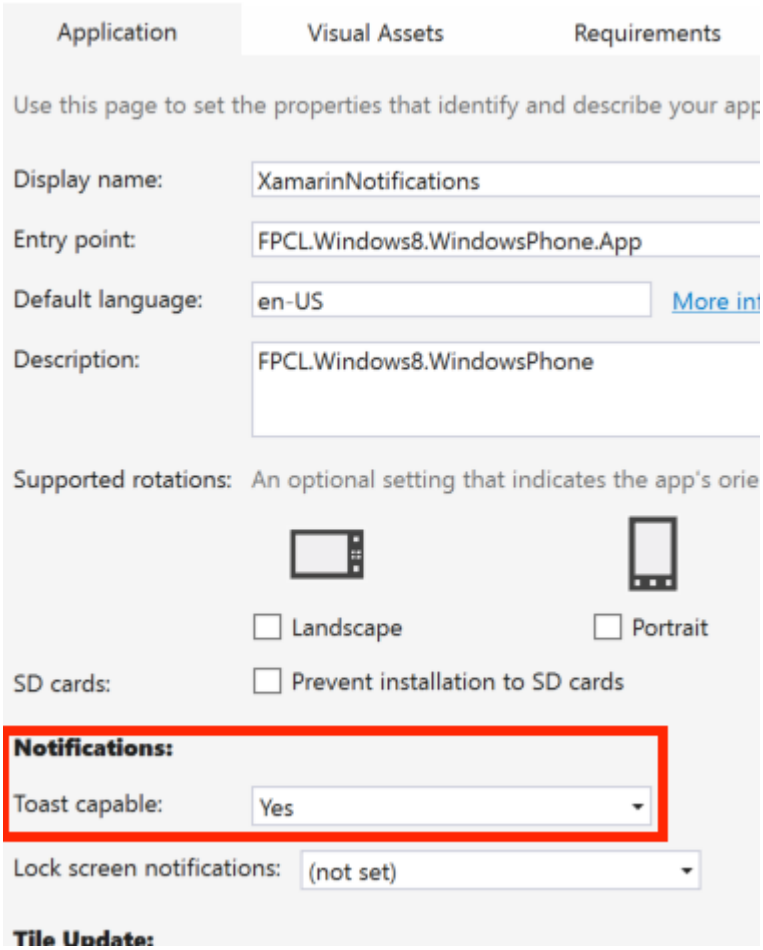


Push-уведомления для Windows Phone с Azure

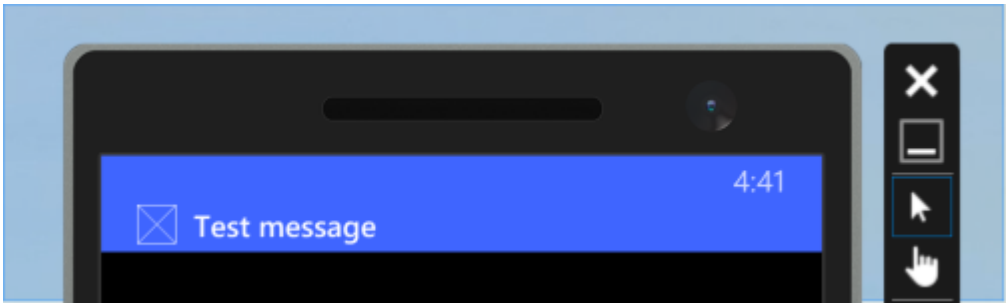
На Windows Phone нужно выполнить что-то вроде кода под ним, чтобы начать работу с push-уведомлениями. Это можно найти в файле `App.xaml.cs`

```
protected async override void OnLaunched(LaunchActivatedEventArgs e)
{
    var channel = await
PushNotificationChannelManager.CreatePushNotificationChannelForApplicationAsync();

    // TODO add connection string here
    var hub = new NotificationHub("XamarinNotifications", "<connection string with listen
access>");
    var result = await hub.RegisterNativeAsync(channel.Uri);

    // Displays the registration ID so you know it was successful
    if (result.RegistrationId != null)
    {
        Settings.DeviceToken = result.RegistrationId;
    }

    // The rest of the default code is here
}
```

Также не забудьте включить возможности в файле `Package.appxmanifest` .



Пример push-уведомления может выглядеть следующим образом:



Прочитайте Всплывающие уведомления онлайн: https://riptutorial.com/ru/xamarin-forms/topic/5042/всплывающие-уведомления

глава 10: Всплывающие уведомления

замечания

AWS Simple Notification Service Lingo:

Конечная точка. Конечной точкой может быть телефон, адрес электронной почты или что-то еще, это то, что AWS SNS может ударить с уведомлением

Тема - по существу группа, содержащая все ваши конечные точки

Подписаться - вы подписываете свой телефон / клиент для получения уведомлений

Lingo:

APNS - Служба уведомления Apple Push. Apple является единственным, кто может отправлять push-уведомления. Вот почему мы предоставляем нашему приложению соответствующий сертификат. Мы предоставляем AWS SNS сертификат, который Apple предоставляет нам для авторизации SNS для отправки уведомления APNS от нашего имени.

GCM - Google Cloud Messaging очень похож на APNS. Google является единственным, кто может напрямую отправлять push-уведомления. Поэтому мы сначала регистрируем наше приложение в GCM и передаем наш токен AWS SNS. SNS обрабатывает все сложные вещи, связанные с GCM и отправляя данные.

Examples

Пример iOS

1. Вам понадобится устройство разработки
2. Перейдите на свою учетную запись Apple Developer и создайте профиль подготовки с включенными Push Notifications
3. Вам потребуется какой-то способ оповестить ваш телефон (AWS, Azure ..etc). **Мы будем использовать AWS здесь**

```
public override bool FinishedLaunching(UIApplication app, NSDictionary options)
{
    global::Xamarin.Forms.Forms.Init();

   //after typical Xamarin.Forms Init Stuff

   //variable to set-up the style of notifications you want, iOS supports 3 types
```

```
   var pushSettings = UIUserNotificationSettings.GetSettingsForTypes(
              UIUserNotificationType.Alert |
              UIUserNotificationType.Badge |
              UIUserNotificationType.Sound,
              null );


        //both of these methods are in iOS, we have to override them and set them up
        //to allow push notifications

        app.RegisterUserNotificationSettings(pushSettings);  //pass the supported push
notifications settings to register app in settings page


}

public override async void RegisteredForRemoteNotifications(UIApplication application, NSData
token)
    {
        AmazonSimpleNotificationServiceClient snsClient = new
AmazonSimpleNotificationServiceClient("your AWS credentials here");

        // This contains the registered push notification token stored on the phone.
        var deviceToken = token.Description.Replace("<", "").Replace(">", "").Replace(" ",
"");

        if (!string.IsNullOrEmpty(deviceToken))
        {
            //register with SNS to create an endpoint ARN, this means AWS can message your
phone
            var response = await snsClient.CreatePlatformEndpointAsync(
            new CreatePlatformEndpointRequest
            {
                Token = deviceToken,
                PlatformApplicationArn = "yourARNwouldgohere" /* insert your platform
application ARN here */
            });

            var endpoint = response.EndpointArn;

            //AWS lets you create topics, so use subscribe your app to a topic, so you can
easily send out one push notification to all of your users
            var subscribeResponse = await snsClient.SubscribeAsync(new SubscribeRequest
            {
                TopicArn = "YourTopicARN here",
                Endpoint = endpoint,
                Protocol = "application"

            });

        }

    }
```

Прочитайте Всплывающие уведомления онлайн: https://riptutorial.com/ru/xamarin-forms/topic/5998/всплывающие-уведомления

глава 11: Доступ к собственным функциям с помощью DependencyService

замечания

Если вы не хотите, чтобы ваш код прерывался, когда реализация не была найдена, сначала проверьте `DependencyService` если она имеет доступную реализацию.

Вы можете сделать это с помощью простой проверки, если она не равна `null` .

```
var speaker = DependencyService.Get<ITextToSpeech>();

if (speaker != null)
{
    speaker.Speak("Ready for action!");
}
```

или, если ваша среда IDE поддерживает C # 6, с оператором с нулевым условием:

```
var speaker = DependencyService.Get<ITextToSpeech>();

speaker?.Speak("Ready for action!");
```

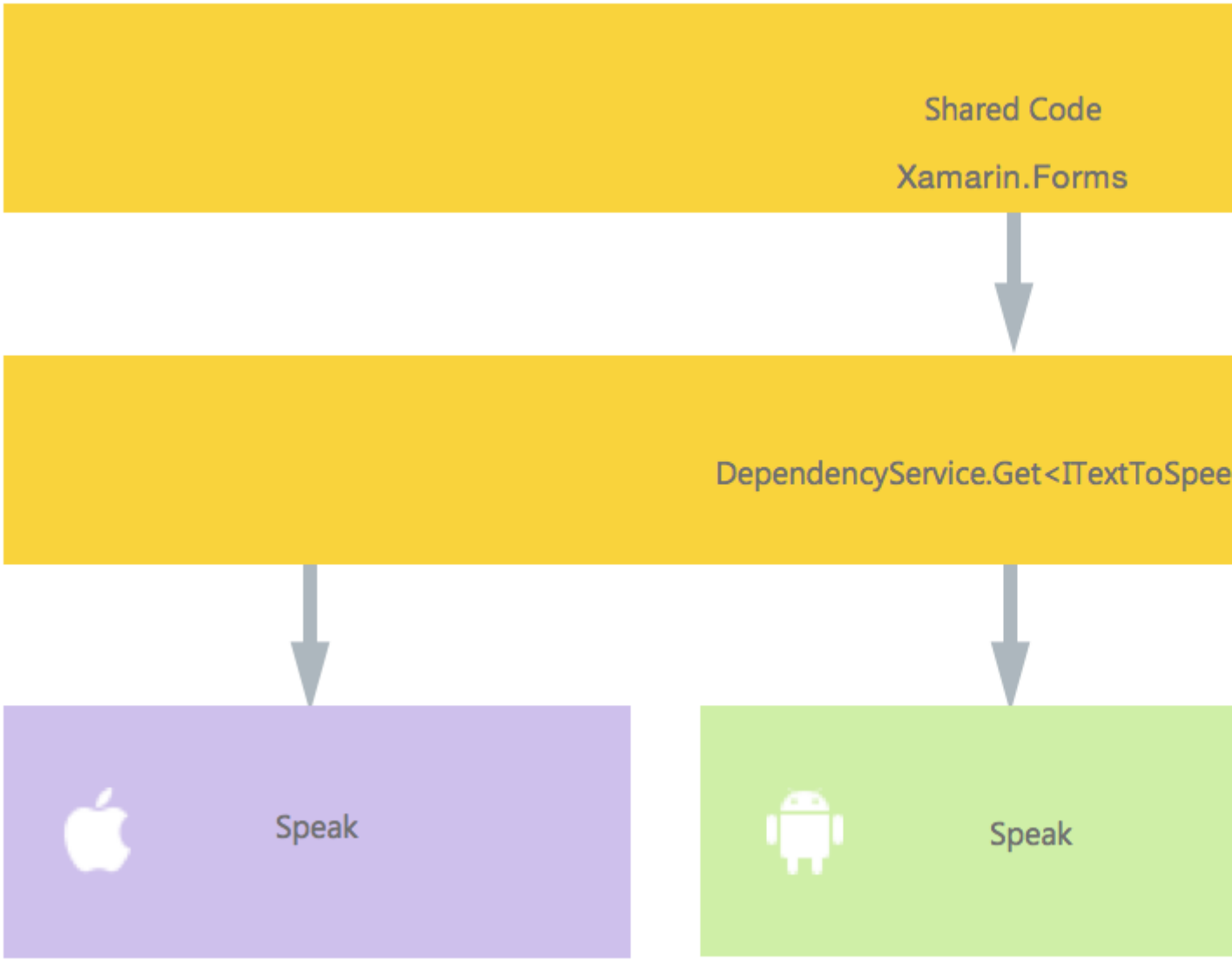
Если вы этого не сделаете, и реализация не будет реализована во время выполнения, ваш код будет генерировать исключение.

Examples

Реализация текста в речь

Хорошим примером функции, запрашивающей специфический для платформы код, является то, что вы хотите реализовать текст в речь (tts). В этом примере предполагается, что вы работаете с общим кодом в библиотеке PCL.

Схематический обзор нашего решения будет выглядеть как снизу.



В нашем общем коде мы определяем интерфейс, который зарегистрирован в `DependencyService` . Здесь мы будем делать наши призывы. Определите интерфейс, как внизу.

```
public interface ITextToSpeech
{
    void Speak (string text);
}
```

Теперь в каждой конкретной платформе нам необходимо создать реализацию этого интерфейса. Начнем с реализации iOS.

Внедрение iOS

```
using AVFoundation;
```

```
public class TextToSpeechImplementation : ITextToSpeech
{
    public TextToSpeechImplementation () {}

    public void Speak (string text)
    {
        var speechSynthesizer = new AVSpeechSynthesizer ();

        var speechUtterance = new AVSpeechUtterance (text) {
            Rate = AVSpeechUtterance.MaximumSpeechRate/4,
            Voice = AVSpeechSynthesisVoice.FromLanguage ("en-US"),
            Volume = 0.5f,
            PitchMultiplier = 1.0f
        };

        speechSynthesizer.SpeakUtterance (speechUtterance);
    }
}
```

В приведенном выше примере кода вы заметите, что для iOS существует определенный код. Подобно типам `AVSpeechSynthesizer` . Они не будут работать в общем коде.

Чтобы зарегистрировать эту реализацию с помощью Xamarin `DependencyService` добавьте этот атрибут над объявлением пространства имен.

```
using AVFoundation;
using DependencyServiceSample.iOS;//enables registration outside of namespace

[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechImplementation))]
namespace DependencyServiceSample.iOS {
    public class TextToSpeechImplementation : ITextToSpeech
//... Rest of code
```

Теперь, когда вы делаете такой вызов в своем общем коде, вводится правильная реализация для платформы, на которой выполняется ваше приложение.

`DependencyService.Get<ITextToSpeech>()` . Подробнее об этом позже.

Реализация Android

Реализация этого кода для Android будет выглядеть как внизу.

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;

public class TextToSpeechImplementation : Java.Lang.Object, ITextToSpeech,
TextToSpeech.IOnInitListener
{
    TextToSpeech speaker;
    string toSpeak;
```

```
    public TextToSpeechImplementation () {}

    public void Speak (string text)
    {
        var ctx = Forms.Context; // useful for many Android SDK features
        toSpeak = text;
        if (speaker == null) {
            speaker = new TextToSpeech (ctx, this);
        } else {
            var p = new Dictionary<string,string> ();
            speaker.Speak (toSpeak, QueueMode.Flush, p);
        }
    }

    #region IOnInitListener implementation
    public void OnInit (OperationResult status)
    {
        if (status.Equals (OperationResult.Success)) {
            var p = new Dictionary<string,string> ();
            speaker.Speak (toSpeak, QueueMode.Flush, p);
        }
    }
    #endregion
}
```

Снова не забудьте зарегистрировать его в `DependencyService` .

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;


[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechImplementation))]
namespace DependencyServiceSample.Droid{
    //... Rest of code
```

Реализация Windows Phone

Наконец, для Windows Phone этот код можно использовать.

```
public class TextToSpeechImplementation : ITextToSpeech
{
    public TextToSpeechImplementation() {}

    public async void Speak(string text)
    {
        MediaElement mediaElement = new MediaElement();

        var synth = new Windows.Media.SpeechSynthesis.SpeechSynthesizer();

        SpeechSynthesisStream stream = await synth.SynthesizeTextToStreamAsync("Hello World");

        mediaElement.SetSource(stream, stream.ContentType);
        mediaElement.Play();
        await synth.SynthesizeTextToStreamAsync(text);
```

```
    }
}
```

И еще раз не забудьте зарегистрировать его.

```
using Windows.Media.SpeechSynthesis;
using Windows.UI.Xaml.Controls;
using DependencyServiceSample.WinPhone;//enables registration outside of namespace

[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechImplementation))]
namespace DependencyServiceSample.WinPhone{
    //... Rest of code
```

Внедрение в общий код

Теперь все на месте, чтобы заставить его работать! Наконец, в вашем общем коде вы можете теперь вызвать эту функцию, используя интерфейс. Во время выполнения будет введена реализация, которая соответствует текущей платформе, на которой она запущена.

В этом коде вы увидите страницу, которая может быть в проекте Xamarin Forms. Он создает кнопку, которая вызывает метод `Speak()` , используя `DependencyService` .

```
public MainPage ()
{
    var speak = new Button {
        Text = "Hello, Forms !",
        VerticalOptions = LayoutOptions.CenterAndExpand,
        HorizontalOptions = LayoutOptions.CenterAndExpand,
    };
    speak.Clicked += (sender, e) => {
        DependencyService.Get<ITextToSpeech>().Speak("Hello from Xamarin Forms");
    };
    Content = speak;
}
```

Результатом будет то, что при запуске приложения и нажатии кнопки будет передан текст.

Все это без необходимости делать такие вещи, как подсказки компилятора и т. Д. Теперь у вас есть единый способ доступа к функциональности платформы с помощью независимого от платформы кода.

Получение номеров приложений и устройств OS - Android и iOS - PCL

В приведенном ниже примере будет собрано номер версии операционной системы устройства и версия приложения (которая определена в свойствах каждого проекта), которая вводится в **имя версии** на Android и **версию** iOS.

Сначала создайте интерфейс в своем проекте PCL:

```
public interface INativeHelper {
    /// <summary>
    /// On iOS, gets the <c>CFBundleVersion</c> number and on Android, gets the
<c>PackageInfo</c>'s <c>VersionName</c>, both of which are specified in their respective
project properties.
    /// </summary>
    /// <returns><c>string</c>, containing the build number.</returns>
    string GetAppVersion();

    /// <summary>
    /// On iOS, gets the <c>UIDevice.CurrentDevice.SystemVersion</c> number and on Android,
gets the <c>Build.VERSION.Release</c>.
    /// </summary>
    /// <returns><c>string</c>, containing the OS version number.</returns>
    string GetOsVersion();
}
```

Теперь мы реализуем интерфейс в проектах Android и iOS.

Android:

```
[assembly: Dependency(typeof(NativeHelper_Android))]

namespace YourNamespace.Droid{
    public class NativeHelper_Android : INativeHelper {

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetAppVersion() {
            Context context = Forms.Context;
            return context.PackageManager.GetPackageInfo(context.PackageName, 0).VersionName;
        }

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetOsVersion() { return Build.VERSION.Release; }
    }
}
```

IOS:

```
[assembly: Dependency(typeof(NativeHelper_iOS))]

namespace YourNamespace.iOS {
    public class NativeHelper_iOS : INativeHelper {

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetAppVersion() { return
Foundation.NSBundle.MainBundle.InfoDictionary[new
Foundation.NSString("CFBundleVersion")].ToString(); }
```

```
        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetOsVersion() { return UIDevice.CurrentDevice.SystemVersion; }
    }
}
```

Теперь использовать код в методе:

```
public string GetOsAndAppVersion {
    INativeHelper helper = DependencyService.Get<INativeHelper>();

    if(helper != null) {
        string osVersion  = helper.GetOsVersion();
        string appVersion = helper.GetBuildNumber()
    }
}
```

Прочитайте Доступ к собственным функциям с помощью DependencyService онлайн: https://riptutorial.com/ru/xamarin-forms/topic/2409/доступ-к-собственным-функциям-с-помощью-dependencyservice

глава 12: жесты

Examples

Сделать снимок с помощью добавления TapGestureRecognizer

В Xamarin.Forms есть пара распознавателей по умолчанию, один из которых - `TapGestureRecognizer` .

Вы можете добавить их практически к любому визуальному элементу. Посмотрите на простую реализацию, которая привязывается к `Image` . Вот как это сделать в коде.

```
var tappedCommand = new Command(() =>
{
    //handle the tap
});

var tapGestureRecognizer = new TapGestureRecognizer { Command = tappedCommand };
image.GestureRecognizers.Add(tapGestureRecognizer);
```

Или в XAML:

```
<Image Source="tapped.jpg">
    <Image.GestureRecognizers>
        <TapGestureRecognizer
                Command="{Binding TappedCommand}"
                NumberOfTapsRequired="2" />
  </Image.GestureRecognizers>
</Image>
```

Здесь команда задается с помощью привязки данных. Как вы можете видеть, вы также можете установить `NumberOfTapsRequired` чтобы включить его для большего количества кранов, прежде чем он примет меры. Значение по умолчанию - 1 крана.

Другими жестами являются Pinch и Pan.

Увеличьте изображение с помощью жестов Pinch

Чтобы сделать `Image` (или любой другой визуальный элемент) масштабируемым, мы должны добавить к нему `PinchGestureRecognizer` . Вот как это сделать в коде:

```
var pinchGesture = new PinchGestureRecognizer();
pinchGesture.PinchUpdated += (s, e) => {
// Handle the pinch
};

image.GestureRecognizers.Add(pinchGesture);
```

Но это также можно сделать из XAML:

```
<Image Source="waterfront.jpg">
  <Image.GestureRecognizers>
    <PinchGestureRecognizer PinchUpdated="OnPinchUpdated" />
  </Image.GestureRecognizers>
</Image>
```

В сопроводительном обработчике событий вы должны указать код, чтобы увеличить изображение. Конечно, другие применения могут быть реализованы.

```
void OnPinchUpdated (object sender, PinchGestureUpdatedEventArgs e)
{
    // ... code here
}
```

Другими жестами являются Tap и Pan.

Показать весь увеличенный контент изображения с помощью PanGestureRecognizer

Когда у вас есть увеличенное `Image` (или другое содержимое), вы можете перетащить `Image` чтобы отобразить весь его контент в увеличенном состоянии.

Это может быть достигнуто путем внедрения PanGestureRecognizer. Из кода это выглядит так:

```
var panGesture = new PanGestureRecognizer();
panGesture.PanUpdated += (s, e) => {
  // Handle the pan
};

image.GestureRecognizers.Add(panGesture);
```

Это также можно сделать из XAML:

```
<Image Source="MonoMonkey.jpg">
  <Image.GestureRecognizers>
    <PanGestureRecognizer PanUpdated="OnPanUpdated" />
  </Image.GestureRecognizers>
</Image>
```

В событии с кодом вы можете теперь обрабатывать панорамирование. Используйте эту подпись метода для ее обработки:

```
void OnPanUpdated (object sender, PanUpdatedEventArgs e)
{
  // Handle the pan
}
```

Поместите контакт, где пользователь коснулся экрана с помощью MR.Gestures

Xamarins, встроенные в распознаватели жестов, обеспечивают только очень базовую сенсорную обработку. Например, нет способа получить положение трогательного пальца. MR.Gestures - это компонент, который добавляет 14 различных событий обработки касания. Положение трогательных пальцев является частью `EventArgs` переданной всем событиям MR.Gestures.

Если вы хотите разместить булавку в любом месте экрана, самым простым способом является использование `MR.Gestures.AbsoluteLayout` который обрабатывает событие `Tapping` .

```
<mr:AbsoluteLayout x:Name="MainLayout" Tapping="OnTapping">
    ...
</mr:AbsoluteLayout>
```

Как вы можете видеть, `Tapping="OnTapping"` также больше напоминает .NET, чем синтаксис Xamarins с вложенными `GestureRecognizers` . Этот синтаксис был скопирован из iOS, и он немного пахнет для разработчиков .NET.

В вашем коде позади вы можете добавить обработчик `OnTapping` следующим образом:

```
private void OnTapping(object sender, MR.Gestures.TapEventArgs e)
{
    if (e.Touches?.Length > 0)
    {
        Point touch = e.Touches[0];
        var image = new Image() { Source = "pin" };
        MainLayout.Children.Add(image, touch);
    }
}
```

Вместо события `Tapping` вы также можете использовать `TappingCommand` и привязываться к ViewModel, но это осложнит ситуацию в этом простом примере.

Больше образцов для MR.Gestures можно найти в приложении GestureSample на GitHub и на веб-сайте MR.Gestures . Они также показывают, как использовать все другие события касания с обработчиками событий, командами, MVVM, ...

Прочитайте жесты онлайн: https://riptutorial.com/ru/xamarin-forms/topic/3914/жесты

глава 13: Использование ListViews

Вступление

В этой документации описывается, как использовать различные компоненты Xamarin Forms ListView

Examples

Потяните, чтобы обновить в XAML и код за

Чтобы включить Pull to Refresh в `ListView` в Xamarin, сначала нужно указать, что он включен `PullToRefresh` а затем указать имя команды, которую вы хотите вызывать при вытаскивании `ListView` :

```
<ListView x:Name="itemListView" IsPullToRefreshEnabled="True" RefreshCommand="Refresh">
```

То же самое можно сделать и в коде:

```
itemListView.IsPullToRefreshEnabled = true;
itemListView.RefreshCommand = Refresh;
```

Затем вы должны указать, что команда `Refresh` делает в вашем коде:

```
public ICommand Refresh
{
    get
    {
        itemListView.IsRefreshing = true; //This turns on the activity
                                          //Indicator for the ListView
        //Then add your code to execute when the ListView is pulled
        itemListView.IsRefreshing = false;
    }
}
```

Прочитайте Использование ListViews онлайн: https://riptutorial.com/ru/xamarin-forms/topic/9487/использование-listviews

глава 14: Клетки Xamarin.Forms

Examples

EntryCell

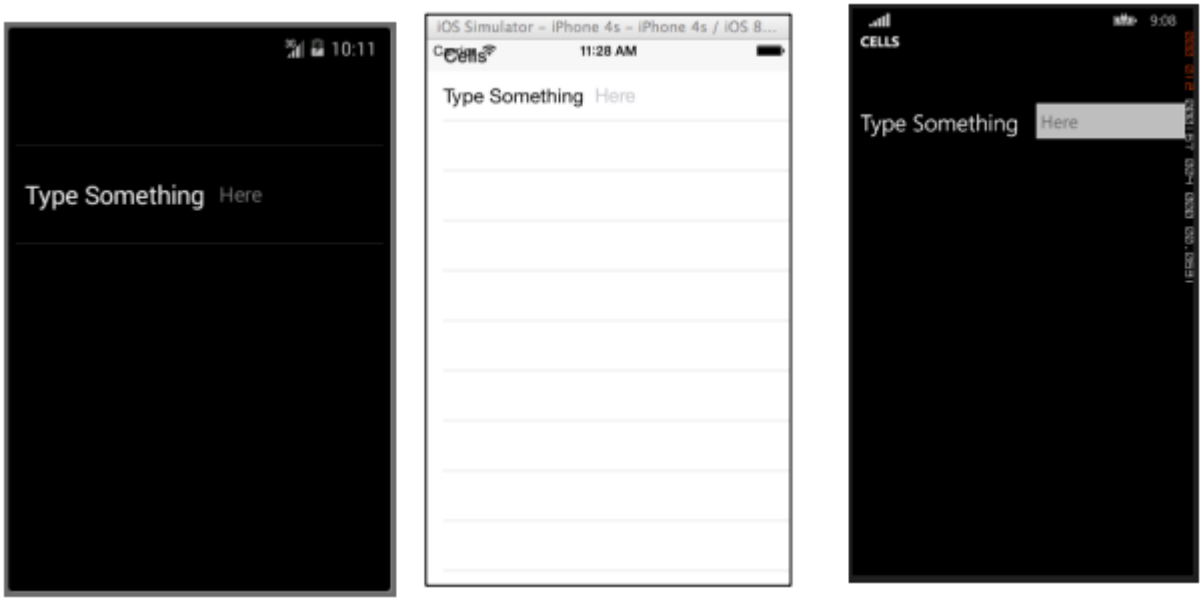
EntryCell - это Cell, который объединяет возможности Label и Entry. EntryCell может быть полезен в сценариях при создании некоторых функций в вашем приложении для сбора данных от пользователя. Они легко могут быть помещены в TableView и рассматриваться как простая форма.

XAML

```
<EntryCell Label="Type Something"
Placeholder="Here"/>
```

Код

```
var entryCell = new EntryCell {
Label = "Type Something",
Placeholder = "Here"
};
```



SwitchCell

SwitchCell - это Cell, который объединяет возможности Label и переключателя on-off. Коммутатор SwitchCell может быть полезен для включения и выключения функций или даже настроек пользователя или параметров конфигурации.

XAML

```
<SwitchCell Text="Switch It Up!" />
```

Код

```
var switchCell = new SwitchCell {
Text = "Switch It Up!"
};
```

TextCell

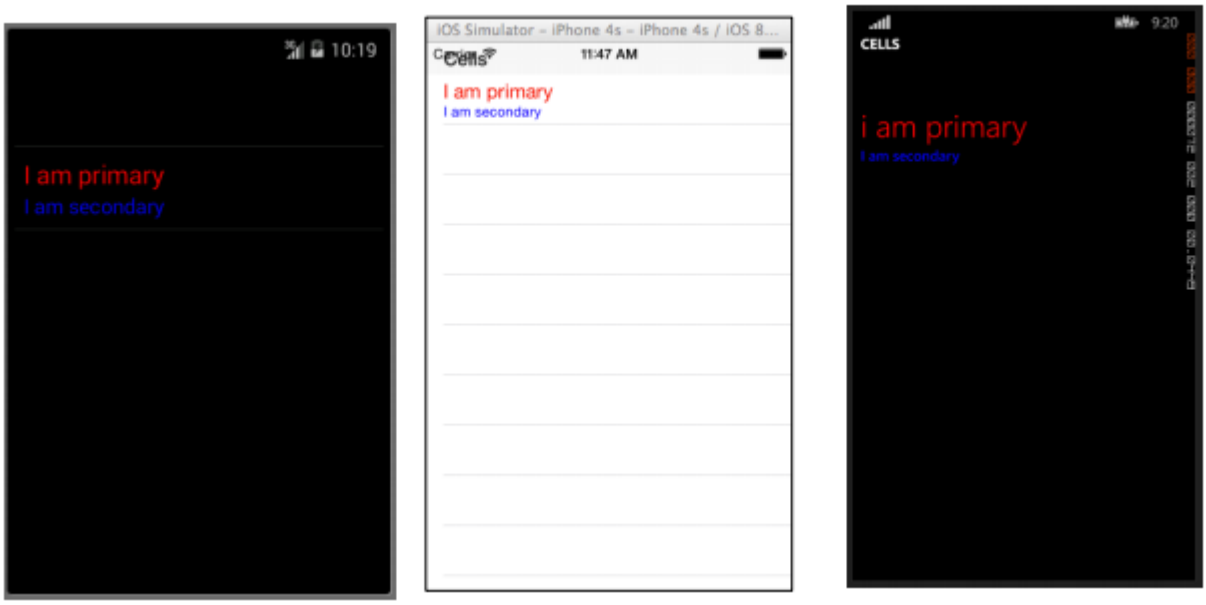
TextCell - это ячейка, которая имеет две отдельные текстовые области для отображения данных. TextCell обычно используется для информационных целей в элементах управления TableView и ListView. Две области текста выровнены по вертикали, чтобы максимизировать пространство внутри Ячейки. Этот тип ячеек также обычно используется для отображения иерархических данных, поэтому, когда пользователь удаляет эту ячейку, он перейдет на другую страницу.

XAML

```
<TextCell Text="I am primary"
TextColor="Red"
Detail="I am secondary"
DetailColor="Blue"/>
```

Код

```
var textCell = new TextCell {
Text = "I am primary",
TextColor = Color.Red,
Detail = "I am secondary",
DetailColor = Color.Blue
};
```



ImageCell

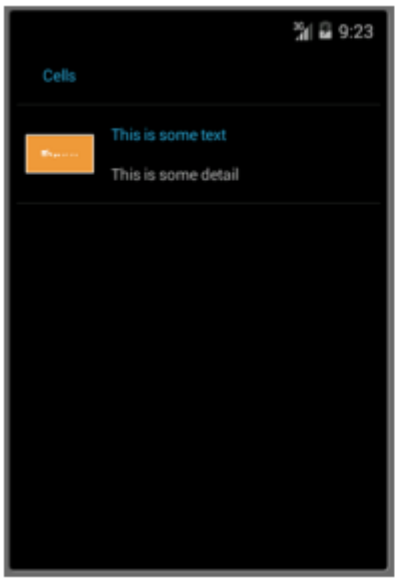
ImageCell - это именно то, на что это похоже. Это простая ячейка, содержащая только изображение. Этот элемент управления работает очень точно с обычным управлением изображения, но с гораздо меньшим количеством звонков и свистов.

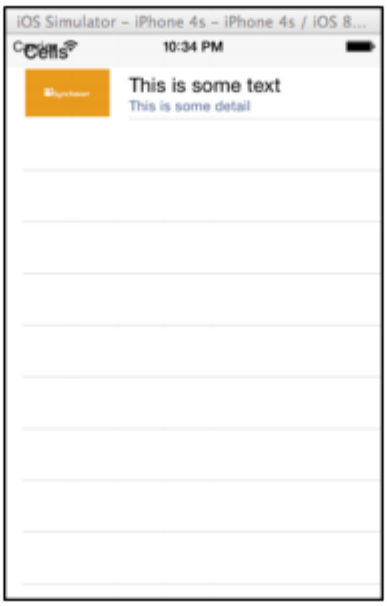
XAML

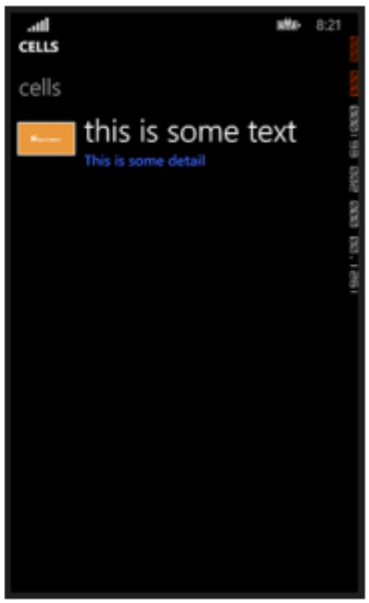
```
<ImageCell ImageSource="http://d2g29cya9iq7ip.cloudfront.net/content/imag
es/company/aboutus-video-bg.png?v=25072014072745")),
 Text="This is some text"
 Detail="This is some detail" />
```

Код

```
var imageCell = new ImageCell {
ImageSource = ImageSource.FromUri(new Uri("http://d2g29cya9iq7ip.clou
109
dfront.net/content/images/company/aboutus-videobg.png?v=25072014072745")),
 Text = "This is some text",
 Detail = "This is some detail"
};
```







ViewCell

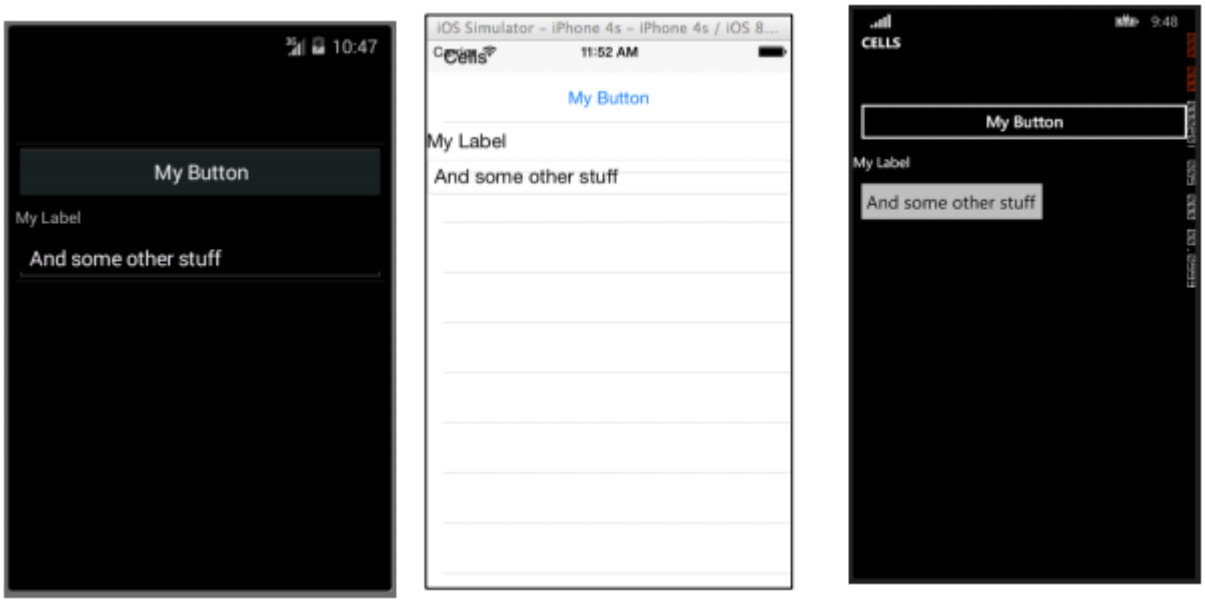
Вы можете рассматривать ViewCell как чистый лист. Это ваш личный холст для создания ячейки, которая выглядит точно так, как вы хотите. Вы даже можете составить его из экземпляров нескольких других объектов View вместе с элементами управления макета. Вы ограничены только вашим воображением. И, возможно, размер экрана.

XAML

```
<ViewCell>
<ViewCell.View>
<StackLayout>
<Button Text="My Button"/>

<Label Text="My Label"/>
<Entry Text="And some other stuff"/>
</StackLayout>
</ViewCell.View>
</ViewCell>
```

Код

```
var button = new Button { Text = "My Button" };
var label = new Label { Text = "My Label" };
var entry = new Entry { Text ="And some other stuff" };
var viewCell = new ViewCell {
View = new StackLayout {
Children = { button, label, entry }
}
};
```



Прочитайте Клетки Xamarin.Forms онлайн: https://riptutorial.com/ru/xamarin-forms/topic/7370/клетки-xamarin-forms

глава 15: Кэширование

Examples

Кэширование с использованием Akavache

О компании Akavache

Akavache - невероятно полезная библиотека, обеспечивающая функциональность кэширования ваших данных. Akavache обеспечивает интерфейс хранения ключевого значения и работает на вершине SQLite3. Вам не нужно синхронизировать вашу схему, поскольку это фактически не-SQL-решение, которое делает его идеальным для большинства мобильных приложений, особенно если вам нужно, чтобы ваше приложение обновлялось часто без потери данных.

Рекомендации для Xamarin

Akavache определенно является лучшей библиотекой кэширования для приложения Xamarin, если вам не нужно работать с сильно относительными данными, бинарными или действительно большими объемами данных. Используйте Akavache в следующих случаях:

- Вам необходимо, чтобы ваше приложение кэшировало данные за определенный период времени (вы можете настроить тайм-аут истечения срока действия для каждого сохраняемого объекта;
- Вы хотите, чтобы ваше приложение работало в автономном режиме;
- Трудно определить и заморозить схему ваших данных. Например, у вас есть списки, содержащие разные типизированные объекты;
- Достаточно для вас иметь простой доступ к данным с ключом и вам не нужно создавать сложные запросы.

