Бесплатная электронная книга

УЧУСЬ iOS

Free unaffiliated eBook created from
Stack Overflow contributors.

#ios

700

Около

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

It is an unofficial and free 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 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 iOS

замечания

Заметки

1- Вам не нужна учетная запись Apple Developer, чтобы начать разработку приложений для iOS. Документацию и инструменты можно бесплатно загрузить с помощью Apple ID. Вы также можете подписывать и устанавливать приложения на *своих персональных устройствах,* используя тот же идентификатор Apple. Если вы хотите распространять или продавать приложения в App Store , вам необходимо зачислить Программу разработчиков Apple, начиная с 99 долларов США (это цена на момент написания статьи и может измениться). Это также добавит инциденты поддержки кода и бета-тестирования для ваших приложений через TestFlight.

2- Создание Apple ID без кредитной карты требует короткого процесса . Если вы не возражаете, связав метод оплаты как часть регистрации, перейдите на страницу https://appleid.apple.com/

- Начать разработку приложений для iOS (Swift)
- Справка Xcode (включая начало работы)
- Загрузки (включая Xcode, если вы не хотите проходить через AppStore)

Связанные теги переполнения стека

- xcode Apple IDE (интегрированная среда разработки) для разработки приложений iOS и macOS
- swift-language Один из основных языков, которые вы можете использовать для разработки в iOS.
- object-c-language Один из основных языков, которые вы можете использовать для разработки в iOS.
- cocoa API Apple для разработки в iOS и macOS.
- sprite-kit Для 2D анимированной графики.
- core-data Для хранения и получения реляционных данных.

Версии

Версия	Дата выхода
iPhone OS 2	2008-07-11
iPhone OS 3	2009-06-17
iOS 4	2010-06-08
iOS 5	2011-10-12
iOS 6	2012-09-19
IOS 7	2013-09-18
iOS 8	2014-09-17
iOS 8.1	2014-10-20
iOS 8.2	2015-03-09
iOS 8.3	2015-04-08
iOS 8.4	2015-06-30
iOS 9	2015-09-16
iOS 9.1	2015-10-22
iOS 9.2	2015-12-08
iOS 9.3	2016-03-21
iOS 10.0.1	2016-09-13
iOS 10.1	2016-10-24
iOS 10.2	2016-12-12
iOS 10.2.1	2017-01-23
iOS 10.3	2017-03-27
iOS 10.3.3	2017-07-19

Examples

Создание приложения Single View по умолчанию

Чтобы разработать приложение для iOS, вы должны начать с приложения под названием Xcode. Существуют и другие альтернативные инструменты, которые вы можете

использовать, но Xcode является официальным инструментом Apple. Обратите внимание, однако, что он работает только на macOS. Последняя официальная версия - Xcode 8.3.3 с Xcode 9 (в настоящее время в бета-версии), которая будет выпущена позднее в этом году.

1. Загрузите свой Mac и установите Xcode из App Store, если он еще не установлен.

   (Если вы предпочитаете не использовать App Store или иметь проблемы, вы также можете загрузить Xcode с веб-сайта Apple Developer , но убедитесь, что вы выбрали последнюю версию, а **не** бета-версию.)

2. Откройте Xcode. Откроется следующее окно:

Welcome to Xc

Version 8.0 (8A218a)

Get started with a playground
Explore new ideas quickly and easily.

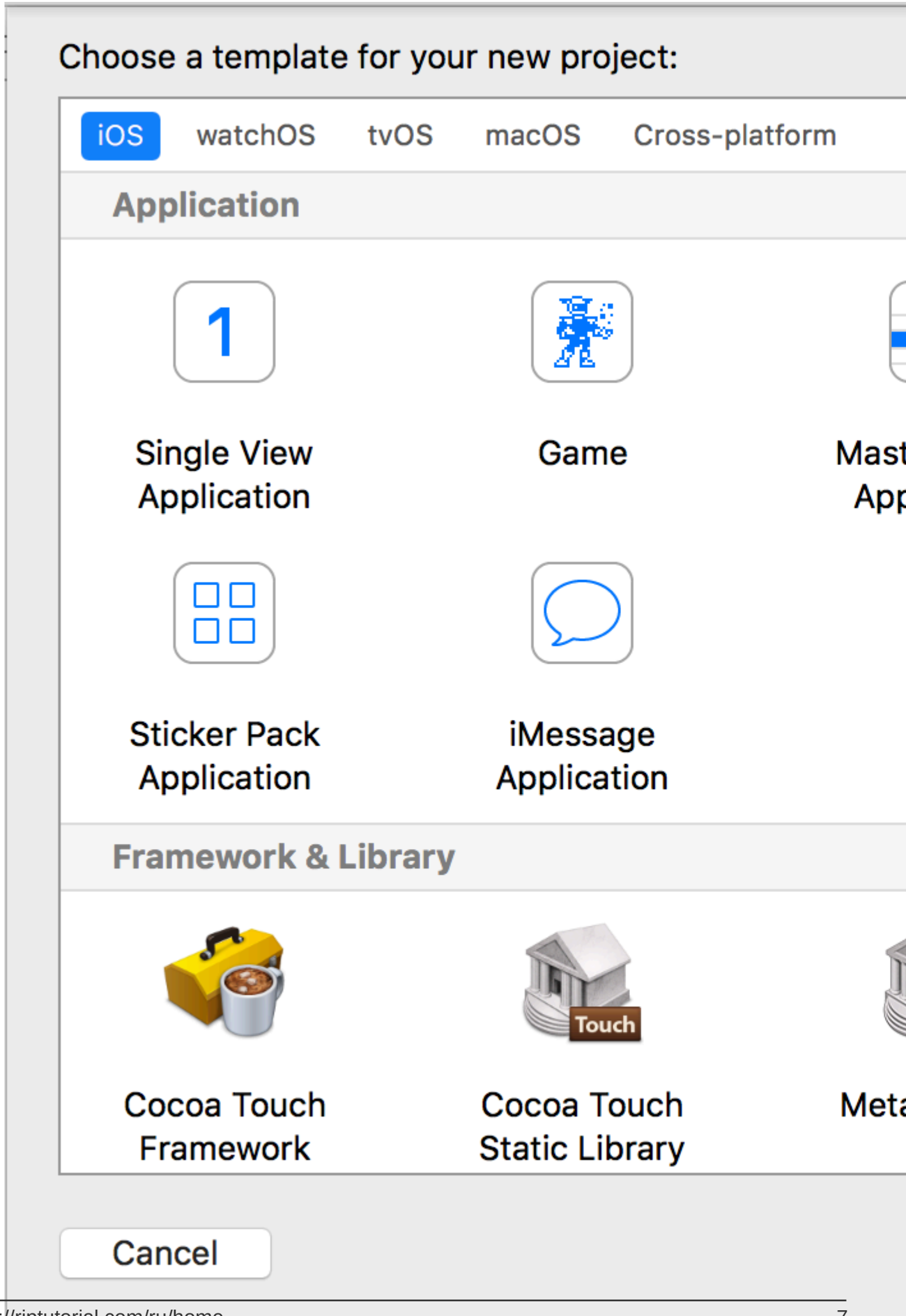
Create a new Xcode project
Create an app for iPhone, iPad, Mac, Ap

Check out an existing project
Start working on something from an SC

В окне представлены следующие параметры:

- **Начало работы с игровой площадкой:** это было введено с языком Swift и Xcode 6. Это интерактивная область, которая может использоваться для написания небольших фрагментов кода для проверки изменений во время выполнения. Это отличный способ для учеников Swift познакомиться с новыми функциями Swift.
  Xcode 6. Это интерактивная область, которая может использоваться для написания небольших фрагментов кода для проверки изменений во время выполнения. Это отличный способ для учеников Swift познакомиться с новыми функциями Swift.
- **Создайте новый проект Xcode:** *выберите этот параметр* , который создает новый проект с настройкой по умолчанию.
- **Проверьте существующий проект:** он используется для проверки проекта из местоположения репозитория, например, для проверки проекта из SVN.

3. Выберите второй вариант. **Создайте новый проект Xcode,** и Xcode попросит вас выполнить начальную настройку проекта:


Choose a template for your new project:
iOS
watchOS
tvOS
macOS
Cross-platform
Application
Single View Application
Game
Sticker Pack Application
iMessage Application
Framework & Library
Cocoa Touch Framework
Cocoa Touch Static Library
Touch
Cancel

Этот мастер используется для выбора шаблона проекта. Есть 5 вариантов:

- **iOS:** используется для создания приложений, библиотек и фреймворков iOS
- **watchOS:** Используется для создания приложений, библиотек и фреймворков watchOS
  watchOS
- **tvOS:** Используется для создания приложений, библиотек и фреймворков tvOS
- **macOS:** используется для создания приложений, библиотек, фреймворков, пакетов AppleScripts и т. д.
- **Кросс-платформенный:** используется для создания кросс-платформенных приложений, шаблонов и содержимого для покупок в приложении

Вы можете видеть, что для вашего приложения существует множество разных шаблонов. Эти шаблоны полезны для повышения вашего развития; они предварительно построены с некоторыми базовыми настройками проекта, такими как интерфейсы пользовательского интерфейса и файлы классов.

Здесь мы будем использовать первый вариант **iOS** .

1. Приложение «Мастер-детали»:

   Этот шаблон содержит объединенный интерфейс мастера и детали: мастер содержит объекты, связанные с интерфейсом детали. Выбор объектов в главном изменяет интерфейс деталей. Вы можете увидеть этот вид пользовательского интерфейса в приложениях «Настройки», «Заметки» и «Контакты» на iPad.

2. Применение на основе страницы:

   Этот шаблон используется для создания приложения на основе страницы. Страницы представляют собой разные виды, которые хранятся в одном контейнере.

3. Приложение с одним взглядом:

   Это обычный шаблон разработки приложений. Это хорошо для начинающих, чтобы изучить поток приложений.

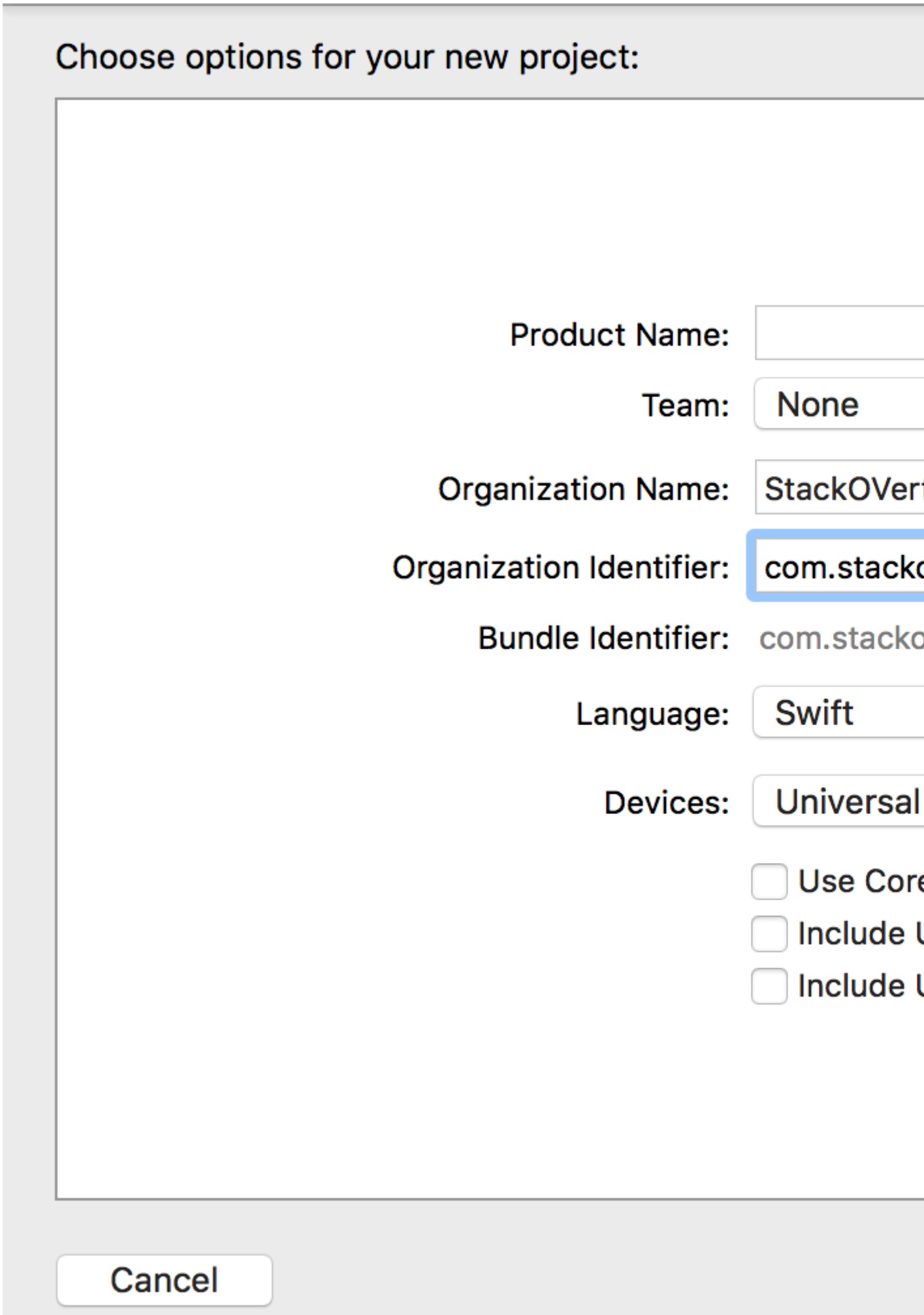
4. Приложение с вкладками:

   Этот шаблон создает вкладки в нижней части приложения. Каждая вкладка имеет другой интерфейс и другой навигационный поток. Вы можете увидеть этот шаблон, используемый в таких приложениях, как Clock, iTunes Store, iBooks и App Store.

5. Игра:

Это отправная точка для разработки игр. Вы можете пойти дальше с такими игровыми технологиями, как SceneKit, SpriteKit, OpenGL ES и Metal.

4. В этом примере мы начнем с **приложения Single View**


Choose options for your new project:
Product Name:
Team:
None
Organization Name:
StackOVer
Organization Identifier:
com.stacko
Bundle Identifier:
com.stacko
Language:
Swift
Devices:
Universal
Use Core
Include U
Include U
Cancel

Мастер поможет вам определить свойства проекта:

- **Название продукта:** название проекта / приложения
- **Название организации:** название организации, в которой вы участвуете
- **Идентификатор организации:** уникальный идентификатор организации, который используется в идентификаторе пакета. Рекомендуется использовать обратную службу имен доменов.
  уникальный идентификатор организации, который используется в идентификаторе пакета. Рекомендуется использовать обратную службу имен доменов.
- **Идентификатор пакета:** *Это поле очень важно.* Он основан на вашем имени проекта и идентификаторе организации, выбирайте с умом. Идентификатор пакета будет использоваться в будущем для установки приложения на устройство и загрузки приложения в iTunes Connect (который является местом, где мы загружаем приложения, которые будут опубликованы в App Store). Это уникальный ключ для идентификации вашего приложения.
- **Язык:** язык программирования, который вы хотели бы использовать. Здесь вы можете изменить Objective-C на Swift, если он не выбран.
- **Устройства.** Поддерживаемые устройства для вашего приложения, которые могут быть изменены позже. Он показывает iPhone, iPad и Universal. Универсальные приложения поддерживают устройства iPhone и iPad, и рекомендуется выбирать этот вариант, когда нет необходимости запускать приложение только на одном устройстве.
- **Использование основных данных.** Если вы хотите использовать Core Data Model в своем проекте, пометьте его как выбранный и создайте файл для `.xcdatamodel` . Вы также можете добавить этот файл позже, если вы не знаете заранее.
- **Include Unit Tests:** Это настраивает цель тестирования устройства и создает классы для модульного тестирования
- **Включить тест пользовательского интерфейса:** это настраивает цель тестирования пользовательского интерфейса и создает классы для тестирования пользовательского интерфейса

Нажмите « **Далее»,** и он попросит вас указать место, где вы хотите создать каталог проекта.
Нажмите « **Создать»,** и вы увидите пользовательский интерфейс Xcode с уже определенной настройкой проекта. Вы можете увидеть некоторые классы и файлы раскадровки.

Это базовый шаблон для приложения с одним представлением.

В левом верхнем углу окна проверьте, выбран ли симулятор (например, «iPhone 6», как показано здесь), а затем нажмите треугольную кнопку RUN.

5. Новое приложение откроет Simulator (это может занять некоторое время при первом запуске, и вам может понадобиться дважды попробовать, если вы увидите ошибку в первый раз). Это приложение предоставляет нам моделирование устройств для создаваемых приложений. Это похоже на реальное устройство! Он содержит некоторые приложения, такие как реальное устройство. Вы можете имитировать ориентацию, местоположение, жест встряхивания, предупреждения о памяти, строку состояния во время вызова, касание пальца, блокировку, перезагрузку, дом и т. Д.

   Вы увидите простое белое приложение, потому что мы еще не внесли никаких изменений в шаблон.

Так что начните свое. это длинный пробег, и вас ждет множество новых возможностей!

Если вы не знаете, куда идти дальше, попробуйте руководство Apple « Jump Right In ». Вы уже выполнили первые несколько шагов, чтобы отключиться.

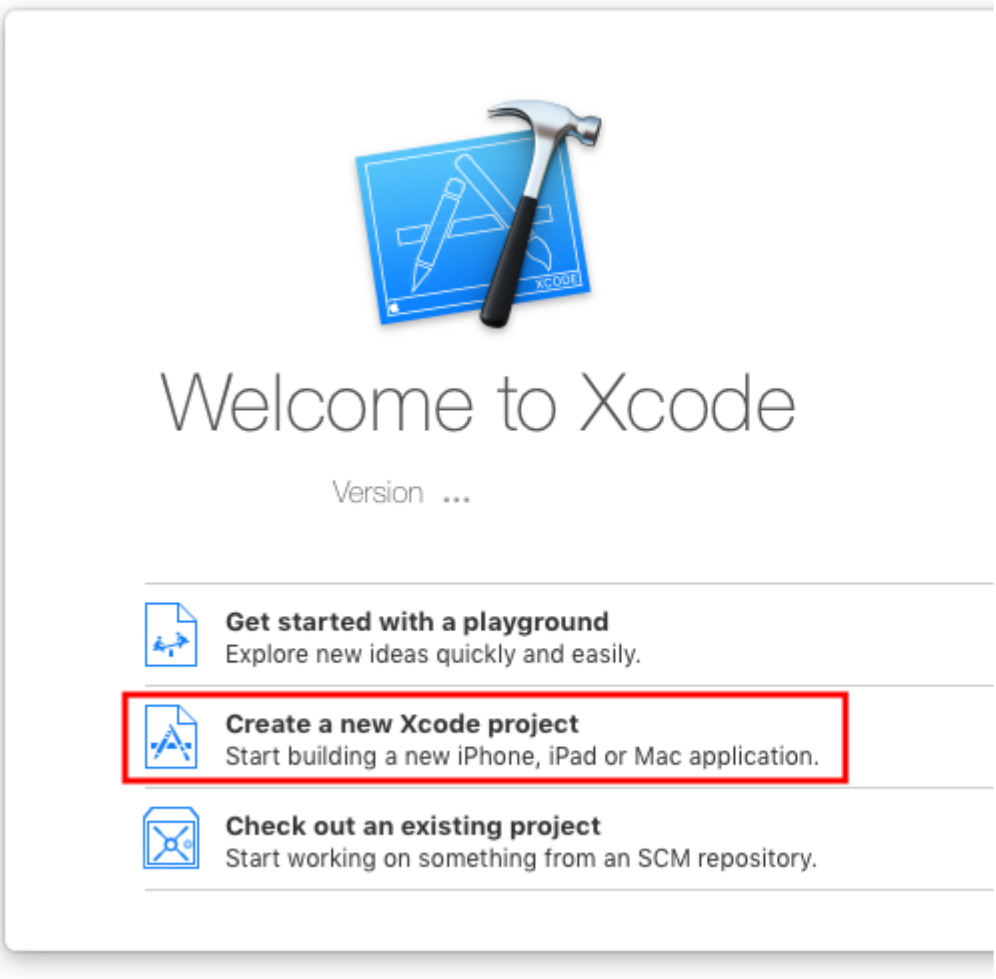
Привет, мир

После настройки Xcode нетрудно получить ваш первый iOS и запустить его.
В следующем примере мы будем:

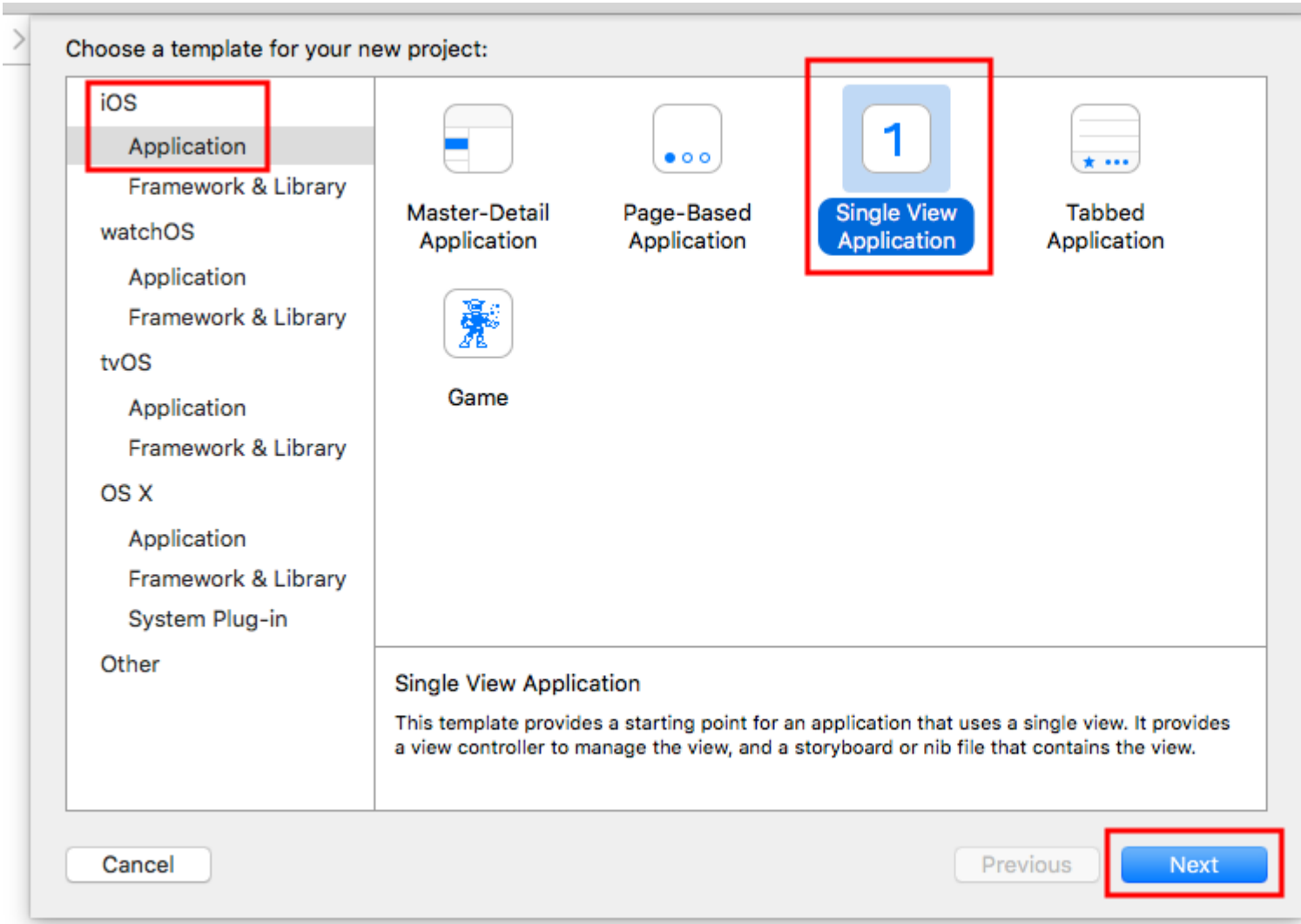
- Начать новый проект
- Добавить ярлык
- Печать сообщения на консоль.
- Запуск в симуляторе

Запуск нового проекта

Когда появится экран приветствия Xcode, выберите « **Создать новый проект Xcode»** .
Кроме того, вы можете сделать **File> New> Project ...** из меню Xcode, если вы уже открыли его.

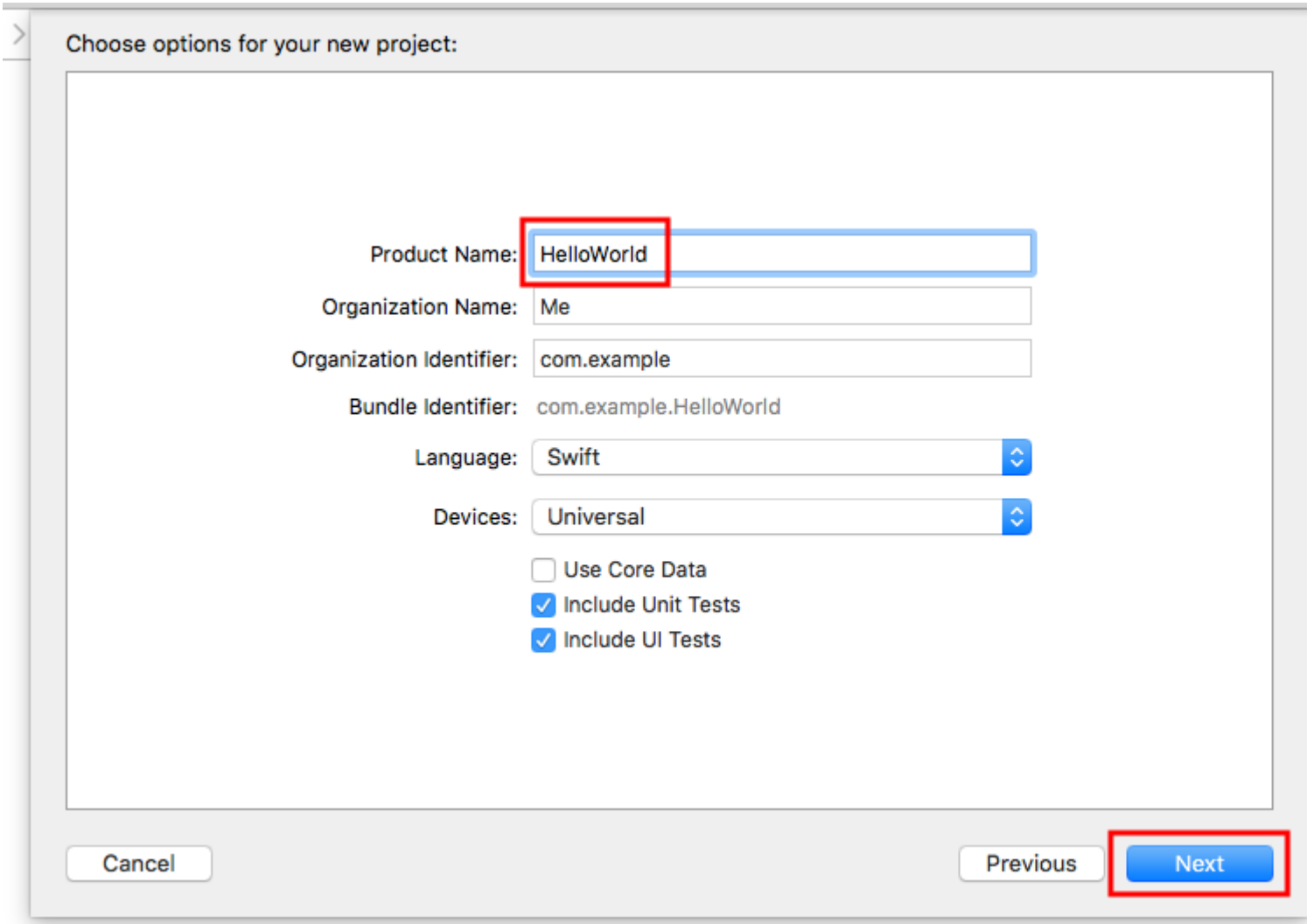


Выберите приложение **Single View** и нажмите « **Далее**» .

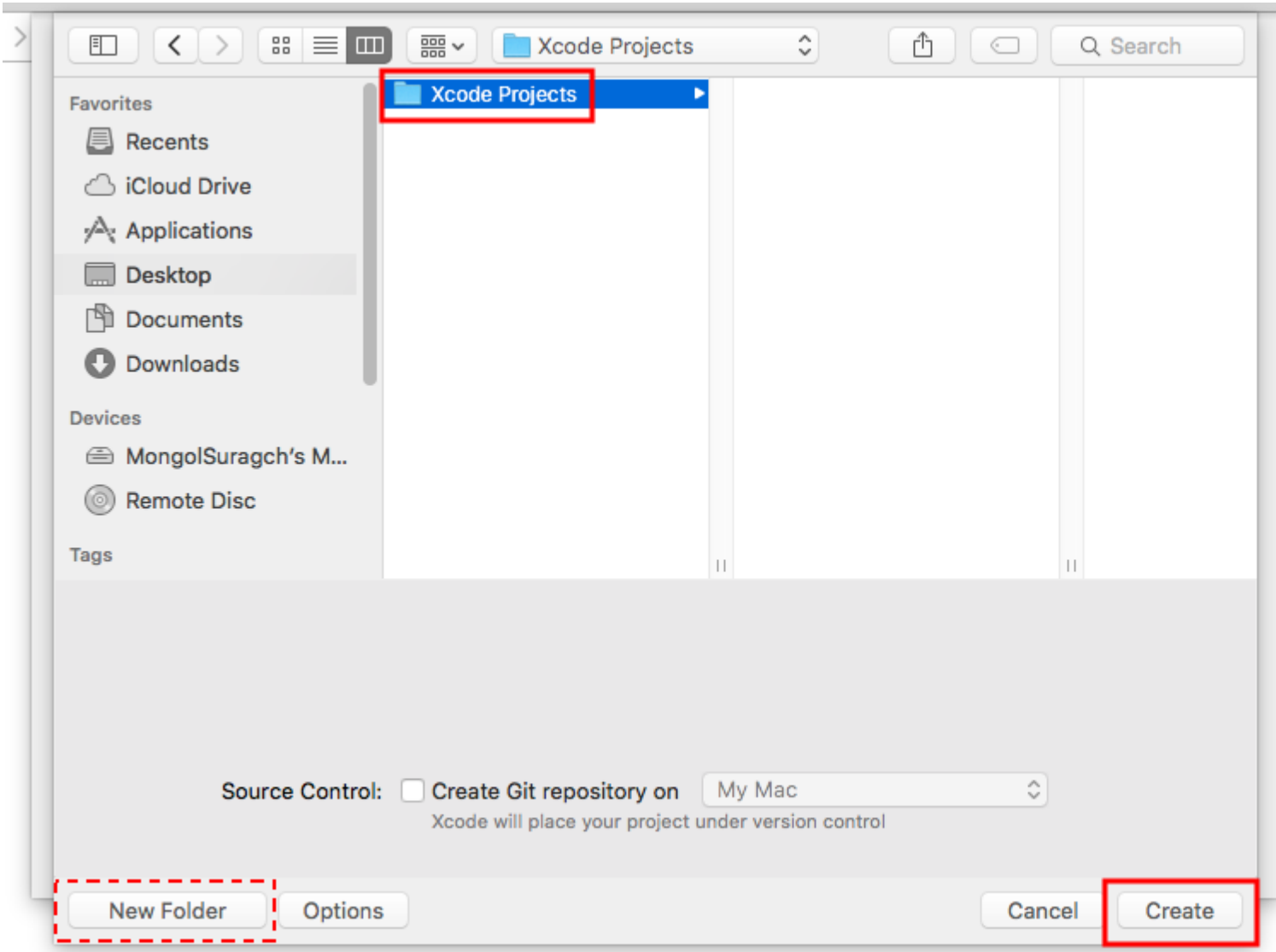


Напишите «HelloWorld» для имени **продукта** (или того, что вы действительно хотите) и в разделе « **Язык**» убедитесь, что выбран **Swift** .

- **Universal** означает, что ваше приложение будет работать как на iPhone, так и на iPad.
- **Использование Core Data** относится к постоянному хранению данных, которое не требуется в нашем приложении Hello World.
- В этом примере мы не будем выполнять **Unit Tests** или **UI Tests** , но это не повредит привычке добавлять их.



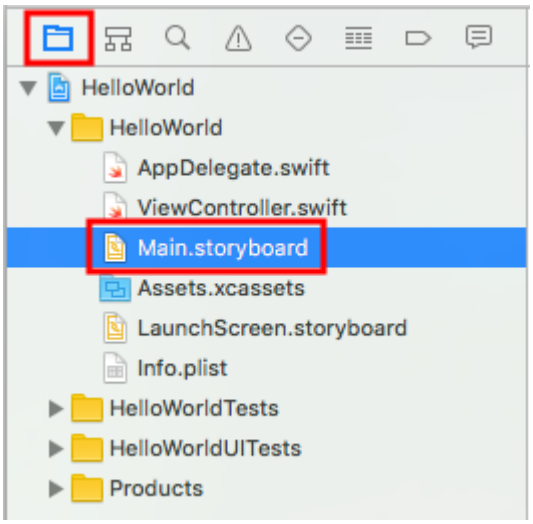
Выберите существующую папку или создайте новую, где вы сохраните проекты Xcode. Это будет дефолт в будущем. Мы создали один из них под названием «Проекты Xcode». Затем нажмите « **Создать**» . Вы можете выбрать Source Control, если хотите (используется при синхронизации с такими сайтами, как GitHub), но в этом примере нам не понадобится.



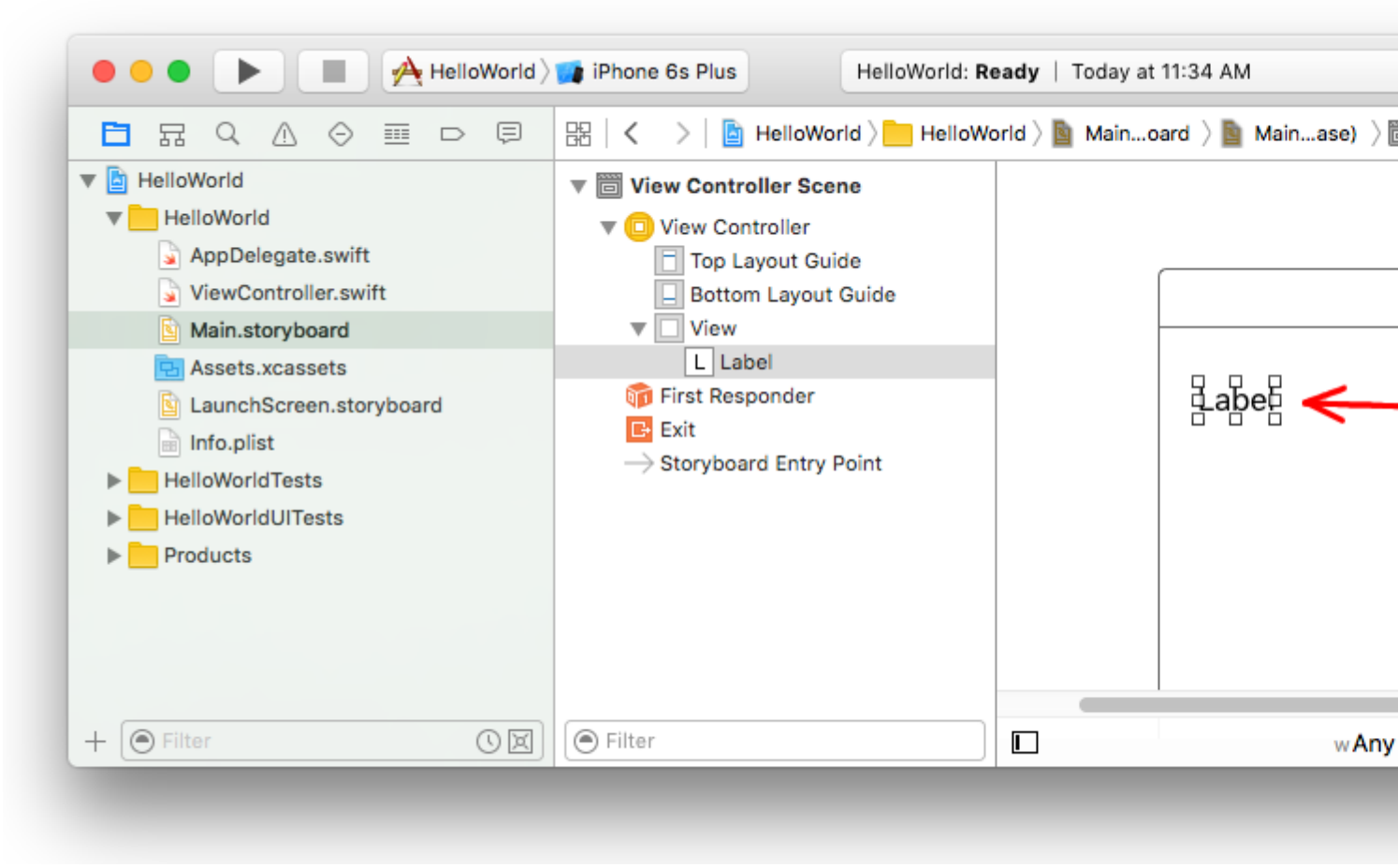
Добавление ярлыка

Это файловая структура проекта Xcode.

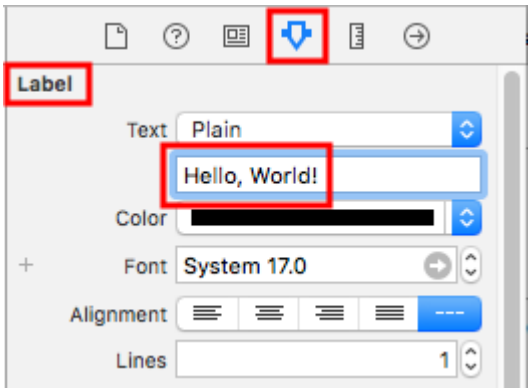
Выберите *Main.storyboard* в Навигаторе проектов.



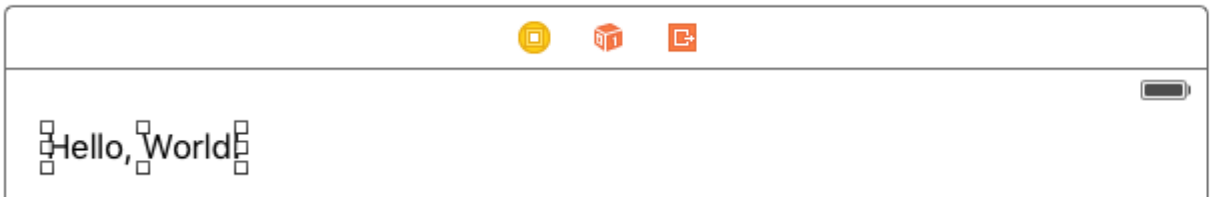
Введите «метку» в поле поиска библиотеки объектов в правом нижнем углу Xcode. Затем перетащите `UILabel` на контроллер просмотра раскадровки. Поместите его, как правило, в область верхнего левого угла.



Убедитесь, что метка выбрана на раскадровке, а затем в **Инспекторе атрибутов** , измените текст на «Hello, World!». Затем вам придется изменить размер и изменить положение метки на раскадровке, так как длина текста больше.

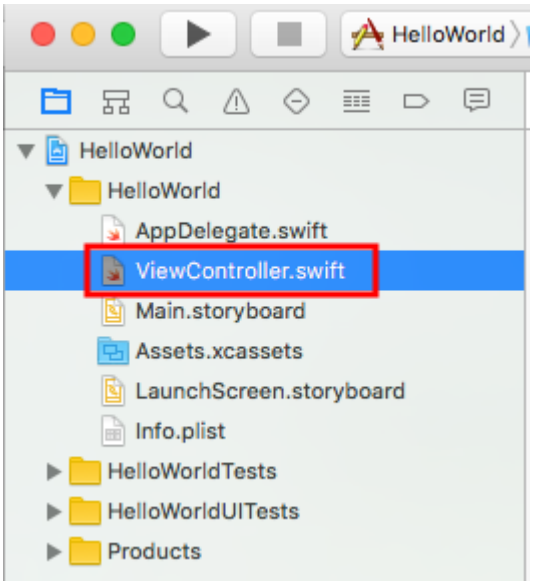


Кроме того, дважды щелкните ярлык на раскадровке, чтобы отредактировать его как « Hello, World!». Во всяком случае, раскадровка должна выглядеть примерно так:



Добавление кода

Выберите *ViewController.swift* в Навигаторе проектов.



Добавьте в метод `viewDidLoad()` `print("Successfully created my first iOS application.")` . Это должно выглядеть примерно так.

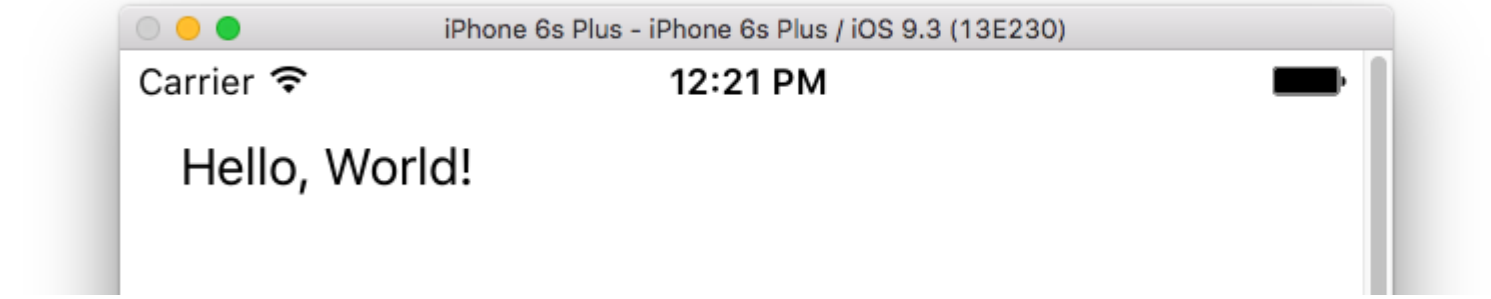
```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
```

```
        // print to the console when app is run
        print("Successfully created my first iOS application.")
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

Запуск приложения в симуляторе

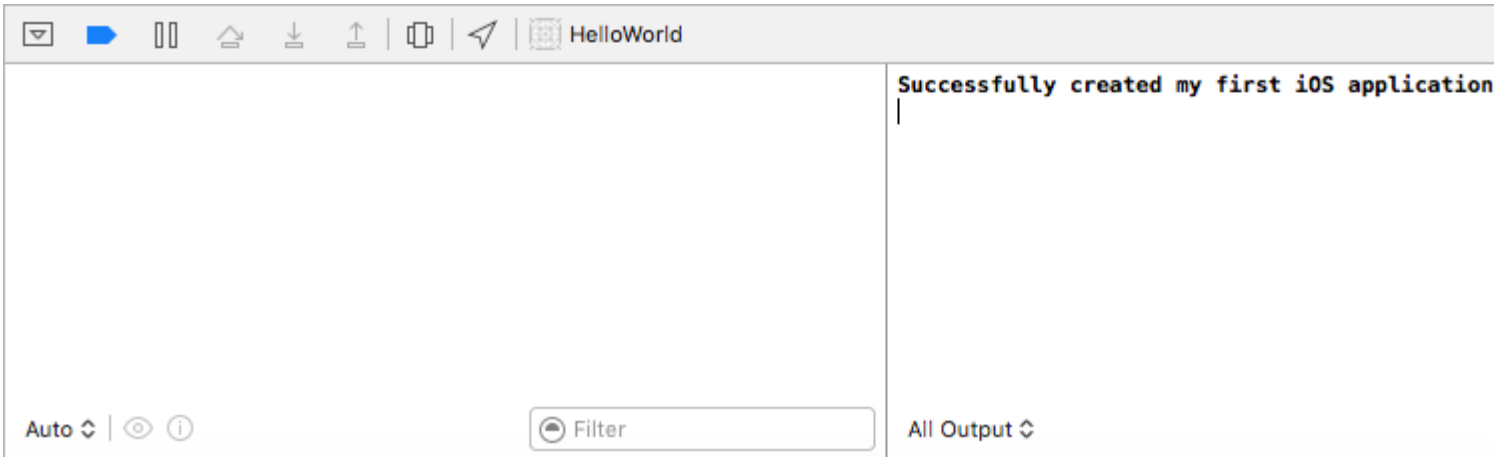
Нажмите кнопку «Выполнить», чтобы создать и запустить приложение. В этом примере текущее симуляторное устройство (называемое «схемой») по умолчанию соответствует iPhone 6s Plus. Новые версии Xcode по умолчанию будут использовать более новые схемы. Вы также можете выбрать другие схемы, щелкнув имя. Мы просто придерживаемся значения по умолчанию.

Симулятор займет некоторое время, чтобы начать с первого запуска. После запуска он должен выглядеть так:



В меню симулятора вы можете выбрать « **Окно»> «Масштаб»,** чтобы уменьшить его, или нажмите cmd + 1/2/3/4/5 для шкалы 100% / 75% / 50% / 33% / 25% соответственно.

Область отладки Xcode (внизу) также должна была напечатать «Успешно создано мое первое приложение для iOS». на консоль. «Успешно создано мое первое приложение для iOS». message - это строка, которую вы напечатали программно в разделе « **Добавить код»** .

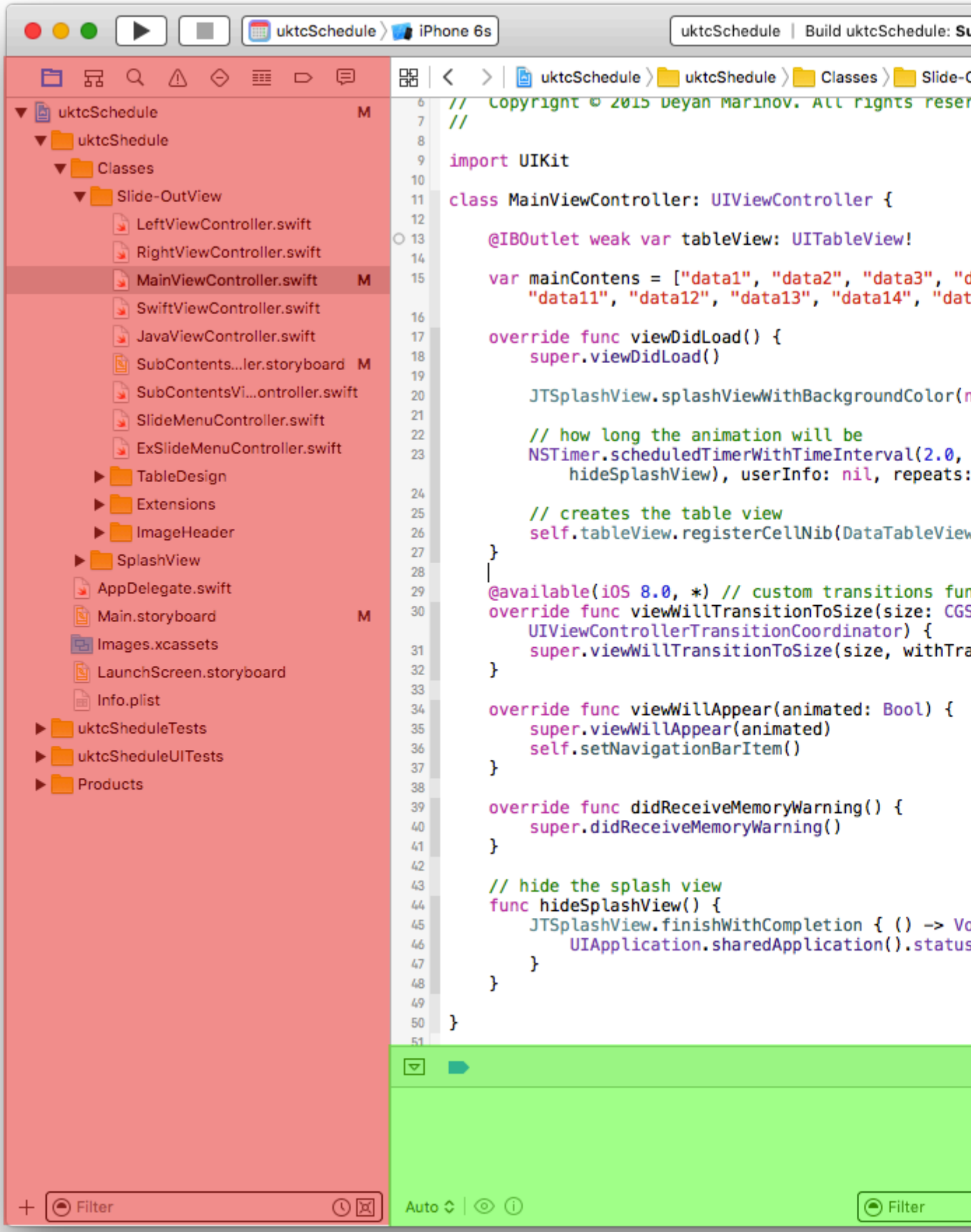


Продолжается

Ниже вы должны узнать о ограничениях автоматического макета. Это поможет вам позиционировать элементы управления на раскадровке, чтобы они выглядели хорошо на любом размере и ориентации устройства.

Интерфейс Xcode

В Xcode у вас есть три отдельные области работы: навигаторы (в красном), область отладки (зеленым) и утилиты (синим цветом).


uktcSchedule
iPhone 6s
uktcSchedule | Build uktcSchedule: Su
uktcSchedule
uktcShedule
Classes
Slide-OutView
LeftViewController.swift
RightViewController.swift
MainViewController.swift M
SwiftViewController.swift
JavaViewController.swift
SubContents...ler.storyboard M
SubContentsVi...ontroller.swift
SlideMenuController.swift
ExSlideMenuController.swift
TableDesign
Extensions
ImageHeader
SplashView
AppDelegate.swift
Main.storyboard M
Images.xcassets
LaunchScreen.storyboard
Info.plist
uktcSheduleTests
uktcSheduleUITests
Products
Filter
// Copyright © 2015 Deyan Marinov. All rights reser
//
import UIKit
class MainViewController: UIViewController {
@IBOutlet weak var tableView: UITableView!
var mainContens = ["data1", "data2", "data3", "d
"data11", "data12", "data13", "data14", "dat
override func viewDidLoad() {
super.viewDidLoad()
JTSplashView.splashViewWithBackgroundColor(
// how long the animation will be
NSTimer.scheduledTimerWithTimeInterval(2.0,
hideSplashView), userInfo: nil, repeats:
// creates the table view
self.tableView.registerCellNib(DataTableView
}
@available(iOS 8.0, *) // custom transitions fun
override func viewWillTransitionToSize(size: CGS
UIViewControllerTransitionCoordinator) {
super.viewWillTransitionToSize(size, withTra
}
override func viewWillAppear(animated: Bool) {
super.viewWillAppear(animated)
self.setNavigationBarItem()
}
override func didReceiveMemoryWarning() {
super.didReceiveMemoryWarning()
}
// hide the splash view
func hideSplashView() {
JTSplashView.finishWithCompletion { () -> Vo
UIApplication.sharedApplication().status
}
}
}
Auto
Filter

Окно рабочей области всегда включает область редактора. Когда вы выбираете файл в своем проекте, его содержимое отображается в области редактора, где Xcode открывает файл в соответствующем редакторе. Например, на изображении выше область редактора MainViewController.swift, быстрый файл кода, который выбран в области Навигатора слева от окна рабочей области.

Область навигатора

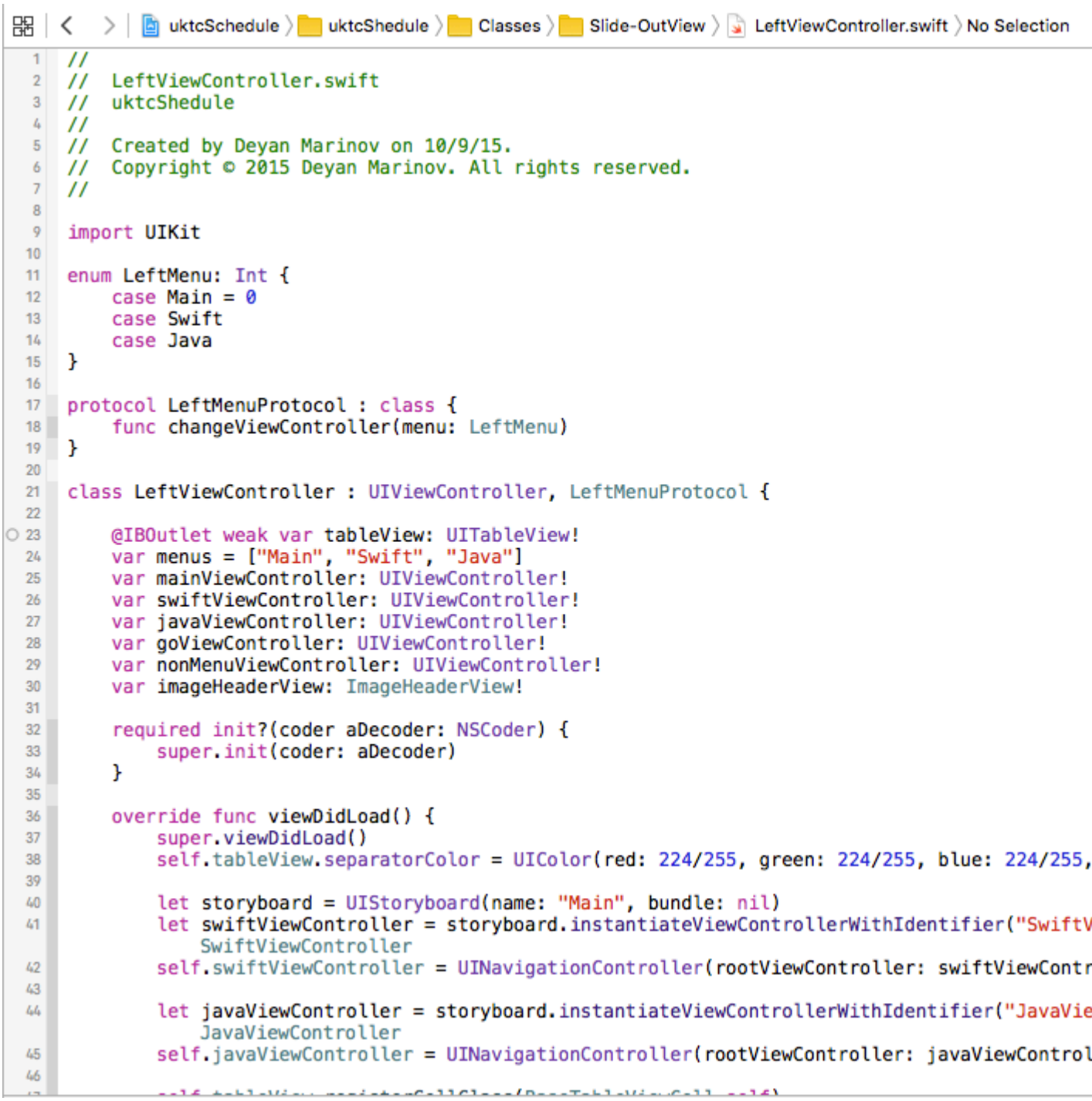
Окно навигатора содержит следующие восемь параметров:

- **Навигатор проекта.** Добавьте, удалите, группируйте и иным образом управляйте файлами в своем проекте или выберите файл для просмотра или редактирования его содержимого в области редактора.
- **Навигатор символов.** Просмотрите символы в своем проекте в виде списка или иерархии. Кнопки слева от панели фильтров позволяют ограничить отображаемые символы комбинацией только классов и протоколов, только символов в вашем проекте или только контейнеров.
- **Найти навигатор** Используйте параметры поиска и фильтры, чтобы быстро найти любую строку в вашем проекте.
- **Навигатор проблем.** Просмотрите такие проблемы, как диагностика, предупреждения и ошибки при открытии, анализе и создании проекта.
- **Тест-навигатор.** Создание, управление, запуск и просмотр модульных тестов.
- **Отладочный навигатор.** Изучите текущие потоки и связанную информацию стека в определенный момент или время во время выполнения программы.
- **Навигатор точек останова.** Точная настройка точек останова путем указания таких характеристик, как условия запуска.
- **Навигатор отчетов.** Просмотрите историю своей сборки, запуска, отладки, непрерывной интеграции и задач управления версиями.

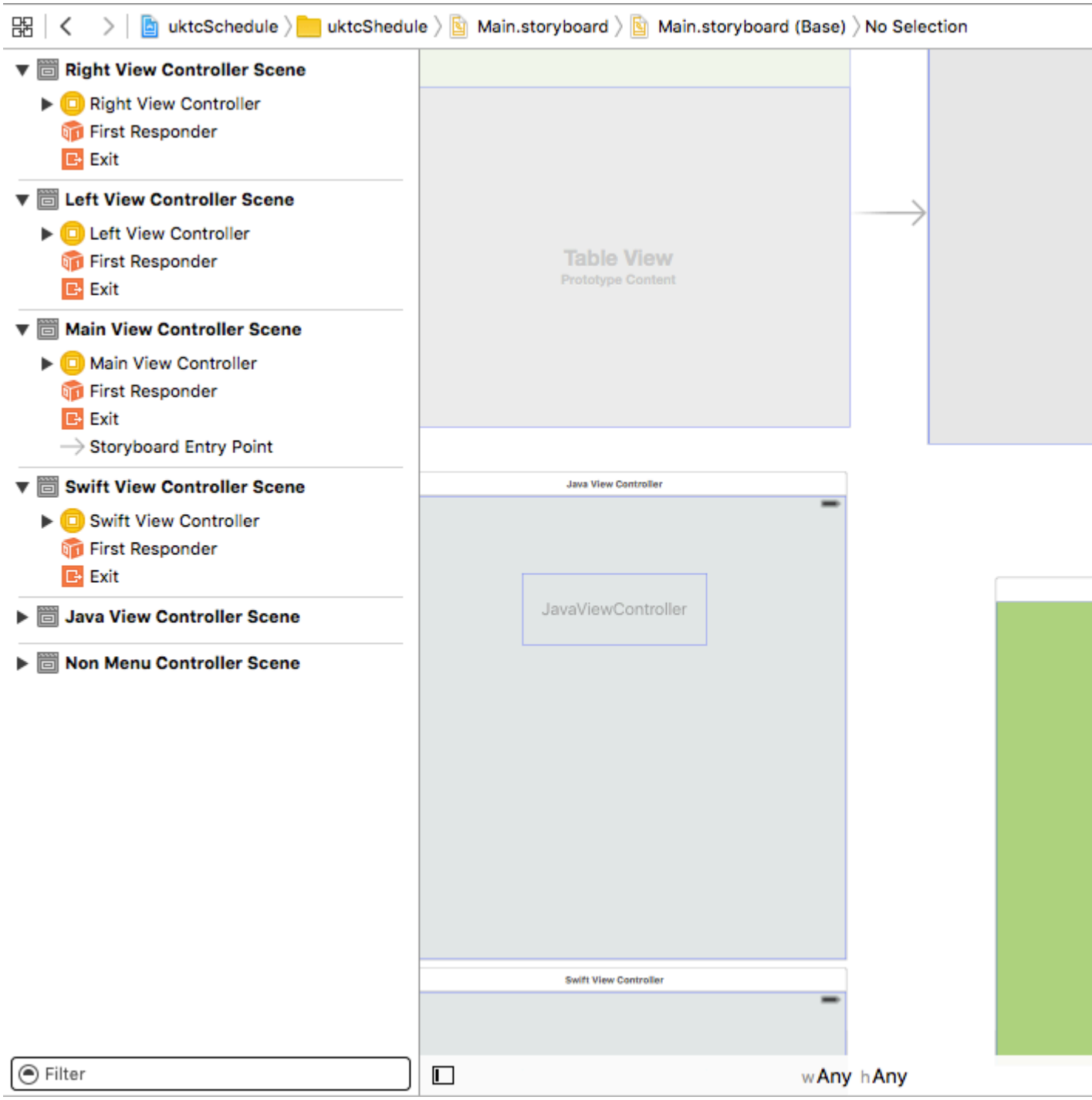
Редакторы

Большинство разработок в Xcode происходит в области редактора, главной области, которая всегда отображается в окне рабочей области. Наиболее часто используемые редакторы:

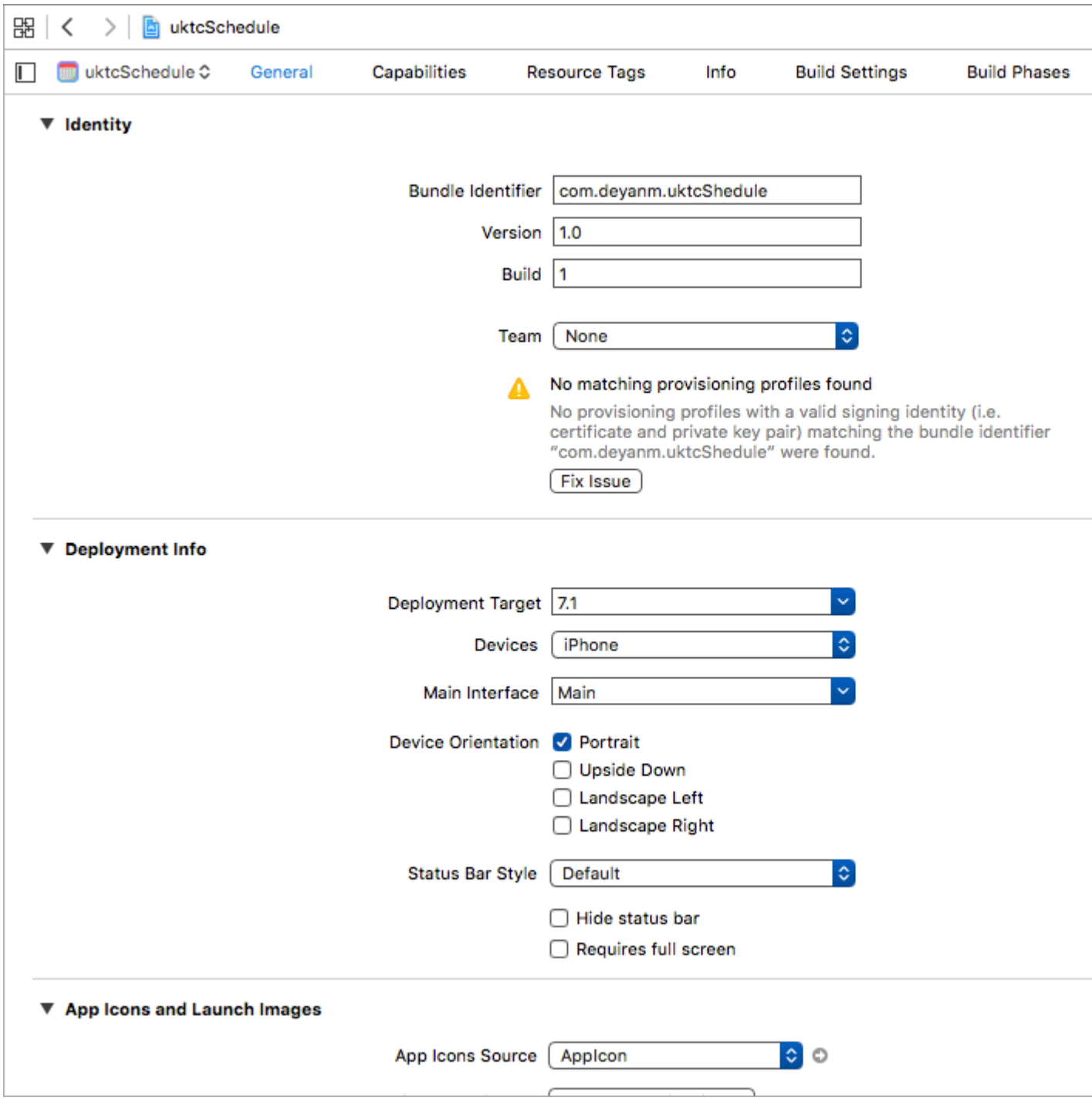
- **Редактор источника.** Напишите и отредактируйте исходный код.



- **Интерфейс Builder.** Графически создавать и редактировать файлы пользовательского интерфейса.



- **Редактор проекта.** Просмотрите и отредактируйте, как ваши приложения должны быть созданы, например, путем указания параметров сборки, целевых архитектур и прав приложений.



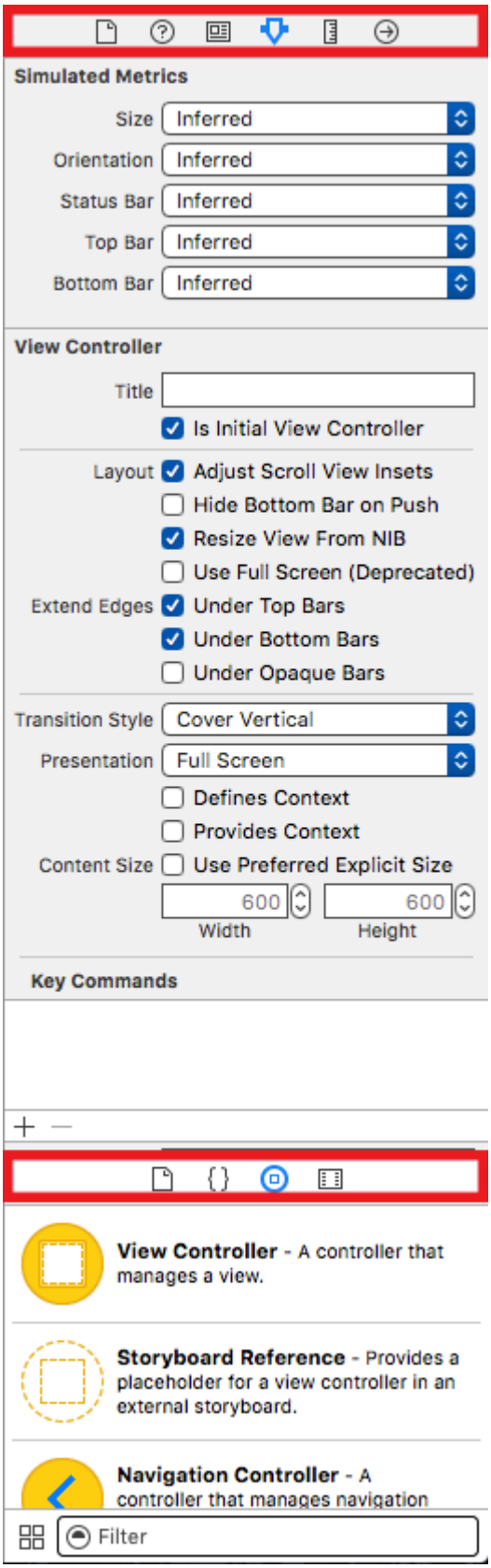
Настройте область редактора для данной задачи с помощью кнопок конфигурации редактора в правой части панели инструментов: 

- **Стандартный редактор.** Заполняет область редактора содержимым выбранного файла.
- **Помощник редактора.** Представляет отдельную панель редактора с содержанием, логически связанным с контентом в стандартной панели редактора. Вы также можете изменить содержимое.
- **Редактор версий.** Показывает различия между выбранным файлом на одной панели и другой версией того же файла во второй панели. Этот редактор работает только тогда, когда ваш проект находится под контролем источника.

Ресурсы и элементы в области коммунальных услуг

Область утилит в правом углу окна рабочей области дает вам быстрый доступ к этим ресурсам: инспекторы, для просмотра и изменения характеристик файла, открытого в редакторе. Библиотеки готовых ресурсов для использования в вашем проекте.

Верхняя панель области служебных программ отображает инспекторов. Нижняя панель предоставляет доступ к библиотекам.



Первая панель (выделенная красным цветом) - **панель Инспектор** , используйте ее, чтобы выбрать инспектора, наиболее подходящего для вашей текущей задачи. В инспекторской панели всегда видны два инспектора (дополнительные инспекторы доступны в некоторых редакторах):

- **Инспектор файлов.** Просмотр и управление метаданными для выбранного файла. Как правило, вы локализуете раскадровки и другие медиафайлы и изменяете настройки файлов пользовательского интерфейса.
- **Быстрая справка.** Подробнее о символе, элементе интерфейса или настройке сборки в файле. Например, Quick Help отображает краткое описание метода, где и как объявляется метод, его область действия, требуемые параметры и доступность платформы и архитектуры.

Используйте **панель библиотеки** (вторая выделена красным цветом) для доступа к готовым к использованию библиотекам ресурсов для вашего проекта:

- **Шаблоны файлов.** Шаблоны для общих типов файлов и кодовых конструкций.
- **Фрагменты кода.** Короткие фрагменты исходного кода для использования в вашем программном обеспечении, такие как объявления классов, потоки управления, объявления блоков и шаблоны для широко используемых технологий Apple.
- **Объекты.** Элементы пользовательского интерфейса вашего приложения.
- **Средства массовой информации.** Файлы, содержащие графику, значки, звуковые файлы и т. П.

  Чтобы использовать библиотеку, перетащите ее прямо в соответствующую область. Например, чтобы использовать фрагмент кода, перетащите его из библиотеки в исходный редактор; для создания исходного файла из шаблона файла, перетащите его шаблон в навигатор проекта.

Чтобы ограничить элементы, отображаемые в выбранной библиотеке, введите соответствующий текст в текстовое поле в **панели «Фильтр»** (нижняя панель). Например, введите «текст» в текстовом поле, чтобы показать все кнопки в библиотеке объектов.

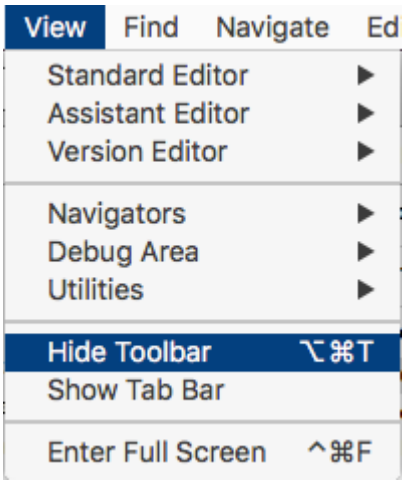
Управление задачами с помощью панели инструментов Workspace

Панель инструментов в верхней части окна рабочей области обеспечивает быстрый доступ к часто используемым командам. **Кнопка** « **Выполнить»** создает и запускает ваши продукты. **Кнопка** « **Стоп»** завершает ваш текущий код. **Меню Scheme** позволяет вам настроить продукты, которые вы хотите построить и запустить. Средство **просмотра активности** показывает ход выполнения задач, выполняемых в настоящее время, путем отображения сообщений о статусе, прогресса сборки и другой информации о вашем проекте.

Кнопки конфигурации редактора (первая группа из трех кнопок) позволяют настраивать область редактора, а **кнопки конфигурации рабочей области** (вторая группа из трех кнопок) скрывают или отображают дополнительные области навигатора, отладки и

служебных программ.

Меню « **Вид**» содержит команды для скрытия или отображения панели инструментов.



Создайте свою первую программу в Swift 3

Здесь я представляю, как создать первую базовую программу на языке Swift 3. Сначала вам нужно иметь знание базового знания в области программирования или не быть готовым изучить его с самого начала.

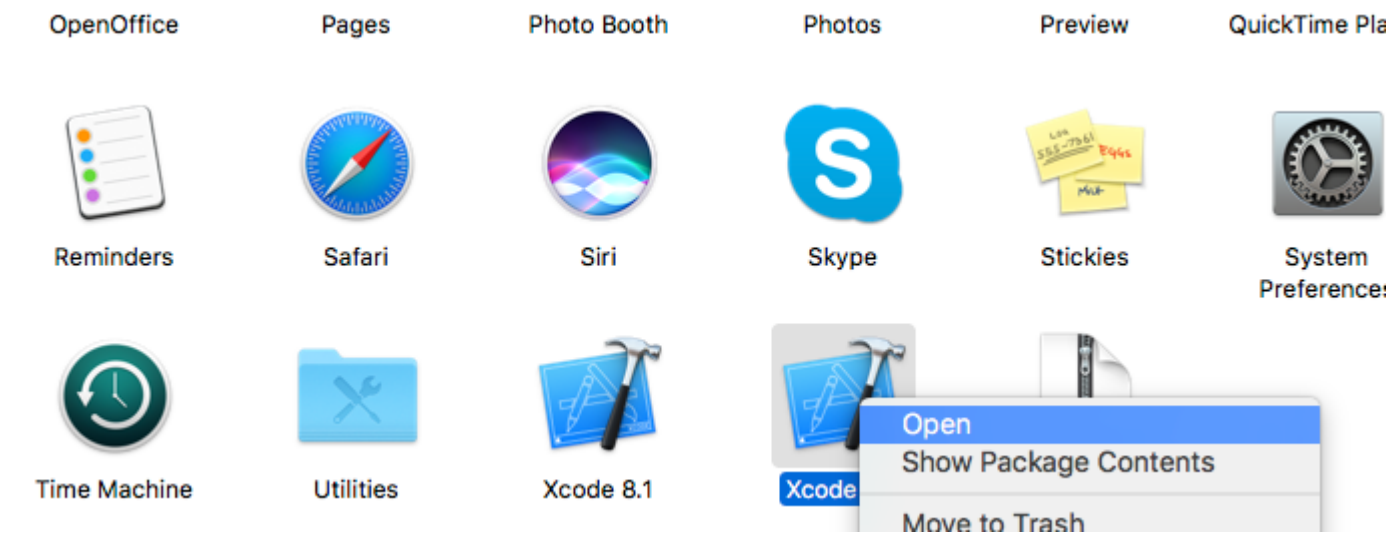
Требования к разработкам:

1. MAC OS - версия 10.11.6 или новее для нового Xcode 8.2
2. Xcode - версия 8.2 Apple Document для внедрения Xcode.

Xcode 8.2 имеет новые функции языка Swift 3 с новыми совместимыми с iOS 10 APi.

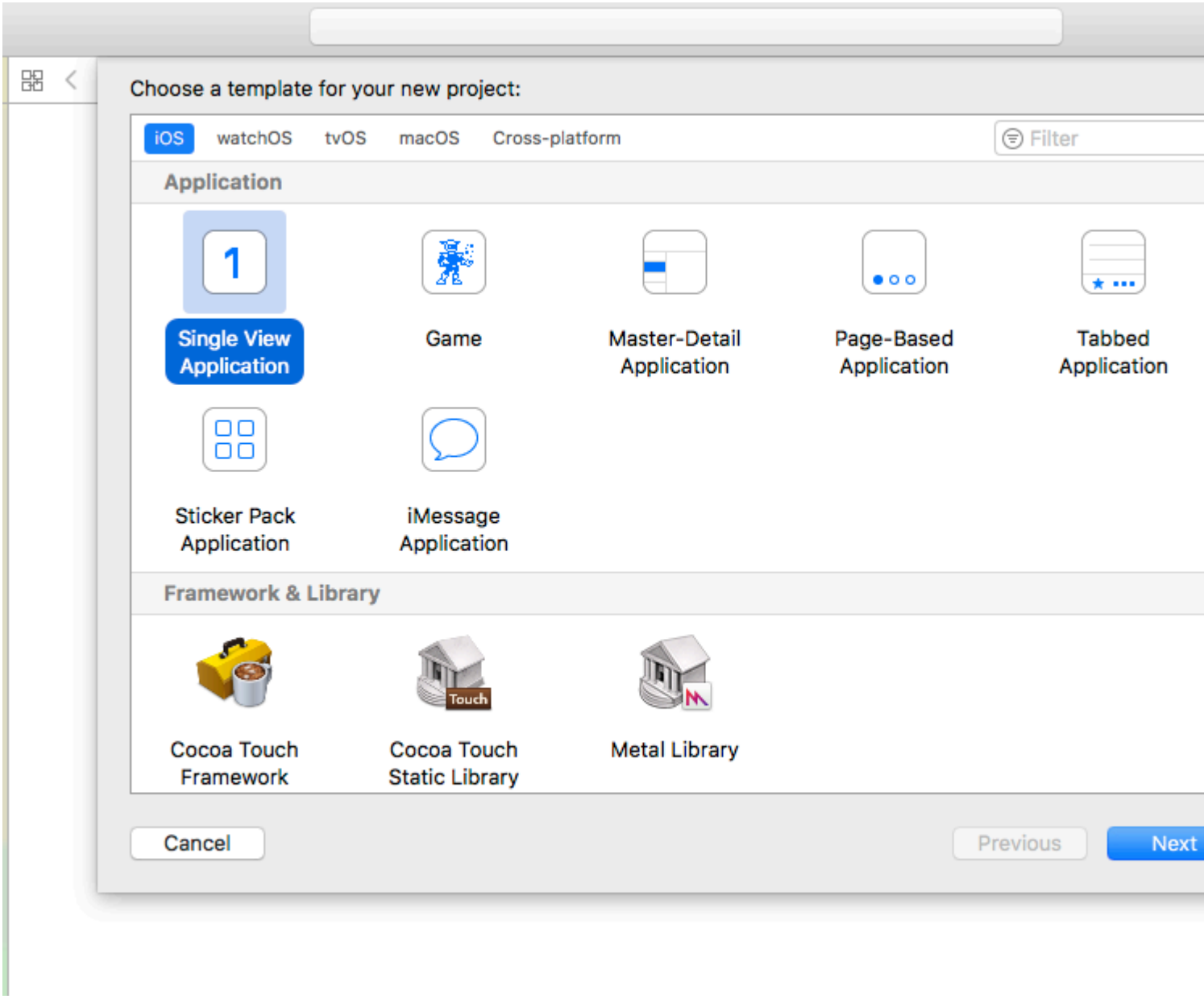
Создайте свою первую программу

Сначала перейдите в приложение и откройте свой Xcode 8.2.



После этого вы увидите экран

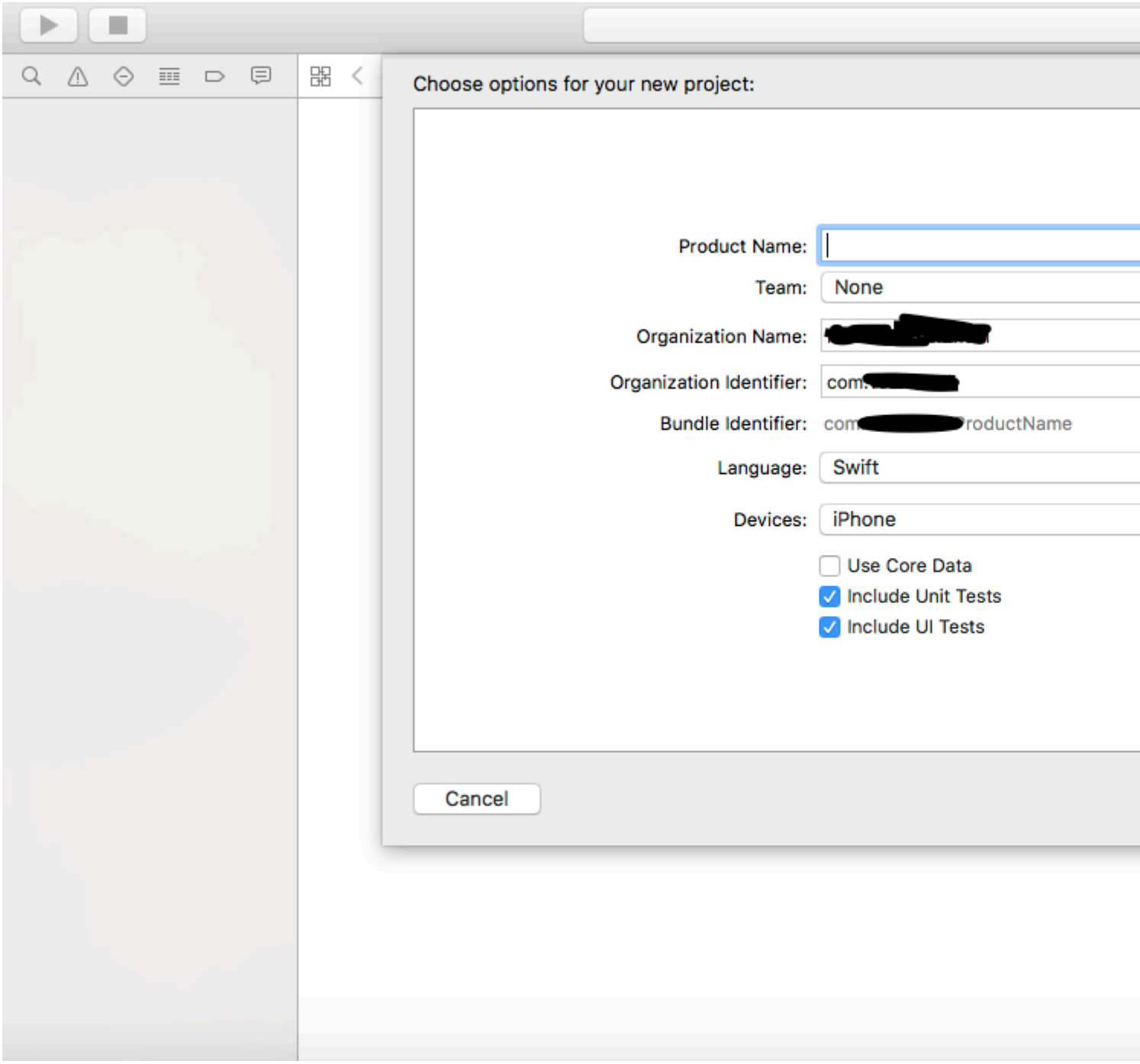
Затем выберите «Создать новый проект», после чего вы увидите следующий экран



Это также очень важная часть внутри Xcode для выбора нашего типа проекта. Нам нужно выбрать наш проект в соответствии с типами ОС. На верхней стороне есть пять типов опций:

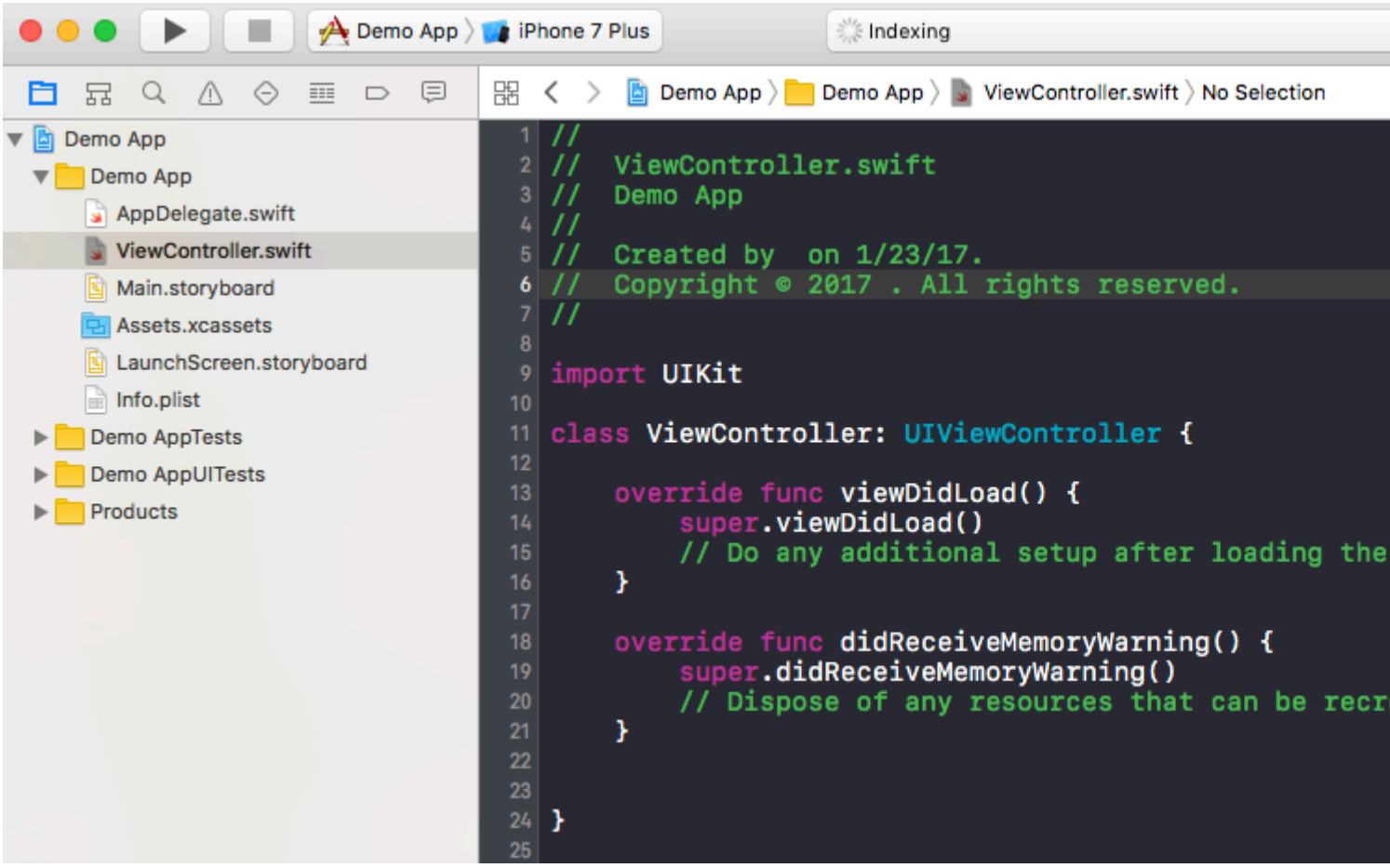
1. IOS
2. watchOS
3. Macos
4. Кросс-платформенная

Теперь мы выбираем платформу iOS для разработки и создаем очень простой проект с одним вариантом приложения:



Затем нам нужно указать имя продукта, это будет представлять ваше имя и имя приложения.

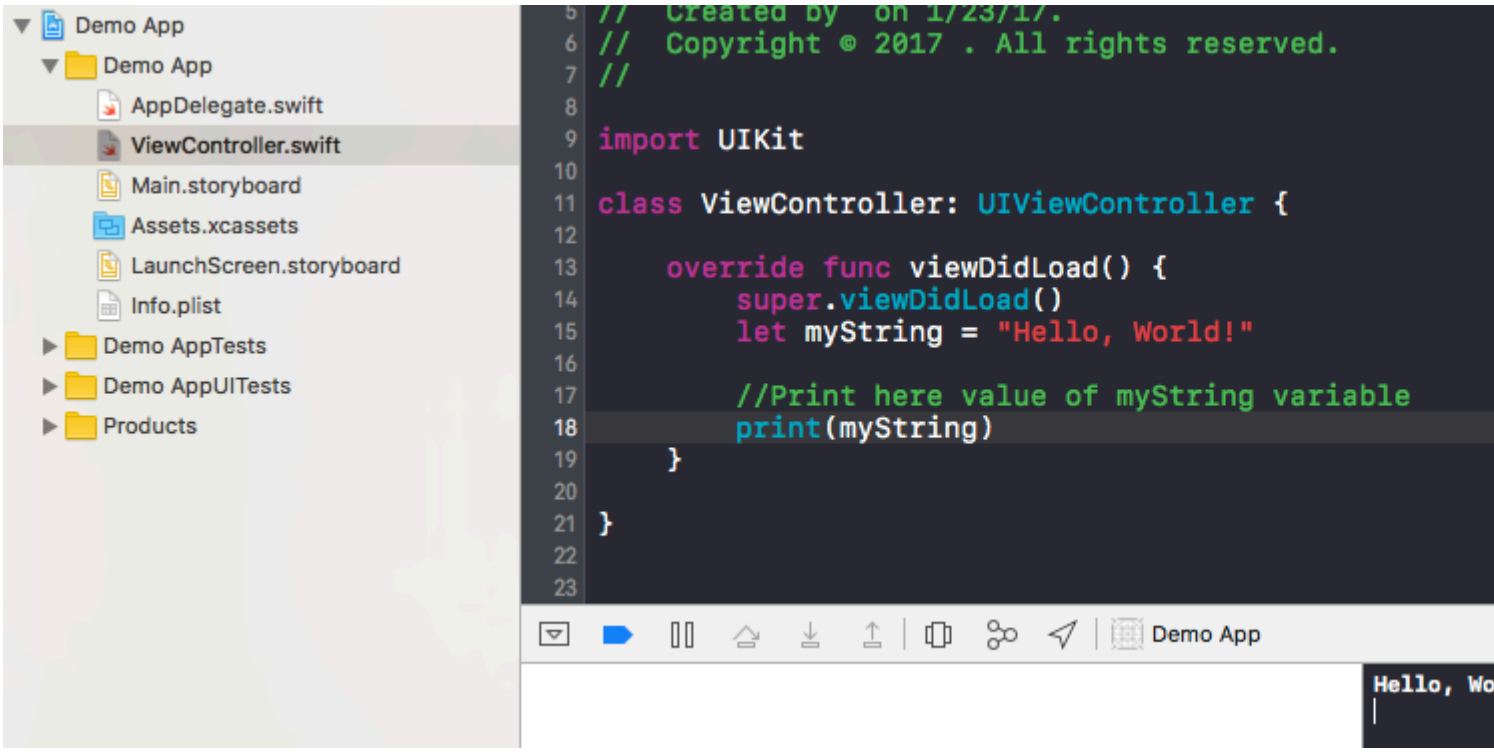
> Название приложения вы можете изменить позже в соответствии с вашими требованиями. Затем нам нужно нажать «Создать», после чего ваш экран будет выглядеть следующим образом:



Внутри этого класса вы можете видеть, что имя файла - ViewController.swift, и внутри класса это имя также является ViewController, которое является наследованием суперкласса UIViewController и, наконец, мы создаем нашу первую переменную, имя которой является **myString** типа String. Добавьте в категорию «super.viewDidLoad ()» следующее:

```
let myString = "Hello, World!"
```

Мы собираемся распечатать содержимое этой переменной. Сначала выберите тип симулятора в левом верхнем углу экрана и нажмите кнопку «Запустить».



После этого ваш вывод будет показан на терминале, который находится справа внизу. Поздравляем, это ваша первая программа Hello World внутри Xcode.

Прочитайте Начало работы с iOS онлайн: https://riptutorial.com/ru/ios/topic/191/начало-работы-с-ios

глава 2: 3D Touch

Examples

3D Touch с быстрым

3D touch был представлен с iPhone 6s Plus. С этим новым уровнем интерфейса добавлены два стиля: Peek и Pop.

Peek и Pop в двух словах

Peek - нажмите

Поп-пресса очень сильно

Проверка поддержки 3D

Вы должны проверить, поддерживает ли устройство поддержку 3D-касания. Вы можете сделать это путем проверки значения *forceTouchCapability* для объекта *UITraitCollection.* UITraitCollection описывает среду интерфейса iOS для вашего приложения.

```
if (traitCollection.forceTouchCapability == .Available) {
    registerForPreviewingWithDelegate(self, sourceView: view)
}
```

Реализация делегата

Вам необходимо реализовать два метода *UIViewControllerPreviewingDelegate* в вашем классе. Один из способов - *заглянуть,* а другой - для *поп-* поведения.

Метод, который будет реализован для *просмотра,* представляет собой *предварительный просмотрконтекста* .

```
func previewingContext(previewingContext: UIViewControllerPreviewing,
viewControllerForLocation location: CGPoint) -> UIViewController? {

    guard let indexPath = self.tableView.indexPathForRowAtPoint(location), cell =
self.tableView.cellForRowAtIndexPath(indexPath) as? <YourTableViewCell> else {
        return nil
    }

    guard let datailVC =
storyboard?.instantiateViewControllerWithIdentifier("<YourViewControllerIdentifier>") as?
<YourViewController> else {
        return nil
    }

    datailVC.peekActive = true
    previewingContext.sourceRect = cell.frame
```

```
    // Do the stuff

    return datailVC

}
```

Метод, который будет реализован для *pop,* представляет собой *предварительный просмотрконтекста* . :)

```
func previewingContext(previewingContext: UIViewControllerPreviewing, commitViewController
viewControllerToCommit: UIViewController) {

    let balanceViewController = viewControllerToCommit as! <YourViewController>

    // Do the stuff

    navigationController?.pushViewController(balanceViewController, animated: true)

}
```

Как вы видите, это перегруженные методы. Вы можете использовать 3D-Touch любым способом, реализуя эти методы.

Objective-C

```
//Checking for 3-D Touch availability
if ([self.traitCollection respondsToSelector:@selector(forceTouchCapability)] &&
        (self.traitCollection.forceTouchCapability == UIForceTouchCapabilityAvailable))
    {
        [self registerForPreviewingWithDelegate:self sourceView:self.view];
    }
//Peek
- (UIViewController *)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
             viewControllerForLocation:(CGPoint)location {

    NSIndexPath *indexPath = [self.tableView indexPathForRowAtPoint:location];
    Country *country = [self countryForIndexPath:indexPath];
    if (country) {
        CountryCell *cell = [self.tableView cellForRowAtIndexPath:indexPath];
        if (cell) {
            previewingContext.sourceRect = cell.frame;
            UINavigationController *navController = [self.storyboard
instantiateViewControllerWithIdentifier:@"UYLCountryNavController"];
            [self configureNavigationController:navController withCountry:country];
            return navController;
        }
    }
    return nil;
}
//Pop
- (void)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
commitViewController:(UIViewController *)viewControllerToCommit {

    [self showDetailViewController:viewControllerToCommit sender:self];
}
```

3 D Пример Touch-C-Touch

Objective-C

```
//Checking for 3-D Touch availability
if ([self.traitCollection respondsToSelector:@selector(forceTouchCapability)] &&
        (self.traitCollection.forceTouchCapability == UIForceTouchCapabilityAvailable))
    {
        [self registerForPreviewingWithDelegate:self sourceView:self.view];
    }
//Peek
- (UIViewController *)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
             viewControllerForLocation:(CGPoint)location {

    NSIndexPath *indexPath = [self.tableView indexPathForRowAtPoint:location];
    Country *country = [self countryForIndexPath:indexPath];
    if (country) {
        CountryCell *cell = [self.tableView cellForRowAtIndexPath:indexPath];
        if (cell) {
            previewingContext.sourceRect = cell.frame;
            UINavigationController *navController = [self.storyboard
instantiateViewControllerWithIdentifier:@"UYLCountryNavController"];
            [self configureNavigationController:navController withCountry:country];
            return navController;
        }
    }
    return nil;
}
//Pop
- (void)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
commitViewController:(UIViewController *)viewControllerToCommit {

    [self showDetailViewController:viewControllerToCommit sender:self];
}
```

Прочитайте 3D Touch онлайн: https://riptutorial.com/ru/ios/topic/6705/3d-touch

глава 3: AFNetworking

Examples

Отправка блока завершения в пользовательском потоке

Всякий раз, когда используется AFNetworking, вызов отправляется в пользовательский поток, предоставляемый AFNetworking. Когда вызов возвращается в блок завершения, он запускается в основном потоке.

В этом примере настраивается поток, который отправляется в блок завершения:

AFNetworking 2.xx:

```
// Create dispatch_queue_t with your name and DISPATCH_QUEUE_SERIAL as for the flag
dispatch_queue_t myQueue = dispatch_queue_create("com.CompanyName.AppName.methodTest",
                  DISPATCH_QUEUE_SERIAL);

// init AFHTTPRequestOperation of AFNetworking
operation = [[AFHTTPRequestOperation alloc] initWithRequest:request];

// Set the FMDB property to run off the main thread
[operation setCompletionQueue:myQueue];
```

AFNetworking 3.xx:

```
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc] init];
[self setCompletionQueue:myQueue];
```

Прочитайте AFNetworking онлайн: https://riptutorial.com/ru/ios/topic/3002/afnetworking

глава 4: Alamofire

Синтаксис

- ответ()
- responseData ()
- responseString (кодирование: NSStringEncoding)
- responseJSON (опции: NSJSONReadingOptions)
- responsePropertyList (параметры: NSPropertyListReadOptions)

параметры

параметр	подробности
метод	`.OPTIONS, .GET, .HEAD, .POST, .PUT, .PATCH, .DELETE, .TRACE, .CONNECT`
URLString	`URLStringConvertible`
параметры	`[String: AnyObject]?`
кодирование	`ParameterEncoding`
заголовки	`[String: String]?`

Examples

Сделать запрос

```
import Alamofire

Alamofire.request(.GET, "https://httpbin.org/get")
```

Автоматическая проверка

```
Alamofire.request("https://httpbin.org/get").validate().responseJSON { response in
switch response.result {
case .success:
    print("Validation Successful")
case .failure(let error):
    print(error)
 }
}
```

Обработка ответов

```
Alamofire.request(.GET, "https://httpbin.org/get", parameters: ["foo": "bar"])
     .responseJSON { response in
         print(response.request)  // original URL request
         print(response.response) // URL response
         print(response.data)     // server data
         print(response.result)   // result of response serialization

         if let JSON = response.result.value {
             print("JSON: \(JSON)")
         }
     }
```

Ручная проверка

```
Alamofire.request(.GET, "https://httpbin.org/get", parameters: ["foo": "bar"])
     .validate(statusCode: 200..<300)
     .validate(contentType: ["application/json"])
     .response { response in
         print(response)
     }
```

Обработчик ответа

```
Alamofire.request(.GET, "https://httpbin.org/get", parameters: ["foo": "bar"])
     .validate()
     .response { request, response, data, error in
         print(request)
         print(response)
         print(data)
         print(error)
      }
```

Обработчики обработанных цепочек

```
Alamofire.request(.GET, "https://httpbin.org/get")
     .validate()
     .responseString { response in
         print("Response String: \(response.result.value)")
     }
     .responseJSON { response in
         print("Response JSON: \(response.result.value)")
     }
```

Прочитайте Alamofire онлайн: https://riptutorial.com/ru/ios/topic/1823/alamofire

глава 5: API iOS Google Places

Examples

Помещение соседних мест из текущего местоположения

Предпосылки

1. Установите контейнеры в свой проект
2. Установите SDK GooglePlaces
3. Активируйте сервисы локации

Сначала нам нужно получить местоположение пользователей, получив их текущую долготу и широту.

1. Импортировать GooglePlaces и GooglePlacePicker

```
import GooglePlaces
import GooglePlacePicker
```

2. Добавить протокол `CLLOcationManagerDelegate`

```
class ViewController: UIViewController, CLLocationManagerDelegate {

}
```

3. создайте свой CLLocationManager ()

```
var currentLocation = CLLocationManager()
```

4. Запрос авторизации

```
currentLocation = CLLocationManager()
currentLocation.requetAlwayAuthorization()
```

5. Создайте кнопку, чтобы вызвать метод GooglePlacePicker

@IBAction func placePickerAction (отправитель: AnyObject) {

```
if CLLOcationManager.authorizationStatues() == .AuthorizedAlways {

        let center =
CLLocationCoordinate2DMake((currentLocation.location?.coordinate.latitude)!,
(currentLocation.location?.coordinate.longitude)!)
        let northEast = CLLocationCoordinate2DMake(center.latitude + 0.001, center.longitude +
0.001)
        let southWest = CLLocationCoordinate2DMake(center.latitude - 0.001, center.longitude -
```

```
0.001)
        let viewport = GMSCoordinateBounds(coordinate: northEast, coordinate: southWest)
        let config = GMSPlacePickerConfig(viewport: viewport)
        placePicker = GMSPlacePicker(config: config)

        placePicker?.pickPlaceWithCallback({ (place: GMSPlace?, error: NSError?) -> Void in
            if let error = error {
                print("Pick Place error: \(error.localizedDescription)")
                return
            }

            if let place = place {
               print("Place name: \(place.name)")
                print("Address: \(place.formattedAddress)")

            } else {
               print("Place name: nil")
                print("Address: nil")
            }
        })
    }
}
```

Прочитайте API iOS Google Places онлайн: https://riptutorial.com/ru/ios/topic/6908/api-ios-google-places

глава 6: API распознавания речи IOS 10

Examples

Речь в текст: распознать речь из пакета, содержащего аудиозапись

```
//import Speech
//import AVFoundation

// create a text field to show speech output
@IBOutlet weak var transcriptionTextField: UITextView!
// we need this audio player to play audio
var audioPlayer: AVAudioPlayer!

override func viewDidLoad()
{
    super.viewDidLoad()
}

// this function is required to stop audio on audio completion otherwise it will play same
audio again and again
func audioPlayerDidFinishPlaying(_ player: AVAudioPlayer, successfully flag: Bool)
{
    player.stop()
}

// this function is required to get a speech recognizer and after that make and request to
speech recognizer
func requestSpeechAuth()
{
    SFSpeechRecognizer.requestAuthorization { authStatus in
        if authStatus == SFSpeechRecognizerAuthorizationStatus.authorized {
            if let path = Bundle.main.url(forResource: "mpthreetest", withExtension: "m4a") {
                do {
                    let sound = try AVAudioPlayer(contentsOf: path)
                    self.audioPlayer = sound
                    self.audioPlayer.delegate = self
                    sound.play()
                }   catch {
                    print("error")
                }

                let recognizer = SFSpeechRecognizer()
                let request = SFSpeechURLRecognitionRequest(url:path)
                recognizer?.recognitionTask(with: request) { (result, error) in
                    if let error = error {
                    print("there is a error\(error)")
                    } else {
// here you are printing out the audio output basically showing it on uitext field
                        self.transcriptionTextField.text =
result?.bestTranscription.formattedString
                    }
                }
            }
        }
    }
```

```
}

// here you are calling requestSpeechAuth function on UIButton press
@IBAction func playButtonPress(_ sender: AnyObject)
{
    requestSpeechAuth()
}
```

Прочитайте API распознавания речи IOS 10 онлайн:
https://riptutorial.com/ru/ios/topic/5986/api-распознавания-речи-ios-10

глава 7: AppDelegate

Вступление

AppDelegate - это протокол, который определяет методы, которые вызывается объектом single-ящика UIApplication в ответ на важные события за всю жизнь приложения.

Обычно используется для выполнения задач при запуске приложения (настройка среды приложения, analitycs (например: Mixpanel / GoogleAnalytics / Crashlitics), стек DB и т. Д.) И завершение работы (например, сохранение контекста БД), обработка запросов URL-адресов и аналогичных приложений задачи.

Examples

Все приложения приложения через методы AppDelegate

Для получения обновлений или для чего-либо, прежде чем приложение будет доступно пользователю, вы можете использовать метод ниже.

AppDidFinishLaunching

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {
    // Write your code before app launch
    return YES;
}
```

Пока приложение Enter на переднем плане:

```
- (void)applicationWillEnterForeground:(UIApplication *)application {
    // Called as part of the transition from the background to the active state; here you can
undo many of the changes made on entering the background.
}
```

Когда приложение запускается, а также фоном для Foreground попадает ниже метода:

```
- (void)applicationDidBecomeActive:(UIApplication *)application {
    // Restart any tasks that were paused (or not yet started) while the application was
inactive. If the application was previously in the background, optionally refresh the user
interface.
}
```

Пока приложение Enter в фоновом режиме:

```
- (void)applicationDidEnterBackground:(UIApplication *)application {
```

```
    // Use this method to release shared resources, save user data, invalidate timers, and
store enough application state information to restore your application to its current state in
case it is terminated later.
    // If your application supports background execution, this method is called instead of
applicationWillTerminate: when the user quits.
}
```

Пока приложение отменяет действие

```
- (void)applicationWillResignActive:(UIApplication *)application {
    // Sent when the application is about to move from active to inactive state. This can
occur for certain types of temporary interruptions (such as an incoming phone call or SMS
message) or when the user quits the application and it begins the transition to the background
state.
    // Use this method to pause ongoing tasks, disable timers, and invalidate graphics
rendering callbacks. Games should use this method to pause the game.
}
```

Пока приложение завершает работу:

```
- (void)applicationWillTerminate:(UIApplication *)application {
    // Called when the application is about to terminate. Save data if appropriate. See also
applicationDidEnterBackground:.
}
```

Роли AppDelegate:

- AppDelegate содержит `startup code` вашего приложения.
- Он реагирует на `key changes state` вашего приложения. В частности, он реагирует как на временные прерывания, так и на изменения состояния выполнения вашего приложения, например, когда ваше приложение переходит с переднего плана на задний план.
- Он `responds to notifications` исходящие извне приложения, такие как удаленные уведомления (также известные как push-уведомления), предупреждения с низкой памятью, уведомления о завершении загрузки и т. Д.
- Он `determines` , должно ли `state preservation` и `restoration` а также помогать в процессе сохранения и восстановления по мере необходимости.
- Он `responds to events` которые нацелены на приложение, и не являются специфическими для ваших приложений или контроллеров представлений. Вы можете использовать его для хранения центральных объектов данных вашего приложения или любого контента, на котором отсутствует собственный контроллер представления.

Открытие ресурса с URL-адресами

Предлагает делегату открыть ресурс, указанный URL-адресом, и предоставляет словарь параметров запуска.

Пример использования:

```
func application(_ app: UIApplication, open url: URL, options: [UIApplicationOpenURLOptionsKey
: Any] = [:]) -> Bool {
        return SomeManager.shared.handle(
            url,
            sourceApplication: options[.sourceApplication] as? String,
            annotation: options[.annotation]
        )
    }
```

Обработка локальных и удаленных уведомлений

Пример использования:

```
/* Instance of your custom APNs/local notification manager */
private var pushManager: AppleNotificationManager!
```

Постановка на учет:

```
func application(application: UIApplication, didRegisterUserNotificationSettings
notificationSettings: UIUserNotificationSettings) {
    // Called to tell the delegate the types of notifications that can be used to get the
user's attention
    pushManager.didRegisterSettings(notificationSettings)
}

func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    // Tells the delegate that the app successfully registered with Apple Push Notification
service (APNs)
    pushManager.didRegisterDeviceToken(deviceToken)
}

func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    // Sent to the delegate when Apple Push Notification service cannot successfully complete
the registration process.
    pushManager.didFailToRegisterDeviceToken(error)
}
```

Удаленная обработка уведомлений:

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject
: AnyObject]) {
    // Remote notification arrived, there is data to be fetched
    // Handling it
    pushManager.handleNotification(userInfo,
                                   background: application.applicationState == .Background
    )
}
```

Обработка локальных уведомлений:

```
func application(application: UIApplication, didReceiveLocalNotification notification:
UILocalNotification) {
    pushManager.handleLocalNotification(notification, background: false)
}
```

Действия по обработке (устаревшие):

```
func application(application: UIApplication, handleActionWithIdentifier identifier: String?,
forRemoteNotification userInfo: [NSObject : AnyObject],
                     completionHandler: () -> Void) {
    pushManager.handleInteractiveRemoteNotification(userInfo, actionIdentifier: identifier,
completion: completionHandler)
}
```

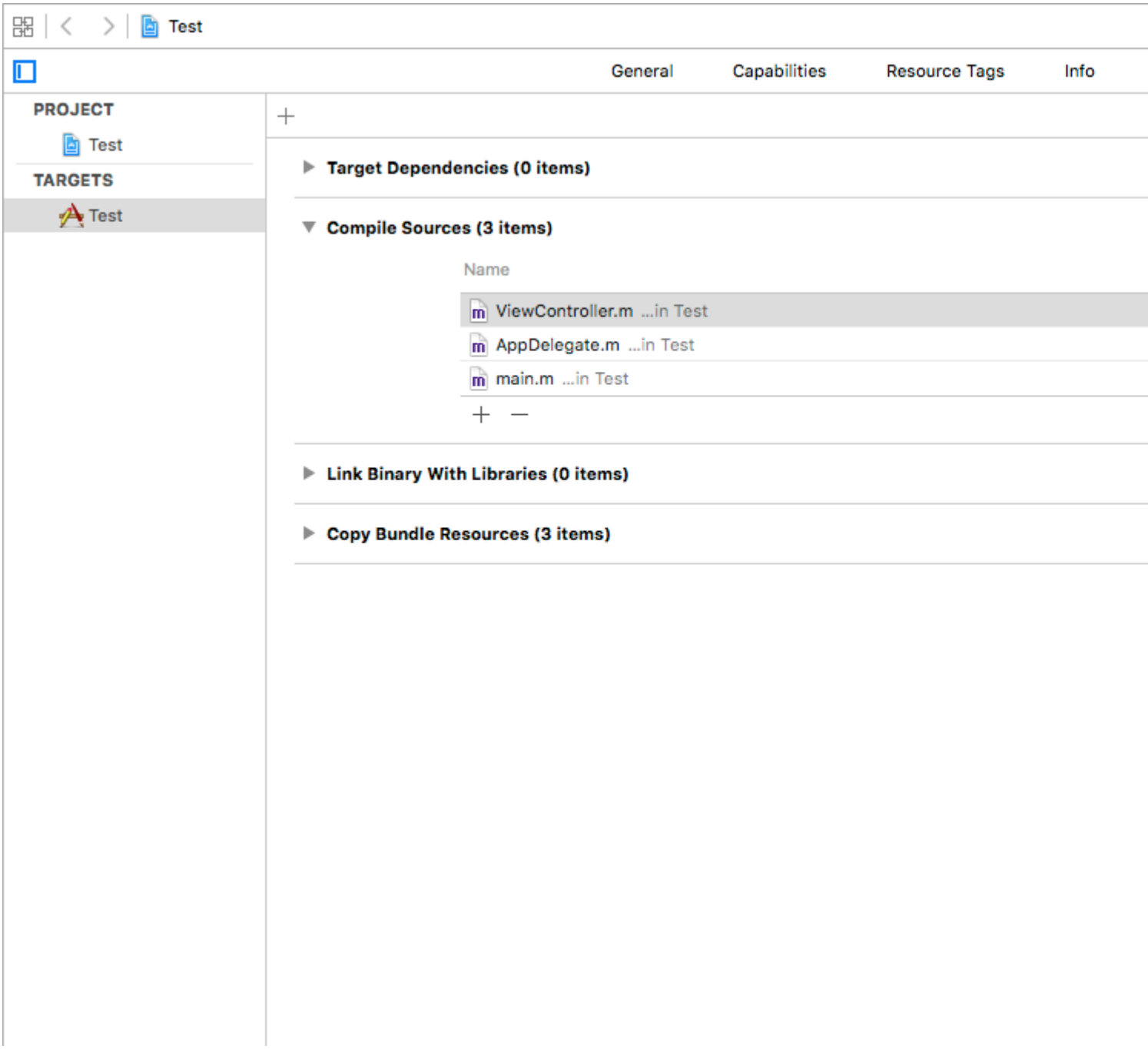
Прочитайте AppDelegate онлайн: https://riptutorial.com/ru/ios/topic/8740/appdelegate

глава 8: ARC (автоматический подсчет ссылок)

Examples

Включить / отключить ARC в файле

ARC может быть отключен для отдельных файлов, добавив флаг компилятора `-fno-objc-arc` для каждого файла. И наоборот, его можно добавить в « *Цели»* ▸ «Сформировать фазы» ▸ «Скомпилировать источники»



Прочитайте ARC (автоматический подсчет ссылок) онлайн:
https://riptutorial.com/ru/ios/topic/4150/arc--автоматический-подсчет-ссылок-

глава 9: attribitedText в UILabel

Вступление

Текущий стилизованный текст, отображаемый меткой.

Вы можете добавить текст `HTML` в `UILabel` используя свойство attributedText или индивидуальный текст `UILabel` с другим свойством

Examples

HTML-текст в UILabel

```
NSString * htmlString = @"<html><body> <b> Example bold text in HTML </b> </body></html>";
NSAttributedString * attrStr = [[NSAttributedString alloc] initWithData:[htmlString
dataUsingEncoding:NSUnicodeStringEncoding] options:@{ NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType } documentAttributes:nil error:nil];

UILabel * yourLabel = [[UILabel alloc] init];
yourLabel.attributedText = attrStr;
```

Установить другое свойство в текст в одном UILabel

Первым шагом, который необходимо выполнить, является создание объекта `NSMutableAttributedString` . Причина, по которой мы создаем `NSMutableAttributedString` вместо `NSAttributedString` заключается в том, что она позволяет нам добавлять к ней строку.

```
NSString *fullStr = @"Hello World!";
NSMutableAttributedString *attString =[[NSMutableAttributedString
alloc]initWithString:fullStr];

// Finding the range of text.
NSRange rangeHello = [fullStr rangeOfString:@"Hello"];
NSRange rangeWorld = [fullStr rangeOfString:@"World!"];

// Add font style for Hello
[attString addAttribute: NSFontAttributeName
                  value: [UIFont fontWithName:@"Copperplate" size:14]
                  range: rangeHello];
// Add text color for Hello
[attString addAttribute: NSForegroundColorAttributeName
                  value: [UIColor blueColor]
                  range: rangeHello];

// Add font style for World!
[attString addAttribute: NSFontAttributeName
                  value: [UIFont fontWithName:@"Chalkduster" size:20]
                  range: rangeWorld];
// Add text color for World!
[attString addAttribute: NSForegroundColorAttributeName
```

```
                  value: [UIColor colorWithRed:(66.0/255.0) green:(244.0/255.0)
blue:(197.0/255.0) alpha:1]
                  range: rangeWorld];

// Set it to UILabel as attributedText
UILabel * yourLabel = [[UILabel alloc] initWithFrame:CGRectMake(10, 150, 200, 100)];
yourLabel.attributedText = attString;
[self.view addSubview:yourLabel];
```

Выход :

Прочитайте attribitedText в UILabel онлайн:
https://riptutorial.com/ru/ios/topic/10927/attribitedtext-в-uilabel

глава 10: AVPlayer и AVPlayerViewController

замечания

импорт AVKit, импорт AVFoundation.

Examples

Воспроизведение мультимедиа с помощью AVPlayerViewController

Objective-C

```
NSURL *url = [[NSURL alloc] initWithString:@"YOUR URL"]; // url can be remote or local

AVPlayer *player = [AVPlayer playerWithURL:url];
// create a player view controller

AVPlayerViewController *controller = [[AVPlayerViewController alloc] init];
[self presentViewController:controller animated:YES completion:nil];
controller.player = player;
[player play];
```

стриж

```
let player = AVPlayer(URL: url) // url can be remote or local

let playerViewController = AVPlayerViewController()
// creating a player view controller
playerViewController.player = player
self.presentViewController(playerViewController, animated: true) {

    playerViewController.player!.play()
}
```

Воспроизведение медиа с помощью AVPlayer и AVPlayerLayer

Цель C

```
NSURL *url = [NSURL URLWithString:@"YOUR URL"];
AVPlayer *player = [AVPlayer playerWithURL:videoURL];
AVPlayerLayer *playerLayer = [AVPlayerLayer playerLayerWithPlayer:player];
playerLayer.frame = self.view.bounds;
[self.view.layer addSublayer:playerLayer];
[player play];
```

стриж

```
let url = NSURL(string: "YOUR URL")
let player = AVPlayer(URL: videoURL!)
let playerLayer = AVPlayerLayer(player: player)
playerLayer.frame = self.view.bounds
self.view.layer.addSublayer(playerLayer)
player.play()
```

Пример AVPlayer

AVPlayer * avPlayer = [AVPlayer playerWithURL: [NSURL URLWithString: @ "YOUR URL"]];

```
    AVPlayerViewController *avPlayerCtrl = [[AVPlayerViewController alloc] init];
    avPlayerCtrl.view.frame = self.view.frame;
    avPlayerCtrl.player = avPlayer;
    avPlayerCtrl.delegate = self;
    [avPlayer play];
    [self presentViewController:avPlayerCtrl animated:YES completion:nil
```

Прочитайте AVPlayer и AVPlayerViewController онлайн:
https://riptutorial.com/ru/ios/topic/5092/avplayer-и-avplayerviewcontroller

глава 11: AVSpeechSynthesizer

Синтаксис

- AVSpeechSynthesizer () // Создает речевой синтезатор
- speaker.speakUtterance (речь) // Преобразует текст в речь

параметры

параметр	подробности
оратор	Объект AVSpeechSynthesizer
речь	Объект AVSpeechUtterance

Examples

Создание основного текста в речь

Используйте метод `speakUtterance: AVSpeechSynthesizer` для преобразования текста в речь. Вам необходимо передать объект `AVSpeechUtterance` этому методу, который содержит текст, который вы хотите произнести.

Цель C

```
AVSpeechSynthesizer *speaker = [[AVSpeechSynthesizer alloc] init];
AVSpeechUtterance *speech    = [AVSpeechUtterance speechUtteranceWithString:@"Hello World"];
[speaker speakUtterance:speech];
```

стриж

```
let speaker = AVSpeechSynthesizer()
let speech = AVSpeechUtterance(string: "Hello World")
speaker.speakUtterance(speech)
```

Прочитайте AVSpeechSynthesizer онлайн:
https://riptutorial.com/ru/ios/topic/1526/avspeechsynthesizer

глава 12: AWS SDK

Examples

Загрузите изображение или видео на S3 с помощью AWS SDK

Прежде чем начать с примера, я бы рекомендовал создать Singleton с членом класса делегата, чтобы вы могли использовать прецедент загрузки файла в фоновом режиме и позволить пользователю продолжать использовать ваше приложение, пока файлы загружаются, даже когда приложение это фон.

Начнем с того, что мы должны создать перечисление, представляющее конфигурацию S3:

```
enum S3Configuration : String
{
    case IDENTITY_POOL_ID   = "YourIdentityPoolId"
    case BUCKET_NAME        = "YourBucketName"
    case CALLBACK_KEY       = "YourCustomStringForCallBackWhenUploadingInTheBackground"
    case CONTENT_TYPE_IMAGE = "image/png"
    case CONTENT_TYPE_VIDEO = "video/mp4"
}
```

Теперь мы должны установить учетные данные при первом запуске приложения, поэтому мы должны установить их внутри `AppDelegate` по методу `didFinishLaunchingWithOptions` (обратите внимание, что вы должны установить свой регион в параметре `regionType`):

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool
{
  let credentialProvider = AWSCognitoCredentialsProvider(regionType: .EUWest1, identityPoolId:
S3Configuration.IDENTITY_POOL_ID.rawValue)
  let configuration = AWSServiceConfiguration(region: .EUWest1, credentialsProvider:
credentialProvider)
  AWSS3TransferUtility.registerS3TransferUtilityWithConfiguration(configuration, forKey:
S3Configuration.CALLBACK_KEY.rawValue)
}
```

Поскольку мы уже находимся в AppDelegate, мы должны реализовать обратный вызов фона, который обрабатывается с помощью SDK AWS:

```
func application(application: UIApplication, handleEventsForBackgroundURLSession identifier:
String, completionHandler: () -> Void)
{
    //  Will print the identifer you have set at the enum: .CALLBACK_KEY
    print("Identifier: " + identifier)
    //  Stores the completion handler.
    AWSS3TransferUtility.interceptApplication(application,
                                              handleEventsForBackgroundURLSession: identifier,
                                              completionHandler: completionHandler)
}
```

Теперь, когда пользователь переместит приложение на задний план, ваша загрузка продолжит фактическую загрузку.

Чтобы загрузить файл с помощью AWS SDK, нам придется записать файл на устройство и предоставить SDK реальный путь. Для примера предположим, что у нас есть UIImage (возможно, видео тоже ..), и мы напишем его в временную папку:

```
// Some image....
let image = UIImage()
let fileURL = NSURL(fileURLWithPath:
NSTemporaryDirectory()).URLByAppendingPathComponent(fileName)
let filePath = fileURL.path!
let imageData = UIImageJPEGRepresentation(image, 1.0)
imageData!.writeToFile(filePath, atomically: true)
```

FileURL и имя_файла будут использоваться для фактической загрузки позже.

Есть два закрытия, которые нам нужно будет определить, которые предоставляются AWS SDK,

1. `AWSS3TransferUtilityUploadCompletionHandlerBlock` - закрытие, которое уведомляет о завершении загрузки (или нет)
2. `AWSS3TransferUtilityUploadProgressBlock` - закрытие, которое уведомляет каждый отправленный байт

Если вы планируете использовать Singleton, вы должны определить эти типы в качестве членов класса. Реализация должна выглядеть так:

```
var completionHandler : AWSS3TransferUtilityUploadCompletionHandlerBlock? =
    { (task, error) -> Void in

        if ((error) != nil)
        {
          print("Upload failed")
        }
        else
        {
          print("File uploaded successfully")
        }
    }

var progressBlock : AWSS3TransferUtilityUploadProgressBlock? =
    { [unowned self] (task, bytesSent:Int64, totalBytesSent:Int64,
totalBytesExpectedToSend:Int64) -> Void in

     let progressInPercentage = Float(Double(totalBytesSent) /
Double(totalBytesExpectedToSend)) * 100
     print(progressInPercentage)
    }
```

ПРИМЕЧАНИЕ. Если вы используете Singleton, вы можете определить делегата, который будет отчитываться о достигнутом прогрессе или когда файл будет выполнен. Если вы не

используете Singleton, вы можете создать статический метод, который будет иметь соответствующие типы:

```
    static func uploadImageToS3(fileURL : NSURL,
                                fileName : String,
                progressFunctionUpdater : Float -> Void,
                            resultBlock : (NSError?) -> Void)
{
 //    Actual implementation .....
 //    ...
 //    ...
}
```

1. `progressFunctionUpdater` - отчитается о функции с прогрессом.
2. `resultBlock` - если вы возвращаете nil, тогда загрузка была еще успешной, вы отправляете объект ошибки

Дамы и господа, фактическая загрузка:

```
        let fileData = NSData(contentsOfFile: fileURL.relativePath!)

        let expression = AWSS3TransferUtilityUploadExpression()
        expression.uploadProgress = progressBlock

        let transferUtility =
AWSS3TransferUtility.S3TransferUtilityForKey(S3Configuration.CALLBACK_KEY.rawValue)

        transferUtility?.uploadData(fileData!,
            bucket: S3Configuration.BUCKET_NAME.rawValue,
            key: fileName,
            contentType: S3Configuration.CONTENT_TYPE_IMAGE.rawData,
            expression: expression,
            completionHander: completionHandler).continueWithBlock
            { (task : AWSTask) -> AnyObject? in

                if let error = task.error
                {
                    print(error)
                }
                if let exception = task.exception
                {
                    print("Exception: " + exception.description)
                }
                if let uploadTask = task.result as? AWSS3TransferUtilityUploadTask
                {
                    print("Upload started...")
                }

                return nil
        }
```

Счастливая загрузка S3 :)

Прочитайте AWS SDK онлайн: https://riptutorial.com/ru/ios/topic/4734/aws-sdk

глава 13: CAAnimation

замечания

CAAnimation - абстрактный класс анимации. Он обеспечивает базовую поддержку протоколов **CAMediaTiming** и **CAAction** . Для анимации слоев основной анимации или объектов набора сцен создайте экземпляры конкретных подклассов **CABasicAnimation** , **CAKeyframeAnimation** , **CAAnimationGroup** или **CATransition** .

Examples

Анимируйте представление из одной позиции в другую.

Objective-C

```
CABasicAnimation *animation = [CABasicAnimation animationWithKeyPath:@"position.x"];
animation.fromValue = @0;
animation.toValue = @320;
animation.duration = 1;

[_label.layer addAnimation:animation forKey:@"basic"];
```

стриж

```
let animation = CABasicAnimation(keyPath: "position.x")
animation.fromValue = NSNumber(value: 0.0)
animation.toValue = NSNumber(value: 320.0)

_label.layer.addAnimation(animation, forKey: "basic")
```

Вид будет перемещаться от 0 до 320 по горизонтали. если вы хотите переместить вид на вертикально, просто замените keypath следующим образом:

```
"position.y"
```

Просмотр анимации - Toss

Objective-C

```
CATransition* transition = [CATransition animation];
transition.startProgress = 0;
transition.endProgress = 1.0;
```

```
transition.type = @"flip";
transition.subtype = @"fromLeft";
transition.duration = 0.8;
transition.repeatCount = 5;
[_label.layer addAnimation:transition forKey:@"transition"];
```

СВИФТ

```
var transition = CATransition()
transition.startProgress = 0
transition.endProgress = 1.0
transition.type = "flip"
transition.subtype = "fromLeft"
transition.duration = 0.8
transition.repeatCount = 5
label.layer.addAnimation(transition, forKey: "transition")
```

Revolve View

```
 CGRect boundingRect = CGRectMake(-150, -150, 300, 300);

CAKeyframeAnimation *orbit = [CAKeyframeAnimation animation];
orbit.keyPath = @"position";
orbit.path = CFAutorelease(CGPathCreateWithEllipseInRect(boundingRect, NULL));
orbit.duration = 4;
orbit.additive = YES;
orbit.repeatCount = HUGE_VALF;
orbit.calculationMode = kCAAnimationPaced;
orbit.rotationMode = kCAAnimationRotateAuto;

[_label.layer addAnimation:orbit forKey:@"orbit"];
```

Просмотр встряхивания

Objective-C

```
CAKeyframeAnimation *animation = [CAKeyframeAnimation animationWithKeyPath:@"position.x"];
animation.values = @[ @0, @10, @-10, @10, @0 ];
animation.keyTimes = @[ @0, @(1 / 6.0), @(3 / 6.0), @(5 / 6.0), @1 ];
animation.duration = 0.4;
animation.additive = YES;
[_label.layer addAnimation:animation forKey:@"shake"];
```

Swift 3

```
let animation = CAKeyframeAnimation(keyPath: "position.x")
animation.values = [ 0, 10, -10, 10, 0 ]
animation.keyTimes = [ 0, NSNumber(value: (1 / 6.0)), NSNumber(value: (3 / 6.0)),
NSNumber(value: (5 / 6.0)), 1 ]
animation.duration = 0.4
animation.isAdditive = true
label.layer.add(animation, forKey: "shake")
```

Анимация Push View

Цель C

```
CATransition *animation = [CATransition animation];
[animation setSubtype:kCATransitionFromRight];//kCATransitionFromLeft
[animation setDuration:0.5];
[animation setType:kCATransitionPush];
[animation setTimingFunction:[CAMediaTimingFunction
functionWithName:kCAMediaTimingFunctionEaseInEaseOut]];
[[yourView layer] addAnimation:animation forKey:@"SwitchToView1"];
```

стриж

```
let animation = CATransition()
animation.subtype = kCATransitionFromRight//kCATransitionFromLeft
animation.duration = 0.5
animation.type = kCATransitionPush
animation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionEaseInEaseOut)
yourView.layer.addAnimation(animation, forKey: "SwitchToView1")
```

Прочитайте CAAnimation онлайн: https://riptutorial.com/ru/ios/topic/981/caanimation

глава 14: CAGradientLayer

Синтаксис

- CAGradientLayer () // Возвращает инициализированный объект CALayer.
- CAGradientLayer (layer: layer) // Переопределить для копирования или инициализации настраиваемых полей указанного слоя.

параметры

параметр	подробности
цвет	Массив объектов `CGColorRef` определяющих цвет каждого градиента. Animatable.
места	Необязательный массив объектов `NSNumber` определяющий местоположение каждого градиентного стопа. Animatable.
ENDPOINT	Конечная точка градиента при рисовании в координатном пространстве слоя. Animatable.
точка отсчета	Начальная точка градиента при рисовании в координатном пространстве слоя. Animatable.
тип	Стиль градиента, нарисованный слоем. По умолчанию используется `kCAGradientLayerAxial` .

замечания

- Используйте `startPoint` и `endPoint` для изменения ориентации `CAGradientLayer` .
- Используйте `locations` чтобы повлиять на распределение / расположение цветов.

Examples

Создание CAGradientLayer

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
```

```
gradientLayer.frame = view.bounds

// Color at the top of the gradient.
let topColor: CGColor = UIColor.red.cgColor

// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.yellow.cgColor

// Set colors.
gradientLayer.colors = [topColor, bottomColor]

// Set locations of the colors.
gradientLayer.locations = [0.0, 1.0]

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, at: 0)
```

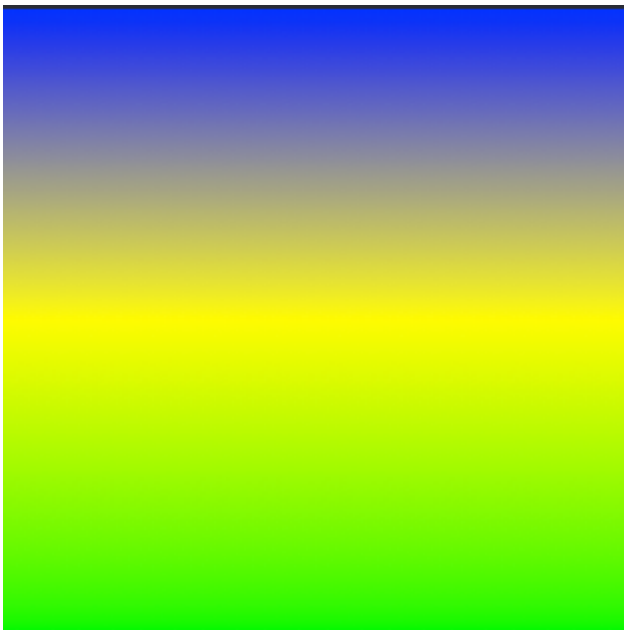
Результат:

Создание CGGradientLayer с несколькими цветами.

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds

// Color at the top of the gradient.
let topColor: CGColor = UIColor.blue.cgColor

// Color at the middle of the gradient.
let middleColor: CGColor = UIColor.yellow.cgColor

// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.green.cgColor
```

```
// Set colors.
gradientLayer.colors = [topColor, middleColor, bottomColor]

// Set locations of the colors.
gradientLayer.locations = [0.0, 0.5, 1.0]

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, at: 0)
```

Результат:

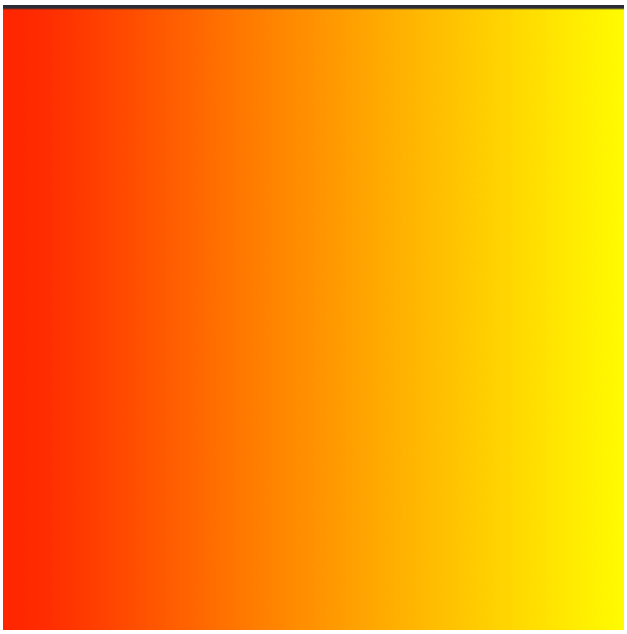


Создание горизонтального CAGradientLayer.

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds

// Color at the top of the gradient.
let topColor: CGColor = UIColor.redColor().CGColor

// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.yellowColor().CGColor

// Set colors.
gradientLayer.colors = [topColor, bottomColor]

// Set start point.
gradientLayer.startPoint = CGPoint(x: 0.0, y: 0.5)

// Set end point.
gradientLayer.endPoint = CGPoint(x: 1.0, y: 0.5)
```

```
// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, atIndex: 0)
```

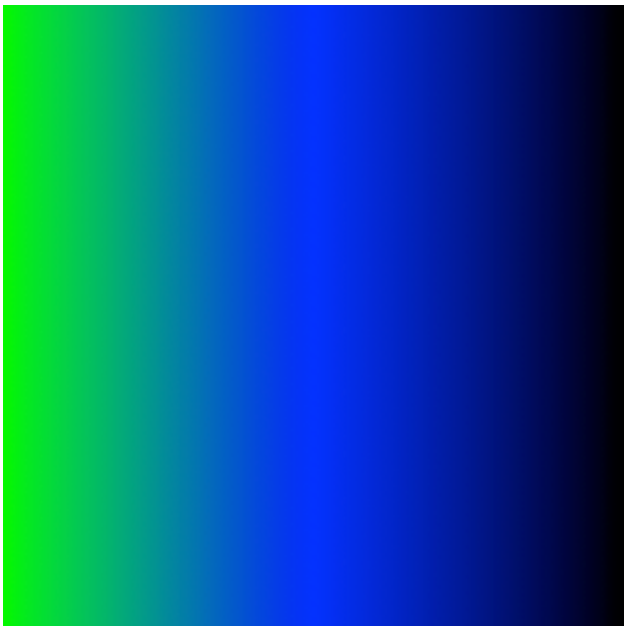
Результат:



Создание горизонтального CAGradientLayer с несколькими цветами.

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds

// Color at the top of the gradient.
let topColor: CGColor = UIColor.greenColor().CGColor

// Color at the middle of the gradient.
let middleColor: CGColor = UIColor.blueColor().CGColor

// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.blackColor().CGColor

// Set colors.
gradientLayer.colors = [topColor, middleColor, bottomColor]

// Set start point.
gradientLayer.startPoint = CGPoint(x: 0.0, y: 0.5)

// Set end point.
gradientLayer.endPoint = CGPoint(x: 1.0, y: 0.5)

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, atIndex: 0)
```

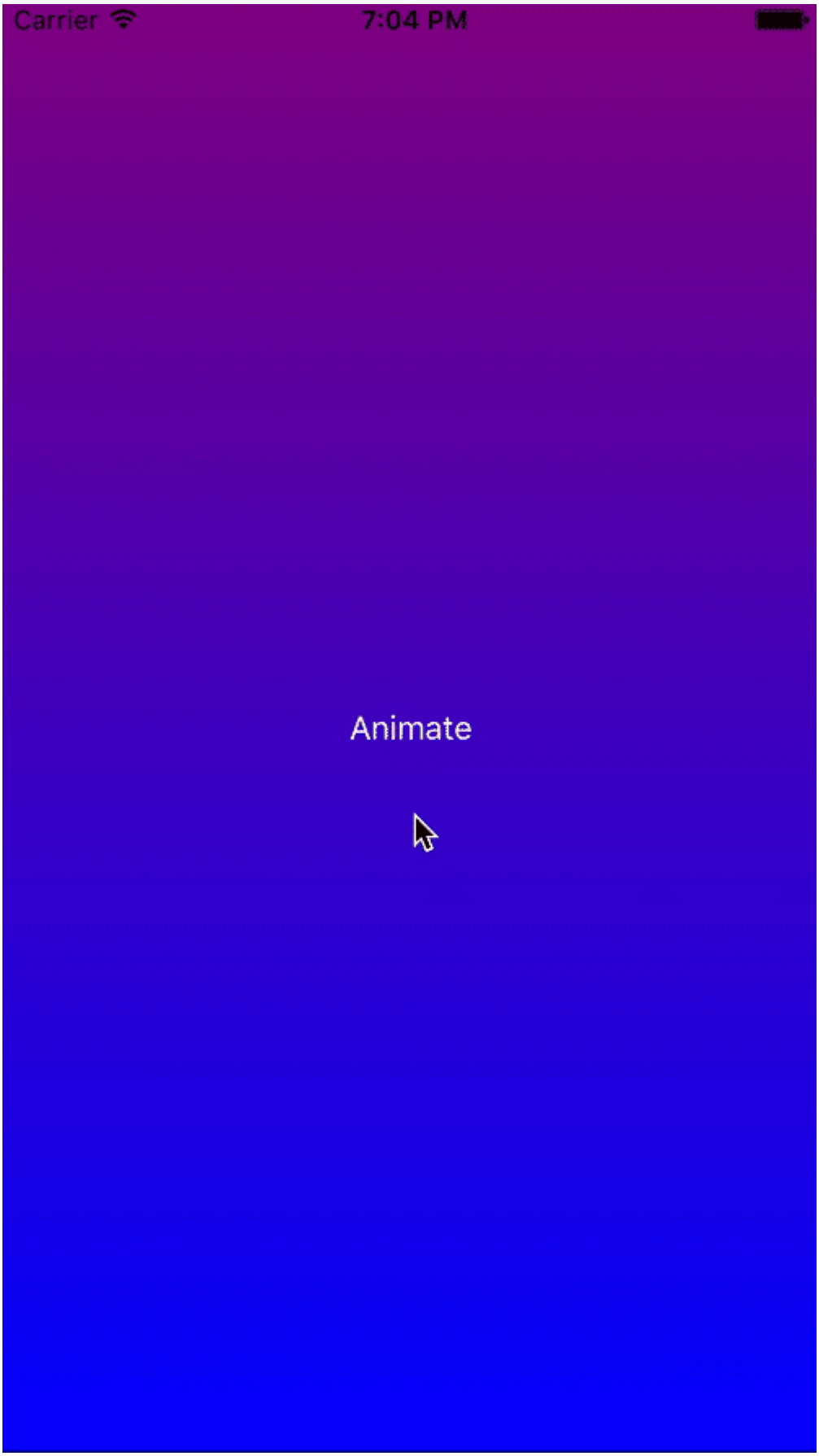
Результат:



Анимация изменения цвета в CAGradientLayer.

```
// Get the current colors of the gradient.
let oldColors = self.gradientLayer.colors

// Define the new colors for the gradient.
let newColors = [UIColor.red.cgColor, UIColor.yellow.cgColor]

// Set the new colors of the gradient.
self.gradientLayer.colors = newColors

// Initialize new animation for changing the colors of the gradient.
let animation: CABasicAnimation = CABasicAnimation(keyPath: "colors")

// Set current color value.
animation.fromValue = oldColors

// Set new color value.
animation.toValue = newColors

// Set duration of animation.
animation.duration = 0.3

// Set animation to remove once its completed.
animation.isRemovedOnCompletion = true

// Set receiver to remain visible in its final state when the animation is completed.
animation.fillMode = kCAFillModeForwards

// Set linear pacing, which causes an animation to occur evenly over its duration.
animation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionLinear)

// Set delegate of animation.
animation.delegate = self

// Add the animation.
self.gradientLayer.addAnimation(animation, forKey: "animateGradientColorChange")
```

Результат:



Прочитайте CAGradientLayer онлайн: https://riptutorial.com/ru/ios/topic/1190/cagradientlayer

глава 15: CALayer

Examples

Создание CALayer

Вы можете создать CALayer и установить его фрейм следующим образом:

Swift:

```
let layer = CALayer()
layer.frame = CGRect(x: 0, y: 0, width: 60, height: 80)
```

Objective-C:

```
CALayer *layer = [[CALayer alloc] init];
layer.frame = CGRectMake(0, 0, 60, 80);
```

Затем вы можете добавить его в качестве подуровня для существующего CALayer:

Swift:

```
existingLayer.addSublayer(layer)
```

Objective-C:

```
[existingLayer addSublayer:layer];
```

Замечания:

Для этого вам необходимо включить структуру QuartzCore.

Swift:

```
 @import QuartzCore
```

Objective-C

```
#import <QuartzCore/QuartzCore.h>
```

Создание частиц с помощью CAEmitterLayer

Класс **CAEmitterLayer** предоставляет систему эмиттеров частиц для Core Animation. Частицы определяются экземплярами **CAEmitterCell** .

Частицы рисуются над цветом фона и границей слоя.

```
        var emitter = CAEmitterLayer()

        emitter.emitterPosition = CGPoint(x: frame.size.width / 2.0, y: -20)
        emitter.emitterShape = kCAEmitterLayerLine
        emitter.emitterSize = CGSize(width: frame.size.width, height: 1)

        emitter.emitterCells = cells
        layer.addSublayer(emitter)
```

Просмотр эмиттера с пользовательским изображением

Например, мы создадим представление, которое содержит слой эмиттера и анимирует частицы.

```
import QuartzCore

class ConfettiView: UIView {
    // main emitter layer
    var emitter: CAEmitterLayer!

    // array of color to emit
    var colors: [UIColor]!

    // intensity of appearance
    var intensity: Float!

    private var active :Bool!

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setup()
    }

    override init(frame: CGRect) {
        super.init(frame: frame)
        setup()
    }

    func setup() {
        // initialization
        colors = [UIColor.redColor(),
                  UIColor.greenColor(),
                  UIColor.blueColor()
                  ]
        intensity = 0.2

        active = false
    }

    func startConfetti() {
        emitter = CAEmitterLayer()

        emitter.emitterPosition = CGPoint(x: frame.size.width / 2.0, y: -20)
        emitter.emitterShape = kCAEmitterLayerLine
        emitter.emitterSize = CGSize(width: frame.size.width, height: 1)
```

```
        var cells = [CAEmitterCell]()
        for color in colors {
            cells.append(confettiWithColor(color))
        }

        emitter.emitterCells = cells
        layer.addSublayer(emitter)
        active = true
    }

    func stopConfetti() {
        emitter?.birthRate = 0
        active = false
    }

    func confettiWithColor(color: UIColor) -> CAEmitterCell {
        let confetti = CAEmitterCell()

        confetti.birthRate = 10.0 * intensity
        confetti.lifetime = 180.0 * intensity
        confetti.lifetimeRange = 0
        confetti.color = color.CGColor
        confetti.velocity = CGFloat(350.0 * intensity)
        confetti.velocityRange = CGFloat(40.0 * intensity)
        confetti.emissionLongitude = CGFloat(M_PI)
        confetti.emissionRange = CGFloat(M_PI_4)
        confetti.spin = CGFloat(3.5 * intensity)
        confetti.spinRange = CGFloat(4.0 * intensity)

        // WARNING: A layer can set this property to a CGImageRef to display the image as its
contents.
        confetti.contents = UIImage(named: "confetti")?.CGImage
        return confetti
    }

    internal func isActive() -> Bool {
        return self.active
    }
}
```

Вам нужно добавить изображение «confetti» или указать rect с **confetti.contentsRect**

Как добавить UIImage в CALayer

Вы можете добавить изображение на `layer` вида, просто используя его свойство `contents` :

```
myView.layer.contents = UIImage(named: "star")?.CGImage
```

- Обратите внимание, что `UIImage` необходимо преобразовать в `CGImage` .

Если вы хотите добавить изображение в свой собственный слой, вы можете сделать это следующим образом:

```
let myLayer = CALayer()
let myImage = UIImage(named: "star")?.CGImage
myLayer.frame = myView.bounds
```

```
myLayer.contents = myImage
myView.layer.addSublayer(myLayer)
```

Изменение внешнего вида

Вышеприведенный код создает такое же представление. Светло-голубой - `UIView` а синяя звезда - `UIImage` .

Однако, как вы видите, он выглядит неровным. Это связано с тем, что `UIImage` меньше, чем `UIView` поэтому он масштабируется, чтобы заполнить представление, которое по умолчанию не указано ничем другим.

Приведенные ниже примеры показывают вариации на слой `contentsGravity` собственности. Код выглядит так:

```
myView.layer.contents = UIImage(named: "star")?.CGImage
myView.layer.contentsGravity = kCAGravityTop
myView.layer.geometryFlipped = true
```

В iOS вы можете установить свойство `geometryFlipped` в значение `true` если вы делаете что-либо с верхней или нижней гравитацией, иначе это будет противоположно тому, что вы ожидаете. (Только гравитация переворачивается вертикально, а не рендеринг содержимого. Если у вас возникли проблемы с переворачиванием содержимого, см. Этот ответ переполнения стека).

Есть два `UIView` ниже примеры для каждого `contentsGravity` настройки, один вид больше , чем `UIImage` , а другой меньше. Таким образом, вы можете увидеть эффекты масштабирования и гравитации.

`kCAGravityResize`

Это значение по умолчанию.

kCAGravityResizeAspect

kCAGravityResizeAspectFill

kCAGravityCenter

kCAGravityTop

kCAGravityBottom

kCAGravityLeft

kCAGravityRight

kCAGravityTopLeft

kCAGravityTopRight

kCAGravityBottomLeft

kCAGravityBottomRight

связанные с

- Свойство режима содержимого вида
- Рисование `UIImage` в `drawRect` с помощью `CGContextDrawImage`
- CALayer Tutorial: Начало работы

Заметки

- Этот пример исходит из этого ответа на переполнение стека .

Добавление преобразований в CALayer (перевод, поворот, масштабирование)

основы

Существует несколько различных преобразований, которые вы можете сделать на слое, но основные из них

- переводить (перемещать)
- масштаб
- вращаться

Чтобы сделать преобразования на `CALayer` , вы устанавливаете свойство `transform` слоя в тип `CATransform3D` . Например, чтобы перевести слой, вы сделали бы что-то вроде этого:

```
myLayer.transform = CATransform3DMakeTranslation(20, 30, 0)
```

Слово `Make` используется в имени для создания исходного преобразования: CATransform3D **Make** Translation. Последующие преобразования, которые применяются, опускают `Make` . См., Например, эту ротацию, за которой следует перевод:

```
let rotation = CATransform3DMakeRotation(CGFloat(30.0 * M_PI / 180.0), 20, 20, 0)
myLayer.transform = CATransform3DTranslate(rotation, 20, 30, 0)
```

Теперь, когда у нас есть основа, как сделать преобразование, давайте посмотрим на некоторые примеры того, как делать каждый. Во-первых, я покажу, как я создаю проект, если вы хотите поиграть с ним.

Настроить

В следующих примерах я создал приложение Single View и добавил `UIView` с голубым фоном в раскадровку. Я подключил представление к контроллеру вида с помощью следующего кода:

```
import UIKit

class ViewController: UIViewController {

    var myLayer = CATextLayer()
    @IBOutlet weak var myView: UIView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // setup the sublayer
        addSubLayer()

        // do the transform
        transformExample()
    }

    func addSubLayer() {
        myLayer.frame = CGRect(x: 0, y: 0, width: 100, height: 40)
        myLayer.backgroundColor = UIColor.blueColor().CGColor
        myLayer.string = "Hello"
        myView.layer.addSublayer(myLayer)
    }

    //******** Replace this function with the examples below ********

    func transformExample() {

        // add transform code here ...


    }

}
```

Существует много разных типов `CALayer` , но я решил использовать `CATextLayer` чтобы преобразования были более четкими визуально.

Переведите

Преобразование трансляции перемещает слой. Основной синтаксис

```
CATransform3DMakeTranslation(tx: CGFloat, ty: CGFloat, tz: CGFloat)
```

где `tx` - изменение координат x, `ty` - изменение y, а `tz` - изменение z.

пример

В iOS начало координат находится в левом верхнем углу, поэтому, если мы хотим переместить слой 90 точек вправо и 50 точек вниз, мы сделаем следующее:

```
myLayer.transform = CATransform3DMakeTranslation(90, 50, 0)
```

Заметки

- Помните, что вы можете вставить это в метод `transformExample()` в код проекта выше.
- Поскольку мы просто будем иметь дело с двумя измерениями здесь, `tz` устанавливается в `0` .
- Красная линия на изображении выше идет от центра исходного местоположения к центру нового места. Это потому, что преобразования выполняются по отношению к опорной точке и точка привязки по умолчанию находится в центре слоя.

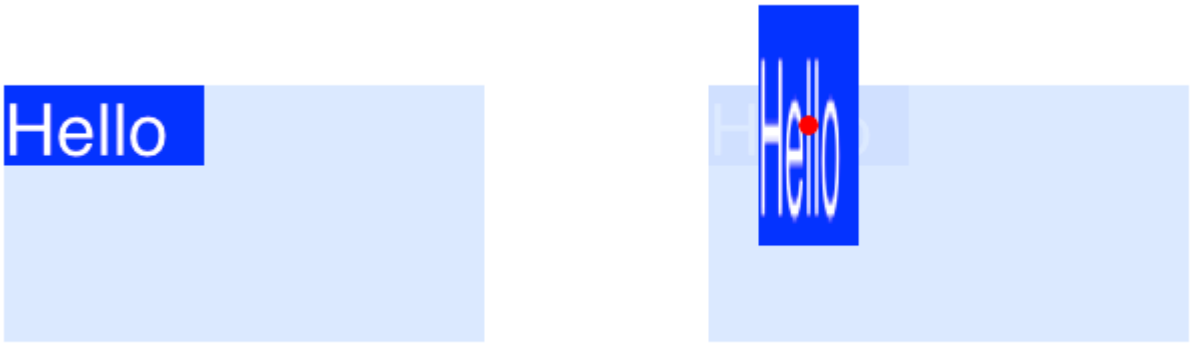
Масштаб

Масштабное преобразование растягивает или скручивает слой. Основной синтаксис

```
CATransform3DMakeScale(sx: CGFloat, sy: CGFloat, sz: CGFloat)
```

где `sx` , `sy` и `sz` - числа, с помощью которых можно масштабировать (умножать) координаты x, y и z соответственно.

пример



Если бы мы хотели половину ширины и утроить высоту, мы сделали бы следующее

```
myLayer.transform = CATransform3DMakeScale(0.5, 3.0, 1.0)
```

Заметки

- Поскольку мы работаем только в двух измерениях, мы просто умножаем координаты z на 1.0, чтобы они не были затронуты.
- Красная точка на изображении выше представляет собой опорную точку. Обратите внимание на то, как масштабирование делается по отношению к точке привязки. То есть, все, либо растянуто в стороне или от точки привязки.

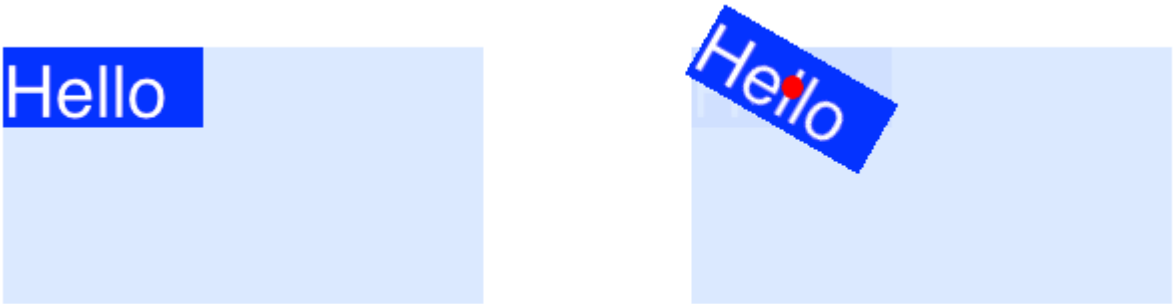
Поворот

Преобразование поворота поворачивает слой вокруг опорной точки (центр слоя по умолчанию). Основной синтаксис

```
CATransform3DMakeRotation(angle: CGFloat, x: CGFloat, y: CGFloat, z: CGFloat)
```

где `angle` - это угол в радианах, чтобы слой был повернут, а `x` , `y` и `z` - оси, вокруг которых можно вращать. Установка оси на 0 отменяет поворот вокруг этой конкретной оси.

пример



Если бы мы хотели повернуть слой по часовой стрелке на 30 градусов, мы сделали бы следующее:

```
let degrees = 30.0
let radians = CGFloat(degrees * M_PI / 180)
myLayer.transform = CATransform3DMakeRotation(radians, 0.0, 0.0, 1.0)
```

Заметки

- Поскольку мы работаем в двух измерениях, мы хотим, чтобы плоскость xy вращалась вокруг оси z. Таким образом, мы устанавливаем `x` и `y` `0.0` и устанавливаем `z` в `1.0` .
- Это повернуло слой по часовой стрелке. Мы могли бы вращаться против часовой стрелки, установив `z` на `-1.0` .
- Красная точка показывает, где находится опорная точка. Вращение осуществляется вокруг точки привязки.

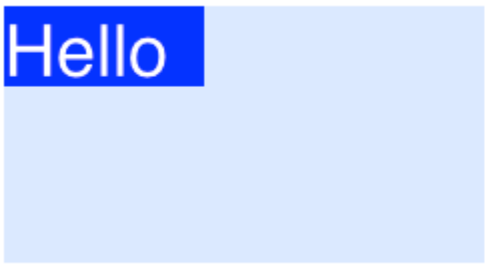
Несколько преобразований

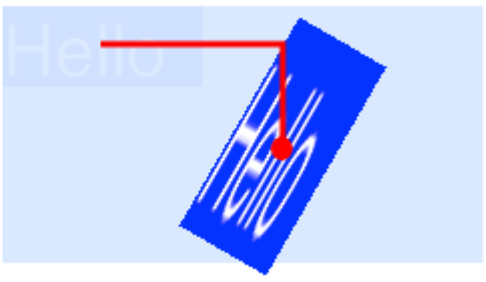
Чтобы объединить несколько преобразований, мы могли бы использовать

```
CATransform3DConcat(a: CATransform3D, b: CATransform3D)
```

Однако мы будем делать одно за другим. Первое преобразование будет использовать `Make` в своем имени. Следующие преобразования не будут использовать `Make` , но они будут принимать предыдущее преобразование в качестве параметра.

пример





На этот раз мы объединим все три предыдущих преобразования.

```
let degrees = 30.0
let radians = CGFloat(degrees * M_PI / 180)

// translate
var transform = CATransform3DMakeTranslation(90, 50, 0)

// rotate
transform = CATransform3DRotate(transform, radians, 0.0, 0.0, 1.0)
```

```
// scale
transform = CATransform3DScale(transform, 0.5, 3.0, 1.0)

// apply the transforms
myLayer.transform = transform
```

Заметки

- Порядок выполнения преобразований в вопросах.
- Все было сделано по отношению к опорной точке (красная точка).

Примечание о якорной точке и позиции

Мы выполнили все наши преобразования выше, не меняя опорную точку. Иногда это необходимо изменить, хотя, например, если вы хотите повернуть вокруг какой-то другой точки, кроме центра. Однако это может быть немного сложно.

Якорная точка и позиция находятся на одном и том же месте. Якорная точка выражается как единица системы координат слоя (по умолчанию `0.5, 0.5`), и позиция выражается в системе координат суперслоя. Они могут быть установлены следующим образом:

```
myLayer.anchorPoint = CGPoint(x: 0.0, y: 1.0)
myLayer.position = CGPoint(x: 50, y: 50)
```

Если вы устанавливаете точку привязки без изменения положения, тогда рамка изменяется так, что позиция будет в нужном месте. Или, точнее, рама пересчитываются на основе новой опорной точки и старой позиции. Это обычно дает неожиданные результаты. Следующие две статьи прекрасно обсуждают это.

- О якорной точке
- Перевести вращение переводом?

Смотрите также

- Граница, закругленные углы и тень на `CALayer`
- Использование границы с контуром Безье для слоя

Этот пример изначально исходит из этого примера переполнения стека .

Отключить анимацию

`CALayer` свойства свойств `CALayer` . Если это нежелательно, их можно отключить следующим образом.

стриж

```
CATransaction.begin()
CATransaction.setDisableActions(true)

// change layer properties that you don't want to animate

CATransaction.commit()
```

Objective-C

```
[CATransaction begin];
[CATransaction setDisableActions:YES];

// change layer properties that you don't want to animate

[CATransaction commit];
```

Закругленные углы

```
layer.masksToBounds = true;
layer.cornerRadius = 8;
```

Тени

Вы можете использовать 5 свойств на каждом уровне для настройки теней:

- `shadowOffset` - это свойство перемещает вашу тень влево / вправо или вверх / вниз

```
self.layer.shadowOffset = CGSizeMake(-1, -1); // 1px left and up

self.layer.shadowOffset = CGSizeMake(1, 1); // 1px down and right
```

- `shadowColor` - устанавливает цвет вашей тени

```
self.layer.shadowColor = [UIColor blackColor].CGColor;
```

- `shadowOpacity` - это непрозрачность тени, от `0` до `1`

```
self.layer.shadowOpacity = 0.2;
```

- `shadowRadius` - это радиус размытия (эквивалент свойства размытия в Sketch или Photoshop)

```
self.layer.shadowRadius = 6;
```

- `shadowPath` - это важное свойство производительности, когда unset iOS основывает тень на альфа-канале представления, который может быть интенсивным с

использованием сложного PNG с альфой. Это свойство позволяет вам форсировать форму для вашей тени и быть более совершенным из-за этого.

Objective-C

```
self.layer.shadowPath = [UIBezierPath bezierPathWithOvalInRect:CGRectMake(0,0,100,100)];
//this does a circular shadow
```

Swift 3

```
self.layer.shadowPath = UIBezierPath(ovalIn: CGRect(x: 0, y: 0, width: 100, height:
100)).cgPath
```

Прочитайте CALayer онлайн: https://riptutorial.com/ru/ios/topic/1462/calayer

глава 16: Carthage iOS Setup

Examples

Carthage Installation Mac

Настройка Карфагена

Загрузите последнюю версию Карфагена от данной ссылки Скачать Ссылку

В разделе «Загрузка» загрузите файл **Carthage.pkg** .

После завершения загрузки установите его двойным щелчком в файле загрузки pkg.

