Бесплатная электронная книга

УЧУСЬ

Xamarin.iOS

Free unaffiliated eBook created from
Stack Overflow contributors.

#xamarin.io

S

Около

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

It is an unofficial and free Xamarin.iOS 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.iOS.

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.iOS

замечания

Xamarin.iOS позволяет создавать собственные приложения для iOS, используя те же элементы управления пользовательским интерфейсом, что и в Objective-C и Xcode, но с гибкостью и элегантностью современного языка (C #), мощью библиотеки базового класса . NET (BCL) , и два первоклассных IDE - Xamarin Studio и Visual Studio - у вас под рукой.

Дополнительную информацию об установке Xamarin.iOS на компьютере Mac или Windows см. В руководствах « Начало работы» в центре разработчика Xamarin

Версии

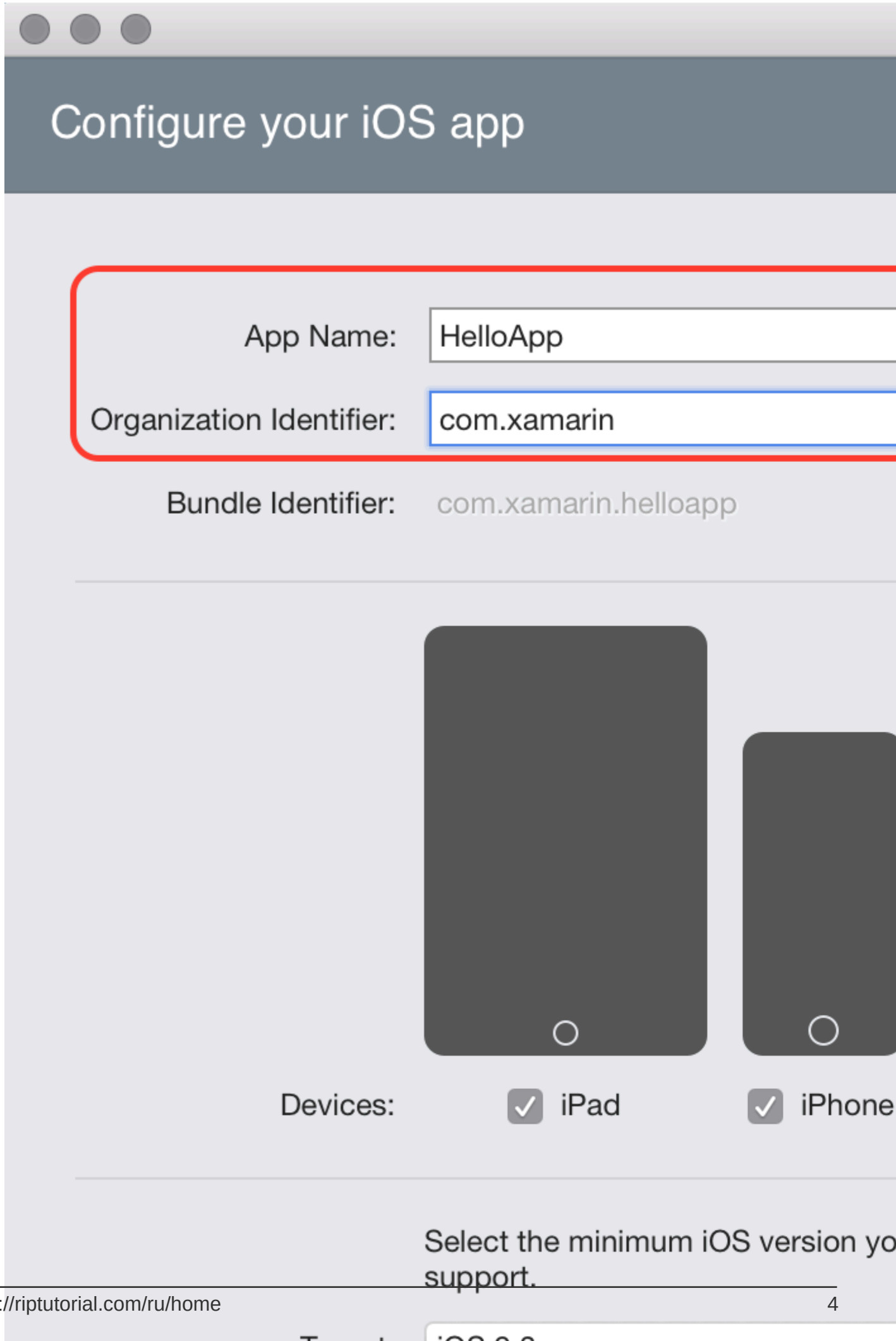
Версия	Дата выхода
1,0	2009-09-14
2,0	2010-04-05
3.0	2010-04-16
4,0	2011-04-06
5.0	2011-10-12
6,0	2012-09-19
7,0	2013-09-18
8,0	2014-09-10
9,0	2015-09-17
9,2	2015-11-17
9,4	2015-12-09
9,6	2016-03-22

Подробную информацию по каждой версии можно найти здесь: https://developer.xamarin.com/releases/ios/

Examples

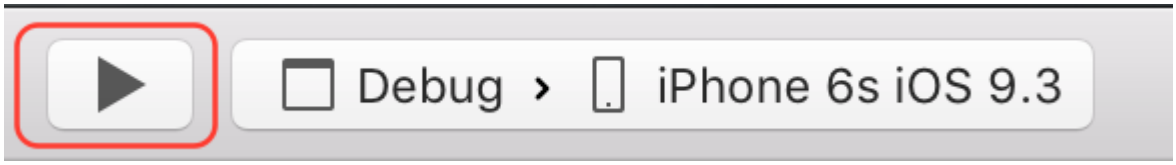
Начало работы в Xamarin Studio

1. Найдите **Файл> Создать> Решение,** чтобы открыть новый диалог проекта.
2. Выберите **приложение Single View** и нажмите **Next**
3. Настройте приложение, установив имя вашего приложения и идентификатор организации и нажмите « **Далее»** :

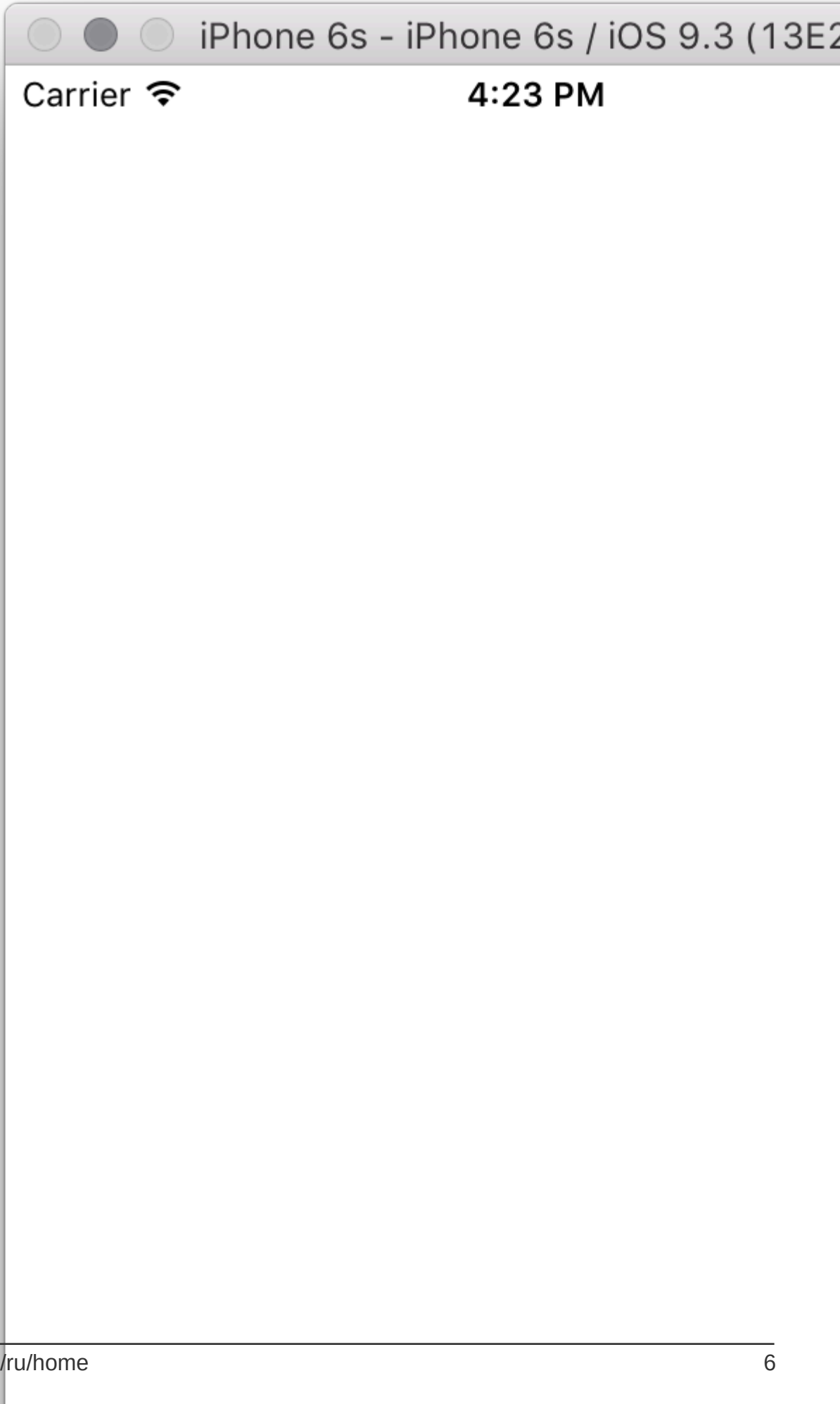

Configure your iOS app
App Name:
HelloApp
Organization Identifier:
com.xamarin
Bundle Identifier:
com.xamarin.helloapp
Devices:
iPad
iPhone
Select the minimum iOS version yo
support.

5. Чтобы запустить приложение, выберите Debug | iPhone 6s iOS 9.x и нажмите кнопку **воспроизведения** :

воспроизведения :

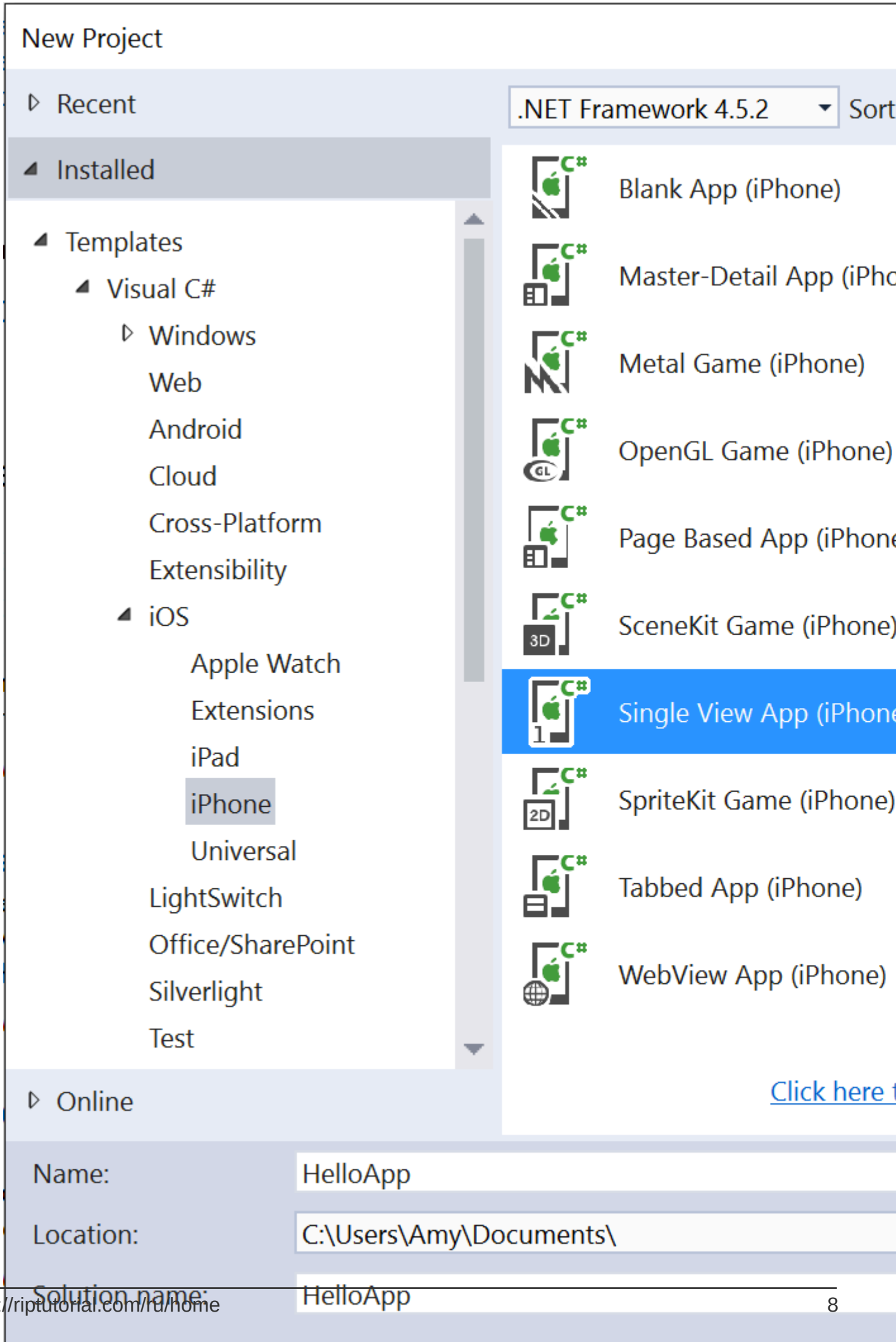


6. Это запустит iOS Simulator и отобразит ваше пустое приложение:

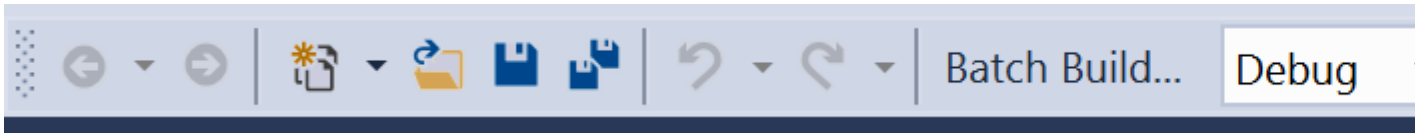

iPhone 6s - iPhone 6s / iOS 9.3 (13E2
Carrier
4:23 PM

чтобы открыть диалог «Новый проект».

2. Перейдите к Visual C #> iOS> iPhone и выберите Single View App:

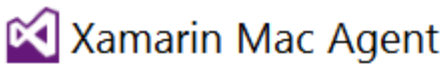

New Project
Recent
.NET Framework 4.5.2
Sort
Installed
Templates
Visual C#
Windows
Web
Android
Cloud
Cross-Platform
Extensibility
iOS
Apple Watch
Extensions
iPad
iPhone
Universal
LightSwitch
Office/SharePoint
Silverlight
Test
Online
Blank App (iPhone)
Master-Detail App (iPho
Metal Game (iPhone)
OpenGL Game (iPhone)
Page Based App (iPhone
SceneKit Game (iPhone)
Single View App (iPhone
SpriteKit Game (iPhone)
Tabbed App (iPhone)
WebView App (iPhone)
Click here t
Name:
HelloApp
Location:
C:\Users\Amy\Documents\
Solution name:
HelloApp

3. Дайте вашему приложению **имя** и нажмите **OK,** чтобы создать проект.
4. Выберите значок агента Mac на панели инструментов, как показано ниже:

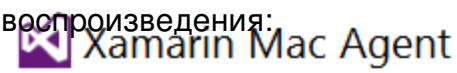


5. Выберите Mac, который будет создавать ваше приложение из списка (убедитесь, что Mac настроен для получения соединения!) И нажмите **Connect** :

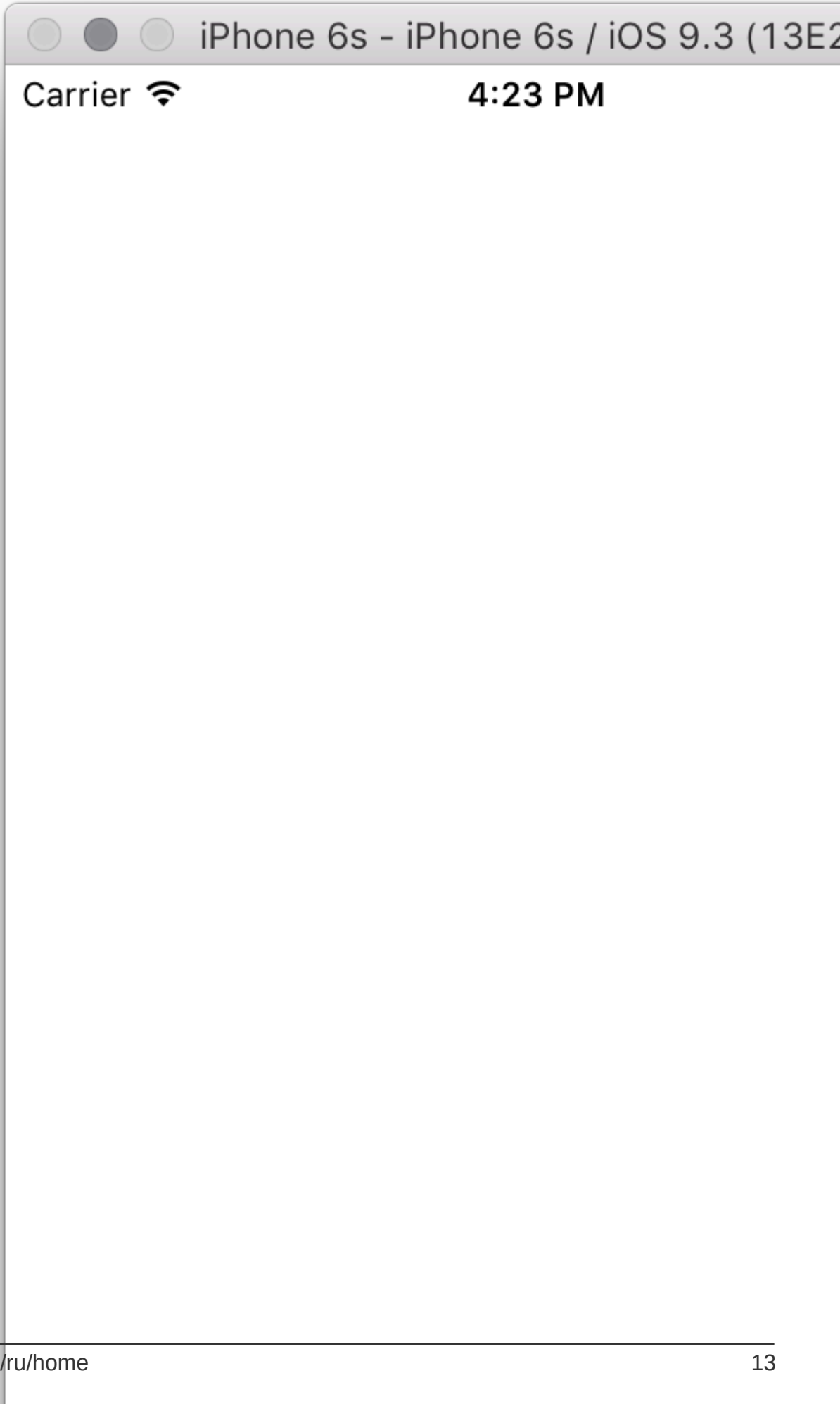
Mac настроен для получения соединения!) И нажмите **Connect** :



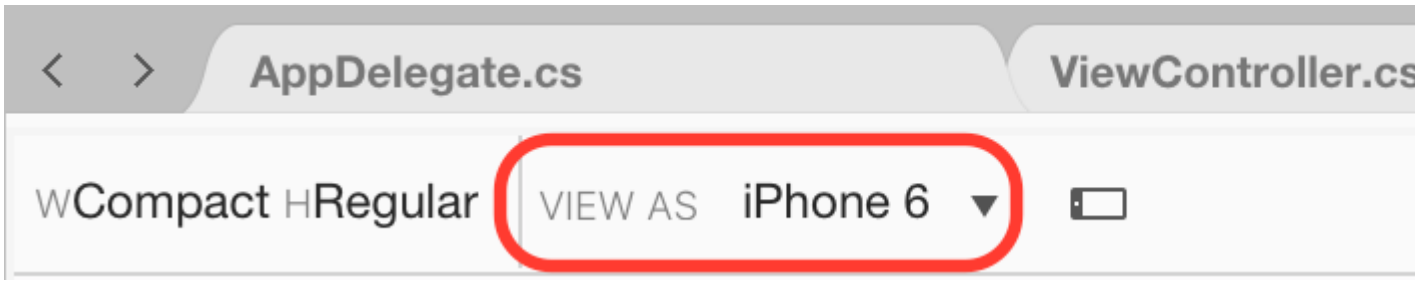
6. Чтобы запустить приложение, выберите **Debug | iPhoneSimulator** и нажмите кнопку воспроизведения:



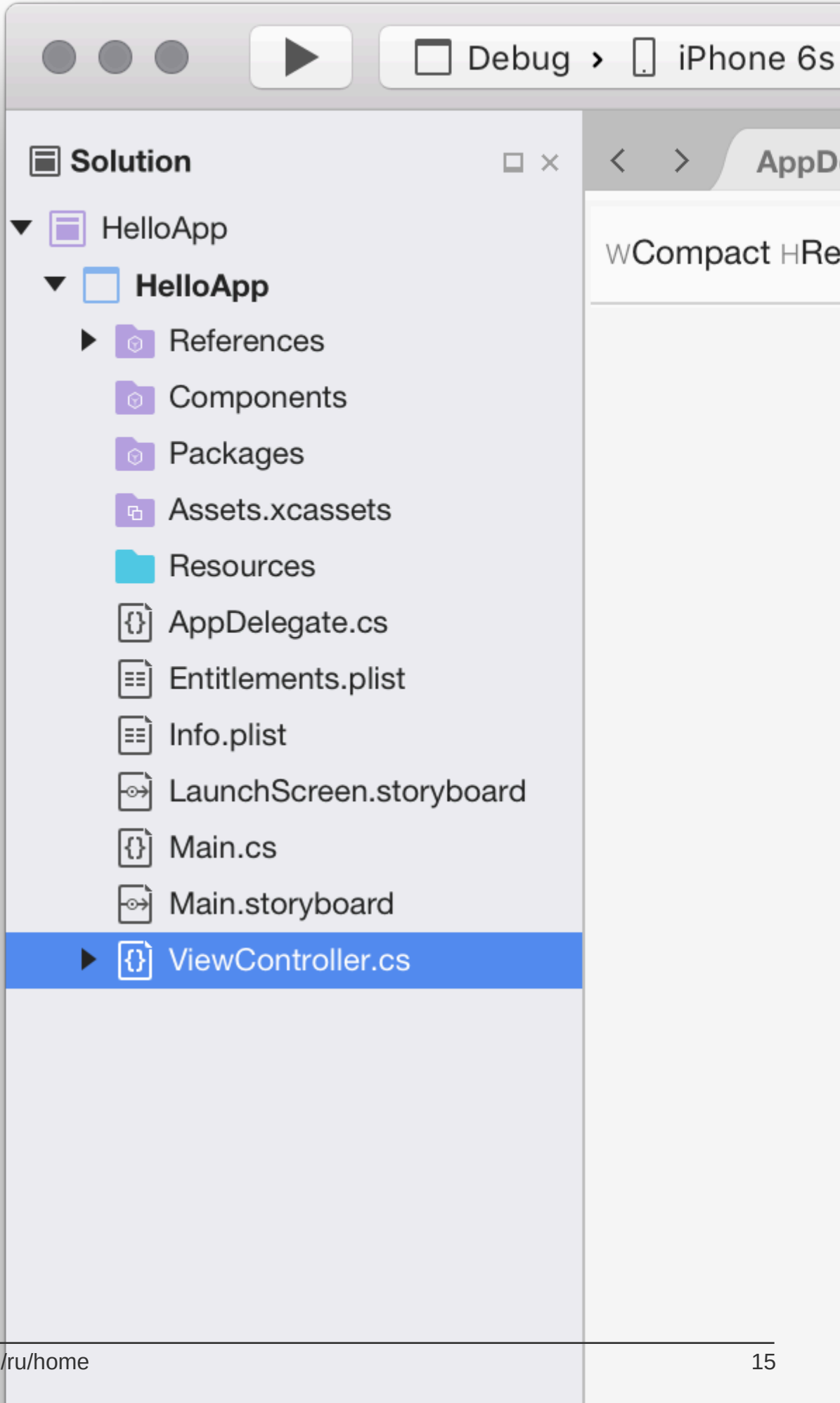
7. Это запустит iOS Simulator на Mac и отобразит ваше пустое приложение:


iPhone 6s - iPhone 6s / iOS 9.3 (13E2
Carrier
4:23 PM

2. Установить **представление** о iPhone 6:



3. Перетащите ярлык и кнопку из панели инструментов на поверхность дизайна, чтобы она выглядела следующим образом:


Debug › iPhone 6s
Solution
HelloApp
HelloApp
References
Components
Packages
Assets.xcassets
Resources
AppDelegate.cs
Entitlements.plist
Info.plist
LaunchScreen.storyboard
Main.cs
Main.storyboard
ViewController.cs
AppD
wCompact HRe

4. На панели «Свойства» дайте ярлыку и нажмите следующие свойства:

ничего такого	название	заглавие
этикетка	lblClicks	[Пусто]
кнопка	нажми на меня	Нажми на меня!