Akavache не является «серебряной пулей» для хранения данных, поэтому дважды подумайте об использовании в следующих случаях:

- У ваших объектов данных много отношений между собой;
- Вам не нужно, чтобы ваше приложение работало в автономном режиме;
- У вас есть огромное количество данных для локального сохранения;
- Вам необходимо перенести данные с версии на версию;
- Вам необходимо выполнить сложные запросы, типичные для SQL, такие как группировка, прогнозы и т. Д.

Фактически вы можете вручную перенести свои данные, просто прочитав их и обновив с помощью обновленных полей.

Простой пример

Взаимодействие с Akavache осуществляется в основном с помощью объекта `BlobCache` .

Большинство методов Akavache возвращают реактивные наблюдаемые, но вы также можете просто ждать их благодаря методам расширения.

```
using System.Reactive.Linq;   // IMPORTANT - this makes await work!

// Make sure you set the application name before doing any inserts or gets
BlobCache.ApplicationName = "AkavacheExperiment";

var myToaster = new Toaster();
await BlobCache.UserAccount.InsertObject("toaster", myToaster);

//
// ...later, in another part of town...
//

// Using async/await
var toaster = await BlobCache.UserAccount.GetObject<Toaster>("toaster");

// or without async/await
Toaster toaster;

BlobCache.UserAccount.GetObject<Toaster>("toaster")
    .Subscribe(x => toaster = x, ex => Console.WriteLine("No Key!"));
```

Обработка ошибок

```
Toaster toaster;

try {
    toaster = await BlobCache.UserAccount.GetObjectAsync("toaster");
} catch (KeyNotFoundException ex) {
    toaster = new Toaster();
}

// Or without async/await:
toaster = await BlobCache.UserAccount.GetObjectAsync<Toaster>("toaster")
    .Catch(Observable.Return(new Toaster()));
```

Прочитайте Кэширование онлайн: https://riptutorial.com/ru/xamarin-forms/topic/3644/кэширование

глава 16: Навигация в Xamarin.Forms

Examples

Навигационный поток

```
using System;
using Xamarin.Forms;

namespace NavigationApp
{
    public class App : Application
    {
        public App()
        {
            MainPage = new NavigationPage(new FirstPage());
        }
    }

    public class FirstPage : ContentPage
    {
        Label FirstPageLabel { get; set; } = new Label();

        Button FirstPageButton { get; set; } = new Button();

        public FirstPage()
        {
            Title = "First page";

            FirstPageLabel.Text = "This is the first page";
            FirstPageButton.Text = "Navigate to the second page";
            FirstPageButton.Clicked += OnFirstPageButtonClicked;

            var content = new StackLayout();
            content.Children.Add(FirstPageLabel);
            content.Children.Add(FirstPageButton);

            Content = content;
        }

        async void OnFirstPageButtonClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SecondPage(), true);
        }
    }

    public class SecondPage : ContentPage
    {
        Label SecondPageLabel { get; set; } = new Label();

        public SecondPage()
        {
            Title = "Second page";

            SecondPageLabel.Text = "This is the second page";
```

```
            Content = SecondPageLabel;
        }
    }
}
```

Поток NavigationPage с помощью XAML

Файл App.xaml.cs (файл App.xaml по умолчанию, поэтому пропущен)

```
using Xamrin.Forms

namespace NavigationApp
{
    public partial class App : Application
    {
        public static INavigation GlobalNavigation { get; private set; }

        public App()
        {
            InitializeComponent();
            var rootPage = new NavigationPage(new FirstPage());

            GlobalNavigation = rootPage.Navigation;

            MainPage = rootPage;
        }
    }
}
```

Файл FirstPage.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="NavigationApp.FirstPage"
    Title="First page">
    <ContentPage.Content>
        <StackLayout>
            <Label
                Text="This is the first page" />
            <Button
                Text="Click to navigate to a new page"
                Clicked="GoToSecondPageButtonClicked"/>
            <Button
                Text="Click to open the new page as modal"
                Clicked="OpenGlobalModalPageButtonClicked"/>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

В некоторых случаях вам нужно открыть новую страницу не в текущей навигации, а в глобальной. Например, если ваша текущая страница содержит нижнее меню, она будет видна при нажатии новой страницы в текущей навигации. Если вам нужна страница, которая будет открыта по всему видимому контенту, скрывающему нижнее меню и

содержимое текущей текущей страницы, вам нужно переместить новую страницу в качестве модальной в глобальную навигацию. См. `App.GlobalNavigation` и пример ниже.

Файл FirstPage.xaml.cs

```
using System;
using Xamarin.Forms;

namespace NavigationApp
{
    public partial class FirstPage : ContentPage
    {
        public FirstPage()
        {
            InitializeComponent();
        }

        async void GoToSecondPageButtonClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SecondPage(), true);
        }

        async void OpenGlobalModalPageButtonClicked(object sender, EventArgs e)
        {
            await App.GlobalNavigation.PushModalAsync(new SecondPage(), true);
        }
    }
}
```

Файл SecondPage.xaml (файл xaml.cs по умолчанию, поэтому пропущен)

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="NavigationApp.SecondPage"
    Title="Second page">
    <ContentPage.Content>
        <Label
            Text="This is the second page" />
    </ContentPage.Content>
</ContentPage>
```

Иерархическая навигация с XAML

По умолчанию шаблон навигации работает как стек страниц, вызывая новейшие страницы на предыдущих страницах. Для этого вам нужно будет использовать объект NavigationPage .

Нажатие новых страниц

```
...
public class App : Application
```

```
{
    public App()
    {
        MainPage = new NavigationPage(new Page1());
    }
}
...
```

Page1.xaml

```
...
<ContentPage.Content>
    <StackLayout>
        <Label Text="Page 1" />
        <Button Text="Go to page 2" Clicked="GoToNextPage" />
    </StackLayout>
</ContentPage.Content>
...
```

Page1.xaml.cs

```
...
public partial class Page1 : ContentPage
{
    public Page1()
    {
        InitializeComponent();
    }

    protected async void GoToNextPage(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new Page2());
    }
}
...
```

Page2.xaml

```
...
<ContentPage.Content>
    <StackLayout>
        <Label Text="Page 2" />
        <Button Text="Go to Page 3" Clicked="GoToNextPage" />
    </StackLayout>
</ContentPage.Content>
...
```

Page2.xaml.cs

```
...
public partial class Page2 : ContentPage
{
    public Page2()
    {
```

```
        InitializeComponent();
    }

    protected async void GoToNextPage(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new Page3());
    }
}
...
```

Поппинг страниц

Обычно пользователь использует кнопку «Назад» для возврата страниц, но иногда вам нужно управлять этим программно, поэтому вам нужно вызвать метод **NavigationPage.PopAsync (),** чтобы вернуться на предыдущую страницу или **NavigationPage.PopToRootAsync (),** чтобы вернуться в начале, такие как ...

Page3.xaml

```
...
<ContentPage.Content>
    <StackLayout>
        <Label Text="Page 3" />
        <Button Text="Go to previous page" Clicked="GoToPreviousPage" />
        <Button Text="Go to beginning" Clicked="GoToStartPage" />
    </StackLayout>
</ContentPage.Content>
...
```

Page3.xaml.cs

```
...
public partial class Page3 : ContentPage
{
    public Page3()
    {
        InitializeComponent();
    }

    protected async void GoToPreviousPage(object sender, EventArgs e)
    {
        await Navigation.PopAsync();
    }

    protected async void GoToStartPage(object sender, EventArgs e)
    {
        await Navigation.PopToRootAsync();
    }
}
...
```

Модальная навигация с XAML

Модальные страницы могут быть созданы тремя способами:

- Из объекта **NavigationPage** для полноэкранных страниц
- Для оповещений и уведомлений
- Для ActionSheets, которые являются всплывающими меню

Полноэкранные модалы

```
...
// to open
await Navigation.PushModalAsync(new ModalPage());
// to close
await Navigation.PopModalAsync();
...
```

Оповещения / подтверждения и уведомления

```
...
// alert
await DisplayAlert("Alert title", "Alert text", "Ok button text");
// confirmation
var booleanAnswer = await DisplayAlert("Confirm?", "Confirmation text", "Yes", "No");
...
```

ActionSheets

```
...
var selectedOption = await DisplayActionSheet("Options", "Cancel", "Destroy", "Option 1",
"Option 2", "Option 3");
...
```

Основная страница корневой страницы

```
public class App : Application
{
    internal static NavigationPage NavPage;

    public App ()
    {
        // The root page of your application
        MainPage = new RootPage();
    }
}
public class RootPage : MasterDetailPage
{
    public RootPage()
    {
        var menuPage = new MenuPage();
        menuPage.Menu.ItemSelected += (sender, e) => NavigateTo(e.SelectedItem as MenuItem);
        Master = menuPage;
```

```
        App.NavPage = new NavigationPage(new HomePage());
        Detail = App.NavPage;
    }
    protected override async void OnAppearing()
    {
        base.OnAppearing();
    }
    void NavigateTo(MenuItem menuItem)
    {
        Page displayPage = (Page)Activator.CreateInstance(menuItem.TargetType);
        Detail = new NavigationPage(displayPage);
        IsPresented = false;
    }
}
```

Навигация по основным деталям

В приведенном ниже коде показано, как выполнять асинхронную навигацию, когда приложение находится в контексте MasterDetailPage.

```
public async Task NavigateMasterDetail(Page page)
{
        if (page == null)
            {
                return;
            }

        var masterDetail = App.Current.MainPage as MasterDetailPage;

        if (masterDetail == null || masterDetail.Detail == null)
            return;

        var navigationPage = masterDetail.Detail as NavigationPage;
        if (navigationPage == null)
        {
            masterDetail.Detail = new NavigationPage(page);
            masterDetail.IsPresented = false;
            return;
        }

        await navigationPage.Navigation.PushAsync(page);


navigationPage.Navigation.RemovePage(navigationPage.Navigation.NavigationStack[navigationPage.Navigatio
- 2]);
        masterDetail.IsPresented = false;
    }
```

Прочитайте Навигация в Xamarin.Forms онлайн: https://riptutorial.com/ru/xamarin-forms/topic/1571/навигация-в-xamarin-forms

глава 17: Навигация в Xamarin.Forms

замечания

Навигация по Xamarin.Forms основан на двух основных навигационных шаблонах: иерархическом и модальном.

Иерархический шаблон позволяет пользователю перемещаться вниз в стек страниц и возвращать нажатие кнопки «назад» / «вверх».

Модальный шаблон - это страница прерывания, требующая определенного действия от пользователя, но обычно ее можно отменить, нажав кнопку отмены. Некоторые примеры - уведомления, предупреждения, диалоговые окна и страницы регистрации / издания.

Examples

Использование модели Inavigation из вида

Первый шаг - создать навигационный интерфейс, который мы будем использовать на модели представления:

```
public interface IViewNavigationService
{
    void Initialize(INavigation navigation, SuperMapper navigationMapper);
    Task NavigateToAsync(object navigationSource, object parameter = null);
    Task GoBackAsync();
}
```

В методе `Initialize` я использую свой настраиваемый mapper, где я храню коллекцию типов страниц с соответствующими ключами.

```
public class SuperMapper
{
    private readonly ConcurrentDictionary<Type, object> _typeToAssociateDictionary = new
ConcurrentDictionary<Type, object>();

    private readonly ConcurrentDictionary<object, Type> _associateToType = new
ConcurrentDictionary<object, Type>();

    public void AddMapping(Type type, object associatedSource)
    {
        _typeToAssociateDictionary.TryAdd(type, associatedSource);
        _associateToType.TryAdd(associatedSource, type);
    }

    public Type GetTypeSource(object associatedSource)
    {
        Type typeSource;
        _associateToType.TryGetValue(associatedSource, out typeSource);
```

```
        return typeSource;
    }

    public object GetAssociatedSource(Type typeSource)
    {
        object associatedSource;
        _typeToAssociateDictionary.TryGetValue(typeSource, out associatedSource);

        return associatedSource;
    }
}
```

Перечисление со страницами:

```
public enum NavigationPageSource
{
    Page1,
    Page2
}
```

Файл `App.cs` :

```
public class App : Application
{
    public App()
    {
        var startPage = new Page1();
        InitializeNavigation(startPage);
        MainPage = new NavigationPage(startPage);
    }

    #region Sample of navigation initialization
    private void InitializeNavigation(Page startPage)
    {
        var mapper = new SuperMapper();
        mapper.AddMapping(typeof(Page1), NavigationPageSource.Page1);
        mapper.AddMapping(typeof(Page2), NavigationPageSource.Page2);

        var navigationService = DependencyService.Get<IViewNavigationService>();
        navigationService.Initialize(startPage.Navigation, mapper);
    }
    #endregion
}
```

В mapper I связан тип некоторой страницы с значением перечисления.

`IViewNavigationService` :

```
[assembly: Dependency(typeof(ViewNavigationService))]
namespace SuperForms.Core.ViewNavigation
{
    public class ViewNavigationService : IViewNavigationService
    {
        private INavigation _navigation;
        private SuperMapper _navigationMapper;
```

```
public void Initialize(INavigation navigation, SuperMapper navigationMapper)
{
    _navigation = navigation;
    _navigationMapper = navigationMapper;
}

public async Task NavigateToAsync(object navigationSource, object parameter = null)
{
    CheckIsInitialized();

    var type = _navigationMapper.GetTypeSource(navigationSource);

    if (type == null)
    {
        throw new InvalidOperationException(
            "Can't find associated type for " + navigationSource.ToString());
    }

    ConstructorInfo constructor;
    object[] parameters;

    if (parameter == null)
    {
        constructor = type.GetTypeInfo()
                          .DeclaredConstructors
                          .FirstOrDefault(c => !c.GetParameters().Any());

        parameters = new object[] { };
    }
    else
    {
        constructor = type.GetTypeInfo()
                          .DeclaredConstructors
                          .FirstOrDefault(c =>
                            {
                                var p = c.GetParameters();
                                return p.Count() == 1 &&
                                    p[0].ParameterType == parameter.GetType();
                            });

        parameters = new[] { parameter };
    }

    if (constructor == null)
    {
        throw new InvalidOperationException(
            "No suitable constructor found for page " + navigationSource.ToString());
    }

    var page = constructor.Invoke(parameters) as Page;

    await _navigation.PushAsync(page);
}

public async Task GoBackAsync()
{
    CheckIsInitialized();

    await _navigation.PopAsync();
}
```

```
        private void CheckIsInitialized()
        {
            if (_navigation == null || _navigationMapper == null)
                throw new NullReferenceException("Call Initialize method first.");
        }
    }
}
```

Я получаю тип страницы, по которой пользователь хочет перейти и создать экземпляр, используя отражение.

И тогда я мог бы использовать навигационную службу на модели просмотра:

```
var navigationService = DependencyService.Get<IViewNavigationService>();
await navigationService.NavigateToAsync(NavigationPageSource.Page2, "hello from Page1");
```

Прочитайте Навигация в Xamarin.Forms онлайн: https://riptutorial.com/ru/xamarin-forms/topic/2507/навигация-в-xamarin-forms

глава 18: Обработка исключений

Examples

Один из способов сообщить об исключениях в iOS

Перейдите в файл `Main.cs` в **проекте iOS** и измените существующий код, как `Main.cs` ниже:

```
    static void Main(string[] args)
    {
        try
        {
            UIApplication.Main(args, null, "AppDelegate");
        }
        catch (Exception ex)
        {
            Debug.WriteLine("iOS Main Exception: {0}", ex);

            var watson = new LittleWatson();
            watson.SaveReport(ex);
        }
    }
```

Интерфейс `ILittleWatson` , используемый в переносном коде, может выглядеть так:

```
public interface ILittleWatson
{
    Task<bool> SendReport();

    void SaveReport(Exception ex);
}
```

Реализация для **проекта iOS** :

```
[assembly: Xamarin.Forms.Dependency(typeof(LittleWatson))]
namespace SomeNamespace
{
    public class LittleWatson : ILittleWatson
    {
        private const string FileName = "Report.txt";

        private readonly static string DocumentsFolder;
        private readonly static string FilePath;

        private TaskCompletionSource<bool> _sendingTask;

        static LittleWatson()
        {
            DocumentsFolder =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
            FilePath = Path.Combine(DocumentsFolder, FileName);
        }
```

```
        public async Task<bool> SendReport()
        {
            _sendingTask = new TaskCompletionSource<bool>();

            try
            {
                var text = File.ReadAllText(FilePath);
                File.Delete(FilePath);
                if (MFMailComposeViewController.CanSendMail)
                {
                    var email = ""; // Put receiver email here.
                    var mailController = new MFMailComposeViewController();
                    mailController.SetToRecipients(new string[] { email });
                    mailController.SetSubject("iPhone error");
                    mailController.SetMessageBody(text, false);
                    mailController.Finished += (object s, MFComposeResultEventArgs args) =>
                    {
                        args.Controller.DismissViewController(true, null);
                        _sendingTask.TrySetResult(true);
                    };

                    ShowViewController(mailController);
                }
            }
            catch (FileNotFoundException)
            {
                // No errors found.
                _sendingTask.TrySetResult(false);
            }

            return await _sendingTask.Task;
        }

        public void SaveReport(Exception ex)
        {
            var exceptionInfo = $"{ex.Message} - {ex.StackTrace}";
            File.WriteAllText(FilePath, exceptionInfo);
        }

        private static void ShowViewController(UIViewController controller)
        {
            var topController = UIApplication.SharedApplication.KeyWindow.RootViewController;
            while (topController.PresentedViewController != null)
            {
                topController = topController.PresentedViewController;
            }

            topController.PresentViewController(controller, true, null);
        }
    }
}
```

И затем, где-то, где приложение запускается, поставьте:

```
        var watson = DependencyService.Get<ILittleWatson>();
        if (watson != null)
        {
            await watson.SendReport();
        }
```

Прочитайте Обработка исключений онлайн: https://riptutorial.com/ru/xamarin-forms/topic/6428/обработка-исключений

глава 19: Общий жизненный цикл приложения Xamarin.Forms? В зависимости от платформы!

Examples

Жизненный цикл Xamarin.Forms - это не фактический жизненный цикл приложения, а кросс-платформенное представление.

Давайте посмотрим на собственные методы жизненного цикла приложения для разных платформ.

Android.

```
//Xamarin.Forms.Platform.Android.FormsApplicationActivity lifecycle methods:
protected override void OnCreate(Bundle savedInstanceState);
protected override void OnDestroy();
protected override void OnPause();
protected override void OnRestart();
protected override void OnResume();
protected override void OnStart();
protected override void OnStop();
```

IOS.

```
//Xamarin.Forms.Platform.iOS.FormsApplicationDelegate lifecycle methods:
public override void DidEnterBackground(UIApplication uiApplication);
public override bool FinishedLaunching(UIApplication uiApplication, NSDictionary
launchOptions);
public override void OnActivated(UIApplication uiApplication);
public override void OnResignActivation(UIApplication uiApplication);
public override void WillEnterForeground(UIApplication uiApplication);
public override bool WillFinishLaunching(UIApplication uiApplication, NSDictionary
launchOptions);
public override void WillTerminate(UIApplication uiApplication);
```

Окна.

```
//Windows.UI.Xaml.Application lifecycle methods:
public event EventHandler<System.Object> Resuming;
public event SuspendingEventHandler Suspending;
protected virtual void OnActivated(IActivatedEventArgs args);
protected virtual void OnFileActivated(FileActivatedEventArgs args);
protected virtual void OnFileOpenPickerActivated(FileOpenPickerActivatedEventArgs args);
protected virtual void OnFileSavePickerActivated(FileSavePickerActivatedEventArgs args);
protected virtual void OnLaunched(LaunchActivatedEventArgs args);
protected virtual void OnSearchActivated(SearchActivatedEventArgs args);
protected virtual void OnShareTargetActivated(ShareTargetActivatedEventArgs args);
```

```
protected virtual void OnWindowCreated(WindowCreatedEventArgs args);

//Windows.UI.Xaml.Window lifecycle methods:
public event WindowActivatedEventHandler Activated;
public event WindowClosedEventHandler Closed;
public event WindowVisibilityChangedEventHandler VisibilityChanged;
```

И теперь **методы** жизненного цикла приложений **Xamarin.Forms** :

```
//Xamarin.Forms.Application lifecycle methods:
protected virtual void OnResume();
protected virtual void OnSleep();
protected virtual void OnStart();
```

То, что вы можете легко сказать, просто наблюдая за списками, многоплановая перспектива жизненного цикла платформы Xamarin.Forms значительно упрощена. Это дает вам общее представление о том, в каком состоянии находится ваше приложение, но в большинстве случаев производства вам нужно будет построить некоторую зависящую от платформы логику.

Прочитайте Общий жизненный цикл приложения Xamarin.Forms? В зависимости от платформы! онлайн: https://riptutorial.com/ru/xamarin-forms/topic/8329/общий-жизненный-цикл-приложения-xamarin-forms--в-зависимости-от-платформы-

глава 20: Относительная компоновка Xamarin

замечания

Использование *ForceLayout* в этом случае

Размер метки и кнопки изменяется в соответствии с текстом внутри них. Поэтому, когда дети добавляются в макет, их размер остается равным нулю как по ширине, так и по высоте. Например:

```
relativeLayout.Children.Add(label,
    Constraint.RelativeToParent(parent => label.Width));
```

Выше выражение вернет 0, так как в данный момент ширина равна нулю. Чтобы обойти это, нам нужно прослушать событие *SizeChanged,* и при изменении размера мы должны заставить макет, чтобы перерисовать его.

```
label.SizeChanged += (s, e) => relativeLayout.ForceLayout();
```

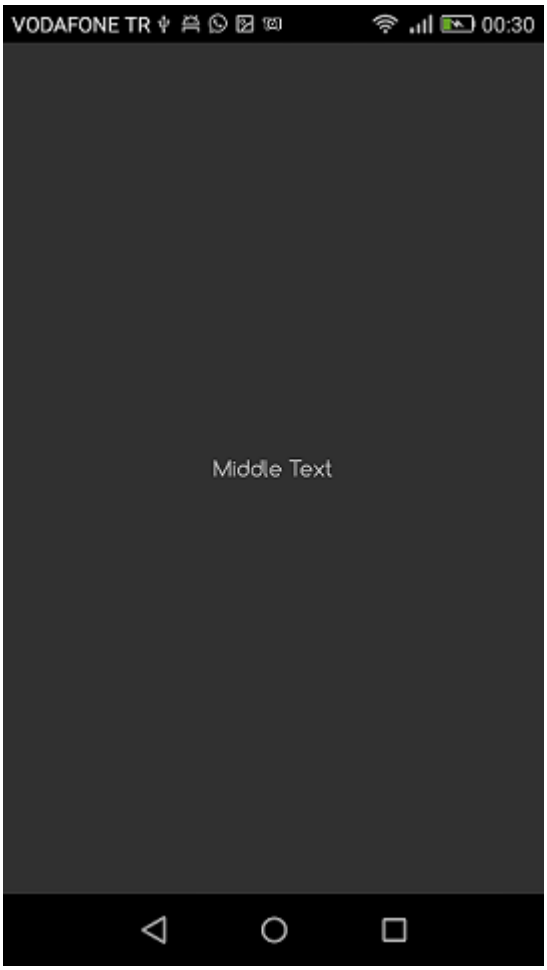
Для такого вида, как *BoxView,* это не нужно. Потому что мы можем определить их размеры при создании экземпляра. Другими словами, в обоих случаях мы можем определить их ширину и высоту как ограничение, когда мы добавляем их в макет. Например:

```
relativeLayout.Children.Add(label,
    Constraint.Constant(0),
    Constraint.Constant(0),
    //Width constraint
    Constraint.Constant(30),
    //Height constraint
    Constraint.Constant(40));
```

Это добавит метку к точке 0, 0. Ширина и высота метки будут 30 и 40. Однако, если текст слишком длинный, некоторые из них могут не отображаться. Если ваш ярлык имеет или может иметь высокую высоту, вы можете использовать свойство *LineBreakMode* метки. Который может обернуть текст. В *перечислении LineBreakMode* имеется много опций.

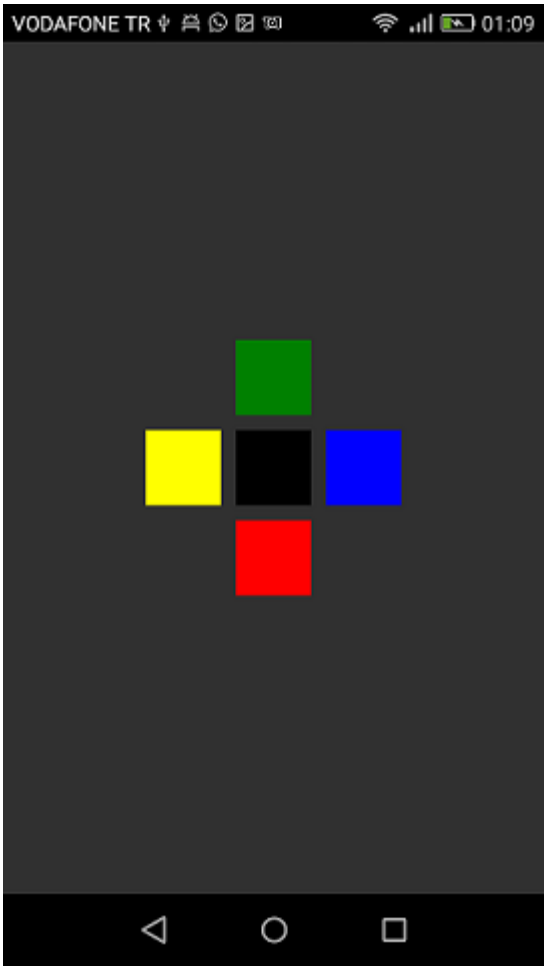
Examples

Страница с простой меткой на середине




```
public class MyPage : ContentPage
{
    RelativeLayout _layout;
    Label MiddleText;

    public MyPage()
    {
        _layout = new RelativeLayout();

        MiddleText = new Label
        {
            Text = "Middle Text"
        };

        MiddleText.SizeChanged += (s, e) =>
        {
            //We will force the layout so it will know the actual width and height of the
label
            //Otherwise width and height of the label remains 0 as far as layout knows
            _layout.ForceLayout();
        };

        _layout.Children.Add(MiddleText
            Constraint.RelativeToParent(parent => parent.Width / 2 - MiddleText.Width / 2),
            Constraint.RelativeToParent(parent => parent.Height / 2 - MiddleText.Height / 2));

        Content = _layout;
    }
}
```

Коробка после коробки




```
public class MyPage : ContentPage
{
    RelativeLayout _layout;

    BoxView centerBox;
    BoxView rightBox;
    BoxView leftBox;
    BoxView topBox;
    BoxView bottomBox;

    const int spacing = 10;
    const int boxSize = 50;

    public MyPage()
    {
        _layout = new RelativeLayout();

        centerBox = new BoxView
        {
            BackgroundColor = Color.Black
        };

        rightBox = new BoxView
        {
            BackgroundColor = Color.Blue,
            //You can both set width and hight here
            //Or when adding the control to the layout
```

```
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        leftBox = new BoxView
        {
            BackgroundColor = Color.Yellow,
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        topBox = new BoxView
        {
            BackgroundColor = Color.Green,
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        bottomBox = new BoxView
        {
            BackgroundColor = Color.Red,
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        //First adding center box since other boxes will be relative to center box
        _layout.Children.Add(centerBox,
            //Constraint for X, centering it horizontally
            //We give the expression as a paramater, parent is our layout in this case
            Constraint.RelativeToParent(parent => parent.Width / 2 - boxSize / 2),
            //Constraint for Y, centering it vertically
            Constraint.RelativeToParent(parent => parent.Height / 2 - boxSize / 2),
            //Constraint for Width
            Constraint.Constant(boxSize),
            //Constraint for Height
            Constraint.Constant(boxSize));

        _layout.Children.Add(leftBox,
            //The x constraint will relate on some level to centerBox
            //Which is the first parameter in this case
            //We both need to have parent and centerBox, which will be called sibling,
            //in our expression paramters
            //This expression will be our second paramater
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X - spacing -
boxSize),
            //Since we only need to move it left,
            //it's Y constraint will be centerBox' position at Y axis
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y)
            //No need to define the size constraints
            //Since we initialize them during instantiation
        );

        _layout.Children.Add(rightBox,
            //The only difference hear is adding spacing and boxSize instead of substracting
them
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X + spacing +
boxSize),
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y)
        );

        _layout.Children.Add(topBox,
```

```
            //Since we are going to move it vertically this thime
            //We need to do the math on Y Constraint
            //In this case, X constraint will be centerBox' position at X axis
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X),
            //We will do the math on Y axis this time
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y  - spacing -
boxSize)
        );

        _layout.Children.Add(bottomBox,
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X),
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y + spacing +
boxSize)
        );

        Content = _layout;
    }
}
```

Прочитайте Относительная компоновка Xamarin онлайн: https://riptutorial.com/ru/xamarin-forms/topic/6583/относительная-компоновка-xamarin

глава 21: Отображать предупреждение

Examples

DisplayAlert

На `Xamarin.Forms Page` по методу `DisplayAlert` может появиться окно предупреждения. Мы можем предоставить Title, Body (текст для предупреждения) и один / два Action Buttons. `Page` предлагает два переопределения метода `DisplayAlert` .

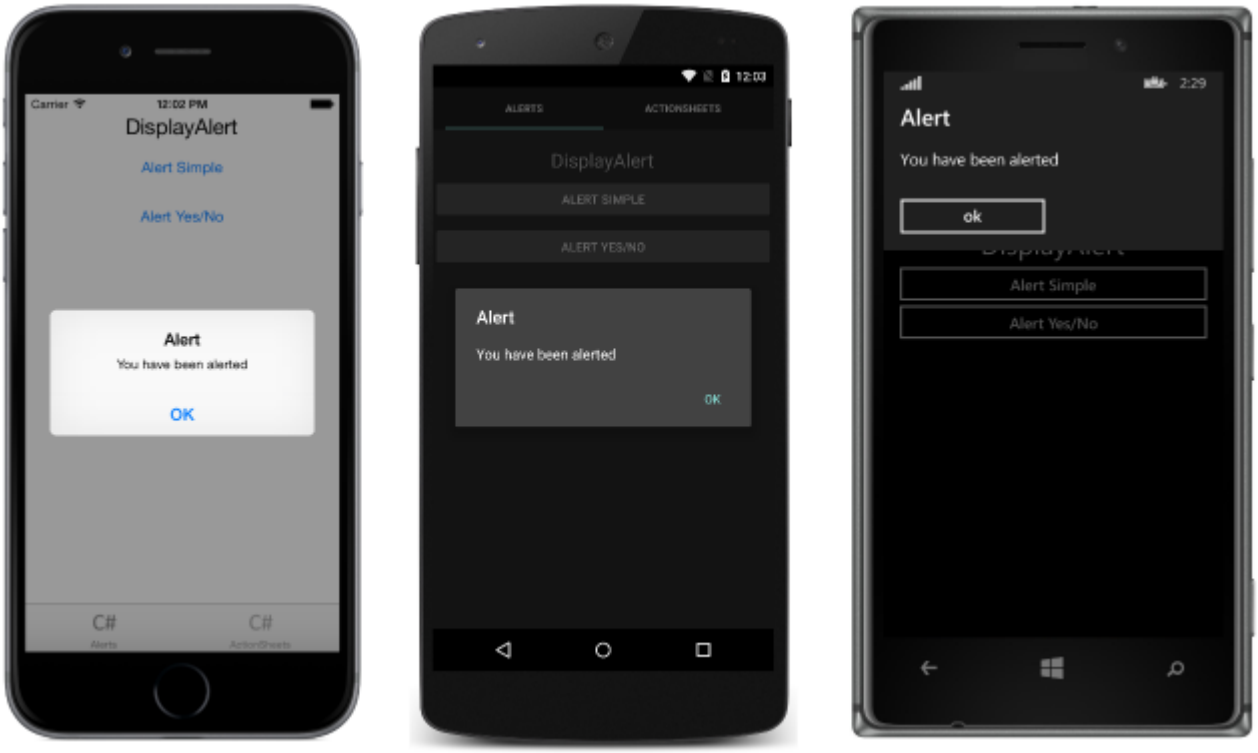
1. `public Task DisplayAlert (String title, String message, String cancel)`

Это переопределение представляет собой диалоговое окно предупреждения пользователю приложения с одной кнопкой отмены. Предупреждение отображает модально и после увольнения пользователь продолжает взаимодействовать с приложением.

Пример :

```
DisplayAlert ("Alert", "You have been alerted", "OK");
```

Над фрагментом будет представлена собственная реализация Alerts на каждой платформе (`AlertDialog` в Android, `UIAlertView` в iOS, `MessageDialog` в Windows), как `AlertDialog` ниже.



2. `public System.Threading.Tasks.Task<bool> DisplayAlert (String title, String message, String`

```
accept, String cancel)
```

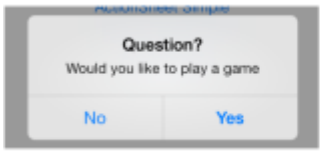
Это переопределение представляет собой диалоговое окно предупреждения для пользователя приложения с кнопкой принятия и отмены. Он фиксирует ответ пользователя, представляя две кнопки и возвращая `boolean` . Чтобы получить ответ от предупреждения, подайте текст для обеих кнопок и дождитесь этого метода. После того, как пользователь выберет одну из опций, ответ будет возвращен в код.

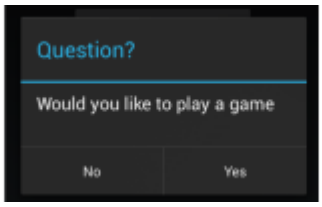
Пример :

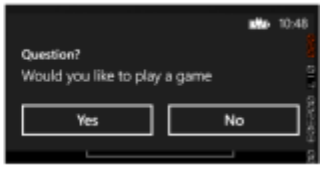
```
var answer = await DisplayAlert ("Question?", "Would you like to play a game", "Yes", "No");
Debug.WriteLine ("Answer: " + (answer?"Yes":"No"));
```

Пример 2: (если условие «Истина» или «ложная проверка» для продолжения)

```
async void listSelected(object sender, SelectedItemChangedEventArgs e)
    {
        var ans = await DisplayAlert("Question?", "Would you like Delete", "Yes", "No");
        if (ans == true)
        {
            //Success condition
        }
        else
        {
            //false conditon
        }
     }
```







Пример предупреждения с помощью только одной кнопки и действия

```
var alertResult = await DisplayAlert("Alert Title", Alert Message, null, "OK");
if(!alertResult)
{
   //do your stuff.
}
```

Здесь мы получим Ok click action.

Прочитайте Отображать предупреждение онлайн: https://riptutorial.com/ru/xamarin-forms/topic/4883/отображать-предупреждение

глава 22: Плагин Xamarin

Examples

Поделиться плагином

Простой способ поделиться сообщением или ссылкой, скопировать текст в буфер обмена или открыть браузер в любом приложении Xamarin или Windows.

Доступно на NuGet: https://www.nuget.org/packages/Plugin.Share/

XAML

```
<StackLayout Padding="20" Spacing="20">
      <Button StyleId="Text" Text="Share Text" Clicked="Button_OnClicked"/>
      <Button StyleId="Link" Text="Share Link" Clicked="Button_OnClicked"/>
      <Button StyleId="Browser" Text="Open Browser" Clicked="Button_OnClicked"/>
      <Label Text=""/>

 </StackLayout>
```

C #

```
 async void Button_OnClicked(object sender, EventArgs e)
        {
            switch (((Button)sender).StyleId)
            {
                case "Text":
                    await CrossShare.Current.Share("Follow @JamesMontemagno on Twitter",
"Share");
                    break;
                case "Link":
                    await CrossShare.Current.ShareLink("http://motzcod.es", "Checkout my
blog", "MotzCod.es");
                    break;
                case "Browser":
                    await CrossShare.Current.OpenBrowser("http://motzcod.es");
                    break;
            }
        }
```

ExternalMaps

Плагин внешних карт Откройте внешние карты, чтобы перейти к определенной геолокации или адресу. Возможность запуска с возможностью навигации в iOS.

Доступно на NuGet: [https://www.nuget.org/packages/Xam.Plugin.ExternalMaps/][1]

XAML

```
  <StackLayout Spacing="10" Padding="10">
      <Button x:Name="navigateAddress" Text="Navigate to Address"/>
      <Button x:Name="navigateLatLong" Text="Navigate to Lat|Long"/>
      <Label Text=""/>

    </StackLayout>
```

Код

```
namespace PluginDemo
{
    public partial class ExternalMaps : ContentPage
    {
        public ExternalMaps()
        {
            InitializeComponent();
            navigateLatLong.Clicked += (sender, args) =>
            {
                CrossExternalMaps.Current.NavigateTo("Space Needle", 47.6204, -122.3491);
            };

            navigateAddress.Clicked += (sender, args) =>
            {
                CrossExternalMaps.Current.NavigateTo("Xamarin", "394 pacific ave.", "San
Francisco", "CA", "94111", "USA", "USA");
            };
        }
    }
}
```

Плагин геолокатора

Легко получить доступ к геолокации через Xamarin.iOS, Xamarin.Android и Windows.

Доступен Nuget: [https://www.nuget.org/packages/Xam.Plugin.Geolocator/][1]

XAML

```
 <StackLayout Spacing="10" Padding="10">
      <Button x:Name="buttonGetGPS" Text="Get GPS"/>
      <Label x:Name="labelGPS"/>
      <Button x:Name="buttonTrack" Text="Track Movements"/>
      <Label x:Name="labelGPSTrack"/>
      <Label Text=""/>

    </StackLayout>
```

Код

```
namespace PluginDemo
{
    public partial class GeolocatorPage : ContentPage
    {
        public GeolocatorPage()
        {
```

```
            InitializeComponent();
            buttonGetGPS.Clicked += async (sender, args) =>
            {
                try
                {
                    var locator = CrossGeolocator.Current;
                    locator.DesiredAccuracy = 1000;
                    labelGPS.Text = "Getting gps";

                    var position = await locator.GetPositionAsync(timeoutMilliseconds: 10000);

                    if (position == null)
                    {
                        labelGPS.Text = "null gps :(";
                        return;
                    }
                    labelGPS.Text = string.Format("Time: {0} \nLat: {1} \nLong: {2}
\nAltitude: {3} \nAltitude Accuracy: {4} \nAccuracy: {5} \nHeading: {6} \nSpeed: {7}",
                        position.Timestamp, position.Latitude, position.Longitude,
                        position.Altitude, position.AltitudeAccuracy, position.Accuracy,
position.Heading, position.Speed);