Чтобы проверить успешную загрузку, выполните следующую команду в версии вашего терминала. Это должно привести к тому, что версия будет установлена как `0.18-19-g743fa0f`

Прочитайте Carthage iOS Setup онлайн: https://riptutorial.com/ru/ios/topic/7404/carthage-ios-setup

глава 17: CAShapeLayer

Синтаксис

1. shapeLayer.fillColor
2. shapeLayer.fillRule
3. shapeLayer.lineCap
4. shapeLayer.lineDashPattern
5. shapeLayer.lineDashPhase
6. shapeLayer.lineJoin

замечания

Класс CAShapeLayer рисует кубический сплайн Безье в его координатном пространстве. Форма компонуется между содержимым слоя и его первым подслоем.

Examples

Базовая операция CAShapeLayer

Использование UIBezierPath для создания кругового пути ShapeLayer

```
CAShapeLayer *circleLayer = [CAShapeLayer layer];
[circleLayer setPath:[[UIBezierPath bezierPathWithOvalInRect:
CGRectMake(50, 50, 100, 100)] CGPath]];
circleLayer.lineWidth = 2.0;
[circleLayer setStrokeColor:[[UIColor redColor] CGColor]];
[circleLayer setFillColor:[[UIColor clearColor] CGColor]];
circleLayer.lineJoin = kCALineJoinRound;  //4 types are available to create a line style
circleLayer.lineDashPattern = [NSArray arrayWithObjects:
[NSNumber numberWithInt:2],[NSNumber numberWithInt:3 ], nil];
// self.origImage is parentView
[[self.view layer] addSublayer:circleLayer];
self.currentShapeLayer = circleLayer;  // public value using to keep that reference of the
shape Layer
self.view.layer.borderWidth = 1.0f;
self.view.layer.borderColor = [[UIColor blueColor]CGColor];  // that will plotted in the
mainview
```

Удалить ShapeLayer

Сохраните ссылку на этот слой. Например, у вас может быть свойство currentShapeLayer: теперь, когда у вас есть ссылка, вы можете легко удалить слой:

Тип 1:

```
[self.currentShapeLayer removeFromSuperlayer];
```

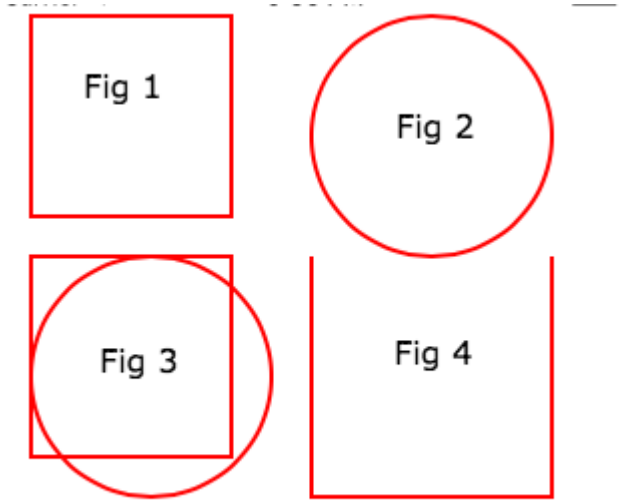
Тип 2:

```
 self.view.layer.sublayers = nil ;                //removed all earlier shapes
```

Другие операции

```
 //Draw Square Shape

CAShapeLayer *squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(20, 20, 100, 100);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = nil;
squareLayer.strokeColor = [[UIColor redColor] CGColor];
squareLayer.path = [UIBezierPath bezierPathWithRect:squareLayer.bounds].CGPath;
[[self.view layer] addSublayer:squareLayer];


//Draw Circle Shape

CAShapeLayer *circleShape = [CAShapeLayer layer];
circleShape.frame = CGRectMake(160, 20, 120, 120);
circleShape.lineWidth = 2.0;
circleShape.fillColor = nil;
circleShape.strokeColor = [[UIColor redColor] CGColor];
circleShape.path = [UIBezierPath bezierPathWithOvalInRect:circleShape.bounds].CGPath;
[[self.view layer] addSublayer:circleShape];


//Subpaths
//UIBezierPath can have any number of “path segments” (or subpaths) so you can effectively
draw as many shapes or lines as you want in a single path object

CAShapeLayer *shapeLayer = [CAShapeLayer layer];
shapeLayer.frame = CGRectMake(20, 140, 200, 200);
shapeLayer.lineWidth = 2.0;
shapeLayer.fillColor = nil;
shapeLayer.strokeColor = [[UIColor redColor] CGColor];

CGMutablePathRef combinedPath= CGPathCreateMutableCopy(circleShape.path);
CGPathAddPath(combinedPath, NULL, squareLayer.path);

shapeLayer.path = combinedPath;
[[self.view layer] addSublayer:shapeLayer];

//Open Path
// Paths do not need to connect their end points back to their starting points. A path that
connects back to its starting point is called a closed path, and one that does not is called
an open path.

shapeLayer = [CAShapeLayer layer];
shapeLayer.frame = CGRectMake(160, 140, 300, 300);
shapeLayer.lineWidth = 2.0;
shapeLayer.fillColor = nil;
shapeLayer.strokeColor = [[UIColor redColor] CGColor];

UIBezierPath *linePath=[UIBezierPath bezierPath];
[linePath moveToPoint:CGPointZero];
[linePath addLineToPoint:CGPointMake(0 , 120)];
[linePath addLineToPoint:CGPointMake(120 , 120)];
```

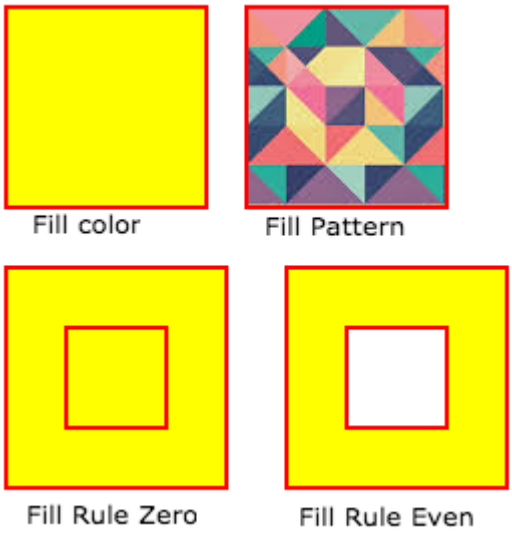
```
[linePath addLineToPoint:CGPointMake(120 , 0)];
shapeLayer.path = linePath.CGPath;
[[self.view layer] addSublayer:shapeLayer];
```



Заглавные буквы // Цвет заливки

```
CAShapeLayer *squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(20, 30, 100, 100);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor yellowColor]CGColor];
squareLayer.strokeColor = [[UIColor redColor] CGColor];
squareLayer.path = [UIBezierPath bezierPathWithRect:squareLayer.bounds].CGPath;
[[self.view layer] addSublayer:squareLayer];

//Fill Pattern Color
//images.jpeg

squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(140, 30, 100, 100);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor colorWithPatternImage:[UIImage
imageNamed:@"images.jpeg"]]CGColor];
squareLayer.strokeColor = [[UIColor redColor] CGColor];
squareLayer.path = [UIBezierPath bezierPathWithRect:squareLayer.bounds].CGPath;
[[self.view layer] addSublayer:squareLayer];


//Fill Rule

//Type 1: kCAFillRuleNonZero
squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(0, 140, 150, 150);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor yellowColor]CGColor];
squareLayer.fillRule = kCAFillRuleNonZero;   // indicate the rule type
squareLayer.strokeColor = [[UIColor redColor] CGColor];
UIBezierPath *outerPath = [UIBezierPath bezierPathWithRect:CGRectInset(squareLayer.bounds,
20.0, 20.0)];
UIBezierPath *innerPath = [UIBezierPath bezierPathWithRect:CGRectInset(squareLayer.bounds,
50.0, 50.0)];
CGMutablePathRef combinedPath= CGPathCreateMutableCopy(outerPath.CGPath);
```

```
CGPathAddPath(combinedPath, NULL, innerPath.CGPath);
squareLayer.path = combinedPath;
[[self.view layer] addSublayer:squareLayer];


//Type 2: kCAFillRuleEvenOdd
squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(140, 140, 150, 150);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor yellowColor]CGColor];
squareLayer.fillRule = kCAFillRuleEvenOdd;   // indicate the rule type
squareLayer.strokeColor = [[UIColor redColor] CGColor];
outerPath = [UIBezierPath bezierPathWithRect:CGRectInset(squareLayer.bounds, 20.0, 20.0)];
innerPath = [UIBezierPath bezierPathWithRect:CGRectInset(squareLayer.bounds, 50.0, 50.0)];
combinedPath= CGPathCreateMutableCopy(outerPath.CGPath);
CGPathAddPath(combinedPath, NULL, innerPath.CGPath);
squareLayer.path = combinedPath;
[[self.view layer] addSublayer:squareLayer];
```



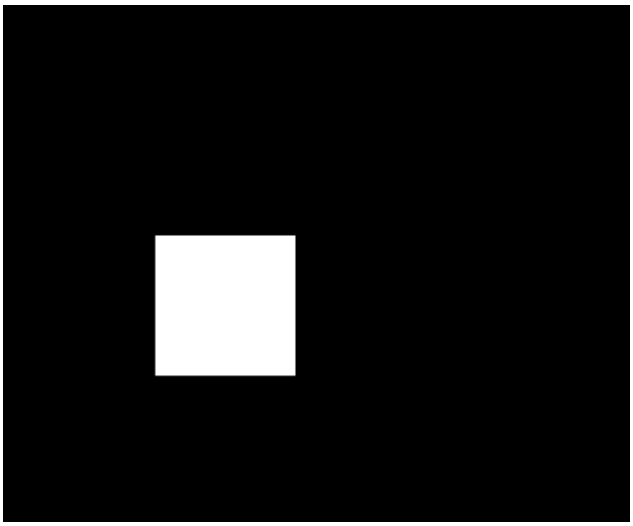
Перечислен список свойств стиля

```
   fillColor
        Fill the color based on the drawed shape.

   fillRule
        Fill Rule the there are two rule is applied to draw the shape.
        1.  kCAFillRuleNonZero
        2.  kCAFillRuleEvenOdd

   lineCap
        Below type used to change the style of the line.
        1.  kCALineCapButt
        2.  kCALineCapRound
        3.  kCALineCapSquare

   lineDashPattern
        The dash pattern applied to the shape’s path when stroked.
        Create DashStyle while you will stroke the line.

   lineDashPhase
        The dash phase applied to the shape’s path when stroked. Animatable.
```

```
   lineJoin
        Line join style for the shape path.Below style use to draw the line join style.
        1.  kCALineJoinMiter
        2.  kCALineJoinRound
        3.  kCALineJoinBevel

   lineWidth
        Which using to set the line width.

   miterLimit
        The miter limit used when stroking the shape’s path. Animatable.

   strokeColor
        Set the stroke color based on the path of the line.

   strokeStart
        When the stroke will start.

   strokeEnd
       When the stroke will end.
```

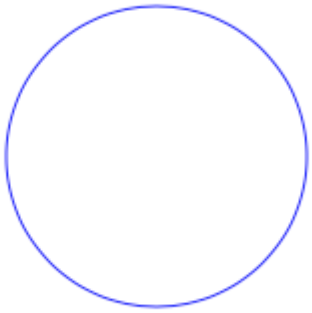
Прямой прямоугольник

```
 CAShapeLayer *mask = [[CAShapeLayer alloc] init];

mask.frame = CGRectMake(50, 50, 100, 100);

CGFloat width = 100;

CGFloat height = 100;

CGMutablePathRef path = CGPathCreateMutable();

CGPathMoveToPoint(path, nil, 30, 30);

CGPathAddLineToPoint(path, nil, width, 30);

CGPathAddLineToPoint(path, nil, width, height);

CGPathAddLineToPoint(path, nil,30, height);

CGPathAddLineToPoint(path, nil, 30, 30);

CGPathCloseSubpath(path);


mask.path = path;

CGPathRelease(path);

self.view.layer.mask = mask;
```



Кружок рисования

```
 CAShapeLayer *circle = [CAShapeLayer layer];

    [circle setPath:[[UIBezierPath bezierPathWithOvalInRect:CGRectMake(100, 100, 150, 150)]
CGPath]];

    [circle setStrokeColor:[[UIColor blueColor] CGColor]];

    [circle setFillColor:[[UIColor clearColor] CGColor]];

    [[self.view layer] addSublayer:circle];
```



CAShapeLayer Анимация

```
CAShapeLayer *circle = [CAShapeLayer layer];

[circle setPath:[[UIBezierPath bezierPathWithOvalInRect:CGRectMake(100, 100, 150, 150)]
CGPath]];

[circle setStrokeColor:[[UIColor blueColor] CGColor]];

[circle setFillColor:[[UIColor clearColor] CGColor]];

[[self.view layer] addSublayer:circle];
```

```
CABasicAnimation *pathAnimation = [CABasicAnimation animationWithKeyPath:@"strokeEnd"];

pathAnimation.duration = 1.5f;

pathAnimation.fromValue = [NSNumber numberWithFloat:0.0f];

pathAnimation.toValue = [NSNumber numberWithFloat:1.0f];

pathAnimation.repeatCount = 10;

pathAnimation.autoreverses = YES;

[circle addAnimation:pathAnimation
              forKey:@"strokeEnd"];
```

Прочитайте CAShapeLayer онлайн: https://riptutorial.com/ru/ios/topic/3575/cashapelayer

глава 18: CGContext Reference

замечания

Открытый непрозрачный тип **CGContextRef** представляет собой назначение 2D-чертежа Quartz. Графический контекст содержит параметры чертежа и всю информацию об устройстве, необходимую для визуализации краски на странице до места назначения, независимо от того, является ли это местом в приложении, растровым изображением, PDF-документом или принтером.

Examples

Строка рисования

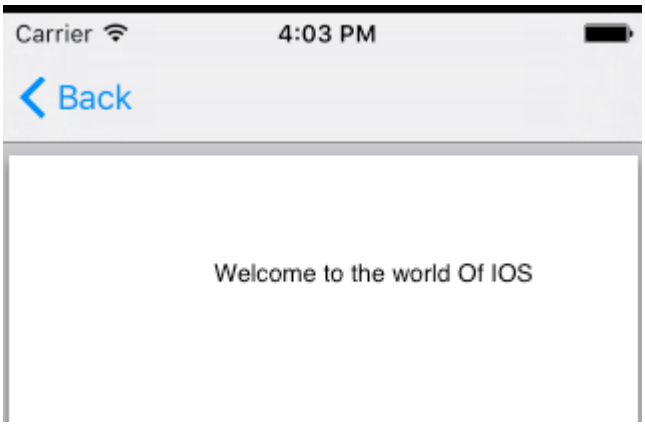
```
CGContextRef context = UIGraphicsGetCurrentContext();

CGContextSetLineWidth(context, 5.0);
CGColorSpaceRef colorspace = CGColorSpaceCreateDeviceRGB();
CGContextMoveToPoint(context, 200, 400);
CGContextAddLineToPoint(context, 100, 100);
CGContextStrokePath(context);
CGColorSpaceRelease(colorspace);
```

Рисовать текст

Draw To требует, чтобы структура **Core Text** была добавлена на этапе сборки

```
[NSString* textToDraw = @"Welcome to the world Of IOS";

    CFStringRef stringRef = (__bridge CFStringRef)textToDraw;

    CFAttributedStringRef currentText = CFAttributedStringCreate(NULL, stringRef, NULL);
    CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedString(currentText);
    CGRect frameRect = CGRectMake(0, 0, 300, 100);
    CGMutablePathRef framePath = CGPathCreateMutable();
    CGPathAddRect(framePath, NULL, frameRect);

    CFRange currentRange = CFRangeMake(0, 0);
```

```
    CTFrameRef frameRef = CTFramesetterCreateFrame(framesetter, currentRange, framePath,
NULL);
    CGPathRelease(framePath);
    CGContextRef currentContext = UIGraphicsGetCurrentContext();

    CGContextSetTextMatrix(currentContext, CGAffineTransformIdentity);
    CGContextTranslateCTM(currentContext, 200, 300);
    CGContextScaleCTM(currentContext, 2, -2);
    CTFrameDraw(frameRef, currentContext);

    CFRelease(frameRef);
    CFRelease(stringRef);
    CFRelease(framesetter);
```



Прочитайте CGContext Reference онлайн: https://riptutorial.com/ru/ios/topic/2664/cgcontext-reference

глава 19: CLLocation

Examples

Фильтр расстояний, используя

Пример :

```
   CLLocationManager *locationManager = [[CLLocationManager alloc] init];
   locationManager.delegate = self;
   locationManager.desiredAccuracy = kCLLocationAccuracyBest;
   locationManager.distanceFilter = 5;
```

Например, в приведенном выше примере кода изменения местоположения менее 5 метров не будут отправляться на обратный вызов, но вместо этого будут проигнорированы.

Получить местоположение пользователя с помощью CLLocationManager

1 - Включить CoreLocation.framework в ваш проект; это достигается нажатием на:

```
root directory -> build phases -> Link Binary With Libraries
```

Нажмите кнопку (+), найдите CoreLocation.framework и нажмите «Добавить».

2- Измените файл info.plist, чтобы получить разрешение на использование местоположения пользователя, открыв его в качестве исходного кода. Добавьте одну из следующих пара ключей: значение под тегом, чтобы спросить об использовании местоположения пользователя во время использования приложения:

```
<key>NSLocationWhenInUseUsageDescription</key>
<string>message to display when asking for permission</string>
```

3- импортировать CoreLocation в ViewController, который будет использовать его.

```
import CoreLocation
```

4- Убедитесь, что ваш ViewController соответствует протоколу CLLocationManagerDelagate

```
class ViewController: UIViewController,CLLocationManagerDelegate {}
```

После этих шагов мы можем создать объект CLLocationManager как переменную экземпляра и использовать его в ViewController.

```
var manager:CLLocationManager!
```

Мы не используем 'let' здесь, потому что мы будем изменять менеджера, чтобы указать его делегата, минимальное расстояние до события обновления и его точность

```
//initialize the manager
manager = CLLocationManager()

//specify delegate
manager.delegate = self

//set the minimum distance the phone needs to move before an update event is triggered (for
example:  100 meters)
manager.distanceFilter = 100

//set Accuracy to any of the following depending on your use case

//let kCLLocationAccuracyBestForNavigation: CLLocationAccuracy
//let kCLLocationAccuracyBest: CLLocationAccuracy
//let kCLLocationAccuracyNearestTenMeters: CLLocationAccuracy
//let kCLLocationAccuracyHundredMeters: CLLocationAccuracy
//let kCLLocationAccuracyKilometer: CLLocationAccuracy
//let kCLLocationAccuracyThreeKilometers: CLLocationAccuracy

manager.desiredAccuracy = kCLLocationAccuracyBest

//ask the user for permission
manager.requestWhenInUseAuthorization()

//Start collecting location information
if #available(iOS 9.0, *) {

   manager.requestLocation()

 } else {

   manager.startUpdatingLocation()

  }
```

Теперь, чтобы получить доступ к обновлениям местоположения, мы можем реализовать функцию, ниже которой называется overtime, что достигается расстояниеФильтр.

```
func locationManager(manager: CLLocationManager, didUpdateLocations locations: [CLLocation])
{}
```

Параметр местоположений представляет собой массив объектов CLLocation, которые представляют фактическое местоположение устройства. Из этих объектов можно получить доступ к следующим атрибутам: `coordinate,altitude, floor, horizontalAccuracy, verticalAccuracy, timestamp, description, course, speed` Точность, вертикальность. Точность `coordinate,altitude, floor, horizontalAccuracy, verticalAccuracy, timestamp, description, course, speed` и `distance(from:)` которое измеряет расстояние между двумя точками.

Примечание. При запросе разрешения на локацию существует два разных типа авторизации.

> «Когда используется» авторизация дает разрешение приложения получать

ваше местоположение только в том случае, если приложение используется или находится на переднем плане.

Разрешение «Всегда» дает разрешения на фоновое изображение приложения, что может привести к сокращению времени автономной работы, если ваше приложение закрыто.

Файл Plist следует отрегулировать по мере необходимости.

Прочитайте CLLocation онлайн: https://riptutorial.com/ru/ios/topic/2002/cllocation

глава 20: CloudKit

замечания

Поддерживаемые типы

- NSData
- NSDate (Дата)
- NSNumber (Int / Double)
- NSString (String)
- NSArray (массив)
- CLLocation
- CKReference
- CKAsset

Подробнее

Панель управления CloudKit

Examples

Регистрация приложения для использования с CloudKit

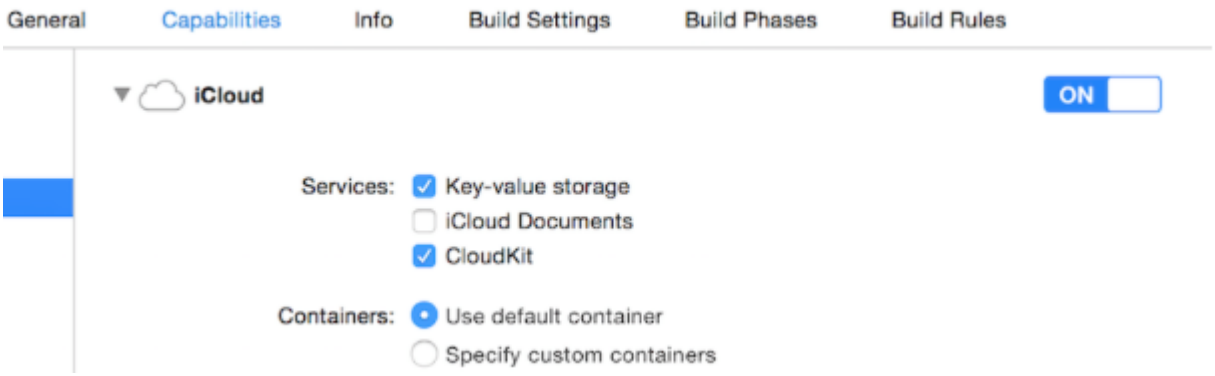
Вам нужно получить файл с правами, чтобы приложение могло получить доступ к iCloud и записывать записи с помощью CloudKit.

Выполните шаги, чтобы предоставить доступ к iCloud из вашего приложения:

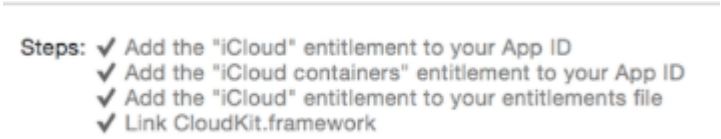
1- Выберите проект в Навигаторе проектов, а затем откройте вкладку «Общие».

2- В разделе «Идентификация» установите идентификатор Apple разработчика в раскрывающемся меню «Команда». (Если он недоступен, добавьте его в меню Xcode -> Настройки -> Учетные записи.

3- Перейдите на вкладку «Возможности» в свойствах проекта и включите iCloud. Затем выберите «Key-Value Storage» и «CloudKit».



4- Убедитесь, что эти элементы отмечены:



Если все элементы отмечены, то ваше приложение будет готово к использованию CloudKit.

Использование облачной панели CloudKit

Все записи, созданные с использованием кода CloudKit, могут быть просмотрены, отредактированы и даже удалены в CloudKit Dashboard. Чтобы открыть панель CloudKit, перейдите сюда .

На панели приборов есть несколько частей:

- Типы записей (которые будут рассмотрены ниже)
- Роли безопасности (где вы можете устанавливать базы данных как общедоступные или частные)
- Типы подписки (которые ваше приложение может регистрировать для уведомлений Apple Push (APN), чтобы уведомить вас, когда запись будет изменена)

Типы записей

Здесь вы получаете список всех существующих типов записей в приложении. Когда вы впервые открываете панель CloudKit Dashboard для приложения, есть тип записи «Пользователи», который вы можете использовать, или просто удалить его и использовать самостоятельно.

На этой странице вы можете вручную ввести данные. Конечно, в большинстве случаев это бессмысленно, потому что IOS SDK может обрабатывать его лучше, чем панель инструментов, но функциональность также существует, если вы предпочитаете. Наибольшее использование этой страницы для просмотра типов.

Сохранение данных в CloudKit

Чтобы сохранить дату в CloudKit, мы должны сделать:

- `CKRecordID` (ключ вашей уникальной записи)
- `CKRecord` (который включает данные)

Создание ключа записи

Чтобы каждый уникальный идентификатор записи был уникальным, мы используем текущую *временную метку* , которая уникальна. Мы получаем `NSDate` метку, используя метод `timeIntervalSinceReferenceDate()` . Он находится в форме ###. ### (# - числа), который мы будем использовать целую часть. Для этого разделим строку:

стриж

```
let timestamp = String(format: "%f", NSDate.timeIntervalSinceReferenceDate())
let timestampParts = timestamp.componentsSeparatedByString(".")
let recordID = CKRecordID(recordName: timestampParts[0])
```

Создание записи

Чтобы сделать запись, мы должны указать тип записи (объясненный в разделе «Использование CloudKit Dashboard») как «Пользователи», идентификатор, который мы сделали сейчас, и данные. Здесь мы добавим образец текста, изображение и текущую дату в запись:

стриж

```
let record = CKRecord(recordType: "Users", recordID: recordID)
record.setObject("Some Text", forKey: "text")
record.setObject(CKAsset(fileURL: someValidImageURL), forKey: "image")
record.setObject(NSDate(), forKey: "date")
```

Objective-C

```
CKRecord *record = [[CKRecord alloc] initWithRecordType: "Users" recordID: recordID];
[record setObject: "Some Text" forKey: "text"];
[record setObject: [CKAsset assetWithFileURL: someValidImageURL] forKey: "image"];
[record setObject: [[NSDate alloc] init] forKey: "date"];
```

Заметка

Здесь мы не добавляли `UIImage` непосредственно в запись, потому что, как упоминалось в Заметках, формат изображения напрямую не поддерживается в CloudKit, поэтому мы преобразовали `UIImage` в `CKAsset` .

Доступ к контейнеру

стриж

```
let container = CKContainer.defaultContainer()
let database = container.privateCloudDatabase // or container.publicCloudDatabase
```

Сохранение записей в базе данных CloudKit

стриж

```
database.saveRecord(record, completionHandler: { (_, error) -> Void in
    print(error ?? "")
})
```

Прочитайте CloudKit онлайн: https://riptutorial.com/ru/ios/topic/4946/cloudkit

глава 21: Core Motion

Examples

Доступ к барометру для получения относительной высоты

стриж

Импортируйте библиотеку Core Motion:

```
import CoreMotion
```

Затем нам нужно создать объект `CMAltimeter` , но общей ошибкой является создание его в `viewDidLoad()` . Если это будет сделано, альтиметр не будет доступен, если нам нужно вызвать метод на нем. Тем не менее, продолжайте и создайте свой объект `CMAltimeter` непосредственно перед `viewDidLoad()` :

```
let altimeter = CMAltimeter()
```

Сейчас:

1. Нам нужно проверить, доступен ли `relativeAltitude` с помощью следующего метода: `CMAltimeter.isRelativeAltitudeAvailable` .
2. Если это вернет `true` , вы можете начать мониторинг изменения высоты с помощью `startRelativeAltitudeUpdatesToQueue`
3. Если ошибок нет, вы должны иметь возможность извлекать данные из `relativeAltitude` свойств и свойств давления.

Ниже приведено определение действия кнопки, чтобы начать мониторинг с помощью нашего барометра.

```
@IBAction func start(sender: AnyObject){
if CMAltimeter.isRelativeAltitudeAvailable() {
    // 2
    altimeter.startRelativeAltitudeUpdatesToQueue(NSOperationQueue.mainQueue(), withHandler: {
data, error in
        // 3
        if (error == nil) {
            println("Relative Altitude: \(data.relativeAltitude)")
            println("Pressure: \(data.pressure)")
        }
    })
}
```

Прочитайте Core Motion онлайн: https://riptutorial.com/ru/ios/topic/7636/core-motion

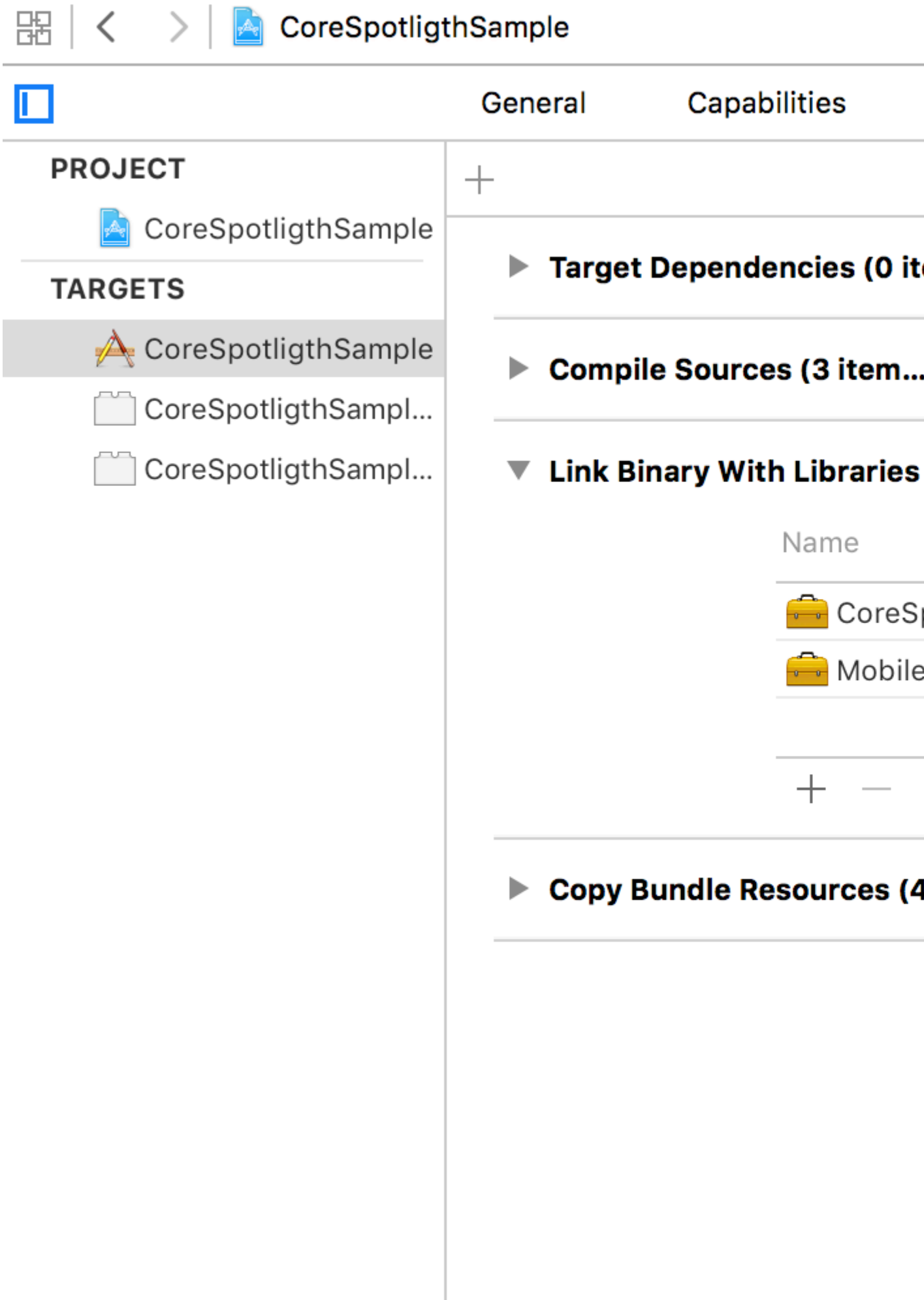
глава 22: Core SpotLight в iOS

Examples

Core-прожектор

Objective-C

1. Создайте новый проект iOS и добавьте *инфраструктуру CoreSpotlight* и *MobileCoreServices* в свой проект.


CoreSpotligthSample
General
Capabilities
PROJECT
CoreSpotligthSample
TARGETS
CoreSpotligthSample
CoreSpotligthSampl...
CoreSpotligthSampl...
Target Dependencies (0 it
Compile Sources (3 item...
Link Binary With Libraries
Name
CoreSp
Mobile
Copy Bundle Resources (4

2. Создайте фактический CSSearchableItem и свяжите уникальныйIdentifier, domainIdentifier и attributeSet. Наконец, индексируйте CSSearchableItem, используя [[CSSearchableIndex defaultSearchableIndex] ...], как показано ниже.

```
//
//  Created by Mayqiyue on 7/10/15.
//  Copyright © 2015 mayqiyue. All rights
//

#import "ViewController.h"
#import <CoreSpotlight/CoreSpotlight.h>
#import <MobileCoreServices/MobileCoreSer

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    [self setupCoreSpotlightSearch];
}

- (void)setupCoreSpotlightSearch {
    CSSearchableItemAttributeSet *attribu
        (NSString *)kUTTypeImage];
    attributeSet.title = @"My First Spotl
    attributeSet.contentDescription = @"'
    attributeSet.keywords = [NSArray arra
    UIImage *image = [UIImage imageNamed:
    NSData *imageData = [NSData dataWithD
    attributeSet.thumbnailData = imageDat

    CSSearchableItem *item = [[CSSearchab
        domainIdentifier:@"spotlight.samp

    [[CSSearchableIndex defaultSearchable
        error) {
        if (!error)
```

Identifier, domainIdentifier и attributeSet. Наконец, индексируйте CSSearchableItem, используя [[CSSearchableIndex defaultSearchableIndex] ...], как показано ниже.

```
//
//  Created by Mayqiyue on 7/10/15.
//  Copyright © 2015 mayqiyue. All rights
//

#import "ViewController.h"
#import <CoreSpotlight/CoreSpotlight.h>
#import <MobileCoreServices/MobileCoreSer

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    [self setupCoreSpotlightSearch];
}

- (void)setupCoreSpotlightSearch {
    CSSearchableItemAttributeSet *attribu
        (NSString *)kUTTypeImage];
    attributeSet.title = @"My First Spotl
    attributeSet.contentDescription = @"'
    attributeSet.keywords = [NSArray arra
    UIImage *image = [UIImage imageNamed:
    NSData *imageData = [NSData dataWithD
    attributeSet.thumbnailData = imageDat

    CSSearchableItem *item = [[CSSearchab
        domainIdentifier:@"spotlight.samp

    [[CSSearchableIndex defaultSearchable
        error) {
        if (!error)
```

используя [[CSSearchableIndex defaultSearchableIndex] ...], как показано ниже.

```
//
//  Created by Mayqiyue on 7/10/15.
//  Copyright © 2015 mayqiyue. All rights
//

#import "ViewController.h"
#import <CoreSpotlight/CoreSpotlight.h>
#import <MobileCoreServices/MobileCoreSer

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    [self setupCoreSpotlightSearch];
}

- (void)setupCoreSpotlightSearch {
    CSSearchableItemAttributeSet *attribu
        (NSString *)kUTTypeImage];
    attributeSet.title = @"My First Spotl
    attributeSet.contentDescription = @"'
    attributeSet.keywords = [NSArray arra
    UIImage *image = [UIImage imageNamed:
    NSData *imageData = [NSData dataWithD
    attributeSet.thumbnailData = imageDat

    CSSearchableItem *item = [[CSSearchab
        domainIdentifier:@"spotlight.samp

    [[CSSearchableIndex defaultSearchable
        error) {
        if (!error)
```

3. OK! Проверьте индекс!

глава 23: CTCallCenter

Examples

Перехват вызовов из вашего приложения даже из фона

Из документации Apple:

> Используйте класс CTCallCenter для получения списка текущих сотовых вызовов и для ответа на изменения состояния для вызовов, например, из состояния набора в подключенное состояние. Такие изменения состояния называются событиями сотового вызова.

Цель CTCallCenter - предоставить разработчику возможность приостановить его состояние приложения во время разговора, чтобы дать пользователю лучший опыт.

Objective-C:

Сначала мы определим новый член класса внутри класса, который мы хотим обработать перехватами:

```
@property (atomic, strong) CTCallCenter *callCenter;
```

Внутри нашего класса init (конструктор) мы выделяем новую память для нашего члена класса:

```
[self setCallCenter:[CTCallCenter new]];
```

Впоследствии мы будем ссылаться на наш новый метод, который фактически обрабатывает перехваты:

```
- (void)registerPhoneCallListener
{
[[self callCenter] setCallEventHandler:^(CTCall * _Nonnull call) {
    NSLog(@"CallEventHandler called - interception in progress");

     if ([call.callState isEqualToString: CTCallStateConnected])
     {
         NSLog(@"Connected");
     }
     else if ([call.callState isEqualToString: CTCallStateDialing])
     {
         NSLog(@"Dialing");
     }
     else if ([call.callState isEqualToString: CTCallStateDisconnected])
     {
         NSLog(@"Disconnected");
```

```
     } else if ([call.callState isEqualToString: CTCallStateIncoming])
     {
         NSLog(@"Incomming");
     }
 }];
}
```

Вот именно, если пользователь будет использовать ваше приложение и получит телефонный звонок, вы можете перехватить этот вызов и обработать свое приложение для сохранения состояния.

Стоит отметить, что есть 4 состояния вызова, которые вы можете перехватить:

```
CTCallStateDialing
CTCallStateIncoming
CTCallStateConnected
CTCallStateDisconnected
```

Swift:

Определите своего члена класса в соответствующем классе и определите его:

```
    self.callCenter = CTCallCenter()
    self.callCenter.callEventHandler = { call in
        //  Handle your interception
        if call.callState == CTCallStateConnected
        {
        }
    }
```

Что произойдет, если ваше приложение находится в фоновом режиме, и вам нужно перехватить вызовы, пока приложение находится в фоновом режиме?

Например, если вы разрабатываете **корпоративное** приложение, вы можете просто добавить 2 возможности (VOIP & Background fetch) на вкладке «Возможности»:

Цель проекта -> Возможности -> Фоновые режимы -> отметка Передача голоса по IP и фоновой выборке

CallKit - ios 10

```
//Header File

<CallKit/CXCallObserver.h>

CXCallObserver *callObserver = [[CXCallObserver alloc] init];

// If queue is nil, then callbacks will be performed on main queue

[callObserver setDelegate:self queue:nil];
```

```
// Don't forget to store reference to callObserver, to prevent it from being released

self.callObserver = callObserver;

// get call status
- (void)callObserver:(CXCallObserver *)callObserver callChanged:(CXCall *)call {
    if (call.hasConnected) {
        // perform necessary actions
    }
}
```

Прочитайте CTCallCenter онлайн: https://riptutorial.com/ru/ios/topic/3007/ctcallcenter

глава 24: CydiaSubstrate tweak

Вступление

Узнайте, как создать настройки cydia субстрата для взломанных iPhone.

Эти настройки позволят вам изменить поведение операционной системы, чтобы действовать так, как вам хотелось бы.

замечания

Установка Theos

https://github.com/theos/theos/wiki/Installation

Examples

Создайте новую настройку с помощью Theos

Используйте nic для создания нового проекта

Введите эту команду в терминал

```
$THEOS/bin/nic.pl
```

```
NIC 2.0 - New Instance Creator
------------------------------
  [1.] iphone/activator_event
  [2.] iphone/application_modern
  [3.] iphone/cydget
  [4.] iphone/flipswitch_switch
  [5.] iphone/framework
  [6.] iphone/ios7_notification_center_widget
  [7.] iphone/library
  [8.] iphone/notification_center_widget
  [9.] iphone/preference_bundle_modern
  [10.] iphone/tool
  [11.] iphone/tweak
  [12.] iphone/xpc_service
Choose a Template (required):
```

Выберите шаблон `[11.] iphone/tweak`

Заполните детали, и вы создадите следующие файлы:

```
-rw-r--r--@  1 gkpln3  staff   214B Jun 12 15:09 Makefile
-rw-r--r--@  1 gkpln3  staff    89B Jun 11 22:58 TorchonFocus.plist
-rw-r--r--   1 gkpln3  staff   2.7K Jun 12 16:10 Tweak.xm
-rw-r--r--   1 gkpln3  staff   224B Jun 11 16:17 control
drwxr-xr-x   3 gkpln3  staff   102B Jun 11 16:18 obj
drwxr-xr-x  16 gkpln3  staff   544B Jun 12 16:12 packages
```

Переопределить метод сохранения снимков iOS

откройте файл `Tweak.xm` используя ваш любимый редактор кода.

привязать к определенному методу из операционной системы.

```
%hook SBScreenShotter
- (void)saveScreenshot:(BOOL)screenshot
{
    %orig;
    NSLog(@"saveScreenshot: is called");
}
%end
```

Обратите внимание, что вы можете выбрать «wether», или не следует вызывать функцию оригинала, например:

```
%hook SBScreenShotter
- (void)saveScreenshot:(BOOL)screenshot
{
    NSLog(@"saveScreenshot: is called");
}
%end
```

будет переопределять функцию без вызова исходного, поэтому скриншоты обрезки не сохраняются.

Прочитайте CydiaSubstrate tweak онлайн:
https://riptutorial.com/ru/ios/topic/10533/cydiasubstrate-tweak

глава 25: DispatchGroup

Вступление

Похожие темы:

Центральная диспетчерская станция

совпадение

Examples

Вступление

Предположим, что у вас несколько потоков. Каждый поток выполняет одну задачу. Вы хотите получать уведомления либо на mainThread ИЛИ в другом потоке, когда все задачи-потоки завершены.

Простейшим решением такой проблемы является `DispatchGroup` .

При использовании группы `DispatchGroup` для каждого запроса вы `enter` группу, и для каждого завершенного запроса вы `leave` группу.

Если в группе больше нет запросов, вы будете `notify` (уведомлены).

Использование:

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()


        let dispatchGroup = DispatchGroup() //Create a group for the tasks.
        let session: URLSession = URLSession.shared

        dispatchGroup.enter() //Enter the group for the first task.

        let firstTask = session.dataTask(with: URLRequest(url: URL(string:
"https://stackoverflow.com")!)) { (data, response, error) in

            //Process Response..

            dispatchGroup.leave() //Leave the group for the first task.
        }
```

```
        dispatchGroup.enter()  //Enter the group for the second task.

        let secondTask = session.dataTask(with: URLRequest(url: URL(string:
"https://google.ca")!)) { (data, response, error) in

            //Process Response..

            dispatchGroup.leave()  //Leave the group for the second task.
        }


        //Get notified on the main thread/queue.. when ALL of the tasks above has been
completed.
        dispatchGroup.notify(queue: DispatchQueue.main) {

            print("Every task is complete")

        }


        //Start the tasks.
        firstTask.resume()
        secondTask.resume()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}
```

С вышесказанным вам не нужно `wait` бесконечно, пока все задачи не будут завершены. Вы можете отобразить загрузчик перед началом всех задач и отпустить загрузчик ПОСЛЕ завершения всех задач. Таким образом, ваш основной поток не блокируется, и ваш код остается чистым.

Предположим, вы также хотели, чтобы задания `ordered` или добавляли свои ответы в массив последовательно. Вы можете сделать следующее:

```
import UIKit

//Locking mechanism..
func synchronized(_ lock: AnyObject, closure: () -> Void) {
    objc_sync_enter(lock)
    closure()
    objc_sync_exit(lock)
}

class ViewController: UIViewController {

    let lock = NSObject() //Object to lock on.
    var responseArray = Array<Data?>() //Array of responses.

    override func viewDidLoad() {
        super.viewDidLoad()

        let dispatchGroup = DispatchGroup()
        let session: URLSession = URLSession.shared
```

```
        dispatchGroup.enter() //Enter the group for the first task.

        let firstTask = session.dataTask(with: URLRequest(url: URL(string:
"https://stackoverflow.com")!)) { (data, response, error) in

            //Process Response..

            synchronized(self.lock, closure: { () -> Void in
                self.responseArray[0] = data ?? nil
            })

            dispatchGroup.leave() //Leave the group for the first task.
        }


        dispatchGroup.enter()  //Enter the group for the second task.

        let secondTask = session.dataTask(with: URLRequest(url: URL(string:
"https://google.ca")!)) { (data, response, error) in

            //Process Response..

            synchronized(self.lock, closure: { () -> Void in
                self.responseArray[1] = data ?? nil
            })

            dispatchGroup.leave()  //Leave the group for the second task.
        }


        //Get notified on the main thread.. when ALL of the requests above has been completed.
        dispatchGroup.notify(queue: DispatchQueue.main) {

            print("Every task is complete..")

            for i in 0..<self.responseArray.count {

                if self.responseArray[i] == nil {
                    print("Request #\(i) Failed.\n")
                }
                else {
                    print("Request #\(i) Succeeded.\n")
                }
            }
        }

        //Two tasks added to the array. Responses are assumed nil until they complete.
        self.responseArray.append(nil)
        self.responseArray.append(nil)

        //Start the tasks.
        firstTask.resume()
        secondTask.resume()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}
```

Заметки

Каждая запись должна иметь выход в `DispatchGroup` . Если вы забыли `leave` после `entering` , вы настраиваете себя. Вы никогда не будете уведомлены, когда задачи будут завершены.

Сумма `enter` должна быть равна размеру `leave` .

Прочитайте DispatchGroup онлайн: https://riptutorial.com/ru/ios/topic/4624/dispatchgroup

глава 26: EventKit

Examples

Запрос разрешения

Ваше приложение не может получить доступ к своим напоминаниям и вашему календарю без разрешения. Вместо этого он должен показывать предупреждение пользователю, запрашивая у него / нее возможность доступа к событиям для приложения.

Чтобы начать работу, импортируйте фреймворк `EventKit` :

стриж

```
import EventKit
```

Objective-C

```
#import <EventKit/EventKit.h>
```

Создание `EKEventStore`

Затем мы `EKEventStore` объект `EKEventStore` . Это объект, из которого мы можем получить доступ к данным календаря и напоминаний:

стриж

```
let eventStore = EKEventStore()
```

Objective-C

```
EKEventStore *eventStore = [[EKEventStore alloc] init];
```

Заметка

Создание объекта `EKEventStore` каждый раз, когда нам нужно получить доступ к календарю, неэффективно. Попробуйте сделать это один раз и использовать

его везде в своем коде.

Проверка доступности

Доступность имеет три разных статуса: Авторизованный, Отклоненный и Не определено. Не определено. Приложение должно предоставлять доступ.

Чтобы проверить доступность, мы используем метод `authorizationStatusForEntityType()` объекта `EKEventStore` :

стриж

```
switch EKEventStore.authorizationStatusForEntityType(EKEntityTypeEvent){
    case .Authorized: //...
    case .Denied: //...
    case .NotDetermined: //...
    default: break
}
```

Objective-C

```
switch ([EKEventStore authorizationStatusForEntityType:EKEntityTypeEvent]){
    case EKAuthorizationStatus.Authorized:
        //...
        break;
    case EKAuthorizationStatus.Denied:
        //...
        break;
    case EKAuthorizationStatus.NotDetermined:
        //...
        break;
    default:
        break;
}
```

Запрос разрешения

Поместите следующий код в случай `NotDetermined` :

стриж

```
eventStore.requestAccessToEntityType(EKEntityTypeEvent, completion: { [weak self]
(userGrantedAccess, _) -> Void in
    if userGrantedAccess{
        //access calendar
    }
```

```
}
```

Доступ к различным типам календарей

Доступ к массиву календарей

Чтобы получить доступ к массиву `EKCalendar` s, мы используем метод `calendarsForEntityType` :

стриж

```
let calendarsArray = eventStore.calendarsForEntityType(EKEntityType.Event) as! [EKCalendar]
```

Итерирование через календари

Просто используйте простой `for` цикла:

стриж

```
for calendar in calendarsArray{
    //...
}
```

Доступ к заголовку и цвету календаря

стриж

```
let calendarColor = UIColor(CGColor: calendar.CGColor)
let calendarTitle = calendar.title
```

Objective-C

```
UIColor *calendarColor = [UIColor initWithCGColor: calendar.CGColor];
NSString *calendarTitle = calendar.title;
```

Добавление события

Создание объекта события

стриж

```
var event = EKEvent(eventStore: eventStore)
```

Objective-C

```
EKEvent *event = [EKEvent initWithEventStore:eventStore];
```

Настройка соответствующего календаря, названия и дат

стриж

```
event.calendar = calendar
event.title = "Event Title"
event.startDate = startDate //assuming startDate is a valid NSDate object
event.endDate = endDate //assuming endDate is a valid NSDate object
```

Добавление события в календарь

стриж

```
try {
    do eventStore.saveEvent(event, span: EKSpan.ThisEvent)
} catch let error as NSError {
    //error
}
```

Objective-C

```
NSError *error;
BOOL *result = [eventStore saveEvent:event span:EKSpanThisEvent error:&error];
if (result == NO){
    //error
}
```

Прочитайте EventKit онлайн: https://riptutorial.com/ru/ios/topic/5854/eventkit

глава 27: FacebookSDK

Examples

Интеграция с FacebookSDK

Шаг 1: Установите SDK

Вы можете установить SDK вручную или через `CocoaPods` . Последний вариант настоятельно рекомендуется.

Поместите эти строки в `Podfile` :

```
target 'MyApp' do
  use_frameworks!

  pod 'FBSDKCoreKit'
  pod 'FBSDKLoginKit'
  pod 'FBSDKShareKit'
end
```

Запустите `pod install` в терминале и откройте `.xcworkspace` вместо `.xcodeproj` .

`FBSDKLoginKit` и `FBSDKShareKit` являются необязательными. Вы можете или не нуждаться в них.

Шаг 2. Создание приложения на Facebook.

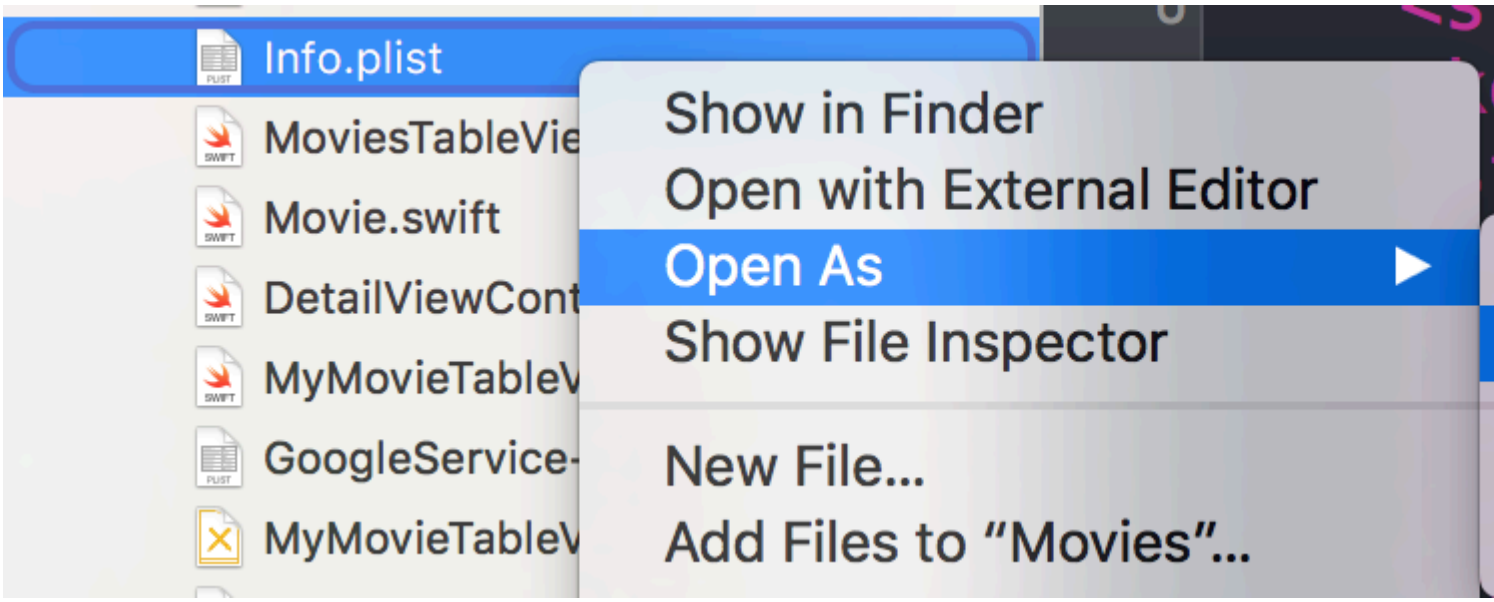
Перейдите в Quick Starts - Facebook для разработчиков, чтобы создать приложение.

Facebook попросит вас загрузить SDK после создания приложения. Вы можете пропустить эту часть, если вы уже установили SDK через CocoaPods.

Шаг 3: Изменить `.plist`

а. Чтобы ваше приложение могло «общаться» с Facebook, вам нужно поместить некоторые настройки в ваш `.plist` файл. Facebook предоставит вам настроенный фрагмент на странице быстрого запуска.

б. Измените файл `.plist` как исходный код.



c. Вставьте свой собственный фрагмент кода в исходный код. **Быть осторожен!** Фрагмент должен быть точно дочерним тегом `<dict>` . Ваш исходный код должен выглядеть примерно так:

```
<plist version="1.0">
<dict>
    // ...
    //some default settings
    // ...
    <key>CFBundleURLTypes</key>
    <array>
        <dict>
            <key>CFBundleURLSchemes</key>
            <array>
                <string>fb{FBAppId}</string>
            </array>
        </dict>
    </array>
    <key>FacebookAppID</key>
    <string>{FBAppId}</string>
    <key>FacebookDisplayName</key>
    <string>{FBAppName}</string>
    <key>LSApplicationQueriesSchemes</key>
    <array>
        <string>fbapi</string>
        <string>fb-messenger-api</string>
        <string>fbauth2</string>
        <string>fbshareextension</string>
    </array>
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSExceptionDomains</key>
    <dict>
        <key>facebook.com</key>
        <dict>
            <key>NSIncludesSubdomains</key>
            <true/>
            <key>NSExceptionRequiresForwardSecrecy</key>
            <false/>
        </dict>
```

```
        <key>fbcdn.net</key>
        <dict>
            <key>NSIncludesSubdomains</key>
            <true/>
            <key>NSExceptionRequiresForwardSecrecy</key>
            <false/>
        </dict>
        <key>akamaihd.net</key>
        <dict>
            <key>NSIncludesSubdomains</key>
            <true/>
            <key>NSExceptionRequiresForwardSecrecy</key>
            <false/>
        </dict>
    </dict>
</dict>
</plist>
```

Если вы вставляете фрагмент в неправильное место, вы столкнетесь с проблемами.

Шаг 4. Расскажите Facebook свой идентификатор пакета на странице быстрого запуска.

=> Как получить идентификатор пакета

Шаг 5: Отредактируйте свой `AppDelegate.swift`

а.

```
import FBSDKCoreKit
```

б.

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    FBSDKApplicationDelegate.sharedInstance().application(application,
didFinishLaunchingWithOptions: launchOptions)
    return true
}

func application(application: UIApplication, openURL url: NSURL, sourceApplication: String?,
annotation: AnyObject) -> Bool {
    return FBSDKApplicationDelegate.sharedInstance().application(application, openURL: url,
sourceApplication: sourceApplication, annotation: annotation)
}
```

Создание собственной пользовательской кнопки «Войти в Facebook»

Иногда мы хотим создать собственный пользовательский интерфейс для кнопки «Войти в Facebook» вместо оригинальной кнопки, которая поставляется с FacebookSDK.

1. В своем раскадровке перетащите свой UIButton и установите его, как бы вы этого ни хотели.

2. Ctrl + перетащите кнопку на контроллер вида в виде IBAction.
3. **Внутри** метода IBAction вы будете симулировать нажатие на фактическую кнопку Facebook следующим образом:

Swift:

```
    let loginButton = FBSDKLoginButton()
    loginButton.delegate = self
    // Your Custom Permissions Array
    loginButton.readPermissions =
    [
                            "public_profile",
                            "email",
                            "user_about_me",
                            "user_photos"
    ]
    // Hiding the button
    loginButton.hidden = true
    self.view.addSubview(loginButton)
    // Simulating a tap for the actual Facebook SDK button
    loginButton.sendActionsForControlEvents(UIControlEvents.TouchUpInside)
```

Objective-C:

```
FBSDKLoginButton *FBButton = [FBSDKLoginButton new];

// Your Custom Permissions Array
FBButton.readPermissions = @[@"public_profile",
                             @"email",
                             @"user_about_me",
                             @"user_photos"
                             ];
FBButton.loginBehavior = FBSDKLoginBehaviorNative;
[FBButton setDelegate:self];
[FBButton setHidden:true];
[loginButton addSubview:FBButton];

[FBButton sendActionsForControlEvents:UIControlEventTouchUpInside];
```

Все готово.

Получение пользовательских данных facebook

После того, как пользователь зарегистрировался в Facebook в своем приложении, пришло время получить данные, запрошенные вами на `FBButton.readPermissions` .

Swift:

```
enum FacebookParametesField : String
{
    case FIELDS_KEY = "fields"
    case FIELDS_VALUE = "id, email, picture, first_name, last_name"
}
```

```
if FBSDKAccessToken.currentAccessToken() != nil
{
//  Getting user facebook data
FBSDKGraphRequest(graphPath: "me",
                  parameters: [FacebookParametesField.FIELDS_KEY.rawValue :
FacebookParametesField.FIELDS_VALUE.rawValue])
.startWithCompletionHandler({ (graphConnection : FBSDKGraphRequestConnection!, result :
AnyObject!, error : NSError!) -> Void in

        if error == nil
        {
            print("Facebook Graph phaze")

            let email = result["email"]
            let facebookToken = FBSDKAccessToken.currentAccessToken().tokenString
            let userFacebookId = result["id"]
            let firstName = result["first_name"]
            let lastName = result["last_name"]

            if let result = result as? Dictionary<String, AnyObject>
            {
                if let picture = result["picture"] as? Dictionary<String,AnyObject>
                {
                    if let data = picture["data"] as? Dictionary <String,AnyObject>
                    {
                        if let url = data["url"] as? String
                        {
                            //  Profile picture URL
                            let profilePictureURL = url
                        }
                    }
                }
            }
        }
    })
}
```

Прочитайте FacebookSDK онлайн: https://riptutorial.com/ru/ios/topic/2972/facebooksdk

глава 28: Fastlane

Examples

инструменты fastlane

fastlane - это инструмент автоматизации сборки с открытым исходным кодом для Android и iOS для разработчиков. Это сокращает время создания сборки. Это инструмент командной строки, который использует Ruby , поэтому вам нужен Ruby на вашем компьютере. У большинства компьютеров Mac уже установлен Ruby по умолчанию.

Установите fastlane

1. Откройте терминал.
2. Запустить `sudo gem install fastlane --verbose`
3. Если вы еще не установили инструменты командной строки Xcode, запустите `xcode-select --install` чтобы установить их
4. Теперь, `cd` в папку проекта (введите `cd` [с пробелом в конце] и перетащите папку проекта в терминал)
5. Запустите `fastlane init` чтобы получить настройку Fastlane.
6. Теперь вы можете использовать все инструменты Fastlane:

Инструменты iOS

- доставить : загрузить скриншоты, метаданные и приложение в App Store
- моментальный снимок : автоматизация снятия локализованных скриншотов вашего приложения iOS на каждом устройстве
- frameit : быстро помещайте скриншоты в нужные рамы устройства
- pem : автоматически создавать и обновлять профили уведомлений push
- вздох : потому что вы предпочтете потратить свое время на строительство, а не на обеспечение безопасности
- произвести : создать новые приложения для iOS на iTunes Connect и Dev Portal, используя командную строку
- cert : автоматически создавать и поддерживать сертификаты подписи кода iOS
- тренажерный зал : создание приложений для iOS никогда не было проще
- match : легко синхронизировать свои сертификаты и профили в вашей команде с помощью Git
- scan : Самый простой способ запустить тесты для приложений iOS и Mac
- космический корабль : библиотека Ruby для доступа к Apple Dev Center и iTunes Connect

iOS TestFlight Tools

- pilot : лучший способ управлять тестерами TestFlight и строить из вашего терминала
- посадка : самый простой способ пригласить тестеров TestFlight

Инструменты для Android

- питания : Загрузить Android приложение и его метаданные в Google Play
- screengrab : автоматизация с помощью локализованных скриншотов вашего приложения для Android на каждом устройстве

Прочитайте Fastlane онлайн: https://riptutorial.com/ru/ios/topic/3574/fastlane

глава 29: FileHandle

Вступление

Прочитать файл в кусках из каталога документов

Examples

Прочитать файл из каталога документов в кусках

Я получаю путь к файлу из каталога документов и читаю этот файл в кусках 1024 и сохраняю (добавляет) объект `NSMutableData` или вы можете напрямую писать в сокет.

```
// MARK: - Get file data as chunks Methode.
func getFileDataInChunks() {

    let doumentDirectoryPath = NSSearchPathForDirectoriesInDomains(.documentDirectory,
.userDomainMask, true)[0] as NSString
    let filePath = doumentDirectoryPath.appendingPathComponent("video.mp4")


    //Check file exits at path or not.
    if FileManager.default.fileExists(atPath: filePath) {

        let chunkSize = 1024 // divide data into 1 kb

        //Create NSMutableData object to save read data.
        let ReadData = NSMutableData()

        do {

            //open file for reading.
            outputFileHandle = try FileHandle(forReadingFrom: URL(fileURLWithPath: filePath))

            // get the first chunk
            var datas = outputFileHandle?.readData(ofLength: chunkSize)

            //check next chunk is empty or not.
            while !(datas?.isEmpty)! {

                //here I write chunk data to ReadData or you can directly write to socket.
                ReadData.append(datas!)

                // get the next chunk
                datas = outputFileHandle?.readData(ofLength: chunkSize)

                print("Running: \(ReadData.length)")
            }

            //close outputFileHandle after reading data complete.
            outputFileHandle?.closeFile()

            print("File reading complete")
```

```
        }catch let error as NSError {
            print("Error : \(error.localizedDescription)")
        }
    }
}
```

После чтения файла завершения вы получите данные файла в `ReadData` переменного Здесь outputFileHandle является объектом `FileHandle`

```
var outputFileHandle:FileHandle?
```

Прочитайте FileHandle онлайн: https://riptutorial.com/ru/ios/topic/10665/filehandle

глава 30: GameplayKit

Examples

Генерация случайных чисел

Хотя `GameplayKit` (который представлен с iOS 9 SDK) посвящен реализации логики игры, он также может быть использован для генерации случайных чисел, что очень полезно в приложениях и играх.

Помимо `GKRandomSource.sharedRandom` который используется в следующих главах, есть три дополнительных типа `GKRandomSource` 's из коробки.

- **GKARC4RandomSource,** который использует алгоритм **ARC4**
- **GKLinearCongruentialRandomSource,** который является быстрым, но не столь случайным `GKRandomSource`
- **GKMersenneTwisterRandomSource,** который реализует алгоритм MersenneTwister. Он медленнее, но более случайным.

В следующей главе мы используем только метод `nextInt()` для `GKRandomSource` . В дополнение к этому есть `nextBool() -> Bool` и `nextUniform() -> Float`

поколение

Во-первых, импортируйте `GameplayKit` :

стриж

```
import GameplayKit
```

Objective-C

```
#import <GameplayKit/GameplayKit.h>
```

Затем, чтобы сгенерировать случайное число, используйте этот код:

стриж

```
let randomNumber = GKRandomSource.sharedRandom().nextInt()
```

Objective-C

```
int randomNumber = [[GKRandomSource sharedRandom] nextInt];
```

> ## Заметка
>
> Функция nextInt () при использовании без параметров возвращает случайное число между -2,147,483,648 и 2,147,483,647, включая самих себя, поэтому мы не уверены, что это всегда положительное или ненулевое число.

Создание числа от 0 до n

Чтобы добиться этого, вы должны дать n `nextIntWithUpperBound()` методу `nextIntWithUpperBound()` :

стриж

```
let randomNumber = GKRandomSource.sharedRandom().nextInt(upperBound: 10)
```

Objective-C

```
int randomNumber = [[GKRandomSource sharedRandom] nextIntWithUpperBound: 10];
```

Этот код даст нам число от 0 до 10, включая себя.

Создание числа от m до n

Для этого вы создаете объект `GKRandomDistribution` с `GKRandomSource` и проходите в границах. `GKRandomDistribution` можно использовать для изменения поведения распределения, такого как `GKGaussianDistribution` или `GKShuffledDistribution` .

После этого объект может использоваться как каждый регулярный `GKRandomSource` поскольку он также реализует протокол `GKRandom` .

стриж

```
let randomizer = GKRandomDistribution(randomSource: GKRandomSource(), lowestValue: 0,
highestValue: 6)
let randomNumberInBounds = randomizer.nextInt()
```

Объектив-С *устарел*

```
int randomNumber = [[GKRandomSource sharedRandom] nextIntWithUpperBound: n - m] + m;
```

Например, чтобы создать случайное число от 3 до 10, вы используете этот код:

стриж

```
let randomNumber = GKRandomSource.sharedRandom().nextInt(upperBound: 7) + 3
```

Объектив-С *устарел*

```
int randomNumber = [[GKRandomSource sharedRandom] nextIntWithUpperBound: 7] + 3;
```

GKEntity и GKComponent

Сущность представляет объект игры, такой как фигура игрока или фигура врага. Поскольку этот объект не делает много без рук и ног, мы можем добавить к нему компоненты. Для создания этой системы в Apple есть `GKEntity` и `GKComponent` .

Предположим, что у нас есть следующий класс для следующих глав:

```
class Player: GKEntity{}
class PlayerSpriteComponent: GKComponent {}
```

GKEntity

Сущность представляет собой набор компонентов и предлагает несколько функций для добавления, удаления и взаимодействия с ее компонентами.

Хотя мы могли бы просто использовать GKEntity, он является общим для подкласса для определенного типа игрового объекта.

Важно, что только один раз можно добавить компонент класса. Если вы добавите второй компонент того же класса, он переопределит первый существующий компонент внутри `GKEntity`

```
let otherComponent = PlayerSpriteComponent()
var player = Player()
player.addComponent(PlayerSpriteComponent())
player.addComponent(otherComponent)
print(player.components.count) //will print 1
print(player.components[0] === otherComponent) // will print true
```

Вы можете спросить, почему. Причиной этого являются методы, называемые `component(for: T.Type)` который возвращает компонент определенного типа объекта.

```
let component = player.component(ofType: PlayerSpriteComponent.self)
```

В дополнение к компонентам-методам у него есть метод `update` который используется для делегирования дельта-времени или текущего времени игровой логики на его компоненты.

```
var player = Player()
player.addComponent(PlayerSpriteComponent())
player.update(deltaTime: 1.0) // will call the update method of the PlayerSpriteComponent
added to it
```

GKComponent

Компонент представляет собой нечто вроде объекта, например визуального компонента или логического компонента.

Если вы вызываете метод обновления объекта, он делегирует его всем его компонентам. Переопределение этого метода используется для управления сущностью.

```
class PlayerSpriteComponent: GKComponent {
    override func update(deltaTime seconds: TimeInterval) {
        //move the sprite depending on the update time
    }
}
```

В дополнение к этому можно переопределить метод `didAddToEntity` и `willRemoveFromEntity` чтобы сообщить другим компонентам о его удалении или добавлении.

Для управления другим компонентом внутри компонента можно получить GKEntity, к которому добавлен компонент.

```
override func update(deltaTime seconds: TimeInterval) {
    let controller = self.entity?.component(ofType: PlayerControlComponent.self)
    //call methods on the controller
}
```

Хотя это возможно, это **не** общий шаблон, поскольку он соединяет два компонента вместе.

GKComponentSystem

Хотя мы просто говорили об использовании механизма делегирования обновлений `GKEntity` для обновления `GKComponents` существует другой способ обновления `GKComponents` который называется `GKComponentSystem` .

Он используется в случае необходимости, чтобы все компоненты определенного типа нуждались в обновлении за один раз.

`GKComponentSystem` создается для определенного типа компонента.

```
let system = GKComponentSystem(componentClass: PlayerSpriteComponent.self)
```

Чтобы добавить компонент, вы можете использовать метод add:

```
system.addComponent(PlayerSpriteComponent())
```

Но более распространенным способом является передача созданного объекта с его компонентами в `GKComponentSystem` и он найдет соответствующий компонент внутри объекта.

```
system.addComponent(foundIn: player)
```

Чтобы обновить все компоненты определенного типа, выполните обновление:

```
system.update(deltaTime: delta)
```

Если вы хотите использовать `GKComponentSystem` вместо механизма обновления на основе `GKComponentSystem` для каждого компонента необходимо иметь `GKComponentSystem` и вызывать обновление для всех систем.

Прочитайте GameplayKit онлайн: https://riptutorial.com/ru/ios/topic/4966/gameplaykit

глава 31: GCD (Grand Central Dispatch)

Вступление

Grand Central Dispatch (GCD) - ответ Apple на многопоточность. Это облегченная структура для выполнения задач синхронно или асинхронно в очередях и обрабатывает потоки ЦП для вас за кулисами.

Связанная тема: параллелизм

Examples

Создание очереди отправки

Вы можете создать свою собственную очередь, используя `dispatch_queue_create`

Objective-C

```
dispatch_queue_t queue = dispatch_queue_create("com.example.myqueue",  DISPATCH_QUEUE_SERIAL);
```

стриж

```
// Before Swift 3
let queue = dispatch_queue_create("com.example.myqueue", DISPATCH_QUEUE_SERIAL)
// Swift 3
let queue = DispatchQueue(label: "com.example.myqueue") //default is serial queue, unless
.concurrent is specified as an attribute otherwise
```

Получение главной очереди

Главной очередью является очередь отправки, в которой все обновления пользовательского интерфейса, и код с изменениями пользовательского интерфейса.

Вам нужно попасть в основную очередь, чтобы обновить интерфейс после завершения асинхронного процесса, такого как `NSURLSession`

Существует два типа вызовов основной очереди: `synchronous` и `asynchronous` . Когда вы вызываете что-то `synchronously` , это означает, что поток, инициировавший эту операцию, будет ждать завершения задачи перед продолжением. `Asynchronous` означает, что он не будет ждать.

Цель кода-C

`Synchronous` вызов главной очереди

```
dispatch_queue_t queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
```

`Asynchronous` вызов главной очереди

```
dispatch_async(dispatch_get_main_queue(), ^{
   // do work here to Usually to update the User Interface
});
```

SWIFT 3

`Asynchronous` вызов главной очереди

```
DispatchQueue.main.async {

}
```

`Synchronous` вызов главной очереди

```
DispatchQueue.main.sync {

}
```

Диспетчерская группа

> DispatchGroup позволяет выполнять агрегатную синхронизацию работы. Вы можете использовать их для отправки нескольких разных рабочих элементов и отслеживания, когда они все завершатся, даже если они могут выполняться в разных очередях. Такое поведение может быть полезно, когда прогресс не может быть достигнут, пока все указанные задачи не будут завершены.

Сценарий, когда это может быть полезно, - это если у вас несколько вызовов webservice, которые все нужно завершить, прежде чем продолжить. Например, вам нужно загрузить несколько наборов данных, которые необходимо обработать с помощью некоторых функций. Вы должны дождаться завершения всех веб-сервисов перед вызовом функции для обработки всех полученных данных.

Swift 3

```
func doLongTasksAndWait () {
    print("starting long running tasks")
    let group = DispatchGroup()          //create a group for a bunch of tasks we are about to
do
    for i in 0...3 {                     //launch a bunch of tasks (eg a bunch of webservice
calls that all need to be finished before proceeding to the next ViewController)
        group.enter()                    //let the group know that something is being added
        DispatchQueue.global().async {   //run tasks on a background thread
            sleep(arc4random() % 4)      //do some long task eg webservice or database lookup
(here we are just sleeping for a random amount of time for demonstration purposes)
            print("long task \(i) done!")
            group.leave()                //let group know that the task is finished
```

```
        }
    }
    group.wait()                          //will block whatever thread we are on here until all
the above tasks have finished (so maybe dont use this function on your main thread)
    print("all tasks done!")
}
```

С другой стороны , если вы не хотите ждать , пока группы , чтобы закончить, но вместо этого хочет , чтобы запустить функцию , когда все задачи выполнили, используйте `notify` функцию вместо `group.wait()`

```
group.notify(queue: DispatchQueue.main) { //the queue: parameter is which queue this block
will run on, if you need to do UI updates, use the main queue
    print("all tasks done!")              //this will execute when all tasks have left the
group
}
```

Пример вывода:

```
starting long running tasks
long task 0 done!
long task 3 done!
long task 1 done!
long task 2 done!
all tasks done!
```

Для получения дополнительной информации см. Документы Apple или соответствующую тему

Диспетчерский семафор

> DispatchSemaphore обеспечивает эффективную реализацию традиционного счетного семафора, который может использоваться для управления доступом к ресурсу в нескольких контекстах выполнения.

Сценарий, когда использовать семафор, может быть, если вы выполняете чтение / запись файлов, если несколько задач пытаются одновременно читать и писать из файла, это может увеличить вашу производительность, чтобы каждая задача дождалась своей очереди так, чтобы чтобы не перегружать контроллер ввода-вывода.

Swift 3

```
func do2TasksAtATime () {
    print("starting long running tasks (2 at a time)")
    let sem = DispatchSemaphore(value: 2)            //this semaphore only allows 2 tasks to
run at the same time (the resource count)
    for i in 0...7 {                                 //launch a bunch of tasks
        DispatchQueue.global().async {               //run tasks on a background thread
            sem.wait()                               //wait here if no resources available
            sleep(2)                                 //do some long task eg file access (here
we are just sleeping for a 2 seconds for demonstration purposes)
```

```
            print("long task \(i) done! \(Date())")
            sem.signal()                                //let the semaphore know this resource is
now available
        }
    }
}
```

Пример вывода: (обратите внимание на отметки времени)

```
starting long running tasks (2 at a time)
long task 0 done! 2017-02-16 07:11:53 +0000
long task 1 done! 2017-02-16 07:11:53 +0000
long task 2 done! 2017-02-16 07:11:55 +0000
long task 3 done! 2017-02-16 07:11:55 +0000
long task 5 done! 2017-02-16 07:11:57 +0000
long task 4 done! 2017-02-16 07:11:57 +0000
long task 6 done! 2017-02-16 07:11:59 +0000
long task 7 done! 2017-02-16 07:11:59 +0000
```

Для получения дополнительной информации см. Документы Apple

Серийные и параллельные диспетчерские очереди

Swift 3

Серийная очередь

```
func serialQueues () {
    let serialQueue = DispatchQueue(label: "com.example.serial") //default queue type is a
serial queue
    let start = Date ()
    for i in 0...3 {                                            //launch a bunch of tasks
        serialQueue.async {                                     //run tasks on a background
thread, using our serial queue
            sleep(2)                                            //do some long task eg
webservice or database lookup
            let timeTaken = Date().timeIntervalSince(start)
            print("serial long task \(i) done! total time taken: \(timeTaken)")
        }
    }
}
```

Пример вывода:

```
serial long task 0 done! total time taken: 2.07241100072861
serial long task 1 done! total time taken: 4.16347700357437
serial long task 2 done! total time taken: 6.23209798336029
serial long task 3 done! total time taken: 8.30682599544525
```

Параллельная очередь

```
func concurrentQueues () {
    let concurrentQueue = DispatchQueue(label: "com.example.concurrent", attributes:
.concurrent) //explicitly specify the queue to be a concurrent queue
```

```
    let start = Date ()
    for i in 0...3 {            //launch a bunch of tasks
        concurrentQueue.async { //run tasks on a background thread, using our concurrent queue
            sleep(2)            //do some long task eg webservice or database lookup
            let timeTaken = Date().timeIntervalSince(start)
            print("concurrent long task \(i) done! total time taken: \(timeTaken)")
        }
    }
}
```

Пример вывода:

```
concurrent long task 3 done! total time taken: 2.07092100381851
concurrent long task 0 done! total time taken: 2.07087397575378
concurrent long task 2 done! total time taken: 2.07086700201035
concurrent long task 1 done! total time taken: 2.07089096307755
```

обсуждение

Как видно из приведенных выше примеров, последовательная очередь будет выполнять каждую задачу в том порядке, в котором они будут отправлены в очередь. Каждая задача будет ждать завершения предыдущей задачи перед выполнением. Что касается параллельной очереди, каждая задача не ждет остальных в очереди и выполняется как можно скорее; преимущество состоит в том, что все задачи в очереди будут выполняться одновременно с отдельными потоками, в результате чего одновременная очередь занимает меньше времени, чем последовательная очередь.

Если порядок выполнения задач не важен, всегда используйте параллельную очередь для обеспечения максимальной эффективности.

Прочитайте GCD (Grand Central Dispatch) онлайн: https://riptutorial.com/ru/ios/topic/4626/gcd--grand-central-dispatch-

глава 32: Healthkit

Examples

HealthKit

Objective-C

Сначала `Target->Capabilities HealthKit Target->Capabilities` и включите `HealthKit` . Это установит запись info.plist.

Создайте новый `CocoaClass` типа `NSObject` Имя файла, которое я дал, - `GSHealthKitManager` а файл заголовка показан ниже

GSHealthKitManager.h

```
#import <Foundation/Foundation.h>
#import <HealthKit/HealthKit.h>
@interface GSHealthKitManager : NSObject

+ (GSHealthKitManager *)sharedManager;

- (void)requestAuthorization;

- (NSDate *)readBirthDate;
- (void)writeWeightSample:(double)weight;
- (NSString *)readGender;

@end
```

GSHealthKitManager.m

```
#import "GSHealthKitManager.h"
#import <HealthKit/HealthKit.h>


@interface GSHealthKitManager ()

@property (nonatomic, retain) HKHealthStore *healthStore;

@end


@implementation GSHealthKitManager

+ (GSHealthKitManager *)sharedManager {
    static dispatch_once_t pred = 0;
    static GSHealthKitManager *instance = nil;
    dispatch_once(&pred, ^{
        instance = [[GSHealthKitManager alloc] init];
        instance.healthStore = [[HKHealthStore alloc] init];
    });
    return instance;
```

```
}

- (void)requestAuthorization {

    if ([HKHealthStore isHealthDataAvailable] == NO) {
        // If our device doesn't support HealthKit -> return.
        return;
    }

    NSArray *readTypes = @[[HKObjectType
characteristicTypeForIdentifier:HKCharacteristicTypeIdentifierDateOfBirth],[HKObjectType
characteristicTypeForIdentifier:HKCharacteristicTypeIdentifierBiologicalSex]];


    [self.healthStore requestAuthorizationToShareTypes:nil readTypes:[NSSet
setWithArray:readTypes] completion:nil];
}

- (NSDate *)readBirthDate {
    NSError *error;
    NSDate *dateOfBirth = [self.healthStore dateOfBirthWithError:&error];   // Convenience
method of HKHealthStore to get date of birth directly.

    if (!dateOfBirth) {
        NSLog(@"Either an error occured fetching the user's age information or none has been
stored yet. In your app, try to handle this gracefully.");
    }

    return dateOfBirth;
}

- (NSString *)readGender
{
    NSError *error;
    HKBiologicalSexObject *gen=[self.healthStore biologicalSexWithError:&error];
    if (gen.biologicalSex==HKBiologicalSexMale)
    {
        return(@"Male");
    }
    else if (gen.biologicalSex==HKBiologicalSexFemale)
    {
        return (@"Female");
    }
    else if (gen.biologicalSex==HKBiologicalSexOther)
    {
        return (@"Other");
    }
    else{
        return (@"Not Set");
    }
}



@end
```

Вызов из ViewController

```
- (IBAction)pressed:(id)sender {
```

```
    [[GSHealthKitManager sharedManager] requestAuthorization];
    NSDate *birthDate = [[GSHealthKitManager sharedManager] readBirthDate];
        NSLog(@"birthdate %@", birthDate);
        NSLog(@"gender 2131321 %@", [[GSHealthKitManager sharedManager] readGender]);


}
```

Выход журнала

```
2016-10-13 14:41:39.568 random[778:26371] birthdate 1992-11-29 18:30:00 +0000
2016-10-13 14:41:39.570 random[778:26371] gender 2131321 Male
```

Прочитайте Healthkit онлайн: https://riptutorial.com/ru/ios/topic/7412/healthkit

глава 33: IBeacon

параметры

параметры	подробности
менеджер	Ссылка на CLLocationManager
область, край	CLRegion может быть круговой областью (область геозонности или маяка)
маяки	Массив CLBeacon содержит все дальномерные маяки

замечания

Маяки - это объекты IOT. Мы фокусируемся на тех, которые соответствуют протоколу iBeacon, стандарту Apple. Каждый маяк - это одностороннее устройство, которое передает 3 вещи

1. UUID
2. Основной
3. Незначительный

Мы можем сканировать iBeacons, настроив наш объект менеджера CLLocation для сканирования маяков для определенного UUID. Будут проверены все маяки с данным UUID.

Менеджер CLLocation также дает вызов для входа и выхода из области маяка.

Examples

Основные операции iBeacon

1. Контрольные маяки настройки

```
func initiateRegion(ref:BeaconHandler){
    let uuid: NSUUID = NSUUID(UUIDString: "<UUID>")
    let beacon = CLBeaconRegion(proximityUUID: uuid, identifier: "")
    locationManager?.requestAlwaysAuthorization()    //cllocation manager obj.
    beacon?.notifyOnEntry = true
    beacon?.notifyOnExit = true
    beacon?.notifyEntryStateOnDisplay = true
    locationManager?.startMonitoringForRegion(beacon!)
    locationManager?.delegate = self;
    // Check if beacon monitoring is available for this device
    if (!CLLocationManager.isMonitoringAvailableForClass(CLBeaconRegion)) {
```

```
        print("error")
    }
    locationManager!.startRangingBeaconsInRegion(self.beacon!)
}
```

2. Ввод и выход из региона

```
func locationManager(manager: CLLocationManager, didEnterRegion region: CLRegion) {
    if(region.isKindOfClass(CLBeaconRegion)) {
        locationManager!.startRangingBeaconsInRegion(self.beacon!)
    }
}

func locationManager(manager: CLLocationManager, didExitRegion region: CLRegion) {
    if(region.isKindOfClass(CLBeaconRegion)) {
        locationManager!.stopRangingBeaconsInRegion(self.beacon!)
    }
}
```

3. Ближний маяк

```
func locationManager(manager: CLLocationManager, didRangeBeacons beacons: [CLBeacon], inRegion
region: CLBeaconRegion) {
    print(beacons.first.major)
}
```

Сканирование конкретных маяков

```
 beacon = CLBeaconRegion(proximityUUID: <#NSUUID#>, major: <#CLBeaconMajorValue#>, identifier:
<#String#>) // listening to all beacons with given UUID and major value
 beacon =    CLBeaconRegion(proximityUUID: <##NSUUID#>, major: <##CLBeaconMajorValue#>, minor:
<##CLBeaconMinorValue#>, identifier: <##String#>) // listening to all beacons with given UUID
and major and minor value
```

Изменение iBeacons

Во-первых, вы должны запросить авторизацию служб определения местоположения

```
let locationManager = CLLocationManager()
locationManager.delegate = self
locationManager.requestWhenInUseAuthorization()
// OR locationManager.requestAlwaysAuthorization()
```

Затем вы можете получить всю информацию `didRangeBeacons` внутри `didRangeBeacons`

```
func locationManager(manager: CLLocationManager, didRangeBeacons beacons: [CLBeacon], inRegion
region: CLBeaconRegion) {
    for beacon in beacons {
        print(beacon.major)
        print(beacon.minor)
    }
}
```

Прочитайте IBeacon онлайн: https://riptutorial.com/ru/ios/topic/1958/ibeacon

глава 34: IBOutlets

замечания

IBOutlet не является ни зарезервированным словом, ни переменной или классом, является синтаксическим сахаром для Interface Builder. После того, как исходный код Objective-C предварительно обработан, он ничего не разрешает.

В Swift это разрешено как ноль.

Он объявлен в `<UIKit/UINibDeclarations.h>` как

```
#ifndef IBOutlet
#define IBOutlet
#endif
```

Examples

Использование IBOutlet в элементе пользовательского интерфейса

В общем, IBOutlets используются для подключения объекта пользовательского интерфейса к другому объекту, в этом случае UIViewController. Соединение служит для того, чтобы объект мог повлиять на мой код или события программным путем. Это можно сделать, просто используя ассистента из раскадровки и щелкнув элемент управления с элемента на раздел свойств .h, а также можно программно и вручную подключить код IBOutlet к вкладке «Соединения» объекта панель служебных программ справа. Вот пример объектного примера UIViewController с выходом метки:

```
//ViewController.h
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

//This is the declaration of the outlet
@property (nonatomic, weak) IBOutlet UILabel *myLabel;

@end

//ViewController.m
#import "ViewController.h"

@implementation ViewController

@synthesize myLabel;

-(void) viewDidLoad {

    [super viewDidLoad];
```

```
    //Editing the properties of the outlet
    myLabel.text = @"TextHere";

}

@end
```

И быстро:

```
import UIKit
class ViewController: UIViewController {
    //This is the declaration of the outlet
    @IBOutlet weak var myLabel: UILabel!


    override func viewDidLoad() {
        super.viewDidLoad()
        //Editing the properties of the outlet
        myLabel.text = "TextHere"
    }
}
```

Связь между объектом раскадровки и запрограммированным объектом может быть проверена как подключенная, если точка слева от декларации выхода в .h заполнена. Пустой круг подразумевал неполное соединение.

Прочитайте IBOutlets онлайн: https://riptutorial.com/ru/ios/topic/4713/iboutlets

глава 35: iOS - Внедрение XMPP с платформой Робби Хэнсона

Examples

iOS XMPP Robbie Hanson Пример с Openfire

SRXMPPDemo

Загрузите пример и все классы здесь - https://github.com/SahebRoy92/SRXMPPDemo

Демонстрация на XMPP в Objective C с различными простыми и сложными функциями, реализованными в нем. Все функции XMPP выполняются функциями **«в полосе»** xmpp. Немногие функции, которые этот проект содержит:

SRXMPP - оболочка Singleton, которая почти имеет все функции, необходимые для **индивидуального** приложения чата.

- один-один чат
- Реализация ключевых данных чата (текстовое сообщение), таким образом сохраняя предыдущие сообщения, автономные сообщения.
- внедрение vCard (информация о профиле пользователя, собственного и других) из XML и основных данных, предоставленных собственными рамками Робби Хэнсона.
- наличие статуса друзей (онлайн / офлайн / ввод текста)

Шаги, чтобы следовать

Вы хотите использовать этот проект в качестве ссылки, тогда вы можете сделать следующее:

1. Установленный Openfire на реальном сервере. Аренда сервера, установка openfire.

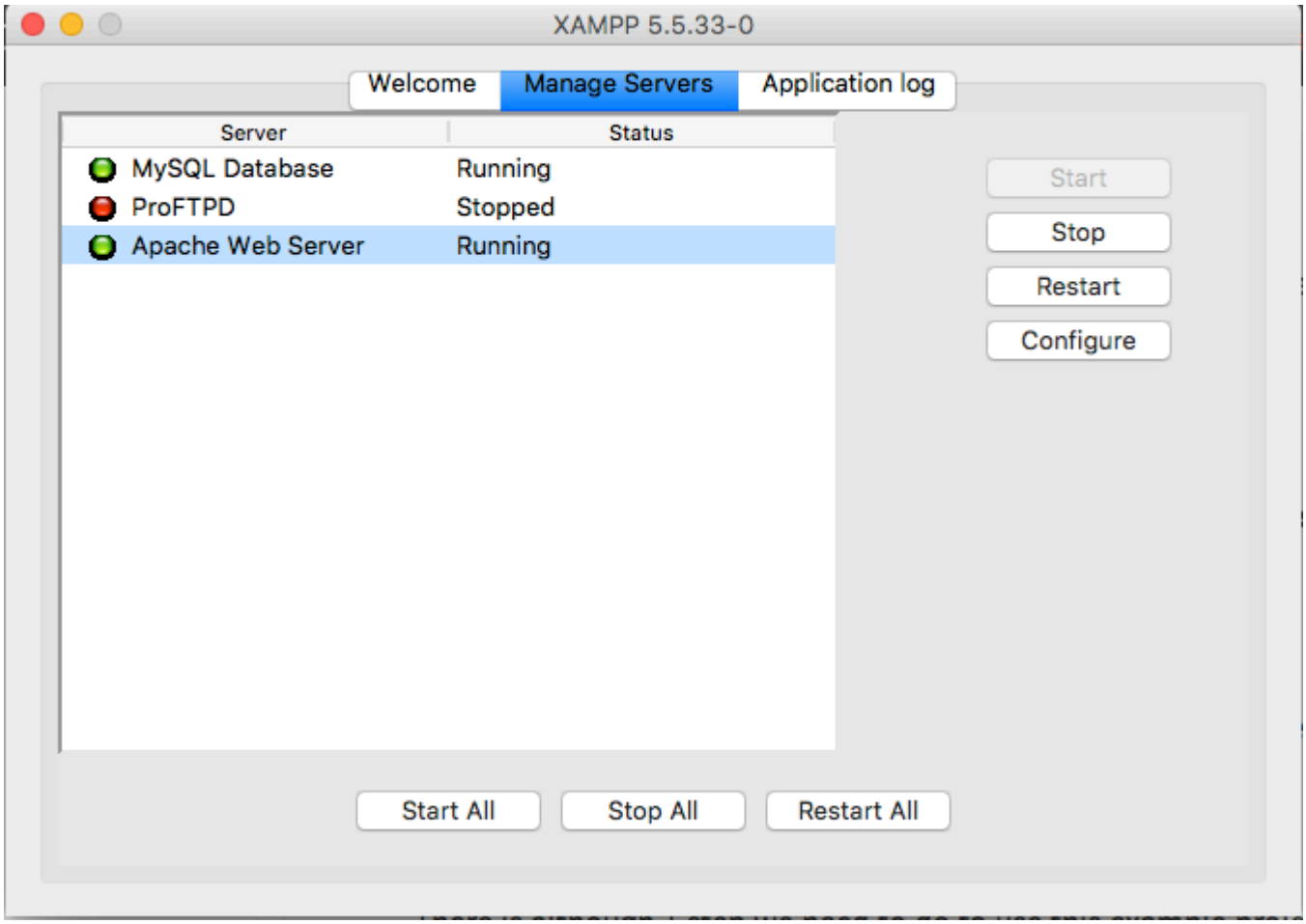
2. Хотите попробовать это без проблем в своем собственном компьютере. Вам нужно загрузить, установить и настроить 3 вещи, чтобы начать

а. Джава -

- Загрузите и установите Java для Mac.

б. XAMPP -

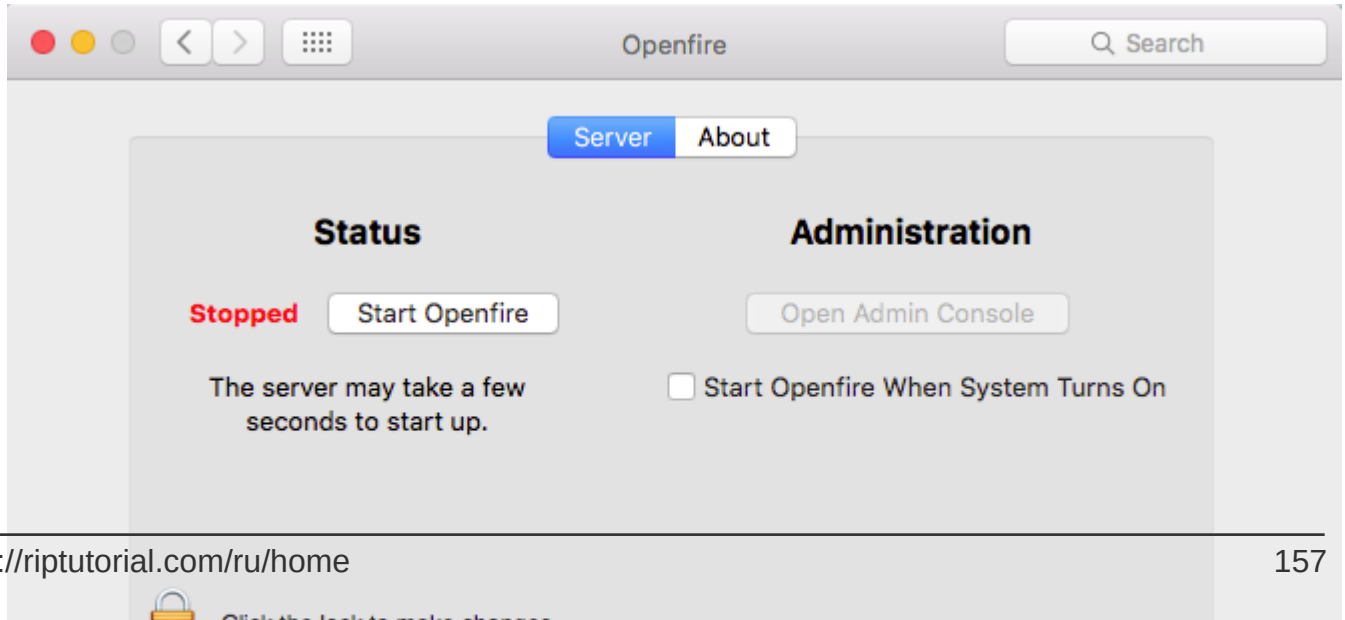
- Установить XAMPP относительно легко.
- После установки просто запустите XAMPP и запустите **Database (SQL)** и **Apache Server** .



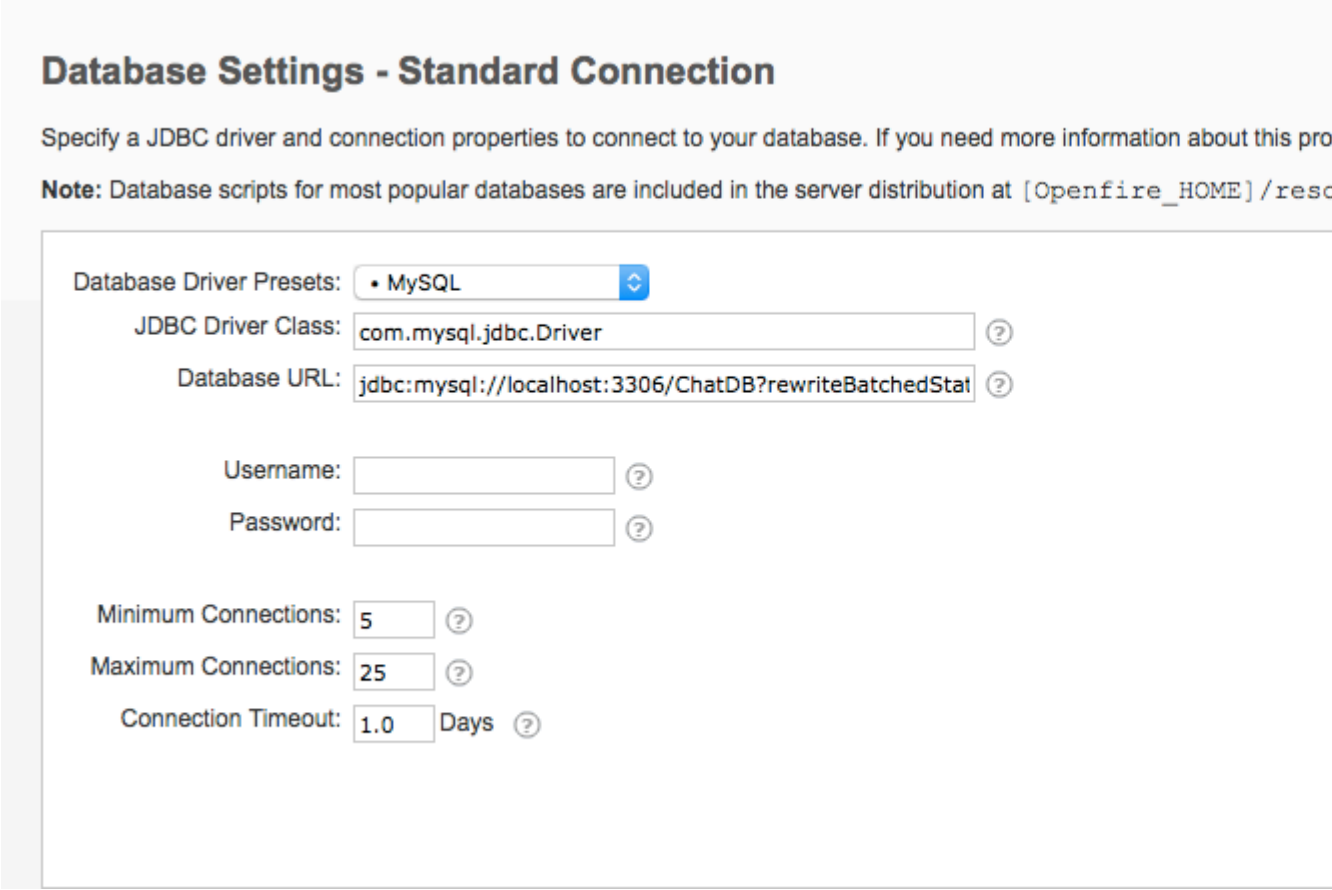
- Затем откройте браузер и вставьте этот URL *[http: // localhost / phpmyadmin /]*
- , Создайте новую БД с левой стороны панели.
- **Назовите DB ничего, кроме как помните это имя, предположим, что мы назовем его ChatDB**

c. Openfire -

- Установите Openfire и запустите приложение и «Запустите Openfire»



- Открыть браузер и вставить этот URL-адрес - [http: // localhost: 9090 / setup / index.jsp] (http: // localhost: 9090 / setup / index.jsp)
- Сделайте обычную настройку
  - Выберите язык>
  - Настройки сервера, оставьте как есть, просто продолжайте>
  - Настройки базы данных, оставьте это как «Стандартное подключение к базе данных как выбранное»>
  - Настройки базы данных - стандартное соединение ". Теперь запомните имя базы данных, которую вы установили, - это **ChatDB** .
  - Выберите префикс драйвера базы данных как * *« MySQL»* . Оставьте класс драйвера JDBC как есть. Теперь в URL-адресе базы данных вы можете видеть, скобки упоминают имя хоста и имя базы данных. Просто измените имя хоста на **«localhost»** и имя базы данных на **«ChatDB»** или любое другое имя базы данных, которое вы установили ранее, при настройке XAMPP. Оставьте имя пользователя и пароль пустым. Заполните данные, подобные изображению здесь.



,

  - Затем выполните полную настройку, указав имя пользователя и пароль и подтвердив это. Это делается. Настройка Openfire.

Теперь эта часть приходит, когда вам нужно изменить крошечную деталь в коде.

Важно. Нам нужно перейти в класс - **SRXMPP.m** , найти NSString extern

SRXMPP_Hostname (в верхней части) и перезаписать его значение

- IP сервера, на котором установлен OpenFire, **ИЛИ**
- если вы его локально установили, замените значение на **«localhost»** .

То есть, вы готовы использовать этот примерный проект и начать кодирование и превратить его в лучший собственный проект.

Этот стартовый пакет поможет вам лучше понять структуру XMPP, а также получить доступ к протоколам XMPP.

Вы можете найти другие протоколы XMPP здесь, на этом сайте - https://xmpp.org/rfcs/rfc3920.html

Развитие все еще остается, и части, где я надеюсь включить их позже,

1. Групповой чат
2. Поддержка отправки изображений

Короче, этот примерный проект вместе с синглтоном имеет почти все функции, которые необходимы для приложения чата One-to-One.

Прочитайте iOS - Внедрение XMPP с платформой Робби Хэнсона онлайн: https://riptutorial.com/ru/ios/topic/1475/ios---внедрение-xmpp-с-платформой-робби-хэнсона

глава 36: iOS TTS

Вступление

Как создать синтезированную речь из текста на устройстве iOS

Examples

Текст в речь

Цель C

```
AVSpeechSynthesizer *synthesizer = [[AVSpeechSynthesizer alloc]init];
AVSpeechUtterance *utterance = [AVSpeechUtterance speechUtteranceWithString:@"Some text"];
[utterance setRate:0.2f];
[synthesizer speakUtterance:utterance];
```

стриж

```
let synthesizer = AVSpeechSynthesizer()
let utterance = AVSpeechUtterance(string: "Some text")
utterance.rate = 0.2
```

Вы также можете изменить голос следующим образом:

```
utterance.voice = AVSpeechSynthesisVoice(language: "fr-FR")
```

И затем загляните

- В Swift 2: `synthesizer.speakUtterance(utterance)`
- В Swift 3: `synthesizer.speak(utterance)`

Не забудьте импортировать AVFoundation

Полезные методы

Вы можете остановить или приостановить всю речь, используя эти два метода:

```
- (BOOL)pauseSpeakingAtBoundary:(AVSpeechBoundary)boundary;
- (BOOL)stopSpeakingAtBoundary:(AVSpeechBoundary)boundary;
```

AVSpeechBoundary указывает, должна ли речь останавливаться или останавливаться

немедленно (`AVSpeechBoundaryImmediate`), или она должна приостанавливаться или останавливаться после слова, произносимого в настоящее время (`AVSpeechBoundaryWord`).

Прочитайте iOS TTS онлайн: https://riptutorial.com/ru/ios/topic/8909/ios-tts

глава 37: MKDistanceFormatter

Examples

Строка с расстояния

Учитывая `CLLocationDistance` (просто `Double` представляющий `CLLocationDistance`), выведите строку, читаемую пользователем:

```
let distance = CLLocationDistance(42)
let formatter = MKDistanceFormatter()
let answer = formatter.stringFromDistance(distance)
// answer = "150 feet"
```

Objective-C

```
CLLocationDistance distance=42;
MKDistanceFormatter *formatter=[[MKDistanceFormatter alloc]init];
NSString *answer=[formatter stringFromDistance:distance];
// answer = "150 feet"
```

По умолчанию это соответствует языку пользователя.

Расстояние

`import Mapkit` Устанавливает `units` в один из `.Default, .Metric, .Imperial, .ImperialWithYards` :

```
formatter.units = .Metric
var answer = formatter.stringFromDistance(distance)
// "40 m"

formatter.units = .ImperialWithYards
answer = formatter.stringFromDistance(distance)
// "50 yards"
```

Objective-C

```
MKDistanceFormatter *formatter=[[MKDistanceFormatter alloc]init];
formatter.units=MKDistanceFormatterUnitsMetric;
NSString *answer=[formatter stringFromDistance:distance];
//40 m

formatter.units=MKDistanceFormatterUnitsImperialWithYards;
NSString *answer=[formatter stringFromDistance:distance];
//50 yards
```

Тип единицы

Установите `unitStyle` в один из `.Default, .Abbreviated, .Full` :

```
formatter.unitStyle = .Full
var answer = formatter.stringFromDistance(distance)
// "150 feet"

formatter.unitStyle = .Abbreviated
answer = formatter.stringFromDistance(distance)
// "150 ft"
```

Objective-C

```
formatter.unitStyle=MKDistanceFormatterUnitStyleFull;
NSString *answer=[formatter stringFromDistance:distance];
// "150 feet"

 formatter.unitStyle=MKDistanceFormatterUnitStyleAbbreviated;
 NSString *answer=[formatter stringFromDistance:distance];
// "150 ft"
```

Прочитайте MKDistanceFormatter онлайн:
https://riptutorial.com/ru/ios/topic/6677/mkdistanceformatter

глава 38: MKMapView

Examples

Добавить MKMapView

стриж

```
let mapView = MKMapView(frame: CGRect(x: 0, y: 0, width: 320, height: 500))
```

Рекомендуется хранить mapView как свойство содержащего `ViewController` так как вы можете захотеть получить к нему доступ в более сложных реализациях.

Цель C

```
self.map = [[MKMapView alloc]initWithFrame:CGRectMake(0, 0, self.view.frame.size.width,
self.view.frame.size.height)];
[self.view addSubview:self.map];
```

Изменить тип карты

Существует 5 различных типов (`MKMapType`), `MKMapView` может отображать.

iPhone OS 3

.Стандартный

Отображает карту улиц, в которой указаны позиции всех дорог и названия дорог.

Swift 2

```
mapView.mapType = .Standard
```

Swift 3

```
mapView.mapType = .standard
```

Objective-C

```
_mapView.mapType = MKMapTypeStandard;
```

iPhone OS 3

.спутник

Отображает спутниковые снимки области.

Swift 2

```
mapView.mapType = .Satellite
```

Swift 3

```
mapView.mapType = .satellite
```

Objective-C

```
_mapView.mapType = MKMapTypeSatellite;
```

iOS 9

.satelliteFlyover

Отображает изображение спутника области с данными эстакады, если таковые имеются.

Swift 2

```
mapView.mapType = .SatelliteFlyover
```

Swift 3

```
mapView.mapType = .satelliteFlyover
```

Objective-C

```
_mapView.mapType = MKMapTypeSatelliteFlyover;
```

iPhone OS 3

.гибридный

Отображает изображение спутника области с информацией о названии дороги и дороги, расположенной сверху.

Swift 2

```
mapView.mapType = .Hybrid
```

Swift 3

```
mapView.mapType = .hybrid
```

Objective-C

```
_mapView.mapType = MKMapTypeHybrid;
```

iOS 9

.hybridFlyover

Отображает гибридное изображение спутника с данными эстакады, если таковые имеются.

Swift 2

```
mapView.mapType = .HybridFlyover
```

Swift 3

```
mapView.mapType = .hybridFlyover
```

Objective-C

```
_mapView.mapType = MKMapTypeHybridFlyover;
```

Установить масштаб / регион для карты

Чтобы установить некоторый уровень масштабирования, скажем, мы хотим увеличить местоположение пользователя с местоположением пользователя в центре и 2 км от площади в радиусе. Затем мы используем следующий код

```
MKUserLocation *userLocation = _mapView.userLocation;
MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance
(userLocation.location.coordinate, 2000, 2000);
[_mapView setRegion:region animated:NO];
```

Реализация локального поиска с использованием MKLocalSearch

MKLocalSearch позволяет пользователям искать местоположение, используя строки естественного языка, такие как «тренажерный зал». Как только поиск будет завершен, класс возвращает список местоположений в пределах указанной области, которая соответствует строке поиска.

Результаты поиска в форме MKMapItem в объекте MKLocalSearchResponse.

попробуем попробовать

```
MKLocalSearchRequest *request =
    [[MKLocalSearchRequest alloc] init];//initialising search request
request.naturalLanguageQuery = @”Gym”; // adding query
request.region = _mapView.region; //setting region
MKLocalSearch *search =
     [[MKLocalSearch alloc]initWithRequest:request];//initiate search

[search startWithCompletionHandler:^(MKLocalSearchResponse
     *response, NSError *error)
{
   if (response.mapItems.count == 0)
       NSLog(@"No Matches");
   else
       for (MKMapItem *item in response.mapItems)
       {
           NSLog(@"name = %@", item.name);
           NSLog(@"Phone = %@", item.phoneNumber);
        }
}];
```

OpenStreetMap Tile-Overlay

В некоторых случаях вы, возможно, не захотите использовать карты по умолчанию, предоставляемые Apple.

Вы можете добавить наложение в свой `mapView` который содержит пользовательские плитки, например, из *OpenStreetMap* .

Предположим, `self.mapView` - это ваш `MKMapView` который вы уже добавили в свой `ViewController` .

Сначала ваш `ViewController` должен соответствовать протоколу `MKMapViewDelegate` .

```
class MyViewController: UIViewController, MKMapViewDelegate
```

Затем вы должны установить `ViewController` качестве делегата `mapView`

```
mapView.delegate = self
```

Затем вы настраиваете наложение для карты. Для этого вам понадобится шаблон URL. URL-адрес должен быть похож на это на всех серверах tile-серверов, и даже если вы сохраните данные карты в автономном режиме: `http://tile.openstreetmap.org/{z}/{x}/{y}.png`

```
let urlTeplate = "http://tile.openstreetmap.org/{z}/{x}/{y}.png"
let overlay = MKTileOverlay(urlTemplate: urlTeplate)
overlay.canReplaceMapContent = true
```

После настройки наложения вы должны добавить его в свой `mapView` .

```
mapView.add(overlay, level: .aboveLabels)
```

Чтобы использовать пользовательские карты, рекомендуется использовать `.aboveLabels` для `level` . В противном случае метки по умолчанию будут видны на вашей пользовательской карте. Если вы хотите увидеть метки по умолчанию, вы можете выбрать здесь `.aboveRoads` .

Если вы сейчас запустите свой проект, вы поймете, что ваша карта по-прежнему будет отображать карту по умолчанию:

Это потому, что мы еще не сказали `mapView` , как визуализировать наложение. Вот почему вы должны были установить делегат раньше. Теперь вы можете добавить `func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer` для вашего контроллера:

```
func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {
    if overlay is MKTileOverlay {
        let renderer = MKTileOverlayRenderer(overlay: overlay)
        return renderer
    } else {
        return MKTileOverlayRenderer()
    }
}
```

Это вернет правильный `MKOverlayRenderer` в ваш `mapView` . Если вы запустите свой проект сейчас, вы должны увидеть такую карту:

Если вы хотите отобразить другую карту, вам просто нужно изменить шаблон URL. В Wiki OSM есть список tile-серверов .

Показать пример UserLocation и UserTracking

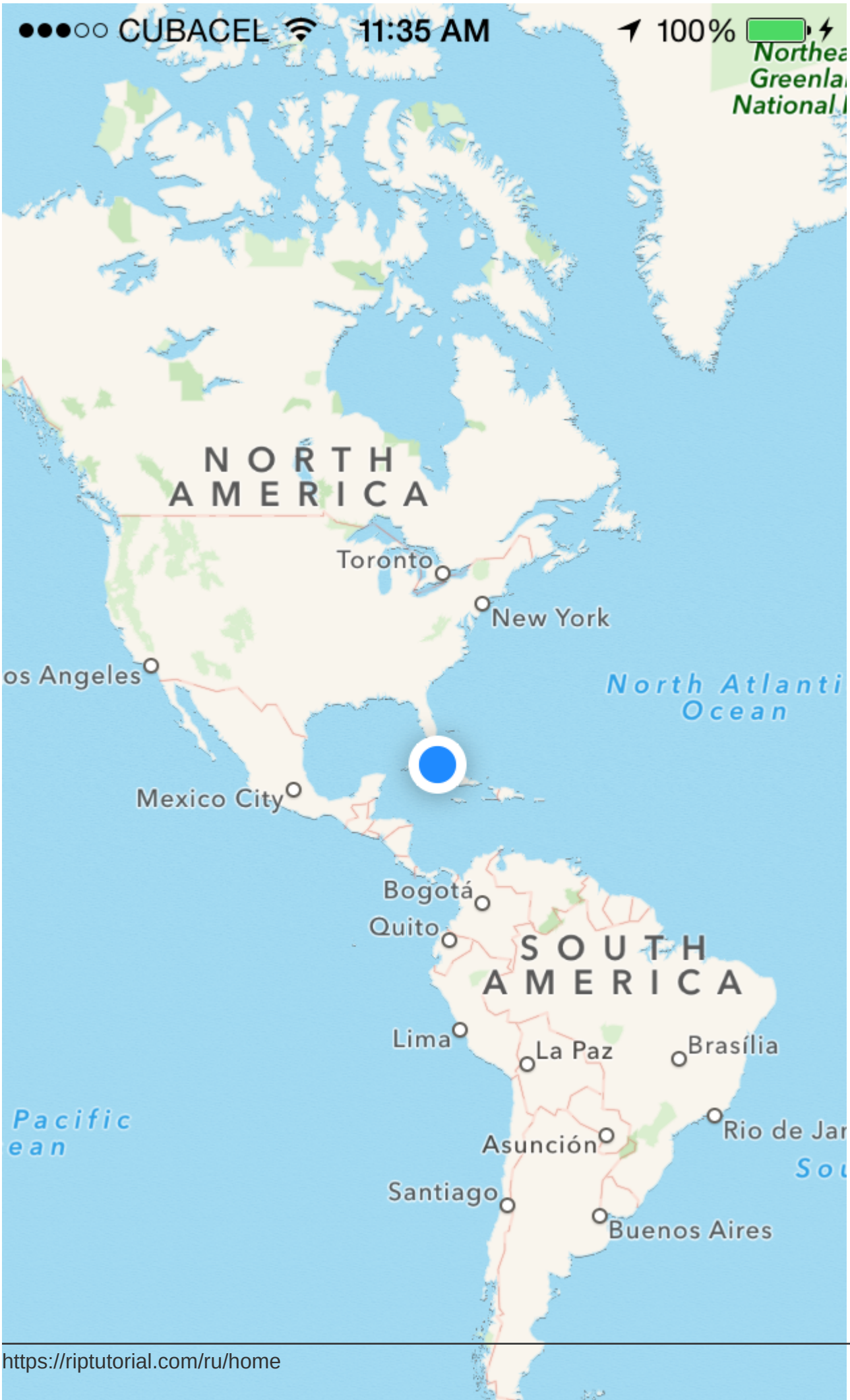
Это покажет местоположение пользователя на карте

Objective-C

```
[self.map setShowsUserLocation:YES];
```

стриж

```
self.map?.showsUserLocation = true
```


CUBACEL
11:35 AM
100%
NORTH AMERICA
Toronto
New York
os Angeles
North Atlanti
Ocean
Mexico City
Bogotá
Quito
SOUTH AMERICA
Lima
La Paz
Brasília
Pacific
ean
Asunción
Rio de Jar
Santiago
Buenos Aires

аннотацию. Теперь сначала создайте объект аннотации.

```
MKPointAnnotation *pointAnnotation = [[MKPointAnnotation alloc] init];
```

Теперь предоставим координату pointAnnotation, так как

```
CLLocationCoordinate2D coordinate = CLLocationCoordinate2DMake(23.054625,72.534562);
pointAnnotation.coordinate = coordinate;
```

Теперь укажите заголовок и субтитры для аннотации,

```
pointAnnotation.title = @"XYZ Point";
pointAnnotation.subtitle = @"Ahmedabad Area";
```

Теперь добавьте эту аннотацию к карте.

```
[self.mapView addAnnotation:pointAnnotation];
```

Да ... Ура. Ты сделал это. Теперь вы можете увидеть аннотацию точки (красный цветной контакт) при заданной координате.

Но теперь, если вы хотите изменить цвет штифта (3 доступных цвета - фиолетовый, красный и зеленый). Затем следуйте этому шагу.

назначить делегата mapview самому себе,

```
self.mapView.delegate = self;
```

Добавьте реализацию MKMapViewDelegate. Теперь добавьте следующий метод,

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView viewForAnnotation:(id
<MKAnnotation>)annotation
{
    // If it's the user location, just return nil, because it have user location's own
annotation, if you want to change that, then use this object;
    if ([annotation isKindOfClass:[MKUserLocation class]])
        return nil;

    if ([annotation isKindOfClass:[MKPointAnnotation class]])
    {
        //Use dequed pin if available
        MKAnnotationView *pinView = [mapView
dequeueReusableAnnotationViewWithIdentifier:@"PinAnnotationView"];

        if (!pinView)
        {
            // If not dequed, then create new.
            pinView = [[MKAnnotationView alloc] initWithAnnotation:annotation
reuseIdentifier:@"PinAnnotationView"];
            pinView.canShowCallout = YES;
            pinView.image = [UIImage imageNamed:@"abc.png"];
            pinView.calloutOffset = CGPointMake(0, 32);
```

```
        } else {
            pinView.annotation = annotation;
        }
        return pinView;
    }
    return nil;
}
```

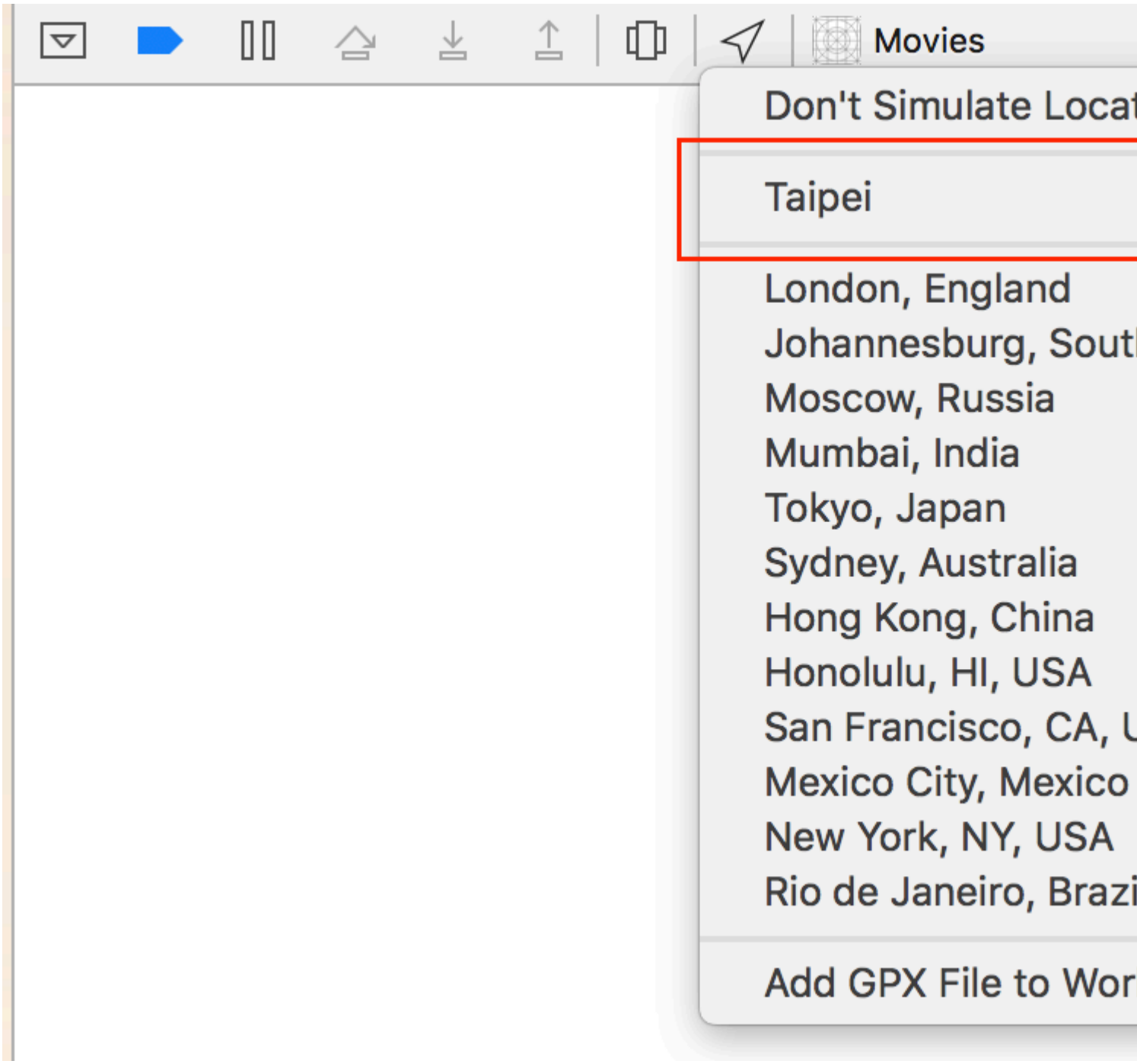
Имитировать настраиваемое местоположение

Шаг 1: В Xcode: Файл -> Создать -> Файл -> Ресурс -> Файл GPX -> Далее -> Дайте GPX-файлу имя (это в этом примере `Taipei`) -> Создать

Шаг 2. Редактирование файла GPX.

```
<?xml version="1.0"?>
<gpx version="1.1" creator="Xcode">
    <wpt lat="25.041865" lon="121.551361"> // Edit the latitude and longitude
        <name>Taipei</name> // Edit the name of the location
        <time>2014-09-24T14:55:37Z</time>
    </wpt>
</gpx>
```

Шаг 3: Когда симулятор запущен:



Вы можете повторить этот процесс для создания нескольких местоположений.

Выделите координаты и масштаб

Когда вы показываете местоположение своим пользователям, вы можете захотеть, чтобы `MKMapView` отображал координату на уровне масштабирования, а не настраивал регион для отображения. Эта функциональность не реализована по умолчанию, поэтому вам необходимо расширить `MKMapView` с помощью методов, которые выполняют комплексный расчет с *координаты* и *уровня масштабирования* до `MKCoordinateRegion` .

```
let MERCATOR_OFFSET = 268435456.0
let MERCATOR_RADIUS = 85445659.44705395
let DEGREES = 180.0
```

```
public extension MKMapView {

    //MARK: Map Conversion Methods

    private func longitudeToPixelSpaceX(longitude:Double)->Double{
        return round(MERCATOR_OFFSET + MERCATOR_RADIUS * longitude * M_PI / DEGREES)
    }

    private func latitudeToPixelSpaceY(latitude:Double)->Double{
        return round(MERCATOR_OFFSET - MERCATOR_RADIUS * log((1 + sin(latitude * M_PI /
DEGREES)) / (1 - sin(latitude * M_PI / DEGREES))) / 2.0)
    }

    private func pixelSpaceXToLongitude(pixelX:Double)->Double{
        return ((round(pixelX) - MERCATOR_OFFSET) / MERCATOR_RADIUS) * DEGREES / M_PI
    }

    private func pixelSpaceYToLatitude(pixelY:Double)->Double{
        return (M_PI / 2.0 - 2.0 * atan(exp((round(pixelY) - MERCATOR_OFFSET) /
MERCATOR_RADIUS))) * DEGREES / M_PI
    }

    private func coordinateSpanWithCenterCoordinate(centerCoordinate:CLLocationCoordinate2D,
zoomLevel:Double)->MKCoordinateSpan{
        // convert center coordiate to pixel space
        let centerPixelX = longitudeToPixelSpaceX(longitude: centerCoordinate.longitude)
        let centerPixelY = latitudeToPixelSpaceY(latitude: centerCoordinate.latitude)
        print(centerCoordinate)
        // determine the scale value from the zoom level
        let zoomExponent:Double = 20.0 - zoomLevel
        let zoomScale:Double = pow(2.0, zoomExponent)
        // scale the map's size in pixel space
        let mapSizeInPixels = self.bounds.size
        let scaledMapWidth = Double(mapSizeInPixels.width) * zoomScale
        let scaledMapHeight = Double(mapSizeInPixels.height) * zoomScale
        // figure out the position of the top-left pixel
        let topLeftPixelX = centerPixelX - (scaledMapWidth / 2.0)
        let topLeftPixelY = centerPixelY - (scaledMapHeight / 2.0)
        // find delta between left and right longitudes
        let minLng = pixelSpaceXToLongitude(pixelX: topLeftPixelX)
        let maxLng = pixelSpaceXToLongitude(pixelX: topLeftPixelX + scaledMapWidth)
        let longitudeDelta = maxLng - minLng
        let minLat = pixelSpaceYToLatitude(pixelY: topLeftPixelY)
        let maxLat = pixelSpaceYToLatitude(pixelY: topLeftPixelY + scaledMapHeight)
        let latitudeDelta = -1.0 * (maxLat - minLat)
        return MKCoordinateSpan(latitudeDelta: latitudeDelta, longitudeDelta: longitudeDelta)
    }

    /**
     Sets the center of the `MKMapView` to a `CLLocationCoordinate2D` with a custom zoom-
level. There is no nee to set a region manually. :-)

     - author: Mylene Bayan (on GitHub)
     */
    public func setCenter(_ coordinate:CLLocationCoordinate2D, zoomLevel:Double,
animated:Bool){
        // clamp large numbers to 28
        var zoomLevel = zoomLevel
        zoomLevel = min(zoomLevel, 28)
        // use the zoom level to compute the region
```

```
        print(coordinate)
        let span = self.coordinateSpanWithCenterCoordinate(centerCoordinate: coordinate,
zoomLevel: zoomLevel)
        let region = MKCoordinateRegionMake(coordinate, span)
        if region.center.longitude == -180.00000000{
            print("Invalid Region")
        }
        else{
            self.setRegion(region, animated: animated)
        }
    }
}
```

(Оригинальную версию Swift 2 от Mylene Bayan можно найти на GitHub)

После того, как вы внесете это `extension` , вы можете установить центральную координату следующим образом:

```
let centerCoordinate = CLLocationCoordinate2DMake(48.136315, 11.5752901) //latitude, longitude
mapView?.setCenter(centerCoordinate, zoomLevel: 15, animated: true)
```

`zoomLevel` - это `Double` значение, обычно от `0` до `21` (что является очень высоким уровнем масштабирования), но допустимы значения до `28` .

Работа с аннотацией

Получить всю аннотацию

```
//following method returns all annotations object added on map
NSArray *allAnnotations = mapView.annotations;
```

Просмотр аннотаций

```
for (id<MKAnnotation> annotation in mapView.annotations)
{
    MKAnnotationView* annotationView = [mapView viewForAnnotation:annotation];
    if (annotationView)
    {
       // Do something with annotation view
       // for e.g change image of annotation view
       annotationView.image = [UIImage imageNamed:@"SelectedPin.png"];
    }
}
```

Удалить все аннотации

```
[mapView removeAnnotations:mapView.annotations]
```

Удалить отдельную аннотацию

```
//getting all Annotation
```

```
NSArray *allAnnotations = self.myMapView.annotations;

if (allAnnotations.count > 0)
{
    //getting first annoation
    id <MKAnnotation> annotation=[allAnnotations firstObject];

    //removing annotation
    [mapView removeAnnotation:annotation];

}
```

Отрегулируйте видимый прямоугольник вида карты, чтобы отобразить все аннотации

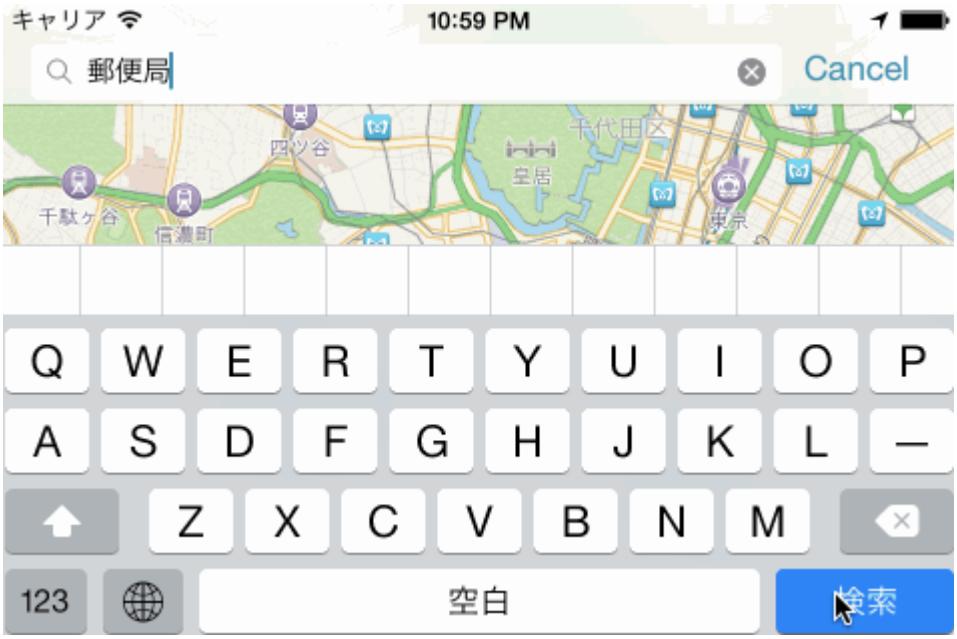
Swift:

```
mapView.showAnnotations(mapView.annotations, animated: true)
```

Objective-C:

```
[mapView showAnnotations:mapView.annotations animated:YES];
```

Демо-версия:



Прочитайте MKMapView онлайн: https://riptutorial.com/ru/ios/topic/915/mkmapview

глава 39: ModelPresentationStyles

Вступление

Модальные стили презентации используются, когда вы переходите от одного контроллера представления к другому. Существует два способа достижения этой настройки. Один - через код, а другой через Interface Builder (с использованием segues). Этот эффект достигается установкой переменной `modalPresentationStyle` в экземпляр `UIModalPresentationStyle` enum. `modalPresentationStyle` представляет собой переменную класса `UIViewController` и используется для указания способа `ViewController` на экране.

замечания

Всегда помните следующее упоминание от Apple.

> В горизонтально компактной среде контроллеры модального просмотра всегда отображаются в полноэкранном режиме. В горизонтально-обычной среде существует несколько различных вариантов представления.

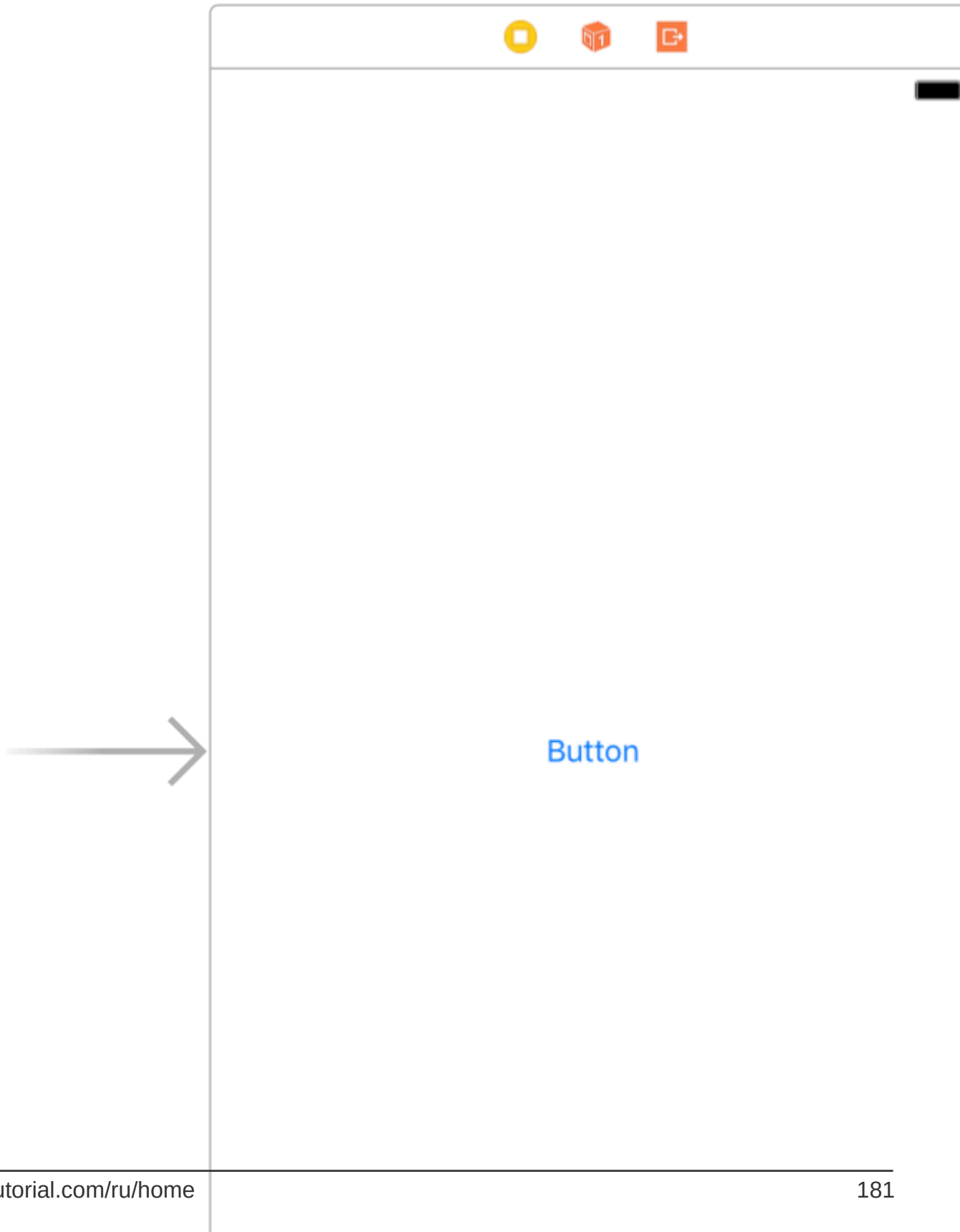
Examples

Изучение ModalPresentationStyle с помощью Interface Builder

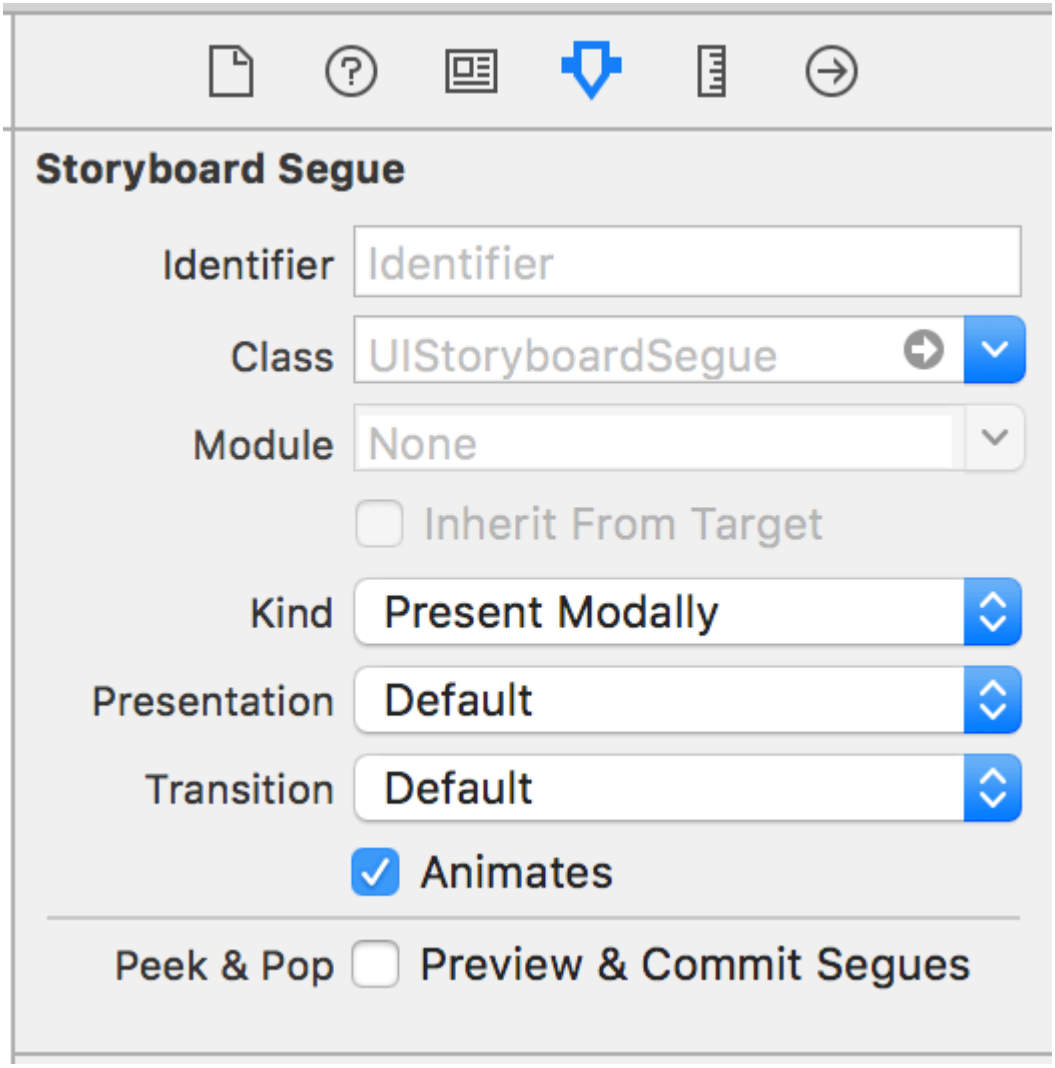
Это будет очень простое приложение, которое будет иллюстрировать разные `ModalpresentationStyle` в iOS. В соответствии с документацией, найденной здесь , существует 9 различных значений для `UIModalPresentationStyle` которые следующие:

1. `fullScreen`
2. `pageSheet`
3. `formSheet`
4. `currentContext`
5. `custom`
6. `overFullScreen`
7. `overCurrentContext`
8. `popover`
9. `none`

Чтобы настроить проект, просто создайте обычный проект iOS и добавьте 2 `ViewControllers` . Поместите `UIButton` в исходный `ViewController` и подключите его к 2-му `ViewController` помощью механизма `Target -> Action` . Чтобы отличить оба `ViewControllers` , установите для свойства фона `UIView` в `ViewController` другой цвет. Если все пойдет хорошо, ваш интерфейс Builder должен выглядеть так,


Button

подробности о том, почему iPad см. В разделе «Примечания»). Когда вы закончите настройку своего проекта, выберите сегю и перейдите к `attributes inspector` . Вы должны уметь видеть что-то вроде этого,



Установите свойство вида « `Present Modally` .

Теперь мы не увидим все эффекты в этом примере, так как некоторые из них требуют немного кода.

Начнем с `fullscreen` . Этот эффект выбирается по умолчанию при выборе `Present Modally` в `Kind` вкладки. Когда вы будете строить и запускать, второй `ViewController` будет занимать весь экран iPad.


iPad Air – iOS 10.3 (14E269)
Carrier
6:11 PM

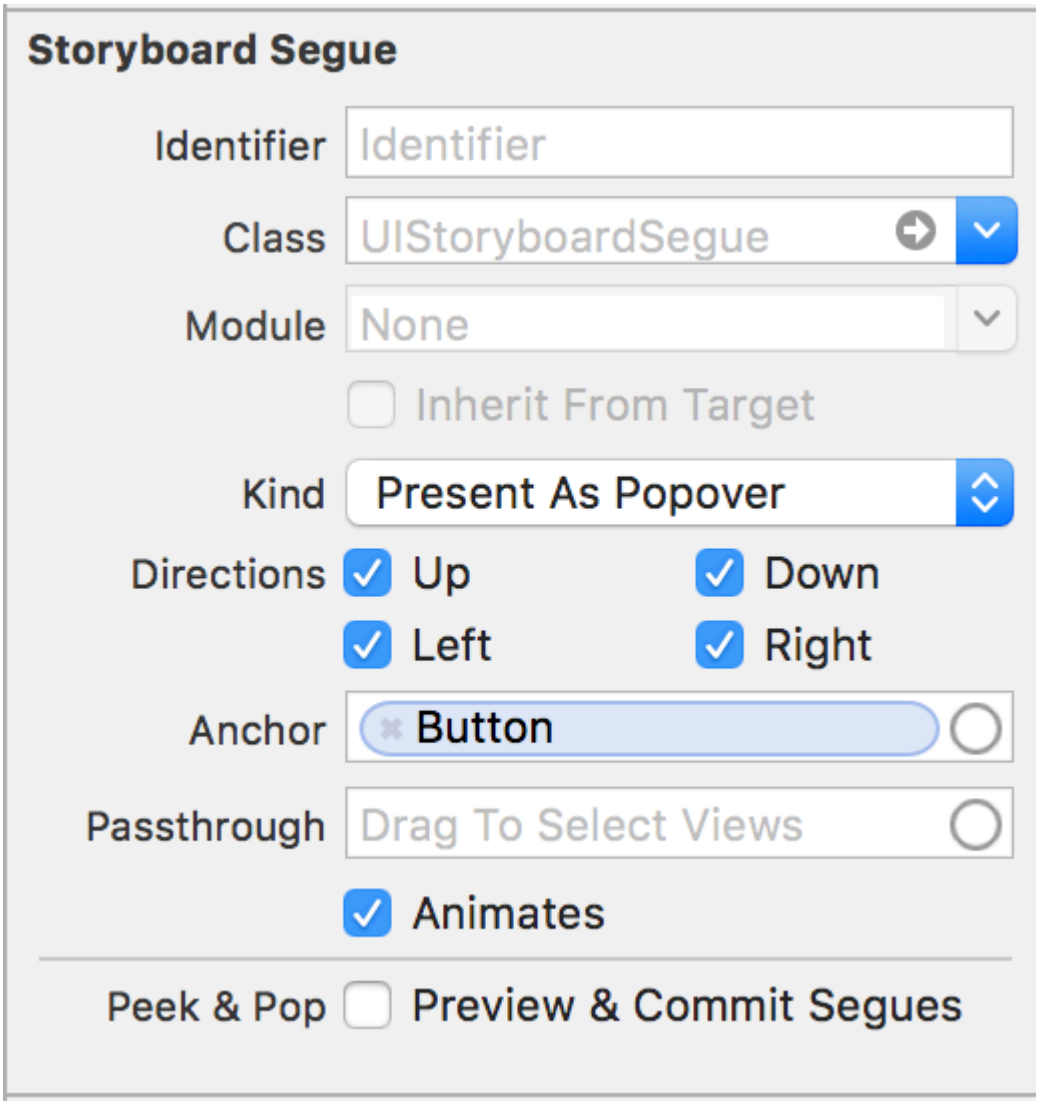
В этом случае, когда устройство находится в портретном режиме, второй `ViewController` аналогичен полноэкранному, но в ландшафтном режиме, второй `ViewController` значительно сужает ширину устройства. Кроме того, любой контент, который не распространяется на 2-й `ViewController` будет недоступен.


iPad Air – iO
Carrier

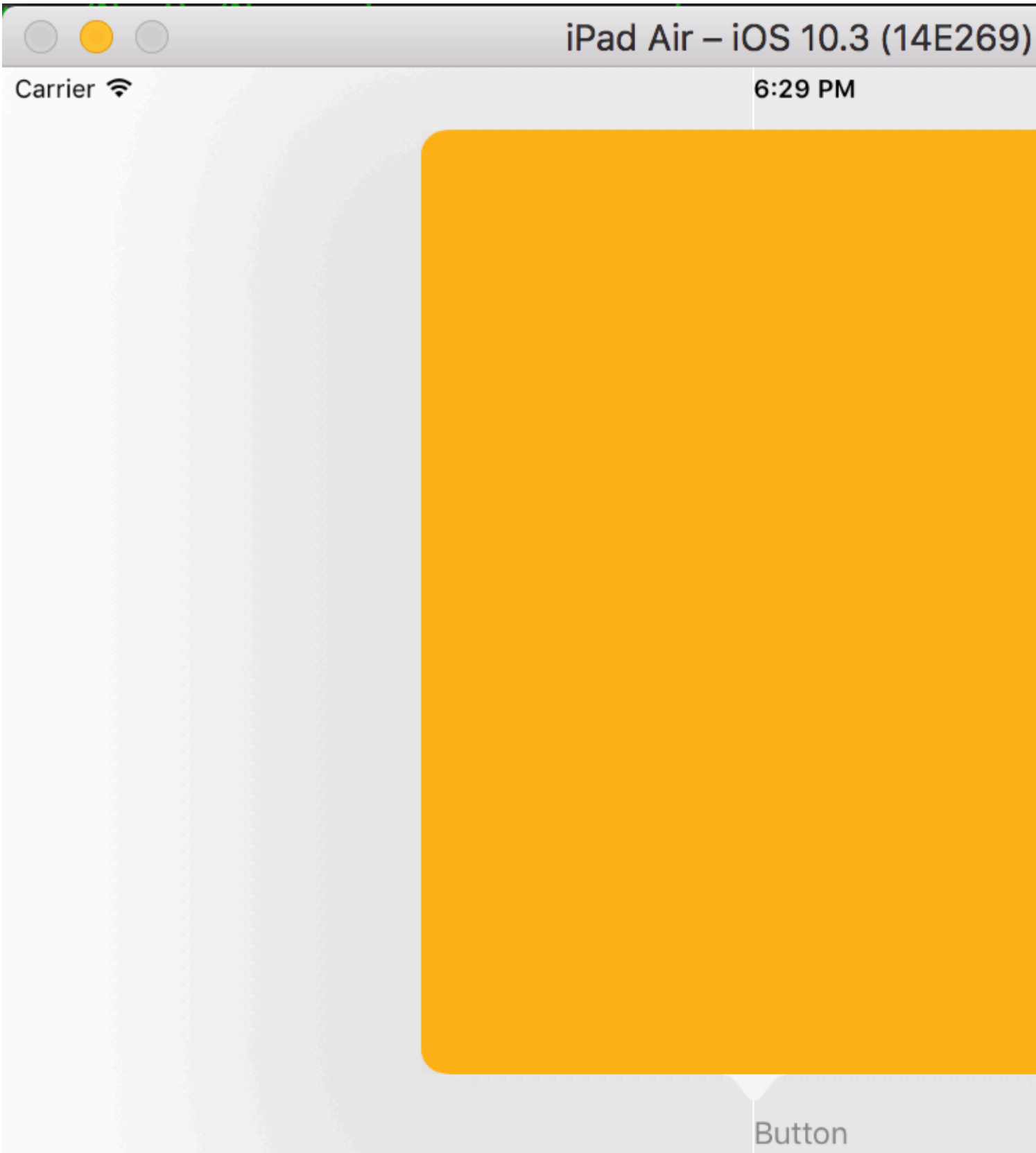
размещается в центре устройства, а размер меньше, чем у устройства. Также, когда устройство находится в ландшафтном режиме, а клавиатура видна, позиция просмотра настраивается вверх, чтобы показать `ViewController` .


iPad Air – iOS 10.3 (14E269)
Carrier
6:23 PM

Чтобы выбрать этот стиль, выберите « `Present as Popover` в `Kind` ». Второй `ViewController` представлен как небольшой popover (размер может быть установлен). Фоновый контент затемнен. Любой щелчок за пределами popover отклонил бы popover. `Attributes Inspector` ваших `Attributes Inspector` должен выглядеть примерно так:



`Anchor` - это элемент пользовательского интерфейса, на который вы хотите указать стрелку навигации. `Directions` - это направления, по которым вы можете указать ваш `Anchor` .


iPad Air – iOS 10.3 (14E269)
Carrier
6:29 PM
Button

https://riptutorial.com/ru/ios/topic/10122/modelpresentationstyles

глава 40: MPMediaPickerDelegate

замечания

Дополнительную информацию о конфиденциальности см. В документации Apple .

Убедитесь, что приложение «Музыка» доступно на вашем iPhone. Он не будет работать в симуляторе.

Examples

Загрузите музыку с помощью MPMediaPickerControllerDelegate и воспроизведите ее с помощью AVAudioPlayer

Пройдите шаги:

- Добавьте «NSAppleMusicUsageDescription» в свой Info.plist для органа конфиденциальности.
- Убедитесь, что ваша музыка доступна на вашем iPhone. Он не будет работать в симуляторе.

iOS 10.0.1

```
import UIKit
import AVFoundation
import MediaPlayer

class ViewController: UIViewController, MPMediaPickerControllerDelegate {

    var avMusicPlayer: AVAudioPlayer!
    var mpMediapicker: MPMediaPickerController!
    var mediaItems = [MPMediaItem]()
    let currentIndex = 0

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    func audioPlayerDidFinishPlaying(_ player: AVAudioPlayer, successfully flag: Bool){
        //What to do?
    }

    func mediaPicker(_ mediaPicker: MPMediaPickerController, didPickMediaItems
mediaItemCollection: MPMediaItemCollection) {
        mediaItems = mediaItemCollection.items
        updatePlayer()
        self.dismiss(animated: true, completion: nil)
    }

    func updatePlayer(){
```

```
        let item = mediaItems[currentIndex]
        // DO-TRY-CATCH try to setup AVAudioPlayer with the path, if successful, sets up the
AVMusicPlayer, and song values.
        if let path: NSURL = item.assetURL as NSURL? {
            do
            {
                avMusicPlayer = try AVAudioPlayer(contentsOf: path as URL)
                avMusicPlayer.enableRate = true
                avMusicPlayer.rate = 1.0
                avMusicPlayer.numberOfLoops = 0
                avMusicPlayer.currentTime = 0
            }
            catch
            {
                avMusicPlayer = nil
            }
        }
    }

    @IBAction func Play(_ sender: AnyObject) {
        //AVMusicPlayer.deviceCurrentTime
        avMusicPlayer.play()
    }

    @IBAction func Stop(_ sender: AnyObject) {
        avMusicPlayer.stop()
    }

    @IBAction func picker(_ sender: AnyObject) {
        mpMediapicker = MPMediaPickerController.self(mediaTypes:MPMediaType.music)
        mpMediapicker.allowsPickingMultipleItems = false
        mpMediapicker.delegate = self
        self.present(mpMediapicker, animated: true, completion: nil)
    }

}
```

Прочитайте MPMediaPickerDelegate онлайн:
https://riptutorial.com/ru/ios/topic/7299/mpmediapickerdelegate

глава 41: MPVolumeView

Вступление

Класс MPVolumeView представляет собой представление объема, чтобы предоставить пользователю ползунок для установки громкости выходного аудиосигнала системы и кнопку выбора маршрута вывода звука.

замечания

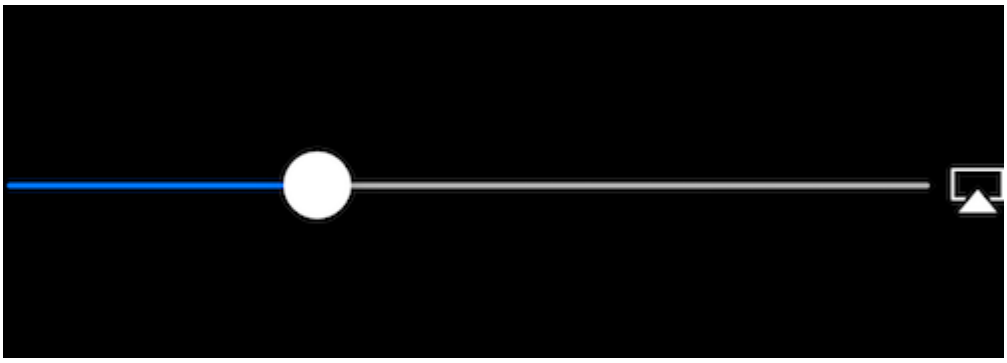
MPVolumeView появляется только при создании и запуске на реальном устройстве iOS и не будет работать в симуляторе.

Examples

Добавление MPVolumeView

```
// Add MPVolumeView in a holder view
let mpVolumeHolderView = UIView(frame: CGRect(x: 0, y: view.bounds.midY, width:
view.bounds.width, height: view.bounds.height))
// Set the holder view's background color to transparent
mpVolumeHolderView.backgroundColor = .clear
let mpVolume = MPVolumeView(frame: mpVolumeHolderView.bounds)
mpVolume.showsRouteButton = true
mpVolumeHolderView.addSubview(mpVolume)
view.addSubview(mpVolumeHolderView)
// the volume view is white, set the parent background to black to show it better in this
example
view.backgroundColor = .black
```

!!! Очень важно отметить, что MPVolumeView работает только на реальном устройстве, а не на симуляторе.



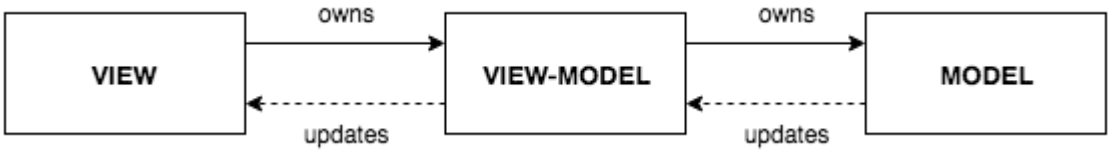
Прочитайте MPVolumeView онлайн: https://riptutorial.com/ru/ios/topic/9038/mpvolumeview

глава 42: MVVM

Examples

MVVM без реактивного программирования

Я начну с очень короткого объяснения, что и зачем использовать шаблон дизайна Model-View-ViewModel (MVVM) в приложениях iOS. Когда iOS впервые появилась, Apple предложила использовать MVC (Model-View-Controller) в качестве шаблона проектирования. Они показали это во всех своих примерах, и все первые разработчики были счастливы использовать его, потому что это прекрасно разделяло проблемы между бизнес-логикой и пользовательским интерфейсом. По мере того, как приложения становились все больше и сложнее, появилась новая проблема, получившая название Massive View Controllers (MVC). Поскольку вся бизнес-логика была добавлена в ViewController, со временем они стали слишком большими и сложными. Чтобы избежать проблемы MVC, в мир iOS - модель-View-ViewModel (MVVM) был введен новый шаблон дизайна.



На приведенной выше диаграмме показано, как выглядит MVVM. У вас есть стандартный ViewController + View (в раскадровке, XIB или Code), который действует как представление MVVM (в более позднем тексте - View будет ссылаться на представление MVVM). В представлении есть ссылка на ViewModel, где наша бизнес-логика. Важно заметить, что ViewModel ничего не знает о представлении и никогда не ссылается на представление. ViewModel имеет ссылку на модель.
Этого достаточно для теоретической части MVVM. Подробнее об этом можно прочитать здесь .

Одна из **основных проблем с MVVM** заключается в том, как обновить View через ViewModel, когда ViewModel не имеет ссылок и даже ничего не знает о представлении.

Основная часть этого примера - показать, как использовать MVVM (точнее, как связать ViewModel и View) без какого-либо реактивного программирования (ReactiveCocoa, ReactiveSwift или RxSwif). Также как примечание: если вы хотите использовать реактивное программирование, еще лучше, так как привязки MVVM выполняются очень просто с его помощью. Но этот пример - о том, как использовать MVVM без реактивного программирования.

Давайте создадим простой пример, чтобы продемонстрировать, как использовать MVVM.

Наш `MVVMExampleViewController` - это простой ViewController с меткой и кнопкой. При нажатии кнопки текст ярлыка должен быть установлен на «Hello». Поскольку решение о взаимодействии с пользователем является частью бизнес-логики, ViewModel должен будет решить, что делать, когда пользователь нажимает кнопку. Представление MVVM не должно вести бизнес-логику.

```
class MVVMExampleViewController: UIViewController {

    @IBOutlet weak var helloLabel: UILabel!

    var viewModel: MVVMExampleViewModel?

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func sayHelloButtonPressed(_ sender: UIButton) {
        viewModel?.userTriggeredSayHelloButton()
    }
}
```

`MVVMExampleViewModel` - это простая ViewModel.

```
class MVVMExampleViewModel {

    func userTriggeredSayHelloButton() {
        // How to update View's label when there is no reference to the View??
    }
}
```

Вы можете задаться вопросом, как установить ссылку ViewModel в представлении. Обычно я это делаю, когда ViewController инициализируется или пока он не будет показан. Для этого простого примера я бы сделал что-то подобное в `AppDelegate` :

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        if let rootVC = window?.rootViewController as? MVVMExampleViewController {
            let viewModel = MVVMExampleViewModel()
            rootVC.viewModel = viewModel
        }

        return true
```

Реальный вопрос: как обновить View из ViewModel без указания ссылки на ViewModel? (Помните, мы не будем использовать ни одну из библиотек iOS с реактивным программированием)

Вы могли бы подумать об использовании KVO, но это слишком усложняло бы ситуацию. Некоторые умные люди задумались над этой проблемой и придумали библиотеку Bond . Вначале библиотека может показаться сложной и немного сложнее понять, поэтому я просто возьму одну ее часть и сделаю наш MVVM полностью функциональным.

Давайте представим `Dynamic` класс, который является основой нашего простого, но полностью функционального шаблона MVVM.

```
class Dynamic<T> {
    typealias Listener = (T) -> Void
    var listener: Listener?

    func bind(_ listener: Listener?) {
        self.listener = listener
    }

    func bindAndFire(_ listener: Listener?) {
        self.listener = listener
        listener?(value)
    }

    var value: T {
        didSet {
            listener?(value)
        }
    }

    init(_ v: T) {
        value = v
    }
}
```

`Dynamic` класс использует Generics и Closures для привязки нашего ViewModel с нашим представлением. Я не буду вдаваться в подробности об этом классе, мы можем сделать это в комментариях (чтобы сделать этот пример короче). Давайте теперь обновим наш `MVVMExampleViewController` и `MVVMExampleViewModel` для использования этих классов.

Наш обновленный `MVVMExampleViewController`

```
class MVVMExampleViewController: UIViewController {

    @IBOutlet weak var helloLabel: UILabel!

    var viewModel: MVVMExampleViewModel?

    override func viewDidLoad() {
        super.viewDidLoad()
        bindViewModel()
    }

    func bindViewModel() {
        if let viewModel = viewModel {
            viewModel.helloText.bind({ (helloText) in
                DispatchQueue.main.async {
                    // When value of the helloText Dynamic variable
                    // is set or changed in the ViewModel, this code will
                    // be executed
                    self.helloLabel.text = helloText
                }
            })
        }
    }
```

```
    @IBAction func sayHelloButtonPressed(_ sender: UIButton) {
        viewModel?.userTriggeredSayHelloButton()
    }
}
```

Обновлен `MVVMExampleViewModel` :

```
    class MVVMExampleViewModel {

    // we have to initialize the Dynamic var with the
    // data type we want
    var helloText = Dynamic("")

    func userTriggeredSayHelloButton() {
        // Setting the value of the Dynamic variable
        // will trigger the closure we defined in the View
        helloText.value = "Hello"
    }
}
```

Вот и все. Теперь `ViewModel` теперь может обновлять `View` без ссылки на `View` .