5. Добавьте следующий код в метод **ViewDidLoad** внутри класса **ViewController** :

```
clickMe.TouchUpInside += (sender, e) =>
{
    totalClicks++;
    if (totalClicks == 1)
    {
        lblClicks.Text = totalClicks + " Click";
    }

   else {
       lblClicks.Text = totalClicks + " Clicks";
   }
};
```

6. Запуск приложения

Прочитайте Начало работы с Xamarin.iOS онлайн: https://riptutorial.com/ru/xamarin-ios/topic/402/начало-работы-с-xamarin-ios

глава 2: UIImageView увеличить в сочетании с UIScrollView

замечания

UIImageView должен быть в scrollview, чтобы это работало.

Метод DoubleTap будет переключаться между minScale и doubleTapScale.

Examples

Двойное нажатие

```
private float minScale = 1f;
private float doubleTapScale = 2f;
private float maxScale = 4f;

private void SetUpDoubleTapZoom()
{
    imageViewToZoom.ContentMode = UIViewContentMode.ScaleAspectFit;
    scrollView.MaximumZoomScale = maxScale;
    scrollView.MinimumZoomScale = minScale;

    var doubleTap = new UITapGestureRecognizer(OnDoubleTap)
    {
        NumberOfTapsRequired = 2
    };

    scrollView.AddGestureRecognizer(doubleTap);
}

private void OnDoubleTap(UIGestureRecognizer gesture)
{
    scrollView.ZoomScale = (scrollView.ZoomScale.Equals(minScale)) ? doubleTapScale :
minScale;
}
```

Усиление зума

```
private float minScale = 1f;
private float maxScale = 4f;

private void SetUpPinchGestureZoom()
{
    imageViewToZoom.ContentMode = UIViewContentMode.ScaleAspectFit;

    scrollView.MaximumZoomScale = maxScale;
    scrollView.MinimumZoomScale = minScale;

    scrollView.ViewForZoomingInScrollView += (UIScrollView sv) => { return imageViewToZoom; };
```

```
}
```

Прочитайте UIImageView увеличить в сочетании с UIScrollView онлайн:
https://riptutorial.com/ru/xamarin-ios/topic/6346/uiimageview-увеличить-в-сочетании-с-uiscrollview

глава 3: Автозаполнение Xalarin iOS Google Места

Вступление

Начиная с начала работы с Xamarin было много вещей, которые я хотел бы, чтобы кто-то еще задокументировал. Здесь я объясню, как использовать 1 аспект автозаполнения мест google, элемент управления пользовательского интерфейса, использующий контроллер результатов. Хотя этот код основан на примерах Google, преобразованных в C # из Swift, я старался изо всех сил убедиться, что это полностью рабочий пример. Надеюсь, что эта документация поможет другим.

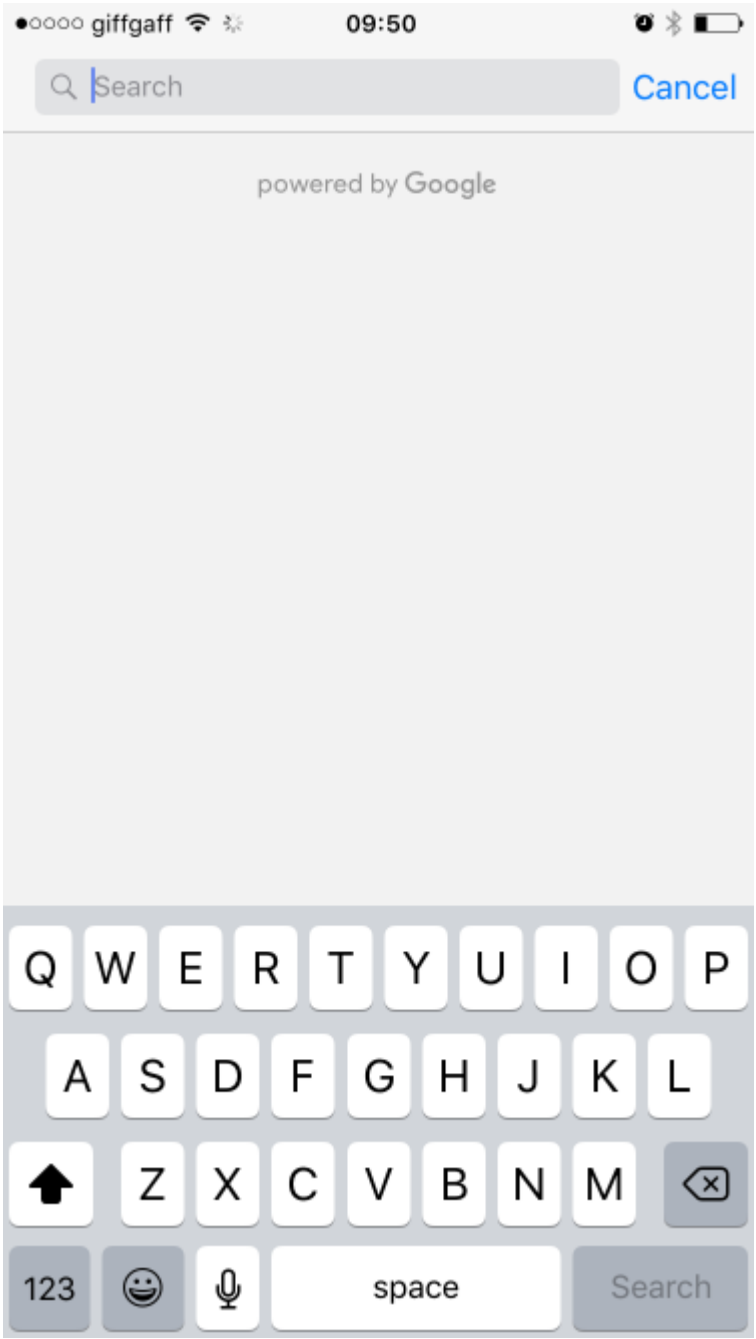
Examples

Добавьте элемент управления пользовательского интерфейса автозаполнения с помощью контроллера результатов.

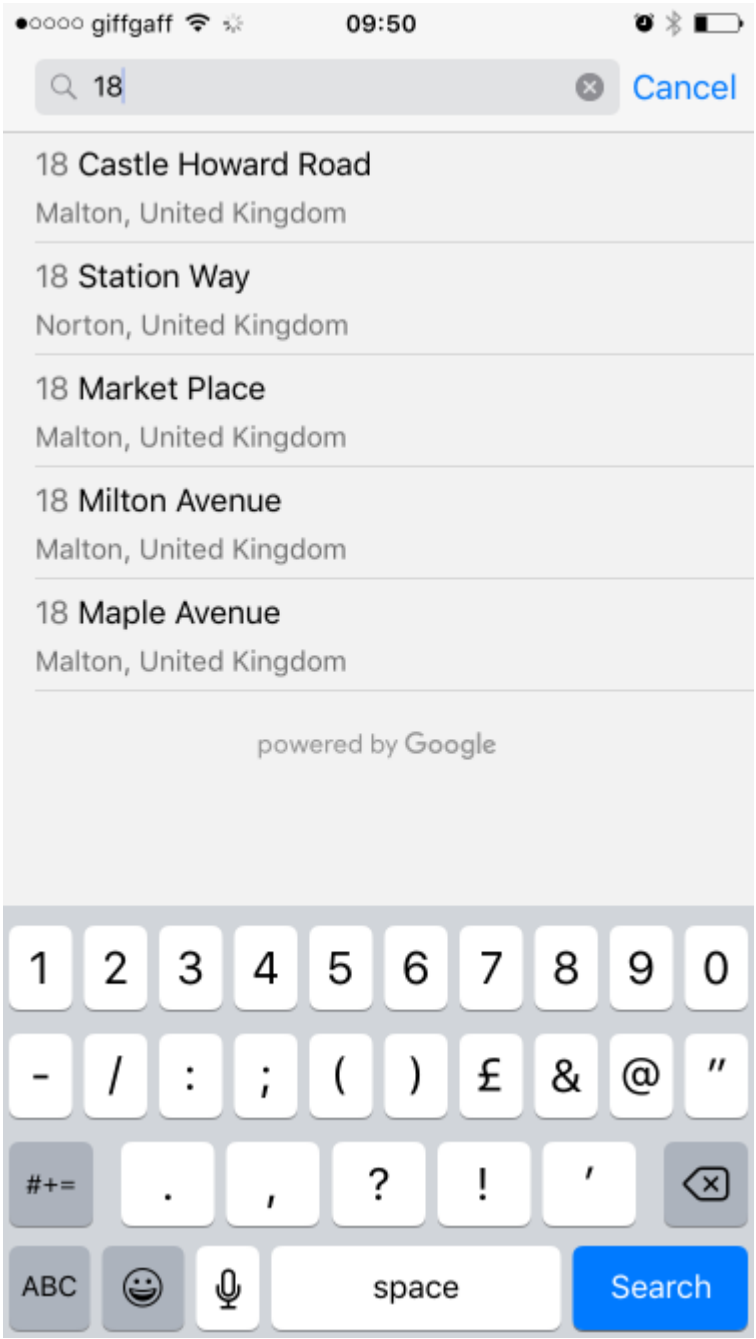
Элемент управления автозаполнения - это диалоговое окно поиска со встроенной функцией автозаполнения. Когда пользователь вводит поисковые запросы, элемент управления представляет список прогнозируемых мест на выбор. Когда пользователь делает выбор, возвращается экземпляр GMSPlace (место в Xamarin), который ваше приложение затем может использовать для получения сведений о выбранном месте.

Как уже упоминалось выше, в этом примере используется контроллер результатов, который позволяет больше контролировать интерфейс ввода текста. Контроллер результатов будет динамически переключать видимость списка результатов на основе фокуса входного интерфейса.

Цель этого кода - отобразить экран, как показано ниже:



Это приведет к автозаполнению вашего адреса при вводе текста, как показано на рисунке ниже:



Инструкции:

1. Сначала нам нужно добавить API-интерфейс google в нашу Visual Studio, он доступен через Nuget, просто найдите Xamarin.Google.iOS.Maps, добавьте его в свой проект iOS, в качестве альтернативы вы можете скачать его с Xamarin Xamarin Google Maps iOS SDK

2. Нам нужно что-то вроде кнопки, чтобы вызвать контроллер просмотра автозаполнения google. В этом примере я делаю это с доской истории и добавил кнопку с именем GoogleButton, вы можете запустить ее с кодом, это не имеет большого значения.

3. В представлении ViewDidLoad в классе контроллеров вашего вида добавьте следующий код. В моем примере ниже я не использую фактическое местоположение

мобильных устройств, мое окончательное решение, конечно же, сделало бы это, но это был тест, и я не хотел реализовывать дополнительный код, пока не доказал, что это сработало или разбавит то, что я пытаюсь показать тебе:

// Код, чтобы открыть автозаполненный контроллер просмотра google.

```
GoogleButton.TouchUpInside += (sender, ea) =>
{

    var FakeCoordinates = new CLLocationCoordinate2D()
    {
        Latitude = 54.135364,
        Longitude = -0.797888
    };

    var north = LocationWithBearing(45, 3000, FakeCoordinates);
    var east = LocationWithBearing(225, 3000, FakeCoordinates);

    var autocompleteController = new AutocompleteViewController();
    autocompleteController.Delegate = new AutoCompleteDelegate();
    autocompleteController.AutocompleteBounds = new CoordinateBounds(north, east);
    PresentViewController(autocompleteController, true, null);

};
```

4. Это необязательно, но я добавил функцию для вычисления локальных границ, я прохожу через 3000, это число находится в метрах, поэтому, если вы хотите, чтобы более крупные начальные границы не изменялись, обратите внимание, что поиск по-прежнему будет найти любой адрес в мире, он просто взвешивает первоначальные результаты, чтобы эти локальные области сначала ограничивались. Эта функция была заимствована из сообщения переполнения стека, я преобразовал ее из Swift в C # для наших целей:

```
public CLLocationCoordinate2D LocationWithBearing(Double bearing, Double distanceMeters,
CLLocationCoordinate2D origin)
{
    var distRadians = distanceMeters/(6372797.6);

    var rbearing = bearing*Math.PI/180.0;

    var lat1 = origin.Latitude*Math.PI/180;
    var lon1 = origin.Longitude*Math.PI/180;

    var lat2 = Math.Asin(Math.Sin(lat1)*Math.Cos(distRadians) +
Math.Cos(lat1)*Math.Sin(distRadians)*Math.Cos(rbearing));
    var lon2 = lon1 + Math.Atan2(Math.Sin(rbearing)*Math.Sin(distRadians)*Math.Cos(lat1),
                   Math.Cos(distRadians) - Math.Sin(lat1)*Math.Sin(lat2));

    return new CLLocationCoordinate2D(latitude: lat2*180/ Math.PI, longitude:
lon2*180/Math.PI);
}
```

5. Этот окончательный фрагмент кода является делегатом для автозаполнения, нам нужен этот делегат для обработки всего того, что google вернется к нам:

```
public  class AutoCompleteDelegate : AutocompleteViewControllerDelegate
{

    public override void DidFailAutocomplete(AutocompleteViewController viewController,
NSError error)
    {
        // TODO: handle the error.
        Debug.Print("Error: " + error.Description);
    }


    public override void DidAutocomplete(AutocompleteViewController viewController, Place
place)
    {
        Debug.Print(place.Name);
        Debug.Print(place.FormattedAddress);

        viewController.DismissViewController(true, null);
    }

    public override void DidRequestAutocompletePredictions(AutocompleteViewController
viewController)
    {
        UIApplication.SharedApplication.NetworkActivityIndicatorVisible = true;
    }

    public override void DidUpdateAutocompletePredictions(AutocompleteViewController
viewController)
    {
        UIApplication.SharedApplication.NetworkActivityIndicatorVisible = true;
    }

    public override void WasCancelled(AutocompleteViewController viewController)
    {
        viewController.DismissViewController(true, null);
    }

}
```

Запустите свой проект, и он должен работать отлично, этот пример довольно сфокусирован, но, надеюсь, он покажет вам базовый пример того, как любой из элементов управления автозаполнением Google будет работать. Спасибо!

Прочитайте Автозаполнение Xalarin iOS Google Места онлайн:
https://riptutorial.com/ru/xamarin-ios/topic/9041/автозаполнение-xalarin-ios-google-места

глава 4: Автоматический макет в Xamarin.iOS

Examples

Добавление ограничений с помощью анкеров IOS 9+

9,0

```
// Since the anchor system simply returns constraints, you still need to add them somewhere.
View.AddConstraints(
    new[] {
        someLabel.TopAnchor.ConstraintEqualTo(TopLayoutGuide.GetBottomAnchor()),
        anotherLabel.TopAnchor.ConstraintEqualTo(someLabel.BottomAnchor, 6),
        oneMoreLabel.TopAnchor.ConstraintEqualTo(anotherLabel.BottomAnchor, 6),

oneMoreLabel.BottomAnchor.ConstraintGreaterThanOrEqualTo(BottomLayoutGuide.GetTopAnchor(), -
10),
    }
);
```

Добавление ограничений с использованием языка Visual Format (VFL)

```
// Using Visual Format Language requires a special look-up dictionary of names<->views.
var views = new NSDictionary(
    nameof(someLabel), someLabel,
    nameof(anotherLabel), anotherLabel,
    nameof(oneMoreLabel), oneMoreLabel
);
// It can also take a look-up dictionary for metrics (such as size values).
// Since we are hard-coding those values in this example, we can give it a `null` or empty
dictionary.
var metrics = (NSDictionary)null;

// Add the vertical constraints to stack everything together.
// `V:` = vertical
// `|…|` = constrain to super view (`View` for this example)
// `-10-` = connection with a gap of 10 pixels (could also be a named parameter from the
metrics dictionary)
// `-[viewName]-` = connection with a control by name looked up in views dictionary (using C#
6 `nameof` for refactoring support)
var verticalConstraints = NSLayoutConstraint.FromVisualFormat(
    $"V:|-20-[{nameof(someLabel)}]-6-[{nameof(anotherLabel)}]-6-[{nameof(oneMoreLabel)}]->=10-
|",
    NSLayoutFormatOptions.AlignAllCenterX,
    metrics,
    views
);
View.AddConstraints(verticalConstraints);
```

Вы можете найти некоторые типы ограничений, такие как пропорции , не могут

передаваться в синтаксисе языка Visual Format (VFL) и должны напрямую обращаться к соответствующим методам.

Использование Cirrious.FluentLayout

Использование NuGet

```
Install-Package Cirrious.FluentLayout
```

Расширенный пример, основанный на примере стартера на странице GitHub , простое имя, ярлыки и поля имен, все они сложены друг над другом:

```
public override void ViewDidLoad()
{
    //create our labels and fields
    var firstNameLabel = new UILabel();
    var lastNameLabel = new UILabel();
    var firstNameField = new UITextField();
    var lastNameField = new UITextField();

    //add them to the View
    View.AddSubviews(firstNameLabel, lastNameLabel, firstNameField, lastNameField);

    //create constants that we can tweak if we do not like the final layout
    const int vSmallMargin = 5;
    const int vMargin = 20;
    const int hMargin = 10;

    //add our constraints
    View.SubviewsDoNotTranslateAutoresizingMaskIntoConstraints();
    View.AddConstraints(
        firstNameLabel.WithSameTop(View).Plus(vMargin),
        firstNameLabel.AtLeftOf(View).Plus(hMargin),
        firstNameLabel.WithSameWidthOf(View),

        firstNameField.WithSameWidth(firstNameLabel),
        firstNameField.WithSameLeft(firstNameLabel),
        firstNameField.Below(firstNameLabel).Plus(vSmallMargin),

        lastNameLabel.Below(firstNameField).Plus(vMargin),
        lastNameLabel.WithSameLeft(firstNameField),
        lastNameLabel.WithSameWidth(firstNameField),

        lastNameField.Below(lastNameLabel).Plus(vSmallMargin),
        lastNameField.WithSameWidth(lastNameLabel),
        lastNameField.WithSameLeft(lastNameLabel));
}
```

Добавление ограничений с масонством

Масонство - это библиотека для объекта-с, но xamarin создали для нее привязку и создали ее как пакет nuget https://www.nuget.org/packages/Masonry/ .

Установка Nuget

```
Install-Package Masonry
```

Это центрирует кнопку на 100 пунктов ниже центральной точки содержащего представления и устанавливает ширину между 200 и 400 точками

```
this.loginBtn.MakeConstraints(make =>
{
    make.Width.GreaterThanOrEqualTo(new NSNumber(200));
    make.Width.LessThanOrEqualTo(new NSNumber(400));
    make.Center.EqualTo(this.View).CenterOffset(new CGPoint(0, 100));
});
```

Это устанавливает масштабированное изображение на 100 точек над центральной точкой содержащего представления, а затем устанавливает ширину в ширину содержащего представления с помощью умножителя 0,5, что означает 50% ширины. Затем он устанавливает высоту в ширину, умноженную на соотношение сторон, которое заставляет изображение масштабироваться, но сохраняя правильное соотношение сторон

```
this.logo.MakeConstraints(make =>
{
    make.Center.EqualTo(this.View).CenterOffset(new CGPoint(0, -100));
    make.Width.EqualTo(this.View).MultipliedBy(0.5f);
    make.Height.EqualTo(this.logo.Width()).MultipliedBy(0.71f);
});
```

Прочитайте Автоматический макет в Xamarin.iOS онлайн: https://riptutorial.com/ru/xamarin-ios/topic/1317/автоматический-макет-в-xamarin-ios

глава 5: Вычисление переменной высоты строки в GetHeightForRow

замечания

Вычисление высоты строк может быть дорогостоящим, и производительность прокрутки может пострадать, если у вас больше данных. В этом случае переопределите `UITableViewSource.EstimatedHeight(UITableView, NSIndexPath)` чтобы быстро предоставить число, достаточное для быстрой прокрутки, например:

```
public override nfloat EstimatedHeight(UITableView tableView, NSIndexPath indexPath)
{
    return 44.0f;
}
```

Examples

Использование GetHeightForRow

Чтобы установить пользовательскую высоту строки, переопределите `UITableViewSource.GetHeightForRow(UITableView,NSIndexPath)` :

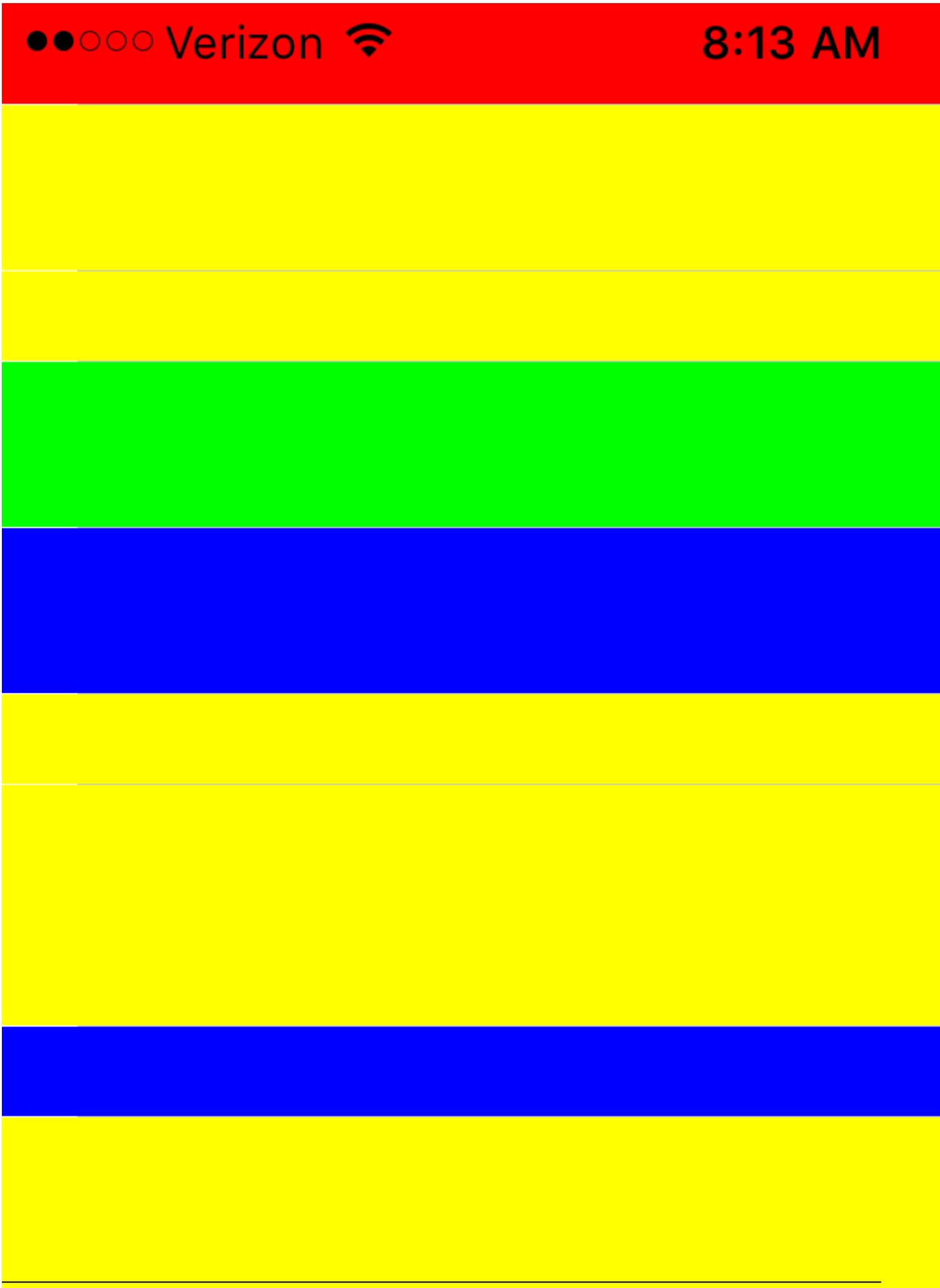
```
public class ColorTableDataSource : UITableViewSource
{
    List<DomainClass> Model { get; set; }

    public override nfloat GetHeightForRow(UITableView tableView, NSIndexPath indexPath)
    {
        var height = Model[indexPath.Row % Model.Count].Height;
        return height;
    }

    //...etc ...
}
```

Класс домена для таблицы (в этом случае он имеет 1 из 3 случайных цветов и 1 из 3 случайных высот):

```
public class DomainClass
{
    static Random rand = new Random(0);
    public UIColor Color { get; protected set; }
    public float Height { get; protected set; }

    static UIColor[] Colors = new UIColor[]
    {
        UIColor.Red,
```

```
        UIColor.Green,
        UIColor.Blue,
        UIColor.Yellow
    };

    public DomainClass()
    {
        Color = Colors[rand.Next(Colors.Length)];
        switch (rand.Next(3))
        {
            case 0:
                Height = 24.0f;
                break;
            case 1:
                Height = 44.0f;
                break;
            case 2:
                Height = 64.0f;
                break;
            default:
                throw new ArgumentOutOfRangeException();
        }
    }

    public override string ToString()
    {
        return string.Format("[DomainClass: Color={0}, Height={1}]", Color, Height);
    }
}
```

Это выглядит так:


Verizon
8:13 AM

https://riptutorial.com/ru/xamarin-ios/topic/6515/вычисление-переменной-высоты-строки-в-getheightforrow

глава 6: Добавить PullToRefresh в UITableView

замечания

Ссылки на объекты:

Таблица UITableView;
TableSource tableSource;
bool useRefreshControl = false;
UIRefreshControl RefreshControl;
List tableItems;