                }
                catch //(Exception ex)
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };

            buttonTrack.Clicked += async (object sender, EventArgs e) =>
            {
                try
                {
                    if (CrossGeolocator.Current.IsListening)
                    {
                        await CrossGeolocator.Current.StopListeningAsync();
                        labelGPSTrack.Text = "Stopped tracking";
                        buttonTrack.Text = "Stop Tracking";
                    }
                    else
                    {
                        if (await CrossGeolocator.Current.StartListeningAsync(30000, 0))
                        {
                            labelGPSTrack.Text = "Started tracking";
                            buttonTrack.Text = "Track Movements";
                        }
                    }
                }
                catch //(Exception ex)
                {
                    //Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };
        }

        protected override void OnAppearing()
        {
```

```
            base.OnAppearing();
            try
            {
                CrossGeolocator.Current.PositionChanged +=
CrossGeolocator_Current_PositionChanged;
                CrossGeolocator.Current.PositionError +=
CrossGeolocator_Current_PositionError;
            }
            catch
            {
            }
        }

        void CrossGeolocator_Current_PositionError(object sender,
Plugin.Geolocator.Abstractions.PositionErrorEventArgs e)
        {

            labelGPSTrack.Text = "Location error: " + e.Error.ToString();
        }

        void CrossGeolocator_Current_PositionChanged(object sender,
Plugin.Geolocator.Abstractions.PositionEventArgs e)
        {
            var position = e.Position;
            labelGPSTrack.Text = string.Format("Time: {0} \nLat: {1} \nLong: {2} \nAltitude:
{3} \nAltitude Accuracy: {4} \nAccuracy: {5} \nHeading: {6} \nSpeed: {7}",
                position.Timestamp, position.Latitude, position.Longitude,
                position.Altitude, position.AltitudeAccuracy, position.Accuracy,
position.Heading, position.Speed);


        }

        protected override void OnDisappearing()
        {
            base.OnDisappearing();
            try
            {
                CrossGeolocator.Current.PositionChanged -=
CrossGeolocator_Current_PositionChanged;
                CrossGeolocator.Current.PositionError -=
CrossGeolocator_Current_PositionError;
            }
            catch
            {
            }
        }
    }
}
```

Медиаплагин

Возьмите или выберите фотографии и видео из API кросс-платформы.

Доступен Nuget: [https://www.nuget.org/packages/Xam.Plugin.Media/][1]

XAML

```
<StackLayout Spacing="10" Padding="10">
      <Button x:Name="takePhoto" Text="Take Photo"/>
      <Button x:Name="pickPhoto" Text="Pick Photo"/>
      <Button x:Name="takeVideo" Text="Take Video"/>
      <Button x:Name="pickVideo" Text="Pick Video"/>
      <Label Text="Save to Gallery"/>
      <Switch x:Name="saveToGallery" IsToggled="false" HorizontalOptions="Center"/>
      <Label Text="Image will show here"/>
      <Image x:Name="image"/>
      <Label Text=""/>

    </StackLayout>
```

Код

```
namespace PluginDemo
{
    public partial class MediaPage : ContentPage
    {
        public MediaPage()
        {
            InitializeComponent();
            takePhoto.Clicked += async (sender, args) =>
            {

                if (!CrossMedia.Current.IsCameraAvailable ||
!CrossMedia.Current.IsTakePhotoSupported)
                {
                    await DisplayAlert("No Camera", ":( No camera avaialble.", "OK");
                    return;
                }
                try
                {
                    var file = await CrossMedia.Current.TakePhotoAsync(new
Plugin.Media.Abstractions.StoreCameraMediaOptions
                    {
                        Directory = "Sample",
                        Name = "test.jpg",
                        SaveToAlbum = saveToGallery.IsToggled
                    });

                    if (file == null)
                        return;

                    await DisplayAlert("File Location", (saveToGallery.IsToggled ?
file.AlbumPath : file.Path), "OK");

                    image.Source = ImageSource.FromStream(() =>
                    {
                        var stream = file.GetStream();
                        file.Dispose();
                        return stream;
                    });
                }
                catch //(Exception ex)
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
```

```
            };

            pickPhoto.Clicked += async (sender, args) =>
            {
                if (!CrossMedia.Current.IsPickPhotoSupported)
                {
                    await DisplayAlert("Photos Not Supported", ":( Permission not granted to
photos.", "OK");
                    return;
                }
                try
                {
                    Stream stream = null;
                    var file = await CrossMedia.Current.PickPhotoAsync().ConfigureAwait(true);


                    if (file == null)
                        return;

                    stream = file.GetStream();
                    file.Dispose();

                    image.Source = ImageSource.FromStream(() => stream);

                }
                catch //(Exception ex)
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };

            takeVideo.Clicked += async (sender, args) =>
            {
                if (!CrossMedia.Current.IsCameraAvailable ||
!CrossMedia.Current.IsTakeVideoSupported)
                {
                    await DisplayAlert("No Camera", ":( No camera avaialble.", "OK");
                    return;
                }

                try
                {
                    var file = await CrossMedia.Current.TakeVideoAsync(new
Plugin.Media.Abstractions.StoreVideoOptions
                    {
                        Name = "video.mp4",
                        Directory = "DefaultVideos",
                        SaveToAlbum = saveToGallery.IsToggled
                    });

                    if (file == null)
                        return;

                    await DisplayAlert("Video Recorded", "Location: " +
(saveToGallery.IsToggled ? file.AlbumPath : file.Path), "OK");

                    file.Dispose();

                }
```

```
                catch //(Exception ex)
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };

            pickVideo.Clicked += async (sender, args) =>
            {
                if (!CrossMedia.Current.IsPickVideoSupported)
                {
                    await DisplayAlert("Videos Not Supported", ":( Permission not granted to
videos.", "OK");
                    return;
                }
                try
                {
                    var file = await CrossMedia.Current.PickVideoAsync();

                    if (file == null)
                        return;

                    await DisplayAlert("Video Selected", "Location: " + file.Path, "OK");
                    file.Dispose();

                }
                catch //(Exception ex)
                {
                    //Xamarin.Insights.Report(ex);
                    //await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };
        }
    }
}
```

Плагин сообщений

Плагин сообщений для Xamarin и Windows для совершения телефонного звонка, отправки смс или отправки электронной почты с использованием приложений обмена сообщениями по умолчанию на разных мобильных платформах.

Доступный Nuget: [https://www.nuget.org/packages/Xam.Plugins.Messaging/][1]

XAML

```
 <StackLayout Spacing="10" Padding="10">
      <Entry Placeholder="Phone Number" x:Name="phone"/>
      <Button x:Name="buttonSms" Text="Send SMS"/>
      <Button x:Name="buttonCall" Text="Call Phone Number"/>
      <Entry Placeholder="E-mail Address" x:Name="email"/>
      <Button x:Name="buttonEmail" Text="Send E-mail"/>
      <Label Text=""/>

    </StackLayout>
```

Код

```
namespace PluginDemo
{
    public partial class MessagingPage : ContentPage
    {
        public MessagingPage()
        {
            InitializeComponent();
            buttonCall.Clicked += async (sender, e) =>
            {
                try
                {
                    // Make Phone Call
                    var phoneCallTask = MessagingPlugin.PhoneDialer;
                    if (phoneCallTask.CanMakePhoneCall)
                        phoneCallTask.MakePhoneCall(phone.Text);
                    else
                        await DisplayAlert("Error", "This device can't place calls", "OK");
                }
                catch
                {
                   // await DisplayAlert("Error", "Unable to perform action", "OK");
                }
            };

            buttonSms.Clicked += async (sender, e) =>
            {
                try
                {

                    var smsTask = MessagingPlugin.SmsMessenger;
                    if (smsTask.CanSendSms)
                        smsTask.SendSms(phone.Text, "Hello World");
                    else
                        await DisplayAlert("Error", "This device can't send sms", "OK");
                }
                catch
                {
                   // await DisplayAlert("Error", "Unable to perform action", "OK");
                }
            };

            buttonEmail.Clicked += async (sender, e) =>
            {
                try
                {
                    var emailTask = MessagingPlugin.EmailMessenger;
                    if (emailTask.CanSendEmail)
                        emailTask.SendEmail(email.Text, "Hello there!", "This was sent from
the Xamrain Messaging Plugin from shared code!");
                    else
                        await DisplayAlert("Error", "This device can't send emails", "OK");
                }
                catch
                {
//await DisplayAlert("Error", "Unable to perform action", "OK");
                }
            };
        }
    }
```

```
}
```

Плагин разрешений

Проверьте, разрешены ли ваши пользователи или запрещены разрешения для общих групп разрешений на iOS и Android.

Кроме того, вы можете запрашивать разрешения с помощью простого межплатформенного асинхронного / ожидаемого API.

Доступен Nuget: https://www.nuget.org/packages/Plugin.Permissions введите ссылку здесь
XAML

XAML

```
 <StackLayout Padding="30" Spacing="10">
      <Button Text="Get Location" Clicked="Button_OnClicked"></Button>
      <Label x:Name="LabelGeolocation"></Label>
      <Button Text="Calendar" StyleId="Calendar"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Camera" StyleId="Camera" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Contacts" StyleId="Contacts"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Microphone" StyleId="Microphone"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Phone" StyleId="Phone" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Photos" StyleId="Photos" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Reminders" StyleId="Reminders"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Sensors" StyleId="Sensors" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Sms" StyleId="Sms" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Storage" StyleId="Storage" Clicked="ButtonPermission_OnClicked"></Button>
      <Label Text=""/>

    </StackLayout>
```

Код

```
bool busy;
        async void ButtonPermission_OnClicked(object sender, EventArgs e)
        {
            if (busy)
                return;

            busy = true;
            ((Button)sender).IsEnabled = false;

            var status = PermissionStatus.Unknown;
            switch (((Button)sender).StyleId)
            {
                case "Calendar":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Calendar);
                    break;
                case "Camera":
```

```
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Camera);
                    break;
                case "Contacts":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Contacts);
                    break;
                case "Microphone":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Microphone);
                    break;
                case "Phone":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Phone);
                    break;
                case "Photos":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Photos);
                    break;
                case "Reminders":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Reminders);
                    break;
                case "Sensors":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Sensors);
                    break;
                case "Sms":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Sms);
                    break;
                case "Storage":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Storage);
                    break;
            }

            await DisplayAlert("Results", status.ToString(), "OK");

            if (status != PermissionStatus.Granted)
            {
                switch (((Button)sender).StyleId)
                {
                    case "Calendar":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Calendar))[Permission.Calendar];
                        break;
                    case "Camera":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Camera))[Permission.Camera];
                        break;
                    case "Contacts":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Contacts))[Permission.Contacts];
                        break;
                    case "Microphone":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Microphone))[Permission.Microphone];

                        break;
                    case "Phone":
```

```
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Phone))[Permission.Phone];
                        break;
                    case "Photos":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Photos))[Permission.Photos];
                        break;
                    case "Reminders":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Reminders))[Permission.Reminders];
                        break;
                    case "Sensors":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Sensors))[Permission.Sensors];
                        break;
                    case "Sms":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Sms))[Permission.Sms];
                        break;
                    case "Storage":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Storage))[Permission.Storage];
                        break;
                }

                await DisplayAlert("Results", status.ToString(), "OK");

            }

            busy = false;
            ((Button)sender).IsEnabled = true;
        }

        async void Button_OnClicked(object sender, EventArgs e)
        {
            if (busy)
                return;

            busy = true;
            ((Button)sender).IsEnabled = false;

            try
            {
                var status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Location);
                if (status != PermissionStatus.Granted)
                {
                    if (await
CrossPermissions.Current.ShouldShowRequestPermissionRationaleAsync(Permission.Location))
                    {
                        await DisplayAlert("Need location", "Gunna need that location", "OK");
                    }

                    var results = await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Location);
                    status = results[Permission.Location];
                }

                if (status == PermissionStatus.Granted)
                {
                    var results = await CrossGeolocator.Current.GetPositionAsync(10000);
```

```
                    LabelGeolocation.Text = "Lat: " + results.Latitude + " Long: " +
results.Longitude;
                }
                else if (status != PermissionStatus.Unknown)
                {
                    await DisplayAlert("Location Denied", "Can not continue, try again.",
"OK");
                }
            }
            catch (Exception ex)
            {

                LabelGeolocation.Text = "Error: " + ex;
            }

            ((Button)sender).IsEnabled = true;
            busy = false;
        }
```

Прочитайте Плагин Xamarin онлайн: https://riptutorial.com/ru/xamarin-forms/topic/7017/плагин-xamarin

глава 23: Поведение на платформе

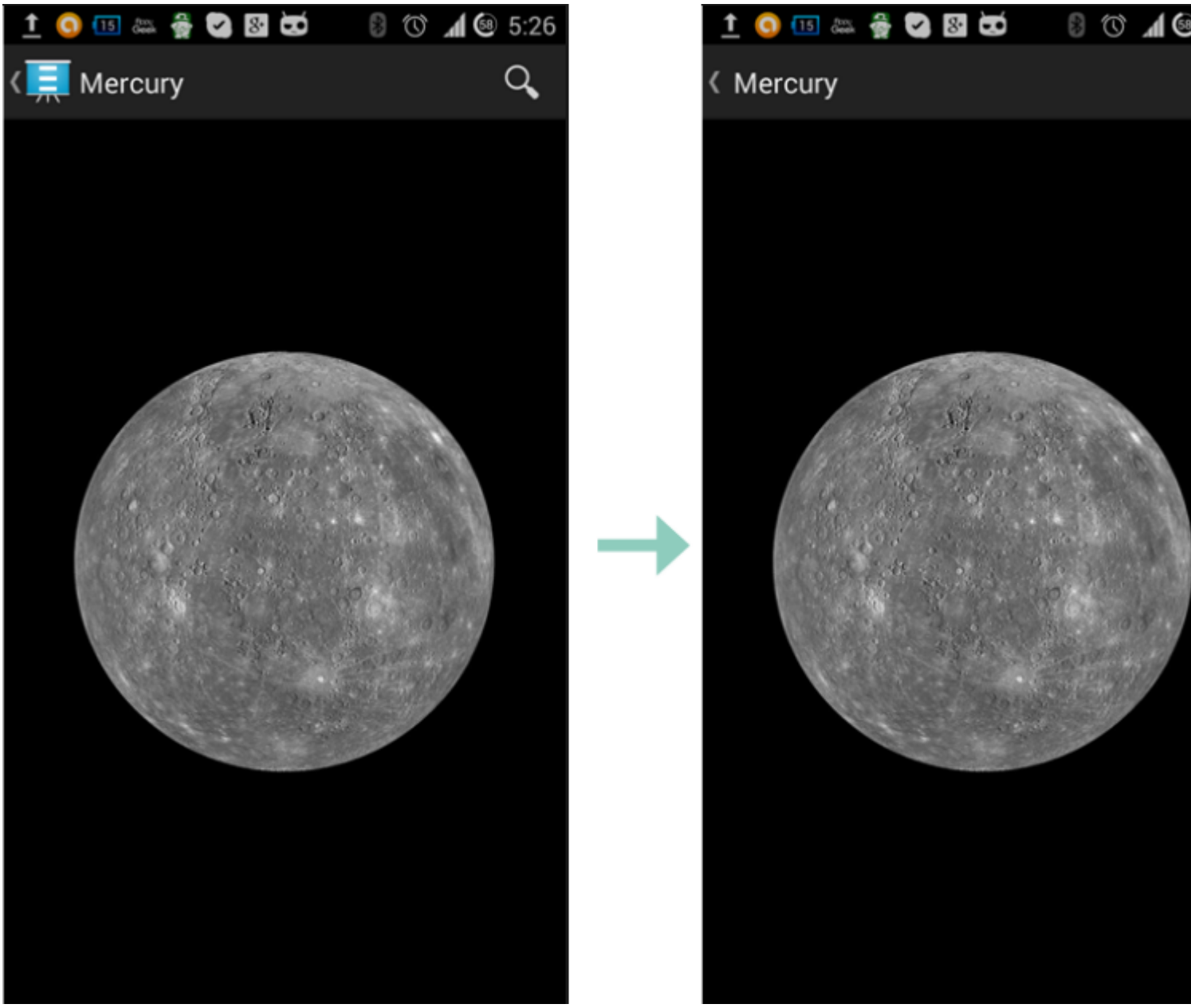
замечания

Целевые платформы

```
if(Device.OS == TargetPlatform.Android)
{

}
else if (Device.OS == TargetPlatform.iOS)
{

}
else if (Device.OS == TargetPlatform.WinPhone)
{

}
else if (Device.OS == TargetPlatform.Windows)
{

}
else if (Device.OS == TargetPlatform.Other)
{

}
```

Examples

Удаление значка в заголовке навигации в Anroid



Использование небольшого прозрачного изображения под названием empty.png

```
public class MyPage : ContentPage
{
    public Page()
    {
        if (Device.OS == TargetPlatform.Android)
            NavigationPage.SetTitleIcon(this, "empty.png");
    }
}
```

Сделать размер шрифта ярлыка меньше в iOS

```
Label label = new Label
{
    Text = "text"
};
if(Device.OS == TargetPlatform.iOS)
{
```

```
    label.FontSize = label.FontSize - 2;
}
```

Прочитайте Поведение на платформе онлайн: https://riptutorial.com/ru/xamarin-forms/topic/6636/поведение-на-платформе

глава 24: Пользовательские рендереры

Examples

Пользовательский рендеринг для ListView

Пользовательские рендереры позволяют разработчикам настраивать внешний вид и поведение элементов управления Xamarin.Forms на каждой платформе. Разработчики могли использовать функции встроенных средств управления.

Например, нам нужно отключить прокрутку в `ListView` . В iOS `ListView` прокручивается, даже если все элементы размещены на экране, и пользователь не сможет прокручивать список. `Xamarin.Forms.ListView` не управляет такой настройкой. В этом случае рендеринг приходит на помощь.

Во-первых, мы должны создать настраиваемый элемент управления в проекте PCL, который объявит некоторое требуемое свойство bindable:

```
public class SuperListView : ListView
{
    public static readonly BindableProperty IsScrollingEnableProperty =
        BindableProperty.Create(nameof(IsScrollingEnable),
                                typeof(bool),
                                typeof(SuperListView),
                                true);

    public bool IsScrollingEnable
    {
        get { return (bool)GetValue(IsScrollingEnableProperty); }
        set { SetValue(IsScrollingEnableProperty, value); }
    }
}
```

Следующим шагом будет создание рендерера для каждой платформы.

IOS:

```
[assembly: ExportRenderer(typeof(SuperListView), typeof(SuperListViewRenderer))]
namespace SuperForms.iOS.Renderers
{
    public class SuperListViewRenderer : ListViewRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<ListView> e)
        {
            base.OnElementChanged(e);

            var superListView = Element as SuperListView;
            if (superListView == null)
                return;
```

```
            Control.ScrollEnabled = superListView.IsScrollingEnable;
        }
    }
}
```

И Android (список Android не имеет прокрутки, если все элементы размещены на экране, поэтому мы не будем отключать прокрутку, но все же мы можем использовать собственные свойства):

```
[assembly: ExportRenderer(typeof(SuperListView), typeof(SuperListViewRenderer))]
namespace SuperForms.Droid.Renderers
{
    public class SuperListViewRenderer : ListViewRenderer
    {
        protected override void
OnElementChanged(ElementChangedEventArgs<Xamarin.Forms.ListView> e)
        {
            base.OnElementChanged(e);

            var superListView = Element as SuperListView;
            if (superListView == null)
                return;
        }
    }
}
```

Свойством `Element` визуализации является мой `SuperListView` управления `SuperListView` из проекта PCL.

`Control` свойство средства визуализации - это собственный контроль. `Android.Widget.ListView` для Android и `UIKit.UITableView` для iOS.

И как мы будем использовать его в `XAML` :

```
<ContentPage x:Name="Page"
         xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         xmlns:controls="clr-namespace:SuperForms.Controls;assembly=SuperForms.Controls"
         x:Class="SuperForms.Samples.SuperListViewPage">


  <controls:SuperListView ItemsSource="{Binding Items, Source={x:Reference Page}}"
                          IsScrollingEnable="false"
                          Margin="20">
    <controls:SuperListView.ItemTemplate>
      <DataTemplate>
        <ViewCell>
          <Label Text="{Binding .}"/>
        </ViewCell>
      </DataTemplate>
    </controls:SuperListView.ItemTemplate>
  </controls:SuperListView>
</ContentPage>
```

`.cs`

файл страницы:

```
public partial class SuperListViewPage : ContentPage
{
    private ObservableCollection<string> _items;

    public ObservableCollection<string> Items
    {
        get { return _items; }
        set
        {
            _items = value;
            OnPropertyChanged();
        }
    }

    public SuperListViewPage()
    {
        var list = new SuperListView();

        InitializeComponent();

        var items = new List<string>(10);
        for (int i = 1; i <= 10; i++)
        {
            items.Add($"Item {i}");
        }

        Items = new ObservableCollection<string>(items);
    }
}
```

Пользовательский рендеринг для BoxView

Custom Renderer помогает добавлять новые свойства и визуализировать их по-разному в собственной платформе, которые иначе не могут быть реализованы с помощью общего кода. В этом примере мы добавим радиус и тень к boxview.

Во-первых, мы должны создать настраиваемый элемент управления в проекте PCL, который объявит некоторое требуемое свойство bindable:

```
namespace Mobile.Controls
{
    public class ExtendedBoxView : BoxView
    {
        /// <summary>
        /// Respresents the background color of the button.
        /// </summary>
        public static readonly BindableProperty BorderRadiusProperty =
BindableProperty.Create<ExtendedBoxView, double>(p => p.BorderRadius, 0);

        public double BorderRadius
        {
            get { return (double)GetValue(BorderRadiusProperty); }
            set { SetValue(BorderRadiusProperty, value); }
        }
```

```
        public static readonly BindableProperty StrokeProperty =
            BindableProperty.Create<ExtendedBoxView, Color>(p => p.Stroke, Color.Transparent);

        public Color Stroke
        {
            get { return (Color)GetValue(StrokeProperty); }
            set { SetValue(StrokeProperty, value); }
        }

        public static readonly BindableProperty StrokeThicknessProperty =
            BindableProperty.Create<ExtendedBoxView, double>(p => p.StrokeThickness, 0);

        public double StrokeThickness
        {
            get { return (double)GetValue(StrokeThicknessProperty); }
            set { SetValue(StrokeThicknessProperty, value); }
        }
    }
}
```

Следующим шагом будет создание рендерера для каждой платформы.

IOS:

```
[assembly: ExportRenderer(typeof(ExtendedBoxView), typeof(ExtendedBoxViewRenderer))]
namespace Mobile.iOS.Renderers
{
 public class ExtendedBoxViewRenderer : VisualElementRenderer<BoxView>
 {
    public ExtendedBoxViewRenderer()
    {
    }

    protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
    {
        base.OnElementChanged(e);
        if (Element == null)
            return;

        Layer.MasksToBounds = true;
        Layer.CornerRadius = (float)((ExtendedBoxView)this.Element).BorderRadius / 2.0f;
    }


    protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
    {
        base.OnElementPropertyChanged(sender, e);
        if (e.PropertyName == ExtendedBoxView.BorderRadiusProperty.PropertyName)
        {
            SetNeedsDisplay();
        }
    }

    public override void Draw(CGRect rect)
    {
        ExtendedBoxView roundedBoxView = (ExtendedBoxView)this.Element;
        using (var context = UIGraphics.GetCurrentContext())
        {
```

```
            context.SetFillColor(roundedBoxView.Color.ToCGColor());
            context.SetStrokeColor(roundedBoxView.Stroke.ToCGColor());
            context.SetLineWidth((float)roundedBoxView.StrokeThickness);

            var rCorner = this.Bounds.Inset((int)roundedBoxView.StrokeThickness / 2,
(int)roundedBoxView.StrokeThickness / 2);

            nfloat radius = (nfloat)roundedBoxView.BorderRadius;
            radius = (nfloat)Math.Max(0, Math.Min(radius, Math.Max(rCorner.Height / 2,
rCorner.Width / 2)));

            var path = CGPath.FromRoundedRect(rCorner, radius, radius);
            context.AddPath(path);
            context.DrawPath(CGPathDrawingMode.FillStroke);
        }
    }

 }
}
```

Снова вы можете настроить, как хотите, внутри метода draw.

И то же самое для Android:

```
[assembly: ExportRenderer(typeof(ExtendedBoxView), typeof(ExtendedBoxViewRenderer))]
namespace Mobile.Droid
{
    /// <summary>
    ///
    /// </summary>
    public class ExtendedBoxViewRenderer : VisualElementRenderer<BoxView>
    {
        /// <summary>
        ///
        /// </summary>
        public ExtendedBoxViewRenderer()
        {
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="e"></param>
        protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
        {
            base.OnElementChanged(e);

            SetWillNotDraw(false);

            Invalidate();
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
```

```
        {
            base.OnElementPropertyChanged(sender, e);

            if (e.PropertyName == ExtendedBoxView.BorderRadiusProperty.PropertyName)
            {
                Invalidate();
            }
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="canvas"></param>
        public override void Draw(Canvas canvas)
        {
            var box = Element as ExtendedBoxView;
            base.Draw(canvas);
            Paint myPaint = new Paint();

            myPaint.SetStyle(Paint.Style.Stroke);
            myPaint.StrokeWidth = (float)box.StrokeThickness;
            myPaint.SetARGB(convertTo255ScaleColor(box.Color.A),
convertTo255ScaleColor(box.Color.R), convertTo255ScaleColor(box.Color.G),
convertTo255ScaleColor(box.Color.B));
            myPaint.SetShadowLayer(20, 0, 5, Android.Graphics.Color.Argb(100, 0, 0, 0));

            SetLayerType(Android.Views.LayerType.Software, myPaint);

            var number = (float)box.StrokeThickness / 2;
            RectF rectF = new RectF(
                        number, // left
                        number, // top
                        canvas.Width - number, // right
                        canvas.Height - number // bottom
                );


            var radius = (float)box.BorderRadius;
            canvas.DrawRoundRect(rectF, radius, radius, myPaint);
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="color"></param>
        /// <returns></returns>
        private int convertTo255ScaleColor(double color)
        {
            return (int) Math.Ceiling(color * 255);
        }
}
```

}

XAML:

Сначала мы ссылаемся на наш контроль с пространством имен, которое мы определили ранее.

```
xmlns:Controls="clr-namespace:Mobile.Controls"
```

Затем мы используем Control следующим образом и используем свойства, определенные в начале:

```
<Controls:ExtendedBoxView
    x:Name="search_boxview"
    Color="#444"
    BorderRadius="5"
    HorizontalOptions="CenterAndExpand"
    />
```

Доступ к рендереру из собственного проекта

```
var renderer = Platform.GetRenderer(visualElement);

if (renderer == null)
{
    renderer = Platform.CreateRenderer(visualElement);
    Platform.SetRenderer(visualElement, renderer);
}

DoSomeThingWithRender(render); // now you can do whatever you want with render
```

Закругленная метка с помощью специального рендеринга для Frame (части PCL и iOS)

Первый шаг: часть PCL

```
using Xamarin.Forms;

namespace ProjectNamespace
{
    public class ExtendedFrame : Frame
    {
        /// <summary>
        /// The corner radius property.
        /// </summary>
        public static readonly BindableProperty CornerRadiusProperty =
            BindableProperty.Create("CornerRadius", typeof(double), typeof(ExtendedFrame),
0.0);

        /// <summary>
        /// Gets or sets the corner radius.
        /// </summary>
        public double CornerRadius
        {
            get { return (double)GetValue(CornerRadiusProperty); }
            set { SetValue(CornerRadiusProperty, value); }
        }
    }
}
```

Второй шаг: часть iOS

```
using ProjectNamespace;
using ProjectNamespace.iOS;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;

[assembly: ExportRenderer(typeof(ExtendedFrame), typeof(ExtendedFrameRenderer))]
namespace ProjectNamespace.iOS
{
    public class ExtendedFrameRenderer : FrameRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<Frame> e)
        {
            base.OnElementChanged(e);

            if (Element != null)
            {
                Layer.MasksToBounds = true;
                Layer.CornerRadius = (float)(Element as ExtendedFrame).CornerRadius;
            }
        }

        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (e.PropertyName == ExtendedFrame.CornerRadiusProperty.PropertyName)
            {
                Layer.CornerRadius = (float)(Element as ExtendedFrame).CornerRadius;
            }
        }
    }
}
```

Третий шаг: код XAML для вызова ExtendedFrame

Если вы хотите использовать его в части XAML, не забудьте написать это:

```
xmlns:controls="clr-namespace:ProjectNamespace;assembly:ProjectNamespace"
```

после

```
xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
```

Теперь вы можете использовать ExtendedFrame следующим образом:

```
<controls:ExtendedFrame
    VerticalOptions="FillAndExpand"
    HorizontalOptions="FillAndExpand"
    BackgroundColor="Gray"
    CornerRadius="35.0">
    <Frame.Content>
        <Label
```

```
            Text="MyText"
            TextColor="Blue"/>
    </Frame.Content>
</controls:ExtendedFrame>
```

Округленный BoxView с возможностью выбора цвета фона

Первый шаг: часть PCL

```
public class RoundedBoxView : BoxView
{
    public static readonly BindableProperty CornerRadiusProperty =
        BindableProperty.Create("CornerRadius", typeof(double), typeof(RoundedEntry),
default(double));

    public double CornerRadius
    {
        get
        {
            return (double)GetValue(CornerRadiusProperty);
        }
        set
        {
            SetValue(CornerRadiusProperty, value);
        }
    }

    public static readonly BindableProperty FillColorProperty =
        BindableProperty.Create("FillColor", typeof(string), typeof(RoundedEntry),
default(string));

    public string FillColor
    {
        get
        {
            return (string) GetValue(FillColorProperty);
        }
        set
        {
            SetValue(FillColorProperty, value);
        }
    }
}
```

Второй шаг: часть дроида

```
[assembly: ExportRenderer(typeof(RoundedBoxView), typeof(RoundedBoxViewRenderer))]
namespace MyNamespace.Droid
{
    public class RoundedBoxViewRenderer : VisualElementRenderer<BoxView>
    {
        protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
        {
            base.OnElementChanged(e);
            SetWillNotDraw(false);
            Invalidate();
        }
```

```
        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);
            SetWillNotDraw(false);
            Invalidate();
        }

        public override void Draw(Canvas canvas)
        {
            var box = Element as RoundedBoxView;
            var rect = new Rect();
            var paint = new Paint
            {
                Color = Xamarin.Forms.Color.FromHex(box.FillColor).ToAndroid(),
                AntiAlias = true,
            };

            GetDrawingRect(rect);

            var radius = (float)(rect.Width() / box.Width * box.CornerRadius);

            canvas.DrawRoundRect(new RectF(rect), radius, radius, paint);
        }
    }
}
```

Третий шаг: часть iOS

```
[assembly: ExportRenderer(typeof(RoundedBoxView), typeof(RoundedBoxViewRenderer))]
namespace MyNamespace.iOS
{
    public class RoundedBoxViewRenderer : BoxRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
        {
            base.OnElementChanged(e);

            if (Element != null)
            {
                Layer.CornerRadius = (float)(Element as RoundedBoxView).CornerRadius;
                Layer.BackgroundColor = Color.FromHex((Element as
RoundedBoxView).FillColor).ToCGColor();
            }
        }

        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (Element != null)
            {
                Layer.CornerRadius = (float)(Element as RoundedBoxView).CornerRadius;
                Layer.BackgroundColor = (Element as RoundedBoxView).FillColor.ToCGColor();
            }
        }
    }
}
```

Прочитайте Пользовательские рендереры онлайн: https://riptutorial.com/ru/xamarin-forms/topic/2949/пользовательские-рендереры

глава 25: Пользовательские шрифты в стилях

замечания

Ресурсы для просмотра:

- Стили Xamarin
- Использование пользовательских шрифтов на iOS и Android с помощью Xamarin.Forms
- Пользовательские рендереры
- Ресурсные словари
- Прикрепленные свойства

Examples

Доступ к обычным шрифтам в Силе

Xamarin.Forms предоставляют отличный механизм для стилизации вашего кросс-платформенного приложения с использованием глобальных стилей.

В мобильном мире ваше приложение должно быть красивым и выделяться из других приложений. Один из этих символов - пользовательские шрифты, используемые в приложении.

Благодаря поддержке питания XAML Styling в Xamarin.Forms только что созданный базовый стиль для всех этикеток с вашими пользовательскими шрифтами.

Чтобы включить пользовательские шрифты в iOS и Android, выполните руководство по использованию пользовательских шрифтов на iOS и Android с почтой Xamarin.Forms, написанной Джеральдом.

Объявите стиль в разделе ресурсов файла App.xaml. Это сделает все стили глобально видимыми.

Из вышеперечисленного Gerald нам нужно использовать свойство StyleId, но оно не является связующим свойством, поэтому для его использования в Style Setter нам нужно создать Attachable Property для него:

```
public static class FontHelper
{
    public static readonly BindableProperty StyleIdProperty =
        BindableProperty.CreateAttached(
```

```
            propertyName: nameof(Label.StyleId),
            returnType: typeof(String),
            declaringType: typeof(FontHelper),
            defaultValue: default(String),
            propertyChanged: OnItemTappedChanged);

    public static String GetStyleId(BindableObject bindable) =>
(String)bindable.GetValue(StyleIdProperty);

    public static void SetStyleId(BindableObject bindable, String value) =>
bindable.SetValue(StyleIdProperty, value);


    public static void OnItemTappedChanged(BindableObject bindable, object oldValue, object
newValue)
    {
        var control = bindable as Element;
        if (control != null)
        {
            control.StyleId = GetStyleId(control);
        }
    }
}
```

Затем добавьте стиль в ресурс App.xaml:

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:h="clr-namespace:My.Helpers"
             x:Class="My.App">

  <Application.Resources>

    <ResourceDictionary>
        <Style x:Key="LabelStyle" TargetType="Label">
            <Setter Property="FontFamily" Value="Metric Bold" />
            <Setter Property="h:FontHelper.StyleId" Value="Metric-Bold" />
        </Style>
    </ResourceDictionary>

  </Application.Resources>

</Application>
```

Согласно вышеприведенному сообщению, нам нужно создать Custom Renderer for Label, который наследуется от платформы LabelRenderer On Android.

```
internal class LabelExRenderer : LabelRenderer
{
    protected override void OnElementChanged(ElementChangedEventArgs<Label> e)
    {
        base.OnElementChanged(e);
        if (!String.IsNullOrEmpty(e.NewElement?.StyleId))
        {
            var font = Typeface.CreateFromAsset(Forms.Context.ApplicationContext.Assets,
e.NewElement.StyleId + ".ttf");
            Control.Typeface = font;
```

```
        }
    }
}
```

Для платформы iOS не требуются специальные средства визуализации.

Теперь вы можете получить стиль в разметке вашей страницы:

Для конкретной метки

```
<Label Text="Some text" Style={StaticResource LabelStyle} />
```

Или примените стиль ко всем меткам на странице, создав стиль на основе LabesStyle

```
<!-- language: xaml -->

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="My.MainPage">

  <ContentPage.Resources>

    <ResourceDictionary>
        <Style TargetType="Label" BasedOn={StaticResource LabelStyle}>
        </Style>
    </ResourceDictionary>

  </ContentPage.Resources>

  <Label Text="Some text" />

</ContentPage>
```

Прочитайте Пользовательские шрифты в стилях онлайн: https://riptutorial.com/ru/xamarin-forms/topic/4854/пользовательские-шрифты-в-стилях

глава 26: Последствия

Вступление

Эффекты упрощают индивидуальные настройки платформы. Когда необходимо изменить свойства Xamarin Forms Control, можно использовать эффекты. Когда необходимо переопределить методы управления Xamarin Forms, можно использовать пользовательские рендереры

Examples

Добавление специфичного для платформы эффекта для элемента управления Entry

1. Создайте новое приложение Xamarin Forms с помощью файла PCL -> Новое решение -> Мультиплатформенное приложение -> Xamarin Forms -> Forms App; Назовите проект как `EffectsDemo`
2. В рамках проекта iOS добавьте новый класс `Effect` который наследует класс `PlatformEffect` и переопределяет методы `OnAttached` , `OnDetached` и `OnElementPropertyChanged` Обратите внимание на два атрибута `ResolutionGroupName` и `ExportEffect` , они необходимы для использования этого эффекта из проекта PCL / shared.