Это действительно простой пример, но я думаю, что у вас есть идея, насколько это возможно. Я не буду вдаваться в подробности о преимуществах MVVM, но как только вы переключитесь с MVC на MVVM, вы не вернетесь. Просто попробуйте и убедитесь сами.

Прочитайте MVVM онлайн: https://riptutorial.com/ru/ios/topic/8775/mvvm

глава 43: MyLayout

Вступление

MyLayout - простая и простая объектно-ориентированная инфраструктура для iOS-представления. MyLayout предоставляет некоторые простые функции для создания разнообразного сложного интерфейса. Он объединяет функции, включая: Autolayout и SizeClass iOS, пять классов макета Android, float и flex-box и бутстрап HTML / CSS. вы можете посетить:

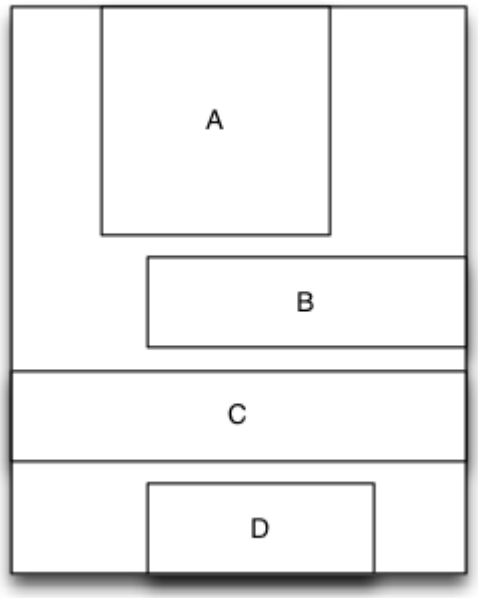
Objective-C: https://github.com/youngsoft/MyLinearLayout Swift: https://github.com/youngsoft/TangramKit

Examples

Простая демонстрация использования MyLayout

1. Существует вид контейнера S, ширина которого равна 100, а высота - на всю высоту надзора. есть четыре подвала A, B, C, D, расположенные сверху вниз.
2. Левое поле Subview A составляет 20% ширины S, правое поле составляет 30% ширины S, высота равна ширине A.
3. Левое поле Subview B составляет 40, ширина заполняется до остаточной ширины S, высота - 40. Ширина Subview C заполняется до S, высота равна 40.
4. Правое поле Subview D - 20, ширина - 50% S, высота - 40

как показано ниже:



```
    MyLinearLayout *S = [MyLinearLayout
linearLayoutWithOrientation:MyLayoutViewOrientation_Vert];
    S.subviewSpace = 10;
    S.widthSize.equalTo(@100);

    UIView *A = UIView.new;
    A.leftPos.equalTo(@0.2);
    A.rightPos.equalTo(@0.3);
    A.heightSize.equalTo(A.widthSize);
    [S addSubview:A];

    UIView *B = UIView.new;
    B.leftPos.equalTo(@40);
    B.widthSize.equalTo(@60);
    B.heightSize.equalTo(@40);
    [S addSubview:B];

    UIView *C = UIView.new;
    C.leftPos.equalTo(@0);
    C.rightPos.equalTo(@0);
    C.heightSize.equalTo(@40);
    [S addSubview:C];

    UIView *D = UIView.new;
    D.rightPos.equalTo(@20);
    D.widthSize.equalTo(S.widthSize).multiply(0.5);
    D.heightSize.equalTo(@40);
    [S addSubview:D];
```

Прочитайте MyLayout онлайн: https://riptutorial.com/ru/ios/topic/9692/mylayout

глава 44: NSArray

Вступление

Вот некоторые полезные служебные функции / методы, которые можно использовать как с расширением Array, чтобы облегчить разработчику выполнение определенных критических операций с массивом с помощью однострочного кода.

замечания

Как только текущий документ получит одобрение, будет добавлено так много усовершенствований для других применений массива. Это мой первый документ и нуждаюсь в вашей помощи и одобрении в моих усилиях.

Examples

Преобразовать массив в строку json

Вызовите эту функцию с аргументом параметра как массив с типом «any». Он вернет вам строку json. Строка Json используется для отправки массива в вызове веб-службы в качестве параметра ввода запроса в Swift.

// ----------

```
let array = [["one" : 1], ["two" : 2], ["three" : 3], ["four" : 4]]

let jsonString = convertIntoJSONString(arrayObject: array)
print("jsonString - \(jsonString)")
```

// ----------

```
func convertIntoJSONString(arrayObject: [Any]) -> String? {

        do {
            let jsonData: Data = try JSONSerialization.data(withJSONObject: arrayObject,
options: [])
            if  let jsonString = NSString(data: jsonData, encoding:
String.Encoding.utf8.rawValue) {
                return jsonString as String
            }

        } catch let error as NSError {
            print("Array convertIntoJSON - \(error.description)")
        }
        return nil
    }
```

Прочитайте NSArray онлайн: https://riptutorial.com/ru/ios/topic/9248/nsarray

глава 45: NSAttributedString

замечания

Установить цвет шрифта с помощью NSAttributedString

Examples

Создание строки, которая имеет пользовательский кернинг (межстрочный интервал)

`NSAttributedString` (и его изменяемый sibling `NSMutableAttributedString`) позволяет создавать строки, которые сложны по своему внешнему виду для пользователя.

Обычным приложением является использование этого для отображения строки и добавления пользовательского кернинга / межстрочного интервала.

Это будет достигнуто следующим образом (где label - `UILabel`), давая другой кернинг для слова «кернинг»,

стриж

```
var attributedString = NSMutableAttributedString("Apply kerning")
attributedString.addAttribute(attribute: NSKernAttributeName, value: 5, range: NSMakeRange(6,
7))
label.attributedText = attributedString
```

Objective-C

```
NSMutableAttributedString *attributedString;
attributedString = [[NSMutableAttributedString alloc] initWithString:@"Apply kerning"];
[attributedString addAttribute:NSKernAttributeName value:@5 range:NSMakeRange(6, 7)];
[label setAttributedText:attributedString];
```

Создать строку с зачеркнутым текстом

Objective-C

```
NSMutableAttributedString *attributeString = [[NSMutableAttributedString alloc]
initWithString:@"Your String here"];
[attributeString addAttribute:NSStrikethroughStyleAttributeName
                    value:@2
                    range:NSMakeRange(0, [attributeString length])];
```

стриж

```
let attributeString: NSMutableAttributedString =  NSMutableAttributedString(string: "Your
String here")
attributeString.addAttribute(NSStrikethroughStyleAttributeName, value: 2, range:
NSMakeRange(0, attributeString.length))
```

Затем вы можете добавить это в свой UILabel:

```
yourLabel.attributedText = attributeString;
```

Добавление атрибутных строк и полужирный текст в Swift

```
let someValue : String = "Something the user entered"
let text = NSMutableAttributedString(string: "The value is: ")
text.appendAttributedString(NSAttributedString(string: someValue, attributes:
[NSFontAttributeName:UIFont.boldSystemFontOfSize(UIFont.systemFontSize())]))
```

Результат выглядит так:

Значение: **Кое-что пользователь ввел**

Изменение цвета слова или строки

Objective-C

```
UIColor *color = [UIColor redColor];
NSString *textToFind = @"redword";

NSMutableAttributedString *attrsString =  [[NSMutableAttributedString alloc]
initWithAttributedString:yourLabel.attributedText];

// search for word occurrence
NSRange range = [yourLabel.text rangeOfString:textToFind];
if (range.location != NSNotFound) {
    [attrsString addAttribute:NSForegroundColorAttributeName value:color range:range];
}

// set attributed text
yourLabel.attributedText = attrsString;
```

стриж

```
let color = UIColor.red;
let textToFind = "redword"

let attrsString =  NSMutableAttributedString(string:yourlabel.text!);

// search for word occurrence
let range = (yourlabel.text! as NSString).range(of: textToFind)
if (range.length > 0) {
     attrsString.addAttribute(NSForegroundColorAttributeName,value:color,range:range)
}

// set attributed text
```

```
yourlabel.attributedText = attrsString
```

Примечание .

Главное здесь - использовать `NSMutableAttributedString` и селектор `addAttribute:value:range` с атрибутом `NSForegroundColorAttributeName` чтобы изменить цвет диапазона строк:

```
NSMutableAttributedString *attrsString =  [[NSMutableAttributedString alloc]
initWithAttributedString:label.attributedText];
[attrsString addAttribute:NSForegroundColorAttributeName value:color range:range];
```

Вы можете использовать другой способ получить диапазон, например: NSRegularExpression.

Удаление всех атрибутов

Objective-C

```
NSMutableAttributedString *mutAttString = @"string goes here";
NSRange range = NSMakeRange(0, mutAttString.length);
[mutAttString setAttributes:@{} range:originalRange];
```

В соответствии с Apple Documentation мы используем `setAttributes` а не `addAttribute` .

стриж

```
mutAttString.setAttributes([:], range: NSRange(0..<string.length))
```

Прочитайте NSAttributedString онлайн: https://riptutorial.com/ru/ios/topic/979/nsattributedstring

глава 46: NSBundle

Examples

Получение основного пакета

1. Получение ссылки на основной комплект с использованием Cocoa.

   Чтобы получить основной пакет в приложении Cocoa, вызовите ***метод*** класса **mainBundle** класса **NSBundle** .

```
   NSBundle *mainBundle;
   // Get the main bundle for the app;
   mainBundle = [NSBundle mainBundle];
```

2. Получение ссылки на основной пакет с использованием Core Foundation.

   Используйте функцию ***CFBundleGetMainBundle*** для извлечения основного пакета для вашего приложения на основе C.

```
   CFBundleRef mainBundle;
   // Get the main bundle for the app
   mainBundle = CFBundleGetMainBundle();
```

Получение связки по пути

1. Поиск пучка какао с использованием его пути

   Чтобы получить связку по определенному пути с использованием Cocoa, вызовите метод ***bundleWithPath:*** class **NSBundle**

```
   NSBundle *myBundle;
   // obtain a reference to a loadable bundle
   myBundle = [NSBundle bundleWithPath:@"/Library/MyBundle.bundle";
```

2. Поиск пакета Cocoa Foundation с использованием его пути

   Чтобы получить пакет по определенному пути с использованием Core Foundation, вызовите функцию ***CFBundleCreate*** и используйте тип **CFURLRef** .

```
   CFURLRef bundleURL;
   CFBundleRef myBundle;
   // Make a CFURLRef from the CFString representation of the bundle's path.
   bundleURL = CFURLCreateWithFileSystemPath(kCFAllocatorDefault,
CFSTR("/Library/MyBundle.bundle"), kCFURLPOSIXPathStyle, true);
   // Make a bundle instance using the URLRef.
   myBundle = CFBundleCreate(kCFAllocatorDefault, bundeURL);
```

```
// You can release the URL now.
CFRelease(bundleURL);
// Use the bundle ...
// Release the bundle when done.
CFRelease(myBundle);
```

Прочитайте NSBundle онлайн: https://riptutorial.com/ru/ios/topic/5862/nsbundle

глава 47: NSData

замечания

Полезные ресурсы

Документация Apple (NSData)

NSData.dataWithContentsOfFile ()

NSData.bytes

Examples

Создание объектов NSData

Использование файла

стриж

```
let data = NSData(contentsOfFile: filePath) //assuming filePath is a valid path
```

Objective-C

```
NSData *data = [NSData dataWithContentsOfFile:filePath]; //assuming filePath is a valid path
```

Использование объекта String

стриж

```
let data = (string as NSString).dataUsingEncoding(NSUTF8StringEncoding) //assuming string is a
String object
```

Objective-C

```
NSData *data = [string dataUsingEncoding:NSUTF8StringEncoding]; //assuming string is a String
object
```

Преобразование NSData в другие типы

Нанизывать

стриж

```
let string = String(NSString(data: data, encoding: NSUTF8StringEncoding)) //assuming data is a valid NSData object
```

Objective-C

```
NSString *string = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
//assuming data is a valid NSData object
[string release];
```

В массив

стриж

```
let array = data.bytes as! NSMutableArray //assuming data is a valid NSData object
```

Objective-C

```
NSMutableArray *array = (NSMutableArray *)[data bytes]; //assuming data is a valid NSData object
```

В байт-массив

стриж

```
let bytesArray = data.bytes as! UInt8 //assuming data is a valid NSData object
```

Objective-C

```
UInt8 *bytesArray = (UInt8 *)data.bytes; //assuming data is a valid NSData object
```

Преобразование NSData в строку HEX

`NSData` может быть представлена как шестнадцатеричная строка, аналогичная тому, что она выводит в своем методе `description` .

стриж

```
extension NSData {

    func hexString() -> String {
        return UnsafeBufferPointer<UInt8>(start: UnsafePointer<UInt8>(bytes), count: length)
            .reduce("") { $0 + String(format: "%02x", $1) }
    }

}
```

Objective-C

```
@implementation NSData (HexRepresentation)

- (NSString *)hexString {
    const unsigned char *bytes = (const unsigned char *)self.bytes;
    NSMutableString *hex = [NSMutableString new];
    for (NSInteger i = 0; i < self.length; i++) {
        [hex appendFormat:@"%02x", bytes[i]];
    }
    return [hex copy];
}

@end
```

Прочитайте NSData онлайн: https://riptutorial.com/ru/ios/topic/5084/nsdata

глава 48: NSDate

Синтаксис

- NSDate () // NSDate объект init для текущей даты и времени
- NSDate (). TimeIntervalSince1970 // Текущая дата и время в количестве секунд с 00:00:00 по UTC 1 января 1970 года.
- NSDate (). Compare (other: NSDate) // Возвращает сравнение текущей даты с другой датой возвращает `NSComparisonResult`

замечания

Существуют различные типы формата даты, которые вы можете установить: Вот полный список из них.

Формат	Значение / Описание	Example1	Example2
Y	Год с не менее 1 разрядом.	175 AD → «175»	2016 AD → «2016»
yy	Год с ровно 2 цифрами.	5 AD → «05»	2016 AD → «16»
yyy	Год, по крайней мере, 3 цифры.	5 AD → «005»	2016 AD → «2016»
гггг	Год, по крайней мере, 4 цифры.	5 AD → «0005»	2016 AD → «2016»
M	Месяц с не менее 1 разрядом.	Июль → «7»	«Ноябрь» → «11»
M.M.	Месяц не менее 2-х цифр.	Июль → «07»	«Ноябрь» → «11»
MMM	Сокращение на три буквы месяца.	Июль → "Jul"	«Ноябрь» → «Ноябрь»
MMMM	Полное название месяца.	Июль → «Июль»	«Ноябрь» → «Ноябрь»
MMMMM	Сокращение аббревиатуры одного месяца (январь, июнь,	Июль → «J»	«Ноябрь» → «N»

Формат	Значение / Описание	Example1	Example2
	июль, все будут иметь «J»).		
d	День с по крайней мере одной цифрой.	8 → «8»	29 → «29»
дд	День с по крайней мере двумя цифрами.	8 → «08»	29 → «29»
«E», «EE» или «EEE»,	3-х дневная аббревиатура имени дня.	Понедельник → «Пн»	Четверг → " Thu"
EEEE	Полное имя дня.	Понедельник → «Понедельник»	Четверг → «Четверг»
EEEEE	1-дневная аббревиатура дневного названия (Thu и Tue будет «T»)	Понедельник → «M»	Четверг → «T»
EEEEEE	2-х дневная аббревиатура имени дня.	Понедельник → «Mo»	Четверг → " Th"
	Период дня (AM / PM).	10 PM → «PM»	2 AM → «AM»
час	1-12 основанный час с по крайней мере 1 цифрой.	10 вечера → «10»	2 AM → «2»
чч	1-12 основанный час с по крайней мере 2 цифрами.	10 вечера → «10»	2 AM → «02»
ЧАС	0-23 балла с по крайней мере 1 цифрой.	10 вечера → «14»	2 AM → «2»
НН	A 0-23 базирующийся час, по крайней мере, 2 цифры.	10 вечера → «14»	2 AM → «02»
м	Минуту, по крайней мере, 1 цифру.	7 → "7"	29 → «29»
мм	Минута, по крайней мере, 2 цифры.	7 → «07»	29 → «29»
s	Вторая с не менее чем 1 цифрой.	7 → "7"	29 → «29»
сс	Второй, по крайней мере, 2 цифры.	7 → «07»	29 → «29»

Есть много других, для получения разного времени, основанного на зоне (z), для получения времени с миллисекундными деталями (S) и т. Д.

Examples

Получить текущую дату

Получение текущей даты очень просто. Вы получаете объект NSDate текущей даты только в одной строке следующим образом:

стриж

```
var date = NSDate()
```

Swift 3

```
var date = Date()
```

Objective-C

```
NSDate *date = [NSDate date];
```

Получить NSDate Object N секунд с текущей даты

Количество секунд с текущей даты и времени для новой даты. Используйте отрицательное значение для указания даты до текущей даты.

Для этого у нас есть метод с именем `dateWithTimerIntervalSinceNow(seconds: NSTimeInterval) -> NSDate` (Swift) или `+ (NSDate*)dateWithTimeIntervalSinceNow:(NSTimeInterval)seconds` (Objective-C).

Например, если вам требуется дата на одну неделю с текущей даты и на одну неделю до текущей даты, мы можем сделать это как.

стриж

```
let totalSecondsInWeek:NSTimeInterval = 7 * 24 * 60 * 60;
//Using negative value for previous date from today
let nextWeek = NSDate().dateWithTimerIntervalSinceNow(totalSecondsInWeek)

//Using positive value for future date from today
let lastWeek = NSDate().dateWithTimerIntervalSinceNow(-totalSecondsInWeek)
```

Swift 3

```
let totalSecondsInWeek:TimeInterval = 7 * 24 * 60 * 60;

//Using positive value to add to the current date
let nextWeek = Date(timeIntervalSinceNow: totalSecondsInWeek)

//Using negative value to get date one week from current date
let lastWeek = Date(timeIntervalSinceNow: -totalSecondsInWeek)
```

Objective-C

```
NSTimeInterval totalSecondsInWeek = 7 * 24 * 60 * 60;
//Using negative value for previous date from today
NSDate *lastWeek = [NSDate dateWithTimeIntervalSinceNow:-totalSecondsInWeek];

//Using positive value for future date from today
NSDate *nextWeek = [NSDate dateWithTimeIntervalSinceNow:totalSecondsInWeek];

NSLog(@"Last Week: %@", lastWeek);
NSLog(@"Right Now: %@", now);
NSLog(@"Next Week: %@", nextWeek);
```

Сравнение даты

Существует 4 метода сравнения дат:

стриж

- `isEqualToDate(anotherDate: NSDate) -> Bool`
- `earlierDate(anotherDate: NSDate) -> NSDate`
- `laterDate(anotherDate: NSDate) -> NSDate`
- `compare(anotherDate: NSDate) -> NSComparisonResult`

Objective-C

- `- (BOOL)isEqualToDate:(NSDate *)anotherDate`
- `- (NSDate *)earlierDate:(NSDate *)anotherDate`
- `- (NSDate *)laterDate:(NSDate *)anotherDate`
- `- (NSComparisonResult)compare:(NSDate *)anotherDate`

Предположим, у нас есть 2 даты:

стриж

```
let date1: NSDate = ... // initialized as  July 7, 2016 00:00:00
let date2: NSDate = ... // initialized as  July 2, 2016 00:00:00
```

Objective-C

```
NSDate *date1 = ... // initialized as  July 7, 2016 00:00:00
NSDate *date2 = ... // initialized as  July 2, 2016 00:00:00
```

Затем, чтобы сравнить их, мы попробуем этот код:

стриж

```
if date1.isEqualToDate(date2) {
    // returns false, as both dates aren't equal
}

earlierDate: NSDate = date1.earlierDate(date2) // returns the earlier date of the two (date 2)
laterDate: NSDate = date1.laterDate(date2) // returns the later date of the two (date1)

result: NSComparisonResult = date1.compare(date2)

if result == .OrderedAscending {
    // true if date1 is earlier than date2
} else if result == .OrderedSame {
    // true if the dates are the same
} else if result == .OrderedDescending {
    // true if date1 is later than date1
}
```

Objective-C

```
if ([date1 isEqualToDate:date2]) {
    // returns false, as both date are not equal
}

NSDate *earlierDate = [date1 earlierDate:date2]; // returns date which comes earlier from both
date, here it will return date2
NSDate *laterDate = [date1 laterDate:date2]; // returns date which comes later from both date,
here it will return date1

NSComparisonResult result = [date1 compare:date2];
if (result == NSOrderedAscending) {
    // fails
    // comes here if date1 is earlier then date2, in our case it will not come here
} else if (result == NSOrderedSame){
    // fails
    // comes here if date1 is same as date2, in our case it will not come here
} else{ // NSOrderedDescending
    // succeeds
    // comes here if date1 is later than date2, in our case it will come here
}
```

Если вы хотите сравнить даты и обрабатывать секунды, недели, месяцы и годы:

Swift 3

```
let dateStringUTC = "2016-10-22 12:37:48 +0000"
let dateFormatter = DateFormatter()
dateFormatter.locale = Locale(identifier: "en_US_POSIX")
dateFormatter.dateFormat = "yyyy-MM-dd HH:mm:ss X"
let date = dateFormatter.date(from: dateStringUTC)!

let now = Date()

let formatter = DateComponentsFormatter()
formatter.unitsStyle = .full
formatter.maximumUnitCount = 2
let string = formatter.string(from: date, to: Date())! + " " + NSLocalizedString("ago",
comment: "added after elapsed time to say how long before")
```

Или вы можете использовать это для каждого компонента:

```
// get the current date and time
let currentDateTime = Date()

// get the user's calendar
let userCalendar = Calendar.current

// choose which date and time components are needed
let requestedComponents: Set<Calendar.Component> = [
    .year,
    .month,
    .day,
    .hour,
    .minute,
    .second
]

// get the components
let dateTimeComponents = userCalendar.dateComponents(requestedComponents, from:
currentDateTime)

// now the components are available
dateTimeComponents.year
dateTimeComponents.month
dateTimeComponents.day
dateTimeComponents.hour
dateTimeComponents.minute
dateTimeComponents.second
```

Получить время Unix Epoch

Чтобы получить Unix Epoch Time , используйте константу `timeIntervalSince1970` :

стриж

```
let date = NSDate() // current date
let unixtime = date.timeIntervalSince1970
```

Objective-C

```
NSDate *date = [NSDate date]; // current date
int unixtime = [date timeIntervalSince1970];
```

NSDateFormatter

Преобразование объекта `NSDate` в строку - всего 3 шага.

1. Создайте объект `NSDateFormatter`

стриж

```
let dateFormatter = NSDateFormatter()
```

Swift 3

```
let dateFormatter = DateFormatter()
```

Objective-C

```
NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
```

2. Установите формат даты, в котором вы хотите, чтобы ваша строка

стриж

```
dateFormatter.dateFormat = "yyyy-MM-dd 'at' HH:mm"
```

Objective-C

```
dateFormatter.dateFormat = @"yyyy-MM-dd 'at' HH:mm";
```

3. Получите форматированную строку

стриж

```
let date = NSDate() // your NSDate object
let dateString = dateFormatter.stringFromDate(date)
```

Swift 3

```
let date = Date() // your NSDate object
let dateString = dateFormatter.stringFromDate(date)
```

Objective-C

```
NSDate *date = [NSDate date]; // your NSDate object
NSString *dateString = [dateFormatter stringFromDate:date];
```

Это даст результат примерно так: `2001-01-02 at 13:00`

> ## Заметка
>
> Создание экземпляра `NSDateFormatter` - дорогостоящая операция, поэтому рекомендуется создать его один раз и повторно использовать, когда это возможно.

Полезное расширение для преобразования даты в строку.

```
extension Date {
        func toString() -> String {
            let dateFormatter = DateFormatter()
            dateFormatter.dateFormat = "MMMM dd yyyy"
            return dateFormatter.string(from: self)
        }
}
```

Полезные ссылки для быстрого создания даты- быстро-получение-человеко-читаемый-date-nsdateformatter .

Для создания форматов даты см. Шаблоны формата даты .

Преобразование NSDate, который состоит из часа и минуты (только) для полного NSDate

Существует много случаев, когда вы создали NSDate только с часовым и минутным

форматом, то есть: 08:12, который возвращается с сервера в виде строки, и вы инициируете экземпляр NSDate только этими **значениями.**

Недостатком этой ситуации является то, что ваш NSDate почти полностью «голый», и вам нужно создать: день, месяц, год, второй и часовой пояс, чтобы этот объект «играл» с другими типами NSDate.

Для примера предположим, что hourAndMinute - это тип NSDate, который состоит из часового и минутного формата:

Objective-C

```
NSDateComponents *hourAndMinuteComponents = [calendar components:NSCalendarUnitHour |
NSCalendarUnitMinute
                                                          fromDate:hourAndMinute];
NSDateComponents *componentsOfDate = [[NSCalendar currentCalendar]
components:NSCalendarUnitDay | NSCalendarUnitMonth | NSCalendarUnitYear
                                                                  fromDate:[NSDate date]];

NSDateComponents *components = [[NSDateComponents alloc] init];
[components setDay: componentsOfDate.day];
[components setMonth: componentsOfDate.month];
[components setYear: componentsOfDate.year];
[components setHour: [hourAndMinuteComponents hour]];
[components setMinute: [hourAndMinuteComponents minute]];
[components setSecond: 0];
[calendar setTimeZone: [NSTimeZone defaultTimeZone]];

NSDate *yourFullNSDateObject = [calendar dateFromComponents:components];
```

Теперь ваш объект является общей противоположностью тому, чтобы быть «голым».

UTC Смещение по времени от NSDate с помощью TimeZone

Здесь это рассчитает смещение времени `UTC` от текущих данных в желаемый часовой пояс.

```
+(NSTimeInterval)getUTCOffSetIntervalWithCurrentTimeZone:(NSTimeZone *)current forDate:(NSDate
*)date  {
    NSTimeZone *utcTimeZone = [NSTimeZone timeZoneWithAbbreviation:@"UTC"];
    NSInteger currentGMTOffset = [current secondsFromGMTForDate:date];
    NSInteger gmtOffset = [utcTimeZone secondsFromGMTForDate:date];
    NSTimeInterval gmtInterval = currentGMTOffset - gmtOffset;
    return gmtInterval;
}
```

Получить тип цикла (12-часовой или 24-часовой)

Проверка, содержит ли текущая дата

символ для AM или PM

Objective-C

```
NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
[formatter setLocale:[NSLocale currentLocale]];
[formatter setDateStyle:NSDateFormatterNoStyle];
[formatter setTimeStyle:NSDateFormatterShortStyle];
NSString *dateString = [formatter stringFromDate:[NSDate date]];
NSRange amRange = [dateString rangeOfString:[formatter AMSymbol]];
NSRange pmRange = [dateString rangeOfString:[formatter PMSymbol]];
BOOL is24h = (amRange.location == NSNotFound && pmRange.location == NSNotFound);
```

Запрос типа временного цикла из `NSDateFormatter`

Objective-C

```
NSString *formatStringForHours = [NSDateFormatter dateFormatFromTemplate:@"j" options:0
locale:[NSLocale currentLocale]];
NSRange containsA = [formatStringForHours rangeOfString:@"a"];
BOOL is24h = containsA.location == NSNotFound;
```

Это использует специальную строку шаблона даты, называемую «j», которая, согласно спецификации ICU ...

> [...] запрашивает предпочтительный формат часов для локали (h, H, K или k), как определено предпочтительным атрибутом элемента часов в дополнительных данных. [...] Обратите внимание, что использование «j» в скелете, переданном API, является единственным способом запросить скелет в качестве предпочтительного временного цикла локали (12-часовой или 24-часовой).

Последнее предложение важно. Это «единственный способ запросить скелет в качестве предпочтительного временного цикла языка». Поскольку `NSDateFormatter` и `NSCalendar` построены на библиотеке ICU, то здесь все верно.

Ссылка

Второй вариант был получен из этого ответа .

Получить NSDate из формата даты JSON "/ Date (1268123281843) /"

До Json.NET 4.5 даты были написаны с использованием формата Microsoft: «/ Date

(1198908717056) /». Если ваш сервер отправляет дату в этом формате, вы можете использовать приведенный ниже код для его сериализации в NSDate:

Objective-C

```
(NSDate*) getDateFromJSON:(NSString *)dateString
{
    // Expect date in this format "/Date(1268123281843)/"
    int startPos = [dateString rangeOfString:@"("].location+1;
    int endPos = [dateString rangeOfString:@")"].location;
    NSRange range = NSMakeRange(startPos,endPos-startPos);
    unsigned long long milliseconds = [[dateString substringWithRange:range] longLongValue];
    NSLog(@"%llu",milliseconds);
    NSTimeInterval interval = milliseconds/1000;
    NSDate *date = [NSDate dateWithTimeIntervalSince1970:interval];
    // add code for date formatter if need NSDate in specific format.
    return date;
}
```

Получить историческое время от NSDate (например: 5 лет назад, 2 года назад, 3 часа назад)

Это можно использовать в различных чат-приложениях, rss-каналах и социальных приложениях, где вам нужно иметь последние фиды с отметками времени:

Objective-C

```
- (NSString *)getHistoricTimeText:(NSDate *)since
{
    NSString *str;
    NSTimeInterval interval = [[NSDate date] timeIntervalSinceDate:since];
    if(interval < 60)
        str = [NSString stringWithFormat:@"%is ago",(int)interval];
    else if(interval < 3600)
    {
        int minutes = interval/60;
        str = [NSString stringWithFormat:@"%im ago",minutes];
    }
    else if(interval < 86400)
    {
        int hours =  interval/3600;

        str = [NSString stringWithFormat:@"%ih ago",hours];
    }
    else
    {
        NSDateFormatter *dateFormater=[[NSDateFormatter alloc]init];
        [dateFormater setLocale:[NSLocale currentLocale]];
        NSString *dateFormat = [NSDateFormatter dateFormatFromTemplate:@"MMM d, YYYY"
options:0 locale:[NSLocale currentLocale]];
        [dateFormater setDateFormat:dateFormat];
        str = [dateFormater stringFromDate:since];

    }
```

```
    return str;
}
```

Прочитайте NSDate онлайн: https://riptutorial.com/ru/ios/topic/1502/nsdate

глава 49: NSHTTPCookieStorage

Examples

Хранить и читать файлы cookie из NSUserDefault

```
import Foundation


class CookiesSingleton {

static let instance : CookiesSingleton = CookiesSingleton()
static var enableDebug = true

func loadCookies() {
    if let cookiesDetails =
NSUserDefaults.standardUserDefaults().objectForKey("customeWebsite")  {
        for (keys,_) in cookiesDetails as! NSDictionary{
                if let  cookieDict = NSUserDefaults.standardUserDefaults().objectForKey(keys
as! String){
                    if let cookie = NSHTTPCookie(properties:cookieDict as! [String:AnyObject])
{
                        NSHTTPCookieStorage.sharedHTTPCookieStorage().setCookie(cookie)
                        if(CookiesSingleton.enableDebug){
                            print("Each Cookies",cookieDict)
                        }
                    }
                }
        }
    }
}

func  removeCookies(){
    NSURLCache.sharedURLCache().removeAllCachedResponses()
    NSURLCache.sharedURLCache().diskCapacity = 0
    NSURLCache.sharedURLCache().memoryCapacity = 0

    let storage : NSHTTPCookieStorage = NSHTTPCookieStorage.sharedHTTPCookieStorage()
    for cookie in storage.cookies! {
        storage.deleteCookie(cookie as NSHTTPCookie)
    }

    NSUserDefaults.standardUserDefaults().setValue("", forKey: "customeWebsite")
    NSUserDefaults.standardUserDefaults().synchronize()

    if(CookiesSingleton.enableDebug){
        print("Cookies Removed")
    }
}

func saveCookies() {

    let cookieArray = NSMutableArray()
    let savedC = NSHTTPCookieStorage.sharedHTTPCookieStorage().cookies

    let allCookiesDic:NSMutableDictionary = NSMutableDictionary()
```

```
    for c : NSHTTPCookie in savedC! {

        let cookieProps = NSMutableDictionary()
        cookieArray.addObject(c.name)
        cookieProps.setValue(c.name, forKey: NSHTTPCookieName)
        cookieProps.setValue(c.value, forKey: NSHTTPCookieValue)
        cookieProps.setValue(c.domain, forKey: NSHTTPCookieDomain)
        cookieProps.setValue(c.path, forKey: NSHTTPCookiePath)
        cookieProps.setValue(c.version, forKey: NSHTTPCookieVersion)
        cookieProps.setValue(NSDate().dateByAddingTimeInterval(2629743), forKey:
NSHTTPCookieExpires)

        allCookiesDic.setValue(cookieProps, forKey: c.name)

    }
    NSUserDefaults.standardUserDefaults().setValue(allCookiesDic, forKey: "customeWebsite")
    NSUserDefaults.standardUserDefaults().synchronize()

    if(CookiesSingleton.enableDebug){
        print("Cookies Saved")
    }
}

}
```

Прочитайте NSHTTPCookieStorage онлайн:
https://riptutorial.com/ru/ios/topic/7312/nshttpcookiestorage

глава 50: NSInvocation

Examples

Объект NSInvocation-C

См. Это оригинальное сообщение от e.James

Согласно ссылке класса NSInvocation от Apple :

> `NSInvocation` - это сообщение Objective-C, статичное, т. `NSInvocation` Это действие, превращенное в объект.

И, *немного* подробнее:

Концепция сообщений занимает центральное место в философии объектива-с. Каждый раз, когда вы вызываете метод или получаете доступ к переменной какого-либо объекта, вы отправляете ему сообщение. `NSInvocation` пригодится, когда вы хотите отправить сообщение объекту в другой момент времени или отправить одно и то же сообщение несколько раз. `NSInvocation` позволяет вам *описать* сообщение, которое вы собираетесь отправить, и затем *вызывать* его (фактически отправить его на целевой объект) позже.

Например, предположим, вы хотите добавить строку в массив. Обычно вы отправляете сообщение `addObject:` следующим образом:

```
[myArray addObject:myString];
```

`NSInvocation` вы хотите использовать `NSInvocation` для отправки этого сообщения в какой-то другой момент времени:

Во-первых, вы должны подготовить объект `NSInvocation` для использования с `NSMutableArray` : `addObject:` selector:

```
NSMethodSignature * mySignature = [NSMutableArray
    instanceMethodSignatureForSelector:@selector(addObject:)];
NSInvocation * myInvocation = [NSInvocation
    invocationWithMethodSignature:mySignature];
```

Затем вы должны указать, какой объект отправить сообщение:

```
[myInvocation setTarget:myArray];
```

Укажите сообщение, которое вы хотите отправить этому объекту:

```
[myInvocation setSelector:@selector(addObject:)];
```

И заполните любые аргументы для этого метода:

```
[myInvocation setArgument:&myString atIndex:2];
```

Обратите внимание, что аргументы объекта должны передаваться указателем. Спасибо Райану МакКуаигу за указание на это, и, пожалуйста, см . Документацию Apple для получения более подробной информации.

На этом этапе `myInvocation` является полным объектом, описывающим сообщение, которое может быть отправлено. Чтобы отправить сообщение, вы должны позвонить:

```
[myInvocation invoke];
```

Этот последний шаг приведет к отправке сообщения, по существу выполняя `[myArray addObject:myString];` ,

Подумайте, как отправить электронное письмо. Вы открываете новое электронное письмо (объект `NSInvocation`), заполняете адрес человека (объекта), которому вы хотите его отправить, введите сообщение для получателя (укажите `selector` и аргументы), а затем нажмите «отправить», (вызов `invoke`).

Дополнительную информацию см. В разделе Использование NSInvocation .

`NSUndoManager` использует объекты `NSInvocation` чтобы он мог *отменить* команды. По существу, вы делаете создание объекта `NSInvocation` чтобы сказать: «Эй, если вы хотите отменить то, что я только что сделал, отправьте это сообщение этому объекту с этими аргументами». Вы предоставляете объект `NSInvocation` для `NSUndoManager` и добавляет этот объект в массив отмененных действий. Если пользователь вызывает «Отменить», `NSUndoManager` просто ищет последнее действие в массиве и вызывает сохраненный объект `NSInvocation` для выполнения необходимых действий.

Подробнее см. « Регистрация операций отмены» .

Прочитайте NSInvocation онлайн: https://riptutorial.com/ru/ios/topic/8276/nsinvocation

глава 51: NSNotificationCenter

Вступление

Уведомления iOS - это простой и эффективный способ отправки данных в связном режиме. То есть отправителю уведомления не нужно заботиться о том, кто (если кто-либо) получает уведомление, он просто отправляет его туда в остальную часть приложения, и его можно подхватить множеством вещей или ничего в зависимости от состояние вашего приложения.

Источник : - HACKING with Swift

параметры

параметр	подробности
название	Название уведомления, для которого зарегистрирован наблюдатель; то есть только уведомления с этим именем используются для добавления блока в очередь операций. Если вы передаете нуль, центр уведомлений не использует имя уведомления, чтобы решить, следует ли добавить блок в очередь операций.
OBJ	Объект, чьи уведомления, которые наблюдатель хочет получить; то есть только уведомления, отправленные этим отправителем, доставляются наблюдателю. Если вы передадите ноль, центр уведомлений не использует отправителя уведомления, чтобы решить, передавать ли его наблюдателю.
очередь	Операционная очередь, к которой должен быть добавлен блок. Если вы передаете нуль, блок запускается синхронно в потоке проводки.
блок	Блок, который будет выполняться при получении уведомления. Блок копируется центром уведомлений и (копией) удерживается до тех пор, пока регистрация наблюдателя не будет удалена.

замечания

Объект NSNotificationCenter (или просто центр уведомлений) предоставляет механизм для передачи информации внутри программы. Объектом NSNotificationCenter является, по существу, таблица отправки уведомлений.

Для получения дополнительной информации ознакомьтесь с документацией Apple здесь.

Examples

Добавление наблюдателя

Соглашение об именовании

Уведомления идентифицируются глобальными объектами NSString, имена которых составлены следующим образом:

`Name of associated class` + `Did | Will` + `UniquePartOfName` + `Notification`

Например:

- NSApplicationDidBecomeActiveNotification
- NSWindowDidMiniaturizeNotification
- NSTextViewDidChangeSelectionNotification
- NSColorPanelColorDidChangeNotification

Swift 2.3

```
NSNotificationCenter.defaultCenter().addObserver(self,
                                                 selector:
#selector(self.testNotification(_:)),
                                                 name: "TestNotification",
                                                 object: nil)
```

Swift 3

```
NSNotificationCenter.default.addObserver(self,
                                         selector: #selector(self.testNotification(_:)),
                                         name: NSNotification.Name(rawValue:
"TestNotification"),
                                         object: nil)
```

Objective-C

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                      selector:@selector(testNotification:)
                                      name:@"TestNotification"
                                      object:nil];
```

PS: Стоит также отметить, что количество раз, когда наблюдатель был добавлен, должен быть точно количеством раз, когда наблюдатель удаляется. Ошибка новобранец заключается в том, чтобы добавить наблюдателя в `viewWillAppear:` из UIViewController, но удаление наблюдателя в `viewDidUnload:` приведет к `viewWillAppear:` количеству нажатий и, таким образом, к утечке наблюдателя, а селектор уведомлений будет вызван излишним образом.

Удаление наблюдателей

Swift 2.3

```
//Remove observer for single notification
NSNotificationCenter.defaultCenter().removeObserver(self, name: "TestNotification", object:
nil)

//Remove observer for all notifications
NotificationCenter.defaultCenter().removeObserver(self)
```

Swift 3

```
//Remove observer for single notification
NotificationCenter.default.removeObserver(self, name: NSNotification.Name(rawValue:
"TestNotification"), object: nil)

//Remove observer for all notifications
NotificationCenter.default.removeObserver(self)
```

Objective-C

```
//Remove observer for single notification
[[NSNotificationCenter defaultCenter] removeObserver:self name:@"TestNotification"
object:nil];

//Remove observer for all notifications
[[NSNotificationCenter defaultCenter] removeObserver:self];
```

Проводка уведомления

стриж

```
NSNotificationCenter.defaultCenter().postNotificationName("TestNotification", object: self)
```

Objective-C

```
[[NSNotificationCenter defaultCenter] postNotificationName:@"TestNotification" object:nil];
```

Проводка уведомления с данными

стриж

```
let userInfo: [String: AnyObject] = ["someKey": myObject]
NSNotificationCenter.defaultCenter().postNotificationName("TestNotification", object: self,
userInfo: userInfo)
```

Objective-C

```
NSDictionary *userInfo = [NSDictionary dictionaryWithObject:myObject forKey:@"someKey"];
[[NSNotificationCenter defaultCenter] postNotificationName: @"TestNotification" object:nil
userInfo:userInfo];
```

Наблюдение за уведомлением

стриж

```
func testNotification(notification: NSNotification) {
    let userInfo = notification.userInfo
    let myObject: MyObject = userInfo["someKey"]
}
```

Objective-C

```
- (void)testNotification:(NSNotification *)notification {
    NSDictionary *userInfo = notification.userInfo;
    MyObject *myObject = [userInfo objectForKey:@"someKey"];
}
```

Добавление / удаление наблюдателя с блоком

Вместо добавления наблюдателя с селектором можно использовать блок:

```
id testObserver = [[NSNotificationCenter defaultCenter] addObserverForName:@"TestNotification"
                                                                     object:nil
```

```
                                                                    queue:nil
                                                                usingBlock:^(NSNotification*
notification) {
    NSDictionary *userInfo = notification.userInfo;
    MyObject *myObject = [userInfo objectForKey:@"someKey"];
}];
```

Затем наблюдателя можно удалить с помощью:

```
[[NSNotificationCenter defaultCenter] removeObserver:testObserver
                                                name:@"TestNotification"
                                              object:nil];
```

Добавить и удалить наблюдателя для имени

```
// Add observer
let observer =
NSNotificationCenter.defaultCenter().addObserverForName("nameOfTheNotification", object: nil,
queue: nil) { (notification) in
    // Do operations with the notification in this block
}

// Remove observer
NSNotificationCenter.defaultCenter().removeObserver(observer)
```

Прочитайте NSNotificationCenter онлайн:
https://riptutorial.com/ru/ios/topic/1601/nsnotificationcenter

глава 52: NSPredicate

Синтаксис

- Форматы струйных форматирования
  - Спецификаторы строки формата C:% d,% s,% f и т. Д.
  - Замещение объекта:% @
  - Замена ключа:% K
- Операторы сравнения предикатов
  - =, ==: Левое выражение равно правому выражению
  - > =, =>: Левое выражение больше или равно правому выражению
  - <=, = <: Левое выражение меньше или равно правому выражению
  - >: Левое выражение больше, чем правое выражение
  - <: Левое выражение меньше, чем правое выражение
  - ! =, <>: Левое выражение не равно правому выражению
  - МЕЖДУ: Левое выражение находится между или равно одному из значений в правом выражении, которое определяет нижнюю и верхнюю границы - ex: BETWEEN {0, 5}
- Предикатные сложные операторы
  - AND, &&: Логическое И
  - ИЛИ, ||: логическое ИЛИ
  - НЕ,!: Логическое НЕ
- Операторы сравнения предикатных строк
  - BEGINSWITH: Левое выражение начинается с правого выражения
  - ENDSWITH: левое выражение заканчивается правым выражением
  - СОДЕРЖАНИЕ: Левое выражение содержит правое выражение
  - LIKE: левое выражение равно правому выражению, с подстановкой подстановки
    - *: Совместить ноль или более символов
    - ?: Сопоставьте один символ

Examples

Создание NSPredicate Использование predicateWithBlock

Objective-C

```
NSPredicate *predicate = [NSPredicate predicateWithBlock:^BOOL(id item,
                                                               NSDictionary *bindings) {
    return [item isKindOfClass:[UILabel class]];
}];
```

стриж

```
let predicate = NSPredicate { (item, bindings) -> Bool in
    return item.isKindOfClass(UILabel.self)
}
```

В этом примере предикат будет соответствовать элементам класса `UILabel` .

Создание NSPredicate с использованием predicateWithFormat

Objective-C

```
NSPredicate *predicate = [NSPredicate predicateWithFormat: @"self[SIZE] = %d", 5)];
```

стриж

```
let predicate = NSPredicate(format: "self[SIZE] >= %d", 5)
```

В этом примере предикат будет соответствовать элементам, которые представляют собой массивы длиной не менее 5.

Создание NSPredicate с переменными замещения

`NSPredicate` может использовать переменные замещения, чтобы позволить привязывать значения на лету.

Objective-C

```
NSPredicate *template = [NSPredicate predicateWithFormat: @"self BEGINSWITH $letter"];
NSDictionary *variables = @{ @"letter": @"r" };
NSPredicate *beginsWithR = [template predicateWithSubstitutionVariables: variables];
```

стриж

```
let template = NSPredicate(format: "self BEGINSWITH $letter")
let variables = ["letter": "r"]
let beginsWithR = template.predicateWithSubstitutionVariables(variables)
```

Предикат шаблона не изменяется `predicateWithSubstitutionVariables` . Вместо этого создается копия, и эта копия получает переменные замещения.

Использование NSPredicate для фильтрации массива

Objective-C

```
NSArray *heroes = @[@"tracer", @"bastion", @"reaper", @"junkrat", @"roadhog"];

NSPredicate *template = [NSPredicate predicateWithFormat:@"self BEGINSWITH $letter"];

NSDictionary *beginsWithRVariables = @{ @"letter": @"r"};
NSPredicate *beginsWithR = [template predicateWithSubstitutionVariables:
beginsWithRVariables];

NSArray *beginsWithRHeroes = [heroes filteredArrayUsingPredicate: beginsWithR];
// ["reaper", "roadhog"]

NSDictionary *beginsWithTVariables = @{ @"letter": @"t"};
NSPredicate *beginsWithT = [template predicateWithSubstitutionVarables: beginsWithTVariables];

NSArray *beginsWithTHeroes = [heroes filteredArrayUsingPredicate: beginsWithT];
// ["tracer"]
```

стриж

```
let heroes = ["tracer", "bastion", "reaper", "junkrat", "roadhog"]

let template = NSPredicate(format: "self BEGINSWITH $letter")

let beginsWithRVariables = ["letter": "r"]
let beginsWithR = template.predicateWithSubstitutionVariables(beginsWithRVariables)

let beginsWithRHeroes = heroes.filter { beginsWithR.evaluateWithObject($0) }
// ["reaper", "roadhog"]

let beginsWithTVariables = ["letter": "t"]
let beginsWithT = template.predicateWithSubstitutionVariables(beginsWithTVariables)

let beginsWithTHeroes = heroes.filter { beginsWithT.evaluateWithObject($0) }
// ["tracer"]
```

Проверка формы с использованием NSPredicate

```
NSString *emailRegex = @"[A-Z0-9a-z]([A-Z0-9a-z._-]{0,64})+[A-Z0-9a-z]+@[A-Z0-9a-z]+([A-Za-z0-
9.-]{0,64})+([A-Z0-9a-z])+\\.[A-Za-z]{2,4}";    NSString *firstNameRegex = @"[0-9A-Za-z\"'-
]{2,32}$";
NSString *firstNameRegex = @"[ 0-9A-Za-z]{2,32}$";
NSString *lastNameRegex = @"[0-9A-Za-z\"'-]{2,32}$";
NSString *mobileNumberRegEx = @"^[0-9]{10}$";
NSString *zipcodeRegEx = @"^[0-9]{5}$";
NSString *SSNRegEx = @"^\\d{3}-?\\d{2}-?\\d{4}$";
NSString *addressRegEx = @"^[ A-Za-z0-9]{2,32}$";
NSString *cityRegEx = @"^[ A-Za-z0-9]{2,25}$";
NSString *PINRegEx = @"^[0-9]{4}$";
NSString *driversLiscRegEx = @"^[0-9a-zA-Z]{5,20}$";
```

```
-(BOOL)validateEmail {
    //Email address field should give an error when the email address begins with ".","-","_"
.
    NSPredicate *emailPredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
emailRegex];
    return ([emailPredicate evaluateWithObject:self.text] && self.text.length <= 64 &&
([self.text rangeOfString:@".."].location == NSNotFound));
}

- (BOOL)validateFirstName {
    NSPredicate *firstNamePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
firstNameRegex];
    return [firstNamePredicate evaluateWithObject:self.text];
}

- (BOOL)validateLastName {
    NSPredicate *lastNamePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
lastNameRegex];
    return [lastNamePredicate evaluateWithObject:self.text];
}

- (BOOL)validateAlphaNumericMin2Max32 {
    NSPredicate *firstNamePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
firstNameRegex];
    return [firstNamePredicate evaluateWithObject:self.text];
}

- (BOOL)validateMobileNumber {
    NSString *strippedMobileNumber =  [[[[self.text stringByReplacingOccurrencesOfString:@"("
withString:@""]
                                        stringByReplacingOccurrencesOfString:@")"
withString:@""]
                                        stringByReplacingOccurrencesOfString:@"-"
withString:@""]
                                        stringByReplacingOccurrencesOfString:@" "
withString:@""];

    NSPredicate *mobileNumberPredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
mobileNumberRegEx];

    return [mobileNumberPredicate evaluateWithObject:strippedMobileNumber];
}

- (BOOL)validateZipcode {
    NSPredicate *zipcodePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
zipcodeRegEx];

    return [zipcodePredicate evaluateWithObject:self.text];
}

- (BOOL)validateSSN {
NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@", SSNRegEx];

return [predicate evaluateWithObject:self.text];
}

- (BOOL)validateAddress {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
addressRegEx];
```

```
    return [predicate evaluateWithObject:self.text];
}

- (BOOL)validateCity {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@", cityRegEx];
    return [predicate evaluateWithObject:self.text];
}

- (BOOL)validatePIN {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@", PINRegEx];

    return [predicate evaluateWithObject:self.text];
}
   - (BOOL)validateDriversLiscNumber {
    if([self.text length] > 20) {
        return NO;
    }
    NSPredicate *driversLiscPredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
driversLiscRegEx];

    return [driversLiscPredicate evaluateWithObject:self.text];
}
```

NSPredicate с условием `AND`,` OR` и `NOT`

Условный предикат будет более чистым и безопасным, используя класс `NSCompoundPredicate` который предоставляет базовые логические операторы для данных предикатов.

Objective-C

И - условие

```
  NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
  NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
  NSPredicate *combinedPredicate = [NSCompoundPredicate andPredicateWithSubpredicates:
@[predicate,anotherPredicate]];
```

ИЛИ - Условие

```
 NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
 NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
 NSPredicate *combinedPredicate = [NSCompoundPredicate orPredicateWithSubpredicates:
@[predicate,anotherPredicate]];
```

НЕ - Состояние

```
 NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
 NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
 NSPredicate *combinedPredicate = [NSCompoundPredicate notPredicateWithSubpredicate:
```

```
@[predicate,anotherPredicate]];
```

Прочитайте NSPredicate онлайн: https://riptutorial.com/ru/ios/topic/5796/nspredicate

глава 53: NSTimer

параметры

параметр	подробности
`interval`	Время, в секундах, чтобы подождать, чтобы запустить таймер; или, повторяя таймеры, время между стрельбой.
`target`	Объект для вызова `selector`
`selector`	В Swift объект `Selector` определяющий метод вызова `target`
`repeats`	Если значение `false` , прогоните таймер только один раз. Если значение `true` , запустите таймер каждый `interval` секунд.

замечания

`NSTimer` позволяет вам отправить сообщение в цель по истечении заданного периода времени.

Examples

Создание таймера

Это создаст таймер, чтобы вызвать метод `doSomething` для `self` через 5 секунд.

стриж

```
let timer = NSTimer.scheduledTimerWithTimeInterval(5,
                                target: self,
                              selector: Selector(doSomething()),
                              userInfo: nil,
                               repeats: false)
```

Swift 3

```
 let timer = Timer.scheduledTimer(timeInterval: 1,
                                          target: self,
                                        selector: #selector(doSomething()),
                                        userInfo: nil,
                                         repeats: true)
```

Objective-C

```
NSTimer *timer = [NSTimer scheduledTimerWithTimeInterval:5.0 target:self
selector:@selector(doSomething) userInfo:nil repeats:NO];
```

Установка повторений на `false/NO` означает, что мы хотим, чтобы таймер срабатывал только один раз. Если мы установим значение `true/YES` , оно будет срабатывать каждые пять секунд, пока оно не будет отменено вручную.

Ручное включение таймера

стриж

```
timer.fire()
```

Objective-C

```
[timer fire];
```

Вызов метода `fire` заставляет NSTimer выполнять задачу, которую он обычно выполнял по расписанию.

В **неповторяющемся таймере** это автоматически отключит таймер. То есть, вызов `fire` до того, как временной интервал будет завершен, приведет только к одному вызову.

В **повторяющемся таймере** это просто вызовет действие без прерывания обычного графика.

Недействительный таймер

стриж

```
timer.invalidate()
```

Objective-C

```
[timer invalidate];
```

Это остановит таймер от срабатывания. **Должен быть вызван из потока, в который был создан таймер,** см . Примечания Apple :

> Вы должны отправить это сообщение из потока, на котором был установлен таймер. Если вы отправляете это сообщение из другого потока, источник входного сигнала, связанный с таймером, не может быть удален из цикла запуска, что может помешать правильному выводу потока.

Примечания. Как только таймер был признан недействительным, его невозможно запустить тот же самый недействительный таймер. Вместо этого вам нужно снова

инициализировать недействительный таймер и вызвать метод пожара.

Параметры частоты таймера

Событие повторного таймера

стриж

```
class ViewController: UIViewController {

    var timer = NSTimer()

    override func viewDidLoad() {
        NSTimer.scheduledTimerWithTimeInterval(1.0, target: self, selector:
Selector(self.timerMethod()), userInfo: nil, repeats: true)
    }

    func timerMethod() {
        print("Timer method called")
    }

    func endTimer() {
        timer.invalidate()
    }
}
```

Swift 3

```
class ViewController: UIViewController {

        var timer = Timer()

        override func viewDidLoad() {
            Timer.scheduledTimer(timeInterval: 1.0, target: self, selector:
#selector(self.timerMethod()), userInfo: nil, repeats: true)
        }

        func timerMethod() {
            print("Timer method called")
        }

        func endTimer() {
            timer.invalidate()
        }
    }
```

При необходимости может быть аннулировано вручную.

стриж

Не повторяющееся событие таймера с задержкой

```
NSTimer.scheduledTimerWithTimeInterval(3.0, target: self, selector:
```

```
Selector(self.timerMethod()), userInfo: nil, repeats: false)
```

Swift 3

```
 Timer.scheduledTimer(timeInterval: 3.0, target: self, selector:
#selector(self.timerMethod()), userInfo: nil, repeats: false)
```

Таймер будет запущен один раз, через 3 секунды после выполнения. Будет автоматически аннулирована, после увольнения.

Передача данных с использованием таймера

Если вы хотите передать некоторые данные с помощью триггера таймера, вы можете сделать это с `userInfo` параметра `userInfo` .

Вот простой подход, который дает краткое представление о том, как вы можете передавать данные в запущенный метод из таймера.

[*Swift 3*]

```
Timer.scheduledTimer(timeInterval: 1.0, target: self, selector:#selector(iGotCall(sender:)),
userInfo: ["Name": "i am iOS guy"], repeats:true)
```

[*Цель - C*]

```
NSTimer* timer = [NSTimer scheduledTimerWithTimeInterval:1.0
                                                  target:self
                                                selector:@selector(iGotCall:)
                                                userInfo:@"i am iOS guy" repeats:YES];
```

Вышеупомянутая строка кода, передающая `["Name": "i am iOS guy"]` в `userInfo` . Итак, теперь, когда `iGotCall` получает вызов, вы можете получить переданное значение в соответствии с фрагментом кода.

[*Swift 3*]

```
func iGotCall(sender: Timer) {
        print((sender.userInfo)!)
    }
```

[*Цель - C*]

```
- (void)iGotCall:(NSTimer*)theTimer {
    NSLog (@"%@", (NSString*)[theTimer userInfo]);
}
```

Прочитайте NSTimer онлайн: https://riptutorial.com/ru/ios/topic/2624/nstimer

глава 54: NSURL

Examples

Как получить последний строковый компонент из NSURL String.

```
NSURL *url = [NSURL URLWithString:@"http://www.example.com/images/apple-tree.jpg"];
NSString *fileName = [url lastPathComponent];
// fileName = "apple-tree.jpg"
```

Как получить последний компонент строки из URL (NSURL) в Swift

Swift 2.3

```
let url = NSURL(string: "http://google.com/lastPath")
let lastPath = url?.lastPathComponent
```

Swift 3.0

```
  let url = URL(string: "http://google.com/lastPath")
  let lastPath = url?.lastPathComponent
```

Прочитайте NSURL онлайн: https://riptutorial.com/ru/ios/topic/4610/nsurl

глава 55: NSURLConnection

Examples

Методы делегата

// совместим протокол NSURLConnectionDelegate.

```
@interface ViewController : UIViewController<NSURLConnectionDelegate>
{
    NSMutableData *_responseData;
}
```

// Реализация методов протокола NSURLConnection.

```
#pragma mark NSURLConnection Delegate Methods

- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
{
    // A response has been received, this is where we initialize the instance var you created
    // so that we can append data to it in the didReceiveData method
    // Furthermore, this method is called each time there is a redirect so reinitializing it
    // also serves to clear it
    _responseData = [[NSMutableData alloc] init];
}

- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data {
    // Append the new data to the instance variable you declared
    [_responseData appendData:data];
}

- (NSCachedURLResponse *)connection:(NSURLConnection *)connection
                  willCacheResponse:(NSCachedURLResponse*)cachedResponse {
    // Return nil to indicate not necessary to store a cached response for this connection
    return nil;
}

- (void)connectionDidFinishLoading:(NSURLConnection *)connection {
    // The request is complete and data has been received
    // You can parse the stuff in your instance variable now

}

- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error {
    // The request has failed for some reason!
    // Check the error var
}
```

Синхронный запрос

```
NSURLRequest * urlRequest = [NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://google.com"]];
```

```
NSURLResponse * response = nil;
NSError * error = nil;
NSData * data = [NSURLConnection sendSynchronousRequest:urlRequest
                                          returningResponse:&response
                                                      error:&error];

if (error == nil)
{
    // Parse data here
}
```

Асинхронный запрос

```
// Create the request instance.
NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://google.com"]];

// Create url connection and fire request
NSURLConnection *conn = [[NSURLConnection alloc] initWithRequest:request delegate:self];
```

Прочитайте NSURLConnection онлайн: https://riptutorial.com/ru/ios/topic/6004/nsurlconnection

глава 56: NSURLSession

замечания

Класс **NSURLSession** и связанные классы предоставляют API для загрузки содержимого. Этот API предоставляет богатый набор методов делегирования для поддержки аутентификации и дает вашему приложению возможность выполнять фоновое скачивание, когда ваше приложение не работает, или в iOS, пока ваше приложение приостановлено.

На высоком уровне **NSURLSession** основан на концепции сеансов и задач. Задача представляет собой одиночный запрос для одного URL-адреса (или одной загрузки для одного URL-адреса). Сессия представляет собой группу связанных запросов.

Операционная система предоставляет один предшествующий сеанс - общий сеанс, который в основном работает как NSURLConnection. Кроме того, вы можете создавать свои собственные сеансы в своем приложении по мере необходимости.

Различные приложения используют сеансы по-разному. Многие приложения создают один сеанс при запуске и просто повторно используют его. Другие приложения получают возможность отменить группу связанных задач (например, веб-браузер, отменяющий все невыполненные запросы при закрытии вкладки) и, таким образом, создать один сеанс для хранения каждой группы связанных запросов.

Первым шагом при использовании NSURLSession является создание объекта конфигурации сеанса. Объект (обычно) многократного использования содержит различные параметры сеанса, которые вы можете настроить для своих конкретных потребностей, таких как максимальный параллелизм, дополнительные заголовки для отправки с каждым запросом, разрешать ли запросы отправлять по сотовому радио (только для iOS), тайм-ауты, хранилище учетных данных, минимальную версию TLS и даже настройки прокси.

Существует три типа конфигураций сеансов, в зависимости от того, как вы должны вести себя в результате:

- **Конфигурации по умолчанию** создают сеансы, которые очень похожи на NSURLConnection.
- **Фоновые конфигурации** создают сеансы, в которых запросы происходят вне процесса, что позволяет продолжить скачивание, даже если приложение больше не работает.
- **Эфемерные конфигурации** создают сеансы, которые не кэшируют что-либо на диск, не хранят файлы cookie на диске и т. Д. И, следовательно, подходят для поддержки таких вещей, как окна браузера инкогнито.

Когда вы создаете конфигурацию фона, вы должны предоставить идентификатор сеанса, который позволит вам повторно связать фоновый сеанс позже (если ваше приложение выйдет или приостановлено или завершено ОС). У вас не должно быть более одного экземпляра сеанса с тем же идентификатором, который активен в вашем приложении, поэтому, как правило, эти конфигурации не могут использоваться повторно. Все остальные конфигурации сеансов можно использовать повторно, чтобы создать столько сеансов, сколько хотите. Поэтому, если вам нужно создать несколько сеансов с аналогичными настройками, вы можете создать конфигурацию один раз и повторно использовать ее каждый раз, когда вы создаете новый сеанс.

После создания сеанса вы можете создавать задачи в этом сеансе. Существует три типа задач:

- **Задачи** данных возвращают данные как объект **NSData** . Они подходят для общего использования, но не поддерживаются в фоновых сеансах.
- **Загрузка задач** возвращает данные в виде файла на диске. Они подходят для больших запросов или для использования в фоновых сеансах.
- **Загрузка задач** загружает данные из объекта NSData или из файла на диске. Вы предоставляете объект данных или файл, который предоставляет тело POST. Данные / файл тела, которые вы предоставляете в задаче, переопределяют любые данные тела / файл, предоставленные в объекте NSURLRequest (если применимо).

Каждый из этих типов позволяет получить данные ответа несколькими различными способами - либо с помощью обратных вызовов на основе блоков, либо путем предоставления делегата на сеансе и реализации методов делегирования.

Кроме того, NSURLSession позволяет вам предоставлять методы делегирования для обработки аутентификации, выполнения пользовательской обработки сертификатов TLS (как для сертификатов клиентов, так и для проверки сервера), изменения поведения кэширования и т. Д.

Examples

Простой запрос GET

```
    // define url
    let url = NSURL(string: "https://urlToGet.com")

    //create a task to get data from a url
    let task = NSURLSession.sharedSession().dataTaskWithURL(url!)
    {

       /*inside this block, we have access to NSData *data, NSURLResponse *response, and
NSError *error returned by the dataTaskWithURL() function*/
      (data, response, error) in

      if error == nil
```

```
      {
          // Data from the request can be manipulated here
      }
      else
      {
          // An error occurred
      }
    }

    //make the request
    task.resume()
```

Цель-C Создать задачу сеанса и данных

```
NSURL *url = [NSURL URLWithString:@"http://www.example.com/"];
NSURLSessionConfiguration *configuration = [NSURLSessionConfiguration
defaultSessionConfiguration];

// Configure the session here.

NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration];

[[session dataTaskWithURL:url
        completionHandler:^(NSData *data, NSURLResponse *response, NSError *error)
{
    // The response object contains the metadata (HTTP headers, status code)

    // The data object contains the response body

    // The error object contains any client-side errors (e.g. connection
    // failures) and, in some cases, may report server-side errors.
    // In general, however, you should detect server-side errors by
    // checking the HTTP status code in the response object.
}] resume];
```

Настройка конфигурации фона

Чтобы создать фоновый сеанс

```
 // Swift:
 let mySessionID = "com.example.bgSession"
 let bgSessionConfig =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(mySessionID)

 let session = NSURLSession(configuration: bgSessionConfig)

 // add tasks here


 // Objective-C:
 NSString *mySessionID = @"com.example.bgSession";
 NSURLSessionConfiguration *configuration =
     [NSURLSessionConfiguration backgroundSessionConfigurationWithIdentifier: mySessionID];
 NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration
                                                       delegate:self]
```

Кроме того, в iOS вы должны настроить поддержку для обработки возобновления приложения. При вызове приложения вашего приложения `application:handleEventsForBackgroundURLSession:completionHandler:` метод (Objective-C) или `application(_:handleEventsForBackgroundURLSession:completionHandler:)` (Swift) вызывается, это означает, что ваше приложение было перезаписано в фоновом режиме для обработки активности в сеансе.

В этом методе вы должны создать новый сеанс с предоставленным идентификатором и настроить его с помощью делегата для обработки событий так же, как обычно на переднем плане. Кроме того, вы должны сохранить предоставленный обработчик завершения в словаре, используя сеанс в качестве ключа.

Когда `URLSessionDidFinishEventsForBackgroundURLSession:` делегата `URLSessionDidFinishEventsForBackgroundURLSession:` (Obj-C) / `URLSessionDidFinishEventsForBackgroundURLSession` (Swift) `URLSessionDidFinishEventsForBackgroundURLSession` чтобы сообщить вам, что больше не нужно обрабатывать события, ваше приложение должно искать обработчик завершения для этого сеанса, удалить сеанс из словаря и вызовите обработчик завершения, тем самым сообщив операционной системе, что у вас больше нет какой-либо выдающейся обработки, связанной с сеансом. (Если вы по-прежнему делаете что-то по какой-то причине, когда получаете этот вызов делегата, подождите, пока не закончите.) Как только вы вызываете этот метод, фоновый сеанс немедленно становится недействительным.

Если ваше приложение затем получает `application:application:didFinishLaunchingWithOptions:` call (вероятно, указывая на то, что пользователь, `application:application:didFinishLaunchingWithOptions:` вашего приложения, когда вы были заняты обработкой фоновых событий), безопасно создать фоновый сеанс с тем же идентификатором, поскольку старый сеанс с этим идентификатор больше не существует.

Если вам интересно узнать подробности, на высоком уровне, когда вы создаете фоновый сеанс, вы делаете две вещи:

- Создание сеанса во внешнем демоне (nsurlsessiond) для обработки загрузок
- Создание сеанса в вашем приложении, который ведет переговоры с этим внешним демоном через NSXPC

Как правило, опасно создавать два сеанса с одним и тем же идентификатором сеанса в одном запуске приложения, потому что оба они пытаются поговорить с тем же сеансом в фоновом демоне. Вот почему официальная документация говорит, что никогда не создавать несколько сеансов с тем же идентификатором. Однако, если первый сеанс был временным сеансом, созданным как часть вызова `handleEventsForBackgroundURLSession` , связь между недействительным сеансом in-app и сеансом в фоновом демонах больше не существует.

Отправка запроса POST с помощью аргументов с использованием

NSURLSession в Objective-C

Существует два распространенных способа кодирования тела запроса POST: кодировка URL (application / x-www-form-urlencoded) и данные формы (multipart / form-data). Большая часть кода аналогична, но способ построения данных тела отличается.

Отправка запроса с использованием URL-кодирования

Будь то у вас есть сервер для вашего небольшого приложения или ваша работа в команде с полным инженером-коннектором, вы захотите поговорить с этим сервером в какой-то момент с вашим приложением iOS.

В следующем коде мы будем составлять строку аргументов, которые сценарий целевого сервера будет использовать, чтобы сделать что-то, что изменяется в зависимости от вашего случая. Например, мы можем отправить строку:

> Имя = Брендон и пароль = ABCDE

На сервер, когда пользователь подписывается на ваше приложение, поэтому сервер может хранить эту информацию в базе данных.

Давайте начнем. Вы захотите создать запрос POST NSURLSession со следующим кодом.

```
// Create the configuration, which is necessary so we can cancel cacheing amongst other
things.
NSURLSessionConfiguration * defaultConfigObject = [NSURLSessionConfiguration
defaultSessionConfiguration];
// Disables cacheing
defaultConfigObject.requestCachePolicy = NSURLRequestReloadIgnoringLocalCacheData;
NSURLSession * defaultSession = [NSURLSession sessionWithConfiguration:defaultConfigObject
delegate:self delegateQueue:[NSOperationQueue mainQueue]];

NSString * scriptURL = [NSString stringWithFormat:@"https://server.io/api/script.php"];
//Converts the URL string to a URL usable by NSURLSession
NSMutableURLRequest * urlRequest = [NSMutableURLRequest requestWithURL:[NSURL
URLWithString:scriptURL]];
NSString * postDataString = [NSString stringWithFormat:@"name=%@&password=%@", [self
nameString], [self URLEncode:passwordString]];
[urlRequest setHTTPMethod:@"POST"];
[urlRequest setHTTPBody:[postDataString dataUsingEncoding:NSUTF8StringEncoding]];

NSURLSessionDataTask * dataTask = [defaultSession dataTaskWithRequest:urlRequest];
// Fire the data task.
[dataTask resume];
```

Вышеприведенный код только что создал и отправил запрос POST на сервер. Помните, что URL-адрес сценария и строка данных POST изменяются в зависимости от вашей ситуации. Если вы читаете это, вы будете знать, что заполнить эти переменные.

Вам также потребуется добавить небольшой метод, который кодирует URL:

```
- (NSString *)URLEncode:(NSString *)originalString encoding:(NSStringEncoding)encoding
{
    return (__bridge_transfer NSString *)CFURLCreateStringByAddingPercentEscapes(
        kCFAllocatorDefault,
        (__bridge CFStringRef)originalString,
        NULL,
        CFSTR(":/?#[]@!$&'()*+,;="),
        CFStringConvertNSStringEncodingToEncoding(encoding));
}
```

Таким образом, когда сервер завершит обработку этих данных, он отправит возврат в ваше приложение iOS. Поэтому нам нужно обработать это возвращение, но как?

Мы используем программирование, основанное на событиях, и используем методы делегатов NSURLSession. Это означает, что когда сервер отправляет ответ, эти методы начнут запускать. Следующие 5 методов - это те, которые будут запускаться во всем запросе ENTIRE, каждый раз, когда он делается:

```
- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveResponse:(NSURLResponse *)response
 completionHandler:(void (^)(NSURLSessionResponseDisposition disposition))completionHandler;

- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveData:(NSData *)data;

- (void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didCompleteWithError:(NSError *)error;

- (void)URLSession:(NSURLSession *)session didReceiveChallenge:(NSURLAuthenticationChallenge
*)challenge completionHandler:(void (^)(NSURLSessionAuthChallengeDisposition, NSURLCredential
*))completionHandler;

- (void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didReceiveChallenge:(NSURLAuthenticationChallenge *)challenge completionHandler:(void
(^)(NSURLSessionAuthChallengeDisposition, NSURLCredential * _Nullable))completionHandler;
```

Ниже вы увидите вышеупомянутые методы, используемые в контексте. Каждая из их целей довольно понятна благодаря Apple, но я все равно прокомментировал их использование:

```
// Response handling delegates
- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveResponse:(NSURLResponse *)response
 completionHandler:(void (^)(NSURLSessionResponseDisposition disposition))completionHandler{
    // Handler allows us to receive and parse responses from the server
    completionHandler(NSURLSessionResponseAllow);
}

- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveData:(NSData *)data{

    // Parse the JSON that came in into an NSDictionary
    NSError * err = nil;
    NSDictionary * jsonDict = [NSJSONSerialization JSONObjectWithData:data
options:NSJSONReadingAllowFragments error:&err];

    if (!err){ // if no error occurred, parse the array of objects as normal
```

```
        // Parse the JSON dictionary 'jsonDict' here
    }else{ // an error occurred so we need to let the user know
        // Handle your error here
    }
}

// Error handling delegate
- (void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didCompleteWithError:(NSError *)error{
    if(error == nil){
        // Download from API was successful
        NSLog(@"Data Network Request Did Complete Successfully.");
    }else{
        // Describes and logs the error preventing us from receiving a response
        NSLog(@"Error: %@", [error userInfo]);

        // Handle network error, letting the user know what happened.
    }
}

// When the session receives a challenge (because of iOS 9 App Transport Security blocking
non-valid SSL certificates) we use the following methods to tell NSURLSession "Chill out, I
can trust me".
// The following is not necessary unless your server is using HTTP, not HTTPS

- (void)URLSession:(NSURLSession *)session didReceiveChallenge:(NSURLAuthenticationChallenge
*)challenge completionHandler:(void (^)(NSURLSessionAuthChallengeDisposition, NSURLCredential
*))completionHandler{
    if([challenge.protectionSpace.authenticationMethod
isEqualToString:NSURLAuthenticationMethodServerTrust]){
        if([challenge.protectionSpace.host isEqualToString:@"DomainNameOfServer.io"]){
            NSURLCredential * credential = [NSURLCredential
credentialForTrust:challenge.protectionSpace.serverTrust];
            completionHandler(NSURLSessionAuthChallengeUseCredential,credential);
        }
    }
}

- (void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didReceiveChallenge:(NSURLAuthenticationChallenge *)challenge completionHandler:(void
(^)(NSURLSessionAuthChallengeDisposition, NSURLCredential * _Nullable))completionHandler{
    if([challenge.protectionSpace.authenticationMethod
isEqualToString:NSURLAuthenticationMethodServerTrust]){
        if([challenge.protectionSpace.host isEqualToString:@"DomainNameOfServer.io"]){
            NSURLCredential * credential = [NSURLCredential
credentialForTrust:challenge.protectionSpace.serverTrust];
            completionHandler(NSURLSessionAuthChallengeUseCredential,credential);
        }
    }
}
```

Итак, это все! Это код, который вам нужен для отправки, получения и анализа запроса API в iOS 9! Хорошо ... это было довольно много кода. Но если они реализованы правильно, как указано выше, это будет безотказно! Обязательно всегда обращайтесь с ошибками, которые были предложены выше.

Отправка запроса с использованием формы

Кодировка URL - это широко совместимый способ кодирования произвольных данных. Тем не менее, это относительно неэффективно для загрузки двоичных данных (например, фотографий), потому что каждый байт без ASCII превращается в трехсимвольный код. Он также не поддерживает вложения файлов, поэтому вам придется передавать имена файлов и данные файла в виде отдельных полей.

Предположим, мы хотим загрузить фотографию таким образом, чтобы она была эффективной и фактически выглядела как файл на стороне сервера. Один из способов сделать это - использовать кодировку формы. Для этого отредактируйте код, создающий NSURLSession, следующим образом:

```
UIImage * imgToSend;

// 2nd parameter of UIImageJPEGRepresentation represents compression quality. 0 being most
compressed, 1 being the least
// Using 0.4 likely stops us hitting the servers upload limit and costs us less server space
NSData * imageData = UIImageJPEGRepresentation(imgToSend, 0.4f);

// Alternatively, if the photo is on disk, you can retrieve it with
// [NSData dataWithContentsOfURL:...]

// Set up the body of the POST request.

// This boundary serves as a separator between one form field and the next.
// It must not appear anywhere within the actual data that you intend to
// upload.
NSString * boundary = @"---------------------------14737809831466499882746641449";

// Body of the POST method
NSMutableData * body = [NSMutableData data];

// The body must start with the boundary preceded by two hyphens, followed
// by a carriage return and newline pair.
//
// Notice that we prepend two additional hyphens to the boundary when
// we actually use it as part of the body data.
//
[body appendData:[[NSString stringWithFormat:@"\r\n--%@\r\n",boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

// This is followed by a series of headers for the first field and then
// TWO CR-LF pairs.
[body appendData:[[NSString stringWithFormat:@"Content-Disposition: form-data;
name=\"tag_name\"\r\n\r\n"] dataUsingEncoding:NSUTF8StringEncoding]];

// Next is the actual data for that field (called "tag_name") followed by
// a CR-LF pair, a boundary, and another CR-LF pair.
[body appendData:[strippedCompanyName dataUsingEncoding:NSUTF8StringEncoding]];
[body appendData:[[NSString stringWithFormat:@"\r\n--%@\r\n", boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

// Encode the filename and image data as the "userfile" CGI parameter.
// This is similar to the previous field, except that it is being sent
// as an actual file attachment rather than a blob of data, which means
// it has both a filename and the actual file contents.
//
// IMPORTANT: The filename MUST be plain ASCII (and if encoded like this,
```

```
//            must not include quotation marks in the filename).
//
NSString * picFileName = [NSString stringWithFormat:@"photoName"];
NSString * appendDataString = [NSString stringWithFormat:@"Content-Disposition: form-data;
name=\"userfile\"; filename=\"%@.jpg\"\r\n", picFileName];
[body appendData:[appendDataString dataUsingEncoding:NSUTF8StringEncoding]];
[body appendData:[@"Content-Type: application/octet-stream\r\n\r\n"
dataUsingEncoding:NSUTF8StringEncoding]];
[body appendData:[NSData dataWithData:imageData]];

// Close the request body with one last boundary with two
// additional hyphens prepended **and** two additional hyphens appended.
[body appendData:[[NSString stringWithFormat:@"\r\n--%@--\r\n", boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

// Create the session
// We can use the delegate to track upload progress and disable cacheing
NSURLSessionConfiguration * defaultConfigObject = [NSURLSessionConfiguration
defaultSessionConfiguration];
defaultConfigObject.requestCachePolicy = NSURLRequestReloadIgnoringLocalCacheData;
NSURLSession * defaultSession = [NSURLSession sessionWithConfiguration: defaultConfigObject
delegate: self delegateQueue: [NSOperationQueue mainQueue]];

// Data uploading task.
NSURL * url = [NSURL URLWithString:@"https://server.io/api/script.php"];
NSMutableURLRequest * request = [NSMutableURLRequest requestWithURL:url];
NSString * contentType = [NSString stringWithFormat:@"multipart/form-data;
boundary=%@",boundary];
[request addValue:contentType forHTTPHeaderField:@"Content-Type"];
request.HTTPMethod = @"POST";
request.HTTPBody = body;
NSURLSessionDataTask * uploadTask = [defaultSession dataTaskWithRequest:request];
[uploadTask resume];
```

Это создает и запускает запрос NSURLSession так же, как и раньше, и в результате методы делегата будут вести себя точно так же. Убедитесь, что сценарий, на который отправляется изображение (расположен на url в `url` переменной), ожидает изображение и может его правильно проанализировать.

Прочитайте NSURLSession онлайн: https://riptutorial.com/ru/ios/topic/2009/nsurlsession

глава 57: NSUserActivity

Вступление

Объект NSUserActivity может использоваться для координации значимых событий в приложении с системой. Он является основой для передачи обслуживания между различными устройствами под управлением iOS и macOS. Кроме того, он также может использоваться для улучшения индексации и увеличения индексации или создания результатов поиска Spotlight для приложения. Начиная с iOS 10, он также может использоваться для координации взаимодействия между вашим приложением и Siri с использованием SiriKit.

замечания

Виды деятельности

Поддерживаемые типы активности должны быть определены в файле `Info.plist` вашего приложения под ключом `NSUserActivityTypes` . Действия привязаны к вашему идентификатору Team Developer Team, что означает, что координация действий ограничена между приложениями, имеющими один и тот же идентификатор команды (например, «Safari» не может принять действие Handoff из «Chrome» или наоборот).

Становление / списание текущей деятельности

Маркировка активности в качестве текущей с помощью функции `becomeCurrent` делает ее доступной для Handoff или Spotlight Indexing. Одновременно может быть только одно действие. Вы можете отметить действие как неактивное без аннулирования, вызвав `resignCurrent` .

Если вы `invalidate` действие, один и тот же экземпляр не может быть снова выполнен.

Не указывайте активность как текущую, предоставляя ее для SiriKit .

Поиск индексирования

Деятельность **не** должна использоваться как механизм индексации общего назначения в вашем приложении. Вместо этого они должны использоваться только в ответ на

инициируемые пользователем действия. Чтобы индексировать весь контент в вашем приложении, используйте CoreSpotlight.

Examples

Создание NSUserActivity

Чтобы создать объект `NSUserActivity` , ваше приложение должно объявить типы поддерживаемых им действий в файле `Info.plist` . Поддерживаемые действия определяются вашим приложением и должны быть уникальными. Активность определяется с использованием схемы именования стиля обратного домена (например, com.panyName.productName.activityName). Вот что может выглядеть запись в вашем Info.plist:

ключ	Значение
NSUserActivityTypes	[Массив]
- item0	com.companyName.productName.activityName01
- item1	com.companyName.productName.activityName02

После того, как вы определили все поддерживаемые типы активности, вы можете начать доступ к ним и использовать их в коде приложения.

Чтобы создать объект `NSUserActivity` вы должны сделать следующее

```
// Initialize the activity object and set its type from one of the ones specified in your
app's plist
NSUserActivity *currentActivity = [[NSUserActivity alloc]
initWithActivityType:@"com.companyName.productName.activityName01"];

// Set the title of the activity.
// This title may be displayed to the user, so make sure it is localized and human-readable
currentActivity.title = @"Current Activity";

// Configure additional properties like userInfo which will be included in the activity
currentActivity.userInfo = @{@"informationKey" : @"value"};

// Configure the activity so the system knows what may be done with it
// It is important that you only set YES to tasks that your application supports
// In this example, we will only enable the activity for use with Handoff
[currentActivity setEligibleForHandoff:YES];
[currentActivity setEligibleForSearch:NO]; // Defaults to NO
[currentActivity setEligibleForPublicIndexing:NO]; // Defaults to NO

// Set this activity as the current user activity
// Only one activity may be current at a time on a device. Calling this method invalidates any
other current activities.
[currentActivity becomeCurrent];
```

После этого вышеуказанная деятельность должна быть доступна для передачи обслуживания (хотя для правильной обработки «Handoff» требуется больше работы).

Прочитайте NSUserActivity онлайн: https://riptutorial.com/ru/ios/topic/10716/nsuseractivity

глава 58: NSUserDefaults

Синтаксис

- `UserDefaults.standard.set(dic, forKey: "LoginSession") //Save value inside userdefaults`
    - `UserDefaults.standard.object(forKey: "LoginSession") as? [String:AnyObject] ?? [:] //Get value from UserDefaults`

замечания

NSUserDefault, которые используются для хранения всего типа DataType, и вы можете получить его значение в любом месте класса приложения. **NSUserDefault**

Examples

Установка значений

Чтобы установить значение в `NSUserDefaults` , вы можете использовать следующие функции:

Swift <3

```
setBool(_:forKey:)
setFloat(_:forKey:)
setInteger(_:forKey:)
setObject(_:forKey:)
setDouble(_:forKey:)
setURL(_:forKey:)
```

Swift 3

В Swift 3 имена функций изменяются для `set` insted of `set` заданного типом.

```
set(_:forKey:)
```

Objective-C

```
-(void)setBool:(BOOL)value forKey:(nonnull NSString *)defaultName;
-(void)setFloat:(float)value forKey:(nonnull NSString *)defaultName;
-(void)setInteger:(NSInteger)value forKey:(nonnull NSString *)defaultName;
-(void)setObject:(nullable id)value forKey:(nonnull NSString *)defaultName;
-(void)setDouble:(double)value forKey:(nonnull NSString *)defaultName;
-(void)setURL:(nullable NSURL *)value forKey:(nonnull NSString *)defaultName;
```

Пример использования:

Swift <3

```
NSUserDefaults.standardUserDefaults.setObject("Netherlands", forKey: "HomeCountry")
```

Swift 3

```
UserDefaults.standard.set("Netherlands", forKey: "HomeCountry")
```

Objective-C

```
[[NSUserDefaults standardUserDefaults] setObject:@"Netherlands" forKey:@"HomeCountry"];
```

Пользовательские объекты

Чтобы сохранить пользовательские объекты в `NSUserDefaults`, вам нужно сделать свой CustomClass подтвержденным протоколом` NSCoding`. Вам необходимо реализовать следующие методы:

стриж

```
public func encodeWithCoder(aCoder: NSCoder) {
    aCoder.encodeObject(name, forKey:"name")
    aCoder.encodeObject(unitId, forKey: "unitId")
}

required public init(coder aDecoder: NSCoder) {
    super.init()
    name = aDecoder.decodeObjectForKey("name") as? String
    unitId = aDecoder.decodeIntegerForKey("unitId") as? NSInteger
}
```

Objective-C

```
- (id)initWithCoder:(NSCoder *)coder {
    self = [super init];
    if (self) {
        name = [coder decodeObjectForKey:@"name"];
        unitId = [coder decodeIntegerForKey:@"unitId"];
    }
    return self;
}
```

```
- (void)encodeWithCoder:(NSCoder*)coder {
    [coder encodeObject:name forKey:@"name"];
    [coder encodeInteger:unitId forKey:@"unitId"];
}
```

Получение значений по умолчанию

Чтобы получить значение в NSUserDefaults, вы можете использовать следующие функции:

стриж

```
arrayForKey(_:)
boolForKey(_:)
dataForKey(_:)
dictionaryForKey(_:)
floatForKey(_:)
integerForKey(_:)
objectForKey(_:)
stringArrayForKey(_:)
stringForKey(_:)
doubleForKey(_:)
URLForKey(_:)
```

Objective-C

```
-(nullable NSArray *)arrayForKey:(nonnull NSString *)defaultName;
-(BOOL)boolForKey:(nonnull NSString *)defaultName;
-(nullable NSData *)dataForKey:(nonnull NSString *)defaultName;
-(nullable NSDictionary<NSString *, id> *)dictionaryForKey:(nonnull NSString *)defaultName;
-(float)floatForKey:(nonnull NSString *)defaultName;
-(NSInteger)integerForKey:(nonnull NSString *)defaultName;
-(nullable id)objectForKey:(nonnull NSString *)key;
-(nullable NSArray<NSString *> *)stringArrayForKey:(nonnull NSString *)defaultName;
-(nullable NSString *)stringForKey:(nonnull NSString *)defaultName;
-(double)doubleForKey:(nonnull NSString *)defaultName;
-(nullable NSURL *)URLForKey:(nonnull NSString *)defaultName;
```

Пример использования:

стриж

```
let homeCountry = NSUserDefaults.standardUserDefaults().stringForKey("HomeCountry")
```

Objective-C

```
NSString *homeCountry = [[NSUserDefaults standardUserDefaults] stringForKey:@"HomeCountry"];
```

Сохранение значений

`NSUserDefaults` периодически записываются на диск системой, но есть моменты, когда вы хотите, чтобы ваши изменения были сохранены немедленно, например, когда приложение переходит в фоновое состояние. Это делается путем вызова `synchronize` .

стриж

```
NSUserDefaults.standardUserDefaults().synchronize()
```

Objective-C

```
[[NSUserDefaults standardUserDefaults] synchronize];
```

Использовать менеджеров для сохранения и чтения данных

Хотя вы можете использовать методы `NSUserDefaults` где угодно, иногда бывает лучше определить менеджера, который сохраняет и читает из `NSUserDefaults` для вас, а затем использовать этот менеджер для чтения или записи ваших данных.

Предположим, что мы хотим сохранить счет пользователя в `NSUserDefaults` . Мы можем создать класс, подобный приведенному ниже, который имеет два метода: `setHighScore` и `highScore` . В любом месте, где вы хотите получить доступ к высоким баллам, создайте экземпляр этого класса.

стриж

```
public class ScoreManager: NSObject {

    let highScoreDefaultKey = "HighScoreDefaultKey"

    var highScore = {
        set {
            // This method includes your implementation for saving the high score
            // You can use NSUserDefaults or any other data store like CoreData or
            // SQLite etc.

            NSUserDefaults.standardUserDefaults().setInteger(newValue, forKey:
highScoreDefaultKey)
            NSUserDefaults.standardUserDefaults().synchronize()
        }
        get {
        //This method includes your implementation for reading the high score

            let score =
NSUserDefaults.standardUserDefaults().objectForKey(highScoreDefaultKey)

            if (score != nil) {
                return score.integerValue;
            } else {
                //No high score available, so return -1
```

```
                return -1;
            }
        }
    }
}
```

Objective-C

```
#import "ScoreManager.h"

#define HIGHSCRORE_KEY @"highScore"

@implementation ScoreManager

- (void)setHighScore:(NSUInteger) highScore {
    // This method includes your implementation for saving the high score
    // You can use NSUserDefaults or any other data store like CoreData or
    // SQLite etc.

    [[NSUserDefaults standardUserDefaults] setInteger:highScore forKey:HIGHSCRORE_KEY];
    [[NSUserDefaults standardUserDefaults] synchronize];
}

- (NSInteger)highScore
{

    //This method includes your implementation for reading the high score

    NSNumber *highScore = [[NSUserDefaults standardUserDefaults] objectForKey:HIGHSCRORE_KEY];
    if (highScore) {
        return highScore.integerValue;
    }else
    {
        //No high score available, so return -1

        return -1;
    }

}

@end
```

Преимущества заключаются в следующем:

1. Реализация вашего процесса чтения и записи происходит только в одном месте, и вы можете изменить его (например, переключиться с `NSUserDefaults` на Core Data), когда захотите, и не беспокоиться об изменении всех мест, в которых вы работаете с высокой оценкой.
2. Просто позвоните только по одному методу, когда вы хотите получить доступ к счету или написать его.
3. Просто отлаживайте его, когда вы видите ошибку или что-то в этом роде.

Заметка

Если вас беспокоит синхронизация, лучше использовать одноэлементный класс, управляющий синхронизацией.

Очистка NSUserDefaults

стриж

```
let bundleIdentifier = NSBundle.mainBundle().bundleIdentifier()

NSUserDefaults.standardUserDefaults().removePersistentDomainForName(bundleIdentifier)
```

Objective-C

```
NSString *bundleIdentifier = [[NSBundle mainBundle] bundleIdentifier];

[[NSUserDefaults standardUserDefaults] removePersistentDomainForName: bundleIdentifier];
```

UserDefaults используется в Swift 3

Каждому приложению необходимо было хранить пользовательские сеансы или связанные с пользователем детали внутри приложения в UserDefaults.Так что мы сделали целую логику внутри класса для более эффективного управления UserDefaults.

Swift 3

```
import Foundation

public struct Session {

    fileprivate static let defaults = UserDefaults.standard

    enum userValues: String {
        case auth_token
        case email
        case fname
        case mobile
        case title
        case userId
        case userType
        case OTP
        case isApproved
    }


    //MARK: - Getting here User Details
    static func getUserSessionDetails()->[String:AnyObject]? {
        let dictionary = defaults.object(forKey: "LoginSession") as? [String:AnyObject]
```

```
        return  dictionary
    }

    //MARK: - Saving Device Token
    static func saveDeviceToken(_ token:String){
        guard (gettingDeviceToken() ?? "").isEmpty else {
            return
        }
        defaults.removeObject(forKey: "deviceToken")
        defaults.set(token, forKey: "deviceToken")
        defaults.synchronize()
    }

    //MARK: - Getting Token here
    static func gettingDeviceToken()->String?{
        let token = defaults.object(forKey: "deviceToken") as? String
        if token == nil{
            return ""
        }else{ return token}
    }

    //MARK: - Setting here User Details
    static func setUserSessionDetails(_ dic :[String : AnyObject]){
        defaults.removeObject(forKey: "LoginSession")
        defaults.set(dic, forKey: "LoginSession")
        defaults.synchronize()
    }

    //MARK:- Removing here all Default Values
    static func userSessionLogout(){
        //Set Activity
        defaults.removeObject(forKey: "LoginSession")
        defaults.synchronize()
    }

    //MARK: - Get value from session here
    static func getUserValues(value: userValues) -> String? {
        let dic = getUserSessionDetails() ?? [:]
        guard let value = dic[value.rawValue] else{
            return ""
        }
        return value as? String
    }

}
```

Использование класса UserDefaults

```
//Saving user Details
Session.setUserSessionDetails(json ?? [:])

//Retriving user Details
let userId = Session.getUserValues(value: .userId) ?? ""
```

Прочитайте NSUserDefaults онлайн: https://riptutorial.com/ru/ios/topic/3150/nsuserdefaults

глава 59: OpenGL

Вступление

OpenGL ES - это графическая библиотека, которую iOS использует для 3D-рендеринга.

Examples

Пример проекта

Пример проекта (Git repo), который можно использовать в качестве отправной точки для выполнения 3D-рендеринга. Код для настройки OpenGL и шейдеров довольно длинный и утомительный, поэтому он не будет хорошо соответствовать этому формату примера. Более поздние фрагменты могут быть показаны в отдельных примерах, в которых подробно описывается, что именно происходит с каждым фрагментом кода, но сейчас вот проект Xcode.

Прочитайте OpenGL онлайн: https://riptutorial.com/ru/ios/topic/9324/opengl

глава 60: plist iOS

Вступление

Plist используется для хранения данных в приложении iOS. Plist сохраняет данные в виде массива и словарей. В plist мы можем сохранить данные как: 1. Статические данные, которые будут использоваться в приложении. 2. Данные, которые будут поступать с сервера.

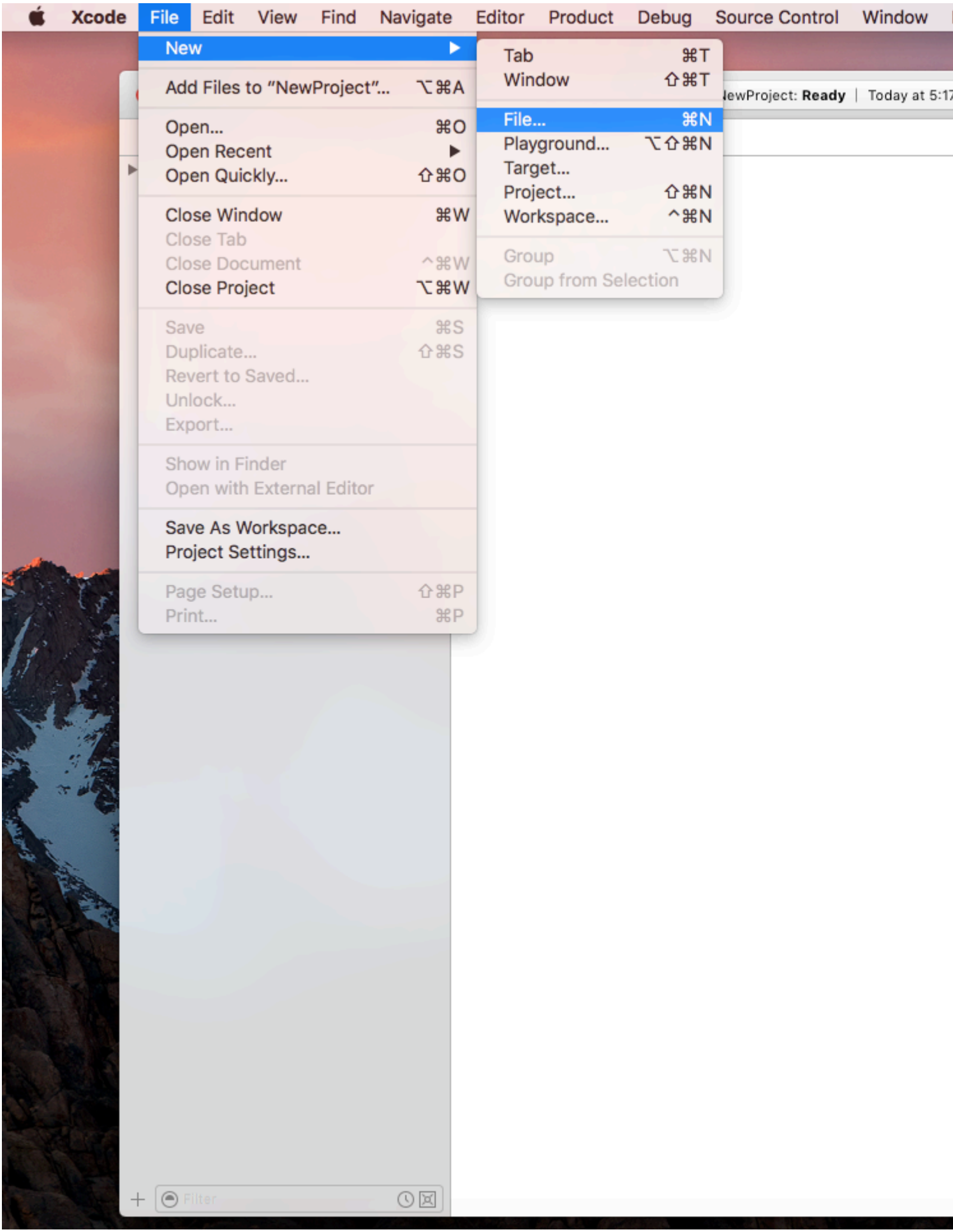
Examples

Пример:

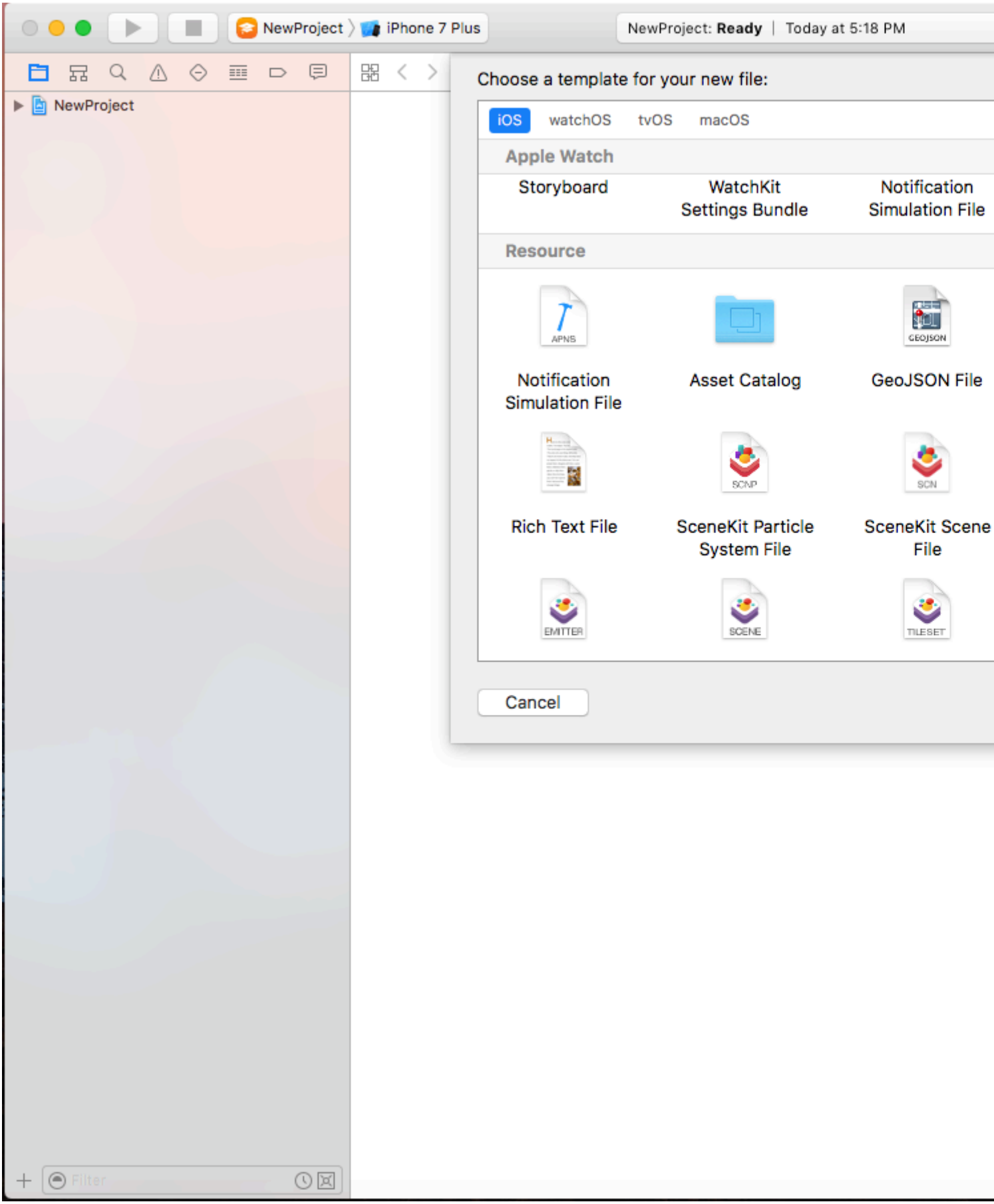
1. Статические данные, которые будут использоваться в приложении.

Чтобы сохранить статические данные в plist, выполните следующие действия:

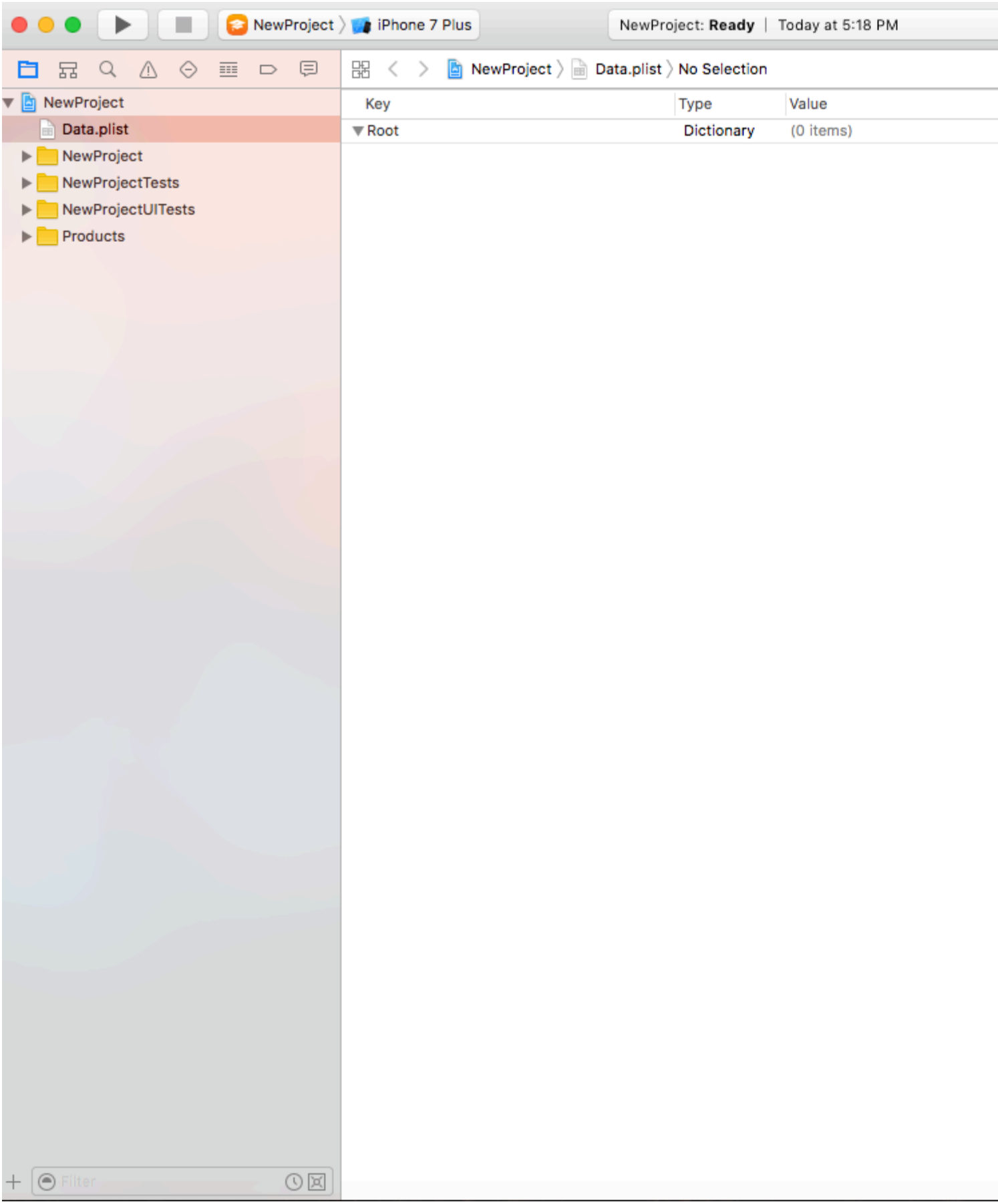
a) Добавить новый файл


Xcode
File
Edit
View
Find
Navigate
Editor
Product
Debug
Source Control
Window
New
Add Files to "NewProject"... ⌥⌘A
Open... ⌘O
Open Recent
Open Quickly... ⇧⌘O
Close Window ⌘W
Close Tab
Close Document ^⌘W
Close Project ⌥⌘W
Save ⌘S
Duplicate... ⇧⌘S
Revert to Saved...
Unlock...
Export...
Show in Finder
Open with External Editor
Save As Workspace...
Project Settings...
Page Setup... ⇧⌘P
Print... ⌘P
Tab ⌘T
Window ⇧⌘T
File... ⌘N
Playground... ⌥⇧⌘N
Target...
Project... ⇧⌘N
Workspace... ^⌘N
Group ⌥⌘N
Group from Selection
Filter

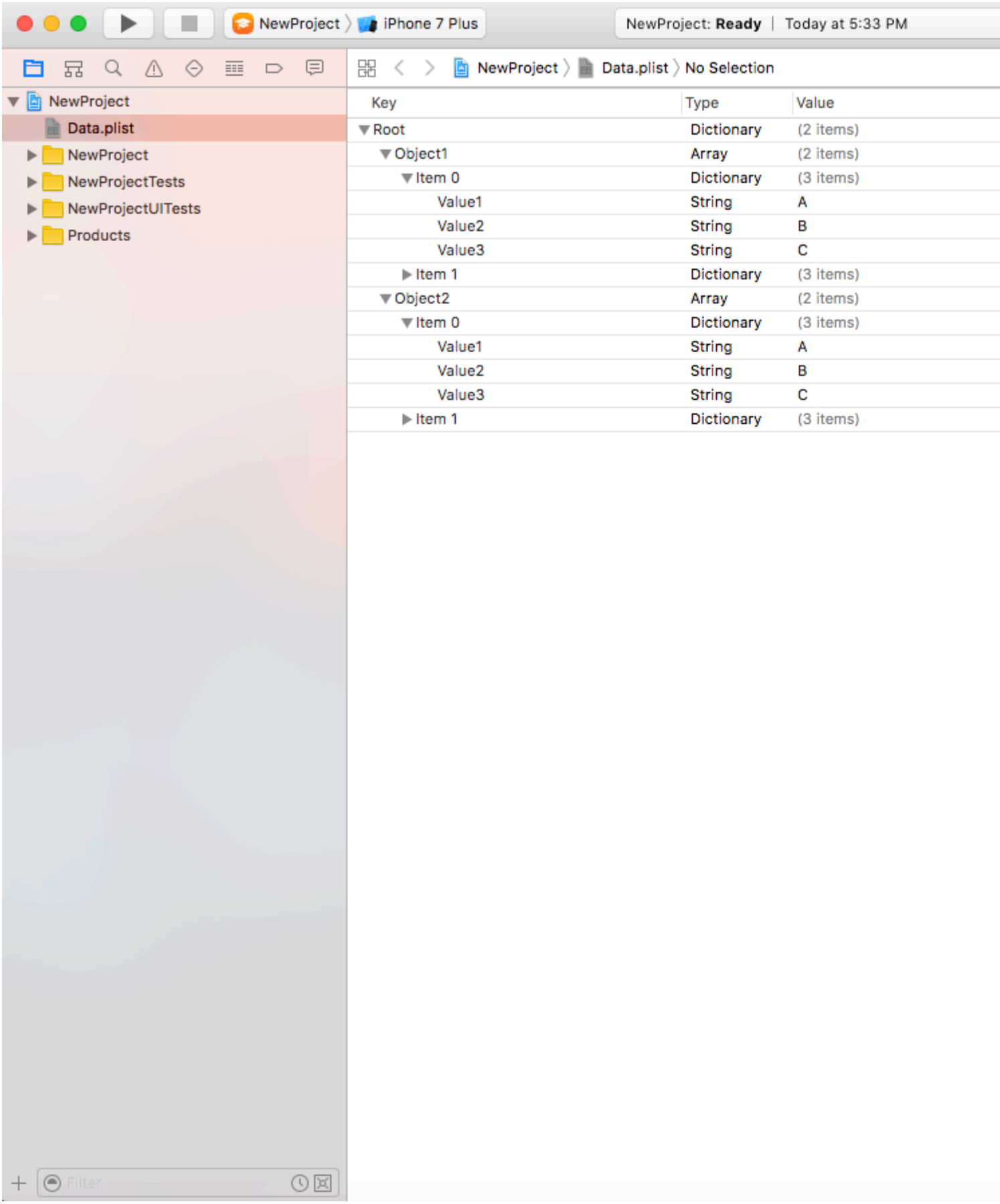
б) Нажмите «Список свойств» в разделе «Ресурсы»



c) Назовите список свойств, и файл будет создан как (data.plist здесь)



d) Вы можете создать слой массивов и словарей как:



// Прочитайте plist из комплекта и извлеките из него Root Dictionary

```
NSDictionary *dictRoot = [NSDictionary dictionaryWithContentsOfFile:[[NSBundle mainBundle]
pathForResource:@"Data" ofType:@"plist"]];
```

// Ваш словарь содержит массив словаря // Теперь вытащите Array из него.

```
NSArray *arrayList = [NSArray arrayWithArray:[dictRoot objectForKey:@"Object1"]];

for(int i=0; i< [arrayList count]; i++)
{
    NSMutableDictionary *details=[arrayList objectAtIndex:i];
}
```

Сохранять и редактировать / удалять данные из Plist

Вы уже создали слой. Этот планшет останется таким же в приложении. Если вы хотите отредактировать данные в этом plist, добавьте новые данные в plist или удалите данные из plist, вы не можете вносить изменения в этот файл.

Для этого вам нужно будет сохранить свой слой в Справочнике документов. Вы можете отредактировать свой plist, сохраненный в каталоге документа.

Сохраните plist в каталоге документа как:

```
NSString *filePath = [[NSBundle mainBundle] pathForResource:@"Data" ofType:@"plist"];

NSDictionary *dict = [[NSDictionary alloc] initWithContentsOfFile:filePath];

NSDictionary *plistDict = dict;

NSFileManager *fileManager = [NSFileManager defaultManager];

NSString *error = nil;

NSData *plistData = [NSPropertyListSerialization dataFromPropertyList:plistDict
format:NSPropertyListXMLFormat_v1_0 errorDescription:&error];

if (![fileManager fileExistsAtPath: plistPath]) {

    if(plistData)
    {
        [plistData writeToFile:plistPath atomically:YES];
    }
}
else
{

}
```

Получите данные Plist как:

```
    NSArray *paths = NSSearchPathForDirectoriesInDomains (NSDocumentDirectory,
NSUserDomainMask, YES);
    NSString *documentsPath = [paths objectAtIndex:0];
    NSString *plistPath = [documentsPath stringByAppendingPathComponent:@"Data.plist"];
    NSDictionary *dict = [[NSDictionary alloc] initWithContentsOfFile:plistPath];

    NSArray *usersArray = [dict objectForKey:@"Object1"];
```

Вы можете отредактировать удаление, добавить новые данные в соответствии с вашим требованием и снова сохранить plist в Document Directory.

Прочитайте plist iOS онлайн: https://riptutorial.com/ru/ios/topic/8141/plist-ios

глава 61: Sirikit

замечания

Различные типы запросов Siri

- Ride Booking (например, заведите меня в Нью-Йорк через MyApp)
- Сообщения (например, отправьте текст Джону с помощью MyApp)
- Поиск фотографий (например, посмотрите фотографии на пляже, сделанные прошлым летом в MyApp)
- Платежи (например, отправляйте 20 долларов США Джону за ужин прошлой ночью, используя MyApp)
- VoIP-вызов (например, Call Mike на моем MyApp)
- Тренировки (например, начать мою ежедневную тренировку из MyApp)
- Климат и радио (специально для CarPlay, например, установите нагреватель на 72 градуса)

Examples

Добавление расширения Siri в приложение

Чтобы интегрировать возможности Siri в ваше приложение, вы должны добавить расширения, как это было бы при создании виджета iOS 10 (старое расширение Today View) или пользовательскую клавиатуру.

Добавление возможностей

1- В настройках проекта выберите цель приложения для iOS и перейдите на вкладку «Возможности»

2- Включение возможности Siri

Добавление расширения

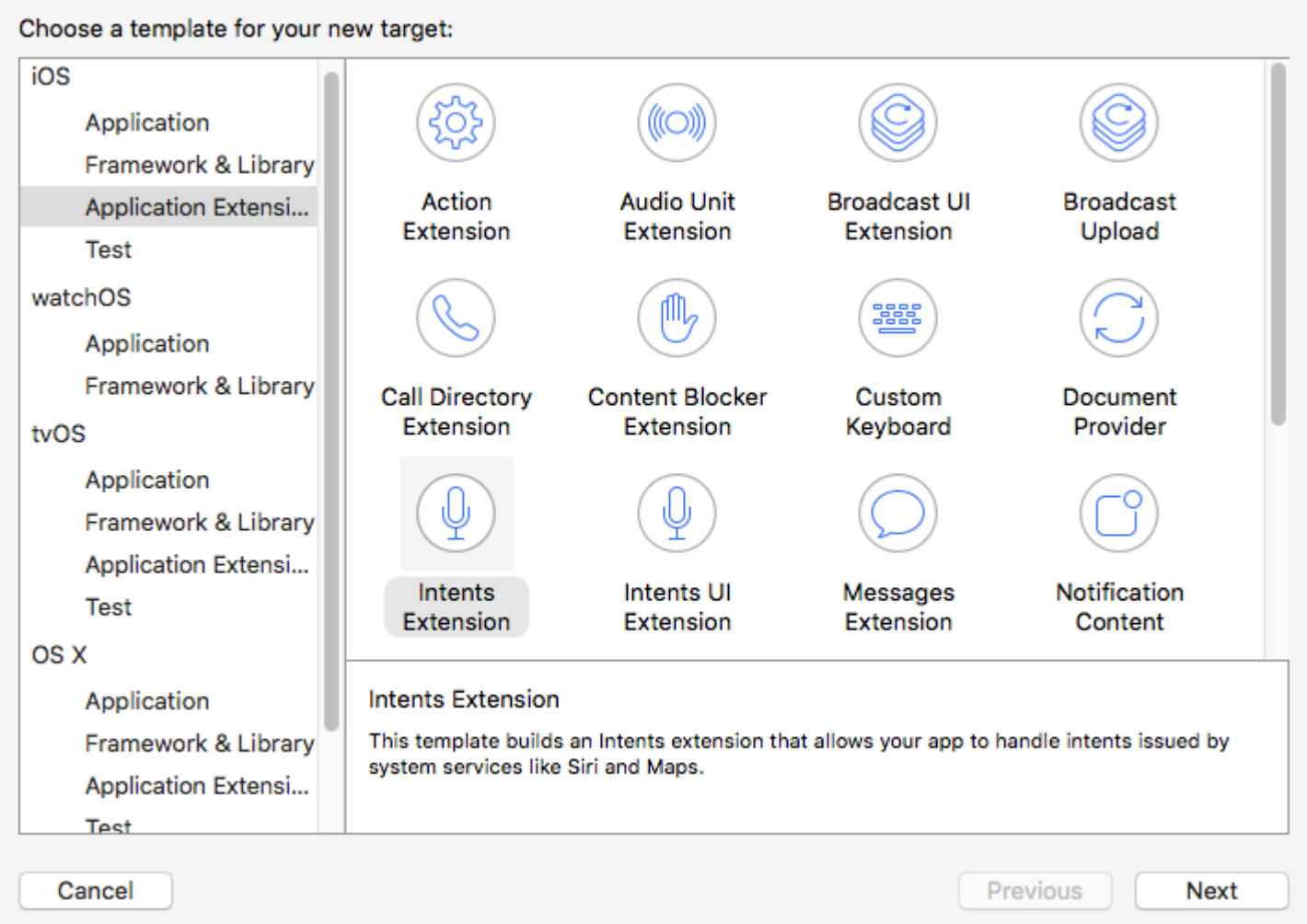
1- Перейдите в Файл -> Создать -> Цель ...

2- Выберите iOS -> Расширение приложения с левой панели

3- Дважды нажмите «Настроить» с правой стороны

По словам Apple:

> Шаблон Intents Extension создает расширение Intents, которое позволяет вашему приложению обрабатывать намерения, предоставляемые системными службами, такими как Siri и Maps.



4- Выберите имя и обязательно установите флажок «Включить расширение пользовательского интерфейса».

Выполняя эти шаги, создаются две новые цели (Intents Extension и UI Extension), и по умолчанию они содержат код Intout Workout. Для разных типов запросов Siri см. Примечания.

Заметка

В любое время, когда вы хотите отладить расширение, просто выберите схему Intent из доступных схем.

Заметка

Вы не можете тестировать приложения SiriKit в Simulator. Вместо этого вам нужно настоящее устройство.

Прочитайте Sirikit онлайн: https://riptutorial.com/ru/ios/topic/5869/sirikit

глава 62: SLComposeViewController

Examples

SLComposeViewController для Twitter, facebook, SinaWelbo и TencentWelbo

Objective-C

Сначала добавьте `Social Framework` в проект XCode.

Импортируйте класс `#import "Social/Social.h"` в требуемый ViewController

Twitter с текстом, изображением и ссылкой

```
//- - To Share text on twitter - -
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeTwitter])
{
        //Tweet
        SLComposeViewController *twitterVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTwitter];
        //To send link together with text
        [twitterVC addURL:[NSURL URLWithString:@"https://twitter.com/IbrahimH_ss_n"]];
        //To add a photo to a link
        [twitterVC addImage:[UIImage imageNamed:@"image"]];
        //Sending link and Image with the tweet
        [twitterVC setInitialText:text];
        /* While adding link and images in a tweet the effective length of a tweet i.e.
        the number of characters which can be entered by the user decreases.
        The default maximum length of a tweet is 140 characters*/
        [self presentViewController:twitterVC animated:YES completion:nil];
}
else
{//Shows alert if twitter is not signed in
       UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to twitter."preferredStyle:UIAlertControllerStyleAlert];
        [self presentViewController:alertCont animated:YES completion:nil];
        UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay"
style:UIAlertActionStyleDefault handler:nil];
        [alertCont addAction:okay];
}
}
```

Facebook с текстом, изображением и ссылкой

```
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeFacebook])
{
    SLComposeViewController *fbVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeFacebook];
    [fbVC setInitialText:text];
    //To send link together with text
    [fbVC addURL:[NSURL URLWithString:@"https://twitter.com/IbrahimH_ss_n"]];
    //To add a photo to a link
    [fbVC addImage:[UIImage imageNamed:@"image"]];
```

```
    [self presentViewController:fbVC animated:YES completion:nil];
}
else
{//Shows alert if twitter is not signed in
       UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to twitter."preferredStyle:UIAlertControllerStyleAlert];
        [self presentViewController:alertCont animated:YES completion:nil];
        UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay"
style:UIAlertActionStyleDefault handler:nil];
        [alertCont addAction:okay];
}
```

SinaWeibo

```
//- - SinaWeibo - -
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeSinaWeibo]){

    SLComposeViewController *SinaWeiboVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeSinaWeibo];
    [SinaWeiboVC setInitialText:text];

    [self presentViewController:SinaWeiboVC animated:YES completion:nil];
}
else
{
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to SinaWeibo."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay" style:UIAlertActionStyleDefault
handler:nil];
    [alertCont addAction:okay];
}
```

TencentWeibo

```
//- -TencentWeibo text share
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeTencentWeibo])
{
    SLComposeViewController *tencentWeiboVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTencentWeibo];
    [tencentWeibo setInitialText:text];
    [self presentViewController:tencentWeibo animated:YES completion:nil];
}
else
{
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to SinaWeibo."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay" style:UIAlertActionStyleDefault
handler:nil];
    [alertCont addAction:okay];
}
```

Прочитайте SLComposeViewController онлайн:
https://riptutorial.com/ru/ios/topic/7366/slcomposeviewcontroller

глава 63: StoreKit

Examples

Получить локализованную информацию о продукте из App Store

Получите локализованную информацию о продукте из набора строк идентификатора продукта, используя `SKProductsRequest` :

```
import StoreKit

let productIdentifierSet = Set(["yellowSubmarine", "pennyLane"])
let productsRequest = SKProductsRequest(productIdentifiers: productIdentifierSet)
```

Чтобы обрабатывать продукты из `productsRequest` , нам нужно назначить делегата запросу, который обрабатывает ответ. Делегат должен соответствовать протоколу `SKProductsRequestDelegate` , что означает, что он должен наследовать от `NSObject` (то есть любого объекта `Foundation`) и реализовать метод `productsRequest` :

```
class PaymentManager: NSObject, SKProductsRequestDelegate {

    var products: [SKProduct] = []

    func productsRequest(request: SKProductsRequest,
                         didReceiveResponse response: SKProductsResponse) {

        products = response.products

    }

}
```

Для того, чтобы начать `productsRequest` мы задаём `PaymentManager` в качестве делегата продукт-запрос, и вызвать `start()` метод по запросу:

```
let paymentManager = PaymentManager()
productsRequest.delegate = paymentManager
productsRequest.start()
```

Если запросы будут успешными, продукты будут представлены в `paymentManager.products` .

Прочитайте StoreKit онлайн: https://riptutorial.com/ru/ios/topic/6025/storekit

глава 64: Swift: изменение rootViewController в AppDelegate для представления основного или входного / входного потока

Вступление

Часто бывает полезно представить первый опыт для новых пользователей вашего приложения. Это может быть по каким-либо причинам, например, побудить их войти (если это требуется для вашей ситуации), объяснить, как использовать приложение, или просто информировать их о новых функциях в обновлении (как отмечает Notes, Photos и Music в iOS11).

замечания

Во-первых, поскольку вы имеете дело с несколькими потоками, здесь можно эффективно использовать раскадровки. По умолчанию ваше приложение использует `Main.storyboard` для вашего основного потока. Ваш бортовой / альтернативный поток может содержаться во вторичной раскадровке, например. `Onboarding.storyboard`

Это имеет ряд преимуществ:

- в команде разработчиков, работа над каждым потоком пользователя может быть отделена
- более четкое управление исходным кодом (git)
- разделение проблем

Когда приложение запускается, вы можете определить, какой поток должен быть представлен. Логика для этого может содержаться в вашем AppDelegate:

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    let isFirstRun = true // logic to determine goes here
    if isFirstRun {
        showOnboarding()
    }
    return true
}
```

Чтобы показать поток Onboarding, стоит подумать о том, как вы хотите справиться с опытом увольнения, как только человек, использующий его, завершил путешествие и

который семантически корректен для того, что вы пытаетесь создать.

подходы:

Двумя основными подходами являются:

1. Поменяйте контроллер корневого представления главного окна приложения
2. Представьте поток бортового потока как модальное путешествие, перекрывающее основной поток.

Реализация этого должна содержаться в расширении AppDelegate.

Examples

Вариант 1: обменять контроллер корневого представления (хорошо)

Есть преимущества для переключения контроллера корневого представления, хотя параметры перехода ограничены теми, которые поддерживаются `UIViewAnimationOptions` , поэтому в зависимости от того, как вы хотите переходить между потоками, может означать, что вы должны реализовать пользовательский переход, который может быть громоздким.

Вы можете показать поток Onboarding, просто установив `UIApplication.shared.keyWindow.rootViewController`

Увольнение обрабатывается путем использования `UIView.transition(with:)` и перехода стиля перехода как `UIViewAnimationOptions` , в этом случае Cross Dissolve. (Флип и завитки также поддерживаются).

Вы также должны установить кадр Main view перед тем, как перейти к нему, поскольку вы создаете его впервые.

```
// MARK: - Onboarding

extension AppDelegate {

    func showOnboarding() {
        if let window = UIApplication.shared.keyWindow, let onboardingViewController =
UIStoryboard(name: "Onboarding", bundle: nil).instantiateInitialViewController() as?
OnboardingViewController {
            onboardingViewController.delegate = self
            window.rootViewController = onboardingViewController
        }
    }

    func hideOnboarding() {
        if let window = UIApplication.shared.keyWindow, let mainViewController =
UIStoryboard(name: "Main", bundle: nil).instantiateInitialViewController() {
            mainViewController.view.frame = window.bounds
```

```
            UIView.transition(with: window, duration: 0.5, options: .transitionCrossDissolve,
animations: {
                window.rootViewController = mainViewController
            }, completion: nil)
        }
    }
}
```

Вариант 2: Существующий альтернативный поток (лучше)

В самой простой реализации поток Onboarding может быть просто представлен в модальном контексте, поскольку семантически Пользователь находится в одном путешествии.

> [Руководство Apple по интерфейсу пользователя - модальность] [1]:
>
> Рассмотрите возможность создания модального контекста только тогда, когда важно привлечь внимание кого-либо, когда задача должна быть завершена или оставлена для продолжения использования приложения или для сохранения важных данных.

Представление модально позволяет простой вариант увольнения в конце путешествия, с небольшим количеством крутизны переключающих контроллеров.

Пользовательские переходы также поддерживаются стандартным образом, поскольку для этого используется API `ViewController.present()` :

```
// MARK: - Onboarding

extension AppDelegate {

    func showOnboarding() {
        if let window = window, let onboardingViewController = UIStoryboard(name:
"Onboarding", bundle: nil).instantiateInitialViewController() as? OnboardingViewController {
            onboardingViewController.delegate = self
            window.makeKeyAndVisible()
            window.rootViewController?.present(onboardingViewController, animated: false,
completion: nil)
        }
    }

    func hideOnboarding() {
        if let window = UIApplication.shared.keyWindow {
            window.rootViewController?.dismiss(animated: true, completion: nil)
        }
    }
}
```

Прочитайте Swift: изменение rootViewController в AppDelegate для представления основного или входного / входного потока онлайн: https://riptutorial.com/ru/ios/topic/10880/swift--изменение-rootviewcontroller-в-appdelegate-для-представления-основного-или-входного---входного-потока

глава 65: SWRevealViewController

замечания

Использование класса SWRevealViewController в качестве основной навигации может не всегда приводить к лучшему пользовательскому опыту. Если боковая панель содержит только 5 или менее записей (или содержимое может быть сжато в 5 или менее записей), вы должны рассмотреть возможность использования панели вкладок по умолчанию.

Панель вкладок интуитивно понятна и позволяет пользователю быстро переключаться между представлениями / контекстами. С другой стороны, навигационная панель боковой панели может выполнять больше действий, чем переключать представление / контекст, и использует меньшее пространство при свертывании.

Для получения дополнительной информации ознакомьтесь с Руководством по интерфейсу iOS от Apple.

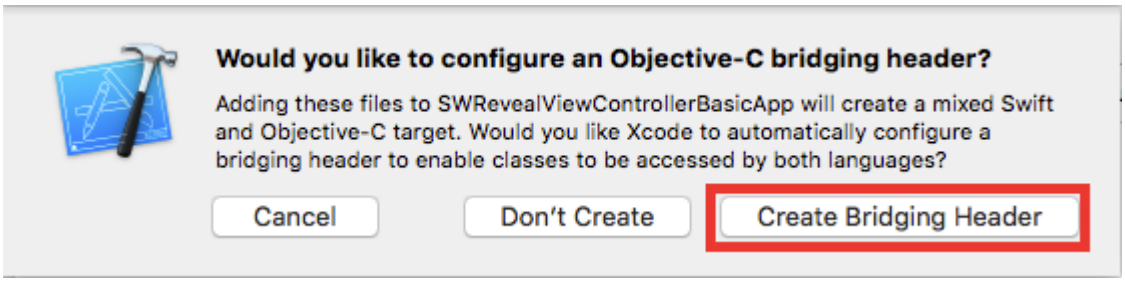
Examples

Настройка базового приложения с помощью SWRevealViewController

Создайте базовое приложение с одним шаблоном приложения для просмотра с быстрым языком

Добавьте `SWRevealViewController.h` и `SWRevealViewController.m`

затем нажмите кнопку Create Bridging Header

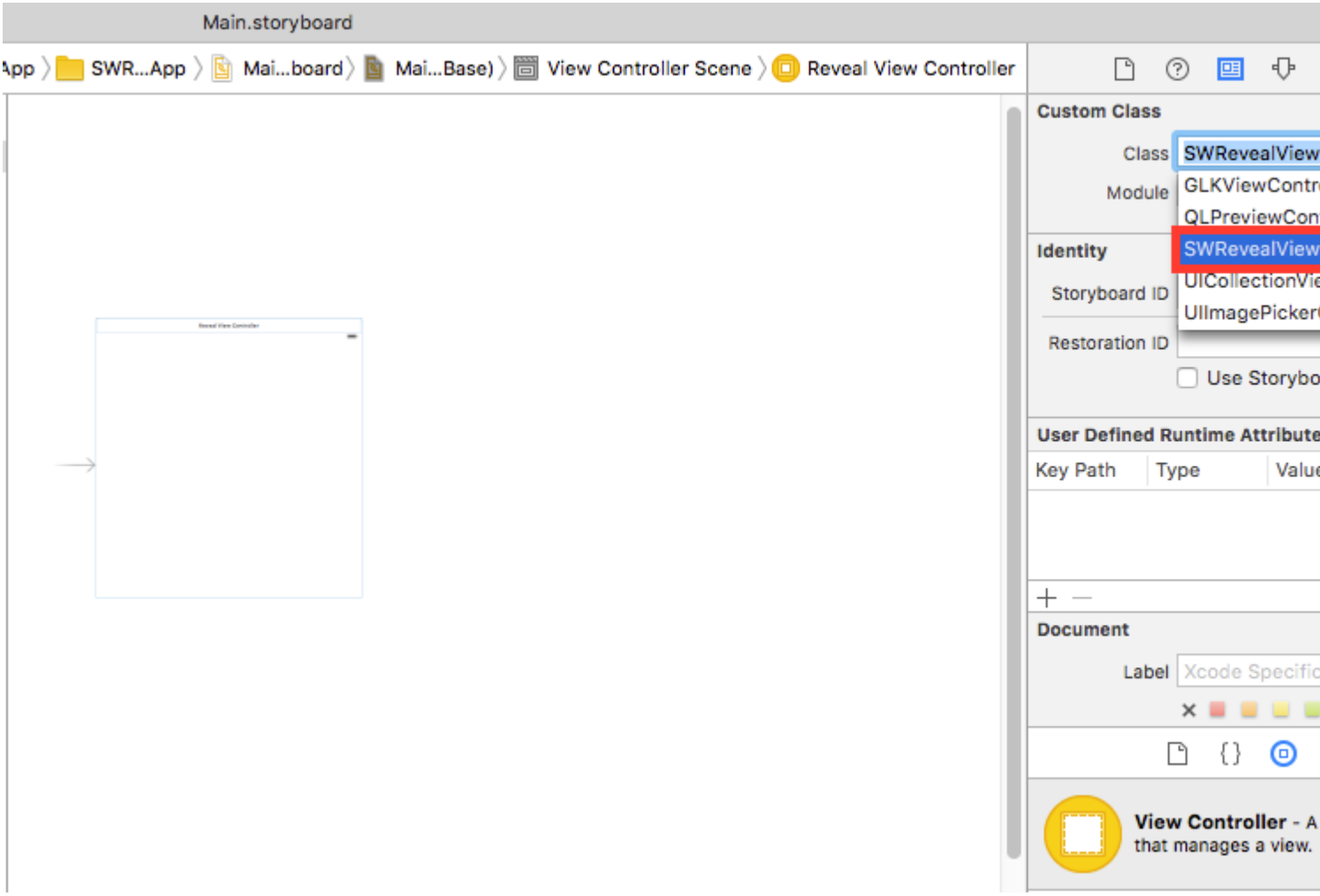


и добавить

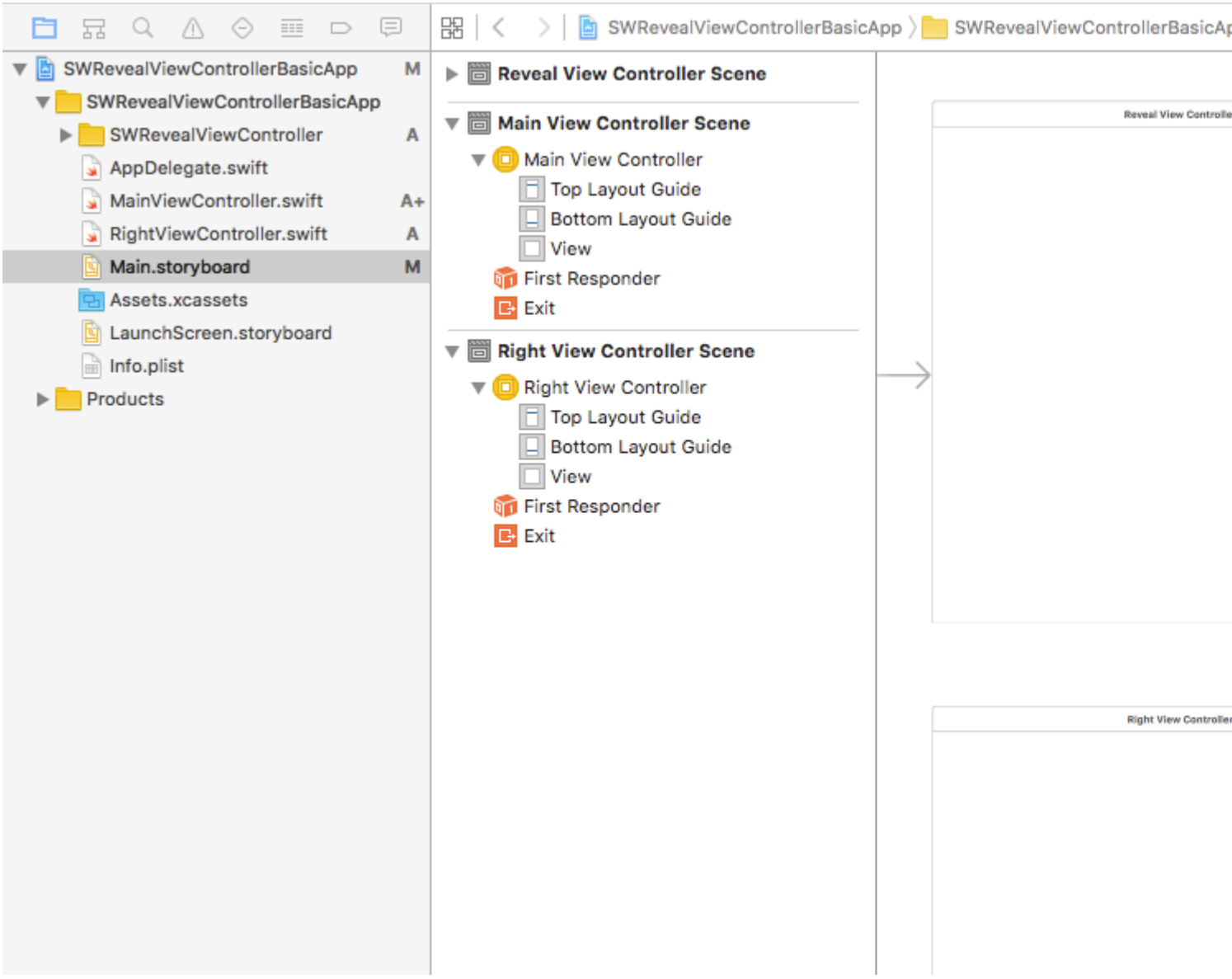
```
#import "SWRevealViewController.h"
```

в заголовке «Бридж»

Затем выберите viewController на раскадровке и измените класс на `SWRevealViewController`

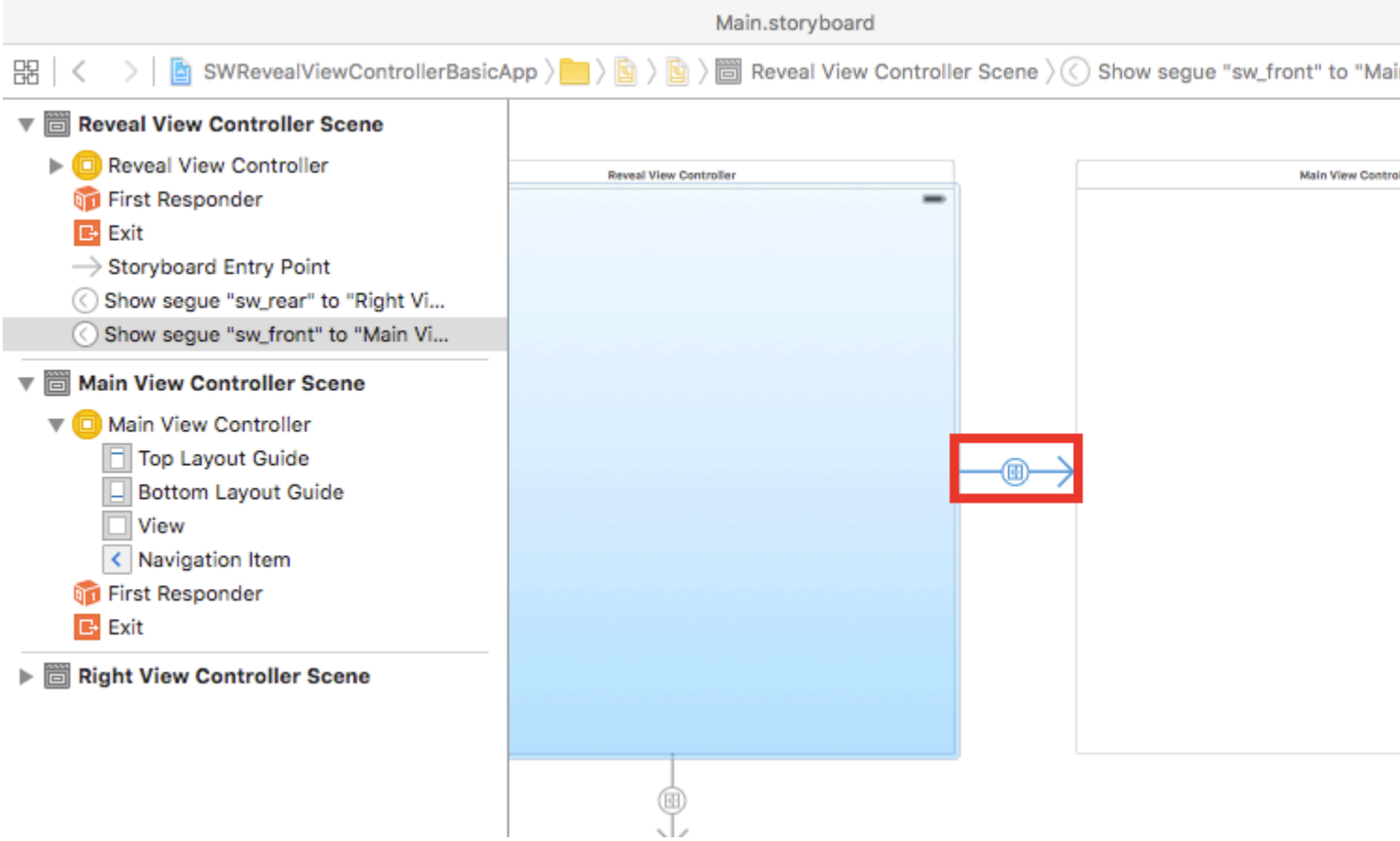


Затем переименуйте viewController в файлы в MainViewController и добавьте новый ViewController с именем RightViewController



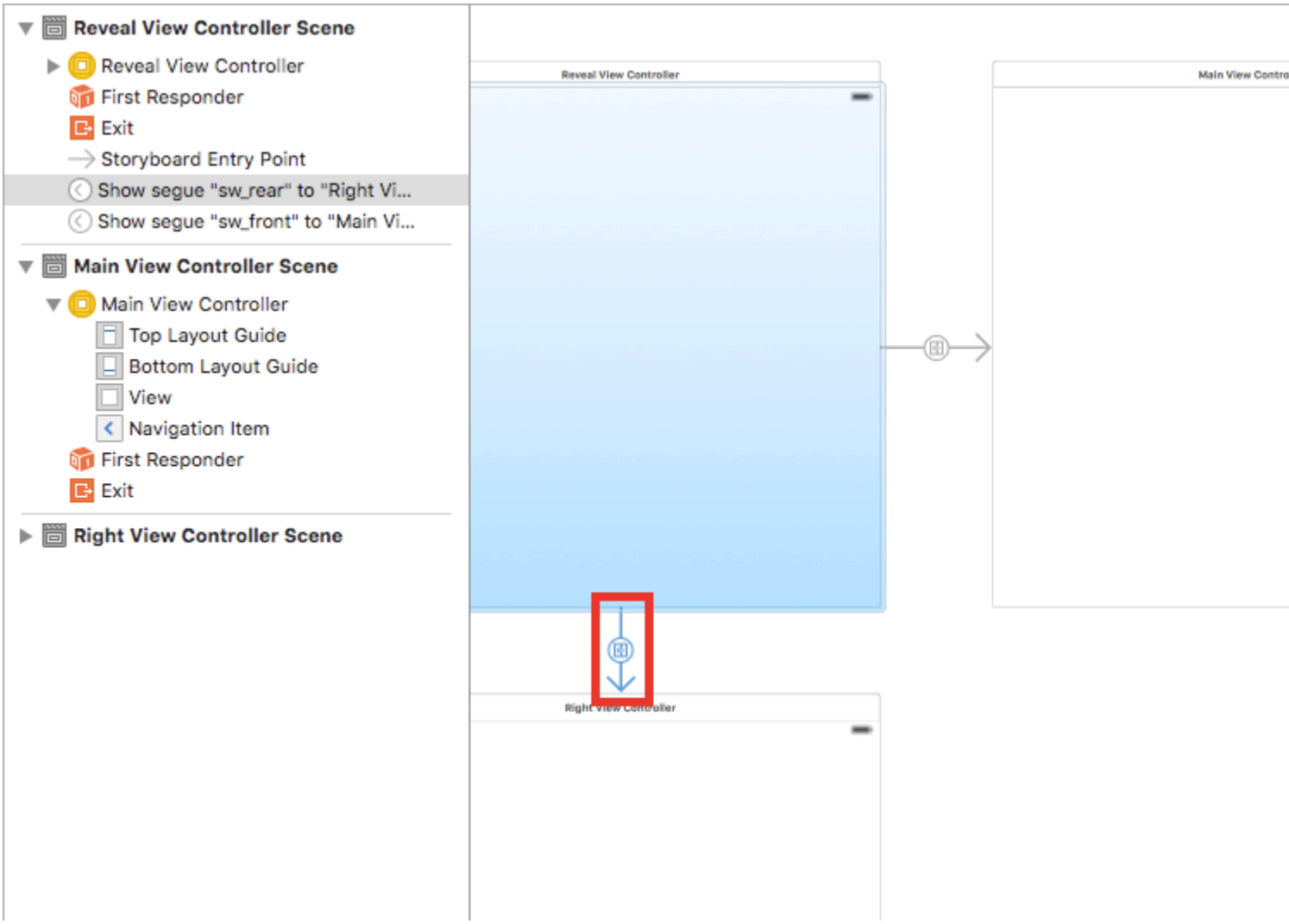
то мы добавим два segues из SWRevealViewController в MainViewController и из SWRevealViewController в RightViewController, тогда нам нужно выбрать первый (от SWRevealViewController to MainViewController) и отредактировать свойства

по набору идентификаторов `sw_front` по набору классов `SWRevealViewControllerSegueSetController`



после этого мы должны сделать то же самое с segue (от SWRevealViewController до RightViewController)

по идентификатору set `sw_rear` по набору классов `SWRevealViewControllerSegueSetController`



то на MainViewController добавьте эту строку в метод `viewDidLoad`

```
self.view.addGestureRecognizer(self.revealViewController().panGestureRecognizer());
```

И это все, у вас есть базовое приложение с интегрированным SWRevealViewController, вы можете `RightViewController` вправо, чтобы показать `RightViewController` как боковое меню

Прочитайте SWRevealViewController онлайн:
https://riptutorial.com/ru/ios/topic/4614/swrevealviewcontroller

глава 66: UIActivityViewController

параметры

Имя параметра	Описание
activityItems	Содержит массив объекта для выполнения операции. Этот массив не должен быть nil и должен содержать хотя бы один объект.
applicationActivities	Массив объектов UIActivity, представляющих пользовательские сервисы, поддерживаемые вашим приложением. Этот параметр может быть равен нулю.

Examples

Инициализация контроллера вида активности

Objective-C

```
NSString *textToShare = @"StackOverflow Documentation!! Together, we can do for Documentation
what we did for Q&A.";
NSURL *documentationURL = [NSURL
URLWithString:@"http://stackoverflow.com/tour/documentation"];

NSArray *objectsToShare = @[textToShare, documentationURL];

UIActivityViewController *activityVC = [[UIActivityViewController alloc]
initWithActivityItems:objectsToShare applicationActivities:nil];

[self presentViewController:activityVC animated:YES completion:nil];
```

стриж

```
let textToShare = "StackOverflow Documentation!! Together, we can do for Documentation what we
did for Q&A."
let documentationURL = NSURL(string:"http://stackoverflow.com/tour/documentation")

let objToShare : [AnyObject] = [textToShare, documentationURL!]

let activityVC = UIActivityViewController(activityItems: objToShare, applicationActivities:
nil)
self.presentViewController(activityVC, animated: true, completion: nil)
```

Прочитайте UIActivityViewController онлайн:

https://riptutorial.com/ru/ios/topic/2889/uiactivityviewcontroller

глава 67: UIAlertController

замечания

> Объект `UIAlertController` отображает пользователю предупреждение. Этот класс заменяет классы `UIActionSheet` и `UIAlertView` для отображения предупреждений. После настройки контроллера предупреждений с `presentViewController:animated:completion:` действиями и стилем, используйте его с помощью `presentViewController:animated:completion:`

Из документации Apple

UIAlertController в Swift

Examples

AlertViews с UIAlertController

`UIAlertView` и `UIActionSheet` устарели в `iOS 8` и более поздних версиях. Итак, Apple представила новый контроллер для `AlertView` и `ActionSheet` под названием `UIAlertController` , изменив `preferredStyle` , вы можете переключаться между `AlertView` и `ActionSheet` . Для него нет метода делегата, потому что все события кнопок обрабатываются в своих блоках.

Простой AlertView

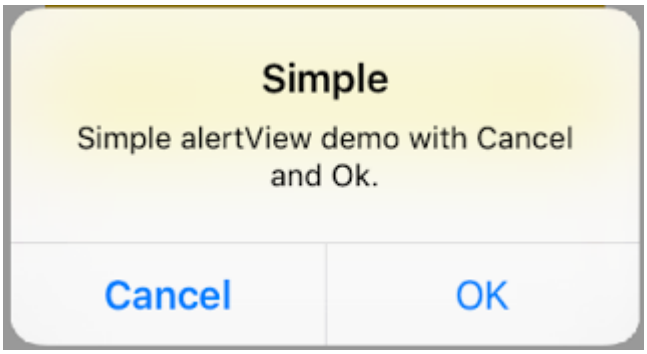
Swift:

```
let alert = UIAlertController(title: "Simple", message: "Simple alertView demo with Cancel and
OK.", preferredStyle: .alert)

alert.addAction(UIAlertAction(title: "Cancel", style: .cancel) { _ in
        print("Cancel")
})
alert.addAction(UIAlertAction(title: "OK", style: .default) { _ in
        print("OK")
})

present(alert, animated: true)
```

Objective-C:

```
 UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"Simple"
message:@"Simple alertView demo with Cancel and OK."
preferredStyle:UIAlertControllerStyleAlert];
    UIAlertAction *cancelAction = [UIAlertAction actionWithTitle:@"Cancel"
style:UIAlertActionStyleCancel handler:^(UIAlertAction * action) {
        NSLog(@"Cancel");
    }];
```

```
    UIAlertAction *okAction = [UIAlertAction actionWithTitle:@"OK"
style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
        NSLog(@"OK");
    }];

    [alertController addAction:cancelAction];
    [alertController addAction:okAction];
    [self presentViewController:alertController animated: YES completion: nil];
```



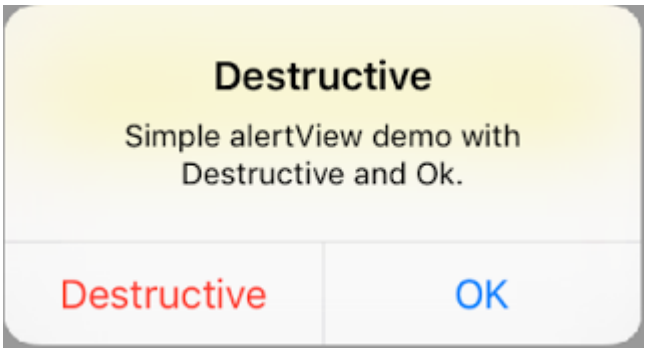
Разрушительный AlertView

Swift:

```
let alert = UIAlertController(title: "Simple", message: "Simple alertView demo with Cancel and
OK.", preferredStyle: .alert)

alert.addAction(UIAlertAction(title: "Destructive", style: .destructive) { _ in
        print("Destructive")
})
alert.addAction(UIAlertAction(title: "OK", style: .default) { _ in
        print("OK")
})

present(alert, animated: true)
```

Objective-C:

```
UIAlertController *alertController = [UIAlertController
alertControllerWithTitle:@"Destructive" message:@"Simple alertView demo with Destructive and
OK." preferredStyle:UIAlertControllerStyleAlert];
    UIAlertAction *destructiveAction = [UIAlertAction actionWithTitle:@"Cancel"
style:UIAlertActionStyleDestructive handler:^(UIAlertAction * action) {
        NSLog(@"Destructive");
    }];
    UIAlertAction *okAction = [UIAlertAction actionWithTitle:@"OK"
style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
        NSLog(@"OK");
    }];

    [alertController addAction:destructiveAction];
    [alertController addAction:okAction];
    [self presentViewController:alertController animated: YES completion: nil];
```



Временный тост-подобный всплывающий

Хорошо для быстрых уведомлений, которые не требуют взаимодействия.

стриж

```
let alert = UIAlertController(title: "Toast", message: "Hello World", preferredStyle: .Alert)

presentViewController(alert, animated: true) {
     let delay_s:Double = 2
     let delayTime = dispatch_time(DISPATCH_TIME_NOW, Int64(delay_s * Double(NSEC_PER_SEC)))
     dispatch_after(delayTime, dispatch_get_main_queue()) {
          alert.dismissViewControllerAnimated(true, completion: nil)
     }
}
```

Добавление текстового поля в UIAlertController, как окно подсказки

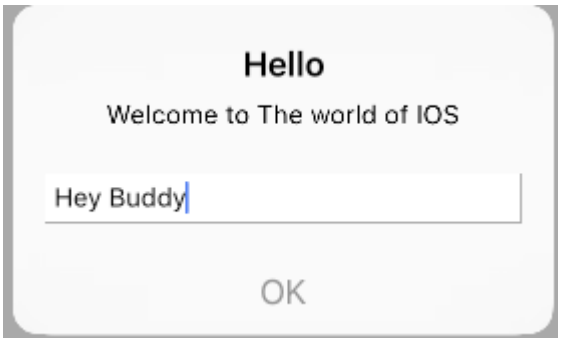
стриж

```
let alert = UIAlertController(title: "Hello",
                              message: "Welcome to the world of iOS",
                              preferredStyle: UIAlertControllerStyle.alert)
let defaultAction = UIAlertAction(title: "OK", style: UIAlertActionStyle.default) { (action)
in

}
defaultAction.isEnabled = false
alert.addAction(defaultAction)

alert.addTextFieldWithConfigurationHandler { (textField) in
     textField.delegate = self
}

present(alert, animated: true, completion: nil)
```

Objective-C

```
UIAlertController* alert = [UIAlertController alertControllerWithTitle:@"Hello"
                                                                message:@"Welcome to the world
```

```
of iOS"

preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction* defaultAction = [UIAlertAction actionWithTitle:@"OK"
                                                        style:UIAlertActionStyleDefault
                                                      handler:^(UIAlertAction * action) {}];

defaultAction.enabled = NO;
[alert addAction:defaultAction];

[alert addTextFieldWithConfigurationHandler:^(UITextField *textField) {
     textField.delegate = self;
}];

[self presentViewController:alert animated:YES completion:nil];
```



Таблицы действий с UIAlertController

С помощью `UIAlertController` действий, такие как устаревший `UIActionSheet` , создаются с использованием того же API, что и для AlertView.

Простой лист действий с двумя кнопками

стриж

```
let alertController = UIAlertController(title: "Demo", message: "A demo with two buttons",
preferredStyle: UIAlertControllerStyle.actionSheet)
```

Objective-C

```
UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"Demo"
message:@"A demo with two buttons" preferredStyle:UIAlertControllerStyleActionSheet];
```

Создайте кнопки «Отмена» и «Хорошо»

стриж

```
let cancelAction = UIAlertAction(title: "Cancel", style: .cancel) { (result : UIAlertAction) -
> Void in
    //action when pressed button
}
let okAction = UIAlertAction(title: "Okay", style: .default) { (result : UIAlertAction) ->
Void in
    //action when pressed button
}
```

Objective-C

```
UIAlertAction *cancelAction = [UIAlertAction actionWithTitle:@"Cancel"
style:UIAlertActionStyleCancel handler:^(UIAlertAction * action) {
        //action when pressed button
    }];

UIAlertAction * okAction = [UIAlertAction actionWithTitle:@"Okay"
style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
        //action when pressed button
    }];
```

И добавьте их в лист действий:

стриж

```
alertController.addAction(cancelAction)
alertController.addAction(okAction)
```

Objective-C

```
[alertController addAction:cancelAction];
[alertController addAction:okAction];
```

Теперь представим `UIAlertController` :

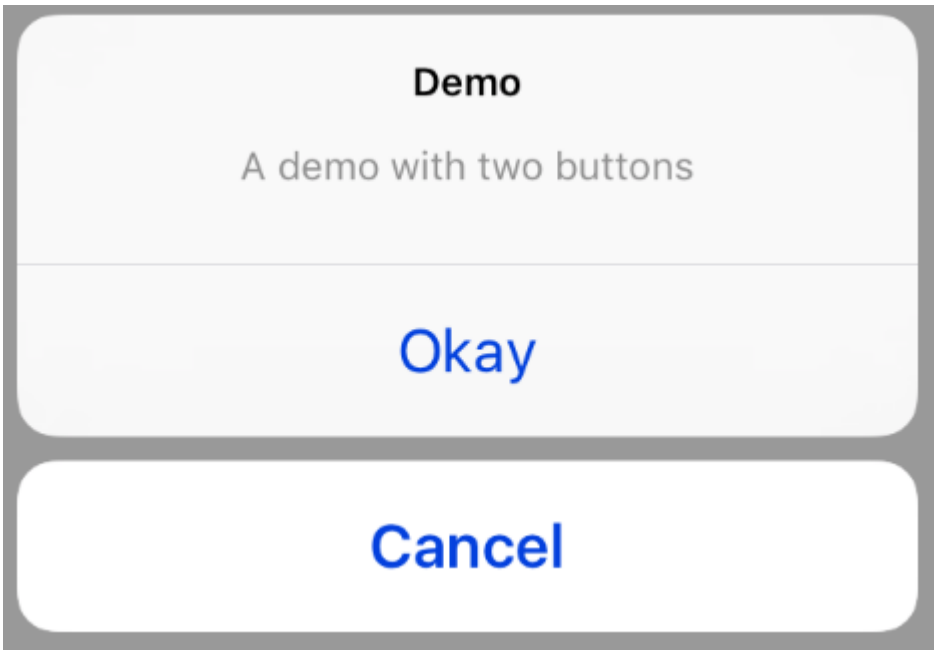
стриж

```
self.present(alertController, animated: true, completion: nil)
```

Objective-C

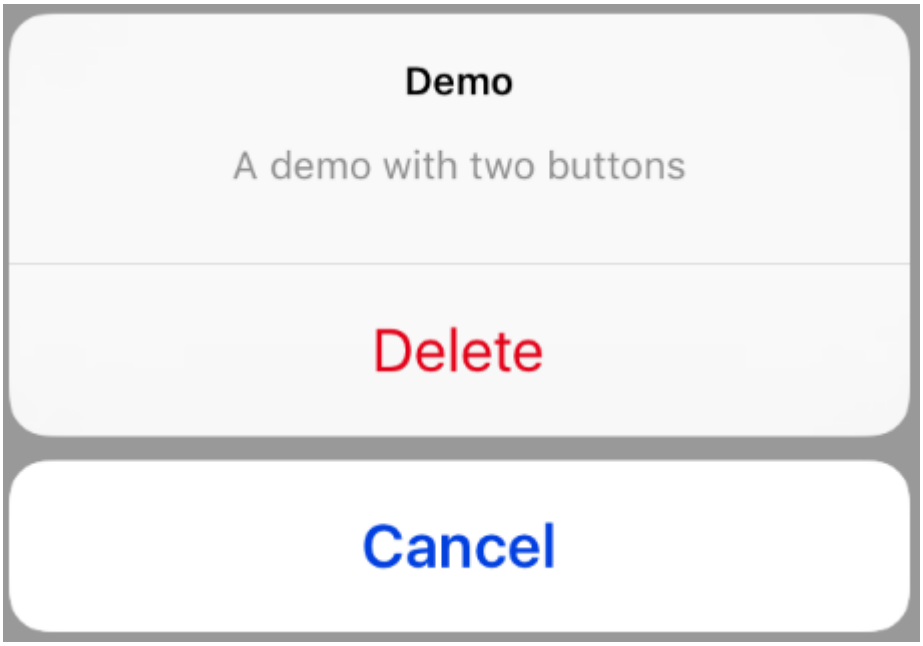
```
[self presentViewController:alertController animated: YES completion: nil];
```

Это должно быть результатом:



Лист действия с деструктивной кнопкой

Использование `UIAlertActionStyle .destructive` для `UIAlertAction` создаст кнопку с красным цветом оттенка.