TableSource и TableItem являются пользовательскими классами

Для полной выборки вы можете использовать fork: https://github.com/adiiaditya/Xamarin.iOS-Samples/tree/master/PullToRefresh

Examples

Добавление UIRefreshControl в UITableView

```
public override async void ViewDidLoad(){
base.ViewDidLoad();
// Perform any additional setup after loading the view, typically from a nib.

Title = "Pull to Refresh Sample";
table = new UITableView(new CGRect(0, 20, View.Bounds.Width, View.Bounds.Height - 20));
//table.AutoresizingMask = UIViewAutoresizing.All;
tableItems = new List<TableItem>();
tableItems.Add(new TableItem("Vegetables") { ImageName = "Vegetables.jpg" });
tableItems.Add(new TableItem("Fruits") { ImageName = "Fruits.jpg" });
tableItems.Add(new TableItem("Flower Buds") { ImageName = "Flower Buds.jpg" });
tableItems.Add(new TableItem("Legumes") { ImageName = "Legumes.jpg" });
tableItems.Add(new TableItem("Tubers") { ImageName = "Tubers.jpg" });
tableSource = new TableSource(tableItems);
table.Source = tableSource;

await RefreshAsync();

AddRefreshControl();

Add(table);
table.Add(RefreshControl);
}

async Task RefreshAsync()
{
// only activate the refresh control if the feature is available
```

```
if (useRefreshControl)
    RefreshControl.BeginRefreshing();

if (useRefreshControl)
    RefreshControl.EndRefreshing();

    table.ReloadData();
}

#region * iOS Specific Code
// This method will add the UIRefreshControl to the table view if
// it is available, ie, we are running on iOS 6+
void AddRefreshControl()
{
if (UIDevice.CurrentDevice.CheckSystemVersion(6, 0))
{
    // the refresh control is available, let's add it
    RefreshControl = new UIRefreshControl();
    RefreshControl.ValueChanged += async (sender, e) =>
    {
        tableItems.Add(new TableItem("Bulbs") { ImageName = "Bulbs.jpg" });
        await RefreshAsync();
    };
    useRefreshControl = true;
   }
}
#endregion
```

Прочитайте Добавить PullToRefresh в UITableView онлайн: https://riptutorial.com/ru/xamarin-ios/topic/6565/добавить-pulltorefresh-в-uitableview

глава 7: Добавить панель поиска в UITableView

замечания

Ссылки на объекты:

Таблица UITableView;
TableSource tableSource;
List tableItems;
UISearchBar searchBar;

Чтобы развернуть полный образец: https://github.com/adiiaditya/Xamarin.iOS-Samples/tree/master/SearchBarWithTableView

Examples

Добавить UISearchBar в UITableView

```
public override void ViewDidLoad()
{
    base.ViewDidLoad();
    // Perform any additional setup after loading the view, typically from a nib.

    //Declare the search bar and add it to the header of the table
    searchBar = new UISearchBar();
    searchBar.SizeToFit();
    searchBar.AutocorrectionType = UITextAutocorrectionType.No;
    searchBar.AutocapitalizationType = UITextAutocapitalizationType.None;
    searchBar.TextChanged += (sender, e) =>
    {
        //this is the method that is called when the user searches
        searchTable();
    };

    Title = "SearchBarWithTableView Sample";
    table = new UITableView(new CGRect(0, 20, View.Bounds.Width, View.Bounds.Height - 20));
    //table.AutoresizingMask = UIViewAutoresizing.All;
    tableItems = new List<TableItem>();

    tableItems.Add(new TableItem("Vegetables") { ImageName = "Vegetables.jpg" });
    tableItems.Add(new TableItem("Fruits") { ImageName = "Fruits.jpg" });
    tableItems.Add(new TableItem("Flower Buds") { ImageName = "Flower Buds.jpg" });
    tableItems.Add(new TableItem("Legumes") { ImageName = "Legumes.jpg" });
    tableItems.Add(new TableItem("Tubers") { ImageName = "Tubers.jpg" });
    tableSource = new TableSource(tableItems);
    table.Source = tableSource;
    table.TableHeaderView = searchBar;
    Add(table);
}
```

```
private void searchTable()
{
    //perform the search, and refresh the table with the results
    tableSource.PerformSearch(searchBar.Text);
    table.ReloadData();
}
```

Класс TableSource будет выглядеть следующим образом:

```
public class TableSource : UITableViewSource
{
    private List<TableItem> tableItems = new List<TableItem>();
    private List<TableItem> searchItems = new List<TableItem>();
    protected string cellIdentifier = "TableCell";

    public TableSource(List<TableItem> items)
    {
        this.tableItems = items;
        this.searchItems = items;
    }

    public override nint RowsInSection(UITableView tableview, nint section)
    {
        return searchItems.Count;
    }

    public override UITableViewCell GetCell(UITableView tableView, NSIndexPath indexPath)
    {
        // request a recycled cell to save memory
        UITableViewCell cell = tableView.DequeueReusableCell(cellIdentifier);


        var cellStyle = UITableViewCellStyle.Default;

        // if there are no cells to reuse, create a new one
        if (cell == null)
        {
            cell = new UITableViewCell(cellStyle, cellIdentifier);
        }

        cell.TextLabel.Text = searchItems[indexPath.Row].Title;
        cell.ImageView.Image = UIImage.FromFile("Images/" +
searchItems[indexPath.Row].ImageName);

        return cell;
    }

    public override nint NumberOfSections(UITableView tableView)
    {
        return 1;
    }

    public void PerformSearch(string searchText)
    {
        searchText = searchText.ToLower();
        this.searchItems = tableItems.Where(x =>
x.Title.ToLower().Contains(searchText)).ToList();
    }
}
```

Прочитайте Добавить панель поиска в UITableView онлайн: https://riptutorial.com/ru/xamarin-ios/topic/6540/добавить-панель-поиска-в-uitableview

глава 8: Добавление UIRefreshControl в представление таблицы

Examples

Добавление UIRefreshControl в TableView

Предположения:

TableView - ссылка на TableView

DataSource - это класс, который наследует UITableViewSource

DataSource.Objects - это открытый список <object> (), доступный для UIViewController

```
private UIRefreshControl refreshControl;

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

    // Set the DataSource for the TableView
    TableView.Source = dataSource = new DataSource(this);

    // Create the UIRefreshControl
    refreshControl = new UIRefreshControl();

    // Handle the pullDownToRefresh event
    refreshControl.ValueChanged += refreshTable;

    // Add the UIRefreshControl to the TableView
    TableView.AddSubview(refreshControl);
}

private void refreshTable(object sender, EventArgs e)
{
    fetchData();
    refreshControl.EndRefreshing();
    TableView.ReloadData();
}

private void fetchData()
{
    var objects = new List<object>();
    // fetch data and store in objects.
    dataSource.Objects = objects;
}
```

Прочитайте Добавление UIRefreshControl в представление таблицы онлайн: https://riptutorial.com/ru/xamarin-ios/topic/4642/добавление-uirefreshcontrol-в-представление-таблицы

глава 9: Добавление UIRefreshControl в представление таблицы

Examples

Добавить простой UIRefreshControl в UIScrollView

Мы предполагаем полностью работающее `UIScrollview` именем `_scrollView` ;

Обратите внимание, что `UITableView` , `UICollectionView` также являются scrollviews, поэтому следующие примеры будут работать над этими элементами пользовательского интерфейса.

Во-первых, создание и распределение

```
UIRefreshControl refreshControl = new UIRefreshControl();
```

Во-вторых, подключение события обновления к методу. Существуют разные способы сделать это.

Стиль 1:

```
refreshControl.ValueChanged += (object sender, EventArgs e) => MyMethodCall();
```

Стиль 2:

```
refreshControl.ValueChanged += (object sender, EventArgs e) =>
{
    //Write code here
};
```

Стиль 3:

```
refreshControl.ValueChanged += HandleRefreshValueChanged;

void HandleRefreshValueChanged(object sender, EventArgs e)
{
    //Write code here
}
```

Третий и последний, добавив контроль обновления непосредственно в наш scrollview.

```
_scrollView.AddSubview(refreshControl);
```

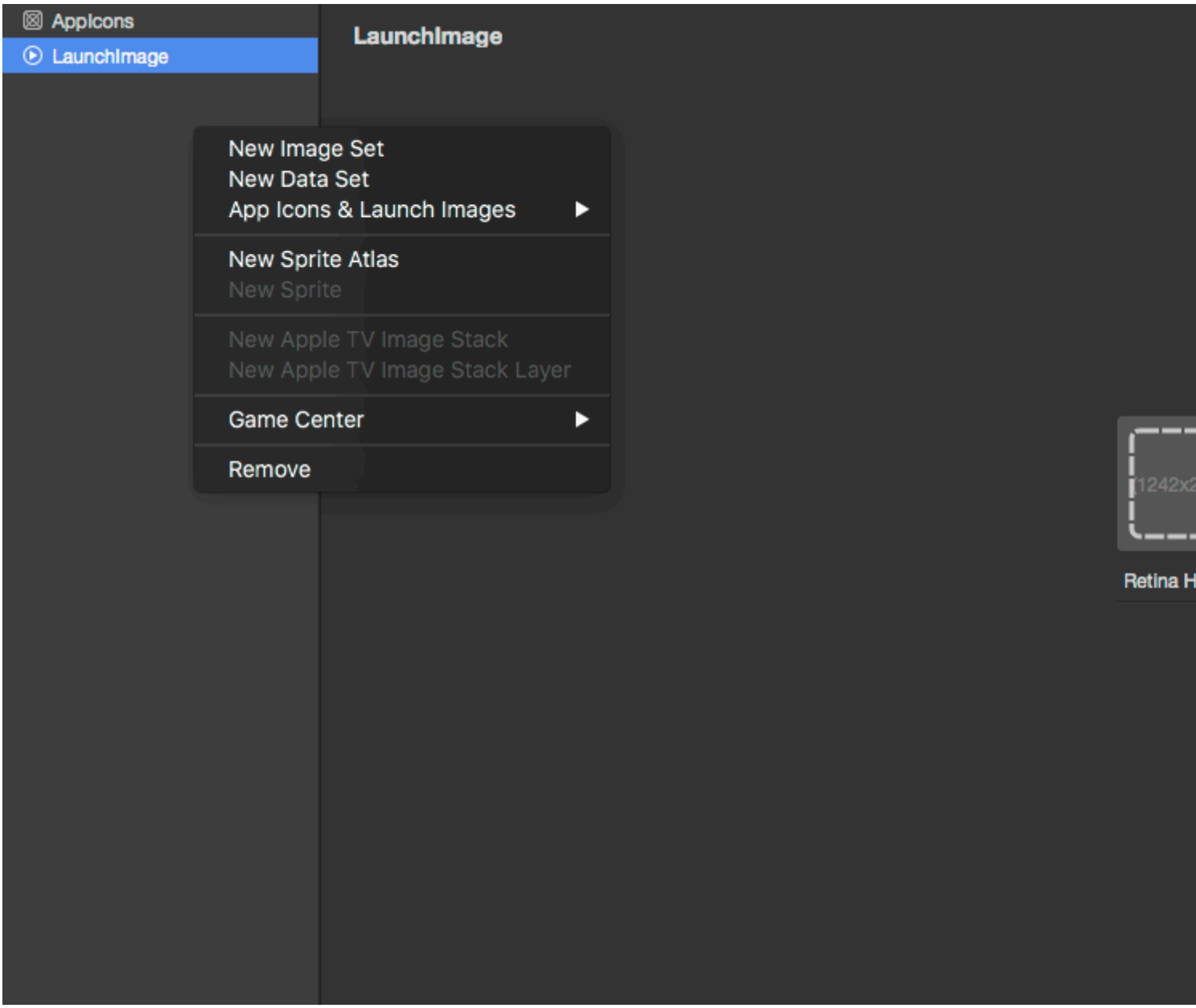
Прочитайте Добавление UIRefreshControl в представление таблицы онлайн: https://riptutorial.com/ru/xamarin-ios/topic/8371/добавление-uirefreshcontrol-в-представление-таблицы

глава 10: Использование каталогов активов

Examples

Добавление объектов изображения в каталог активов

Вот как выглядит Каталог активов в Xamarin Studio,



Как показано на рисунке выше, существует 5 типов активов, которые вы можете создать в каталоге.

Я покрою только набор изображений, потому что он самый простой.

Когда вы создаете новый набор изображений. Вы получите такие варианты

Чтобы добавить изображения в каталог, вы можете просто щелкнуть по пунктирным квадратам и выбрать изображение, которое вы хотите установить для определенного параметра.

В XCode у вас есть опции 1x, 2x и 3x для покрытия последних размеров экрана устройства iOS. Но у Xamarin есть один дополнительный вариант Vector, с помощью которого вы можете загрузить PDF-форматированное векторное изображение, которое будет автоматически масштабироваться в зависимости от устройства, на котором работает ваше приложение.

Для iPhone-изображений Xamarin сохраняет специальный размер i4 для iOS7, который используется для iPhone с 4-дюймовым экраном (5, 5S и SE).

Пожалуйста, обратитесь к документации Xamarin о том, как добавить изображения в приложение iOS для получения дополнительной информации.

Прочитайте Использование каталогов активов онлайн: https://riptutorial.com/ru/xamarin-ios/topic/6630/использование-каталогов-активов

глава 11: Использование каталогов активов iOS для управления изображениями

замечания

Каталоги активов - это способ управления несколькими разрешениями объектов изображения iOS. Чтобы отображать оптимальные изображения, iOS использует 1x, 2x и 3 x версии каждого изображения в зависимости от плотности экрана устройства. Версия 1x предназначена только для очень старых устройств без сетчатки, поэтому нет необходимости в приложениях, поддерживающих только iOS 9.

Каталоги активов помогут поддерживать прореживание и нарезку приложений, оптимизируя ресурсы, которые пользователи загружают для установки приложения из App Store.

Examples

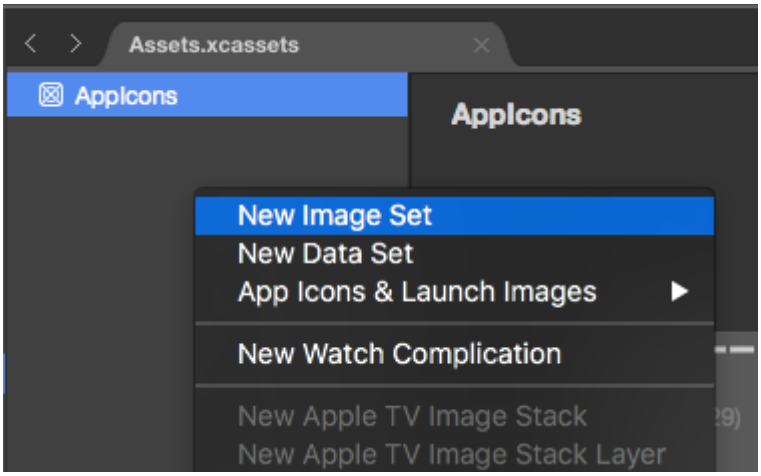
Загрузка изображения каталога активов

Загрузите изображение из каталога активов с помощью `UIImage.FromBundle(string imageName)`

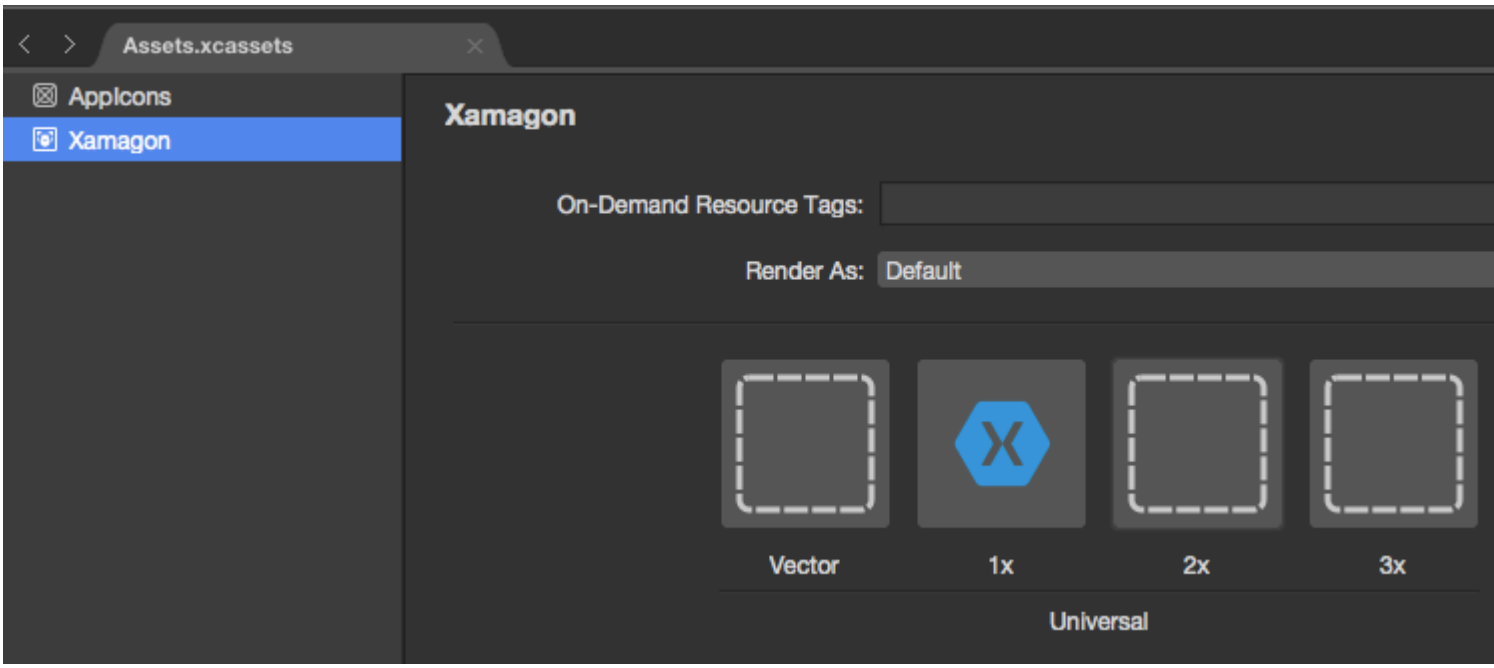
```
UIImage image = UIImage.FromBundle("ImageName");
// use the name of the image set from the asset catalog
```

Вы можете использовать изображение для `UIImageView` или что-то еще, что вам нужно сделать.

Управление изображениями в каталоге активов



Каталоги активов позволяют управлять изображениями, значками приложений и запусками изображений. Набор изображений используется для изображений, отображаемых в приложении. Универсальные изображения, как правило, являются лучшим вариантом. Вы можете либо использовать векторное изображение (например, PDF), которое будет масштабироваться для всех экранов, либо включать 1x, 2x и 3x вариант, и iOS выберет соответствующую версию изображения для текущего устройства пользователя.



Вы можете изменить имя любого набора в каталоге активов, дважды щелкнув по имени. Изображения могут быть добавлены путем перетаскивания или щелчка по изображению, которое вы хотите заполнить для выбора файлов.

Добавление образов каталога объектов в раскадровку

Изображения каталога объектов могут использоваться из раскадровки, как любой другой вид изображения, добавленный в проект. Они будут автоматически заполнены как опция в `UIImageView` и других представлениях, которые поддерживают добавление изображения.

Прочитайте Использование каталогов активов iOS для управления изображениями онлайн: https://riptutorial.com/ru/xamarin-ios/topic/6241/использование-каталогов-активов-ios-для-управления-изображениями

глава 12: Как использовать каталоги активов

Examples

Использование каталогов активов

Чтобы использовать Каталог активов, вам необходимо сделать следующее:

1. Дважды щелкните файл Info.plist в обозревателе решений, чтобы открыть его для редактирования.
2. Прокрутите вниз до раздела «Иконки приложений».
3. В раскрывающемся списке «Источник» убедитесь, что выбраны AppIcons.
4. В обозревателе решений дважды щелкните файл Assets.xcassets, чтобы открыть его для редактирования.
5. Выберите AppIcons из списка активов, чтобы отобразить Редактор значков.
6. Или нажмите на указанный тип значка и выберите файл изображения для требуемого типа / размера или перетащите изображение из папки и опустите его на нужный размер.
7. Нажмите кнопку «Открыть», чтобы включить изображение в проект и установить его в xcasset.

Прочитайте Как использовать каталоги активов онлайн: https://riptutorial.com/ru/xamarin-ios/topic/6539/как-использовать-каталоги-активов

глава 13: Методы изменения размера для UIImage

Examples

Изменение размера изображения - с соотношением сторон

```
// resize the image to be contained within a maximum width and height, keeping aspect ratio
public static UIImage MaxResizeImage(this UIImage sourceImage, float maxWidth, float
maxHeight)
{
    var sourceSize = sourceImage.Size;
    var maxResizeFactor = Math.Min(maxWidth / sourceSize.Width, maxHeight /
sourceSize.Height);
    if (maxResizeFactor > 1) return sourceImage;
    var width = maxResizeFactor * sourceSize.Width;
    var height = maxResizeFactor * sourceSize.Height;
    UIGraphics.BeginImageContext(new CGSize(width, height));
    sourceImage.Draw(new CGRect(0, 0, width, height));
    var resultImage = UIGraphics.GetImageFromCurrentImageContext();
    UIGraphics.EndImageContext();
    return resultImage;
}
```

Изменение размера изображения - без соотношения сторон

```
// resize the image (without trying to maintain aspect ratio)
public static UIImage ResizeImage(this UIImage sourceImage, float width, float height)
{
    UIGraphics.BeginImageContext(new SizeF(width, height));
    sourceImage.Draw(new RectangleF(0, 0, width, height));
    var resultImage = UIGraphics.GetImageFromCurrentImageContext();
    UIGraphics.EndImageContext();
    return resultImage;
}
```

Обрезать изображение без изменения размера

```
// crop the image, without resizing
public static UIImage CropImage(this UIImage sourceImage, int crop_x, int crop_y, int width,
int height)
{
    var imgSize = sourceImage.Size;
    UIGraphics.BeginImageContext(new SizeF(width, height));
    var context = UIGraphics.GetCurrentContext();
    var clippedRect = new RectangleF(0, 0, width, height);
    context.ClipToRect(clippedRect);
    var drawRect = new CGRect(-crop_x, -crop_y, imgSize.Width, imgSize.Height);
    sourceImage.Draw(drawRect);
    var modifiedImage = UIGraphics.GetImageFromCurrentImageContext();
    UIGraphics.EndImageContext();
```

```
    return modifiedImage;
}
```

Прочитайте Методы изменения размера для UIImage онлайн:
https://riptutorial.com/ru/xamarin-ios/topic/6542/методы-изменения-размера-для-uiimage

глава 14: Оповещения

Examples

Показать оповещение

Для предупреждений с iOS 8 вы должны использовать `UIAlertController` но для версий раньше вы использовали бы `UIAlertView` , который теперь устарел.

8,0

```
var alert = UIAlertController.Create(title, message, UIAlertControllerStyle.Alert);
alert.AddAction(UIAlertAction.Create(otherTitle, UIAlertActionStyle.Destructive, (action) => {
    // otherTitle();
}));
alert.AddAction(UIAlertAction.Create(cancelTitle, UIAlertActionStyle.Cancel, null));
this.PresentViewController(alert, true, null);
```

8,0

```
var alert = new UIAlertView (title, message, null, cancelTitle, otherTitle);
alert.Clicked += (object sender, UIButtonEventArgs e) => {
    if(e.ButtonIndex == 1)
       // otherTitle();
};
alert.Show ();
```

Отобразить оповещение входа в систему

Следующий код для iOS 8 и ниже для создания оповещения входа.

```
// Create the UIAlertView
var loginAlertView = new UIAlertView(title, message, null, cancelTitle, okTitle);

// Setting the AlertViewStyle to UIAlertViewStyle.LoginAndPasswordInput
loginAlertView.AlertViewStyle = UIAlertViewStyle.LoginAndPasswordInput;

// Getting the fields Username and Password
var usernameTextField = loginAlertView.GetTextField(0);
var passwordTextField = loginAlertView.GetTextField(1);

// Setting a placeholder
usernameTextField.Placeholder = "user@stackoverflow.com";
passwordTextField.Placeholder = "Password";

// Adding the button click handler.
loginAlertView.Clicked += (alertViewSender, buttonArguments) =>
{

    // Check if cancel button is pressed
    if (buttonArguments.ButtonIndex == loginAlertView.CancelButtonIndex)
    {
```

```
        // code
    }

    // In our case loginAlertView.FirstOtherButtonIndex is equal to the OK button
    if (buttonArguments.ButtonIndex == loginAlertView.FirstOtherButtonIndex)
    {
        // code
    }
};

// Show the login alert dialog
loginAlertView.Show();
```

Отображение листа действий

`UIAlertController` доступный с iOS8, позволяет использовать один и тот же объект предупреждения для листов действий или более классических предупреждений. Единственное различие заключается в том, что `UIAlertControllerStyle` передается как параметр при создании.

Эта строка изменяется от AlertView к ActionSheet, по сравнению с некоторыми другими примерами, доступными здесь:

```
var alert = UIAlertController.Create(title, message, UIAlertControllerStyle.ActionSheet);
```

Способ добавления действий к контроллеру все тот же:

```
alert.AddAction(UIAlertAction.Create(otherTitle, UIAlertActionStyle.Destructive, (action) => {
    // ExecuteSomeAction();
}));
alert.AddAction(UIAlertAction.Create(cancelTitle, UIAlertActionStyle.Cancel, null));

//Add additional actions if necessary
```

Обратите внимание, что если у вас есть безпараметрический метод void, вы можете использовать его как последний параметр `.AddAction()` .