- `OnAttached` - это метод, в котором логика настройки
- `OnDetached` - это метод, в котором происходит очистка и де-регистрация
- `OnElementPropertyChanged` - это метод, который запускается при изменении свойств разных элементов. Чтобы определить правильное свойство, проверьте правильность изменения свойства и добавьте свою логику. В этом примере `OnFocus` даст `Blue` цвет, а `OutofFocus` предоставит `Red` Color

```
using System;
using EffectsDemo.iOS;
using UIKit;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;

[assembly: ResolutionGroupName("xhackers")]
[assembly: ExportEffect(typeof(FocusEffect), "FocusEffect")]
namespace EffectsDemo.iOS
{
public class FocusEffect : PlatformEffect
{
public FocusEffect()
{
}
```

```
 UIColor backgroundColor;
 protected override void OnAttached()
 {
     try
     {
         Control.BackgroundColor = backgroundColor = UIColor.Red;
     }
     catch (Exception ex)
     {
         Console.WriteLine("Cannot set attacked property" + ex.Message);
     }
 }

 protected override void OnDetached()
 {
     throw new NotImplementedException();
 }

 protected override void
OnElementPropertyChanged(System.ComponentModel.PropertyChangedEventArgs args)
 {
     base.OnElementPropertyChanged(args);

     try
     {
         if (args.PropertyName == "IsFocused")
         {
             if (Control.BackgroundColor == backgroundColor)
             {
                 Control.BackgroundColor = UIColor.Blue;
             }
             else
             {
                 Control.BackgroundColor = backgroundColor;
             }
         }
     }
     catch (Exception ex)
     {
         Console.WriteLine("Cannot set property " + ex.Message);
     }
 }
```

}}

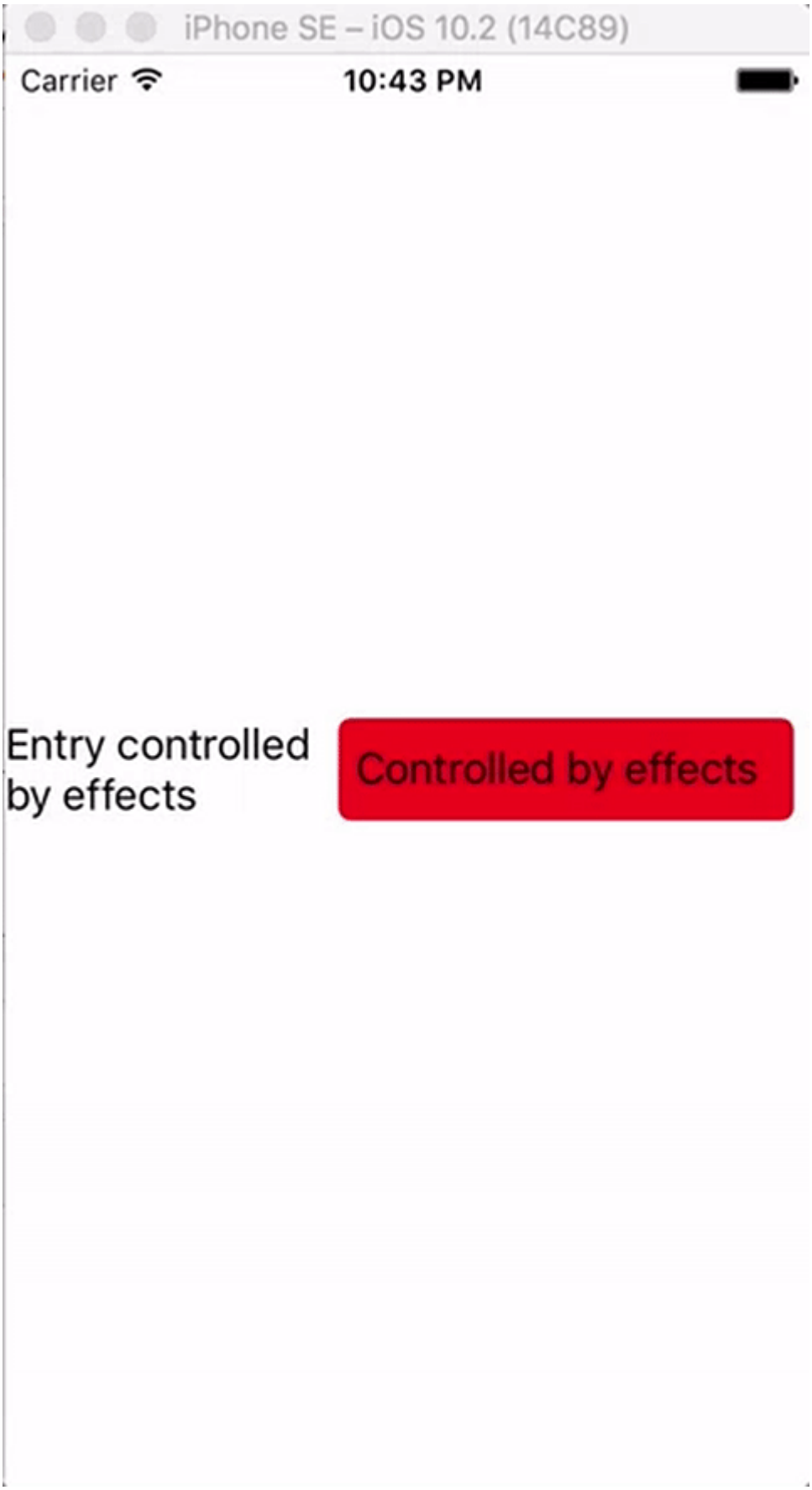
3. Чтобы использовать этот эффект в приложении, в проекте `PCL` создайте новый класс с именем `FocusEffect` который наследуется от `RoutingEffect` . Это необходимо для того, чтобы PCL создавал экземпляр конкретной реализации эффекта. Пример кода ниже:

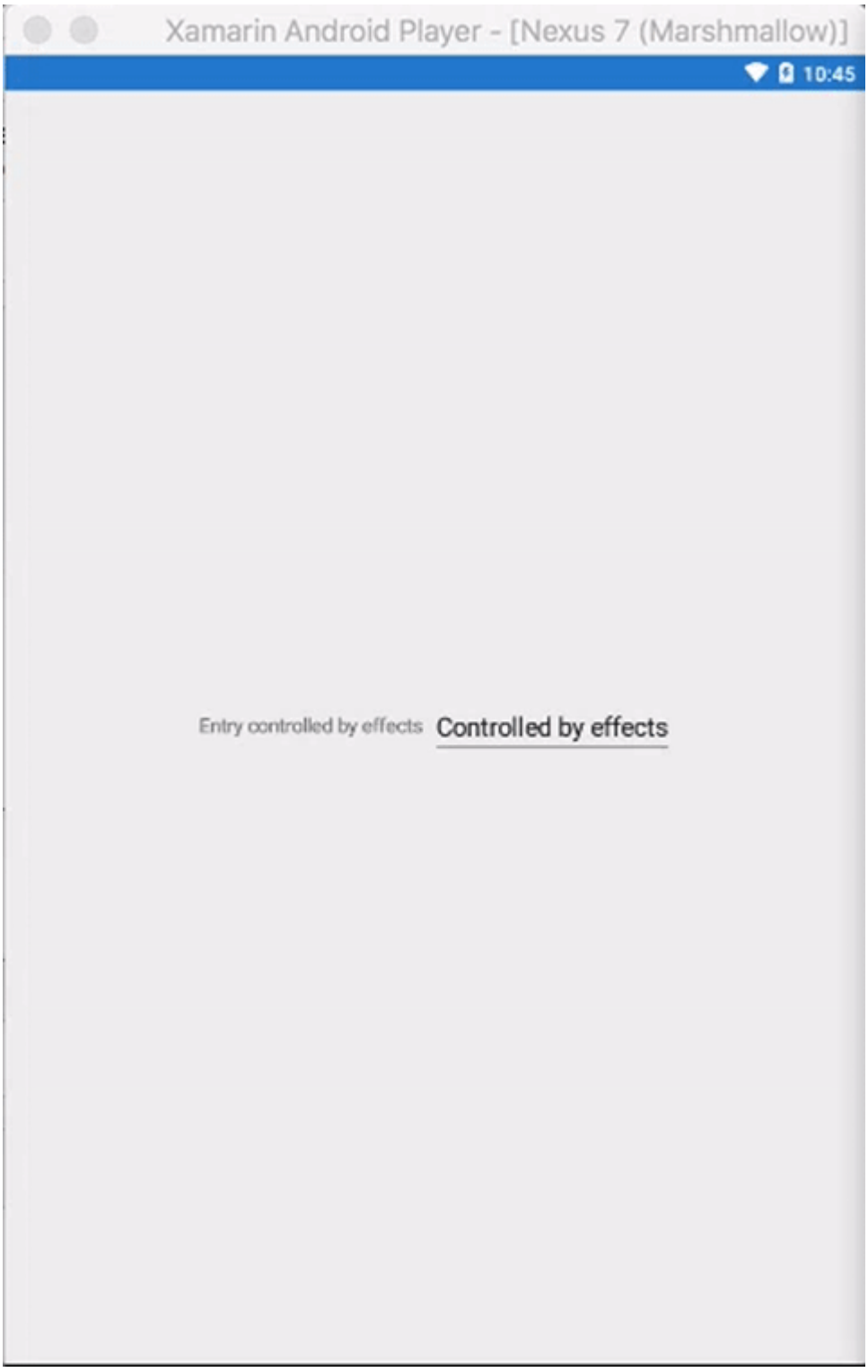
```
using Xamarin.Forms;
namespace EffectsDemo
{
    public class FocusEffect : RoutingEffect
    {
        public FocusEffect() : base("xhackers.FocusEffect")
        {
        }
    }
```

```
}
```

4. Добавить эффект в элемент управления `Entry` в XAML

```
<?xml version="1.0" encoding="utf-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-
namespace:EffectsDemo" x:Class="EffectsDemo.EffectsDemoPage">
<StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Center">
<Label Text="Effects Demo" HorizontalOptions="StartAndExpand" VerticalOptions="Center"
></Label>
<Entry Text="Controlled by effects" HorizontalOptions="FillAndExpand"
VerticalOptions="Center">
    <Entry.Effects>
        <local:FocusEffect>
        </local:FocusEffect>
    </Entry.Effects>
</Entry>
</StackLayout>
</ContentPage>
```


iPhone SE – iOS 10.2 (14C89)
Carrier
10:43 PM
Entry controlled by effects
Controlled by effects



Поскольку эффект был реализован только в версии iOS, когда приложение запускается в `iOS Simulator` при фокусировке изменений цвета фона `Entry` и ничего не происходит в `Android Emulator` поскольку `Effect` не был создан в проекте `Droid`

Прочитайте Последствия онлайн: https://riptutorial.com/ru/xamarin-forms/topic/9252/последствия

глава 27: Почему Xamarin формирует и когда использовать Xamarin Forms

замечания

Вы можете обратиться к официальной документации Xamarin Forms, чтобы узнать больше:

https://www.xamarin.com/forms

Examples

Почему Xamarin формирует и когда использовать Xamarin Forms

Xamarin становится все более популярным - трудно решить, когда использовать Xamarin.Forms и когда Xamarin.Platform (так Xamarin.iOS и Xamarin.Android).

Прежде всего, вы должны знать, для каких приложений вы можете использовать Xamarin.Forms:

1. Прототипы - для визуализации того, как ваше приложение будет выглядеть на разных устройствах.
2. Приложения, которые не требуют функциональных возможностей платформы (например, API), - но здесь обратите внимание на то, что Xamarin работает изо всех сил, чтобы обеспечить как можно больше кросс-платформенной совместимости.
3. Приложения, где разделение кода имеет решающее значение - важнее, чем пользовательский интерфейс.
4. Приложения, в которых данные отображаются важнее, чем расширенные функции

Есть также много других факторов:

1. Кто будет отвечать за разработку приложений - если ваша команда состоит из опытных разработчиков мобильных устройств, они смогут легко обращаться с Xamarin.Forms. Но если у вас есть один разработчик на платформу (собственная разработка), то формы могут быть более сложной задачей.
2. Также обратите внимание, что с Xamarin.Forms вы по-прежнему иногда сталкиваетесь с некоторыми проблемами - платформа Xamarin.Forms все еще улучшается.
3. Быстрое развитие иногда очень важно - чтобы сократить затраты и время, вы можете

решить использовать формы.

4. При разработке корпоративных приложений без какой-либо расширенной функциональности лучше использовать Xamarin.Forms - он позволяет обмениваться кодом режима, а не событием в мобильной области, но в целом. Некоторые части кода могут совместно использоваться на многих платформах.

Вы не должны использовать Xamarin.Forms, когда:

1. Вам необходимо создать пользовательские функции и получить доступ к API-интерфейсам платформы
2. Вы должны создать пользовательский интерфейс для мобильного приложения
3. Когда некоторые функции не готовы для Xamarin.Forms (например, некоторые специфические действия на мобильном устройстве)
4. Ваша команда состоит из разработчиков мобильных приложений на платформе (мобильная разработка на Java и / или Swift / Objective C)

Прочитайте Почему Xamarin формирует и когда использовать Xamarin Forms онлайн: https://riptutorial.com/ru/xamarin-forms/topic/6869/почему-xamarin-формирует-и-когда-использовать-xamarin-forms

глава 28: Работа с картами

замечания

Если вы собираетесь запустить свой проект на другом компьютере, вам нужно будет создать для него новый ключ API, поскольку отпечатки пальцев SHA-1 не будут совпадать для разных машин сборки.

Вы можете изучить проект, описанный в примере *Добавление карты в Xamarin.Forms* здесь

Examples

Добавление карты в Xamarin.Forms (Xamarin Studio)

Вы можете просто использовать собственные API карт на каждой платформе с помощью Xamarin Forms. Все, что вам нужно, это загрузить пакет *Xamarin.Forms.Maps* из nuget и установить его в каждый проект (включая проект PCL).

Инициализация карт

Прежде всего, вы должны добавить этот код в свои проекты на платформе. Для этого вам нужно добавить `Xamarin.FormsMaps.Init` метода `Xamarin.FormsMaps.Init` , как в приведенных ниже примерах.

проект iOS

Файл AppDelegate.cs

```
[Register("AppDelegate")]
public partial class AppDelegate : Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
{
    public override bool FinishedLaunching(UIApplication app, NSDictionary options)
    {
        Xamarin.Forms.Forms.Init();
        Xamarin.FormsMaps.Init();

        LoadApplication(new App());

        return base.FinishedLaunching(app, options);
    }
}
```

Android-проект

Файл MainActivity.cs

```
[Activity(Label = "MapExample.Droid", Icon = "@drawable/icon", Theme = "@style/MyTheme",
MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize |
ConfigChanges.Orientation)]
public class MainActivity : Xamarin.Forms.Platform.Android.FormsAppCompatActivity
{
    protected override void OnCreate(Bundle bundle)
    {
        TabLayoutResource = Resource.Layout.Tabbar;
        ToolbarResource = Resource.Layout.Toolbar;

        base.OnCreate(bundle);

        Xamarin.Forms.Forms.Init(this, bundle);
        Xamarin.FormsMaps.Init(this, bundle);

        LoadApplication(new App());
    }
}
```

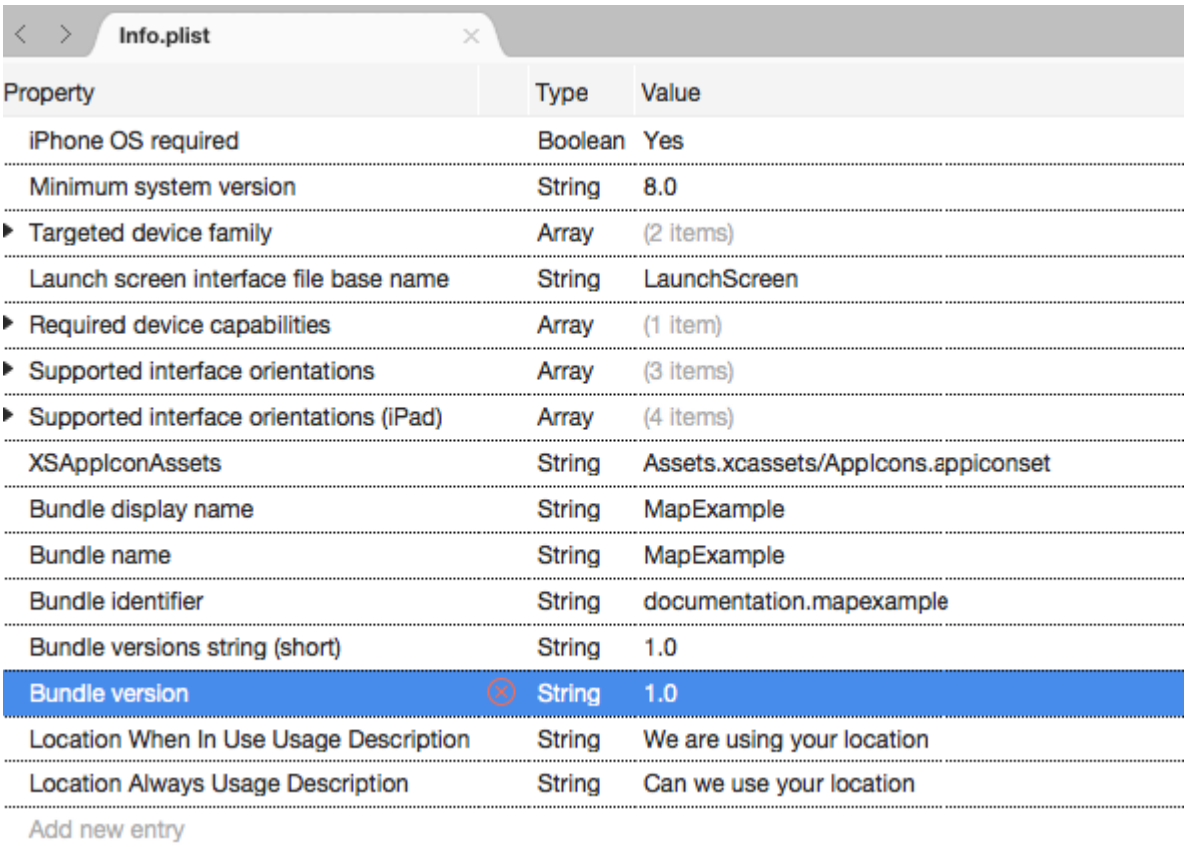
Конфигурация платформы

Дополнительные шаги настройки требуются на некоторых платформах перед отображением карты.

проект iOS

В проекте iOS вам просто нужно добавить 2 файла в файл *Info.plist* :

- `NSLocationWhenInUseUsageDescription` *string* со значением `We are using your location`
- `NSLocationAlwaysUsageDescription` *string* со значением `Can we use your location`



Android-проект

Чтобы использовать Карты Google, вы должны создать ключ API и добавить его в свой проект. Следуйте приведенной ниже инструкции, чтобы получить этот ключ:

1. (Необязательно) Найдите, где находится инструмент инструмента keytool (по умолчанию это `/System/Library/Frameworks/JavaVM.framework/Versions/Current/Commands`)

2. (Необязательно) Откройте терминал и перейдите в папку вашего ключа:

   ```
   cd /System/Library/Frameworks/JavaVM.framework/Versions/Current/Commands
   ```

3. Выполните следующую команду keytool:

   ```
   keytool -list -v -keystore "/Users/[USERNAME]/.local/share/Xamarin/Mono for
   Android/debug.keystore" -alias androiddebugkey -storepass android -keypass android
   ```

   Где [USERNAME], очевидно, ваша текущая папка пользователя. Вы должны получить что-то подобное в выводе:

   ```
   Alias name: androiddebugkey
   Creation date: Jun 30, 2016
   Entry type: PrivateKeyEntry
   Certificate chain length: 1
   Certificate[1]:
   Owner: CN=Android Debug, O=Android, C=US
   Issuer: CN=Android Debug, O=Android, C=US
   ```

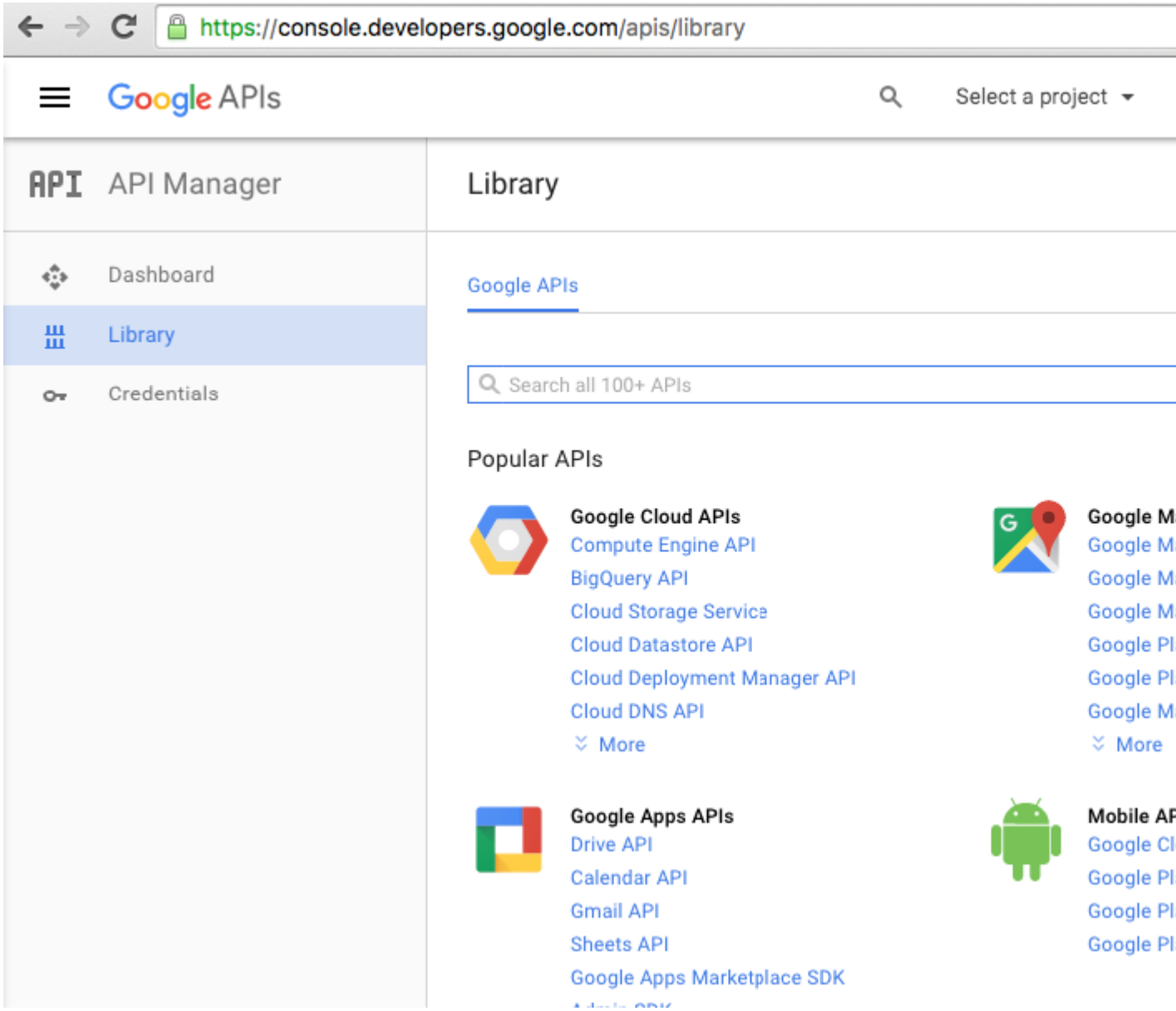
```
Serial number: 4b5ac934
Valid from: Thu Jun 30 10:22:00 EEST 2016 until: Sat Jun 23 10:22:00 EEST 2046
Certificate fingerprints:
         MD5:  4E:49:A7:14:99:D6:AB:9F:AA:C7:07:E2:6A:1A:1D:CA
         SHA1: 57:A1:E5:23:CE:49:2F:17:8D:8A:EA:87:65:44:C1:DD:1C:DA:51:95
         SHA256:
70:E1:F3:5B:95:69:36:4A:82:A9:62:F3:67:B6:73:A4:DD:92:95:51:44:E3:4C:3D:9E:ED:99:03:09:9F:90:3F

         Signature algorithm name: SHA256withRSA
         Version: 3
```

4. Все, что нам нужно в этом выпуске, - это отпечаток сертификата SHA1. В нашем случае это равно:

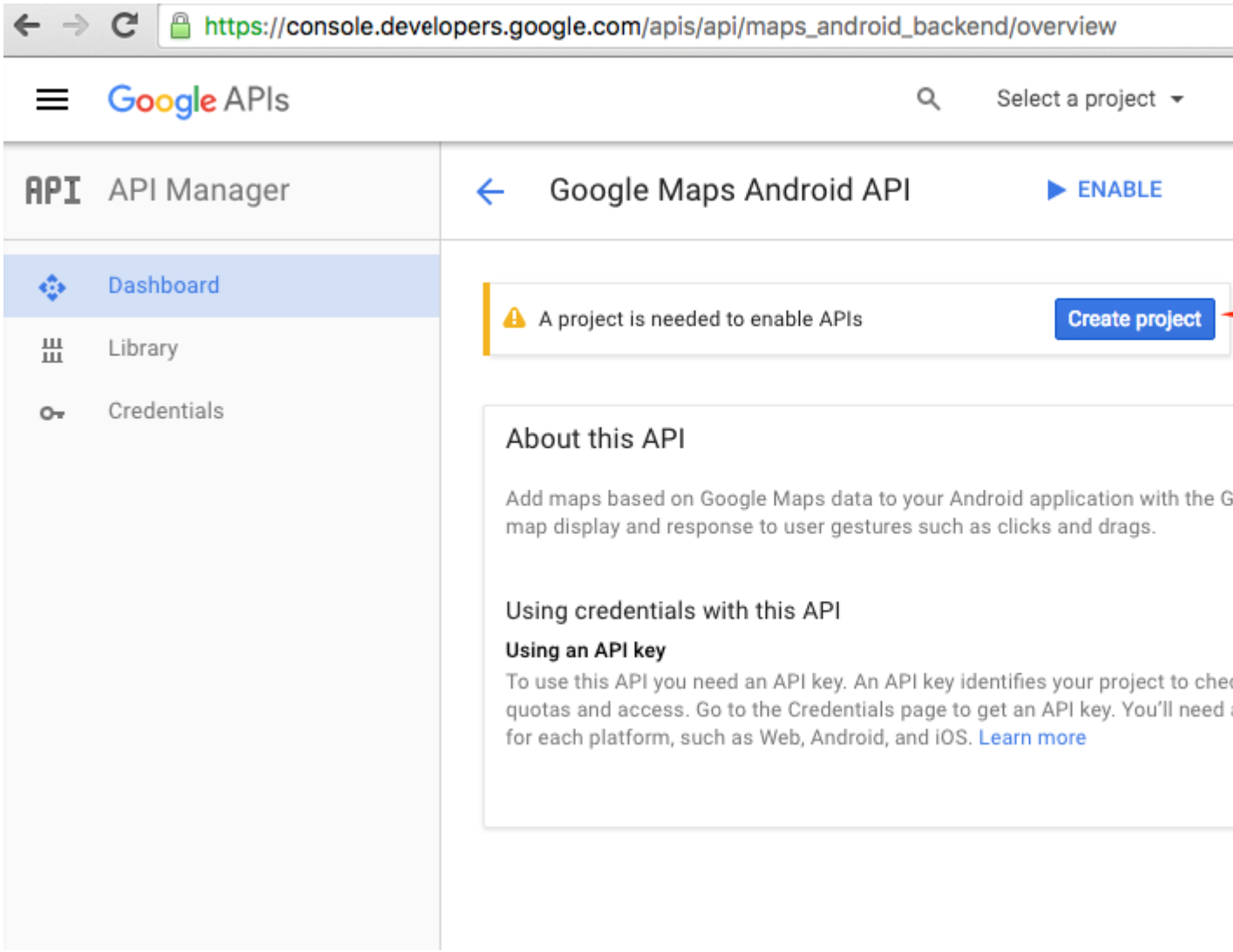
   ```
   57:A1:E5:23:CE:49:2F:17:8D:8A:EA:87:65:44:C1:DD:1C:DA:51:95
   ```

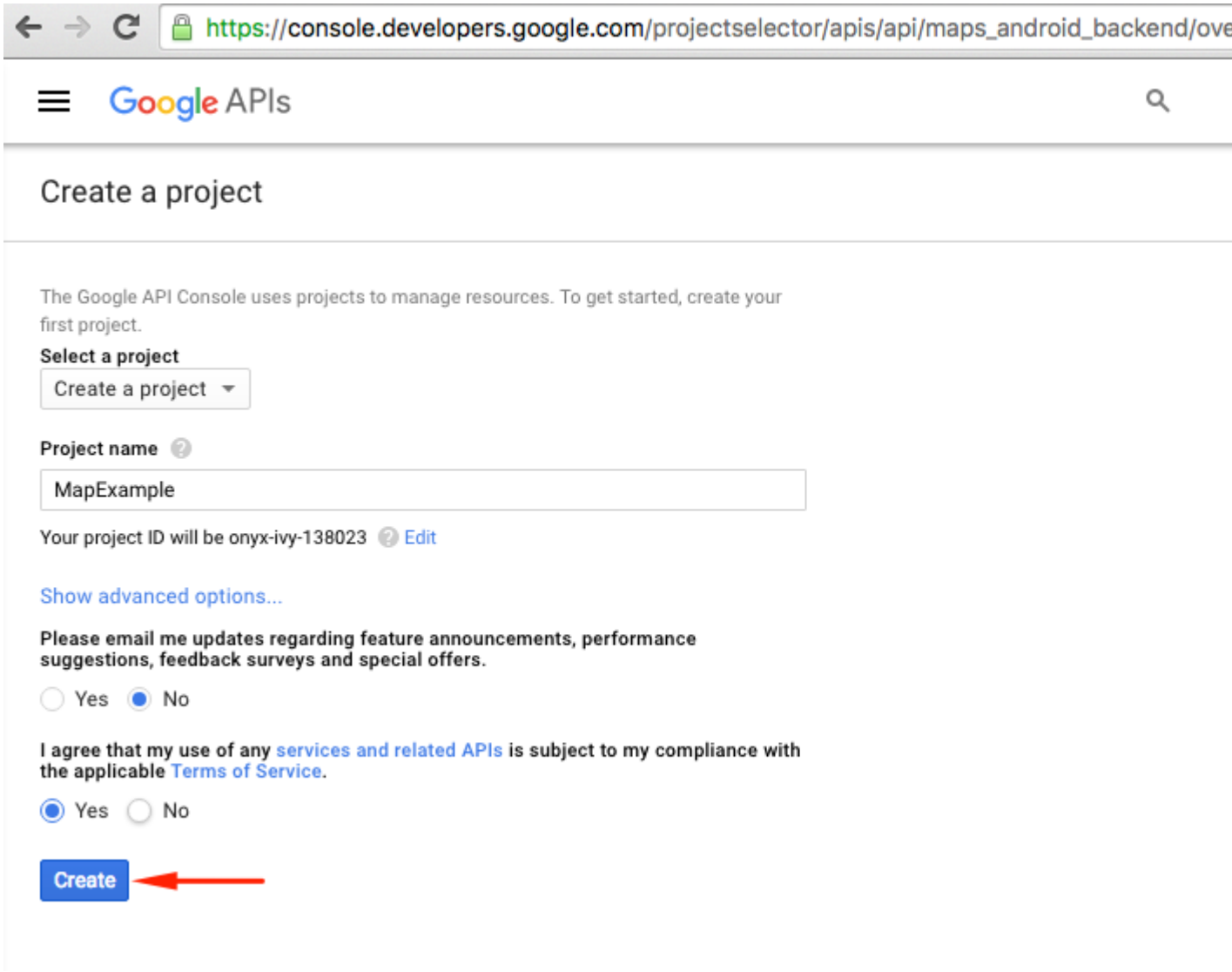
   Скопируйте или сохраните где-нибудь этот ключ. Это нам понадобится позже.

5. Перейдите в Google Developers Console , в нашем случае мы должны добавить API Android для Google Maps , поэтому выберите:

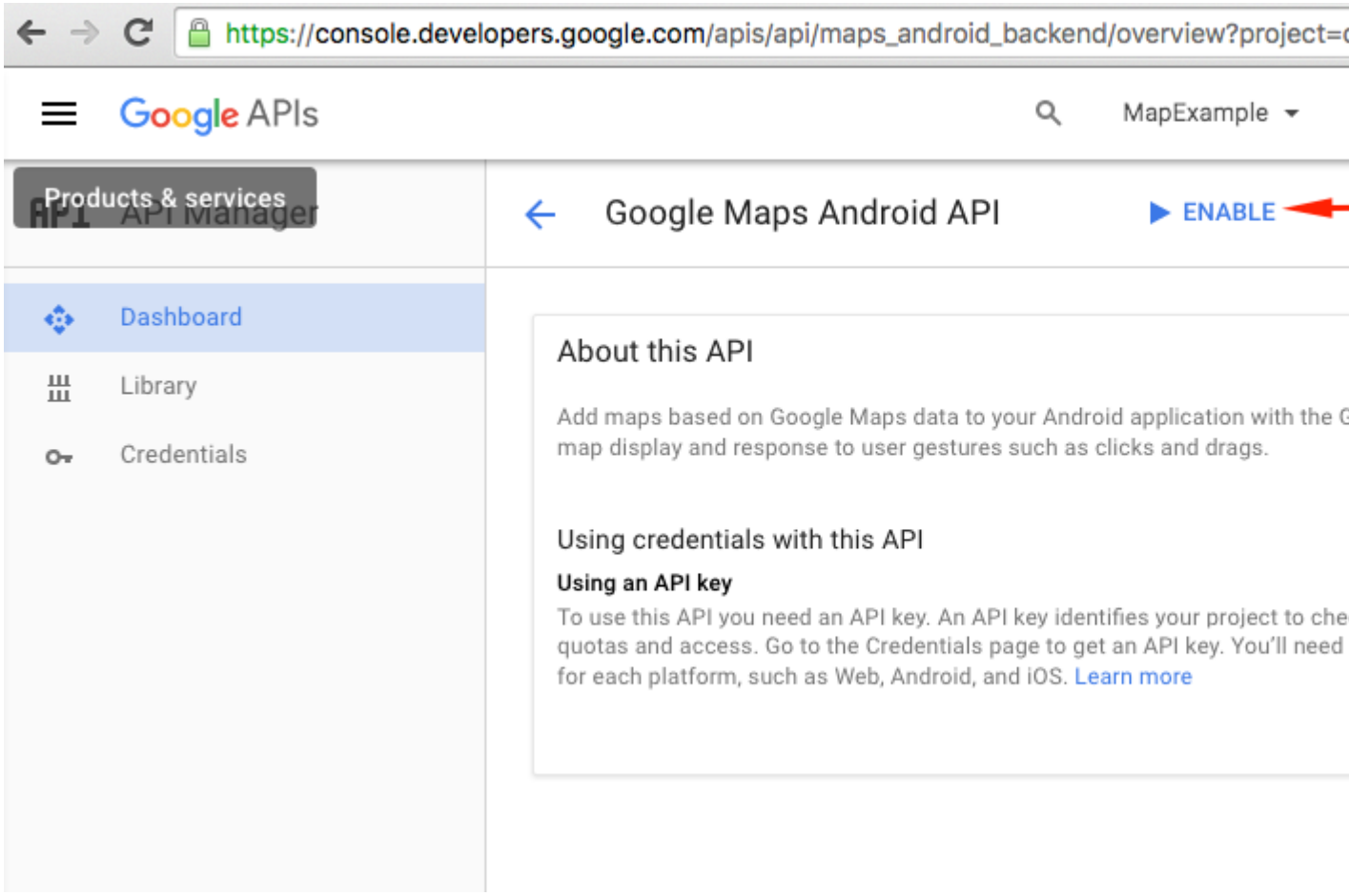


6. Google попросит вас создать проект для включения API, следовать этому совету и создать проект:

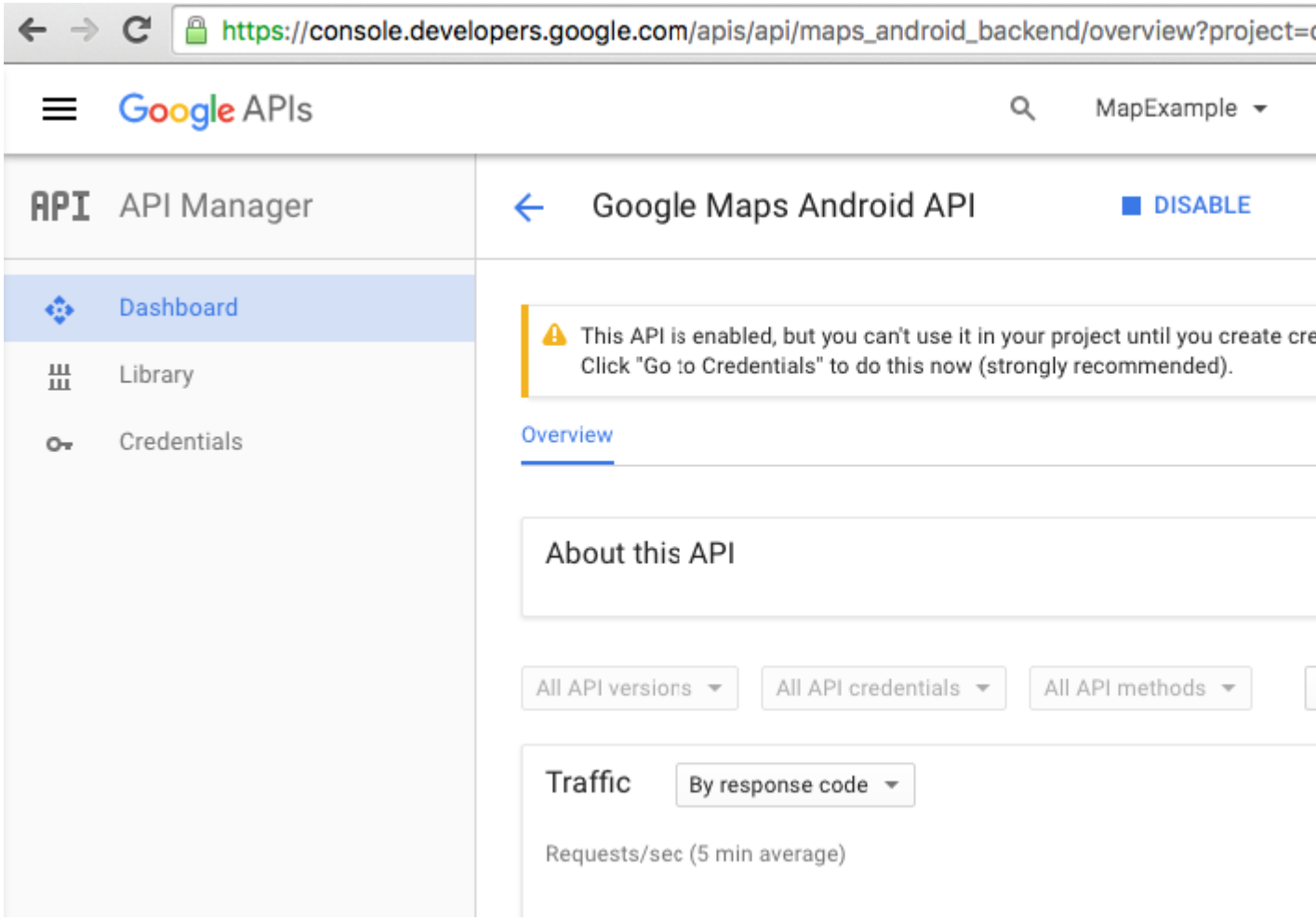

https://console.developers.google.com/apis/api/maps_android_backend/overview
Google APIs
Select a project
API Manager
Google Maps Android API
ENABLE
Dashboard
Library
Credentials
A project is needed to enable APIs
Create project
About this API
Add maps based on Google Maps data to your Android application with the G
map display and response to user gestures such as clicks and drags.
Using credentials with this API
Using an API key
To use this API you need an API key. An API key identifies your project to chec
quotas and access. Go to the Credentials page to get an API key. You'll need
for each platform, such as Web, Android, and iOS. Learn more