Для этого примера `okAction` сверху был заменен этим `UIAlertAction` :

стриж

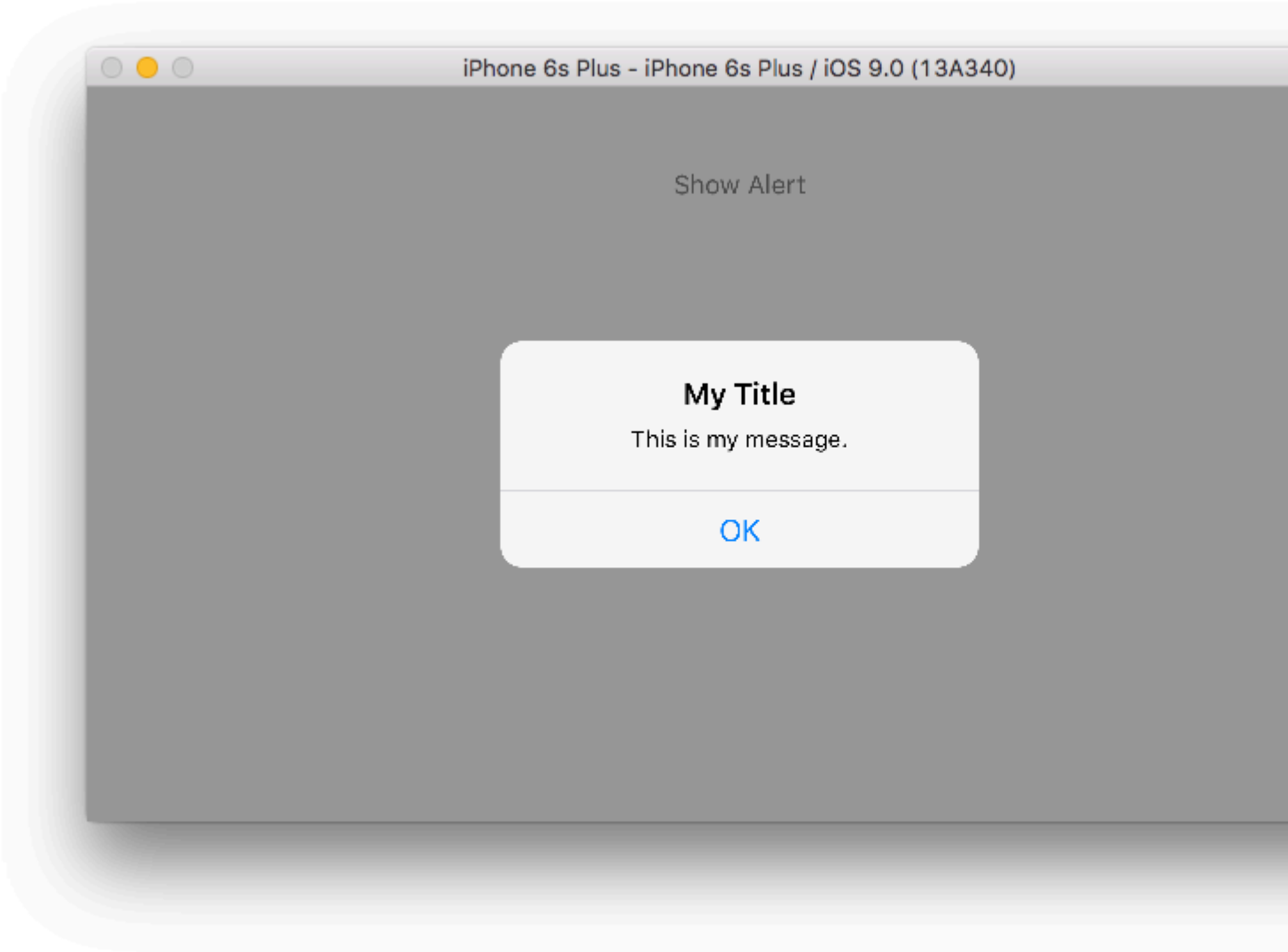
```
let destructiveAction = UIAlertAction(title: "Delete", style: .destructive) { (result :
UIAlertAction) -> Void in
    //action when pressed button
}
```

Objective-C

```
UIAlertAction * destructiveAction = [UIAlertAction actionWithTitle:@"Delete"
style:UIAlertActionStyleDestructive handler:^(UIAlertAction * action) {
            //action when pressed button
        }];
```

Отображение и обработка предупреждений

Одна кнопка

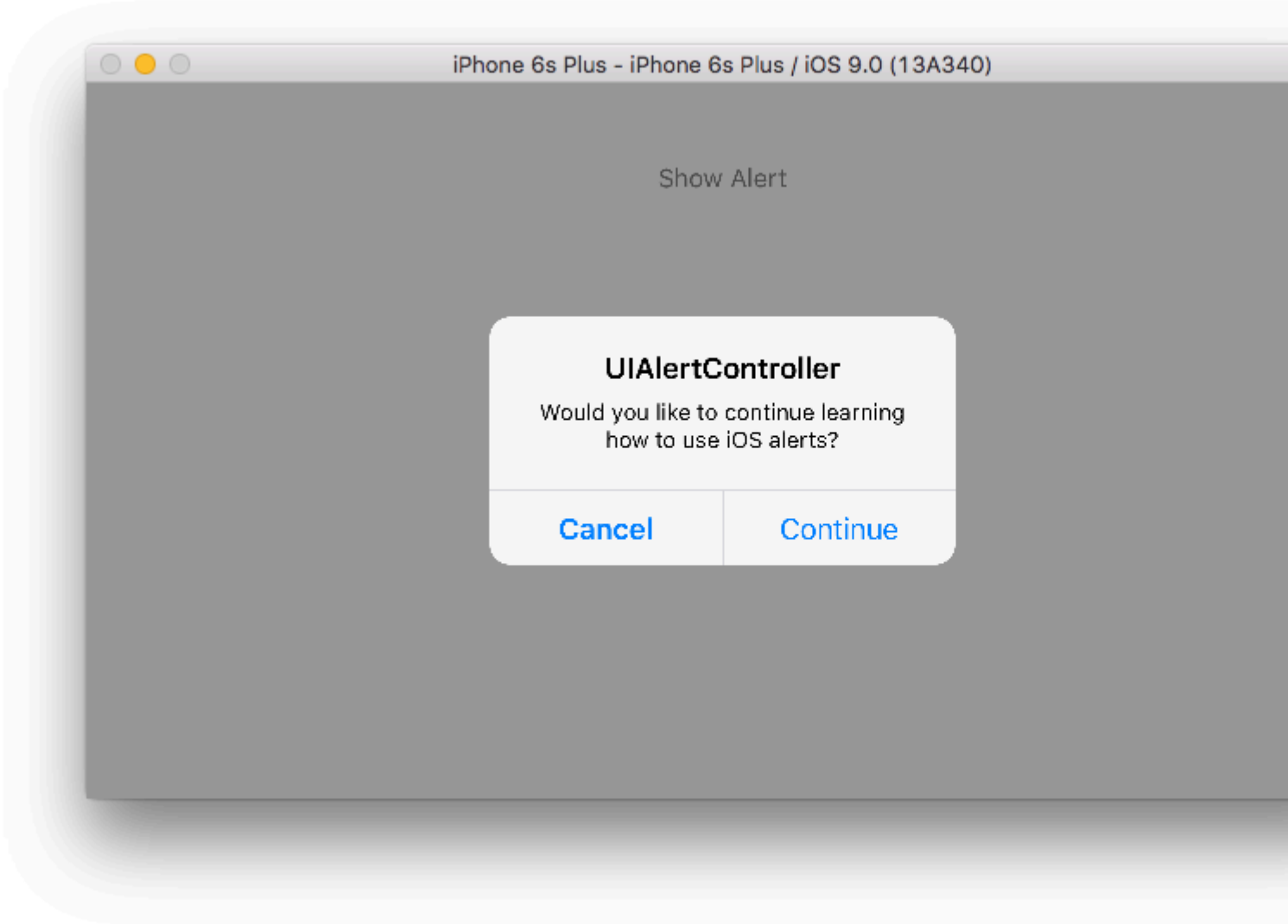


стриж

```
class ViewController: UIViewController {

    @IBAction func showAlertButtonTapped(sender: UIButton) {
```

```
        // create the alert
        let alert = UIAlertController(title: "My Title", message: "This is my message.",
preferredStyle: UIAlertControllerStyle.Alert)

        // add an action (button)
        alert.addAction(UIAlertAction(title: "OK", style: UIAlertActionStyle.Default, handler:
nil))

        // show the alert
        self.presentViewController(alert, animated: true, completion: nil)
    }
}
```

Две кнопки

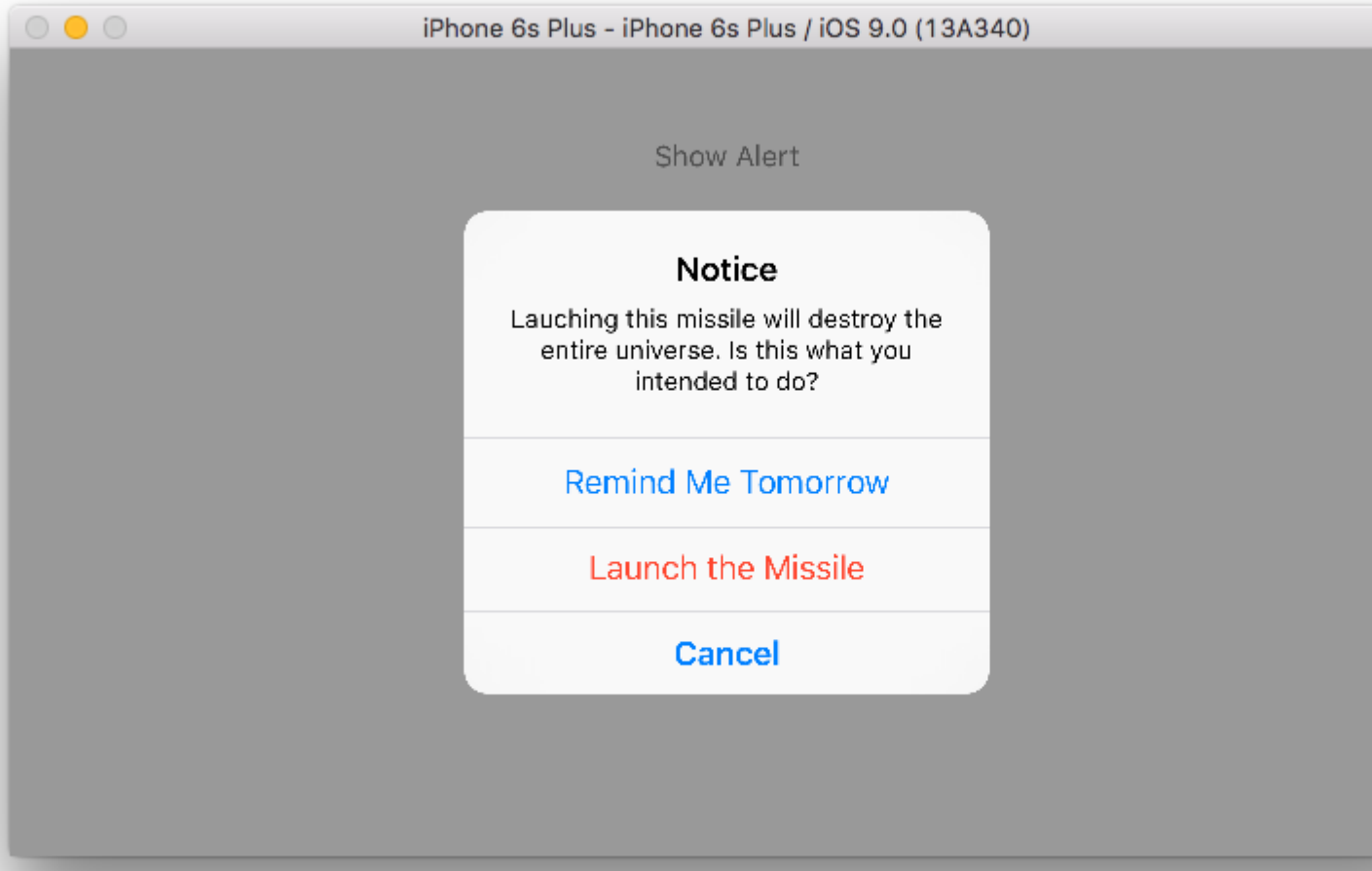


стриж

```
class ViewController: UIViewController {

    @IBAction func showAlertButtonTapped(sender: UIButton) {
```

```
        // create the alert
        let alert = UIAlertController(title: "UIAlertController", message: "Would you like to
continue learning how to use iOS alerts?", preferredStyle: UIAlertControllerStyle.Alert)

        // add the actions (buttons)
        alert.addAction(UIAlertAction(title: "Continue", style: UIAlertActionStyle.Default,
handler: nil))
        alert.addAction(UIAlertAction(title: "Cancel", style: UIAlertActionStyle.Cancel,
handler: nil))

        // show the alert
        self.presentViewController(alert, animated: true, completion: nil)
    }
}
```

Три кнопки



стриж

```
class ViewController: UIViewController {
```

```
    @IBAction func showAlertButtonTapped(sender: UIButton) {

        // create the alert
        let alert = UIAlertController(title: "Notice", message: "Lauching this missile will
destroy the entire universe. Is this what you intended to do?", preferredStyle:
UIAlertControllerStyle.Alert)

        // add the actions (buttons)
        alert.addAction(UIAlertAction(title: "Remind Me Tomorrow", style:
UIAlertActionStyle.Default, handler: nil))
        alert.addAction(UIAlertAction(title: "Cancel", style: UIAlertActionStyle.Cancel,
handler: nil))
        alert.addAction(UIAlertAction(title: "Launch the Missile", style:
UIAlertActionStyle.Destructive, handler: nil))

        // show the alert
        self.presentViewController(alert, animated: true, completion: nil)
    }
}
```

Обрабатывающие кнопки

В приведенных выше примерах `handler` был `nil` . Вы можете заменить `nil` на закрытие, чтобы что-то сделать, когда пользователь нажал кнопку, как показано ниже:

стриж

```
alert.addAction(UIAlertAction(title: "Launch the Missile", style:
UIAlertActionStyle.Destructive, handler: { action in

    // do something like...
    self.launchMissile()

}))
```

Заметки

- Многим кнопкам не обязательно нужно использовать разные типы `UIAlertActionStyle` . Все они могут быть `.Default` .
- Для более чем трех кнопок рассмотрите использование листа действий. Настройка очень похожа. Вот пример.

Выделение кнопки действия

Контроллер предупреждения имеет свойство, которое используется для добавления акцентов в действие, добавленное в контроллер предупреждения. Это свойство можно использовать, чтобы выделить конкретное действие для внимания пользователя. Для цели

C;

```
@property(nonatomic, strong) UIAlertAction *preferredAction
```

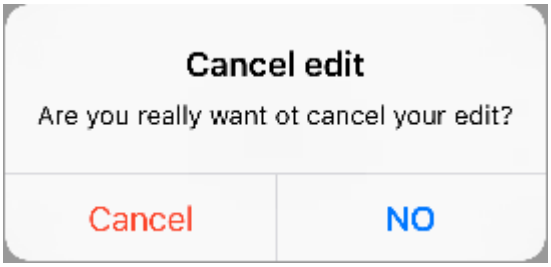
Действие, **которое уже добавлено в контроллер предупреждений,** может быть присвоено этому свойству. Контроллер предупреждений выделит это действие.

Это свойство можно использовать только с UIAlertControllerStyleAlert.

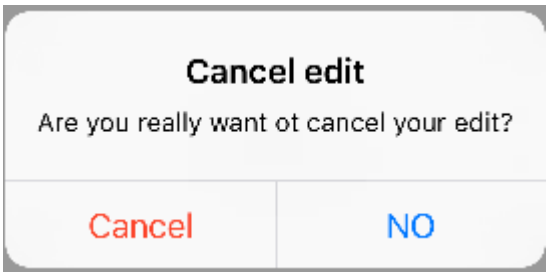
В следующем примере показано, как его использовать.

```
UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"Cancel
edit" message:@"Are you really want to cancel your edit?"
preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction *cancel = [UIAlertAction actionWithTitle:@"Cancel"
style:UIAlertActionStyleCancel handler:^(UIAlertAction * action) {
    NSLog(@"Cancel");
}];

UIAlertAction *no = [UIAlertAction actionWithTitle:@"NO" style:UIAlertActionStyleDefault
handler:^(UIAlertAction * action) {
    NSLog(@"Highlighted button is pressed.");
}];

[alertController addAction:cancel];
[alertController addAction:no];

//add no action to preffered action.
//Note
//the action should already be added to alert controller
alertController.preferredAction = no;

[self presentViewController:alertController animated: YES completion: nil];
```

Alert Controller с **предпочтительным набором действий** . Кнопка **NO** выделена.



Контроллер предупреждения с **предпочтительным действием не установлен** . Кнопка **NO** не подсвечена.



Прочитайте UIAlertController онлайн: https://riptutorial.com/ru/ios/topic/874/uialertcontroller

глава 68: UIAppearance

Examples

Установить внешний вид всех экземпляров класса

Чтобы настроить внешний вид всех экземпляров класса, получите доступ к прокси-объекту нужного класса. Например:

Установите цвет оттенка UIButton

Swift:

```
UIButton.appearance().tintColor = UIColor.greenColor()
```

Objective-C:

```
[UIButton appearance].tintColor = [UIColor greenColor];
```

Установить цвет фона UIButton

Swift:

```
UIButton.appearance().backgroundColor = UIColor.blueColor()
```

Objective-C:

```
[UIButton appearance].backgroundColor = [UIColor blueColor];
```

Установите цвет текста UILabel

Swift:

```
UILabel.appearance().textColor = UIColor.redColor()
```

Objective-C:

```
[UILabel appearance].textColor = [UIColor redColor];
```

Установите цвет фона UILabel

Swift:

```
UILabel.appearance().backgroundColor = UIColor.greenColor()
```

Objective-C:

```
[UILabel appearance].backgroundColor = [UIColor greenColor];
```

Установить цвет оттенка UINavigationBar

Swift:

```
UINavigationBar.appearance().tintColor = UIColor.cyanColor()
```

Objective-C:

```
[UINavigationBar appearance].tintColor = [UIColor cyanColor];
```

Установите цвет фона UINavigationBar

Swift:

```
UINavigationBar.appearance().backgroundColor = UIColor.redColor()
```

Objective-C:

```
[UINavigationBar appearance].backgroundColor = [UIColor redColor];
```

Внешний вид класса, когда он содержится в классе контейнера

Используйте `appearanceWhenContainedInInstancesOfClasses:` чтобы настроить внешний вид, например, класса, содержащегося в экземпляре класса контейнера. Например , настройки `UILabel` «s `textColor` и `backgroundColor` в `ViewController` класса будет выглядеть следующим образом :

Установите цвет текста UILabel

Swift:

```
UILabel.appearanceWhenContainedInInstancesOfClasses([ViewController.self]).textColor =
UIColor.whiteColor()
```

Objective-C:

```
[UILabel appearanceWhenContainedInInstancesOfClasses:@[[ViewController class]]].textColor =
[UIColor whiteColor];
```

Установите цвет фона UILabel

Swift:

```
UILabel.appearanceWhenContainedInInstancesOfClasses([ViewController.self]).backgroundColor =
UIColor.blueColor()
```

Objective-C:

```
[UILabel appearanceWhenContainedInInstancesOfClasses:@[[ViewController
class]]].backgroundColor = [UIColor blueColor];
```

Прочитайте UIAppearance онлайн: https://riptutorial.com/ru/ios/topic/3422/uiappearance

глава 69: UIBarButtonItem

параметры

параметр	Описание
заглавие	Название UIBarButtonItem
стиль	Стиль UIBarButtonItem
цель	Объект для получения действия UIBarButtonItem
действие	Селектор (метод), который должен выполняться при нажатии кнопки UIBarButtonItem

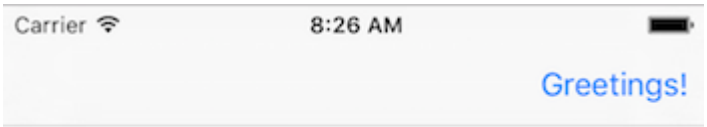
замечания

Ссылка на `self.navigationItem` предполагает, что UIViewController встроен в UINavigationController.

Examples

Создание UIBarButtonItem

```
//Swift
let barButtonItem = UIBarButtonItem(title: "Greetings!", style: .Plain, target: self, action:
#selector(barButtonTapped))
self.navigationItem.rightBarButtonItem = barButtonItem

//Objective-C
UIBarButtonItem *barButtonItem = [[UIBarButtonItem alloc] initWithTitle:@"Greetings!"
style:UIBarButtonItemStylePlain target:self action:@selector(barButtonTaped)];
self.navigationItem.rightBarButtonItem = barButtonItem;
```

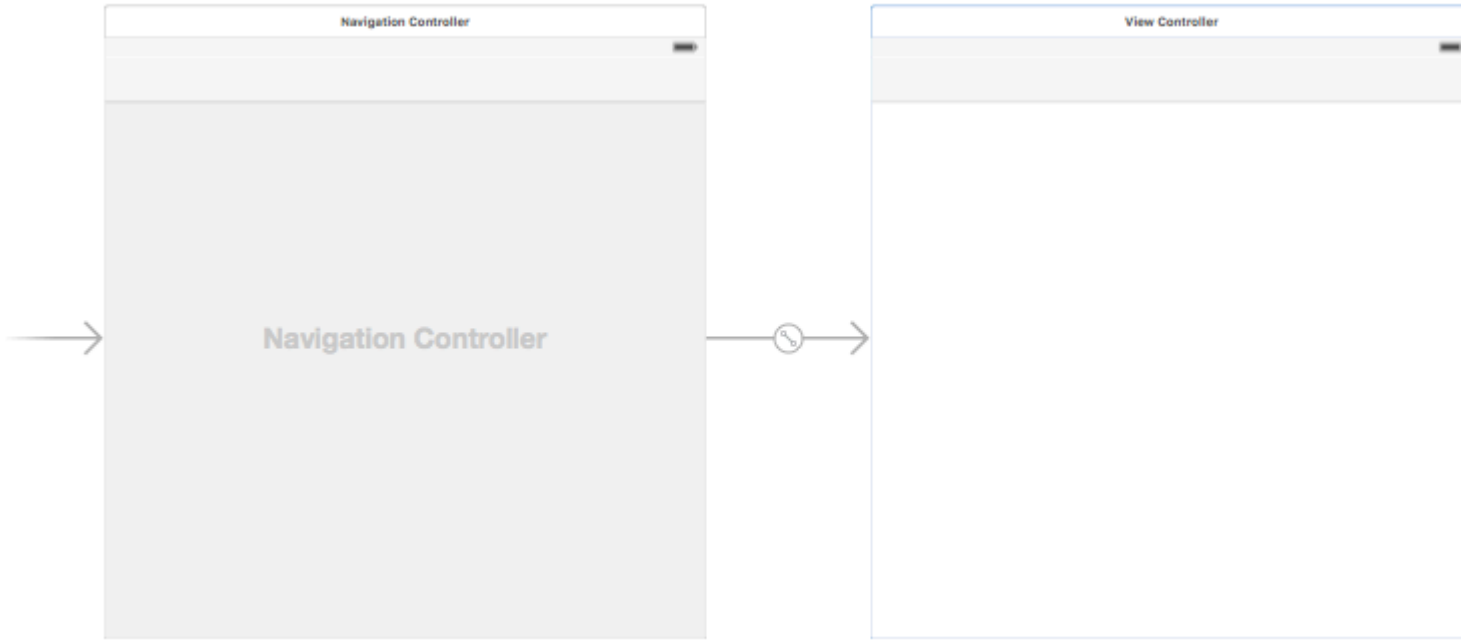


Создание UIBarButtonItem в построителе интерфейса

В приведенном ниже примере показано, как добавить кнопку панели навигации (называемую `UIBarButtonItem`) в Interface Builder.

Добавьте контроллер навигации на свою раскадровку

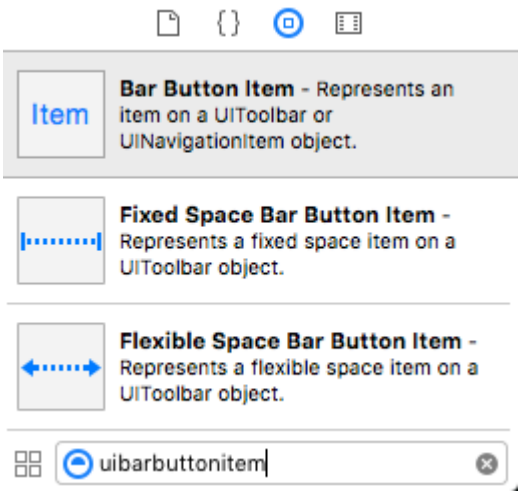
Выберите свой контроллер просмотра, а затем в меню Xcode выберите « **Редактор»> «Вставить»> «Контроллер навигации»** .



Кроме того, вы можете добавить `UINavigationBar` из библиотеки объектов.

Добавить элемент бара

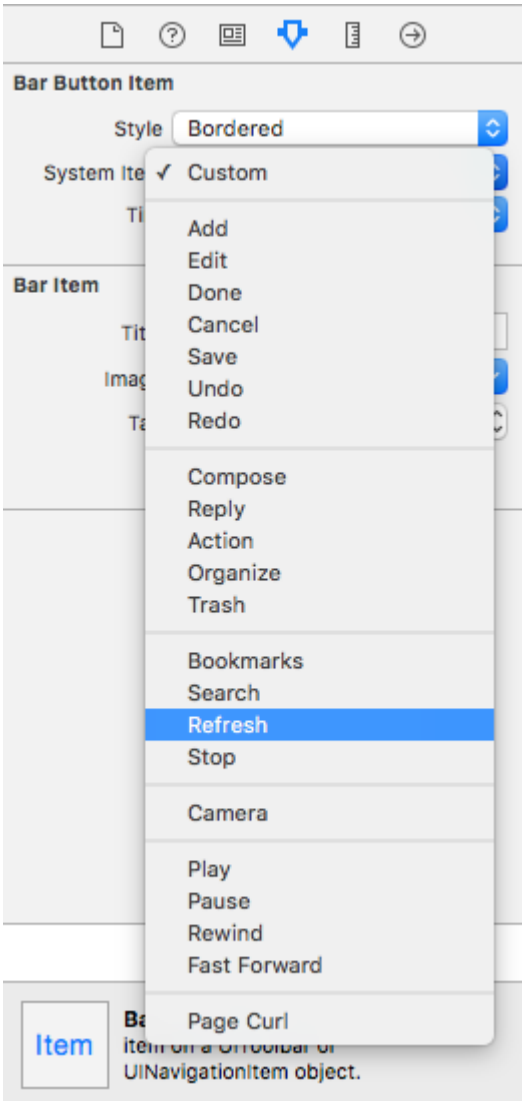
Перетащите элемент `UIBarButtonItem` из библиотеки объектов в верхнюю панель навигации.



Он должен выглядеть так:

Установить атрибуты

Вы можете дважды щелкнуть «Item», чтобы изменить текст на «Refresh», но есть реальный значок *Refresh,* который вы можете использовать. Просто выберите Инспектор атрибутов для `UIBarButtonItem` и для **элемента системы** выберите **Refresh** .



Это даст вам значок Refresh по умолчанию.

Добавить действие IB

Управляйте перетаскиванием из `UIBarButtonItem` в контроллер просмотра, чтобы добавить `@IBAction` .

```
class ViewController: UIViewController {

    @IBAction func refreshBarButtonItemTap(sender: UIBarButtonItem) {

        print("How refreshing!")
    }

}
```

Вот и все.

Заметки

- Этот пример изначально исходит из этого ответа переполнения стека .

Элемент кнопки «Бар» Исходное изображение без цвета оттенка

Если `barButtonItem` имеет свойство ненулевого изображения (например, задано в `barButtonItem` интерфейса).

Objective-C

```
   barButtonItem.image = [barButtonItem.image
imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
```

Прочитайте UIBarButtonItem онлайн: https://riptutorial.com/ru/ios/topic/1543/uibarbuttonitem

глава 70: UIBezierPath

Examples

Как применить угловой радиус к прямоугольникам, нарисованным UIBezierPath

Угловой радиус для всех 4 кромок:

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) cornerRadius: 11];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Угловой радиус для верхнего левого края:

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopLeft cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Угловой радиус для верхнего правого края:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopRight cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

угловой радиус для нижнего левого края:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomLeft cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

угловой радиус для нижнего правого края:

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomRight cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

угловой радиус для нижних краев:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomLeft |
```

```
UIRectCornerBottomRight cornerRadii: CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

угловой радиус для верхних краев:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopLeft | UIRectCornerTopRight
cornerRadii: CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Как создать простые формы с помощью UIBezierPath

Для простого круга:

```
UIBezierPath* ovalPath = [UIBezierPath bezierPathWithOvalInRect: CGRectMake(0,0,50,50)];
[UIColor.grayColor setFill];
[ovalPath fill];
```

Swift:

```
let ovalPath = UIBezierPath(ovalInRect: CGRect(x: 0, y: 0, width: 50, height: 50))
UIColor.grayColor().setFill()
ovalPath.fill()
```

Для простого прямоугольника:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect: CGRectMake(0,0,50,50)];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Swift:

```
let rectanglePath = UIBezierPath(rect: CGRect(x: 0, y: 0, width: 50, height: 50))
UIColor.grayColor().setFill()
rectanglePath.fill()
```

Для простой линии:


```
UIBezierPath* bezierPath = [UIBezierPath bezierPath];
[bezierPath moveToPoint: CGPointMake(x1,y1)];
[bezierPath addLineToPoint: CGPointMake(x2,y2)];
[UIColor.blackColor setStroke];
bezierPath.lineWidth = 1;
[bezierPath stroke];
```

Swift:

```
let bezierPath = UIBezierPath()
bezierPath.moveToPoint(CGPoint(x: x1, y: y1))
bezierPath.addLineToPoint(CGPoint(x: x2, y: y2))
UIColor.blackColor().setStroke()
bezierPath.lineWidth = 1
bezierPath.stroke()
```

Для половины круга:

```
 CGRect ovalRect = CGRectMake(x,y,width,height);
UIBezierPath* ovalPath = [UIBezierPath bezierPath];
[ovalPath addArcWithCenter: CGPointMake(0, 0) radius: CGRectGetWidth(ovalRect) / 2 startAngle:
180 * M_PI/180 endAngle: 0 * M_PI/180 clockwise: YES];
[ovalPath addLineToPoint: CGPointMake(0, 0)];
[ovalPath closePath];
```

```
CGAffineTransform ovalTransform = CGAffineTransformMakeTranslation(CGRectGetMidX(ovalRect),
CGRectGetMidY(ovalRect));
ovalTransform = CGAffineTransformScale(ovalTransform, 1, CGRectGetHeight(ovalRect) /
CGRectGetWidth(ovalRect));
[ovalPath applyTransform: ovalTransform];

[UIColor.grayColor setFill];
[ovalPath fill];
```

Swift:

```
let ovalRect = CGRect(x: 0, y: 0, width: 50, height: 50)
let ovalPath = UIBezierPath()
ovalPath.addArcWithCenter(CGPoint.zero, radius: ovalRect.width / 2, startAngle: 180 *
CGFloat(M_PI)/180, endAngle: 0 * CGFloat(M_PI)/180, clockwise: true)
ovalPath.addLineToPoint(CGPoint.zero)
ovalPath.closePath()

var ovalTransform = CGAffineTransformMakeTranslation(CGRectGetMidX(ovalRect),
CGRectGetMidY(ovalRect))
ovalTransform = CGAffineTransformScale(ovalTransform, 1, ovalRect.height / ovalRect.width)
ovalPath.applyTransform(ovalTransform)

UIColor.grayColor().setFill()
ovalPath.fill()
```

Для простого треугольника:

```
UIBezierPath* polygonPath = [UIBezierPath bezierPath];
[polygonPath moveToPoint: CGPointMake(x1, y1)];
[polygonPath addLineToPoint: CGPointMake(x2, y2)];
[polygonPath addLineToPoint: CGPointMake(x3, y2)];
[polygonPath closePath];
[UIColor.grayColor setFill];
[polygonPath fill];
```

Swift:

```
let polygonPath = UIBezierPath()
polygonPath.moveToPoint(CGPoint(x: x1, y: y1))
polygonPath.addLineToPoint(CGPoint(x: x2, y: y2))
polygonPath.addLineToPoint(CGPoint(x: x3, y: y3))
polygonPath.closePath()
UIColor.grayColor().setFill()
polygonPath.fill()
```

UIBezierPath + AutoLayout

Для пути безье для изменения размера в зависимости от рамки просмотра переопределите drawRect представления, что вы рисуете путь безье:

```
- (void)drawRect:(CGRect)frame
{
    UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect:
CGRectMake(CGRectGetMinX(frame), CGRectGetMinY(frame), CGRectGetWidth(frame),
CGRectGetHeight(frame))];
    [UIColor.grayColor setFill];
    [rectanglePath fill];
}
```

Как применить тени к UIBezierPath

Рассмотрим простой прямоугольник, который нарисован траекторией безье.

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect:
CGRectMake(x,y,width,height)];
 [UIColor.grayColor setFill];
 [rectanglePath fill];
```

Основная тень внешнего заполнения:

```
CGContextRef context = UIGraphicsGetCurrentContext();

NSShadow* shadow = [[NSShadow alloc] init];
[shadow setShadowColor: UIColor.blackColor];
[shadow setShadowOffset: CGSizeMake(7.1, 5.1)];
[shadow setShadowBlurRadius: 5];

UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect: CGRectMake(x,y,width,height)];
CGContextSaveGState(context);
CGContextSetShadowWithColor(context, shadow.shadowOffset, shadow.shadowBlurRadius,
[shadow.shadowColor CGColor]);
[UIColor.grayColor setFill];
```

```
[rectanglePath fill];
CGContextRestoreGState(context);
```

Основная внутренняя заливка:

```
CGContextRef context = UIGraphicsGetCurrentContext();

NSShadow* shadow = [[NSShadow alloc] init];
[shadow setShadowColor: UIColor.blackColor];
[shadow setShadowOffset: CGSizeMake(9.1, -7.1)];
[shadow setShadowBlurRadius: 6];

UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect: CGRectMake(x,y,width,height)];
[UIColor.grayColor setFill];
[rectanglePath fill];

CGContextSaveGState(context);
UIRectClip(rectanglePath.bounds);
CGContextSetShadowWithColor(context, CGSizeZero, 0, NULL);

CGContextSetAlpha(context, CGColorGetAlpha([shadow.shadowColor CGColor]));
CGContextBeginTransparencyLayer(context, NULL);
{
    UIColor* opaqueShadow = [shadow.shadowColor colorWithAlphaComponent: 1];
    CGContextSetShadowWithColor(context, shadow.shadowOffset, shadow.shadowBlurRadius,
[opaqueShadow CGColor]);
    CGContextSetBlendMode(context, kCGBlendModeSourceOut);
    CGContextBeginTransparencyLayer(context, NULL);

    [opaqueShadow setFill];
    [rectanglePath fill];

    CGContextEndTransparencyLayer(context);
}
CGContextEndTransparencyLayer(context);
CGContextRestoreGState(context);
```

Проектирование и нанесение пути Безье

В этом примере показан процесс разработки формы, которую вы хотите нарисовать на виде. Используется конкретная форма, но используемые вами концепции могут применяться к любой форме.

Как нарисовать маршрут Безье в

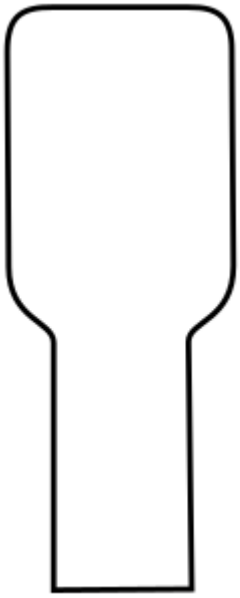
пользовательском представлении

Это основные шаги:

1. Конструируйте контур формы, которую вы хотите.
2. Разделите контур маршрута на сегменты линий, дуг и кривых.
3. Постройте этот путь программно.
4. Нарисуйте путь либо в `drawRect` либо используя `CAShapeLayer` .

Дизайн формы

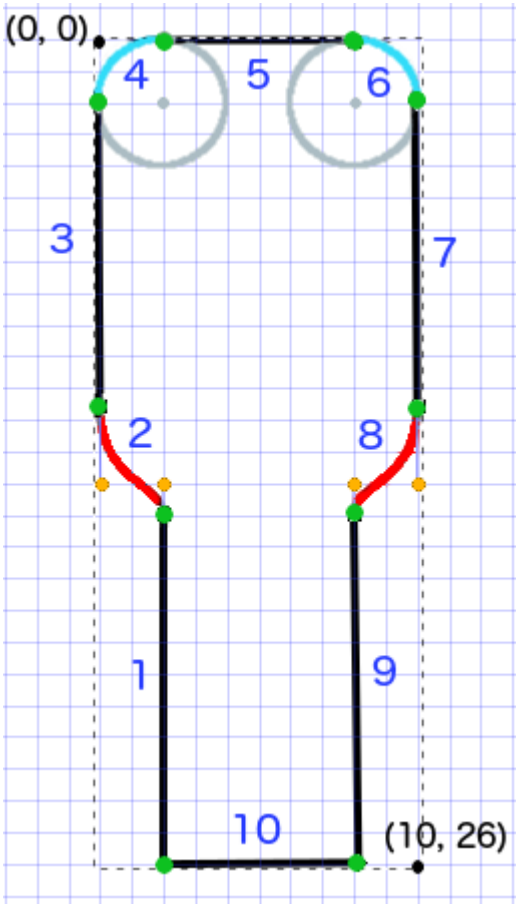
Вы могли бы сделать что угодно, но в качестве примера я выбрал форму ниже. Это может быть всплывающий ключ на клавиатуре.



Разделить путь на сегменты

Оглянитесь на свой дизайн формы и разделите его на более простые элементы линий (для прямых линий), дуги (для кругов и круглых углов) и кривые (для чего-либо еще).

Вот как выглядит наш пример:



- Черные - сегменты линии
- Светло-голубой - сегменты дуги
- Красный - кривые
- Оранжевые точки - это контрольные точки для кривых
- Зеленые точки - это точки между сегментами пути
- Пунктирные линии показывают ограничивающий прямоугольник
- Синие числа - это сегменты в том порядке, в котором они будут добавлены программно

Построить путь программно

Мы будем произвольно начинать в левом нижнем углу и работать по часовой стрелке. Я использую сетку в изображении, чтобы получить значения x и y для точек. Я здесь все жестко программирую, но, конечно, вы не сделали бы этого в реальном проекте.

Основной процесс:

1. Создайте новый `UIBezierPath`
2. Выберите начальную точку на пути с помощью `moveToPoint`
3. Добавить сегменты в путь

- line: `addLineToPoint`
- arc: `addArcWithCenter`

- кривая: `addCurveToPoint`

4. Закрыть путь с помощью `closePath`

Вот код, чтобы сделать путь на изображении выше.

```
func createBezierPath() -> UIBezierPath {

    // create a new path
    let path = UIBezierPath()

    // starting point for the path (bottom left)
    path.moveToPoint(CGPoint(x: 2, y: 26))

    // **********************
    // ***** Left side *****
    // **********************

    // segment 1: line
    path.addLineToPoint(CGPoint(x: 2, y: 15))

    // segment 2: curve
    path.addCurveToPoint(CGPoint(x: 0, y: 12), // ending point
        controlPoint1: CGPoint(x: 2, y: 14),
        controlPoint2: CGPoint(x: 0, y: 14))

    // segment 3: line
    path.addLineToPoint(CGPoint(x: 0, y: 2))

    // **********************
    // ****** Top side *****
    // **********************

    // segment 4: arc
    path.addArcWithCenter(CGPoint(x: 2, y: 2), // center point of circle
        radius: 2, // this will make it meet our path line
        startAngle: CGFloat(M_PI), // π radians = 180 degrees = straight left
        endAngle: CGFloat(3*M_PI_2), // 3π/2 radians = 270 degrees = straight up
        clockwise: true) // startAngle to endAngle goes in a clockwise direction

    // segment 5: line
    path.addLineToPoint(CGPoint(x: 8, y: 0))

    // segment 6: arc
    path.addArcWithCenter(CGPoint(x: 8, y: 2),
        radius: 2,
        startAngle: CGFloat(3*M_PI_2), // straight up
        endAngle: CGFloat(0), // 0 radians = straight right
        clockwise: true)

    // **********************
    // ***** Right side ****
    // **********************

    // segment 7: line
    path.addLineToPoint(CGPoint(x: 10, y: 12))

    // segment 8: curve
    path.addCurveToPoint(CGPoint(x: 8, y: 15), // ending point
        controlPoint1: CGPoint(x: 10, y: 14),
```

```
        controlPoint2: CGPoint(x: 8, y: 14))

    // segment 9: line
    path.addLineToPoint(CGPoint(x: 8, y: 26))

    // *********************
    // **** Bottom side ****
    // *********************

    // segment 10: line
    path.closePath() // draws the final line to close the path

    return path
}
```

Примечание. Некоторые из приведенного выше кода можно уменьшить, добавив строку и дугу в одну команду (поскольку дуга имеет подразумеваемую начальную точку). Подробнее см. Здесь .

Нарисуйте путь

Мы можем рисовать путь либо в слое, либо в `drawRect` .

Способ 1: рисование пути в слое

Наш пользовательский класс выглядит так. Мы добавляем наш путь Безье к новому `CAShapeLayer` когда представление инициализируется.

```
import UIKit
class MyCustomView: UIView {

    override init(frame: CGRect) {
        super.init(frame: frame)
        setup()
    }

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setup()
    }

    func setup() {

        // Create a CAShapeLayer
        let shapeLayer = CAShapeLayer()

        // The Bezier path that we made needs to be converted to
        // a CGPath before it can be used on a layer.
        shapeLayer.path = createBezierPath().CGPath

        // apply other properties related to the path
        shapeLayer.strokeColor = UIColor.blueColor().CGColor
        shapeLayer.fillColor = UIColor.whiteColor().CGColor
        shapeLayer.lineWidth = 1.0
        shapeLayer.position = CGPoint(x: 10, y: 10)
```

```
        // add the new layer to our custom view
        self.layer.addSublayer(shapeLayer)
    }

    func createBezierPath() -> UIBezierPath {

        // see previous code for creating the Bezier path
    }
}
```

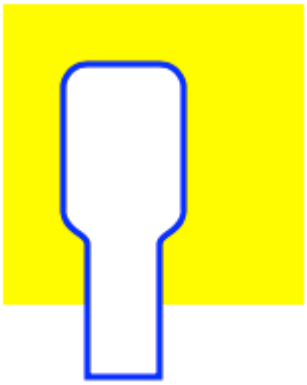
И создание нашего представления в View Controller, как это

```
override func viewDidLoad() {
    super.viewDidLoad()

    // create a new UIView and add it to the view controller
    let myView = MyCustomView()
    myView.frame = CGRect(x: 100, y: 100, width: 50, height: 50)
    myView.backgroundColor = UIColor.yellowColor()
    view.addSubview(myView)

}
```

Мы получаем...

Хм, это немного мало, потому что я жестко закодировал все числа. Я могу масштабировать размер пути вверх, хотя вот так:

```
let path = createBezierPath()
let scale = CGAffineTransformMakeScale(2, 2)
path.applyTransform(scale)
shapeLayer.path = path.CGPath
```



Способ 2: рисовать путь в `drawRect`

Использование `drawRect` выполняется медленнее, чем рисование на слой, поэтому это не рекомендуемый метод, если он вам не нужен.

Вот пересмотренный код для нашего пользовательского представления:

```
import UIKit
class MyCustomView: UIView {

    override func drawRect(rect: CGRect) {

        // create path (see previous code)
        let path = createBezierPath()

        // fill
        let fillColor = UIColor.whiteColor()
        fillColor.setFill()

        // stroke
        path.lineWidth = 1.0
        let strokeColor = UIColor.blueColor()
        strokeColor.setStroke()

        // Move the path to a new location
        path.applyTransform(CGAffineTransformMakeTranslation(10, 10))

        // fill and stroke the path (always do these last)
        path.fill()
        path.stroke()

    }

    func createBezierPath() -> UIBezierPath {

        // see previous code for creating the Bezier path
    }
}
```

который дает нам тот же результат ...

Дальнейшее обучение

Отличные статьи для понимания путей Безье.

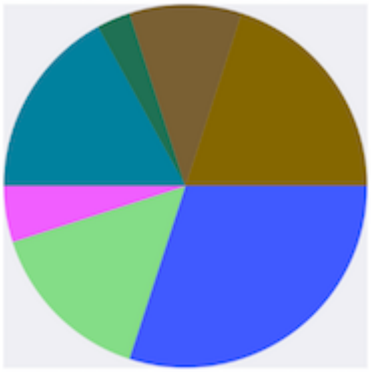
- Думая, как путь Безье (все, что я когда-либо читал от этого автора, хорошо, и вдохновение для моего примера выше пришло отсюда).
- Кодирующая математика: Эпизод 19 - Кривые Безье (развлекательные и хорошие визуальные иллюстрации)
- Кривые Безье (как они используются в графических приложениях)
- Кривые Безье (хорошее описание того, как производятся математические формулы)

Заметки

- Этот пример изначально исходит из этого ответа переполнения стека .
- В ваших реальных проектах вы, вероятно, не должны использовать жестко закодированные числа, а скорее получите размеры с границ ваших взглядов.

просмотр и просмотр столбцов с UIBezierPath

- просмотр пирога



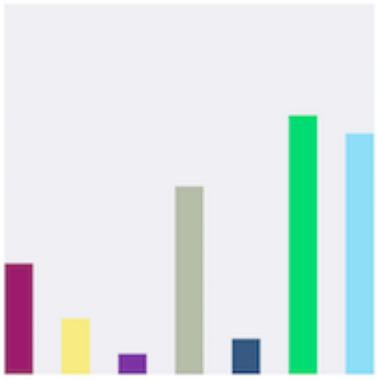
```
- (void)drawRect:(CGRect)rect {

    NSArray *data = @[@30, @15, @5, @17, @3, @10, @20];

    // 1. context
```

```
    CGContextRef cxtRef = UIGraphicsGetCurrentContext();

    CGPoint center = CGPointMake(150, 150);
    CGFloat radius = 150;
    __block CGFloat startAngle = 0;
    [data enumerateObjectsUsingBlock:^(NSNumber * _Nonnull obj, NSUInteger idx, BOOL *
_Nonnull stop) {

        // 2. create path
        CGFloat endAngle = obj.floatValue / 100 * M_PI * 2 + startAngle;
        UIBezierPath *circlePath = [UIBezierPath bezierPathWithArcCenter:center radius:radius
startAngle:startAngle endAngle:endAngle clockwise:YES];
        [circlePath addLineToPoint:center];

        // 3. add path
        CGContextAddPath(cxtRef, circlePath.CGPath);

        // set color
        [[UIColor colorWithRed:((float)arc4random_uniform(256) / 255.0)
green:((float)arc4random_uniform(256) / 255.0) blue:((float)arc4random_uniform(256) / 255.0)
alpha:1.0] setFill];

        // 4. render
        CGContextDrawPath(cxtRef, kCGPathFill);

        // reset angle
        startAngle = endAngle;
    }];
}
```

```
override func draw(_ rect: CGRect) {
    // define data to create pie chart
    let data: [Int] = [30, 15, 5, 17, 3, 10, 20]

    // 1. find center of draw rect
    let center: CGPoint = CGPoint(x: rect.midX, y: rect.midY)

    // 2. calculate radius of pie
    let radius = min(rect.width, rect.height) / 2.0

    var startAngle: CGFloat = 0.0
    for value in data {

      // 3. calculate end angle for slice
      let endAngle = CGFloat(value) / 100.0 * CGFloat.pi * 2.0 + startAngle

      // 4. create UIBezierPath for slide
      let circlePath = UIBezierPath(arcCenter: center, radius: radius, startAngle: startAngle,
endAngle: endAngle, clockwise: true)

      // 5. add line to center to close path
      circlePath.addLine(to: center)

      // 6. set fill color for current slice
      UIColor(red: (CGFloat(arc4random_uniform(256)) / 255.0), green:
(CGFloat(arc4random_uniform(256)) / 255.0), blue: (CGFloat(arc4random_uniform(256)) / 255.0),
alpha: 1.0).setFill()

      // 7. fill slice path
      circlePath.fill()
```

```
      // 8. set end angle as start angle for next slice
      startAngle = endAngle
    }
  }
```

- вид столбца



```
- (void)drawRect:(CGRect)rect {

    NSArray *data = @[@300, @150.65, @55.3, @507.7, @95.8, @700, @650.65];

    // 1.
    CGContextRef cxtRef = UIGraphicsGetCurrentContext();

    NSInteger columnCount = 7;
    CGFloat width = self.bounds.size.width / (columnCount + columnCount - 1);
    for (NSInteger i = 0; i < columnCount; i++) {

        // 2.
        CGFloat height = [data[i] floatValue] / 1000 * self.bounds.size.height;  // floatValue
        CGFloat x = 0 + width * (2 * i);
        CGFloat y = self.bounds.size.height - height;
        UIBezierPath *rectPath = [UIBezierPath bezierPathWithRect:CGRectMake(x, y, width,
height)];
        CGContextAddPath(cxtRef, rectPath.CGPath);

        // 3.
        [[UIColor colorWithRed:((float)arc4random_uniform(256) / 255.0)
green:((float)arc4random_uniform(256) / 255.0) blue:((float)arc4random_uniform(256) / 255.0)
alpha:1.0] setFill];
        CGContextDrawPath(cxtRef, kCGPathFill);
    }
}
```

```
override func draw(_ rect: CGRect) {
    // define data for chart
    let data: [CGFloat] = [300, 150.65, 55.3, 507.7, 95.8, 700, 650.65]

    // 1. calculate number of columns
    let columnCount = data.count

    // 2. calculate column width
    let columnWidth = rect.width / CGFloat(columnCount + columnCount - 1)

    for (columnIndex, value) in data.enumerated() {
      // 3. calculate column height
```

```
      let columnHeight = value / 1000.0 * rect.height

      // 4. calculate column origin
      let columnOrigin = CGPoint(x: (columnWidth * 2.0 * CGFloat(columnIndex)), y:
(rect.height - columnHeight))

      // 5. create path for column
      let columnPath = UIBezierPath(rect: CGRect(origin: columnOrigin, size: CGSize(width:
columnWidth, height: columnHeight)))

      // 6. set fill color for current column
      UIColor(red: (CGFloat(arc4random_uniform(256)) / 255.0), green:
(CGFloat(arc4random_uniform(256)) / 255.0), blue: (CGFloat(arc4random_uniform(256)) / 255.0),
alpha: 1.0).setFill()

      // 7. fill column path
      columnPath.fill()
    }
  }
```

Прочитайте UIBezierPath онлайн: https://riptutorial.com/ru/ios/topic/3186/uibezierpath

глава 71: UIButton

Вступление

UIButton : UIControl перехватывает события касания и отправляет сообщение о действии целевому объекту при его прослушивании. Вы можете установить название, изображение и другие свойства внешнего вида кнопки. Кроме того, вы можете указать другой вид для каждого состояния кнопки.

замечания

Типы кнопок

Тип кнопки определяет ее внешний вид и поведение. После создания кнопки вы не можете изменить ее тип. Наиболее часто используемыми типами кнопок являются пользовательские и системные типы, но при необходимости используются другие типы

- UIButtonTypeCustom

```
No button style.
```

- UIButtonTypeSystem

```
A system style button, such as those shown in navigation bars and toolbars.
```

- UIButtonTypeDetailDisclosure

```
A detail disclosure button.
```

- UIButtonTypeInfoLight

```
An information button that has a light background.
```

- UIButtonTypeInfoDark

```
An information button that has a dark background.
```

- UIButtonTypeContactAdd

```
A contact add button.
```

При создании пользовательской кнопки, которая является кнопкой с типом custom,

изначально на кадре кнопки устанавливаются (0, 0, 0, 0). Прежде чем добавлять кнопку в свой интерфейс, вы должны обновить фрейм до более подходящего значения.

Examples

Создание UIButton

UIButtons можно инициализировать в кадре:

стриж

```
let button = UIButton(frame: CGRect(x: x, y: y, width: width, height: height)
```

Цель C

```
UIButton *button = [[UIButton alloc] initWithFrame:CGRectMake(x, y, width, height)];
```

Конкретный тип UIButton можно создать следующим образом:

стриж

```
let button = UIButton(type: .Custom)
```

Цель C

```
UIButton *button = [UIButton buttonWithType:UIButtonTypeCustom];
```

где `type` - `type` `UIButtonType` :

```
enum UIButtonType : Int {
    case Custom
    case System
    case DetailDisclosure
    case InfoLight
    case InfoDark
    case ContactAdd
    static var RoundedRect: UIButtonType { get }
}
```

Установить заголовок

стриж

```
button.setTitle(titleString, forState: controlState)
```

Цель C

```
[button setTitle:(NSString *) forState:(UIControlState)];
```

Чтобы установить заголовок по умолчанию на «Hello, World!»

стриж

```
button.setTitle("Hello, World!", forState: .normal)
```

Цель C

```
[button setTitle:@"Hello, World!" forControlState:UIControlStateNormal];
```

Установить цвет заголовка

```
//Swift
button.setTitleColor(color, forControlState: controlState)

//Objective-C
[button setTitleColor:(nullable UIColor *) forState:(UIControlState)];
```

Чтобы установить цвет заголовка в синий

```
//Swift
button.setTitleColor(.blue, for: .normal)

//Objective-C
[button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal]
```

Горизонтальное выравнивание содержимого

стриж

```
//Align contents to the left of the frame
button.contentHorizontalAlignment = .left

//Align contents to the right of the frame
button.contentHorizontalAlignment = .right

//Align contents to the center of the frame
button.contentHorizontalAlignment = .center

//Make contents fill the frame
button.contentHorizontalAlignment = .fill
```

Цель C

```
//Align contents to the left
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentLeft;

//Align contents to the right
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentRight;
```

```
//Align contents to the center
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentCenter;

//Align contents to fill the frame
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentFill;
```

Получение метки заголовка

Базовый заголовок заголовка, если таковой существует, может быть извлечен с использованием

стриж

```
var label: UILabel? = button.titleLabel
```

Цель C

```
UILabel *label = button.titleLabel;
```

Это можно использовать для установки шрифта метки заголовка, например

стриж

```
button.titleLabel?.font = UIFont.boldSystemFontOfSize(12)
```

Цель C

```
button.titleLabel.font = [UIFont boldSystemFontOfSize:12];
```

Отключение UIButton

Кнопка может быть отключена

стриж

```
myButton.isEnabled = false
```

Objective-C:

```
myButton.enabled = NO;
```

Кнопка станет серой:

Если вы не хотите кнопку , чтобы изменить внешний вид при отключении набора `adjustsImageWhenDisabled` K `false` / `NO`

Добавление действия в UIButton через код (программно)

Чтобы добавить метод к кнопке, сначала создайте метод действия:

Objective-C

```
-(void)someButtonAction:(id)sender {
  // sender is the object that was tapped, in this case its the button.
    NSLog(@"Button is tapped");
}
```

стриж

```
func someButtonAction() {
    print("Button is tapped")
}
```

Теперь, чтобы добавить этот метод действий к вашей кнопке, вы должны написать следующую строку кода:

Цель C

```
[yourButtonInstance addTarget:self action:@selector(someButtonAction)
forControlEvents:UIControlEventTouchUpInside];
```

стриж

```
yourButtonInstance.addTarget(self, action: #selector(someButtonAction), forControlEvents:
.TouchUpInside)
```

Для параметра ControlEvents действительны все члены `ENUM` UIControlEvents .

Настройка шрифта

стриж

```
myButton.titleLabel?.font =  UIFont(name: "YourFontName", size: 20)
```

Цель C

```
myButton.titleLabel.font = [UIFont fontWithName:@"YourFontName" size:20];
```

Прикрепление метода к кнопке

Чтобы добавить метод к кнопке, сначала создайте метод действия:

Objective-C

```
-(void) someButtonAction{
    NSLog(@"Button is tapped");

}
```

стриж

```
func someButtonAction() {
        print("Button is tapped")
    }
```

Теперь, чтобы добавить этот метод действий к вашей кнопке, вы должны написать следующую строку кода:

Цель C

```
[yourButtonInstance addTarget:self action:@selector(someButtonAction)
forControlEvents:UIControlEventTouchUpInside];
```

стриж

```
yourButtonInstance.addTarget(self, action: #selector(someButtonAction), forControlEvents:
.touchUpInside)
```

Для ControlEvents действительны все члены `ENUM` UIControlEvents .

Получите размер UIButton строго на основе его текста и шрифта

Чтобы получить точный размер текста UIButton на основе его шрифта, используйте функцию `intrinsicContentSize` .

стриж

```
button.intrinsicContentSize.width
```

Objective-C

```
button.intrinsicContentSize.width;
```

Установить изображение

стриж

```
button.setImage(UIImage(named:"test-image"), forState: .normal)
```

Цель C

```
[self.button setImage:[UIImage imageNamed:@"test-image"] forState:UIControlStateNormal];
```

Несколько состояний управления

Вы также можете установить изображение для нескольких `UIControlStates` , например , чтобы установить тот же образ для `Selected` и `Highlighted` состояний:

стриж

```
button.setImage(UIImage(named:"test-image"), forState:[.selected, .highlighted])
```

Цель C

```
[self.button setImage:[UIImage imageNamed:@"test-image"]
forState:UIControlStateSelected|UIControlStateHighlighted];
```

Прочитайте UIButton онлайн: https://riptutorial.com/ru/ios/topic/516/uibutton

глава 72: UICollectionView

Examples

Создать программный сборник

стриж

```
func createCollectionView() {
    let layout: UICollectionViewFlowLayout = UICollectionViewFlowLayout()
    let collectionView = UICollectionView(frame: CGRect(x: 0, y: 0, width: view.frame.width,
height: view.frame.height), collectionViewLayout: layout)
    collectionView.dataSource = self
    collectionView.delegate = self
    view.addSubview(collectionView)
}
```

Objective-C

```
- (void)createCollectionView {
    UICollectionViewFlowLayout *layout = [[UICollectionViewFlowLayout alloc] init];
    UICollectionView *collectionView = [[UICollectionView alloc] initWithFrame:CGRectMake(0,
0, self.view.frame.size.width, self.view.frame.size.height) collectionViewLayout:layout];
    [collectionView setDataSource:self];
    [collectionView setDelegate:self];
    [self.view addSubview:collectionView];
}
```

Swift - UICollectionViewDelegateFlowLayout

```
// MARK: - UICollectionViewDelegateFlowLayout
extension ViewController: UICollectionViewDelegateFlowLayout {
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, sizeForItemAtIndexPath indexPath: NSIndexPath) -> CGSize {
        return CGSize(width: 50, height: 50)
    }
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, insetForSectionAtIndex section: Int) -> UIEdgeInsets {
        return UIEdgeInsets(top: 5, left: 5, bottom: 5, right: 5)
    }
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumLineSpacingForSectionAtIndex section: Int) -> CGFloat {
        return 5.0
    }
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumInteritemSpacingForSectionAtIndex section: Int) -> CGFloat {
        return 5.0
    }
}
```

Создание UICollectionView

Инициализируйте `UICollectionView` с помощью кадра `CGRect` :

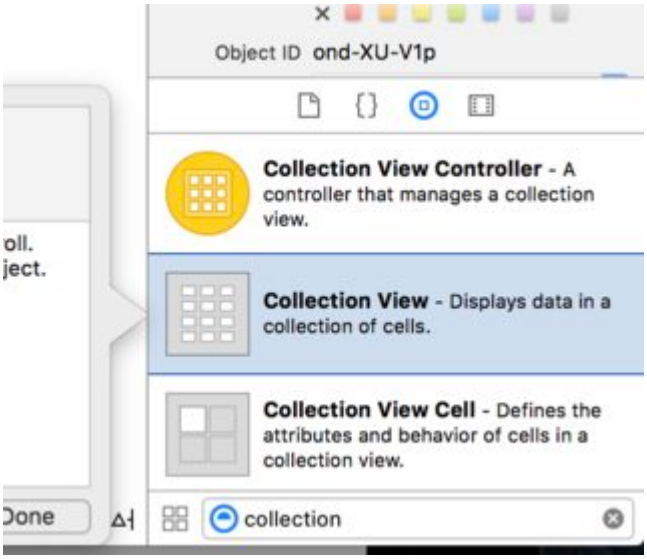
Swift:

```
let collection = UICollectionView(frame: CGRect(x: 0, y: 0, width: 200, height: 21))
```

Цель C:

```
UICollectionView *collection = [[UICollectionView alloc] initWithFrame:CGRectMake(0, 0, 200,
21)];
```

Вы также можете создать `UICollectionView` в интерфейсе Builder



UICollectionView - источник данных

В каждом представлении коллекции должен быть объект `Datasource` . Объект `Datasource` - это контент, отображаемый вашим приложением в `UICollectionView` . Как минимум, все `Datasource` объекты должны реализовывать `collectionView:numberOfItemsInSection:` и `collectionView:cellForItemAtIndexPath:` методы.

Необходимые методы

стриж

```
func collectionView(collectionView: UICollectionView, numberOfItemsInSection section: Int) ->
Int {
    // Return how many items in section
    let sectionArray = _data[section]
    return sectionArray.count
}

func collectionView(collectionView: UICollectionView, cellForItemAtIndexPath indexPath:
NSIndexPath) -> UICollectionViewCell {

   let cell = collectionView.dequeueReusableCellWithReuseIdentifier(MyCellID)
```

```
    // If you use a custom cell class then cast the cell returned, like:
    // as! MyCollectionViewCellClass
    // or you will have errors when you try to use features of that class.

    //Customize your cell here, default UICollectionViewCells do not contain any inherent
    //text or image views (like UITableView), but some could be added,
    //or a custom UICollectionViewCell sub-class could be used
    return cell
}
```

Цель C

```
- (NSInteger)collectionView:(UICollectionView*)collectionView
numberOfItemsInSection:(NSInteger)section {
    // Return how many items in section
    NSArray *sectionArray = [_data objectAtIndex:section];
    return [sectionArray count];
}

- (UICollectionViewCell *)collectionView:(UICollectionView *)collectionView
                  cellForItemAtIndexPath:(NSIndexPath *)indexPath {
    // Return a cell
    UICollectionViewCell *newCell = [self.collectionView
                                     dequeueReusableCellWithReuseIdentifier:MyCellID

                                                               forIndexPath:indexPath];
    //Customize your cell here, default UICollectionViewCells do not contain any inherent
    //text or image views (like UITableView), but some could be added,
    //or a custom UICollectionViewCell sub-class could be used
    return newCell;
}
```

Основной пример Swift для коллекции

Создать новый проект

Это может быть просто приложение с одним представлением.

Добавить код

Создайте новый файл класса Cocoa Touch Class (File> New> File ...> iOS> Cocoa Touch Class). Назовите его `MyCollectionViewCell` . Этот класс будет содержать выходы для просмотров, которые вы добавляете в свою ячейку в раскадровке.

```
import UIKit
class MyCollectionViewCell: UICollectionViewCell {

    @IBOutlet weak var myLabel: UILabel!
}
```

Мы подключим эту розетку позже.

Откройте ViewController.swift и убедитесь, что у вас есть следующий контент:

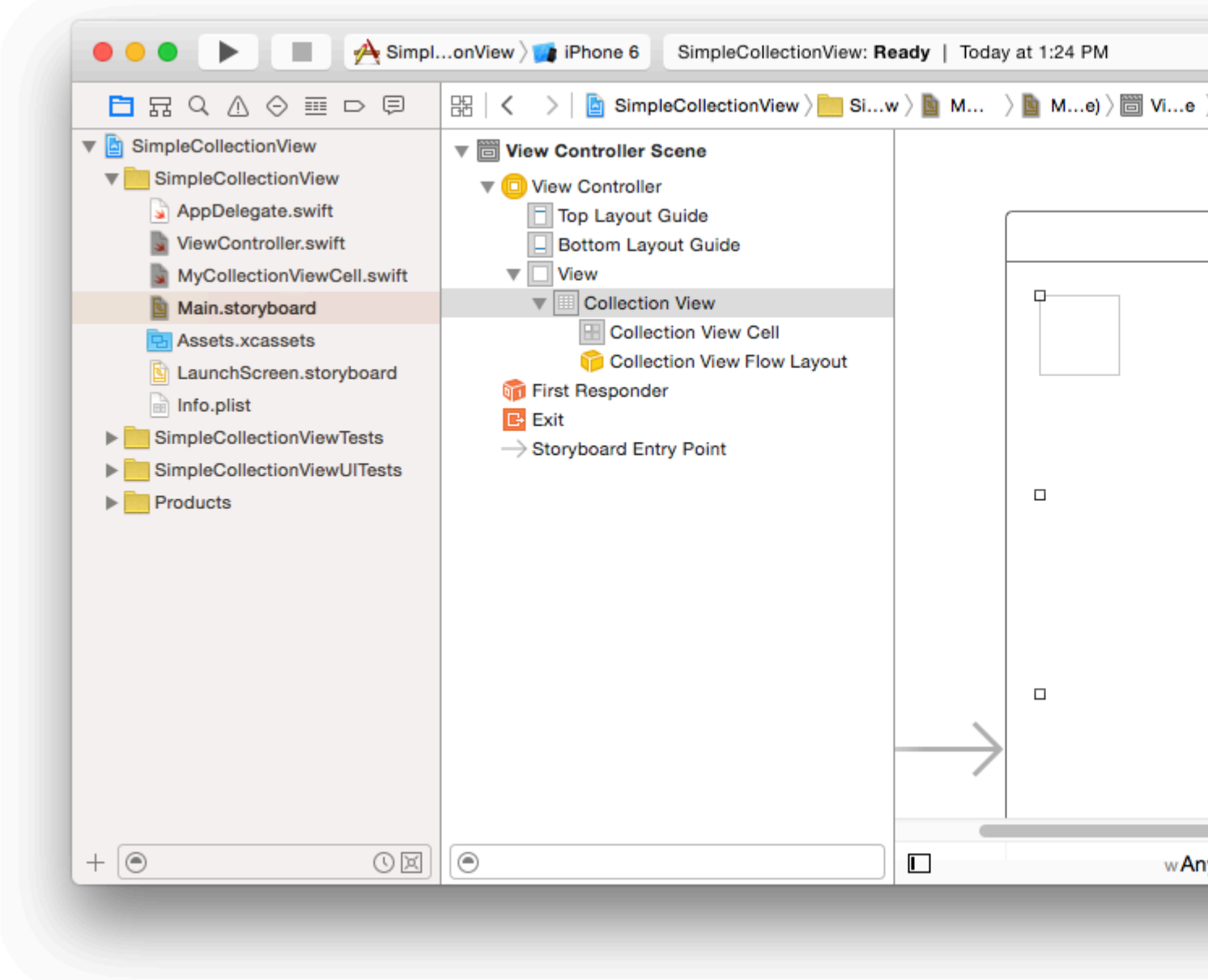
```
import UIKit
class ViewController: UIViewController, UICollectionViewDataSource, UICollectionViewDelegate {

    let reuseIdentifier = "cell" // also enter this string as the cell identifier in the
storyboard
    var items = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14",
"15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
"30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40", "41", "42", "43", "44",
"45", "46", "47", "48"]


    // MARK: - UICollectionViewDataSource protocol

    // tell the collection view how many cells to make
    func collectionView(collectionView: UICollectionView, numberOfItemsInSection section: Int)
-> Int {
        return self.items.count
    }

    // make a cell for each cell index path
    func collectionView(collectionView: UICollectionView, cellForItemAtIndexPath indexPath:
NSIndexPath) -> UICollectionViewCell {

        // get a reference to our storyboard cell
        let cell = collectionView.dequeueReusableCellWithReuseIdentifier(reuseIdentifier,
forIndexPath: indexPath) as! MyCollectionViewCell

        // Use the outlet in our custom class to get a reference to the UILabel in the cell
        cell.myLabel.text = self.items[indexPath.item]
        cell.backgroundColor = UIColor.yellowColor() // make cell more visible in our example
project

        return cell
    }

    // MARK: - UICollectionViewDelegate protocol

    func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath:
NSIndexPath) {
        // handle tap events
        print("You selected cell #\(indexPath.item)!")
    }
}
```

Заметки

- `UICollectionViewDataSource` и `UICollectionViewDelegate` - это протоколы, которые следуют в представлении коллекции. Вы также можете добавить протокол `UICollectionViewDelegateFlowLayout` для программного изменения размера представлений, но это необязательно.
- Мы просто помещаем простые строки в нашу сетку, но вы могли бы, конечно, сделать изображения позже.

Настройка раскадровки

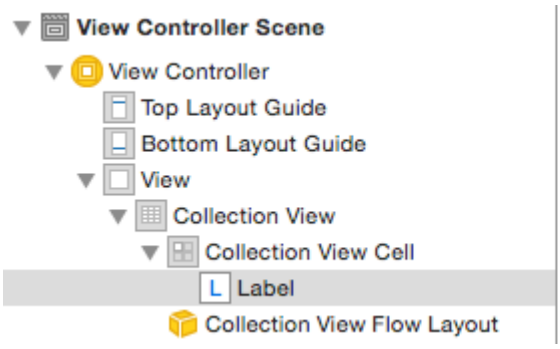
Перетащите представление коллекции в контроллер просмотра в своем раскадровке. Вы можете добавить ограничения, чтобы заполнить родительский вид, если хотите.



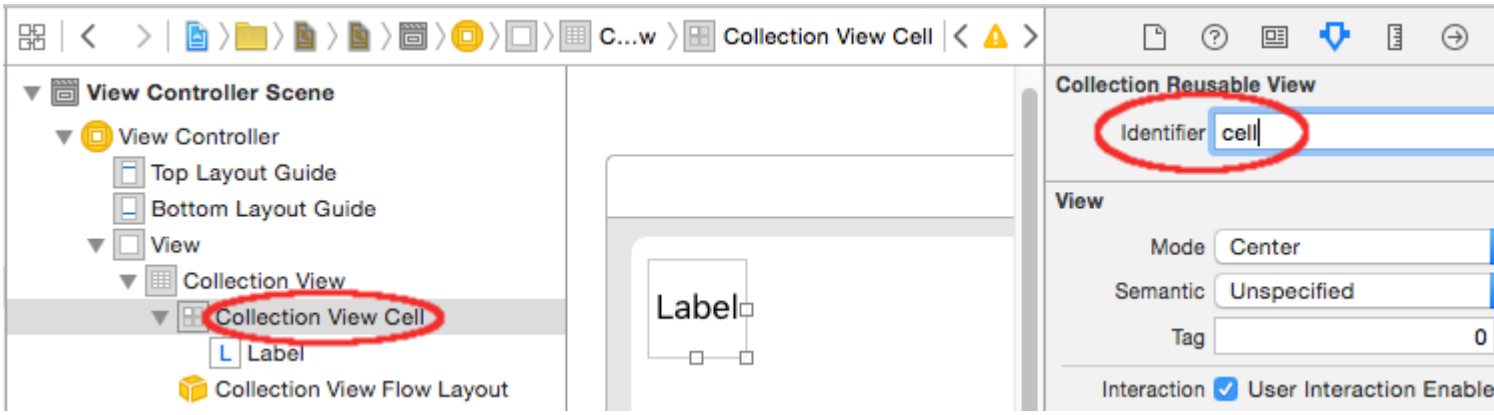
Убедитесь, что ваши настройки по умолчанию в инспекторе атрибутов также

- Предметы: 1
- Макет: поток

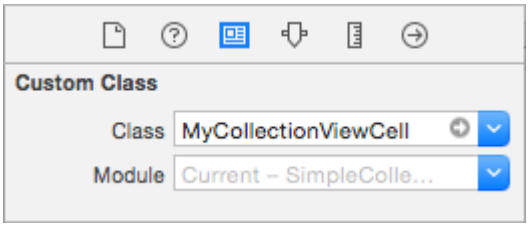
Маленькая коробка в левом верхнем углу коллекции представляет собой коллекцию View Cell. Мы будем использовать его как нашу прототипную ячейку. Перетащите ярлык в ячейку и центрируйте ее. Вы можете изменить размер границ ячеек и добавить ограничения, чтобы поместить ярлык, если хотите.



Напишите «ячейку» (без кавычек) в поле «Идентификатор» Инспектора атрибутов для ячейки «Просмотр коллекции». Обратите внимание, что это то же значение, что и `let reuseIdentifier = "cell"` в ViewController.swift.

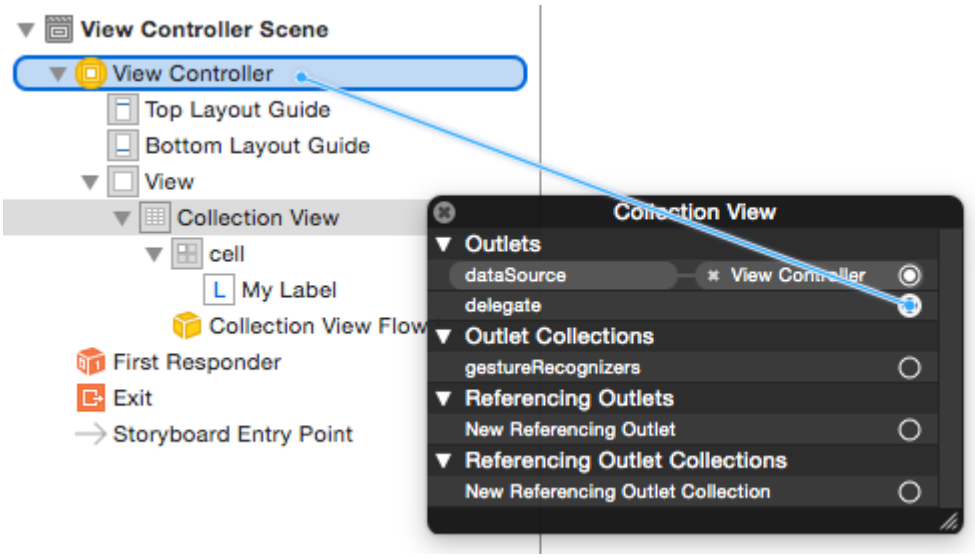


И в Identity Inspector для ячейки задайте имя класса `MyCollectionViewCell` , наш собственный класс, который мы создали.



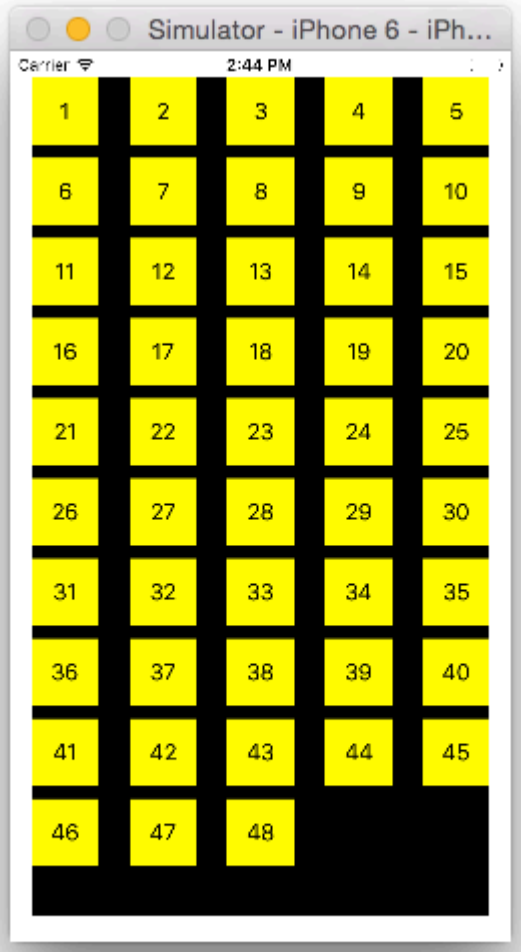
Подключите розетки

- `myLabel` ярлык в ячейке коллекции к `myLabel` в классе `MyCollectionViewCell` . (Вы можете управлять перетаскиванием .)
- `dataSource delegate` коллекции и `dataSource` к контроллеру представления. (Щелкните правой кнопкой мыши Вид коллекции в структуре документа. Затем щелкните и перетащите стрелку «плюс» вверх, чтобы просмотреть контроллер просмотра.)



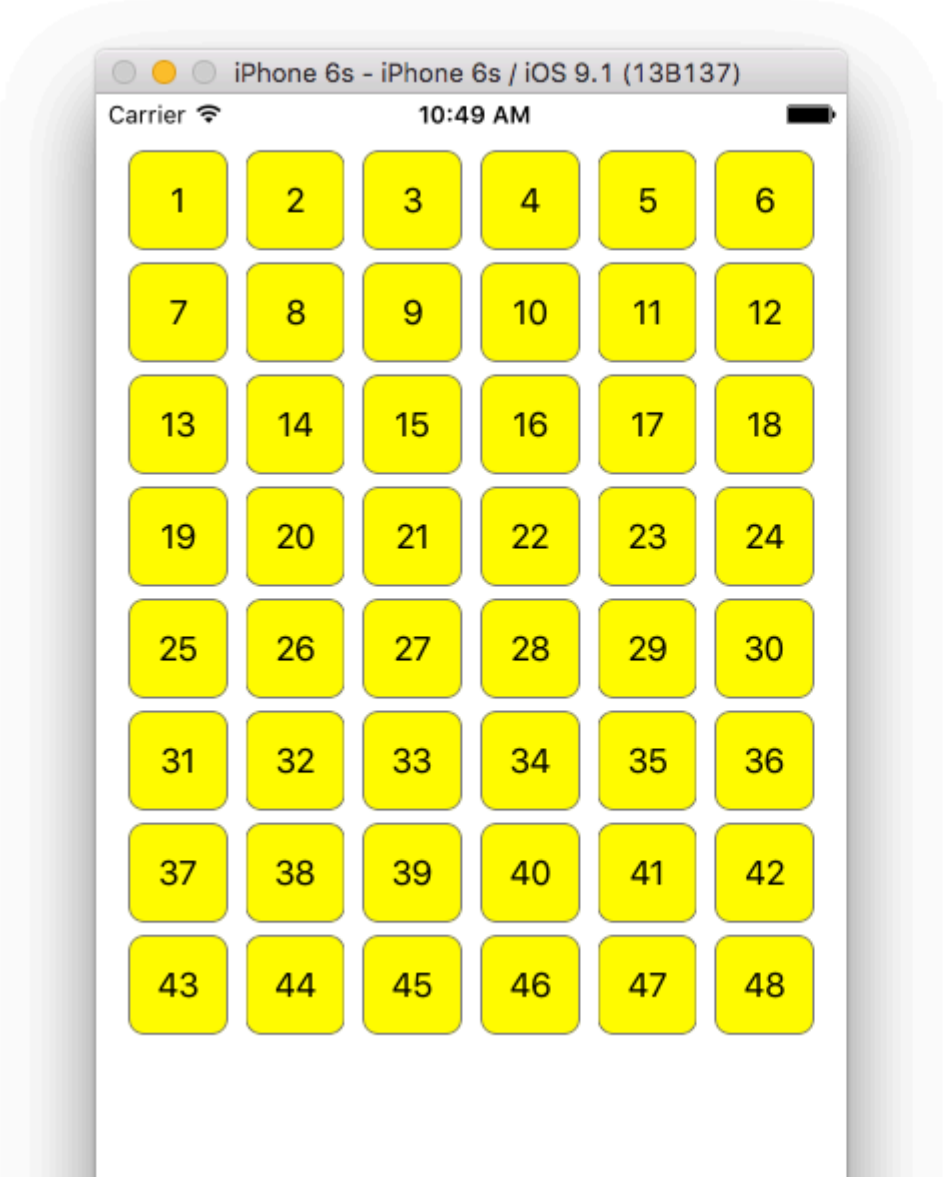
Законченный

Вот как это выглядит после добавления ограничений для центрирования метки в ячейке и привязки коллекции к стенам родителя.



Создание улучшений

Если вы хотите внести улучшения в внешний вид, см. Оригинальное сообщение, откуда приходит этот пример .



Дальнейшее обучение

- Простой учебник UICollectionView
- Учебник UICollectionView Часть 1: Начало работы
- Учебное пособие по UICollectionView Часть 2: Многоразовые представления и выбор ячеек

Выполнение пакетных обновлений

Вы можете анимировать сложные изменения в виде коллекции с `performBatchUpdates` метода `performBatchUpdates` . Внутри блока обновлений вы можете указать несколько модификаций, чтобы они сразу же оживили.

```
collecitonView.performBatchUpdates({
    // Perform updates
}, nil)
```

Внутри блока обновления вы можете выполнять вставки, удаления, перемещения и перезагрузки. Вот как определить, какой indexPath использовать:

Тип	NSIndexPath
вставка	Индекс в новом массиве
делеция	Индекс в старом массиве
Переехать	from: old array, to: new array
перезагружать	либо новый, либо старый массив (это не имеет значения)

Вы должны только перезагружать ячейки, которые не были перемещены, но их содержимое изменилось. Важно отметить, что перемещение не будет обновлять содержимое ячейки, а только перемещать его местоположение.

Чтобы убедиться, что пакетное обновление будет выполнено правильно, убедитесь, что набор индексов для `deletion` , `move-from` и `reload` уникален, а набор индексов для `insertion` , `move-to` и `reload` уникален.

Вот пример правильного пакетного обновления:

```
let from = [1, 2, 3, 4, 5]
let to = [1, 3, 6, 4, 5]

collecitonView.performBatchUpdates({
    collectionView.insertItemsAtIndexPaths([NSIndexPath(forItem: 2, inSection: 0)])
    collectionView.deleteItemsAtIndexPaths([NSIndexPath(forItem: 1, inSection: 0)])
    collectionView.moveItemAtIndexPath(NSIndexPath(forItem: 2, inSection: 0),
                                       toIndexPath: NSIndexPath(forItem: 1, inSection:0))
}, nil)
```

Настройка UICollectionViewDelegate и выбор позиции

Иногда, если действие должно быть привязано к выбору ячейки коллекции, вам необходимо реализовать протокол `UICollectionViewDelegate` .

Предположим, что представление коллекции находится внутри `UIViewController` `MyViewController` .

Objective-C

В *MyViewController.h* объявляется, что он реализует протокол `UICollectionViewDelegate` , как `UICollectionViewDelegate` ниже

```
@interface MyViewController : UIViewController <UICollectionViewDelegate, .../* previous
existing delegate, as UICollectionDataSource *>
```

стриж

В *MyViewController.swift* добавьте следующие

```
class MyViewController : UICollectionViewDelegate {
}
```

Метод, который должен быть реализован, - это

Objective-C

```
-(void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath
*)indexPath
{
}
```

стриж

```
func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath:
NSIndexPath)
{
}
```

В качестве примера мы можем установить цвет фона выбранной ячейки на зеленый.

Objective-C

```
-(void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath
*)indexPath
{
    UICollectionViewCell* cell = [collectionView cellForItemAtIndexPath:indexPath];
    cell.backgroundColor = [UIColor greenColor];
}
```

стриж

```
class MyViewController : UICollectionViewDelegate {
    func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath:
NSIndexPath)
    {
        var cell : UICollectionViewCell = collectionView.cellForItemAtIndexPath(indexPath)!
        cell.backgroundColor = UIColor.greenColor()
    }
}
```

Управление представлением нескольких коллекций с помощью DataSource и Flowlayout

Здесь мы управляем несколькими коллекциями, там делегируем методы с событиями didselect.

```
extension ProductsVC: UICollectionViewDelegate, UICollectionViewDataSource{

        // MARK: - UICollectionViewDataSource
        func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection
section: Int) -> Int {
            guard collectionView == collectionCategory else {
                return arrOfProducts.count
            }
            return arrOfCategory.count
        }

        func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath:
IndexPath) -> UICollectionViewCell {

            guard collectionView == collectionProduct else {
                  let cell = collectionView.dequeueReusableCell(withReuseIdentifier:
"ProductCategoryCell", for: indexPath) as! ProductCategoryCell
                  cell.viewBackground.layer.borderWidth = 0.5
                  //Do some thing as per use
                  return cell
            }

            let cell = collectionView.dequeueReusableCell(withReuseIdentifier: cellIdentifier,
for: indexPath) as! ProductCell
            cell.contentView.layer.borderWidth = 0.5
            cell.contentView.layer.borderColor = UIColor.black.cgColor
            let json = arrOfProducts[indexPath.row]
            //Do something as per use

            return cell
        }

        func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath:
IndexPath) {
            guard collectionView == collectionCategory else {
                let json = arrOfProducts[indexPath.row]
                // Do something for  collectionProduct here
                return
            }
            let json = arrOfCategory[indexPath.row] as [String: AnyObject]
            let id = json["cId"] as? String ?? ""
            // Do something
        }
    }

    extension ProductsVC: UICollectionViewDelegateFlowLayout{

        // MARK: - UICollectionViewDelegateFlowLayout
        func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {

            let collectionWidth = collectionView.bounds.width
            guard collectionView == collectionProduct else {
                var itemWidth = collectionWidth / 4 - 1;

                if(UI_USER_INTERFACE_IDIOM() == .pad) {
                    itemWidth = collectionWidth / 4 - 1;
```

```
                }
                return CGSize(width: itemWidth, height: 50)
            }

            var itemWidth = collectionWidth / 2 - 1;
            if(UI_USER_INTERFACE_IDIOM() == .pad) {
                itemWidth = collectionWidth / 4 - 1;
            }
            return CGSize(width: itemWidth, height: 250);
        }

        func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumInteritemSpacingForSectionAt section: Int) -> CGFloat {
            return 1
        }

        func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumLineSpacingForSectionAt section: Int) -> CGFloat {
            return 1
        }

    }
```

Прочитайте UICollectionView онлайн: https://riptutorial.com/ru/ios/topic/2399/uicollectionview

глава 73: UIColor

Examples

Создание UIColor

Существует множество способов создания `UIColor` :

стриж

- Используя один из предопределенных цветов:

```
let redColor = UIColor.redColor()
let blueColor: UIColor = .blueColor()

// In Swift 3, the "Color()" suffix is removed:
let redColor = UIColor.red
let blueColor: UIColor = .blue
```

  Если компилятор уже знает, что переменная является экземпляром `UIColor` вы можете пропустить тип все вместе:

```
let view = UIView()
view.backgroundColor = .yellowColor()
```

- Использование значения оттенков серого и альфа:

```
let grayscaleColor = UIColor(white: 0.5, alpha: 1.0)
```

- Использование оттенка, насыщенности, яркости и альфа:

```
let hsbColor = UIColor(
    hue: 0.4,
    saturation: 0.3,
    brightness: 0.7,
    alpha: 1.0
)
```

- Использование значений RGBA:

```
let rgbColor = UIColor(
    red: 30.0 / 255,
    green: 70.0 / 255,
    blue: 200.0 / 255,
    alpha: 1.0
)
```

- Использование рисунка:

```
let patternColor = UIColor(patternImage: UIImage(named: "myImage")!)
```

Objective-C

- Используя один из предопределенных цветов:

```
UIColor *redColor = [UIColor redColor];
```

- Использование значения оттенков серого и альфа:

```
UIColor *grayscaleColor = [UIColor colorWithWhite: 0.5 alpha: 1.0];
```

- Использование оттенка, насыщенности, яркости и альфа:

```
UIColor *hsbColor = [UIColor
    colorWithHue: 0.4
    saturation: 0.3
    brightness: 0.7
    alpha: 1.0
];
```

- Использование значений RGBA:

```
UIColor *rgbColor = [UIColor
    colorWithRed: 30.0 / 255.0
    green: 70.0 / 255.0
    blue: 200.0 / 255.0
    alpha: 1.0
];
```

- Использование рисунка:

```
UIColor *pattenColor = [UIColor colorWithPatternImage:[UIImage
imageNamed:@"myImage.png"]];
```

Недокументированные методы

На `UIColor` существует множество недокументированных методов, которые предоставляют альтернативные цвета или функциональные возможности. Их можно найти в личном заголовочном файле `UIColor` . Я буду документировать использование двух частных методов, `styleString()` и `_systemDestructiveTintColor()` .

`styleString`

Начиная с iOS 2.0 существует частный метод экземпляра в `UIColor` называемый `styleString` который возвращает строковое представление RGB или RGBA для цвета, даже для цветов типа `whiteColor` вне пространства RGB.

Objective-C:

```
@interface UIColor (Private)

- (NSString *)styleString;

@end

// ...

[[UIColor whiteColor] styleString]; // rgb(255,255,255)
[[UIColor redColor] styleString]; // rgb(255,0,0)
[[UIColor lightTextColor] styleString]; // rgba(255,255,255,0.600000)
```

В Swift вы можете использовать заголовок для моста, чтобы открыть интерфейс. С чистым Swift вам нужно будет создать протокол `@objc` с помощью частного метода и `unsafeBitCast` `UIColor` с протоколом:

```
@objc protocol  UIColorPrivate {
    func styleString() -> String
}

let white = UIColor.whiteColor()
let red = UIColor.redColor()
let lightTextColor = UIColor.lightTextColor()

let whitePrivate = unsafeBitCast(white, UIColorPrivate.self)
let redPrivate = unsafeBitCast(red, UIColorPrivate.self)
let lightTextColorPrivate = unsafeBitCast(lightTextColor, UIColorPrivate.self)

whitePrivate.styleString() // rgb(255,255,255)
redPrivate.styleString() // rgb(255,0,0)
lightTextColorPrivate.styleString() // rgba(255,255,255,0.600000)
```

`_systemDestructiveTintColor()`

Существует недокументированный метод класса в `UIColor` называемый `_systemDestructiveTintColor` который возвращает красный цвет, используемый деструктивными системными кнопками:

```
let red = UIColor.performSelector("_systemDestructiveTintColor").takeUnretainedValue()
```

Он возвращает неуправляемый объект, который вы должны вызывать `.takeUnretainedValue()` , поскольку `.takeUnretainedValue()` цвета не было передано нашему собственному объекту.

Как и в случае любого недокументированного API, вы должны проявлять осторожность при попытке использовать этот метод:

```
if UIColor.respondsToSelector("_systemDestructiveTintColor") {
    if let red = UIColor.performSelector("_systemDestructiveTintColor").takeUnretainedValue()
as? UIColor {
        // use the color
    }
```

```
}
```

или с использованием протокола:

```
@objc protocol UIColorPrivateStatic {
    func _systemDestructiveTintColor() -> UIColor
}

let privateClass = UIColor.self as! UIColorPrivateStatic
privateClass._systemDestructiveTintColor() // UIDeviceRGBColorSpace 1 0.231373 0.188235 1
```

Цвет с альфа-компонентом

Вы можете установить непрозрачность для определенного `UIColor` не создавая новый, используя `init(red:_,green:_,blue:_,alpha:_)` .

стриж

```
let colorWithAlpha = UIColor.redColor().colorWithAlphaComponent(0.1)
```

Swift 3

```
//In Swift Latest Version
_ colorWithAlpha = UIColor.red.withAlphaComponent(0.1)
```

Objective-C

```
UIColor * colorWithAlpha = [[UIColor redColor] colorWithAlphaComponent:0.1];
```

Создание пользовательских атрибутов применяется к типу данных CGColor

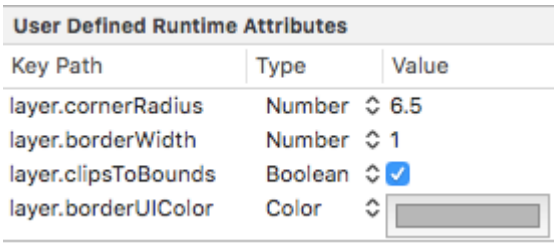
По умолчанию Interface Builder не принимает тип данных `CGColor` , поэтому позволяет добавлять `CGColor` с использованием пользовательских атрибутов в построителе интерфейса; можно использовать расширение следующим образом:

Быстрое расширение:

```
extension CALayer {
    func borderUIColor() -> UIColor? {
        return borderColor != nil ? UIColor(CGColor: borderColor!) : nil
    }

    func setBorderUIColor(color: UIColor) {
        borderColor = color.CGColor
    }
```

```
}
```

Новый пользовательский атрибут (borderUIColor) будет распознаваться и применяться без проблем.



Создание UIColor из шестнадцатеричного числа или строки

Вы можете создать `UIColor` из шестнадцатеричного числа или строки, например 0xff00cc, "#FFFFFF"

стриж

Int Value

```
extension UIColor {
    convenience init(hex: Int, alpha: CGFloat = 1.0) {
        let r = CGFloat((hex >> 16) & 0xff) / 255
        let g = CGFloat((hex >> 08) & 0xff) / 255
        let b = CGFloat((hex >> 00) & 0xff) / 255
        self.init(red: r, green: g, blue: b, alpha: alpha)
    }
}
```

Пример:

```
let color = UIColor(hex: 0xff00cc, alpha: 1.0)
```

Обратите внимание, что для `alpha` предоставляется значение по умолчанию `1.0` , поэтому его можно использовать следующим образом:

```
let color = UIColor(hex: 0xff00cc)
```

Строковое значение

```
extension UIColor {
    convenience init(hexCode: String) {
        let hex =
hexCode.stringByTrimmingCharactersInSet(NSCharacterSet.alphanumericCharacterSet().invertedSet)
        var int = UInt32()
        NSScanner(string: hex).scanHexInt(&int)
        let a, r, g, b: UInt32
```

```
        switch hex.characters.count {
        case 3:
            (a, r, g, b) = (255, (int >> 8) * 17, (int >> 4 & 0xF) * 17, (int & 0xF) * 17)
        case 6:
            (a, r, g, b) = (255, int >> 16, int >> 8 & 0xFF, int & 0xFF)
        case 8:
            (a, r, g, b) = (int >> 24, int >> 16 & 0xFF, int >> 8 & 0xFF, int & 0xFF)
        default:
            (a, r, g, b) = (1, 1, 1, 0)
        }

        self.init(red: CGFloat(r) / 255, green: CGFloat(g) / 255, blue: CGFloat(b) / 255,
alpha: CGFloat(a) / 255)
    }
}
```

Пример использования:

Шестигранник с альфой

```
let color = UIColor("#80FFFFFF")
```

Hex без альфы (`color` альфа будет равна 1,0)

```
let color = UIColor("#FFFFFF")
let color = UIColor("#FFF")
```

Objective-C

Int Value

```
@interface UIColor (Hex)
+ (UIColor *)colorWithHex:(NSUInteger)hex alpha:(CGFloat)alpha;
@end

@implementation UIColor (Hex)
+ (UIColor *)colorWithHex:(NSUInteger)hex alpha:(CGFloat)alpha {
    return [UIColor colorWithRed:((CGFloat)((hex & 0xFF0000) >> 16))/255.0
                           green:((CGFloat)((hex & 0xFF00) >> 8))/255.0
                            blue:((CGFloat)(hex & 0xFF))/255.0
                           alpha:alpha];
}
@end
```

Пример:

```
UIColor *color = [UIColor colorWithHex:0xff00cc alpha:1.0];
```

Строковое значение

```
- (UIColor*) hex:(NSString*)hexCode {
```

```
    NSString *noHashString = [hexCode stringByReplacingOccurrencesOfString:@"#"
withString:@""];
    NSScanner *scanner = [NSScanner scannerWithString:noHashString];
    [scanner setCharactersToBeSkipped:[NSCharacterSet symbolCharacterSet]];

    unsigned hex;
    if (![scanner scanHexInt:&hex]) return nil;
    int a;
    int r;
    int g;
    int b;

    switch (noHashString.length) {
        case 3:
            a = 255;
            r = (hex >> 8) * 17;
            g = ((hex >> 4) & 0xF) * 17;
            b = ((hex >> 0) & 0xF) * 17;
            break;
        case 6:
            a = 255;
            r = (hex >> 16);
            g = (hex >> 8) & 0xFF;
            b = (hex) & 0xFF;
            break;
        case 8:
            a = (hex >> 24);
            r = (hex >> 16) & 0xFF;
            g = (hex >> 8) & 0xFF;
            b = (hex) & 0xFF;
            break;

        default:
            a = 255.0;
            r = 255.0;
            b = 255.0;
            g = 255.0;
            break;
    }

    return [UIColor colorWithRed:r / 255.0f green:g / 255.0f blue:b / 255.0f alpha:a / 255];
}
```

Пример использования:

Шестигранник с альфой

```
UIColor* color = [self hex:@"#80FFFFFF"];
```

Нех без альфы (`color` альфа будет равна 1)

```
UIColor* color = [self hex:@"#FFFFFF"];
UIColor* color = [self hex:@"#FFF"];
```

Регулировка яркости цвета от UIColor

В приведенном ниже примере кода вы получите скорректированную версию этого цвета, где более высокий процент будет ярче, а более низкий процент будет темнее.

Objective-C

```
+ (UIColor *)adjustedColorForColor:(UIColor *)c : (double)percent
{
    if (percent < 0) percent = 0;

    CGFloat r, g, b, a;
    if ([c getRed:&r green:&g blue:&b alpha:&a])
        return [UIColor colorWithRed:MAX(r * percent, 0.0)
                               green:MAX(g * percent, 0.0)
                                blue:MAX(b * percent, 0.0)
                               alpha:a];
    return nil;
}
```

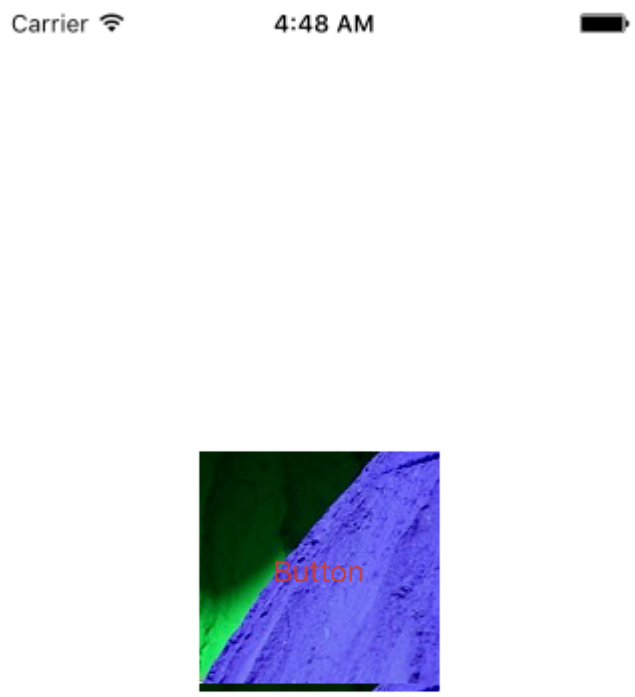
стриж

```
func adjustedColorForColor( c: UIColor, var percent: CGFloat) -> UIColor {
    if percent < 0 {
        percent = 0
    }

    var r,g,b,a: CGFloat
    r = 0.0
    g = 0.0
    b = 0.0
    a = 0.0

    if c.getRed(&r, green: &g, blue: &b, alpha: &a) {
        return UIColor(red: max(r * percent, 0.0), green: max(g * percent, 0.0), blue: max(b *
percent, 0.0), alpha: a)
    }

    return UIColor()
}
```

UIColor из шаблона изображения

Вы можете создать объект `UIColor` используя шаблон изображения, используя метод `UIColor(patternImage:_)` .

```
btn.backgroundColor = UIColor(patternImage: UIImage(named: "image")!)
```



Более светлый и темный оттенок данного UIColor

В приведенном ниже примере кода показано, как вы можете получить более светлый и темный оттенок заданного цвета, полезный в приложениях с динамическими темами

Для **темного цвета**

```
+ (UIColor *)darkerColorForColor:(UIColor *)c
{
    CGFloat r, g, b, a;
    if ([c getRed:&r green:&g blue:&b alpha:&a])
        return [UIColor colorWithRed:MAX(r - 0.2, 0.0)
                               green:MAX(g - 0.2, 0.0)
                                blue:MAX(b - 0.2, 0.0)
                               alpha:a];
    return nil;
}
```

Для более **светлых цветов**

```
+ (UIColor *)lighterColorForColor:(UIColor *)c
```

```
{
    CGFloat r, g, b, a;
    if ([c getRed:&r green:&g blue:&b alpha:&a])
        return [UIColor colorWithRed:MIN(r + 0.2, 1.0)
                               green:MIN(g + 0.2, 1.0)
                                blue:MIN(b + 0.2, 1.0)
                               alpha:a];
    return nil;
}
```

См. Раздел «Визуальные различия» ниже, учитывая, что данный цвет `[UIColor orangeColor]`

Прочитайте UIColor онлайн: https://riptutorial.com/ru/ios/topic/956/uicolor

глава 74: UIControl - Обработка событий с помощью блоков

Examples

Вступление

Как правило, при использовании `UIControl` или `UIButton` мы добавляем `selector` в качестве действия обратного вызова, когда событие происходит на кнопке или в управлении, например, при нажатии пользователем или прикосновении к элементу управления.

Например, мы сделали бы следующее:

```
import UIKit

class ViewController: UIViewController {
    @IBOutlet weak var button: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()

        let button = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 44))
        button.addTarget(self, action: #selector(self.onButtonPress(_:)), for: .touchUpInside)
        self.view.addSubview(button)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    func onButtonPress(_ button: UIButton!) {
        print("PRESSED")
    }
}
```

Когда дело доходит до `selector` , компилятор должен знать только, что он существует. Это можно сделать с помощью `protocol` и не выполнять.

Например, следующее приведет к сбою вашего приложения:

```
import UIKit

@objc
protocol ButtonEvent {
    @objc optional func onButtonPress(_ button: UIButton)
}

class ViewController: UIViewController, ButtonEvent {
    @IBOutlet weak var button: UIButton!
```

```
    override func viewDidLoad() {
        super.viewDidLoad()

        let button = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 44))
        button.addTarget(self, action: #selector(ButtonEvent.onButtonPress(_:)), for:
.touchUpInside)
        self.view.addSubview(button)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}
```

Это связано с тем, что ваше приложение НЕ реализует функцию `onButtonPress` .

Теперь, что бы вы могли сделать все это вместе с инициализацией кнопки? Что делать, если вам не нужно указывать обратные вызовы и вместо этого можно указать блоки, которые можно добавлять и удалять в любое время? Зачем беспокоиться о внедрении селекторов?

Решение

```
import Foundation
import UIKit

protocol RemovableTarget {
    func enable();
    func disable();
}

extension UIControl {
    func addEventHandler(event: UIControlEvents, runnable: (control: UIControl) -> Void) ->
RemovableTarget {

        class Target : RemovableTarget {
            private var event: UIControlEvents
            private weak var control: UIControl?
            private var runnable: (control: UIControl) -> Void

            private init(event: UIControlEvents, control: UIControl, runnable: (control:
UIControl) -> Void) {
                self.event = event
                self.control = control
                self.runnable = runnable
            }

            @objc
            private func run(_ control: UIControl) {
                runnable(control: control)
            }

            private func enable() {
                control?.addTarget(self, action: #selector(Target.run(_:)), for: event)
                objc_setAssociatedObject(self, unsafeAddress(of: self), self,
.OBJC_ASSOCIATION_RETAIN)
            }
```

```
            private func disable() {
                control?.removeTarget(self, action: #selector(Target.run(_:)), for:
self.event)
                objc_setAssociatedObject(self, unsafeAddress(of: self), nil,
.OBJC_ASSOCIATION_ASSIGN)
            }
        }

        let target = Target(event: event, control: self, runnable: runnable)
        target.enable()
        return target
    }
}
```

Вышеприведенное является простым расширением `UIControl` . Он добавляет внутренний закрытый класс, который имеет функцию `func run(_ control: UIControl)` callback `func run(_ control: UIControl)` которая используется как действие событий.

Затем мы используем `object association` для добавления и удаления цели, потому что она не будет сохранена `UIControl` .

Функция обработчика событий возвращает `Protocol` , чтобы скрыть внутреннюю работу класса `Target` а также позволить вам `enable` и `disable` цель в любой момент времени.

Пример использования:

```
import Foundation
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()


        //Create a button.
        let button = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 44))

        //Add an event action block/listener -- Handles Button Press.
        let target = button.addEventHandler(event: .touchUpInside) { (control) in
            print("Pressed")
        }


        self.view.addSubview(button)

        //Example of enabling/disabling the listener/event-action-block.
        DispatchQueue.main.after(when: DispatchTime.now() + 5) {
            target.disable() //Disable the listener.

            DispatchQueue.main.after(when: DispatchTime.now() + 5) {
                target.enable() //Enable the listener.
            }
        }
    }
```

```
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}
```

Прочитайте UIControl - Обработка событий с помощью блоков онлайн:
https://riptutorial.com/ru/ios/topic/3180/uicontrol---обработка-событий-с-помощью-блоков

глава 75: UIDatePicker

замечания

`UIDatePicker` не наследуется от `UIPickerView` , но он управляет пользовательским объектом `UIDatePicker` как подвью.

Examples

Создать выбор даты

стриж

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
```

Objective-C

```
UIDatePicker *datePicker = [[UIDatePicker alloc] initWithFrame:CGRectMake(x: 0, y: 0, width:
320, height: 200)];
```

Установка минимально-максимальной даты

Вы можете установить минимальную и максимальную дату, которые может показывать UIDatePicker.

Минимальная дата

```
[datePicker setMinimumDate:[NSDate date]];
```

Максимальная дата

```
[datePicker setMaximumDate:[NSDate date]];
```

Режимы

UIDatePicker имеет различные режимы выбора.

```
enum UIDatePickerMode : Int {
    case Time
    case Date
```

```
    case DateAndTime
    case CountDownTimer
}
```

- `Time` Выбор даты отображает часы, минуты и (необязательно) обозначение AM / PM.
- `Date` - выбор даты показывает месяцы, дни месяца и годы.
- `DateAndTime` - `DateAndTime` даты отображает даты (как единый день недели, месяц и день месяца) плюс часы, минуты и (необязательно) обозначение AM / PM.
- `CountDownTimer` - `CountDownTimer` даты отображает значения часов и минут, например [1 | 53]. Приложение должно установить таймер для стрельбы в надлежащий интервал и установить выбор даты в секундах.

Установка свойства datePickerMode

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.datePickerMode = .Date
```

Установка минутного интервала

Вы можете изменить свойство `minuteInterval` чтобы установить интервал, отображаемый колесом минут. Значение по умолчанию равно 1, максимальное значение - 30.

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.minuteInterval = 15
```

Длительность отсчета

Значение `NSTimeInterval` этого свойства указывает секунды, с которых отсчет даты в режиме обратного отсчета. Если режим выбора даты не является `CountDownTimer` , это значение игнорируется. Максимальное значение - 86 399 секунд (23:59)

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.countDownDuration = 60 * 60
```

Прочитайте UIDatePicker онлайн: https://riptutorial.com/ru/ios/topic/5643/uidatepicker

глава 76: UIDevice

параметры

Имущество	Описание
название	Имя, идентифицирующее устройство.
systemName: String	Имя операционной системы, запущенной на устройстве, представленном получателем.
модель: String	Модель устройства.
systemVersion: String	Текущая версия операционной системы.

замечания

Класс UIDevice предоставляет экземпляр Singleton, представляющий текущее устройство. Из этого экземпляра вы можете получить информацию об устройстве, такую как имя, модель устройства и имя и версия операционной системы.

Examples

Получить имя модели устройства iOS

Swift 2

```
import UIKit

extension UIDevice {

    var modelName: String {
        var systemInfo = utsname()
        uname(&systemInfo)
        let machineMirror = Mirror(reflecting: systemInfo.machine)
        let identifier = machineMirror.children.reduce("") { identifier, element in
            guard let value = element.value as? Int8 where value != 0 else { return identifier
}
            return identifier + String(UnicodeScalar(UInt8(value)))
        }

        switch identifier {
        case "iPod5,1":                                 return "iPod Touch 5"
        case "iPod7,1":                                 return "iPod Touch 6"
        case "iPhone3,1", "iPhone3,2", "iPhone3,3":     return "iPhone 4"
```

```
        case "iPhone4,1":                               return "iPhone 4s"
        case "iPhone5,1", "iPhone5,2":                  return "iPhone 5"
        case "iPhone5,3", "iPhone5,4":                  return "iPhone 5c"
        case "iPhone6,1", "iPhone6,2":                  return "iPhone 5s"
        case "iPhone7,2":                               return "iPhone 6"
        case "iPhone7,1":                               return "iPhone 6 Plus"
        case "iPhone8,1":                               return "iPhone 6s"
        case "iPhone8,2":                               return "iPhone 6s Plus"
        case "iPhone9,1", "iPhone9,3":                  return "iPhone 7"
        case "iPhone9,2", "iPhone9,4":                  return "iPhone 7 Plus"
        case "iPhone8,4":                               return "iPhone SE"
        case "iPad2,1", "iPad2,2", "iPad2,3", "iPad2,4":return "iPad 2"
        case "iPad3,1", "iPad3,2", "iPad3,3":           return "iPad 3"
        case "iPad3,4", "iPad3,5", "iPad3,6":           return "iPad 4"
        case "iPad4,1", "iPad4,2", "iPad4,3":           return "iPad Air"
        case "iPad5,3", "iPad5,4":                      return "iPad Air 2"
        case "iPad2,5", "iPad2,6", "iPad2,7":           return "iPad Mini"
        case "iPad4,4", "iPad4,5", "iPad4,6":           return "iPad Mini 2"
        case "iPad4,7", "iPad4,8", "iPad4,9":           return "iPad Mini 3"
        case "iPad5,1", "iPad5,2":                      return "iPad Mini 4"
        case "iPad6,3", "iPad6,4", "iPad6,7", "iPad6,8":return "iPad Pro"
        case "AppleTV5,3":                              return "Apple TV"
        case "i386", "x86_64":                          return "Simulator"
        default:                                        return identifier
        }
    }
}

if UIDevice.currentDevice().modelName == "iPhone 6 Plus" {
    // is an iPhone 6 Plus
}
```

Swift 3

```
import UIKit

public extension UIDevice {

    var modelName: String {
        var systemInfo = utsname()
        uname(&systemInfo)
        let machineMirror = Mirror(reflecting: systemInfo.machine)
        let identifier = machineMirror.children.reduce("") { identifier, element in
            guard let value = element.value as? Int8 , value != 0 else { return identifier
}
            return identifier + String(UnicodeScalar(UInt8(value)))
        }

        switch identifier {
        case "iPod5,1":                                 return "iPod Touch 5"
        case "iPod7,1":                                 return "iPod Touch 6"
        case "iPhone3,1", "iPhone3,2", "iPhone3,3":     return "iPhone 4"
        case "iPhone4,1":                               return "iPhone 4s"
        case "iPhone5,1", "iPhone5,2":                  return "iPhone 5"
        case "iPhone5,3", "iPhone5,4":                  return "iPhone 5c"
        case "iPhone6,1", "iPhone6,2":                  return "iPhone 5s"
        case "iPhone7,2":                               return "iPhone 6"
        case "iPhone7,1":                               return "iPhone 6 Plus"
        case "iPhone8,1":                               return "iPhone 6s"
        case "iPhone8,2":                               return "iPhone 6s Plus"
```

```
        case "iPhone9,1", "iPhone9,3":                  return "iPhone 7"
        case "iPhone9,2", "iPhone9,4":                  return "iPhone 7 Plus"
        case "iPhone8,4":                               return "iPhone SE"
        case "iPad2,1", "iPad2,2", "iPad2,3", "iPad2,4":return "iPad 2"
        case "iPad3,1", "iPad3,2", "iPad3,3":           return "iPad 3"
        case "iPad3,4", "iPad3,5", "iPad3,6":           return "iPad 4"
        case "iPad4,1", "iPad4,2", "iPad4,3":           return "iPad Air"
        case "iPad5,3", "iPad5,4":                      return "iPad Air 2"
        case "iPad2,5", "iPad2,6", "iPad2,7":           return "iPad Mini"
        case "iPad4,4", "iPad4,5", "iPad4,6":           return "iPad Mini 2"
        case "iPad4,7", "iPad4,8", "iPad4,9":           return "iPad Mini 3"
        case "iPad5,1", "iPad5,2":                      return "iPad Mini 4"
        case "iPad6,3", "iPad6,4", "iPad6,7", "iPad6,8":return "iPad Pro"
        case "AppleTV5,3":                              return "Apple TV"
        case "i386", "x86_64":                          return "Simulator"
        default:                                        return identifier
        }
    }
}

if UIDevice.current.modelName == "iPhone 7" {
    // is an iPhone 7
}
```

Получение статуса батареи и уровня заряда батареи

```
override func viewDidLoad() {
    super.viewDidLoad()
    NotificationCenter.default.addObserver(self, selector:
Selector(("batteryStateDidChange:")), name: NSNotification.Name.UIDeviceBatteryStateDidChange,
object: nil)
    NotificationCenter.default.addObserver(self, selector:
Selector(("batteryLevelDidChange:")), name: NSNotification.Name.UIDeviceBatteryLevelDidChange,
object: nil)

    // Stuff...
}

func batteryStateDidChange(notification: NSNotification){
    // The stage did change: plugged, unplugged, full charge...
}

func batteryLevelDidChange(notification: NSNotification){

    let batteryLevel = UIDevice.current.batteryLevel
    if batteryLevel < 0.0 {
        print(" -1.0 means battery state is UIDeviceBatteryStateUnknown")
        return
    }

    print("Battery Level : \(batteryLevel * 100)%")
    // The battery's level did change (98%, 99%, ...)
}
```

Идентификация устройства и работа

```
 UIDevice *deviceInfo = [UIDevice currentDevice];
```

```
 NSLog(@"Device Name %@", deviceInfo.name);
 //Ex: myIphone6s
 NSLog(@"System Name %@", deviceInfo.systemName);
 //Device Name iPhone OS
 NSLog(@"System Version %@", deviceInfo.systemVersion);
 //System Version 9.3
 NSLog(@"Model %@", deviceInfo.model);
 //Model iPhone
 NSLog(@"Localized Model %@", deviceInfo.localizedModel);
 //Localized Model iPhone
 int device=deviceInfo.userInterfaceIdiom;
 //UIUserInterfaceIdiomPhone=0
 //UIUserInterfaceIdiomPad=1
 //UIUserInterfaceIdiomTV=2
 //UIUserInterfaceIdiomCarPlay=3
 //UIUserInterfaceIdiomUnspecified=-1
 NSLog(@"identifierForVendor %@", deviceInfo.identifierForVendor);
 //identifierForVendor <__NSConcreteUUID 0x7a10ae20> 556395DC-0EB4-4FD5-BC7E-B16F612ECC6D
```

Получение ориентации устройства

```
UIDevice *deviceInfo = [UIDevice currentDevice];
int d = deviceInfo.orientation;
```

`deviceInfo.orientation` возвращает значение UIDeviceOrientation, которое показано ниже:

```
UIDeviceOrientationUnknown 0
UIDeviceOrientationPortrait 1
UIDeviceOrientationPortraitUpsideDown 2
UIDeviceOrientationLandscapeLeft 3
UIDeviceOrientationLandscapeRight 4
UIDeviceOrientationFaceUp 5
UIDeviceOrientationFaceDown 6
```

Прослушивание изменений ориентации устройства в контроллере просмотра:

```
- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    [[UIDevice currentDevice] beginGeneratingDeviceOrientationNotifications];
    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(deviceOrientationDidChange)
                                                 name:UIDeviceOrientationDidChangeNotification
                                               object:nil];
}

-(void)deviceOrientationDidChange
{
    UIDeviceOrientation orientation = [[UIDevice currentDevice] orientation];
    if (orientation == UIDeviceOrientationPortrait || orientation ==
UIDeviceOrientationPortraitUpsideDown) {
         [self changedToPortrait];
    } else if (orientation == UIDeviceOrientationLandscapeLeft || orientation ==
UIDeviceOrientationLandscapeRight) {
         [self changedToLandscape];
    }
```

```
-(void)changedToPortrait
{
    // Function Body
}

-(void)changedToLandscape
{
    // Function Body
}
```

Чтобы отключить проверку любого изменения ориентации:

```
- (void)viewWillDisappear:(BOOL)animated {
    [super viewWillDisappear:animated];
    [[UIDevice currentDevice] endGeneratingDeviceOrientationNotifications];
}
```

Получение состояния аккумулятора устройства

```
//Get permission for Battery Monitoring
[[UIDevice currentDevice] setBatteryMonitoringEnabled:YES];
UIDevice *myDevice = [UIDevice currentDevice];

[myDevice setBatteryMonitoringEnabled:YES];
double batLeft = (float)[myDevice batteryLevel] * 100;
NSLog(@"%.f",batLeft);

int d = myDevice.batteryState;
//Returns an Integer Value
//UIDeviceBatteryStateUnknown 0
//UIDeviceBatteryStateUnplugged 1
//UIDeviceBatteryStateCharging 2
//UIDeviceBatteryStateFull 3

//Using notifications for Battery Monitoring
-(void)startMonitoringForBatteryChanges
{
// Enable monitoring of battery status
[[UIDevice currentDevice] setBatteryMonitoringEnabled:YES];
// Request to be notified when battery charge or state changes
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(checkBatteryStatus)
name:UIDeviceBatteryLevelDidChangeNotification object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(checkBatteryStatus)
name:UIDeviceBatteryStateDidChangeNotification object:nil];
}
-(void) checkBatteryStatus
{
NSLog (@"Battery Level is %.f",[[UIDevice currentDevice] batteryLevel]*100);
 int d=[[UIDevice currentDevice] batteryState];
    if (d==0)
    {
        NSLog(@"Unknown");
    }
    else if (d==1)
    {
        NSLog(@"Unplugged");
    }
```

```
    else if (d==2)
    {
        NSLog(@"Charging");
    }
    else if (d==3)
    {
        NSLog(@"Battery Full");
    }
}
```

Использование датчика приближения

```
//Enabling the proximity Sensor
- (void)viewWillAppear:(BOOL)animated {

    [super viewWillAppear:animated];
    [[UIDevice currentDevice] setProximityMonitoringEnabled:YES];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(sensorStateMonitor:) name:@"UIDeviceProximityStateDidChangeNotification"
object:nil];
}

- (void)sensorStateMonitor:(NSNotificationCenter *)notification
{
    if ([[UIDevice currentDevice] proximityState] == YES)
    {
        NSLog(@"Device is close to user.");
    }

    else
    {
        NSLog(@"Device is not closer to user.");
    }
}
```

Прочитайте UIDevice онлайн: https://riptutorial.com/ru/ios/topic/4878/uidevice

глава 77: UIFeedbackGenerator

Вступление

`UIFeedbackGenerator` и его подклассы предлагают публичный интерфейс Taptic Engine®, который можно найти на устройствах iOS, начиная с iPhone 7. Haptics, фирменные Taptics, обеспечивают тактильную обратную связь для событий на экране. В то время как многие системные средства управления предоставляют `UIFeedbackGenerator` , разработчики могут использовать подклассы `UIFeedbackGenerator` для добавления haptics к пользовательским элементам управления и другим событиям. `UIFeedbackGenerator` - абстрактный класс, который нельзя использовать напрямую, а разработчики используют один из своих подклассов.

Examples

Эффект триггера

Пример показывает, как запускать ударное `UIImpactFeedbackGenerator` используя `UIImpactFeedbackGenerator` после нажатия кнопки.

стриж

```
class ViewController: UIViewController
{
    lazy var button: UIButton =
    {
        let button = UIButton()
        button.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(button)
        button.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
        button.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        button.setTitle("Impact", for: .normal)
        button.setTitleColor(UIColor.gray, for: .normal)
        return button
    }()

    // Choose between heavy, medium, and light for style
    let impactFeedbackGenerator = UIImpactFeedbackGenerator(style: .heavy)

    override func viewDidLoad()
    {
        super.viewDidLoad()
        button.addTarget(self, action: #selector(self.didPressButton(sender:)), for:
.touchUpInside)

        // Primes feedback generator for upcoming events and reduces latency
        impactFeedbackGenerator.prepare()
    }

    func didPressButton(sender: UIButton)
```

```
    {
        // Triggers haptic
        impactFeedbackGenerator.impactOccurred()
    }
}
```

Objective-C

```
@interface ViewController ()
@property (nonatomic, strong) UIImpactFeedbackGenerator *impactFeedbackGenerator;
@property (nonatomic, strong) UIButton *button;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.button addTarget:self action:@selector(didPressButton:)
forControlEvents:UIControlEventTouchUpInside];

    // Choose between heavy, medium, and light for style
    self.impactFeedbackGenerator = [[UIImpactFeedbackGenerator alloc]
initWithStyle:UIImpactFeedbackStyleHeavy];

    // Primes feedback generator for upcoming events and reduces latency
    [self.impactFeedbackGenerator prepare];
}

- (void)didPressButton:(UIButton *)sender
{
    // Triggers haptic
    [self.impactFeedbackGenerator impactOccurred];
}

#pragma mark - Lazy Init
- (UIButton *)button
{
    if (!_button)
    {
        _button = [[UIButton alloc]init];
        _button.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_button];
        [_button.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_button.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
        [_button setTitle:@"Impact" forState:UIControlStateNormal];
        [_button setTitleColor:[UIColor grayColor] forState:UIControlStateNormal];
    }
    return _button;
}

@end
```

Прочитайте UIFeedbackGenerator онлайн:
https://riptutorial.com/ru/ios/topic/10048/uifeedbackgenerator

глава 78: UIFont

Вступление

UIFont - это класс, который используется для получения и настройки информации, связанной с шрифтом. Он наследует от `NSObject` и соответствует `Hashable` , `Equatable` , `CVarArg` и `NSCopying` .

Examples

Объявление и инициализация UIFont

Вы можете объявить `UIFont` следующим образом:

```
var font: UIFont!
```

`UIFont` имеет больше методов `init()` :

- `UIFont.init(descriptor: UIFontDescriptor, size: CGFloat)`
- `UIFont.init(name: String, size: CGFloat)`

Поэтому вы можете инициализировать `UIFont` следующим образом:

```
let font = UIFont(name: "Helvetica Neue", size: 15)
```

Стандартным шрифтом является `System` , размер `17` .

Изменение шрифта метки

Чтобы изменить шрифт текста метки, вам необходимо получить доступ к свойству `font` :

```
label.font = UIFont(name:"Helvetica Neue", size: 15)
```

Приведенный выше код изменит шрифт метки на `Helvetica Neue` , размер `15` . Помните, что вы должны правильно называть имя шрифта, иначе он выкинет эту ошибку, потому что инициализированное значение выше является необязательным и, следовательно, может быть nil:

> Неожиданно было найдено нуль при развертывании необязательного значения

Прочитайте UIFont онлайн: https://riptutorial.com/ru/ios/topic/9792/uifont

глава 79: UIGestureRecognizer

Examples

UITapGestureRecognizer

Инициализируйте `UITapGestureRecognizer` с целью, `self` в этом случае, и `action` которое является методом, который имеет один параметр: `UITapGestureRecognizer` .

После инициализации добавьте его в представление, чтобы он распознал краны.

стриж

```
override func viewDidLoad() {
    super.viewDidLoad()
    let recognizer = UITapGestureRecognizer(target: self,
                                            action: #selector(handleTap(_:)))
    view.addGestureRecognizer(recognizer)
}

func handleTap(recognizer: UITapGestureRecognizer) {

}
```

Objective-C

```
- (void)viewDidLoad {
    [super viewDidLoad];
    UITapGestureRecognizer *recognizer =
        [[UITapGestureRecognizer alloc] initWithTarget:self
                                                action:@selector(handleTap:)];
    [self.view addGestureRecognizer:recognizer];
}

- (void)handleTap:(UITapGestureRecognizer *)recognizer {

}
```

Пример увольнения клавиатуры через UITapGestureRecognizer:

Во-первых, вы создаете функцию для отклонения клавиатуры:

```
func dismissKeyboard() {
        view.endEditing(true)
    }
```

Затем вы добавляете распознаватель жестов кран в свой контроллер просмотра, вызывая метод, который мы только что сделали

```
let tap: UITapGestureRecognizer = UITapGestureRecognizer(target: self, action:
"dismissKeyboard")
        view.addGestureRecognizer(tap)
```

Пример получения местоположения жестов UITapGestureRecognizer (Swift 3):

```
func handleTap(gestureRecognizer: UITapGestureRecognizer) {
print("tap working")
if gestureRecognizer.state == UIGestureRecognizerState.recognized
   {
     print(gestureRecognizer.location(in: gestureRecognizer.view))
   }
}
```

UIPanGestureRecognizer

Панорамирующие распознаватели обнаруживают жесты перетаскивания. Следующий пример добавляет изображение в контроллер просмотра и позволяет пользователю перетащить его на экран.

Objective-C

```
- (void)viewDidLoad {
    [super viewDidLoad];

    UIImageView *imageView = [[UIImageView alloc] initWithImage:[UIImage
imageNamed:@"imageToDrag"]];
    [imageView sizeToFit];
    imageView.userInteractionEnabled = YES;
    [self.view addSubview:imageView];

    UIPanGestureRecognizer *pan = [[UIPanGestureRecognizer alloc] initWithTarget:self
action:@selector(handlePan:)];
    [imageView addGestureRecognizer:pan];
}

- (void)handlePan:(UIPanGestureRecognizer *)recognizer {
    CGPoint translation = [recognizer translationInView:self.view];
    recognizer.view.center = CGPointMake(recognizer.view.center.x + translation.x,
                                         recognizer.view.center.y + translation.y);
    [recognizer setTranslation:CGPointZero inView:self.view];
}
```

стриж

```
override func viewDidLoad() {
    super.viewDidLoad()

    let imageView = UIImageView.init(image: UIImage.init(named: "imageToDrag"))
    imageView.sizeToFit()
    imageView.isUserInteractionEnabled = true
    self.view.addSubview(imageView)

    let pan = UIPanGestureRecognizer.init(target: self, action:
#selector(handlePan(recognizer:)))
```

```
    imageView.addGestureRecognizer(pan)
}

func handlePan(recognizer: UIPanGestureRecognizer) {
    let translation = recognizer.translation(in: self.view)
    if let view = recognizer.view {
        view.center = CGPoint(x: view.center.x + translation.x, y: view.center.y +
translation.y)
    }
    recognizer.setTranslation(CGPoint.zero, in: self.view)
}
```

> **Примечание.** Хотя `UIPanGestureRecognizer` полезен для обнаружения любых жестов перетаскивания, если вы просто хотите обнаружить основной жест, например, пользователь, перетаскивающий свой палец влево / вправо или вверх / вниз, используйте `UISwipeGestureRecognizer` . `UIPanGestureRecognizer` - лучший выбор, если вам нужен доступ к таким методам, как `translationInView:` или `velocityInView:`

UITapGestureRecognizer (Double Tap)

Двойной кран, как один кран, также использует `UITapGestureRecognizer` . Вы просто установите для параметра `numberOfTapsRequired` значение `2` .

стриж

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Double Tap
    let doubleTapGesture = UITapGestureRecognizer(target: self, action:
#selector(handleDoubleTap))
    doubleTapGesture.numberOfTapsRequired = 2
    doubleTapView.addGestureRecognizer(doubleTapGesture)
}

// Double tap action
func handleDoubleTap() {
    label.text = "Double tap recognized"
}
```

Заметки

- Примерный проект можно найти здесь .
- Вы можете распознать тройной `numberOfTapsRequired` установив для параметра `numberOfTapsRequired` значение `3` .

UILongPressGestureRecognizer

`UILongPressGestureRecognizer` позволяет прослушивать длительное нажатие на

представление. Вы можете установить длительность задержки до вызова метода действия.

стриж

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Long Press
    let longPressGesture = UILongPressGestureRecognizer(target: self, action:
#selector(handleLongPress(_:)))
    longPressView.addGestureRecognizer(longPressGesture)
}

// Long press action
func handleLongPress(gesture: UILongPressGestureRecognizer) {
    if gesture.state == UIGestureRecognizerState.Began {
        label.text = "Long press recognized"
    }
}
```

Заметки

- Более полный образец проекта можно найти здесь .
- Измените `minimumPressDuration` чтобы установить длительность длительного нажатия.

UISwipeGestureRecognizer

Прокручивание жестов позволяет вам слушать, как пользователь быстро перемещает палец по экрану в определенном направлении.

стриж

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Swipe (right and left)
    let swipeRightGesture = UISwipeGestureRecognizer(target: self, action:
#selector(handleSwipe(_:)))
    let swipeLeftGesture = UISwipeGestureRecognizer(target: self, action:
#selector(handleSwipe(_:)))
    swipeRightGesture.direction = UISwipeGestureRecognizerDirection.Right
    swipeLeftGesture.direction = UISwipeGestureRecognizerDirection.Left
    swipeView.addGestureRecognizer(swipeRightGesture)
    swipeView.addGestureRecognizer(swipeLeftGesture)
}

// Swipe action
func handleSwipe(gesture: UISwipeGestureRecognizer) {
    label.text = "Swipe recognized"

    // example task: animate view off screen
```

```
    let originalLocation = swipeView.center
    if gesture.direction == UISwipeGestureRecognizerDirection.Right {
        label.text = "Swipe right"
    } else if gesture.direction == UISwipeGestureRecognizerDirection.Left {
        label.text = "Swipe left"
    }
}
```

Objective-C

```
- (void)viewDidLoad
{
[super viewDidLoad];
UISwipeGestureRecognizer *swipeLeft = [[UISwipeGestureRecognizer alloc] initWithTarget:self
action:@selector(handleSwipe:)];
UISwipeGestureRecognizer *swipeRight = [[UISwipeGestureRecognizer alloc] initWithTarget:self
action:@selector(handleSwipe:)];

// Setting the swipe direction.
[swipeLeft setDirection:UISwipeGestureRecognizerDirectionLeft];
[swipeRight setDirection:UISwipeGestureRecognizerDirectionRight];

// Adding the swipe gesture on image view
[self.view addGestureRecognizer:swipeLeft];
[self.view addGestureRecognizer:swipeRight];

}
//Handling Swipe Gesture Events

- (void)handleSwipe:(UISwipeGestureRecognizer *)swipe {

    if (swipe.direction == UISwipeGestureRecognizerDirectionLeft) {
        NSLog(@"Left Swipe");
    }

    if (swipe.direction == UISwipeGestureRecognizerDirectionRight) {
        NSLog(@"Right Swipe");
    }

}
```

Заметки

- Более полный пример проекта можно найти здесь .

UIPinchGestureRecognizer

Пинчи - это жест с двумя пальцами, где пальцы движутся ближе или дальше друг от друга. Этот жест обычно используется для изменения размера представления.

стриж

```
override func viewDidLoad() {
    super.viewDidLoad()
```

```
    // Pinch
    let pinchGesture = UIPinchGestureRecognizer(target: self, action:
#selector(handlePinch(_:)))
    pinchView.addGestureRecognizer(pinchGesture)
}

// Pinch action
func handlePinch(gesture: UIPinchGestureRecognizer) {
    label.text = "Pinch recognized"

    if gesture.state == UIGestureRecognizerState.Changed {
        let transform = CGAffineTransformMakeScale(gesture.scale, gesture.scale)
        pinchView.transform = transform
    }
}
```

Заметки

- Более полный пример проекта можно найти здесь .

UIRotationGestureRecognizer

Два пальца, вращающихся вокруг центра, можно прослушать с помощью `UIRotationGestureRecognizer` . Обычно это используется для поворота представления.

стриж

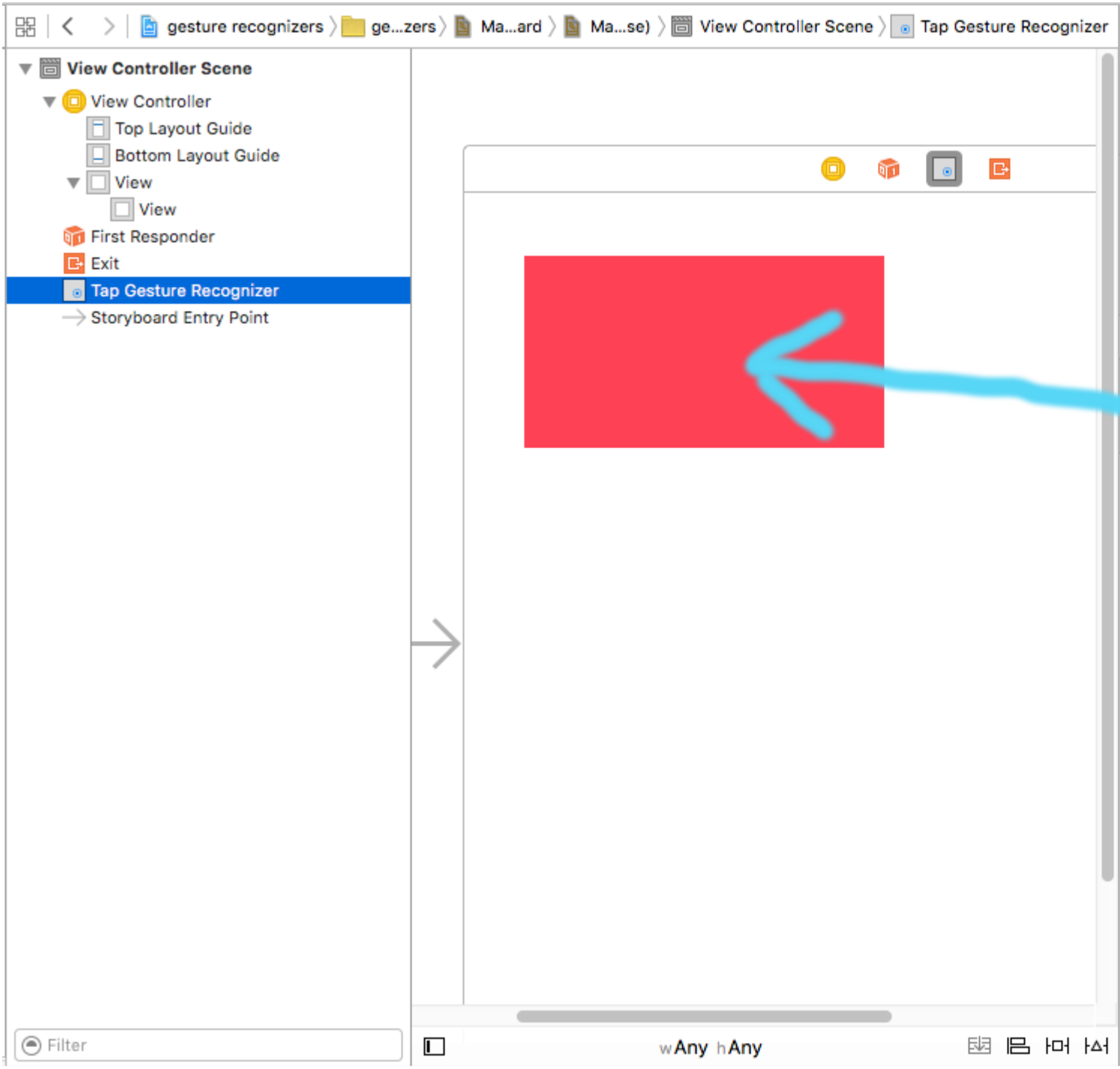
```
override func viewDidLoad() {
    super.viewDidLoad()

    // Rotate
    let rotateGesture = UIRotationGestureRecognizer(target: self, action:
#selector(handleRotate(_:)))
    rotateView.addGestureRecognizer(rotateGesture)
}

// Rotate action
func handleRotate(gesture: UIRotationGestureRecognizer) {
    label.text = "Rotate recognized"

    if gesture.state == UIGestureRecognizerState.Changed {
        let transform = CGAffineTransformMakeRotation(gesture.rotation)
        rotateView.transform = transform
    }
}
```

Заметки

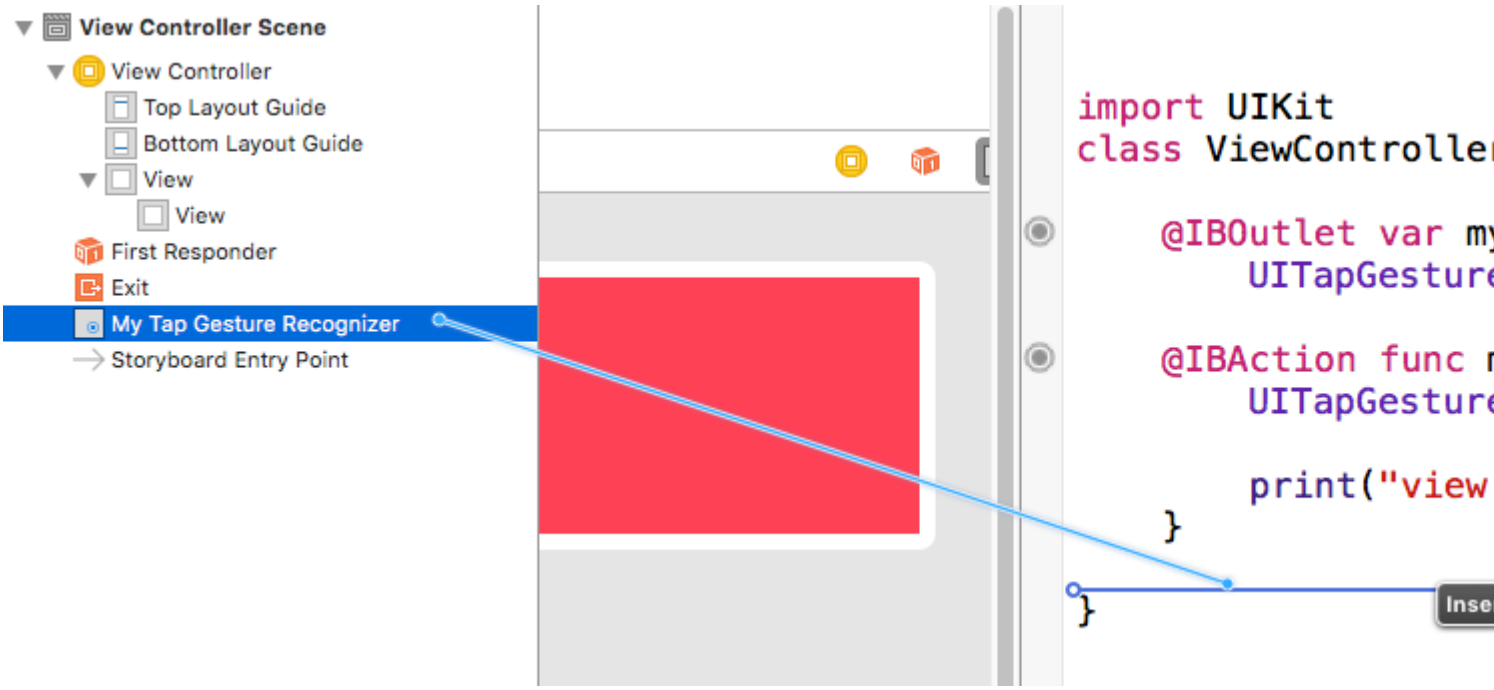
- Примерный проект можно найти здесь .

Добавление распознавателя жестов в Interface Builder

Перетащите распознаватель жестов из библиотеки объектов на свой вид.



Управляйте перетаскиванием с жестов в структуре документа на ваш код контроллера просмотра, чтобы сделать выход и действие.



Заметки

- Этот пример исходит из этого более полного примера проекта, демонстрирующего распознаватели жестов.

Прочитайте UIGestureRecognizer онлайн:
https://riptutorial.com/ru/ios/topic/1289/uigesturerecognizer

глава 80: UIImage

замечания

Тема разработчика Apple для UIImage

Examples

Создание UIImage

С локальным изображением

стриж

```
let image = UIImage(named: "imageFromBundleOrAsset")
```

Objective-C

```
UIImage *image = [UIImage imageNamed:@"imageFromBundleOrAsset"];
```

Заметка

Метод `imageNamed` кэширует содержимое изображения в память. Загрузка большого количества изображений таким образом может привести к появлению предупреждений о низкой памяти, которые могут привести к прекращению действия приложения. Это можно устранить, используя метод `imageWithContentsOfFile UIImage` , который не использует кеширование.

С NSData

стриж

```
let imageData = Data(base64Encoded: imageString, options:
Data.Base64DecodingOptions.ignoreUnknownCharacters)

let image = UIImage(data: imageData!)
```

C UIColor

стриж

```
let color = UIColor.red
let size = CGSize(width: 200, height: 200)

UIGraphicsBeginImageContextWithOptions(size, false, 0.0)
UIGraphicsGetCurrentContext()!.setFillColor(color.cgColor)
UIGraphicsGetCurrentContext()!.fill(CGRect(origin: .zero, size: size))
let colorImage = UIGraphicsGetImageFromCurrentImageContext()
UIGraphicsEndImageContext()
```

Objective-C

```
UIColor *color=[UIColor redColor];
CGRect frame = CGRectMake(0, 0, 80, 100);
UIGraphicsBeginImageContext(frame.size);
CGContextRef context = UIGraphicsGetCurrentContext();
CGContextSetFillColorWithColor(context, [color CGColor]);
CGContextFillRect(context, frame);
UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
```

C содержимым файла

Objective-C

Пример:

```
UIImage *image = [UIImage imageWithContentsOfFile:[[NSBundle mainBundle]
pathForResource:[cellCountry objectForKey:@"Country_Flag"] ofType:nil]];
```

Использование массива:

Пример:

```
NSMutableArray *imageArray = [[NSMutableArray alloc] init];

for (int imageNumber = 1; self.myPhoto != nil; imageNumber++) {
    NSString *fileName = [NSString stringWithFormat:@"%@.jpg", self.myPhoto];

    // check if a file exists
    if ([UIImage imageNamed:fileName]) {
        // if it exists, add it to the array
        [imageArray addObject:[UIImage imageWithContentsOfFile:[[NSBundle
```

```
mainBundle]pathForResource:[NSString stringWithFormat:@"%@", fileName] ofType:@""]]];
    } else {
        break;
    }
}
```

// Использование массива изображений для анимации здесь:

```
self.myImageView.animationImages = imageArray;
```

Создание и инициализация объектов изображения с содержимым файла

Создание и возврат объекта изображения путем загрузки данных изображения из файла по указанному пути.

Пример:

```
 UIImage *image = [UIImage imageWithContentsOfFile:[[NSBundle mainBundle]
pathForResource:[cellCountry objectForKey:@"Country_Flag"] ofType:nil]];
```

Использование массива:

пример

```
 NSMutableArray *imageArray = [[NSMutableArray alloc] init];

for (int imageNumber = 1; self.myPhoto != nil; imageNumber++) {
    NSString *fileName = [NSString stringWithFormat:@"%@.jpg", self.myPhoto];

    // check if a file exists
    if ([UIImage imageNamed:fileName]) {
        // if it exists, add it to the array
        [imageArray addObject:[UIImage imageWithContentsOfFile:[[NSBundle
mainBundle]pathForResource:[NSString stringWithFormat:@"%@", fileName] ofType:@""]]];
    } else {
        break;
    }
}

//Using image array for animations here
self.myImageView.animationImages = imageArray;
```

Съемное изображение с крышками

В приведенном ниже примере пузырька сообщения: углы изображения должны оставаться неизменными, что определено `UIEdgeInsets` , но границы и центр изображения должны расширяться, чтобы покрывать новый размер.

```
let insets = UIEdgeInsetsMake(12.0, 20.0, 22.0, 12.0)
let image = UIImage(named: "test")
image?.resizableImageWithCapInsets(insets, resizingMode: .Stretch)
```

Сравнение изображений

> Метод `isEqual:` единственный надежный способ определить, содержат ли два изображения одни и те же данные изображения. Создаваемые объекты изображения могут отличаться друг от друга, даже если вы инициализируете их с теми же данными кэшированного изображения. Единственный способ определить их равенство - использовать метод `isEqual:` который сравнивает фактические данные изображения. В листинге 1 показаны правильные и неправильные способы сравнения изображений.
>
> Источник: Apple Documentation

стриж

```
// Load the same image twice.
let image1 = UIImage(named: "MyImage")
let image2 = UIImage(named: "MyImage")

// The image objects may be different, but the contents are still equal
if let image1 = image1, image1.isEqual(image2) {
    // Correct. This technique compares the image data correctly.
}

if image1 == image2 {
    // Incorrect! Direct object comparisons may not work.
}
```

Objective-C

```
// Load the same image twice.
UIImage* image1 = [UIImage imageNamed:@"MyImage"];
UIImage* image2 = [UIImage imageNamed:@"MyImage"];

// The image objects may be different, but the contents are still equal
if ([image1 isEqual:image2]) {
   // Correct. This technique compares the image data correctly.
}

if (image1 == image2) {
```

```
   // Incorrect! Direct object comparisons may not work.
}
```

Создать UIImage с помощью UIColor

стриж

```
let color = UIColor.redColor()
let size = CGSize(width: 200, height: 200)

UIGraphicsBeginImageContextWithOptions(size, false, 0.0)
CGContextSetFillColorWithColor(UIGraphicsGetCurrentContext(), color.CGColor)
CGContextFillRect(UIGraphicsGetCurrentContext(), CGRect(origin: .zero, size: size))
let colorImage = UIGraphicsGetImageFromCurrentImageContext()
UIGraphicsEndImageContext()
```

Swift 3

```
let color = UIColor.red()
let size = CGSize(width: 200, height: 200)

UIGraphicsBeginImageContextWithOptions(size, false, 0.0)
if let context = UIGraphicsGetCurrentContext() {
    context.setFillColor(color.cgColor)
    context.fill(CGRect(origin: .zero, size: size))
    let colorImage = UIGraphicsGetImageFromCurrentImageContext()
}
UIGraphicsEndImageContext()
```

Objective-C:

Добавьте этот метод как расширение `UIImage` :

```
+ (UIImage *)createImageWithColor: (UIColor *)color {
    CGRect rect=CGRectMake(0.0f, 0.0f, 1.0f, 1.0f);
    UIGraphicsBeginImageContext(rect.size);
    CGContextRef context = UIGraphicsGetCurrentContext();
    CGContextSetFillColorWithColor(context, [color CGColor]);
    CGContextFillRect(context, rect);

    UIImage *theImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return theImage;
}
```

Градиентное изображение с цветами

Создание градиента `UIImage` с цветами в `CGRect`

Swift:

```
extension UIImage {
    static func gradientImageWithBounds(bounds: CGRect, colors: [CGColor]) -> UIImage {
        let gradientLayer = CAGradientLayer()
        gradientLayer.frame = bounds
        gradientLayer.colors = colors

        UIGraphicsBeginImageContext(gradientLayer.bounds.size)
        gradientLayer.render(in: UIGraphicsGetCurrentContext()!)
        let image = UIGraphicsGetImageFromCurrentImageContext()
        UIGraphicsEndImageContext()
        return image!
    }
}
```

Использование:

```
let image = UIImage.gradientImageWithBounds(CGRect(x: 0, y: 0, width: 200, height: 200),
colors: [UIColor.yellowColor().CGColor, UIColor.blueColor().CGColor])
```

Objective-C:

```
+ (UIImage *)gradientImageWithBounds:(CGRect)bounds colors:(NSArray *)colors {
    CAGradientLayer *gradientLayer = [CAGradientLayer layer];
    gradientLayer.frame = bounds;
    gradientLayer.colors = colors;

    UIGraphicsBeginImageContext(gradientLayer.bounds.size);
    [gradientLayer renderInContext:UIGraphicsGetCurrentContext()];
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return image;
}
```

Градиентный фоновый слой для границ

```
+ (CALayer *)gradientBGLayerForBounds:(CGRect)bounds colors:(NSArray *)colors
{
    CAGradientLayer * gradientBG = [CAGradientLayer layer];
    gradientBG.frame = bounds;
    gradientBG.colors = colors;
    return gradientBG;
}
```

Преобразование UIImage в / из base64 кодирования

кодирование

```
//convert the image to NSData first
let imageData:NSData = UIImagePNGRepresentation(image)!
// convert the NSData to base64 encoding
let strBase64:String =
```

```
imageData.base64EncodedStringWithOptions(.Encoding64CharacterLineLength)
```

расшифровка

```
let dataDecoded:NSData = NSData(base64EncodedString: strBase64, options:
NSDataBase64DecodingOptions(rawValue: 0))!
let decodedimage:UIImage = UIImage(data: dataDecoded)!
```

Сделайте снимок UIView

```
//Here self.webView is the view whose screenshot I need to take
//The screenshot is saved in jpg format in the application directory to avoid any loss of
quality in retina display devices i.e. all current devices running iOS 10
UIGraphicsBeginImageContextWithOptions(self.webView.bounds.size, NO, [UIScreen
mainScreen].scale);
[self.webView.layer renderInContext:UIGraphicsGetCurrentContext()];
UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
NSString  *jpgPath = [NSHomeDirectory() stringByAppendingPathComponent:@"Documents/Test.jpg"];
[UIImageJPEGRepresentation(image, 1.0) writeToFile:jpgPath atomically:YES];
UIImage *pop=[[UIImage alloc]initWithContentsOfFile:jpgPath];
//pop is the final image in jpg format and high quality with the exact resolution of the view
you selected in pixels and not just points
```

Применить UIColor к UIImage

Используйте тот же UIImage с несколькими приложениями для основной темы, просто применив UIColor к экземпляру UIImage следующим образом.

```
// *** Create an UIImage instance with RenderingMode AlwaysTemplate ***
UIImage *imgMenu = [[UIImage imageNamed:@"iconMenu"]
imageWithRenderingMode:UIImageRenderingModeAlwaysTemplate];

// *** Now Apply `tintColor` to `UIImageView` of UIImageView or UIButton and convert image in
given color ***
[btn setImage:imgMenu forState:UIControlStateNormal]; // Set UIImage in UIButton.

[button.imageView setTintColor:[UIColor blueColor]]; // It changes image color of UIButton to
blue color
```

Теперь скажем, вы хотите сделать то же самое с UIImageView, а затем использовать следующий код

```
[imageView setImage:imgMenu]; // Assign UIImage to UIImageView
[imageView setTintColor:[UIColor greenColor]]; // Change imageview image color to green.
[imageView setTintColor:[UIColor redColor]]; // Change imageview image color to red.
```

Изменить цвет UIImage

Swift Добавьте это расширение в UIImage:

```
extension UIImage {
    func maskWithColor(color: UIColor) -> UIImage? {

        let maskImage = self.CGImage
        let width = self.size.width
        let height = self.size.height
        let bounds = CGRectMake(0, 0, width, height)

        let colorSpace = CGColorSpaceCreateDeviceRGB()
        let bitmapInfo = CGBitmapInfo(rawValue: CGImageAlphaInfo.PremultipliedLast.rawValue)
        let bitmapContext = CGBitmapContextCreate(nil, Int(width), Int(height), 8, 0,
colorSpace, bitmapInfo.rawValue) //needs rawValue of bitmapInfo

        CGContextClipToMask(bitmapContext, bounds, maskImage)
        CGContextSetFillColorWithColor(bitmapContext, color.CGColor)
        CGContextFillRect(bitmapContext, bounds)

        //is it nil?
        if let cImage = CGBitmapContextCreateImage(bitmapContext) {
            let coloredImage = UIImage(CGImage: cImage)

            return coloredImage

        } else {
            return nil
        }
    }
}
```

Затем, чтобы изменить цвет вашего UIImage

```
my_image.maskWithColor(UIColor.blueColor())
```

Найдено по этой ссылке

Прочитайте UIImage онлайн: https://riptutorial.com/ru/ios/topic/1409/uiimage

глава 81: UIImagePickerController

Вступление

UIImagePickerController предоставляет почти готовое решение, позволяющее пользователю выбирать изображение с их устройства или делать снимок с помощью камеры, а затем представлять это изображение. Согласившись с UIImagePickerControllerDelegate, вы можете создать логику, которая указывает в вашем приложении, как представить изображение и что с ним делать (используя didFinishPickingMediaWithInfo), а также что делать, если пользователь отказывается выбирать изображение или делать снимок (используя imagePickerControllerDidCancel).

Examples

Общее использование UIImagePickerController

Шаг 1. Создайте контроллер, установите делегат и совместите его с протоколом.

```
//Swift
class ImageUploadViewController: UIViewController, UIImagePickerControllerDelegate,
UINavigationControllerDelegate {

    let imagePickerController = UIImagePickerController()

    override func viewDidLoad() {
        super.viewDidLoad()
        imagePickerController.delegate = self
    }
}

//Objective-C
@interface ImageUploadViewController : UIViewController
<UIImagePickerControllerDelegate,UINavigationControllerDelegate> {

    UIImagePickerController *imagePickerController;

}

@end

@implementation ImageUploadViewController

- (void)viewDidLoad {

    [super viewDidLoad];

    imagePickerController.delegate = self;

}

@end
```

note: Мы фактически ничего не реализуем в `UINavigationControllerDelegate` , но `UIImagePickerController` наследует от `UINavigationController` и изменяет поведение `UINavigationController` . Поэтому нам все равно нужно сказать, что наш контроллер просмотра соответствует `UINavigationControllerDelegate` .

Шаг 2. Когда вам нужно показать `UIImagePickerController` :

```
//Swift
self.imagePickerController.sourceType = .Camera // options: .Camera , .PhotoLibrary ,
.SavedPhotosAlbum
self.presentViewController(self.imagePickerController, animated: true, completion: nil)

//Objective-C
imagePickerController.sourceType = UIImagePickerControllerSourceTypeCamera; // options:
UIImagePickerControllerSourceTypeCamera, UIImagePickerControllerSourceTypePhotoLibrary,
UIImagePickerControllerSourceTypeSavedPhotosAlbum
[self presentViewController:imagePickerController animated:YES completion:nil];
```

Шаг 3: Внедрите методы делегата:

```
//Swift
func imagePickerController(picker: UIImagePickerController, didFinishPickingMediaWithInfo
info: [String : AnyObject]) {
    if let pickedImage = info[UIImagePickerControllerOriginalImage] as? UIImage {
        // Your have pickedImage now, do your logic here
    }
    self.dismissViewControllerAnimated(true, completion: nil)
}

func imagePickerControllerDidCancel(picker: UIImagePickerController) {
    self.dismissViewControllerAnimated(true, completion: nil)
}

//Objective-C
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {

   UIImage *pickedImage = info[UIImagePickerControllerOriginalImage];

    if (pickedImage) {

        //You have pickedImage now, do your logic here

    }

    [self dismissViewControllerAnimated:YES completion:nil];

}

- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker {

    [self dismissViewControllerAnimated:YES completion:nil];

}
```

Прочитайте UIImagePickerController онлайн:

https://riptutorial.com/ru/ios/topic/3023/uiimagepickercontroller

глава 82: UIImageView

Examples

Создайте UIImageView

Чтобы программно создать `UIImageView` , все, что вам нужно сделать, это создать экземпляр `UIImageView` :

```
//Swift
let imageView = UIImageView()

//Objective-C
UIImageView *imageView = [[UIImageView alloc] init];
```

Вы можете установить размер и положение `UIImageView` с помощью `CGRect` :

```
//Swift
imageView.frame = CGRect(x: 0, y: 0, width: 200, height: 200)

//Objective-C
imageView.frame = CGRectMake(0,0,200,200);
```

Или вы можете установить размер во время инициализации:

```
//Swift
UIImageView(frame: CGRect(x: 0, y: 0, width: 200, height: 200))

//Objective-C
UIImageView *imageView = [[UIImageView alloc] initWithFrame:CGRectMake(0,0,200,200);

//Alternative way of defining frame for UIImageView
UIImageView *imageView = [[UIImageView alloc] init];
CGRect imageViewFrame = imageView.frame;
imageViewFrame.size.width = 200;
imageViewFrame.size.height = 200;
imageViewFrame.origin.x = 0;
imageViewFrame.origin.y = 0;
imageView.frame = imageViewFrame;
```

> Примечание. Вы должны импортировать `UIKit` для использования `UIImageView` .