Например, предположим, что я хочу, чтобы код `private void DoStuff(){...}` выполнялся, когда я `private void DoStuff(){...}` «OK»:

```
UIAlertAction action = UIAlertAction.Create("OK", UIAlertActionStyle.Cancel, DoStuff);
alert.AddAction(action);
```

Заметьте, что я не использую () после DoStuff при создании действия.

То, как вы представляете контроллер, выполняется так же, как и любой другой контроллер:

```
this.PresentViewController(alert, true, null);
```

Диалоговое окно «Модифицировать оповещение»

Общепринятой практикой было использование `NSRunLoop` для отображения модального `UIAlertView` для блокировки выполнения кода до тех пор, пока пользовательский ввод не будет обработан в iOS; пока Apple не выпустила iOS7, она сломала несколько существующих приложений. К счастью, есть лучший способ реализовать его с асинхронным / ожиданием C #.

Вот новый код, использующий шаблон async / await для отображения модального UIAlertView:

```
Task ShowModalAletViewAsync (string title, string message, params string[] buttons)
{
    var alertView = new UIAlertView (title, message, null, null, buttons);
    alertView.Show ();
    var tsc = new TaskCompletionSource ();

    alertView.Clicked += (sender, buttonArgs) => {
        Console.WriteLine ("User clicked on {0}", buttonArgs.ButtonIndex);
        tsc.TrySetResult(buttonArgs.ButtonIndex);
    };
    return tsc.Task;
}

//Usage
async Task PromptUser() {
    var result = await ShowModalAletViewAsync
                ("Alert", "Do you want to continue?", "Yes", "No"); //process the result
}
```

Прочитайте Оповещения онлайн: https://riptutorial.com/ru/xamarin-ios/topic/433/оповещения

глава 15: Параллельное программирование в Xamarin.iOS

Examples

Управление пользовательским интерфейсом из фоновых потоков

Фоновые потоки не могут изменять пользовательский интерфейс; почти все методы UIKit должны быть вызваны в основной поток.

Из подкласса `NSObject` (включая любой `UIViewController` или `UIView`):

```
InvokeOnMainThread(() =>
{
    // Call UI methods here
});
```

Из стандартного класса C #:

```
UIApplication.SharedApplication.InvokeOnMainThread(() =>
{
    // Call UI methods here
});
```

`InvokeOnMainThread` ждет, пока ваш код будет запущен в основном потоке для выполнения перед продолжением. Если вам не нужно ждать, используйте `BeginInvokeOnMainThread` .

Использование Async и ожидание

Вы можете использовать методы async для обработки асинхронных исполнений. Например, запросы POST и GET. Скажем ниже, это ваш метод получения данных.

```
Task<List> GetDataFromServer(int type);
```

Вы можете вызвать этот метод, как показано ниже

```
var result = await GetDataFromServer(1);
```

Однако в реальной практике этот метод будет находиться в интерфейсе уровня сервиса. Там лучший способ сделать это - создать отдельный метод для вызова этого и обновить пользовательский интерфейс, показанный ниже.

```
//Calling from viewDidLoad
void async ViewDidLoad()
```

```
{
    await GetDataListFromServer(1);
    //Do Something else
}

//New method call to handle the async task
private async Task GetArchivedListFromServer(int type)
{
    var result = await GetDataFromServer(type);
    DataList.AddRange(result.toList());
    tableView.ReloadData();
}
```

В приведенном выше фрагменте кода будет вызван метод GetDataListFromServer и он отправит веб-запрос. Тем не менее, он не будет блокировать поток пользовательского интерфейса, пока он не получит ответ от сервера. Он будет двигаться вниз по линии после `await GetDataListFromServer(1)` . Однако внутри метода `private async Task GetArchivedListFromServer(int type)` он будет ждать, пока он получит ответ от сервера, чтобы выполнить строки после того, как `var result = await GetDataFromServer(type);` ,

Прочитайте Параллельное программирование в Xamarin.iOS онлайн:
https://riptutorial.com/ru/xamarin-ios/topic/1364/параллельное-программирование-в-xamarin-ios

глава 16: Подключение к Microsoft Cognitive Services

замечания

В этом примере мы использовали пакет Microsoft.ProjectOxford.Vision NuGet: https://www.nuget.org/packages/Microsoft.ProjectOxford.Vision/

Чтобы узнать больше о Microsoft Cognitive Services, обратитесь к официальной документации: https://www.microsoft.com/cognitive-services/en-us/computer-vision-api

Пожалуйста, найдите загруженный образец на моем GitHub: https://github.com/Daniel-Krzyczkowski/XamarinIOS/tree/master/XamariniOS_CognitiveServices

Я также прикрепляю ссылку на свой блог, где я представил, как использовать когнитивные услуги с приложением Xamarin Forms: http://mobileprogrammer.pl

Examples

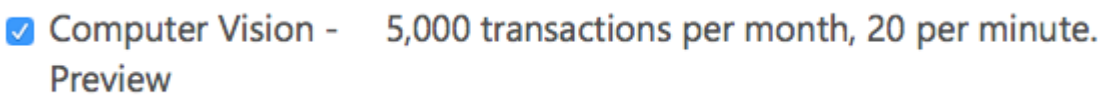
Подключение к Microsoft Cognitive Services

В этом примере вы узнаете, как использовать Microsoft Cognitive Services с мобильным приложением Xamarin iOS. Мы будем использовать API компьютерного зрения, чтобы определить, что находится на картинке.

После создания проекта Xamarin.iOS добавьте ниже пакет NuGet к проекту:

https://www.nuget.org/packages/Microsoft.ProjectOxford.Vision/

С помощью этой библиотеки мы сможем использовать Cognitive Services в нашем приложении iOS. Я предполагаю, что у вас уже зарегистрирована учетная запись Microsoft, и вы включили Computer Vision Api, как на экране ниже:



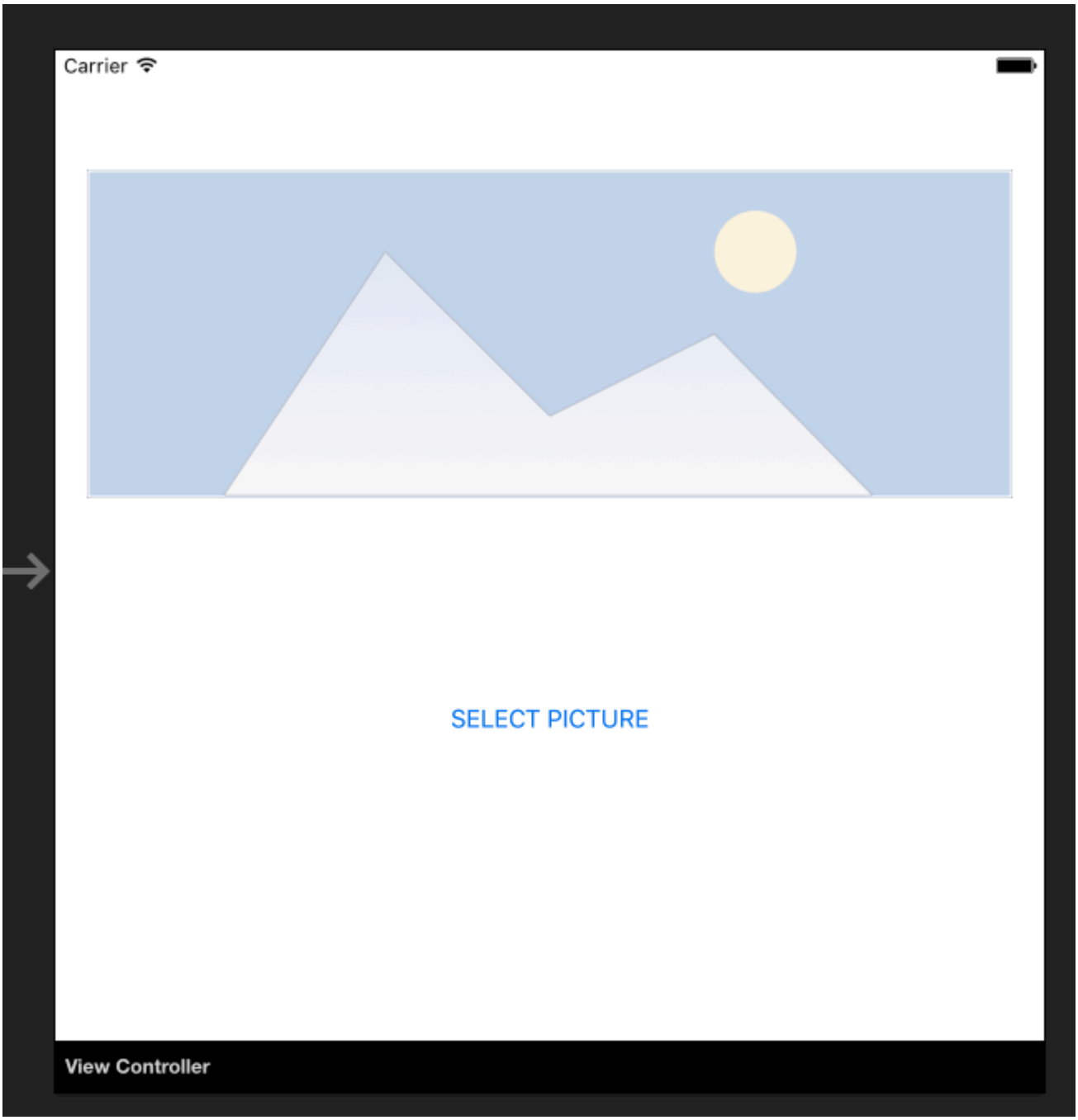
Как только вы нажмете «Подписаться» внизу, появится Api Key:

Computer Vision - Preview	5,000 transactions per month, 20 per minute.	Key 1: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate | Show | Copy Key 2: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate | Show | Copy

Теперь мы можем начать настраивать доступ к Cognitive Services из приложения iOS. Во-первых, нам нужно получить некоторую картину для анализа. Для этого мы можем использовать компонент Xamarin Media ниже:
https://components.xamarin.com/view/mediaplugin

Как только он будет успешно установлен, создадим простой пользовательский интерфейс с изображением и кнопкой, чтобы выбрать изображение из галереи. Размер элементов управления зависит от вас.

Откройте Main.storyboard и добавьте элементы управления UIImageView и UIButton по умолчанию ViewController. Добавьте их имена: «SelectedPictureImageView» и «SelectButton»:



Теперь мы должны добавить обработчик события «Touch Up Inside» для обработки выбора изображения:

```
partial void SelectButtonClick(UIButton sender)
{
    selectImage();
}

async void selectImage()
{
    var selectedImage = await CrossMedia.Current.PickPhotoAsync();
    SelectedPictureImageView.Image =  new
UIImage(NSData.FromStream(selectedImage.GetStream()));
}
```

Теперь мы хотели бы отобразить аналитическую информацию, как только Cognitive

Services вернет информацию. Добавьте метку под кнопкой «АнализLabel»:

SELECT PICTURE

Analysis result....

Пришло время подключить Computer Vision API!

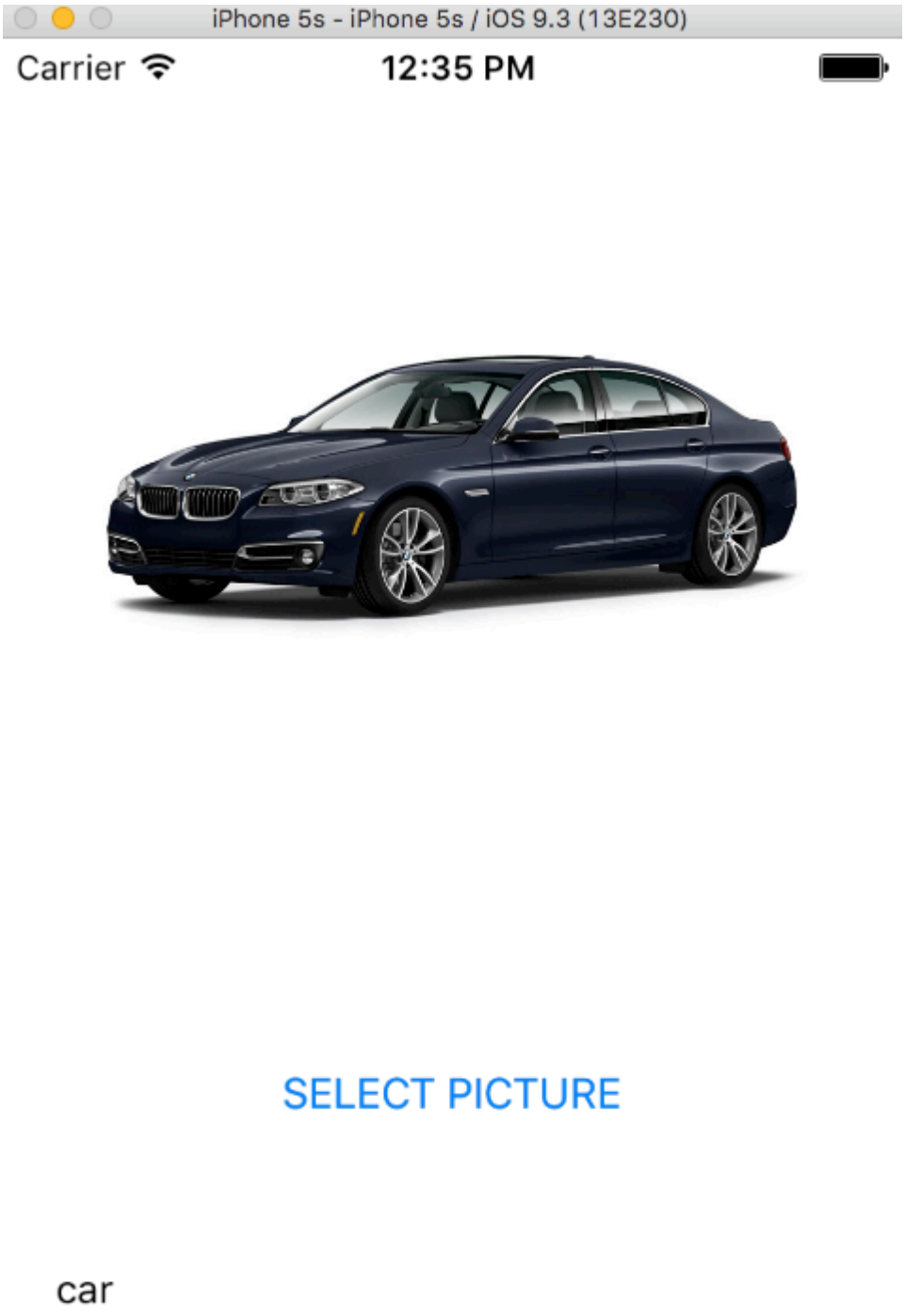
Чтобы получить информацию о выбранном изображении, добавьте ниже метод. Не забудьте вставить ключ API!

```
async Task analyseImage(Stream imageStream)
{
    try
    {
        VisionServiceClient visionClient = new VisionServiceClient("<<YOUR API KEY HERE>>");
        VisualFeature[] features = { VisualFeature.Tags, VisualFeature.Categories,
VisualFeature.Description };
        var analysisResult = await visionClient.AnalyzeImageAsync(imageStream,
features.ToList(), null);
        AnalysisLabel.Text = string.Empty;
        analysisResult.Description.Tags.ToList().ForEach(tag => AnalysisLabel.Text =
AnalysisLabel.Text + tag + "\n");
    }
    catch (Microsoft.ProjectOxford.Vision.ClientException ex)
    {
        AnalysisLabel.Text = ex.Error.Message;
    }
}
```

Теперь вы можете вызвать его в методе selectImage:

```
async void selectImage()
{
    var selectedImage = await CrossMedia.Current.PickPhotoAsync();
    SelectedPictureImageView.Image =  new
UIImage(NSData.FromStream(selectedImage.GetStream()));
    await analyseImage(selectedImage.GetStream());
}
```

После выбора изображения Microsoft Cognitive Services проанализирует его и вернет результат:



Помните, что изображение не может быть слишком большим - в этом случае вы получите информацию, как показано ниже:

SELECT PICTURE

Input image is too large.

Есть много других услуг, которые вы можете попробовать использовать. Пожалуйста, обратитесь к официальной документации (ссылка прилагается), чтобы узнать больше.

Прочитайте Подключение к Microsoft Cognitive Services онлайн:
https://riptutorial.com/ru/xamarin-ios/topic/6122/подключение-к-microsoft-cognitive-services

глава 17: Работа с Xib и раскадровки в Xamarin.iOS

Examples

Открытие Xib / Sotryboard в Xcode Interface Builder вместо

Xamarin sturio открывает Xib-файл и раскадровки по умолчанию в Xamarin Designer. Пользователь может щелкнуть правой кнопкой мыши по файлу и `Open With` -> `Xcode Interface Builder '

Прочитайте Работа с Xib и раскадровки в Xamarin.iOS онлайн:
https://riptutorial.com/ru/xamarin-ios/topic/6182/работа-с-xib-и-раскадровки-в-xamarin-ios

глава 18: Рекомендации по переносу из UILocalNotification в систему уведомлений пользователей

Examples

UserNotifications

1. Вам придется импортировать UserNotifications

```
 @import UserNotifications;
```

2. Разрешение на подтверждение для localNotification

```
let center = UNUserNotificationCenter.current()
center.requestAuthorization([.alert, .sound]) { (granted, error) in
    // Enable or disable features based on authorization.
}
```

3. Теперь мы обновим номер значка значка приложения

```
 @IBAction  func triggerNotification(){
 let content = UNMutableNotificationContent()
 content.title = NSString.localizedUserNotificationString(forKey: "Tom said:", arguments:
nil)
 content.body = NSString.localizedUserNotificationString(forKey: "Hello Mike Let's go.",
arguments: nil)
 content.sound = UNNotificationSound.default()
 content.badge = UIApplication.shared().applicationIconBadgeNumber + 1;
 content.categoryIdentifier = "com.mike.localNotification"
 //Deliver the notification in two seconds.
 let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 1.0, repeats: true)
 let request = UNNotificationRequest.init(identifier: "TwoSecond", content: content,
trigger: trigger)

 //Schedule the notification.
 let center = UNUserNotificationCenter.current()
 center.add(request)
 }

 @IBAction func stopNotification(_ sender: AnyObject) {
 let center = UNUserNotificationCenter.current()
 center.removeAllPendingNotificationRequests()
 }
```

Прочитайте Рекомендации по переносу из UILocalNotification в систему уведомлений пользователей онлайн: https://riptutorial.com/ru/xamarin-ios/topic/6382/рекомендации-по-переносу-из-uilocalnotification-в-систему-уведомлений-пользователей

глава 19: Связывание скоростных библиотек

Вступление

Легко следовать руководству, которое приведет вас к процессу связывания файлов Swift .framework для использования в проекте Xamarin.

замечания

1. При создании библиотеки в Xcode у нее есть возможность включить быстрые библиотеки. Не надо! Они будут включены в ваше окончательное приложение как NAME.app/Frameworks/LIBRARY.framework/Frameworks/libswift*.dylib, но они должны быть включены как NAME.app/Frameworks/libswift*.dylib

2. Вы можете найти эту информацию в другом месте, но стоит упомянуть: не включать Bitcode в библиотеку. На данный момент Xamarin не включает Bitcode для iOS, и Apple требует, чтобы все библиотеки поддерживали одни и те же архитектуры.

Examples

Связывание быстрой библиотеки в Xamarin.iOS

Связывание быстрой библиотеки в Xamarin.iOS следует тому же процессу для Objective-C, как показано на https://developer.xamarin.com/guides/ios/advanced_topics/binding_objective-c/ , но с некоторыми оговорками.

1. Быстрый класс должен наследовать от NSObject, который должен быть привязан.
2. Компилятор Swift будет переводить имена классов и протоколов во что-то другое, если вы не используете аннотацию @objc (например, @objc (MyClass)) в своих быстрых классах, чтобы указать явное имя цели c.
3. Во время выполнения ваш APP должен включать некоторые быстрые основные библиотеки вместе с вашей привязанной инфраструктурой в папке под названием Frameworks;
4. Когда приложение загружается в AppStore, оно должно содержать папку SwiftSupport вместе с папкой «Полезная нагрузка». Они находятся внутри файла IPA.

Здесь вы можете найти простой пример привязки:
https://github.com/Flash3001/Xamarin.BindingSwiftLibrarySample

И полный пример привязки: https://github.com/Flash3001/iOSCharts.Xamarin

Ниже приведены шаги:

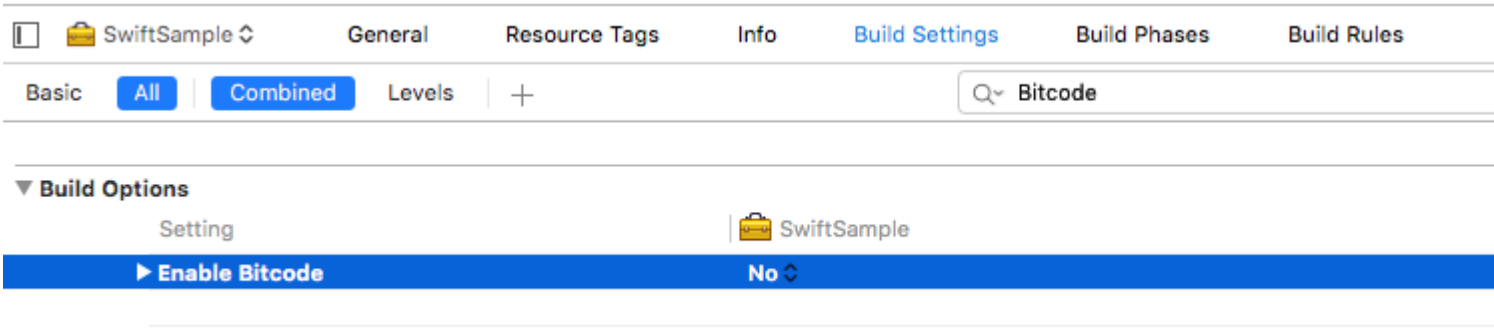
1.1. Подготовьте классы Swift, которые хотите экспортировать.

Для любого класса Swift, который вы хотите использовать, вам нужно либо наследовать из NSObject, либо сделать имя Objective-C явным, используя аннотацию objc. В противном случае компилятор Swift будет генерировать разные имена. Ниже приведен пример кода, как выглядит класс Swift. Обратите внимание, что не имеет значения, какой класс он наследует, если корневой класс наследуется от NSObject.