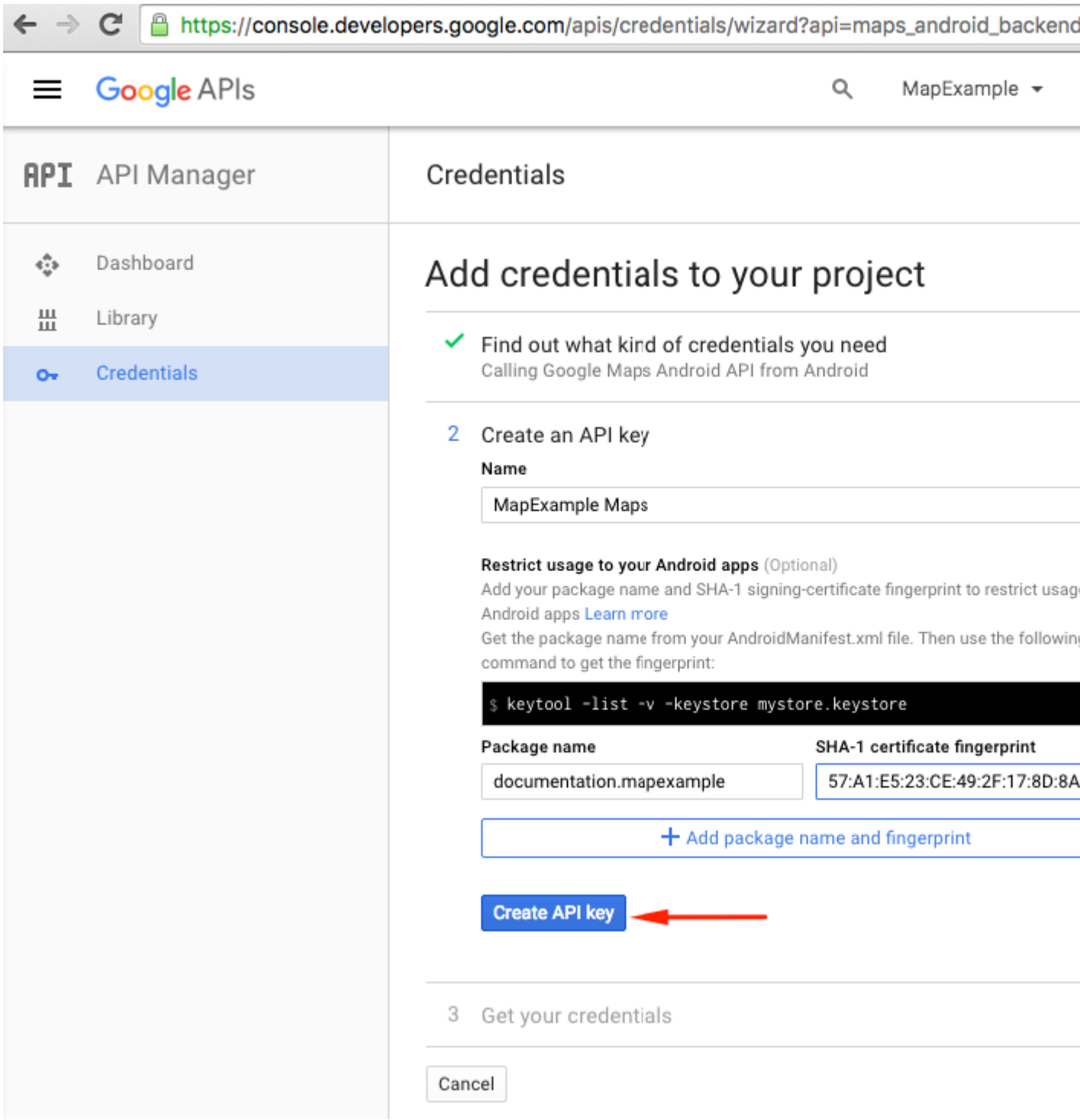
7. Включите API Карт Google для своего проекта:



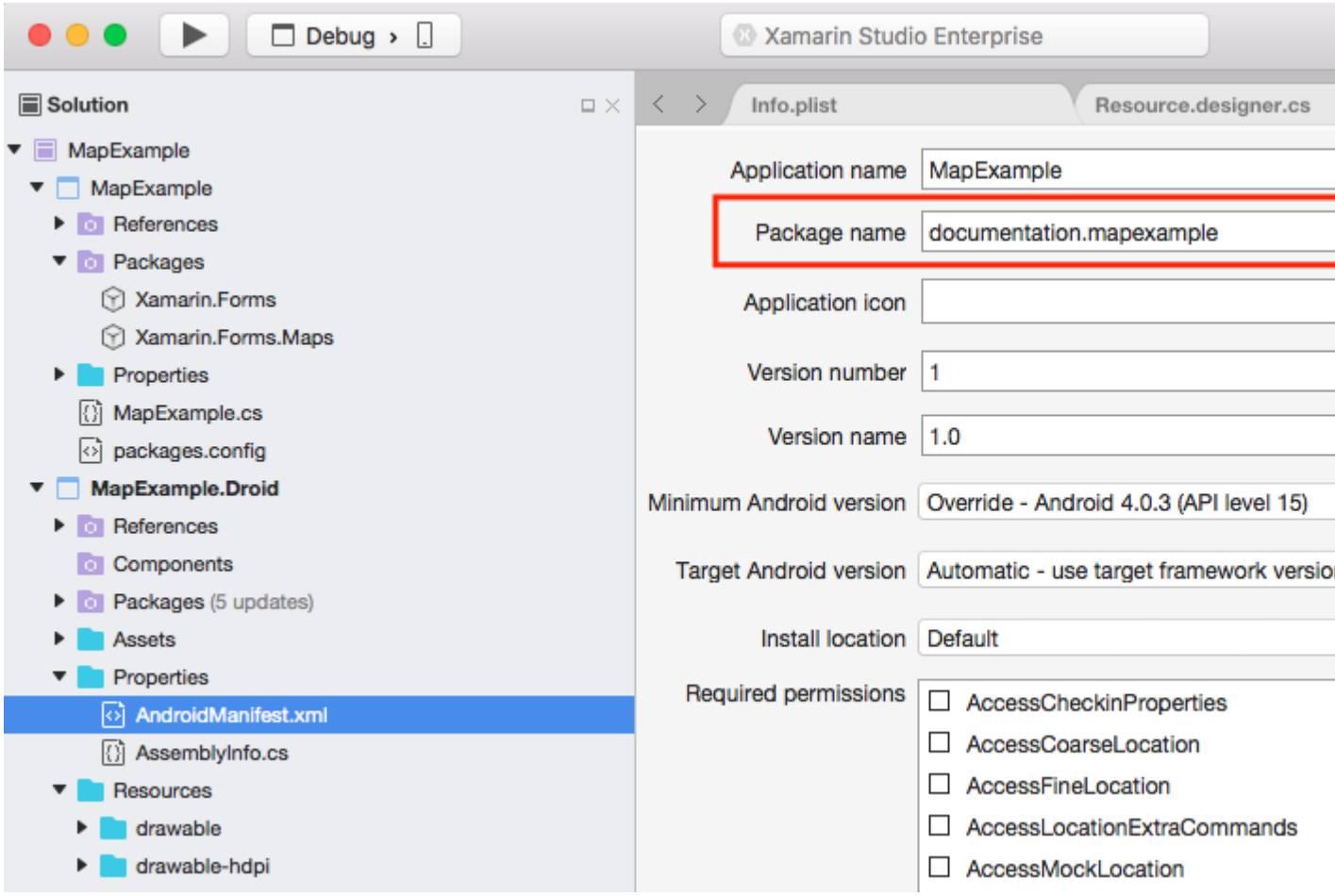
После того, как вы включили api, вам нужно создать учетные данные для своего приложения. Следуйте этому совету:



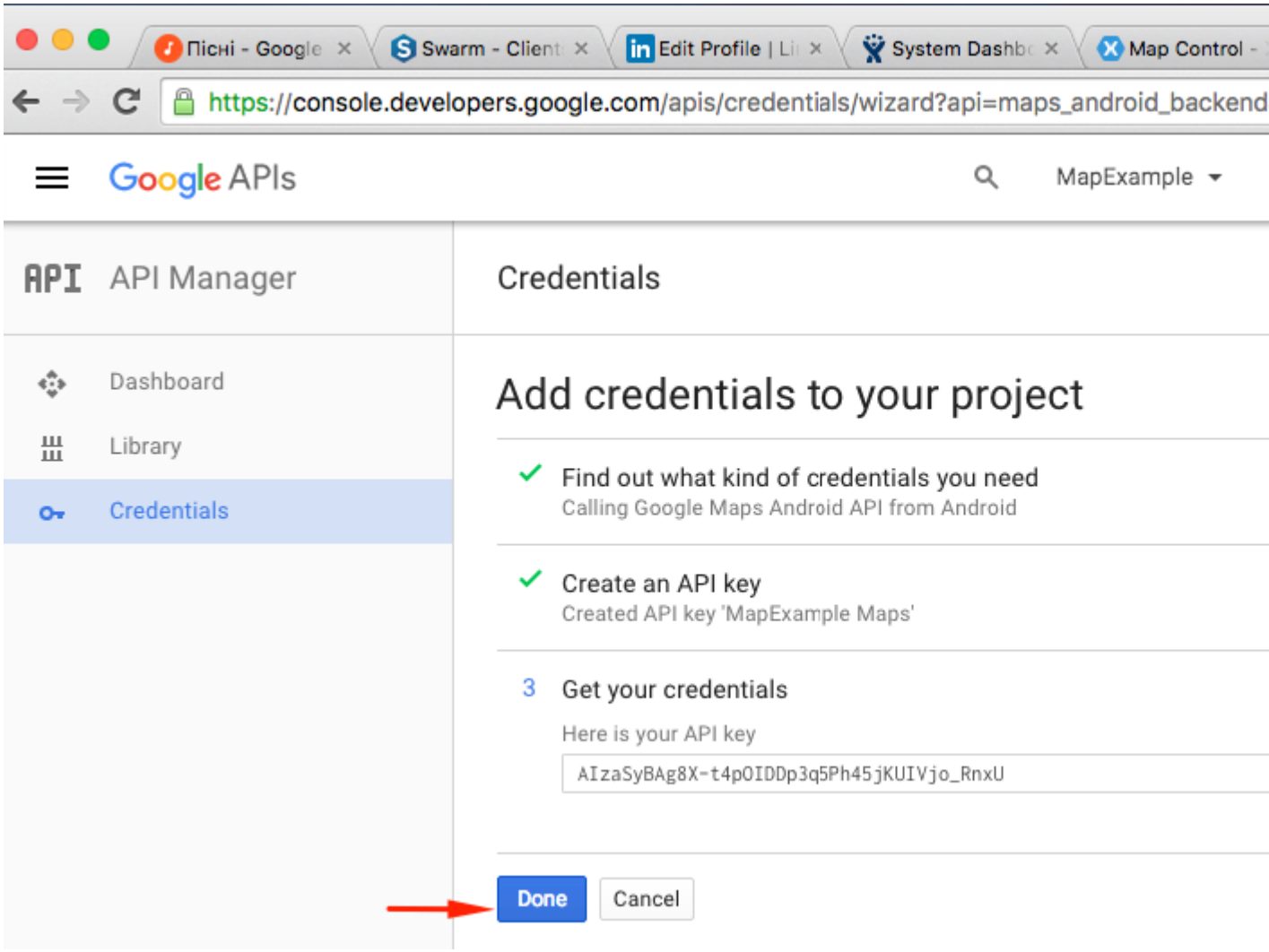
8. На следующей странице выберите платформу Android, нажмите «Какие мандаты мне нужны?». , введите имя для вашего ключа API, нажмите «Добавить имя пакета и отпечаток пальца», введите имя своего пакета и ваш отпечаток SHA1 с шага 4 и, наконец, создайте ключ API:



Чтобы найти название своего пакета в Xamarin Studio, перейдите в свое решение . Droid -> AndroidManifest.xml:



9. После создания скопируйте новый ключ API (не забудьте нажать кнопку «Готово» после) и вставьте его в файл `AndroidManifest.xml` :



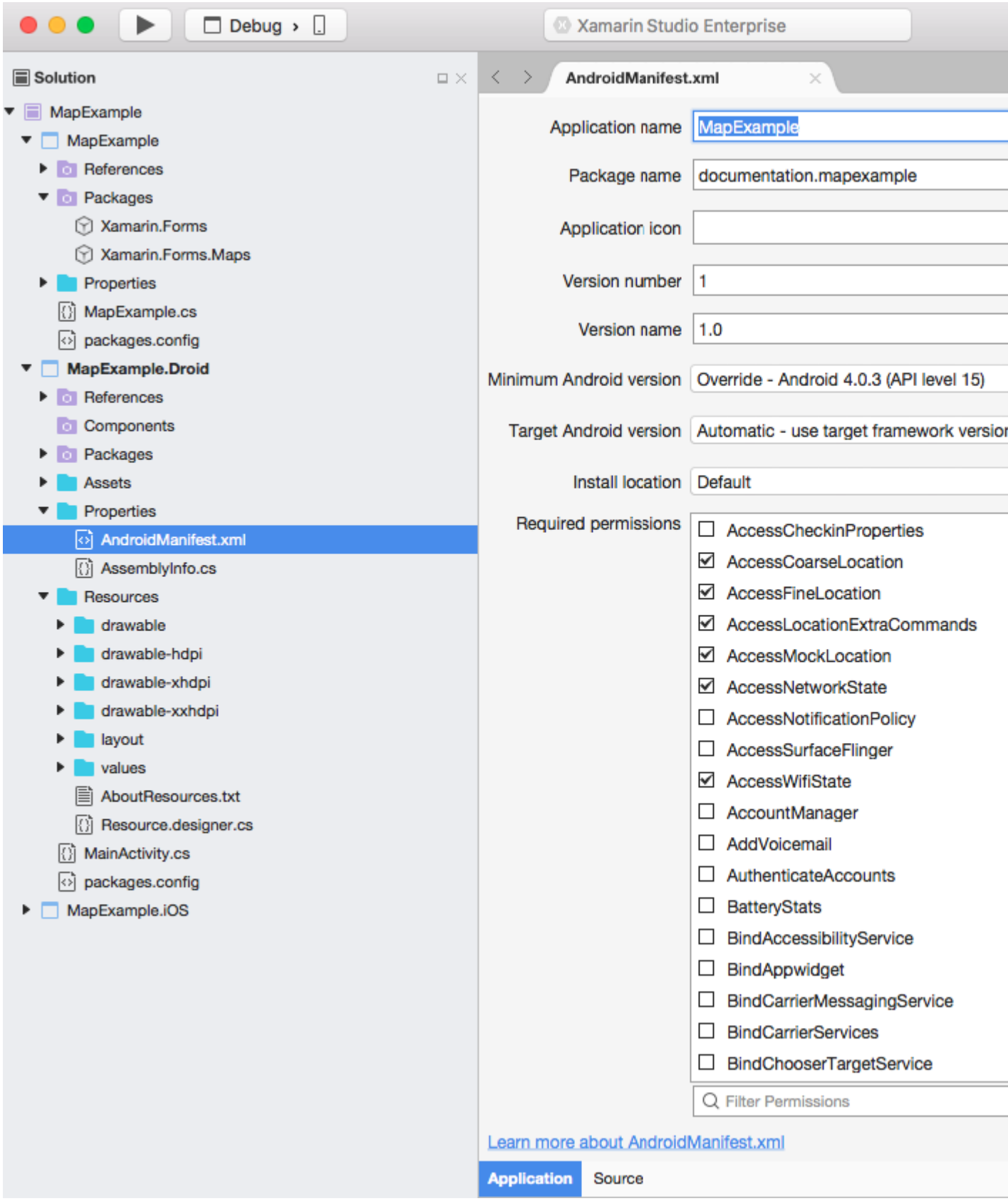
Файл AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    package="documentation.mapexample">
    <uses-sdk
        android:minSdkVersion="15" />
    <application
        android:label="MapExample">
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyBAg8X-t4pOIDDp3q5Ph45jKUIVjo_RnxU" />
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
    </application>
</manifest>
```

Вам также необходимо включить некоторые разрешения в манифесте, чтобы включить некоторые дополнительные функции:

- Доступ к грубому местоположению

- Доступ к Fine Location
- Доступ к дополнительным командам
- Доступ к локальному местоположению
- Состояние сети доступа
- Доступ к Wi-Fi
- интернет



Хотя для загрузки данных Карты требуется два последних разрешения. Узнайте о разрешениях на Android, чтобы узнать больше. Это все шаги для настройки Android.

> *Примечание* . Если вы хотите запустить приложение на симуляторе Android, вам необходимо установить на него сервисы Google Play. Следуйте этому руководству, чтобы установить Play Services на Xamarin Android Player.

Если вы не можете найти обновление сервисов Google Play после установки магазина воспроизведения, вы можете обновить его непосредственно из своего приложения, где у вас есть зависимость от служб карт

Добавление карты

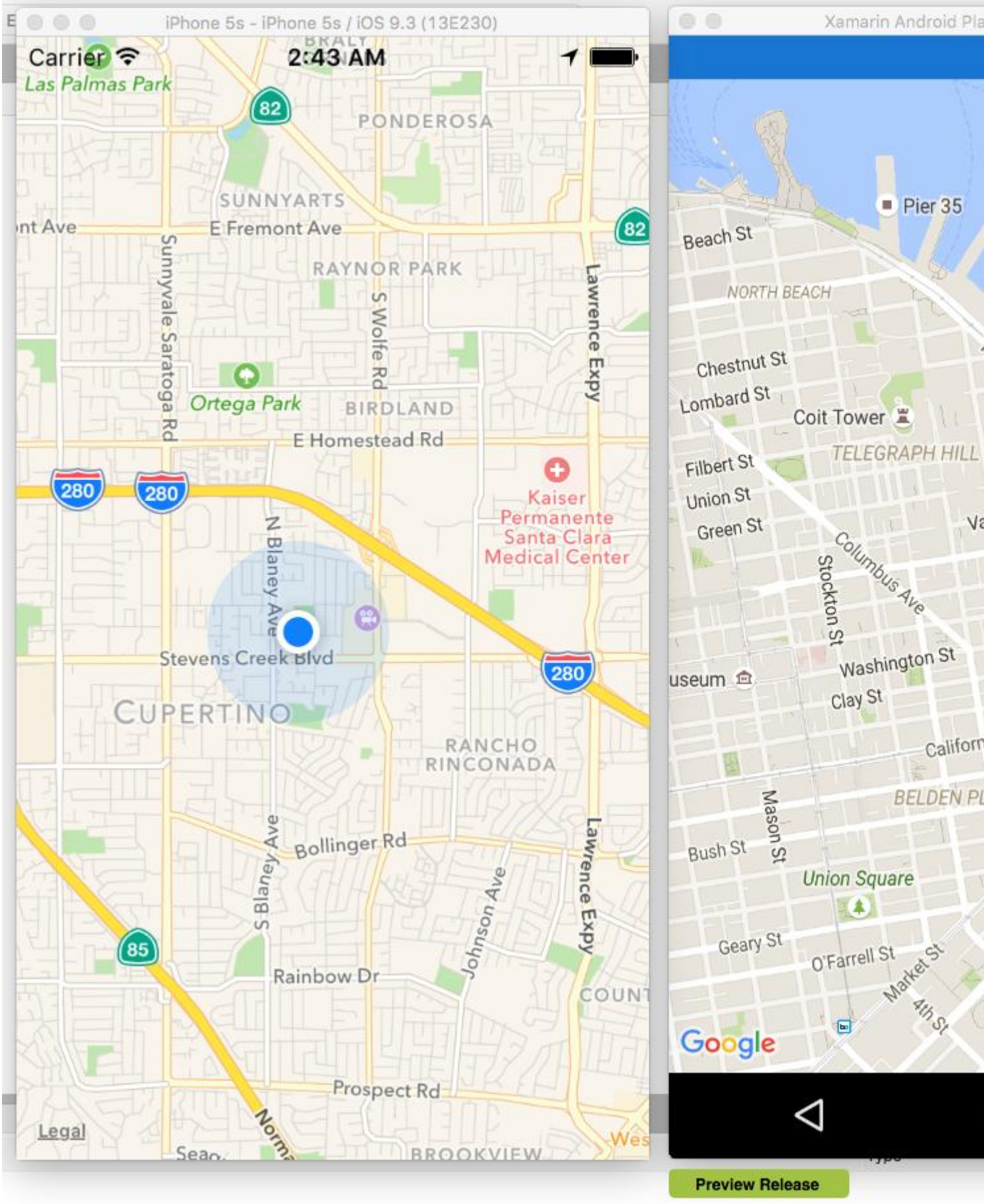
Добавление вида карты в ваш кроссплатформенный проект довольно прост. Вот пример того, как вы можете это сделать (я использую проект PCL без XAML).

Проект PCL

Файл MapExample.cs

```
public class App : Application
{
    public App()
    {
        var map = new Map();
        map.IsShowingUser = true;

        var rootPage = new ContentPage();
        rootPage.Content = map;

        MainPage = rootPage;
    }
}
```

Это все. Теперь, если вы запустите приложение на iOS или Android, оно покажет вам вид карты:



Прочитайте Работа с картами онлайн: https://riptutorial.com/ru/xamarin-forms/topic/3917/работа-с-картами

глава 29: Работа с локальными базами данных

Examples

Использование SQLite.NET в общем проекте

SQLite.NET - это библиотека с открытым исходным кодом, которая позволяет добавлять поддержку локальных баз данных с использованием `SQLite` версии 3 в проекте `Xamarin.Forms` .

В приведенных ниже шагах показано, как включить этот компонент в общий проект `Xamarin.Forms` :

1. Загрузите последнюю версию класса SQLite.cs и добавьте ее в общий проект.
2. Каждая таблица, которая будет включена в базу данных, должна быть смоделирована как класс в общем проекте. Таблица определяется добавлением по меньшей мере двух атрибутов в классе: `Table` (для класса) и `PrimaryKey` (для свойства).

В этом примере новый класс с именем `Song` добавляется в общий проект, определяемый следующим образом:

```
using System;
using SQLite;

namespace SongsApp
{
    [Table("Song")]
    public class Song
    {
        [PrimaryKey]
        public string ID { get; set; }
        public string SongName { get; set; }
        public string SingerName { get; set; }
    }
}
```

3. Затем добавьте новый класс под названием `Database` , который наследуется от класса `SQLiteConnection` (включенного в SQLite.cs). В этом новом классе определяется код доступа к базе данных, создания таблиц и операций CRUD для каждой таблицы. Пример кода показан ниже:

```
using System;
using System.Linq;
using System.Collections.Generic;
using SQLite;
```

```
namespace SongsApp
{
    public class BaseDatos : SQLiteConnection
    {
        public BaseDatos(string path) : base(path)
        {
            Initialize();
        }

        void Initialize()
        {
            CreateTable<Song>();
        }

        public List<Song> GetSongs()
        {
            return Table<Song>().ToList();
        }

        public Song GetSong(string id)
        {
            return Table<Song>().Where(t => t.ID == id).First();
        }

        public bool AddSong(Song song)
        {
            Insert(song);
        }

        public bool UpdateSong(Song song)
        {
            Update(song);
        }

        public void DeleteSong(Song song)
        {
            Delete(song);
        }
    }
}
```

4. Как вы могли видеть на предыдущем шаге, конструктор нашего класса `Database` включает в себя параметр `path` , который представляет местоположение файла, в котором хранится файл базы данных SQLite. Статический объект `Database` может быть объявлен в `App.cs` `path` зависит от платформы:

```
public class App : Application
{
    public static Database DB;

    public App ()
    {
        string dbFile = "SongsDB.db3";

#if __ANDROID__
        string docPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
        var dbPath = System.IO.Path.Combine(docPath, dbFile);
```

```
#else
#if __IOS__
        string docPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
        string libPath = System.IO.Path.Combine(docPath, "..", "Library");
        var dbPath = System.IO.Path.Combine(libPath, dbFile);
#else
        var dbPath = System.IO.Path.Combine(ApplicationData.Current.LocalFolder.Path, dbFile);
#endif
#endif

        DB = new Database(dbPath);

        // The root page of your application
        MainPage = new SongsPage();
    }
}
```

5. Теперь просто вызовите объект `DB` через класс `App` любое время, когда вам нужно выполнить операцию CRUD в таблице `Songs` . Например, чтобы вставить новую `Song` после того, как пользователь нажал кнопку, вы можете использовать следующий код:

```
void AddNewSongButton_Click(object sender, EventArgs a)
{
    Song s = new Song();
    s.ID = Guid.NewGuid().ToString();
    s.SongName = songNameEntry.Text;
    s.SingerName = singerNameEntry.Text;

    App.DB.AddSong(song);
}
```

Работа с локальными базами данных с использованием xamarin.forms в visual studio 2015

Пример SQlite Шаг за шагом Объяснение

1. В приведенных ниже шагах показано, как включить этот компонент в общий проект Xamarin.Forms: добавить пакеты в (pcl, Andriod, Windows, Ios). Добавить ссылки. Нажмите « **Управление пакетами Nuget»** -> нажмите «Обзор», чтобы установить **SQLite.Net.Core- PCL** , **SQLite Net Extensions** после завершения установки проверяют его один раз в ссылках, затем

2. Чтобы добавить Class **Employee.cs** ниже кода

```
using SQLite.Net.Attributes;

    namespace DatabaseEmployeeCreation.SqlLite
    {
        public   class Employee
        {
            [PrimaryKey,AutoIncrement]
            public int Eid { get; set; }
            public string Ename { get; set; }
```

```
            public string Address { get; set; }
            public string phonenumber { get; set; }
            public string email { get; set; }
        }
    }
```

3. Чтобы добавить один интерфейс ISQLite

```
using SQLite.Net;
           //using SQLite.Net;
           namespace DatabaseEmployeeCreation.SqlLite.ViewModel
           {
               public interface ISQLite
               {
                   SQLiteConnection GetConnection();
               }
           }
```

4. Создайте один класс для логики и методов базы данных, приведенных ниже.

используя SQLite.Net; используя System.Collections.Generic; используя System.Linq; использование Xamarin.Forms; namespace DatabaseEmployeeCreation.SqlLite.ViewModel {public class DatabaseLogic {static object locker = new object (); База данных SQLiteConnection;

```
    public DatabaseLogic()
    {
        database = DependencyService.Get<ISQLite>().GetConnection();
        // create the tables
        database.CreateTable<Employee>();
    }

    public IEnumerable<Employee> GetItems()
    {
        lock (locker)
        {
            return (from i in database.Table<Employee>() select i).ToList();
        }
    }

    public IEnumerable<Employee> GetItemsNotDone()
    {
        lock (locker)
        {
            return database.Query<Employee>("SELECT * FROM [Employee]");
        }
    }

    public Employee GetItem(int id)
    {
        lock (locker)
        {
            return database.Table<Employee>().FirstOrDefault(x => x.Eid == id);
        }
    }

    public int SaveItem(Employee item)
    {
```

```
        lock (locker)
        {
            if (item.Eid != 0)
            {
                database.Update(item);
                return item.Eid;
            }
            else
            {
                return database.Insert(item);
            }
        }
    }

    public int DeleteItem(int Eid)
    {
        lock (locker)
        {
            return database.Delete<Employee>(Eid);
        }
    }
}
```

}

5. для создания xaml.forms EmployeeRegistration.xaml

```
    <?xml version="1.0" encoding="utf-8" ?>
    <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
                 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                 x:Class="DatabaseEmployeeCreation.SqlLite.EmployeeRegistration"
      Title="{Binding Name}" >
      <StackLayout VerticalOptions="StartAndExpand" Padding="20">

        <Label Text="Ename" />
        <Entry x:Name="nameEntry" Text="{Binding Ename}"/>
        <Label Text="Address" />
        <Editor x:Name="AddressEntry" Text="{Binding Address}"/>
        <Label Text="phonenumber" />
        <Entry x:Name="phonenumberEntry" Text="{Binding phonenumber}"/>
        <Label Text="email" />
        <Entry x:Name="emailEntry" Text="{Binding email}"/>

        <Button Text="Add" Clicked="addClicked"/>

       <!-- <Button Text="Delete" Clicked="deleteClicked"/>-->

        <Button Text="Details" Clicked="DetailsClicked"/>

        <!--  <Button Text="Edit" Clicked="speakClicked"/>-->

      </StackLayout>
    </ContentPage>
```

EmployeeRegistration.cs

```
    using DatabaseEmployeeCreation.SqlLite.ViewModel;
    using DatabaseEmployeeCreation.SqlLite.Views;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;

namespace DatabaseEmployeeCreation.SqlLite
{
    public partial class EmployeeRegistration : ContentPage
    {
        private int empid;
        private Employee obj;

        public EmployeeRegistration()
        {
            InitializeComponent();

        }

        public EmployeeRegistration(Employee obj)
        {
            this.obj = obj;
            var eid = obj.Eid;
            Navigation.PushModalAsync(new EmployeeRegistration());
            var Address = obj.Address;
            var email = obj.email;
            var Ename = obj.Ename;
            var phonenumber = obj.phonenumber;
            AddressEntry. = Address;
            emailEntry.Text = email;
            nameEntry.Text = Ename;

            //AddressEntry.Text = obj.Address;
            //emailEntry.Text = obj.email;
            //nameEntry.Text = obj.Ename;
            //phonenumberEntry.Text = obj.phonenumber;

            Employee empupdate = new Employee(); //updateing Values
            empupdate.Address = AddressEntry.Text;
            empupdate.Ename = nameEntry.Text;
            empupdate.email = emailEntry.Text;
            empupdate.Eid = obj.Eid;
            App.Database.SaveItem(empupdate);
            Navigation.PushModalAsync(new EmployeeRegistration());

        }

        public EmployeeRegistration(int empid)
        {
            this.empid = empid;
            Employee lst = App.Database.GetItem(empid);
            //var Address = lst.Address;
            //var email = lst.email;
            //var Ename = lst.Ename;
            //var phonenumber = lst.phonenumber;
            //AddressEntry.Text = Address;
            //emailEntry.Text = email;
            //nameEntry.Text = Ename;
            //phonenumberEntry.Text = phonenumber;
```

```
                // to retriva values based on id to
                AddressEntry.Text = lst.Address;
                emailEntry.Text = lst.email;
                nameEntry.Text = lst.Ename;
                phonenumberEntry.Text = lst.phonenumber;

                Employee empupdate = new Employee(); //updateing Values
                empupdate.Address = AddressEntry.Text;
                empupdate.email = emailEntry.Text;
                App.Database.SaveItem(empupdate);
                Navigation.PushModalAsync(new EmployeeRegistration());
            }

            void addClicked(object sender, EventArgs e)
            {
                //var createEmp = (Employee)BindingContext;
                Employee emp = new Employee();
                emp.Address = AddressEntry.Text;
                emp.email = emailEntry.Text;
                emp.Ename = nameEntry.Text;
                emp.phonenumber = phonenumberEntry.Text;
                App.Database.SaveItem(emp);
                this.Navigation.PushAsync(new EmployeeDetails());

            }
            //void deleteClicked(object sender, EventArgs e)
            //{
            //    var emp = (Employee)BindingContext;
            //    App.Database.DeleteItem(emp.Eid);
            //    this.Navigation.PopAsync();
            //}
            void DetailsClicked(object sender, EventArgs e)
            {
                var empcancel = (Employee)BindingContext;
                this.Navigation.PushAsync(new EmployeeDetails());
            }
            //    void speakClicked(object sender, EventArgs e)
            //    {
            //        var empspek = (Employee)BindingContext;
            //        //DependencyService.Get<ITextSpeak>().Speak(empspek.Address + " " +
empspek.Ename);
            //    }
        }
    }
```

6. для отображения EmployeeDetails под кодом ниже

```
using DatabaseEmployeeCreation;
using DatabaseEmployeeCreation.SqlLite;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;

namespace DatabaseEmployeeCreation.SqlLite.Views
{
```

```
public partial class EmployeeDetails : ContentPage
{
    ListView lv = new ListView();
    IEnumerable<Employee> lst;
    public EmployeeDetails()
    {
        InitializeComponent();
        displayemployee();
    }

    private void displayemployee()
    {
        Button btn = new Button()
        {

            Text = "Details",
            BackgroundColor = Color.Blue,
        };
        btn.Clicked += Btn_Clicked;
        //IEnumerable<Employee> lst = App.Database.GetItems();
        //IEnumerable<Employee> lst1 = App.Database.GetItemsNotDone();
        //IEnumerable<Employee> lst2 = App.Database.GetItemsNotDone();
        Content = new StackLayout()
        {
            Children = { btn },
        };
    }

    private void Btn_Clicked(object sender, EventArgs e)
    {
        lst = App.Database.GetItems();

        lv.ItemsSource = lst;
        lv.HasUnevenRows = true;
        lv.ItemTemplate = new DataTemplate(typeof(OptionsViewCell));

        Content = new StackLayout()
        {
            Children = { lv },
        };

    }
}
```

```
public class OptionsViewCell : ViewCell
{

    int empid;
    Button btnEdit;
    public OptionsViewCell()
    {
    }
    protected override void OnBindingContextChanged()
    {
        base.OnBindingContextChanged();

        if (this.BindingContext == null)
            return;

        dynamic obj = BindingContext;
```

```
            empid = Convert.ToInt32(obj.Eid);
            var lblname = new Label
            {
                BackgroundColor = Color.Lime,
                Text = obj.Ename,
            };

            var lblAddress = new Label
            {
                BackgroundColor = Color.Yellow,
                Text = obj.Address,
            };

            var lblphonenumber = new Label
            {
                BackgroundColor = Color.Pink,
                Text = obj.phonenumber,
            };

            var lblemail = new Label
            {
                BackgroundColor = Color.Purple,
                Text = obj.email,
            };

            var lbleid = new Label
            {
                BackgroundColor = Color.Silver,
                Text = (empid).ToString(),
            };

            //var lblname = new Label
            //{
            //    BackgroundColor = Color.Lime,
            //    // HorizontalOptions = LayoutOptions.Start
            //};
            //lblname.SetBinding(Label.TextProperty, "Ename");

            //var lblAddress = new Label
            //{
            //    BackgroundColor = Color.Yellow,
            //    //HorizontalOptions = LayoutOptions.Center,
            //};
            //lblAddress.SetBinding(Label.TextProperty, "Address");

            //var lblphonenumber = new Label
            //{
            //    BackgroundColor = Color.Pink,
            //    //HorizontalOptions = LayoutOptions.CenterAndExpand,
            //};
            //lblphonenumber.SetBinding(Label.TextProperty, "phonenumber");

            //var lblemail = new Label
            //{
            //    BackgroundColor = Color.Purple,
            //    // HorizontalOptions = LayoutOptions.CenterAndExpand
            //};
            //lblemail.SetBinding(Label.TextProperty, "email");
            //var lbleid = new Label
            //{
            //    BackgroundColor = Color.Silver,
```

```
                //    // HorizontalOptions = LayoutOptions.CenterAndExpand
                //};
                //lbleid.SetBinding(Label.TextProperty, "Eid");
                Button btnDelete = new Button
                {
                    BackgroundColor = Color.Gray,

                    Text = "Delete",
                    //WidthRequest = 15,
                    //HeightRequest = 20,
                    TextColor = Color.Red,
                    HorizontalOptions = LayoutOptions.EndAndExpand,
                };
                btnDelete.Clicked += BtnDelete_Clicked;
                //btnDelete.PropertyChanged += BtnDelete_PropertyChanged;

                btnEdit = new Button
                {
                    BackgroundColor = Color.Gray,
                    Text = "Edit",
                    TextColor = Color.Green,
                };
                // lbleid.SetBinding(Label.TextProperty, "Eid");
                btnEdit.Clicked += BtnEdit_Clicked1; ;
                //btnEdit.Clicked += async (s, e) =>{
                //    await App.Current.MainPage.Navigation.PushModalAsync(new
EmployeeRegistration());
                //};

                View = new StackLayout()
                {
                    Orientation = StackOrientation.Horizontal,
                    BackgroundColor = Color.White,
                    Children = { lbleid, lblname, lblAddress, lblemail, lblphonenumber,
btnDelete, btnEdit },
                };

                //View = new StackLayout()
                //{ HorizontalOptions = LayoutOptions.Center, WidthRequest = 10,
BackgroundColor = Color.Yellow, Children = { lblAddress } };

                //View = new StackLayout()
                //{ HorizontalOptions = LayoutOptions.End, WidthRequest = 30, BackgroundColor
= Color.Yellow, Children = { lblemail } };

                //View = new StackLayout()
                //{ HorizontalOptions = LayoutOptions.End, BackgroundColor = Color.Green,
Children = { lblphonenumber } };




                //string Empid =c.eid ;

            }

            private async void BtnEdit_Clicked1(object sender, EventArgs e)
            {
               Employee obj= App.Database.GetItem(empid);
                if (empid > 0)
                {
```

```
                    await App.Current.MainPage.Navigation.PushModalAsync(new
EmployeeRegistration(obj));
                }
                else {
                await App.Current.MainPage.Navigation.PushModalAsync(new
EmployeeRegistration(empid));
                }
            }



            private void BtnDelete_Clicked(object sender, EventArgs e)
            {
                // var eid = Convert.ToInt32(empid);
                // var item = (Xamarin.Forms.Button)sender;
                int eid = empid;
                App.Database.DeleteItem(eid);
            }
            //private void BtnDelete_PropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
            //{
            // var ename=  e.PropertyName;
            //}
        }

        //private void BtnDelete_Clicked(object sender, EventArgs e)
        //{
        //    var eid = 8;
        //    var item = (Xamarin.Forms.Button)sender;

        //    App.Database.DeleteItem(eid);
        //}
    }
```

7. Чтобы реализовать метод в Android и ios GetConnection ()

```
    using System;
    using Xamarin.Forms;
    using System.IO;
    using DatabaseEmployeeCreation.Droid;
    using DatabaseEmployeeCreation.SqlLite.ViewModel;
    using SQLite;
    using SQLite.Net;

    [assembly: Dependency(typeof(SQLiteEmployee_Andriod))]
    namespace DatabaseEmployeeCreation.Droid
    {
        public class SQLiteEmployee_Andriod : ISQLite
        {
            public SQLiteEmployee_Andriod()
            {
            }

            #region ISQLite implementation
            public SQLiteConnection GetConnection()
            {
                //var sqliteFilename = "EmployeeSQLite.db3";
                //string documentsPath =
System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal); // Documents
folder
```

```
                //var path = Path.Combine(documentsPath, sqliteFilename);

                //// This is where we copy in the prepopulated database
                //Console.WriteLine(path);
                //if (!File.Exists(path))
                //{
                //    var s =
Forms.Context.Resources.OpenRawResource(Resource.Raw.EmployeeSQLite);  // RESOURCE NAME ###

                //    // create a write stream
                //    FileStream writeStream = new FileStream(path, FileMode.OpenOrCreate,
FileAccess.Write);
                //    // write to the stream
                //    ReadWriteStream(s, writeStream);
                //}

                //var conn = new SQLiteConnection(path);

                //// Return the database connection
                //return conn;
                var filename = "DatabaseEmployeeCreationSQLite.db3";
                var documentspath =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
                var path = Path.Combine(documentspath, filename);
                var platform = new SQLite.Net.Platform.XamarinAndroid.SQLitePlatformAndroid();
                var connection = new SQLite.Net.SQLiteConnection(platform, path);
                return connection;
            }

            //public  SQLiteConnection GetConnection()
            //{
            //    var filename = "EmployeeSQLite.db3";
            //    var documentspath =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            //    var path = Path.Combine(documentspath, filename);

            //    var platform = new
SQLite.Net.Platform.XamarinAndroid.SQLitePlatformAndroid();
            //    var connection = new SQLite.Net.SQLiteConnection(platform, path);
            //    return connection;
            //}
            #endregion

            /// <summary>
            /// helper method to get the database out of /raw/ and into the user filesystem
            /// </summary>
            void ReadWriteStream(Stream readStream, Stream writeStream)
            {
                int Length = 256;
                Byte[] buffer = new Byte[Length];
                int bytesRead = readStream.Read(buffer, 0, Length);
                // write the required bytes
                while (bytesRead > 0)
                {
                    writeStream.Write(buffer, 0, bytesRead);
                    bytesRead = readStream.Read(buffer, 0, Length);
                }
                readStream.Close();
                writeStream.Close();
            }
        }
```

```
    }
```

Я надеюсь, что этот вышеприведенный пример очень прост, и я объяснил

Прочитайте Работа с локальными базами данных онлайн: https://riptutorial.com/ru/xamarin-forms/topic/5997/работа-с-локальными-базами-данных

глава 30: Связывание данных

замечания

Возможные исключения

System.ArrayTypeMismatchException: Попытка получить доступ к элементу как к типу, несовместимому с массивом.

Это исключение может возникнуть при попытке привязать коллекцию к свойству non-bindable, когда включена предварительная компиляция XAML. Общим примером является попытка привязки к `Picker.Items` . Увидеть ниже.

System.ArgumentException: объект типа « Xamarin.Forms.Binding» не может быть преобразован в тип «System.String».

Это исключение может возникнуть при попытке привязать коллекцию к свойству без привязки, когда предварительная компиляция XAML отключена. Общим примером является попытка привязки к `Picker.Items` . Увидеть ниже.

Свойство `Picker.Items` не связывает

Этот код вызовет ошибку:

```
<!-- BAD CODE: will cause an error -->
<Picker Items="{Binding MyViewModelItems}" SelectedIndex="0" />
```

Исключение может быть одним из следующих:

> System.ArrayTypeMismatchException: Попытка получить доступ к элементу как к типу, несовместимому с массивом.

или же

> System.ArgumentException: объект типа «Xamarin.Forms.Binding» не может быть преобразован в тип «System.String».

В частности, свойство `Items` не привязывается. Решения включают создание собственного пользовательского элемента управления или использование стороннего элемента управления, такого как `BindablePicker` от FreshEssentials . После установки пакета FreshEssentials NuGet в вашем проекте доступен элемент управления `BindablePicker` пакета с привязываемым свойством `ItemsSource` :

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:fe="clr-namespace:FreshEssentials;assembly=FreshEssentials"
             xmlns:my="clr-namespace:MyAssembly;assembly=MyAssembly"
             x:Class="MyNamespace.MyPage">
  <ContentPage.BindingContext>
    <my:MyViewModel />
  </ContentPage.BindingContext>
  <ContentPage.Content>
    <fe:BindablePicker ItemsSource="{Binding MyViewModelItems}" SelectedIndex="0" />
  </ContentPage.Content>
</ContentPage>
```

Examples

Основное привязку к ViewModel

EntryPage.xaml:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:vm="clr-namespace:MyAssembly.ViewModel;assembly=MyAssembly"
             x:Class="MyAssembly.EntryPage">
  <ContentPage.BindingContext>
    <vm:MyViewModel />
  </ContentPage.BindingContext>
  <ContentPage.Content>
    <StackLayout VerticalOptions="FillAndExpand"
                 HorizontalOptions="FillAndExpand"
                 Orientation="Vertical"
                 Spacing="15">
      <Label Text="Name:" />
      <Entry Text="{Binding Name}" />
      <Label Text="Phone:" />
      <Entry Text="{Binding Phone}" />
      <Button Text="Save" Command="{Binding SaveCommand}" />
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

MyViewModel.cs:

```
using System;
using System.ComponentModel;

namespace MyAssembly.ViewModel
```

```
{
    public class MyViewModel : INotifyPropertyChanged
    {
        private string _name = String.Empty;
        private string _phone = String.Empty;

        public string Name
        {
            get { return _name; }
            set
            {
                if (_name != value)
                {
                    _name = value;
                    OnPropertyChanged(nameof(Name));
                }
            }
        }

        public string Phone
        {
            get { return _phone; }
            set
            {
                if (_phone != value)
                {
                    _phone = value;
                    OnPropertyChanged(nameof(Phone));
                }
            }
        }

        public ICommand SaveCommand { get; private set; }

        public MyViewModel()
        {
            SaveCommand = new Command(SaveCommandExecute);
        }

        private void SaveCommandExecute()
        {

        }

        public event PropertyChangedEventHandler PropertyChanged;

        protected virtual void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Прочитайте Связывание данных онлайн: https://riptutorial.com/ru/xamarin-forms/topic/3915/связывание-данных

глава 31: Создание настраиваемых элементов управления

Examples

Создайте пользовательский элемент управления вводами Xamarin Forms (не требуется родной)

Ниже приведен пример чистого пользовательского элемента управления Xamarin Forms. Никакой пользовательский рендеринг для этого не выполняется, но может быть легко реализован, по сути, в моем собственном коде, я использую этот самый элемент управления вместе с настраиваемым визуализатором как для `Label` и для `Entry` .

Пользовательский элемент управления - это `ContentView` с `BoxView` `Label` , « `Entry` и « `BoxView` внутри него, удерживаемый на месте с использованием 2 `StackLayout` . Мы также определяем несколько связующих свойств, а также событие `TextChanged` .

Пользовательские свойства связывания работают, определяясь как они ниже, и имеют элементы внутри элемента управления (в данном случае `Label` и `Entry`), привязанные к настраиваемым свойствам связывания. Некоторые из свойств `BindingPropertyChangedDelegate` также должны реализовать `BindingPropertyChangedDelegate` , чтобы ограниченные элементы меняли свои значения.