Назначение изображения в UIImageView

Вы можете назначить изображение в `UIImageView` во время инициализации или позже с помощью свойства `image` :

```
//Swift
UIImageView(image: UIImage(named: "image1"))
```

```
UIImageView(image: UIImage(named: "image1"), highlightedImage: UIImage(named: "image2"))

imageView.image = UIImage(named: "image1")

//Objective-C
[[UIImageView alloc] initWithImage:[UIImage imageNamed:@"image1"];

[[UIImageView alloc] initWithImage:[UIImage imageNamed:@"image1"] highlightedImage:[UIImage
imageNamed:@"image2"]];

imageView.image = [UIImage imageNamed:@"image1"];
```

Анимация UIImageView

Вы можете анимировать `UIImageView` , быстро отображая изображения на нем в последовательности, используя свойства анимации `UIImageView` :

```
imageView.animationImages = [UIImage(named: "image1")!,
                             UIImage(named: "image2")!,
                             UIImage(named: "image3")!,
                             UIImage(named: "image4")!,
                             UIImage(named: "image5")!,
                             UIImage(named: "image6")!,
                             UIImage(named: "image7")!,
                             UIImage(named: "image8")!]
imageView.animationDuration = 0.3
imageView.animationRepeatCount = 1
```

Свойством `animationImages` является `Array` `UIImages` который запускается сверху вниз, когда анимация запускается.

Свойство `animationDuration` - это `Double` высказывание, сколько секунд будет выполняться анимация.

Свойство `animationRepeatCount` - это `Int` который говорит, сколько раз анимация будет запущена.

Чтобы запустить и остановить анимацию, вы можете вызвать соответствующие методы для этого:

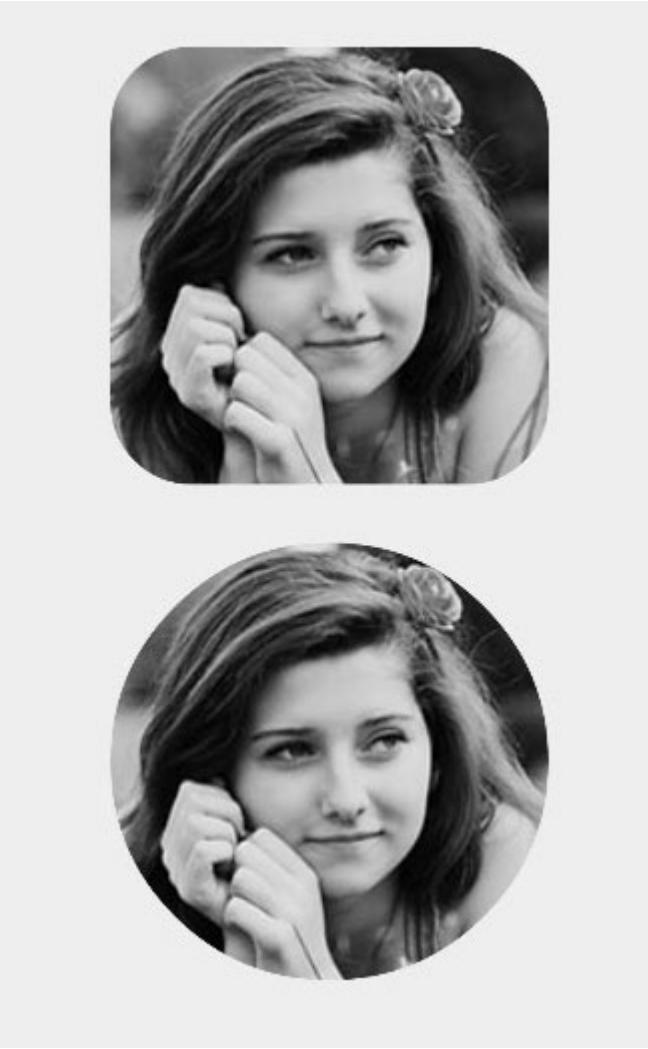
```
imageView.startAnimating()
imageView.stopAnimating()
```

Существует метод `isAnimating()` который возвращает `Boolean` значение, указывающее, работает ли анимация в данный момент или нет.

Обратите внимание, что это не очень эффективный способ создания анимаций: он довольно медленный и ресурсоемкий. Подумайте, используя слои или спрайты для получения лучших результатов

Создание изображения в круг или закругление

В этом примере показано, как сделать `UIView` или `UIImageView` округленным с некоторым радиусом следующим образом:



Objective-C

```
someImageView.layer.cornerRadius = CGRectGetHeight(someImageView.frame) / 2;
someImageView.clipsToBounds = YES;
```

стриж

```
someImageView.layer.cornerRadius = someImageView.frame.height/2
// this should alleviate the performance hit that adding transparency may cause - see
http://stackoverflow.com/a/6254531/189804
// Be sure to check scrolling performance with Instruments if you take this approach.
someImageView.layer.shouldRasterize = true
someImageView.clipsToBounds = true // All parts of the image that are outside its bounds (the
frame) are cut out (makes the rounded corners visible)
```

Предполагается, что если вы используете автозапуск, то вы `someImageView.layer.cornerRadius`

код `viewDidLayoutSubviews` в `viewDidLayoutSubviews` . Это позволит обновить изображение `cornerRadius` если изображение изменит размер.

```
override func viewDidLayoutSubviews() {
    super.viewDidLayoutSubviews()
    someImageView.layer.cornerRadius = someImageView.frame.size.width/2
    someImageView.layer.masksToBounds = true
}
```

UIImage, замаскированный ярлыком

Это делает изображение замаскированным в форме букв метки:

Objective-C

```
self.maskImage.layer.mask = self.maskLabel.layer;
self.maskImage.layer.masksToBounds = YES;
```

Swift 3

```
maskImageView.mask = maskLabel
maskImageView.masksToBounds = true
```

Вот результат:

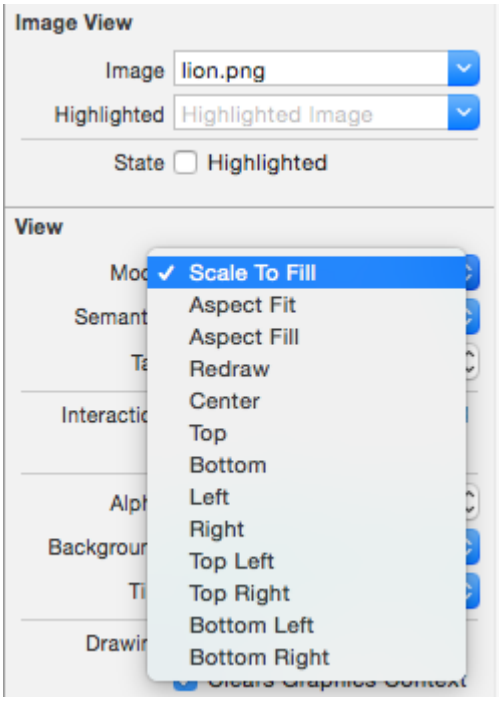
Изменение цвета изображения

```
//Swift
imageView.tintColor = UIColor.redColor()
imageView.image = imageView.image?.imageWithRenderingMode(.AlwaysTemplate)

//Swift 3
imageView.tintColor = UIColor.red
imageView.image = imageView.image?.withRenderingMode(.alwaysTemplate)

//Objective-C
imageView.tintColor = [UIColor redColor];
imageView.image = [imageView.image imageWithRenderingMode:UIImageRenderingModeAlwaysTemplate]
```

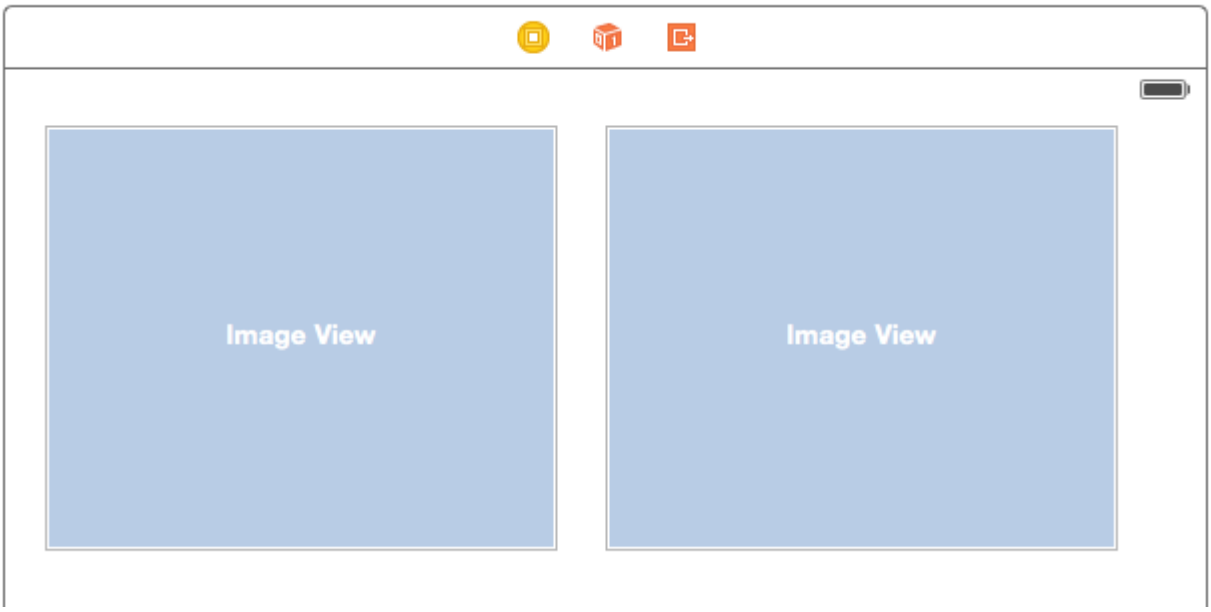
Как свойство Mode влияет на изображение

Свойство режима содержимого представления указывает, как его содержимое должно

быть выложено. В Interface Builder в Инспекторе атрибутов могут быть выбраны различные режимы.



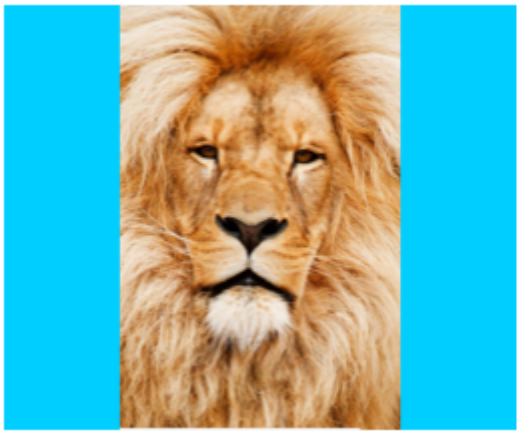
Давайте посмотрим, как работают различные режимы.



Масштаб для заполнения

Высота и ширина изображения растягиваются в соответствии с размером `UIImageView` .

Aspect Fit



Самая длинная сторона (высота или ширина) изображения растягивается в соответствии с представлением. Это делает изображение настолько большим, насколько возможно, пока оно покажет все изображение, а не искажает высоту или ширину. (Я установил фон `UIImageView` на синий, чтобы его размер был ясным.)

Аспект заполнения


Самая короткая сторона (высота или ширина) изображения растягивается в соответствии с представлением. Как и «Aspect Fit», пропорции изображения не искажаются от их исходного соотношения сторон.

Перерисовка

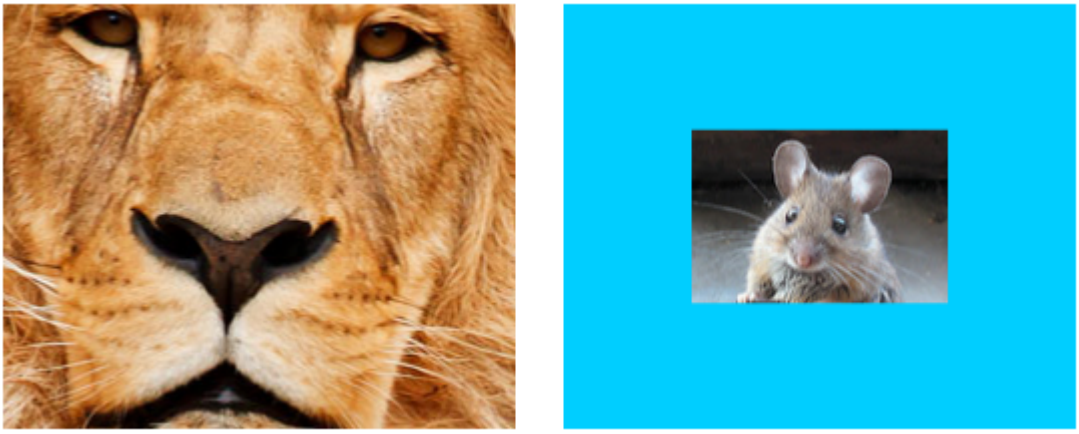

Перерисовать только для пользовательских представлений, которые должны выполнять собственное масштабирование и изменение размера. Мы не используем пользовательский вид, поэтому нам не следует использовать Redraw. Обратите внимание: здесь `UIImageView` просто дает нам тот же результат, что и Scale to Fill, но он делает больше работы за кулисами.

О Redraw, в документации Apple говорится:

> Режимы содержимого хороши для повторного использования содержимого вашего представления, но вы также можете настроить режим содержимого на значение `UIViewContentModeRedraw` когда вы специально хотите, чтобы ваши пользовательские представления перерисовывались во время операций масштабирования и изменения размера. Настройка режима контента вашего представления на это значение заставляет систему вызывать метод `drawRect:` ваш вид в ответ на изменения геометрии. В общем, вам следует избегать

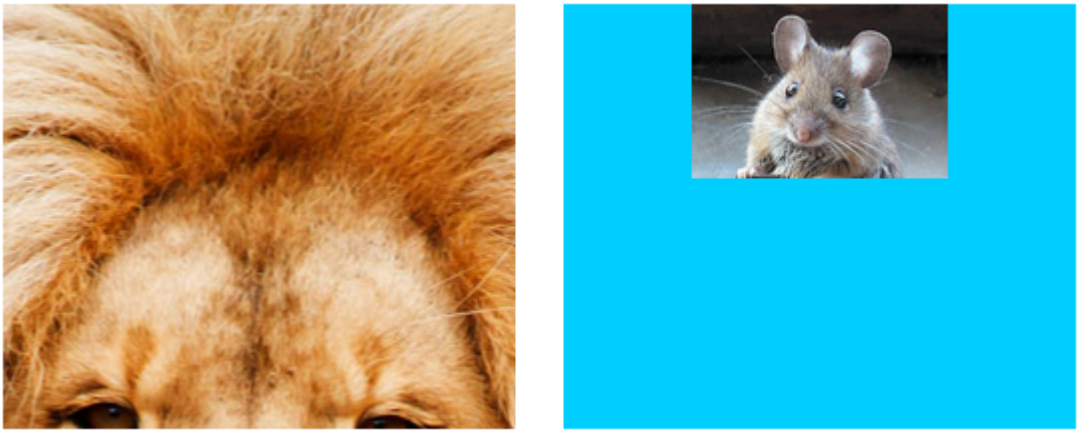
использования этого значения, когда это возможно, и вы, безусловно, не должны использовать его со стандартными представлениями системы.

Центр



Изображение сосредоточено в представлении, но длина и ширина изображения не растянуты.

верхний



Верхний край изображения центрируется по горизонтали в верхней части изображения, а длина и ширина изображения не растянуты.

Низ

Нижний край изображения центрируется горизонтально в нижней части изображения, а длина и ширина изображения не растянуты.

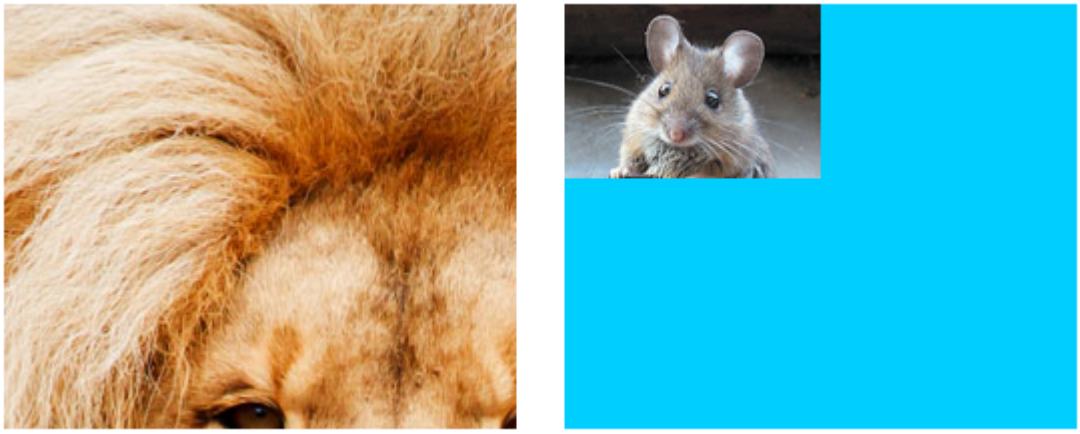
Оставил

Левый край изображения центрируется по вертикали слева от представления, а длина и ширина изображения не растянуты.

Правильно

Правый край изображения центрирован по вертикали справа от представления, а длина и ширина изображения не растянуты.

Верхний левый



Верхний левый угол изображения расположен в верхнем левом углу окна. Длина и ширина изображения не растянуты.

В правом верхнем углу

Верхний правый угол изображения расположен в верхнем правом углу окна. Длина и ширина изображения не растянуты.

Нижний левый

Нижний левый угол изображения расположен в левом нижнем углу окна. Длина и ширина изображения не растянуты.

Нижний правый

Нижний правый угол изображения расположен в правом нижнем углу окна. Длина и ширина изображения не растянуты.

Заметки

- Этот пример исходит отсюда .
- Если содержимое (в нашем случае изображения) имеет тот же размер, что и представление (в нашем случае `UIImageView`), то изменение режима содержимого не будет иметь заметной разницы.
- См. Этот и этот вопрос для обсуждения режимов контента для представлений, отличных от `UIImageView` .
- В Swift, чтобы программно настроить режим содержимого, вы делаете следующее:

```
imageView.contentMode = UIViewContentMode.scaleToFill
imageView.contentMode = UIViewContentMode.scaleAspectFit
imageView.contentMode = UIViewContentMode.scaleAspectFill
imageView.contentMode = UIViewContentMode.redraw
imageView.contentMode = UIViewContentMode.center
imageView.contentMode = UIViewContentMode.top
imageView.contentMode = UIViewContentMode.bottom
imageView.contentMode = UIViewContentMode.left
imageView.contentMode = UIViewContentMode.right
imageView.contentMode = UIViewContentMode.topLeft
imageView.contentMode = UIViewContentMode.topRight
imageView.contentMode = UIViewContentMode.bottomLeft
imageView.contentMode = UIViewContentMode.bottomRight
```

Прочитайте UIImageView онлайн: https://riptutorial.com/ru/ios/topic/695/uiimageview

глава 83: UILabel

Вступление

Класс UILabel реализует текстовое представление только для чтения. Вы можете использовать этот класс для рисования одной или нескольких строк статического текста, например тех, которые вы можете использовать для идентификации других частей вашего пользовательского интерфейса. Базовый класс UILabel обеспечивает поддержку как простого, так и сложного стиля текста ярлыка. Вы также можете контролировать аспекты внешнего вида, например, использует ли метка тень или рисует с подсветкой. При необходимости вы можете настроить внешний вид своего текста, добавив подклассом.

Синтаксис

- UILabel.numberOfLines: Int // получить или установить максимальное количество строк, которые может иметь метка. 0 неограниченно
- UILabel.text: String? // получить или установить текст, отображаемый ярлыком
- UILabel.textColor: UIColor! // получить или установить цвет текста на ярлыке
- UILabel.tintColor: UIColor! // получить или установить цвет оттенка метки
- UILabel.attributedText: NSAttributedString? // получить или установить атрибутный текст метки
- UILabel.font: UIFont! // получить или установить шрифт текста на ярлыке
- UILabel.textAlignment: NSTextAlignment // получить или установить выравнивание текста

замечания

UILabels - это виды, которые могут использоваться для отображения одной или нескольких строк текста. Он содержит несколько способов стилизации текста, например, тени, цвета текста и шрифты.

UILabels также могут отображать Attributed Strings, который представляет собой текстовую + встроенную разметку для применения стилей к частям текста.

UILabel не соответствует протоколу UIAppearance, поэтому вы не можете использовать методы прокси-сервера UIAppearance для настройки внешнего вида UILabels. См. Обсуждение для большего.

Ссылка Apple Developer здесь

Examples

Изменение текста в существующей этикетке

Изменение текста существующей `UILabel` может быть выполнено путем доступа и изменения `text` свойства `UILabel` . Это можно сделать напрямую, используя `String` литералы или косвенно используя переменные.

Установка текста со `String` литералами

стриж

```
label.text = "the new text"
```

Objective-C

```
// Dot Notation
label.text = @"the new text";

// Message Pattern
[label setText:@"the new text"];
```

Установка текста с переменной

стриж

```
let stringVar = "basic String var"
label.text = stringVar
```

Objective-C

```
NSString * stringVar = @"basic String var";

// Dot Notation
label.text = stringVar;

// Message Pattern
[label setText: stringVar];
```

Цвет текста

Вы можете использовать свойство `textColor` метки, чтобы применить цвет текста ко всему тексту метки.

стриж

```
label.textColor = UIColor.redColor()
```

```
label.textColor = UIColor(red: 64.0/255.0, green: 88.0/255.0, blue: 41.0/225.0, alpha: 1)
```

Swift 3

```
label.textColor = UIColor.red
label.textColor = UIColor(red: 64.0/255.0, green: 88.0/255.0, blue: 41.0/225.0, alpha: 1)
```

Objective-C

```
label.textColor = [UIColor redColor];
label.textColor = [UIColor colorWithRed:64.0f/255.0f green:88.0f/255.0f blue:41.0f/255.0f
alpha:1.0f];
```

Применение цвета текста к части текста

Вы также можете изменять цвет текста (или другие атрибуты) частей текста с помощью `NSAttributedString` :

Objective-C

```
attributedString = [[NSMutableAttributedString alloc] initWithString:@"The grass is green; the
sky is blue."];
[attributedString addAttribute: NSForegroundColorAttributeName value:[UIColor greenColor]
range:NSMakeRange(13, 5)];
[attributedString addAttribute: NSForegroundColorAttributeName value:[UIColor blueColor]
range:NSMakeRange(31, 4)];
label.attributedText = attributesString;
```

стриж

```
let attributedString = NSMutableAttributedString(string: "The grass is green; the sky is
blue.")
attributedString.addAttribute(NSForegroundColorAttributeName, value: UIColor.green(), range:
NSRange(location: 13, length: 5))
attributedString.addAttribute(NSForegroundColorAttributeName, value: UIColor.blue(), range:
NSRange(location: 31, length: 4))
label.attributedText = attributedString
```

Выравнивание текста

стриж

```
label.textAlignment = NSTextAlignment.left
//or the shorter
label.textAlignment = .left
```

Любое значение в перечислении `NSTextAlignment` действительно: `.left` , `.center` , `.right` , `.justified .natural`

Objective-C

```
label.textAlignment = NSTextAlignmentLeft;
```

Любое значение в перечислении `NSTextAlignment` действительно: `NSTextAlignmentLeft` , `NSTextAlignmentCenter` , `NSTextAlignmentRight` , `NSTextAlignmentJustified` , `NSTextAlignmentNatural`

Вертикальное выравнивание в `UILabel` не поддерживается из коробки: вертикально выравнивать текст вверху в UILabel

Создание UILabel

С рамкой

Когда вы знаете точные размеры, которые вы хотите установить для своей метки, вы можете инициализировать `UILabel` с помощью рамки `CGRect` .

стриж

```
let frame = CGRect(x: 0, y: 0, width: 200, height: 21)
let label = UILabel(frame: frame)
view.addSubview(label)
```

Objective-C

```
CGRect frame = CGRectMake(0, 0, 200, 21);
UILabel *label = [[UILabel alloc] initWithFrame:frame];
[view addSubview:label];
```

С автоматической компоновкой

Вы можете добавить ограничения на `UILabel` если хотите, чтобы iOS динамически вычислял свой фрейм во время выполнения.

стриж

```
let label = UILabel()
label.backgroundColor = .red
label.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(label)

NSLayoutConstraint.activate([
    //stick the top of the label to the top of its superview:
```

```
    label.topAnchor.constraint(equalTo: view.topAnchor)

    //stick the left of the label to the left of its superview
    //if the alphabet is left-to-right, or to the right of its
    //superview if the alphabet is right-to-left:
    label.leadingAnchor.constraint(equalTo: view.leadingAnchor)

    //stick the label's bottom to the bottom of its superview:
    label.bottomAnchor.constraint(equalTo: view.bottomAnchor)

    //the label's width should be equal to 100 points:
    label.widthAnchor.constraint(equalToConstant: 100)
])
```

Objective-C

```
UILabel *label = [[UILabel alloc] init];
```

C Objective-c + Visual Format Language (VFL)

```
  UILabel *label = [UILabel new];
  label.translatesAutoresizingMaskIntoConstraints = NO;
  [self.view addSubview label];
  // add horizontal constraints with 5 left and right padding from the leading and trailing

  [self.view addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"V:|-5-
[labelName]-5-|"
                                                                     options:0
                                                                     metrics:nil


views:@{@"labelName":label}]];
  // vertical constraints that will use the height of the superView with no padding on top and
bottom
  [self.view addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"H:|[labelName]|"
                                                                     options:0
                                                                     metrics:nil


views:@{@"labelName":label}]]
```

Документацию VFL можно найти здесь

После того, как метка была создана, обязательно задайте размеры с помощью автоматической компоновки. Xcode будет отображать ошибки, если это сделано неправильно.

С интерфейсом Builder

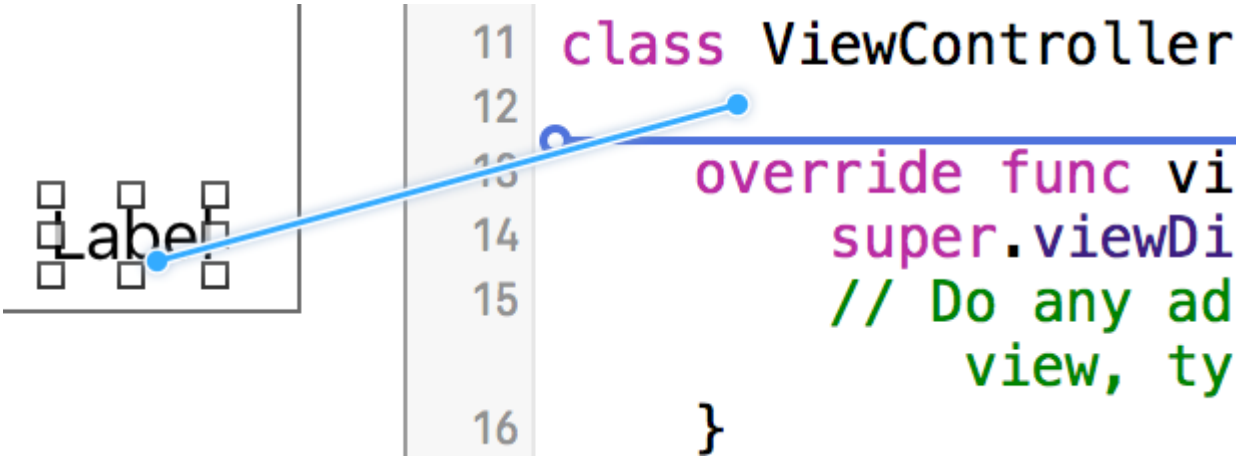
Вы также используете Interface Builder, чтобы добавить `UILabel` в свой файл `Storyboard` или `.xib` , перетащив `Label` с панели библиотеки объектов и отбросив его в виде на холсте:

Вместо того, чтобы программно определять кадр (положение и размер) для `UILabel` , `Storyboard` или `.xib` позволяет использовать Auto Layout для добавления ограничений в элемент управления.

Чтобы получить доступ к этому ярлыку, созданному из `storyboard` или `xib` создайте IBOutlet этого ярлыка.

Связывание между построителем интерфейса и контроллером View

После того, как вы добавили `UILabel` в свою `Storyboard` или файл `.xib` вы можете связать ее с вашим кодом, нажав `Control ^` а затем перетащив мышь между `UILabel` на ваш `ViewController` или вы можете перетащить ее в код, щелкнув правой кнопкой мыши имеют такой же эффект.



В диалоговом окне свойств вы можете установить имя `UILabel` и установить его как `strong` или `weak` . Для получения дополнительной информации о `strong` и `weak` , см. Это ,

Другой способ - сделать выход программным путем следующим образом:

стриж

```
@IBOutlet weak var nameLabel : UILabel!
```

Objective-C

```
@property (nonatomic, weak) IBOutlet UILabel *nameLabel;
```

Установить шрифт

стриж

```
let label = UILabel()
```

Objective-C

```
UILabel *label = [[UILabel alloc] init];
or
UILabel *label = [UILabel new]; // convenience method for calling alloc-init
```

Изменение размера шрифта по умолчанию

стриж

```
label.font = UIFont.systemFontOfSize(17)
```

Swift 3

```
label.font = UIFont.systemFont(ofSize: 17)
```

Objective-C

```
label.font = [UIFont systemFontOfSize:17];
```

Используйте определенный вес шрифта

iOS 8.2

стриж

```
label.font = UIFont.systemFontOfSize(17, weight: UIFontWeightBold)
```

Swift3

```
label.font = UIFont.systemFont(ofSize: 17, weight: UIFontWeightBold)
```

Objective-C

```
label.font = [UIFont systemFontOfSize:17 weight:UIFontWeightBold];
```

iOS 8.2

стриж

```
label.font = UIFont.boldSystemFontOfSize(17)
```

Swift3

```
label.font = UIFont.boldSystemFont(ofSize: 17)
```

Objective-C

```
label.font = [UIFont boldSystemFontOfSize:17];
```

Используйте текстовый стиль динамического типа.

Размер шрифта и точки будет основываться на предпочтительном размере чтения пользователя.

стриж

```
label.font = UIFont.preferredFontForTextStyle(UIFontTextStyleBody)
```

Swift 3

```
label.font = UIFont.preferredFont(forTextStyle: .body)
```

Objective-C

```
label.font = [UIFont preferredFontForTextStyle:UIFontTextStyleBody];
```

Использовать другой шрифт

стриж

```
label.font = UIFont(name: "Avenir", size: 15)
```

Objective-C

```
label.font = [UIFont fontWithName:@"Avenir" size:15];
```

Изменить размер шрифта

Способ установки размера шрифта, не зная семейства шрифтов, заключается в использовании свойства **шрифта** `UILabel` .

стриж

```
label.font = label.font.fontWithSize(15)
```

Swift 3

```
label.font = label.font.withSize(15)
```

Objective-C

```
label.font = [label.font fontWithSize:15];
```

Использовать пользовательский шрифт Swift

См. Эту ссылку

Количество строк

Когда вы создаете ярлык и устанавливаете его текст более чем одной строки, которую он может отобразить, он будет усечен, и вы увидите только одну строку текста, заканчивающуюся тремя точками (...). Это связано с тем, что свойство с именем `numberOfLines` установлено в 1, и поэтому отображается только одна строка. Это распространенная ошибка при обработке `UILabel` s, и многие люди думают об этом как об ошибке, или могут использовать несколько ярлыков, чтобы показать больше, чем текст, но просто отредактировав это свойство, мы можем сказать, что `UILabel` принять до указанного количества строк. Например, если для этого свойства установлено значение 5, метка может отображать 1, 2, 3, 4 или 5 строк данных.

Установка значения программно

Чтобы установить это свойство, просто присвойте ему новое целое число:

стриж

```
label.numberOfLines = 2
```

Objective-C

```
label.numberOfLines = 2;
```

Заметка

Это свойство можно установить равным 0. Однако это не означает, что он не примет никаких строк, а означает, что метка может иметь столько строк, сколько необходимо (ака «Бесконечность»):

стриж

```
label.numberOfLines = 0
```

Objective-C

```
label.numberOfLines = 0;
```

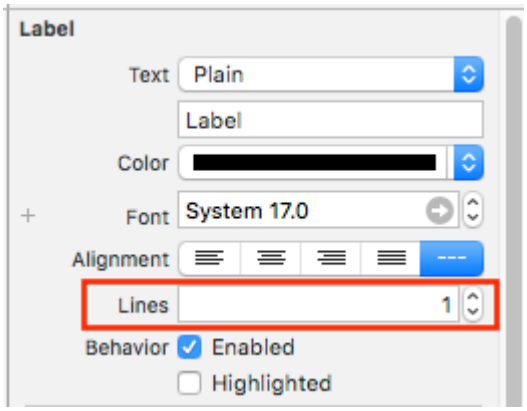
> # Заметка
>
> Если метка имеет ограничение по высоте, ограничение будет соблюдаться. В этом случае `label.numberOfLines = 0` может работать не так, как ожидалось.

> # Заметка
>
> Для более сложного многострочного текста UITextView может быть лучше подходит. *

Установка значения в построителе интерфейса

Вместо программного программирования `numberOfLines` вы можете использовать `Storyboard` или `.xib` и установить свойство `numberOfLines` . Таким образом, мы достигаем тех же результатов, что и вышеуказанный код.

Как показано ниже:



Размер по размеру

Предположим, у вас есть `UILabel` на вашем `storyboard` и вы создали `IBOutlet` для него в `ViewController.swift` / `ViewController.m` и назвали его `labelOne` .

Чтобы изменения были легко видны, измените `backgroundColor` и `textColor` меткиOne в методе `viewDidLoad` :

Функция `sizeToFit` используется, когда вы хотите автоматически изменять размер метки на основе содержимого, хранящегося в нем.

стриж

```
labelOne.backgroundColor = UIColor.blueColor()
labelOne.textColor = UIColor.whiteColor()
labelOne.text = "Hello, World!"
labelOne.sizeToFit()
```

Swift 3

```
labelOne.backgroundColor = UIColor.blue
labelOne.textColor = UIColor.white
labelOne.text = "Hello, World!"
labelOne.sizeToFit()
```

Objective-C

```
labelOne.backgroundColor = [UIColor blueColor];
labelOne.textColor = [UIColor whiteColor];
labelOne.text = @"Hello, World!";
[labelOne sizeToFit];
```

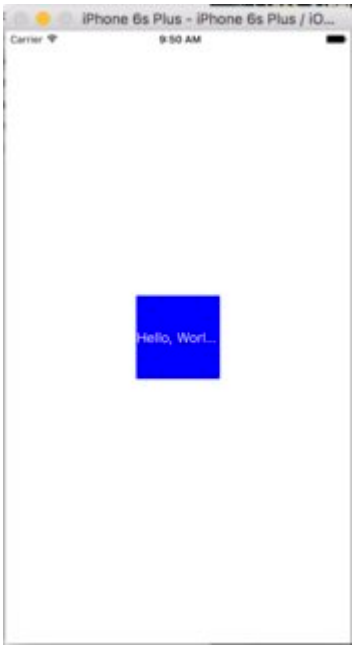
Вывод для вышеуказанного кода:

Как вы можете видеть, никаких изменений нет, поскольку текст отлично подходит для labelOne. sizeToFit только изменяет рамку метки.

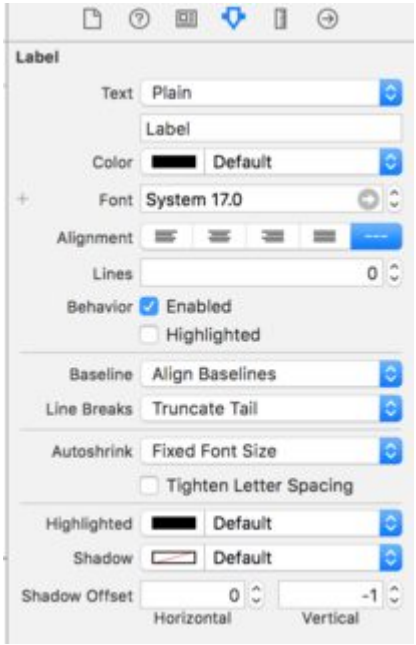
Давайте изменим текст на несколько более длинный:

```
labelOne.text = "Hello, World! I’m glad to be alive!"
```

Теперь labelOne выглядит так:



Даже вызов `sizeToFit` ничего не меняет. Это происходит из-за того, что по умолчанию числовое число, отображаемое UILabel, равно 1. Давайте изменим его на ноль на раскадровке:



На этот раз, когда мы запустим приложение, labelOne появится правильно:

Свойство `numberOfLines` также может быть изменено в файле `ViewController` :

```
// Objective-C
labelOne.numberOfLines = 0;

// Swift
labelOne.numberOfLines = 0
```

Фоновый цвет

стриж

```
label.backgroundColor = UIColor.redColor()

label.backgroundColor = .redColor()
```

Swift 3

```
label.backgroundColor = UIColor.red
```

Objective-C

```
label.backgroundColor = [UIColor redColor];
```

Добавить тени в текст

стриж

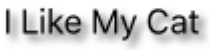
```
label1.layer.shadowOffset = CGSize(width: 3, height: 3)
label1.layer.shadowOpacity = 0.7
label1.layer.shadowRadius = 2
```

Swift 3

```
label1.layer.shadowOffset = CGSize(width: 3, height: 3)
label1.layer.shadowOpacity = 0.7
label1.layer.shadowRadius = 2
```

Objective-C

```
label1.layer.shadowOffset = CGSizeMake(3, 3);
label1.layer.shadowOpacity = 0.7;
label1.layer.shadowRadius = 2;
```



Переменная высота с использованием ограничений

Вы можете создать `UILabel` с динамической высотой, используя автоматическую компоновку.

Вам нужно установить `numberOfLines` в ноль (0) и добавить минимальную высоту, установив ограничения с отношением типа `.GreaterThanOrEqual` в `.Height`

iOS 6

стриж

```
label.numberOfLines = 0

let heightConstraint = NSLayoutConstraint(
    item: label,
    attribute: .Height,
    relatedBy: .GreaterThanOrEqual,
    toItem: nil,
    attribute: .NotAnAttribute,
    multiplier: 0,
    constant: 20
)

label.addConstraint(heightConstraint)
```

iOS 9

стриж

```
label.numberOfLines = 0
```

```
label.translatesAutoresizingMaskIntoConstraints = false
label.heightAnchor.constraintGreaterThanOrEqualToConstant(20).active = true
```

LineBreakMode

Использование кода

```
UILabel.lineBreakMode: NSLineBreakMode
```

стриж

```
label.lineBreakMode = .ByTruncatingTail
```

- .ByWordWrapping
- .ByCharWrapping
- .ByClipping
- .ByTruncatingHead
- .ByTruncatingTail
- .ByTruncatingMiddle

Swift 3

```
label.lineBreakMode = .byTruncatingTail
```

- .byWordWrapping
- .byCharWrapping
- .byClipping
- .byTruncatingHead
- .byTruncatingTail
- .byTruncatingMiddle

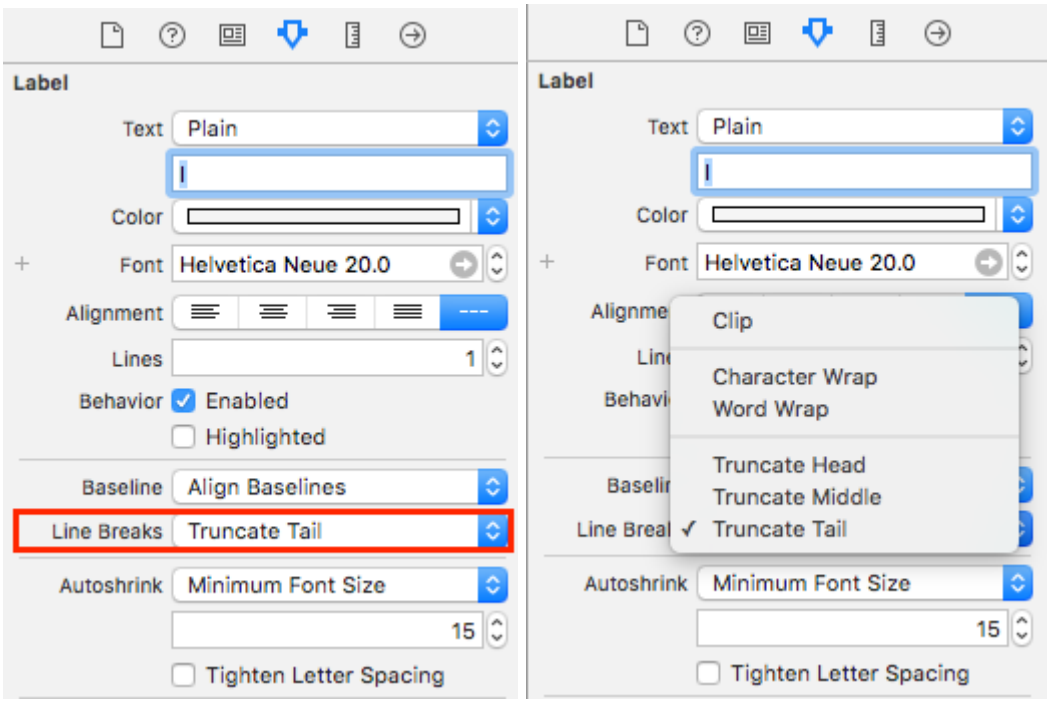
Objective-C

```
[label setLineBreakMode:NSLineBreakByTruncatingTail];
```

- NSLineBreakByWordWrapping
- NSLineBreakByCharWrapping
- NSLineBreakByClipping
- NSLineBreakByTruncatingHead
- NSLineBreakByTruncatingTail
- NSLineBreakByTruncatingMiddle

Использование раскадровки

Это также можно установить в инспекторе атрибутов UILabel:



Константы

- Word Wrapping - обертывание происходит на границах слов, если только само слово не подходит для одной строки
- Char Wrapping - обертывание происходит до первого символа, который не подходит
- Линии отсечения просто не вытягиваются за край текстового контейнера
- Усекающая головка - линия отображается так, что конец подходит в контейнере, а недостающий текст в начале строки обозначается эллипсовым символом
- Усечение хвоста - строка отображается так, что начало вставляется в контейнер, а отсутствующий текст в конце строки обозначается эллипсовым символом
- Усечение среднего - линия отображается так, что начало и конец вписываются в контейнер, а отсутствующий текст в середине обозначается эллипсовым символом

Вычислить границы содержимого (например, динамические высоты ячеек)

Обычный пример использования для вычисления фрейма, на который будет выполняться метка, - это правильно подобрать ячейки представления таблицы. Рекомендуемый способ использования этого метода - использовать метод `NSString boundingRectWithSize:options:attributes:context:`

`options`

принимают параметры рисования строк:

- `NSStringDrawingUsesLineFragmentOrigin` следует использовать для меток с несколькими строками
- `NSStringDrawingTruncatesLastVisibleLine` следует добавить с помощью `|` оператора, если имеется максимальное количество строк

`attributes` является `NSDictionary` атрибутов, которые влияют на атрибуты строк (полный список: Apple Docs), но факторы, влияющие на высоту, включают:

- **NSFontAttributeName** : очень важно, размер и семейство шрифтов являются важной частью отображаемого размера метки.
- **NSParagraphStyleAttributeName** : для настройки отображения текста. Это включает межстрочный интервал, выравнивание текста, стиль усечения и несколько других опций. Если вы явно не изменили ни одно из этих значений, вам не нужно слишком беспокоиться об этом, но может быть важно, если вы переключили некоторые значения на IB.

`context` должен быть равен `nil` поскольку основной случай использования `NSStringDrawingContext` заключается в том, чтобы разрешить размер шрифта, чтобы он соответствовал указанному прямоугольнику, что не должно быть, если мы вычисляем динамическую высоту.

Цель C

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{

    UITableViewCell *cell = [tableView cellForRowAtIndexPath:indexPath];

    NSString *labelContent = cell.theLabel.text;
    // you may choose to get the content directly from the data source if you have done
minimal customizations to the font or are comfortable with hardcoding a few values
//    NSString *labelContent = [self.dataSource objectAtIndexPath:indexPath];

    // value may be hardcoded if retrieved from data source
    NSFont *labelFont = [cell.theLabel font];

    // The NSParagraphStyle, even if you did not code any changes these values may have been
altered in IB
    NSMutableParagraphStyle *paragraphStyle = [NSMutableParagraphStyle new];
    paragraphStyle.lineBreakMode = NSLineBreakByWordWrapping;
    paragraphStyle.alignment = NSTextAlignmentCenter;

    NSDictionary *attributes = @{NSFontAttributeName: labelFont,
                                 NSParagraphStyleAttributeName: paragraphStyle};

    // The width is also important to the height
    CGFloat labelWidth = CGRectGetWidth(cell.theLabel.frame);
    // If you have been hardcoding up to this point you will be able to get this value by
subtracting the padding on left and right from tableView.bounds.size.width
//    CGFloat labelWidth = CGRectGetWidth(tableView.frame) - 20.0f - 20.0f;
```

```
    CGRect bodyBounds = [labelContent boundingRectWithSize:CGSizeMake(width, CGFLOAT_MAX)
options:NSStringDrawingUsesLineFragmentOrigin attributes:attributes context:nil];

    return CGRectGetHeight(bodyBounds) + heightOfObjectsOnTopOfLabel +
heightOfObjectBelowLabel;
}
```

Swfit 3

```
override func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) ->
CGFloat {
    var cell = tableView.cellForRow(atIndexPath: indexPath)!
    var labelContent = cell.theLabel.text
    var labelFont = cell.theLabel.font
    var paragraphStyle = NSMutableParagraphStyle()

    paragraphStyle.lineBreakMode = .byWordWrapping
    paragraphStyle.alignment = .center

    var attributes = [NSFontAttributeName: labelFont, NSParagraphStyleAttributeName:
paragraphStyle]

    var labelWidth: CGFloat = cell.theLabel.frame.width

    var bodyBounds = labelContent.boundingRect(withSize: CGSize(width: width, height:
CGFLOAT_MAX), options: .usesLineFragmentOrigin, attributes: attributes, context: nil)

    return bodyBounds.height + heightOfObjectsOnTopOfLabel + heightOfObjectBelowLabel
}
```

И наоборот, если у вас есть установленное максимальное количество строк, вам сначала нужно вычислить высоту одной строки, чтобы убедиться, что мы не получаем значение выше допустимого размера:

```
    // We calculate the height of a line by omitting the NSStringDrawingUsesLineFragmentOrigin
option, which will assume an infinitely wide label
    CGRect singleLineRect = [labelContent boundingRectWithSize:CGSizeMake(CGFLOAT_MAX,
CGFLOAT_MAX)

options:NSStringDrawingTruncatesLastVisibleLine
                                                context:nil];
    CGFloat lineHeight = CGRectGetHeight(singleLineRect);
    CGFloat maxHeight = lineHeight * cell.theLabel.numberOfLines;

    // Now you can call the method appropriately
    CGRect bodyBounds = [labelContent boundingRectWithSize:CGSizeMake(width, maxHeight)
options:(NSStringDrawingUsesLineFragmentOrigin|NSStringDrawingTruncatesLastVisibleLine)
attributes:attributes context:nil];

    return CGRectGetHeight(bodyBounds) + heightOfObjectsOnTopOfLabel +
heightOfObjectBelowLabel;
```

Настраиваемая метка

ПРИМЕЧАНИЕ. В большинстве случаев лучше использовать `UIButton` вместо

создания `UILabel` вы можете нажать. Используйте этот пример, если вы уверены, что по какой-то причине вы не хотите использовать `UIButton` .

1. Создать ярлык
2. Включить взаимодействие с пользователем
3. Добавить `UITapGestureRecognizer`

Ключом к созданию интерактивной `UILabel` является возможность взаимодействия с пользователем.

стриж

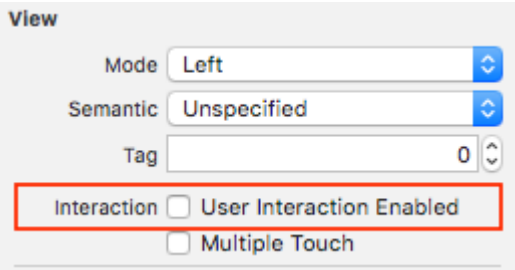
```
let label = UILabel()
label.userInteractionEnabled = true

let gesture = UITapGestureRecognizer(target: self, action: #selector(labelClicked(_:)))
label.addGestureRecognizer(gesture)
```

Objective-C

```
UILabel *label = [[UILabel alloc] init];
[label setUserInteractionEnabled:YES];

UITapGestureRecognizer* gesture = [[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(labelClicked:)];
[label addGestureRecognizer:gesture];
```

Установка «userInteractionEnabled» в инспекторе атрибутов раскадровки

Вместо использования кода вы можете выбрать UILabel внутри раскадровки и проверить опцию:



Динамическая рамка ярлыка с неизвестной длиной текста

Иногда нам приходится изменять размер UILabel на основе динамического содержимого,

где длина текста неизвестна. В этом примере ширина UILabel фиксируется в 280 точках, а высота бесконечна, скажем, 9999. Оцените кадр относительно стиля текста и maximumLabelSize.

Objective-C

```
UILabel * label = [[UILabel alloc] init];

NSString *message = @"Some dynamic text for label";

//set the text and style if any.
label.text = message;

label.numberOfLines = 0;

CGSize maximumLabelSize = CGSizeMake(280, 9999); //280:max width of label and 9999-max height
of label.

// use font information from the UILabel to calculate the size
CGSize expectedLabelSize = [label sizeThatFits:maximumLabelSize];

//Deprecated in iOS 7.0
//CGSize expectedLabelSize = [message sizeWithFont:label.font
constrainedToSize:maximumLabelSize lineBreakMode:NSLineBreakByWordWrapping];

// create a frame that is filled with the UILabel frame data
CGRect newFrame = label.frame;

// resizing the frame to calculated size
newFrame.size.height = expectedLabelSize.height;

// put calculated frame into UILabel frame
label.frame = newFrame;
```

стриж

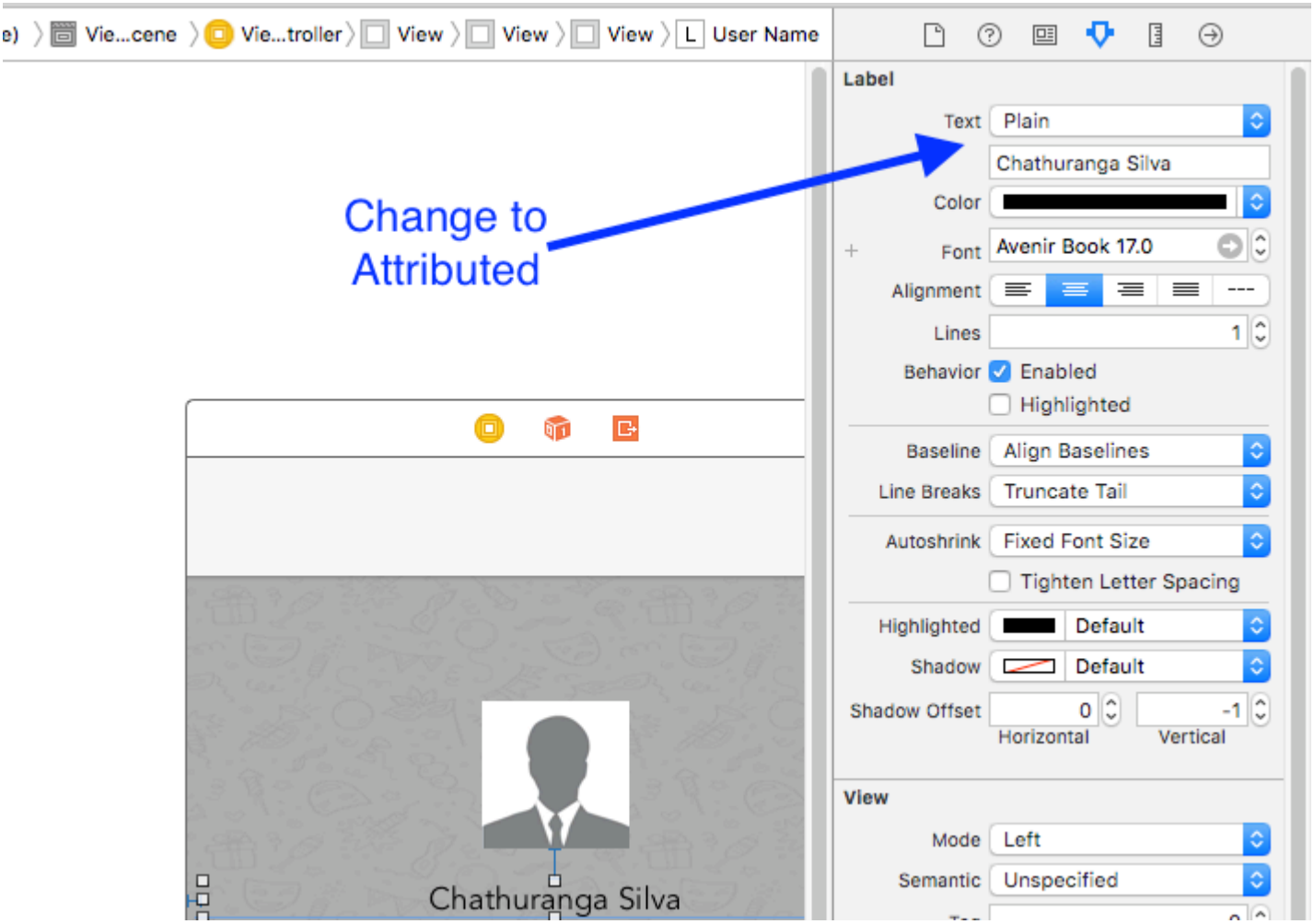
```
var message: String = "Some dynamic text for label"
//set the text and style if any.
label.text = message
label.numberOfLines = 0
var maximumLabelSize: CGSize = CGSize(width: 280, height: 9999)
var expectedLabelSize: CGSize = label.sizeThatFits(maximumLabelSize)
// create a frame that is filled with the UILabel frame data
var newFrame: CGRect = label.frame
// resizing the frame to calculated size
newFrame.size.height = expectedLabelSize.height
// put calculated frame into UILabel frame
label.frame = newFrame
```

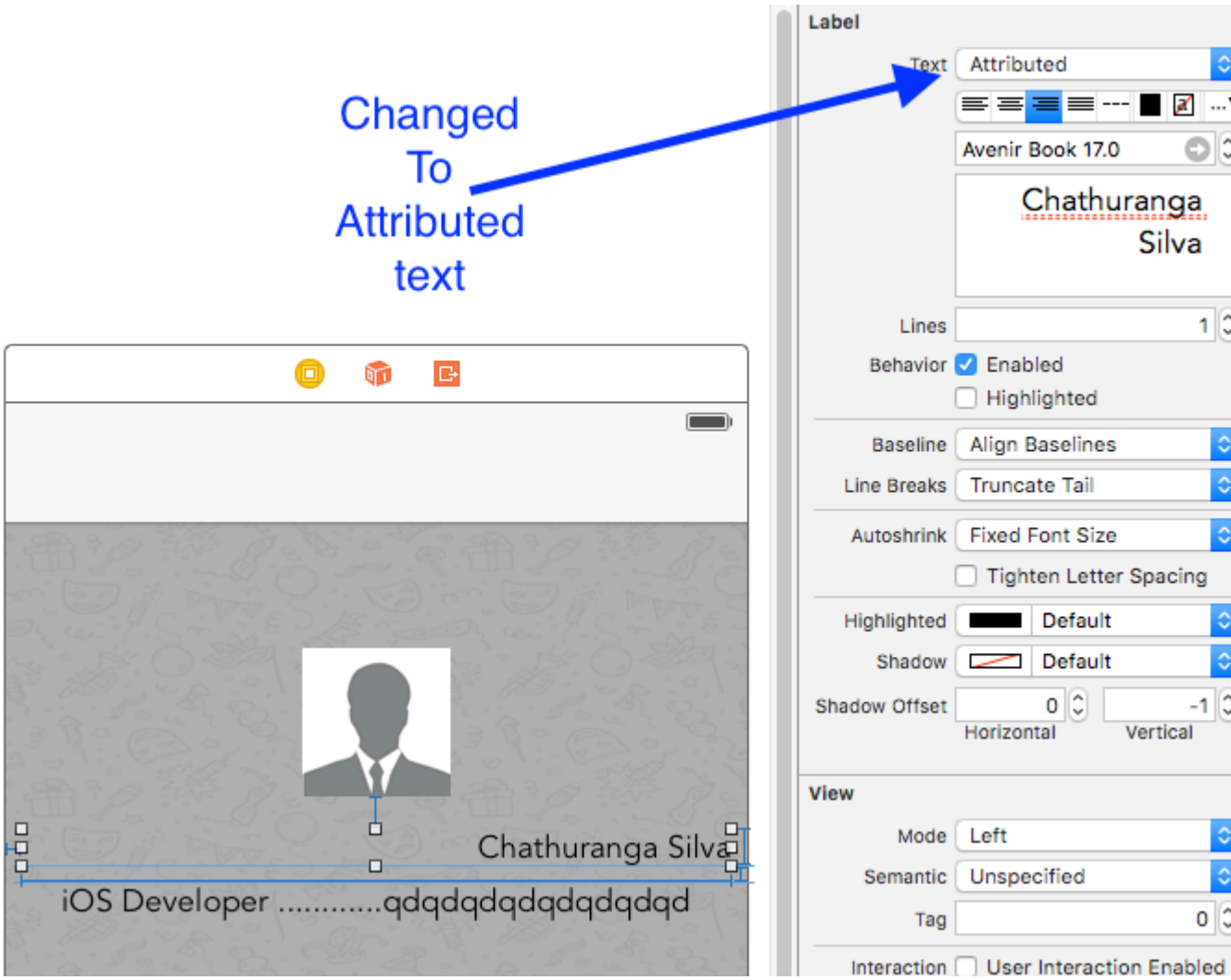
Текст, обозначенный ярлыком

01. Подчеркнутый текст: - Single / Double Line, Strike Through: - Single / Double Line

Шаг 1

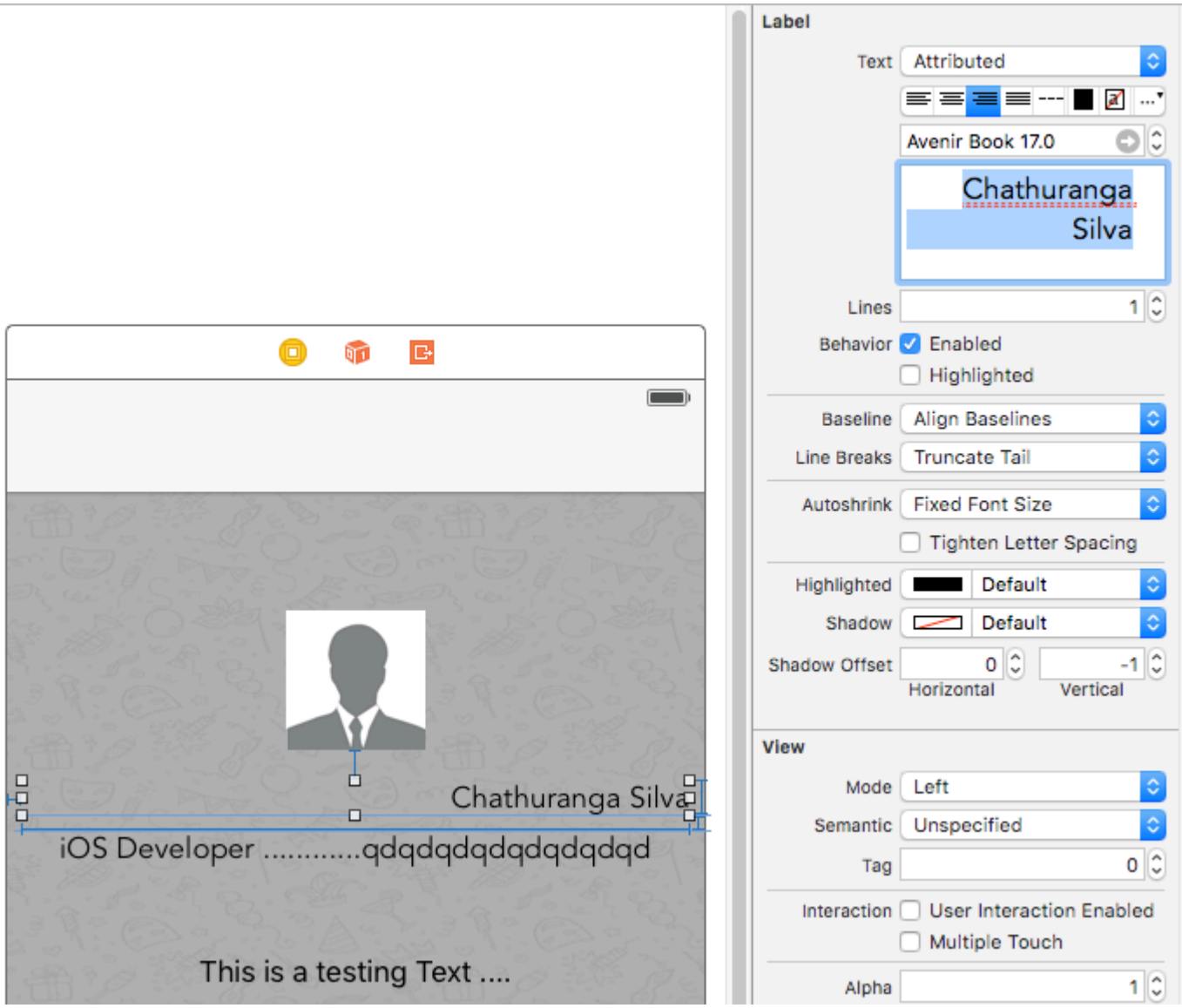
Выберите ярлык и измените тип метки Plain на Attributed





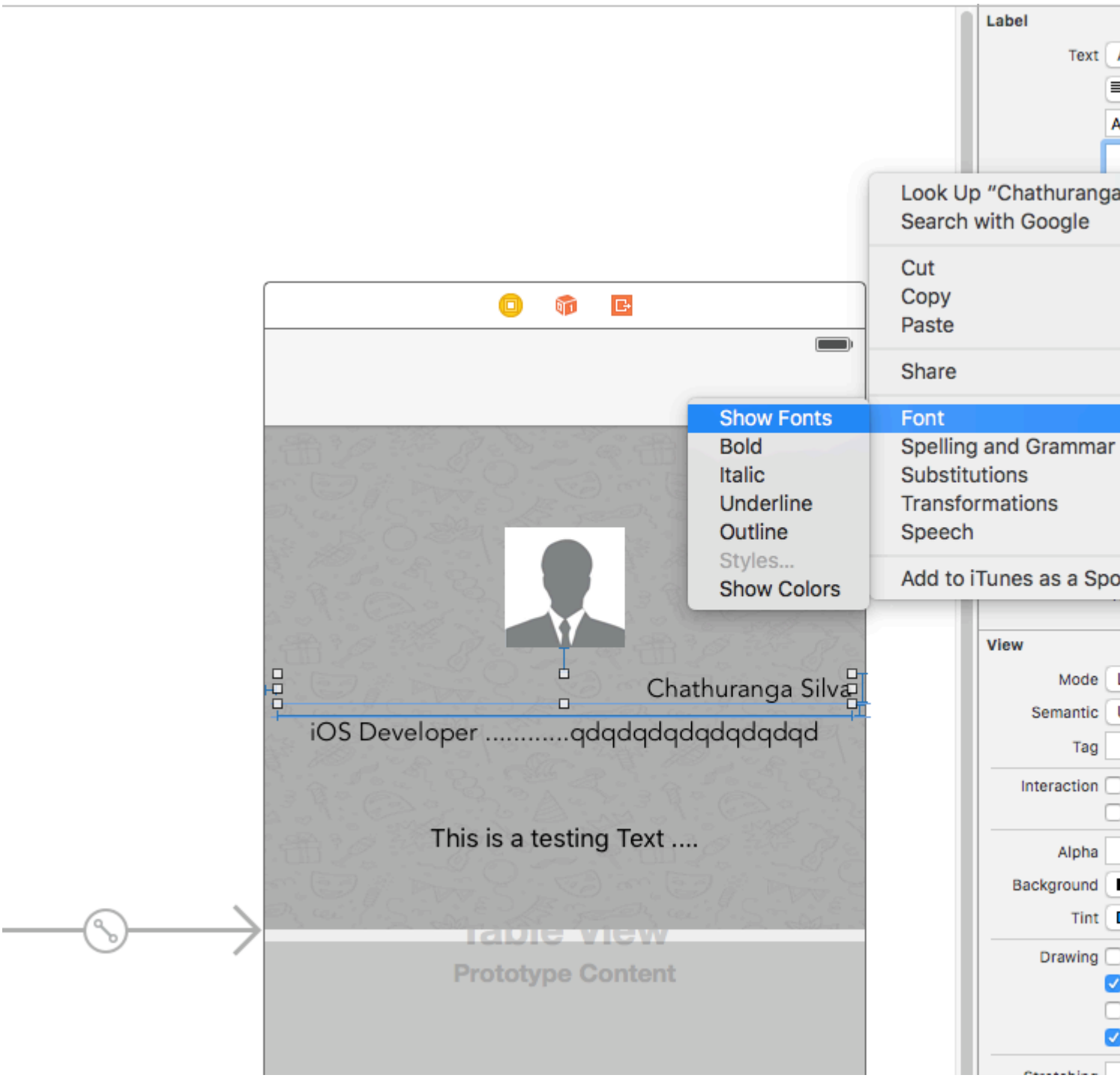
Шаг 2

Нажмите на текст ярлыка и щелкните правой кнопкой мыши



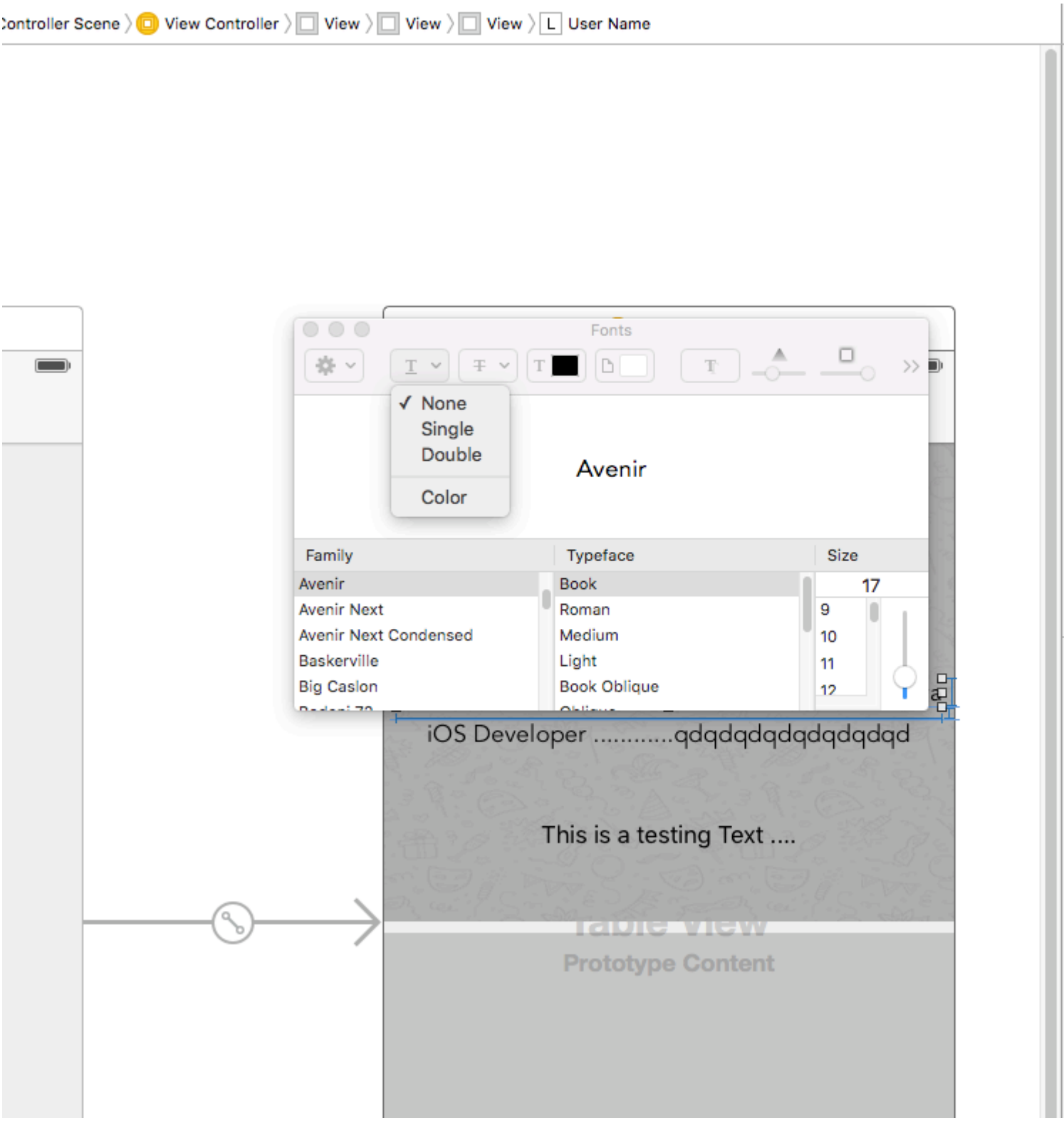
Шаг 3

Затем нажмите «Шрифт» -> «Показать шрифты».

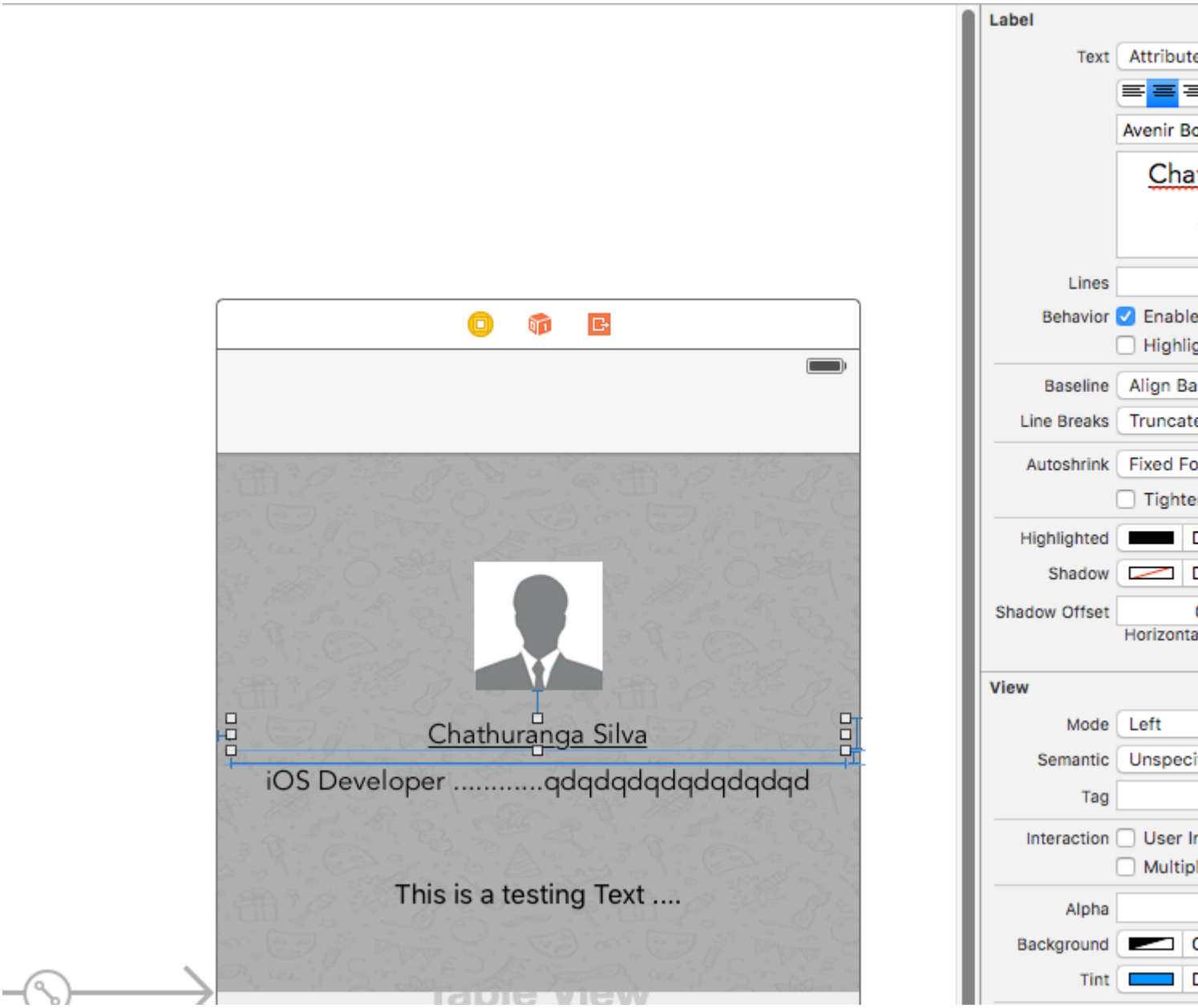


Шаг 4

Затем появится вид на шрифт и нажмите кнопку подчеркивания, чтобы подчеркнуть текст или нажмите кнопку прокрутки, чтобы сделать текст зачеркнутым. И выберите одну строку или двойную линию.

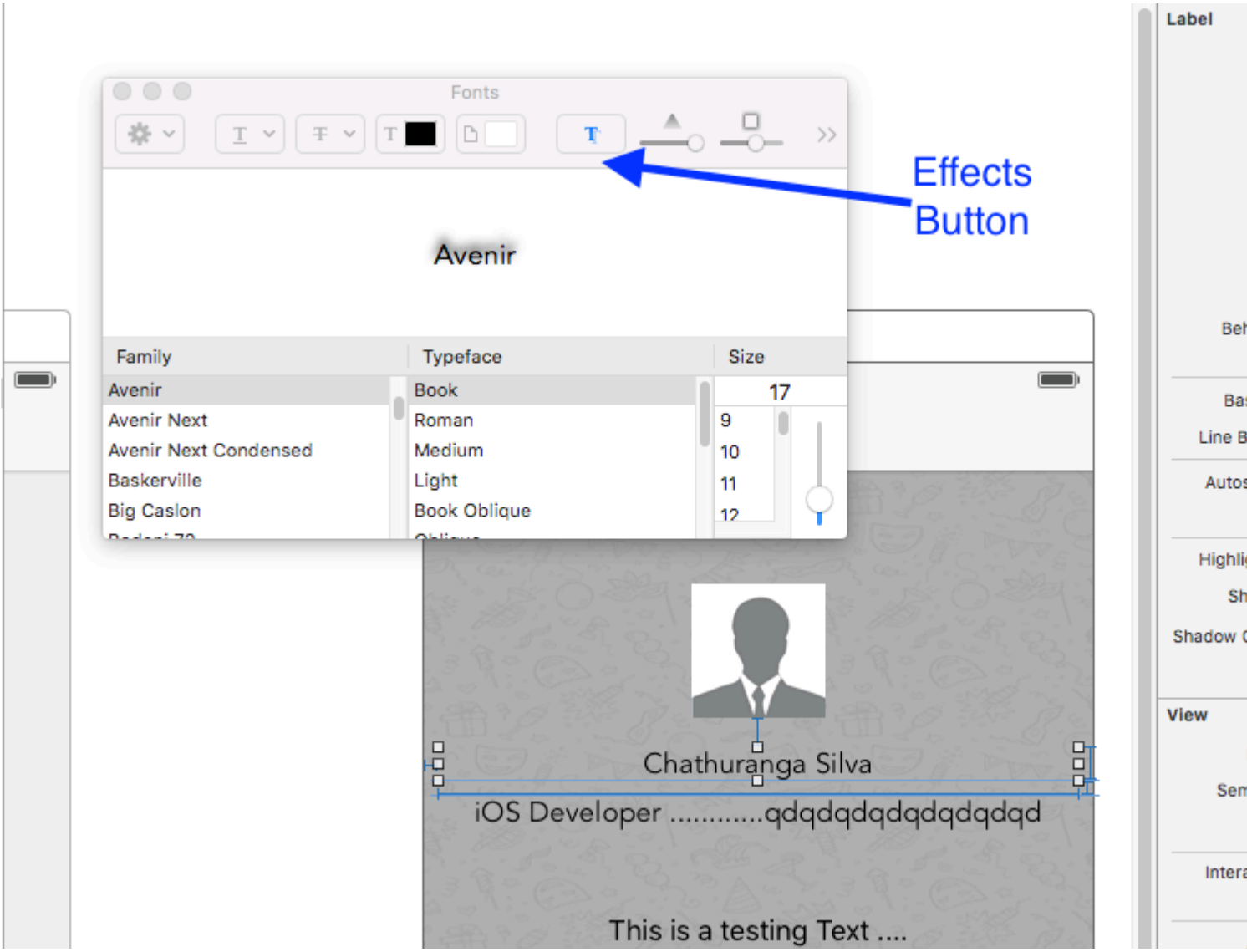


Наконец, нажмите «Ввод», и ярлык будет отображаться как подчеркивание или зачеркивание в соответствии с вашим выбором.

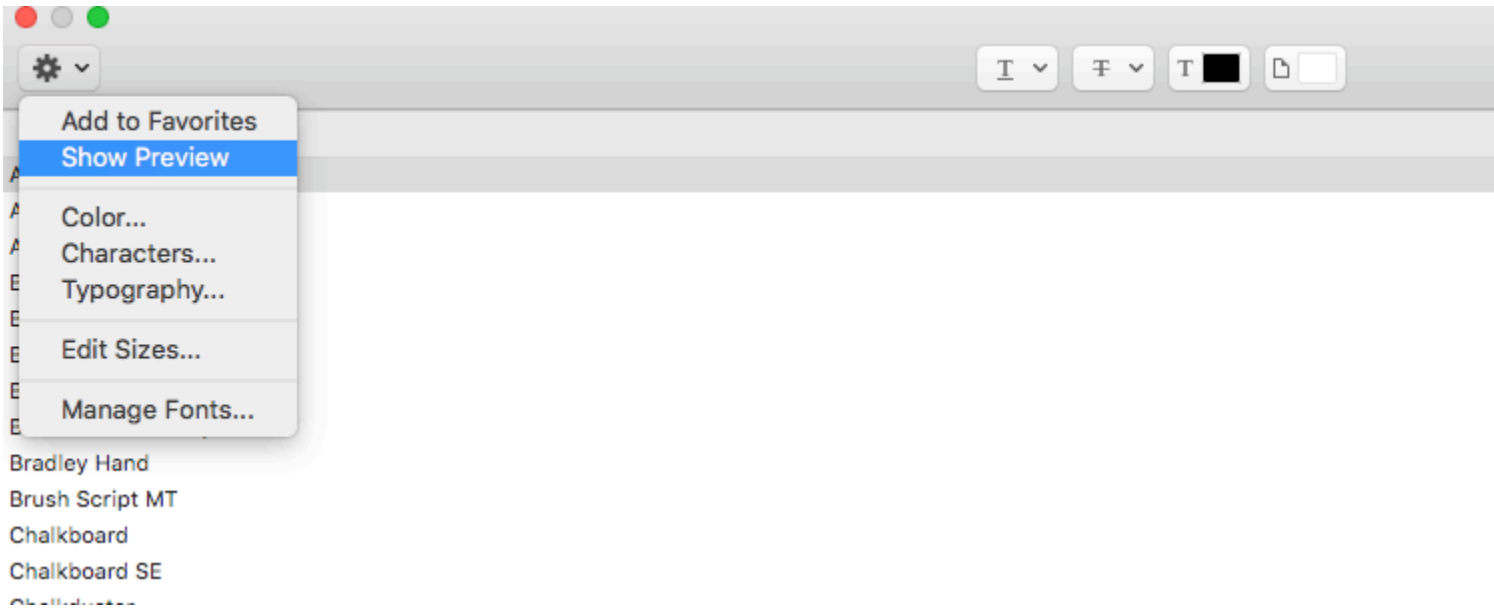


02. Добавить текст shaddow / background blur effects

Получите представление «Шрифт», как описано выше, и нажмите кнопку «Эффекты».



Если вы не видите предварительный просмотр, щелкните изображение показа в настройках



Наконец, измените shaddow и смещение в соответствии с вашими предпочтениями.

Обосновать текст

стриж

```
let sampleText = "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est
laborum."

// Create label
let label = UILabel(frame: CGRectMake(0, 0, view.frame.size.width, 400))
label.numberOfLines = 0
label.lineBreakMode = NSLineBreakMode.ByWordWrapping

// Justify text through paragraph style
let paragraphStyle = NSMutableParagraphStyle()
paragraphStyle.alignment = NSTextAlignment.Justified
let attributes = [NSParagraphStyleAttributeName: paragraphStyle,
NSBaselineOffsetAttributeName: NSNumber(float: 0)]
let attributedString = NSAttributedString(string: sampleText, attributes: attributes)
label.attributedText = attributedString
view.addSubview(label)
```

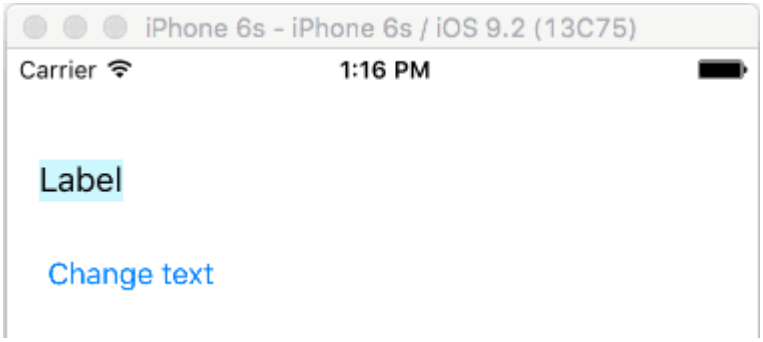
Objective-C

```
  NSString *sampleText = @"Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim
id est laborum.";

    // Create label
    UILabel *label = [[UILabel alloc] initWithFrame:CGRectMake(0, 0,
self.view.frame.size.width, 400)];
    label.numberOfLines = 0;
    label.lineBreakMode = NSLineBreakByWordWrapping;


    // Justify text through paragraph style
    NSMutableParagraphStyle *paragraphStyle = [[NSMutableParagraphStyle alloc] init];
    paragraphStyle.alignment = NSTextAlignmentJustified;
    NSAttributedString *attributedString = [[NSAttributedString alloc]
initWithString:sampleText attributes:@{
                       NSParagraphStyleAttributeName : paragraphStyle,
```

```
                    NSBaselineOffsetAttributeName : [NSNumber numberWithFloat:0]
                    }];
    label.attributedText = attributedString;
    [self.view addSubview:label];
```

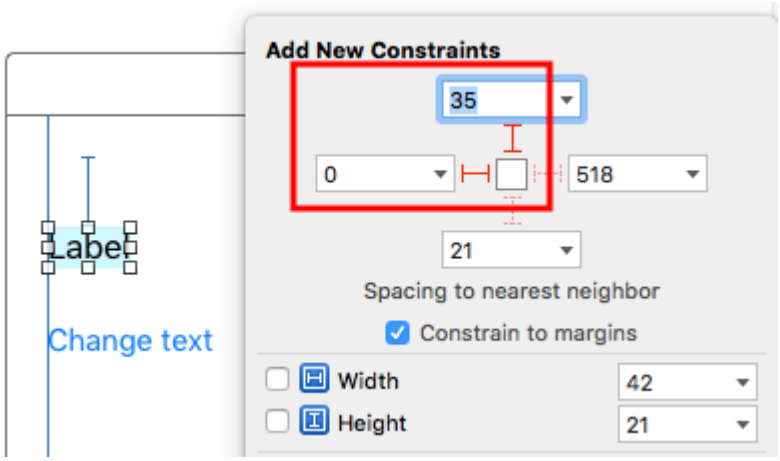
Автоматическая метка в соответствии с текстом

В этом примере показано, как ширина метки может автоматически изменяться при изменении текстового содержимого.



Прикрепите левый и верхний края

Просто используйте автомасштабирование, чтобы добавить ограничения, чтобы прикрепить левую и верхнюю стороны метки.



После этого он автоматически изменит размер.

Заметки

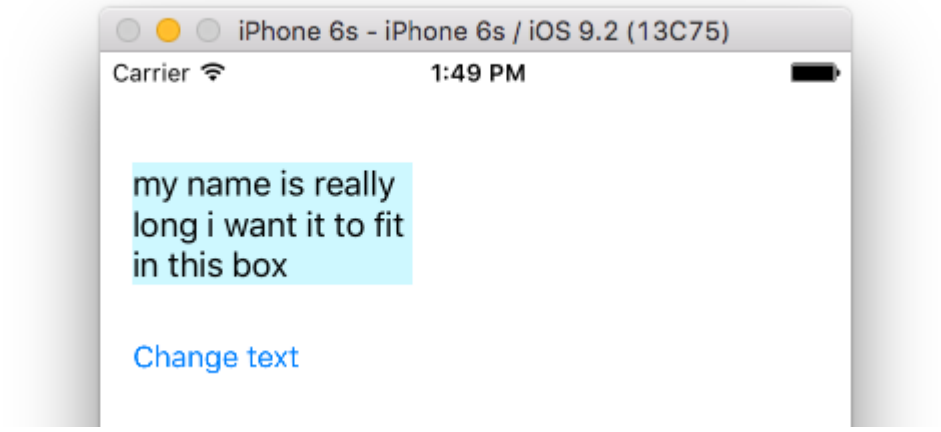
- Этот пример исходит из этого ответа переполнения стека .
- Не добавляйте ограничения для ширины и высоты. Ярлыки имеют *внутренний* размер, основанный на их текстовом содержимом.

- Не нужно устанавливать `sizeToFit` при использовании автоматического макета. Полный код для примера проекта приведен здесь:

```
import UIKit
class ViewController: UIViewController {

    @IBOutlet weak var myLabel: UILabel!

    @IBAction func changeTextButtonTapped(sender: UIButton) {
        myLabel.text = "my name is really long i want it to fit in this box"
    }
}
```

- Этот метод также может использоваться для правильного размещения нескольких меток по горизонтали, как в этом примере .

- Если вы хотите, чтобы ваш ярлык `myLabel.preferredMaxLayoutWidth = 150 // or whatever` к строке, установите количество строк в 0 в IB и добавьте `myLabel.preferredMaxLayoutWidth = 150 // or whatever` в коде. (Кнопка также прикреплена к нижней части этикетки, чтобы она сдвигалась вниз, когда высота этикетки увеличивалась.)



Получите размер UILabel строго на основе его текста и шрифта

`NSString` предоставляет метод `boundingRectWithSize` который можно использовать для прогнозирования полученного CGSize `UILabel` на основе его текста и шрифта без необходимости создания `UILabel`

Objective-C

```
[[text boundingRectWithSize:maxSize options:(NSStringDrawingTruncatesLastVisibleLine |
NSStringDrawingUsesLineFragmentOrigin) attributes:@{NSFontAttributeName: fontName}
context:nil] size];
```

стриж

```
let nsText = text as NSString?
nsText?.boundingRectWithSize(maxSize, options: [.TruncatesLastVisibleLine,
.UsesLineFragmentOrigin], attributes: [NSFontAttributeName: fontName], context: nil).size
```

стриж

Создать ярлык и метку «Ограничение ограничения высоты». Добавьте ниже код, в котором вы будете прикреплять текст к метке.

```
@IBOutlet var lblDescriptionHeightConstration: NSLayoutConstraint!
@IBOutlet weak var lblDescription: UILabel!

let maxWidth = UIScreen.mainScreen().bounds.size.width - 40
let sizeOfLabel = self.lblDesc.sizeThatFits(CGSize(width: maxWidth, height: CGFloat.max))
self.lblDescriptionHeightConstration.constant = sizeOfLabel.height
```

Примечание: «40» - это пространство левой и правой сторон экрана.

Цвет выделенного и выделенного текста

Objective-C

```
UILabel *label = [[UILabel alloc] init];
label.highlighted = YES;
label.highlightedTextColor = [UIColor redColor];
```

стриж

```
let label = UILabel()
label.highlighted = true
label.highlightedTextColor = UIColor.redColor()
```

Swift 3

```
let label = UILabel()
label.isHighlighted = true
label.highlightedTextColor = UIColor.red
```

Прочитайте UILabel онлайн: https://riptutorial.com/ru/ios/topic/246/uilabel

глава 84: UILocalNotification

Вступление

Локальные уведомления позволяют вашему приложению уведомлять пользователя о содержании, которое не требует использования сервера.

В отличие от удаленных уведомлений, которые запускаются с сервера, локальные уведомления планируются и запускаются в приложении. Уведомления в целом направлены на то, чтобы увеличить взаимодействие пользователя с приложением, пригласить или соблазнить пользователя открывать и взаимодействовать с ним.

UILocalNotification устарела в iOS 10. Вместо этого используйте Framework UserNotifications.

замечания

Не путайте UILocalNotification с помощью push-уведомлений. Устройство UILocalNotification запускается вашим устройством, и по расписанию копируется в систему.

Ссылки:

- Ссылка на класс UILocalNotification
- UILocalNotification при переполнении стека

Examples

Планирование локального уведомления

Убедитесь, что вы видите Регистрация для местных уведомлений, чтобы это работало:

стриж

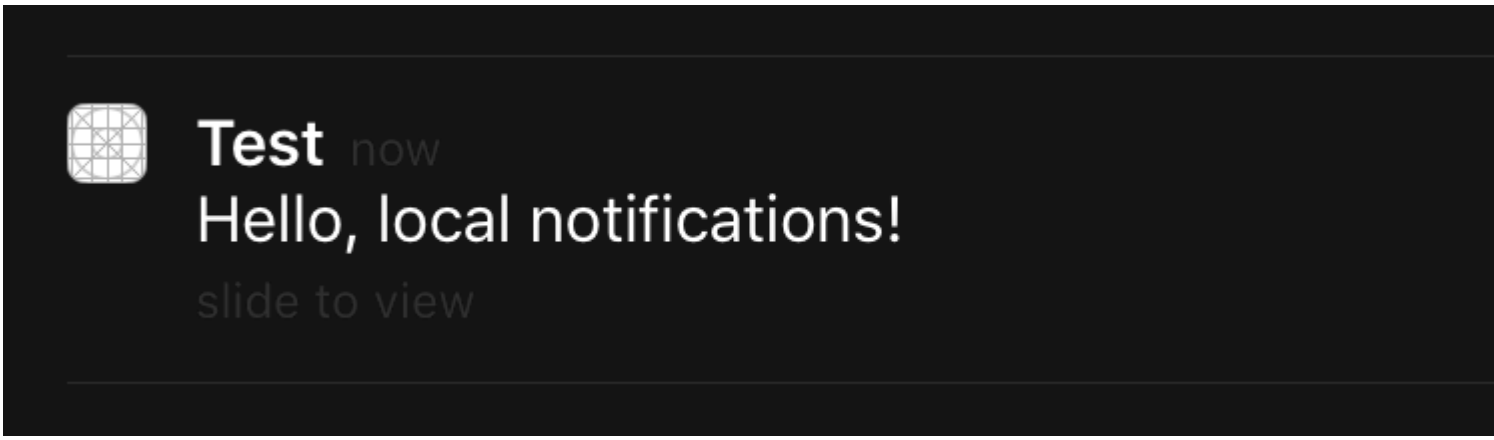
```
let notification = UILocalNotification()
notification.alertBody = "Hello, local notifications!"
notification.fireDate = NSDate().dateByAddingTimeInterval(10) // 10 seconds after now
UIApplication.sharedApplication().scheduleLocalNotification(notification)
```

Objective-C

```
UILocalNotification *notification = [[UILocalNotification alloc] init];
notification.alertBody = @"Hello, local notifications!";
notification.fireDate = [NSDate dateWithTimeIntervalSinceNow:10]; // 10 seconds after now
[[UIApplication sharedApplication] scheduleLocalNotification:notification];
```

Чтобы увидеть уведомление в симуляторе iOS, введите ^⌘H (control-command-H), чтобы

вернуться домой, а затем введите ⌘L (command-L), чтобы заблокировать устройство. Подождите несколько секунд, и появится уведомление (этот вид будет отличаться в зависимости от типа уведомления, описанного в разделе «Регистрация для локальных уведомлений»):



Проведите по уведомлению, чтобы вернуться в приложение (обратите внимание, что если вы вызвали это в представлении диспетчера `viewDidLoad` , `viewWillAppear` , `viewDidAppear` и т. Д., Уведомление будет запланировано снова).

Регистрация для местных уведомлений

iOS 8

Чтобы предоставлять пользователям локальные уведомления, вам необходимо зарегистрировать свое приложение на устройстве:

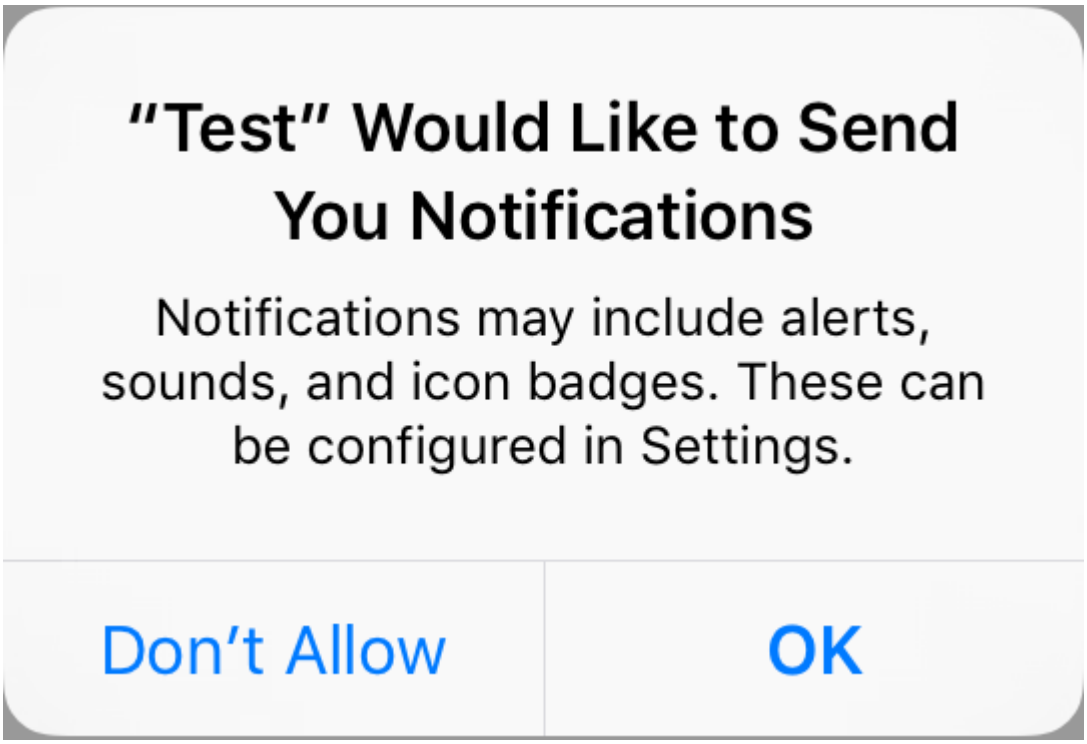
стриж

```
let settings = UIUserNotificationSettings(forTypes: [.Badge, .Sound, .Alert], categories: nil)
UIApplication.sharedApplication().registerUserNotificationSettings(settings)
```

Objective-C

```
UIUserNotificationSettings *settings = [UIUserNotificationSettings
settingsForTypes:(UIUserNotificationTypeBadge | UIUserNotificationTypeSound |
UIUserNotificationTypeAlert) categories:nil];
[[UIApplication sharedApplication] registerUserNotificationSettings:settings];
```

Это будет предупреждать при первом вызове:



Независимо от того, что пользователь выбирает, предупреждение не будет отображаться снова, и изменения должны быть инициированы пользователем в настройках.

Ответ на полученное местное уведомление

ВАЖНО: этот метод делегата вызывается только на переднем плане.

стриж

```
func application(application: UIApplication, didReceiveLocalNotification notification:
UILocalNotification) {

}
```

Objective-C

```
- (void)application:(UIApplication *)application
didReceiveLocalNotification:(UILocalNotification *)notification {

}
```

Этот метод обычно переопределяется в AppDelegate, который соответствует протоколу UIApplicationDelegate.

Управление локальными уведомлениями с использованием UUID

Часто вам нужно будет управлять своими уведомлениями, имея возможность отслеживать их и отменять.

Отслеживать уведомление

Вы можете назначить UUID (универсальный уникальный идентификатор) для уведомления, чтобы вы могли отслеживать его:

стриж

```
let notification = UILocalNotification()
let uuid = NSUUID().uuidString
notification.userInfo = ["UUID": uuid]
UIApplication.shared.scheduleLocalNotification(notification)
```

Objective-C

```
UILocalNotification *notification = [[UILocalNotification alloc] init];
NSString *uuid = [[NSUUID UUID] UUIDString];
notification.userInfo = @{ @"UUID": uuid };
[[UIApplication sharedApplication] scheduleLocalNotification:notification];
```

Отмена уведомления

Чтобы отменить уведомление, мы сначала получаем список всех уведомлений, а затем находим тот, у которого есть соответствующий UUID. Наконец, мы отменим его.

стриж

```
let scheduledNotifications = UIApplication.shared.scheduledLocalNotifications

guard let scheduledNotifications = scheduledNotifications else {
    return
}

for notification in scheduledNotifications where "\(notification.userInfo!["UUID"]!)" ==
UUID_TO_CANCEL {
    UIApplication.sharedApplication().cancelLocalNotification(notification)
}
```

Objective-C

```
NSArray *scheduledNotifications = [[UIApplication sharedApplication]
scheduledLocalNotifications];

for (UILocalNotification *notification in scheduledNotifications) {
    if ([[notification.userInfo objectForKey:"UUID"] compare: UUID_TO_CANCEL]) {
        [[UIApplication sharedApplication] cancelLocalNotification:notification];
        break;
    }
}
```

Вероятно, вы захотите сохранить все эти UUID в Core Data или Realm.

Немедленное представление локального уведомления

Если вы хотите немедленно указать местное уведомление, вы должны позвонить:

Swift 3

```
UIApplication.shared.presentLocalNotificationNow(notification)
```

Swift 2

```
UIApplication.sharedApplication().presentLocalNotificationNow(notification)
```

Objective-C

```
[[UIApplication sharedApplication] presentLocalNotificationNow:notification];
```

Преимущество использования этого заключается в том, что вам не нужно будет устанавливать свойства `fireDate` и `timeZone` вашего объекта `UILocalNotification` .

Звук уведомления

Пользовательские звуки могут быть предоставлены для уведомлений, созданных вашим приложением. Когда система отображает предупреждение для локального уведомления или значки значка приложения, он воспроизводит этот звук (пока пользователь не отключил звуки уведомлений).

Значение по умолчанию - nil, что означает, что для вашего уведомления не воспроизводится звук.

Чтобы предоставить настраиваемый звук, добавьте файл `.caf` , `.wav` или `.aiff` в свой пакет приложений. Звуки, длившиеся более 30 секунд, не поддерживаются. Подача звука, который не соответствует этим требованиям, приведет к воспроизведению звука по умолчанию (`UILocalNotificationDefaultSoundName`).

Objective-C

```
UILocalNotification *notification = [UILocalNotification new];
notification.soundName = @"nameOfSoundInBundle.wav"; // Use
UILocalNotificationDefaultSoundName for the default alert sound
```

стриж

```
let notification = UILocalNotification()
notification.soundName = "nameOfSoundInBundle.wav"
```

Регистрация и расписание локального уведомления в Swift 3.0 (iOS 10)

Постановка на учет

в **AppDelegate**

```
import UserNotifications
```

в методе **didFinishLaunchingWithOptions** ,

```
UNUserNotificationCenter.current().requestAuthorization(options: [.alert,.sound,.badge]) {
(granted, error) in

// Here you can check Request is Granted or not.

}
```

Создание и расписание уведомлений.

```
    let content = UNMutableNotificationContent()
    content.title = "10 Second Notification Demo"
    content.subtitle = "From Wolverine"
    content.body = "Notification after 10 seconds - Your pizza is Ready!!"
    content.categoryIdentifier = "myNotificationCategory"

    let trigger = UNTimeIntervalNotificationTrigger(
        timeInterval: 10.0,
        repeats: false)

    let request = UNNotificationRequest(
        identifier: "10.second.message",
        content: content,
        trigger: trigger
    )
    UNUserNotificationCenter.current().add(request, withCompletionHandler: nil)
```

Когда эта часть кода запускается, если вы разрешили разрешение на уведомление, вы получите уведомление.

Чтобы проверить его правильно, убедитесь, что ваше приложение в фоновом режиме.

что нового в UILocalNotification с iOS10

Вы можете использовать `UILocalNotification` , старые API также отлично работают с iOS10, но нам лучше использовать API в структуре пользовательских уведомлений. Есть также некоторые новые функции, вы можете использовать только с iOS10 User Notifications.

Это также случается с удаленным уведомлением, для получения дополнительной информации: здесь .

Новые возможности:

1. Теперь вы можете либо представить оповещение, звук или увеличить значок, в то

время как приложение находится на переднем плане тоже с iOS 10
2. Теперь вы можете обрабатывать все события в одном месте, когда пользователь нажал (или сдвинул) кнопку действия, даже когда приложение уже было убито.
3. Поддержка 3D-касания вместо раздвижного жестов.
4. Теперь вы можете удалить специфическое локальное уведомление только одним кодом строки.
5. Поддержка Rich Notification с пользовательским интерфейсом.

Нам очень легко преобразовать API `UILocalNotification` API-интерфейсы iOS10 User Notifications, они действительно похожи.

Я пишу демо здесь, чтобы показать, как использовать новые и старые API одновременно: **iOS10AdaptationTips** .

Например,

С реализацией Swift:

1. import UserNotifications

```
///    Notification become independent from UIKit
import UserNotifications
```

2. запросить авторизацию для localNotification

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

3. расписание localNotification

4. обновить значок значка значка приложения

```
@IBAction  func triggerNotification(){
    let content = UNMutableNotificationContent()
    content.title = NSString.localizedUserNotificationString(forKey: "Elon said:",
arguments: nil)
    content.body = NSString.localizedUserNotificationString(forKey: "Hello Tom Get up,
let's play with Jerry!", arguments: nil)
    content.sound = UNNotificationSound.default()
    content.badge = UIApplication.shared().applicationIconBadgeNumber + 1;
    content.categoryIdentifier = "com.elonchan.localNotification"
    // Deliver the notification in five seconds.
    let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 60.0, repeats:
true)
    let request = UNNotificationRequest.init(identifier: "FiveSecond", 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()
    // or you can remove specifical notification:
    // center.removePendingNotificationRequests(withIdentifiers: ["FiveSecond"])
}
```

Реализация Objective-C:

1. import UserNotifications

```
// Notifications are independent from UIKit
#import <UserNotifications/UserNotifications.h>
```

2. запросить авторизацию для localNotification

```
UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];
[center requestAuthorizationWithOptions:(UNAuthorizationOptionBadge |
UNAuthorizationOptionSound | UNAuthorizationOptionAlert)
                      completionHandler:^(BOOL granted, NSError * _Nullable error) {
                          if (!error) {
                              NSLog(@"request authorization succeeded!");
                              [self showAlert];
                          }
                      }];
```

3. расписание localNotification

4. обновить значок значка значка приложения

```
UNMutableNotificationContent *content = [[UNMutableNotificationContent alloc] init];
content.title = [NSString localizedUserNotificationStringForKey:@"Elon said:"
                                                      arguments:nil];
content.body = [NSString localizedUserNotificationStringForKey:@"Hello Tom Get up, let's
play with Jerry!"
                                                    arguments:nil];
content.sound = [UNNotificationSound defaultSound];

// 4. update application icon badge number
content.badge = [NSNumber numberWithInteger:([UIApplication
sharedApplication].applicationIconBadgeNumber + 1)];
// Deliver the notification in five seconds.
UNTimeIntervalNotificationTrigger *trigger = [UNTimeIntervalNotificationTrigger
                                              triggerWithTimeInterval:5.f
                                              repeats:NO];
UNNotificationRequest *request = [UNNotificationRequest
requestWithIdentifier:@"FiveSecond"
                                                                    content:content
                                                                    trigger:trigger];
/// 3. schedule localNotification
UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];
[center addNotificationRequest:request withCompletionHandler:^(NSError * _Nullable error)
{
```

```
    if (!error) {
        NSLog(@"add NotificationRequest succeeded!");
    }
}];
```

Перейдите сюда для получения дополнительной информации: **iOS10AdaptationTips** .

#updated

> Завершение приложения из-за неперехваченного исключения « NSInternalInconsistencyException», причина: «интервал времени должен быть не менее 60, если повторяется»

```
let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 60, repeats: true)
```

Прочитайте UILocalNotification онлайн: https://riptutorial.com/ru/ios/topic/635/uilocalnotification

глава 85: UINavigationController

замечания

Из документации :

> Класс UINavigationController реализует специализированный контроллер представлений, который управляет навигацией по иерархическому контенту. Этот навигационный интерфейс позволяет эффективно представлять ваши данные и облегчает пользователю навигацию по этому контенту. Обычно вы используете этот класс как есть, но вы также можете подклассы настраивать поведение класса.

Examples

Включение навигационного контроллера

К предыдущему контроллеру представления

Чтобы вернуться к предыдущей странице, вы можете сделать это:

стриж

```
navigationController?.popViewControllerAnimated(true)
```

Objective-C

```
[self.navigationController popViewControllerAnimated:YES];
```

К контроллеру корневого представления

Чтобы попасть в корень стека навигации, вы можете сделать это:

стриж

```
navigationController?.popToRootViewControllerAnimated(true)
```

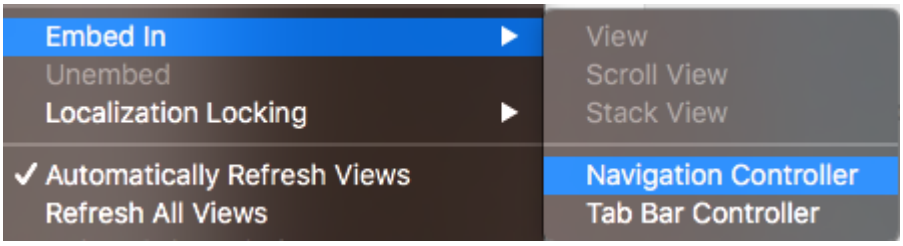
Цель С

```
[self.navigationController popToRootViewControllerAnimated:YES];
```

Создание навигационного контроллера

В своем раскадровке выберите ViewController, который вы хотите встроить в контроллер навигации.

Затем перейдите в Редактор> Вставить в> Контроллер навигации



И это создаст ваш контроллер навигации

Программно встраивать контроллер вида в навигационный контроллер

стриж

```
//Swift
let viewController = UIViewController()
let navigationController = UINavigationController(rootViewController: viewController)

//Objective-C
UIViewController *viewController = [[UIViewController alloc] init];
UINavigationController *navigationController = [[UINavigationController alloc]
initWithRootViewController:viewController];
```

Нажатие контроллера просмотра в стек навигации

```
//Swift
let fooViewController = UIViewController()
navigationController?.pushViewController(fooViewController, animated: true)

//Objective-C
UIViewController *fooViewController = [[UIViewController alloc] init];
[navigationController pushViewController:fooViewController animated:YES];
```

Цель

`UINavigationController` используется для формирования древовидной иерархии контроллеров представлений, которая называется `navigation stack` .

С точки зрения разработчиков:

Вы можете подключить самостоятельно сделанный контроллер и получить все преимущества бесплатного менеджера иерархии и общего приглашенного пользовательского интерфейса бесплатно. `UINavigationController` анимирует переход к новым контроллерам и автоматически предоставляет вам функциональные возможности. `UINavigationController` также предоставляет доступ ко всем другим контроллерам в `navigation stack` которые могут помочь получить доступ к некоторым функциям или данным.

С точки зрения пользователя:

`UINavigationController` помогает запомнить, где пользователь в данный момент (название панели навигации), и как он может вернуться (встроенная кнопка назад) на один из предыдущих экранов.

Прочитайте UINavigationController онлайн:
https://riptutorial.com/ru/ios/topic/1079/uinavigationcontroller

глава 86: UIPageViewController

Вступление

UIPageViewController предоставляет пользователям возможность легко переключаться между несколькими видами с помощью жестов салфетки. Чтобы создать UIPageViewController, вы должны реализовать методы UIPageViewControllerDataSource. К ним относятся методы возврата как UIPageViewController до, так и после текущего UIPageViewController вместе с методами presentationCount и presentationIndex.

Синтаксис

1. UIPageViewControllerTransitionStyle
2. UIPageViewControllerNavigationOrientation
3. UIPageViewControllerSpineLocation
4. UIPageViewControllerNavigationDirection

замечания

Ссылка Apple Developer здесь

Examples

Создавать горизонтальный пейджинг UIPageViewController

1. Init массива контроллеров представлений, которые будут управляться UIPageViewController. Добавьте класс контроллера базового представления, который имеет `identifier` свойства, который будет использоваться для идентификации контроллеров представлений при работе с методами источника данных UIPageViewController. Пусть контроллеры представления наследуют этот базовый класс.

```
UIViewController *firstVC = [[UIViewController alloc] init];
firstVC.identifier = 0
UIViewController *secondVC = [[UIViewController alloc] init];
secondVC.identifier = 1
NSArray *viewControllers = [[NSArray alloc] initWithObjects: firstVC, secondVC, nil];
```

2. Создайте экземпляр UIPageViewController.

```
UIPageViewController *pageViewController = [[UIPageViewController alloc]
initWithTransitionStyle:UIPageViewControllerTransitionStyleScroll
```

```
navigationOrientation:UIPageViewControllerNavigationOrientationHorizontal

options:nil];
```

3. Источником данных является текущий класс, который должен реализовывать протокол `UIPageViewControllerDataSource` .

```
pageViewController.dataSource = self;
```

4. `setViewControllers` добавит только первый контроллер представления, следующий будет добавлен в стек с использованием методов источника данных

```
if (viewControllers.count) {
    [pageViewController setViewControllers:@[[viewControllers objectAtIndex:0]]
                                 direction:UIPageViewControllerNavigationDirectionForward
                                  animated:NO
                                completion:nil];
}
```

5. Добавьте UIPageViewController в качестве контроллера детского представления, чтобы он получал от него `appearance` вид контроллера `appearance` и события `rotation` .

```
 [self addChildViewController:pageViewController];
 pageViewController.view.frame = self.view.frame;
 [self.view addSubview:pageViewController.view];
 [pageViewController didMoveToParentViewController:self];
```

6. Внедрение методов UIPageViewControllerDataSource

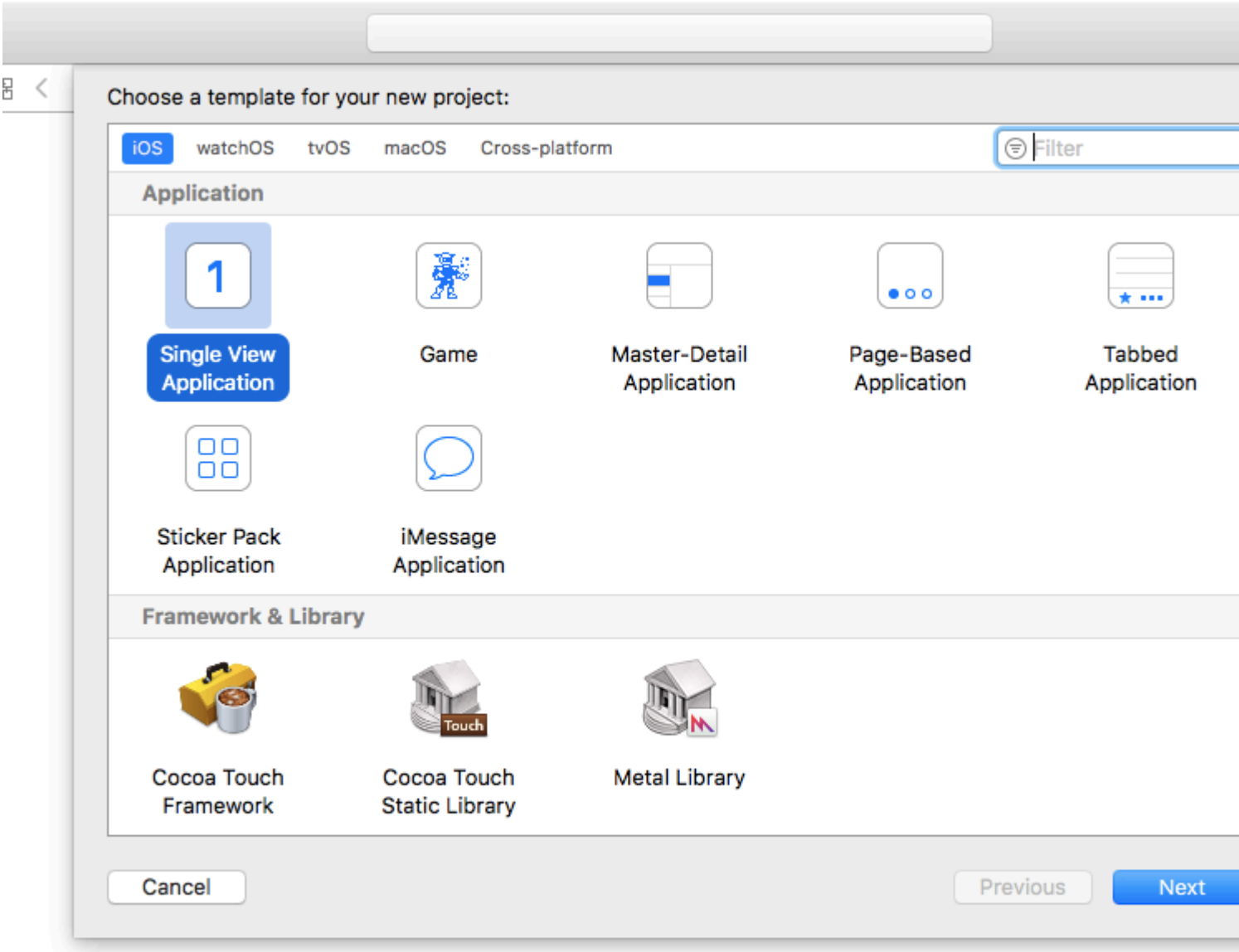
```
- (UIViewController *)pageViewController:(UIPageViewController *)pageViewController
      viewControllerBeforeViewController:(UIViewController *)viewController
{
    index = [(Your View Controler Base Class *)viewController identifier];
    index--;
    return [self childViewControllerAtIndex:index];
}

- (UIViewController *)pageViewController:(UIPageViewController *)pageViewController
       viewControllerAfterViewController:(UIViewController *)viewController
{
    index = [(Your View Controler Base Class *)viewController identifier];
    index++;
    return [self childViewControllerAtIndex:index];
}

- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController
{
    return [viewControllers count];
}

- (NSInteger)presentationIndexForPageViewController:(UIPageViewController *)pageViewController
{
    return index;
}
```

7. Метод Utility, который возвращает контроллер представления с использованием индекса, если индекс не соответствует границам, он возвращает nil.

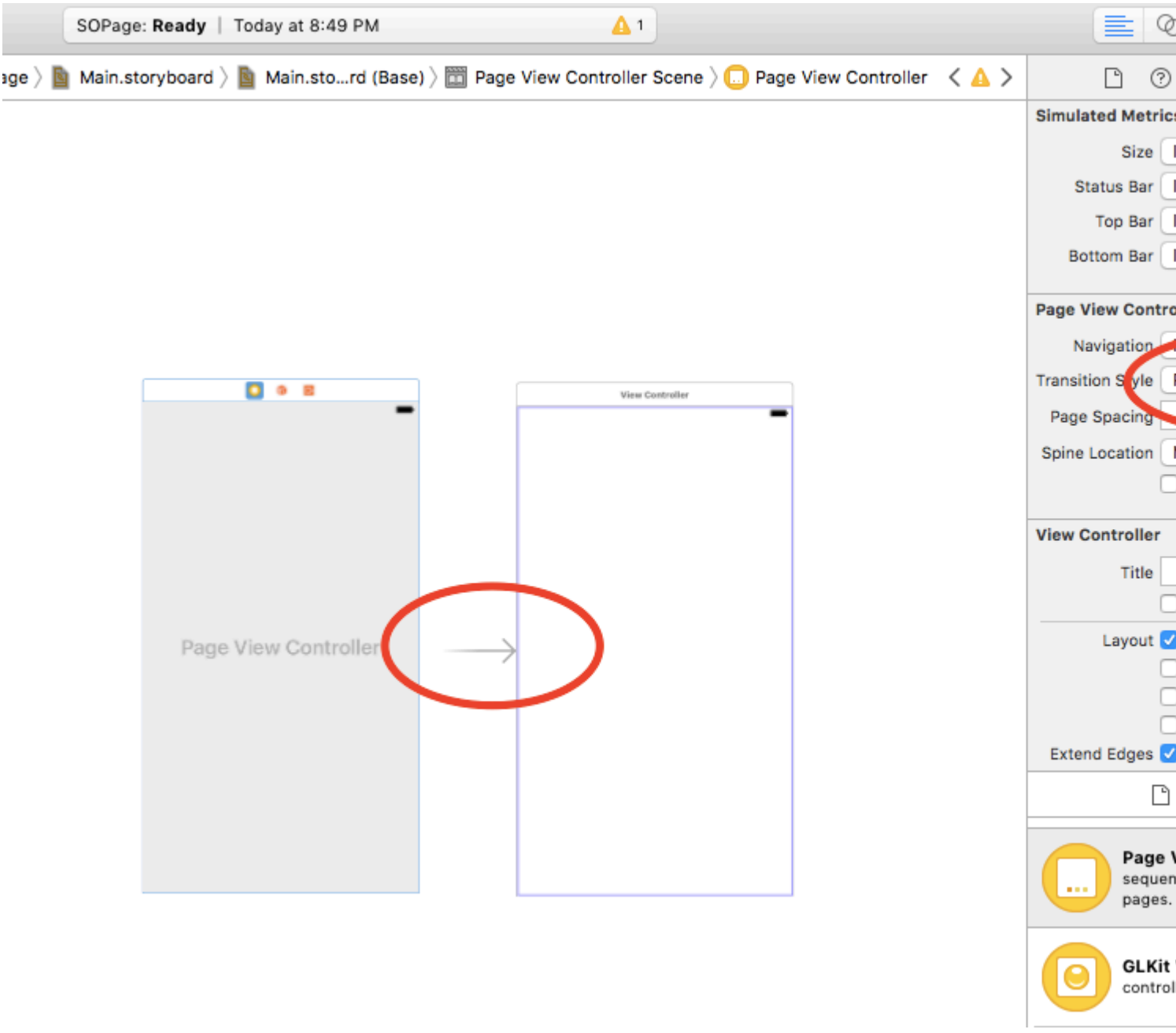
```
- (UIViewController *)childViewControllerAtIndex:(NSInteger)index
{
    if (index <= ([viewControllers count] - 1)) {
        return [viewControllers objectAtIndex:index];
    } else {
        return nil;
    }
}
```

Простой способ создания горизонтальных диспетчеров просмотра страниц (бесконечные страницы)

1. Давайте создадим новый проект, я выбираю приложение Single View для лучшей демонстрации



2. Перетащите контроллер просмотра страницы в раскадровку, после чего вам нужно изменить две вещи:
    1. Установите контроллер просмотра страницы в качестве начального контроллера представления
    2. Измените стиль перехода на прокрутку



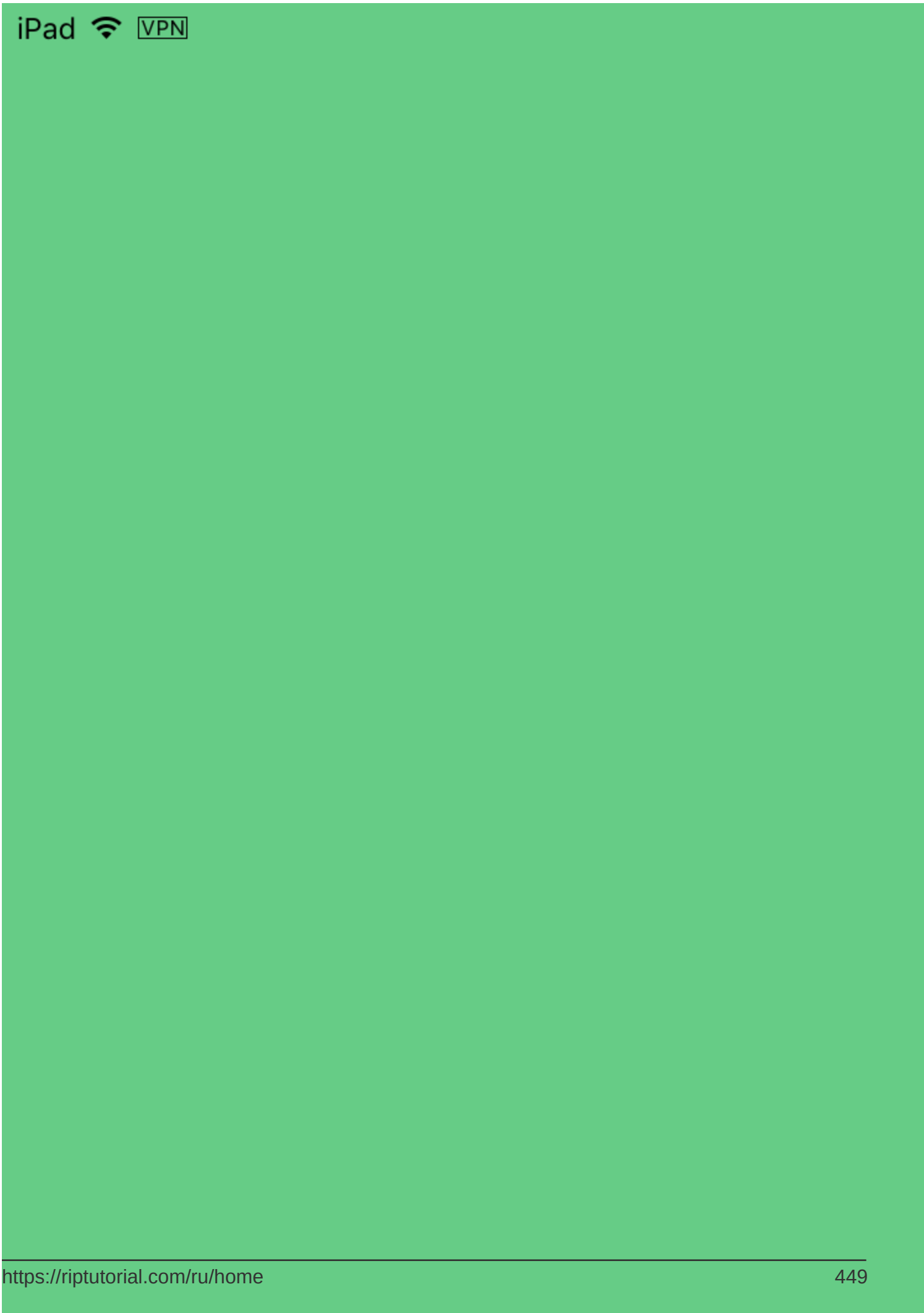
3. И вам нужно создать класс UIPageViewController, а затем установить его как пользовательский класс контроллера просмотра страницы на раскадровке

4. Вставьте этот код в свой класс UIPageViewController, вы должны получить красочное бесконечное выгружаемое приложение :)

```
class PageViewController: UIPageViewController, UIPageViewControllerDataSource {

    override func viewDidLoad() {
```

```
        self.dataSource = self
        let controller = createViewController()
        self.setViewControllers([controller], direction: .forward, animated: false,
completion: nil)
    }

    func pageViewController(_ pageViewController: UIPageViewController,
viewControllerBefore viewController: UIViewController) -> UIViewController? {
        let controller = createViewController()
        return controller
    }

    func pageViewController(_ pageViewController: UIPageViewController,
viewControllerAfter viewController: UIViewController) -> UIViewController? {
        let controller = createViewController()
        return controller
    }

    func createViewController() -> UIViewController {
        var randomColor: UIColor {
            return UIColor(hue: CGFloat(arc4random_uniform(360))/360, saturation: 0.5,
brightness: 0.8, alpha: 1)
        }
        let storyboard = UIStoryboard(name: "Main", bundle: nil)
        let controller = storyboard.instantiateViewController(withIdentifier: "View
Controller")
        controller.view.backgroundColor = randomColor
        return controller
    }
}
```

Вот как выглядит последний проект, с каждым свитком вы получаете контроллер вида с разным цветом:


iPad
VPN

https://riptutorial.com/ru/ios/topic/3291/uipageviewcontroller

глава 87: UIPheonix - простой, гибкий, динамичный и масштабируемый интерфейс пользовательского интерфейса

Вступление

Вдохновленный разработкой игр UIPheonix - это супер простая, гибкая, динамичная и масштабируемая концепция пользовательского интерфейса + для создания многоразовых приложений с компонентами / управляемыми приложениями для macOS, iOS и tvOS. Тот же API применяется для кросс-платформенной разработки! Подумайте об этом, используя блоки Lego, вы можете использовать похожие и легко перемещать их как пирог.

https://github.com/MKGitHub/UIPheonix

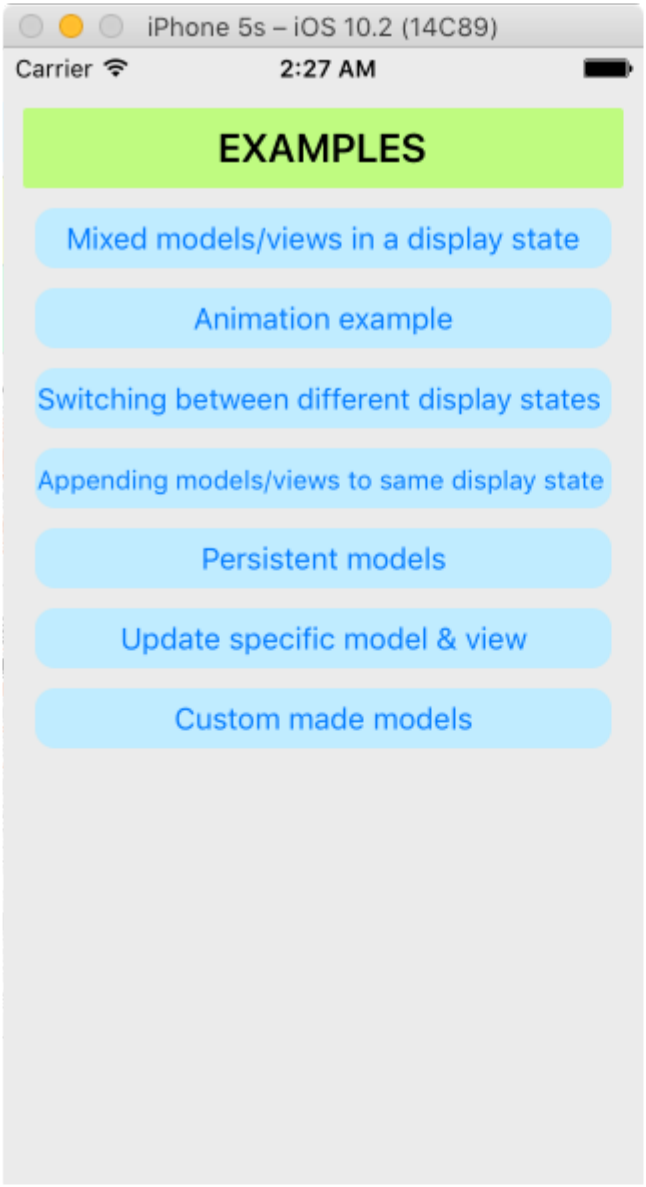
замечания

- Забудьте о статических макетах, проблемах с ограничениями и предупредительных взрывах на консоли.
- Забудьте весь код клея, весь шаблонный код и все очень распространенные излишне сконструированные ненужные кучи кода мусора в ваших приложениях.
- Постройте и внесите изменения в свой интерфейс пользователя быстро.
- Сделайте свой пользовательский интерфейс повторно используемым.
- Сосредоточьтесь на создании своего приложения, а не на проблемах макета.
- Минимальная настройка, минимальное воздействие на ваше приложение, легкий вес, отсутствие зависимостей, отсутствие боли, но столько выигрыша!
- Создает элементы коллекции и таблицы, поэтому вы можете легко смешивать и сопоставлять.
- Не заменяет технологии Apple нестандартными реализациями, поэтому вы всегда будете в безопасности и обновлены, и вы можете легко вернуться в любое время.
- Демо-приложения для macOS, iOS и tvOS (Kung Fu!)

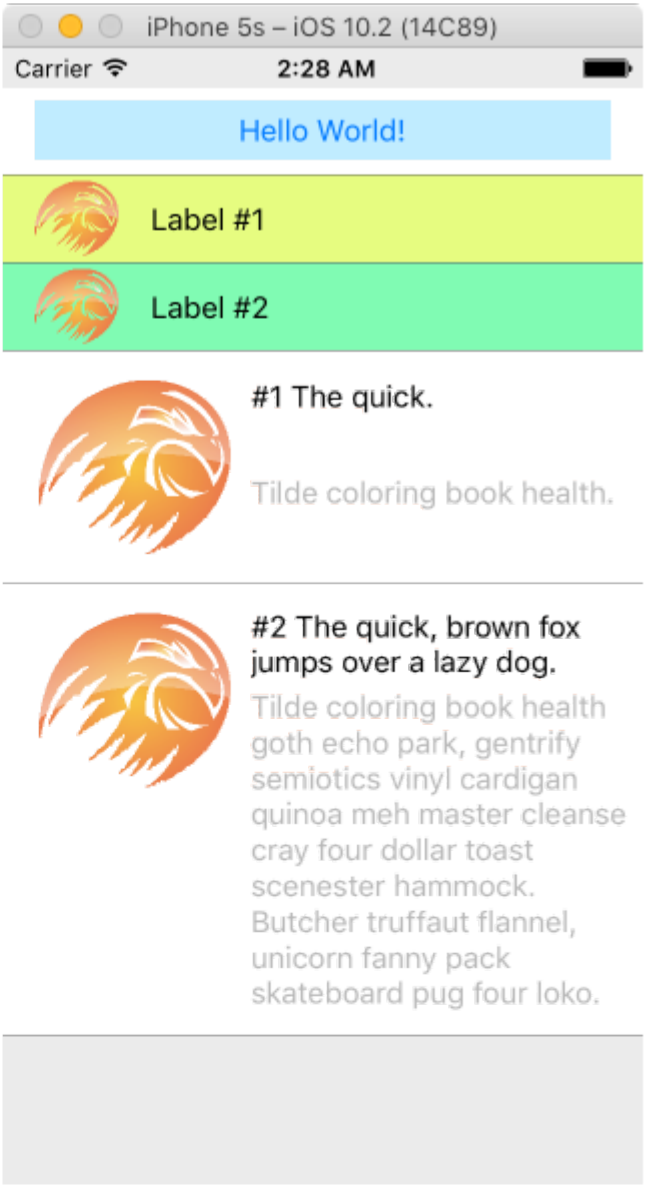
Examples

Примеры компонентов пользовательского интерфейса

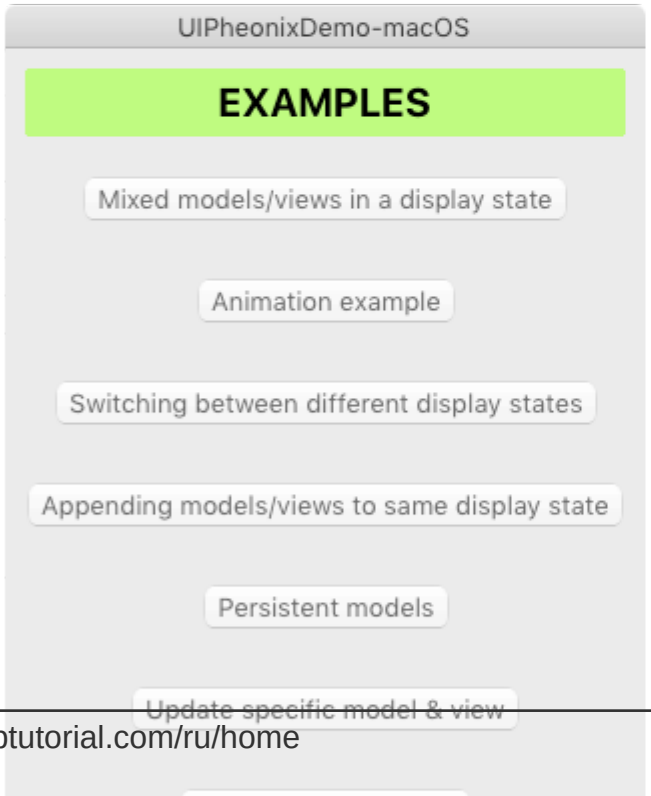
iOS - Collection View



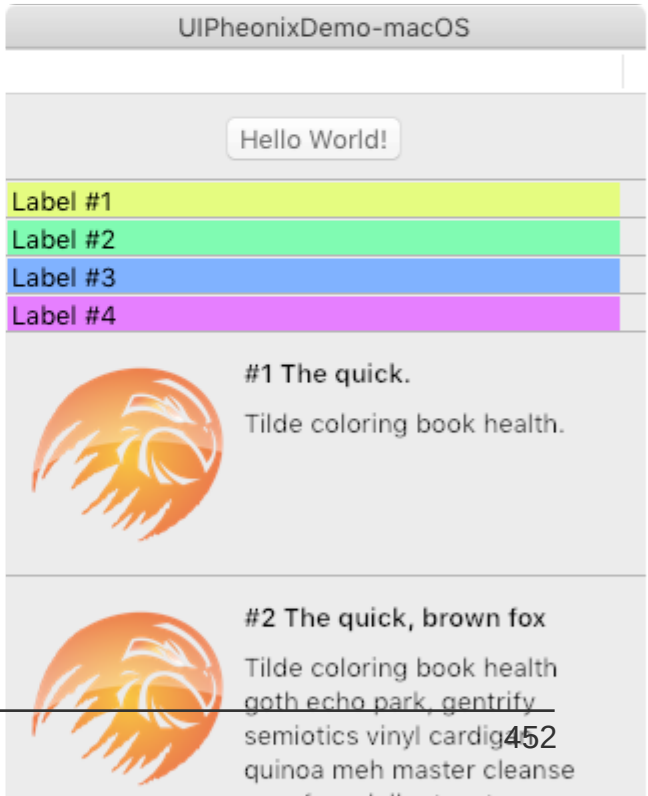
iOS - Table View



macOS - Collection View



macOS - Table View



простой, гибкий, динамичный и масштабируемый интерфейс пользовательского интерфейса онлайн: https://riptutorial.com/ru/ios/topic/9120/uipheonix---простой--гибкий--динамичный-и-масштабируемый-интерфейс-пользовательского-интерфейса

глава 88: UIPickerView

Examples

Основной пример

стриж

```
class PickerViewExampleViewController : UIViewController, UIPickerViewDelegate,
UIPickerViewDataSource {
    @IBOutlet weak var btnFolder: UIButton!
    let pickerView = UIPickerView()
    let pickerViewRows = ["First row,", "Secound row,","Third row,","Fourth row"]

    override func viewDidLoad() {
        super.viewDidLoad()
        self.btnFolder.addTarget(self, action: #selector(CreateListVC.btnFolderPress),
forControlEvents: UIControlEvents.TouchUpInside)
    }

    @objc private func btnFolderPress() {
        self.pickerView.delegate = self
        self.pickerView.dataSource = self
        self.view.addSubview(self.pickerView)
    }

    //MARK: UIPickerViewDelegate

    func pickerView(pickerView: UIPickerView, titleForRow row: Int, forComponent component:
Int) -> String? {
        return self.pickerViewRows[row]
    }

    //MARK: UIPickerViewDataSource

    func numberOfComponentsInPickerView(pickerView: UIPickerView) -> Int {
        return 1
    }

    func pickerView(pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
        return self.pickerViewRows.count
    }

}
```

Objective-C

```
@property (nonatomic,strong) UIPickerView *countryPicker;
@property (nonatomic,strong) NSArray *countryNames;
```

```
- (void)viewDidLoad {
    [super viewDidLoad];
    _countryNames = @[@"Australia (AUD)", @"China (CNY)",
                          @"France (EUR)", @"Great Britain (GBP)", @"Japan (JPY)",@"INDIA
(IN)",@"AUSTRALIA (AUS)",@"NEW YORK (NW)"];

   [self pickcountry];
 }

-(void)pickcountry {
    _countryPicker = [[UIPickerView alloc]init];

    _countryPicker.delegate = self;
    _countryPicker.dataSource = self;

    [[UIPickerView appearance] setBackgroundColor:[UIColor colorWithRed:21/255.0
green:17/255.0 blue:50/255.0 alpha:1.0]];
}

#pragma mark-  pickerView Delegates And datasource

- (NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerView {
    return 1;
}

- (NSInteger)pickerView:(UIPickerView *)pickerView
numberOfRowsInComponent:(NSInteger)component {
    return _countryNames.count;
}

- (NSString *)pickerView:(UIPickerView *)pickerView
             titleForRow:(NSInteger)row
            forComponent:(NSInteger)component {
    return _countryNames[row];
}

- (void)pickerView:(UIPickerView *)pickerView didSelectRow:(NSInteger)row
inComponent:(NSInteger)component {
      NSString *pickedCountryName = _countryNames[row];
}
```

Изменение цвета pickerView Цвет фона и цвет текста

Objective-C

```
//Displays the country pickerView with black background and white text
[self. countryPicker setValue:[UIColor whiteColor] forKey:@"textColor"];
[self. countryPicker setValue:[UIColor blackColor] forKey:@"backgroundColor"];
```

стриж

```
let color1 = UIColor(colorLiteralRed: 1, green: 1, blue: 1, alpha: 1)
let color2 = UIColor(colorLiteralRed: 0, green: 0, blue: 0, alpha: 1)
pickerView2.setValue(color1, forKey: "textColor")
pickerView2.setValue(color2, forKey: "backgroundColor")
```

Прочитайте UIPickerView онлайн: https://riptutorial.com/ru/ios/topic/4242/uipickerview

глава 89: UIRefreshControl TableView

Вступление

Объект UIRefreshControl предоставляет стандартный элемент управления, который можно использовать для инициирования обновления содержимого табличного представления. Вы связываете элемент управления обновлением с таблицей через связанный с ним объект контроллера табличного представления. Контроллер табличного представления обрабатывает работу по добавлению элемента управления в визуальный внешний вид таблицы и управление отображением этого элемента управления в ответ на соответствующие жесты пользователя.

Examples

Пример Objective-C

Сначала объявите свойство, подобное этому в ViewController

```
@property (nonatomic) UIRefreshControl *refreshControl;
```

Позже в `viewDidLoad()` настройте refreshControl, как показано ниже:

```
self.refreshControl = [[UIRefreshControl alloc]init];
[self.tableView addSubview:self.refreshControl];
[self.refreshControl addTarget:self action:@selector(refreshTable)
forControlEvents:UIControlEventValueChanged];
//Setting the tint Color of the Activity Animation
self.refreshControl.tintColor = [UIColor redColor];
//Setting the attributed String to the text
NSMutableAttributedString * string = [[NSMutableAttributedString alloc]
initWithString:@"firstsecondthird"];
[string addAttribute:NSForegroundColorAttributeName value:[UIColor redColor]
range:NSMakeRange(0,5)];
[string addAttribute:NSForegroundColorAttributeName value:[UIColor greenColor]
range:NSMakeRange(5,6)];
[string addAttribute:NSForegroundColorAttributeName value:[UIColor blueColor]
range:NSMakeRange(11,5)];
self.refreshControl.attributedTitle = string;
```

Теперь функция `refreshTable` определяется как:

```
- (void)refreshTable {
    //TODO: refresh your data
    [self.refreshControl endRefreshing];
    [self.refreshControl beginRefreshing];
    [self.tableView reloadData];
    [self.refreshControl endRefreshing];
}
```

Настроить refreshControl на tableView:

```
UIRefreshControl *refreshControl = [[UIRefreshControl alloc] init];
[refreshControl addTarget:self action:@selector(pullToRefresh:)
forControlEvents:UIControlEventValueChanged];
self.scrollView.alwaysBounceVertical = YES;
[self.scrollView addSubview:refreshControl];

- (void)pullToRefresh:(UIRefreshControl*) sender{
//Do work off the main thread
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
    // Simulate network traffic (sleep for 2 seconds)
    [NSThread sleepForTimeInterval:2];
    //Update data
    //Call complete on the main thread
    dispatch_sync(dispatch_get_main_queue(), ^{
        //Update network activity UI
        NSLog(@"COMPLETE");
        [sender endRefreshing];
    });
});
```

}

Прочитайте UIRefreshControl TableView онлайн:
https://riptutorial.com/ru/ios/topic/8278/uirefreshcontrol-tableview

глава 90: UIScrollView

Examples

Создание UIScrollView

Создайте экземпляр `UIScrollView` с `CGRect` качестве фрейма.

стриж

```
let scrollview = UIScrollView.init(frame: CGRect(x: 0, y: 0, width: 320, height: 400))
```

Objective-C

```
UIScrollView *scrollview = [[UIScrollView alloc] initWithFrame:CGRectMake(0, 0, 320, 400)];
```

Просмотр содержимого

Свойство `contentSize` должно быть установлено в размере прокручиваемого содержимого. Указывает размер прокручиваемой области. Прокрутка видима при прокручиваемой области, т.е. `contentSize` больше, чем размер кадра `UIScrollView` .

C Autolayout:

Когда содержимое представления прокрутки настраивается с использованием автозапуска, оно должно быть явно размером как по вертикали, так и по горизонтали и иметь все 4 ребра, прикрепленные к содержащему прокрутку. Таким образом, `contentSize` рассчитывается автоматически на основе содержимого прокрутки, а также обновляется при изменении макета содержимого.

Вручную:

стриж

```
scrollview.contentSize = CGSize(width: 640, height: 800)
```

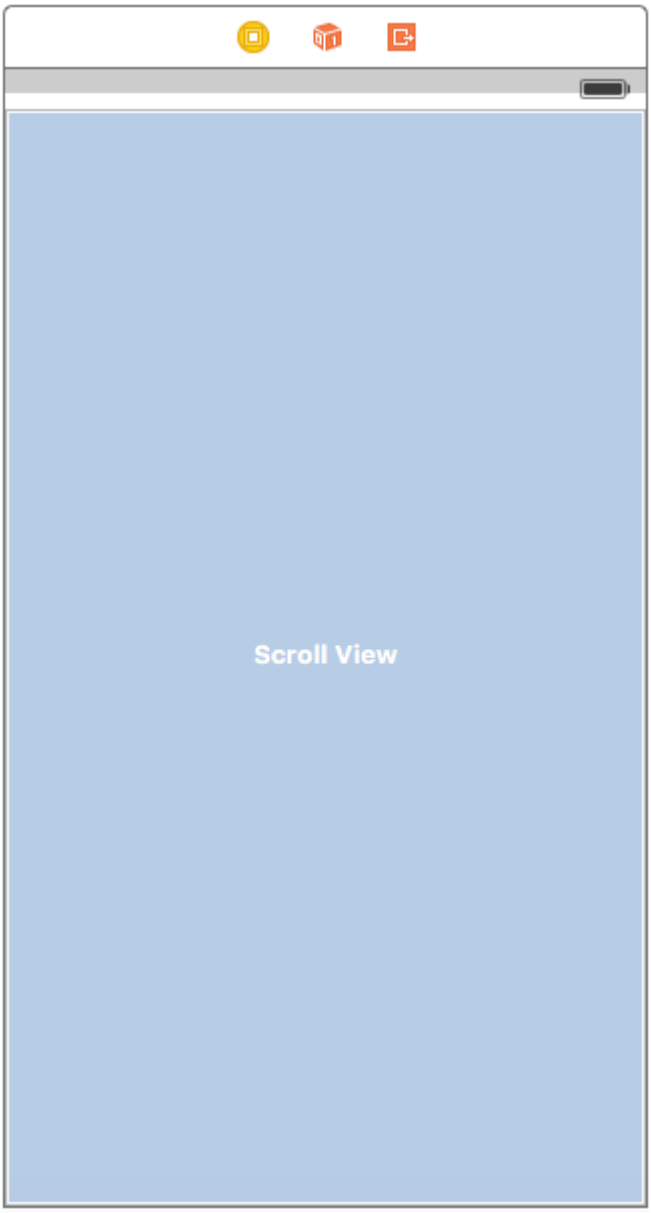
Objective-C

```
scrollview.contentSize = CGSizeMake(640, 800);
```

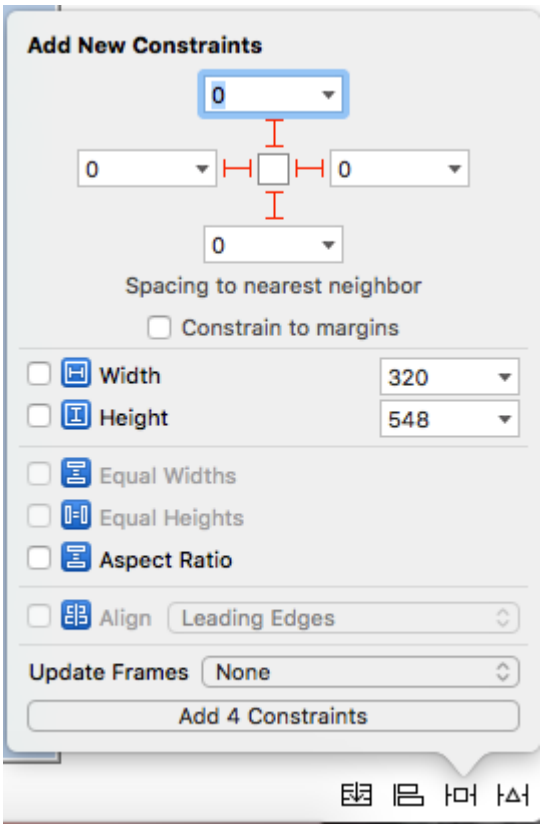
ScrollView с AutoLayout

Простые шаги для использования scrollview с автозапуском.

- Создание нового проекта с помощью приложения с одним представлением
- Выберите элемент управления представлением по умолчанию и измените его размер экрана на iPhone-4inch из инспектора атрибутов.
- Добавьте scrollview к виду viewcontroller следующим образом и установите цвет фона в синий



- Добавьте ограничения на него, как показано ниже.



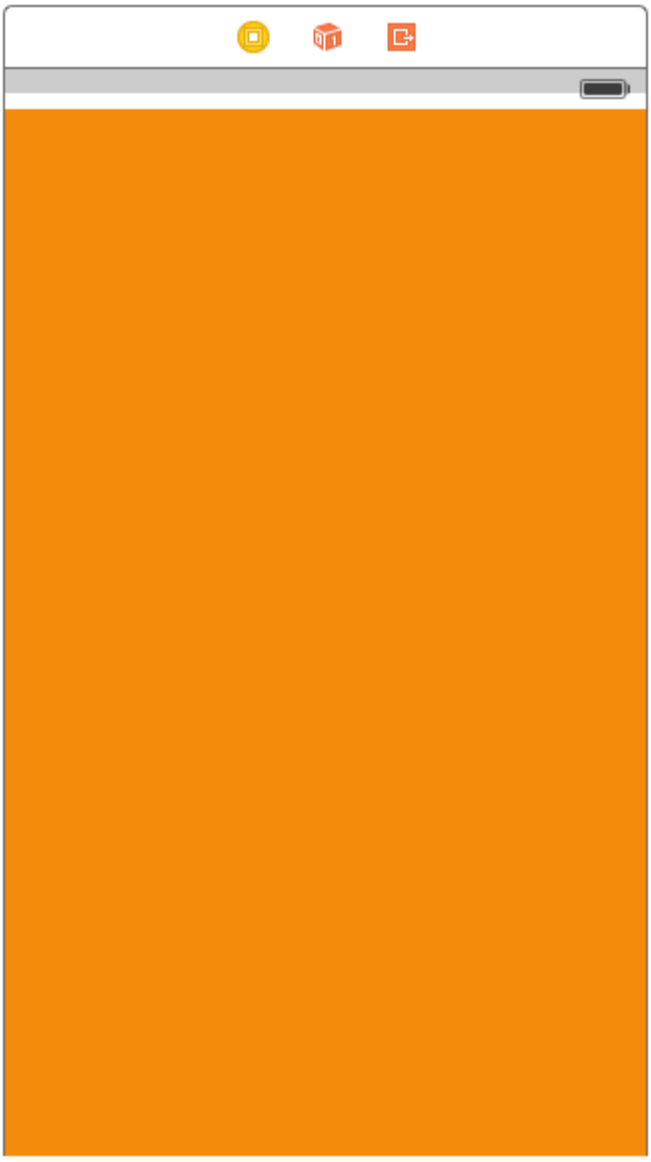
Что это будет делать, просто придерживайтесь каждого края scrollview для просмотра viewcontroller

Сценарий 1:

Теперь позвольте сказать, что наш контент огромен, и мы хотим, чтобы он прокручивался как по горизонтали, так и по вертикали.

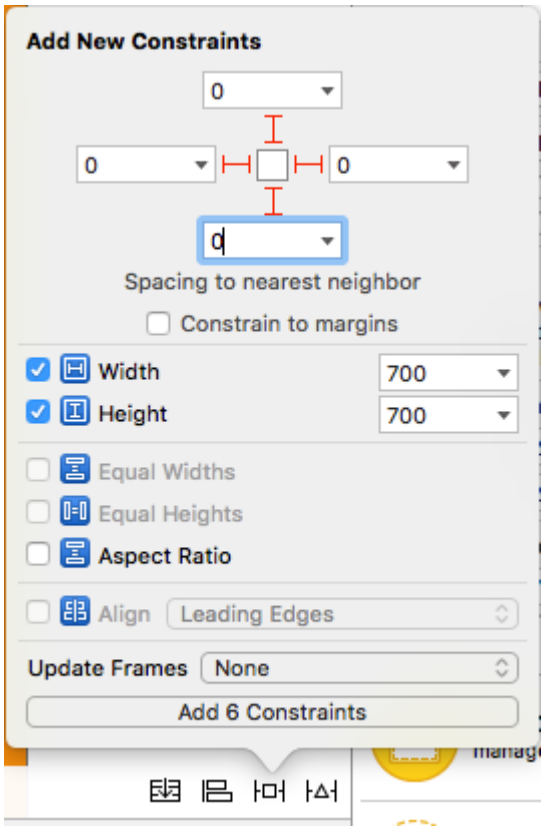
За это,

- Добавьте UIView в scrollview рамки (0,0700,700). Давайте дадим оранжевый цвет фона, чтобы идентифицировать его по-разному.



Далее идет важная часть, нам нужно ее прокручивать по горизонтали и вертикали.

- Выберите оранжевый вид и добавьте следующие ограничения



Позвольте мне объяснить, что мы сделали на предыдущем шаге.

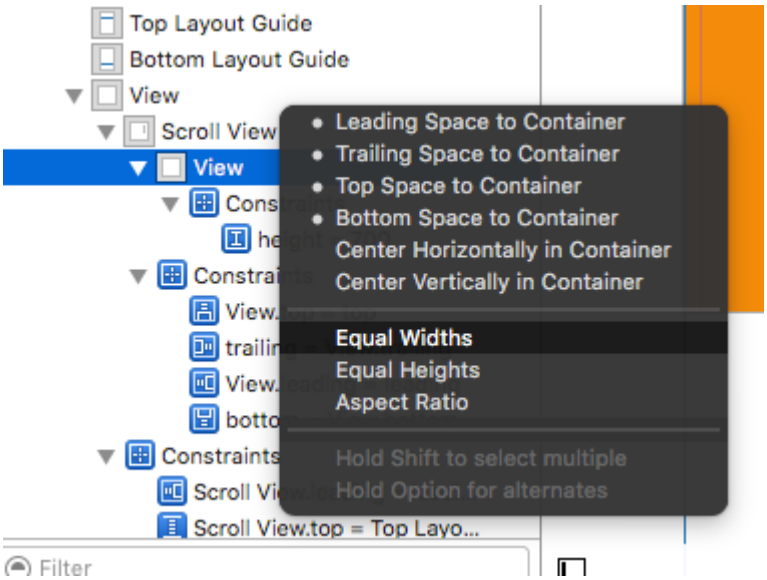
- Мы установили высоту и ширину до 700.
- Мы устанавливаем конечное пространство в scrollview = 0, которое сообщает scrollview, что контент горизонтально прокручивается.
- Мы устанавливаем нижнее пространство в scrollview = 0, которое сообщает прокручиванию, что контент вертикально прокручивается.

Теперь запустите проект и проверьте.

Сценарий 2. ***Давайте рассмотрим сценарий, в котором мы знаем, что ширина содержимого будет такой же, как ширина ширины прокрутки, но высота больше, чем scrollview.***

Выполните шаги для прокрутки содержимого по вертикали.

- Удалите ограничение ширины в приведенном выше случае.
- Измените ширину оранжевого вида в соответствии с шириной прокрутки.
- Ctrl-перетаскивание из оранжевого вида для просмотра прокрутки и добавление ограничения **ширины** .



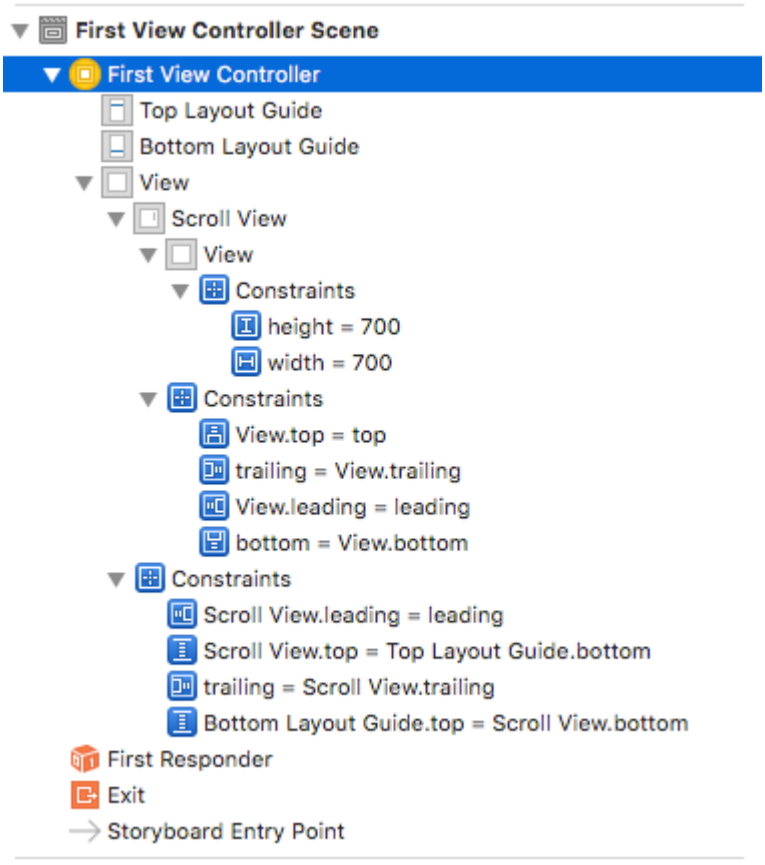
- И сделано !!! Просто запустите и проверьте, прокручивается ли она вертикально

Сценарий 3:

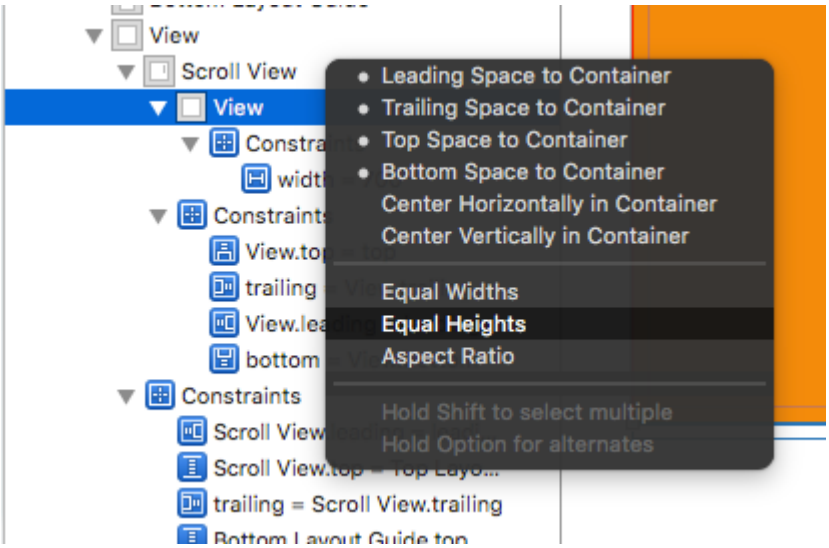
Теперь мы хотим прокручивать только по горизонтали, а не по вертикали.

Выполните шаги для горизонтальной прокрутки содержимого.

- Отмените все изменения для достижения ограничений, как показано ниже (т.е. **восстановить исходные ограничения, которые достигли вертикальной и горизонтальной прокрутки**)



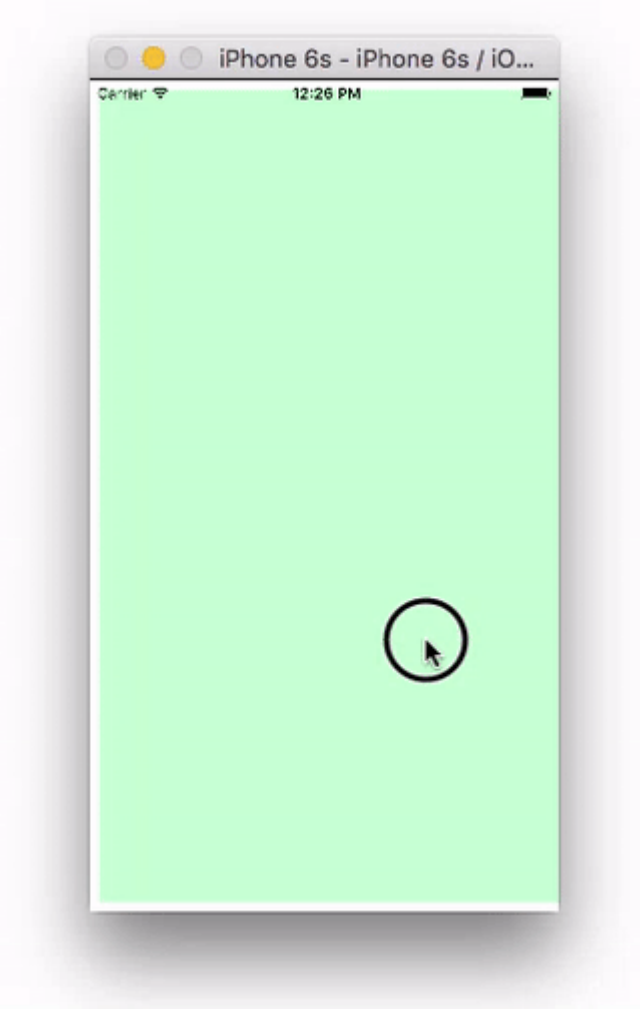
- Проверьте рамку оранжевого вида, которая должна быть (0,0700,700)
- Удалить ограничение высоты оранжевого вида.
- Измените высоту оранжевого вида, чтобы она соответствовала высоте прокрутки.
- Ctrl-перетащить с оранжевого вида, чтобы просмотреть прокрутку и добавить ограничение **равных высот** .



- И сделано !!! Просто запустите и проверьте, прокручивается ли она вертикально

Прокрутка содержимого с включенной автоматической компоновкой

Этот проект является самодостаточным примером, полностью выполненным в Interface Builder. Вы должны иметь возможность работать через 10 минут или меньше. Затем вы можете применить концепции, которые вы узнали, к своему собственному проекту.



Здесь я просто использую `UIView` но они могут представлять любой понравившийся вам вид (например, кнопка, ярлык и т. Д.). Я также выбрал горизонтальную прокрутку, потому что скриншоты раскадровки более компактны для этого формата. Однако принципы одинаковы для вертикальной прокрутки.

Ключевые идеи

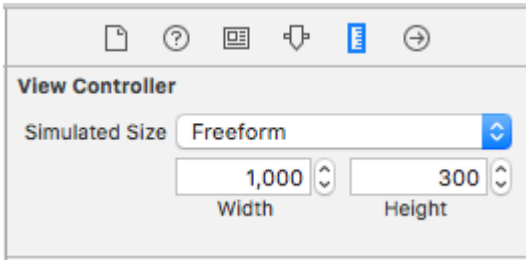
- `UIScrollView` должен использовать только одно подвью. Это «UIView», который служит для просмотра содержимого, чтобы содержать все, что вы хотите прокрутить.
- Сделайте представление содержимого, а *родительский* элемент прокрутки имеет равные высоты для горизонтальной прокрутки. (Равная ширина для вертикальной прокрутки)
- Убедитесь, что все прокручиваемое содержимое имеет заданную ширину и закреплено со всех сторон.

Начать новый проект

Это может быть просто приложение с одним представлением.

Раскадровка

В этом примере мы создадим горизонтальное представление прокрутки. Выберите контроллер просмотра, а затем выберите Freeform в инспекторе размеров. Сделайте ширину `1,000` и высоту `300` . Это просто дает нам комнату на раскадровке, чтобы добавить контент, который будет прокручиваться.

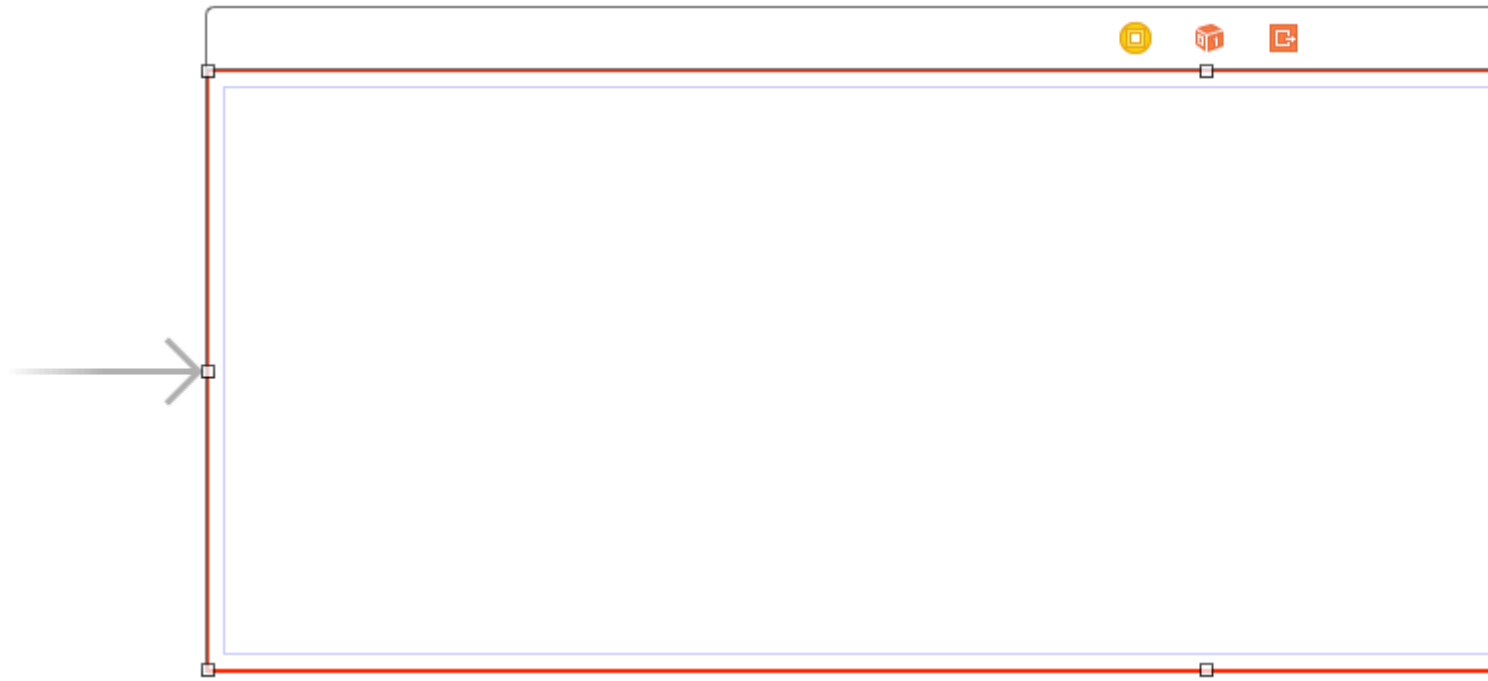


Добавить прокрутку

Добавьте `UIScrollView` и соедините все четыре стороны с корневым представлением контроллера вида.

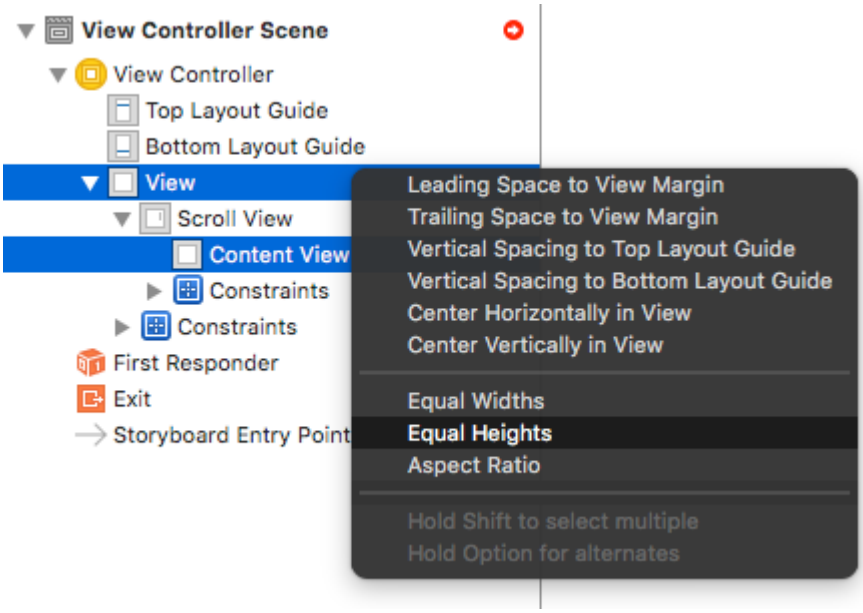
Добавление содержимого

Добавьте `UIView` в качестве представления в прокрутку. *Это ключ.* Не пытайтесь добавить много просмотров в прокрутку. Просто добавьте один `UIView` . Это будет просмотр вашего контента для других просмотров, которые вы хотите прокрутить. Подключите просмотр содержимого к списку прокрутки со всех четырех сторон.



Равные высоты

В настоящее время в структуре документа, `Command` нажмите как представление содержимого и *родительский просмотр* представления скроллинга для того , чтобы выбрать их оба. Затем установите высоту равным (Control </ kbd перетащить из представления содержимого в вид прокрутки>). *Это также ключ.* Поскольку мы прокручиваем по горизонтали, представление содержимого прокрутки не будет знать, насколько оно должно быть высоким, если мы не установим его таким образом.



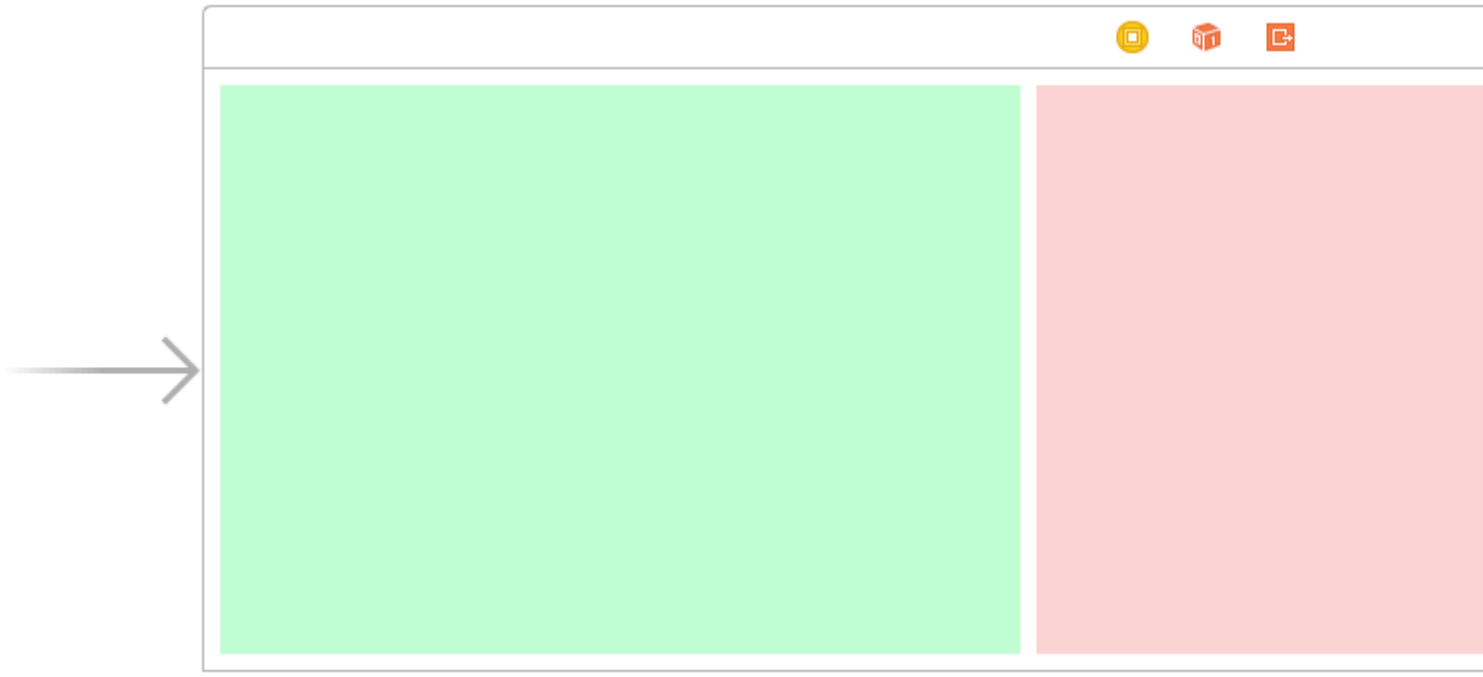
Замечания:

- Если бы мы производили прокрутку содержимого по вертикали, мы бы установили ширину представления содержимого равным ширине родительского представления

прокрутки.

Добавить контент

Добавьте три `UIView` и дайте им все ограничения. Я использовал 8 пунктов поля для всего.



Ограничения:

- Зеленый вид: нажмите верхний, левый и нижний края. Сделайте ширину 400.
- Красный вид: прикрепите верхний, левый и нижний края. Сделайте ширину 300.
- Фиолетовый вид: выровняйте все четыре края. Сделайте ширину любым оставшимся пространством (268 в этом случае).

Установка ограничений ширины также является ключевой, так что представление прокрутки знает, насколько широким будет просмотр содержимого.

Законченный

Это все. Теперь вы можете запустить свой проект. Он должен вести себя как прокручиваемое изображение в верхней части этого ответа.

Дальнейшее обучение

- iOS: как сделать работу AutoLayout на ScrollView
- Как настроить UIScrollView с автоматической компоновкой в интерфейсе Builder
- Учебник YouTube для видео: UIScrollView - как сохранить свои взгляды на экране

Включить / отключить прокрутку

Свойство `scrollEnabled` хранит `Boolean` значение, определяющее, включена ли прокрутка. Если значение этого свойства истинно / ДА, прокрутка включена, в противном случае нет. Значение по умолчанию - `true`

стриж

```
scrollview.isScrollEnabled = true
```

Objective-C

```
scrollview.scrollEnabled = YES;
```

Увеличение / уменьшение UIImageView

Создать экземпляр UIScrollView

```
let scrollview = UIScrollView.init(frame: self.view.bounds)
```

А затем установите следующие свойства:

```
scrollView.minimumZoomScale = 0.1
scrollView.maximumZoomScale = 4.0
scrollView.zoomScale = 1.0
scrollview.delegate = self as? UIScrollViewDelegate
```

Чтобы увеличить и уменьшить изображение, мы должны указать размер, который пользователь может увеличить и уменьшить. Мы делаем это, устанавливая значения `maximumZoomScale` свойств `minimumZoomScale` и `maximumZoomScale` . По умолчанию оба они установлены в 1.0.

И `zoomScale` до 1.0, которые определяют коэффициент масштабирования для минимального и максимального масштабирования.

Для поддержки масштабирования мы должны установить делегат для просмотра прокрутки. Объект делегата должен соответствовать протоколу `UIScrollViewDelegate` . Этот класс делегата должен реализовать метод `viewForZoomingInScrollView()` и вернуть представление для увеличения.

Измените свой ViewController как показано

```
class ViewController: UIViewController, UIScrollViewDelegate
```

Затем добавьте следующую функцию делегата в класс.