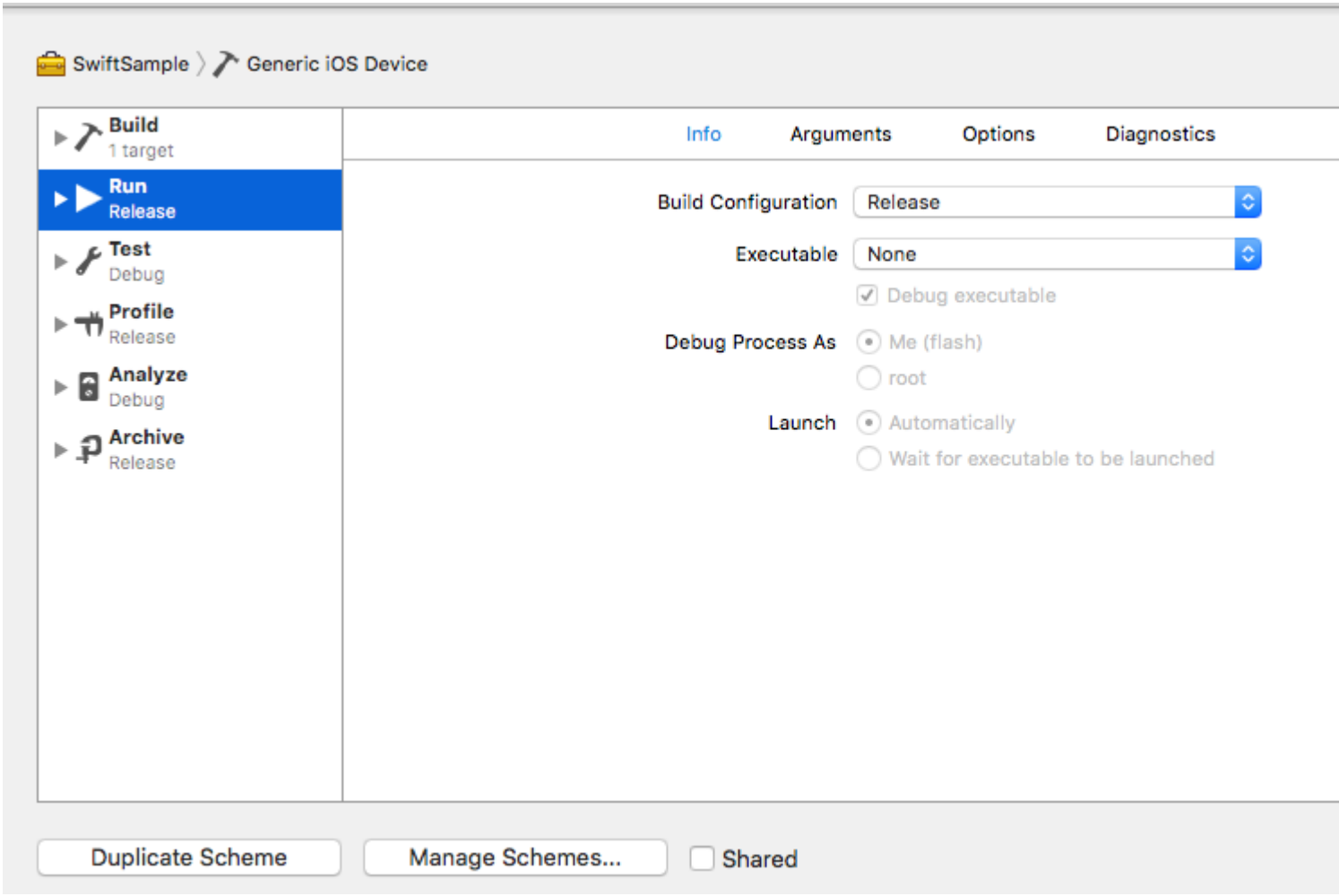
```
//Add this to specify explicit objective c name
@objc(MyClass)
open class MyClass: NSObject {
    open func getValue() -> String
    {
        return "Value came from MyClass.swift!";
    }
}
```

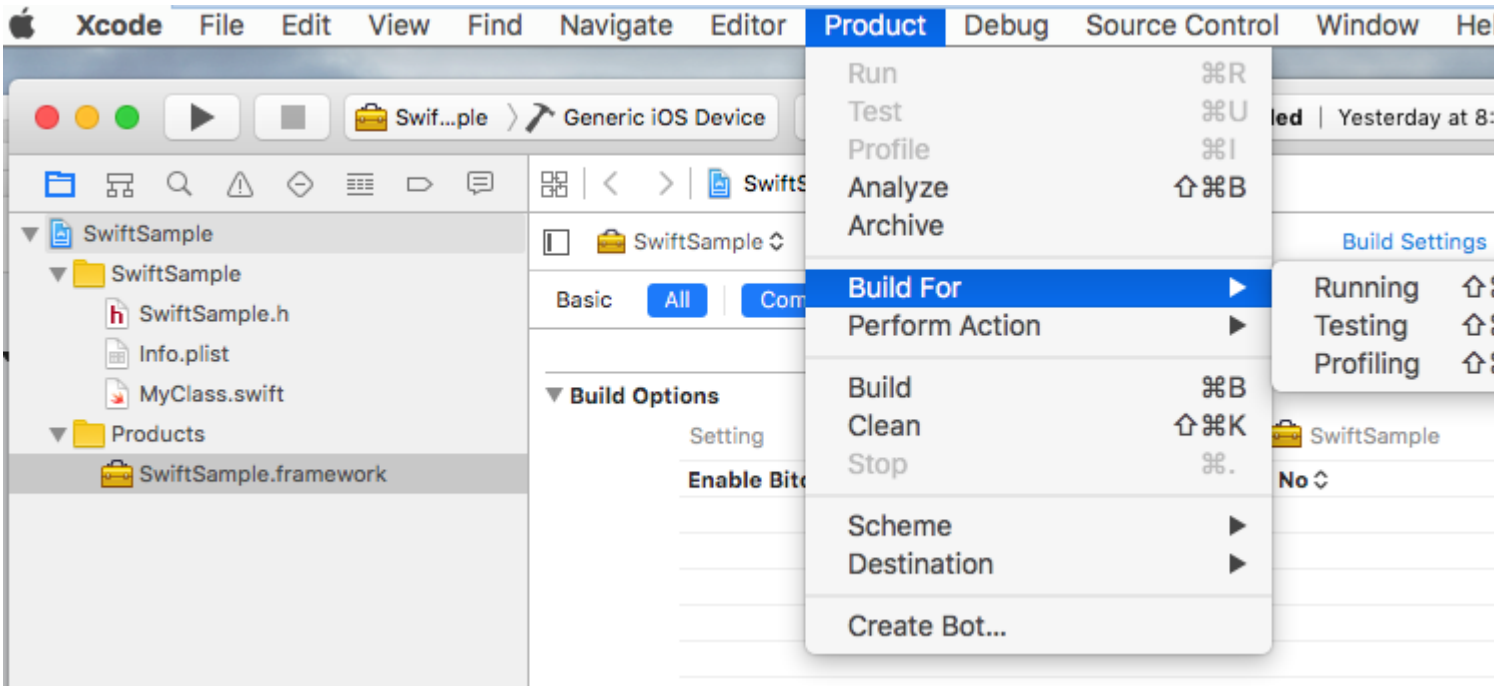
1.2 Построение структуры

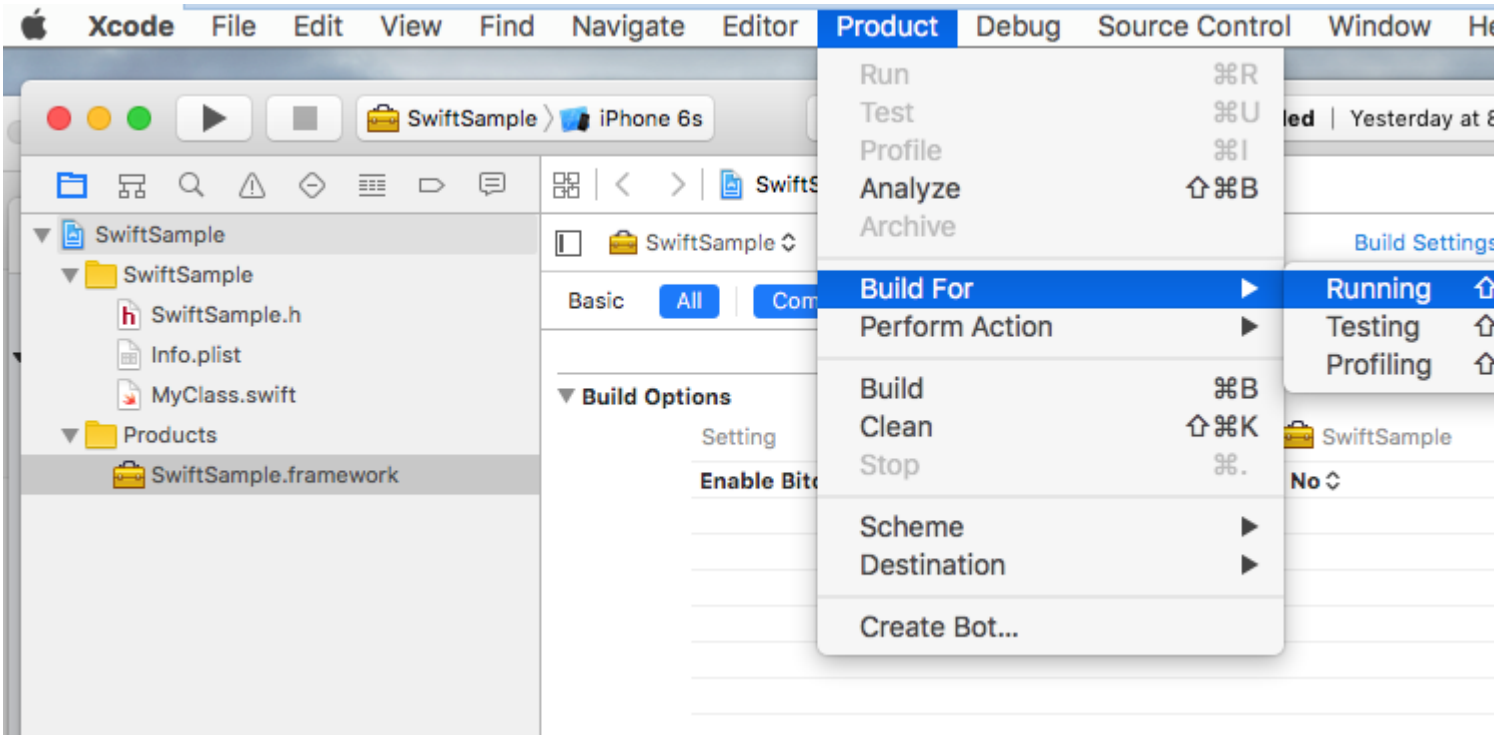
Отключить биткод. *

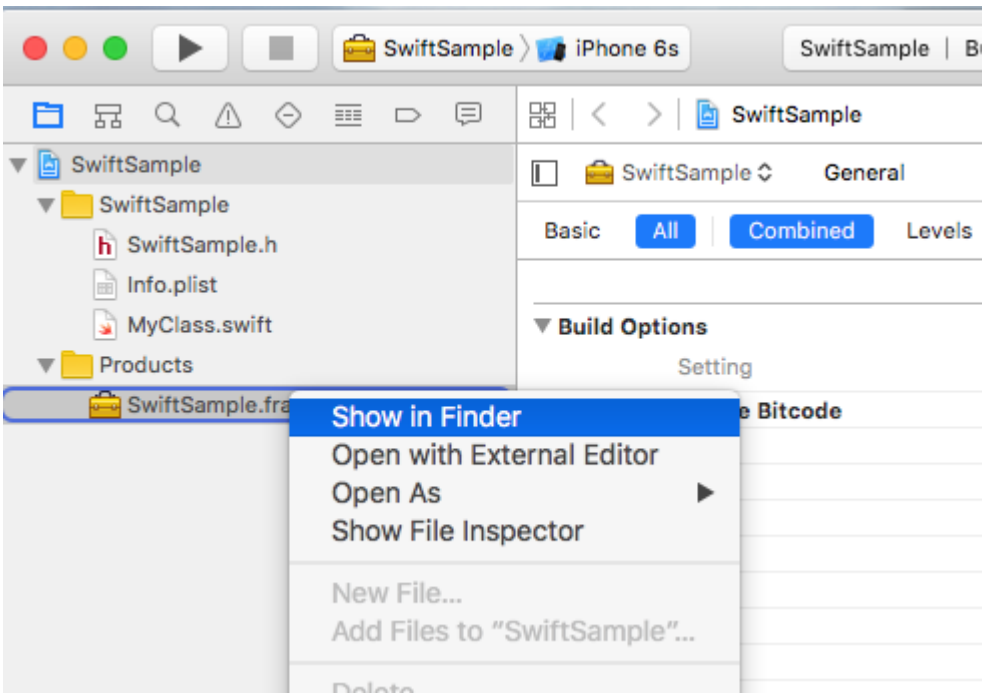


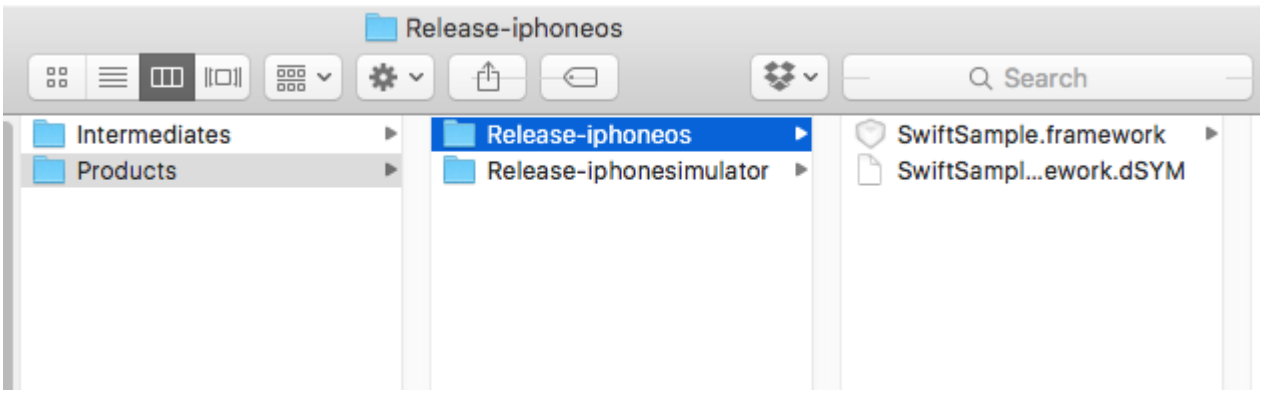
Сборка для выпуска для устройства и симулятора. *


SwiftSample
Generic iOS Device
Build
1 target
Run
Release
Test
Debug
Profile
Release
Analyze
Debug
Archive
Release
Info
Arguments
Options
Diagnostics
Build Configuration
Release
Executable
None
Debug executable
Debug Process As
Me (flash)
root
Launch
Automatically
Wait for executable to be launched
Duplicate Scheme
Manage Schemes...
Shared


Xcode
File
Edit
View
Find
Navigate
Editor
Product
Debug
Source Control
Window
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean ⇧⌘K
Stop ⌘.
Scheme
Destination
Create Bot...
SwiftSample
SwiftSample.h
Info.plist
MyClass.swift
Products
SwiftSample.framework
Build Options
Build Settings
Running
Testing
Profiling





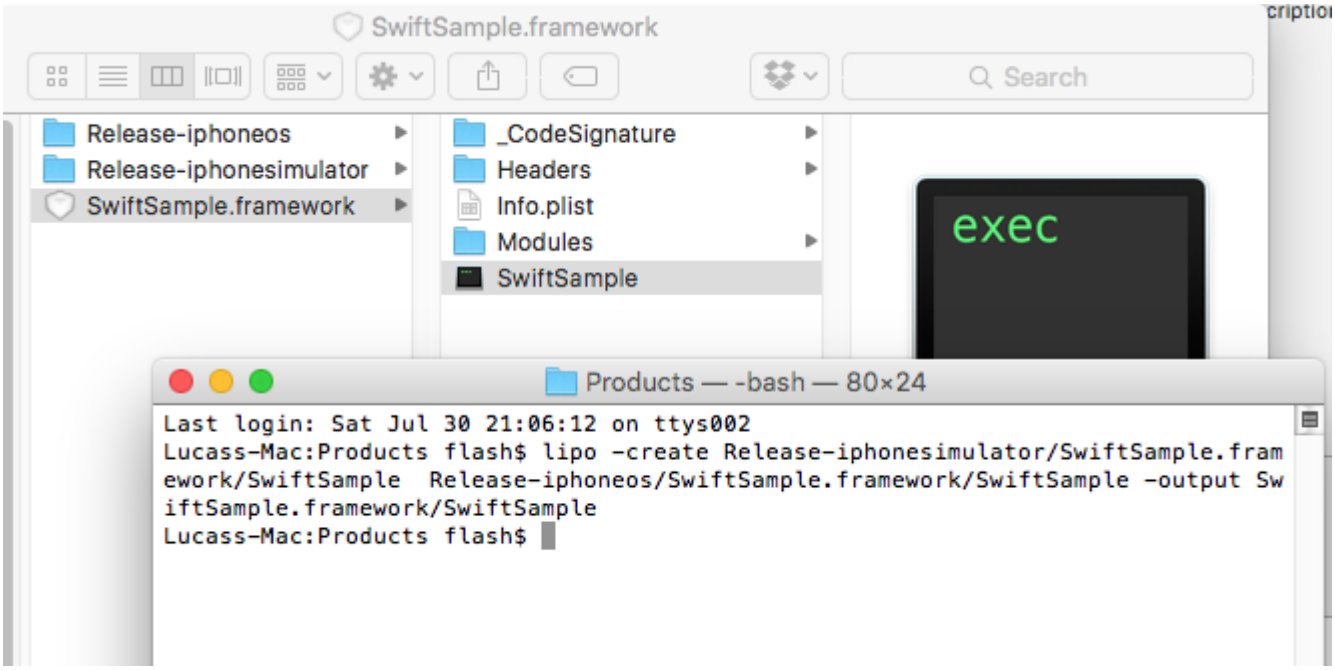


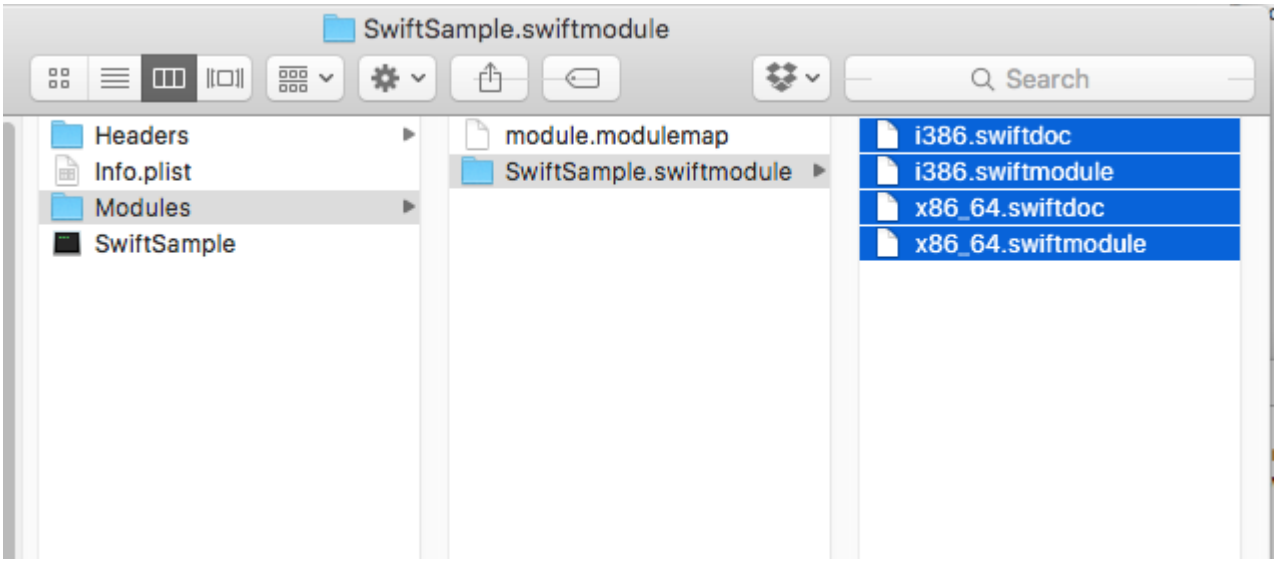
- Не относится только к привязке Swift.

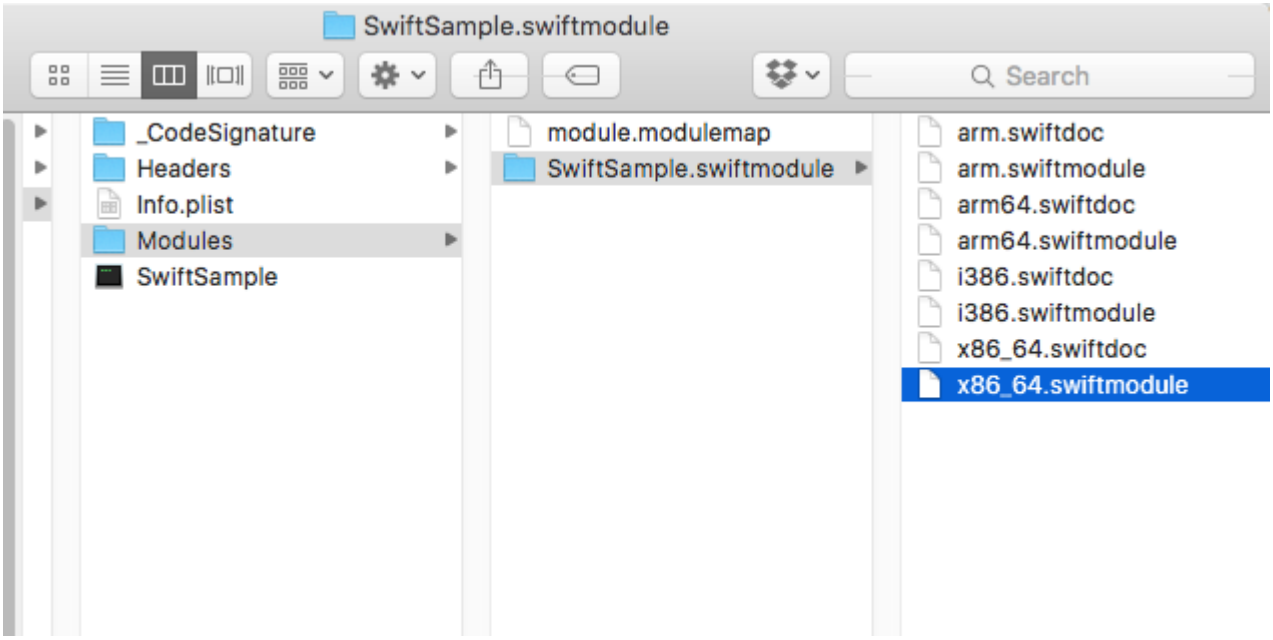
2. Создайте жировую библиотеку

Рамка содержит несколько файлов, один из которых нужно немного поесть - NAME.framework / NAME (без расширения).

- Копировать релиз-iphoneos / NAME.framework в NAME.framework
- Создайте библиотеку FAT, используя:
    - **lipo -create Release-iphonesimulator / NAME.framework / NAME Release-iphoneos / NAME.framework / NAME -output NAME.framework / NAME**
- Скопируйте файлы в Release-iphonesimulator / NAME.framework / Modules / NAME.swiftmodule в NAME.framework / Modules / NAME.swiftmodule (до сих пор он содержал только файлы из iphoneos)



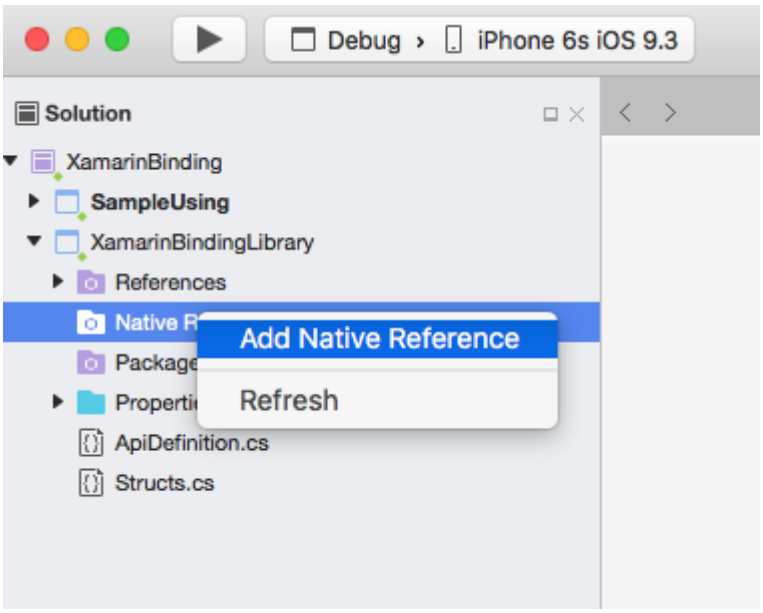


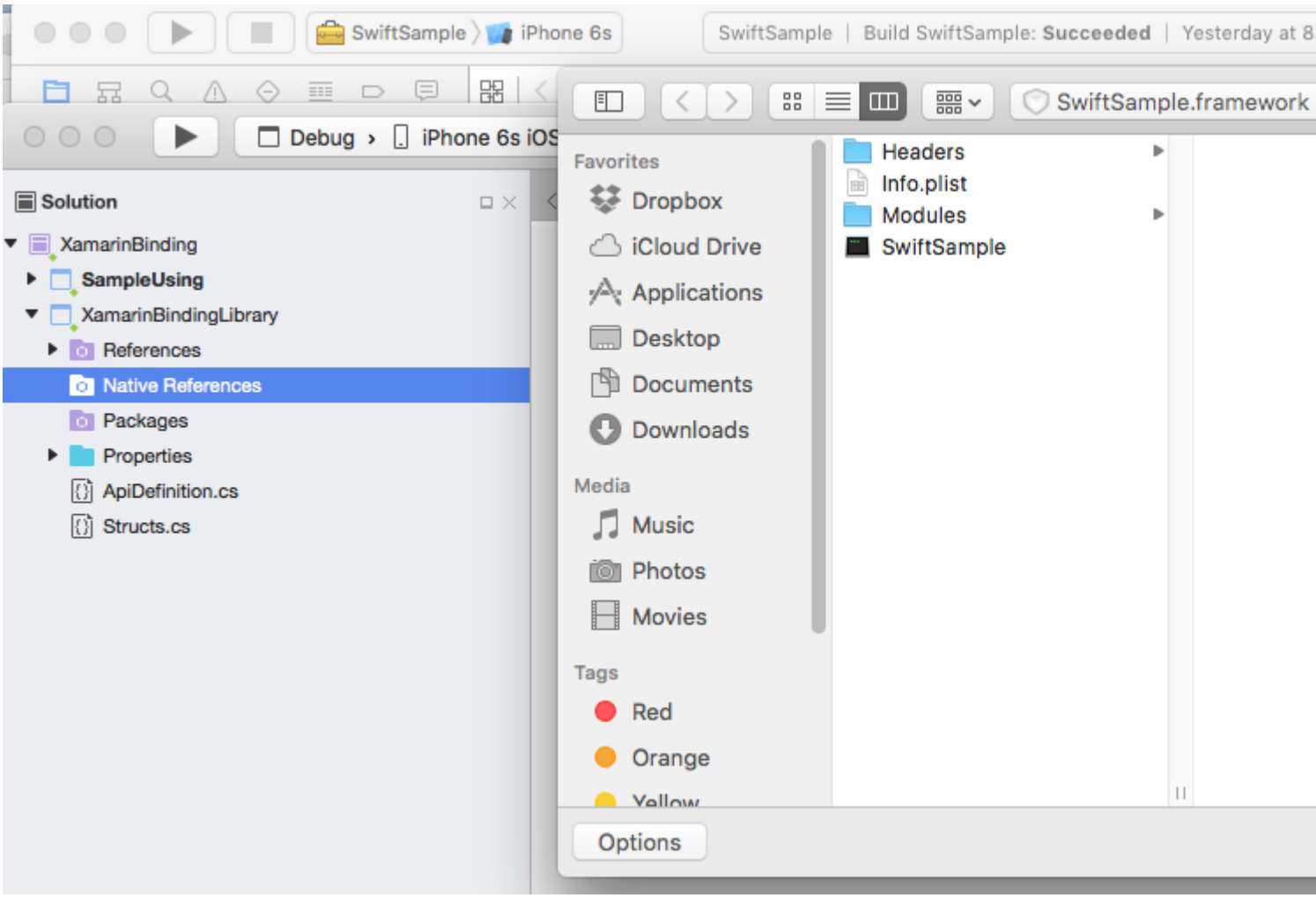


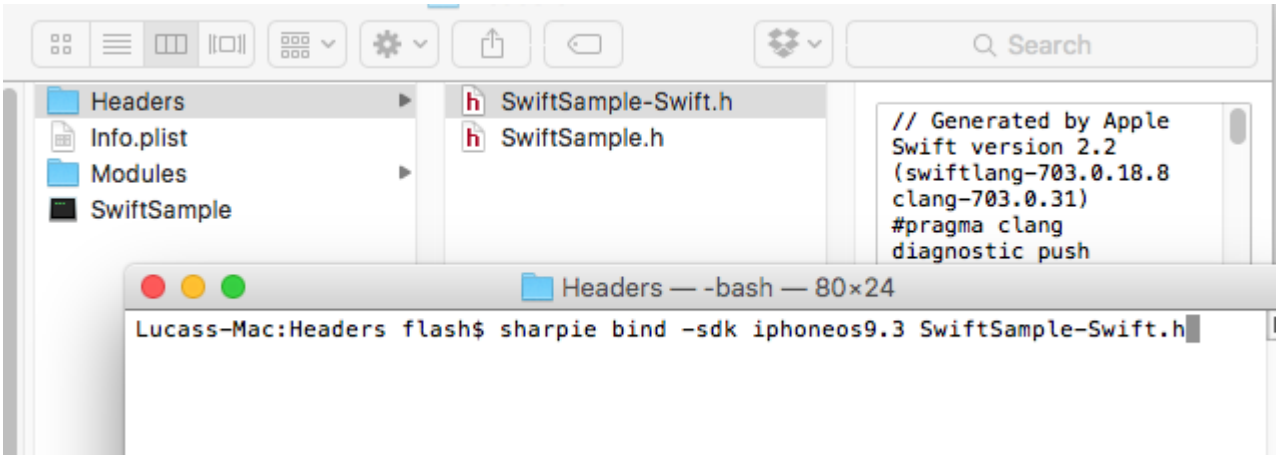
3. Импортируйте библиотеку

Предположим, вы уже создали проект Binding в File -> New -> iOS -> Binding Library.