```
public class InputFieldContentView : ContentView {

    #region Properties

    /// <summary>
    /// Attached to the <c>InputFieldContentView</c>'s <c>ExtendedEntryOnTextChanged()</c>
event, but returns the <c>sender</c> as <c>InputFieldContentView</c>.
    /// </summary>
    public event System.EventHandler<TextChangedEventArgs> OnContentViewTextChangedEvent; //In
OnContentViewTextChangedEvent() we return our custom InputFieldContentView control as the
sender but we could have returned the Entry itself as the sender if we wanted to do that
instead.

    public static readonly BindableProperty LabelTextProperty =
BindableProperty.Create("LabelText", typeof(string), typeof(InputFieldContentView),
string.Empty);

    public string LabelText {
        get { return (string)GetValue(LabelTextProperty); }
        set { SetValue(LabelTextProperty, value); }
    }

    public static readonly BindableProperty LabelColorProperty =
BindableProperty.Create("LabelColor", typeof(Color), typeof(InputFieldContentView),
Color.Default);
```

```
    public Color LabelColor {
        get { return (Color)GetValue(LabelColorProperty); }
        set { SetValue(LabelColorProperty, value); }
    }

    public static readonly BindableProperty EntryTextProperty =
BindableProperty.Create("EntryText", typeof(string), typeof(InputFieldContentView),
string.Empty, BindingMode.TwoWay, null, OnEntryTextChanged);

    public string EntryText {
        get { return (string)GetValue(EntryTextProperty); }
        set { SetValue(EntryTextProperty, value); }
    }

    public static readonly BindableProperty PlaceholderTextProperty =
BindableProperty.Create("PlaceholderText", typeof(string), typeof(InputFieldContentView),
string.Empty);

    public string PlaceholderText {
        get { return (string)GetValue(PlaceholderTextProperty); }
        set { SetValue(PlaceholderTextProperty, value); }
    }

    public static readonly BindableProperty UnderlineColorProperty =
BindableProperty.Create("UnderlineColor", typeof(Color), typeof(InputFieldContentView),
Color.Black, BindingMode.TwoWay, null, UnderlineColorChanged);

    public Color UnderlineColor {
        get { return (Color)GetValue(UnderlineColorProperty); }
        set { SetValue(UnderlineColorProperty, value); }
    }

    private BoxView _underline;

    #endregion

    public InputFieldContentView() {

        BackgroundColor   = Color.Transparent;
        HorizontalOptions = LayoutOptions.FillAndExpand;

        Label label = new Label {
            BindingContext    = this,
            HorizontalOptions = LayoutOptions.StartAndExpand,
            VerticalOptions   = LayoutOptions.Center,
            TextColor         = Color.Black
        };

        label.SetBinding(Label.TextProperty, (InputFieldContentView view) => view.LabelText,
BindingMode.TwoWay);
        label.SetBinding(Label.TextColorProperty, (InputFieldContentView view) =>
view.LabelColor, BindingMode.TwoWay);

        Entry entry = new Entry {
            BindingContext          = this,
            HorizontalOptions       = LayoutOptions.End,
            TextColor               = Color.Black,
            HorizontalTextAlignment = TextAlignment.End
        };
```

```
        entry.SetBinding(Entry.PlaceholderProperty, (InputFieldContentView view) =>
view.PlaceholderText, BindingMode.TwoWay);
        entry.SetBinding(Entry.TextProperty, (InputFieldContentView view) => view.EntryText,
BindingMode.TwoWay);

        entry.TextChanged += OnTextChangedEvent;

        _underline = new BoxView {
            BackgroundColor   = Color.Black,
            HeightRequest     = 1,
            HorizontalOptions = LayoutOptions.FillAndExpand
        };

        Content = new StackLayout {
            Spacing           = 0,
            HorizontalOptions = LayoutOptions.FillAndExpand,
            Children          = {
                new StackLayout {
                    Padding           = new Thickness(5, 0),
                    Spacing           = 0,
                    HorizontalOptions = LayoutOptions.FillAndExpand,
                    Orientation       = StackOrientation.Horizontal,
                    Children          = { label, entry }
                }, _underline
            }
        };

        SizeChanged += (sender, args) => entry.WidthRequest = Width * 0.5 - 10;
    }

    private static void OnEntryTextChanged(BindableObject bindable, object oldValue, object
newValue) {
        InputFieldContentView contentView = (InputFieldContentView)bindable;
        contentView.EntryText             = (string)newValue;
    }

    private void OnTextChangedEvent(object sender, TextChangedEventArgs args) {
        if(OnContentViewTextChangedEvent != null) { OnContentViewTextChangedEvent(this, new
TextChangedEventArgs(args.OldTextValue, args.NewTextValue)); } //Here is where we pass in
'this' (which is the InputFieldContentView) instead of 'sender' (which is the Entry control)
    }

    private static void UnderlineColorChanged(BindableObject bindable, object oldValue, object
newValue) {
        InputFieldContentView contentView      = (InputFieldContentView)bindable;
        contentView._underline.BackgroundColor = (Color)newValue;
    }
}
```

А вот картина конечного продукта на прошивке (изображение показывает , как он выглядит при использовании пользовательского визуализатора для `Label` и `Entry` , которая используется для удаления границы на прошивке и указать специальный шрифт для обоего элементов):

Один вопрос , который я столкнулся получал `BoxView.BackgroundColor` измениться , когда

`UnderlineColor` изменилось. Даже после привязки `BoxView` «s `BackgroundColor` свойства, оно не изменится , пока я не добавил `UnderlineColorChanged` делегата.

Ярлык со связываемой коллекцией пространств

Я создал пользовательскую метку с оболочкой вокруг свойства `FormattedText` :

```
public class MultiComponentLabel : Label
{
    public IList<TextComponent> Components { get; set; }

    public MultiComponentLabel()
    {
        var components = new ObservableCollection<TextComponent>();
        components.CollectionChanged += OnComponentsChanged;
        Components = components;
    }

    private void OnComponentsChanged(object sender, NotifyCollectionChangedEventArgs e)
    {
        BuildText();
    }

    private void OnComponentPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
    {
        BuildText();
    }

    private void BuildText()
    {
        var formattedString = new FormattedString();
        foreach (var component in Components)
        {
            formattedString.Spans.Add(new Span { Text = component.Text });
            component.PropertyChanged -= OnComponentPropertyChanged;
            component.PropertyChanged += OnComponentPropertyChanged;
        }

        FormattedText = formattedString;
    }
}
```

Я добавил коллекцию пользовательских `TextComponent` s:

```
public class TextComponent : BindableObject
{
    public static readonly BindableProperty TextProperty =
        BindableProperty.Create(nameof(Text),
                                typeof(string),
                                typeof(TextComponent),
                                default(string));

    public string Text
    {
        get { return (string)GetValue(TextProperty); }
        set { SetValue(TextProperty, value); }
```

```
    }
}
```

И когда коллекция текстовых компонентов изменяется или свойство `Text` отдельных изменений компонентов, я перестраиваю свойство `FormattedText` базы `Label` .

И как я использовал его в `XAML` :

```
<ContentPage x:Name="Page"
         xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         xmlns:controls="clr-namespace:SuperForms.Controls;assembly=SuperForms.Controls"
         x:Class="SuperForms.Samples.MultiComponentLabelPage">
<controls:MultiComponentLabel Margin="0,20,0,0">
    <controls:MultiComponentLabel.Components>
      <controls:TextComponent Text="Time"/>
      <controls:TextComponent Text=": "/>
      <controls:TextComponent Text="{Binding CurrentTime, Source={x:Reference Page}}"/>
    </controls:MultiComponentLabel.Components>
  </controls:MultiComponentLabel>
</ContentPage>
```

Codebehind страницы:

```
public partial class MultiComponentLabelPage : ContentPage
{
    private string _currentTime;

    public string CurrentTime
    {
        get { return _currentTime; }
        set
        {
            _currentTime = value;
            OnPropertyChanged();
        }
    }

    public MultiComponentLabelPage()
    {
        InitializeComponent();
        BindingContext = this;
    }

    protected override void OnAppearing()
    {
        base.OnAppearing();

        Device.StartTimer(TimeSpan.FromSeconds(1), () =>
        {
            CurrentTime = DateTime.Now.ToString("hh : mm : ss");
            return true;
        });
    }
}
```

Создание настраиваемого элемента управления Entry с использованием свойства MaxLength

Элемент управления Xamarin Forms `Entry` не имеет свойства `MaxLength` . Чтобы достичь этого, вы можете расширить `Entry` как `Entry` ниже, добавив свойство Bindable `MaxLength` . Затем вам просто нужно подписаться на событие `TextChanged` в `Entry` и проверить длину `Text` при его вызове:

```
class CustomEntry : Entry
{
    public CustomEntry()
    {
        base.TextChanged += Validate;
    }

    public static readonly BindableProperty MaxLengthProperty =
BindableProperty.Create(nameof(MaxLength), typeof(int), typeof(CustomEntry), 0);

    public int MaxLength
    {
        get { return (int)GetValue(MaxLengthProperty); }
        set { SetValue(MaxLengthProperty, value); }
    }

    public void Validate(object sender, TextChangedEventArgs args)
    {
        var e = sender as Entry;
        var val = e?.Text;

        if (string.IsNullOrEmpty(val))
            return;

        if (MaxLength > 0 && val.Length > MaxLength)
            val = val.Remove(val.Length - 1);

        e.Text = val;
    }
}
```

Использование в XAML:

```
   <ContentView xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         xmlns:customControls="clr-namespace:CustomControls;assembly=CustomControls"
         x:Class="Views.TestView">
<ContentView.Content>
    <customControls:CustomEntry MaxLength="10" />
</ContentView.Content>
```

Прочитайте Создание настраиваемых элементов управления онлайн: https://riptutorial.com/ru/xamarin-forms/topic/3913/создание-настраиваемых-элементов-управления

глава 32: Создание настраиваемых элементов управления

Вступление

В каждом представлении `Xamarin.Forms` имеется сопроводительный рендерер для каждой платформы, который создает экземпляр встроенного элемента управления. Когда представление отображается на конкретной платформе, `ViewRenderer` экземпляр класса `ViewRenderer` .

Процесс для этого заключается в следующем:

Создайте пользовательский элемент управления Xamarin.Forms.

Потребляйте пользовательский контроль из Xamarin.Forms.

Создайте собственный рендер для элемента управления на каждой платформе.

Examples

Внедрение CheckBox Control

В этом примере мы реализуем пользовательский флажок для Android и iOS.

Создание пользовательского элемента управления

```
namespace CheckBoxCustomRendererExample
{
    public class Checkbox : View
    {
        public static readonly BindableProperty IsCheckedProperty =
BindableProperty.Create<Checkbox, bool>(p => p.IsChecked, true, propertyChanged: (s, o, n) =>
{ (s as Checkbox).OnChecked(new EventArgs()); });
        public static readonly BindableProperty ColorProperty =
BindableProperty.Create<Checkbox, Color>(p => p.Color, Color.Default);

        public bool IsChecked
        {
            get
            {
                return (bool)GetValue(IsCheckedProperty);
            }
            set
            {
                SetValue(IsCheckedProperty, value);
            }
        }
```

```
        public Color Color
        {
            get
            {
                return (Color)GetValue(ColorProperty);
            }
            set
            {
                SetValue(ColorProperty, value);
            }
        }

        public event EventHandler Checked;

        protected virtual void OnChecked(EventArgs e)
        {
            if (Checked != null)
                Checked(this, e);
        }
    }
}
```

Мы начнем с Android Custom Renderer, создав новый класс (`CheckboxCustomRenderer`) в части `Android` нашего решения.

Несколько важных деталей:

- Нам нужно отметить верхнюю часть нашего класса атрибутом ExportRenderer, чтобы рендеринг был зарегистрирован с помощью `Xamarin.Forms` . Таким образом, `Xamarin.Forms` будет использовать этот рендерер, когда он попытается создать наш объект `Checkbox` на `Android` .
- Мы выполняем большую часть нашей работы в методе `OnElementChanged` , где мы создаем и настраиваем наш собственный элемент управления.

Потребление пользовательского контроля

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-
namespace:CheckBoxCustomRendererExample"
x:Class="CheckBoxCustomRendererExample.CheckBoxCustomRendererExamplePage">
    <StackLayout Padding="20">
        <local:Checkbox Color="Aqua" />
    </StackLayout>
</ContentPage>
```

Создание пользовательского рендерера на каждой платформе

Процесс создания пользовательского класса визуализатора выглядит следующим образом:

1. Создайте подкласс класса `ViewRenderer<T1,T2>` который отображает настраиваемый элемент управления. Аргументом первого типа должен быть пользовательский элемент управления, на который выполняется средство визуализации, в данном случае `CheckBox` . Аргументом второго типа должен быть собственный элемент управления, который будет реализовывать пользовательский элемент управления.
2. Переопределите метод `OnElementChanged` который отображает пользовательский контроль и логику записи для его настройки. Этот метод вызывается, когда `Xamarin.Forms` соответствующее управление `Xamarin.Forms` .
3. Добавьте атрибут `ExportRenderer` в собственный класс визуализатора, чтобы указать, что он будет использоваться для визуализации настраиваемого `Xamarin.Forms` управления `Xamarin.Forms` . Этот атрибут используется для регистрации пользовательского рендеринга с помощью `Xamarin.Forms` .

Создание пользовательского рендеринга для Android

```
[assembly: ExportRenderer(typeof(Checkbox), typeof(CheckBoxRenderer))]
namespace CheckBoxCustomRendererExample.Droid
{
    public class CheckBoxRenderer : ViewRenderer<Checkbox, CheckBox>
    {
        private CheckBox checkBox;

        protected override void OnElementChanged(ElementChangedEventArgs<Checkbox> e)
        {
            base.OnElementChanged(e);
            var model = e.NewElement;
            checkBox = new CheckBox(Context);
            checkBox.Tag = this;
            CheckboxPropertyChanged(model, null);
            checkBox.SetOnClickListener(new ClickListener(model));
            SetNativeControl(checkBox);
        }
        private void CheckboxPropertyChanged(Checkbox model, String propertyName)
        {
            if (propertyName == null || Checkbox.IsCheckedProperty.PropertyName ==
propertyName)
            {
                checkBox.Checked = model.IsChecked;
            }

            if (propertyName == null || Checkbox.ColorProperty.PropertyName == propertyName)
            {
                int[][] states = {
                    new int[] { Android.Resource.Attribute.StateEnabled}, // enabled
                    new int[] {Android.Resource.Attribute.StateEnabled}, // disabled
                    new int[] {Android.Resource.Attribute.StateChecked}, // unchecked
                    new int[] { Android.Resource.Attribute.StatePressed}  // pressed
                };
                var checkBoxColor = (int)model.Color.ToAndroid();
                int[] colors = {
                    checkBoxColor,
                    checkBoxColor,
                    checkBoxColor,
                    checkBoxColor
```

```
                };
                var myList = new Android.Content.Res.ColorStateList(states, colors);
                checkBox.ButtonTintList = myList;

            }
        }

        protected override void OnElementPropertyChanged(object sender,
PropertyChangedEventArgs e)
        {
            if (checkBox != null)
            {
                base.OnElementPropertyChanged(sender, e);

                CheckboxPropertyChanged((Checkbox)sender, e.PropertyName);
            }
        }

        public class ClickListener : Java.Lang.Object, IOnClickListener
        {
            private Checkbox _myCheckbox;
            public ClickListener(Checkbox myCheckbox)
            {
                this._myCheckbox = myCheckbox;
            }
            public void OnClick(global::Android.Views.View v)
            {
                _myCheckbox.IsChecked = !_myCheckbox.IsChecked;
            }
        }
    }
}
```

Создание пользовательского рендерера для iOS

Поскольку в iOS флажок не установлен, мы сначала создадим `CheckBoxView` а затем создадим рендер для нашего флажка Xamarin.Forms.

`CheckBoxView` основан на двух изображениях checked_checkbox.png и unchecked_checkbox.png, поэтому свойство `Color` будет проигнорировано.

Вид CheckBox:

```
namespace CheckBoxCustomRendererExample.iOS
{
    [Register("CheckBoxView")]
    public class CheckBoxView : UIButton
    {
        public CheckBoxView()
        {
            Initialize();
        }

        public CheckBoxView(CGRect bounds)
            : base(bounds)
        {
```

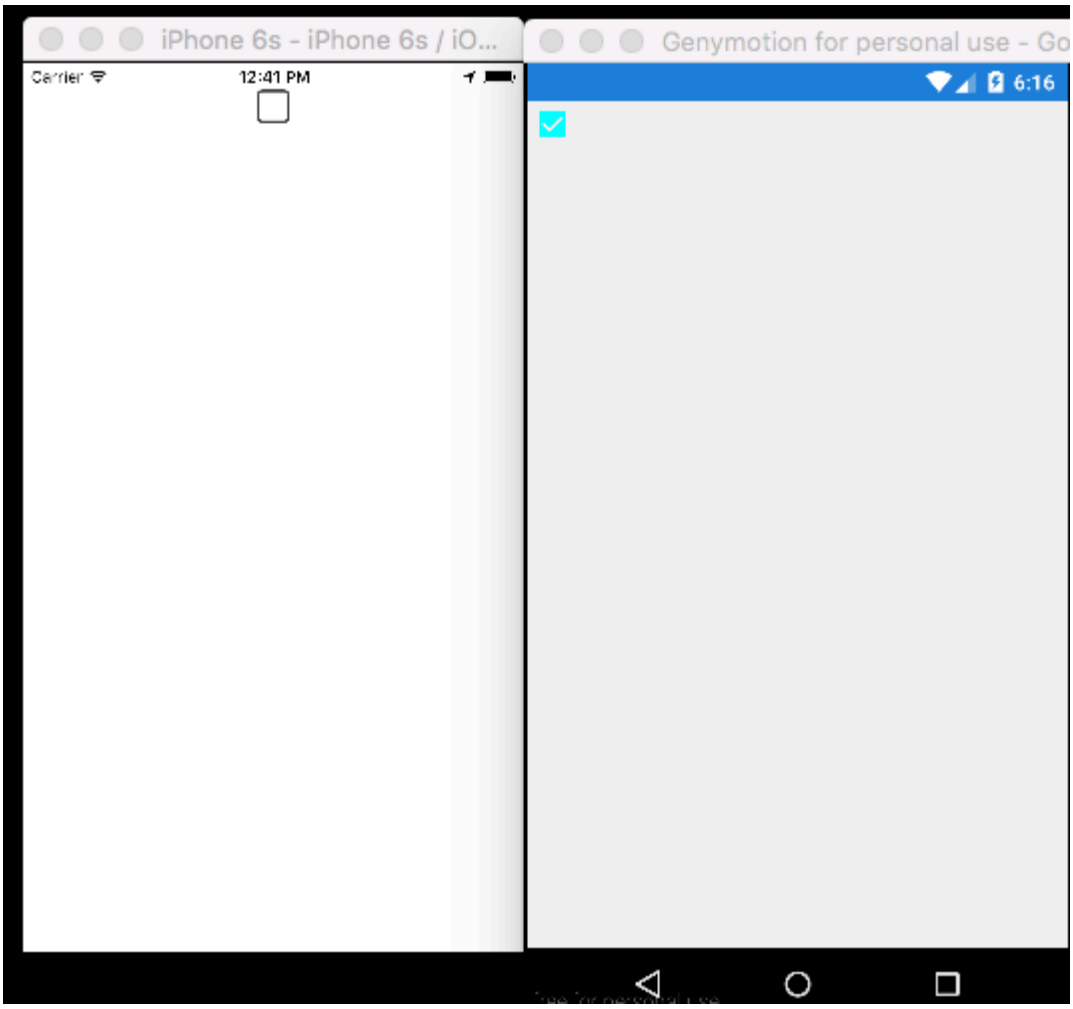
```
            Initialize();
        }

        public string CheckedTitle
        {
            set
            {
                SetTitle(value, UIControlState.Selected);
            }
        }

        public string UncheckedTitle
        {
            set
            {
                SetTitle(value, UIControlState.Normal);
            }
        }

        public bool Checked
        {
            set { Selected = value; }
            get { return Selected; }
        }

        void Initialize()
        {
            ApplyStyle();

            TouchUpInside += (sender, args) => Selected = !Selected;
            // set default color, because type is not UIButtonType.System
            SetTitleColor(UIColor.DarkTextColor, UIControlState.Normal);
            SetTitleColor(UIColor.DarkTextColor, UIControlState.Selected);
        }

        void ApplyStyle()
        {
            SetImage(UIImage.FromBundle("Images/checked_checkbox.png"),
UIControlState.Selected);
            SetImage(UIImage.FromBundle("Images/unchecked_checkbox.png"),
UIControlState.Normal);
        }
    }
}
```

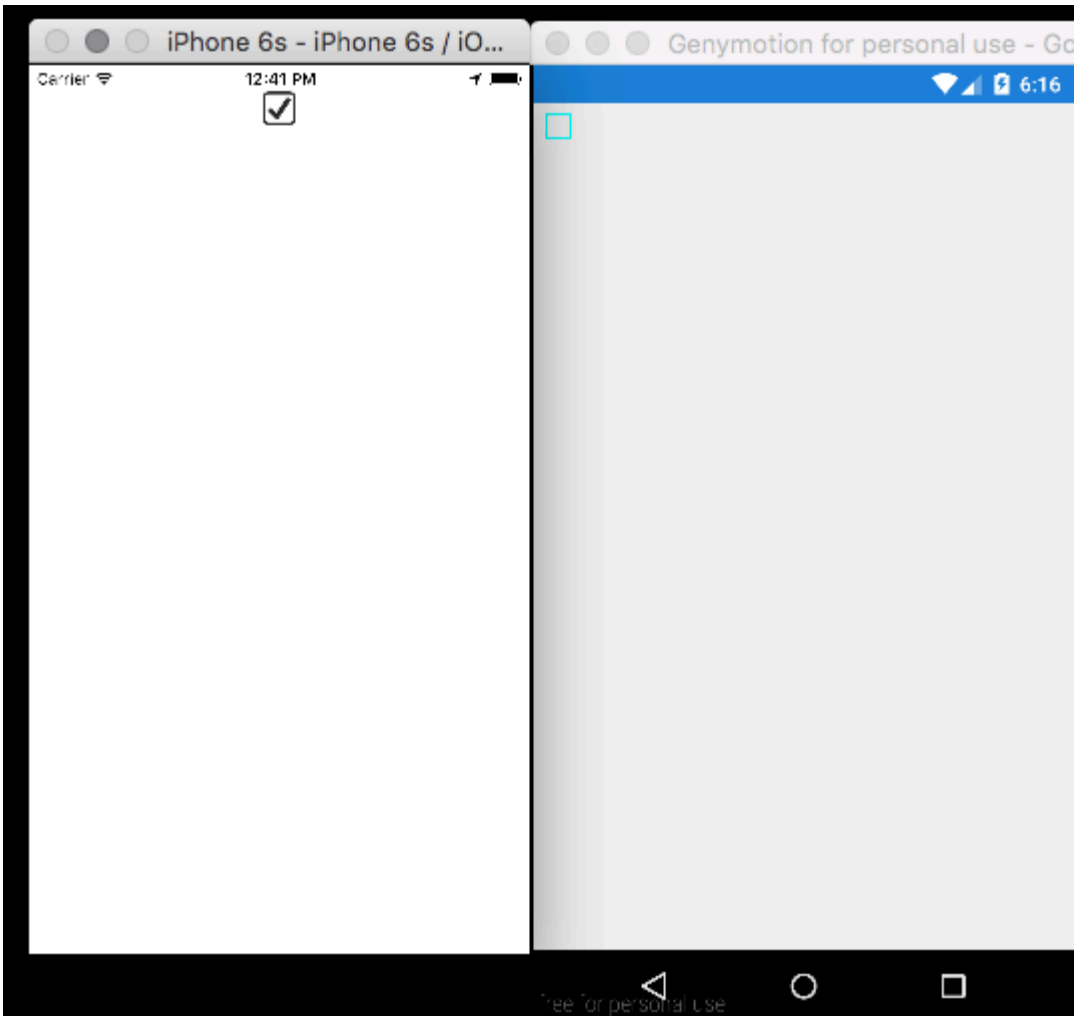
Пользовательский рендеринг CheckBox:

```
[assembly: ExportRenderer(typeof(Checkbox), typeof(CheckBoxRenderer))]
namespace CheckBoxCustomRendererExample.iOS
{
    public class CheckBoxRenderer : ViewRenderer<Checkbox, CheckBoxView>
    {

        /// <summary>
        /// Handles the Element Changed event
        /// </summary>
        /// <param name="e">The e.</param>
        protected override void OnElementChanged(ElementChangedEventArgs<Checkbox> e)
        {
            base.OnElementChanged(e);
```

```
            if (Element == null)
                return;

            BackgroundColor = Element.BackgroundColor.ToUIColor();
            if (e.NewElement != null)
            {
                if (Control == null)
                {
                    var checkBox = new CheckBoxView(Bounds);
                    checkBox.TouchUpInside += (s, args) => Element.IsChecked =
Control.Checked;
                    SetNativeControl(checkBox);
                }
                Control.Checked = e.NewElement.IsChecked;
            }

            Control.Frame = Frame;
            Control.Bounds = Bounds;

        }

        /// <summary>
        /// Handles the <see cref="E:ElementPropertyChanged" /> event.
        /// </summary>
        /// <param name="sender">The sender.</param>
        /// <param name="e">The <see cref="PropertyChangedEventArgs"/> instance containing the
event data.</param>
        protected override void OnElementPropertyChanged(object sender,
PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (e.PropertyName.Equals("Checked"))
            {
                Control.Checked = Element.IsChecked;
            }
        }
    }
}
```

Результат:


iPhone 6s - iPhone 6s / iO...
Carrier
12:41 PM
Genymotion for personal use - Go
6:16



Прочитайте Создание настраиваемых элементов управления онлайн: https://riptutorial.com/ru/xamarin-forms/topic/5975/создание-настраиваемых-элементов-управления

глава 33: Создание настраиваемых элементов управления

Examples

Создание пользовательской кнопки

```
/// <summary>
/// Button with some additional options
/// </summary>
public class TurboButton : Button
{
    public static readonly BindableProperty StringDataProperty = BindableProperty.Create(
      propertyName: "StringData",
      returnType: typeof(string),
      declaringType: typeof(ButtonWithStorage),
      defaultValue: default(string));

    public static readonly BindableProperty IntDataProperty = BindableProperty.Create(
      propertyName: "IntData",
      returnType: typeof(int),
      declaringType: typeof(ButtonWithStorage),
      defaultValue: default(int));

    /// <summary>
    /// You can put here some string data
    /// </summary>
    public string StringData
    {
        get { return (string)GetValue(StringDataProperty); }
        set { SetValue(StringDataProperty, value); }
    }

    /// <summary>
    /// You can put here some int data
    /// </summary>
    public int IntData
    {
        get { return (int)GetValue(IntDataProperty); }
        set { SetValue(IntDataProperty, value); }
    }

    public TurboButton()
    {
        PropertyChanged += CheckIfPropertyLoaded;
    }

    /// <summary>
    /// Called when one of properties is changed
    /// </summary>
    private void CheckIfPropertyLoaded(object sender, PropertyChangedEventArgs e)
    {
        //example of using PropertyChanged
        if(e.PropertyName == "IntData")
```

```
        {
            //IntData is now changed, you can operate on updated value
        }
    }
}
```

Использование в XAML:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="SomeApp.Pages.SomeFolder.Example"
    xmlns:customControls="clr-namespace:SomeApp.CustomControls;assembly=SomeApp">
    <StackLayout>
        <customControls:TurboButton x:Name="exampleControl" IntData="2" StringData="Test" />
    </StackLayout>
</ContentPage>
```

Теперь вы можете использовать свои свойства в с #:

```
exampleControl.IntData
```

Обратите внимание, что вам нужно указать сами, где ваш класс TurboButton размещен в вашем проекте. Я сделал это в этой строке:

```
xmlns:customControls="clr-namespace:SomeApp.CustomControls;assembly=SomeApp"
```

Вы можете свободно изменять «customControls» на другое имя. Это зависит от вас, как вы это называете.

Прочитайте Создание настраиваемых элементов управления онлайн: https://riptutorial.com/ru/xamarin-forms/topic/6592/создание-настраиваемых-элементов-управления

глава 34: Специальные визуальные корректировки для платформы

Examples

Коррекция идиомы

Индивидуальные корректировки с помощью идиомы можно выполнять с помощью кода C #, например, для изменения ориентации макета, отображается ли представление или телефон или планшет.

```
if (Device.Idiom == TargetIdiom.Phone)
{
    this.panel.Orientation = StackOrientation.Vertical;
}
else
{
    this.panel.Orientation = StackOrientation.Horizontal;
}
```

Эти функции также доступны непосредственно из кода XAML:

```
<StackLayout x:Name="panel">
  <StackLayout.Orientation>
    <OnIdiom x:TypeArguments="StackOrientation">
      <OnIdiom.Phone>Vertical</OnIdiom.Phone>
      <OnIdiom.Tablet>Horizontal</OnIdiom.Tablet>
    </OnIdiom>
  </StackLayout.Orientation>
</StackLayout>
```

Корректировка платформы

Корректировки могут быть сделаны для определенных платформ из кода C #, например, для изменения заполнения для всех целевых платформ.

```
if (Device.OS == TargetPlatform.iOS)
{
    panel.Padding = new Thickness (10);
}
else
{
    panel.Padding = new Thickness (20);
}
```

Вспомогательный метод также доступен для сокращенных объявлений C #:

```
panel.Padding = new Thickness (Device.OnPlatform(10,20,0));
```

Эти функции также доступны непосредственно из кода XAML:

```
<StackLayout x:Name="panel">
  <StackLayout.Padding>
    <OnPlatform x:TypeArguments="Thickness"
      iOS="10"
      Android="20" />
  </StackLayout.Padding>
</StackLayout>
```

Использование стилей

При работе с XAML, используя централизованный `Style` вы можете обновить набор стилизованных представлений из одного места. Все настройки идиомы и платформы также могут быть интегрированы в ваши стили.

```
<Style TargetType="StackLayout">
  <Setter Property="Padding">
    <Setter.Value>
      <OnPlatform x:TypeArguments="Thickness"
                  iOS="10"
                  Android="20"/>
    </Setter.Value>
  </Setter>
</Style>
```

Использование пользовательских представлений

Благодаря этим инструментам настройки вы можете создавать пользовательские представления, которые могут быть интегрированы на вашу страницу.

Выберите « `File > New > File... > Forms > Forms ContentView (Xaml)` и создайте представление для каждого конкретного макета: `TabletHome.xaml` и `PhoneHome.xaml` .

Затем выберите `File > New > File... > Forms > Forms ContentPage` и создайте `HomePage.cs` который содержит:

```
using Xamarin.Forms;

public class HomePage : ContentPage
{
    public HomePage()
    {
        if (Device.Idiom == TargetIdiom.Phone)
        {
            Content = new PhoneHome();
        }
        else
        {
            Content = new TabletHome();
```

```
            }

        }
    }
```

Теперь у вас есть `HomePage` который создает другую иерархию представлений для идиом `Phone` и `Tablet` .

Прочитайте Специальные визуальные корректировки для платформы онлайн: https://riptutorial.com/ru/xamarin-forms/topic/5012/специальные-визуальные-корректировки-для-платформы

глава 35: Страница Xamarin.Forms

Examples

TabbedPage

TabbedPage похож на NavigationPage, поскольку он позволяет и управляет простой навигацией между несколькими дочерними страницами. Разница в том, что, вообще говоря, на каждой платформе отображается какой-то бар в верхней или нижней части экрана, который отображает большинство (если не всех) доступных дочерних страниц. В приложениях Xamarin.Forms вкладка TabbedPage обычно полезна, когда у вас есть небольшое предопределенное количество страниц, с которыми пользователи могут перемещаться между ними, например меню или простой мастер, который можно расположить в верхней или нижней части экрана.

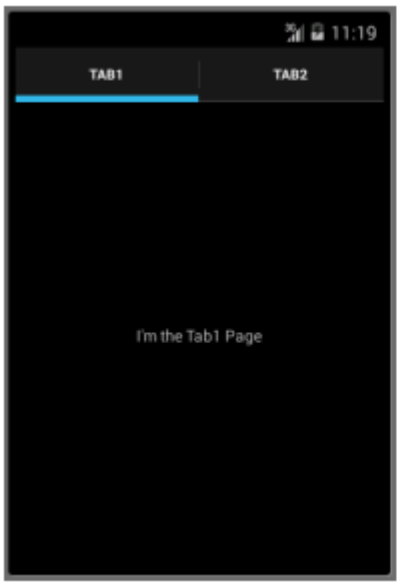
XAML

```
<?xml version="1.0" encoding="utf-8" ?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="XamlBasics.SampleXaml">
  <TabbedPage.Children>
    <ContentPage Title="Tab1">
      <Label Text="I'm the Tab1 Page"
             HorizontalOptions="Center"
             VerticalOptions="Center"/>
    </ContentPage>
    <ContentPage Title="Tab2">
    <Label Text="I'm the Tab2 Page"
           HorizontalOptions="Center"
           VerticalOptions="Center"/>
    </ContentPage>
  </TabbedPage.Children>
</TabbedPage>
```

Код

```
var page1 = new ContentPage {
Title = "Tab1",
Content = new Label {
Text = "I'm the Tab1 Page",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
}
};
var page2 = new ContentPage {
Title = "Tab2",
Content = new Label {
Text = "I'm the Tab2 Page",
HorizontalOptions = LayoutOptions.Center,
```

```
66
VerticalOptions = LayoutOptions.Center
}
};
var tabbedPage = new TabbedPage {
Children = { page1, page2 }
};
```



ContentPage

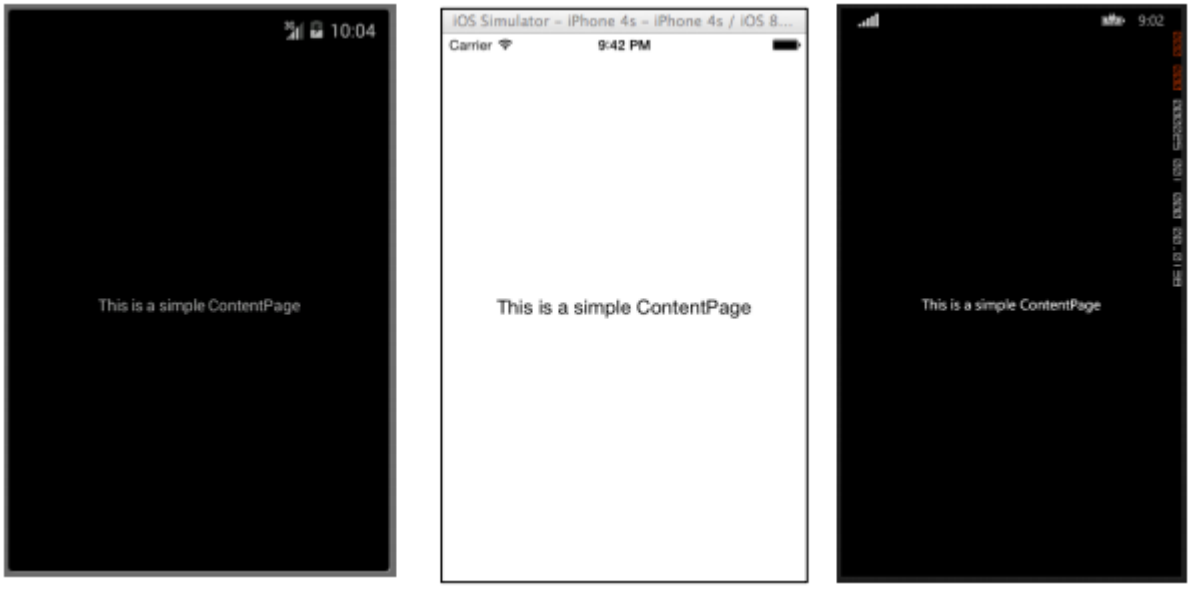
ContentPage: отображает один вид.

XAML

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="XamlBasics.SampleXaml">
<Label Text="This is a simple ContentPage"
HorizontalOptions="Center"
VerticalOptions="Center" />
</ContentPage>
```

Код

```
var label = new Label {
Text = "This is a simple ContentPage",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
};
var contentPage = new ContentPage {
Content = label
};
```



MasterDetailPage

MasterDetailPage: управляет двумя отдельными страницами (панелями) информации.