```
func viewForZoomingInScrollView(scrollView: UIScrollView) -> UIView? {
    return imageView
}
```

Теперь создайте экземпляр UIImageView

Сделайте эту переменную переменной класса

```
var imageView:UIImageView = UIImageView.init(image: UIImage.init(named: "someImage.jpg"))
```

А затем добавьте его в scrollview

```
scrollView?.addSubview(imageView)
```

Ссылка

- Руководство по программированию прокрутки для iOS
- Учебник UIScrollView

Обнаружение при завершении прокрутки UIScrollView с помощью методов делегирования

scrollViewDidEndDecelerating: это говорит делегату, что просмотр прокрутки завершил замедление движения прокрутки.

Цель C:

```
- (void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView {
    [self stoppedScrolling];
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate {
    if (!decelerate) {
        [self stoppedScrolling];
    }
}

- (void)stoppedScrolling {
    // done, do whatever
}
```

Swift:

```
func scrollViewDidEndDragging(scrollView: UIScrollView, willDecelerate decelerate: Bool) {
    if !decelerate {
        stoppedScrolling()
    }
```

```
}

func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
    stoppedScrolling()
}

func stoppedScrolling() {
    // done, do whatever
}
```

Ограничить направление прокрутки

Вы можете ограничить направления, которые пользователь может прокрутить, используя следующий код:

```
func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if scrollView.contentOffset.x != 0 {
        scrollView.contentOffset.x = 0
    }
}
```

Каждый раз, когда пользователь прокручивается по оси x, смещение содержимого scrollView возвращается к 0.
Вы можете, очевидно, изменить `x` s на `y` s и тем самым заблокировать направление только по горизонтали.

Вам также необходимо убедиться, что вы поместили этот код в метод `scrollViewDidScroll(_ scrollView: UIScrollView)` . В противном случае вы не сможете заставить его работать.

Кроме того, обязательно импортируйте `UIScrollViewDelegate` в объявление класса, например:

```
class ViewController: UIViewController, UIScrollViewDelegate
```

... и установите делегат scrollView для себя в каком-то методе, например `viewDidLoad(_:)`

```
scrollView.delegate = self
```

Прочитайте UIScrollView онлайн: https://riptutorial.com/ru/ios/topic/1575/uiscrollview

глава 91: UIScrollView AutoLayout

Examples

ScrollableController

При использовании Autolayout с `UIScrollView` он НЕ изменяет размер в зависимости от размера его содержимого или подсмотров.

Чтобы получить автоматическую прокрутку `UIScrollView` когда его содержимое становится слишком большим, чтобы соответствовать видимой области, нам нужно добавить `ContentView` и некоторые ограничения, которые позволяют `UIScrollView` определять размер его содержимого, а его ширину и высоту в родительском Посмотреть.

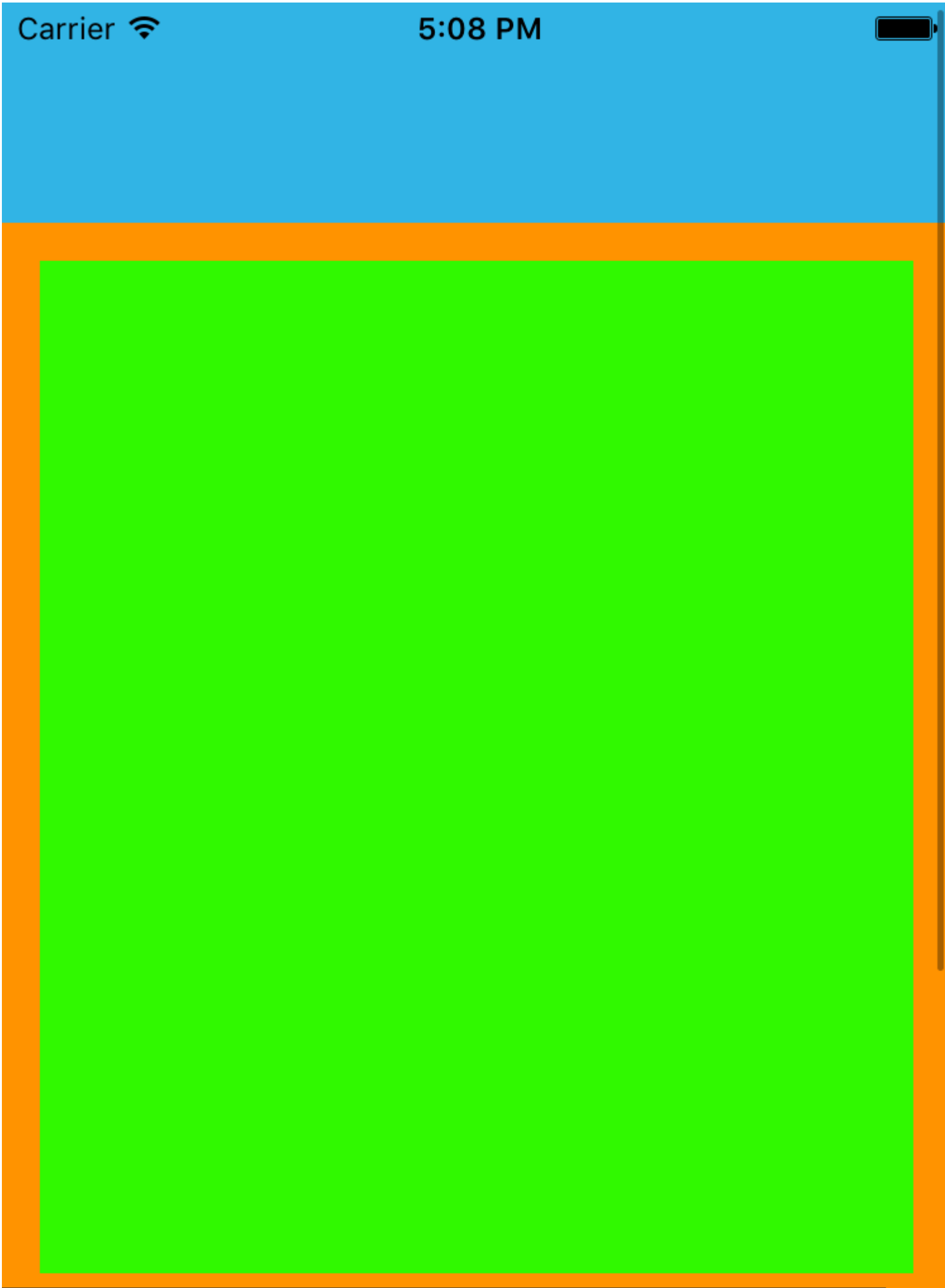
```
import Foundation
import UIKit

class ScrollableController : UIViewController {

    private var scrollView: UIScrollView!
    private var contentView: UIView!

    override func viewDidLoad() {
        super.viewDidLoad()

        //Setup
        self.initControls()
        self.setTheme()
        self.layoutScrollView()
        self.layoutContentView()

        //Add child views
        self.addChildViews()
    }

    func initControls() {
        self.scrollView = UIScrollView()
        self.contentView = UIView()
    }

    func setTheme() {
        self.scrollView.backgroundColor = UIColor.blue()
        self.contentView.backgroundColor = UIColor.orange()
    }

    func layoutScrollView() {
        self.view.addSubview(self.scrollView)

        let views: NSDictionary = ["scrollView": self.scrollView]
        var constraints = Array<String>()

        //Constrain the scrollView to our controller's self.view.
        constraints.append("H:|-0-[scrollView]-0-|")
        constraints.append("V:|-0-[scrollView]-0-|")
```

```
        for constraint in constraints {
            self.view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutFormatOptions(rawValue: 0), metrics: nil, views: views as!
[String : AnyObject]))
        }

        self.scrollView.translatesAutoresizingMaskIntoConstraints = false
    }

    func layoutContentView() {
        self.scrollView.addSubview(self.contentView)

        let views: NSDictionary = ["contentView": self.contentView, "view": self.view]
        var constraints = Array<String>()

        //Constrain the contentView to the scrollView.
        constraints.append("H:|-0-[contentView]-0-|")
        constraints.append("V:|-0-[contentView]-0-|")

        for constraint in constraints {
            self.scrollView.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutFormatOptions(rawValue: 0), metrics: nil, views: views as!
[String : AnyObject]))
        }

        //Disable Horizontal Scrolling by making the contentView EqualWidth with our
controller's self.view (ScrollView's parentView).
        self.view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
"H:[contentView(==view)]", options: NSLayoutFormatOptions(rawValue: 0), metrics: nil, views:
views as! [String : AnyObject]))

        self.contentView.translatesAutoresizingMaskIntoConstraints = false
    }

    func addChildViews() {
        //Init
        let greenView = UIView()
        let whiteView = UIView()

        //Theme
        greenView.backgroundColor = UIColor.green()
        whiteView.backgroundColor = UIColor.orange()

        //Layout -- Child views are added to the 'ContentView'
        self.contentView.addSubview(greenView)
        self.contentView.addSubview(whiteView)

        let views: NSDictionary = ["greenView": greenView, "whiteView": whiteView];
        var constraints = Array<String>()

        //Constrain the greenView to the contentView with a height of 400 and 15 spacing all
around.
        constraints.append("H:|-15-[greenView]-15-|")
        constraints.append("V:|-15-[greenView(400)]")

        //Constrain the whiteView below the greenView with 15 spacing all around and a height
of 500.
        constraints.append("H:|-15-[whiteView]-15-|")
        constraints.append("V:[greenView]-15-[whiteView(500)]-15-|")
```

```
        for constraint in constraints {
            self.contentView.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutFormatOptions(rawValue: 0), metrics: nil, views: views as!
[String : AnyObject]))
        }

        greenView.translatesAutoresizingMaskIntoConstraints = false
        whiteView.translatesAutoresizingMaskIntoConstraints = false
    }
}
```

Теперь мы видим, что greenView (высота 400) + whiteView (высота 500) больше нашего экрана. Это приведет к тому, что contentSize ScrollView будет расти, чтобы соответствовать представлениям BOTH, позволяя ему прокручивать по вертикали.

Мы отключили горизонтальную прокрутку с `EqualWidth` ограничения `EqualWidth` на `contentView` и `self.view`


Carrier
5:08 PM

с помощью раскадровки

При использовании scrollviews в раскадровке лучше рассчитать размер контента в соответствии с количеством представлений, присутствующим в scrollview, а не указывать размер содержимого программно со статическим значением.

Ниже приведены шаги по динамическому увеличению размера содержимого.

Шаг 1 :

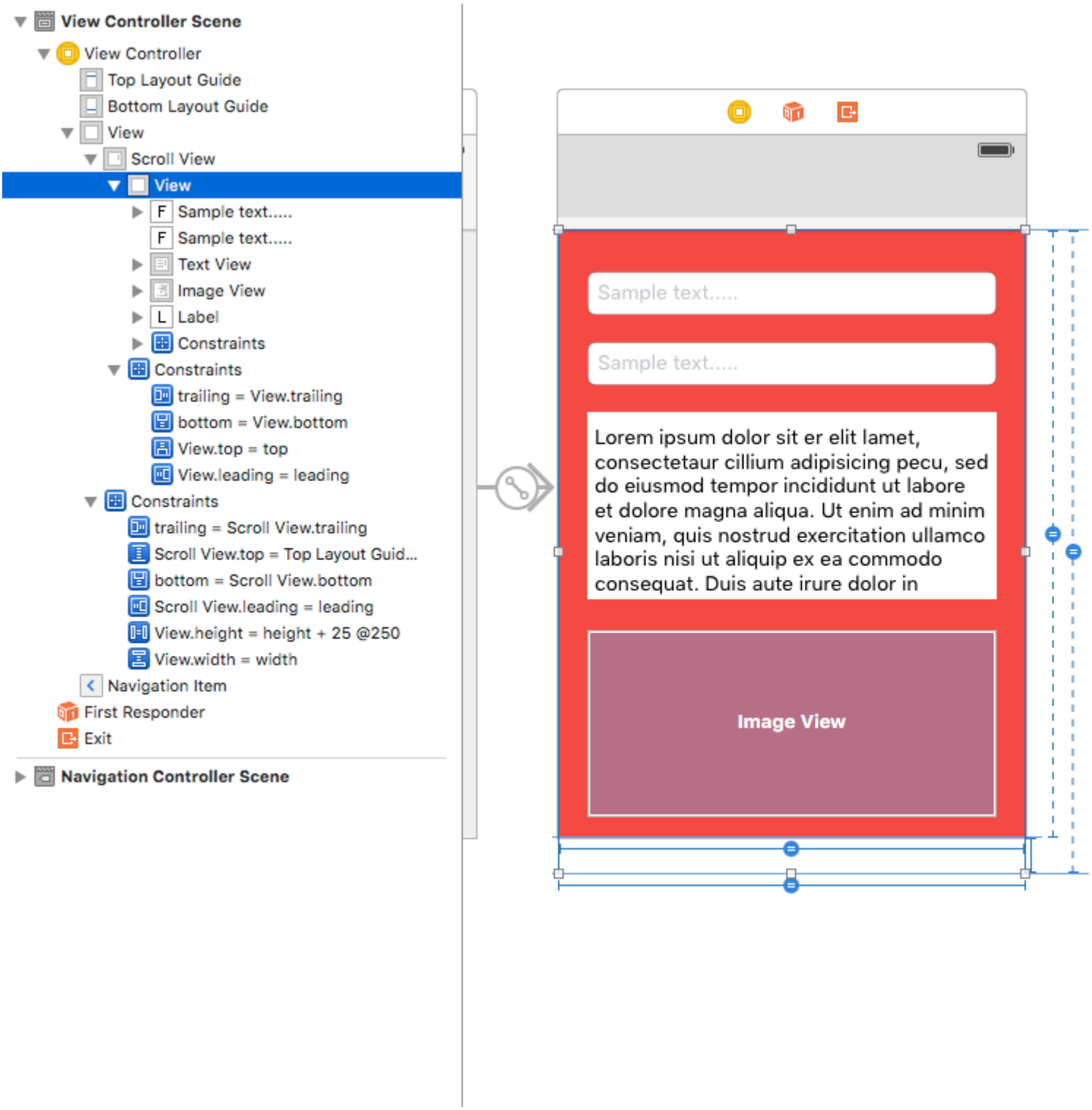
Добавьте Scrollview для просмотра в раскадровке и добавьте основные, конечные, верхние и нижние ограничения (Все значения равны нулю).

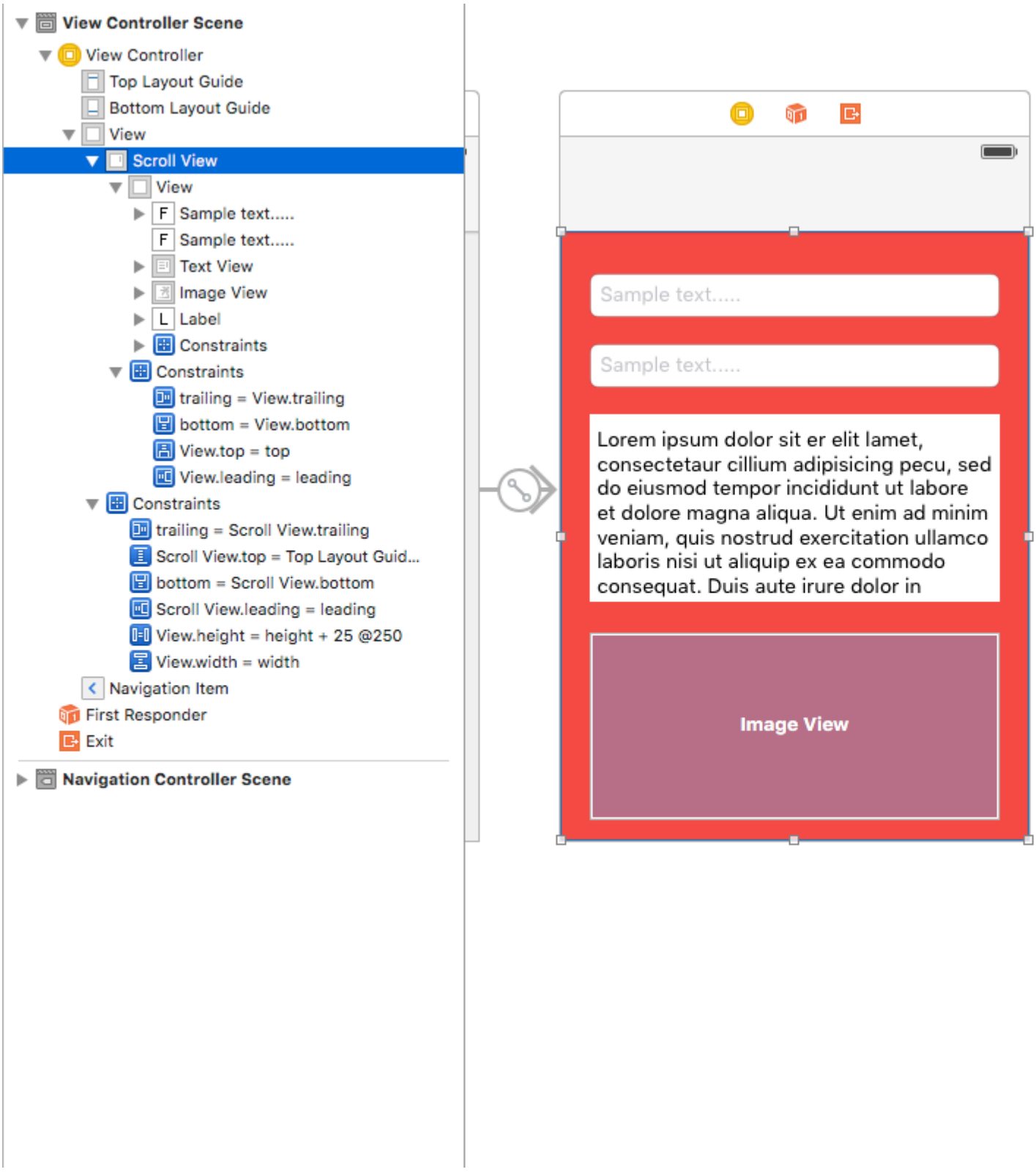
Шаг 2 :

Не добавляйте непосредственно виды, которые вам нужны, прямо в scrollview. Сначала добавьте одно представление в scrollview (это будет наш просмотр содержимого для всех элементов интерфейса). Добавьте ниже ограничения на это представление.

1. Ведущие, конечные, верхние и нижние ограничения (Все значения равны нулю).
2. Добавьте равную высоту, равную ширину в Основной вид (т. Е. Содержащий прокрутку). Для равной высоты установите приоритет на низкий. (Это важный шаг для настройки размера контента).
3. Высота этого содержимого будет соответствовать количеству просмотров, добавленных в представление. скажем, если вы добавили последнее представление, это одна метка, а его позиция Y - 420, а высота - 20, тогда ваш контент будет 440.

Шаг 3: добавьте ограничения ко всем представлениям, которые вы добавили в виде содержимого, в соответствии с вашим требованием.


View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Scroll View
View
Sample text.....
Sample text.....
Text View
Image View
Label
Constraints
Constraints
trailing = View.trailing
bottom = View.bottom
View.top = top
View.leading = leading
Constraints
trailing = Scroll View.trailing
Scroll View.top = Top Layout Guid...
bottom = Scroll View.bottom
Scroll View.leading = leading
View.height = height + 25 @250
View.width = width
Navigation Item
First Responder
Exit
Navigation Controller Scene
Sample text.....
Sample text.....
Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
Image View



Прочитайте UIScrollView AutoLayout онлайн:
https://riptutorial.com/ru/ios/topic/4671/uiscrollview-autolayout

глава 92: UIScrollView с дочерним элементом StackView

Examples

Сложный StackView в примере Scrollview

Ниже приведен пример того, что можно сделать с вложенными StackViews, предоставляя пользователю впечатление непрерывного прокрутки с использованием сложных элементов пользовательского интерфейса или выравниваний.

Предотвращение неоднозначной компоновки

Частый вопрос о StackViews внутри Scrollviews возникает из-за неоднозначности с предупреждениями / heigh на построителе интерфейса. Как объяснил этот ответ , необходимо:

1. Добавить в UIScrollView UIView (contentScrollView);
2. В этом contentScrollView установите верхнее, нижнее, левое и правое поля на 0
3. Установите также выравнивание по горизонтали и по вертикали;

Прокрутка к содержимому внутри вложенных StackViews

Большая информация о прокрутке состоит в том, чтобы определить смещение, необходимое для представления (например) **текстовое поле внутри StackView с внутри ScrollView** .

Если вы попытаетесь получить **позицию** `Textfield.frame.minY` **может быть 0** , потому что кадр minY учитывает только расстояние между элементом и вершиной StackView. Таким образом, вы должны **учитывать все другие родительские стеки** / **представления.**

Хорошим обходным путем для этого является:

1 - Внедрение расширения ScrollView

```
extension UIScrollView {

    func scrollToShowView(view: UIView){
        var offset = view.frame.minY
        var superview = view.superview
        while((superview  != nil)){
            offset += (superview?.frame.minY)!
            superview = superview?.superview
        }

        offset -= 100 //optional margin added on offset

        self.contentOffset = CGPoint.init(x: 0, y: offset)
    }

}
```

Это рассмотрит все родительское представление и суммирует необходимое смещение для scrollview, представляющего необходимый вид на экране (например, текстовое поле, которое не может оставаться за клавиатурой пользователя)

Пример использования:

```
func textViewDidBeginEditing(_ textView: UITextView) {
    self.contentScrollView.scrollToShowView(view: textView)
}
```

Прочитайте UIScrollView с дочерним элементом StackView онлайн:

https://riptutorial.com/ru/ios/topic/9404/uiscrollview-с-дочерним-элементом-stackview

глава 93: UISearchController

Синтаксис

- UISearchController (searchResultsController: UIViewController?) // Передача nil в качестве параметра, если контроллер обновления поиска также отображает содержимое, доступное для поиска.
- func updateSearchResults (для searchController: UISearchController) // Необходимый метод для реализации при использовании протокола UISearchResultsUpdating

параметры

параметр	подробности
`UISearchController.searchBar`	Панель поиска для установки в вашем интерфейсе. *(только для чтения)*
`UISearchController.searchResultsUpdater`	Объект, ответственный за обновление содержимого контроллера результатов поиска.
`UISearchController.isActive`	Представленное состояние интерфейса поиска.
`UISearchController.obscuresBackgroundDuringPresentation`	Логическое значение, указывающее, скрывается ли основное содержимое во время поиска.
`UISearchController.dimsBackgroundDuringPresentation`	Логическое значение, указывающее, является ли основной контент затухающим во время поиска.
`UISearchController.hidesNavigationBarDuringPresentation`	Логическое значение указывает, следует ли скрывать навигационную панель при поиске.
`UIViewController.definesPresentationContext`	Логическое значение, указывающее, рассматривается

параметр	подробности
	ли это представление диспетчера вида, когда контроллер вида или один из его потомков представляет контроллер вида.
`UIViewController.navigationItem.titleView`	Пользовательский вид отображается в центре панели навигации, когда приемник является верхним элементом, в котором может быть помещена панель поиска.
`UITableViewController.tableView.tableHeaderView`	Возвращает вспомогательный вид, который отображается над таблицей, в которой может быть помещена панель поиска.

замечания

Ссылка на UIKit Framework:

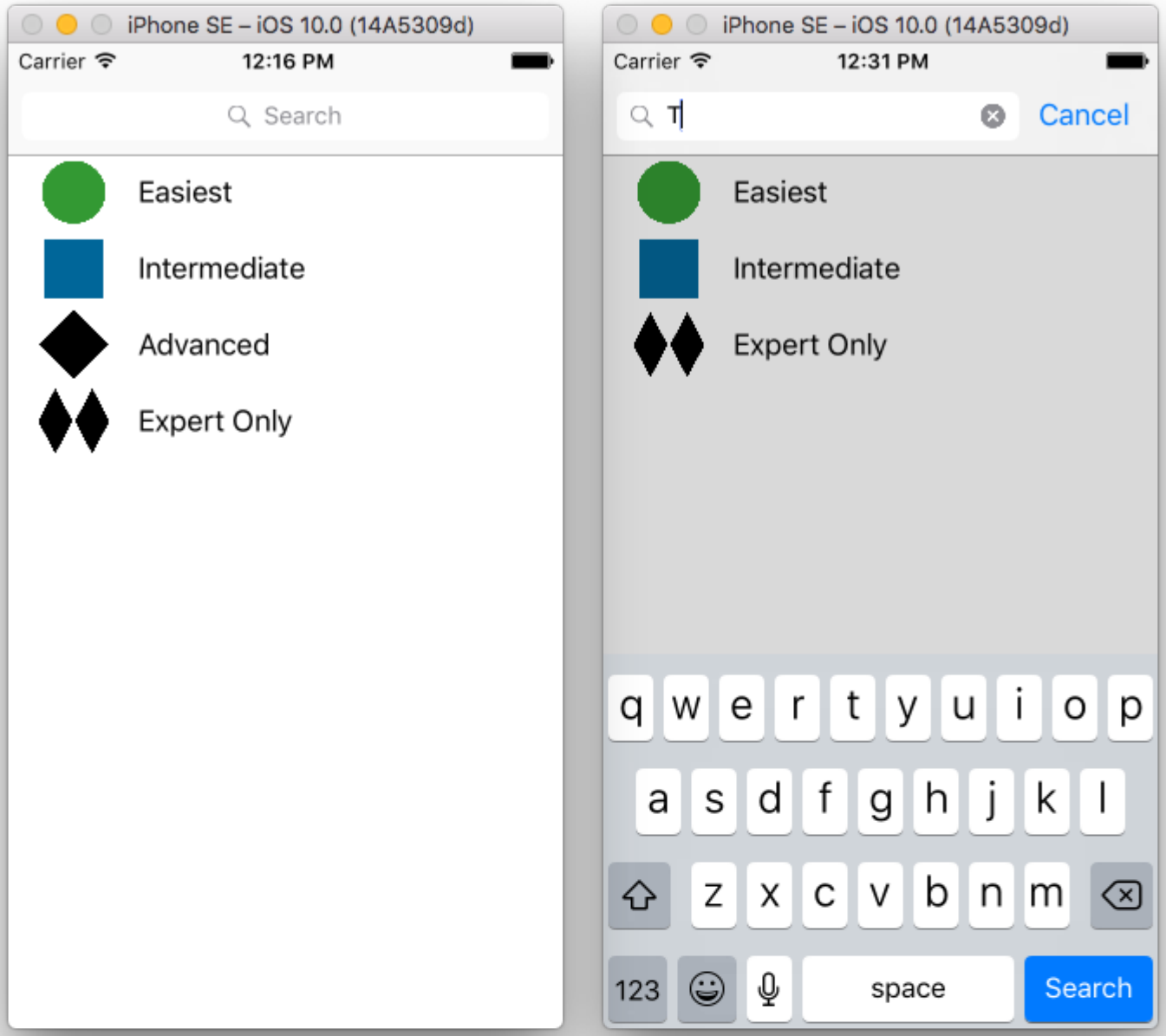
UISearchController

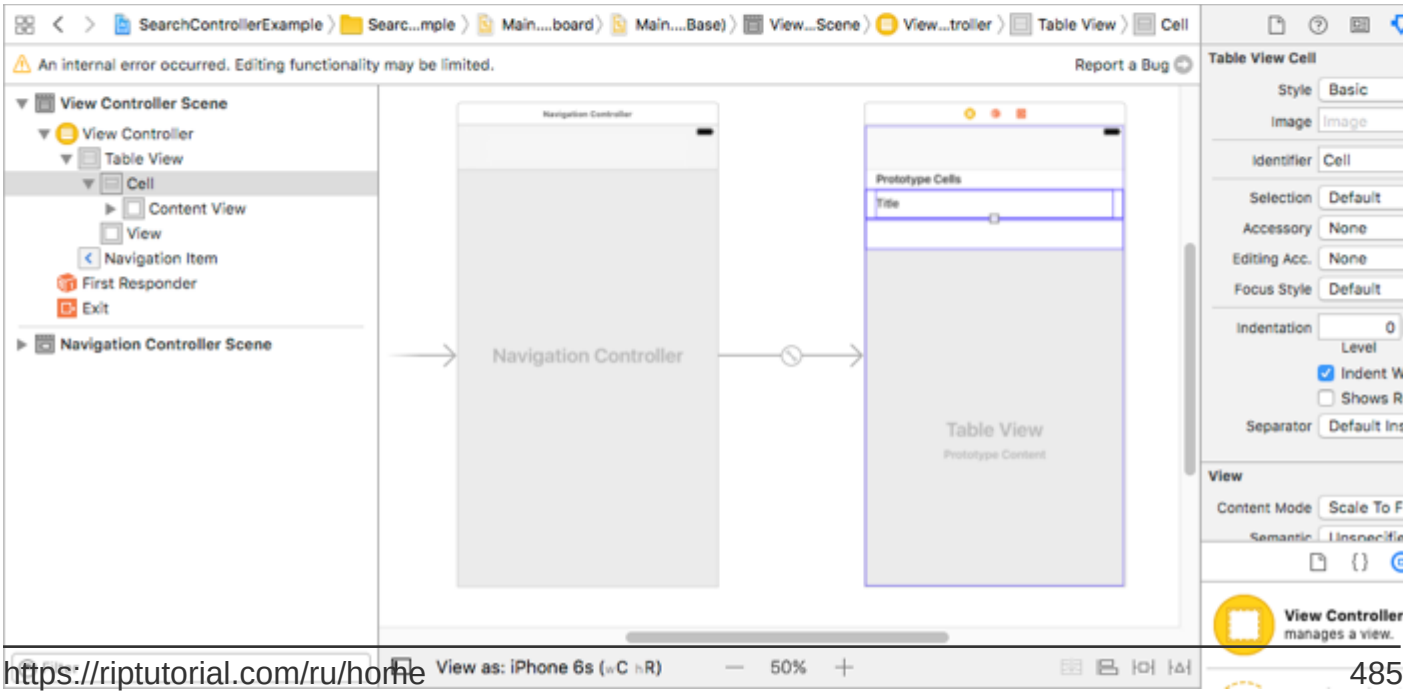
UISearchResultsUpdating

Examples

Панель поиска в заголовке панели навигации

В этом примере используется контроллер поиска для фильтрации данных внутри контроллера табличного представления. Строка поиска помещается в панель навигации, в которую встроена таблица.


iPhone SE – iOS 10.0 (14A5309d)
Carrier
12:16 PM
Search
Easiest
Intermediate
Advanced
Expert Only
iPhone SE – iOS 10.0 (14A5309d)
Carrier
12:31 PM
T
Cancel
Easiest
Intermediate
Expert Only
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
Search


SearchControllerExample
An internal error occurred. Editing functionality may be limited.
Report a Bug
View Controller Scene
View Controller
Table View
Cell
Content View
View
Navigation Item
First Responder
Exit
Navigation Controller Scene
Navigation Controller
Prototype Cells
Title
Table View
Table View Cell
Style
Basic
Image
Identifier
Cell
Selection
Default
Accessory
None
Editing Acc.
None
Focus Style
Default
Indentation
Level
Width
Indent While Editing
Separator
Default Insets
View
Content Mode
Scale To Fill
View Controller
manages a view.
View as: iPhone 6s
50%

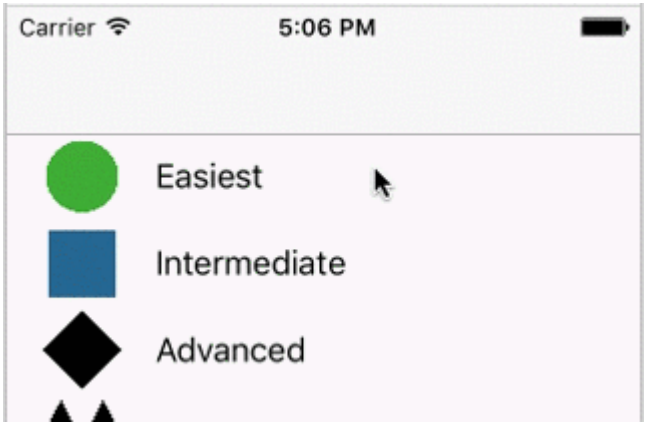
который содержит панель навигации). Затем установите наш пользовательский класс ViewController для наследования из `UITableViewController` и `UISearchResultsUpdating` протокол `UISearchResultsUpdating` .

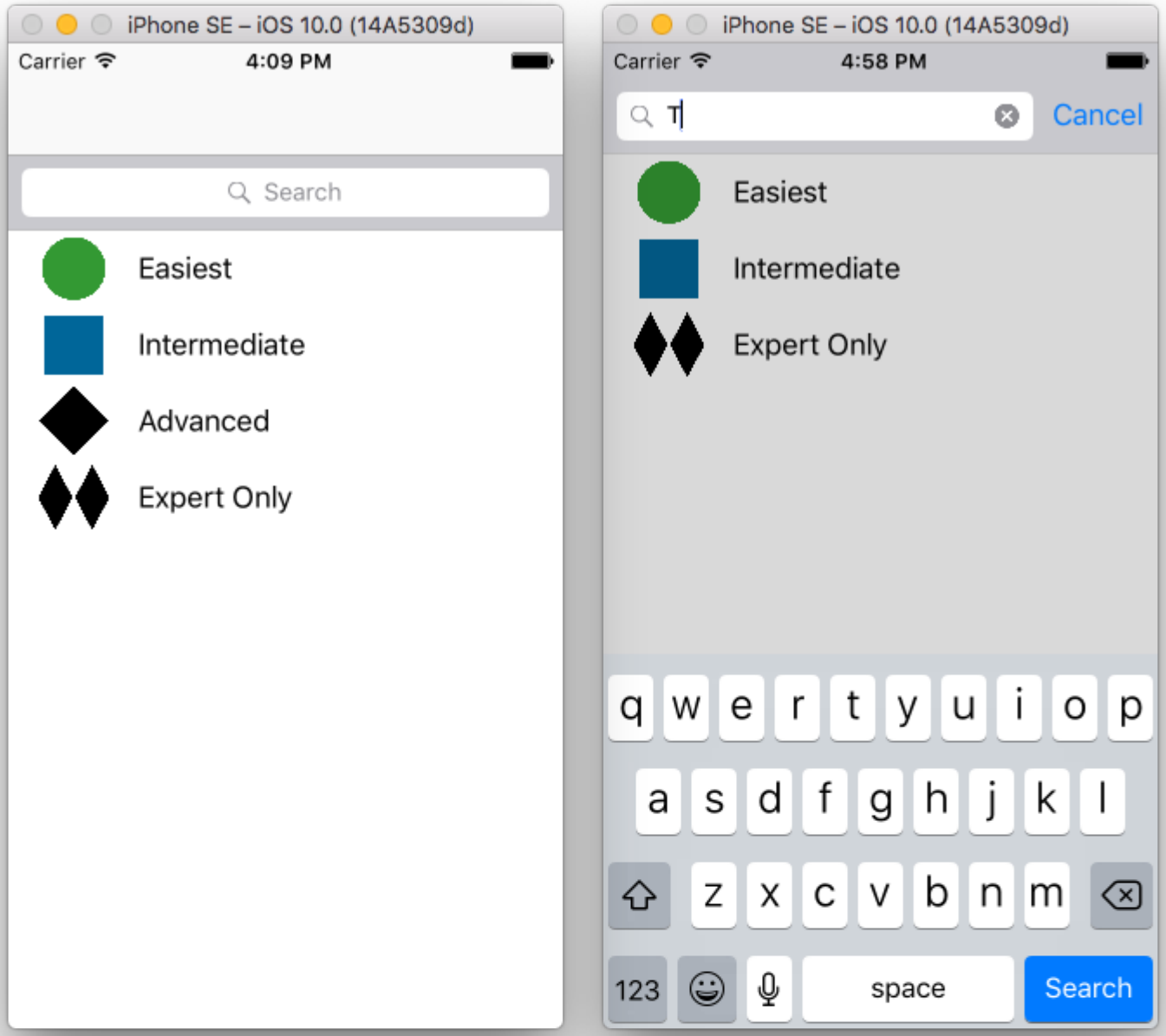
```
class ViewController: UITableViewController, UISearchResultsUpdating {

    let entries = [(title: "Easiest", image: "green_circle"),
                   (title: "Intermediate", image: "blue_square"),
                   (title: "Advanced", image: "black_diamond"),
                   (title: "Expert Only", image: "double_black_diamond")]

    // An empty tuple that will be updated with search results.
    var searchResults : [(title: String, image: String)] = []

    let searchController = UISearchController(searchResultsController: nil)

    override func viewDidLoad() {
        super.viewDidLoad()

        searchController.searchResultsUpdater = self
        self.definesPresentationContext = true

        // Place the search bar in the navigation item's title view.
        self.navigationItem.titleView = searchController.searchBar

        // Don't hide the navigation bar because the search bar is in it.
        searchController.hidesNavigationBarDuringPresentation = false
    }

    func filterContent(for searchText: String) {
        // Update the searchResults array with matches
        // in our entries based on the title value.
        searchResults = entries.filter({ (title: String, image: String) -> Bool in
            let match = title.range(of: searchText, options: .caseInsensitive)
            // Return the tuple if the range contains a match.
            return match != nil
        })
    }

    // MARK: - UISearchResultsUpdating method

    func updateSearchResults(for searchController: UISearchController) {
        // If the search bar contains text, filter our data with the string
        if let searchText = searchController.searchBar.text {
            filterContent(for: searchText)
            // Reload the table view with the search result data.
            tableView.reloadData()
        }
    }

    // MARK: - UITableViewController methods

    override func numberOfSections(in tableView: UITableView) -> Int { return 1 }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) ->
Int {
        // If the search bar is active, use the searchResults data.
        return searchController.isActive ? searchResults.count : entries.count
    }
```

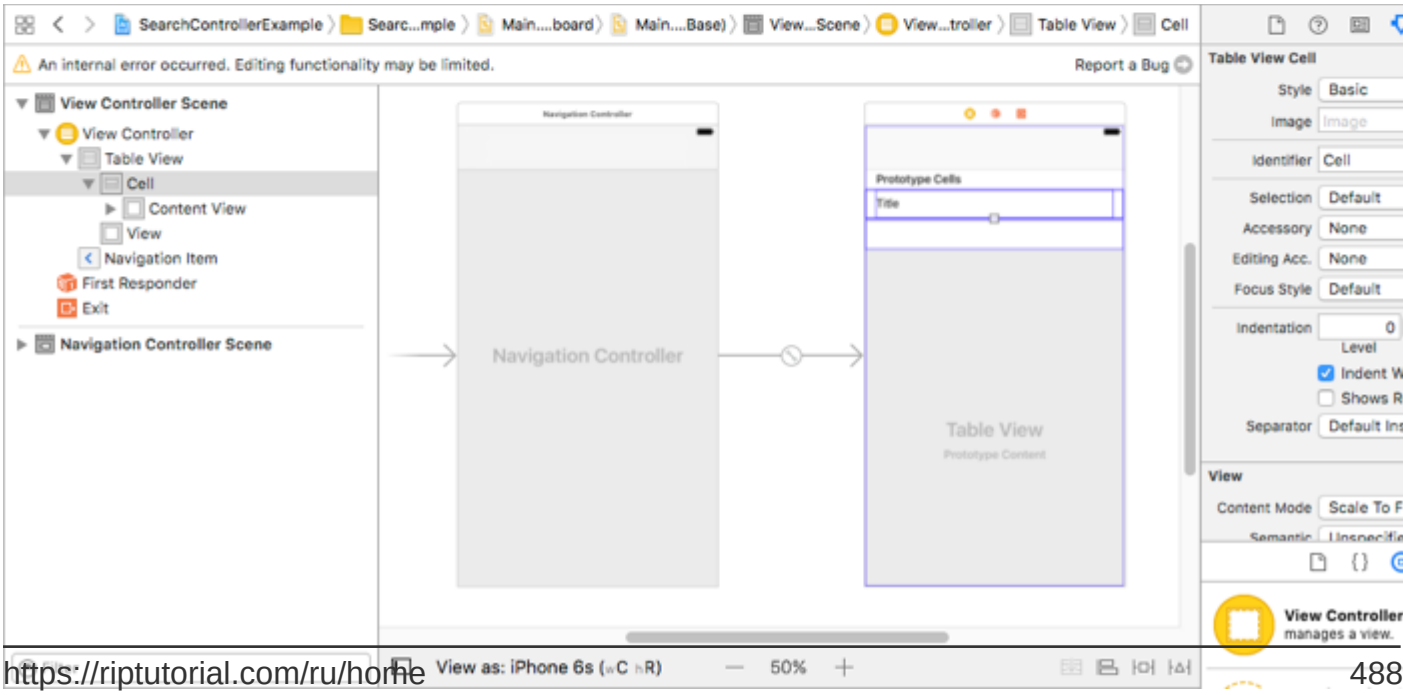
```
    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        // If the search bar is active, use the searchResults data.
        let entry = searchController.isActive ?
                    searchResults[indexPath.row] : entries[indexPath.row]

        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
        cell.textLabel?.text = entry.title
        cell.imageView?.image = UIImage(named: entry.image)
        return cell
    }
}
```

Панель поиска в заголовке таблицы

В этом примере используется контроллер поиска для фильтрации ячеек в контроллере табличного представления. Строка поиска помещается внутри заголовка представления таблицы. Содержимое табличного представления смещается с той же высотой, что и панель поиска, так что панель поиска сначала скрыта. При прокрутке вверх по верхнему краю представления таблицы открывается панель поиска. Затем, когда панель поиска становится активной, она скрывает панель навигации.




iPhone SE – iOS 10.0 (14A5309d)
Carrier
4:09 PM
Search
Easiest
Intermediate
Advanced
Expert Only
iPhone SE – iOS 10.0 (14A5309d)
Carrier
4:58 PM
Cancel
Easiest
Intermediate
Expert Only
space
Search


SearchControllerExample
An internal error occurred. Editing functionality may be limited.
Report a Bug
View Controller Scene
View Controller
Table View
Cell
Content View
View
Navigation Item
First Responder
Exit
Navigation Controller Scene
Navigation Controller
Prototype Cells
Title
Table View
Table View Cell
Style
Basic
Identifier
Cell
Selection
Default
Accessory
None
Editing Acc.
None
Focus Style
Default
Indentation
Indent While Editing
Separator
Default Insets
View
Content Mode
Scale To Fill
View Controller
manages a view.
View as: iPhone 6s
50%

который содержит панель навигации). Затем установите наш пользовательский класс ViewController для наследования из UITableViewController и примените протокол UISearchResultsUpdating.

```
class ViewController: UITableViewController, UISearchResultsUpdating {

    let entries = [(title: "Easiest", image: "green_circle"),
                   (title: "Intermediate", image: "blue_square"),
                   (title: "Advanced", image: "black_diamond"),
                   (title: "Expert Only", image: "double_black_diamond")]

    // An empty tuple that will be updated with search results.
    var searchResults : [(title: String, image: String)] = []

    let searchController = UISearchController(searchResultsController: nil)

    override func viewDidLoad() {
        super.viewDidLoad()

        searchController.searchResultsUpdater = self
        self.definesPresentationContext = true

        // Place the search bar in the table view's header.
        self.tableView.tableHeaderView = searchController.searchBar

        // Set the content offset to the height of the search bar's height
        // to hide it when the view is first presented.
        self.tableView.contentOffset = CGPoint(x: 0, y:
searchController.searchBar.frame.height)
    }

    func filterContent(for searchText: String) {
        // Update the searchResults array with matches
        // in our entries based on the title value.
        searchResults = entries.filter({ (title: String, image: String) -> Bool in
            let match = title.range(of: searchText, options: .caseInsensitive)
            // Return the tuple if the range contains a match.
            return match != nil
        })
    }

    // MARK: - UISearchResultsUpdating method

    func updateSearchResults(for searchController: UISearchController) {
        // If the search bar contains text, filter our data with the string
        if let searchText = searchController.searchBar.text {
            filterContent(for: searchText)
            // Reload the table view with the search result data.
            tableView.reloadData()
        }
    }

    // MARK: - UITableViewController methods

    override func numberOfSections(in tableView: UITableView) -> Int { return 1 }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) ->
Int {
        // If the search bar is active, use the searchResults data.
        return searchController.isActive ? searchResults.count : entries.count
```

```
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        // If the search bar is active, use the searchResults data.
        let entry = searchController.isActive ?
                    searchResults[indexPath.row] : entries[indexPath.row]

        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
        cell.textLabel?.text = entry.title
        cell.imageView?.image = UIImage(named: entry.image)
        return cell
    }
}
```

Реализация

Во-первых, сделайте свой класс совместимым с `UISearchResultsUpdating` .

```
class MyTableViewController: UITableViewController, UISearchResultsUpdating {}
```

Добавьте свойство контроллера поиска:

```
class MyTableViewController: UTableViewController, UISearchResultsUpdating {
    let searchController = UISearchController(searchResultsController: nil)
}
```

Добавьте панель поиска:

```
override func viewDidLoad() {
    super.viewDidLoad()

    searchController.searchResultsUpdater = self
    searchController.hidesNavigationBarDuringPresentation = false
    searchController.dimsBackgroundDuringPresentation = false
    searchController.searchBar.sizeToFit()
    self.tableView.tableHeaderView = searchController.searchBar
}
```

И, наконец, `updateSearchResultsForSearchController` метод `updateSearchResultsForSearchController` который поступает из протокола `UISearchResultsUpdating` :

```
func updateSearchResultsForSearchController(searchController: UISearchController) {

}
```

UISerachController в Objective-C

```
Delegate: UISearchBarDelegate, UISearchControllerDelegate, UISearchBarDelegate

@property (strong, nonatomic)  UISearchController *searchController;
```

```
- (void)searchBarConfiguration
{
    self.searchController = [[UISearchController alloc] initWithSearchResultsController:nil];
    self.searchController.searchBar.delegate = self;
    self.searchController.hidesNavigationBarDuringPresentation = NO;

    // Hides search bar initially.  When the user pulls down on the list, the search bar is
revealed.
    [self.tableView setContentOffset:CGPointMake(0,
self.searchController.searchBar.frame.size.height)];

    self.searchController.searchBar.backgroundColor = [UIColor DarkBlue];
    self.searchController.searchBar.tintColor = [UIColor DarkBlue];

    self.tableView.contentOffset = CGPointMake(0,
CGRectGetHeight(_searchController.searchBar.frame));
    self.tableView.tableHeaderView = _searchController.searchBar;
    _searchController.searchBar.delegate = self;
    _searchController.searchBar.showsCancelButton = YES;
    self.tapGestureRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(resetSearchbarAndTableView)];
    [self.view addGestureRecognizer:self.tapGestureRecognizer];

}

- (void)resetSearchbarAndTableView{
// Reload your tableview and resign keyboard.
}


- (void)searchBarCancelButtonClicked:(UISearchBar *)searchBar{
// Search cancelled
}
- (void)searchBarSearchButtonClicked:(UISearchBar *)searchBar{
// Implement filtration of your data as per your need using NSPredicate or else.
// then reload your data control like Tableview.
}
```

Прочитайте UISearchController онлайн:
https://riptutorial.com/ru/ios/topic/2813/uisearchcontroller

глава 94: UISegmentedControl

Вступление

Объектом UISegmentedControl является горизонтальное управление, состоящее из нескольких сегментов, причем каждый сегмент функционирует как дискретная кнопка. Сегментированный контроль дает компактное средство для группировки нескольких элементов управления.

Examples

Создание UISegmentedControl через код

1. Создайте новый экземпляр UISegmentedControl, заполненный тремя элементами (сегментами):

```
let mySegmentedControl = UISegmentedControl (items: ["One", "Two", "Three"])
```

2. Рамка установки;

```
mySegmentedControl.frame = CGRect(x: 0.0, y: 0.0, width: 300, height: 50)
```

3. Сделать выбор по умолчанию (а не индексировать сегменты на 0):

```
mySegmentedControl.selectedSegmentIndex = 0
```

4. Настроить цель:

```
mySegmentedControl.addTarget(self, action: #selector(segmentedValueChanged(_:)), for:
.valueChanged)
```

5 Изменено значение ручки:

```
func segmentedValueChanged(_ sender:UISegmentedControl!) {
    print("Selected Segment Index is : \(sender.selectedSegmentIndex)")
}
```

6. Добавить UISegmentedControl в иерархию представлений

```
yourView.addSubview(mySegmentedControl)
```

Прочитайте UISegmentedControl онлайн:
https://riptutorial.com/ru/ios/topic/9963/uisegmentedcontrol

глава 95: UISlider

Examples

UISlider

Objective-C

Объявить свойство ползунка в `ViewController.h` или в интерфейсе `ViewController.m`

```
@property (strong, nonatomic)UISlider *slider;

//Define frame of slider and add to view
CGRect frame = CGRectMake(0.0, 100.0, 320.0, 10.0);
UISlider *slider = [[UISlider alloc] initWithFrame:frame];
[slider addTarget:self action:@selector(sliderAction:)
forControlEvents:UIControlEventValueChanged];
[self.slider setBackgroundColor:[UIColor clearColor]];
self.slider.minimumValue = 0.0;
self.slider.maximumValue = 50.0;
//sending a NO/False would update the value of slider only when the user is no longer touching
the screen. Hence sending only the final value
self.slider.continuous = YES;
self.slider.value = 25.0;
[self.view addSubview slider];
```

Управление событием смены ползунка

```
- (IBAction)sliderAction:(id)sender {
    NSLog(@"Slider Value %f", sender.value);
}
```

Пример SWIFT

```
 let frame = CGRect(x: 0, y: 100, width: 320, height: 10)
 let slider = UISlider(frame: frame)
 slider.addTarget(self, action:  #selector(sliderAction), for: .valueChanged)
 slider.backgroundColor = .clear
 slider.minimumValue = 0.0
 slider.maximumValue = 50.0
 //sending a NO/False would update the value of slider only when the user is no longer
touching the screen. Hence sending only the final value
 slider.isContinuous = true
 slider.value = 25.0
 view.addSubview(slider)
```

Обработка события изменения слайдера

```
 func sliderAction(sender:UISlider!)
 {
    print("value--\(sender.value)")
```

```
}
```

Добавление пользовательского изображения большого пальца

Чтобы добавить пользовательское изображение для большого пальца слайдера, просто вызовите метод setThumbImage с вашим пользовательским изображением:

Swift 3.1:

```
let slider = UISlider()
let thumbImage = UIImage
slider.setThumbImage(thumbImage, for: .normal)
```

Прочитайте UISlider онлайн: https://riptutorial.com/ru/ios/topic/7402/uislider

глава 96: UISplitViewController

замечания

`UISplitViewController` - это контейнерный класс, например `UITabViewController` , `UINavigationController` . Он разделяет основной вид на два контроллера просмотра MasterViewController (PrimaryViewController) и detailViewController (SecondaryViewController). мы можем отправить массив с двумя контроллерами представлений, и Apple рекомендует использовать `UISplitViewController` в качестве rootviewcontroller для вашего приложения. Чтобы взаимодействовать между диспетчерами view, я использую `NSNotificationCenter` .

Examples

Мастер и деталь Просмотр взаимодействия с использованием делегатов в Objective C

`UISplitViewController` должен быть rootViewController вашего приложения.

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions
{
// Override point for customization after application launch.
self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]]
self.window.backgroundColor = [UIColor blackColor];
[self.window makeKeyAndVisible];
self.window.clipsToBounds = YES;
SplitViewController *spView = [[SplitViewController alloc]init];
self.window.rootViewController = spView;
[self.window makeKeyAndVisible];
return YES;
}
```

Просто создайте объект для `UISplitViewController` и настройте этот диспетчер представлений как rootviewcontroller для вашего приложения.

SplitViewController.h

```
#import <UIKit/UIKit.h>
#import "MasterViewController.h"
#import "DetailViewController.h"
@interface ViewController : UISplitViewController
{
DetailViewController *detailVC;
MasterViewController *masterVC;
NSMutableArray *array;
}
```

```
@end
```

`MasterViewController` всегда находится на левой стороне устройства, где вы можете установить ширину в `UISplitViewCOntroller` делегатов `DetailViewController` а `DetailViewController` - с правой стороны приложения

SplitViewController.m

```
#import "ViewController.h"
#define ANIMATION_LENGTH 0.3
@interface ViewController ()
@end

@implementation ViewController
- (void)viewDidLoad
{
[super viewDidLoad];
masterVC = [[MasterViewController alloc]init];
detailVC = [[DetailViewController alloc]init];
[masterVC setDetailDelegate:(id)detailVC];
NSArray *vcArray = [NSArray arrayWithObjects:masterVC, detailVC, nil];
self.preferredDisplayMode = UISplitViewControllerDisplayModeAutomatic;
self.viewControllers = vcArray;
self.delegate = (id)self;
self.presentsWithGesture = YES;
}
```

Созданный мастер и деталь ViewControllers добавляются в массив, который настроен на `self.viewControllers` в `UISplitViewController` . `self.preferredDisplayMode` - это режим, установленный для отображения основной и `DetailViewController` Apple Documentation для DisplayMode . `self.presentsWithGesture` позволяет жестом салфетки для отображения `MasterViewcontroller`

MasterViewController.h

```
#import <UIKit/UIKit.h>

@protocol DetailViewDelegate <NSObject>
@required
- (void)sendSelectedNavController:(UIViewController *)viewController;
@end

@interface MasterViewController : UIViewController
{
    UITableView *mainTableView;
    NSMutableArray *viewControllerArray;
}
@property (nonatomic, retain) id<DetailViewDelegate> detailDelegate;
@end
```

Создайте делегат `DetailViewDelegate` с помощью `sendSelectedNavController:(UIViewController *)viewController` для отправки `UIViewController` в `DetailViewcontroller` . Затем в `MasterViewController` mainTableView является табличным представлением в левой части. ViewControllerArray содержит все `UIViewControllers` которые должны отображаться в

DetailViewController

MasterViewController.m

```
#import "MasterViewController.h"

@implementation MasterViewController
@synthesize detailDelegate;

-(void)viewDidLoad
{
[super viewDidLoad];

UIViewController *dashBoardVC = [[UIViewController alloc]init];
[dashBoardVC.view setBackgroundColor:[UIColor redColor]];
UIViewController *inventVC = [[UIViewController alloc]init];
[inventVC.view setBackgroundColor:[UIColor whiteColor]];
UIViewController *alarmVC = [[UIViewController alloc]init];
[alarmVC.view setBackgroundColor: [UIColor purpleColor]];
UIViewController *scanDeviceVC = [[UIViewController alloc]init];
[scanDeviceVC.view setBackgroundColor:[UIColor cyanColor]];
UIViewController *serverDetailVC = [[UIViewController alloc]init];
[serverDetailVC.view setBackgroundColor: [UIColor whiteColor]];
viewControllerArray = [[NSMutableArray
alloc]initWithObjects:dashBoardVC,inventVC,alarmVC,scanDeviceVC,serverDetailVC,nil];
mainTableView = [[UITableView alloc]initWithFrame:CGRectMake(0, 50,self.view.frame.size.width,
self.view.frame.size.height-50) style:UITableViewStylePlain];
[mainTableView setDelegate:(id)self];
[mainTableView setDataSource:(id)self];
[mainTableView setSeparatorStyle:UITableViewCellSeparatorStyleNone];
[mainTableView setScrollsToTop:NO];
[self.view addSubview:mainTableView];
}

- (CGFloat)tableView:(UITableView *)tableView
heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    return 100;
}
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:    (NSInteger)section
{
    return [viewControllerArray count];
}

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;    //count of section
}

- (UITableViewCell *)tableView:(UITableView *)tableView
     cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    NSString *cellId = [NSString
stringWithFormat:@"Cell%li%ld",(long)indexPath.section,(long)indexPath.row];
UITableViewCell *cell =[tableView   dequeueReusableCellWithIdentifier:cellId];

if (cell == nil)
{
   cell = [[UITableViewCell alloc]initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:cellId];
}
```

```
[cell.contentView setBackgroundColor:[UIColor redColor]];
cell.textLabel.text =[NSString stringWithFormat:@"My VC at index %ld",(long)indexPath.row];
return cell;
}

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    [detailDelegate sendSelectedNavController:[viewControllerArray
objectAtIndex:indexPath.row]];
}

@end
```

Создайте `UIViewControllers` и `UIViewControllers` его в массив. Представление таблицы инициализируется, затем в методе `didSelectRowAtIndexPath` я отправляю `UIViewController` в `DetailViewController` с помощью `detailDelegate` с соответствующим `UIViewController` в массиве в качестве параметра

DetailViewController.h

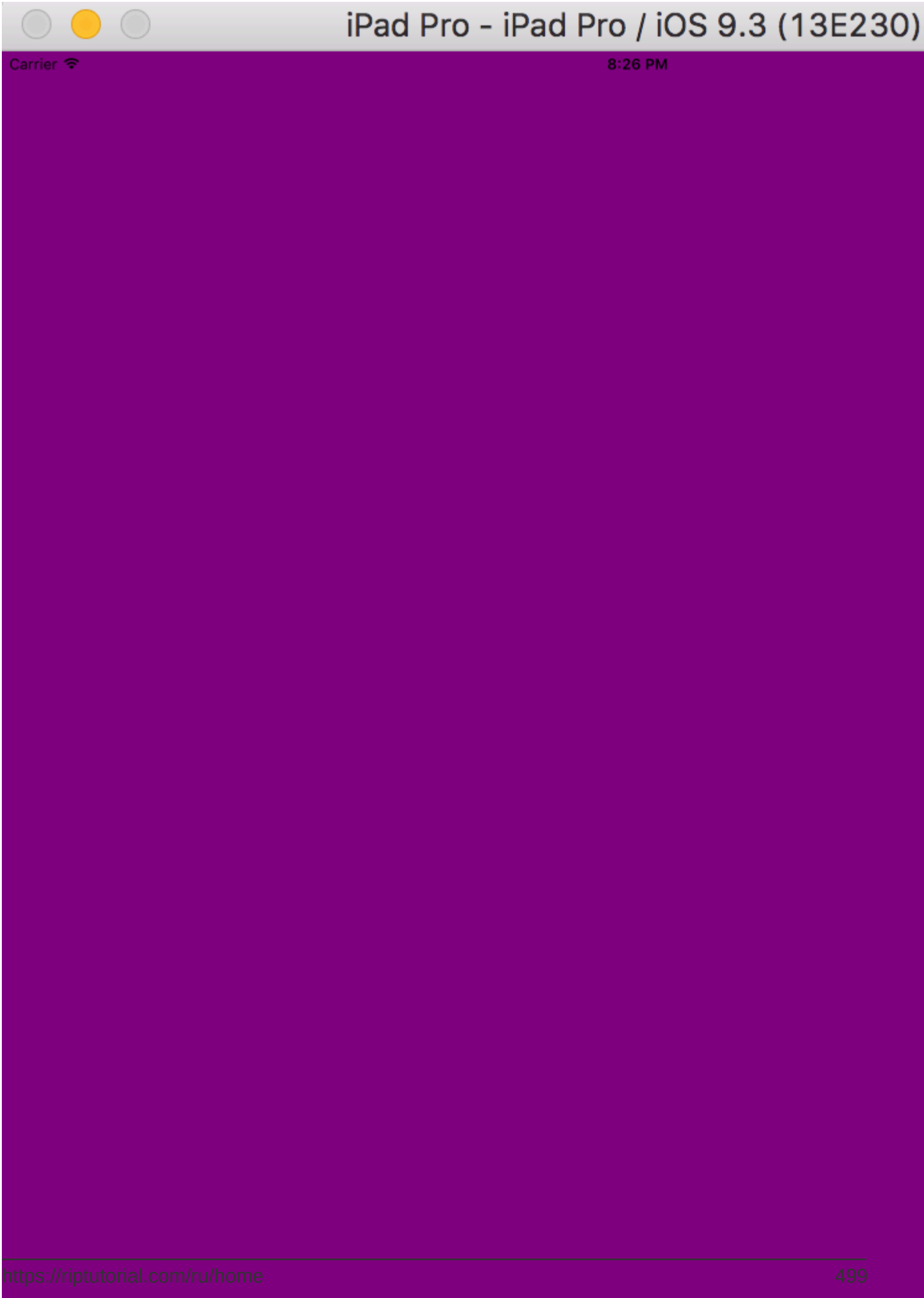
```
#import <UIKit/UIKit.h>

@interface DetailViewController : UIViewController<UICollectionViewDelegate>
{
    UIViewController *tempNav;
}
@end
```

DetailViewController.m

```
#import "DetailViewController.h"

@implementation DetailViewController
-(void)viewDidLoad
{
    [super viewDidLoad];
    [self.view setBackgroundColor:[UIColor whiteColor]];
}
-(void)sendSelectedNavController:(UIViewController *)navController
{
    NSArray *viewsToRemove = [self.view subviews];
    for (UIView *v in viewsToRemove) {
        [v removeFromSuperview];
    }
    tempNav = navController;
    [self.view addSubview:tempNav.view];
}
@end
```

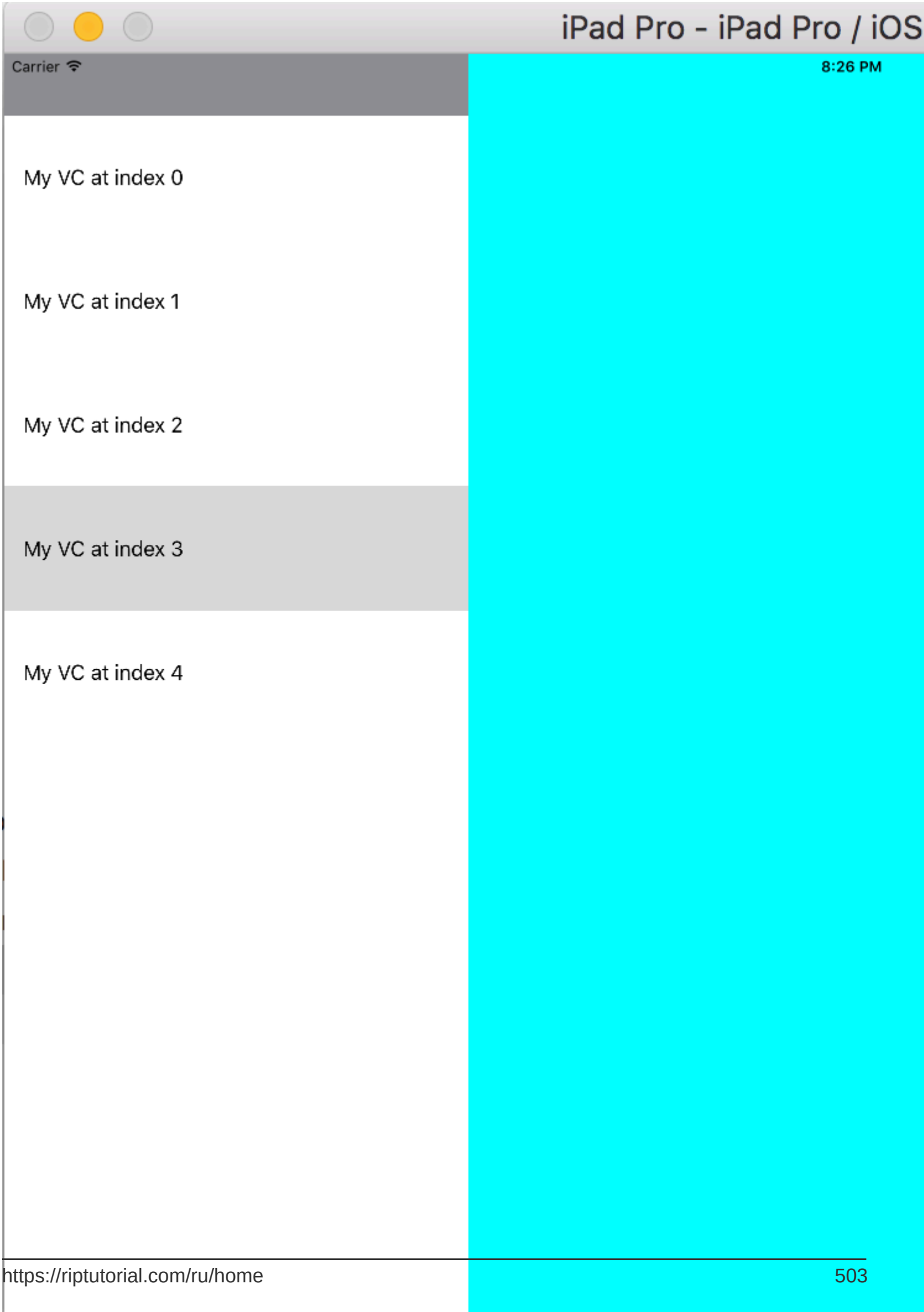
`sendSelectedNavController` объявляется здесь с удалением всех просмотров в `DetailViewController` и добавлением прошедшего `UIViewController` из `MasterViewController`

Добавление некоторых снимков экрана приложения


iPad Pro - iPad Pro / iOS 9.3 (13E230)
Carrier
8:26 PM

`preferredDisplayMode` как автоматический при прокрутке экрана, мы получаем `MasterViewController` как `MasterViewController` на `MasterViewController` ниже, но в ландшафтном режиме мы получаем как `MasterViewController` и `DetailViewController`


iPad Pro - iPad Pro / iOS 9.3 (13E230)
Carrier
8:26 PM
My VC at index 0
My VC at index 1
My VC at index 2
My VC at index 3
My VC at index 4


iPad Pro - iPad Pro / iOS
Carrier
8:26 PM
My VC at index 0
My VC at index 1
My VC at index 2
My VC at index 3
My VC at index 4

https://riptutorial.com/ru/ios/topic/4833/uisplitviewcontroller

глава 97: UISplitViewController

замечания

В iOS 8 и более поздних версиях вы можете использовать класс `UISplitViewController` на всех устройствах iOS, в предыдущих версиях iOS класс доступен только на iPad. `UISplitViewController` - это контейнерный класс, например `UITabViewController` , `UINavigationController` . Он разделяет основной вид на два `UIViewControllers` `masterViewController` (PrimaryViewController) и `detailViewController` (SecondaryViewController). мы можем отправить `NSArray` с двумя `UIViewControllers` и Apple рекомендует `UISplitViewController` как rootviewcontroller для вашего приложения. Для взаимодействия между `UIViewControllers` я использую `NSNotificationCenter` .

Examples

Взаимодействие между представлением мастера и деталями с использованием делегатов в Objective C

`UISplitViewController` должен быть контроллер корневого представления окна вашего приложения

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions
{
self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]]
self.window.backgroundColor = [UIColor blackColor];
[self.window makeKeyAndVisible];
self.window.clipsToBounds = YES;
SplitViewController *spView = [[SplitViewController alloc]init];
self.window.rootViewController = spView;
[self.window makeKeyAndVisible];
return YES;
}
```

Просто создайте объект для своего `UISplitVIewController` и установите его как rootViewController для своего приложения.

SplitViewController.h

```
#import <UIKit/UIKit.h>
#import "MasterViewController.h"
#import "DetailViewController.h"
@interface ViewController : UISplitViewController
{
DetailViewController *detailVC;
```

```
MasterViewController *masterVC;
NSMutableArray *array;
}
@end
```

`MasterViewController` - это `UIViewController` который установлен в левой части устройства, вы можете установить ширину в `UISplitViewController` используя `maximumPrimaryColumnWidth` и `DetailViewController` на правой стороне

SplitViewController.m

```
#import "ViewController.h"
#define ANIMATION_LENGTH 0.3
@interface ViewController ()
@end

@implementation ViewController
- (void)viewDidLoad
{
[super viewDidLoad];
masterVC = [[MasterViewController alloc]init];
detailVC = [[DetailViewController alloc]init];
[masterVC setDetailDelegate:(id)detailVC];
NSArray *vcArray = [NSArray arrayWithObjects:masterVC, detailVC, nil];
self.preferredDisplayMode = UISplitViewControllerDisplayModeAutomatic;
self.viewControllers = vcArray;
self.delegate = (id)self;
self.presentsWithGesture = YES;
}
```

Мастер и детализация `UIViewController` добавляются в `NSArray` который настроен на `self.viewControllers` . `self.preferredDisplayMode` - это режим, установленный для отображения `MasterViewController` и `DetailViewController` . `self.presentsWithGesture` позволяет жестом салфетки для отображения `MasterViewController`

MasterViewController.h

```
#import <UIKit/UIKit.h>

@protocol DetailViewDelegate <NSObject>
@required
- (void)sendSelectedNavController:(UIViewController *)viewController;
@end

@interface MasterViewController : UIViewController
{
    UITableView *mainTableView;
    NSMutableArray *viewControllerArray;
}
@property (nonatomic, retain) id<DetailViewDelegate> detailDelegate;
@end
```

Создание `DetailViewDelegate` делегата с `sendSelectedNavController` метод отправки `UIViewControllers` к `DetailViewController` . Затем в `MasterViewController` создается `UITableView` .

`ViewControllerArray` содержит все `UIViewControllers` которые должны отображаться в `DetailViewController`

MasterViewController.m

```
#import "MasterViewController.h"

@implementation MasterViewController
@synthesize detailDelegate;

-(void)viewDidLoad
{
[super viewDidLoad];

UIViewController *dashBoardVC = [[UIViewController alloc]init];
[dashBoardVC.view setBackgroundColor:[UIColor redColor]];
UIViewController *inventVC = [[UIViewController alloc]init];
[inventVC.view setBackgroundColor:[UIColor whiteColor]];
UIViewController *alarmVC = [[UIViewController alloc]init];
[alarmVC.view setBackgroundColor: [UIColor purpleColor]];
UIViewController *scanDeviceVC = [[UIViewController alloc]init];
[scanDeviceVC.view setBackgroundColor:[UIColor cyanColor]];
UIViewController *serverDetailVC = [[UIViewController alloc]init];
[serverDetailVC.view setBackgroundColor: [UIColor whiteColor]];
viewControllerArray = [[NSMutableArray
alloc]initWithObjects:dashBoardVC,inventVC,alarmVC,scanDeviceVC,serverDetailVC,nil];
mainTableView = [[UITableView alloc]initWithFrame:CGRectMake(0, 50,self.view.frame.size.width,
self.view.frame.size.height-50) style:UITableViewStylePlain];
[mainTableView setDelegate:(id)self];
[mainTableView setDataSource:(id)self];
[mainTableView setSeparatorStyle:UITableViewCellSeparatorStyleNone];
[mainTableView setScrollsToTop:NO];
[self.view addSubview:mainTableView];
}

- (CGFloat)tableView:(UITableView *)tableView
heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    return 100;
}
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:    (NSInteger)section
{
    return [viewControllerArray count];
}

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;    //count of section
}

- (UITableViewCell *)tableView:(UITableView *)tableView
     cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    NSString *cellId = [NSString
stringWithFormat:@"Cell%li%ld",(long)indexPath.section,(long)indexPath.row];
UITableViewCell *cell =[tableView   dequeueReusableCellWithIdentifier:cellId];

if (cell == nil)
{
   cell = [[UITableViewCell alloc]initWithStyle:UITableViewCellStyleDefault
```

```
reuseIdentifier:cellId];
}
[cell.contentView setBackgroundColor:[UIColor redColor]];
cell.textLabel.text =[NSString stringWithFormat:@"My VC at index %ld",(long)indexPath.row];
return cell;
}

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    [detailDelegate sendSelectedNavController:[viewControllerArray
objectAtIndex:indexPath.row]];
}

@end
```

Создал некоторый `UIViewController` и добавил его в `NSMutableArray` . `UITableView` инициализируется затем по методу `didselectrowatindexpath` Я отправляю `UIViewController` в `DetailViewController` используя делегат `detailDelegate` с соответствующим `UIViewController` в `NSMutableArray` в качестве параметра

DetailViewController.h

```
#import <UIKit/UIKit.h>

@interface DetailViewController : UIViewController<UICollectionViewDelegate>
{
    UIViewController *tempNav;
}
@end
```

DetailViewController.m

```
#import "DetailViewController.h"

@implementation DetailViewController
-(void)viewDidLoad
{
    [super viewDidLoad];
    [self.view setBackgroundColor:[UIColor whiteColor]];
}
-(void)sendSelectedNavController:(UIViewController *)navController
{
    NSArray *viewsToRemove = [self.view subviews];
    for (UIView *v in viewsToRemove) {
        [v removeFromSuperview];
    }
    tempNav = navController;
    [self.view addSubview:tempNav.view];
}
@end
```

Здесь `sendSelectedNavController` с удалением всех `UIView` в `DetailViewController` и добавлением прошедшего `UIViewController` из `MasterViewController` .

Прочитайте UISplitViewController онлайн:
https://riptutorial.com/ru/ios/topic/4844/uisplitviewcontroller

глава 98: UIStackView

Examples

Создание горизонтального представления стека программно

Swift 3

```
let stackView = UIStackView()
stackView.axis = .horizontal
stackView.alignment = .fill // .leading .firstBaseline .center .trailing .lastBaseline
stackView.distribution = .fill // .fillEqually .fillProportionally .equalSpacing
.equalCentering

let label = UILabel()
label.text = "Text"
stackView.addArrangedSubview(label)
// for horizontal stack view, you might want to add width constraint to label or whatever view
you're adding.
```

стриж

```
let stackView = UIStackView()
stackView.axis = .Horizontal
stackView.alignment = .Fill // .Leading .FirstBaseline .Center .Trailing .LastBaseline
stackView.distribution = .Fill // .FillEqually .FillProportionally .EqualSpacing
.EqualCentering

let label = UILabel(frame: CGRectZero)
label.text = "Label"
stackView.addArrangedSubview(label)
// for horizontal stack view, you might want to add width constraint to label or whatever view
you're adding.
```

Objective-C

```
UIStackView *stackView = [[UIStackView alloc] init];
stackView.axis = UILayoutConstraintAxisHorizontal;
stackView.alignment = UIStackViewAlignmentFill; //UIStackViewAlignmentLeading,
UIStackViewAlignmentFirstBaseline, UIStackViewAlignmentCenter, UIStackViewAlignmentTrailing,
UIStackViewAlignmentLastBaseline
stackView.distribution = UIStackViewDistributionFill; //UIStackViewDistributionFillEqually,
UIStackViewDistributionFillProportionally, UIStackViewDistributionEqualSpacing,
UIStackViewDistributionEqualCentering

UILabel *label = [[UILabel alloc] initWithFrame:CGRectZero];
label.text = @"Label";
[stackView addArrangedSubview:label];
//For horizontal stack view, you might want to add a width constraint to your label or
whatever view you are adding.
```

Создавать вертикальное представление стека программно

стриж

```
let stackView = UIStackView()
stackView.axis = .Vertical
stackView.alignment = .Fill // .Leading .FirstBaseline .Center .Trailing .LastBaseline
stackView.distribution = .Fill // .FillEqually .FillProportionally .EqualSpacing
.EqualCentering

let label = UILabel(frame: CGRectZero)
label.text = "Label"
stackView.addArrangedSubview(label)
// for vertical stack view, you might want to add height constraint to label or whatever view
you're adding.
```

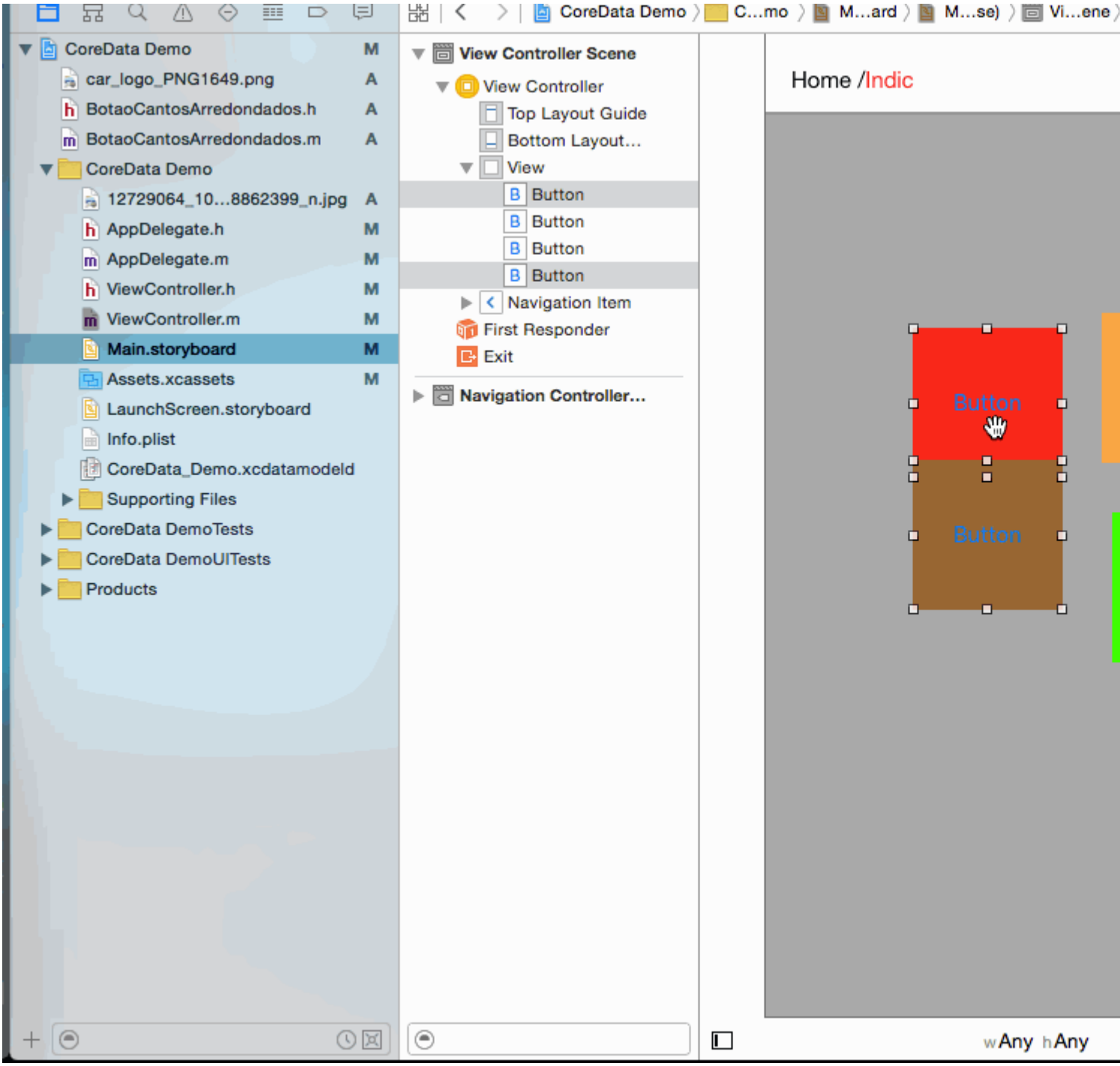
Objective-C

```
UIStackView *stackView = [[UIStackView alloc] init];
stackView.axis = UILayoutConstraintAxisVertical;
stackView.alignment = UIStackViewAlignmentFill; //UIStackViewAlignmentLeading,
UIStackViewAlignmentFirstBaseline, UIStackViewAlignmentCenter, UIStackViewAlignmentTrailing,
UIStackViewAlignmentLastBaseline
stackView.distribution = UIStackViewDistributionFill; //UIStackViewDistributionFillEqually,
UIStackViewDistributionFillProportionally, UIStackViewDistributionEqualSpacing,
UIStackViewDistributionEqualCentering

UILabel *label = [[UILabel alloc] initWithFrame:CGRectZero];
label.text = @"Label";
[stackView addArrangedSubview:label];
//For vertical stack view, you might want to add a height constraint to your label or whatever
view you are adding.
```

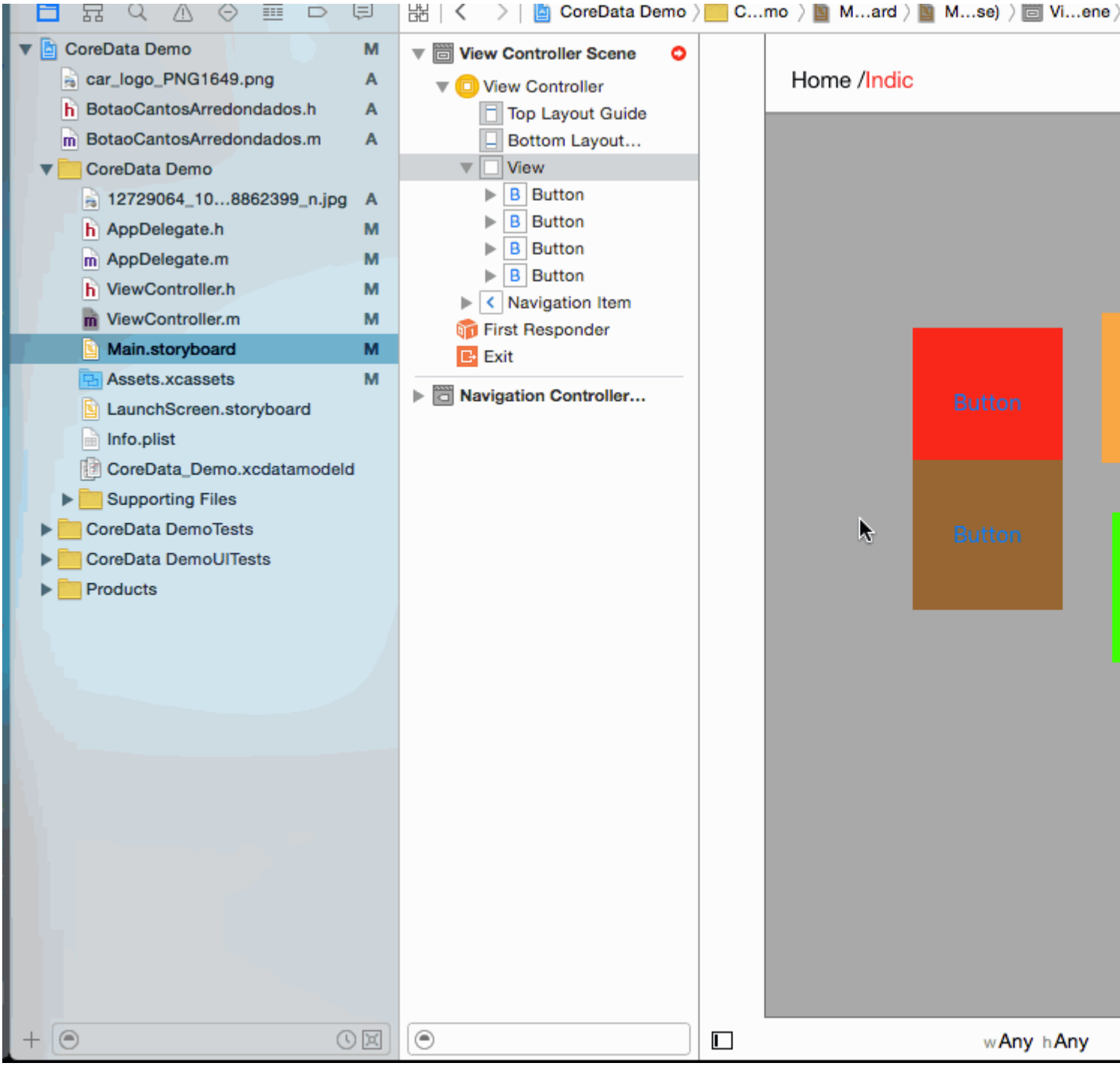
Центровые кнопки с UIStackview

Шаг 1: - возьмите 4 кнопки в своей раскадровке. Button1, Button2, Button 3, Button4

Шаг 2: - Дайте фиксированную высоту и ширину для всех кнопок.

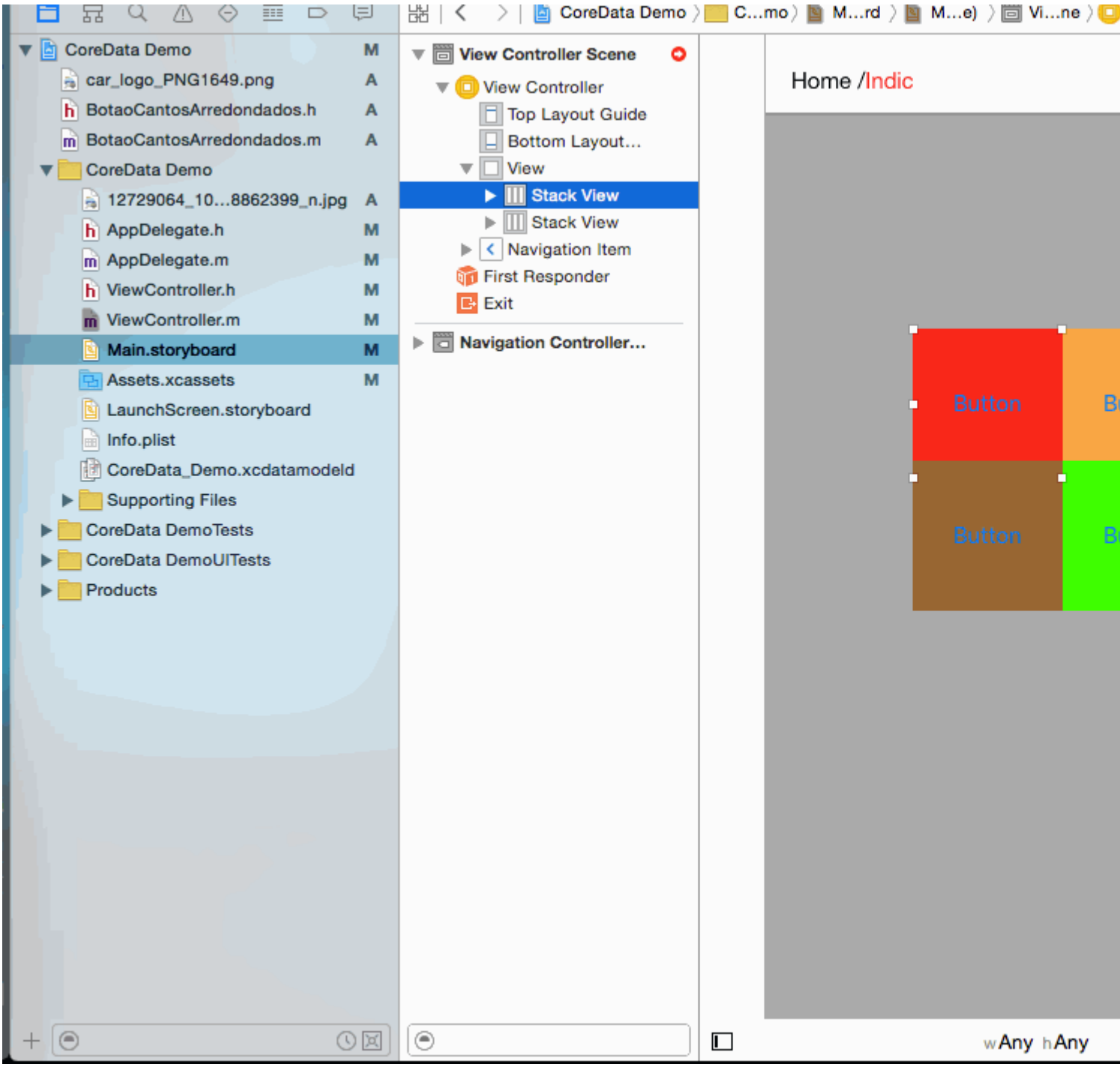


Шаг 3: - Все пары 2 - 2 кнопок в 2 stackview.

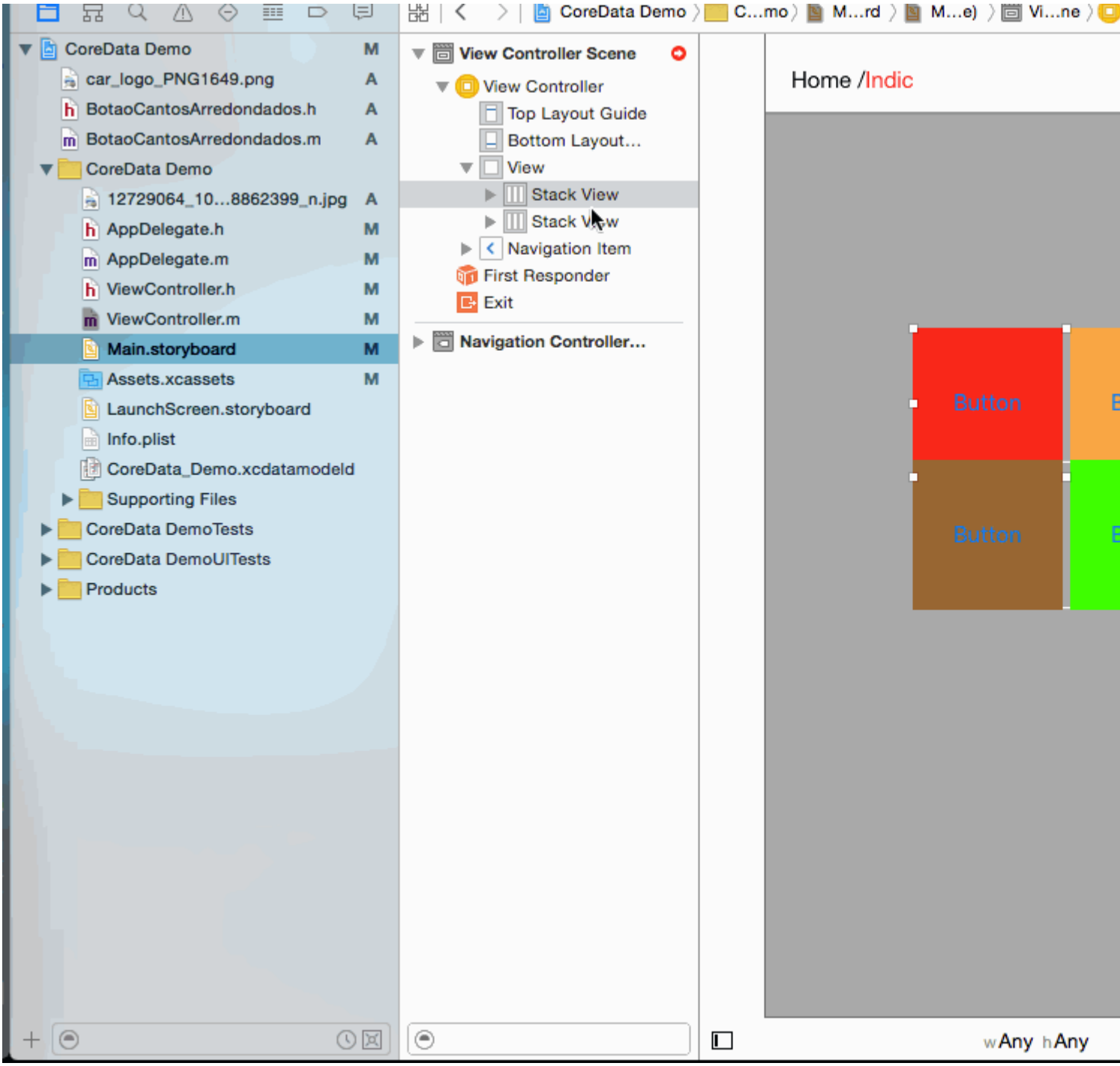


Шаг 4: - Установите свойство UIStackview для обоих.

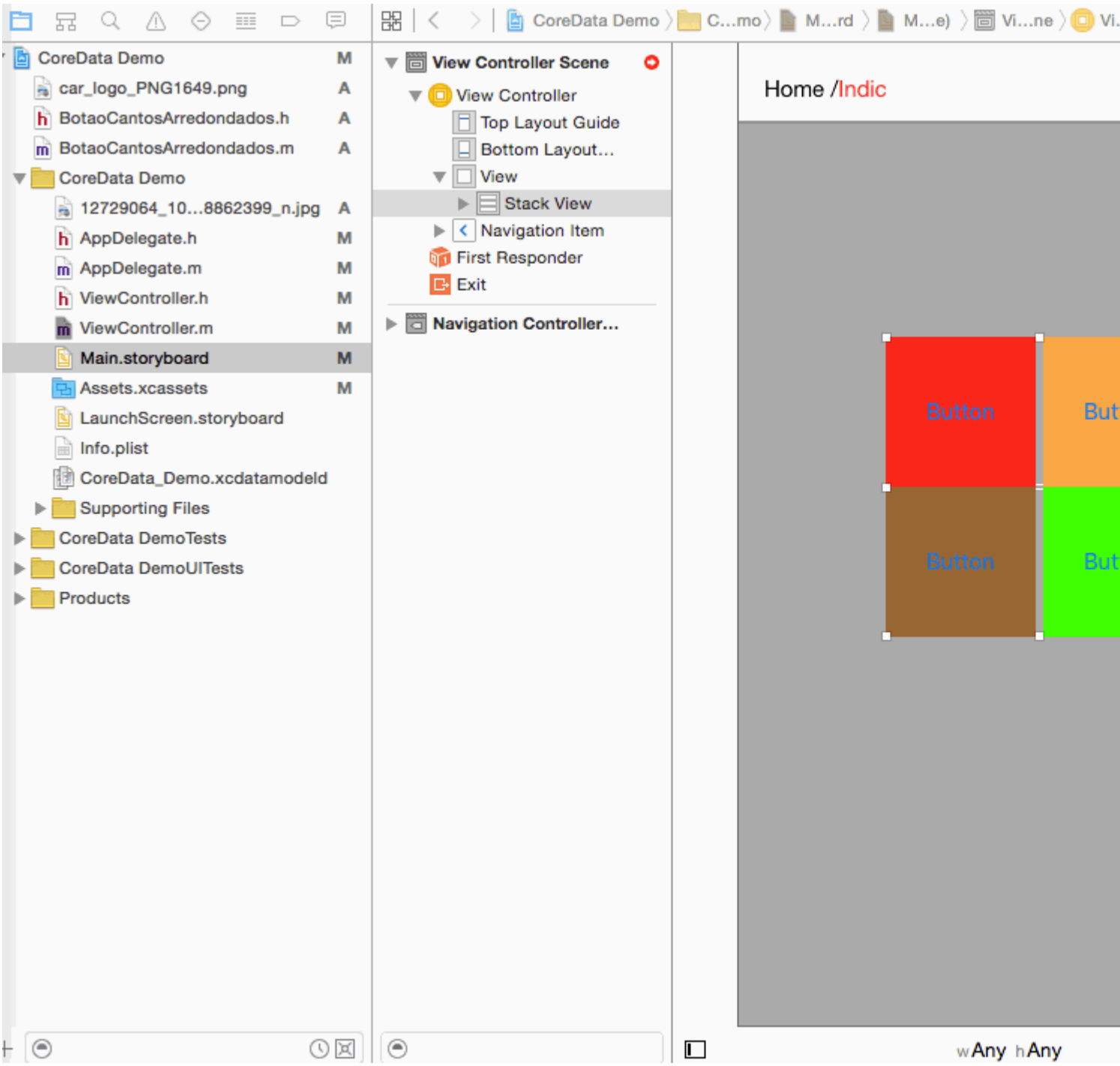
```
    Distribution -> Fill Equally
         Spacing -> 5 (as per your requirement)
```



Шаг 5: - Добавьте оба Stackview в один Stackview

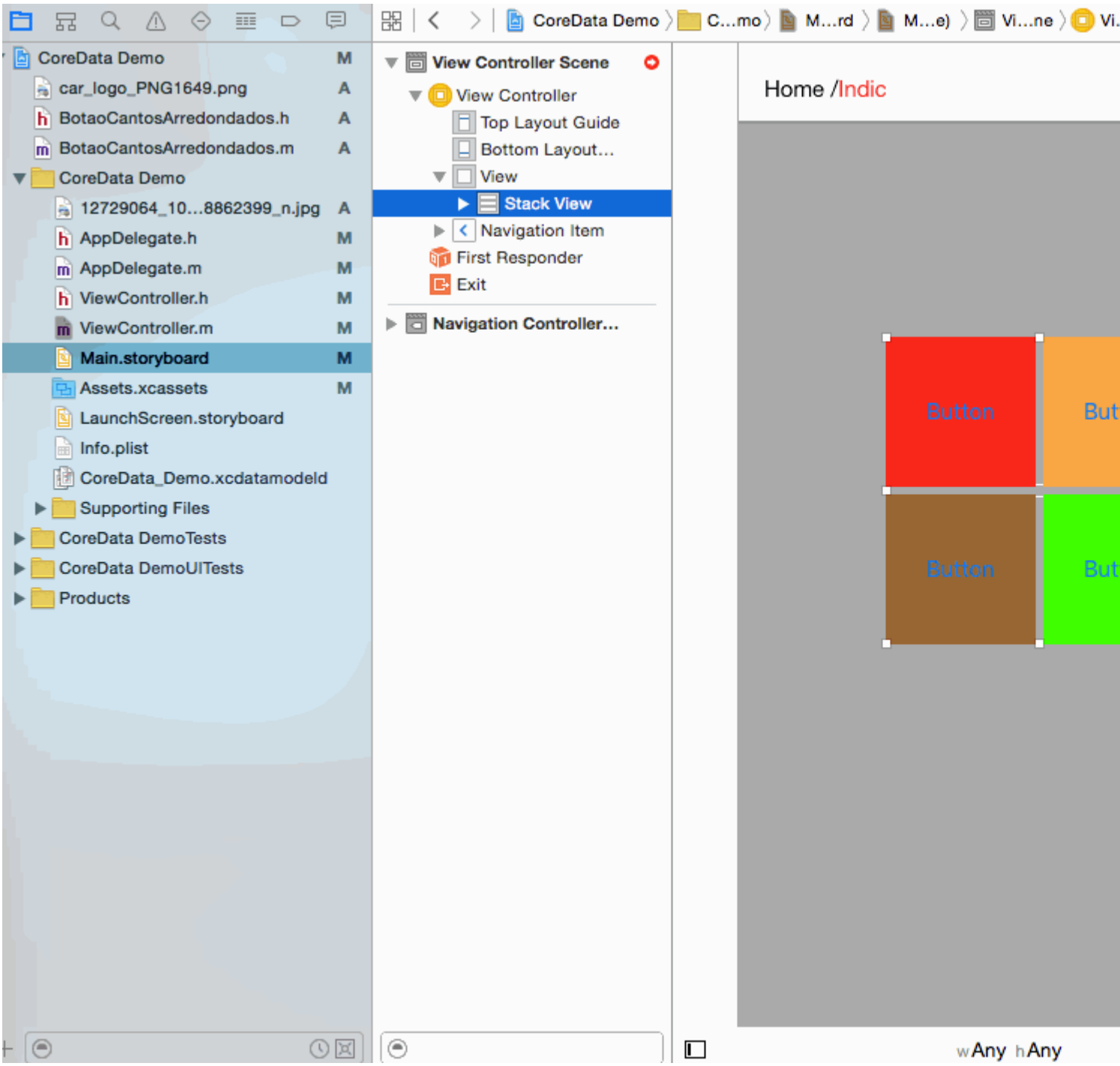


Шаг 6: - Установить `Distribution = Fill equally Spacing =5` в главном стеке (установлено в соответствии с вашим требованием)

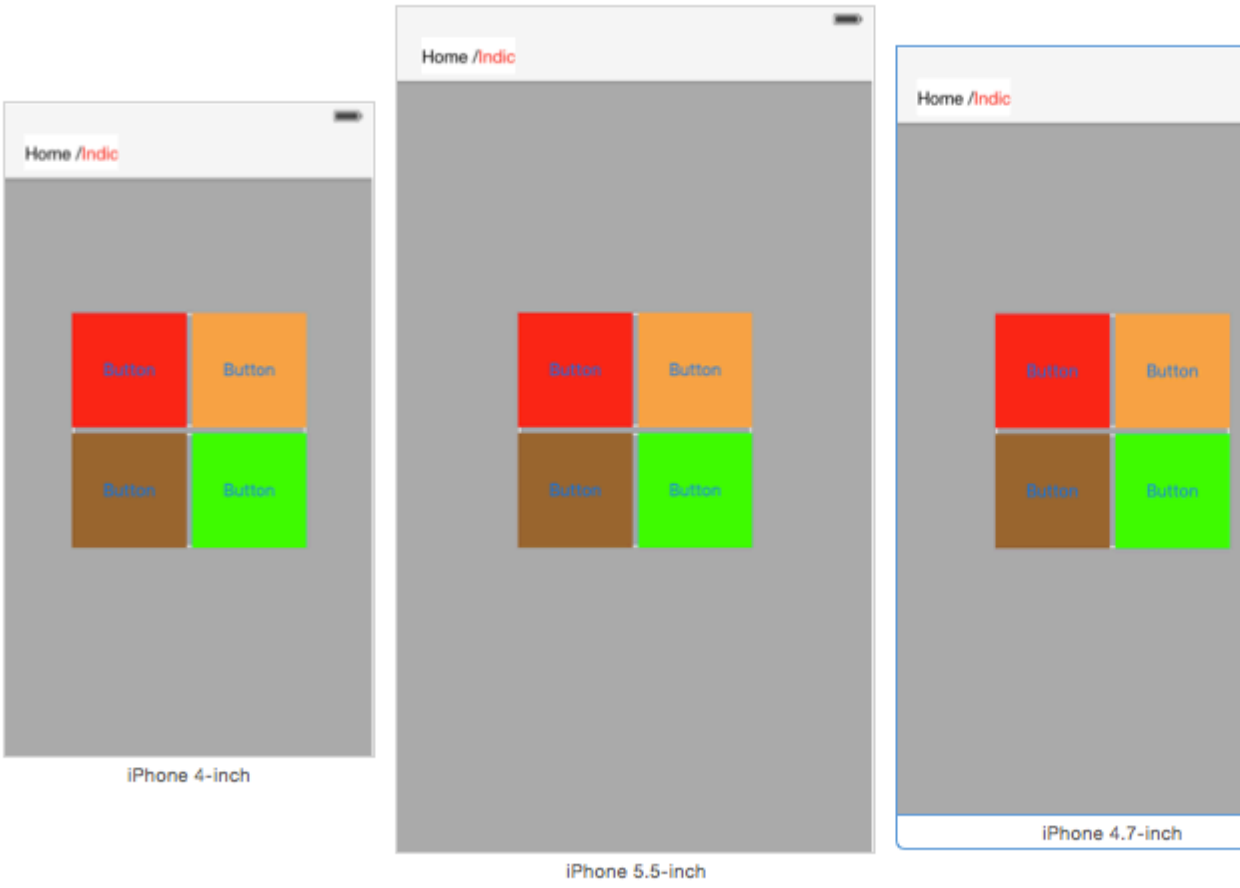


Шаг 7: - Теперь установите Constrain в основное окно стека

```
    center Horizontally in container

    center vertically in container

      and select Update Frame.
```



Шаг 8: - Пришло время для вывода для всех устройств.



Прочитайте UIStackView онлайн: https://riptutorial.com/ru/ios/topic/1390/uistackview

глава 99: UIStoryboard

Вступление

Объект UIStoryboard инкапсулирует график контроллера представления, хранящийся в файле ресурсов раскадровки Interface Builder. Этот график контроллера просмотра представляет собой контроллеры представлений для всех или части пользовательского интерфейса вашего приложения.

Examples

Получение программного обеспечения UIStoryboard

SWIFT:

Получение экземпляра **UIStoryboard** программно может быть выполнено следующим образом:

```
    let storyboard = UIStoryboard(name: "Main", bundle: nil)
```

где:

- **name** => имя раскадровки без расширения
- **bundle** => пакет, содержащий файл раскадровки и связанные с ним ресурсы. Если вы укажете nil, этот метод будет выглядеть в основном пакете текущего приложения.

Например, вы можете использовать созданный выше экземпляр для доступа к определенному **UIViewController, созданному** в этой раскадровке:

```
   let viewController = storyboard.instantiateViewController(withIdentifier: "yourIdentifier")
```

Objective-C:

Получение экземпляра **UIStoryboard** в Objective-C можно сделать следующим образом:

```
 UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"MainStoryboard" bundle:nil];
```

Пример доступа к **UIViewController, созданный в** рамках этой раскадровки:

```
MyViewController *myViewController = [storyboard
```

```
instantiateViewControllerWithIdentifier:@"MyViewControllerIdentifier"];
```

Откройте другую раскадровку

```
let storyboard = UIStoryboard(name: "StoryboardName", bundle: nil)
let vc = storyboard.instantiateViewController(withIdentifier: "ViewControllerID") as
YourViewController
self.present(vc, animated: true, completion: nil)
```

Прочитайте UIStoryboard онлайн: https://riptutorial.com/ru/ios/topic/8795/uistoryboard

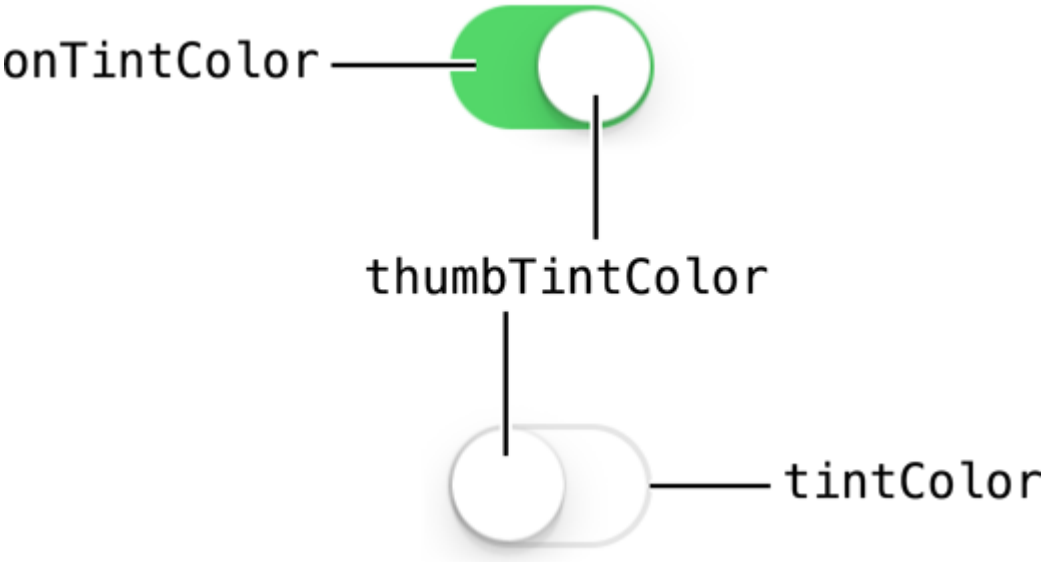
глава 100: UISwitch

Синтаксис

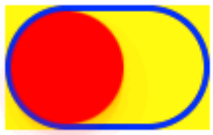
- (Instancetype) initWithFrame: (CGRect) кадр;
- (void) setOn: (BOOL) для анимированных: (BOOL) анимированных;
- (nullable instancetype) initWithCoder: (NSCoder *) aDecoder;

замечания

1. Справочник по UISwitch: документация Apple



2. Еще одна ссылка: Enoch Huang



Examples

Включение / выключение

Objective-C

```
[mySwitch setOn:YES];
//or
```

```
[mySwitch setOn:YES animated:YES];
```

стриж

```
mySwitch.setOn(false)
//or
mySwitch.setOn(false, animated: false)
```

Установить цвет фона

Objective-C

```
mySwitch.backgroundColor = [UIColor yellowColor];
[mySwitch setBackgroundColor: [UIColor yellowColor]];
mySwitch.backgroundColor =[UIColor colorWithRed:255/255.0 green:0/255.0 blue:0/255.0
alpha:1.0];
mySwitch.backgroundColor= [UIColor colorWithWhite: 0.5 alpha: 1.0];
mySwitch.backgroundColor=[UIColor colorWithHue: 0.4 saturation: 0.3 brightness:0.7 alpha:
1.0];
```

стриж

```
mySwitch.backgroundColor = UIColor.yellow
mySwitch.backgroundColor = UIColor(red: 255.0/255, green: 0.0/255, blue: 0.0/255, alpha: 1.0)
mySwitch.backgroundColor = UIColor(white: 0.5, alpha: 1.0)
mySwitch.backgroundColor = UIColor(hue: 0.4,saturation: 0.3,brightness: 0.7,alpha: 1.0)
```

Установить цвет оттенка

Objective-C

```
//for off-state
mySwitch.tintColor = [UIColor blueColor];
[mySwitch setTintColor: [UIColor blueColor]];

//for on-state
mySwitch.onTintColor = [UIColor cyanColor];
[mySwitch setOnTintColor: [UIColor cyanColor]];
```

стриж

```
//for off-state
mySwitch.tintColor = UIColor.blueColor()

//for on-state
mySwitch.onTintColor = UIColor.cyanColor()
```

Установить изображение для состояния включения / выключения

Objective-C

```
//set off-image
mySwitch.offImage = [UIImage imageNamed:@"off_image"];
[mySwitch setOffImage:[UIImage imageNamed:@"off_image"]];

//set on-image
mySwitch.onImage = [UIImage imageNamed:@"on_image"];
[mySwitch setOnImage:[UIImage imageNamed:@"on_image"]];
```

стриж

```
//set off-image
mySwitch.offImage = UIImage(named: "off_image")

//set on-image
mySwitch.onImage = UIImage(named: "on_image")
```

Прочитайте UISwitch онлайн: https://riptutorial.com/ru/ios/topic/2182/uiswitch

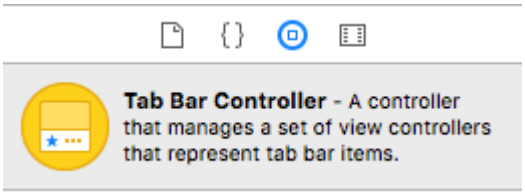
глава 101: UITabBarController

Examples

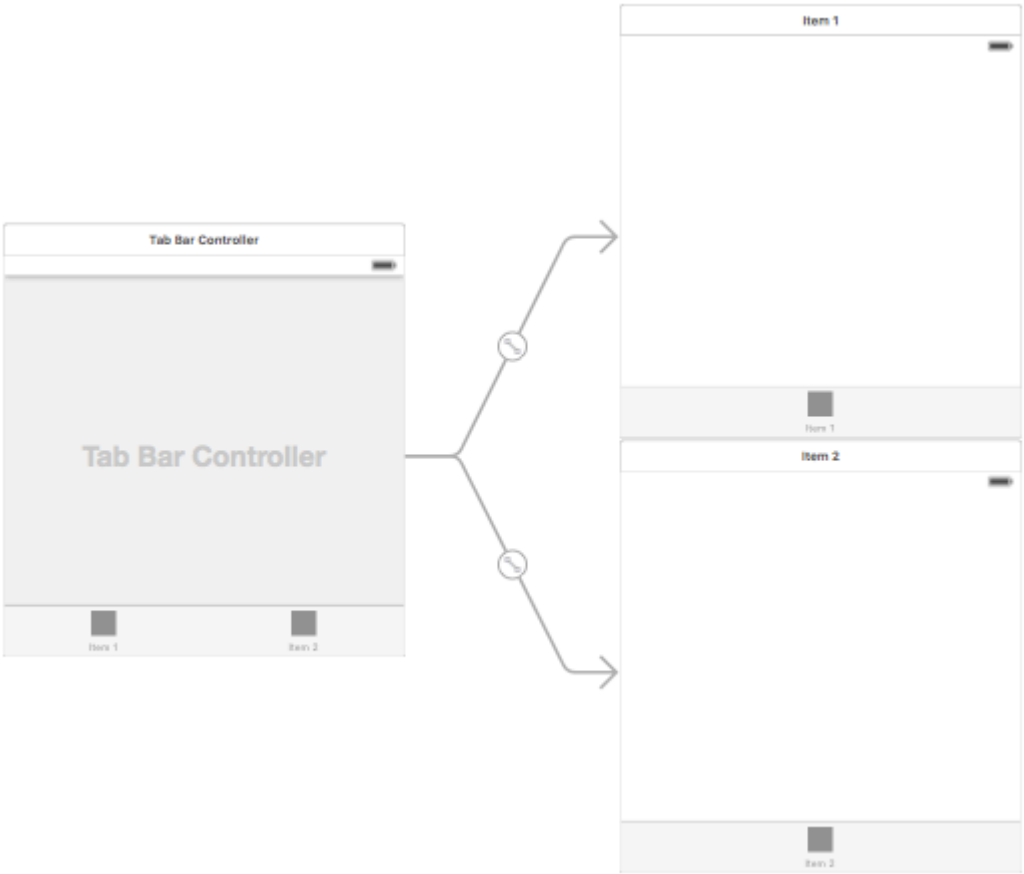
Создать экземпляр

«Панель вкладок» обычно встречается в большинстве приложений для iOS и используется для представления различных видов на каждой вкладке.

Чтобы создать контроллер панели вкладок с помощью построителя интерфейсов, перетащите контроллер панели вкладок из библиотеки объектов в холст.



По умолчанию контроллер панели вкладок имеет два вида. Чтобы добавить дополнительные виды, перетащите элемент управления панели вкладок в новый вид и выберите «просмотр контроллеров» в раскрывающемся списке.



Изменение заголовка вкладки и заголовка элемента

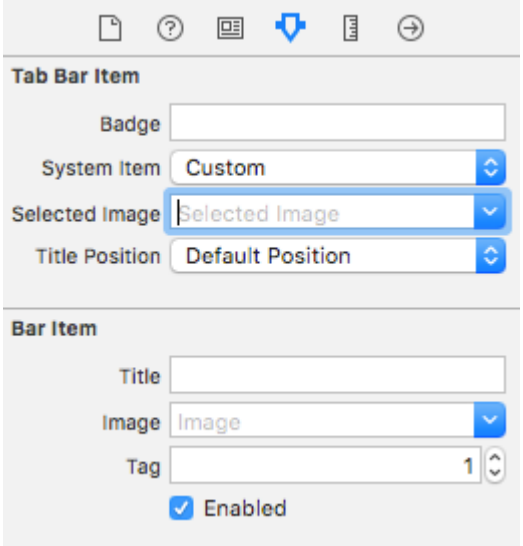
Использование Истории:

Выберите элемент панели вкладок из соответствующего контроллера представления и перейдите к инспектору атрибутов

Если вы хотите использовать встроенную иконку и заголовок, установите «Системный элемент» в соответствующее значение.

Для пользовательского значка добавьте необходимые изображения в папку «Ресурсы» и установите «Системный элемент» с более раннего времени на «Пользовательский».

Теперь установите значок, который будет отображаться, когда вкладка выбрана из раскрывающегося списка «выбранное изображение», и значок вкладки по умолчанию из раскрывающегося списка «изображение». Добавьте соответствующий заголовок в поле « title».



Программный:

В `viewDidLoad()` контроллера представления добавьте следующий код:

Objective-C:

```
self.title = @"item";

self.tabBarItem.image = [UIImage imageNamed:@"item"];
self.tabBarItem.selectedImage = [UIImage imageNamed:@"item_selected"];
```

Swift:

```
self.title = "item"
```

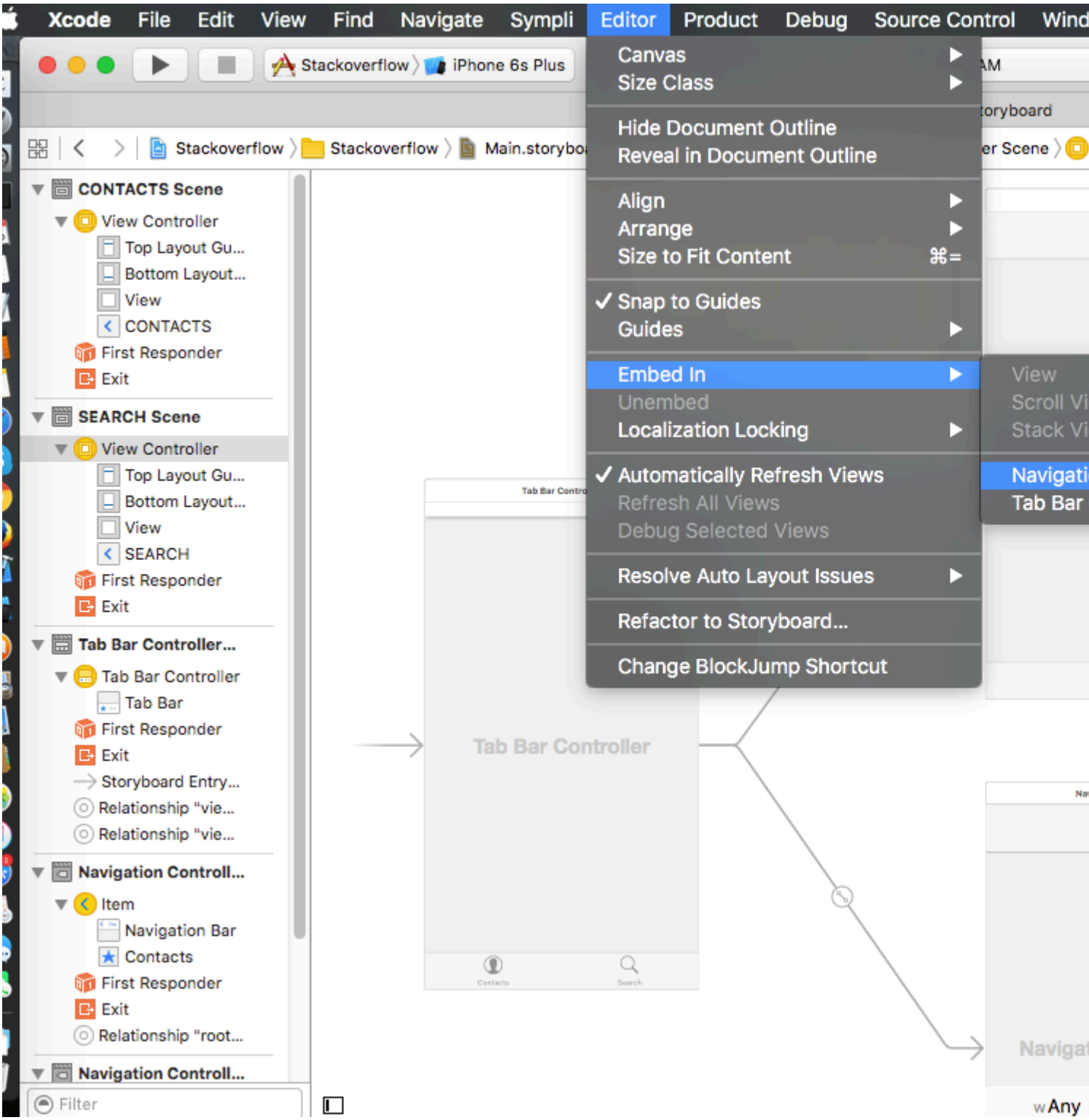
```
self.tabBarItem.image = UIImage(named: "item")
self.tabBarItem.selectedImage = UIImage(named: "item_selected")
```

Контроллер навигации с TabBar

Контроллер навигации может быть встроен в каждую вкладку с помощью раскадровки. Это может быть похоже на скриншот.

Чтобы добавить контроллер навигации в контроллер просмотра, подключающийся из контроллера панели вкладок, вот поток

- Выберите контроллер вида, для которого нам нужно добавить контроллер навигации. Здесь пусть это будет отображаться в качестве индикатора выбора.
- В меню « **Редактор»** Xcode выберите « **Вставить в -> Контроллер навигации»**



Настройка цвета вкладки

```
[[UITabBar appearance] setTintColor:[UIColor whiteColor]];
[[UITabBar appearance] setBarTintColor:[UIColor tabBarBackgroundColor]];
[[UITabBar appearance] setBackgroundColor:[UIColor tabBarInactiveColor]];
[[UINavigationBar appearance] setBarTintColor:[UIColor appBlueColor]];
[[UINavigationBar appearance] setTintColor:[UIColor whiteColor]];
[[UINavigationBar appearance] setBarStyle:UIBarStyleBlack];
```

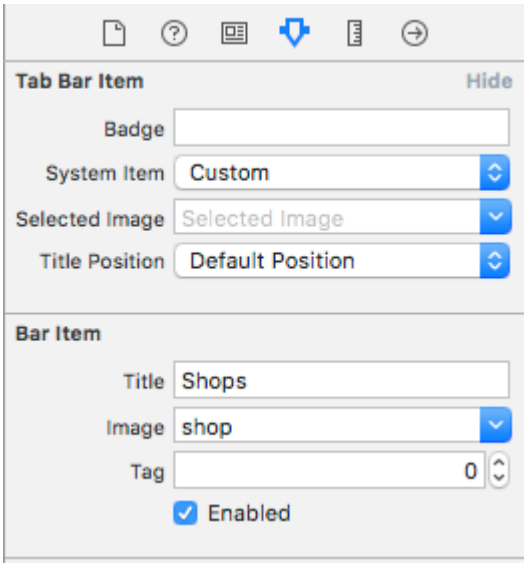
UITabBarController с пользовательским выбором цвета

Создание UITabBarController в `Swift 3` Изменение цвета и названия изображения в соответствии с выбором с изменением выбранного цвета вкладки.

```
import UIKit

class TabbarController: UITabBarController {

    override func viewDidLoad() {
        super.viewDidLoad()

        self.navigationController?.isNavigationBarHidden = true

        UITabBar.appearance().tintColor = UIColor.purple

        // set red as selected background color
        let numberOfItems = CGFloat(tabBar.items!.count)
        let tabBarItemSize = CGSize(width: tabBar.frame.width / numberOfItems, height:
tabBar.frame.height)
        tabBar.selectionIndicatorImage =
UIImage.imageWithColor(UIColor.lightText.withAlphaComponent(0.5), size:
tabBarItemSize).resizableImage(withCapInsets: UIEdgeInsets.zero)

        // remove default border
        tabBar.frame.size.width = self.view.frame.width + 4
        tabBar.frame.origin.x = -2

    }

    override func viewWillAppear(_ animated: Bool) {
        // For Images
        let firstViewController:UIViewController = NotificationVC()
        // The following statement is what you need
        let customTabBarItem:UITabBarItem = UITabBarItem(title: nil, image: UIImage(named:
"notification@2x")?.withRenderingMode(UIImageRenderingMode.alwaysOriginal), selectedImage:
UIImage(named: "notification_sel@2x"))
        firstViewController.tabBarItem = customTabBarItem

        for item in self.tabBar.items! {
            let unselectedItem = [NSForegroundColorAttributeName: UIColor.white]
            let selectedItem = [NSForegroundColorAttributeName: UIColor.purple]

            item.setTitleTextAttributes(unselectedItem, for: .normal)
            item.setTitleTextAttributes(selectedItem, for: .selected)
        }
    }

}


extension UIImage {
    class func imageWithColor(_ color: UIColor, size: CGSize) -> UIImage {
        let rect: CGRect = CGRect(origin: CGPoint(x: 0,y :0), size: CGSize(width: size.width,
height: size.height))
        UIGraphicsBeginImageContextWithOptions(size, false, 0)
        color.setFill()
        UIRectFill(rect)
        let image: UIImage = UIGraphicsGetImageFromCurrentImageContext()!
        UIGraphicsEndImageContext()
        return image
    }
```

```
}
```

Выбор изображения для панели вкладок и установка названия вкладки здесь



Выбор другой вкладки

Создать программный контроллер панели управления без раскадровки

```
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    var firstTabNavigationController : UINavigationController!
    var secondTabNavigationControoller : UINavigationController!
    var thirdTabNavigationController : UINavigationController!
    var fourthTabNavigationControoller : UINavigationController!
    var fifthTabNavigationController : UINavigationController!



    func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        Fabric.with([Crashlytics.self])

       window = UIWindow(frame: UIScreen.main.bounds)


        window?.backgroundColor = UIColor.black


        let tabBarController = UITabBarController()

        firstTabNavigationController = UINavigationController.init(rootViewController:
FirstViewController())
        secondTabNavigationControoller = UINavigationController.init(rootViewController:
SecondViewController())
        thirdTabNavigationController = UINavigationController.init(rootViewController:
ThirdViewController())
        fourthTabNavigationControoller = UINavigationController.init(rootViewController:
FourthViewController())
        fifthTabNavigationController = UINavigationController.init(rootViewController:
FifthViewController())

        tabBarController.viewControllers = [firstTabNavigationController,
```

```
secondTabNavigationControoller, thirdTabNavigationController, fourthTabNavigationControoller,
fifthTabNavigationController]


        let item1 = UITabBarItem(title: "Home", image: UIImage(named: "ico-home"), tag: 0)
        let item2 = UITabBarItem(title: "Contest", image:  UIImage(named: "ico-contest"), tag:
1)
        let item3 = UITabBarItem(title: "Post a Picture", image:  UIImage(named: "ico-photo"),
tag: 2)
        let item4 = UITabBarItem(title: "Prizes", image:  UIImage(named: "ico-prizes"), tag:
3)
        let item5 = UITabBarItem(title: "Profile", image:  UIImage(named: "ico-profile"), tag:
4)

        firstTabNavigationController.tabBarItem = item1
        secondTabNavigationControoller.tabBarItem = item2
        thirdTabNavigationController.tabBarItem = item3
        fourthTabNavigationControoller.tabBarItem = item4
        fifthTabNavigationController.tabBarItem = item5

        UITabBar.appearance().tintColor = UIColor(red: 0/255.0, green: 146/255.0, blue:
248/255.0, alpha: 1.0)

        self.window?.rootViewController = tabBarController

        window?.makeKeyAndVisible()

        return true
    }
```

Прочитайте UITabBarController онлайн: https://riptutorial.com/ru/ios/topic/2763/uitabbarcontroller

глава 102: UITableView

Вступление

Простой, широко используемый, но очень мощный вид, который может представлять данные в форме списка, используя строки и один столбец. Пользователи могут перемещаться по вертикали через элементы в представлении таблицы и, возможно, манипулировать и выбирать контент.

Синтаксис

- - (CGFloat) tableView: (UITableView *) tableView heightForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (CGFloat) tableView: (UITableView *) tableView heightForHeaderInSection: (NSInteger) раздел;
- - (CGFloat) tableView: (UITableView *) tableView heightForFooterInSection: (NSInteger);
- - (UIView *) tableView: (UITableView *) tableView viewForHeaderInSection: (NSInteger);
- - (UIView *) tableView: (UITableView *) tableView viewForFooterInSection: (NSInteger);
- - (UITableViewCellAccessoryType) tableView: (UITableView *) tableView accessoryTypeForRowWithIndexPath: (NSIndexPath *) indexPath
- - (void) tableView: (UITableView *) tableView accessoryButtonTappedForRowWithIndexPath: (NSIndexPath *) indexPath;
- - (NSIndexPath *) tableView: (UITableView *) tableView willSelectRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSIndexPath *) tableView: (UITableView *) tableView willDeselectRowAtIndexPath: (NSIndexPath *) indexPath
- - (void) tableView: (UITableView *) tableView didSelectRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView didDeselectRowAtIndexPath: (NSIndexPath *) indexPath
- - (UITableViewCellEditingStyle) tableView: (UITableView *) tableView editingStyleForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSString *) tableView: (UITableView *) tableView titleForDeleteConfirmationButtonForRowAtIndexPath: (NSIndexPath *) indexPath

- - (BOOL) tableView: (UITableView *) tableView shouldIndentWhileEditingRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView willBeginEditingRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView didEndEditingRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSIndexPath *) tableView: (UITableView *) tableView targetIndexPathForMoveFromRowAtIndexPath: (NSIndexPath *) sourceIndexPath toProposedIndexPath: (NSIndexPath *) предложилDestinationIndexPath;
- - (NSInteger) tableView: (UITableView *) tableView indentationLevelForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSInteger) tableView: (UITableView *) tableView numberOfRowsInSection: (NSInteger);
- - (UITableViewCell *) tableView: (UITableView *) tableView cellForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSInteger) numberOfSectionsInTableView: (UITableView *) tableView;
- - (NSString *) tableView: (UITableView *) tableView titleForHeaderInSection: (NSInteger); // фиксированный стиль шрифта. использовать пользовательский вид (UILabel), если вы хотите что-то другое
- - (NSString *) tableView: (UITableView *) tableView titleForFooterInSection: (NSInteger);
- - (BOOL) tableView: (UITableView *) tableView canEditRowAtIndexPath: (NSIndexPath *) indexPath;
- - (BOOL) tableView: (UITableView *) tableView canMoveRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSArray *) sectionIndexTitlesForTableView: (UITableView *) tableView;
- - (NSInteger) tableView: (UITableView *) tableView sectionForSectionIndexTitle: (NSString *) title atIndex: (NSInteger) индекс;
- - (void) tableView: (UITableView *) tableView commitEditingStyle: (UITableViewCellEditingStyle) editStyle forRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView moveRowAtIndexPath: (NSIndexPath *) sourceIndexPath toIndexPath: (NSIndexPath *) destinationIndexPath;

замечания

`UITableView` является подклассом `UIScrollView` . Классы, которые следуют за протоколом

`UITableViewDelegate` , также следуют протоколу `UIScrollViewDelegate` . `UITableView` может быть полезен для отображения длинных или неопределенных списков через свои ячейки, в то время как `UIScrollView` лучше, когда размер отображаемых представлений известен заранее.

Examples

Самостоятельные ячейки

В iOS 8 Apple представила ячейку для самостоятельной калибровки. Макет вашего UITableViewCells с Autolayout явно, и UITableView позаботится обо всем остальном. Высота строки вычисляется автоматически, по умолчанию значение `rowHeight` равно UITableViewAutomaticDimension.

UITableView свойство `estimatedRowHeight` используется , когда само-проклейки ячейка вычисления.

> Когда вы создаете ячейку просмотра таблицы размеров, вы должны установить это свойство и использовать ограничения для определения размера ячейки.
>
> *- Apple, UITableView Документация*

```
self.tableView.estimatedRowHeight = 44.0
```

Обратите внимание, что высота делегата `heightForRowAtIndexPath` не *нужна,* если вы хотите иметь динамическую высоту для всех ячеек. Просто установите указанное свойство в случае необходимости и перед повторной загрузкой или загрузкой таблицы. Тем не менее, вы можете установить высоту конкретной ячейки, а другие динамические - с помощью следующей функции:

стриж

```
override func tableView(tableView: UITableView, heightForRowAtIndexPath indexPath:
NSIndexPath) -> CGFloat {
    switch indexPath.section {
    case 1:
        return 60
    default:
        return UITableViewAutomaticDimension
    }
}
```

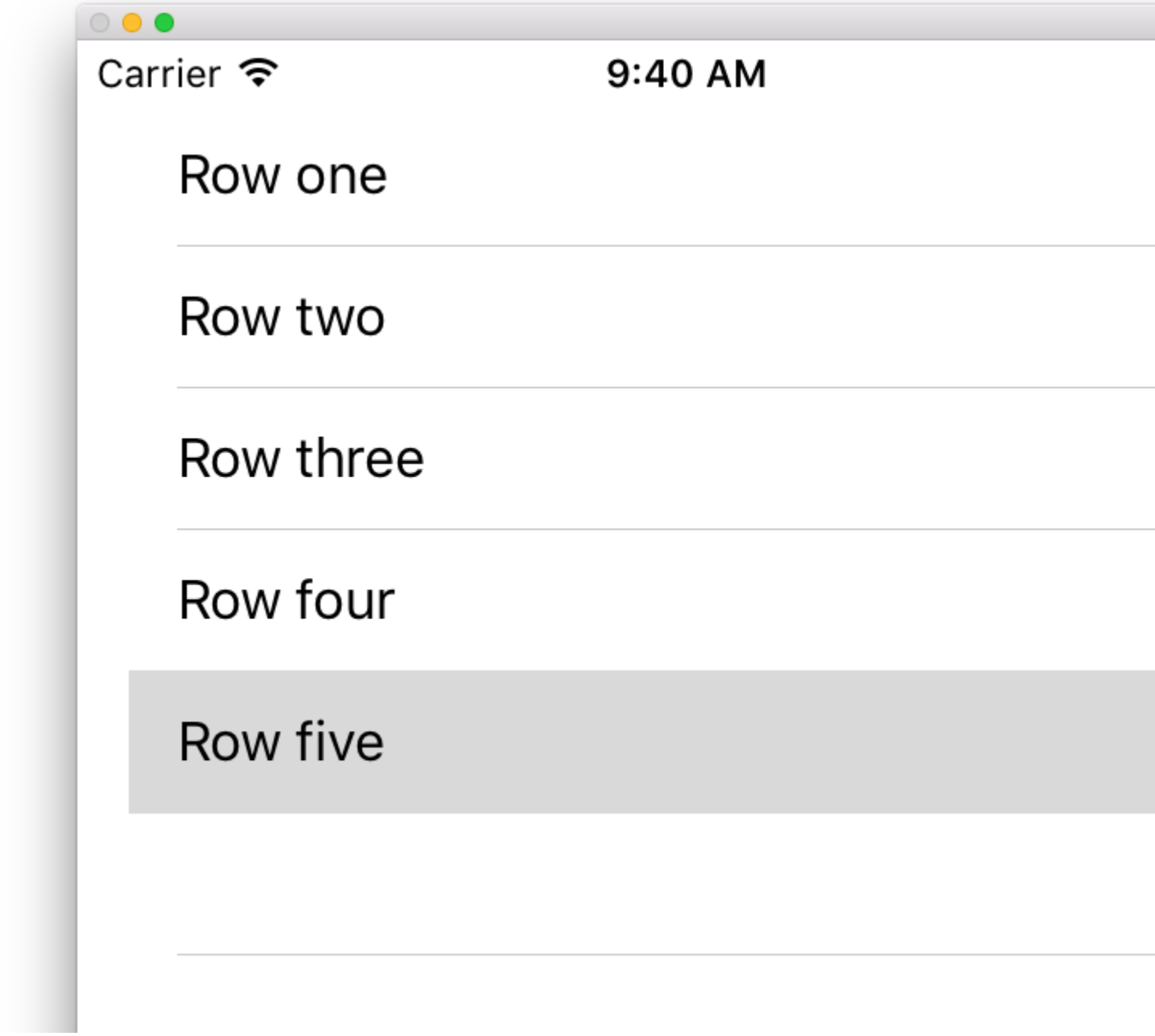
Objective-C

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
  switch (indexPath.section) {
      case 1:
```

```
        return 60;
      default:
        return UITableViewAutomaticDimension;
    }
}
```

Создание UITableView

Просмотр таблицы представляет собой список строк, которые можно выбрать. Каждая строка заполняется из источника данных. В этом примере создается простой вид таблицы, в котором каждая строка представляет собой одну строку текста.



Добавьте UITableView в свою раскадровку

Хотя существует несколько способов создания `UITableView` , одним из самых простых является добавление его в раскадровку. Откройте свою раскадровку и перетащите `UITableView` на свой `UIViewController` . Обязательно используйте Auto Layout, чтобы правильно выровнять таблицу (со всех четырех сторон).

Заполнение таблицы данными

Чтобы динамически отображать контент (т. Е. Загружать его из источника данных, такого как массив, модель Core Data, сетевой сервер и т. Д.) В вашем представлении таблицы, вам необходимо настроить источник данных.

Создание простого источника данных

Источником данных, как указано выше, может быть что угодно с данными. Его полностью зависит от вас, как отформатировать его и что в нем. Единственное требование - вы должны быть в состоянии прочитать его позже, чтобы вы могли заполнить каждую строку своей таблицы данными, когда это необходимо.

В этом примере мы просто установим массив с некоторыми строками (текстом) в качестве нашего источника данных:

стриж

```
let myDataArray: [String] = ["Row one", "Row two", "Row three", "Row four", "Row five"]
```

Objective-C

```
// You'll need to define this variable as a global variable (like an @property) so that you
can access it later when needed.
NSArray *myDataArray = @[@"Row one", @"Row two", @"Row three", @"Row four", @"Row five"];
```

Настройка источника данных в контроллере просмотра

Убедитесь, что ваш контроллер просмотра соответствует протоколу `UITableViewDataSource` .

стриж

```
class ViewController: UIViewController, UITableViewDataSource {
```

Objective-C

```
@interface ViewController : UIViewController <UITableViewDataSource>
```

Как только ваш контроллер представления объявит, что он будет **соответствовать**

`UITableViewDataSource` (это то, что мы только что сделали выше), вам *необходимо* реализовать по крайней мере следующие методы в классе контроллера вида:

- `tableView:numberOfRowsInSection` , это спрашивает, сколько строк должно `tableView:numberOfRowsInSection` вашем представлении таблицы.

```
// Swift

func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
     return self.myDataArray.count
}
```

- `tableView:cellForRowAtIndexPath` , запросы, которые вы создаете и возвращаете ячейку для каждой строки, указанной в `tableView:numberOfRowsInSection` . Итак, если вы сказали, что вам нужно 10 строк, этот метод будет называться десять раз для каждой строки, и вам нужно создать ячейку для каждой из этих строк.

```
// Swift

func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
     // Create a new cell here. The cellReuseIdentifier needs to match the reuse
identifier from the cell in your Storyboard
     let cell: UITableViewCell =
tableView.dequeueReusableCellWithIdentifier(cellReuseIdentifier) as UITableViewCell!

     // Set the label on your cell to the text from your data array
     cell.textLabel?.text = self.myDataArray[indexPath.row]

     return cell
 }
```

ПРЕДУПРЕЖДЕНИЕ : вы **НЕ** можете возвращать `cellForRowAtIndexPath:` для любых ячеек в `cellForRowAtIndexPath:` Это приведет к сбою вашего приложения, и вы увидите следующую ошибку в консоли:

```
Uncaught exception 'NSInternalInconsistencyException', reason: 'UITableView
dataSource must return a cell from tableView:cellForRowAtIndexPath:'
```

Подключение источника данных в виде таблицы к контроллеру просмотра

Вы можете сделать это с помощью кода, установив таблицы `dataSource` свойство `self` на контроллере представления. Или вы можете выбрать вид таблицы в своем раскадровке, открыть инспектор атрибутов, выбрать панель «Outlets» и перетащить с `dataSource` на ваш контроллер представления (**ПРИМЕЧАНИЕ** . Убедитесь, что вы подключаетесь к UIViewCONTROLLER, а **не** UIView или другому объекту *в* вашем UIViewController).

Обработка ряда строк

Когда пользователь нажимает на строку в вашем представлении таблицы, как правило, вы хотите что-то сделать - ответить. Во многих приложениях, когда вы нажимаете на строку, отображается дополнительная информация об этом элементе, который вы использовали. Подумайте о приложении «Сообщения»: когда вы нажимаете на строку, показывающую один из ваших контактов, разговор с этим человеком затем отображается на экране.

Чтобы сделать это, вы должны соответствовать протоколу `UITableViewDelegate` . Это похоже на соответствие протоколу источника данных. Однако на этот раз вы просто добавите его рядом с `UITableViewDataSource` и разделите его запятой. Поэтому он должен выглядеть так:

стриж

```
class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {
```

Objective-C

```
@interface ViewController : UIViewController <UITableViewDataSource, UITableViewDelegate>
```

Для делегата табличного представления нет необходимых методов. Однако для обработки выбора строк вам необходимо использовать следующий метод:

- `tableView:didSelectRowAtIndexPath` , это вызывается всякий раз, когда строка прослушивается, что позволяет вам что-то делать в ответ. В нашем примере мы просто напечатаем заявление о подтверждении в журнале Xcode.

```
// Swift

func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
     print("You tapped cell number \(indexPath.row).")
 }

// Objective-C

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath
*)indexPath {
     NSLog(@"You tapped cell number %ld.", (long)indexPath.row);
}
```

Окончательное решение

См. Ниже полную настройку с помощью только кода, никаких объяснений.