Поддержка Xamarin по импорту .frameworks. Просто щелкните правой кнопкой мыши «Родные ссылки» и нажмите «Добавить собственную ссылку». Найдите новую структуру жира и добавьте ее.







4. Создайте ApiDefinition на основе файла LIBRARY-Swift.h внутри заголовков.

Вы можете сделать это вручную, но это будет нехорошо. Вы можете использовать Objetive Sharpie. Инструмент Xamarin использует для связывания своих собственных библиотек.

Как использовать его на https://developer.xamarin.com/guides/cross-platform/macios/binding/objective-sharpie/

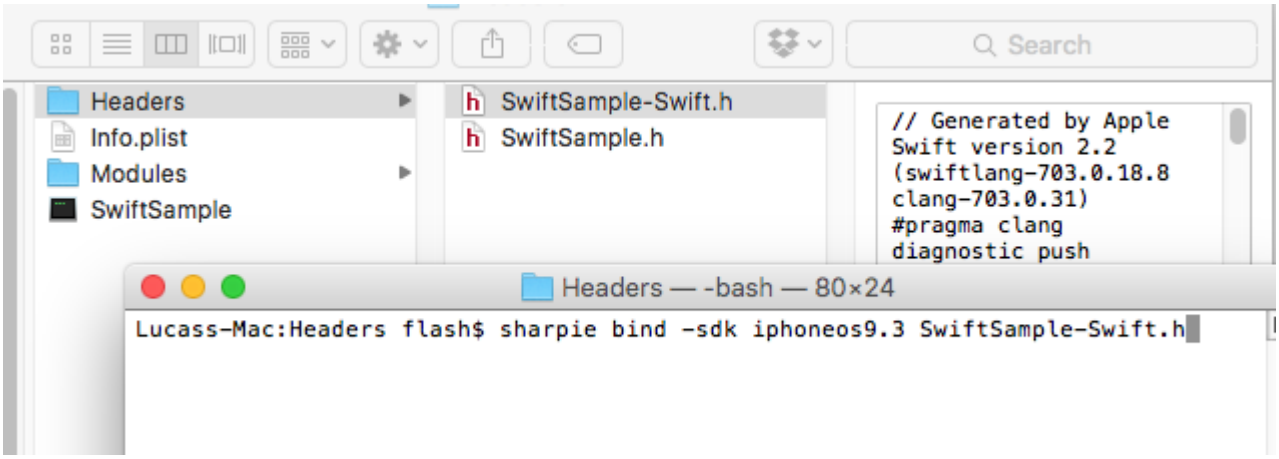
Основная команда будет выглядеть примерно так: ***sharpie bind -sdk iphoneos9.3 NAME-Swift.h***

Если вы получаете `System.Reflection.TargetInvocationException` это, вероятно, потому, что у вас установлена другая версия SDK. Выполните следующую команду для проверки с установленным вами SDK iPhone OS:

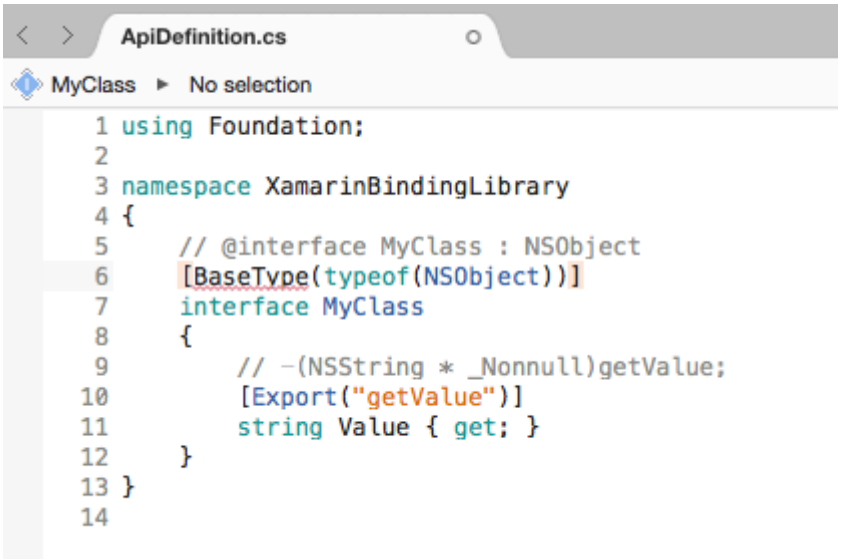
```
sharpie xcode -sdks
```

Файл ***NAME-Swift.h*** находится в ***NAME.framework / Headers / NAME-Swift.h***

Примечание. Быстрые классы должны унаследовать от «NSObject», иначе ***NAME-Swift.h*** не будет импортировать ваши классы, а Objetive Sharpie ничего не преобразит.



Замените содержимое вашего проекта привязки ApiDefinition.cs новым созданным.



5. Измените все [Protocol] и [BaseType], чтобы включить имя класса в среду

выполнения Objective-C.

Если исходный класс или протокол Swift не содержит аннотацию @objc (MyClass), как указано в шаге 1.1, они будут иметь свои внутренние имена Objective-C, поэтому вам нужно будет сопоставить их с правильными.

Все имена доступны в файле NAME-Swift.h в следующем формате:

```
SWIFT_CLASS("_TtC11SwiftSample7MyClass")
@interface MyClass : NSObject
```

А также

```
SWIFT_PROTOCOL("_TtP6Charts17ChartDataProvider_")
@protocol ChartDataProvider
```

Чтобы установить имя, вы используете свойство BaseTypeAttribute.Name https://developer.xamarin.com/guides/cross-platform/macios/binding/binding-types-reference/#BaseType.Name для классов и ProcotolAttribute.Name https: // developer.xamarin.com/api/property/MonoTouch.Foundation.ProtocolAttribute.Name/ для протоколов.

```
[BaseType(typeof(NSObject), Name = "_TtC11SwiftSample7MyClass")]
interface MyClass
```

Делать это вручную не круто. Вы можете использовать этот инструмент https://github.com/Flash3001/SwiftClassify, чтобы вставить все имена. (Это незавершенное производство, но это довольно просто, просто просмотрев код, который вы получите, как он работает).

6.1 Включите все зависимости Swift для запуска.

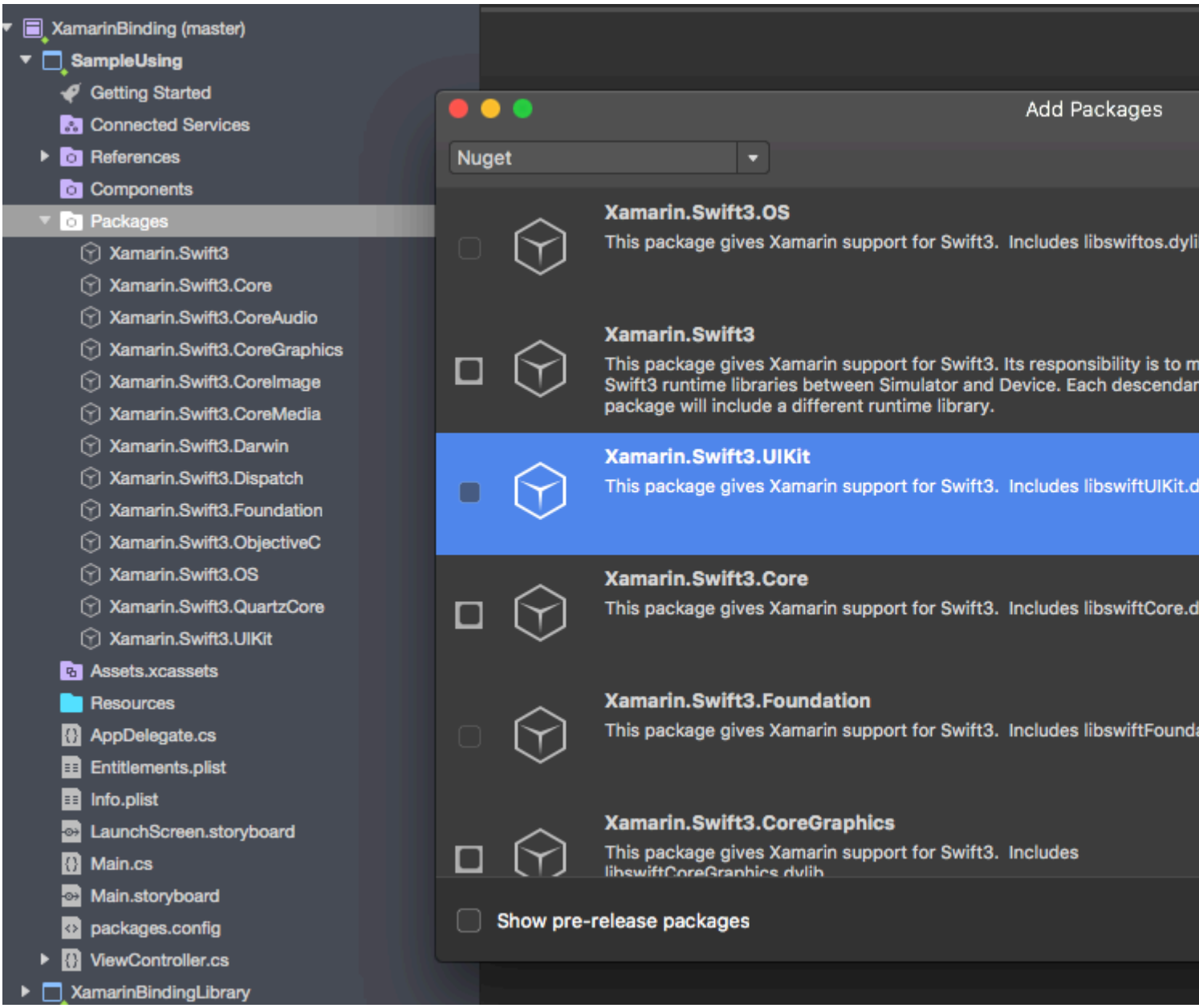
Если вы попытаетесь использовать библиотеку в приложении и попытаетесь запустить ее прямо сейчас, она сработает. Ошибка связана с отсутствием libswiftCore.dylib

Что-то вроде этого:

```
Dyld Error Message:
  Library not loaded: @rpath/libswiftCore.dylib
  Referenced from: /Users/USER/Library/Developer/CoreSimulator/Devices/AC440891-C819-4050-8CAB-CE15AB4B3830/data/Containers/Bundle/Application/27D2EC87-5042-4FA7-9B80-A24A8971FB48/SampleUsing.app/Frameworks/SwiftSample.framework/SwiftSample
  Reason: image not found
```

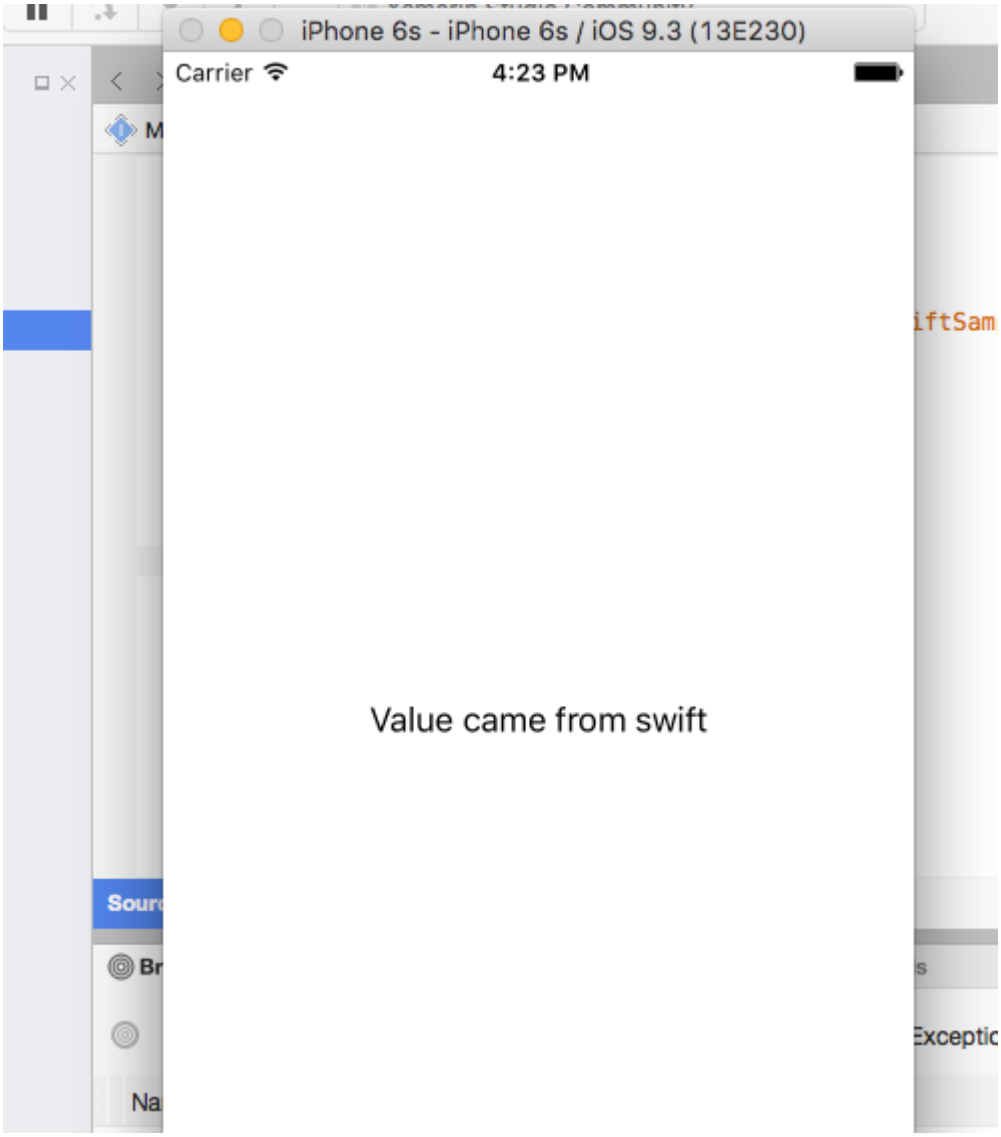
Xamarin.iOS не предоставляет официальную поддержку для привязки библиотеки Swift. Поэтому вы должны вручную включить быстрые основные библиотеки в папки Frameworks и SwiftSupport. Файлы для папки Frameworks отличаются для Simulator и Device. Их можно найти в / Applications/Xcode.app/Contents/Developer//XcodeDefault.xctoolchain/usr/lib/swift.Toolchains

Вместо ручного копирования файлов внутри папки Framework вы можете использовать эту библиотеку https://github.com/Flash3001/Xamarin.Swift3.Support . Он включает в себя каждую отдельную потребность Swift 3.1, каждый из которых находится в одном пакете NuGet.



Как вы можете видеть, пакет Nuget включен в потребительское приложение, а не в самом привязке. Если вы попытаетесь включить его в привязку, вы получите ошибки компиляции.

Если вы создаете пакет Nuget, вы можете указать Nuget включить его в качестве зависимости.



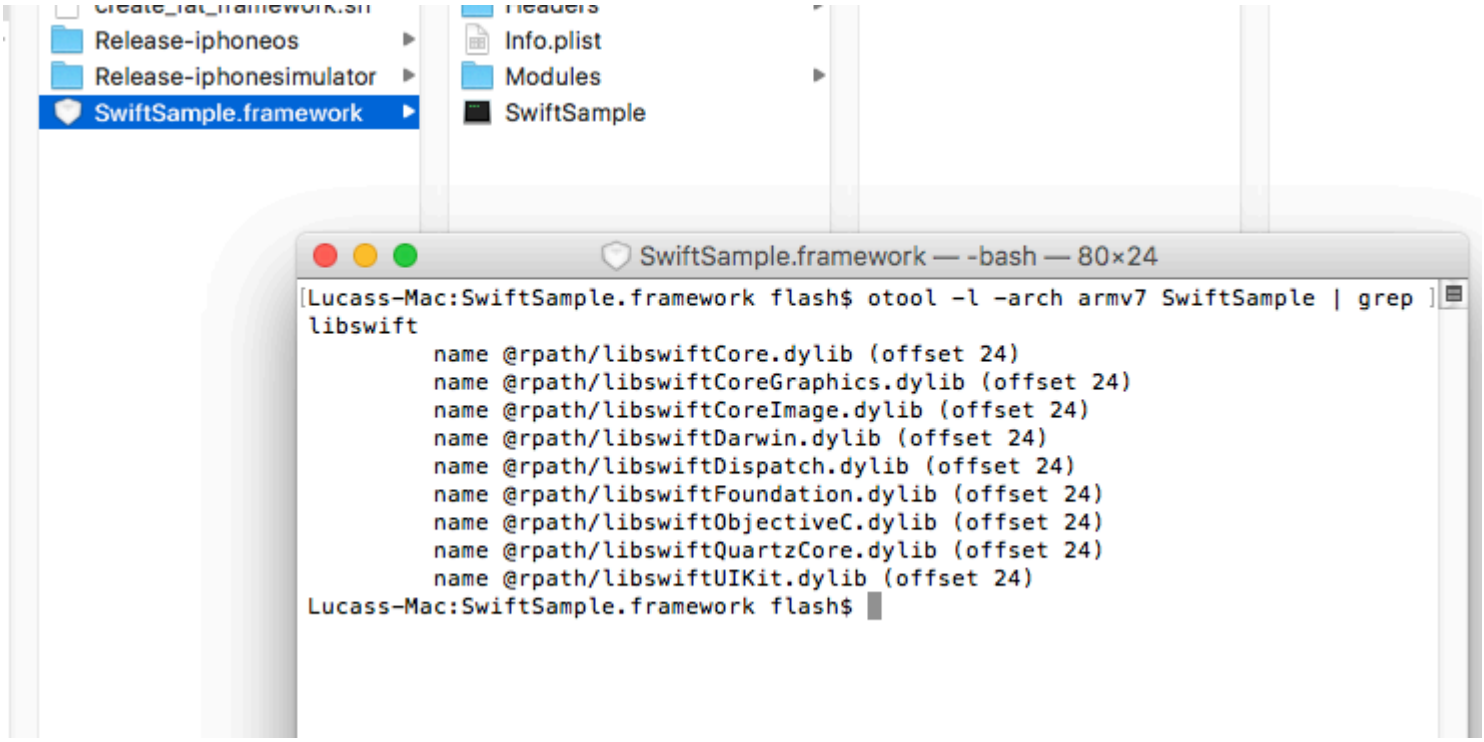
6.2. Определение зависимостей Swift.

Важно, чтобы выяснить, какой пакет необходимо включить в свой проект. Обычно требуется простая привязка:

```
    libswiftCore.dylib
    libswiftCoreGraphics.dylib
    libswiftCoreImage.dylib
    libswiftDarwin.dylib
    libswiftDispatch.dylib
    libswiftFoundation.dylib
    libswiftObjectiveC.dylib
    libswiftQuartzCore.dylib
    libswiftUIKit.dylib
```

Чтобы перечислить каждую зависимость, вы можете запустить следующую команду внутри вашего LibraryName.framework

```
otool -l -arch armv7 LibraryName | grep libswift
```



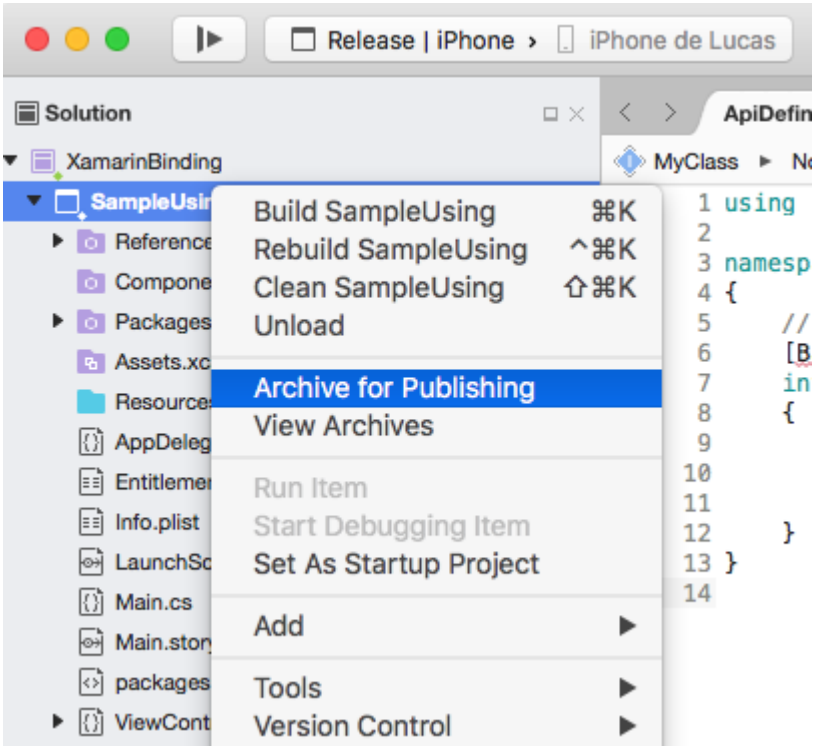
Не включайте все пакеты, доступные в NuGet для Swift3, так как они могут увеличить размер вашего приложения.

7. Включите SwiftSupport, чтобы отправить приложение в AppStore.

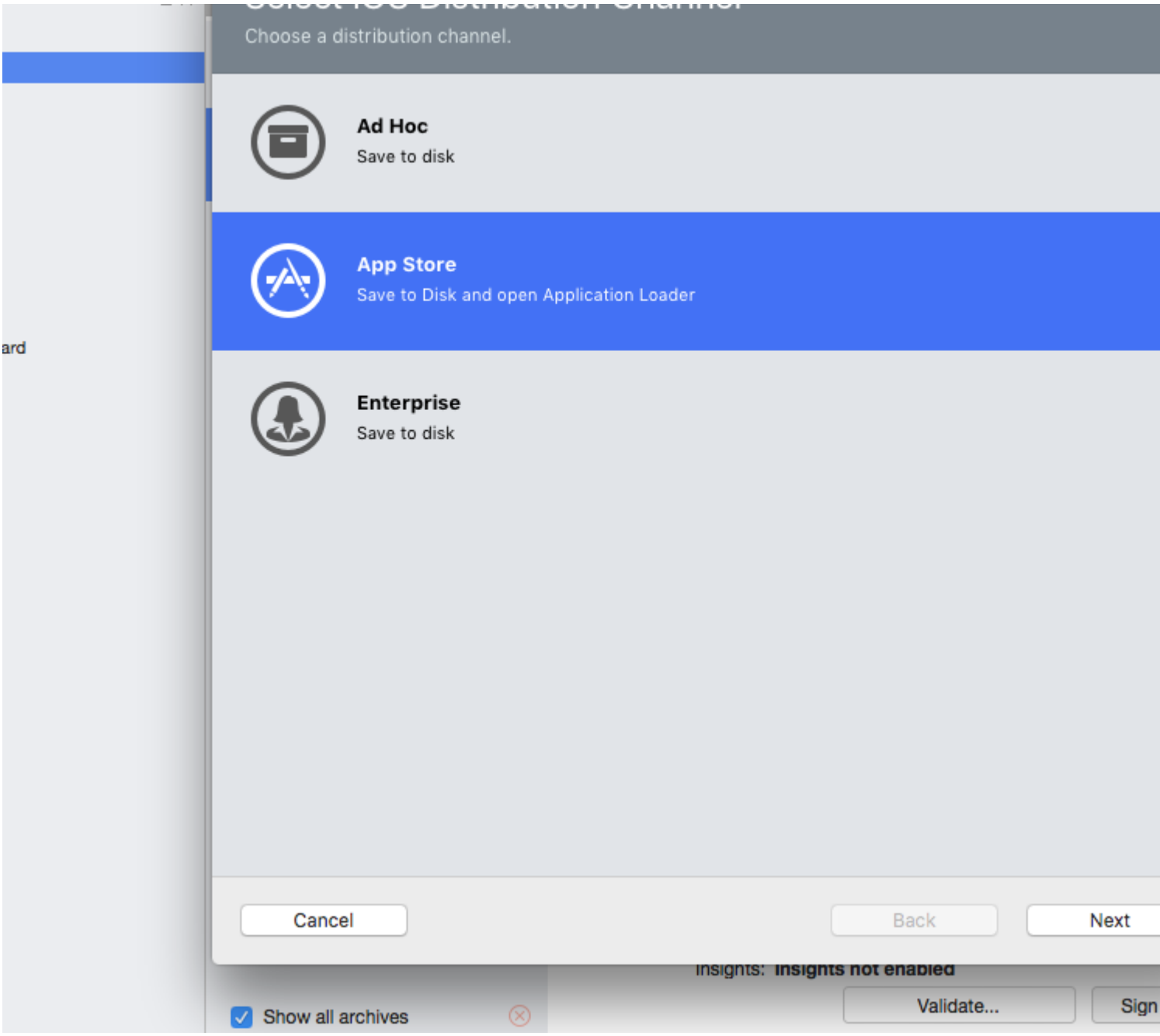
Apple требует, чтобы ваше приложение отправлялось вместе с папкой SwiftSupport вместе с папкой Payload. Оба находятся внутри вашего пакета IPA.