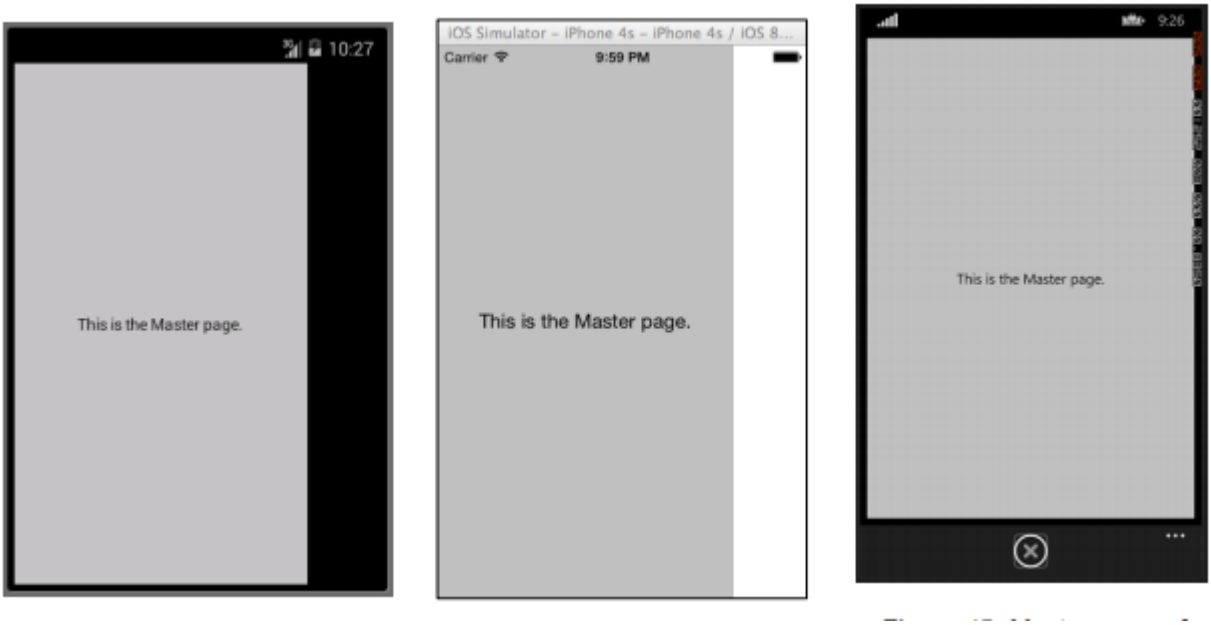
XAML

```
<?xml version="1.0" encoding="utf-8" ?>
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="XamlBasics.SampleXaml">
<MasterDetailPage.Master>
<ContentPage Title = "Master" BackgroundColor = "Silver">
<Label Text="This is the Master page."
TextColor = "Black"
HorizontalOptions="Center"
VerticalOptions="Center" />
</ContentPage>
</MasterDetailPage.Master>
<MasterDetailPage.Detail>
<ContentPage>
<Label Text="This is the Detail page."
HorizontalOptions="Center"
VerticalOptions="Center" />
</ContentPage>
</MasterDetailPage.Detail>
</MasterDetailPage>
```

Код

```
var masterDetailPage = new MasterDetailPage {
Master = new ContentPage {
Content = new Label {
Title = "Master",
BackgroundColor = Color.Silver,

TextColor = Color.Black,
Text = "This is the Master page.",
HorizontalOptions = LayoutOptions.Center,
```

```
VerticalOptions = LayoutOptions.Center
}
},
Detail = new ContentPage {
Content = new Label {
Title = "Detail",
Text = "This is the Detail page.",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
}
}
};
```



Прочитайте Страница Xamarin.Forms онлайн: https://riptutorial.com/ru/xamarin-forms/topic/7018/страница-xamarin-forms

глава 36: Тестирование блоков BDD в Xamarin.Forms

замечания

- Контейнер / резольвер DI, который мы используем внутри этой библиотеки, - Autofac.
- Структура тестирования - NUnit 3x.
- Вы должны иметь возможность использовать эту библиотеку с любой инфраструктурой Xamarin.Forms
- Проект источника и примера доступен здесь

Examples

Простой Specflow для тестирования команд и навигации с помощью NUnit Test Runner

Почему нам это надо?

Нынешний способ проведения модульного тестирования в Xamarin.Forms осуществляется с помощью платформы, поэтому ваш тест должен работать в среде iOS, Android, Windows или Mac UI: запуск тестов в среде IDE

Xamarin также обеспечивает потрясающее тестирование пользовательского интерфейса с помощью предложения Xamarin.TestCloud , но при желании внедрить методы BDD dev и иметь возможность тестировать ViewModels и Commands, в то время как он работает дешево на отдельных тестовых запусках локально или на сервере сборки, нет встроенный.

Я разработал библиотеку, которая позволяет использовать Specflow с Xamarin.Forms, чтобы легко реализовать ваши функции из определений Scenarios до ViewModel независимо от любой среды MVVM, используемой для приложения (например, XLabs , MVVMCross , Prism)

Если вы новичок в BDD, проверьте, не выпущен ли Specflow .

Использование:

- Если у вас его еще нет, установите расширение Visual Studio Visual Studio здесь (или из вашей визуальной студии IDE): https://visualstudiogallery.msdn.microsoft.com/c74211e7-cb6e-4dfa-855d-df0ad4a37dd6

- Добавьте библиотеку классов в проект Xamarin.Forms. Это ваш тестовый проект.
- Добавьте пакет SpecFlow.Xamarin.Forms из nuget в тестовые проекты.
- Добавьте класс к тестовому проекту, который наследует «TestApp», и зарегистрируйте пары views / viewmodels, а также добавьте регистрацию DI, как показано ниже:

```
public class DemoAppTest : TestApp
{
    protected override void SetViewModelMapping()
    {
        TestViewFactory.EnableCache = false;

        // register your views / viewmodels below
        RegisterView<MainPage, MainViewModel>();
    }

    protected override void InitialiseContainer()
    {
        // add any di registration here
        // Resolver.Instance.Register<TInterface, TType>();
        base.InitialiseContainer();
    }
}
```

- Добавьте класс SetupHook в свой тестовый проект, чтобы добавить к нему специальные захваты. Вам нужно будет загружать тестовое приложение, как показано ниже, предоставляя класс, который вы создали выше, и исходную модель представления вашего приложения:

```
[Binding]
public class SetupHooks : TestSetupHooks
{
    /// <summary>
    ///     The before scenario.
    /// </summary>
    [BeforeScenario]
    public void BeforeScenario()
    {
        // bootstrap test app with your test app and your starting viewmodel
        new TestAppBootstrap().RunApplication<DemoAppTest, MainViewModel>();
    }
}
```

- Вам нужно будет добавить блок catch к вашим представлениям xamarin.forms codebehind, чтобы игнорировать фреймворк xamarin.forms, заставляющий вас запускать приложение ui (чего мы не хотим делать):

```
    public YourView()
    {
        try
        {
            InitializeComponent();
```

```
            }
            catch (InvalidOperationException soe)
            {
                if (!soe.Message.Contains("MUST"))
                    throw;
            }
        }
```

- Добавьте функцию specflow в ваш проект (используя шаблоны vs specflow, поставляемые с расширением vs specflow)
- Создайте / сгенерируйте класс шагов, который наследует TestStepBase, передавая параметр stageContext на базу.
- Используйте навигационные службы и помощники для навигации, выполнения команд и тестирования моделей просмотров:

```
  [Binding]
    public class GeneralSteps : TestStepBase
    {
        public GeneralSteps(ScenarioContext scenarioContext)
            : base(scenarioContext)
        {
            // you need to instantiate your steps by passing the scenarioContext to the base
        }

        [Given(@"I am on the main view")]
        public void GivenIAmOnTheMainView()
        {
            Resolver.Instance.Resolve<INavigationService>().PushAsync<MainViewModel>();


Resolver.Instance.Resolve<INavigationService>().CurrentViewModelType.ShouldEqualType<MainViewModel>();

        }

        [When(@"I click on the button")]
        public void WhenIClickOnTheButton()
        {
            GetCurrentViewModel<MainViewModel>().GetTextCommand.Execute(null);
        }

        [Then(@"I can see a Label with text ""(.*)""")]
        public void ThenICanSeeALabelWithText(string text)
        {
            GetCurrentViewModel<MainViewModel>().Text.ShouldEqual(text);
        }
    }
```

Расширенное использование для MVVM

Чтобы добавить к первому примеру, чтобы проверить инструкции навигации, которые происходят в приложении, мы должны предоставить ViewModel крючок для навигации. Для достижения этой цели:

- Добавьте пакет SpecFlow.Xamarin.Forms.IViewModel от NuGet к вашему проекту PCL Xamarin.Forms
- Внедрите интерфейс IViewModel в ViewModel. Это просто откроет свойство Xamarin.Forms Inavigation:
- `public class MainViewModel : INotifyPropertyChanged, IViewModel.IViewModel { public INavigation Navigation { get; set; }`
- В тестовой структуре вы сможете выбрать и управлять внутренней навигацией
- Вы можете использовать любые фреймворки MVVM для вашего приложения (например, XLabs , MVVMCross , Prism, чтобы назвать несколько). Пока интерфейс IViewModel реализован в вашей ViewModel, инфраструктура подберет его.

Прочитайте Тестирование блоков BDD в Xamarin.Forms онлайн:
https://riptutorial.com/ru/xamarin-forms/topic/6172/тестирование-блоков-bdd-в-xamarin-forms

глава 37: Тестирование устройства

Examples

Тестирование моделей просмотра

Прежде чем мы начнем ...

С точки зрения уровней приложений, ваш ViewModel - это класс, содержащий всю бизнес-логику и правила, которые делают приложение таким, каким оно должно соответствовать требованиям. Также важно сделать его максимально независимым, уменьшая ссылки на пользовательский интерфейс, уровень данных, встроенные функции и вызовы API и т. Д. Все это делает вашу виртуальную машину проверкой.
Короче говоря, ваша ViewModel:

- Не следует зависеть от классов пользовательского интерфейса (просмотров, страниц, стилей, событий);
- Не следует использовать статические данные других классов (насколько это возможно);
- Следует внедрить бизнес-логику и подготовить данные, которые должны быть использованы для пользовательского интерфейса;
- Должны использоваться другие компоненты (база данных, HTTP, пользовательский интерфейс) через интерфейсы, разрешаемые с помощью Injection Dependency.

  Ваша ViewModel может иметь свойства других типов виртуальных машин. Например, `ContactsPageViewModel` будет иметь тип сбора, такой как `ObservableCollection<ContactListItemViewModel>`

Бизнес-требования

Допустим, у нас есть следующие возможности для реализации:

```
As an unauthorized user
I want to log into the app
So that I will access the authorized features
```

После выяснения пользовательской истории мы определили следующие сценарии:

```
Scenario: trying to log in with valid non-empty creds
  Given the user is on Login screen
   When the user enters 'user' as username
    And the user enters 'pass' as password
```

```
    And the user taps the Login button
   Then the app shows the loading indicator
    And the app makes an API call for authentication

Scenario: trying to log in empty username
  Given the user is on Login screen
   When the user enters '  ' as username
    And the user enters 'pass' as password
    And the user taps the Login button
   Then the app shows an error message saying 'Please, enter correct username and password'
    And the app doesn't make an API call for authentication
```

Мы останемся только с этими двумя сценариями. Конечно, должно быть гораздо больше случаев, и вы должны определить их все до фактического кодирования, но для нас достаточно хорошо ознакомиться с модульным тестированием моделей просмотра.

Давайте рассмотрим классический подход TDD и начнем с написания пустого тестируемого класса. Затем мы будем писать тесты и сделаем их зелеными, реализовав бизнес-функциональность.

Общие классы

```
public abstract class BaseViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

Сервисы

Помните ли вы, что наша модель представления не должна использовать классы UI и HTTP напрямую? Вы должны определить их как абстракции и не зависеть от деталей реализации .

```
/// <summary>
/// Provides authentication functionality.
/// </summary>
public interface IAuthenticationService
{
    /// <summary>
    /// Tries to authenticate the user with the given credentials.
    /// </summary>
    /// <param name="userName">UserName</param>
    /// <param name="password">User's password</param>
    /// <returns>true if the user has been successfully authenticated</returns>
    Task<bool> Login(string userName, string password);
```

```
}

/// <summary>
/// UI-specific service providing abilities to show alert messages.
/// </summary>
public interface IAlertService
{
    /// <summary>
    /// Show an alert message to the user.
    /// </summary>
    /// <param name="title">Alert message title</param>
    /// <param name="message">Alert message text</param>
    Task ShowAlert(string title, string message);
}
```

Создание заглушки ViewModel

Хорошо, у нас будет класс страницы для экрана входа, но сначала начнем с ViewModel:

```
public class LoginPageViewModel : BaseViewModel
{
    private readonly IAuthenticationService authenticationService;
    private readonly IAlertService alertService;

    private string userName;
    private string password;
    private bool isLoading;

    private ICommand loginCommand;

    public LoginPageViewModel(IAuthenticationService authenticationService, IAlertService
alertService)
    {
        this.authenticationService = authenticationService;
        this.alertService = alertService;
    }

    public string UserName
    {
        get
        {
            return userName;
        }
        set
        {
            if (userName!= value)
            {
                userName= value;
                OnPropertyChanged();
            }
        }
    }

    public string Password
    {
        get
        {
```

```
            return password;
        }
        set
        {
            if (password != value)
            {
                password = value;
                OnPropertyChanged();
            }
        }
    }

    public bool IsLoading
    {
        get
        {
            return isLoading;
        }
        set
        {
            if (isLoading != value)
            {
                isLoading = value;
                OnPropertyChanged();
            }
        }
    }

    public ICommand LoginCommand => loginCommand ?? (loginCommand = new Command(Login));

    private void Login()
    {
        authenticationService.Login(UserName, Password);
    }
}
```

Мы определили два свойства `string` и команду для привязки к пользовательскому интерфейсу. Мы не будем описывать, как создать класс страницы, разметку XAML и связать ViewModel с этим в этом разделе, поскольку они не имеют ничего конкретного.

Как создать экземпляр LoginPageViewModel?

Я думаю, вы, вероятно, создали виртуальные машины только с конструктором. Теперь, поскольку вы можете видеть, что наша виртуальная машина зависит от 2-х услуг, которые вводятся в качестве параметров конструктора, так что не просто сделать `var viewModel = new LoginPageViewModel()` . Если вы не знакомы с Dependency Injection, это лучший момент, чтобы узнать об этом. Надлежащее модульное тестирование невозможно без знания и соблюдения этого принципа.

тесты

Теперь давайте напишем некоторые тесты в соответствии с перечисленными выше случаями использования. Прежде всего вам нужно создать новую сборку (только библиотеку классов или выбрать специальный проект тестирования, если вы хотите использовать инструменты тестирования модулей Microsoft). Назовите это что-то вроде `ProjectName.Tests` и добавьте ссылку на ваш оригинальный проект PCL.

В этом примере я собираюсь использовать NUnit и Moq, но вы можете использовать любые тестовые библиотеки по вашему выбору. Ничего особенного с ними не будет.

Хорошо, это тестовый класс:

```
[TestFixture]
public class LoginPageViewModelTest
{
}
```

Письменные тесты

Ниже приведены методы тестирования для первых двух сценариев. Попробуйте сохранить 1 метод тестирования на 1 ожидаемый результат и не проверять все в одном тесте. Это поможет вам получить более четкие отчеты о том, что не удалось в коде.

```
[TestFixture]
public class LoginPageViewModelTest
{
    private readonly Mock<IAuthenticationService> authenticationServiceMock =
        new Mock<IAuthenticationService>();
    private readonly Mock<IAlertService> alertServiceMock =
        new Mock<IAlertService>();

    [TestCase("user", "pass")]
    public void LogInWithValidCreds_LoadingIndicatorShown(string userName, string password)
    {
        LoginPageViewModel model = CreateViewModelAndLogin(userName, password);

        Assert.IsTrue(model.IsLoading);
    }

    [TestCase("user", "pass")]
    public void LogInWithValidCreds_AuthenticationRequested(string userName, string password)
    {
        CreateViewModelAndLogin(userName, password);

        authenticationServiceMock.Verify(x => x.Login(userName, password), Times.Once);
    }

    [TestCase("", "pass")]
    [TestCase("   ", "pass")]
```

```
    [TestCase(null, "pass")]
    public void LogInWithEmptyuserName_AuthenticationNotRequested(string userName, string
password)
    {
        CreateViewModelAndLogin(userName, password);

        authenticationServiceMock.Verify(x => x.Login(It.IsAny<string>(), It.IsAny<string>()),
Times.Never);
    }

    [TestCase("", "pass", "Please, enter correct username and password")]
    [TestCase("   ", "pass", "Please, enter correct username and password")]
    [TestCase(null, "pass", "Please, enter correct username and password")]
    public void LogInWithEmptyUserName_AlertMessageShown(string userName, string password,
string message)
    {
        CreateViewModelAndLogin(userName, password);

        alertServiceMock.Verify(x => x.ShowAlert(It.IsAny<string>(), message));
    }

    private LoginPageViewModel CreateViewModelAndLogin(string userName, string password)
    {
        var model = new LoginPageViewModel(
            authenticationServiceMock.Object,
            alertServiceMock.Object);

        model.UserName = userName;
        model.Password = password;

        model.LoginCommand.Execute(null);

        return model;
    }
}
```

И здесь мы идем:

Теперь цель состоит в том, чтобы написать правильную реализацию для метода `Login` ViewModel и все.

Реализация бизнес-логики

```
private async void Login()
{
    if (String.IsNullOrWhiteSpace(UserName) || String.IsNullOrWhiteSpace(Password))
    {
        await alertService.ShowAlert("Warning", "Please, enter correct username and
password");
    }
    else
    {
        IsLoading = true;
        bool isAuthenticated = await authenticationService.Login(UserName, Password);
    }
}
```

И после повторных тестов:

Теперь вы можете сохранить свой код новыми тестами, делая его более стабильным и безопасным для регрессии.

Прочитайте Тестирование устройства онлайн: https://riptutorial.com/ru/xamarin-forms/topic/3529/тестирование-устройства

глава 38: Триггеры и поведение

Examples

Пример запуска Xamarin Forms

`Trigger` s - простой способ добавить некоторую отзывчивость UX к вашему приложению. Один простой способ сделать это , чтобы добавить `Trigger` , который меняет `Label` «s `TextColor` в зависимости от того связанной с ним `Entry` имеет текст , введенную в него или нет.

Использование `Trigger` для этого позволяет `Label.TextColor` менять `Label.TextColor` серого (когда текст не вводится) на черный (как только пользователи вводят текст):

Конвертер (каждому конвертеру присваивается переменная `Instance` которая используется в привязке, так что новый экземпляр класса не создается каждый раз, когда он используется):

```
/// <summary>
/// Used in a XAML trigger to return <c>true</c> or <c>false</c> based on the length of
<c>value</c>.
/// </summary>
public class LengthTriggerConverter : Xamarin.Forms.IValueConverter {

    /// <summary>
    /// Used so that a new instance is not created every time this converter is used in the
XAML code.
    /// </summary>
    public static LengthTriggerConverter Instance = new LengthTriggerConverter();

    /// <summary>
    /// If a `ConverterParameter` is passed in, a check to see if <c>value</c> is greater than
<c>parameter</c> is made. Otherwise, a check to see if <c>value</c> is over 0 is made.
    /// </summary>
    /// <param name="value">The length of the text from an Entry/Label/etc.</param>
    /// <param name="targetType">The Type of object/control that the text/value is coming
from.</param>
    /// <param name="parameter">Optional, specify what length to test against (example: for 3
Letter Name, we would choose 2, since the 3 Letter Name Entry needs to be over 2 characters),
if not specified, defaults to 0.</param>
    /// <param name="culture">The current culture set in the device.</param>
    /// <returns><c>object</c>, which is a <c>bool</c> (<c>true</c> if <c>value</c> is greater
than 0 (or is greater than the parameter), <c>false</c> if not).</returns>
    public object Convert(object value, System.Type targetType, object parameter, CultureInfo
culture)     { return DoWork(value, parameter); }
    public object ConvertBack(object value, System.Type targetType, object parameter,
CultureInfo culture) { return DoWork(value, parameter); }

    private static object DoWork(object value, object parameter) {
        int parameterInt = 0;

        if(parameter != null) { //If param was specified, convert and use it, otherwise, 0 is
```

```
used

            string parameterString = (string)parameter;

            if(!string.IsNullOrEmpty(parameterString)) { int.TryParse(parameterString, out
parameterInt); }
        }

        return (int)value > parameterInt;
    }
}
```

XAML (код XAML использует `x:Name` на `Entry` , чтобы выяснить , в `Entry.Text` собственности составляет более 3 символов.):

```
<StackLayout>
    <Label Text="3 Letter Name">
        <Label.Triggers>
            <DataTrigger TargetType="Label"
                         Binding="{Binding Source={x:Reference NameEntry},
                                           Path=Text.Length,
                                           Converter={x:Static
helpers:LengthTriggerConverter.Instance},
                                           ConverterParameter=2}"
                          Value="False">
                <Setter Property="TextColor"
                        Value="Gray"/>
            </DataTrigger>
        </Label.Triggers>
    </Label>
    <Entry x:Name="NameEntry"
           Text="{Binding MealAmount}"
           HorizontalOptions="StartAndExpand"/>
</StackLayout>
```

Мульти триггеры

MultiTrigger не требуется часто, но есть ситуации, когда это очень удобно. MultiTrigger ведет себя аналогично Trigger или DataTrigger, но имеет несколько условий. Все условия должны быть правдой для стрельбителей Setters. Вот простой пример:

```
<!-- Text field needs to be initialized in order for the trigger to work at start -->
<Entry x:Name="email" Placeholder="Email" Text="" />
<Entry x:Name="phone" Placeholder="Phone" Text="" />
<Button Text="Submit">
    <Button.Triggers>
        <MultiTrigger TargetType="Button">
            <MultiTrigger.Conditions>
                <BindingCondition Binding="{Binding Source={x:Reference email},
Path=Text.Length}" Value="0" />
                <BindingCondition Binding="{Binding Source={x:Reference phone},
Path=Text.Length}" Value="0" />
            </MultiTrigger.Conditions>
            <Setter Property="IsEnabled" Value="False" />
        </MultiTrigger>
    </Button.Triggers>
```

```
</Button>
```

Пример состоит из двух разных записей, телефона и электронной почты, и один из них должен быть заполнен. MultiTrigger отключает кнопку отправки, когда оба поля пусты.

Прочитайте Триггеры и поведение онлайн: https://riptutorial.com/ru/xamarin-forms/topic/6012/триггеры-и-поведение

глава 39: Услуги зависимостей

замечания

API-интерфейс для конкретной платформы доступа из PCL или совместного проекта.

Examples

Доступ к камере и галерее.

(Код доступа) [https://github.com/vDoers/vDoersCameraAccess]

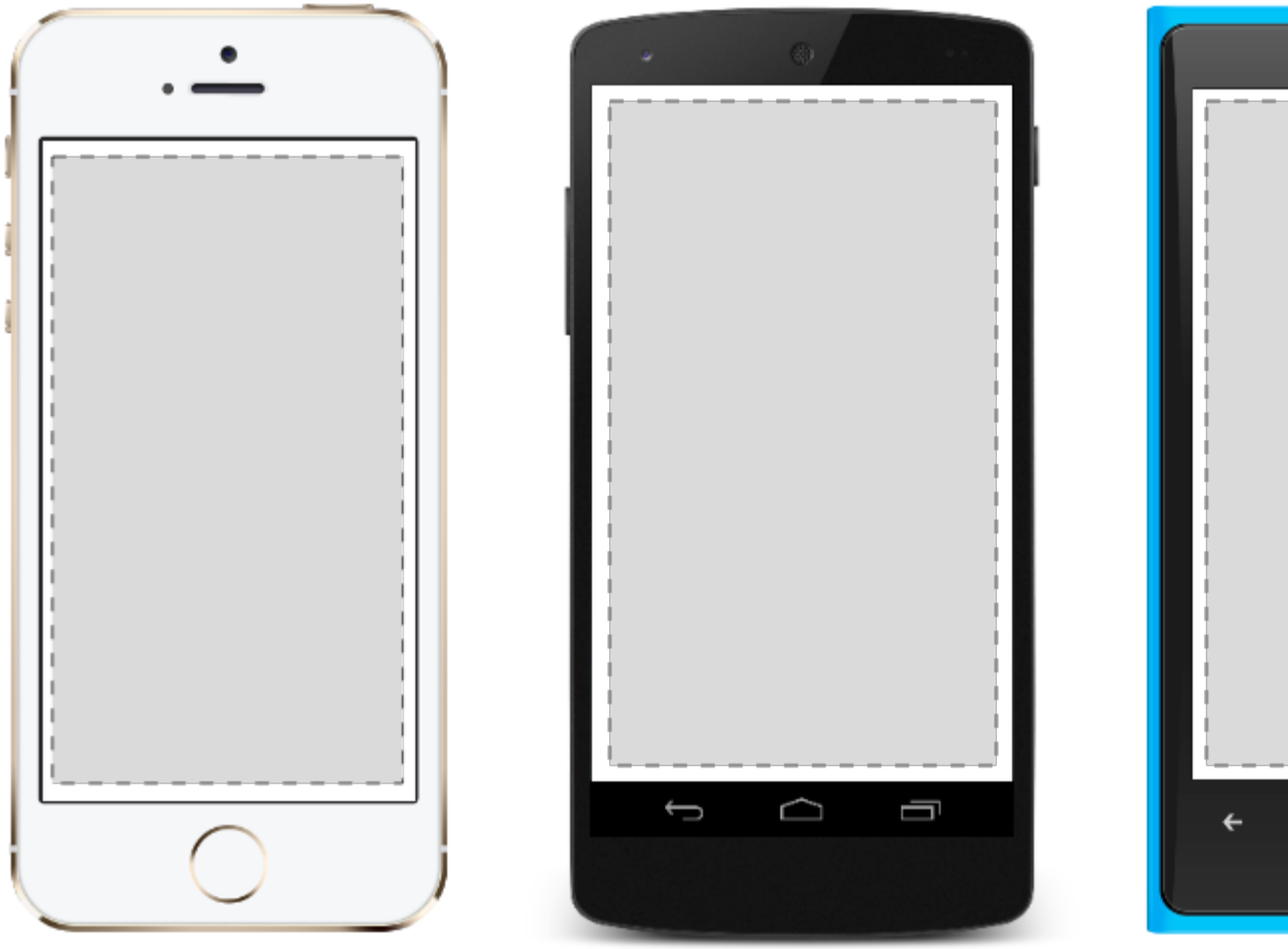
Прочитайте Услуги зависимостей онлайн: https://riptutorial.com/ru/xamarin-forms/topic/6127/услуги-зависимостей

глава 40: Форматы макетов Xamarin

Examples

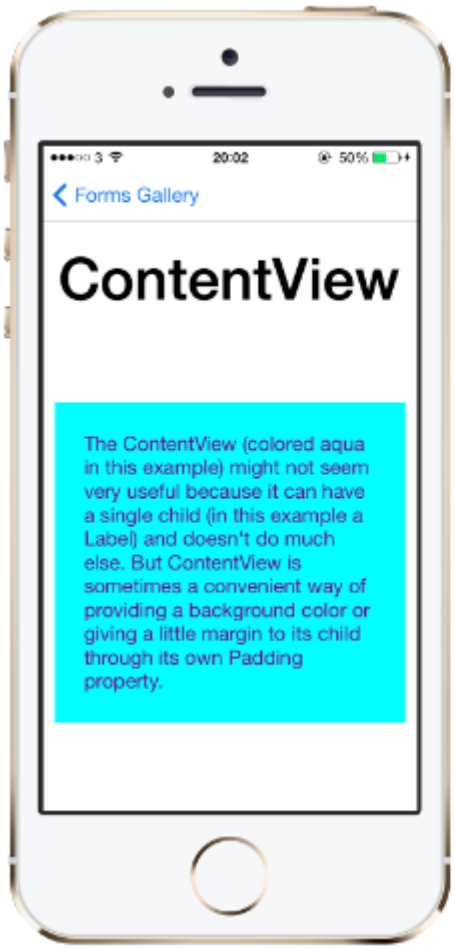
ContentPresenter

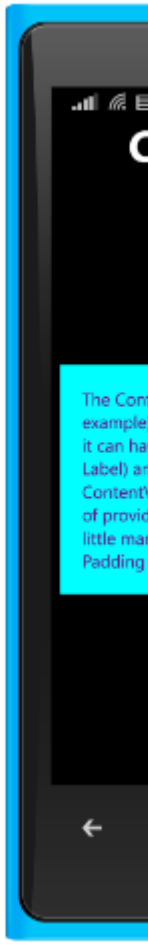
Менеджер макетов для шаблонных просмотров. Используется в ControlTemplate, чтобы отметить, где появится содержимое, которое будет представлено.



ContentView

Элемент с одним контентом. ContentView очень мало использует свое собственное. Его цель - служить базовым классом для пользовательских сложных представлений.





XAML

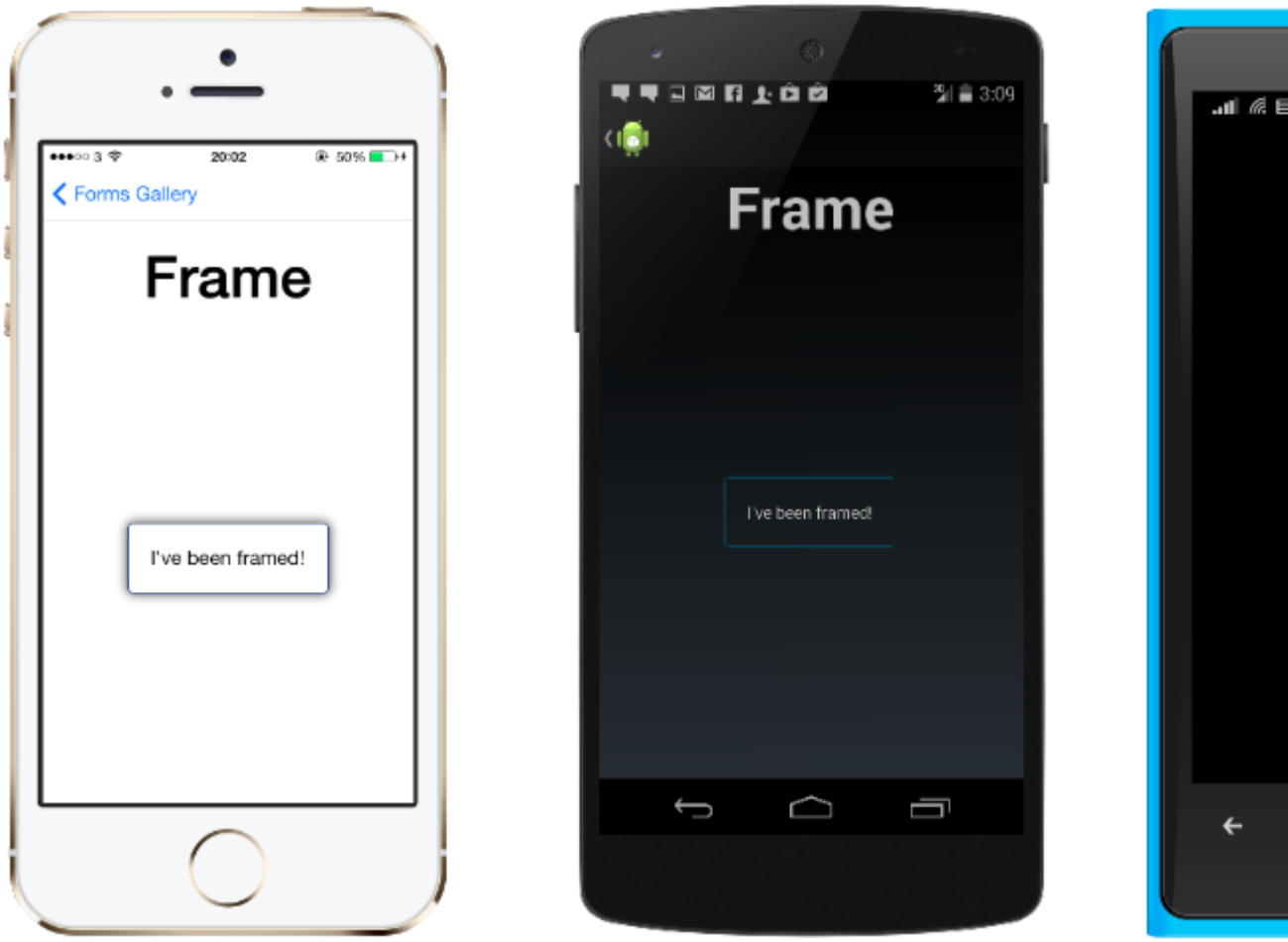
```
<ContentView>
<Label Text="Hi, I'm a simple Label inside of a simple ContentView"
HorizontalOptions="Center"
VerticalOptions="Center"/>
</ContentView>
```

Код

```
var contentView = new ContentView {
Content = new Label {
 Text = "Hi, I'm a simple Label inside of a simple ContentView",
 HorizontalOptions = LayoutOptions.Center,
 VerticalOptions = LayoutOptions.Center
}
};
```

Рамка

Элемент, содержащий один дочерний элемент, с некоторыми параметрами кадрирования. Рамка имеет значение по умолчанию Xamarin.Forms.Layout.Padding 20.



XAML