стриж

```
import UIKit
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

    // Data model: These strings will be the data for the table view cells
    let myDataArray: [String] = ["Row one", "Row two", "Row three", "Row four", "Row five"]

    // cell reuse id (cells that scroll out of view can be reused)
    let cellReuseIdentifier = "cell"

    // don't forget to hook this up from the storyboard
    @IBOutlet var myTableView: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Register the table view cell class and its reuse id
        myTableView.registerClass(UITableViewCell.self, forCellReuseIdentifier:
cellReuseIdentifier)

        // This view controller itself will provide the delegate methods and row data for the
table view.
        myTableView.delegate = self
        myTableView.dataSource = self
    }

    // number of rows in table view
    func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return self.myDataArray.count
    }

    // create a cell for each table view row
    func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {

        // create a new cell if needed or reuse an old one
        let cell:UITableViewCell =
tableView.dequeueReusableCellWithIdentifier(cellReuseIdentifier) as UITableViewCell!

        // set the text from the data model
        cell.textLabel?.text = self.myDataArray[indexPath.row]

        return cell
    }

    // method to run when table view cell is tapped
    func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
        print("You tapped cell number \(indexPath.row).")
    }
}
```

Objective-C

ViewController.h

```
#import <UIKit/UIKit.h>

@interface ViewController: UIViewController <UITableViewDelegate, UITableViewDataSource> {
    IBOutlet UITableView *myTableView;
    NSArray *myDataArray;
```

```
}

@end
```

ViewController.m

```
#import "ViewController.h"

// cell reuse id (cells that scroll out of view can be reused)
NSString * _Nonnull cellReuseIdentifier = @"cell";

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];

    // Data model: These strings will be the data for the table view cells
    myDataArray = @[@"Row one", @"Row two", @"Row three", @"Row four", @"Row five"];

    // Register the table view cell class and its reuse id
    [myTableView registerClass:[UITableViewCell class]
forCellReuseIdentifier:cellReuseIdentifier];

    // This view controller itself will provide the delegate methods and row data for the
table view.
    myTableView.delegate = self;
    myTableView.dataSource = self;
}

// number of rows in table view
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    return myDataArray.count;
}

// create a cell for each table view row
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath {
    // create a new cell if needed or reuse an old one
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:cellReuseIdentifier];

    // set the text from the data model
    cell.textLabel.text = myDataArray[indexPath.row];

    return cell;
}

// method to run when table view cell is tapped
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath{
    NSLog(@"You tapped cell number %ld.", (long)indexPath.row);
}

@end
```

Делегат и источник данных

`UITableViewDelegate` используется для управления отображением таблицы, а `UITableViewDataSource` используется для определения данных `UITableView` . Существует два

необходимых метода и множество дополнительных, которые можно использовать для настройки размера, разделов, заголовков и ячеек в `UITableView` .

UITableViewDataSource

Необходимые методы

`numberOfRowsInSection:` этот метод определяет, сколько ячеек будет отображаться в каждом разделе таблицы.

Objective-C

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    // Return the number of rows for the table view. Usually populated from an array,
    // or can be statically defined.
    return self.myArray.count;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // Return the number of rows for the table view. Usually populated from an array,
    // or can be statically defined.
    return self.myArray.count
}
```

`cellForRowAtIndexPath:` этот метод позволяет создавать и настраивать ячейки `UITableView` . Должен возвращать `UITableViewCell` или пользовательский подкласс.

Примечание: Использование `dequeueReusableCellWithIdentifier:forIndexPath:` требует, чтобы класс или СИБ был зарегистрирован для этого идентификатора , используя `UITableView` «s `registerClass:forCellReuseIdentifier:` или `registerNib:forCellReuseIdentifier:` методы. Обычно это будет выполняться в `UIViewController` `viewDidLoad` .

Objective-C

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath {
    MyCustomCell *cell = [tableView dequeueReusableCellWithIdentifier:@"MyCustomCell"
                                                         forIndexPath:indexPath];

    // All additional customization goes here
    cell.titleLabel.text = [NSString stringWithFormat:@"Title Row %lu", indexPath.row];

    return cell;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
    let cell = tableView.dequeueReusableCellWithIdentifier("MyCustomCell",
forIndexPath:indexPath)

    // All additional customization goes here
    cell.titleLabel.text = String(format:"Title Row %lu", indexPath.row)

    return cell
}
```

Дополнительные методы

`titleForHeaderInSection:` определяет строку как заголовок для каждого заголовка раздела в представлении таблицы. Этот метод позволяет изменять заголовок, дальнейшая настройка может быть выполнена путем определения представления для заголовка.

Objective-C

```
- (NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section {
    switch(section) {
        case 0:
            return @"Title 1";
            break;

        case 1:
            return @"Title 2";
            break;

        default:
            return nil;
            break;
    }
}
```

Swift 3

```
func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
    switch section {
        case 0:
            return "Title 1"
        case 1:
            return "Title 2"
        default:
            return nil
    }
}
```

`titleForFooterInSection:` Определяет строку как заголовок для каждого заголовка раздела в представлении таблицы.

Objective-C

```
- (NSString *)tableView:(UITableView *)tableView titleForFooterInSection:(NSInteger)section {
    return @"Footer text";
}
```

Swift 3

```
func tableView(_ tableView: UITableView, titleForFooterInSection section: Int) -> String? {
    return "Footer text"
}
```

`canEditRowAtIndexPath:` Используется для определения того, должен ли отображаться пользовательский интерфейс редактирования для указанной строки. Должен возвращать `YES` если указанную строку можно удалить или добавить.

Objective-C

```
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath {
    return YES;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, canEditRowAtIndexPath indexPath: NSIndexPath) -> Bool
{
    return true
}
```

`commitEditingStyle:forRowAtIndexPath` работу, необходимую для обработки добавления или удаления указанной строки. Например, удалите ячейку из `UITableView` с помощью анимации и удалите связанный объект из модели данных таблицы.

Objective-C

```
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath
*)indexPath {
    switch (editingStyle) {
        case UITableViewCellEditingStyleInsert:
            // Insert new data into the backing data model here
            [self insertNewDataIntoDataModel];
            [tableView insertRowsAtIndexPaths:@[indexPath]
withRowAnimation:UITableViewRowAnimationAutomatic];
            break;
        case UITableViewCellEditingStyleDelete:
            [self removeDataFromDataModelAtIndex:indexPath.row];
            [tableView deleteRowsAtIndexPaths:@[indexPath]
withRowAnimation:UITableViewRowAnimationAutomatic];
            break;
        default:
            // Nothing to perform if the editingStyle was neither Insert or Delete
```

```
            break;

    }
}
```

Swift 3

```
func tableView(_ tableView: UITableView, commitEditingStyle editingStyle:
UITableViewCellEditingStyle, forRowAtIndexPath indexPath: NSIndexPath) {
    switch editingStyle {
        case .Insert:
            self.insertNewDataIntoDataModel()
            tableView.insertRowsAtIndexPaths([indexPath], withRowAnimation:.Automatic)
        case .Delete:
            self.removeDataFromDataModelAtIndex(indexPath.row)
            tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation:.Automatic)
        default:
            // Nothing to perform if the editingStyle was neither Insert or Delete
    }
}
```

`editActions:forRowAt` Позволяет добавлять дополнительные действия или кнопки в режим редактирования строки внутри `UITableview` . Например, если вы хотите две кнопки, кнопку редактирования и удаления, когда пользователь перебирает для редактирования строки, вы должны использовать этот метод.

Swift 3

```
override func tableView(_ tableView: UITableView, editActionsForRowAt indexPath: IndexPath) ->
[UITableViewRowAction]? {
    // In the handler you will get passed the action as well as the indexPath for
    // the row that is being edited
    let editAction = UITableViewRowAction(style: .normal, title: "Edit", handler: { [unowned
self] action, indexPath in
        // Do something when edit is tapped
    })

    // Change the color of the edit action
    editAction.backgroundColor = UIColor.blue

    let deleteAction = UITableViewRowAction(style: .destructive, title: "Delete", handler: {
[unowned self] action, indexPath in
        // Handel the delete event
    })


    return [deleteAction, editAction]
}
```

UITableViewDelegate

Все методы в `UITableViewDelegate` являются необязательными, но делегат, который их

реализует, предоставит дополнительные возможности для `UITableView` .

`numberOfSectionsInTableView:` по умолчанию это возвращает 1, но поддержка нескольких разделов активируется путем возврата другого количества разделов.

Objective-C

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return self.numSections;
}
```

Swift 3

```
func numberOfSectionsInTableView(_ tableView: UITableView) -> Int {
    return self.numSections
}
```

`viewForHeaderInSection` Позволяет настроить настраиваемое представление как заголовок раздела.

Objective-C

```
- (UIView *)tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger)section {

    UIView *view = [[UIView alloc] initWithFrame:CGRectMake(0, 0,
CGRectGetWidth(tableView.frame), 22)];
    view.backgroundColor = [UIColor groupTableViewBackgroundColor];

    UILabel *label = [[UILabel alloc] init];
    label.font = [UIFont systemFontOfSize:12];
    label.textColor = [UIColor darkGrayColor];

    switch (section) {
        case 1: {
            label.text = @"Title";
            label.frame = labelFrame;

            UIButton *more = [[UIButton alloc] initWithFrame:btnFrame];
            [more setTitle:@"See more" forState:UIControlStateNormal];
            [more.titleLabel setFont:[UIFont systemFontOfSize:12]];
            [view addSubview:more];
        }   break;

        default:
            label.frame = CGRectMake(0, 0, 0, 0);
            break;
    }

    [view addSubview:label];
    return view;
}
```

Swift 3

```
func tableView(_ tableView: UITableView,  viewForHeaderInSection section: Int) -> UIView? {
    let view = UIView(frame: CGRect(x: 0, y: 0, width: tableView.frame.size.width, height:
22))
    view.backgroundColor = UIColor.groupTableViewBackgroundColor()

    let label = UILabel()
    label.font = UIFont.systemFontOfSize(12)
    label.textColor = UIColor.darkGrayColor()

    switch section {
        case 1:
            label.text = "Title"
            label.frame = labelFrame

            let more = UIButton(frame: btnFrame)
            more.setTitle("See more", forState:.Normal)
            view.addSubview(more)

        default:
            label.frame = CGRect.zero
    }

    view.addSubview(label)
    return view;
}
```

`heightForRowAtIndexPath:` определение высоты каждой ячейки в виде таблицы.

Objective-C

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    return 44;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, heightForRowAtIndexPath indexPath: NSIndexPath) ->
CGFloat {
    return 44
}
```

`heightForHeaderInSection:` и `heightForFooterInSection` Определите высоту для верхнего и `heightForFooterInSection` колонтитула каждого раздела в виде таблицы

Objective-C

```
- (CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section {
    return 33;
}
```

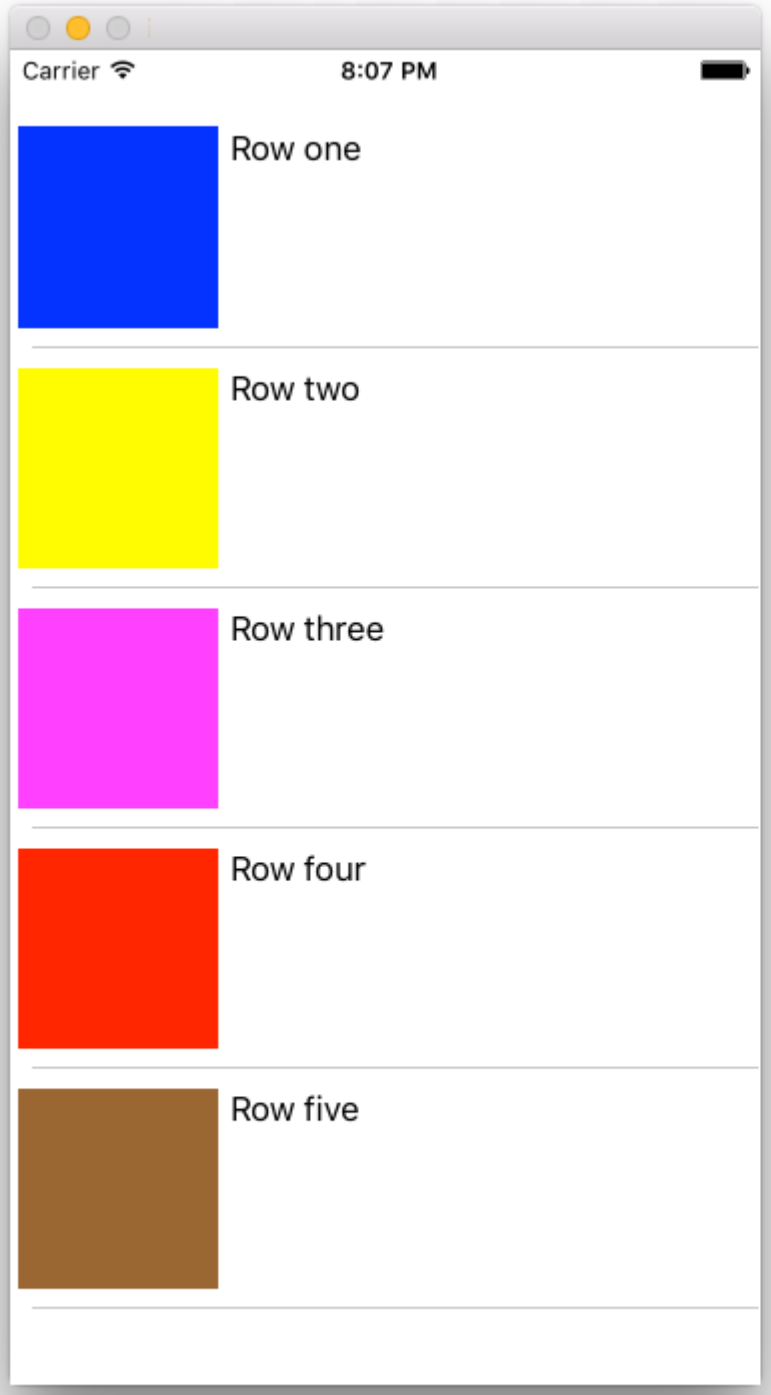
Swift 3

```
func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
    return 33
```

```
}
```

Пользовательские ячейки

Настройка `UITableViewCell` позволяет использовать очень мощные, динамичные и гибкие интерфейсы. Благодаря обширной настройке и в сочетании с другими технологиями вы можете делать такие вещи, как: обновление определенных свойств или элементов интерфейса по мере их изменения, анимации или рисования вещей в ячейке, эффективного загрузки видео по мере прокрутки пользователя или даже отображения изображений при загрузке с сеть. Возможности здесь почти бесконечны. Ниже приведен простой пример того, как может выглядеть пользовательская ячейка.



В этом разделе рассматриваются основы и, надеюсь, будут расширены, чтобы детализировать более сложные процессы, подобные описанным выше.

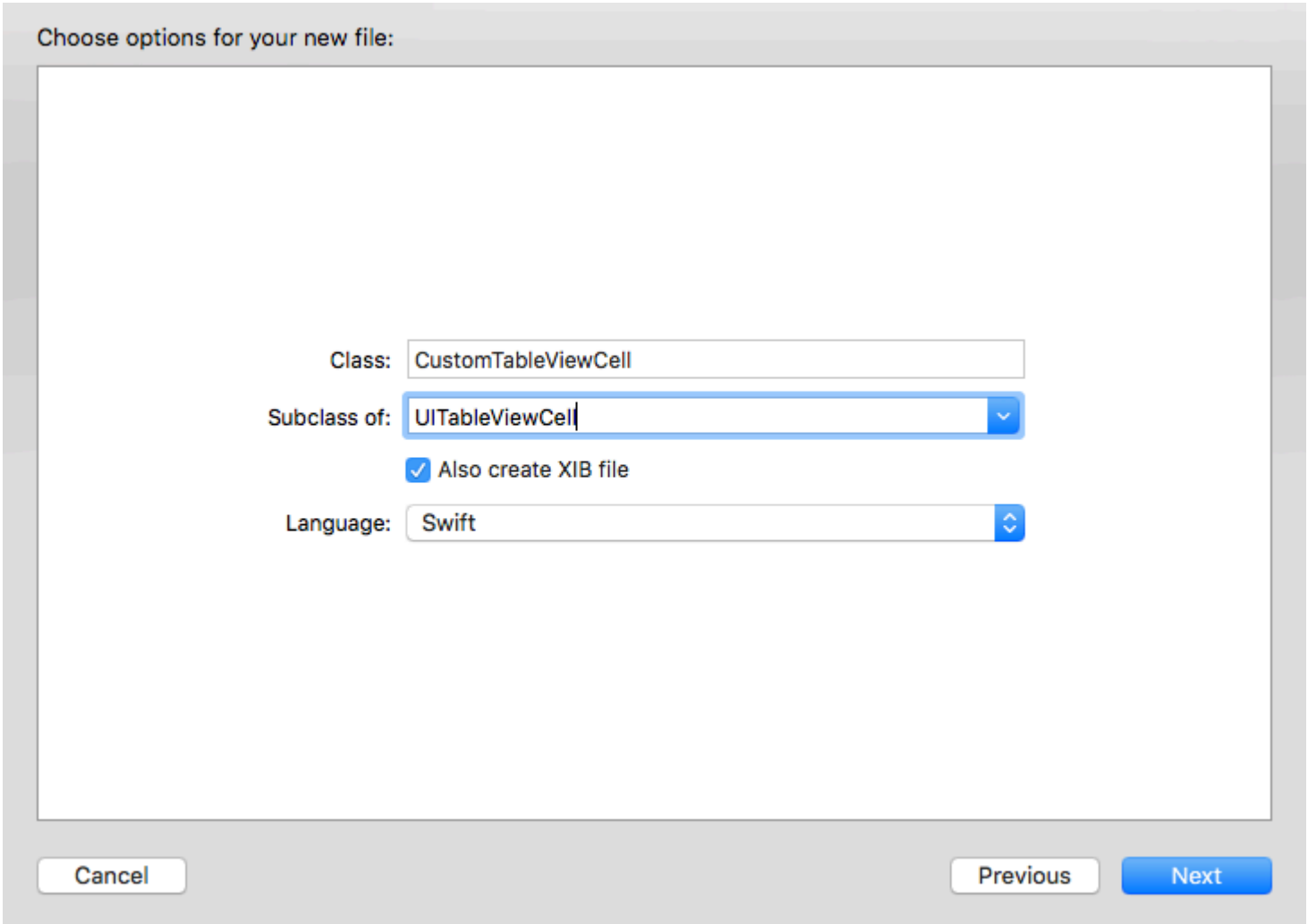
Создание пользовательской ячейки

Во-первых, создайте новый подкласс `UITableViewCell` (создайте новый класс Touch Cocoa в Xcode и установите `UITableViewCell` в качестве суперкласса). Ниже приведен код вашего кода, который может выглядеть после подкласса.

стриж

```
class CustomTableViewCell: UITableViewCell {
    static var identifier: String {
        return NSStringFromClass(self)
    }

    var customLabel: UILabel!

    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
        customLabel = UILabel(frame: CGRect(x: 0, y: 0, width: contentView.frame.width,
height: contentView.frame.height))
        customLabel.textAlignment = .center
        contentView.addSubview(customLabel)
    }
}
```

При необходимости, добавьте «Создавать XIB-файл» при создании нового файла для настройки с помощью Interface Builder. В случае, если вы это сделаете, подключите `customLabel` как `@IBOutlet`



В `UIViewController` содержащем `tableView` , зарегистрируйте класс новой пользовательской ячейки (см. Ниже). *Обратите внимание: это необходимо, только если вы не создаете ячейку*

с помощью раскадровки в интерфейсе табличного представления.

стриж

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Register Cell Class
    tableView.register(CustomTableViewCell.self, forCellReuseIdentifier:
CustomTableViewCell.identifier)
}
```

Если вы решили использовать XIB-файл, `registerNib` вместо этого:

стриж

```
// Register Nib
tableView.register(UINib(nibName: CustomTableViewCell.identifier, bundle: nil),
forCellReuseIdentifier: CustomTableViewCell.identifier)
```

Теперь, когда ваш `tableView` знает о вашей пользовательской ячейке, вы можете удалить ее из `cellForRowAtIndexPath` :

стриж

```
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
    // Load the CustomTableViewCell. Make sure the identifier supplied here matches the one
from your cell
    let cell: CustomTableViewCell =
tableView.dequeueReusableCellWithIdentifier(CustomTableViewCell.identifier) as!
CustomTableViewCell

    // This is where the magic happens - setting a custom property on your very own cell
    cell.customLabel.text = "My Custom Cell"

    return cell
}
```

Расширение и сворачивание UITableViewCells

В своей раскадровке добавьте объект UITableView на свой UIViewController и дайте ему охватить весь вид. Установите соединения `UITableviewDataSource` и `UITableviewDelegate` .

Objective-C

В файле `.h`

```
NSMutableArray *arrayForBool;
NSMutableArray *sectionTitleArray;
```

В вашем файле `.m`

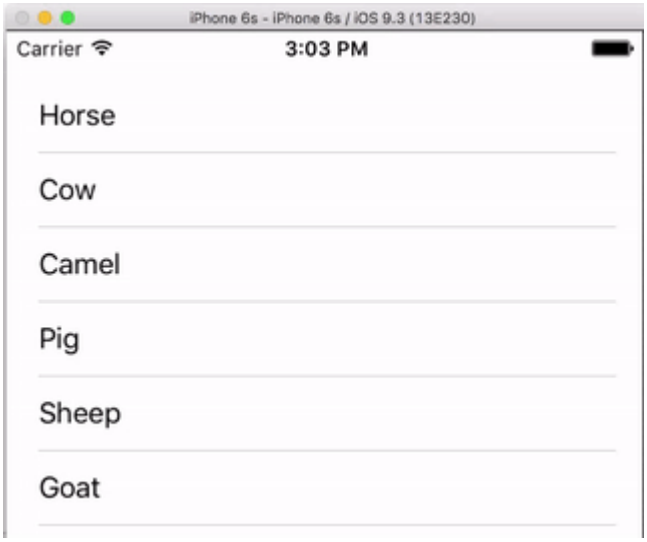
```
- (void)viewDidLoad  {
    [super viewDidLoad];

    arrayForBool = [[NSMutableArray alloc] init];
    sectionTitleArray = @[@"Sam",@"Sanju",@"John",@"Staffy"];

    for (int i=0; i<[sectionTitleArray count]; i++) {
        [arrayForBool addObject:[NSNumber numberWithBool:NO]];
    }

    _tableView.dataSource = self;
    _tableView.delegate = self;
}

 // Declare number of rows in section
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
     if ([[arrayForBool objectAtIndex:section] boolValue]) {
         return section+2;
    } else {
         return 0;
    }
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath {

static NSString *cellid=@"hello";
UITableViewCell *cell=[tableView dequeueReusableCellWithIdentifier:cellid];
if (cell==nil) {
    cell=[[UITableViewCell alloc]initWithStyle:UITableViewCellStyleSubtitle
reuseIdentifier:cellid];
}
   BOOL manyCells  = [[arrayForBool objectAtIndex:indexPath.section] boolValue];

         /** If the section supposed to be closed******/
    if(!manyCells){
        cell.backgroundColor=[UIColor clearColor];
        cell.textLabel.text=@"";
    }
      /** If the section supposed to be Opened******/
    else{
        cell.textLabel.text=[NSString stringWithFormat:@"%@ %d",[sectionTitleArray
objectAtIndex:indexPath.section],indexPath.row+1];
        cell.backgroundColor=[UIColor whiteColor];
        cell.selectionStyle=UITableViewCellSelectionStyleNone ;
    }
cell.textLabel.textColor=[UIColor blackColor];

 /** Add a custom Separator with cell*/
  UIView* separatorLineView = [[UIView alloc]initWithFrame:CGRectMake(15, 40,
_expandableTableView.frame.size.width-15, 1)];
separatorLineView.backgroundColor = [UIColor blackColor];
[cell.contentView addSubview:separatorLineView];
return cell;
}

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
return [sectionTitleArray count];
}
```

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath  *)indexPath
{

/*************** Close the section, once the data is selected
**********************************/
[arrayForBool replaceObjectAtIndex:indexPath.section withObject:[NSNumber numberWithBool:NO]];

 [_expandableTableView reloadSections:[NSIndexSet indexSetWithIndex:indexPath.section]
withRowAnimation:UITableViewRowAnimationAutomatic];

 }

 - (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath
*)indexPath
{
if ([[arrayForBool objectAtIndex:indexPath.section] boolValue]) {
    return 40;
}
return 0;

   }

- (UIView *)tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger)section
{

UIView *sectionView=[[UIView alloc]initWithFrame:CGRectMake(0, 0, 280,40)];
sectionView.tag=section;
UILabel *viewLabel=[[UILabel alloc]initWithFrame:CGRectMake(10, 0,
_expandableTableView.frame.size.width-10, 40)];
viewLabel.backgroundColor=[UIColor clearColor];
viewLabel.textColor=[UIColor blackColor];
viewLabel.font=[UIFont systemFontOfSize:15];
viewLabel.text=[NSString stringWithFormat:@"List of %@",[sectionTitleArray
objectAtIndex:section]];
[sectionView addSubview:viewLabel];
         /********** Add a custom Separator with Section view ******************/
UIView* separatorLineView = [[UIView alloc] initWithFrame:CGRectMake(15, 40,
_expandableTableView.frame.size.width-15, 1)];
separatorLineView.backgroundColor = [UIColor blackColor];
[sectionView addSubview:separatorLineView];

/********** Add UITapGestureRecognizer to SectionView   **************/

UITapGestureRecognizer  *headerTapped   = [[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(sectionHeaderTapped:)];
[sectionView addGestureRecognizer:headerTapped];

return  sectionView;


}

- (void)sectionHeaderTapped:(UITapGestureRecognizer *)gestureRecognizer{

NSIndexPath *indexPath = [NSIndexPath indexPathForRow:0 inSection:gestureRecognizer.view.tag];
if (indexPath.row == 0) {
    BOOL collapsed  = [[arrayForBool objectAtIndex:indexPath.section] boolValue];
    for (int i=0; i<[sectionTitleArray count]; i++) {
        if (indexPath.section==i) {
            [arrayForBool replaceObjectAtIndex:i withObject:[NSNumber
```

```
numberWithBool:!collapsed]];
        }
    }
    [_expandableTableView reloadSections:[NSIndexSet
indexSetWithIndex:gestureRecognizer.view.tag]
withRowAnimation:UITableViewRowAnimationAutomatic];

   }
 }
```

Проведите по удалению строк

Я всегда думаю, что хорошо иметь очень простой, самодостаточный пример, так что ничего не предполагается, когда я изучаю новую задачу. Этот ответ заключается в том, что для удаления строк `UITableView` . Проект выполняется следующим образом:



Этот проект основан на примере UITableView для Swift .

Добавить код

Создайте новый проект и замените код ViewController.swift следующим.

```
import UIKit
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

    // These strings will be the data for the table view cells
    var animals: [String] = ["Horse", "Cow", "Camel", "Pig", "Sheep", "Goat"]

    let cellReuseIdentifier = "cell"

    @IBOutlet var tableView: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // It is possible to do the following three things in the Interface Builder
        // rather than in code if you prefer.
```

```
        self.tableView.registerClass(UITableViewCell.self, forCellReuseIdentifier:
cellReuseIdentifier)
        tableView.delegate = self
        tableView.dataSource = self
    }

    // number of rows in table view
    func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return self.animals.count
    }

    // create a cell for each table view row
    func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {

        let cell:UITableViewCell =
self.tableView.dequeueReusableCellWithIdentifier(cellReuseIdentifier) as UITableViewCell!

        cell.textLabel?.text = self.animals[indexPath.row]

        return cell
    }

    // method to run when table view cell is tapped
    func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
        print("You tapped cell number \(indexPath.row).")
    }

    // this method handles row deletion
    func tableView(tableView: UITableView, commitEditingStyle editingStyle:
UITableViewCellEditingStyle, forRowAtIndexPath indexPath: NSIndexPath) {

        if editingStyle == .Delete {

            // remove the item from the data model
            animals.removeAtIndex(indexPath.row)

            // delete the table view row
            tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Fade)

        } else if editingStyle == .Insert {
            // Not used in our example, but if you were adding a new row, this is where you
would do it.
        }
    }

}
```

Единственный ключевой метод в приведенном выше коде, который включает удаление строк, является последним. Здесь снова для акцента:

```
func tableView(tableView: UITableView, commitEditingStyle editingStyle:
UITableViewCellEditingStyle, forRowAtIndexPath indexPath: NSIndexPath) {

    if editingStyle == .Delete {

        // remove the item from the data model
        animals.removeAtIndex(indexPath.row)
```

```
        // delete the table view row
        tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Fade)
    }
}
```

Раскадровка

Добавьте `UITableView` в контроллер просмотра в раскадровке. Используйте автоматический макет, чтобы прикрепить четыре стороны таблицы к краям View View. Управляйте перетаскиванием из представления таблицы в раскадровке в `@IBOutlet var tableView: UITableView!` строка в коде.

Законченный

Это все. Теперь вы сможете запустить свое приложение и удалить строки, проверив их влево и нажав «Удалить».

Заметки

- Это доступно только для iOS 8. Дополнительную информацию см. В этом ответе .
- Если вам нужно изменить количество отображаемых кнопок или текст кнопки, см. Этот ответ для получения более подробной информации.

дальнейшее чтение

- Как сделать свистящую панель для просмотра таблицы с действиями - без использования орехов со списком прокрутки
- Документация Apple

Сепараторные линии

Редактирование ширины разделительных линий

Вы можете настроить, чтобы линии разделителя вашего табличного вида расширяли до разных ширины по таблице, изменяя свойство `layoutMargins:` свойство в вашей ячейке (`layoutMargins:`). Это может быть достигнуто несколькими способами.

Изменение разделительных линий для определенных ячеек

В любом из ваших таблиц рассмотрите метод `cellForRowAtIndexPath:` *или* метод `willDisplayCell:` установите свойство `UIEdgeInsetsZero` ячейки `layoutMargins:` свойство `UIEdgeInsetsZero` (распространяется на всю ширину таблицы) или на все, что вы можете пожелать здесь.

Objective-C

```
[cell setLayoutMargins:UIEdgeInsetsZero];

// May also use separatorInset
[cell setSeparatorInset:UIEdgeInsetsZero];
```

стриж

```
func tableView(tableView: UITableView, willDisplayCell cell: UITableViewCell,
forRowAtIndexPath indexPath: NSIndexPath) {
    cell.separatorInset = UIEdgeInsetsZero
    cell.layoutMargins = UIEdgeInsetsZero
}

func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell
{
    cell.separatorInset = UIEdgeInsetsZero
    cell.layoutMargins = UIEdgeInsetsZero
}
```

Удалить все разделительные линии

Тонкие серые линии между каждой ячейкой могут быть не совсем тем, что вы ищете. Скрыть их от взгляда довольно просто.

В вашем `UIViewController` `viewDidLoad:` добавьте следующий код. Вы также можете установить это свойство в любое время, прежде чем загружать или перезагружать представление таблицы (необязательно должно быть в `viewDidLoad:` .

Swift:

```
 tableView.separatorStyle = .None
```

Objective-C:

```
tableView.separatorStyle = UITableViewCellSeparatorStyleNone;
```

В качестве альтернативы, свойство можно изменить в вашей раскадровке или XIB, выбрав `separator` tableView и параметров (под инспектором атрибутов) в `None` .

Скрыть лишние разделительные линии

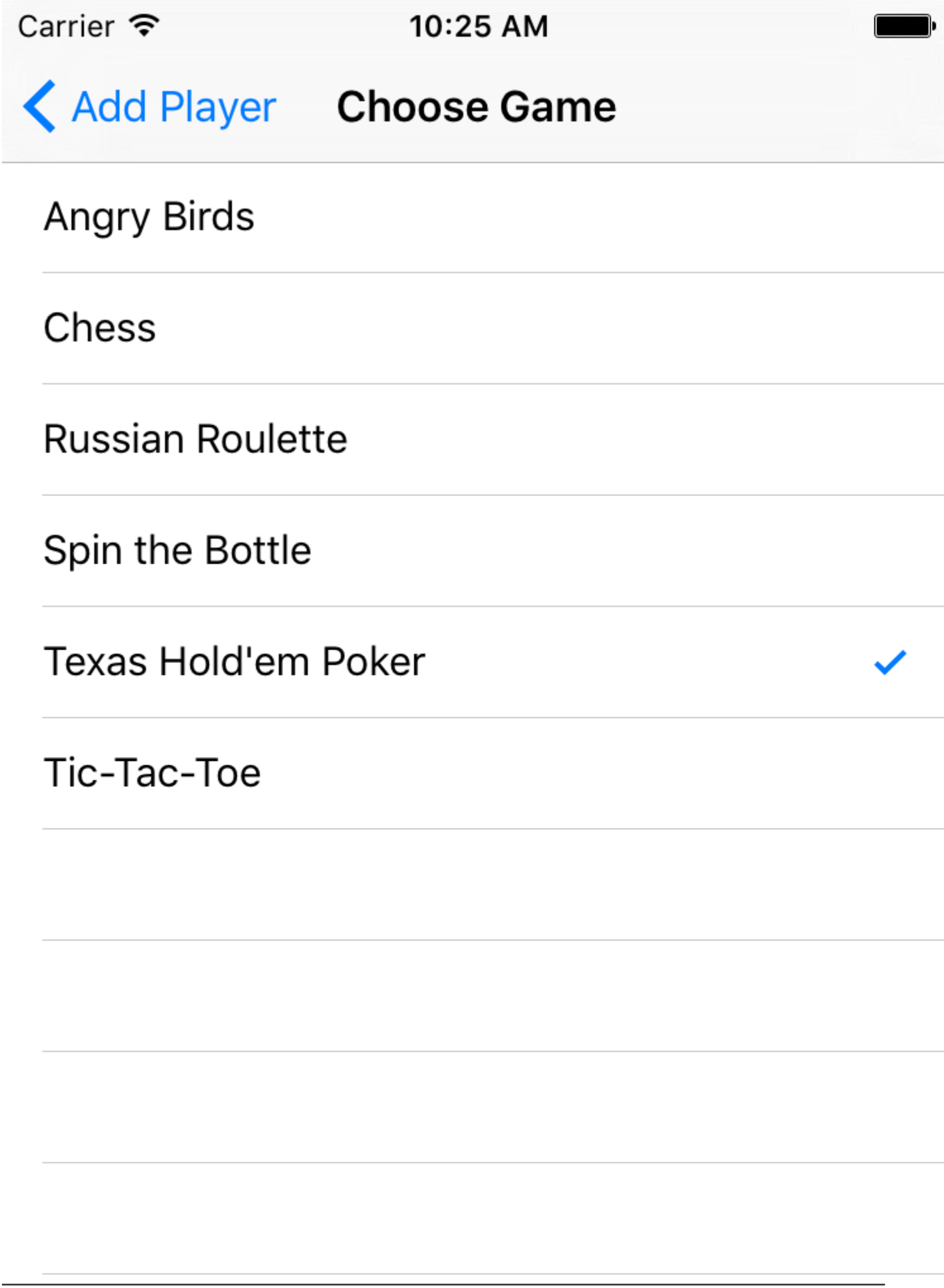
Вы можете скрыть строки разделителя `UITableViewCell` для пустых ячеек, установив пустой нижний колонтитул внизу UITableView:

стриж

```
tableView.tableFooterView = UIView()
```

Objective-C

```
tableView.tableFooterView = [[UIView alloc] initWithFrame:CGRectZero];
```


Carrier
10:25 AM
Add Player
Choose Game
Angry Birds
Chess
Russian Roulette
Spin the Bottle
Texas Hold'em Poker
Tic-Tac-Toe

глава 103: UITableViewCell

Вступление

Загрузка пользовательского xib-файла ячейки использует класс класса ячейки, нет необходимости регистрировать файл nib

Examples

Xib-файл UITableViewCell

Создайте класс класса ячейки `UITableView` .

Файл UITableViewCell + RRCell.h

```
#import <UIKit/UIKit.h>

@interface UITableViewCell (RRCell)

-(id)initWithOwner:(id)owner;

@end
```

Файл UITableViewCell + RRCell.m

```
#import "UITableViewCell+RRCell.h"

@implementation UITableViewCell (RRCell)

#pragma clang diagnostic push
#pragma clang diagnostic ignored "-Wobjc-designated-initializers"

-(id)initWithOwner:(id)owner {

    if (self = [super init]) {

        NSArray *nib = [[NSBundle mainBundle]loadNibNamed:NSStringFromClass([self class])
owner:self options:nil];
        self = [nib objectAtIndex:0];
    }
    return self;
}

#pragma clang diagnostic pop



@end
```

Импортируйте класс класса ячейки, чтобы использовать этот метод в методе `cellForRowAtIndexPath`

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath
{
    //Creted custom cell xib file to load by cell category class
    CustomCell *cell = [[CustomCell alloc]initWithOwner:self];

    return cell;
}
```

Прочитайте UITableViewCell онлайн: https://riptutorial.com/ru/ios/topic/10101/uitableviewcell

глава 104: UITableViewController

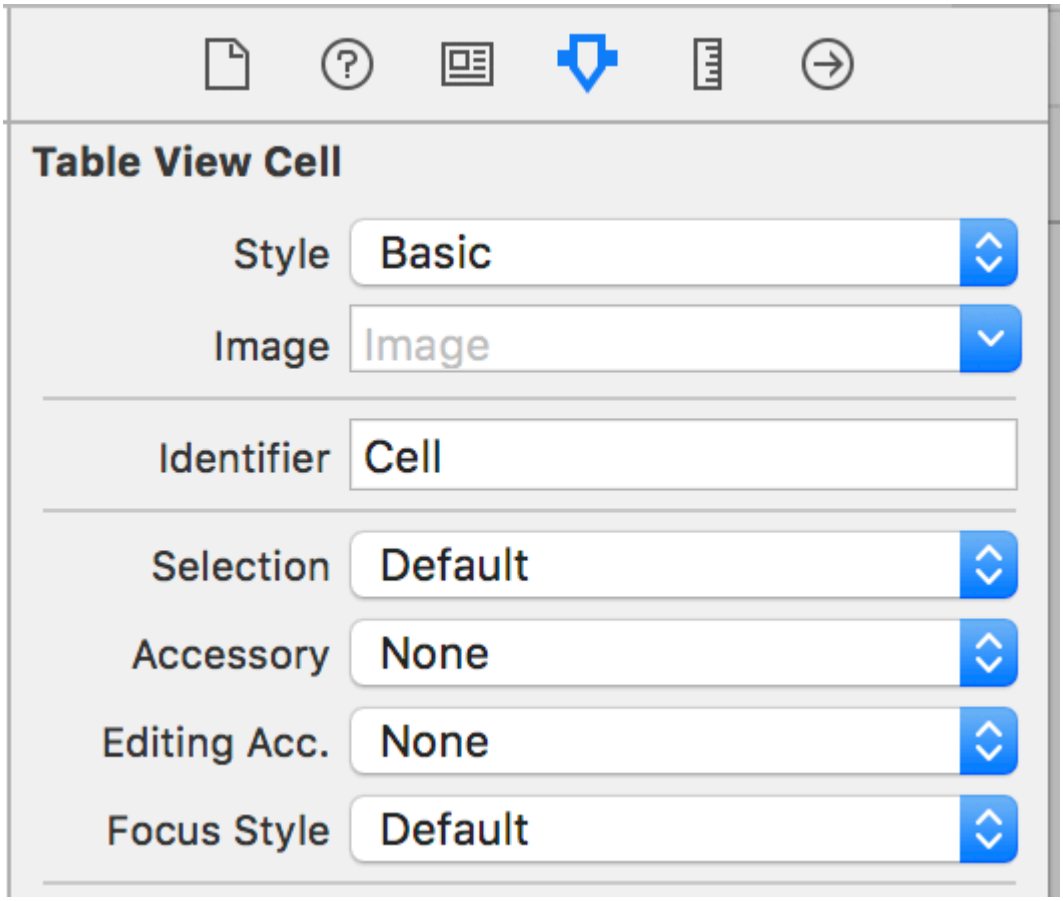
Вступление

Объект контроллера UITableViewController, который управляет представлением таблицы. Для некоторого определенного сценария рекомендуется использовать UITableViewController, например, если у вас много ячеек, а у некоторых есть UITextfield.

Examples

TableView с динамическими свойствами с помощью tableviewCellStyle basic.

```
override func numberOfSections(in tableView: UITableView) -> Int {
    // You need to return minimum one to show the cell inside the tableview
    return 1
}

override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // return the number of rows inside the tableview.
    return 3
}


override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {

    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
// identifier string should be same as what you have entered in the cell Attribute inspector -
> identifier (see the image).

    // Configure the cell...
    cell.textLabel?.text = "Cell \(indexPath.row) :" + "Hello"
//cell have different style Custom, basic, right detail, left detail, subtitle.
//For custom you can use your own objects and constrains, for other styles all
//is ready just select according to your design. (see the image for changing the style)

    return cell
}

override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        // this delegate method will trigger when you click a cell
}
```



TableView с пользовательской ячейкой

Для пользовательской ячейки tableview вам нужен класс, который является подклассом из `UITableViewCell` , примерного класса, который вы можете увидеть ниже.

```
class TableViewCell: UITableViewCell {

@IBOutlet weak var lblTitle: UILabel!

override func awakeFromNib() {
    super.awakeFromNib()
    // Initialization code
}

override func setSelected(_ selected: Bool, animated: Bool) {
    super.setSelected(selected, animated: animated)

    // Configure the view for the selected state
}

}
```

Ваши делегаты на столе

```
override func numberOfSections(in tableView: UITableView) -> Int {
        // You need to return minimum one to show the cell inside the tableview
        return 1
    }
```

```
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // return the number of rows inside the tableview.
    return 3
}


override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {

    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath) as!
TableViewCell
    // identifier string should be same as what you have entered in the cell Attribute
inspector -> identifier.

    // Configure the cell...
    cell.lblTitle.text = "Cell \(indexPath.row) :" + "Hello"


    return cell
}

override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    // this delegate method will trigger when you click a cell
}
```

Прочитайте UITableViewController онлайн:
https://riptutorial.com/ru/ios/topic/10953/uitableviewcontroller

глава 105: UITextField

Вступление

UITextField является частью структуры UIKit и используется для отображения области для ввода текстового ввода от пользователя с помощью экранной клавиатуры

Синтаксис

- UITextField.text: String // получить или установить текст, отображаемое в поле.
- UITextField.attributedText: NSAttributedString // получает или задает атрибут текста, отображаемого в поле.
- UITextField.textColor: UIColor // получить или установить цвет текста в поле
- UITextField.font: UIFont // получить или установить шрифт текста в поле
- UITextField.textAlignment: NSTextAlignment // по умолчанию - NSLeftTextAlignment
- UITextField.borderStyle: UITextBorderStyle // по умолчанию UITextBorderStyleNone. Если установлено значение UITextBorderStyleRoundedRect, пользовательские фоновые изображения игнорируются.
- UITextField.placeholder: String // по умолчанию - nil. строка нарисована 70% серого
- UITextField.attributedPlaceholder: NSAttributedString // получить или установить атрибут-заполнитель поля
- UITextField.clearsOnBeginEditing: Bool // default - НЕТ, который перемещает курсор на выбранное местоположение. если ДА, весь текст очищен
- UITextField.adjustsFontSizeToFitWidth: Bool // default - НЕТ. если ДА, текст будет уменьшаться до minFontSize вдоль базовой линии
- UITextField.minimumFontSize: CGFloat // по умолчанию 0.0. фактический минимум может быть прикреплен к чему-то читаемому. используется, если adjustsFontSizeToFitWidth - ДА
- UITextField.delegate: UITextFieldDelegate? // default - nil. слабая ссылка
- UITextField.clearButtonMode: UITextFieldViewMode // устанавливает, когда появляется кнопка очистки. default - UITextFieldViewModeNever
- UITextField.leftView: UIView? // например увеличительное стекло
- UITextField.leftViewMode: UITextFieldViewMode // устанавливает, когда появляется левое представление. default - UITextFieldViewModeNever
- UITextField.rightView: UIView? // кнопка закладки
- UITextField.rightViewMode: UITextFieldViewMode // устанавливает, когда появляется правильный вид. default - UITextFieldViewModeNever
- UITextField.inputView: UIView? // Представлено, когда объект становится первым ответчиком. Если установлено значение nil, возвращается к следующей цепочке ответчиков. Если он установлен в режиме первого ответчика, он не будет действовать до тех пор, пока не будет вызван reloadInputViews.

- UITextField.inputAccessoryView: UIView?
- UITextField.isSecureTextEntry: Bool // например, если поле содержит конфиденциальный ввод, например пароль или номер карты

Examples

Инициализировать текстовое поле

стриж

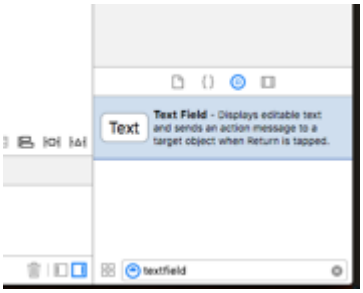
```
let frame = CGRect(x: 0, y: 0, width: 100, height: 100)
let textField = UITextField(frame: frame)
```

Objective-C

```
CGRect *frame = CGRectMake(0, 0, 100, 100);
UITextField *textField = [[UITextField alloc] initWithFrame:frame];
```

Интерфейс Builder

Вы также можете добавить `UITextField` в раскадровку, перетащив его из библиотеки объектов.



Вид вспомогательного аксессуара (панель инструментов)

Добавьте дополнительный вид над клавиатурой. Это обычно используется для добавления следующих / предыдущих кнопок или дополнительных кнопок, таких как Done / Submit (особенно для клавиш клавиатуры номер / телефон / десятичная клавиатура, у которых нет встроенного ключа возврата).

стриж

```
let textField = UITextField() // initialized however

let toolbar = UIToolbar(frame: CGRect(x: 0, y: 0, width: view.frame.size.width, height: 0)
```

```
let flexibleSpace = UIBarButtonItem(barButtonSystemItem: .FlexibleSpace, target: nil, action:
nil)

let doneButton = UIBarButtonItem(barButtonSystemItem: .Done, target: self, action:
Selector("done"))

let items = [flexibleSpace, doneButton]  // pushes done button to right side

toolbar.setItems(items, animated: false) // or toolbar.items = ...
toolbar.sizeToFit()

textField.inputAccessoryView = toolbar
```

Objective-C

```
UITextField *textField = [[UITextField alloc] init];

UIToolbar *toolbar = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
self.view.frame.size.width, 0)];

UIBarButtonItem *flexibleSpace = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil action:nil];
UIBarButtonItem *doneButton = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemDone target:self action:@selector(done)];
NSArray *items = @[
                   flexibleSpace,
                   doneButton
                   ];

[toolbar setItems:items];
[toolbar sizeToFit];

textField.inputAccessoryView = toolbar;
```

Автоматическая замена строчных букв заглавными

стриж

```
textField.autocapitalizationType = .None
```

Objective-C

```
textField.autocapitalizationType = UITextAutocapitalizationTypeNone;
```

Все варианты:

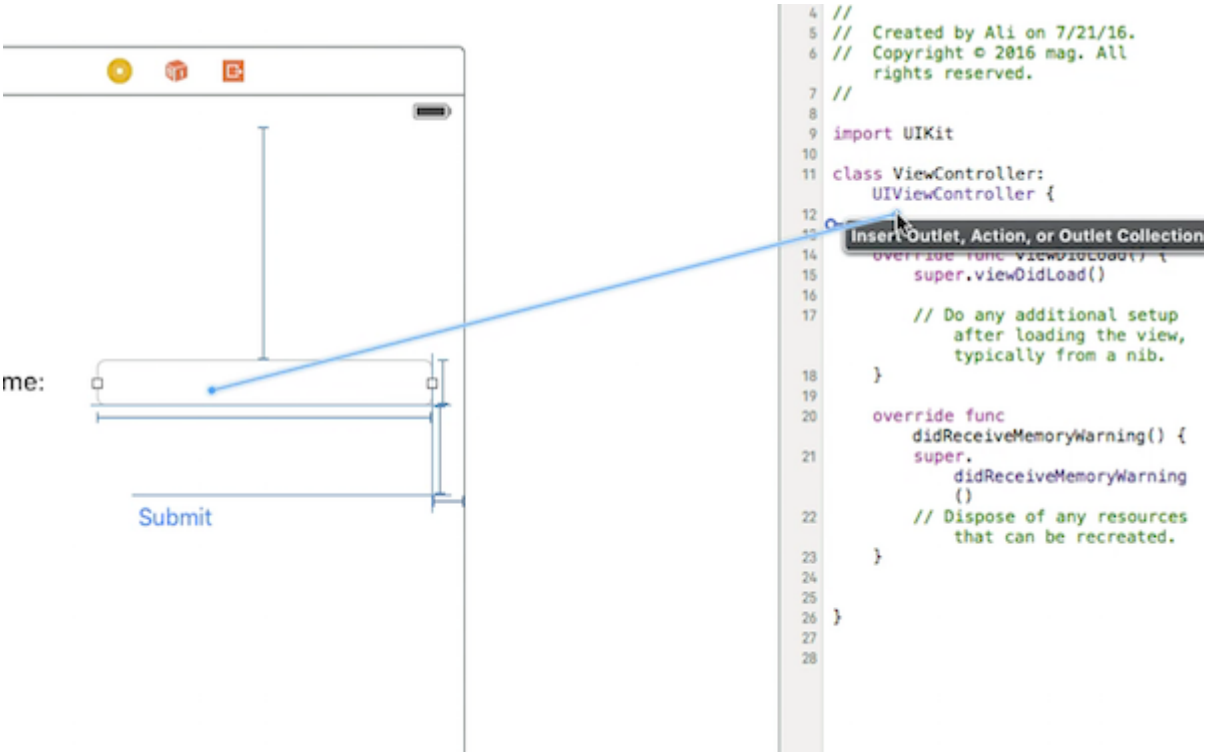
- `.None` \ `UITextAutocapitalizationTypeNone` : не автокапитализировать что-либо
- `.Words` \ `UITextAutocapitalizationTypeWords` : автокапитализировать каждое слово
- `.Sentences` \ `UITextAutocapitalizationTypeSentences` : автокапитализировать первое слово в предложении

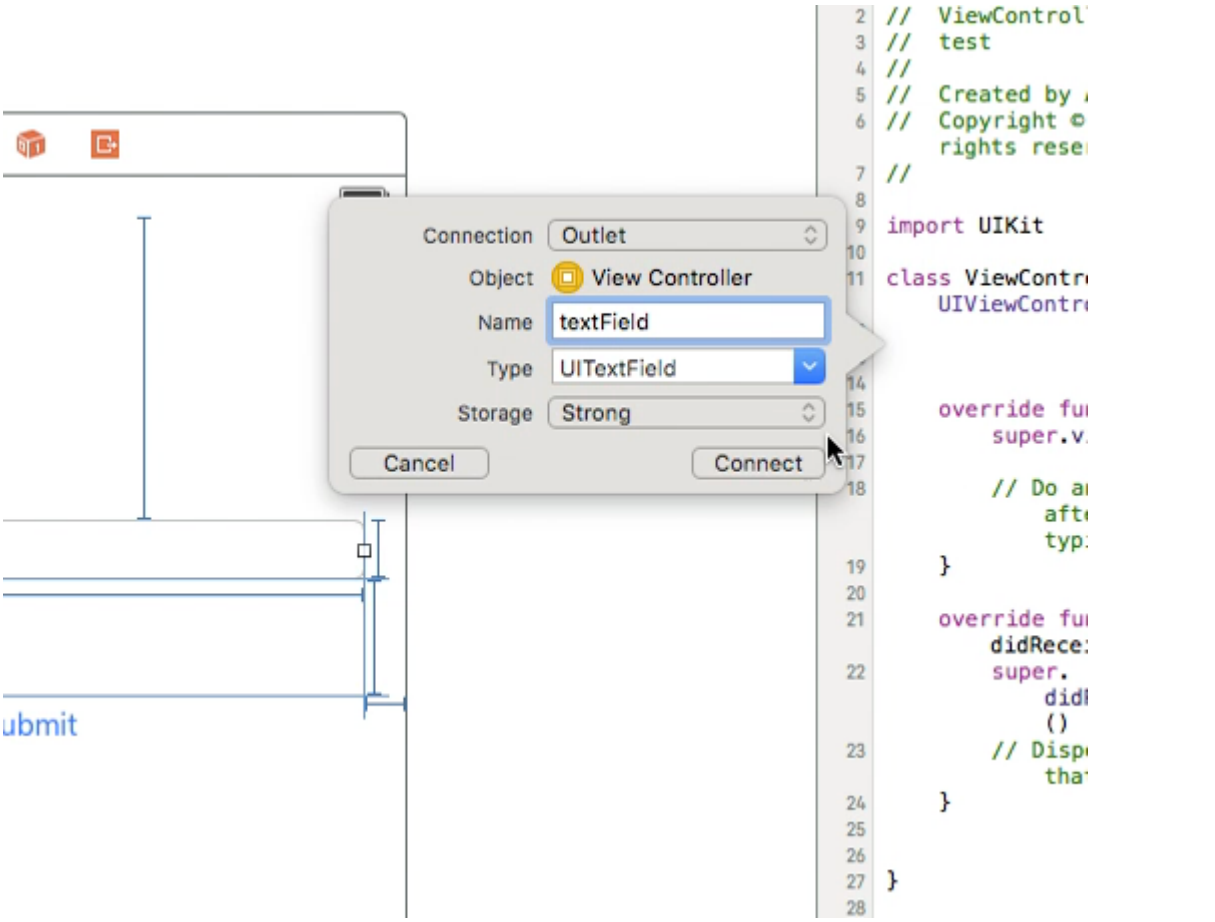
- `.AllCharacters` \ `UITextAutocapitalizationTypeAllCharacters` : автокапитализировать каждую букву (например, закрытие крышки)

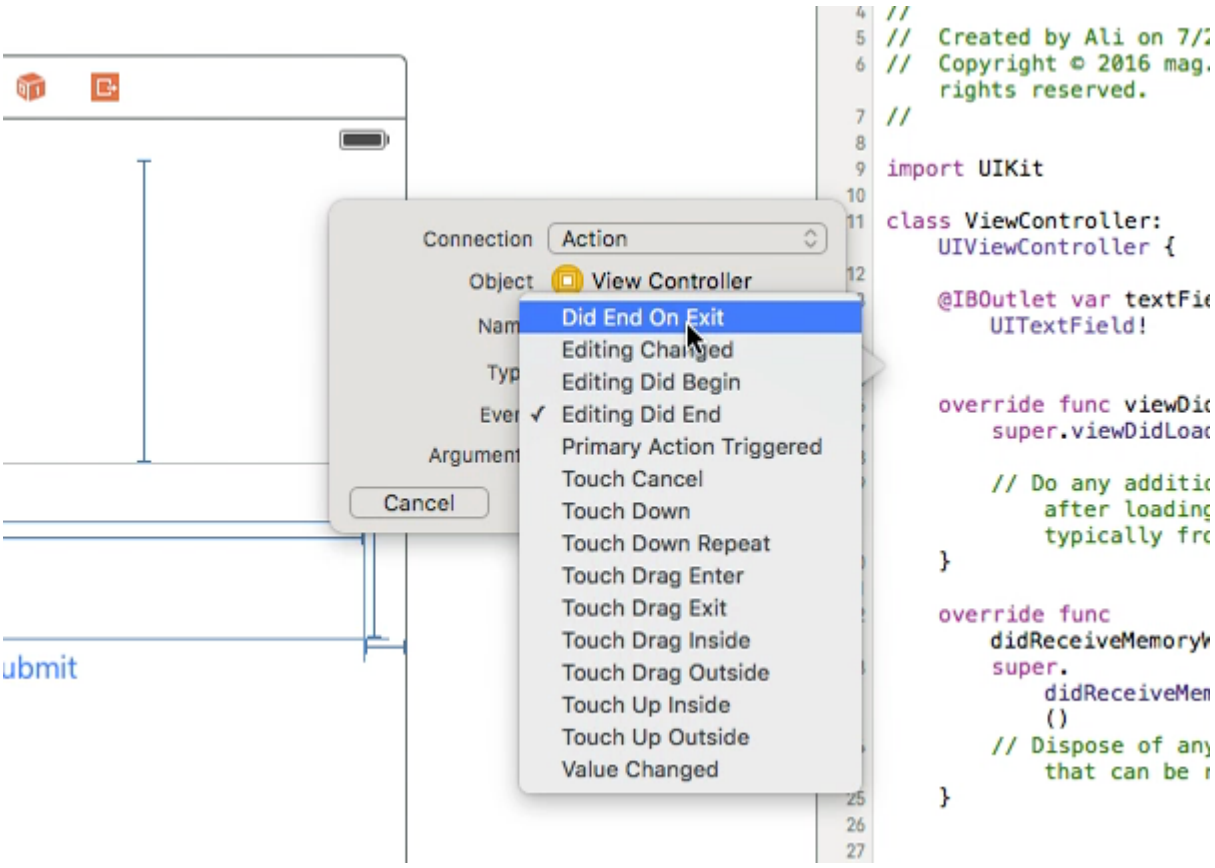
Отключить клавиатуру

стриж

Ctrl + Перетащите из поля UItextfield в MainStoryboard в класс ViewController и создайте UITextField Outlet







После этого снова выберите UItextField и Ctrl + перетащите в класс ViewController, но на этот раз выберите « **Действие»,** а в хранилище выберите « **Завершить на выходе»,** затем нажмите «Подключиться».

в действии, которое вы только что создали, введите имя вашего UItextField `.resignFirstResponder()`

```
   @IBAction func textFieldResign(sender: AnyObject) {
        yourTextFieldName.resignFirstResponder()
    }
```

Это будет касаться скрытия клавиатуры при нажатии клавиши возврата на клавиатуре.

Другой пример скрытия клавиатуры при нажатии клавиши возврата:

мы добавляем протокол `UITextFieldDelegate` рядом с `UIViewController`

в функции vieDidLoad мы добавляем `self.yourTextFieldName.delegate = self`

И, наконец, мы добавляем это

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
                yourTextFieldName.resignFirstResponder()
                return true
            }
```

Последний код:

```
class ViewController: UIViewController, UITextFieldDelegate  {

@IBOutlet var textField: UITextField!

    func textFieldShouldReturn(textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }

override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?){
    view.endEditing(true)
    super.touchesBegan(touches, withEvent: event)
}


override func viewDidLoad() {
    super.viewDidLoad()
    self.textField.delegate = self
        }
```

}

Objective-C

```
[textField resignFirstResponder];
```

Установить выравнивание

стриж

```
textField.textAlignment = .Center
```

Objective-C

```
[textField setTextAlignment: NSTextAlignmentCenter];
```

В этом примере мы установили `NSTextAlignment` в центр. Вы также можете установить значение `.Left` , `.Right` , `.Justified` и `.Natural` .

`.Natural` - это выравнивание по умолчанию для текущей локализации. Это означает, что для языков слева направо (например, английский) выравнивание равно `.Left` ; для языков справа налево, это. `.Right` .

KeyboardType

Чтобы изменить внешний вид клавиатуры, следующие типы могут быть установлены индивидуально для каждого свойства `UITextFields` : `keyboardType`

```
typedef NS_ENUM(NSInteger, UIKeyboardType) {
    UIKeyboardTypeDefault,                // Default type for the current input method.
    UIKeyboardTypeASCIICapable,           // Displays a keyboard which can enter ASCII
characters, non-ASCII keyboards remain active
    UIKeyboardTypeNumbersAndPunctuation,  // Numbers and assorted punctuation.
    UIKeyboardTypeURL,                    // A type optimized for URL entry (shows . / .com
prominently).
    UIKeyboardTypeNumberPad,              // A number pad (0-9). Suitable for PIN entry.
    UIKeyboardTypePhonePad,               // A phone pad (1-9, *, 0, #, with letters under the
numbers).
    UIKeyboardTypeNamePhonePad,           // A type optimized for entering a person's name or
phone number.
    UIKeyboardTypeEmailAddress,           // A type optimized for multiple email address entry
(shows space @ . prominently).
    UIKeyboardTypeDecimalPad NS_ENUM_AVAILABLE_IOS(4_1),   // A number pad with a decimal
point.
    UIKeyboardTypeTwitter NS_ENUM_AVAILABLE_IOS(5_0),      // A type optimized for twitter
text entry (easy access to @ #)
    UIKeyboardTypeWebSearch NS_ENUM_AVAILABLE_IOS(7_0),    // A default keyboard type with
URL-oriented addition (shows space . prominently).

    UIKeyboardTypeAlphabet = UIKeyboardTypeASCIICapable, // Deprecated
};
```

Перемещение прокрутки, когда UITextView становится первым респондентом

Наблюдайте за уведомлениями `UIKeyboardWillShowNotification` и `UIKeyboardWillHideNotification` , обновляйте `scrollView` содержимого `scrollView` соответствии с высотой клавиатуры, а

затем прокрутите до сосредоточенного `scrollView` управления.

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    // register for keyboard notifications
    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(keyboardWillShow:)
                                                 name:UIKeyboardWillShowNotification
                                               object:self.view.window];
    // register for keyboard notifications
    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(keyboardWillHide:)
                                                 name:UIKeyboardWillHideNotification
                                               object:self.view.window];
}

// Called when UIKeyboardWillShowNotification is sent
- (void)keyboardWillShow:(NSNotification*)notification
{
    // if we have no view or are not visible in any window, we don't care
    if (!self.isViewLoaded || !self.view.window) {
        return;
    }

    NSDictionary *userInfo = [notification userInfo];

    CGRect keyboardFrameInWindow;
    [[userInfo objectForKey:UIKeyboardFrameEndUserInfoKey] getValue:&keyboardFrameInWindow];

    // the keyboard frame is specified in window-level coordinates. this calculates the frame
as if it were a subview of our view, making it a sibling of the scroll view
    CGRect keyboardFrameInView = [self.view convertRect:keyboardFrameInWindow fromView:nil];

    CGRect scrollViewKeyboardIntersection = CGRectIntersection(_scrollView.frame,
keyboardFrameInView);
    UIEdgeInsets newContentInsets = UIEdgeInsetsMake(0, 0,
scrollViewKeyboardIntersection.size.height, 0);

    // this is an old animation method, but the only one that retains compaitiblity between
parameters (duration, curve) and the values contained in the userInfo-Dictionary.
    [UIView beginAnimations:nil context:NULL];
    [UIView setAnimationDuration:[[userInfo
objectForKey:UIKeyboardAnimationDurationUserInfoKey] doubleValue]];
    [UIView setAnimationCurve:[[userInfo objectForKey:UIKeyboardAnimationCurveUserInfoKey]
intValue]];

    _scrollView.contentInset = newContentInsets;
    _scrollView.scrollIndicatorInsets = newContentInsets;

    /*
     * Depending on visual layout, _focusedControl should either be the input field
(UITextField,..) or another element
     * that should be visible, e.g. a purchase button below an amount text field
     * it makes sense to set _focusedControl in delegates like -textFieldShouldBeginEditing:
if you have multiple input fields
     */
    if (_focusedControl) {
        CGRect controlFrameInScrollView = [_scrollView convertRect:_focusedControl.bounds
fromView:_focusedControl]; // if the control is a deep in the hierarchy below the scroll view,
```

```
this will calculate the frame as if it were a direct subview
        controlFrameInScrollView = CGRectInset(controlFrameInScrollView, 0, -10); // replace
10 with any nice visual offset between control and keyboard or control and top of the scroll
view.

        CGFloat controlVisualOffsetToTopOfScrollview = controlFrameInScrollView.origin.y -
_scrollView.contentOffset.y;
        CGFloat controlVisualBottom = controlVisualOffsetToTopOfScrollview +
controlFrameInScrollView.size.height;

        // this is the visible part of the scroll view that is not hidden by the keyboard
        CGFloat scrollViewVisibleHeight = _scrollView.frame.size.height -
scrollViewKeyboardIntersection.size.height;

        if (controlVisualBottom > scrollViewVisibleHeight) { // check if the keyboard will
hide the control in question
            // scroll up until the control is in place
            CGPoint newContentOffset = _scrollView.contentOffset;
            newContentOffset.y += (controlVisualBottom - scrollViewVisibleHeight);

            // make sure we don't set an impossible offset caused by the "nice visual offset"
            // if a control is at the bottom of the scroll view, it will end up just above the
keyboard to eliminate scrolling inconsistencies
            newContentOffset.y = MIN(newContentOffset.y, _scrollView.contentSize.height -
scrollViewVisibleHeight);

            [_scrollView setContentOffset:newContentOffset animated:NO]; // animated:NO
because we have created our own animation context around this code
        } else if (controlFrameInScrollView.origin.y < _scrollView.contentOffset.y) {
            // if the control is not fully visible, make it so (useful if the user taps on a
partially visible input field
            CGPoint newContentOffset = _scrollView.contentOffset;
            newContentOffset.y = controlFrameInScrollView.origin.y;

            [_scrollView setContentOffset:newContentOffset animated:NO]; // animated:NO
because we have created our own animation context around this code
        }
    }

    [UIView commitAnimations];
}


// Called when the UIKeyboardWillHideNotification is sent
- (void)keyboardWillHide:(NSNotification*)notification
{
    // if we have no view or are not visible in any window, we don't care
    if (!self.isViewLoaded || !self.view.window) {
        return;
    }

    NSDictionary *userInfo = notification.userInfo;

    [UIView beginAnimations:nil context:NULL];
    [UIView setAnimationDuration:[[userInfo
valueForKey:UIKeyboardAnimationDurationUserInfoKey] doubleValue]];
    [UIView setAnimationCurve:[[userInfo valueForKey:UIKeyboardAnimationCurveUserInfoKey]
intValue]];

    // undo all that keyboardWillShow-magic
    // the scroll view will adjust its contentOffset apropriately
```

```
    _scrollView.contentInset = UIEdgeInsetsZero;
    _scrollView.scrollIndicatorInsets = UIEdgeInsetsZero;

    [UIView commitAnimations];
}
```

Получить фокус клавиатуры и скрыть клавиатуру

Получить фокус

стриж

```
textField.becomeFirstResponder()
```

Objective-C

```
[textField becomeFirstResponder];
```

Уходить в отставку

стриж

```
textField.resignFirstResponder()
```

Objective-C

```
[textField resignFirstResponder];
```

Замените клавиатуру на UIPickerView

В некоторых случаях вы хотите показать своим пользователям `UIPickerView` с предопределенным содержимым для `UITextField` вместо клавиатуры.

Создание пользовательского UIPickerView

Сначала вам нужен пользовательский класс-оболочка для `UIPickerView` соответствующий протоколам `UIPickerViewDataSource` и `UIPickerViewDelegate` .

```
class MyPickerView: UIPickerView, UIPickerViewDataSource, UIPickerViewDelegate
```

Вам необходимо реализовать следующие методы для DataSource и Delegate:

```
public func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) ->
```

```
Int {
    if data != nil {
        return data!.count
    } else {
        return 0
    }
}

public func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 1
}

public func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent
component: Int) -> String? {
    if data != nil {
        return data![row]
    } else {
        return ""
    }
}
```

Для обработки данных `MyPickerView` нужны `data` свойств, `selectedValue` и `textFieldBeingEdited` :

```
/**
 The data for the `UIPickerViewDelegate`

 Always needs to be an array of `String`! The `UIPickerView` can ONLY display Strings
 */
public var data: [String]? {
    didSet {
        super.delegate = self
        super.dataSource = self
        self.reloadAllComponents()
    }
}

/**
 Stores the UITextField that is being edited at the moment
 */
public var textFieldBeingEdited: UITextField?

/**
 Get the selected Value of the picker
 */
public var selectedValue: String {
    get {
        if data != nil {
            return data![selectedRow(inComponent: 0)]
        } else {
            return ""
        }
    }
}
```

Подготовьте ViewController

`ViewController` , содержащий ваш текстовый фильтр, должен иметь свойство для вашего пользовательского `UIPickerView` . (Предполагая, что у вас уже есть другое свойство или

`@IBOutlet` содержащий ваш textField)

```
/**
 The picker view to present as keyboard
 */
var picker: MyPickerView?
```

В вашем `viewDidLoad()` вам нужно инициализировать `picker` и немного настроить его:

```
picker = MyPickerView()
picker?.autoresizingMask = [.flexibleHeight, .flexibleWidth]
picker?.backgroundColor = UIColor.white()

picker?.data = ["One", "Two", "Three", "Four", "Five"] //The data shown in the picker
```

Теперь вы можете добавить `MyPicker` качестве `inputView` вашего `UITextField` :

```
textField.inputView = picker
```

Отклонение клавиатуры-подборщика

Теперь вы заменили клавиатуру на `UIPickerView` , но нет возможности ее отклонить. Это можно сделать с помощью custom `.inputAccessoryView` :

Добавьте свойство `pickerAccessory` в `ViewController` .

```
/**
 A toolbar to add to the keyboard when the `picker` is presented.
 */
var pickerAccessory: UIToolbar?
```

В `viewDidLoad()` вам необходимо создать `UIToolbar` для `inputAccessoryView` :

```
pickerAccessory = UIToolbar()
pickerAccessory?.autoresizingMask = .flexibleHeight

//this customization is optional
pickerAccessory?.barStyle = .default
pickerAccessory?.barTintColor = UIColor.red()
pickerAccessory?.backgroundColor = UIColor.red()
pickerAccessory?.isTranslucent = false
```

Вы должны установить рамку своей панели инструментов. Чтобы вписаться в дизайн iOS, рекомендуется использовать высоту `44.0` :

```
var frame = pickerAccessory?.frame
frame?.size.height = 44.0
pickerAccessory?.frame = frame!
```

Для хорошего пользовательского интерфейса вы должны добавить две кнопки («Готово»

и «Отмена»), но она также будет работать только с тем, кто отклоняет клавиатуру.

```
let cancelButton = UIBarButtonItem(barButtonSystemItem: .cancel, target: self, action:
#selector(ViewController.cancelBtnClicked(_:)))
cancelButton.tintColor = UIColor.white()
let flexSpace = UIBarButtonItem(barButtonSystemItem: .flexibleSpace, target: nil, action: nil)
//a flexible space between the two buttons
let doneButton = UIBarButtonItem(barButtonSystemItem: .done, target: self, action:
#selector(ViewController.doneBtnClicked(_:)))
doneButton.tintColor = UIColor.white()

//Add the items to the toolbar
pickerAccessory?.items = [cancelButton, flexSpace, doneButton]
```

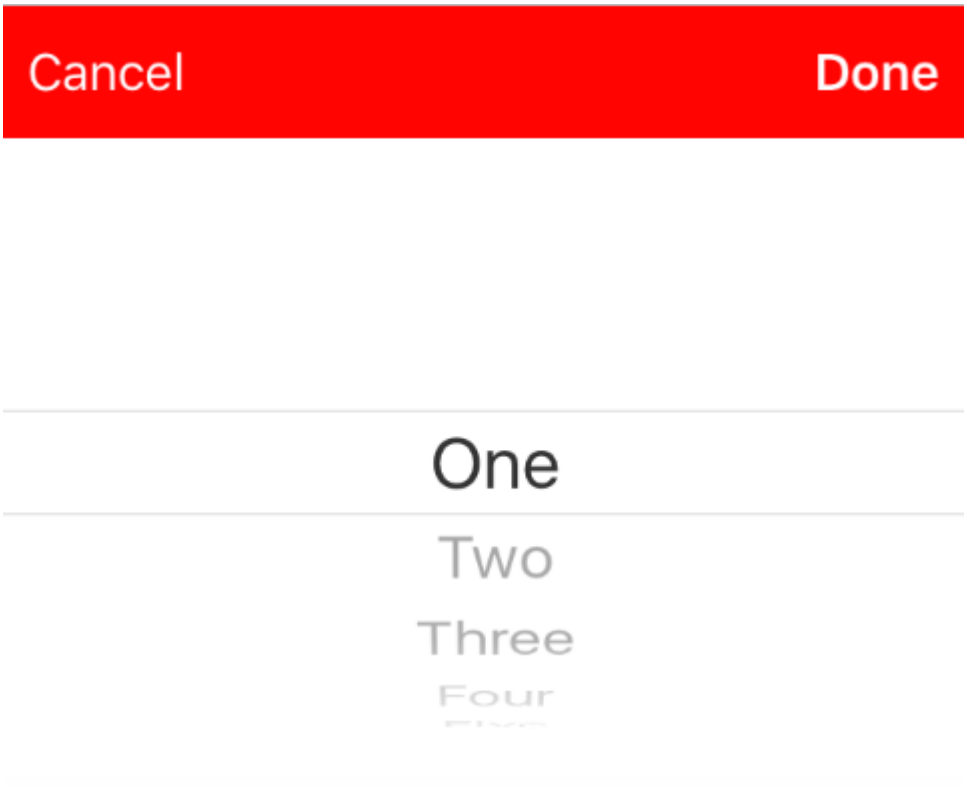
Теперь вы можете добавить панель инструментов в качестве `inputAccessoryView`

```
textField.inputAccessoryView = pickerAccessory
```

Прежде чем вы сможете создать свой проект, вам нужно реализовать методы, вызывающие кнопки:

```
/**
 Called when the cancel button of the `pickerAccessory` was clicked. Dismsses the picker
 */
func cancelBtnClicked(_ button: UIBarButtonItem?) {
    textField?.resignFirstResponder()
}

/**
 Called when the done button of the `pickerAccessory` was clicked. Dismisses the picker and
puts the selected value into the textField
 */
func doneBtnClicked(_ button: UIBarButtonItem?) {
    textField?.resignFirstResponder()
    textField.text = picker?.selectedValue
}
```

Запустите проект, коснитесь `textField` и вместо клавиатуры вы увидите такой сборщик:



Выберите значение программно (необязательно)

Если вы не хотите, чтобы первая строка была выбрана автоматически, вы можете установить выбранную строку, как в `UIPickerView` :

```
picker?.selectRow(3, inComponent: 0, animated: false) //Will select the row at index 3
```

Отключите клавиатуру, когда пользователь нажимает кнопку возврата

Создайте свой контроллер просмотра для управления редактированием текста для текстового поля.

```
class MyViewController: UITextFieldDelegate {

    override viewDidLoad() {
        super.viewDidLoad()

        textField.delegate = self
    }

}
```

`textFieldShouldReturn` вызывается каждый раз, когда нажата кнопка возврата на клавиатуре.

Swift:

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
    textField.resignFirstResponder()
```

```
    return true;
}
```

Objective-C:

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField {
   [textField resignFirstResponder];
   return true;
}
```

Получение и установка положения курсора

Полезная информация

В самом начале текста текстового поля:

```
let startPosition: UITextPosition = textField.beginningOfDocument
```

Самый конец текста текстового поля:

```
let endPosition: UITextPosition = textField.endOfDocument
```

Текущий выбранный диапазон:

```
let selectedRange: UITextRange? = textField.selectedTextRange
```

Получить позицию курсора

```
if let selectedRange = textField.selectedTextRange {

    let cursorPosition = textField.offsetFromPosition(textField.beginningOfDocument,
toPosition: selectedRange.start)

    print("\(cursorPosition)")
}
```

Установка положения курсора

Чтобы установить положение, все эти методы фактически устанавливают диапазон с одинаковыми начальными и конечными значениями.

К началу

```
let newPosition = textField.beginningOfDocument
textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

К концу

```
let newPosition = textField.endOfDocument
textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

В одну позицию слева от текущей позиции курсора

```
// only if there is a currently selected range
if let selectedRange = textField.selectedTextRange {

    // and only if the new position is valid
    if let newPosition = textField.positionFromPosition(selectedRange.start, inDirection:
UITextLayoutDirection.Left, offset: 1) {

        // set the new position
        textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
    }
}
```

В произвольное положение

Начните с начала и переместите 5 символов вправо.

```
let arbitraryValue: Int = 5
if let newPosition = textField.positionFromPosition(textField.beginningOfDocument,
inDirection: UITextLayoutDirection.Right, offset: arbitraryValue) {

    textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
}
```

связанные с

Выбрать весь текст

```
textField.selectedTextRange = textField.textRangeFromPosition(textField.beginningOfDocument,
toPosition: textField.endOfDocument)
```

Выберите диапазон текста

```
// Range: 3 to 7
let startPosition = textField.positionFromPosition(textField.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: 3)
let endPosition = textField.positionFromPosition(textField.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: 7)
```

```
if startPosition != nil && endPosition != nil {
    textField.selectedTextRange = textField.textRangeFromPosition(startPosition!, toPosition:
endPosition!)
}
```

Вставить текст в текущую позицию курсора

```
textField.insertText("Hello")
```

Заметки

- Этот пример изначально исходит из этого ответа переполнения стека .
- Этот ответ использует текстовое поле, но те же понятия применимы к `UITextView` .
- Используйте `textField.becomeFirstResponder()` чтобы уделить внимание текстовому полю и `textField.becomeFirstResponder()` клавиатуру.
- См. Этот ответ для получения текста в определенном диапазоне.

связанные с

- Как создать диапазон в Swift (косвенно относится к вопросу о том, почему мы должны использовать `selectedTextRange` здесь, а не только `selectedRange` Range)

Скрыть мигающую каретку

Чтобы скрыть мигающий кареток, вам необходимо переопределить caretRectForPosition UITextField и вернуть CGRectZero.

Swift 2.3 <

```
public override func caretRectForPosition(position: UITextPosition) -> CGRect {
    return CGRectZero
}
```

Swift 3

```
override func caretRect(for position: UITextPosition) -> CGRect {
    return CGRect.zero
}
```

Objective-C

```
- (CGRect) caretRectForPosition:(UITextPosition*) position{
return CGRectZero;
}
```

Изменить цвет и шрифт заполнителя

Мы можем изменить стиль заполнителя, установив `attributedPlaceholder` (а `NSAttributedString`).

```
var placeholderAttributes = [String: AnyObject]()
placeholderAttributes[NSForegroundColorAttributeName] = color
placeholderAttributes[NSFontAttributeName] = font

if let placeholder = textField.placeholder {
    let newAttributedPlaceholder = NSAttributedString(string: placeholder, attributes:
placeholderAttributes)
    textField.attributedPlaceholder = newAttributedPlaceholder
}
```

В этом примере мы меняем только `color` и `font` . Вы можете изменить другие свойства, такие как подчеркивание или стиль зачеркивания. `NSAttributedString` свойств, которые можно изменить, обратитесь к `NSAttributedString` .

Создать UITextField

Инициализируйте `UITextField` с помощью CGRect в качестве фрейма:

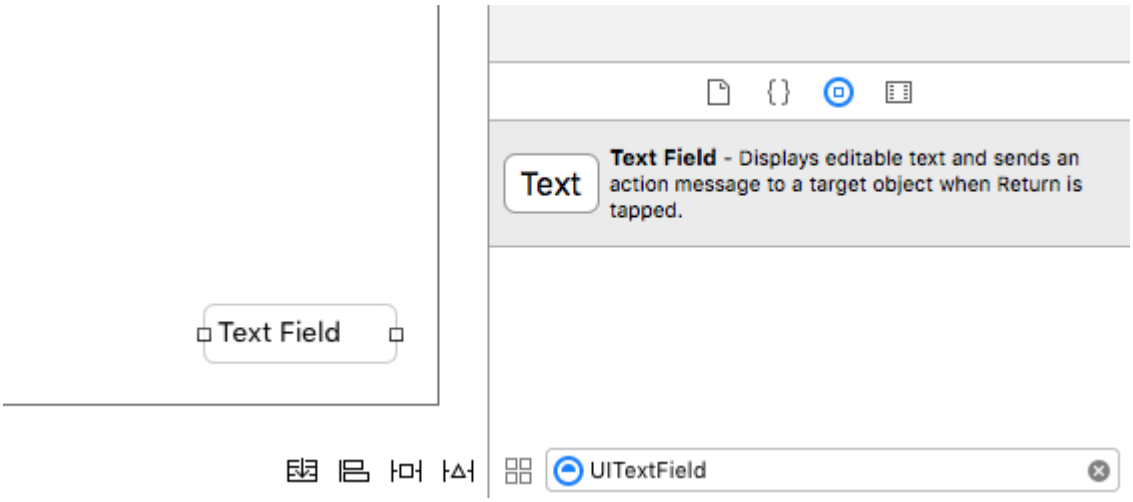
стриж

```
let textfield = UITextField(frame: CGRect(x: 0, y: 0, width: 200, height: 21))
```

Objective-C

```
UITextField *textField = [[UITextField alloc] initWithFrame:CGRectMake(0, 0, 200, 21)];
```

Вы также можете создать `UITextField` в Interface Builder:



Прочитайте UITextField онлайн: https://riptutorial.com/ru/ios/topic/1630/uitextfield

глава 106: UITextView

Examples

Изменить текст

стриж

```
textView.text = "Hello, world!"
```

Objective-C:

```
textView.text = @"Hello, world!";
```

Установить атрибутный текст

```
// Modify some of the attributes of the attributed string.
let attributedText = NSMutableAttributedString(attributedString: textView.attributedText!)

// Use NSString so the result of rangeOfString is an NSRange.
let text = textView.text! as NSString

// Find the range of each element to modify.
let tintedRange = text.range(of: NSLocalizedString("tinted", comment: ""))
let highlightedRange = text.range(of: NSLocalizedString("highlighted", comment: ""))

// Add tint.
attributedText.addAttribute(NSForegroundColorAttributeName, value: UIColor.blue, range:
tintedRange)

// Add highlight.
attributedText.addAttribute(NSBackgroundColorAttributeName, value: UIColor.yellow, range:
highlightedRange)

textView.attributedText = attributedText
```

Изменить выравнивание текста

стриж

```
textView.textAlignment = .left
```

Objective-C

```
textView.textAlignment = NSTextAlignmentLeft;
```

Методы UITextViewDelegate

Ответ на редактирование уведомлений

- `textViewShouldBeginEditing(_:)`
- `textViewDidBeginEditing(_:)`
- `textViewShouldEndEditing(_:)`
- `textViewDidEndEditing(_:)`

Ответ на изменение текста

- `textView(_:shouldChangeTextIn:replacementText:)`
- `textViewDidChange(_:)`

Ответ на URL

- `textView(_: UITextView, shouldInteractWithURL: NSURL, inRange: NSRange) -> Bool`

Изменить шрифт

стриж

```
//System Font
textView.font = UIFont.systemFont(ofSize: 12)

//Font of your choosing
textView.font = UIFont(name: "Font Name", size: 12)
```

Objective-C

```
//System Font
textView.font = [UIFont systemFontOfSize:12];

//Font of your choosing
textView.font = [UIFont fontWithName:@"Font Name" size:12];
```

Изменить цвет текста

стриж

```
textView.textColor = UIColor.red
```

Objective-C

```
textView.textColor = [UIColor redColor];
```

UITextView с текстом HTML

```
NSString *htmlString = @"<p> This is an <b>HTML</b> text</p>";
NSAttributedString *attributedString = [[NSMutableAttributedString alloc]
                                                    initWithData: [htmlString
dataUsingEncoding:NSUnicodeStringEncoding]
```

```
                                                        options: @{
NSDocumentTypeDocumentAttribute: NSHTMLTextDocumentType }
                                                        documentAttributes: nil
                                                        error: nil
                                                        ];
                _yourTextView.attributedText = attributedString;
                // If you want to modify the font
                field.font = [UIFont fontWithName:@"Raleway-Regular" size:15];
```

Автообнаружение ссылок, адресов, дат и т. Д.

`UITextView` имеет встроенную поддержку автоматического обнаружения множества данных. Данные, которые могут быть автоматически обнаружены в настоящее время, включают:

```
enum {
   UIDataDetectorTypePhoneNumber   = 1 << 0,
   UIDataDetectorTypeLink          = 1 << 1,
   UIDataDetectorTypeAddress       = 1 << 2,
   UIDataDetectorTypeCalendarEvent = 1 << 3,
   UIDataDetectorTypeNone          = 0,
   UIDataDetectorTypeAll           = NSUIntegerMax
};
```

Включение автоматического обнаружения

```
// you may add as many as you like by using the `|` operator between options
textView.dataDetectorTypes = (UIDataDetectorTypeLink | UIDataDetectorTypePhoneNumber);
```

Если включено, текст будет отображаться как гиперссылка в `UITextView`

Доступные данные

Чтобы разрешить `UITextView` по ссылке (что приведет к различным действиям в зависимости от типа данных), вы должны убедиться, что `UITextView` можно выбрать, но не редактировать и включить взаимодействие с пользователем

```
textView.editable = NO;
textView.selectable = YES;
textView.userInteractionEnabled = YES; // YES by default
```

Проверьте, нет ли пустого или нулевого

стриж

```
if let text = self.textView.text where !text.isEmpty {
    // Do stuff for text
} else {
```

```
    // Do stuff for nil text or empty string
}
```

Objective-C

```
if (self.textView.text.length > 0){
    // Do stuff for text
}   else {
    // Do stuff for nil text or empty string
}
```

Получение и настройка проводника

Полезная информация

В самом начале текста текстового поля:

```
let startPosition: UITextPosition = textView.beginningOfDocument
```

Самый конец текста текстового поля:

```
let endPosition: UITextPosition = textView.endOfDocument
```

Текущий выбранный диапазон:

```
let selectedRange: UITextRange? = textView.selectedTextRange
```

Получить позицию курсора

```
if let selectedRange = textView.selectedTextRange {

    let cursorPosition = textView.offsetFromPosition(textView.beginningOfDocument, toPosition:
selectedRange.start)

    print("\(cursorPosition)")
}
```

Установка положения курсора

Чтобы установить положение, все эти методы фактически устанавливают диапазон с одинаковыми начальными и конечными значениями.

К началу

```
let newPosition = textView.beginningOfDocument
textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

К концу

```
let newPosition = textView.endOfDocument
textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

В одну позицию слева от текущей позиции курсора

```
// only if there is a currently selected range
if let selectedRange = textView.selectedTextRange {

    // and only if the new position is valid
    if let newPosition = textView.positionFromPosition(selectedRange.start, inDirection:
UITextLayoutDirection.Left, offset: 1) {

        // set the new position
        textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
    }
}
```

В произвольное положение

Начните с начала и переместите 5 символов вправо.

```
let arbitraryValue: Int = 5
if let newPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: arbitraryValue) {

    textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
}
```

связанные с

Выбрать весь текст

```
textView.selectedTextRange = textView.textRangeFromPosition(textView.beginningOfDocument,
toPosition: textView.endOfDocument)
```

Выберите диапазон текста

```
// Range: 3 to 7
let startPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: 3)
let endPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: 7)
```

```
if startPosition != nil && endPosition != nil {
    textView.selectedTextRange = textView.textRangeFromPosition(startPosition!, toPosition:
endPosition!)
}
```

Вставить текст в текущую позицию курсора

```
textView.insertText("Hello")
```

Заметки

- Этот пример изначально исходит из адаптации этого ответа на переполнение стека .
- Этот ответ использует текстовое поле, но те же понятия применимы к `UITextView` .
- Используйте `textView.becomeFirstResponder()` чтобы уделить внимание текстовому полю и `textView.becomeFirstResponder()` клавиатуру.
- См. Этот ответ для получения текста в определенном диапазоне.

связанные с

- Как создать диапазон в Swift (косвенно относится к вопросу о том, почему мы должны использовать `selectedTextRange` здесь, а не только `selectedRange` Range)

Удалите дополнительные прокладки, чтобы они соответствовали точно измеренному тексту.

По умолчанию `UITextView` имеет дополнительные прокладки. Иногда это раздражает, особенно если вы хотите измерить некоторый текст без экземпляра представления и поместить их в какой-либо области точно.

Сделайте это, чтобы удалить такие прокладки.

```
messageTextView.textContainerInset = UIEdgeInsetsZero
messageTextView.textContainer.lineFragmentPadding = 0
```

Теперь вы можете измерять размер текста с помощью `NSAttributedString.boundingRectWithSize(...)` и изменять размер `UITextView` только для соответствия его тексту.

```
let budget = getSomeCGSizeBudget()
let text = getSomeAttributedString()
let textSize = text.boundingRectWithSize(budget, options: [.UsesLineFragmentOrigin,
```

```
.UsesFontLeading], context: nil).size
messageTextView.frame.size = textSize // Just fits.
```

Прочитайте UITextView онлайн: https://riptutorial.com/ru/ios/topic/1043/uitextview

глава 107: UIView

Синтаксис

1. // Объектив-С
2. [UIView new] // Получить объект представления, выделенный и инициализированный
3. [[UIView alloc] initWithFrame: (Pass CGRect)] // Получить представление, выделенное и инициализированное фреймом
4. [[UIView alloc] init] // Получить объект представления, выделенный и инициализированный
5. // Swift
6. UIView () // Создает экземпляр UIView с флагом CGRect.zero
7. UIView (frame: CGRect) // Создает экземпляр UIView, определяющий кадр
8. UIView.addSubview (UIView) // добавление другого экземпляра UIView в качестве подзадачи
9. UIView.hidden // Получить или установить видимость вида
10. UIView.alpha // Получить или установить непрозрачность представления
11. UIView.setNeedsLayout () // Заставляет просмотр обновлять макет

замечания

Класс **UIView** определяет прямоугольную область на экране и интерфейсы для управления содержимым в этой области. Во время выполнения объект представления обрабатывает рендеринг любого содержимого в своей области, а также обрабатывает любые взаимодействия с этим контентом.

Examples

Создать UIView

Objective-C

```
CGRect myFrame = CGRectMake(0, 0, 320, 35)
UIView *view = [[UIView alloc] initWithFrame:myFrame];
```

```
//Alternative way of defining the frame
UIView *view = [[UIView alloc] init];
CGRect myFrame = view.frame;
myFrame.size.width = 320;
myFrame.size.height = 35;
myFrame.origin.x = 0;
myFrame.origin.y = 0;
view.frame = myFrame;
```

стриж

```
let myFrame = CGRect(x: 0, y: 0, width: 320, height: 35)
let view = UIView(frame: myFrame)
```

Сделать округленный вид

Чтобы сделать округленный `UIView` , укажите `cornerRadius` для `layer` .

Это также применимо к любому классу, который наследуется от `UIView` , например `UIImageView` .

Программный

Быстрый код

```
someImageView.layoutIfNeeded()
someImageView.clipsToBounds = true
someImageView.layer.cornerRadius = 10
```

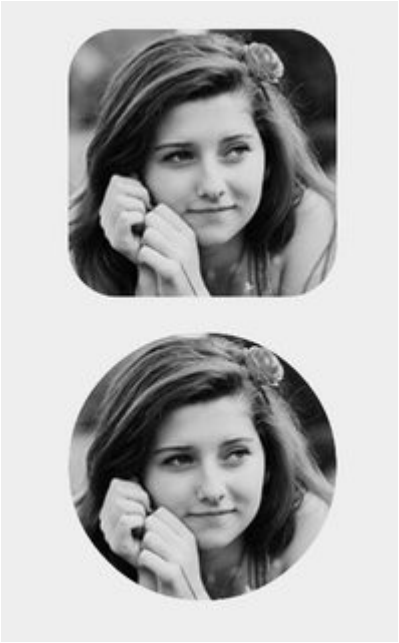
Код цели-C

```
[someImageView layoutIfNeeded];
someImageView.clipsToBounds = YES;
someImageView.layer.cornerRadius = 10;
```

пример

```
//Swift code
topImageView.layoutIfNeeded()
bottomImageView.layoutIfNeeded()
topImageView.clipsToBounds = true
topImageView.layer.cornerRadius = 10
bottomImageView.clipsToBounds = true
bottomImageView.layer.cornerRadius = bottomImageView.frame.width / 2

//Objective-C code
[topImageView layoutIfNeeded]
[bottomImageView layoutIfNeeded];
topImageView.clipsToBounds = YES;
topImageView.layer.cornerRadius = 10;
bottomImageView.clipsToBounds = YES;
bottomImageView.cornerRadius = CGRectGetWidth(bottomImageView.frame) / 2;
```

Вот результат, показывающий эффект округленного вида с использованием указанного углового радиуса:



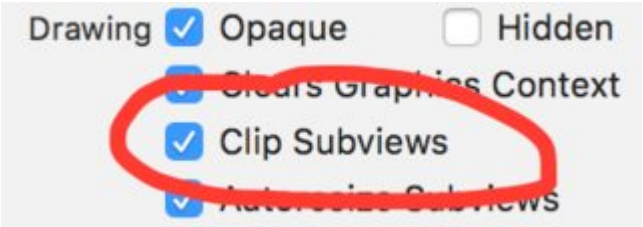
Заметка

Для этого вам необходимо включить структуру QuartzCore.

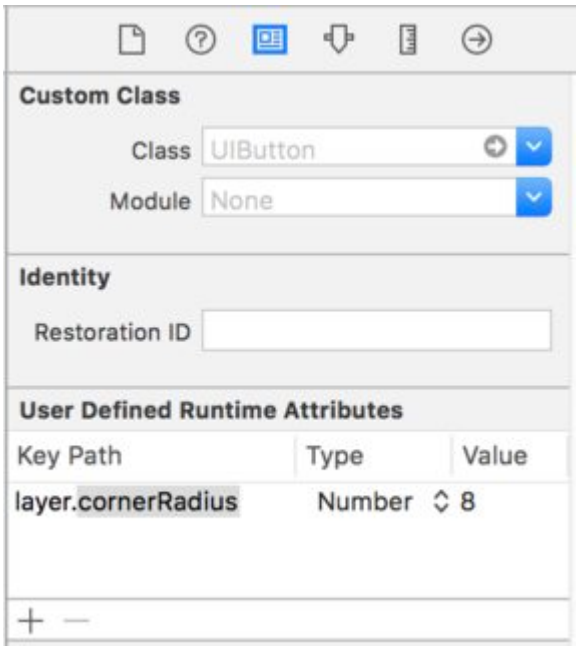
```
#import <QuartzCore/QuartzCore.h>
```

Конфигурация раскадровки

Эффект закругленного обзора также может быть достигнут `non-programmatically` , установив соответствующие свойства в **Storyboard** .



Поскольку свойства `layer` не отображаются в Storyboard, вам необходимо изменить атрибут `cornerRadius` помощью атрибута User Defined Runtime Attributes.



Быстрое расширение

Вы можете использовать это удобное расширение для применения округленного представления, если оно имеет такую же ширину и высоту.

```
extension UIView {
    @discardableResult
    public func setAsCircle() -> Self {
        self.clipsToBounds = true
        let frameSize = self.frame.size
        self.layer.cornerRadius = min(frameSize.width, frameSize.height) / 2.0
        return self
    }
}
```

Чтобы использовать его:

```
yourView.setAsCircle()
```

Как сделать снимок

Вы можете сделать снимок из `UIView` следующим образом:

стриж

```
let snapshot = view.snapshotView(afterScreenUpdates: true)
```

Objective-C

```
UIView *snapshot = [view snapshotViewAfterScreenUpdates: YES];
```

Использование IBInspectable и IBDesignable

Один (или два) из самых новых возможностей в последних версиях Xcode являются `IBInspectable` свойства и `IBDesignable` `UIView` s. Они не имеют ничего общего с функциональностью вашего приложения, но вместо этого влияют на опыт разработчика в Xcode. Цель состоит в том, чтобы визуально проверять пользовательские представления в приложении iOS без его запуска. Предположим, что у вас есть пользовательское представление с именем `CustomView` которое наследуется от `UIView` . В этом пользовательском представлении отобразится строка текста с обозначенным цветом. Вы также можете не отображать текст. Нам понадобятся три свойства:

```
var textColor: UIColor = UIColor.blackColor()
var text: String?
var showText: Bool = true
```

Затем мы можем переопределить функцию `drawRect` в классе:

```
if showText {
    if let text = text {
        let s = NSString(string: text)
        s.drawInRect(rect,
            withAttributes: [
                NSForegroundColorAttributeName: textColor,
                NSFontAttributeName: UIFont(name: "Helvetica Neue", size: 18)!
            ])
    }
}
```

Предполагая, что свойство `text` установлено, это приведет к рисованию строки в верхнем левом углу представления при запуске приложения. Проблема в том, что мы не будем знать, как это выглядит, без запуска приложения. Здесь `IBInspectable` и `IBDesignable` входят. `IBInspectable` позволяет визуально устанавливать значения свойств представления в Xcode, как и со встроенными элементами управления. `IBDesignable` покажет нам визуальный предварительный просмотр в раскадровке. Вот как должен выглядеть класс:

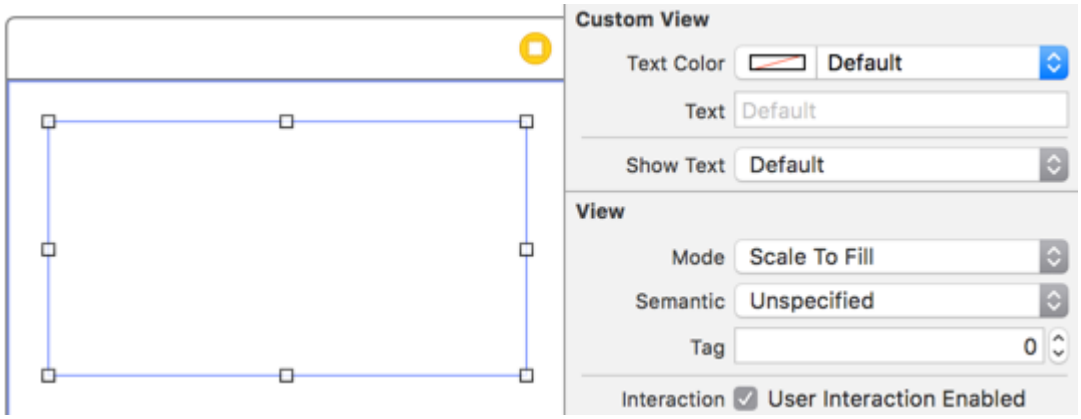
```
@IBDesignable
class CustomView: UIView {
    @IBInspectable var textColor: UIColor = UIColor.blackColor()
    @IBInspectable var text: String?
    @IBInspectable var showText: Bool = true

    override func drawRect(rect: CGRect) {
        // ...
    }
}
```

Или в Objective C:

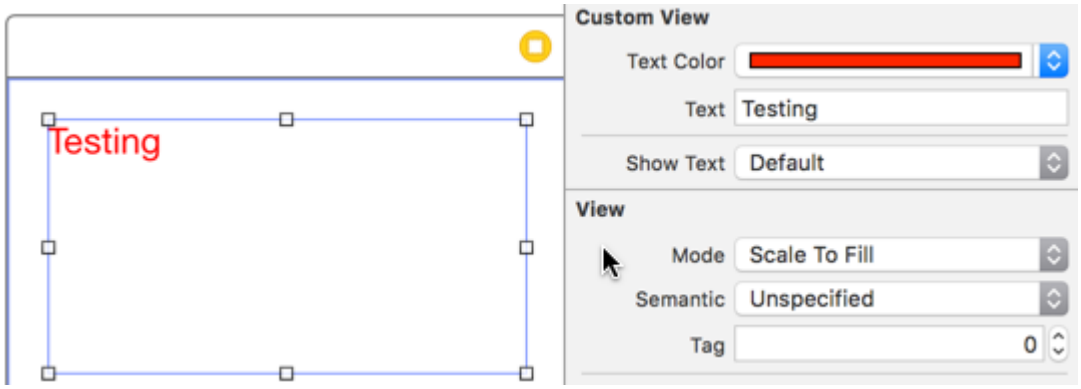
```
IB_DESIGNABLE
@interface CustomView: UIView
```

```
@property (nonatomic, strong) IBInspectable UIColor* textColor;
@property (nonatomic, strong) IBInspectable NSString* text;
@property (nonatomic, assign) IBInspectable BOOL showText;

@end

@implementation CustomView

- (instancetype)init {
    if(self = [super init]) {
        self.textColor = [UIColor blackColor];
        self.showText = YES;
    }
    return self;
}

- (void)drawRect:(CGRect)rect {
    //...
}

@end
```

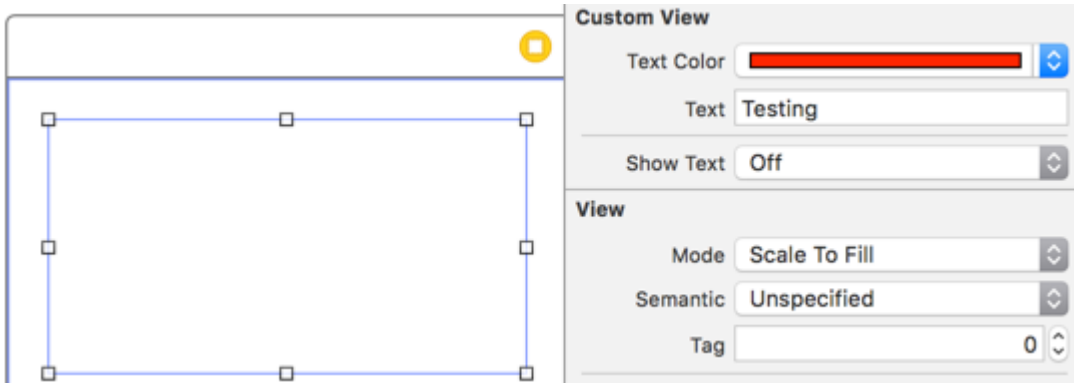
На следующих снимках экрана показано, что происходит в Xcode. Первое - это то, что происходит после добавления пересмотренного класса. Обратите внимание, что для трех свойств есть три новых элемента интерфейса. *Цвет текста* отобразит подборку цветов, *текст* - это только поле ввода, а « *Показать текст»* предоставит нам опции « `Off` И « `On` Которые являются `false` и `true` соответственно.



Далее следует изменить *цвет текста* на красный с помощью панели выбора цвета. Кроме того, был предоставлен некоторый текст, чтобы показать функцию `drawRect` . Обратите внимание, что представление в Interface Builder также обновлено.

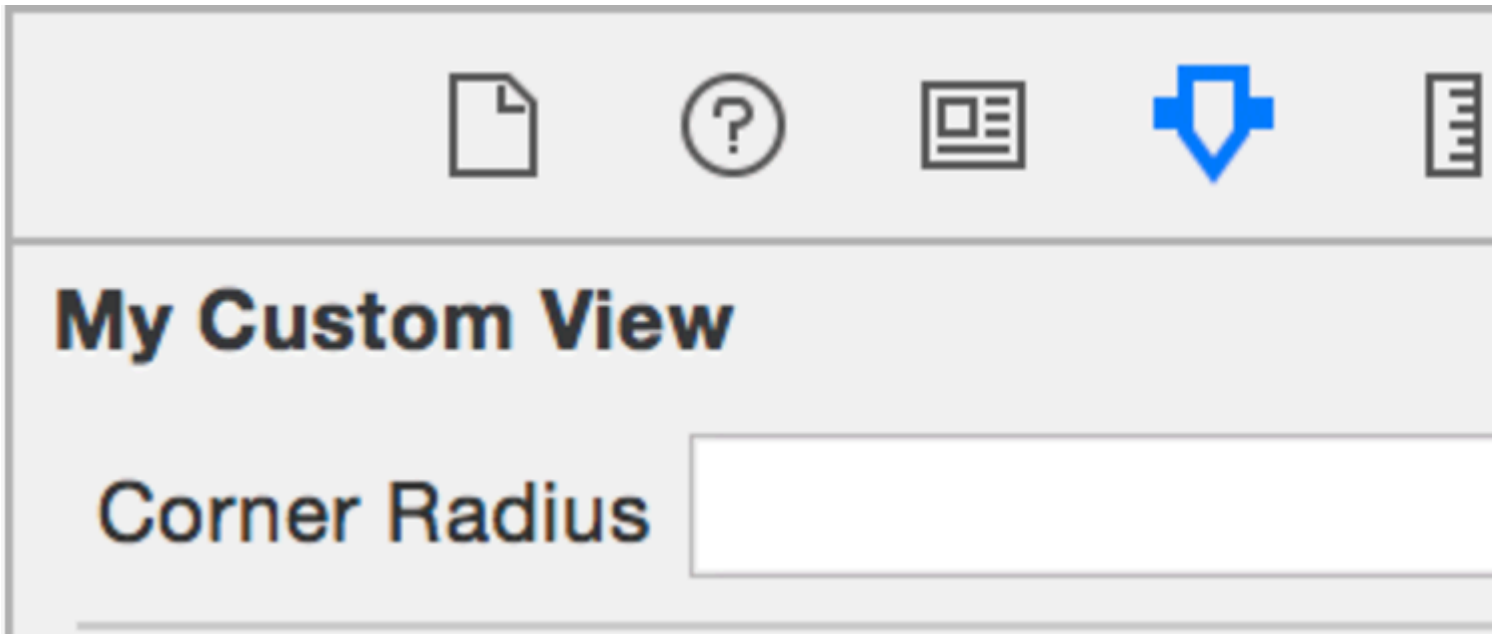


Наконец, установка « *Показать текст* в положение« `Off` В инспекторе свойств приводит к исчезновению текстового отображения в« Интерфейс-Builder ».



Тем не менее, мы все придумали ситуацию, когда нам нужно создать округленный `UIView` на нескольких представлениях в вашей `Storyboard` Вместо объявления `IBDesignable` для всех просмотров `Storyboard` лучше создать `Extension UIView` и получить пользовательский интерфейс, построенный только для вашего `UIView` через проект, чтобы создать закругленный вид, установив угловой радиус. Конфигурируемый радиус границы на любом UIView, который вы создаете в раскадровке.

```
extension UIView {

    @IBInspectable var cornerRadius:CGFloat {
        set {
            layer.cornerRadius = newValue
            clipsToBounds = newValue > 0
        }
        get {
            return layer.cornerRadius
        }
    }

}
```



Анимация UIView

```
let view = UIView(frame: CGRect(x: 0, y: 0, width: 100, height: 100))
view.backgroundColor = UIColor.orange
self.view.addSubview(view)
UIView.animate(withDuration: 0.75, delay: 0.5, options: .curveEaseIn, animations: {
    //This will cause view to go from (0,0) to
    // (self.view.frame.origin.x,self.view.frame.origin.y)
    view.frame.origin.x = self.view.frame.origin.x
    view.frame.origin.y = self.view.frame.origin.y
}) { (finished) in
    view.backgroundColor = UIColor.blueColor()
}
```

Расширение UIView для атрибутов размера и кадра

Если мы хотим получить x-кординат происхождения представления, тогда нам нужно написать так:

```
view.frame.origin.x
```

Для ширины нам нужно написать:

```
view.frame.size.width
```

Но если мы добавим простое расширение к `UIView` , мы можем получить все атрибуты очень просто, например:

```
view.x
view.y
view.width
view.height
```

Это также поможет установить такие атрибуты, как:

```
view.x = 10
view.y = 10
view.width = 100
view.height = 200
```

И простое расширение было бы:

```
extension UIView {

    var x: CGFloat {
        get {
            return self.frame.origin.x
        }
        set {
            self.frame = CGRect(x: newValue, y: self.frame.origin.y, width:
self.frame.size.width, height: self.frame.size.height)
        }
    }

    var y: CGFloat {
        get {
            return self.frame.origin.y
        }
        set {
            self.frame = CGRect(x: self.frame.origin.x, y: newValue, width:
self.frame.size.width, height: self.frame.size.height)
        }
    }

    var width: CGFloat {
        get {
            return self.frame.size.width
        }
        set {
            self.frame = CGRect(x: self.frame.origin.x, y: self.frame.origin.y, width:
newValue, height: self.frame.size.height)
        }
    }

    var height: CGFloat {
        get {
            return self.frame.height
        }
        set {
            self.frame = CGRect(x: self.frame.origin.x, y: self.frame.origin.y, width:
self.frame.size.width, height: newValue)
        }
    }
}
```

Нам нужно добавить этот файл класса в проект, и он будет доступен для использования во всем проекте!

Программно управлять вводом и удалением UIView в другой UIView и из

него

Предположим, у вас есть `parentView` в который вы хотите вставить новый `subView` программно (например, когда вы хотите вставить `UIImageView` в представление `UIViewController`), чем вы можете сделать это, как `UIViewController` ниже.

Objective-C

```
[parentView addSubview:subView];
```

стриж

```
parentView.addSubview(subView)
```

Вы также можете добавить subView под другим `subView2` , который уже является подвидным представлением parentView, используя следующий код:

Objective-C

```
[parentView insertSubview:subView belowSubview:subView2];
```

стриж

```
parentView.insertSubview(subView, belowSubview: subView2)
```

Если вы хотите вставить его над `subView2` вы можете сделать это следующим образом:

Objective-C

```
[parentView insertSubview:subView aboveSubview:subView2];
```

стриж

```
parentView.insertSubview(subView, aboveSubview: subView2)
```

Если где-то в вашем коде вам нужно принести некоторый `subView` на передний `subView` , так что, прежде всего, в других `parentView` , вы можете сделать это следующим образом:

Objective-C

```
[parentView bringSubviewToFront:subView];
```

стриж

```
parentView.bringSubviewToFront(subView)
```

Наконец, если вы хотите удалить `subView` из `parentView` , вы можете сделать следующее:

Objective-C

```
[subView removeFromSuperview];
```

стриж

```
subView.removeFromSuperview()
```

Создание UIView с использованием Autolayout

```
UIView *view = [[UIView alloc] init];

[self.view addSubview:view];

//Use the function if you want to use height as constraint
[self addView:view onParentView:self.view withHeight:200.f];


//Use this function if you want to add view with respect to parent and should resize with it
[self addFullResizeConstraintForSubview:view addedOnParentView:self.view];
```

функции

Функция добавления вида с фиксированной высотой с использованием ограничений автоопределения

```
-(void)addView:(UIView*)subView onParentView:(UIView*)parentView withHeight:(CGFloat)height{

subView.translatesAutoresizingMaskIntoConstraints = NO;

NSLayoutConstraint *trailing =[NSLayoutConstraint
                                    constraintWithItem:subView
                                    attribute:NSLayoutAttributeTrailing
                                    relatedBy:NSLayoutRelationEqual
                                    toItem:parent
                                    attribute:NSLayoutAttributeTrailing
                                    multiplier:1.0
                                    constant:10.f];


NSLayoutConstraint *top = [NSLayoutConstraint
                                constraintWithItem:subView
                                attribute:NSLayoutAttributeTop
                                relatedBy:NSLayoutRelationEqual
                                toItem:parent
                                attribute:NSLayoutAttributeTop
                                multiplier:1.0
                                constant:10.f];

NSLayoutConstraint *leading = [NSLayoutConstraint
                                    constraintWithItem:subView
                                    attribute:NSLayoutAttributeLeading
```

```
                                    relatedBy:NSLayoutRelationEqual
                                    toItem:parent
                                    attribute:NSLayoutAttributeLeading
                                    multiplier:1.0
                                    constant:10.f];
[parent addConstraint:trailing];
[parent addConstraint:top];
[parent addConstraint:leading];

NSLayoutConstraint *heightConstraint =[NSLayoutConstraint
                                            constraintWithItem:subView
                                            attribute:NSLayoutAttributeHeight
                                            relatedBy:NSLayoutRelationEqual
                                            toItem:nil
                                            attribute:0
                                            multiplier:0.0
                                            constant:height];

[subView addConstraint:heightConstraint];

}
```

Функция добавляет ограничение полного размера для созданного UIView.

```
-(void)addFullResizeConstraintForSubview:(UIView*)subView
addedOnParentView:(UIView*)parentView{

subView.translatesAutoresizingMaskIntoConstraints = NO;

NSLayoutConstraint *trailing =[NSLayoutConstraint
                                    constraintWithItem:subView
                                    attribute:NSLayoutAttributeTrailing
                                    relatedBy:NSLayoutRelationEqual
                                    toItem:parent
                                    attribute:NSLayoutAttributeTrailing
                                    multiplier:1.0
                                    constant:10.f];


NSLayoutConstraint *top = [NSLayoutConstraint
                                constraintWithItem:subView
                                attribute:NSLayoutAttributeTop
                                relatedBy:NSLayoutRelationEqual
                                toItem:parent
                                attribute:NSLayoutAttributeTop
                                multiplier:1.0
                                constant:10.f];

NSLayoutConstraint *leading = [NSLayoutConstraint
                                    constraintWithItem:subView
                                    attribute:NSLayoutAttributeLeading
                                    relatedBy:NSLayoutRelationEqual
                                    toItem:parent
                                    attribute:NSLayoutAttributeLeading
                                    multiplier:1.0
                                    constant:10.f];

NSLayoutConstraint *bottom =[NSLayoutConstraint
                                  constraintWithItem:subView
                                  attribute:NSLayoutAttributeBottom
```

```
                                relatedBy:NSLayoutRelationEqual
                                toItem:parent
                                attribute:NSLayoutAttributeBottom
                                multiplier:1.0
                                constant:0.f];
[parent addConstraint:trailing];
[parent addConstraint:top];
[parent addConstraint:leading];
[parent addConstraint:bottom];
}
```

Использование внутреннего содержимого

При создании подкласса UIView внутренний размер содержимого помогает избежать установки жестко заданных ограничений высоты и ширины

основной взгляд на то, как класс может использовать этот

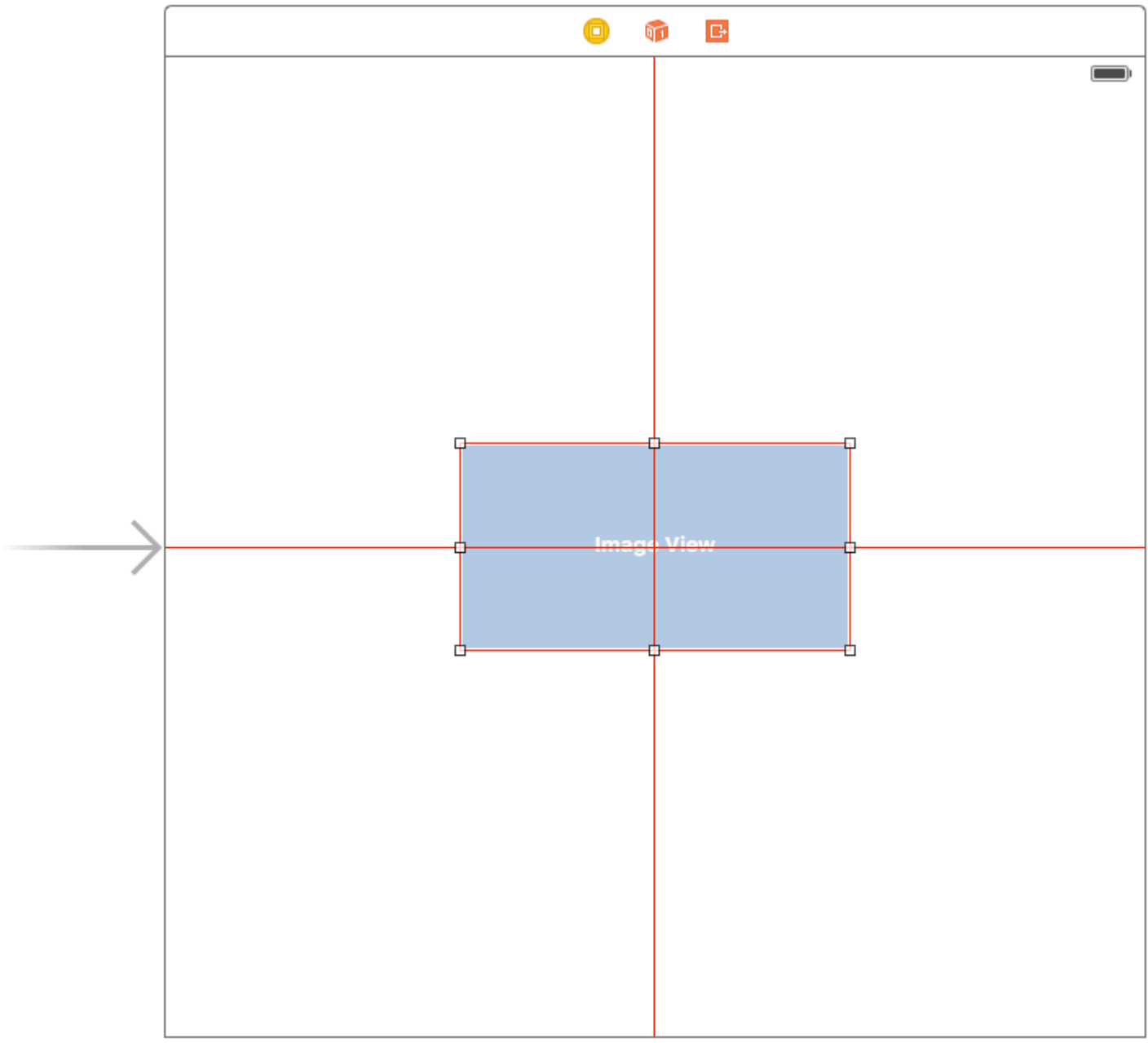
```
class ImageView: UIView {
    var image: UIImage {
        didSet {
            invalidateIntrinsicContentSize()
        }
    }
    // omitting initializers
    // convenience init(image: UIImage)

    override func intrinsicContentSize() -> CGSize {
        return CGSize(width: image.size.width, height: image.size.height)
    }
}
```

Если вы хотите только обеспечить один размер по существу, вы можете `UIViewNoIntrinsicMetric` значение `UIViewNoIntrinsicMetric` для значения, которое вы хотите игнорировать.

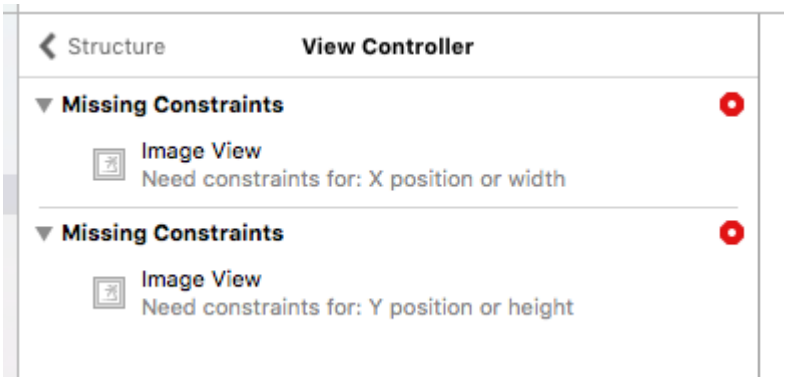
```
 override func intrinsicContentSize() -> CGSize {
     return CGSize(width: UIViewNoIntrinsicMetric, height: image.size.width)
 }
```

Преимущества при использовании с AutoLayout и Interface Builder

Можно было взять этот ImageView (или UIImageView) и установить горизонтальное выравнивание в центр наблюдения X и вертикальное выравнивание к центру наблюдения Y.

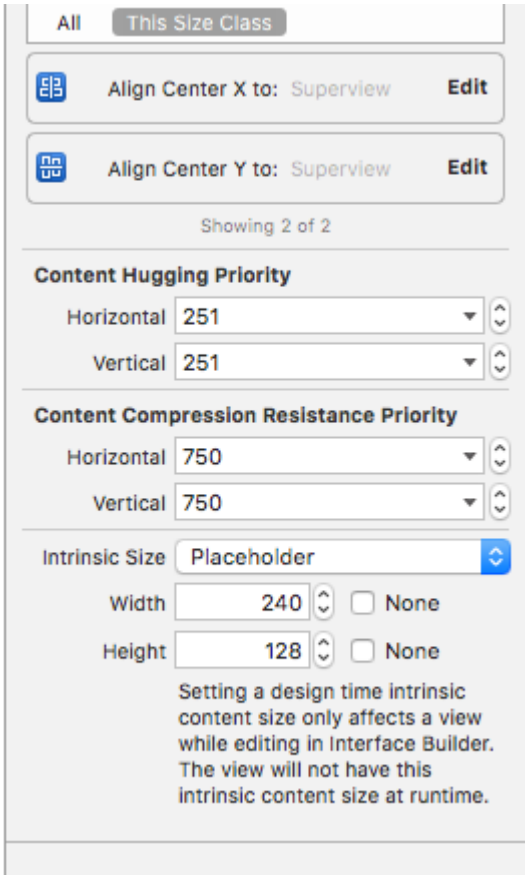


Разработчик интерфейса будет жаловаться на вас в этот момент, указав следующее предупреждение:



Именно здесь входит `Placeholder Intrinsic Size` .

Перейдя в панель инспектора размера и до раскрывающегося списка Intrinsic Size, вы можете переключить это значение с Default на Placeholder.



и теперь построитель интерфейсов удалит предыдущие предупреждения, и вы можете использовать этот размер, чтобы иметь динамически размерные представления, выложенные в построителе интерфейса.

Встряхните взгляд

```
extension UIView {
    func shake() {
        let animation = CAKeyframeAnimation(keyPath: "transform.translation.x")
        animation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionLinear)
        animation.duration = 0.6
        animation.values = [-10.0, 10.0, -7.0, 7.0, -5.0, 5.0, 0.0 ]
```

```
        layer.add(animation, forKey: "shake")
    }
}
```

Эта функция может использоваться, чтобы привлечь внимание к определенному виду, слегка встряхнув его.

Прочитайте UIView онлайн: https://riptutorial.com/ru/ios/topic/858/uiview

глава 108: UIViewController

Examples

Наследование

Подклассификация `UIControl` дает нам доступ к следующим методам:

- `beginTrackingWithTouch` вызывается, когда палец сначала прикасается к границам элемента управления.
- `continueTrackingWithTouch` вызывается многократно, когда палец скользит по элементу управления и даже за пределами границ элемента управления.
- `endTrackingWithTouch` вызывается, когда палец поднимается с экрана.

MyCustomControl.swift

```
import UIKit

// These are out self-defined rules for how we will communicate with other classes
protocol ViewControllerCommunicationDelegate: class {
    func myTrackingBegan()
    func myTrackingContinuing(location: CGPoint)
    func myTrackingEnded()
}

class MyCustomControl: UIControl {

    // whichever class wants to be notified of the touch events must set the delegate to
itself
    weak var delegate: ViewControllerCommunicationDelegate?

    override func beginTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?) -> Bool {

        // notify the delegate (i.e. the view controller)
        delegate?.myTrackingBegan()

        // returning true means that future events (like continueTrackingWithTouch and
endTrackingWithTouch) will continue to be fired
        return true
    }

    override func continueTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?) -> Bool
{

        // get the touch location in our custom control's own coordinate system
        let point = touch.locationInView(self)

        // Update the delegate (i.e. the view controller) with the new coordinate point
        delegate?.myTrackingContinuing(point)

        // returning true means that future events will continue to be fired
        return true
    }
```

```
    override func endTrackingWithTouch(touch: UITouch?, withEvent event: UIEvent?) {

        // notify the delegate (i.e. the view controller)
        delegate?.myTrackingEnded()
    }
}
```

ViewController.swift

Таким образом, диспетчер представлений настроен как делегат и реагирует на события касания от нашего пользовательского элемента управления.

```
import UIKit
class ViewController: UIViewController, ViewControllerCommunicationDelegate {

    @IBOutlet weak var myCustomControl: MyCustomControl!
    @IBOutlet weak var trackingBeganLabel: UILabel!
    @IBOutlet weak var trackingEndedLabel: UILabel!
    @IBOutlet weak var xLabel: UILabel!
    @IBOutlet weak var yLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        myCustomControl.delegate = self
    }

    func myTrackingBegan() {
        trackingBeganLabel.text = "Tracking began"
    }

    func myTrackingContinuing(location: CGPoint) {
        xLabel.text = "x: \(location.x)"
        yLabel.text = "y: \(location.y)"
    }

    func myTrackingEnded() {
        trackingEndedLabel.text = "Tracking ended"
    }
}
```

Заметки

- Альтернативные методы достижения одного и того же результата без подкласса включают добавление цели или использование распознавателя жестов.
- Нет необходимости использовать делегат с этими методами, если они используются только самим настраиваемым элементом управления. Мы могли бы просто добавить оператор `print` чтобы показать, как вызываются события. В этом случае код будет упрощен до

  ```
  import UIKit
  class MyCustomControl: UIControl {

      override func beginTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?) ->
  ```

```
Bool {
          print("Began tracking")
          return true
      }

      override func continueTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?)
-> Bool {
          let point = touch.locationInView(self)
          print("x: \(point.x), y: \(point.y)")
          return true
      }

      override func endTrackingWithTouch(touch: UITouch?, withEvent event: UIEvent?) {
          print("Ended tracking")
      }
  }
```

Создать экземпляр

стриж

```
let viewController = UIViewController()
```

Objective-C

```
UIViewController *viewController = [UIViewController new];
```

Установить программный вид

стриж

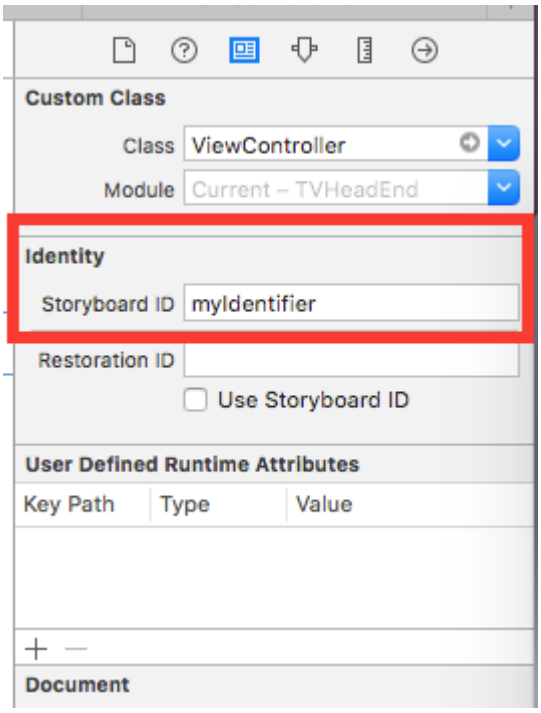
```
class FooViewController: UIViewController {

  override func loadView() {
    view = FooView()
  }

}
```

Создавать с помощью раскадровки

```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
```

С идентификатором :

Дайте сцену идентификатор раскадровки в инспекторе идентификации раскадровки.

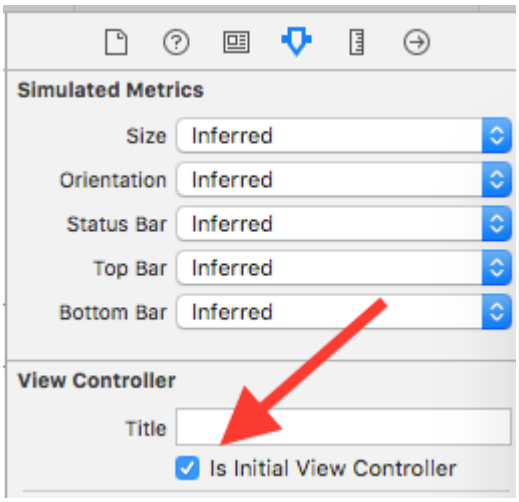


Введите код:

```
UIViewController *controller = [storyboard
instantiateViewControllerWithIdentifier:@"myIdentifier"];
```

Создайте исходный элемент управления представлением :

Внутри раскадровки выберите контроллер представления, затем выберите инспектор атрибутов, установите флажок «Исходный контроллер просмотра».




```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
UIViewController *controller = [storyboard instantiateInitialViewController];
```

Доступ к контроллеру просмотра контейнера

Когда контроллер представления представлен в контроллере панели вкладок, вы можете получить доступ к контроллеру панели вкладок следующим образом:

стриж

```
let tabBarController = viewController.tabBarController
```

Objective-C

```
UITabBarController *tabBarController = self.tabBarController;
```

Когда контроллер просмотра является частью навигационного стека, вы можете получить доступ к контроллеру навигации следующим образом:

стриж

```
let navigationController = viewController.navigationController
```

Objective-C

```
UINavigationController *navigationController = self.navigationController;
```

Добавление / удаление контроллера детского представления

Чтобы добавить контроллер детского представления:

```
- (void)displayContentController:(UIViewController *)vc {
    [self addChildViewController:vc];
    vc.view.frame = self.view.frame;
    [self.view addSubview:vc.view];
    [vc didMoveToParentViewController:self];
}
```

Чтобы удалить контроллер детского представления:

```
- (void)hideContentController:(UIViewController *)vc {
    [vc willMoveToParentViewController:nil];
    [vc.view removeFromSuperview];
    [vc removeFromParentViewController];
}
```

Прочитайте UIViewController онлайн: https://riptutorial.com/ru/ios/topic/1956/uiviewcontroller

глава 109: UIWebView

замечания

Функции делегата UIWebView: -

Задачи-C Declerations

```
- (BOOL)webView:(UIWebView *)webView
shouldStartLoadWithRequest:(NSURLRequest *)request
 navigationType:(UIWebViewNavigationType)navigationType;

- (void)webView:(UIWebView *)webView
didFailLoadWithError:(NSError *)error;

- (void)webViewDidFinishLoad:(UIWebView *)webView;

- (void)webViewDidStartLoad:(UIWebView *)webView;
```

Examples

Создание экземпляра UIWebView

стриж

```
let webview = UIWebView(frame: CGRect(x: 0, y: 0, width: 320, height: 480))
```

Objective-C

```
UIWebView *webview = [[UIWebView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];

//Alternative way of defining frame for UIWebView
UIWebView *webview = [[UIWebView alloc] init];
CGRect webviewFrame = webview.frame;
webviewFrame.size.width = 320;
webviewFrame.size.height = 480;
webviewFrame.origin.x = 0;
webviewFrame.origin.y = 0;
webview.frame = webviewFrame;
```

Создание запроса URL

Загружать контент в веб-просмотр из `url` адреса

стриж

```
webview.loadRequest(NSURLRequest(URL: NSURL(string: "http://www.google.com")!))
```

Objective-C

```
[webview loadRequest:[NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://www.google.com"]]];
```

Остановка загрузки веб-содержимого

Метод `stopLoading()` останавливает текущий процесс загрузки webview.

стриж

```
webview.stopLoading()
```

Objective-C

```
[webview stopLoading];
```

Обновить текущий веб-контент

стриж

```
webview.reload()
```

Objective-C

```
[webview reload];
```

Определение размера содержимого

Во многих случаях, например, при использовании веб-представлений в ячейках таблицы, важно определить размер содержимого отображаемой HTML-страницы. После загрузки страницы это можно вычислить в методе делегата `UIWebViewDelegate` :

```
- (void) webViewDidFinishLoad:(UIWebView *) aWebView {
    CGRect frame = aWebView.frame;
    frame.size.height = 1;
    aWebView.frame = frame;
    CGSize fittingSize = [aWebView sizeThatFits:CGSizeZero];
    frame.size = fittingSize;
    aWebView.frame = frame;

    NSLog(@"size: %f, %f", fittingSize.width, fittingSize.height);
}
```

В коде используется дополнительный трюк, позволяющий коротко установить высоту веб-представления на 1 до измерения размера фитинга. В противном случае он просто сообщит о текущем размере кадра. После измерения мы сразу установили высоту на

фактическую высоту содержимого.

Источник

Загрузить HTML-строку

Веб-представления полезны для загрузки локально сгенерированных строк HTML.

```
NSString *html = @"<!DOCTYPE html><html><body>Hello World</body></html>";
[webView loadHTMLString:html baseURL:nil];
```

стриж

```
  let htmlString = "<h1>My First Heading</h1><p>My first paragraph.</p>"
  webView.loadHTMLString(htmlString, baseURL: nil)
```

Может быть указан локальный базовый URL. Это полезно для ссылки на изображения, таблицы стилей или сценарии из набора приложений:

```
NSString *html = @"<!DOCTYPE html><html><head><link href='style.css' rel='stylesheet'
type='text/css'></head><body>Hello World</body></html>";
[self loadHTMLString:html baseURL:[NSURL fileURLWithPath:[[NSBundle mainBundle]
resourcePath]]];
```

В этом случае `style.css` загружается локально из каталога ресурсов приложения. Конечно, можно указать удаленный URL.

Загрузить JavaScript

Мы можем запускать пользовательский JavaScript в `UIWebView` используя метод `stringByEvaluatingJavaScriptFromString()` Этот метод возвращает результат запуска скрипта JavaScript, переданного в параметре сценария, или nil, если сбой сценария.

стриж

Загрузить скрипт из String

```
webview.stringByEvaluatingJavaScriptFromString("alert('This is JavaScript!');")
```

Загрузить сценарий из локального файла

```
//Suppose you have javascript file named "JavaScript.js" in project.
let filePath = NSBundle.mainBundle().pathForResource("JavaScript", ofType: "js")
        do {
            let jsContent = try String.init(contentsOfFile: filePath!, encoding:
NSUTF8StringEncoding)
            webview.stringByEvaluatingJavaScriptFromString(jsContent)
        }
        catch let error as NSError{
```

```
            print(error.debugDescription)
        }
```

Objective-C

Загрузить скрипт из String

```
[webview stringByEvaluatingJavaScriptFromString:@"alert('This is JavaScript!');"];
```

Загрузить сценарий из локального файла

```
//Suppose you have javascript file named "JavaScript.js" in project.
NSString *filePath = [[NSBundle mainBundle] pathForResource:@"JavaScript" ofType:@"js"];
NSString *jsContent = [NSString stringWithContentsOfFile:filePath
encoding:NSUTF8StringEncoding error:nil];
[webview stringByEvaluatingJavaScriptFromString:jsContent];
```

Примечание. Метод `stringByEvaluatingJavaScriptFromString:` ждет синхронного завершения для оценки JavaScript. Если вы загружаете веб-контент, код JavaScript которого вы не проверили, вызов этого метода может повесить ваше приложение. Лучшая практика заключается в принятии `WKWebView` класса и использовать его `evaluateJavaScript:completionHandler:` метод. Но `WKWebView` доступен от iOS 8.0 и более поздних `WKWebView` .

Загрузите файлы документов, такие как .pdf, .txt, .doc и т. Д.

Вместо веб-страниц мы также можем загружать файлы документов в iOS WebView, например .pdf, .txt, .doc и т. `loadData` Метод `loadData` используется для загрузки `NSData` в webview.

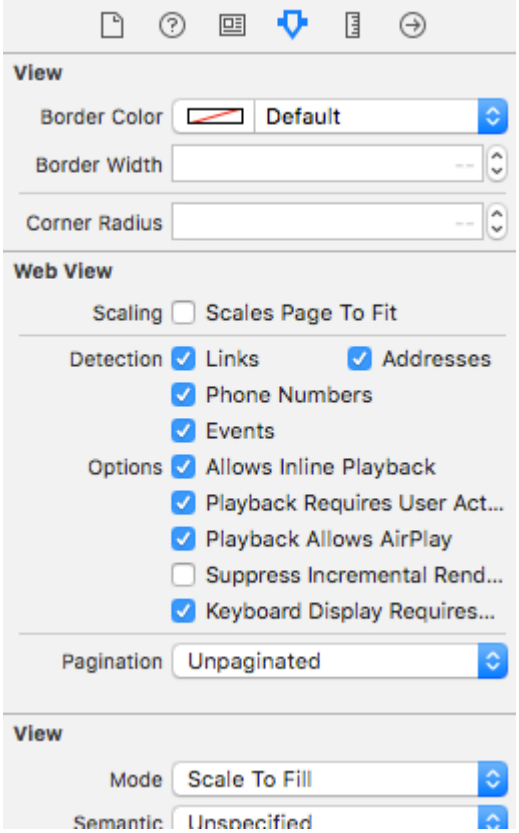
стриж

```
//Assuming there is a text file in the project named "home.txt".
let localFilePath = NSBundle.mainBundle().pathForResource("home", ofType:"txt");
let data = NSFileManager.defaultManager().contentsAtPath(localFilePath!);
webview.loadData(data!, MIMEType: "application/txt", textEncodingName:"UTF-8" , baseURL:
NSURL())
```

Objective-C

```
//Assuming there is a text file in the project named "home.txt".
NSString *localFilePath = [[NSBundle mainBundle] pathForResource:@"home" ofType:@"txt"];
NSData *data = [[NSFileManager defaultManager] contentsAtPath:localFilePath];
[webview loadData:data MIMEType:@"application/txt" textEncodingName:@"UTF-8" baseURL:[NSURL
new]];
```

Сделать ссылки Что внутри UIWebview clickable



В vc.h

```
@interface vc : UIViewController<UIWebViewDelegate>
```

в vc.m

```
- (BOOL)webView:(UIWebView *)webView shouldStartLoadWithRequest:(NSURLRequest *)request
navigationType:(UIWebViewNavigationType)navigationType{


    if (navigationType == UIWebViewNavigationTypeLinkClicked){
        //open it on browser if you want to open it in same web view remove return NO;
        NSURL *url = request.URL;
        if ([[UIApplication sharedApplication] canOpenURL:url]) {
            [[UIApplication sharedApplication] openURL:url];
        }
        return NO;

    }

    return YES;

}
```

Загружать локальный HTML-файл в webView

Во-первых, добавьте файл HTML в свой проект (если вас попросят выбрать параметры для добавления файла, выберите « *Копировать элементы», если это необходимо*)

Следующая строка кода загружает содержимое HTML-файла в webView

```
webView.loadRequest(NSURLRequest(URL: NSURL(fileURLWithPath:
NSBundle.mainBundle().pathForResource("YOUR HTML FILE", ofType: "html")!})
```

- Если ваш HTML-файл называется index.html, замените **ВАШ HTML-ФАЙЛ индексом**
- Вы можете использовать этот код либо в *viewDidLoad (),* либо в *viewDidAppear ()* или любой другой функции

Прочитайте UIWebView онлайн: https://riptutorial.com/ru/ios/topic/1452/uiwebview

глава 110: UUID (универсальный уникальный идентификатор)

замечания

Чтобы сохранить UUID, мы можем использовать SSKeychainUtility . Пример можно найти на странице Github

Examples

Создание UUID

Случайный UUID

стриж

```
func randomUUID() -> NSString{
    return NSUUID.UUID().UUIDString()
}
```

Objective-C

```
+ (NSString *)randomUUID {
    if(NSClassFromString(@"NSUUID")) { // only available in iOS >= 6.0
        return [[NSUUID UUID] UUIDString];
    }
    CFUUIDRef uuidRef = CFUUIDCreate(kCFAllocatorDefault);
    CFStringRef cfuuid = CFUUIDCreateString(kCFAllocatorDefault, uuidRef);
    CFRelease(uuidRef);
    NSString *uuid = [((__bridge NSString *) cfuuid) copy];
    CFRelease(cfuuid);
    return uuid;
}
```

Идентификатор поставщика

iOS 6

В одной строке мы можем получить UUID, как показано ниже:

стриж

```
let UDIDString = UIDevice.currentDevice().identifierForVendor?.UUIDString
```

Objective-C

```
NSString *UDIDString = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
```

`identifierForVendor` является уникальным идентификатором, который остается неизменным для каждого приложения одного поставщика на одном устройстве, если только все приложения поставщика не удаляются с этого устройства. См . Документацию Apple о том, когда этот `UUID` изменяется.

Apple IFA против IFV (Apple Identifier для рекламодателей против идентификатора для поставщиков)

- Вы можете использовать IFA для измерения кликов по объявлениям и IFV для измерения установок приложений.
- IFA имеет встроенные механизмы конфиденциальности, которые делают его идеальным для рекламы. Напротив, IFV предназначен разработчикам для внутреннего использования для измерения пользователей, которые устанавливают свои приложения.

ЕСЛИ

- Класс ASIdentifierManager обеспечивает
    - **advertisingIdentifier: UUID** : алфавитно-цифровая строка, уникальная для каждого устройства, используемая только для рекламы рекламы.
    - **isAdvertisingTrackingEnabled** : Логическое значение, указывающее, имеет ли пользователь ограниченное отслеживание объявлений.

IFV

- Класс ASIdentifierManager обеспечивает
    - **identifierForVendor: UUID** : буквенно-цифровая строка, которая однозначно идентифицирует устройство у поставщика приложения.

Найти устройство MPC и IFV здесь .

Создать строку UUID для устройств iOS

Здесь мы можем создать `UUID String` в одной строке.

Представляет строки UUID, которые могут использоваться для уникальной идентификации типов, интерфейсов и других элементов.

Swift 3.0

```
print(UUID().uuidString)
```

Это очень полезно для идентификации нескольких устройств с уникальным идентификатором.

Прочитайте UUID (универсальный уникальный идентификатор) онлайн: https://riptutorial.com/ru/ios/topic/3629/uuid--универсальный-уникальный-идентификатор-

глава 111: WCSessionDelegate

Вступление

`WCSessionDelegate` работает с часами OS2 +, используя WatchConnectivity. var watchSession: WCSession? func startWatchSession () {if (WCSession.isSupported ()) {watchSession = WCSession.default () watchSession !.delegate = self watchSession! .activate ()}} Внедрить требуемый метод: - `didReceiveApplicationContext`

Examples

Контроллер набора часов (WKInterfaceController)

```
import WatchConnectivity

var watchSession : WCSession?

    override func awake(withContext context: Any?) {
        super.awake(withContext: context)
        // Configure interface objects here.
        startWatchSession()
    }

func startWatchSession(){

        if(WCSession.isSupported()){
            watchSession = WCSession.default()
            watchSession!.delegate = self
            watchSession!.activate()
        }
    }

//Callback in below delegate method when iOS app triggers event
func session(_ session: WCSession, didReceiveApplicationContext applicationContext: [String :
Any]) {
        print("did ReceiveApplicationContext at watch")
    }
```

Прочитайте WCSessionDelegate онлайн:
https://riptutorial.com/ru/ios/topic/8289/wcsessiondelegate

глава 112: WKWebView

Вступление

WKWebView является центральным элементом современного API WebKit, представленного в iOS 8 и OS X Yosemite. Он заменяет UIWebView в UIKit и WebView в AppKit, предлагая согласованный API на двух платформах.

Имея отзывчивую прокрутку 60 кадров в секунду, встроенные жесты, оптимизированную связь между приложением и веб-страницей и тот же механизм JavaScript, что и Safari, WKWebView является одним из самых значительных объявлений, которые выходят из WWDC 2014.

Examples

Создание простого веб-браузера

```
import UIKit
import WebKit


class ViewController: UIViewController, UISearchBarDelegate, WKNavigationDelegate,
WKUIDelegate {

    var searchbar: UISearchBar! //All web-browsers have a search-bar.
    var webView: WKWebView! //The WKWebView we'll use.
    var toolbar: UIToolbar! //Toolbar at the bottom just like in Safari.
    var activityIndicator: UIActivityIndicatorView! //Activity indicator to let the user know
the page is loading.

    override func viewDidLoad() {
        super.viewDidLoad()

        self.initControls()
        self.setTheme()
        self.doLayout()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }


    func initControls() {
        self.searchbar = UISearchBar()

        //WKUserContentController allows us to add Javascript scripts to our webView that will
run either at the beginning of a page load OR at the end of a page load.

        let configuration = WKWebViewConfiguration()
        let contentController = WKUserContentController()
        configuration.userContentController = contentController
```

```
        //create the webView with the custom configuration.
        self.webView = WKWebView(frame: .zero, configuration: configuration)

        self.toolbar = UIToolbar()
        self.layoutToolbar()

        self.activityIndicator = UIActivityIndicatorView(activityIndicatorStyle: .gray)
        self.activityIndicator.hidesWhenStopped = true
    }

    func setTheme() {
        self.edgesForExtendedLayout = UIRectEdge(rawValue: 0)
        self.navigationController?.navigationBar.barTintColor = UIColor.white()

        //Theme the keyboard and searchBar. Setup delegates.
        self.searchbar.delegate = self
        self.searchbar.returnKeyType = .go
        self.searchbar.searchBarStyle = .prominent
        self.searchbar.placeholder = "Search or enter website name"
        self.searchbar.autocapitalizationType = .none
        self.searchbar.autocorrectionType = .no

        //Set the WebView's delegate.
        self.webView.navigationDelegate = self //Delegate that handles page navigation
        self.webView.uiDelegate = self //Delegate that handles new tabs, windows, popups,
layout, etc..

        self.activityIndicator.transform = CGAffineTransform(scaleX: 1.5, y: 1.5)
    }

    func layoutToolbar() {
        //Browsers typically have a back button, forward button, refresh button, and
newTab/newWindow button.

        var items = Array<UIBarButtonItem>()

        let space = UIBarButtonItem(barButtonSystemItem: .flexibleSpace, target: nil, action:
nil)

        items.append(UIBarButtonItem(title: "<", style: .plain, target: self, action:
#selector(onBackButtonPressed)))
        items.append(space)
        items.append(UIBarButtonItem(title: ">", style: .plain, target: self, action:
#selector(onForwardButtonPressed)))
        items.append(space)
        items.append(UIBarButtonItem(barButtonSystemItem: .refresh, target: self, action:
#selector(onRefreshPressed)))
        items.append(space)
        items.append(UIBarButtonItem(barButtonSystemItem: .organize, target: self, action:
#selector(onTabPressed)))

        self.toolbar.items = items
    }

    func doLayout() {
        //Add the searchBar to the navigationBar.
        self.navigationItem.titleView = self.searchbar

        //Add all other subViews to self.view.
        self.view.addSubview(self.webView)
```

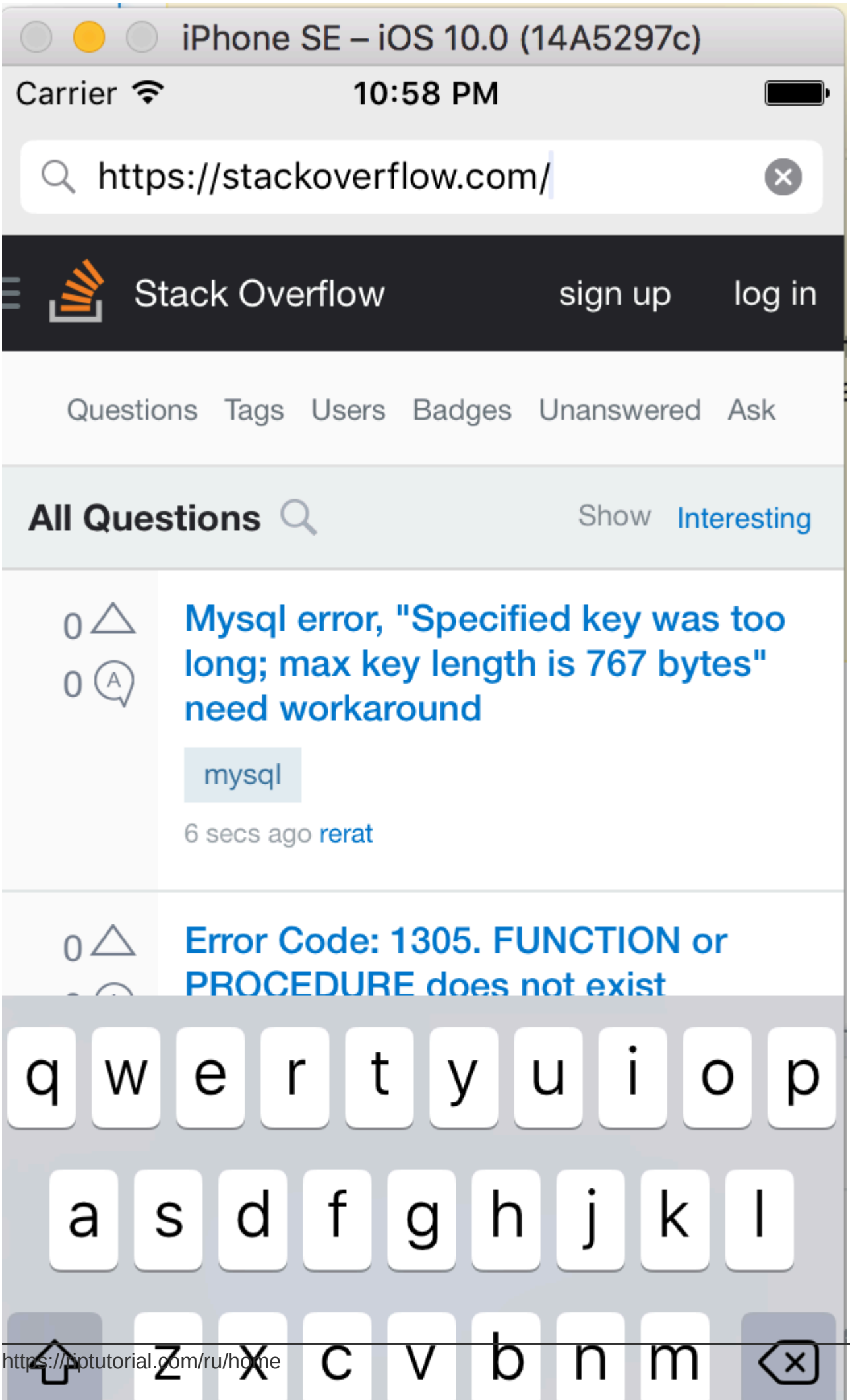
```
        self.view.addSubview(self.toolbar)
        self.view.addSubview(self.activityIndicator)

        //Setup which views will be constrained.

        let views: [String: AnyObject] = ["webView": self.webView, "toolbar": self.toolbar,
"activityIndicator": self.activityIndicator];
        var constraints = Array<String>();

        constraints.append("H:|-0-[webView]-0-|")
        constraints.append("H:|-0-[toolbar]-0-|")
        constraints.append("V:|-0-[webView]-0-[toolbar(50)]-0-|")


        //constrain the subviews using the above visual constraints.

        for constraint in constraints {
            self.view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutFormatOptions(rawValue: 0), metrics: nil, views: views))
        }

        for view in self.view.subviews {
            view.translatesAutoresizingMaskIntoConstraints = false
        }


        //constraint the activity indicator to the center of the view.
        self.view.addConstraint(NSLayoutConstraint(item: self.activityIndicator, attribute:
.centerX, relatedBy: .equal, toItem: self.view, attribute: .centerX, multiplier: 1.0,
constant: 0.0))
        self.view.addConstraint(NSLayoutConstraint(item: self.activityIndicator, attribute:
.centerY, relatedBy: .equal, toItem: self.view, attribute: .centerY, multiplier: 1.0,
constant: 0.0))
    }

    //Searchbar Delegates

    func searchBarSearchButtonClicked(_ searchBar: UISearchBar) {
        self.searchbar.resignFirstResponder()

        if let searchText = self.searchbar.text, url = URL(string: searchText) {
            //Get the URL from the search bar. Create a new NSURLRequest with it and tell the
webView to navigate to that URL/Page. Also specify a timeout for if the page takes too long.
Also handles cookie/caching policy.

            let request = URLRequest(url: url, cachePolicy: .useProtocolCachePolicy,
timeoutInterval: 30)
            self.webView.load(request)
        }
    }


    //Toolbar Delegates

    func onBackButtonPressed(button: UIBarButtonItem) {
        if (self.webView.canGoBack) { //allow the user to go back to the previous page.
            self.webView.goBack()
        }
    }

    func onForwardButtonPressed(button: UIBarButtonItem) {
```

```
        if (self.webView.canGoForward) { //allow the user to go forward to the next page.
            self.webView.goForward()
        }
    }

    func onRefreshPressed(button: UIBarButtonItem) {
        self.webView.reload()  //reload the current page.
    }

    func onTabPressed(button: UIBarButtonItem) {
        //TODO: Open a new tab or web-page.
    }


    //WebView Delegates

    func webView(_ webView: WKWebView, decidePolicyFor navigationAction: WKNavigationAction,
decisionHandler: (WKNavigationActionPolicy) -> Void) {

        decisionHandler(.allow) //allow the user to navigate to the requested page.
    }

    func webView(_ webView: WKWebView, decidePolicyFor navigationResponse:
WKNavigationResponse, decisionHandler: (WKNavigationResponsePolicy) -> Void) {

        decisionHandler(.allow) //allow the webView to process the response.
    }

    func webView(_ webView: WKWebView, didStartProvisionalNavigation navigation:
WKNavigation!) {
        self.activityIndicator.startAnimating()
    }

    func webView(_ webView: WKWebView, didFailProvisionalNavigation navigation: WKNavigation!,
withError error: NSError) {
        self.activityIndicator.stopAnimating()

        //Handle the error. Display an alert to the user telling them what happened.

        let alert = UIAlertController(title: "Error", message: error.localizedDescription,
preferredStyle: .alert)
        let action = UIAlertAction(title: "OK", style: .default) { (action) in
            alert.dismiss(animated: true, completion: nil)
        }
        alert.addAction(action)
        self.present(alert, animated: true, completion: nil)
    }

    func webView(_ webView: WKWebView, didFinish navigation: WKNavigation!) {
        self.activityIndicator.stopAnimating()

        //Update our search bar with the webPage's final endpoint-URL.
        if let url = self.webView.url {
            self.searchbar.text = url.absoluteString ?? self.searchbar.text
        }
    }

    func webView(_ webView: WKWebView, didReceiveServerRedirectForProvisionalNavigation
navigation: WKNavigation!) {
        //When the webview receives a "Redirect" to a different page or endpoint, this is
called.
```

```
    }

    func webView(_ webView: WKWebView, didCommit navigation: WKNavigation!) {
        //When the content for the webpage starts arriving, this is called.
    }

    func webView(_ webView: WKWebView, didFail navigation: WKNavigation!, withError error:
NSError) {

    }

    func webView(_ webView: WKWebView, didReceive challenge: URLAuthenticationChallenge,
completionHandler: (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {

        completionHandler(.performDefaultHandling, .none) //Handle SSL connections by default.
We aren't doing SSL pinning or custom certificate handling.