Вы можете использовать этот скрипт https://github.com/bq/ipa-packager, чтобы выполнить эту работу за вас.

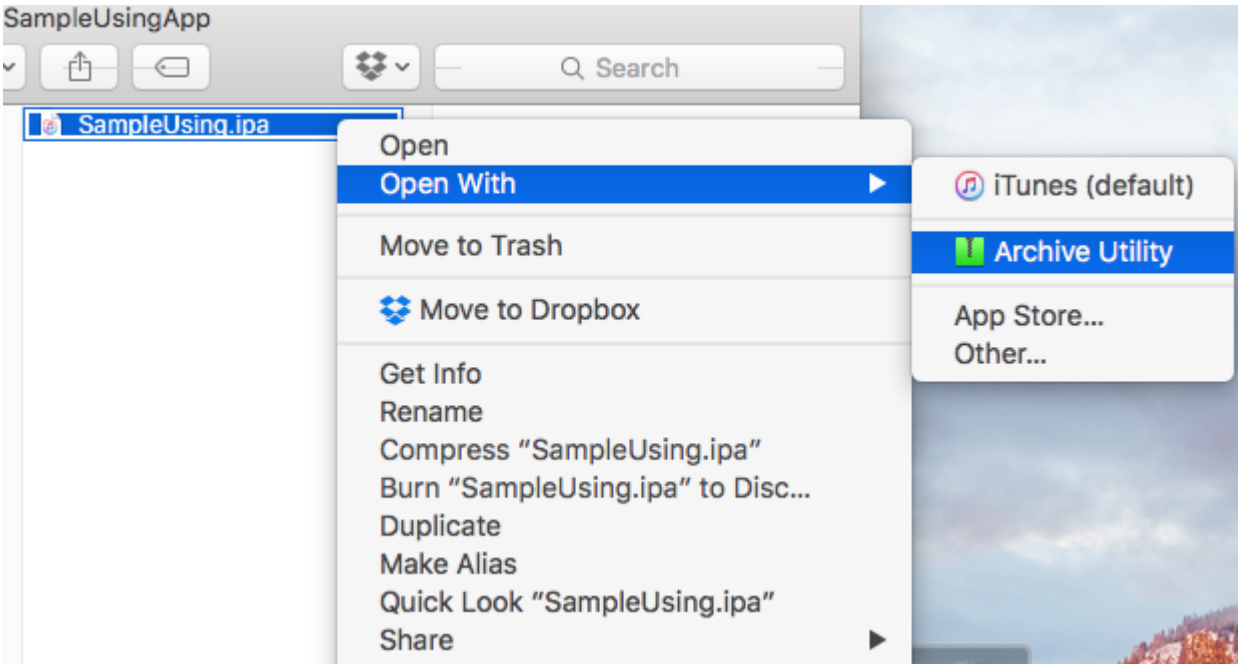
Этот процесс является единственным, который должен выполнять пользователь библиотеки. Каждый раз, когда он пытается нажать приложение в AppStore.



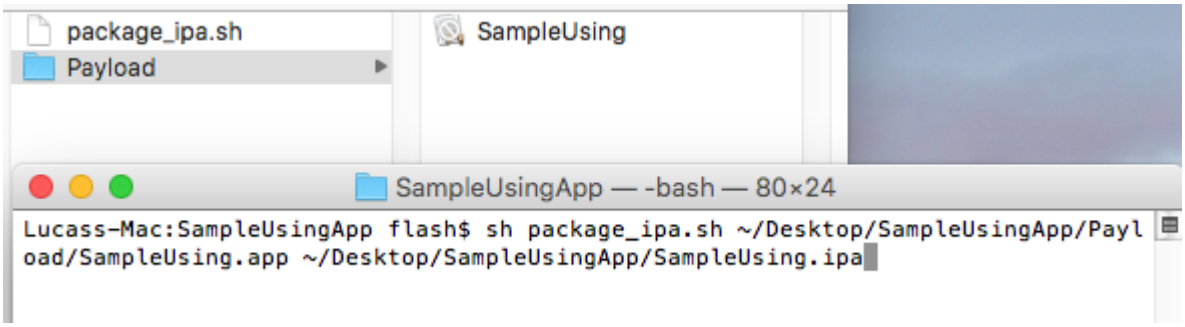
Нажмите «Подписать и распространить» и «Сохранить на диск».



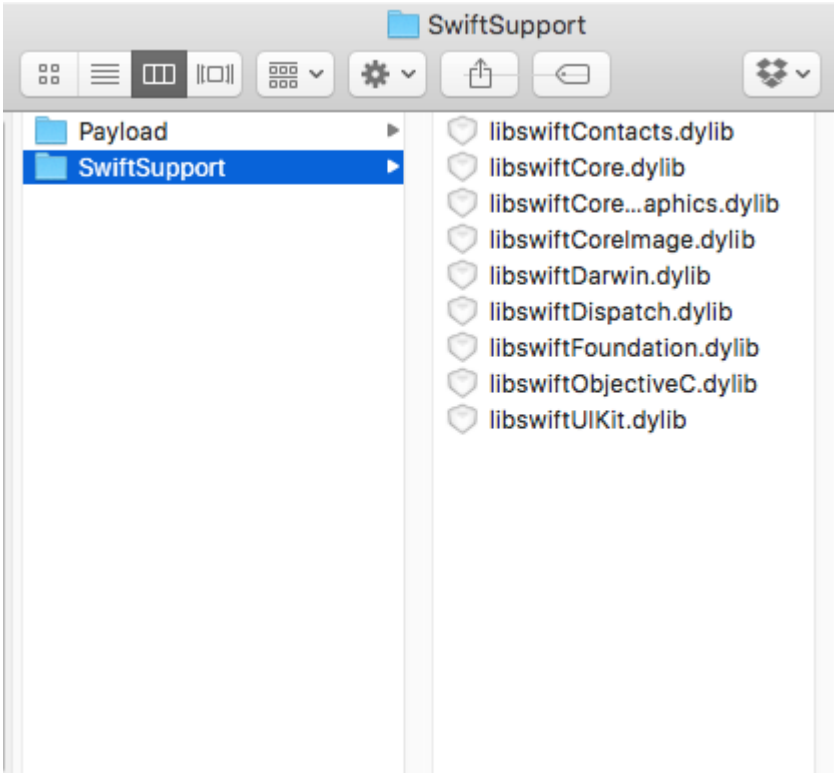
Разархивируйте ваш .IPA



Создайте новый IPA, используя сценарий,



Если вы разблокируете файл, он будет содержать папку SwiftSupport.



замечания

При создании библиотеки в Xcode у нее есть возможность включить быстрые библиотеки. Не надо! Они будут включены в ваше окончательное приложение как NAME.app/Frameworks/LIBRARY.framework/Frameworks/libswift*.dylib, но они должны быть включены как NAME.app/Frameworks/libswift*.dylib

Вы можете найти эту информацию в другом месте, но стоит упомянуть: не включать Bitcode в библиотеку. На данный момент Xamarin не включает Bitcode для iOS, и Apple требует, чтобы все библиотеки поддерживали одни и те же архитектуры.

отказ

Это руководство создано Lucas Teixeira . Все кредиты принадлежат ему. Спасибо, Лукас.

Прочитайте Связывание скоростных библиотек онлайн: https://riptutorial.com/ru/xamarin-ios/topic/6091/связывание-скоростных-библиотек

глава 20: Сенсорный ID

параметры

колонка	колонка
клетка	клетка

замечания

Во-первых, установите, может ли устройство принимать вход Touch ID.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
```

Если это так, мы можем отобразить интерфейс Touch ID с помощью:

```
context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
```

Есть три части информации, которую мы должны передать в `EvaluatePolicy` - сама политика, строка, объясняющая, зачем нужна аутентификация, и обработчик ответа. Обработчик ответа сообщает приложению, что он должен делать в случае успешной или неудачной аутентификации.

Одна из предостережений Local Authentication заключается в том, что он должен запускаться на переднем плане, поэтому обязательно используйте `InvokeOnMainThread` для обработчика ответа:

```
var replyHandler = new LAContextReplyHandler((success, error) =>
    {

        this.InvokeOnMainThread(() =>
        {
            if (success)
            {
                Console.WriteLine("You logged in!");
                PerformSegue("AuthenticationSegue", this);
            }
            else {
                //Show fallback mechanism here
            }
        });

    });
```

Чтобы определить, была ли изменена база данных авторизованных отпечатков пальцев, вы

можете проверить непрозрачную структуру (NSData), возвращенную `context.EvaluatedPolicyDomainState` . Вашему приложению необходимо будет сохранить и сравнить состояние политики для обнаружения изменений. Одно дело отметить, что Apple заявляет:

> Однако характер этих изменений не может быть определен по этим данным.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
{
    var policyState = context.EvaluatedPolicyDomainState;

    var replyHandler = new LAContextReplyHandler((success, error) =>
    {

        this.InvokeOnMainThread(() =>
        {
            if (success)
            {
                Console.WriteLine("You logged in!");
                PerformSegue("AuthenticationSegue", this);
            }
            else {
                //Show fallback mechanism here
            }
        });

    });
    context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
};
```

Examples

Добавить Touch ID в ваше приложение

Во-первых, установите, может ли устройство принимать вход Touch ID.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
```

Если это так, мы можем отобразить интерфейс Touch ID с помощью:

```
context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
```

Есть три части информации, которую мы должны передать в `EvaluatePolicy` - сама политика, строка, объясняющая, зачем нужна аутентификация, и обработчик ответа. Обработчик ответа сообщает приложению, что он должен делать в случае успешной или неудачной аутентификации.

Одна из предостережений Local Authentication заключается в том, что он должен запускаться на переднем плане, поэтому обязательно используйте `InvokeOnMainThread` для обработчика ответа:

```
var replyHandler = new LAContextReplyHandler((success, error) =>
{
    this.InvokeOnMainThread(() =>
    {
        if (success)
        {
            Console.WriteLine("You logged in!");
            PerformSegue("AuthenticationSegue", this);
        }
        else {
            //Show fallback mechanism here
        }
    });
});
```

Чтобы определить, была ли изменена база данных авторизованных отпечатков пальцев, вы можете проверить непрозрачную структуру (NSData), возвращенную `context.EvaluatedPolicyDomainState` . Вашему приложению необходимо будет сохранить и сравнить состояние политики для обнаружения изменений. Одно дело отметить, что Apple заявляет:

> Однако характер этих изменений не может быть определен по этим данным.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
{
    var policyState = context.EvaluatedPolicyDomainState;

    var replyHandler = new LAContextReplyHandler((success, error) =>
    {

        this.InvokeOnMainThread(() =>
        {
            if (success)
            {
                Console.WriteLine("You logged in!");
                PerformSegue("AuthenticationSegue", this);
            }
            else {
                //Show fallback mechanism here
            }
        });

    });
    context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
};
```

Пример кнопки

```
partial void AuthenticateMe(UIButton sender)
```

```
{
    var context = new LAContext();
    //Describes an authentication context
    //that allows apps to request user authentication using Touch ID.
    NSError AuthError;
    //create the reference for error should it occur during the authentication.
    var myReason = new NSString("To add a new chore");
    //this is the string displayed at the window for touch id

    if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
    // check if the device have touchId capabilities.
    {
        var replyHandler = new LAContextReplyHandler((success, error) =>
        {

            this.InvokeOnMainThread(() =>
            {
                if (success)
                {
                    Console.WriteLine("You logged in!");
                    PerformSegue("AuthenticationSegue", this);
                }
                else {
                    //Show fallback mechanism here
                }
            });

        });
        context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);//send touch id request
    };
}
```

Использование брелка

Рабочий источник - https://github.com/benhysell/V.TouchIdExample

Длинное описание формы - http://benjaminhysell.com/archive/2014/11/authentication-in-xamarin-ios-with-touch-id-or-passcode/

```
//Simple View with a switch to enable / disable Touch ID and
//a button to invoke authentication

/// <summary>
/// Enable/Disable Touch ID
/// </summary>
/// <param name="sender">Sender.</param>
partial void TouchIdEnableDisable(UISwitch sender)
{
    if (sender.On)
    {
        //enable Touch ID
        //set our record
        //note what you fill in here doesn't matter, just needs to be
        //consistent across all uses of the record
        var secRecord = new SecRecord(SecKind.GenericPassword)
        {
```

```
            Label = "Keychain Item",
            Description = "fake item for keychain access",
            Account = "Account",
            Service = "com.yourcompany.touchIdExample",
            Comment = "Your comment here",
            ValueData = NSData.FromString("my-secret-password"),
            Generic = NSData.FromString("foo")
        };

        secRecord.AccessControl = new
SecAccessControl(SecAccessible.WhenPasscodeSetThisDeviceOnly,
SecAccessControlCreateFlags.UserPresence);
        SecKeyChain.Add(secRecord);

        authenticateButton.Enabled = true;
    }
    else
    {
        //disable Touch ID
        var record = new SecRecord(SecKind.GenericPassword)
        {
            Service = "com.yourcompany.touchIdExample",
            UseOperationPrompt = "Authenticate to Remove Touch ID / Passcode from Test App"
        };

        SecStatusCode result;

        //query one last time to ensure they can remove it
        SecKeyChain.QueryAsRecord(record, out result);
        if (SecStatusCode.Success == result || SecStatusCode.ItemNotFound == result)
        {
            //remove the record
            SecKeyChain.Remove(record);
            authenticateButton.Enabled = false;
        }
        else
        {
            //could not authenticate, leave switch on
            sender.On = true;
         }
    }
}

/// <summary>
/// Show Touch ID to user and evaluate authentication
/// </summary>
/// <param name="sender">Sender.</param>
partial void AuthenticateUser(UIButton sender)
{
    var rec = new SecRecord(SecKind.GenericPassword)
    {
        Service = "com.yourcompany.touchIdExample",
        UseOperationPrompt = "Authenticate to access Test App"
    };
    SecStatusCode res;
    SecKeyChain.QueryAsRecord(rec, out res);
    if (SecStatusCode.Success == res || SecStatusCode.ItemNotFound == res)
    {
        //Success!!
        //add your code here to continue into your application
        AuthenticatedLabel.Hidden = false;
```

```
    }
    else
    {
        //Failure
        AuthenticatedLabel.Hidden = true;
    }
}
```

Прочитайте Сенсорный ID онлайн: https://riptutorial.com/ru/xamarin-ios/topic/577/сенсорный-id

глава 21: Создавайте и используйте пользовательские ячейки таблицы прототипов в xamarin.iOS с помощью раскадровки

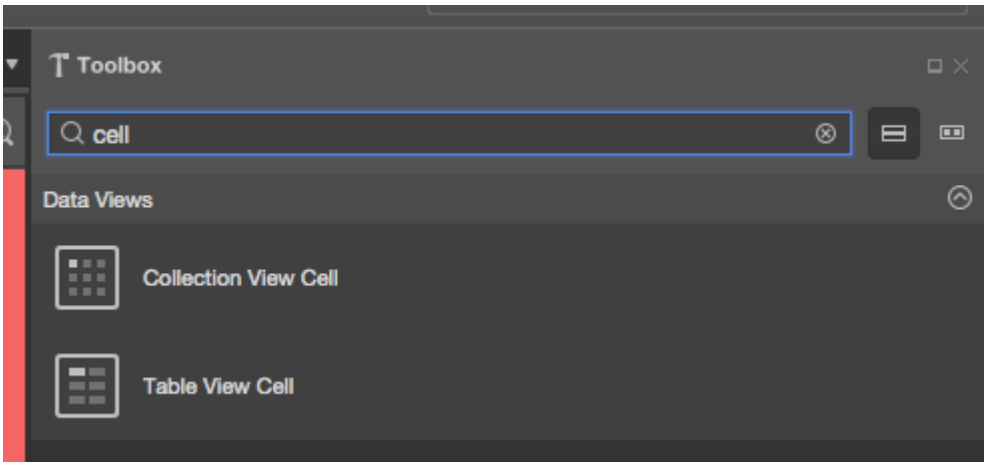
Examples

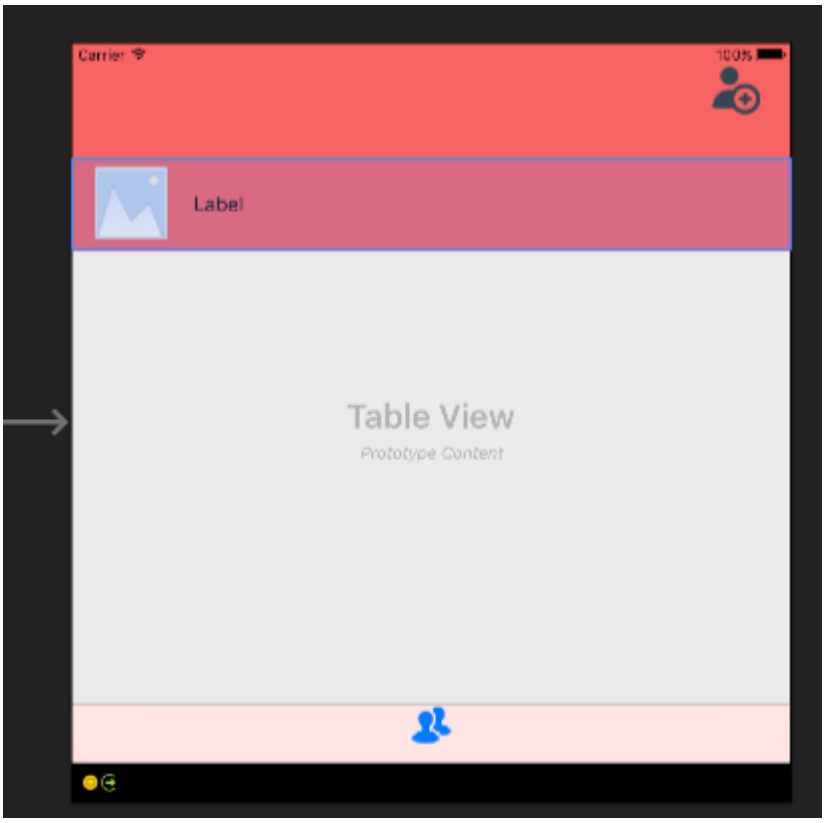
Создание пользовательской ячейки с помощью раскадровки

Open Storyboard, где у вас есть ViewController с TableView:

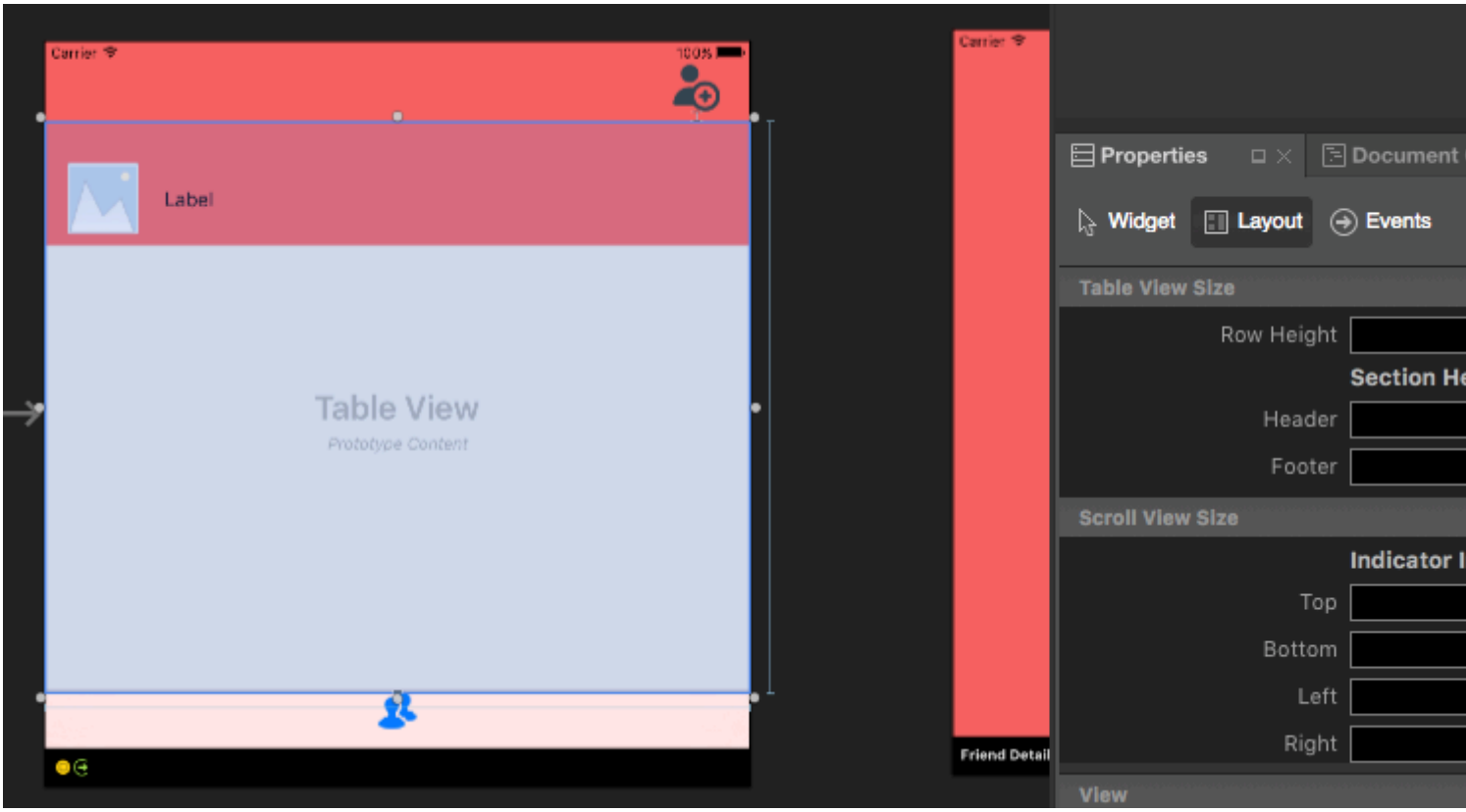
Добавить ячейку прототипа (если ранее не было добавлено ячейки):

Настройте ячейку, как вы хотите (в моем случае есть пользовательский UIImage и Label):



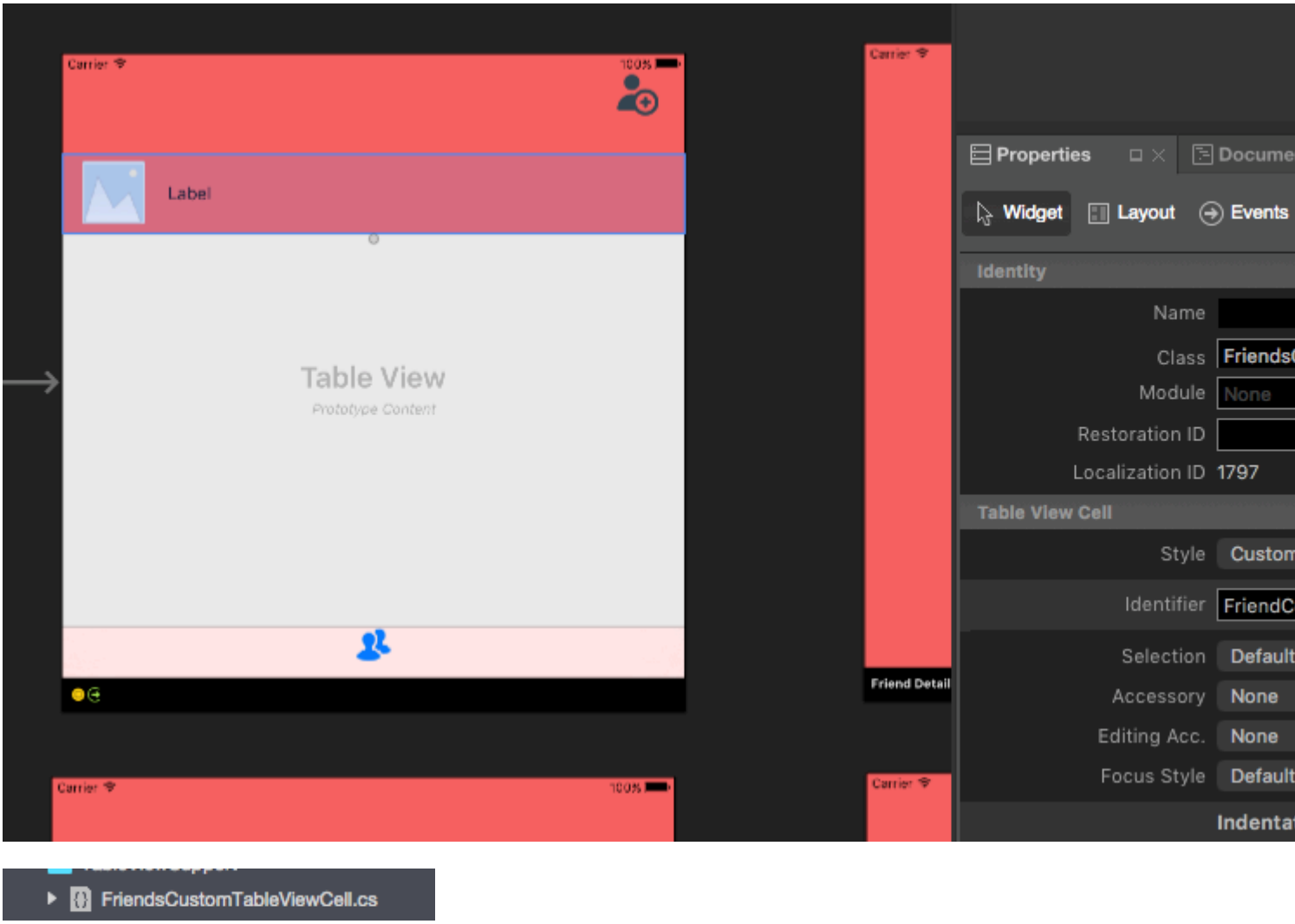


Не забудьте установить высоту ячейки. Для этого выберите весь TableView и в окне «Свойства» выберите вкладку «Макет». В верхней части окна свойств вы увидите «высота строки» - поместите соответствующее значение:



Теперь выберите ячейку прототипа еще раз. В окне «Свойства» введите имя класса (он будет создавать для него класс кода). В моем случае это «FriendsCustomTableViewCell».