```
<Frame>
<Label Text="I've been framed!"
HorizontalOptions="Center"
VerticalOptions="Center" />
</Frame>
```

Код

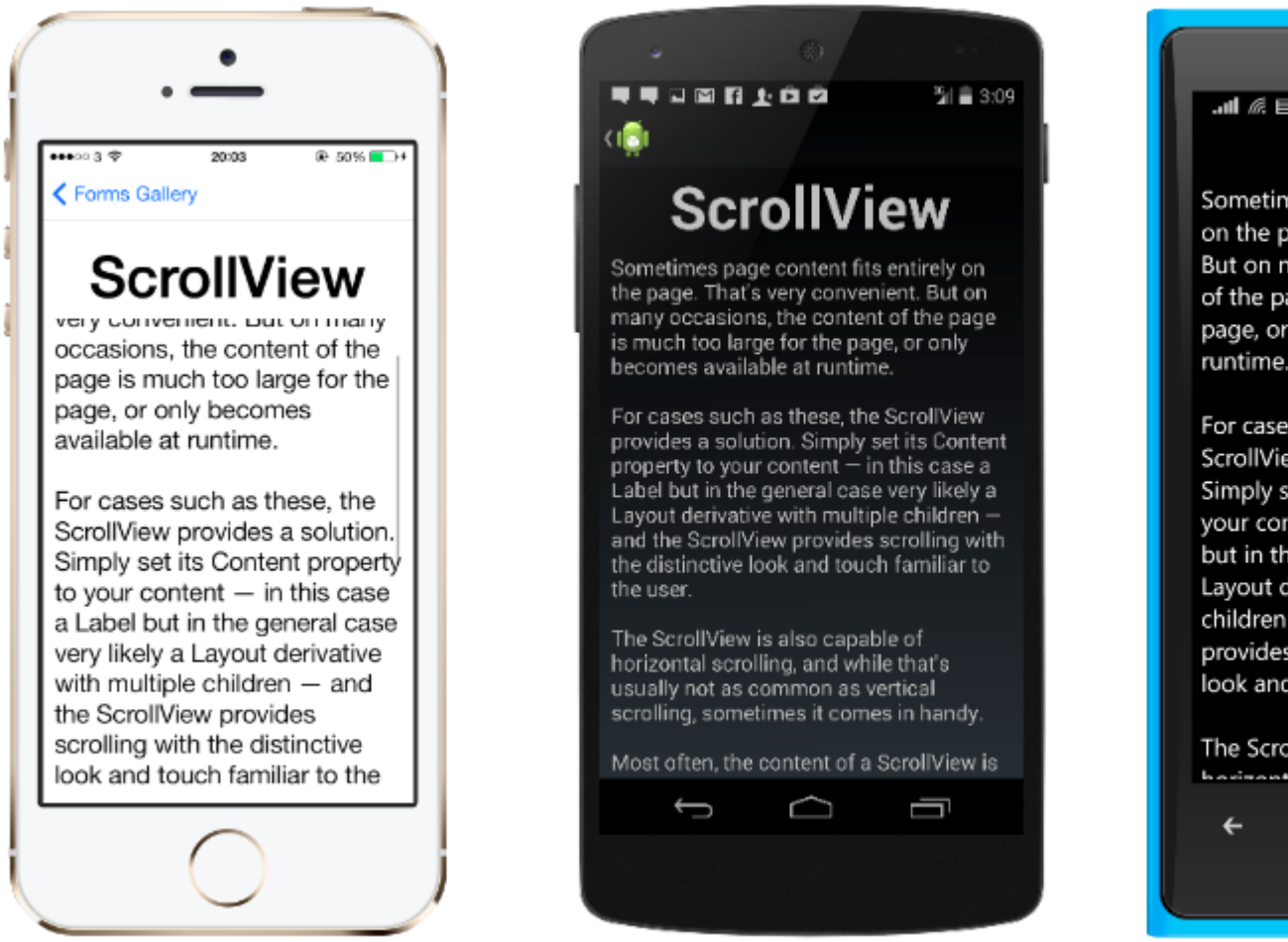
```
var frameView = new Frame {
Content = new Label {
 Text = "I've been framed!",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
},
OutlineColor = Color.Red
};
```

ScrollView

Элемент, способный прокручиваться, если это требует `Content` .

`ScrollView` содержит макеты и позволяет им прокручивать экран. `ScrollView` также используется, чтобы позволить представлениям автоматически перемещаться к видимой

части экрана при показе клавиатуры.



Примечание. `ScrollViews` не должны быть вложенными. Кроме того, `ScrollViews` не должны `ScrollViews` другие элементы управления, которые обеспечивают прокрутку, например `ListView` и `WebView` .

`ScrollView` легко определить. В XAML:

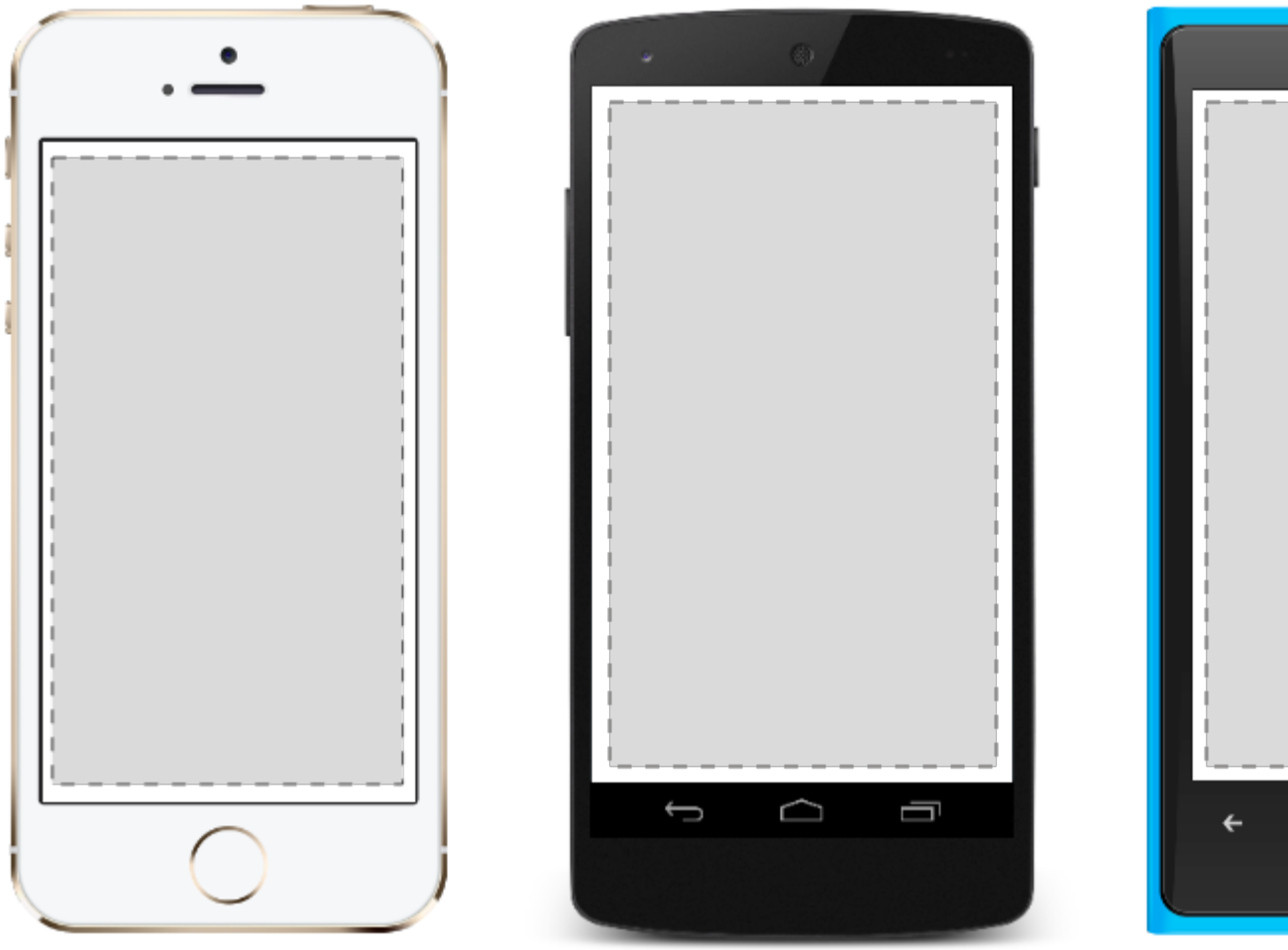
```
<ContentPage.Content>
    <ScrollView>
        <StackLayout>
            <BoxView BackgroundColor="Red" HeightRequest="600" WidthRequest="150" />
            <Entry />
        </StackLayout>
    </ScrollView>
</ContentPage.Content>
```

То же определение в коде:

```
var scroll = new ScrollView();
Content = scroll;
var stack = new StackLayout();
stack.Children.Add(new BoxView { BackgroundColor = Color.Red,   HeightRequest = 600,
WidthRequest = 600 });
stack.Children.Add(new Entry());
```

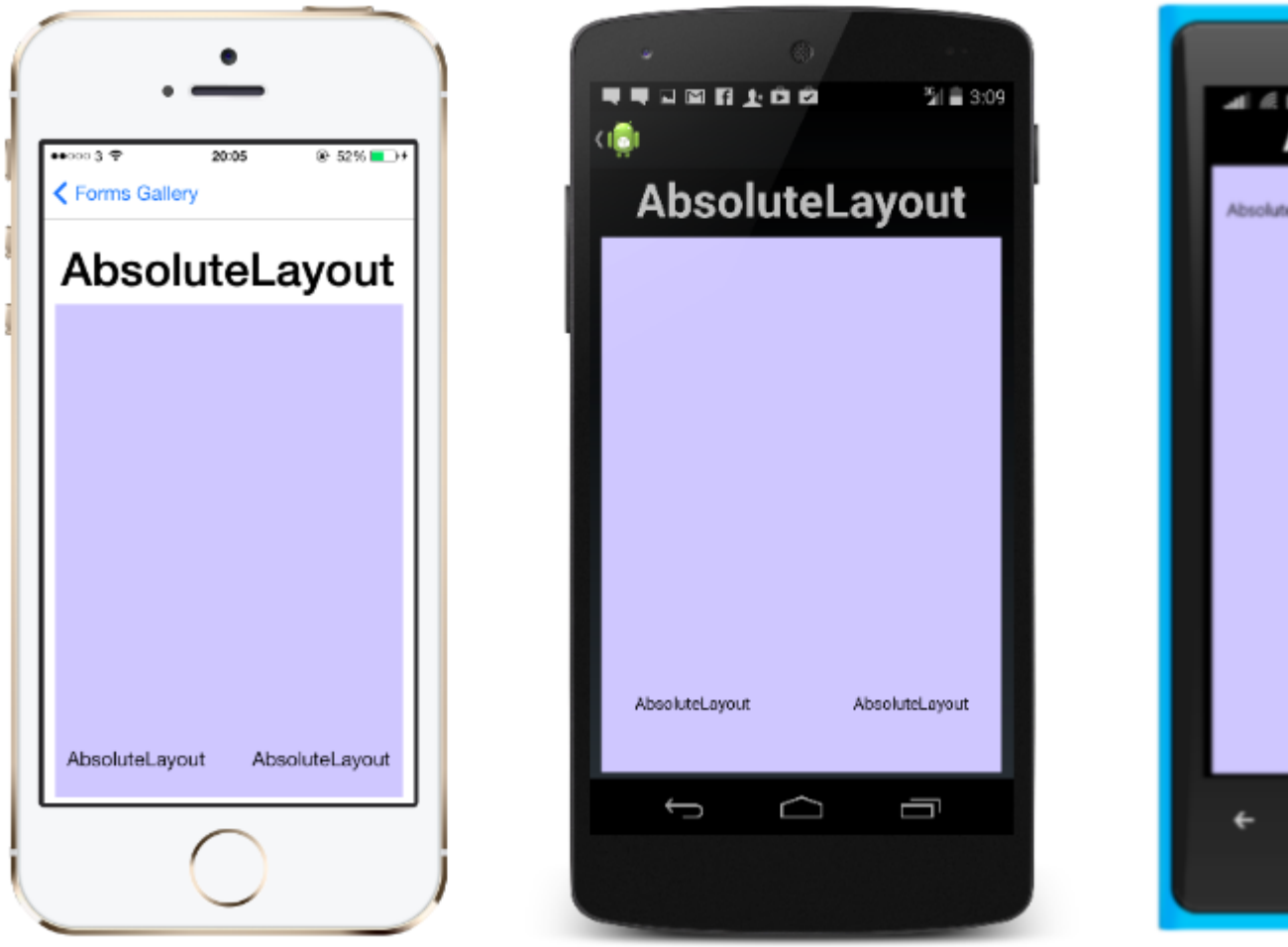
TemplatedView

Элемент, который отображает контент с помощью шаблона управления и базовый класс для `ContentView` .



AbsoluteLayout

`AbsoluteLayout` позиции и размеры дочерних элементов, пропорциональных его размеру и положению или по абсолютным значениям. Представления для детей могут быть расположены и измерены с использованием пропорциональных значений или статических значений, а пропорциональные и статические значения могут быть смешаны.



Определение `AbsoluteLayout` в XAML выглядит так:

```
<AbsoluteLayout>
    <Label Text="I'm centered on iPhone 4 but no other device"
        AbsoluteLayout.LayoutBounds="115,150,100,100" LineBreakMode="WordWrap"  />
    <Label Text="I'm bottom center on every device."
        AbsoluteLayout.LayoutBounds=".5,1,.5,.1" AbsoluteLayout.LayoutFlags="All"
        LineBreakMode="WordWrap"  />
    <BoxView Color="Olive"  AbsoluteLayout.LayoutBounds="1,.5, 25, 100"
        AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Red" AbsoluteLayout.LayoutBounds="0,.5,25,100"
        AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Blue" AbsoluteLayout.LayoutBounds=".5,0,100,25"
        AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Blue" AbsoluteLayout.LayoutBounds=".5,0,1,25"
        AbsoluteLayout.LayoutFlags="PositionProportional, WidthProportional" />
</AbsoluteLayout>
```

Тот же макет будет выглядеть так в коде:

```
Title = "Absolute Layout Exploration - Code";
var layout = new AbsoluteLayout();

var centerLabel = new Label {
Text = "I'm centered on iPhone 4 but no other device.",
LineBreakMode = LineBreakMode.WordWrap};
```

```
AbsoluteLayout.SetLayoutBounds (centerLabel, new Rectangle (115, 159, 100, 100));
// No need to set layout flags, absolute positioning is the default

var bottomLabel = new Label { Text = "I'm bottom center on every device.", LineBreakMode =
LineBreakMode.WordWrap };
AbsoluteLayout.SetLayoutBounds (bottomLabel, new Rectangle (.5, 1, .5, .1));
AbsoluteLayout.SetLayoutFlags (bottomLabel, AbsoluteLayoutFlags.All);

var rightBox = new BoxView{ Color = Color.Olive };
AbsoluteLayout.SetLayoutBounds (rightBox, new Rectangle (1, .5, 25, 100));
AbsoluteLayout.SetLayoutFlags (rightBox, AbsoluteLayoutFlags.PositionProportional);

var leftBox = new BoxView{ Color = Color.Red };
AbsoluteLayout.SetLayoutBounds (leftBox, new Rectangle (0, .5, 25, 100));
AbsoluteLayout.SetLayoutFlags (leftBox, AbsoluteLayoutFlags.PositionProportional);

var topBox = new BoxView{ Color = Color.Blue };
AbsoluteLayout.SetLayoutBounds (topBox, new Rectangle (.5, 0, 100, 25));
AbsoluteLayout.SetLayoutFlags (topBox, AbsoluteLayoutFlags.PositionProportional);

var twoFlagsBox = new BoxView{ Color = Color.Blue };
AbsoluteLayout.SetLayoutBounds (topBox, new Rectangle (.5, 0, 1, 25));
AbsoluteLayout.SetLayoutFlags (topBox, AbsoluteLayoutFlags.PositionProportional |
AbsoluteLayout.WidthProportional);

layout.Children.Add (bottomLabel);
layout.Children.Add (centerLabel);
layout.Children.Add (rightBox);
layout.Children.Add (leftBox);
layout.Children.Add (topBox);
```

Элемент AbsoluteLayout в Xamarin.Forms позволяет указать, где именно на экране вы хотите, чтобы дочерние элементы отображались, а также их размер и форма (границы).

Существует несколько различных способов установить границы дочерних элементов на основе перечисления AbsoluteLayoutFlags, которые используются во время этого процесса. Перечисление **AbsoluteLayoutFlags** содержит следующие значения:

- **Все** : Все размеры пропорциональны.
- **HeightProportional** : Высота пропорциональна расположению.
- **Нет** : интерпретация не выполняется.
- **PositionProportional** : Объединяет XProportional и YProportional.
- **SizeProportional** : Объединяет WidthProportional и HeightProportional.
- **WidthProportional:** Ширина пропорциональна макета.
- **XProportional** : свойство X пропорционально компоновке.
- **YProportional** : свойство Y пропорционально расположению.

Процесс работы с макетом контейнера AbsoluteLayout может показаться немного интуитивным вначале, но с небольшим использованием он станет привычным. Когда вы создадите дочерние элементы, чтобы установить их в абсолютном положении в контейнере, вам нужно выполнить три шага. Вы хотите установить флаги, назначенные элементам, используя метод **AbsoluteLayout.SetLayoutFlags ()** . Вы также захотите использовать метод **AbsoluteLayout.SetLayoutBounds (),** чтобы дать элементам свои

границы. Наконец, вы захотите добавить дочерние элементы в коллекцию Children. Так как Xamarin.Forms является абстракционным слоем между Xamarin и специфическими для устройства реализациями, позиционные значения могут быть независимы от пикселей устройства. Здесь упоминаются ранее упомянутые флаги компоновки. Вы можете выбрать, как процесс компоновки элементов управления Xamarin.Forms должен интерпретировать значения, которые вы определяете.

сетка

Макет, содержащий представления, расположенные в строках и столбцах.

Это типичное определение `Grid` в XAML.

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="2*" />
    <RowDefinition Height="*" />
    <RowDefinition Height="200" />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="*" />
  </Grid.ColumnDefinitions>
```

```
  <ContentView Grid.Row="0" Grid.Column="0"/>
  <ContentView Grid.Row="1" Grid.Column="0"/>
  <ContentView Grid.Row="2" Grid.Column="0"/>

  <ContentView Grid.Row="0" Grid.Column="1"/>
  <ContentView Grid.Row="1" Grid.Column="1"/>
  <ContentView Grid.Row="2" Grid.Column="1"/>

</Grid>
```

Такая же `Grid` определенная в коде, выглядит так:

```
var grid = new Grid();
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength(2, GridUnitType.Star) });
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength (1, GridUnitType.Star)
});
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength(200)});
grid.ColumnDefinitions.Add (new ColumnDefinition{ Width = new GridLength (200) });
```

Чтобы добавить элементы в сетку: В `XAML` :

```
<Grid>

   <--DEFINITIONS...--!>

   <ContentView Grid.Row="0" Grid.Column="0"/>
   <ContentView Grid.Row="1" Grid.Column="0"/>
   <ContentView Grid.Row="2" Grid.Column="0"/>

   <ContentView Grid.Row="0" Grid.Column="1"/>
   <ContentView Grid.Row="1" Grid.Column="1"/>
   <ContentView Grid.Row="2" Grid.Column="1"/>

</Grid>
```

В коде C #:

```
var grid = new Grid();
//DEFINITIONS...
var topLeft = new Label { Text = "Top Left" };
var topRight = new Label { Text = "Top Right" };
var bottomLeft = new Label { Text = "Bottom Left" };
var bottomRight = new Label { Text = "Bottom Right" };
grid.Children.Add(topLeft, 0, 0);
grid.Children.Add(topRight, 0, 1);
grid.Children.Add(bottomLeft, 1, 0);
grid.Children.Add(bottomRight, 1, 1);
```

Для `Height` и `Width` доступно несколько единиц.

- **Авто** - автоматически размеры, соответствующие содержанию в строке или столбце. Указан как GridUnitType.Auto в C # или как Auto в XAML.
- **Пропорциональные** столбцы и строки пропорционально оставшемуся пространству. Указывается как значение и GridUnitType.Star в C # и как # * в XAML, причем #

является вашим желаемым значением. Указание одной строки / столбца на * приведет к заполнению доступного пространства.

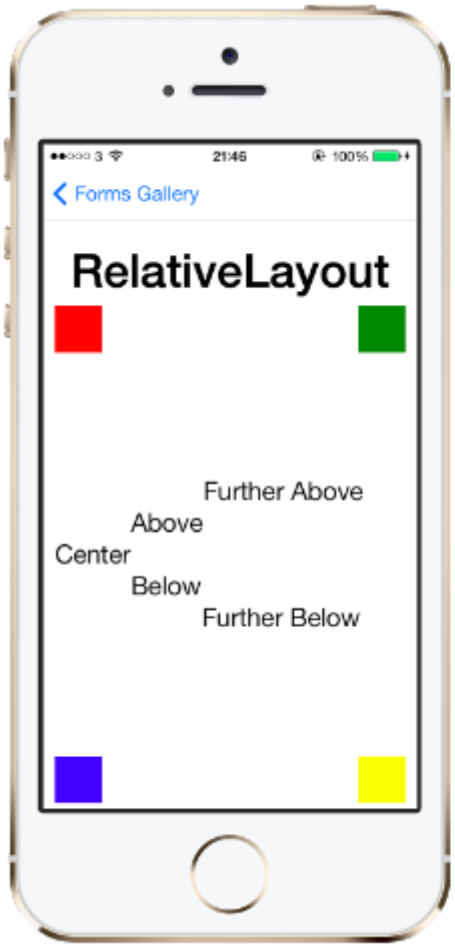
- **Абсолютные** размеры столбцов и строк с определенными фиксированными значениями высоты и ширины. Указывается как значение и GridUnitType.Absolute в C # и как # в XAML, причем # является вашим желаемым значением.

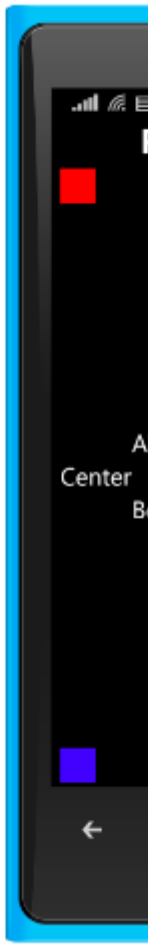
Примечание. Значения ширины столбцов по умолчанию заданы как «Авто» по умолчанию в Xamarin.Forms, что означает, что ширина определяется по размеру детей. Обратите внимание, что это отличается от реализации XAML на платформах Microsoft, где ширина по умолчанию - *, которая заполняет доступное пространство.

RelativeLayout

`Layout` который использует `Constraints` для компоновки своих детей.

`RelativeLayout` используется для размещения и размера представлений относительно свойств макета или представлений сестры. В отличие от `AbsoluteLayout` , `RelativeLayout` не имеет понятия движущегося якоря и не имеет средств для позиционирования элементов относительно нижнего или правого краев макета. `RelativeLayout` поддерживает элементы позиционирования за пределами собственных границ.





RelativeLayout в XAML выглядит следующим образом:

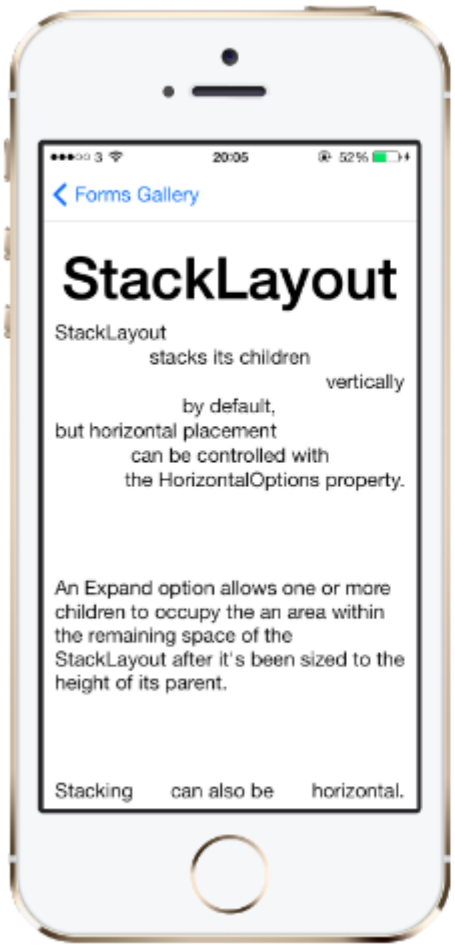
```
<RelativeLayout>
    <BoxView Color="Red" x:Name="redBox"
        RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToParent,
            Property=Height,Factor=.15,Constant=0}"
        RelativeLayout.WidthConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Width,Factor=1,Constant=0}"
        RelativeLayout.HeightConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Height,Factor=.8,Constant=0}" />
    <BoxView Color="Blue"
        RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToView,
            ElementName=redBox,Property=Y,Factor=1,Constant=20}"
        RelativeLayout.XConstraint="{ConstraintExpression Type=RelativeToView,
            ElementName=redBox,Property=X,Factor=1,Constant=20}"
        RelativeLayout.WidthConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Width,Factor=.5,Constant=0}"
        RelativeLayout.HeightConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Height,Factor=.5,Constant=0}" />
</RelativeLayout>
```

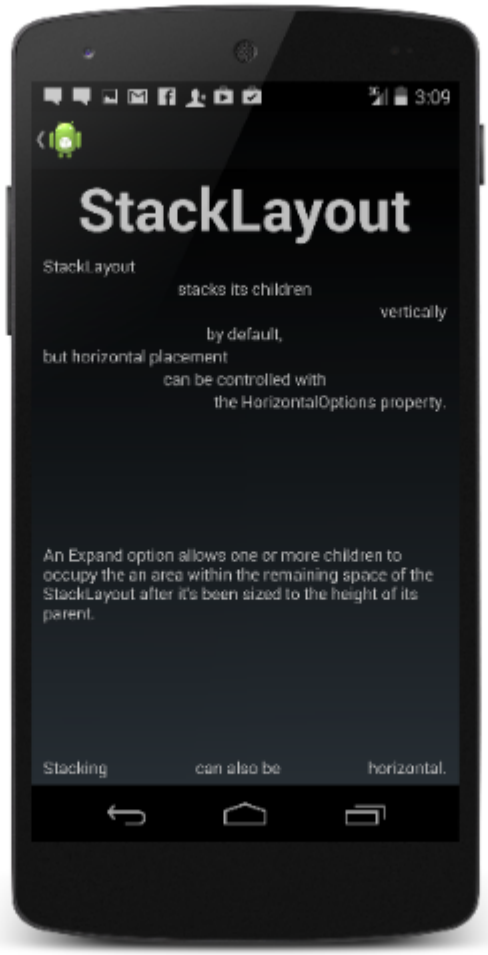
Такой же макет может быть выполнен с помощью этого кода:

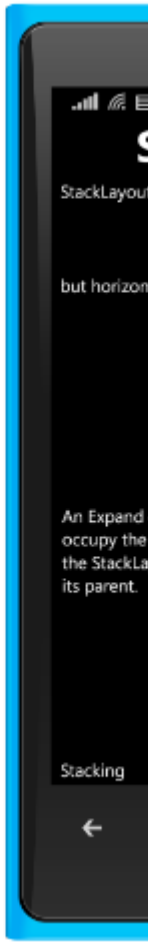
```
layout.Children.Add (redBox, Constraint.RelativeToParent ((parent) => {
        return parent.X;
    }), Constraint.RelativeToParent ((parent) => {
        return parent.Y * .15;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Width;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Height * .8;
    }));
layout.Children.Add (blueBox, Constraint.RelativeToView (redBox, (Parent, sibling) => {
        return sibling.X + 20;
    }), Constraint.RelativeToView (blueBox, (parent, sibling) => {
        return sibling.Y + 20;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Width * .5;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Height * .5;
    }));
```

StackLayout

`StackLayout` организует представления в одномерной строке («стек»), горизонтально или вертикально. `Views` в `StackLayout` могут быть рассчитаны на основе пространства в макете с использованием параметров макета. Позиционирование определяется просмотрами заказов, которые были добавлены в макет и варианты макета представлений.







Использование в XAML

```
<StackLayout>
    <Label Text="This will be on top" />
    <Button Text="This will be on the bottom" />
</StackLayout>
```

Использование в коде

```
StackLayout stackLayout = new StackLayout
{
    Spacing = 0,
    VerticalOptions = LayoutOptions.FillAndExpand,
    Children =
    {
        new Label
        {
            Text = "StackLayout",
            HorizontalOptions = LayoutOptions.Start
        },
        new Label
        {
```

```
            Text = "stacks its children",
            HorizontalOptions = LayoutOptions.Center
        },
        new Label
        {
            Text = "vertically",
            HorizontalOptions = LayoutOptions.End
        },
        new Label
        {
            Text = "by default,",
            HorizontalOptions = LayoutOptions.Center
        },
        new Label
        {
            Text = "but horizontal placement",
            HorizontalOptions = LayoutOptions.Start
        },
        new Label
        {
            Text = "can be controlled with",
            HorizontalOptions = LayoutOptions.Center
        },
        new Label
        {
            Text = "the HorizontalOptions property.",
            HorizontalOptions = LayoutOptions.End
        },
        new Label
        {
            Text = "An Expand option allows one or more children " +
                   "to occupy the an area within the remaining " +
                   "space of the StackLayout after it's been sized " +
                   "to the height of its parent.",
            VerticalOptions = LayoutOptions.CenterAndExpand,
            HorizontalOptions = LayoutOptions.End
        },
        new StackLayout
        {
            Spacing = 0,
            Orientation = StackOrientation.Horizontal,
            Children =
            {
                new Label
                {
                    Text = "Stacking",
                },
                new Label
                {
                    Text = "can also be",
                    HorizontalOptions = LayoutOptions.CenterAndExpand
                },
                new Label
                {
                    Text = "horizontal.",
                },
            }
        }
    }
};
```

Прочитайте Форматы макетов Xamarin онлайн: https://riptutorial.com/ru/xamarin-forms/topic/6273/форматы-макетов-xamarin

глава 41: Храм Хамарина

Examples

Нажмите «Жест»

С помощью Tap Gesture вы можете активировать любой пользовательский интерфейс (изображения, кнопки, стоп-листы, ...):

(1) В коде, используя событие:

```
var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.Tapped += (s, e) => {
    // handle the tap
};
image.GestureRecognizers.Add(tapGestureRecognizer);
```

(2) В коде, используя `ICommand` (например, с MVVM-Pattern):

```
var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.SetBinding (TapGestureRecognizer.CommandProperty, "TapCommand");
image.GestureRecognizers.Add(tapGestureRecognizer);
```

(3) Или в Xaml (с событием и `ICommand` требуется только один):

```
<Image Source="tapped.jpg">
    <Image.GestureRecognizers>
        <TapGestureRecognizer Tapped="OnTapGestureRecognizerTapped" Command="{Binding
TapCommand"} />
  </Image.GestureRecognizers>
</Image>
```

Прочитайте Храм Хамарина онлайн: https://riptutorial.com/ru/xamarin-forms/topic/7994/храм-хамарина

глава 42: Храм Хамарина

Examples

Жестокое событие

Когда мы устанавливаем контроль над Label, Label не предоставляет никаких событий. <Ярлык x: Name = "lblSignUp Text =" У вас нет учетной записи? "/>, Как показано только для обозначения только для ярлыков.

Когда пользователь хочет заменить Button с помощью Label, мы даем событие для Label. Как показано ниже:

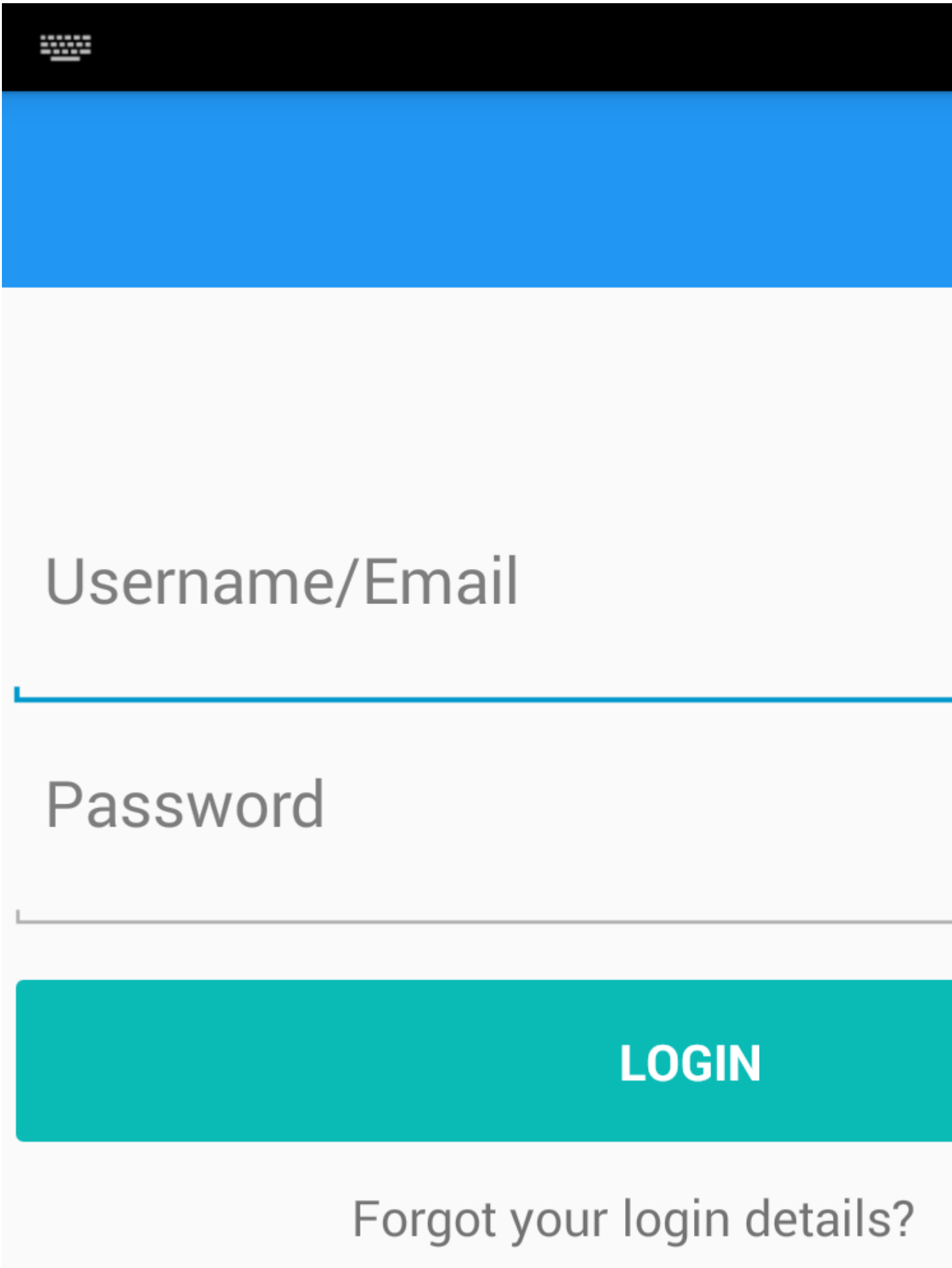
XAML

```
<Label x:Name="lblSignUp" Text="Don't have an account?" Grid.Row="8" Grid.Column="1"
Grid.ColumnSpan="2">
  <Label.GestureRecognizers>
    <TapGestureRecognizer
   Tapped="lblSignUp_Tapped"/>
  </Label.GestureRecognizers>
```

C #

```
var lblSignUp_Tapped = new TapGestureRecognizer();
lblSignUp_Tapped.Tapped += (s,e) =>
{
//
//  Do your work here.
//
};
lblSignUp.GestureRecognizers.Add(lblSignUp_Tapped);
```

Экран Ниже показано событие метки. Экран 1: ярлык «У вас нет учетной записи?» как показано ниже.


Username/Email
Password
LOGIN
Forgot your login details?

храм-хамарина

глава 43: Чтение AppSettings в Xamarin.Forms

Examples

Чтение файла app.config в проекте Xamarin.Forms Xaml

В то время как на всех мобильных платформах есть собственное управление настройками api, нет встроенных способов считывания настроек из старого старого файла app.config типа .net style.config; Это связано с множеством веских причин, в частности, управление конфигурацией инфраструктуры .NET. Api находится на стороне тяжеловеса, и каждая платформа имеет свою собственную файловую систему api.

Таким образом, мы создали простую библиотеку PCLAppConfig , красиво упакованную для немедленного потребления.

Эта библиотека использует прекрасную библиотеку PCLStorage

В этом примере предполагается, что вы разрабатываете проект Xamarin.Forms Xaml, где вам нужно будет получить доступ к настройкам из вашей общей модели просмотра.

1. Инициализируйте ConfigurationManager.AppSettings в каждом из ваших проектов платформы сразу после инструкции «Xamarin.Forms.Forms.Init», как показано ниже:

iOS (AppDelegate.cs)

```
global::Xamarin.Forms.Forms.Init();
ConfigurationManager.Initialise(PCLAppConfig.FileSystemStream.PortableStream.Current);
LoadApplication(new App());
```

Android (MainActivity.cs)

```
global::Xamarin.Forms.Forms.Init(this, bundle);
ConfigurationManager.Initialise(PCLAppConfig.FileSystemStream.PortableStream.Current);
LoadApplication(new App());
```

UWP / Windows 8.1 / WP 8.1 (App.xaml.cs)

```
Xamarin.Forms.Forms.Init(e);
ConfigurationManager.Initialise(PCLAppConfig.FileSystemStream.PortableStream.Current);
```

2. Добавьте файл app.config в общий проект PCL и добавьте записи appSettings, как и в любом файле app.config.

```
<configuration>
    <appSettings>
        <add key="config.text" value="hello from app.settings!" />
    </appSettings>
</configuration>
```

3. Добавьте этот файл приложения PCL app.config **в виде связанного файла** во всех проектах вашей платформы. Для android убедитесь, что действие сборки установлено на **«AndroidAsset»** , для UWP установите действие сборки в **«Содержимое»,**

4. Доступ к настройкам: `ConfigurationManager.AppSettings["config.text"];`

Прочитайте Чтение AppSettings в Xamarin.Forms онлайн: https://riptutorial.com/ru/xamarin-forms/topic/5911/чтение-appsettings-в-xamarin-forms

кредиты

S. No	Главы	Contributors
1	Начало работы с Xamarin.Forms	Akshay Kulkarni, chrisntr, Community, Demitrian, hankide, jdstaerk, Manohar, patridge, Sergey Metlov, spaceplane
2	CarouselView - предварительная версия	dpserge
3	Contact Picker - Xamarin Forms (Android и iOS)	Pucho Eric
4	DependencyService	Steven Thewissen
5	MessagingCenter	Gerald Versluis
6	OAuth2	Eng Soon Cheah
7	SQL Database и API в форматах Xamarin.	RIYAZ
8	Xamarin.Forms Просмотров	Aaron Thompson, Eng Soon Cheah
9	Всплывающие уведомления	Gerald Versluis, user1568891
10	Доступ к собственным функциям с помощью DependencyService	Gerald Versluis, hankide, hvaughan3, Sergey Metlov
11	жесты	doerig, Gerald Versluis, Michael Rumpler
12	Использование ListViews	cvanbeek
13	Клетки Xamarin.Forms	Eng Soon Cheah

14	Кэширование	Sergey Metlov
15	Навигация в Xamarin.Forms	Fernando Arreguín, jimmgarr, Lucas Moura Veloso, Paul, Sergey Metlov, Taras Shevchuk, Willian D. Andrade
16	Обработка исключений	Yehor Hromadskyi
17	Общий жизненный цикл приложения Xamarin.Forms? В зависимости от платформы!	Zverev Eugene
18	Относительная компоновка Xamarin	Ege Aydın
19	Отображать предупреждение	aboozz pallikara, GvSharma, Sreeraj, Yehor Hromadskyi
20	Плагин Xamarin	Eng Soon Cheah
21	Поведение на платформе	Ege Aydın
22	Пользовательские рендереры	Bonelol, hankide, Nicolas Bodin-Ripert, Nicolas Bodin-Ripert, nishantvodoo, Yehor Hromadskyi, Zverev Eugene
23	Пользовательские шрифты в стилях	Roma Rudyak
24	Последствия	Swaminathan Vetri
25	Почему Xamarin формирует и когда использовать Xamarin Forms	Daniel Krzyczkowski, mike
26	Работа с картами	Taras Shevchuk
27	Работа с локальными базами данных	Luis Beltran, Manohar
28	Связывание данных	Andrew, Matthew, Yehor Hromadskyi

29	Создание настраиваемых элементов управления	hvaughan3, spaceplane, Yehor Hromadskyi
30	Специальные визуальные корректировки для платформы	Alois, GalaxiaGuy, Paul
31	Страница Xamarin.Forms	Eng Soon Cheah
32	Тестирование блоков BDD в Xamarin.Forms	Ben Ishiyama-Levy
33	Тестирование устройства	jerone, Sergey Metlov
34	Триггеры и поведение	hamalaiv, hvaughan3
35	Услуги зависимостей	RIYAZ
36	Форматы макетов Xamarin	Eng Soon Cheah, Gerald Versluis, Lucas Moura Veloso
37	Храм Хамарина	Joehl
38	Чтение AppSettings в Xamarin.Forms	Ben Ishiyama-Levy, GvSharma