После этого укажите «Идентификатор» для вашей ячейки. Как вы видите, мой « FriendCell». Последнее, что нужно установить, - это свойство «Стиль», настроенное на пользовательский. Поле «Имя» должно быть пустым. После того, как вы нажмете «enter» после ввода «Class», файл с кодом будет автоматически создан:



Теперь код для ячейки должен выглядеть следующим образом:

```
public partial class FriendsCustomTableViewCell : UITableViewCell
{
    public FriendsCustomTableViewCell (IntPtr handle) : base (handle)
    {
    }

    public FriendsCustomTableViewCell(NSString cellId, string friendName, UIImage friendPhoto)
: base (UITableViewCellStyle.Default, cellId)
    {
        FriendNameLabel.Text = friendName;
        FriendPhotoImageView.Image = friendPhoto;
    }

    //This methods is to update cell data when reuse:
    public void UpdateCellData(string friendName, UIImage friendPhoto)
    {
        FriendNameLabel.Text = friendName;
        FriendPhotoImageView.Image = friendPhoto;
```

```
    }
}
```

В UITableViewSource вы должны объявить cellIdentifier в верхней части класса (в моем случае это «FriendCell»), а в методе «GetCell» вы должны бросать ячейки и устанавливать для них данные:

```
string cellIdentifier = "FriendCell";

public override UITableViewCell GetCell(UITableView tableView, NSIndexPath indexPath)
{
    FriendsCustomTableViewCell cell = (FriendsCustomTableViewCell)
tableView.DequeueReusableCell(cellIdentifier);
    Friend friend = _friends[indexPath.Row];

    //---- if there are no cells to reuse, create a new one
    if (cell == null)
    { cell = new FriendsCustomTableViewCell(new NSString(cellIdentifier), friend.FriendName,
new UIImage(NSData.FromArray(friend.FriendPhoto))); }

    cell.UpdateCellData(friend.UserName, new UIImage(NSData.FromArray(friend.FriendPhoto)));

    return cell;
}
```

Прочитайте Создавайте и используйте пользовательские ячейки таблицы прототипов в xamarin.iOS с помощью раскадровки онлайн: https://riptutorial.com/ru/xamarin-ios/topic/5907/создавайте-и-используйте-пользовательские-ячейки-таблицы-прототипов-в-xamarin-ios-с-помощью-раскадровки

глава 22: Управление скриншотом в многозадачном коммутаторе iOS

Вступление

В Руководстве по программированию приложений для iOS :

Перед перемещением на задний план удалите конфиденциальную информацию из представлений.

Когда приложение переходит на задний план, система делает снимок главного окна приложения, который затем кратко отображается при переходе приложения на передний план.

замечания

Адаптировано из фактического вопроса StackOverflow Управление скриншотом в многозадачном коммутаторе iOS7 и ответ Obj-c Ответ

Examples

Показать изображение для снимка

```
public override void DidEnterBackground(UIApplication application)
 {
    //to add the background image in place of 'active' image
    var backgroundImage = new UIImageView();
    backgroundImage.Tag = 1234;
    backgroundImage.Image = UIImage.FromBundle("Background");
    backgroundImage.Frame = this.window.Frame;
    this.window.AddSubview(backgroundImage);
    this.window.BringSubviewToFront(backgroundImage);
}

public override void WillEnterForeground(UIApplication application)
{
    //remove 'background' image
    var backgroundView = this.window.ViewWithTag(1234);
    if(null != backgroundView)
        backgroundView.RemoveFromSuperview();
}
```

Прочитайте Управление скриншотом в многозадачном коммутаторе iOS онлайн: https://riptutorial.com/ru/xamarin-ios/topic/8681/управление-скриншотом-в-многозадачном-коммутаторе-ios

глава 23: Ящик навигации Xamarin.iOS

Синтаксис

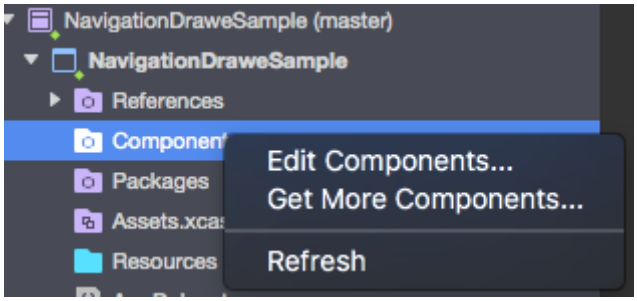
1. Компонент навигации Flayout: https://components.xamarin.com/view/flyoutnavigation

Examples

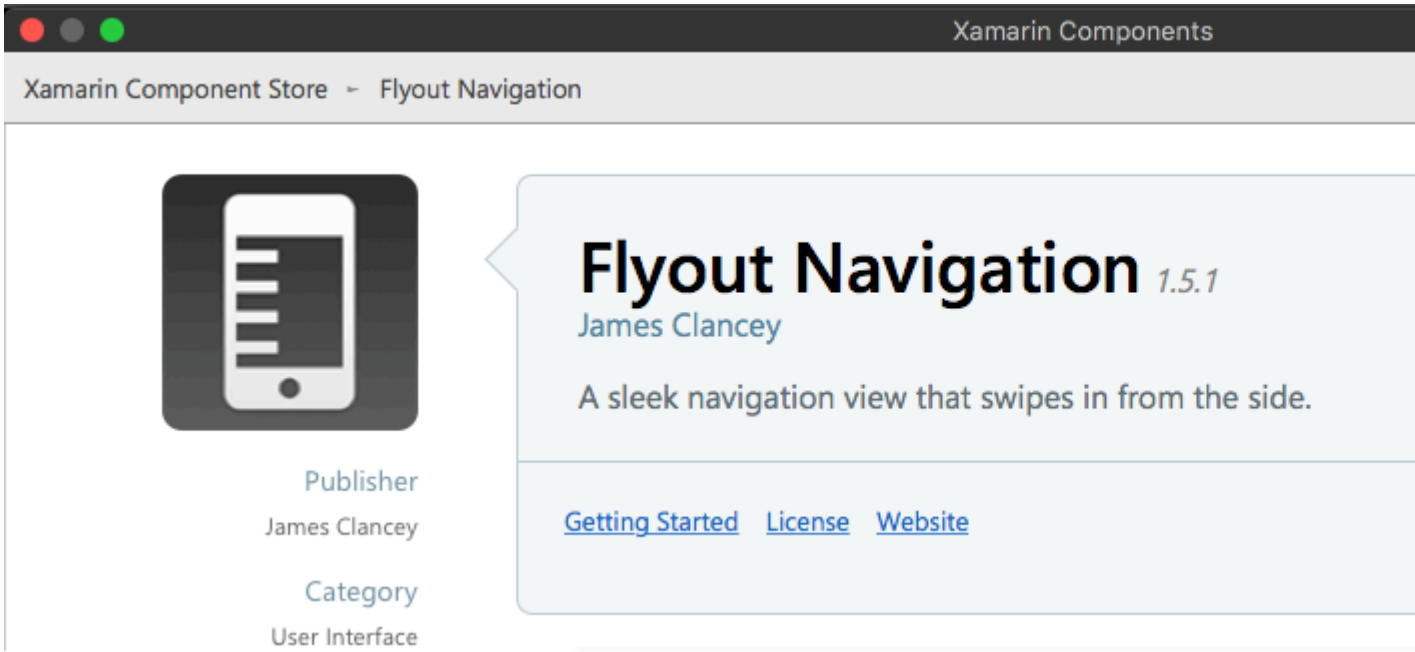
Ящик навигации Xamarin.iOS

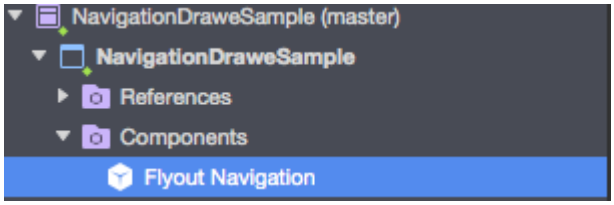
1. Создайте новый пустой проект Xamarin.iOS (приложение Single View).

2. Щелкните правой кнопкой мыши на папке «Компоненты» и выберите «Получить дополнительные компоненты»:

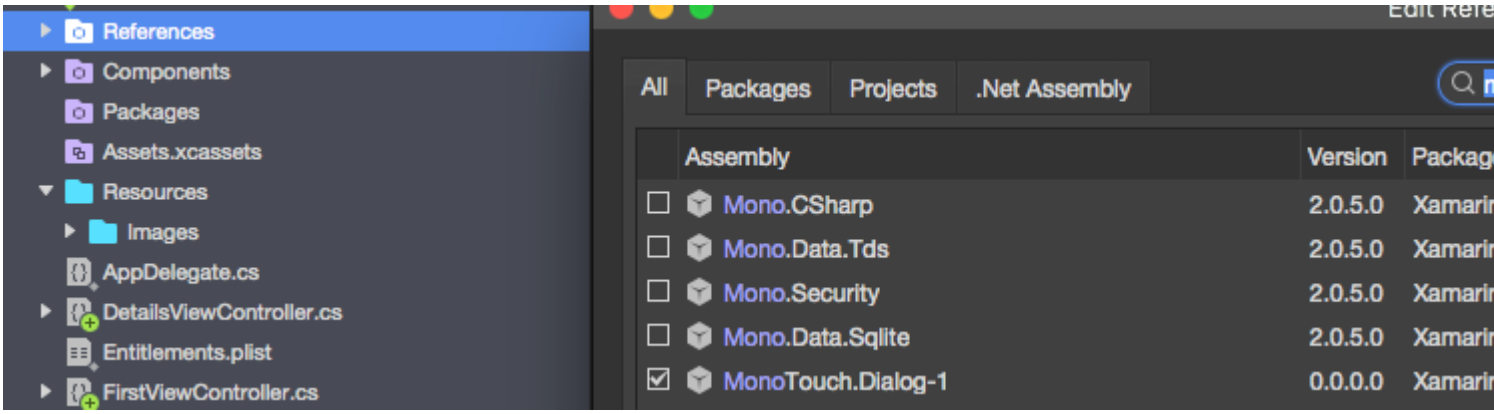


3. В поле поиска введите: «Flout Navigation» и добавьте ниже компонент в ваше приложение:

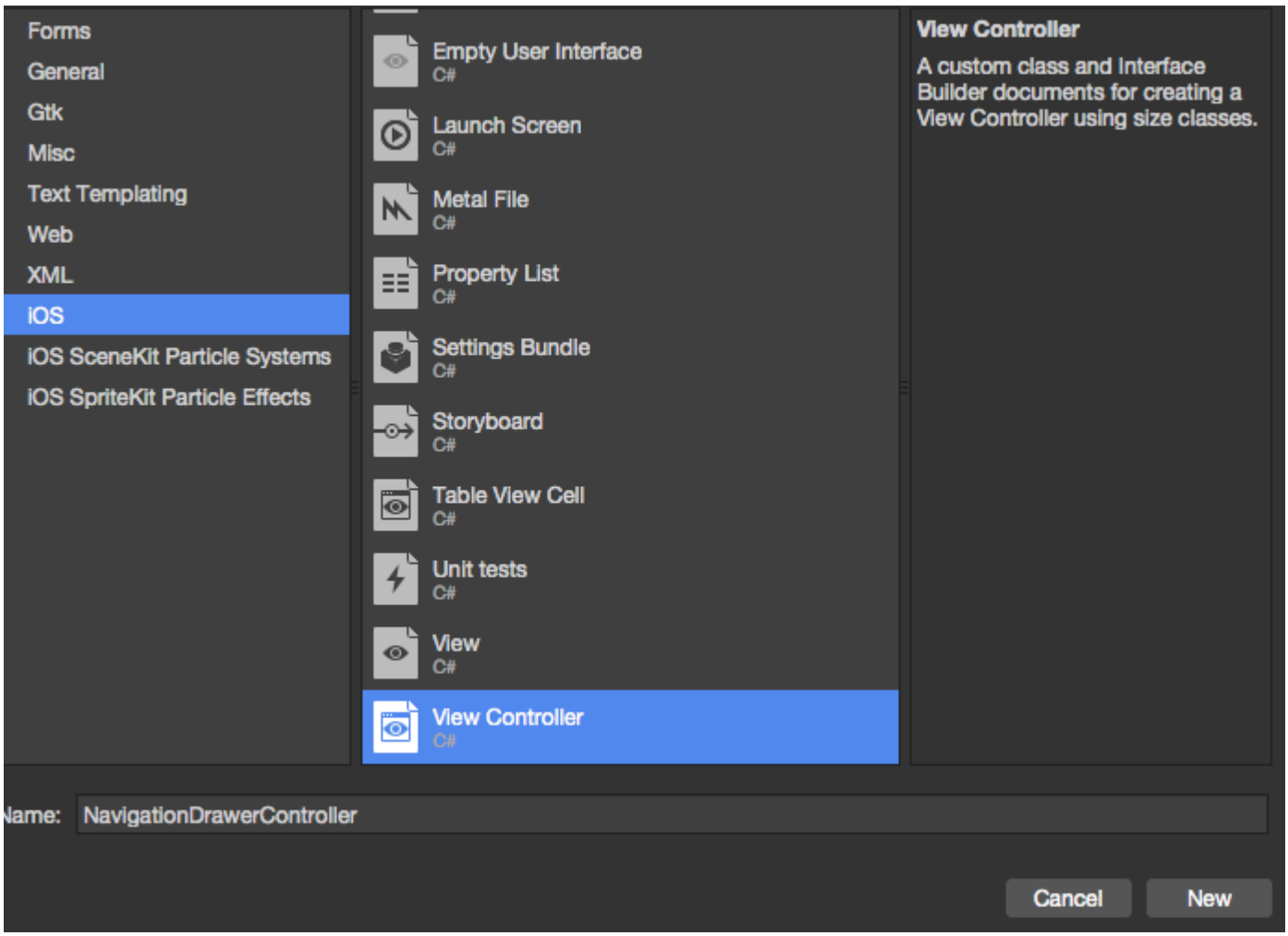




Помните также, чтобы добавить ссылку «Mono.Touch.Dialog-1»:



4. Теперь жестко нажмите на проект и добавьте новый UIViewController под названием « NavigationDrawerController»:



5. Теперь код для класса «NavigationDrawerController» должен выглядеть следующим образом:

```
public partial class NavigationDrawerController : UIViewController
{
    public NavigationDrawerController(IntPtr handle) : base(handle)
    {
    }
```

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

        NavigationItem.LeftBarButtonItem = getMenuItem();
        NavigationItem.RightBarButtonItem = new UIBarButtonItem { Width = 40 };
    }

    UIBarButtonItem getMenuItem()
    {
        var item = new UIBarButtonItem();
        item.Width = 40;
        //Please provide your own icon or take mine from the GitHub sample:
        item.Image = UIImage.FromFile("Images/menu_button@2x.png");
        item.Clicked += (sender, e) =>
        {
            if (ParentViewController is MainNavigationController)
                (ParentViewController as MainNavigationController).ToggleMenu();
        };

        return item;
    }
}
```

Не стоит беспокоиться о том, что «MainNavigationController» выделен красным цветом - мы добавим его на следующем шаге.

6. Теперь откройте файл «Main.storyboard»:

a) Добавьте один UIViewController:

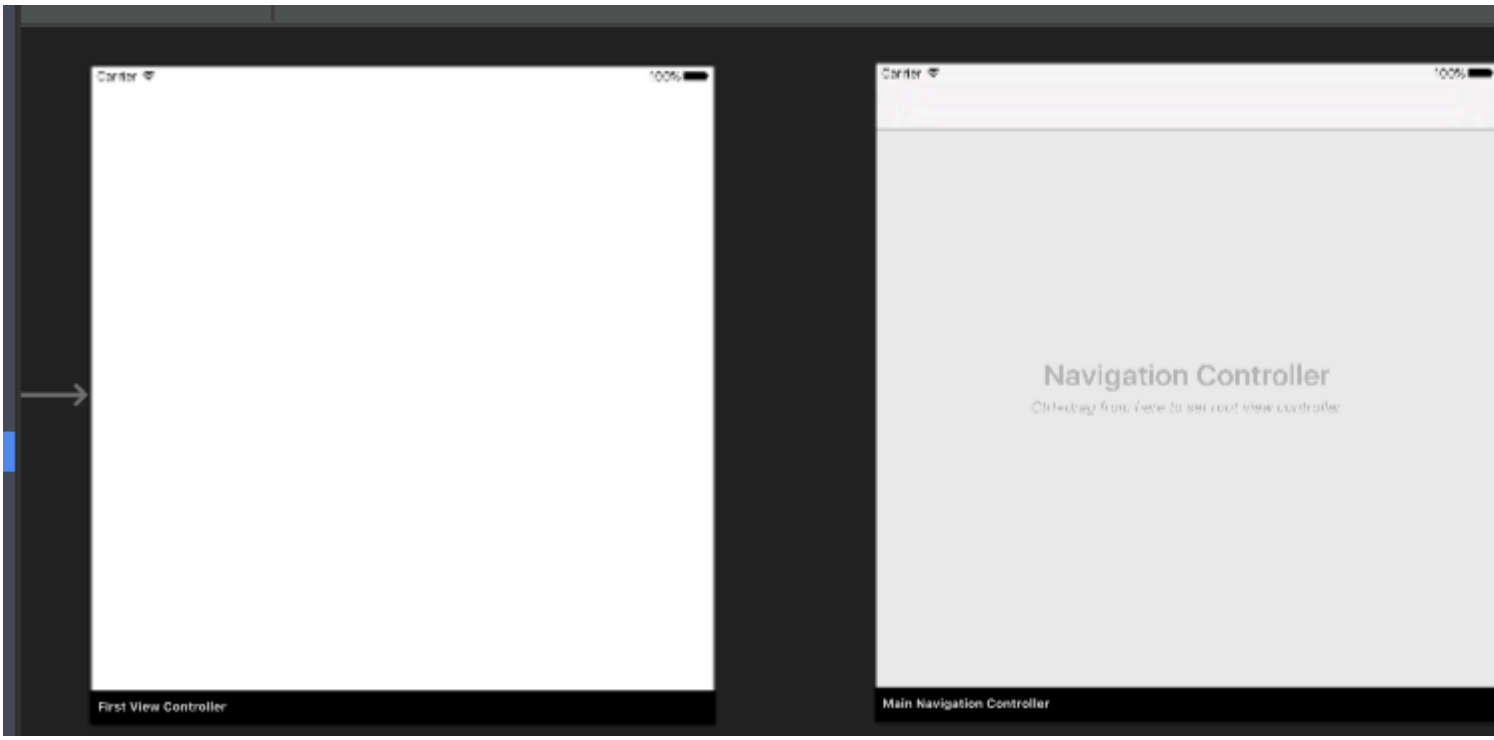
Заполните поля «Class» и «StoryboardID» с таким именем: «FirstViewController»

b) После этого добавьте контроллер навигации с корневым UIViewController:

Заполните поля «Class» и «StoryboardID» с таким именем: «MainNavigationController» для навигационного контроллера

Заполните поля «Класс» и «StoryboardID» с таким именем: «DetailsViewController» для корневого контроллера

Xamarin (или Visual) Studio создаст классы кода для вышеуказанных контроллеров.



7. Теперь откройте класс «FirstViewController» и вставьте ниже кода:

```
public partial class FirstViewController : UIViewController
{
    public FirstViewController (IntPtr handle) : base (handle)
    {
    }

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

    void createNavigationFlyout()
    {
        var navigation = new FlyoutNavigationController
        {
            //Here are sections definied for the drawer:
            NavigationRoot = new RootElement("Navigation")
            {
                new Section ("Pages")
                {
                    new StringElement ("MainPage")
                }
            },

            //Here are controllers definied for the drawer (in this case navigation controller
with one root):
             ViewControllers = new[]
             {

(MainNavigationController)Storyboard.InstantiateViewController("MainNavigationController")
             }
        };

        View.AddSubview(navigation.View);
```

```
    }
}
```

8. Откройте класс «MainNavigationController» и вставьте ниже кода:

```
public partial class MainNavigationController : UINavigationController
{
     public MainNavigationController (IntPtr handle) : base (handle)
     {
     }
     //Responsible for opening/closing drawer:
     public void ToggleMenu()
     {
         if (ParentViewController is FlyoutNavigationController)
         (ParentViewController as FlyoutNavigationController).ToggleMenu();
     }
 }
```

9. Последний класс под названием «DetailsViewController» должен выглядеть так:

```
public partial class DetailsViewController : NavigationDrawerController
{
    public DetailsViewController (IntPtr handle) : base(handle)
    {
    }
}
```

Обратите внимание, что «DetailsViewController» происходит от «NavigationDrawerController», который мы создали в начале.

Вот и все. Теперь вы можете настроить ящик, как хотите. Пожалуйста, также найдите готовый образец на моем GitHub:

https://github.com/Daniel-Krzyczkowski/XamarinIOS/tree/master/Xamarin.iOS.NavigationDrawer

Прочитайте Ящик навигации Xamarin.iOS онлайн: https://riptutorial.com/ru/xamarin-ios/topic/6574/ящик-навигации-xamarin-ios

кредиты

S. No	Главы	Contributors
1	Начало работы с Xamarin.iOS	Amy Burns, chrisntr, Community, Dominic, hankide, Sergey, valdetero
2	UIImageView увеличить в сочетании с UIScrollView	Citroenfris, valdetero
3	Автозаполнение Xalarin iOS Google Места	Conrad
4	Автоматический макет в Xamarin.iOS	ben, patridge, Tom Hawkin, valdetero
5	Вычисление переменной высоты строки в GetHeightForRow	Larry OBrien, valdetero
6	Добавить PullToRefresh в UITableView	Aditya Kumar, valdetero
7	Добавить панель поиска в UITableView	Aditya Kumar, valdetero
8	Добавление UIRefreshControl в представление таблицы	manishKungwani, valdetero
9	Использование каталогов активов	aniket.ghode, mnoronha
10	Использование каталогов активов iOS для управления изображениями	dylansturg, valdetero

11	Как использовать каталоги активов	Aditya Kumar
12	Методы изменения размера для UIImage	Frauke Nonnenmacher, raymondis, valdetero
13	Оповещения	chrisntr, Gil Sand, patridge, Pilatus, Prashant C, valdetero
14	Параллельное программирование в Xamarin.iOS	Ashan, Pilatus, Tom Gilder, valdetero
15	Подключение к Microsoft Cognitive Services	Daniel Krzyczkowski, valdetero
16	Работа с Xib и раскадровки в Xamarin.iOS	lukya
17	Рекомендации по переносу из UILocalNotification в систему уведомлений пользователей	Aditya Kumar
18	Связывание скоростных библиотек	Alex Sorokoletov, Elad Nava, Esam Sherif, J. Rahmati, James Mundy, Lucas Teixeira
19	Сенсорный ID	Amy Burns, ben, DannyC, Matthew, Peter Zhong, valdetero
20	Создавайте и используйте пользовательские ячейки таблицы прототипов в xamarin.iOS с помощью раскадровки	Daniel Krzyczkowski, valdetero
21	Управление скриншотом в многозадачном	ben

	коммутаторе iOS	
22	Ящик навигации Xamarin.iOS	Daniel Krzyczkowski, valdetero