    }


    //WebView's UINavigation Delegates

    //This is called when a webView or existing loaded page wants to open a new window/tab.
    func webView(_ webView: WKWebView, createWebViewWith configuration:
WKWebViewConfiguration, for navigationAction: WKNavigationAction, windowFeatures:
WKWindowFeatures) -> WKWebView? {

        //The view that represents the new tab/window. This view will have an X button at the
top left corner + a webView.
        let container = UIView()

        //New tabs need an exit button.
        let XButton = UIButton()
        XButton.addTarget(self, action: #selector(onWebViewExit), for: .touchUpInside)
        XButton.layer.cornerRadius = 22.0

        //Create the new webView window.
        let webView = WKWebView(frame: .zero, configuration: configuration)
        webView.navigationDelegate = self
        webView.uiDelegate = self

        //Layout the tab.
        container.addSubview(XButton)
        container.addSubview(webView)

        let views: [String: AnyObject] = ["XButton": XButton, "webView": webView];
        var constraints = Array<String>()

        constraints.append("H:|-(-22)-[XButton(44)]")
        constraints.append("H:|-0-[webView]-0-|")
        constraints.append("V:|-(-22)-[XButton(44)]-0-[webView]-0-|")


        //constrain the subviews.
        for constraint in constraints {
            container.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutFormatOptions(rawValue: 0), metrics: nil, views: views))
        }

        for view in container.subviews {
            view.translatesAutoresizingMaskIntoConstraints = false
```

```
        }

        //TODO: Add the containerView to self.view or present it with a new controller. Keep
track of tabs..

        return webView
    }

    func onWebViewExit(button: UIButton) {
        //TODO: Destroy the tab. Remove the new tab from the current window or controller.
    }
}
```

Отображение пользовательской кнопки `GO` на клавиатуре:


iPhone SE – iOS 10.0 (14A5297c)
Carrier
10:58 PM
https://stackoverflow.com/
Stack Overflow
sign up
log in
Questions
Tags
Users
Badges
Unanswered
Ask
All Questions
Show
Interesting
0
0
Mysql error, "Specified key was too long; max key length is 767 bytes" need workaround
mysql
6 secs ago rerat
0
Error Code: 1305. FUNCTION or PROCEDURE does not exist
q w e r t y u i o p
a s d f g h j k l
z x c v b n m

`.AtDocumentEnd`

Отправлять сообщения с JavaScript и обрабатывать их с внутренней стороны

Сообщения могут быть отправлены из JavaScript с использованием следующего кода

```
window.webkit.messageHandlers.{NAME}.postMessage()
```

Здесь, как создать обработчик сообщения сценария для обработки сообщений:

```
class NotificationScriptMessageHandler: NSObject, WKScriptMessageHandler {
    func userContentController(userContentController: WKUserContentController,
didReceiveScriptMessage message: WKScriptMessage!) {
        if message.name == "{NAME}" {
            // to be sure of handling the correct message
            print(message.body)
        }
    }
}
```

Здесь, как настроить обработчик сообщения сценария в WKWebView:

```
let configuration = WKWebViewConfiguration()
let userContentController = WKUserContentController()
let handler = NotificationScriptMessageHandler()
userContentController.addScriptMessageHandler(handler, name: "{NAME}")
configuration.userContentController = userContentController
let webView = WKWebView(frame: self.view.bounds, configuration: configuration)
```

> **ПРИМЕЧАНИЕ.** Добавление такого же обработчика `"{NAME}"` с `addScriptMessageHandler:name:` более одного раза приводит к исключению `NSInvalidArgumentException` .

Прочитайте WKWebView онлайн: https://riptutorial.com/ru/ios/topic/3602/wkwebview

глава 113: Xcode Build & Archive из командной строки

Синтаксис

- `xcodebuild [-project name.xcodeproj] -scheme schemename [[-destination destinationspecifier] ...] [-destination-timeout value] [-configuration configurationname] [-sdk [sdkfullpath | sdkname]] [action ...] [buildsetting=value ...] [-userdefault=value ...]`

параметры

вариант	Описание
-проект	Создайте проект name.xcodeproj.
-схема	Требуется при создании рабочего пространства.
-место назначения	Использовать целевое устройство
-конфигурация	Использовать конфигурацию сборки
-sdk	указанный SDK

замечания

Запустите `xcodebuild` из каталога, содержащего ваш проект, для создания проекта Xcode. Чтобы создать рабочую область Xcode, вы должны передать параметры **-workspace** и **-схемы** для определения сборки. Параметры схемы будут контролировать, какие цели создаются и как они построены, хотя вы можете передать другие параметры xcodebuild, чтобы переопределить некоторые параметры схемы.

Examples

Строительство и архив

Телосложение:

```
xcodebuild -exportArchive -exportFormat ipa \
    -archivePath "/Users/username/Desktop/MyiOSApp.xcarchive" \
    -exportPath "/Users/username/Desktop/MyiOSApp.ipa" \
    -exportProvisioningProfile "MyCompany Distribution Profile"
```

Архив:

```
xcodebuild -project <ProjectName.xcodeproj>
    -scheme <ProjectName>
    -sdk iphonesimulator
    -configuration Debug
    -destination "platform=iOS Simulator,name=<Device>,OS=9.3"
    clean build
```

Прочитайте Xcode Build & Archive из командной строки онлайн:
https://riptutorial.com/ru/ios/topic/5027/xcode-build--amp--archive-из-командной-строки

глава 114: Автоматическая компоновка

Вступление

Автоматическая компоновка динамически вычисляет размер и положение всех представлений в вашей иерархии представлений на основе ограничений, размещенных на этих представлениях. Источник

Синтаксис

- NSLayoutConstraint (item: Any, attribute: NSLayoutAttribute, relatedBy: NSLayoutRelation, toItem: Any ?, attribute: NSLayoutAttribute, множитель: CGFloat, константа: CGFloat) // Создавать противопоказание программно

Examples

Настройка ограничений программно

Пример кода котельной

```
override func viewDidLoad() {
    super.viewDidLoad()

    let myView = UIView()
    myView.backgroundColor = UIColor.blueColor()
    myView.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(myView)

    // Add constraints code here
    // ...
}
```

В приведенных ниже примерах стиль Anchor является предпочтительным методом над `NSLayoutConstraint` , однако он доступен только для iOS 9, поэтому, если вы поддерживаете iOS 8, вам все равно следует использовать стиль `NSLayoutConstraint` .

Закрепление

Анкерный стиль

```
let margins = view.layoutMarginsGuide
myView.leadingAnchor.constraintEqualToAnchor(margins.leadingAnchor, constant: 20).active =
true
```

- В дополнение к `leadingAnchor` , есть также `trailingAnchor` , `topAnchor` и `bottomAnchor` .

Стиль NSLayoutConstraint

```
NSLayoutConstraint(item: myView, attribute: NSLayoutAttribute.Leading, relatedBy:
NSLayoutRelation.Equal, toItem: view, attribute: NSLayoutAttribute.LeadingMargin, multiplier:
1.0, constant: 20.0).active = true
```

- В дополнение к `.Leading` также есть `.Trailing` , `.Top` и `.Bottom` .
- В дополнение к `.LeadingMargin` также есть `.TrailingMargin` , `.TopMargin` и `.BottomMargin` .

Стиль визуального форматирования

```
NSLayoutConstraint.constraintsWithVisualFormat("H:|-20-[myViewKey]", options: [], metrics:
nil, views: ["myViewKey": myView])
```

Ширина и высота

Анкерный стиль

```
myView.widthAnchor.constraintEqualToAnchor(nil, constant: 200).active = true
myView.heightAnchor.constraintEqualToAnchor(nil, constant: 100).active = true
```

Стиль NSLayoutConstraint

```
NSLayoutConstraint(item: myView, attribute: NSLayoutAttribute.Width, relatedBy:
NSLayoutRelation.Equal, toItem: nil, attribute: NSLayoutAttribute.NotAnAttribute, multiplier:
1, constant: 200).active = true
NSLayoutConstraint(item: myView, attribute: NSLayoutAttribute.Height, relatedBy:
NSLayoutRelation.Equal, toItem: nil, attribute: NSLayoutAttribute.NotAnAttribute, multiplier:
1, constant: 100).active = true
```

Стиль визуального форматирования

```
NSLayoutConstraint.constraintsWithVisualFormat("H:[myViewKey(200)]", options: [], metrics:
nil, views: ["myViewKey": myView])
NSLayoutConstraint.constraintsWithVisualFormat("V:[myViewKey(100)]", options: [], metrics:
nil, views: ["myViewKey": myView])
```

Центр в контейнере

Анкерный стиль

```
myView.centerXAnchor.constraintEqualToAnchor(view.centerXAnchor).active = true
myView.centerYAnchor.constraintEqualToAnchor(view.centerYAnchor).active = true
```

Стиль NSLayoutConstraint

```
NSLayoutConstraint(item: myView, attribute: NSLayoutAttribute.CenterX, relatedBy:
NSLayoutRelation.Equal, toItem: view, attribute: NSLayoutAttribute.CenterX, multiplier: 1,
constant: 0).active = true
NSLayoutConstraint(item: myView, attribute: NSLayoutAttribute.CenterY, relatedBy:
NSLayoutRelation.Equal, toItem: view, attribute: NSLayoutAttribute.CenterY, multiplier: 1,
constant: 0).active = true
```

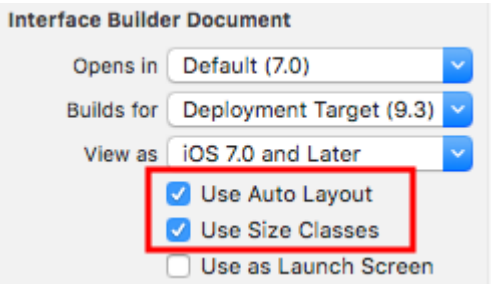
Стиль визуального форматирования

```
NSLayoutConstraint.constraintsWithVisualFormat("V:[viewKey]-(<=0)-[myViewKey]", options:
NSLayoutFormatOptions.AlignAllCenterX, metrics: nil, views: ["myViewKey": myView, "viewKey":
view])
NSLayoutConstraint.constraintsWithVisualFormat("H:[viewKey]-(<=0)-[myViewKey]", options:
NSLayoutFormatOptions.AlignAllCenterY, metrics: nil, views: ["myViewKey": myView, "viewKey":
view])
```

Как использовать автоматический макет

Автоматическая компоновка используется для организации представлений, чтобы они выглядели хорошо на любом устройстве и ориентации. Ограничения - это правила, которые рассказывают, как все должно быть заложено. Среди прочего они включают закрепление краев, центрирование и установочные размеры.

Автоматический макет включен по умолчанию, но вы можете дважды проверить это. Если вы *нажмете Main.storyboard* в Навигаторе проектов, а затем покажете Инспектор файлов. Убедитесь, что отмечены флажки Auto Layout и Size Classes:

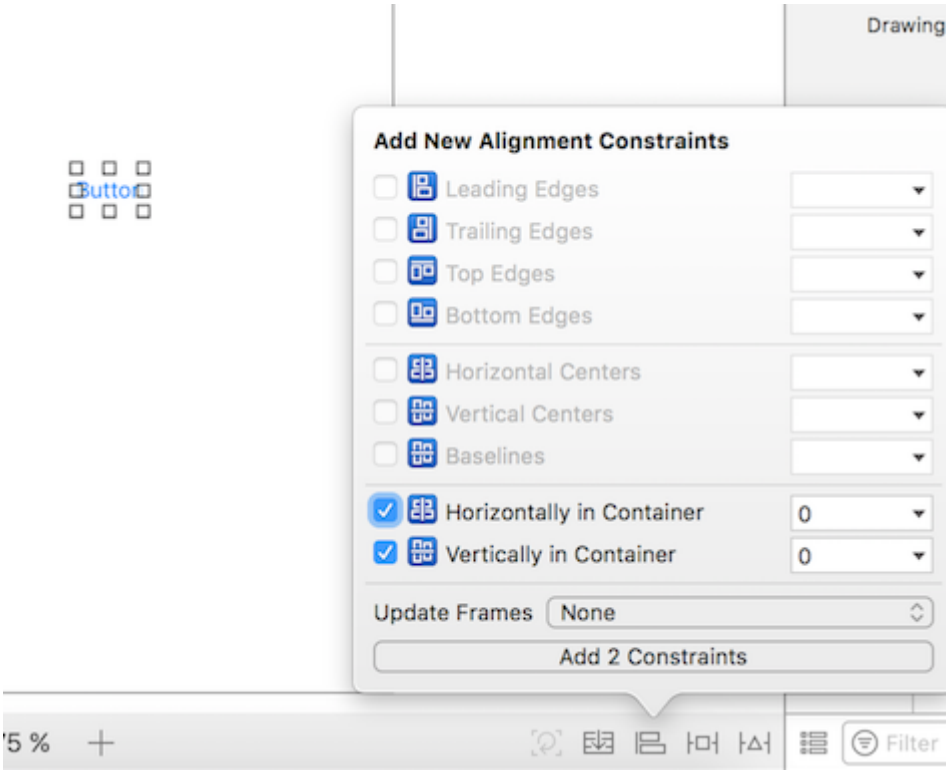


Ограничения автоматической компоновки можно задать в построителе интерфейса или в коде. В интерфейсе Builder вы найдете инструменты автоматического макета в правом нижнем углу. Нажав на них, вы увидите различные варианты установки ограничений для представления.

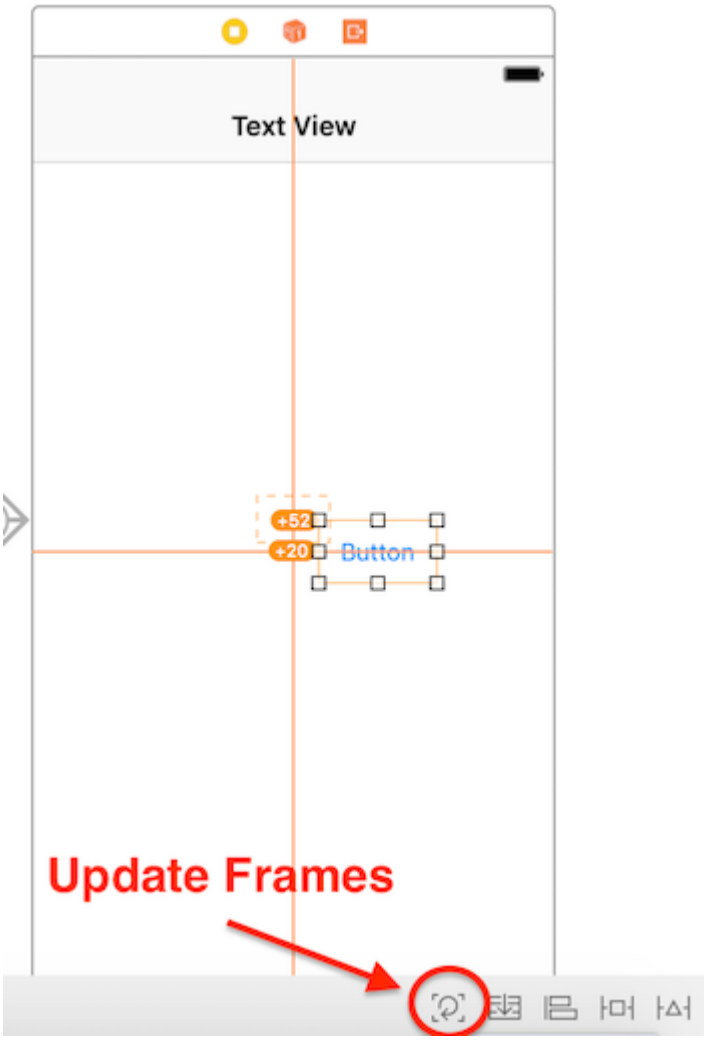
Если вы хотите иметь разные ограничения для разных размеров или ориентации устройства, вы можете установить их в WAny hayny Size Class, которые находятся в нижней части.

Центровые ограничения

Выберите свою кнопку (или любой другой вид, который хотите центрировать) на **раскадровке** . Затем нажмите кнопку выравнивания в правом нижнем углу. Выберите « `Horizontally in Container` и « `Vertically in Container` . Нажмите «Добавить 2 ограничения».



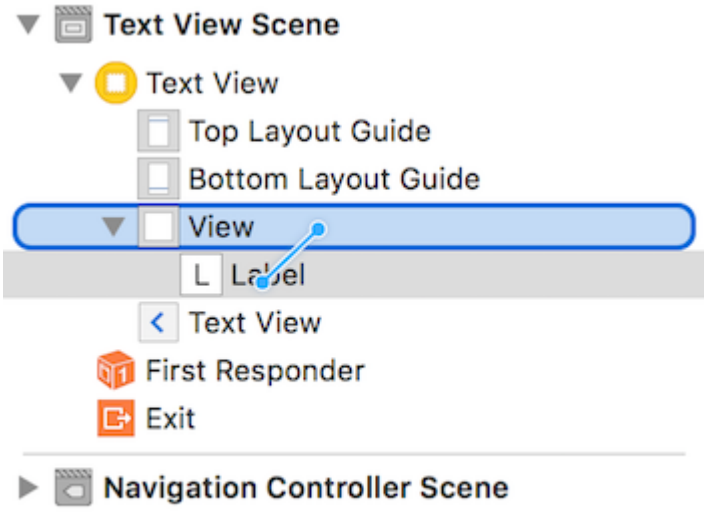
Если бы он не был полностью центрирован, вам, возможно, понадобится сделать еще одну вещь. Нажмите кнопку «Обновить фреймы», расположенную на два слева от кнопки «Вставить в стек» на нижней панели.



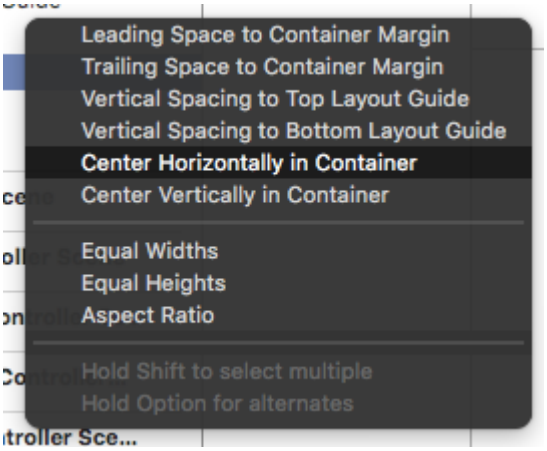
Вы также можете «обновлять фреймы по мере необходимости», нажимая вместе ⌘ + ⌥ + = (Command + Option и equals) после выбора представления, это может сэкономить некоторое время.

Теперь, когда вы запускаете приложение, оно должно быть центрировано независимо от того, какой размер устройства вы используете.

Другой способ просмотра точек с помощью Interface Builder - это перетаскивание с помощью элемента управления. Предположим, вы хотите `UILabel` в представлении. Откройте `Document Outline` в своем раскадровке, нажав кнопку боковой панели внизу слева. Нажмите и перетащите ярлык в представление, удерживая `ctrl` (control), и появится синяя линия:



После выпуска появится меню параметров ограничения:



Выберите «Центр по горизонтали в контейнере» и «Центрировать по вертикали в контейнере». Обновляйте кадры по мере необходимости и вуаля! Центральная этикетка.

В качестве альтернативы вы можете добавлять ограничения программно. Создайте ограничения и добавьте их в нужные элементы пользовательского интерфейса и представления, как описано в следующем примере, где мы создаем кнопку и выравниваем ее по центру, по горизонтали и по вертикали в ее супервизор:

Objective-C

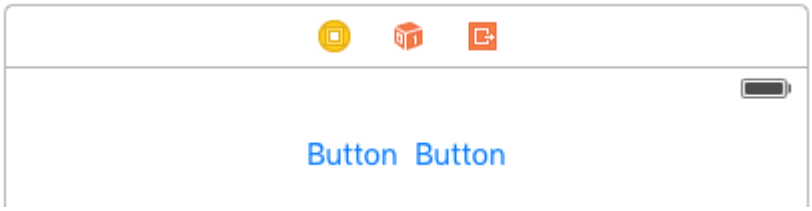
```
- (void)viewDidLoad
{
    [super viewDidLoad];
    UIButton *yourButton = [[UIButton alloc] initWithFrame:CGRectMake(0, 0, 100, 18)];
    [yourButton setTitle:@"Button" forState:UIControlStateNormal];


    [self.view addConstraint:[NSLayoutConstraint constraintWithItem:yourButton
attribute:NSLayoutAttributeCenterY relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeCenterY multiplier:1 constant:0]]; //Align veritcally center to
superView
```

```
    [self.view addConstraint:[NSLayoutConstraint constraintWithItem:yourButton
attribute:NSLayoutAttributeCenterX relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeCenterX multiplier:1 constant:0]]; //Align horizontally center to
superView

    [self.view addSubview:yourButton]; //Add button to superView
}
```

стриж

```
override func viewDidLoad()
{
    super.viewDidLoad()
    let yourButton: UIButton = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 18))
    yourButton.setTitle("Button", forState: .Normal)

    let centerVertically = NSLayoutConstraint(item: yourButton,
                                    attribute: .CenterX,
                                    relatedBy: .Equal,
                                       toItem: view,
                                    attribute: .CenterX,
                                   multiplier: 1.0,
                                     constant: 0.0)
    let centerHorizontally = NSLayoutConstraint(item: yourButton,
                                    attribute: .CenterY,
                                    relatedBy: .Equal,
                                       toItem: view,
                                    attribute: .CenterY,
                                   multiplier: 1.0,
                                     constant: 0.0)
    NSLayoutConstraint.activateConstraints([centerVertically, centerHorizontally])
}
```

Пространственные виды



Обычно требуется, чтобы два вида были бок о бок, в центре их наблюдения. Общий ответ, заданный в Stack Overflow, заключается в том, чтобы вставлять эти два представления в `UIView` и `UIView` . Это необязательно или рекомендуется. Из документов UILayoutGuide :

> Существует ряд затрат, связанных с добавлением фиктивных представлений в вашу иерархию представлений. Во-первых, есть затраты на создание и поддержание самого представления. Во-вторых, фиктивный вид является полноправным членом иерархии представлений, что означает, что он добавляет служебные данные для каждой задачи, выполняемой иерархией. Хуже всего, невидимый фиктивный вид может перехватывать сообщения, предназначенные для других представлений, что вызывает проблемы, которые очень трудно

найти.

Вы можете использовать `UILayoutGuide` для этого, вместо добавления кнопок в ненужный `UIView` . `UILayoutGuide` - это, по сути, прямоугольное пространство, которое может взаимодействовать с Auto Layout. Вы устанавливаете `UILayoutGuide` с левой и правой сторон кнопок и устанавливаете их ширину равными. Это будет центрировать кнопки. Вот как это сделать в коде:

Стиль визуального форматирования

```
view.addSubview(button1)
view.addSubview(button2)

let leftSpace = UILayoutGuide()
view.addLayoutGuide(leftSpace)

let rightSpace = UILayoutGuide()
view.addLayoutGuide(rightSpace)

let views = [
    "leftSpace" : leftSpace,
    "button1" : button1,
    "button2" : button2,
    "rightSpace" : rightSpace
]

// Lay the buttons and layout guides out horizontally in a line.
// Put the layout guides on each end.
NSLayoutConstraint.activateConstraints(NSLayoutConstraint.constraintsWithVisualFormat("H:|[leftSpace][b
[button2][rightSpace]|", options: [], metrics: nil, views: views))

// Now set the layout guides widths equal, so that the space on the
// left and the right of the buttons will be equal
leftSpace.widthAnchor.constraintEqualToAnchor(rightSpace.widthAnchor).active = true
```

Анкерный стиль

```
let leadingSpace = UILayoutGuide()
let trailingSpace = UILayoutGuide()
view.addLayoutGuide(leadingSpace)
view.addLayoutGuide(trailingSpace)

leadingSpace.widthAnchor.constraintEqualToAnchor(trailingSpace.widthAnchor).active = true

leadingSpace.leadingAnchor.constraintEqualToAnchor(view.leadingAnchor).active = true
leadingSpace.trailingAnchor.constraintEqualToAnchor(button1.leadingAnchor).active = true

trailingSpace.leadingAnchor.constraintEqualToAnchor(button2.trailingAnchor).active = true
trailingSpace.trailingAnchor.constraintEqualToAnchor(view.trailingAnchor).active = true
```

Вам также нужно будет добавить к этому вертикальные ограничения, но это будет центрировать кнопки в представлении без добавления каких-либо «фиктивных» представлений! Это избавит систему от потери времени процессора при отображении этих «фиктивных» представлений. В этом примере используются кнопки, но вы можете

поменять кнопки для любого вида, на которое вы хотите установить ограничения.

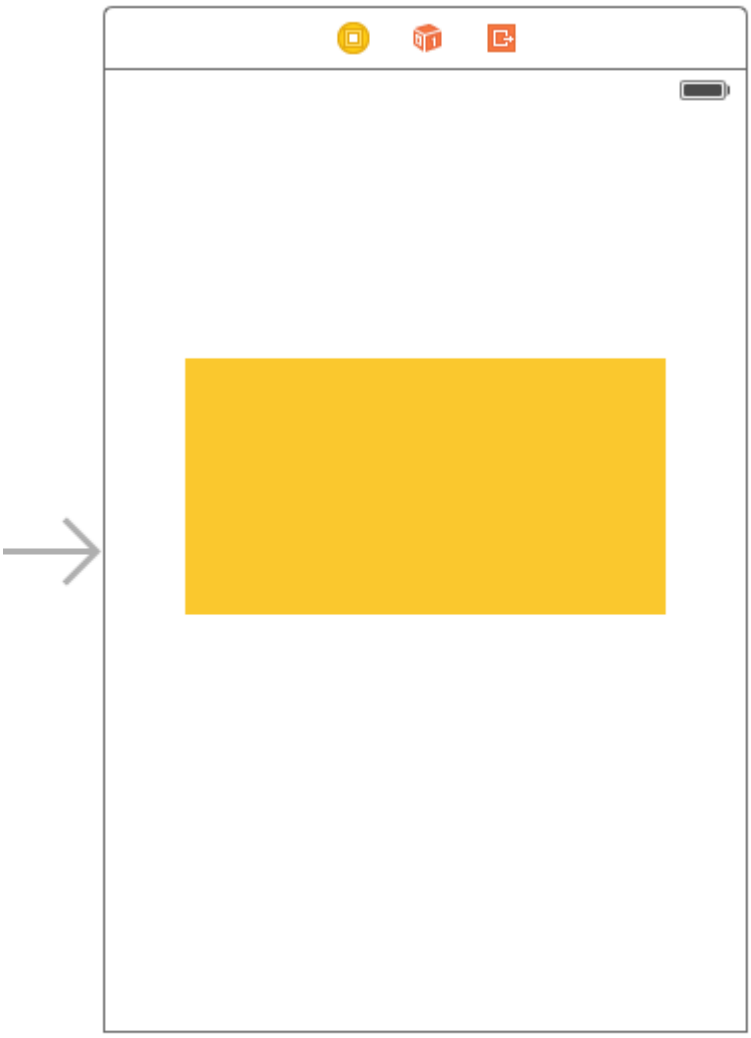
Если вы поддерживаете iOS 8 или ранее, самым простым способом создания этого макета является добавление скрытых фиктивных представлений. С помощью iOS 9 вы можете заменить фиктивные виды макетами.

Примечание. Интерфейс Builder еще не поддерживает макеты (Xcode 7.2.1). Поэтому, если вы хотите их использовать, вы должны создать свои ограничения в коде. Источник .

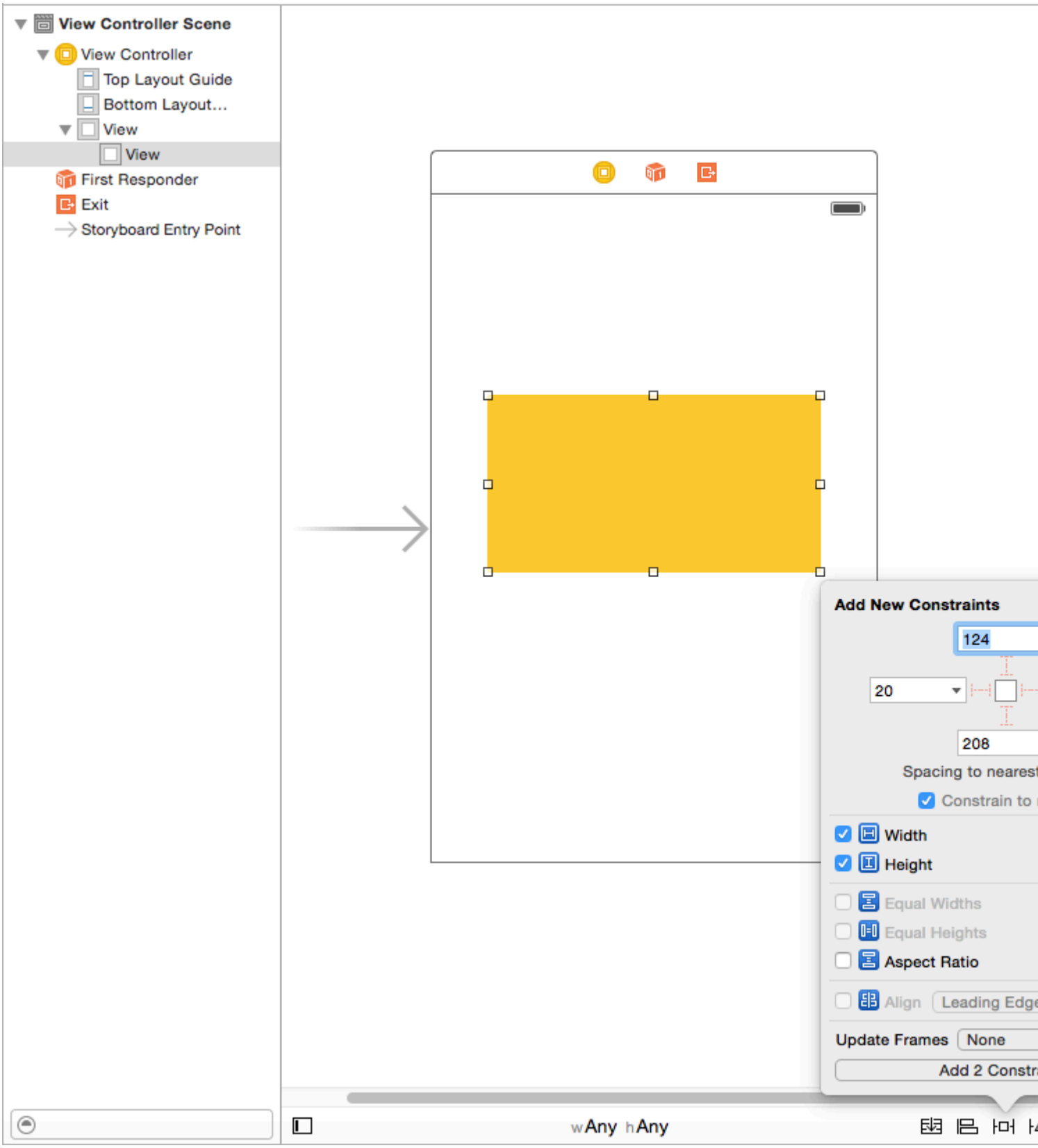
Внутренний размер UILabel

Мы должны создать представление, которое будет иметь префикс изображения для текста. текст может иметь переменную длину. Мы должны достичь результата, когда текст изображения + всегда находится в центре родительского представления.

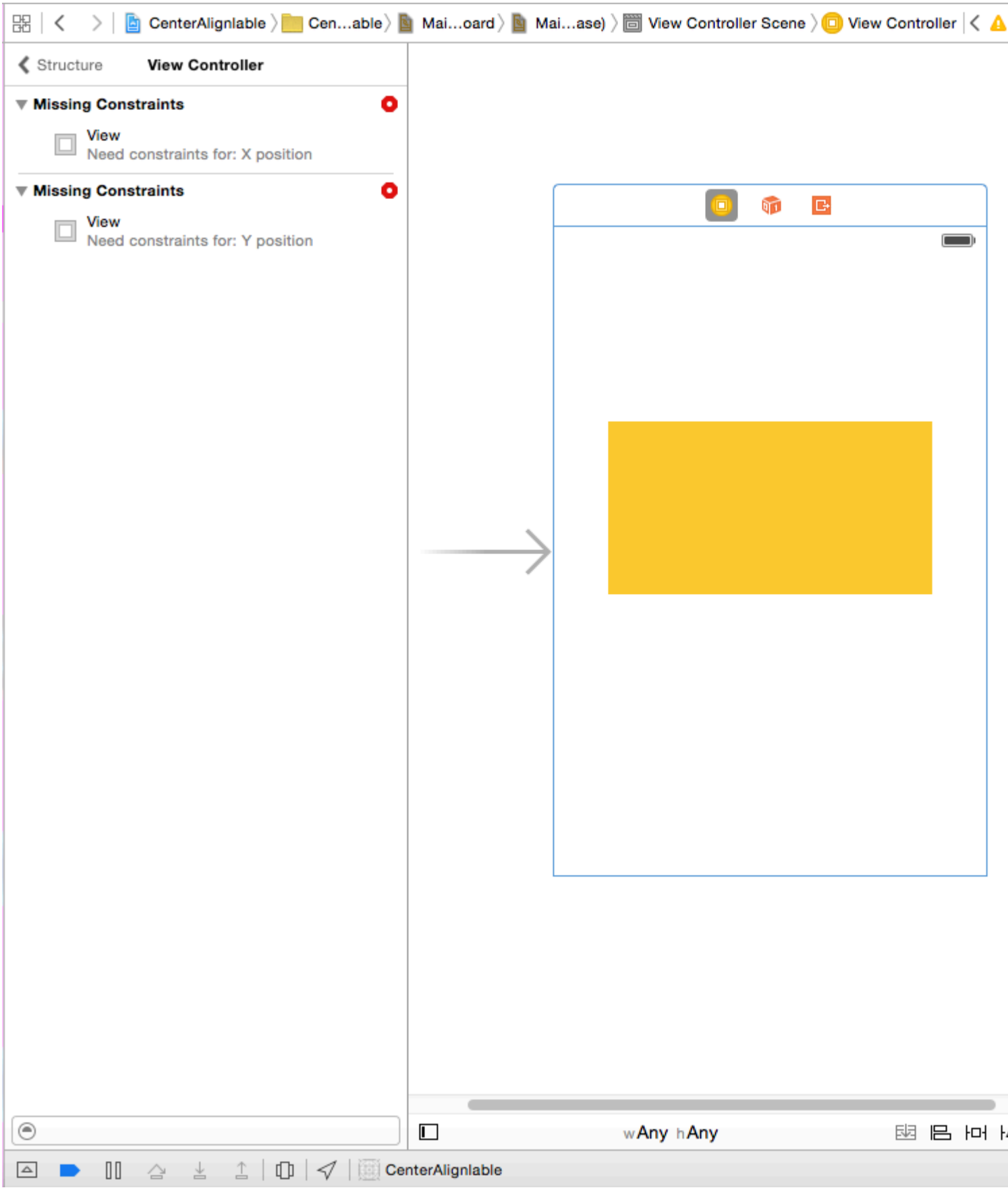
Шаг 1: Сначала создайте проект с одним видом и назовите его чем-то по своему выбору и откройте кулак из раскрывающегося списка. Сделайте вид с некоторым разумным размером и установите его цвет фона на желтый. Я изменил размер моего монитора представления на 3.5 ". взгляд должен выглядеть примерно так



Шаг 2. Теперь мы добавим ограничения на желтый вид. Для начала добавим ограничения ширины и высоты. Подождите минуту, мы не сказали, что представление будет иметь динамическую ширину? Хорошо, мы вернемся к нему позже. Добавьте следующие ограничения в соответствии с приведенным ниже изображением не беспокоятся о значении ширины, любое значение будет просто отлично для ширины, просто держите его достаточно большим, чтобы мы могли правильно добавлять автоопределения.



После добавления этих двух ограничений вы увидите, что XCode дает вам ошибки, как показано на рисунке ниже, чтобы увидеть их и понять.

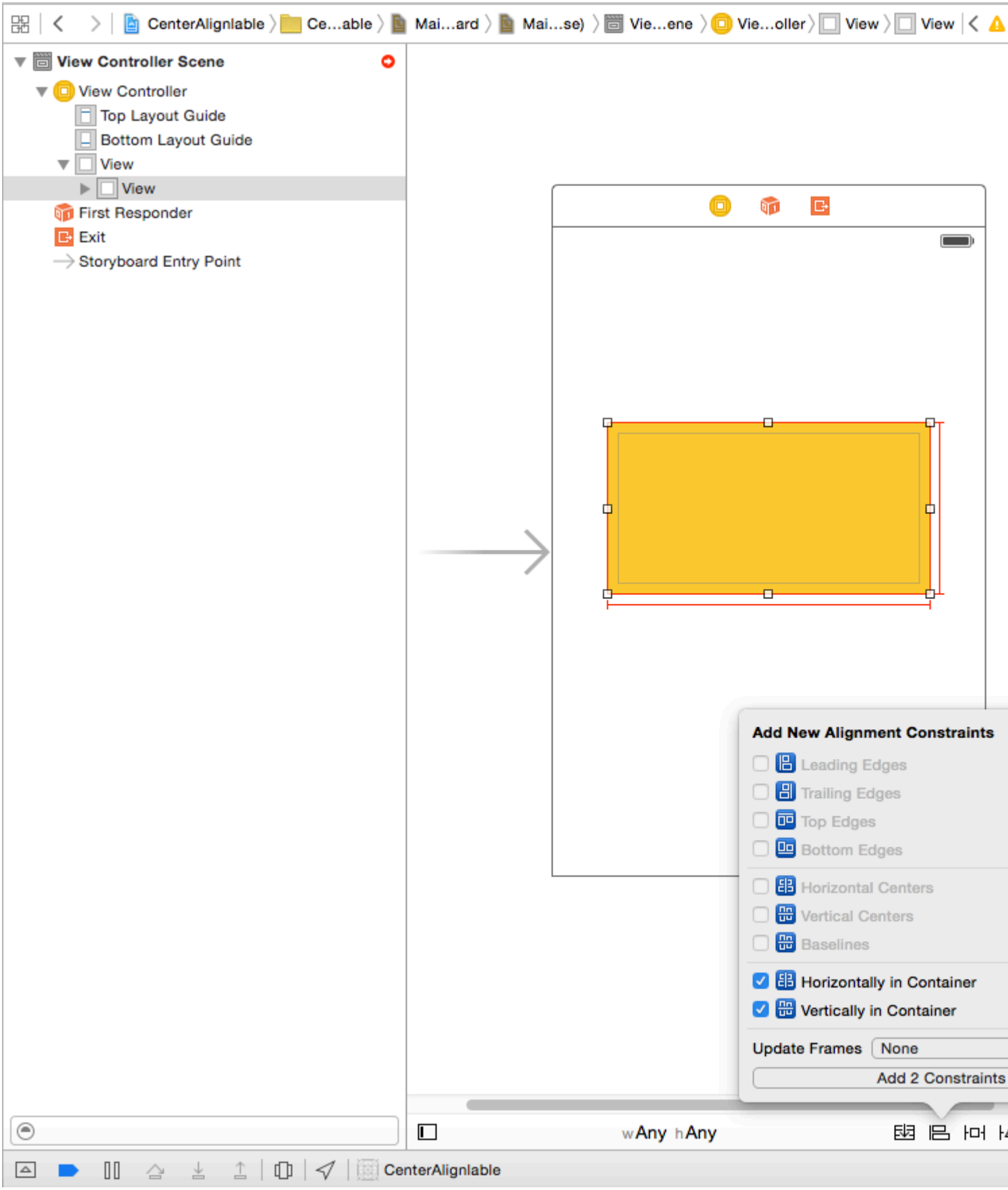


У нас есть две ошибки (красная означает ошибку). Как обсуждалось выше, можно пересмотреть часть двусмысленности

Недостающие ограничения: Необходимые ограничения для: X-позиции: - Как

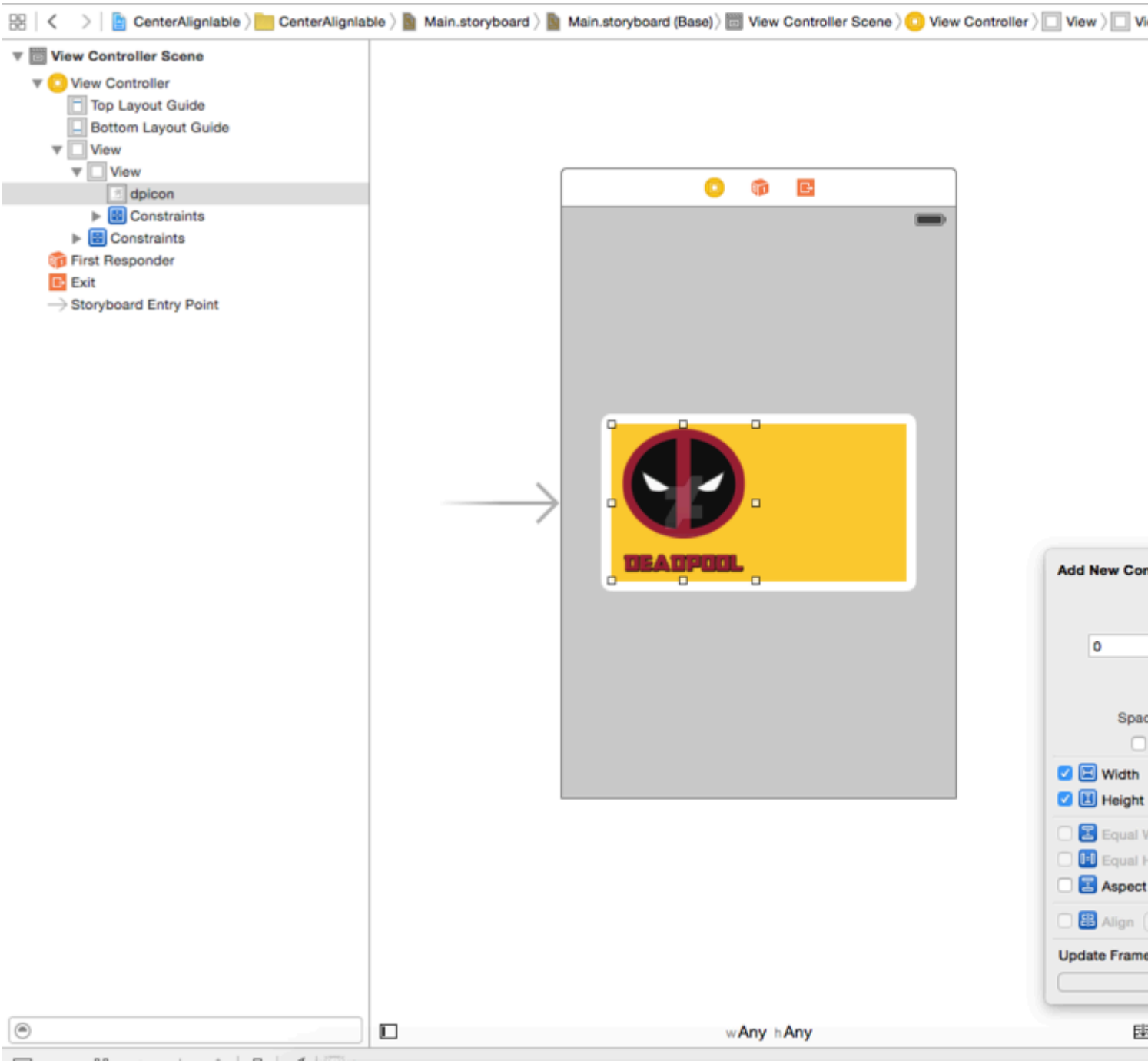
обсуждалось выше, мы дали представление ширину и высоту, чтобы определить ее «БОЛОНЫ», но мы не дали ее происхождения, поэтому ее «РАМКА» не определена. Autolayout не может определить, какова будет позиция X нашего желтого вида

Отсутствующие ограничения: Необходимые ограничения для: Y-позиции: - Как обсуждалось выше, мы дали представление ширину и высоту, чтобы определить ее « BOUNDS», но мы не дали ее происхождения, поэтому ее «FRAME» не определена. Autolayout не может определить, какова будет Y-позиция нашего желтого обзора. Чтобы решить эту проблему, мы должны дать автоопределению что-то, чтобы изменить X и Y. Поскольку мы не можем установить фреймы, мы сделаем это автоматически. Добавьте следующие ограничения в соответствии с ниже, я объясню позже



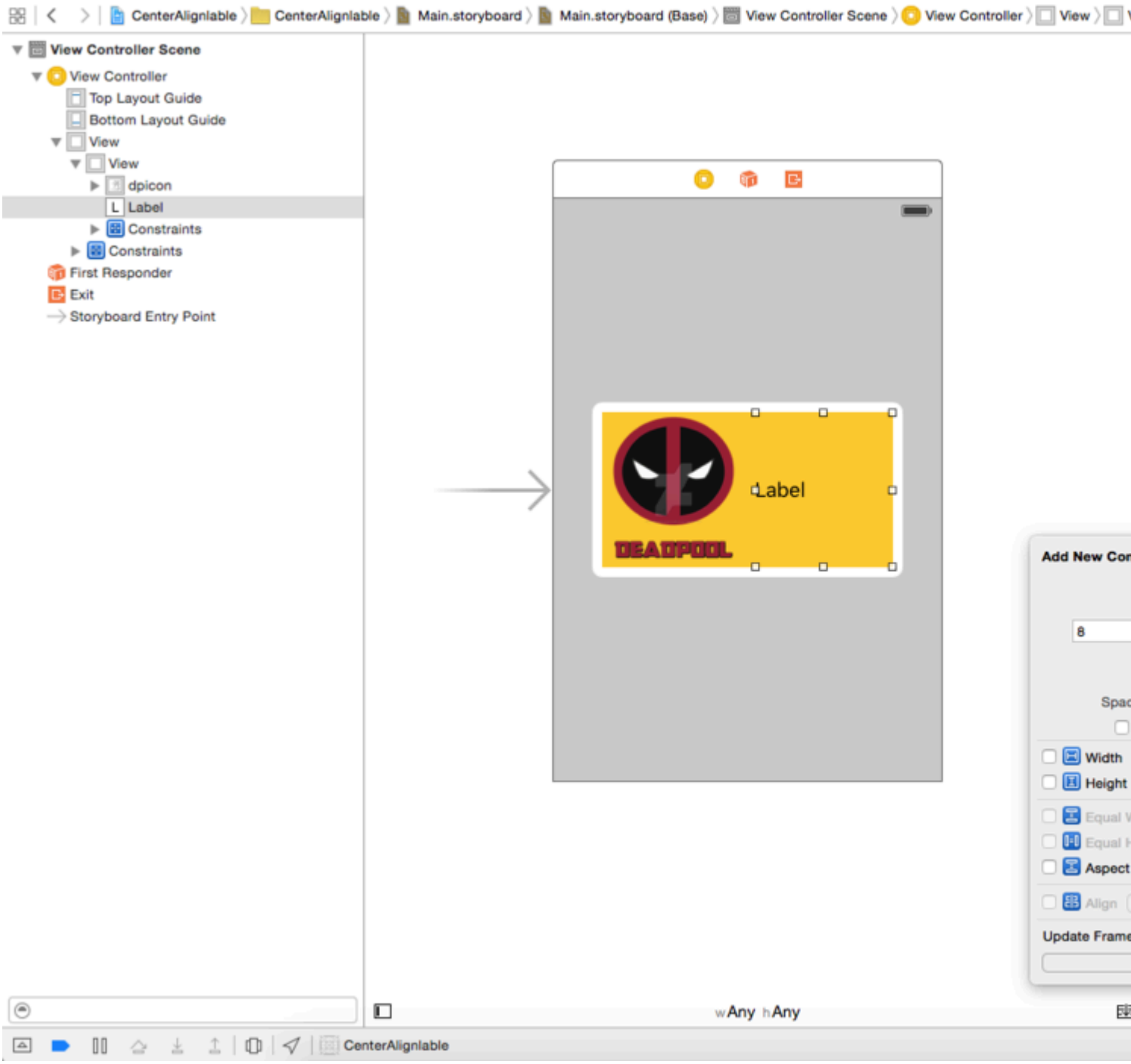
Что мы сделали, мы добавили «Вертикальный центр» и «Горизонтальный центр», которые ограничивают автоспуск, что наш желтый вид всегда будет находиться в центре по горизонтали: так что X в определенном порядке с вертикальным ограничением и определяется Y (вы возможно, придется отрегулировать рамку).

Шаг 3: Теперь наш базовый желтый вид готов. Мы добавим префиксное изображение в качестве подсмотра нашего желтого представления со следующими ограничениями. Вы можете выбрать любое изображение по вашему выбору.



Поскольку у нас фиксированный размер для нашего префикса, мы будем иметь фиксированную ширину ширины для этого изображения. Добавьте ограничения и переходите к следующему шагу.

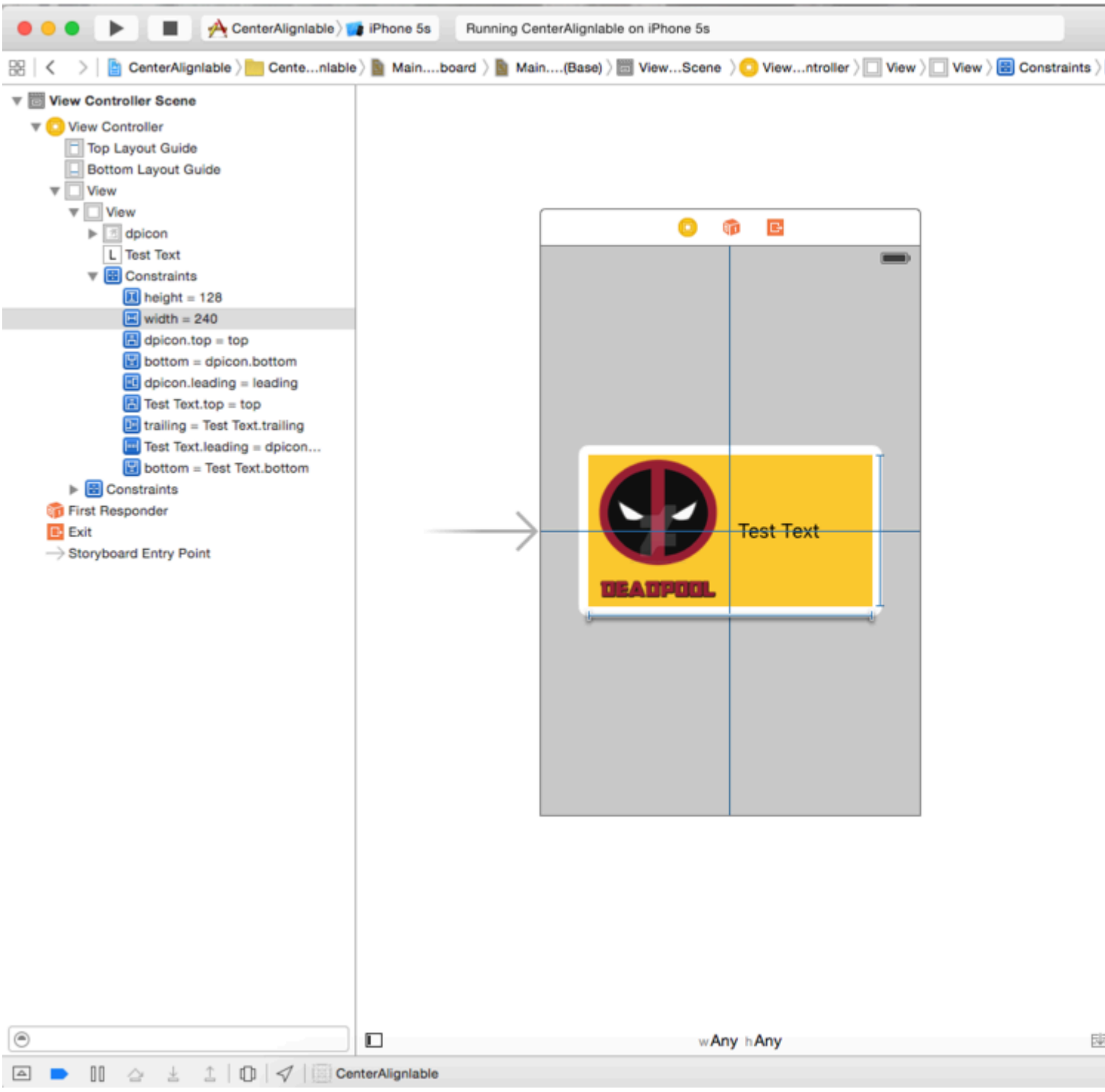
Шаг 4: добавьте UILabel в качестве дополнительного представления нашего желтого представления и добавьте следующие ограничения



Как вы можете видеть, я дал только относительные ограничения для нашего UILabel.Its 8 баллов из префикса и 0,0,0 верхний трейлинг и снизу из желтого представления. Поскольку мы хотим, чтобы ширина была динамической, мы не будем ограничивать ширину или высоту ,

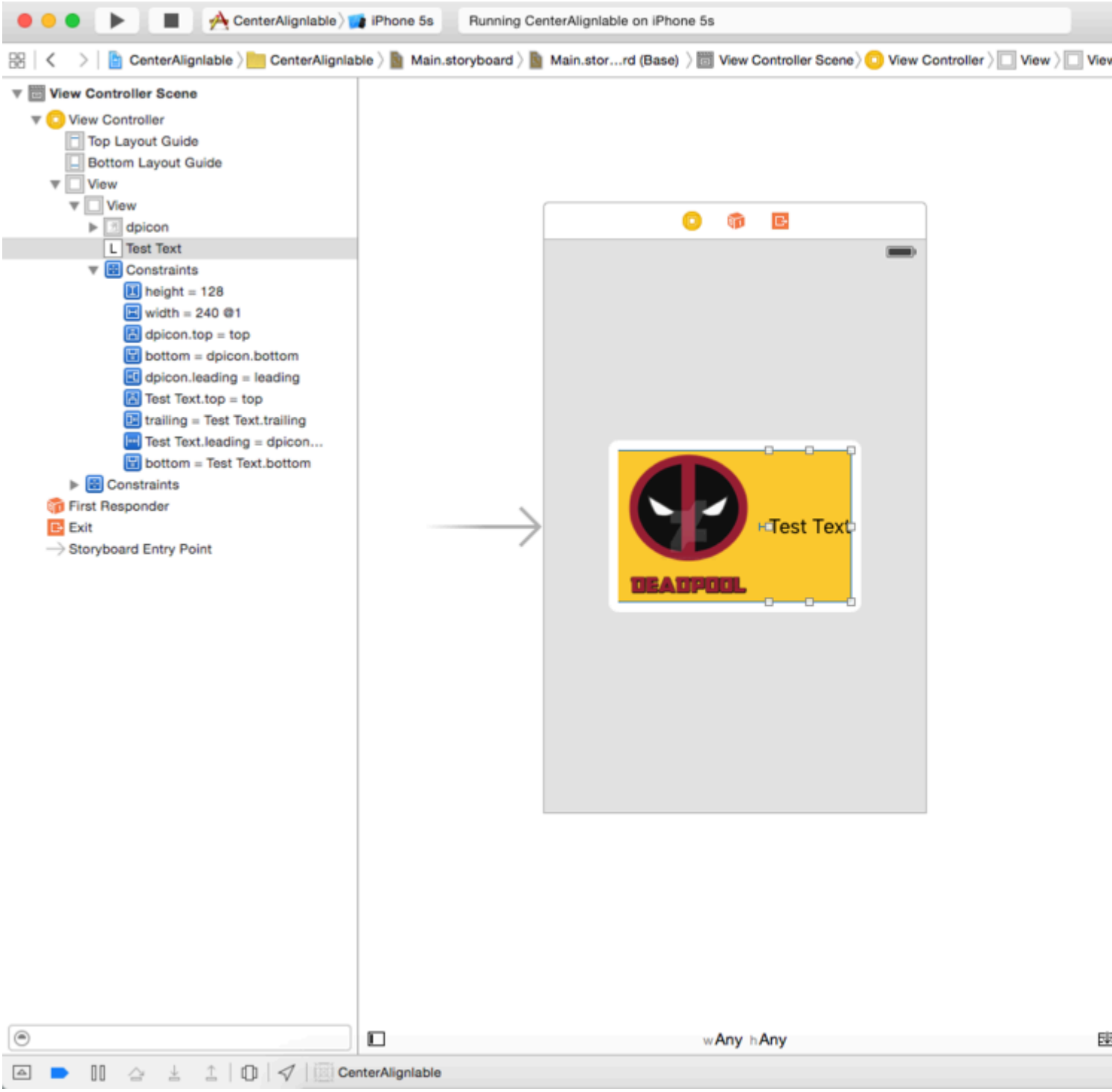
B: Почему мы не получаем никаких ошибок сейчас, мы не дали никакой ширины и высоты? Ans: - Мы получаем сообщение об ошибке или предупреждении только в том случае, если автомат не может разрешить любую вещь, которая необходима для визуализации представления на экране. Пусть это ширина или начало по высоте. Поскольку наша метка относится к желтому представлению и префиксному изображению и их кадрам четко определен автозапуск, способный вычислить рамку нашей метки.

Шаг 5: Теперь, если вспомнить, мы поймем, что мы дали фиксированный взгляд на желтый вид, но хотим, чтобы он был динамически зависим от текста нашей метки. Поэтому мы изменим нашу ширину. Ограничение желтого цвета. Ширина желтого вида необходимо устранить двусмысленность, но мы хотим, чтобы он был отменен во время выполнения на основе содержимого UILabel. Поэтому мы выберем наш желтый вид и перейдем к диспетчеру размера и уменьшим приоритет ограничения ширины до 1, чтобы он был превышен. Следуйте приведенному ниже изображению.



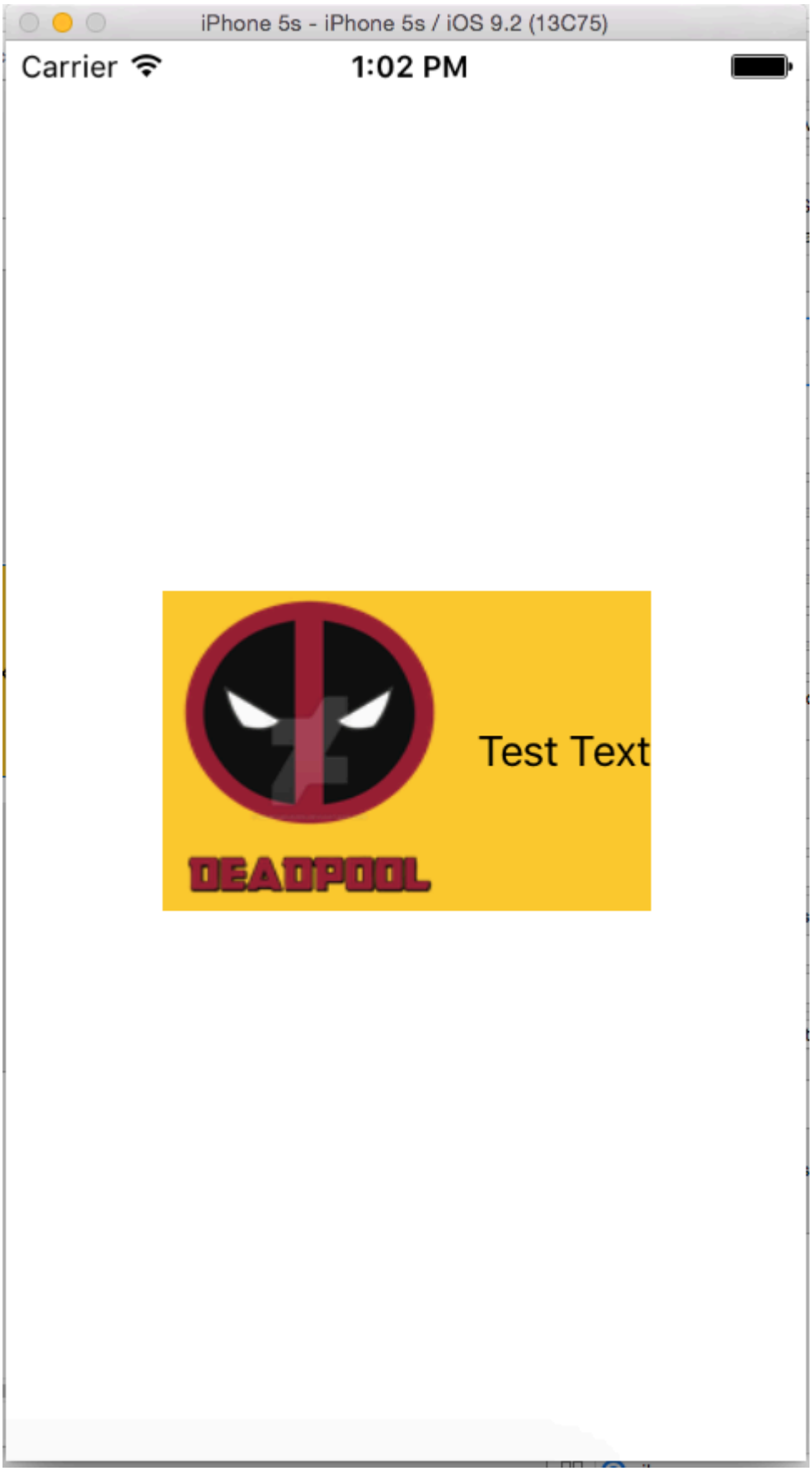
Шаг 6: Мы хотим, чтобы UILabel расширялась в соответствии с текстом и выдвигала наш желтый вид. Итак, мы уменьшили приоритет желтой ширины. Теперь мы увеличим приоритет сопротивления сжатию текста нашего UILabel. Мы хотим, чтобы наше мнение

уменьшалось как так что мы увеличим приоритет обнимания содержимого UILabel.



Как вы можете видеть, мы увеличили приоритет обхода контента до 500 и приоритет сопротивления сжатию до 751, который будет успешно превышать 1 приоритет ограничения ширины.

Теперь создадим и запустим, вы увидите что-то следующее.



Как анимировать с помощью автоматической компоновки

Без автоматической компоновки анимация выполняется с изменением фрейма представления с течением времени. С помощью Auto Layout ограничения задают рамку

представления, поэтому вам нужно анимировать ограничения. Это направление делает анимацию сложнее визуализировать.

Ниже приведены способы анимации с помощью Auto Layout:

1. **Измените константу ограничения** после создания с помощью периодических вызовов (`CADisplayLink` , `dispatch_source_t` , `dispatch_after` , `NSTimer`). Затем вызовите `layoutIfNeeded` чтобы обновить ограничение. Пример:

Objective-C:

```
self.someConstraint.constant = 10.0;
[UIView animateWithDuration:0.25 animations:^{
    [self.view layoutIfNeeded];
}];
```

Swift:

```
self.someConstraint.constant = 10.0
UIView.animate(withDuration: 0.25, animations: self.view.layoutIfNeeded)
```

2. **Измените ограничения** и вызовите `[view layoutIfNeeded]` внутри блока анимации. Это интерполирует между двумя положениями, игнорируя ограничения во время анимации.

```
[UIView animateWithDuration:0.5 animations:^{
    [view layoutIfNeeded];
}]
```

3. **Измените приоритет ограничений** . Это менее интенсивно процессор, чем добавление и удаление ограничений.

4. **Удалите все ограничения и используйте маски автосохранения** . Для более поздних вам нужно установить `view.translatesAutoresizingMaskIntoConstraints = YES` .

5. **Используйте ограничения, которые не мешают намеченной анимации** .

6. **Используйте представление контейнера** . Поместите надпись с использованием ограничений. Затем добавьте subview с ограничениями, которые не сражаются с анимацией, например: центр относительно супервизора. Это выгружает часть ограничений в супервизор, поэтому они не сражаются с анимацией в подвью.

7. **Анимированные слои вместо этого видят** . Преобразования слоев не вызывают автоматический макет.

```
CABasicAnimation* ba = [CABasicAnimation animationWithKeyPath:@"transform"];
ba.autoreverses = YES;
ba.duration = 0.3;
```

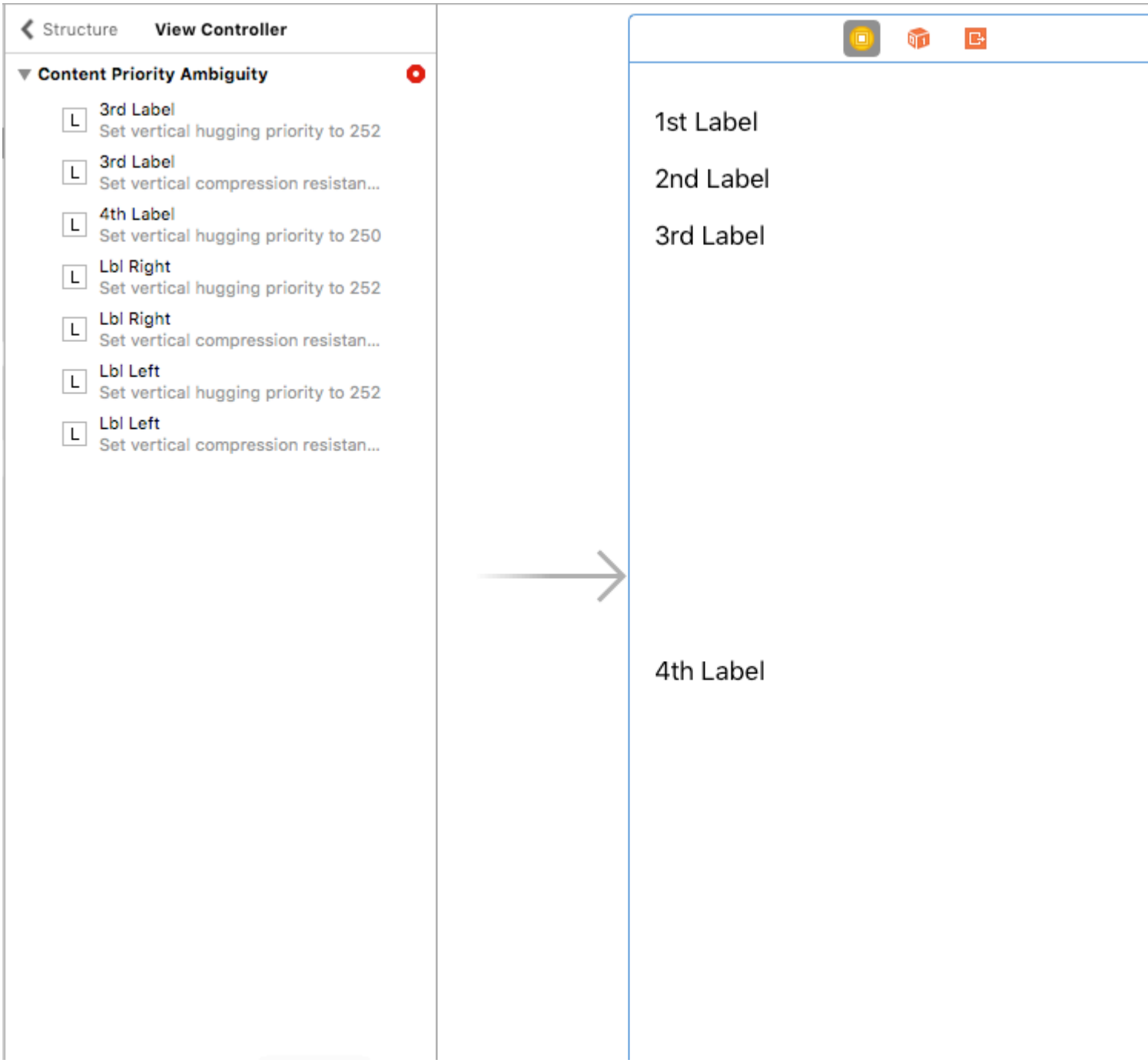
```
ba.toValue = [NSValue valueWithCATransform3D:CATransform3DMakeScale(1.1, 1.1, 1)];
[v.layer addAnimation:ba forKey:nil];
```

8. **Переопределить layoutSubviews** . Вызовите `[super layoutSubviews]` и настройте ограничения.
9. **Измените рамку в viewDidLayoutSubviews** . `layoutSubviews` применяется в `layoutSubviews` , поэтому после его выполнения измените его в `viewDidLayoutSubviews` .
10. **Откажитесь от автоматического макета** и вручную установите представления. Вы можете сделать это переопределением `layoutSubviews` / `layout` без вызова реализации суперкласса.

Быстрый совет: если родительский `layoutIfNeeded()` анимированного представления не интерполируется (т. `layoutIfNeeded()` Анимация переходит из состояния начала в конец), вызовите `layoutIfNeeded()` в самом глубоком представлении, `layoutIfNeeded()` родителем анимированного представления (другими словами , на которые не влияет анимация). Я точно не знаю, почему это работает.

Разрешить конфликт приоритетов UILabel

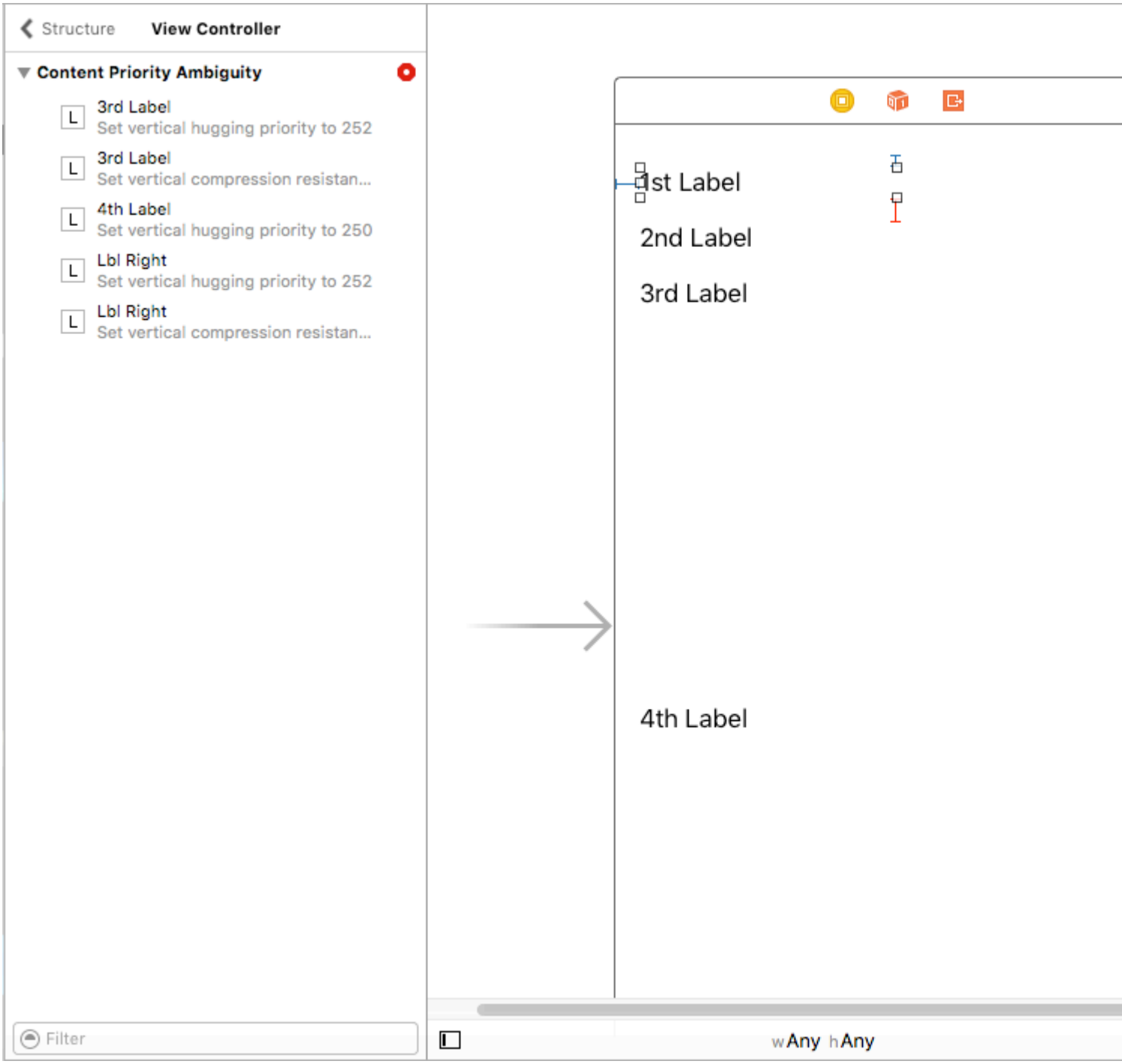
Проблема . Когда вы используете много ярлыков внутри представления, вы можете получить **предупреждение** :

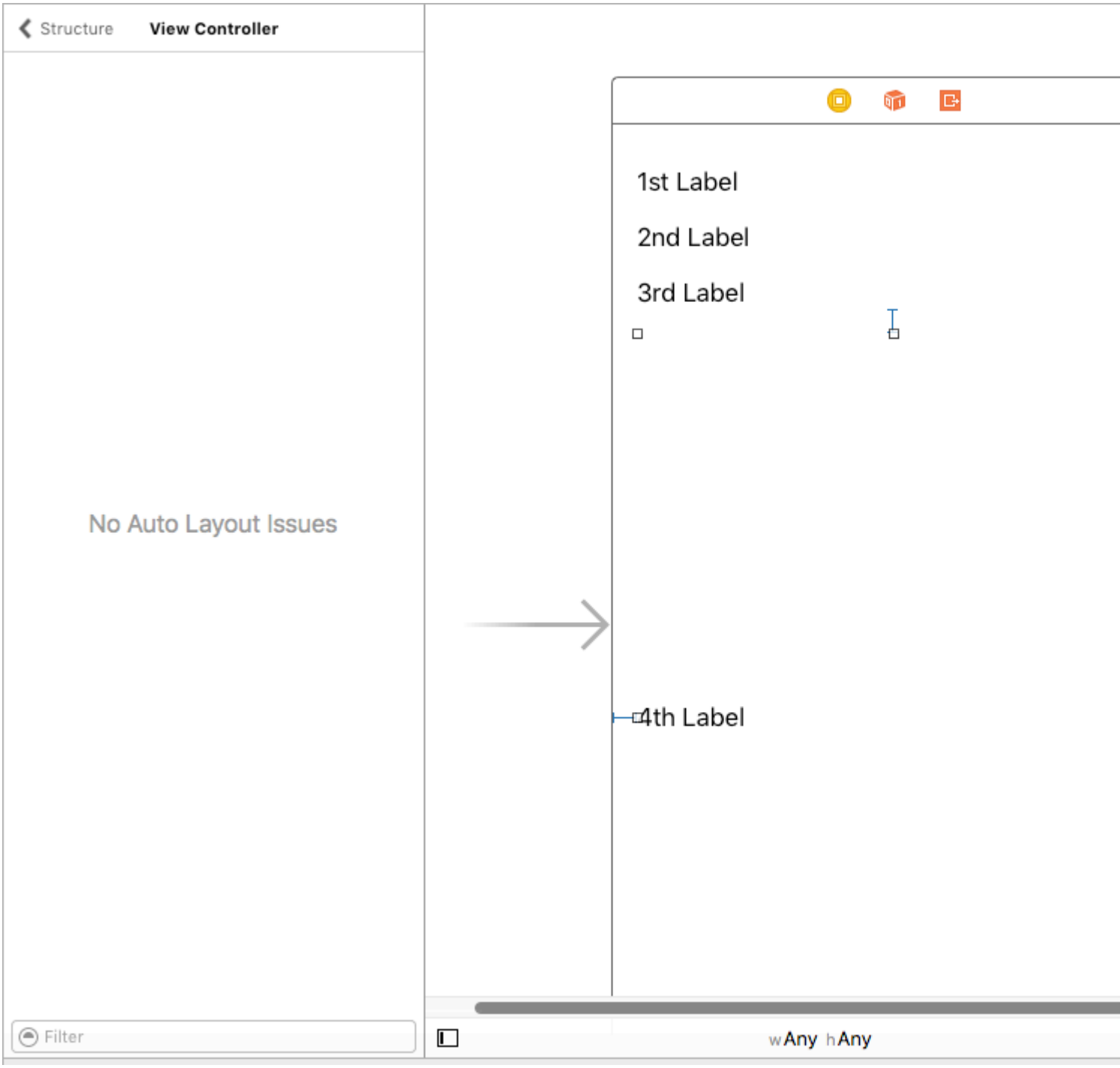


Как мы можем исправить это **предупреждение** ?

Решение . Мы рассчитываем и устанавливаем приоритеты в порядке. Приоритеты должны отличаться от ярлыков. Это означает, что важно, получит более высокий приоритет. Например, в моем случае я устанавливаю вертикальные приоритеты для своих ярлыков следующим образом:

Я установил наивысший приоритет для 1-й метки и самый низкий для 4-й метки.


Structure
View Controller
Content Priority Ambiguity
3rd Label
Set vertical hugging priority to 252
3rd Label
Set vertical compression resistan...
4th Label
Set vertical hugging priority to 250
Lbl Right
Set vertical hugging priority to 252
Lbl Right
Set vertical compression resistan...
Filter
1st Label
2nd Label
3rd Label
4th Label
wAny hAny



В ViewController, я думаю, вам трудно увидеть эффект этих приоритетов. Тем не менее, очень ясно, что UITableViewCell + оценивает высоту ячейки.

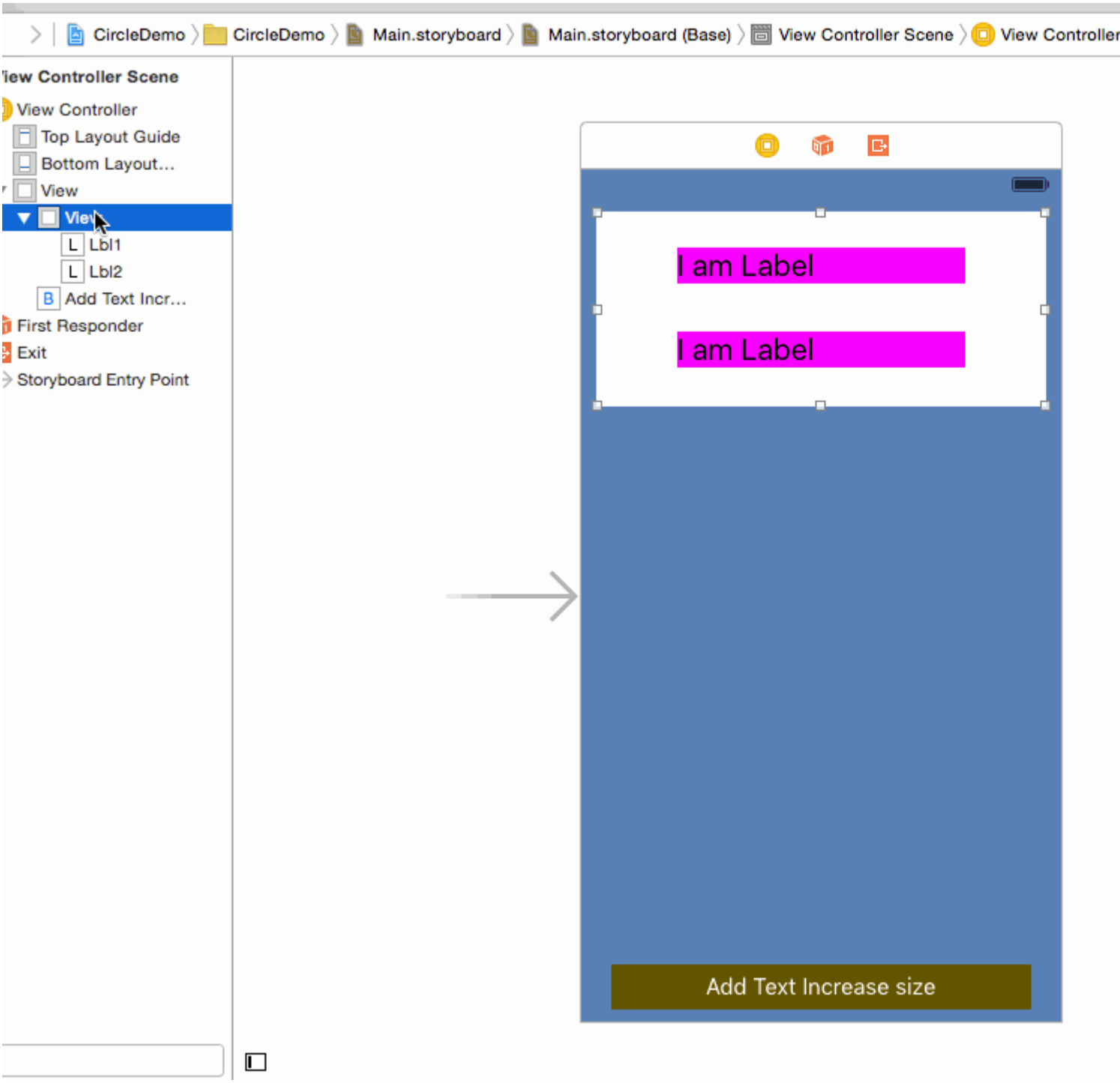
Надеюсь, эта помощь.

Размер UILabel и Parentview Согласно тексту в UILabel

Пошаговое руководство: -

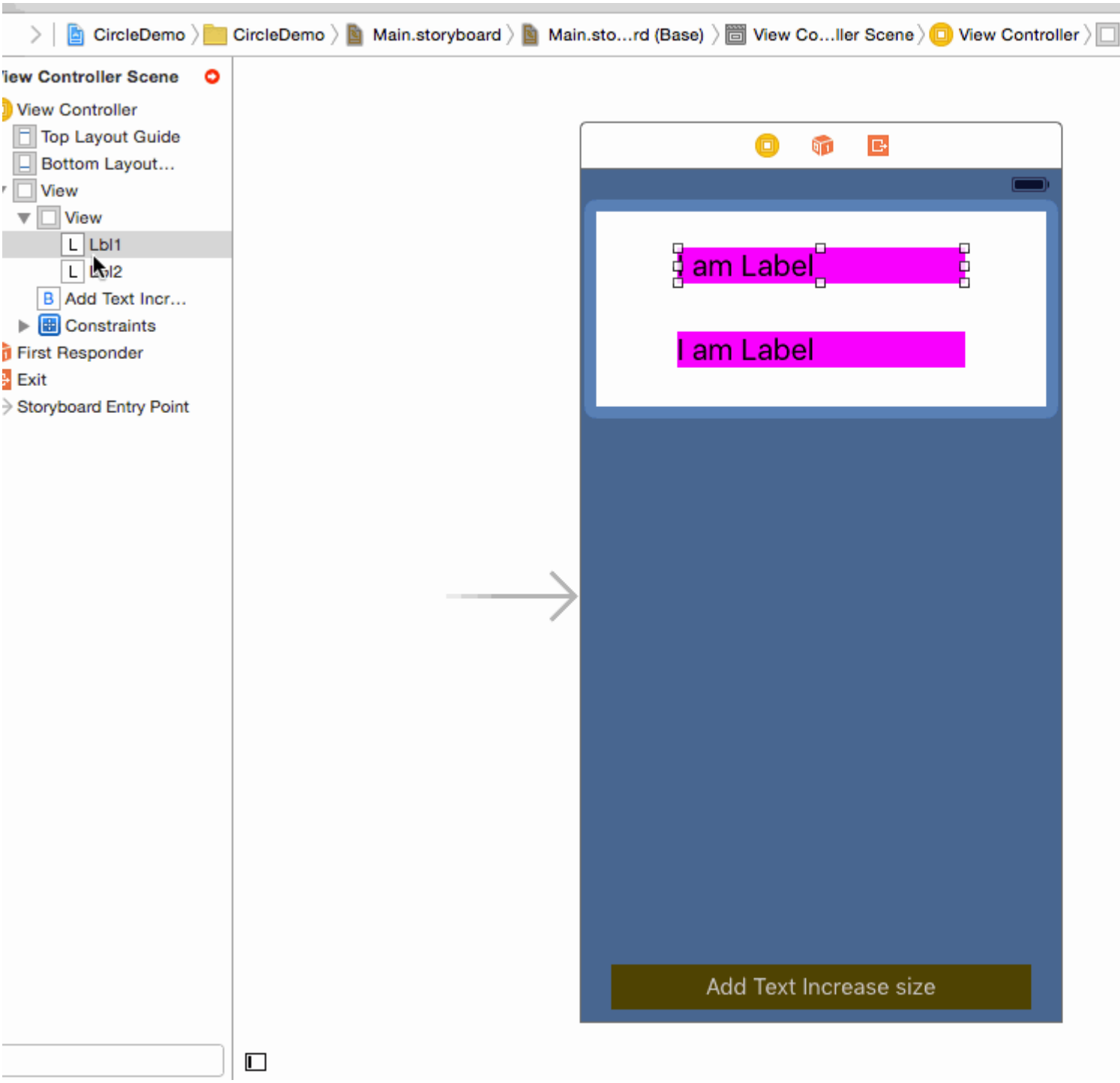
Шаг 1: - Установите ограничение на UIView

1. Ведущий. 2) Верх. 3) Трейлинг. (Из основного окна)



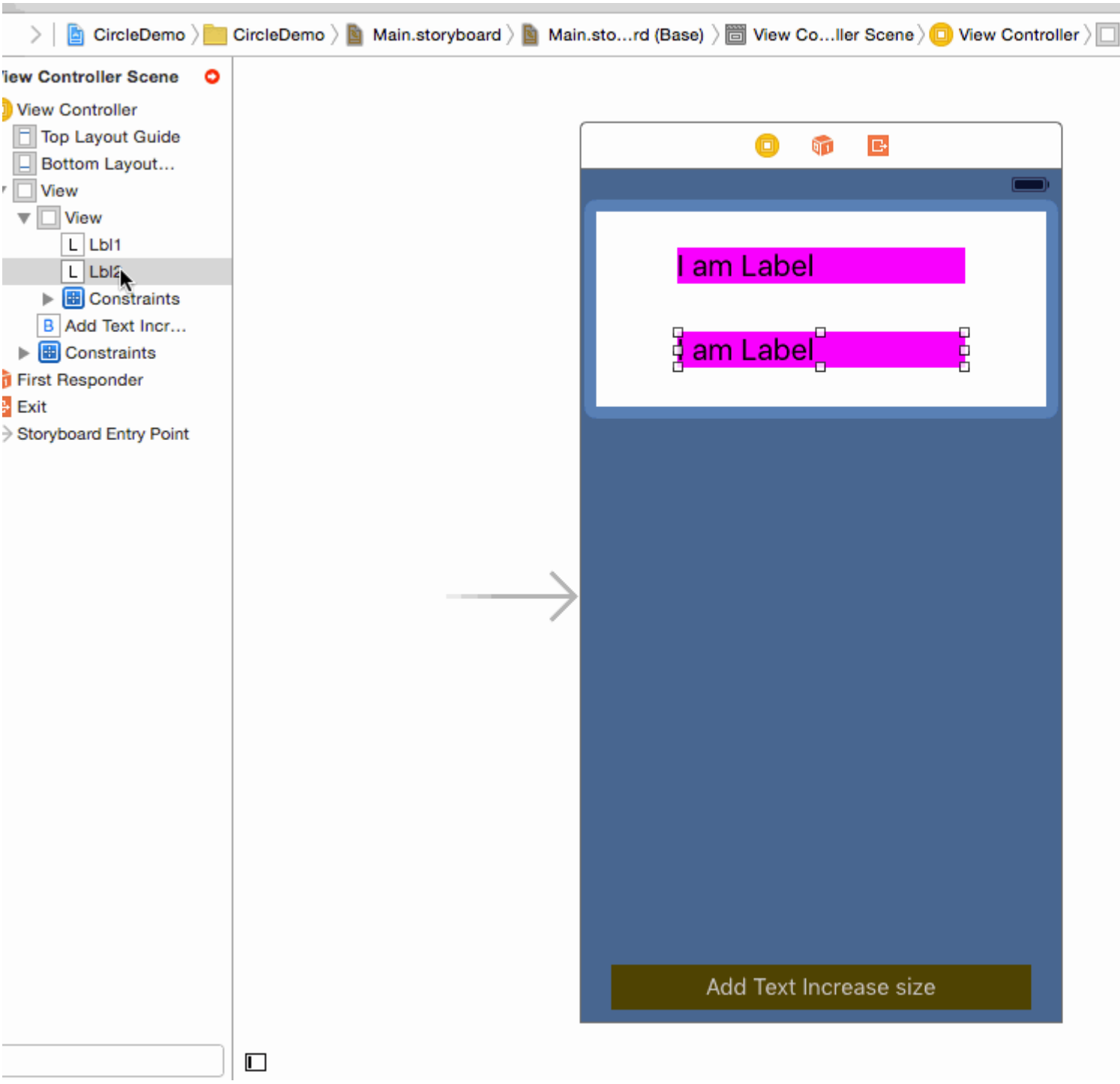
Шаг 2: - Установите ограничение на метку 1

1. Leading 2) Top 3) Trailing (Из его супервизора)

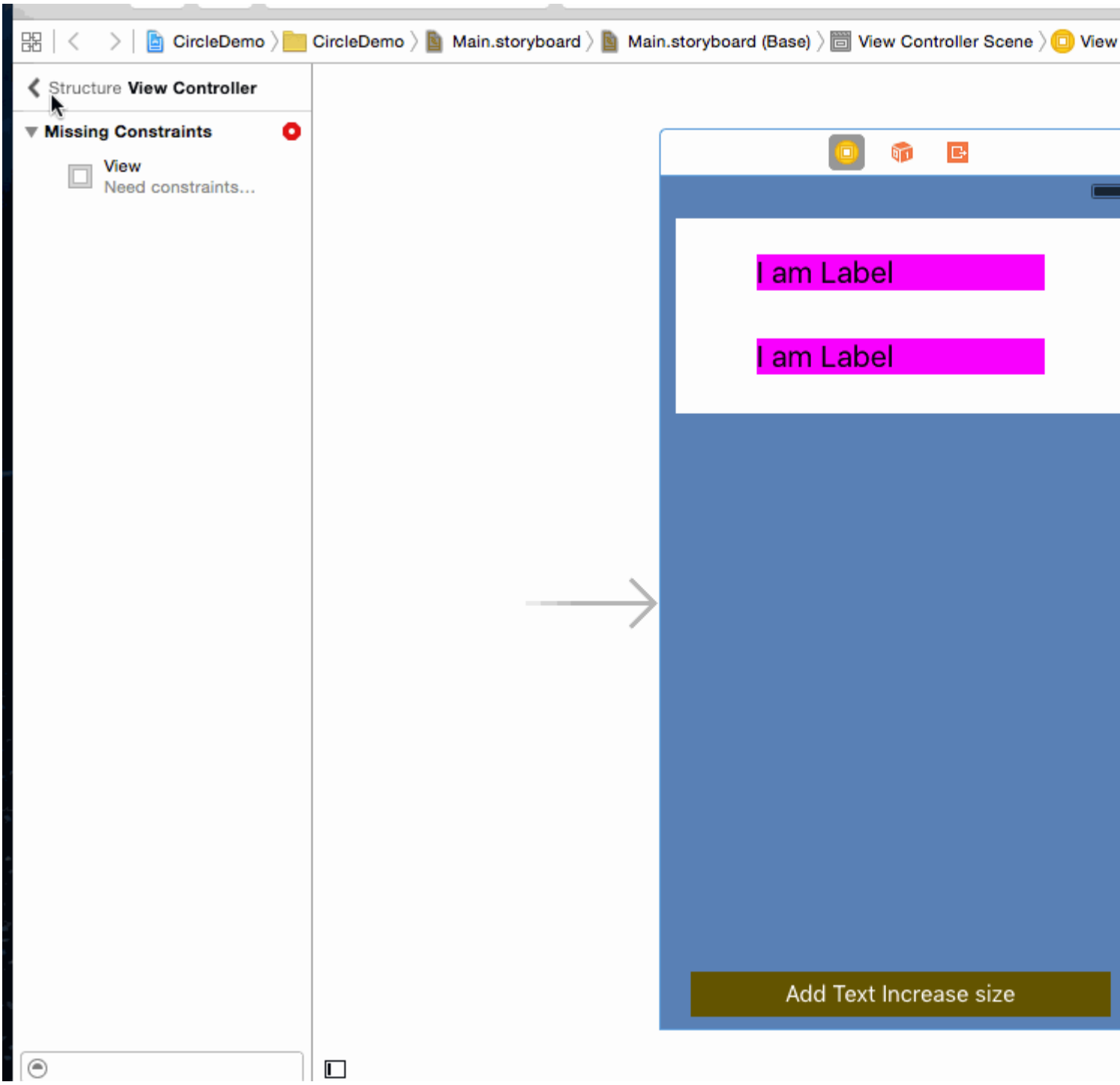


Шаг 3: - Установите ограничение на метку 2

1. Leading 2) Top 3) Trailing (из своего супервизора)

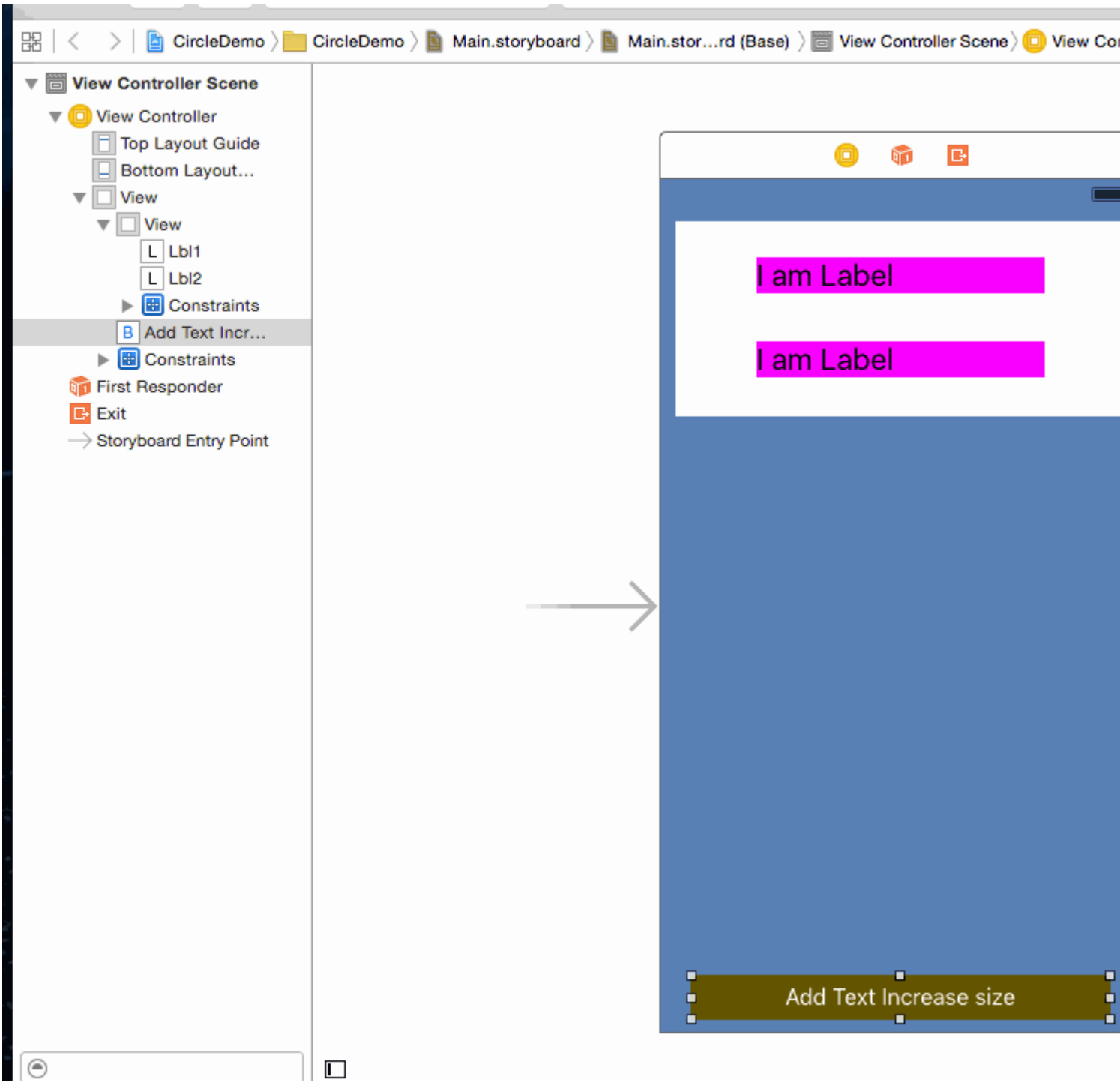


Шаг 4: - Самое сложное дать нижнюю часть UILabel из UIView.



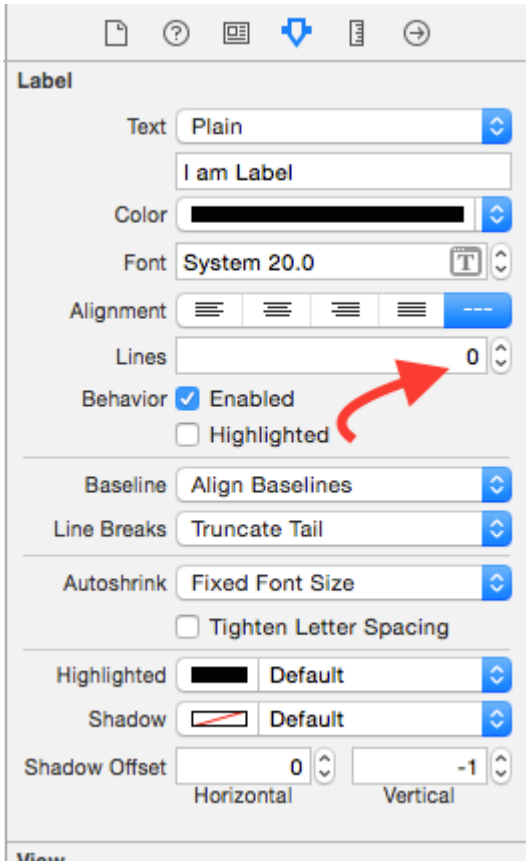
Шаг 5: - (Необязательно) Установите ограничение на UIButton

1. Ведущий 2) Нижний 3) Трейлинг 4) Фиксированная высота (из основного вида)


CircleDemo
CircleDemo
Main.storyboard
Main.stor...rd (Base)
View Controller Scene
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout...
View
View
Lbl1
Lbl2
Constraints
Add Text Incr...
Constraints
First Responder
Exit
Storyboard Entry Point
I am Label
I am Label
Add Text Increase size

Выход :-

Примечание. - Убедитесь, что вы установили Number of lines = 0 в свойстве Label.



Я надеюсь, что эта информация достаточно, чтобы понять Autoresize UIView в соответствии с высотой UILabel и Autoresize UILabel. Согласно тексту.

Основы языка Visual Format: ограничения в коде!

HVFL - это язык, предназначенный для ограничения элементов пользовательского интерфейса простым и быстрым способом. Как правило, VFL имеет преимущество перед традиционной настройкой пользовательского интерфейса в Interface Builder, потому что он гораздо читабельнее, доступен и компактен.

Вот пример VFL, в котором три UIViews ограничены слева направо, заполняя `superView.width`, с помощью `aGradeView`

```
"H:|[bgView][aGradeView(40)][bGradeView(40)]|"
```

Есть две оси, в которых мы можем ограничить объекты пользовательского интерфейса, по горизонтали и по вертикали.

Каждая строка VFL всегда начинается с `H:` или `V:` Если ни один из них не присутствует, по умолчанию используется значение `H:`

Двигаясь дальше, у нас есть конвейер. `|` Этот символ или труба относится к надзору. Если вы более подробно рассмотрите фрагмент кода VFL выше, вы заметите два из этих конвейеров.

Это означает два горизонтальных конца надзора, внешние и внешние границы.

Затем вы увидите квадратные скобки, в первом наборе квадратных скобок у нас есть `bgView` . Когда у нас есть квадратные скобки, это относится к элементу пользовательского интерфейса, теперь вы можете задаться вопросом, как установить связь между именем и фактическим элементом пользовательского интерфейса, возможно, выходом?

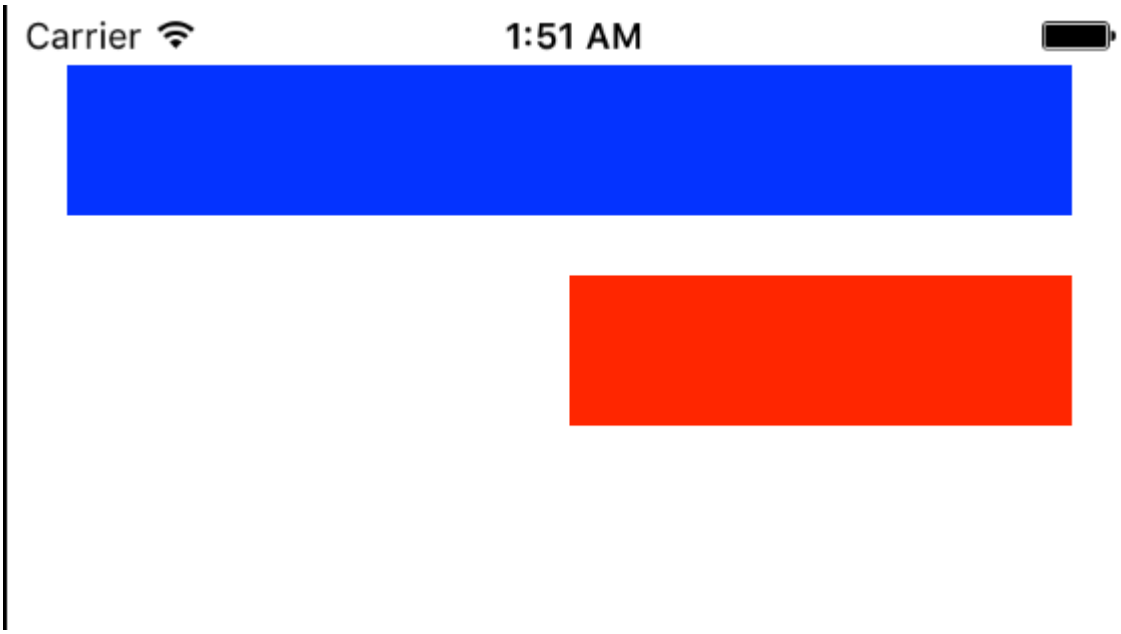
Я расскажу об этом в конце сообщения.

Если вы посмотрите на вторую пару квадратных скобок `[aGradeView(50)]` , мы также `[aGradeView(50)]` круглые скобки внутри, когда это присутствует, она определяет ширину / высоту в зависимости от осей, которая в этом случае равна 50 пикселей в ширину.

Первые квадратные скобки `[bgView]` не имеют явно определенной ширины, что означает, что она будет охватывать как можно больше.

Хорошо, это все для основ, больше о передовых материалах в другом примере.

например:




```
    // 1. create views
    UIView *blueView = [[UIView alloc] init];
    blueView.backgroundColor = [UIColor blueColor];
    [self.view addSubview:blueView];

    UIView *redView = [[UIView alloc] init];
    redView.backgroundColor = [UIColor redColor];
    [self.view addSubview:redView];

    // 2. forbid Autoresizing
    blueView.translatesAutoresizingMaskIntoConstraints = NO;
    redView.translatesAutoresizingMaskIntoConstraints = NO;

    // 3. make contraints
```

```
    // horizontal
    NSArray *blueH = [NSLayoutConstraint constraintsWithVisualFormat:@"H:|-20-[blueView]-20-|"
options:NSLayoutFormatAlignAllLeft metrics:nil views:@{@"blueView" : blueView}];
    [self.view addConstraints:blueH];

    // vertical
    NSArray *blueVandRedV = [NSLayoutConstraint constraintsWithVisualFormat:@"V:|-20-
[blueView(50)]-20-[redView(==blueView)]" options:NSLayoutFormatAlignAllTrailing metrics:nil
views:@{@"blueView" : blueView, @"redView" : redView}];
    [self.view addConstraints:blueVandRedV];

    NSLayoutConstraint *redW = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeWidth relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeWidth multiplier:0.5 constant:0];
    [self.view addConstraint:redW];
```

Смешанное использование автоматической компоновки с неавтоматической компоновкой

Иногда вы можете выполнить некоторые дополнительные действия для вычислений **Auto Layout,** выполненных самим `UIKit` .

Пример: если у вас есть `UIView` , который имеет `maskLayer` , возможно , потребуется обновить `maskLayer` как только **Auto Layout** меняет `UIView` «S `frame`

```
// CustomView.m
- (void)layoutSubviews {
    [super layoutSubviews];
    // now you can assume Auto Layout did its job
    // you can use view's frame in your calculations
    CALayer maskLayer = self.maskLayer;
    maskLayer.bounds = self.bounds;
    ...
}
```

или если вы хотите предпринять дополнительные действия для **автоматического макета** в `ViewController`

```
- (void)viewDidLayoutSubviews {
    [super viewDidLayoutSubviews];
    // now you can assume all your subviews are positioned/resized correctly
    self.customView.frame = self.containerView.frame;
}
```

Пропорциональный макет

Ограничение, созданное как

```
NSLayoutConstraint(item: myView, attribute: NSLayoutAttribute.Leading, relatedBy:
NSLayoutRelation.Equal, toItem: view, attribute: NSLayoutAttribute.LeadingMargin, multiplier:
1.0, constant: 20.0)
```

или, с математической точки зрения:

```
view.attribute * multiplier + constant          (1)
```

Вы можете использовать множитель для создания пропорционального макета для разного коэффициента.

Пример:

Бирюзовый вид (V1) представляет собой квадрат с шириной пропорциональной ширины супервизора с отношением 1: 1.1

Квадрат Гэри (V2) является подпунктом V1. Нижнее пространство, заданное константой = 60, Конечное пространство, заданное множителем = 1.125 и константой = 0

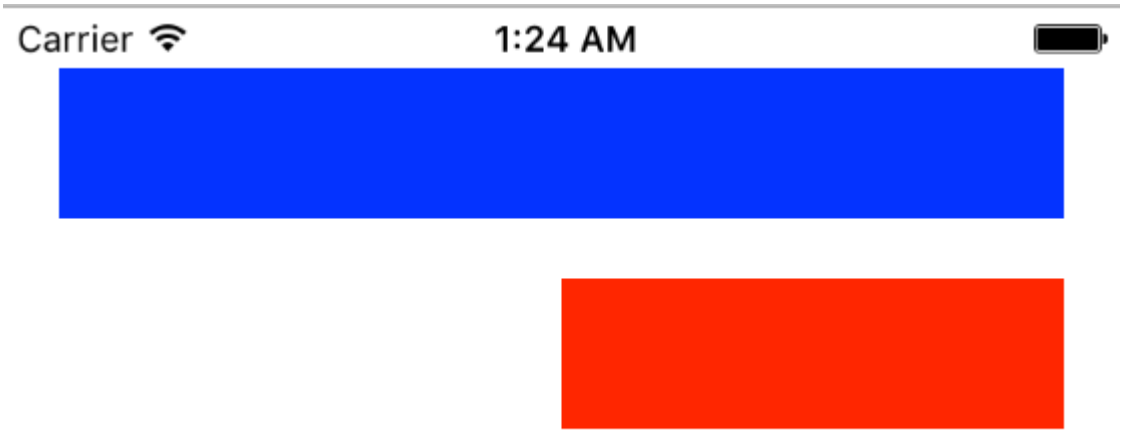
Пространственное пространство устанавливается пропорционально, нижнее пространство устанавливается как константа.

равен 0 (например, ведущее пространство), формула ограничения (1), будет равна 0. Вам нужно изменить второй элемент ограничения или установить ограничение относительно поля, чтобы view.attribute! = 0.

NSLayoutConstraint: Препятствия в коде!

Когда мы работаем над фреймворком, если ограничения не слишком сложны, нам лучше использовать Interface Builder или NSLayoutConstraint в коде, чтобы сделать его достаточно небольшим, вместо импорта масонства или SnapKit.

например:



- Objective-C

```
    // 1. create views
    UIView *blueView = [[UIView alloc] init];
    blueView.backgroundColor = [UIColor blueColor];
    [self.view addSubview:blueView];

    UIView *redView = [[UIView alloc] init];
    redView.backgroundColor = [UIColor redColor];
    [self.view addSubview:redView];

    // 2. forbid Autoresizing
    blueView.translatesAutoresizingMaskIntoConstraints = NO;
    redView.translatesAutoresizingMaskIntoConstraints = NO;

    // 3. make contraints
    // 3.1 blueView
    NSLayoutConstraint *blueLeft = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeLeft relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeLeft multiplier:1 constant:20];
    [self.view addConstraint:blueLeft];

    NSLayoutConstraint *blueTop = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeTop relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeTop multiplier:1 constant:20];
    [self.view addConstraint:blueTop];

    NSLayoutConstraint *blueRight = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeRight relatedBy:NSLayoutRelationEqual toItem:self.view
```

```
attribute:NSLayoutAttributeRight multiplier:1 constant:-20];
    [self.view addConstraint:blueRight];

    NSLayoutConstraint *blueHeight = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeHeight relatedBy:NSLayoutRelationEqual toItem:nil
attribute:NSLayoutAttributeNotAnAttribute multiplier:1 constant:50];
    [self.view addConstraint:blueHeight];

    // 3.2 redView
    NSLayoutConstraint *redTop = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeTop relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeBottom multiplier:1 constant:20];
    [self.view addConstraint:redTop];

    NSLayoutConstraint *redRight = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeRight relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeRight multiplier:1 constant:-20];
    [self.view addConstraint:redRight];

    NSLayoutConstraint *redHeight = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeHeight relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeHeight multiplier:1 constant:0];
    [self.view addConstraint:redHeight];

    NSLayoutConstraint *redWidth = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeWidth relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeWidth multiplier:0.5 constant:0];
    [self.view addConstraint:redWidth];
```

Прочитайте Автоматическая компоновка онлайн: https://riptutorial.com/ru/ios/topic/792/автоматическая-компоновка

глава 115: Архитектура MVP

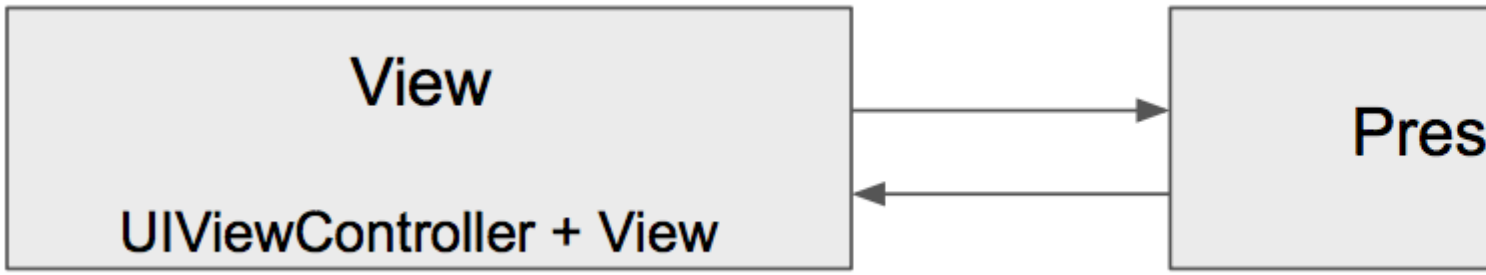
Вступление

MVP - это архитектурный образец, являющийся производством Model-View-Controller. Он представлен тремя различными компонентами: Model, View и Presenter. Он был разработан для облегчения автоматизированного модульного тестирования и улучшения разделения проблем в логике представления.

В примерах вы найдете простой проект, построенный с учетом шаблона MVP.

замечания

Компоненты:



- **Модель** является интерфейсом, ответственным за данные домена (отображаться или иным образом воздействовать на графический интерфейс)
- **View** отвечает за уровень представления (GUI)
- **Ведущий** - это «средний человек» между моделью и видом. Он реагирует на действия пользователя, выполняемые в представлении, извлекает данные из Модели и форматирует их для отображения в представлении

Компонентные обязанности:

модель	Посмотреть	Ведущий
Общается с уровнем БД	Предоставляет данные	Выполняет запросы к модели
Поднятие соответствующих событий	Получает события	Форматирует данные из модели
	Очень простая логика проверки	Отправляет форматированные данные в представление

модель	Посмотреть	Ведущий
		Комплексная логика проверки

Различия между MVC и MVP :

- Просмотр в MVC тесно связан с контроллером, часть просмотра MVP состоит из UIView и UIViewController
- MVP View настолько тупым, насколько это возможно и не содержит почти никакой логики (например, в MVVM), MVC View имеет некоторую бизнес-логику и может запрашивать модель
- MVP View обрабатывает жесты пользователей и взаимодействие делегатов с презентатором, в MVC контроллер обрабатывает жесты и команды Model
- MVP очень поддерживает модульное тестирование, MVC имеет ограниченную поддержку
- MVC Controller имеет множество зависимостей UIKit, у MVP Presenter нет ни одного

Плюсы:

- MVP делает UIViewController частью компонента View, он тупой, пассивный и ... менее массивный;]
- Большая часть бизнес-логики инкапсулируется из-за немых просмотров, это дает отличную возможность тестирования. Мощные объекты могут быть введены для проверки части домена.
- Отдельные лица легче держать в голове, обязанности четко разделены.

Cons

- Вы напишете больше кода.
- Барьер для неопытных разработчиков или для тех, кто еще не работает с шаблоном.

Examples

Dog.swift

```
import Foundation

enum Breed: String {
    case bulldog  = "Bulldog"
    case doberman = "Doberman"
    case labrador = "Labrador"
}

struct Dog {
    let name:  String
    let breed: String
    let age:   Int
```

```
}
```

DoggyView.swift

```
import Foundation

protocol DoggyView: NSObjectProtocol {
    func startLoading()
    func finishLoading()
    func setDoggies(_ doggies: [DoggyViewData])
    func setEmpty()
}
```

DoggyService.swift

```
import Foundation

typealias Result = ([Dog]) -> Void

class DoggyService {

    func deliverDoggies(_ result: @escaping Result) {

        let firstDoggy  = Dog(name: "Alfred", breed: Breed.labrador.rawValue, age: 1)
        let secondDoggy = Dog(name: "Vinny",  breed: Breed.doberman.rawValue, age: 5)
        let thirdDoggy  = Dog(name: "Lucky",  breed: Breed.labrador.rawValue, age: 3)

        let delay = DispatchTime.now() + Double(Int64(Double(NSEC_PER_SEC)*2)) /
Double(NSEC_PER_SEC)

        DispatchQueue.main.asyncAfter(deadline: delay) {
            result([firstDoggy,
                    secondDoggy,
                    thirdDoggy])
        }
    }
}
```

DoggyPresenter.swift

```
import Foundation

class DoggyPresenter {

    // MARK: - Private
    fileprivate let dogService: DoggyService
    weak fileprivate var dogView: DoggyView?

    init(dogService: DoggyService){
        self.dogService = dogService
    }

    func attachView(_ attach: Bool, view: DoggyView?) {
        if attach {
            dogView = nil
        } else {
```

```
            if let view = view { dogView = view }
        }
    }

    func getDogs(){
        self.dogView?.startLoading()

        dogService.deliverDoggies { [weak self] doggies in
            self?.dogView?.finishLoading()

        if doggies.count == 0 {
            self?.dogView?.setEmpty()
        } else {
            self?.dogView?.setDoggies(doggies.map {
                return DoggyViewData(name: "\($0.name) \($0.breed)",
                                     age: "\($0.age)")
                })
            }
        }
    }
}

struct DoggyViewData {
    let name: String
    let age:  String
}
```

DoggyListViewController.swift

```
import UIKit

class DoggyListViewController: UIViewController, UITableViewDataSource {

    @IBOutlet weak var emptyView: UIView?
    @IBOutlet weak var tableView: UITableView?
    @IBOutlet weak var spinner:   UIActivityIndicatorView?

    fileprivate let dogPresenter = DoggyPresenter(dogService: DoggyService())
    fileprivate var dogsToDisplay = [DoggyViewData]()

    override func viewDidLoad() {
        super.viewDidLoad()

        tableView?.dataSource = self
        spinner?.hidesWhenStopped = true
        dogPresenter.attachView(true, view: self)
        dogPresenter.getDogs()
    }

    // MARK: DataSource
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return dogsToDisplay.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let cell = UITableViewCell(style: .subtitle, reuseIdentifier: "Cell")
        let userViewData = dogsToDisplay[indexPath.row]
        cell.textLabel?.text = userViewData.name
```

```
        cell.detailTextLabel?.text = userViewData.age
        return cell
    }
}

extension DoggyListViewController: DoggyView {

    func startLoading() {
        spinner?.startAnimating()
    }

    func finishLoading() {
        spinner?.stopAnimating()
    }

    func setDoggies(_ doggies: [DoggyViewData]) {
        dogsToDisplay = doggies
        tableView?.isHidden = false
        emptyView?.isHidden = true;
        tableView?.reloadData()
    }

    func setEmpty() {
        tableView?.isHidden = true
        emptyView?.isHidden = false;
    }
}
```

Прочитайте Архитектура MVP онлайн: https://riptutorial.com/ru/ios/topic/9467/архитектура-mvp

глава 116: Безопасность

Вступление

Безопасность в iOS связана с безопасностью данных, безопасностью транспорта, защитой кода и т. Д.

Examples

Безопасность транспорта с использованием SSL

Приложения iOS должны быть написаны таким образом, чтобы обеспечить безопасность данных, которые транспортируются по сети.
SSL - это общий способ сделать это.

Всякий раз, когда приложение пытается вызвать веб-службы, чтобы вытащить или перенаправить данные на серверы, он должен **использовать SSL через HTTP, то есть HTTPS** .
Для этого **приложение должно вызывать** `https://server.com/part` такие веб-службы, а не `http://server.com/part` .
В этом случае приложение должно доверять серверу `server.com` с использованием сертификата SSL.

Ниже приведен пример проверки доверия к серверу,

Внедрить `URLSessionDelegate` как:

```
func urlSession(_ session: URLSession, didReceive challenge: URLAuthenticationChallenge,
completionHandler: @escaping (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {

    if challenge.protectionSpace.authenticationMethod == NSURLAuthenticationMethodServerTrust
{
        let serverTrust:SecTrust = challenge.protectionSpace.serverTrust!

        func acceptServerTrust() {
            let credential:URLCredential = URLCredential(trust: serverTrust)
            challenge.sender?.use(credential, for: challenge)
            completionHandler(.useCredential, URLCredential(trust:
challenge.protectionSpace.serverTrust!))
        }

        let success = SSLTrustManager.shouldTrustServerTrust(serverTrust, forCert:
"Server_Public_SSL_Cert")
        if success {
            acceptServerTrust()
            return
        }
    }
    else if challenge.protectionSpace.authenticationMethod ==
```

```
NSURLAuthenticationMethodClientCertificate {
        completionHandler(.rejectProtectionSpace, nil);
        return
    }
    completionHandler(.cancelAuthenticationChallenge, nil)
}
```

Вот менеджер доверия: (не удалось найти код Swift)

```
@implementation SSLTrustManager
+ (BOOL)shouldTrustServerTrust:(SecTrustRef)serverTrust forCert:(NSString*)certName {
// Load up the bundled certificate.
NSString *certPath = [[NSBundle mainBundle] pathForResource:certName ofType:@"der"];
NSData *certData = [[NSData alloc] initWithContentsOfFile:certPath];
CFDataRef certDataRef = (__bridge_retained CFDataRef)certData;
SecCertificateRef cert = SecCertificateCreateWithData(NULL, certDataRef);

// Establish a chain of trust anchored on our bundled certificate.
CFArrayRef certArrayRef = CFArrayCreate(NULL, (void *)&cert, 1, NULL);
SecTrustSetAnchorCertificates(serverTrust, certArrayRef);

// Verify that trust.
SecTrustResultType trustResult;
SecTrustEvaluate(serverTrust, &trustResult);

// Clean up.
CFRelease(certArrayRef);
CFRelease(cert);
CFRelease(certDataRef);

// Did our custom trust chain evaluate successfully?
return trustResult == kSecTrustResultUnspecified;
}
@end
```

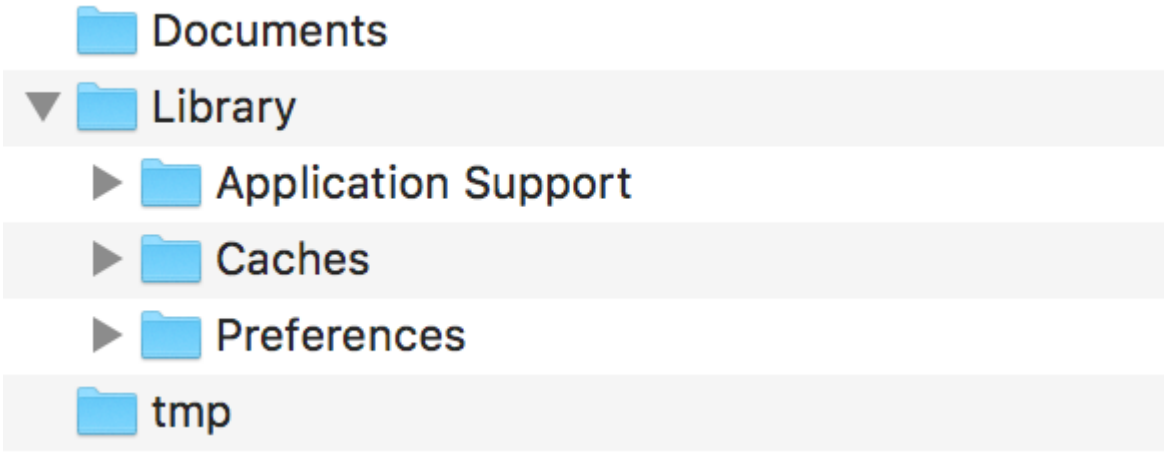
Server_Public_SSL_Cert.der является общедоступным SSL-сервером.

Используя этот подход, наше приложение может убедиться в том, что оно связано с предполагаемым сервером, и никто не перехватывает связь приложения с сервером.

Защита данных в резервных копиях iTunes

Если мы хотим, чтобы наши данные приложений были защищены от резервных копий iTunes, мы должны пропустить данные наших приложений из резервной копии в iTunes. Всякий раз, когда устройство iOS резервируется с помощью iTunes на macOS, все данные, хранящиеся во всех приложениях, копируются в эту резервную копию и сохраняются на резервном компьютере.

Но мы можем исключить данные приложения из этой резервной копии, используя ключ `URLResourceKey.isExcludedFromBackupKey` .
Вот структура каталогов нашего приложения:



Примечание. Обычно конфиденциальные данные хранятся в каталоге «Поддержка приложений».

например, если мы хотим исключить все наши данные, хранящиеся в каталоге **поддержки приложений,** мы можем использовать вышеупомянутый ключ следующим образом:

```
    let urls = FileManager.default.urls(for: .applicationSupportDirectory, in:
.userDomainMask)
    let baseURL = urls[urls.count-1];

    let bundleIdentifier = Bundle.main.object(forInfoDictionaryKey: "CFBundleIdentifier") as!
String
    let pathURL = baseURL.appendingPathComponent(bundleIdentifier)
    let persistentStoreDirectoryPath = pathURL.path
    if !FileManager.default.fileExists(atPath: persistentStoreDirectoryPath) {
        do {
            try FileManager.default.createDirectory(atPath: path, withIntermediateDirectories:
true, attributes: nil)
        }catch {
            //handle error
        }
    }
    let dirURL = URL.init(fileURLWithPath: persistentStoreDirectoryPath, isDirectory: true)
    do {
        try (dirURL as NSURL).setResourceValue((true), forKey: .isExcludedFromBackupKey)
    } catch {
        //handle error
    }
```

Существует множество инструментов для просмотра резервных копий iTunes для всех резервных копий данных, чтобы подтвердить, работает ли над ним подход или нет. iExplorer - хороший инструмент для поиска резервных копий iTunes.

Прочитайте Безопасность онлайн: https://riptutorial.com/ru/ios/topic/9999/безопасность

глава 117: Безопасность транспорта приложений (ATS)

параметры

параметр	подробности
`NSAppTransportSecurity`	Настройка ATS
`NSAllowsArbitraryLoads`	Установите `YES` чтобы отключить ATS повсюду. В iOS 10 и более поздних версиях и macOS 10.12 и более поздних версиях значение этого ключа игнорируется, если в файле Info.plist вашего приложения присутствует какой-либо из следующих ключей: NSAllowsArbitraryLoadsInMedia, NSAllowsArbitraryLoadsInWebContent, NSAllowsLocalNetworking
`NSAllowsArbitraryLoadsInMedia`	Установите `YES` для отключения ATS для носителя, загруженного с использованием API-интерфейсов из структуры AV Foundation. ***(iOS 10+, macOS 10.12+)***
`NSAllowsArbitraryLoadsInWebContent`	Установите `YES` для отключения ATS в веб-представлениях вашего приложения (`WKWebView` , `UIWebView` , `WebView`), не затрагивая ваши соединения NSURLSession. ***(iOS 10+, macOS 10.12+)***
`NSAllowsLocalNetworking`	Установите для `YES` для отключения для подключения к неквалифицированным доменам и к .local доменам. ***(iOS 10+, macOS 10.12+)***
`NSExceptionDomains`	Настройка исключений для определенных доменов
`NSIncludesSubdomains`	Установите `YES` для применения исключений ко всем субдоменам выбранного домена.

параметр	подробности
`NSRequiresCertificateTransparency`	Установите `YES` чтобы для сертификатов сервера (X.509) в домене были представлены действительные, подписанные отметки времени прозрачности (CT), из известных журналов CT. ***(iOS 10+, macOS 10.12+)***
`NSExceptionAllowsInsecureHTTPLoads`	Установите для `YES` чтобы разрешить HTTP в выбранном домене.
`NSExceptionRequiresForwardSecrecy`	По умолчанию `YES` ; Установите `NO` чтобы отключить Forward Secrecy и принять больше шифров.
`NSExceptionMinimumTLSVersion`	По умолчанию `TLSv1.2` ; Возможные значения: `TLSv1.0` , `TLSv1.1` , `TLSv1.2`
`NSThirdPartyExceptionAllowsInsecureHTTPLoads`	Подобно `NSExceptionAllowsInsecureHTTPLoads` , но для доменов, в которых у вас нет контроля над
`NSThirdPartyExceptionRequiresForwardSecrecy`	Подобно `NSExceptionRequiresForwardSecrecy` , но для доменов, на которые у вас нет контроля над
`NSThirdPartyExceptionMinimumTLSVersion`	Подобно `NSExceptionMinimumTLSVersion` , но для доменов, на которые у вас нет контроля над

замечания

Безопасность транспорта приложений - это функция безопасности в iOS и macOS. Это не позволяет приложениям устанавливать незащищенные соединения. По умолчанию приложения могут использовать только безопасные HTTPS-соединения.

Если приложение необходимо подключиться к серверу через HTTP, исключения должны быть определены в `Info.plist` . (см. примеры для получения дополнительной информации об этом)

Примечание. В 2017 году Apple будет применять ATS. Это означает, что вы больше не можете загружать приложения, имеющие исключения ATS, определенные в `Info.plist` . Если вы можете предоставить хорошие аргументы, почему вы должны использовать HTTP, вы можете связаться с Apple, и они могут позволить вам определять исключения.

(Источник: WWDC 2016 - Сессия 706)

Более подробную информацию о конфигурации безопасности для транспорта приложений можно найти в документации CocoaKeys .

Examples

Загрузка всего содержимого HTTP

Apple представила ATS с iOS 9 в качестве новой функции безопасности для улучшения конфиденциальности и безопасности между приложениями и веб-службами. ATS по умолчанию не выполняет все запросы без HTTPS. Хотя это может быть очень хорошо для производственных сред, это может быть неприятностью во время тестирования.

ATS настроен в файле `Info.plist` целевого объекта с `NSAppTransportSecurity` словаря `NSAppTransportSecurity` (`App Transport Security Settings` в редакторе Xcode Info.plist). Чтобы разрешить весь HTTP-контент, добавьте параметр `Allow Arbitrary Loads` boolean (`NSAllowsArbitraryLoads`) и установите для него значение `YES` . Это не рекомендуется для производственных приложений, и если требуется HTTP-контент, рекомендуется его выборочно активировать.

Выборочно загружать контент HTTP

Подобно включению всего HTTP-содержимого, вся конфигурация происходит в соответствии с `App Transport Security Settings` . Добавьте словарь `NSExceptionDomains Exception Domains` (`NSExceptionDomains`) в настройки ATS верхнего уровня.

Для каждого домена добавьте элемент словаря в Исключительные домены, где ключ является доменом, о котором идет речь. Установите `NSExceptionAllowsInsecureHTTPLoads` в `YES` чтобы отключить требование HTTPS для этого домена.

Конечным точкам требуется SSL

Представленные в iOS 9, все конечные точки должны соответствовать спецификации HTTPS.
Любые конечные точки, не использующие SSL, не будут с предупреждением в журнале консоли. В вашем приложении появится сообщение об ошибке подключения к Интернету.

Чтобы настроить исключения: поместите в файл Info.plist следующее:

1. Разрешить конкретный домен (testdomain.com) **только:**

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSExceptionDomains</key>
<dict>
```

```
    <key>testdomain.com</key>
    <dict>
        <key>NSIncludesSubdomains</key>
        <true/>
        <key>NSExceptionAllowsInsecureHTTPLoads</key>
        <true/>
    </dict>
</dict>
```

Ключ, разрешающий такое поведение, - `NSExceptionAllowsInsecureHTTPLoads` . В этом случае приложение будет разрешать только HTTP-соединение с указанным доменом (testdomain.com) и блокировать все другие HTTP-соединения.

Ключевой `NSIncludesSubdomains` указывает, что все и все **поддомены** упомянутого домена (testdomain.com) также должны быть разрешены.

2. Разрешить любой домен:

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

В этом случае приложение разрешает HTTP-подключение к **любому** домену. По состоянию на 1 января 2017 года использование этого флага приведет к тщательной проверке App Store, и разработчики приложений должны будут объяснить, почему они должны использовать это исключение в первую очередь. Возможные пояснения:

- Приложение, которое загружает зашифрованный мультимедийный контент, который не содержит персонализированной информации.
- Соединения с устройствами, которые не могут быть обновлены для использования защищенных соединений.
- Подключение к серверу, которому управляет другой объект, и не поддерживает безопасные соединения.

Прочитайте Безопасность транспорта приложений (ATS) онлайн:
https://riptutorial.com/ru/ios/topic/5435/безопасность-транспорта-приложений--ats-

глава 118: блок

Синтаксис

- В качестве переменной:

  returnType (^ blockName) (parameterTypes) = ^ returnType (parameters) {...};

- Как собственность:

  @property (nonatomic, copy) returnType (^ blockName) (parameterTypes);

- В качестве параметра метода:

  - (void) methodWithBlock: (returnType (^) (parameterTypes)) blockName;

- Как typedef:

  typedef returnType (^ TypeName) (parameterTypes);

  TypeName blockName = ^ returnType (parameters) {...};

Examples

UIView Animations

```
[UIView animateWithDuration:1.0
    animations:^{
        someView.alpha = 0;
        otherView.alpha = 1;
    }
    completion:^(BOOL finished) {
    [someView removeFromSuperview];
}];
```

Карат «^» определяет блок. Например, `^{ … }` является блоком. Более конкретно, это блок, который возвращает «void» и не принимает аргументов. Это эквивалентно методу, например: «- (void)», но нет никакого неотъемлемого имени, связанного с блоком кода.

Определить блок, который может принимать аргументы, работает очень точно. Чтобы предоставить аргумент блоку, вы определяете блок следующим образом: ^ *(BOOL someArg, NSString someStr) {...}* *. Когда вы используете вызовы API, поддерживающие блоки, вы будете писать блоки, которые выглядят схожим с этим, особенно для блоков анимации или блоков NSURLConnection, как показано в приведенном выше примере.

Пользовательский блок завершения для пользовательских методов

1- Определите свой собственный пользовательский блок

```
typedef void(^myCustomCompletion)(BOOL);
```

2- Создайте собственный метод, который принимает ваш настраиваемый блок завершения в качестве параметра.

```
-(void) customMethodName:(myCustomCompletion) compblock{
    //do stuff
    // check if completion block exist; if we do not check it will throw an exception
    if(complblock)
       compblock(YES);
  }
```

3- Как использовать блок в методе

```
[self customMethodName:^(BOOL finished) {
if(finished){
    NSLog(@"success");
}
}];
```

Изменить захваченную переменную

Блок будет захватывать переменные, которые появлялись в том же лексическом масштабе. Обычно эти переменные захватываются как значение «const»:

```
int val = 10;
void (^blk)(void) = ^{
    val = 20; // Error! val is a constant value and cannot be modified!
};
```

Чтобы изменить переменную, вам нужно использовать модификатор типа __block storage.

```
__block int val = 10;
void (^blk)(void) = ^{
    val = 20; // Correct! val now can be modified as an ordinary variable.
};
```

Прочитайте блок онлайн: https://riptutorial.com/ru/ios/topic/6888/блок

глава 119: Богатые уведомления

Вступление

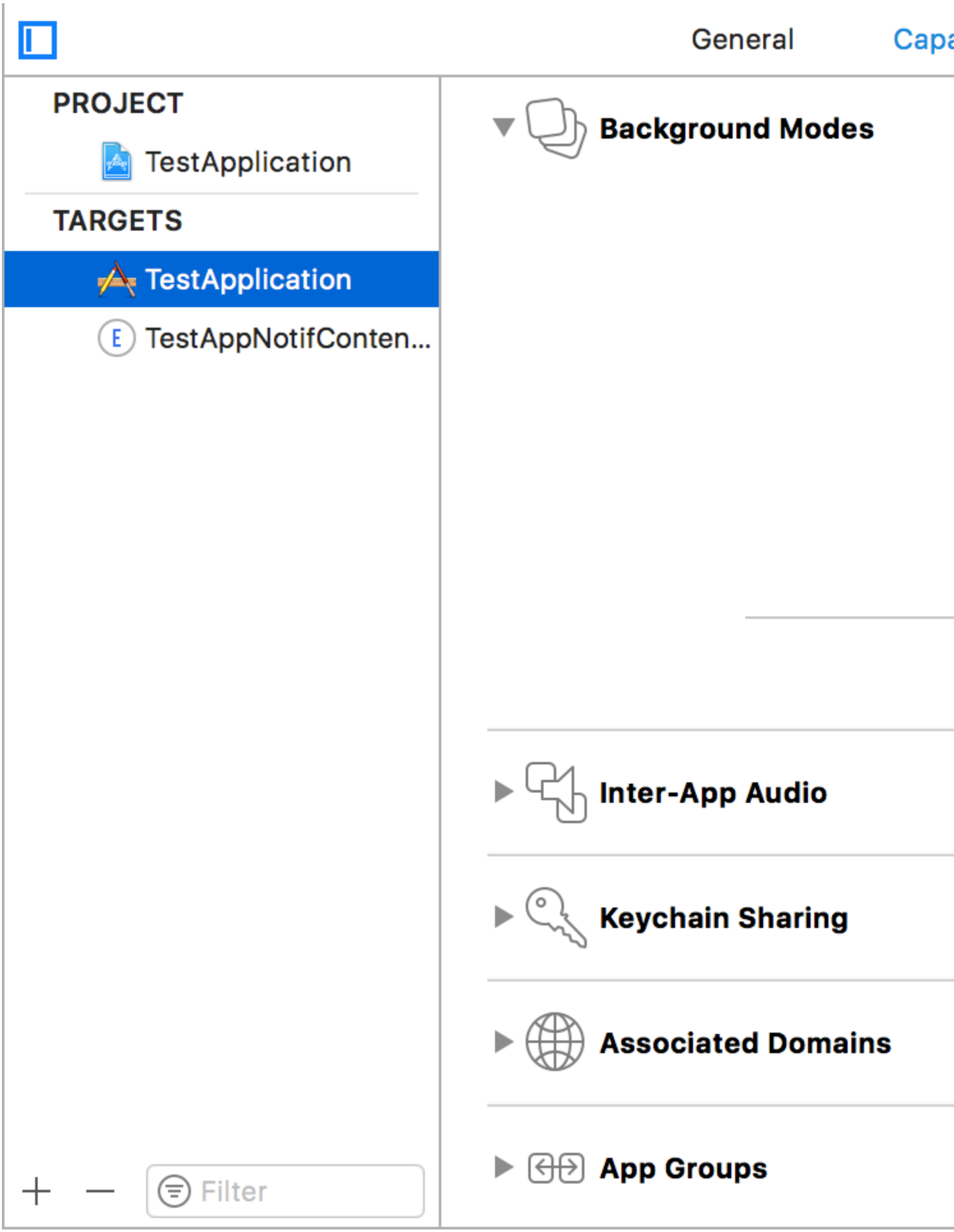
«Богатые уведомления» позволяют настраивать внешний вид локальных и удаленных уведомлений, когда они появляются на устройстве пользователя. Уведомление Rich в основном включает UNNotificationServiceExtension и UNNotificationContentExtension, т.е. отображение обычного уведомления расширенным образом

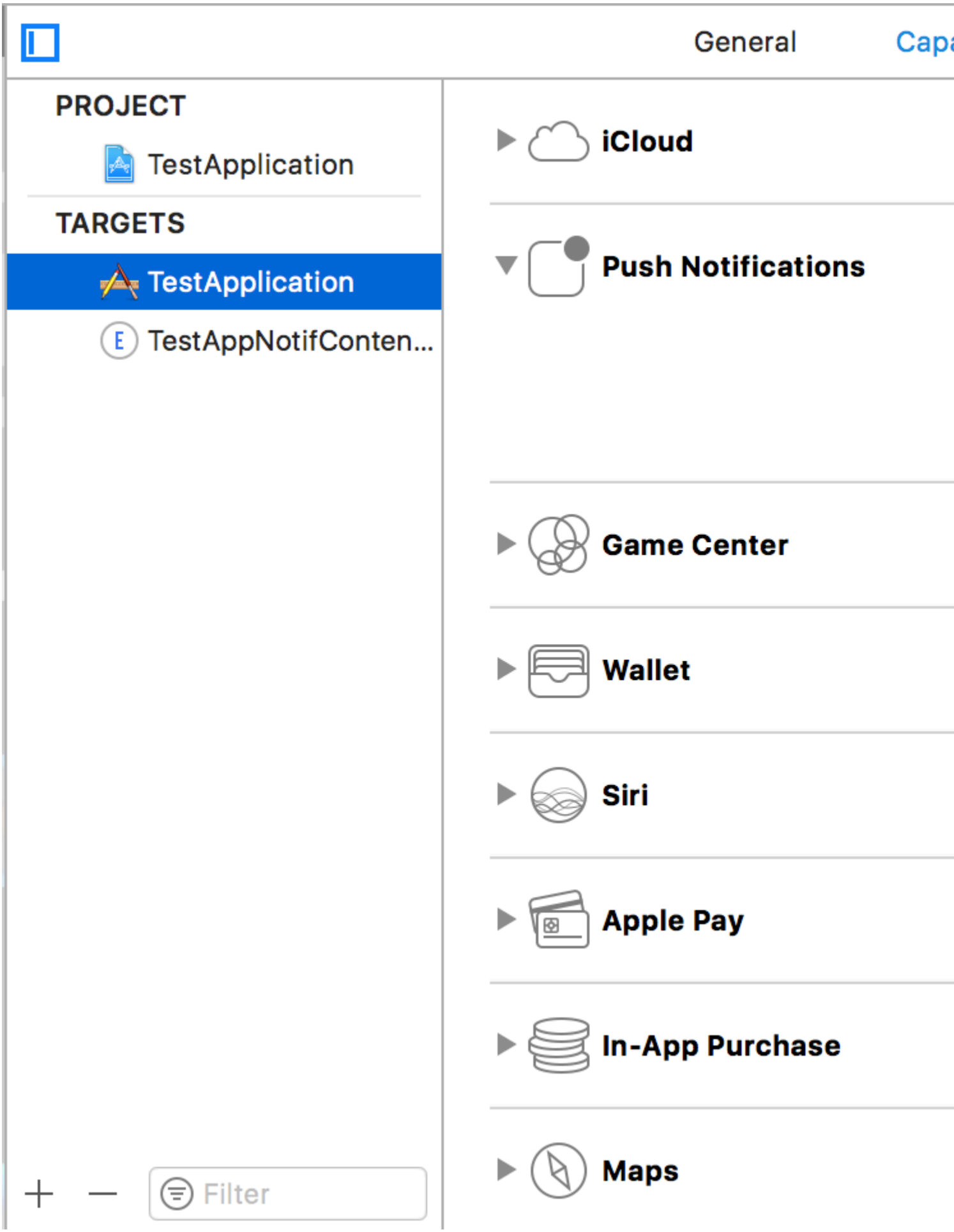
Examples

Создание простого UNNotificationContentExtension

Шаг 1

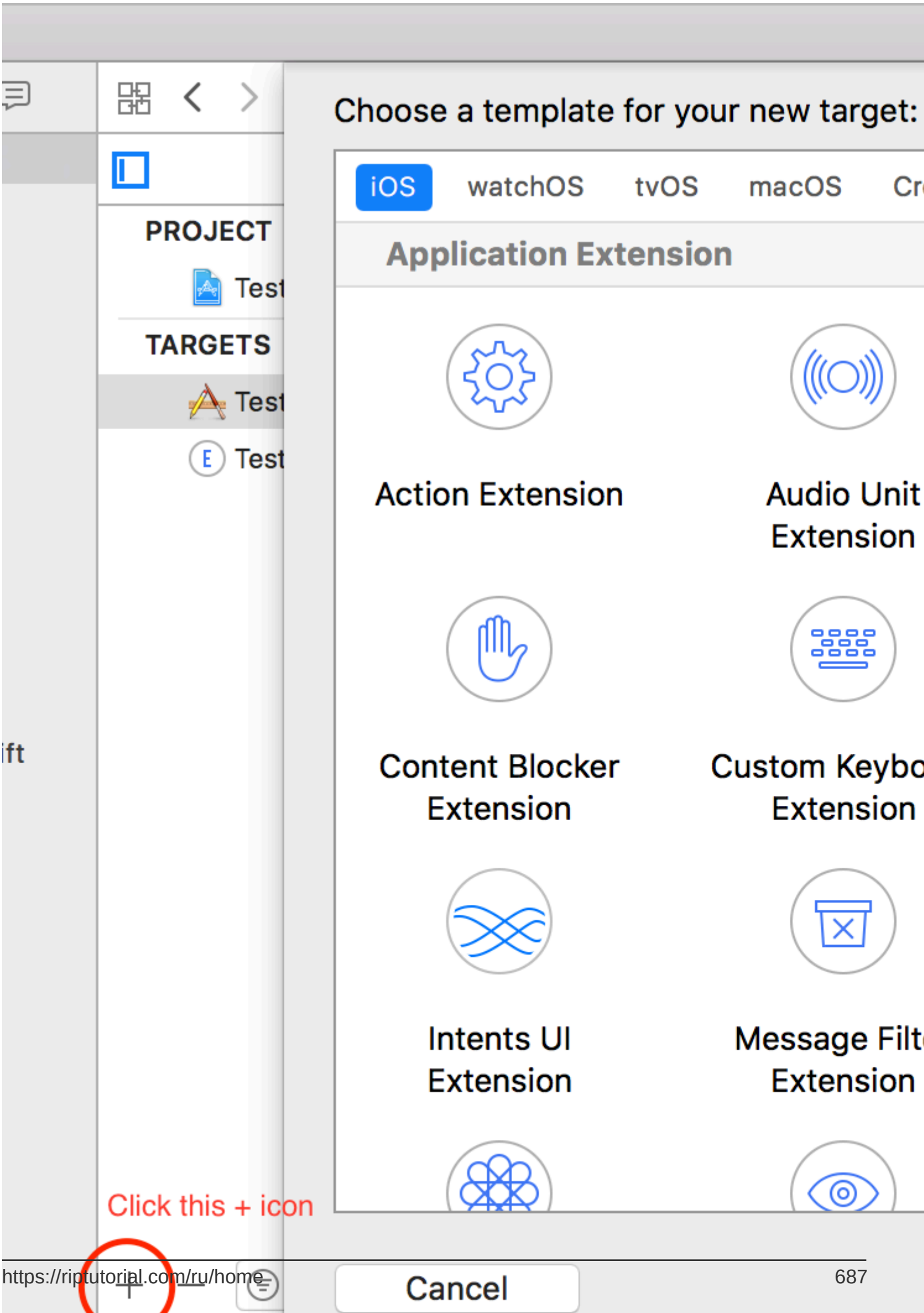
Обеспечение соответствия среды для уведомления. Удостоверьтесь, что вы включили **фоновый режим** и **Push-уведомление**


General
Capa
PROJECT
TestApplication
TARGETS
TestApplication
TestAppNotifConten...
Filter
Background Modes
Inter-App Audio
Keychain Sharing
Associated Domains
App Groups


General
Capa
PROJECT
TestApplication
TARGETS
TestApplication
TestAppNotifConten...
Filter
iCloud
Push Notifications
Game Center
Wallet
Siri
Apple Pay
In-App Purchase
Maps

Шаг 2. Создание UNNotificationContentExtension

Нажмите на значок « **+»** внизу, который создает целевой шаблон и выберите «Расширение содержимого уведомлений» -> далее -> создать имя для расширения содержимого -> завершить


Choose a template for your new target:
iOS
watchOS
tvOS
macOS
Application Extension
Action Extension
Audio Unit Extension
Content Blocker Extension
Custom Keybo Extension
Intents UI Extension
Message Filt Extension
PROJECT
Test
TARGETS
Test
Test
Click this + icon
Cancel

означает, как отображается содержимое уведомлений, они выполняются при длительном нажатии полученного уведомления

- UNNotificationExtensionOverridesDefaultTitle: мы можем предоставить пользовательский заголовок для нашего уведомления по умолчанию, оно отображает имя приложения `self.title = myTitle`
- UNNotificationDefaultContentHidden: этот логический элемент определяет, должен ли скрытый элемент уведомления по умолчанию скрываться или нет
- UNNotificationCategory: Категория создана в `UNUserNotificationCenter` в вашем приложении. Здесь это может быть строка или массив строк, поэтому каждая категория может предоставлять разные типы данных, из которых мы можем создавать разные пользовательские интерфейсы. Посылаемая нами полезная информация должна содержать название категории, чтобы отобразить это конкретное расширение
- UNNotificationExtensionInitialContentSizeRatio: размер исходного содержимого, т.е. при отображении ContentExtension в первый раз начальный размер по ширине устройства. здесь 1 обозначает, что высота будет равна ширине

Шаг 4: Создание `UNNotificationAction` и `UNNotificationCategory` в нашем приложении

В приложении AppDelegate.swift приложения `didFinishLaunchingWithOptions` добавьте

```
    let userNotificationAction:UNNotificationAction = UNNotificationAction.init(identifier:
"ID1", title: "வணக்கம்", options: .destructive)
    let userNotificationAction2:UNNotificationAction = UNNotificationAction.init(identifier:
"ID2", title: "Success", options: .destructive)

    let notifCategory:UNNotificationCategory = UNNotificationCategory.init(identifier:
"CATID1", actions: [userNotificationAction,userNotificationAction2], intentIdentifiers:
["ID1","ID2"] , options:.customDismissAction)

    UNUserNotificationCenter.current().delegate = self
    UNUserNotificationCenter.current().setNotificationCategories([notifCategory])
    UIApplication.shared.registerForRemoteNotifications()
```

Мы создали два `UNNotificationAction` с идентификаторами `ID1` и `ID2` и добавили эти действия в `UNNotificationCategory` с идентификатором `CATID1` (идентификатор категории в файле info.plist ContentExtension тот же, то, что мы создали здесь, должно использоваться в полезной нагрузке и в файле plist). Мы устанавливаем категорию в `UNUserNotificationCenter` нашего приложения, а в следующей строке мы регистрируемся для уведомления, которое вызывает функцию `didRegisterForRemoteNotificationsWithDeviceToken` где мы получаем токен устройства

Примечание. Не забывайте `import UserNotifications` в свой AppDelegate.swift и добавить `UNUserNotificationCenterDelegate`

Шаг 5: Образец полезной нагрузки для NotificationContent

```
 'aps': {
    'badge': 0,
    'alert': {
        'title': "Rich Notification",
        'body': "Body of RICH NOTIFICATION",
        },
    'sound' : "default",
    'category': "CATID1",
    'mutable-content':"1",
    },
'attachment': "2"
```

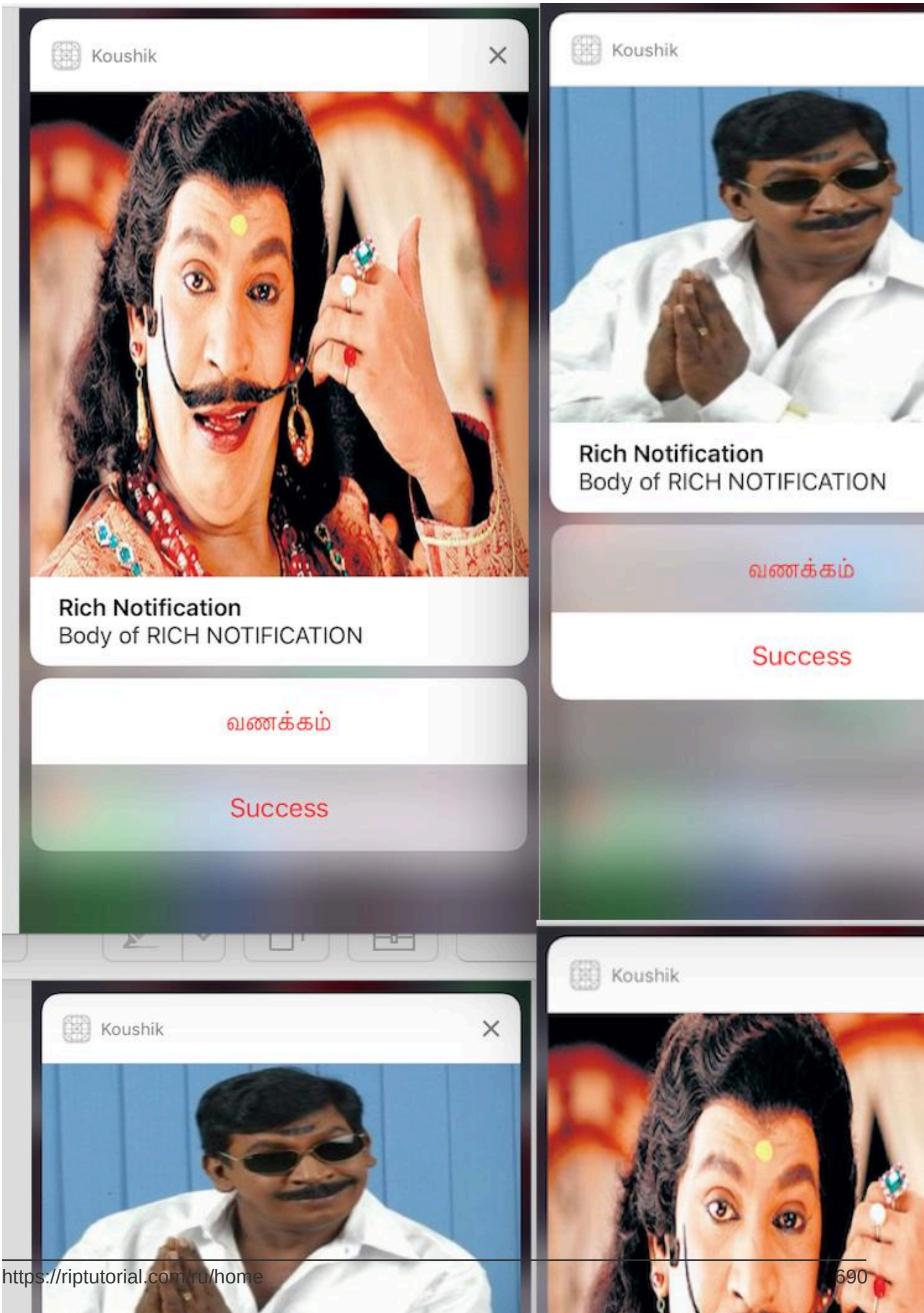
Шаг 6: Настройка ContentExtension

Соответствующие действия для категории автоматически отображаются во время выполнения действия уведомления. Позволяет нам увидеть код, как его выполнение

```
import UIKit
import UserNotifications
import UserNotificationsUI

class NotificationViewController: UIViewController, UNNotificationContentExtension {

@IBOutlet var imageView: UIImageView?
override func viewDidLoad() {
    super.viewDidLoad()
}

func didReceive(_ notification: UNNotification) {
     self.title = "Koushik"
    imageView?.backgroundColor = UIColor.clear
    imageView?.image = #imageLiteral(resourceName: "welcome.jpeg")
}

func didReceive(_ response: UNNotificationResponse, completionHandler completion: @escaping
(UNNotificationContentExtensionResponseOption) -> Void) {

    self.title = "Koushik"
    imageView?.image = UIImage.init(named: "Success.jpeg")

    if(response.actionIdentifier == "ID1")
    {
       imageView?.image = UIImage.init(named: "Success.jpeg")
    }
    else
    {
        imageView?.image = UIImage.init(named: "welcome.jpeg")
    }

    }
}
```

Шаг 7: Результат

После получения и долгого нажатия / нажатия на просмотр уведомления уведомление выглядит так:


Koushik
Rich Notification
Body of RICH NOTIFICATION
வணக்கம்
Success
Koushik
Rich Notification
Body of RICH NOTIFICATION
வணக்கம்
Success
Koushik
Koushik

`UNNotificationExtensionOverrideDefaultTitle` как ДА. На шаге 3 мы `UNNotificationExtensionDefaultContentHidden` как НЕТ, если его YES, тогда уведомление будет выглядеть как изображения 3 и 4.

Прочитайте Богатые уведомления онлайн: https://riptutorial.com/ru/ios/topic/10769/богатые-уведомления

глава 120: Брелок

Синтаксис

- kSecClassGenericPassword // Ключ значения, представляющий не-интернет-пароль
- kSecClassInternetPassword // Ключ значения, представляющий пароль в Интернете
- kSecClassCertificate // Ключ значения, представляющий сертификат
- kSecClassCertificate // Ключ значения, представляющий ключ
- kSecClassIdentity // Ключ значения, представляющий идентификатор, который является сертификатом плюс ключ

замечания

iOS хранит конфиденциальную информацию, такую как пароли, ключи шифрования, сертификаты и удостоверения в защищенной области хранения, называемой Keychain. Эта область хранения полностью управляется сопроцессором, называемым Secure Enclave, который встроен в процессор приложений. Поскольку Keychain изолирован в iOS, элементы keychain могут извлекаться только приложением, которое их там помещает в первую очередь.

В некоторых случаях вы должны включить совместное использование ключей в возможностях Xcode, чтобы избежать ошибок.

Чтобы взаимодействовать с цепочкой ключей, мы используем инфраструктуру ac, называемую Keychain Services. Для получения дополнительной информации см. Руководство по программированию устройств Keychain для Apple .

Поскольку Keychain Services находится ниже уровня `Foundation` , он ограничен использованием типов `CoreFoundation` . В результате большинство объектов внутренне представлены как `CFDictionary` s, где `CFString` s являются их ключами и различными типами `CoreFoundation` качестве их значений.

Хотя службы Keychain Services включены как часть структуры `Security` , импорт `Foundation` обычно является хорошим вариантом, поскольку он включает некоторые вспомогательные функции в бэкэнд.

Кроме того, если вы не хотите напрямую обращаться к службам Keychain, Apple предоставляет примерный проект Generic Keychain Swift, который предоставляет типы Swift, которые используют функции Keychain Services за кулисами.

Examples

Добавление пароля в брелок

Каждый элемент `CFDictionary` чаще всего представлен как `CFDictionary` . Вы можете, однако, просто использовать `NSDictionary` в Objective-C и воспользоваться мостом, или в Swift вы можете использовать `Dictionary` и явно бросать в `CFDictionary` .

Вы можете создать пароль со следующим словарем:

стриж

```
var dict = [String : AnyObject]()
```

Во-первых, вам нужна пара ключ / значение, которая позволяет Keychain знать, что это пароль. Обратите внимание, что поскольку наша клавиша dict - это `String` мы должны `CFString` любой `CFString` в `String` в Swift 3. CFString не может использоваться как ключ к Swift Dictionary, потому что это не Hashable.

стриж

```
dict[kSecClass as String] = kSecClassGenericPassword
```

Далее, наш пароль может иметь ряд атрибутов, чтобы описать его и помочь нам найти его позже. Вот список атрибутов для общих паролей .

стриж

```
// The password will only be accessible when the device is unlocked
dict[kSecAttrAccessible as String] = kSecAttrAccessibleWhenUnlocked
// Label may help you find it later
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
// Service name
dict[kSecAttrService as String] = "MyService" as CFString
```

Наконец, нам нужны наши фактические данные. Не задерживайте это слишком долго в памяти. Это должно быть `CFData` .

стриж

```
dict[kSecValueData as String] = "my_password!!".data(using: .utf8) as! CFData
```

Наконец, функция добавления ключей Keychain Services хочет знать, как она должна возвращать вновь созданный элемент keychain. Поскольку вы не должны долго держаться за данные в памяти, вот как вы можете вернуть атрибуты только:

стриж

```
dict[kSecReturnAttributes as String] = kCFBooleanTrue
```

Теперь мы построили наш предмет. Добавим:

стриж

```
var result: AnyObject?
let status = withUnsafeMutablePointer(to: &result) {
    SecItemAdd(dict as CFDictionary, UnsafeMutablePointer($0))
}
let newAttributes = result as! Dictionary<String, AnyObject>
```

Это помещает новые атрибуты dict внутри `result` . `SecItemAdd` принимает словарь, который мы построили, а также указатель на то, где мы хотим получить наш результат. Затем функция возвращает `OSStatus` указанием успеха или кода ошибки. Коды результатов описаны здесь .

Поиск пароля в брелках

Чтобы построить запрос, мы должны представить его как `CFDictionary` . Вы также можете использовать `NSDictionary` в Objective-C или `Dictionary` в Swift и отбрасывать в `CFDictionary` .

Нам нужен ключ класса:

стриж

```
var dict = [String : AnyObject]()
dict[kSecClass as String] = kSecClassGenericPassword
```

Затем мы можем указать атрибуты, чтобы сузить наш поиск:

стриж

```
// Label
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
```

```
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
// Service name
dict[kSecAttrService as String] = "MyService" as CFString
```

Мы также можем указать специальные клавиши-модификаторы поиска, описанные здесь .

Наконец, мы должны сказать, как мы хотим вернуть наши данные. Ниже мы `CFData` чтобы только частный пароль был возвращен как объект `CFData` :

стриж

```
dict[kSecReturnData as String] = kCFBooleanTrue
```

Теперь давайте посмотрим:

стриж

```
var queryResult: AnyObject?
let status = withUnsafeMutablePointer(to: &queryResult) {
    SecItemCopyMatching(dict as CFDictionary, UnsafeMutablePointer($0))
}
// Don't keep this in memory for long!!
let password = String(data: queryResult as! Data, encoding: .utf8)!
```

Здесь `SecItemCopyMatching` принимает словарь запросов и указатель на то, куда вы хотите, чтобы результат был выполнен. Он возвращает `OSStatus` с кодами результатов. Вот возможности.

Обновление пароля в цепочке ключей

Как обычно, сначала нам нужен `CFDictionary` для представления элемента, который мы хотим обновить. Это должно содержать все старые значения для элемента, включая старые личные данные. Затем требуется `CFDictionary` любых атрибутов или самих данных, которые вы хотели бы изменить.

Итак, сначала создадим ключ класса и список атрибутов. Эти атрибуты могут сузить наш поиск, но вы должны включать в себя любые атрибуты и старые значения, если вы их измените.

стриж

```
var dict = [String : AnyObject]()
```

```
dict[kSecClass as String] = kSecClassGenericPassword
// Label
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
```

Теперь мы должны добавить старые данные:

стриж

```
dict[kSecValueData as String] = "my_password!!".data(using: .utf8) as! CFData
```

Теперь давайте создадим те же атрибуты, но другой пароль:

стриж

```
var newDict = [String : AnyObject]()
newDict[kSecClass as String] = kSecClassGenericPassword
// Label
newDict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
newDict[kSecAttrAccount as String] = "My Name" as CFString
// New password
newDict[kSecValueData as String] = "new_password!!".data(using: .utf8) as! CFData
```

Теперь мы просто передаем его в службы Keychain:

стриж

```
let status = SecItemUpdate(dict as CFDictionary, newDict as CFDictionary)
```

`SecItemUpdate` возвращает код состояния. Результаты описаны здесь .

Удаление пароля из брелка

Нам нужно только одно, чтобы удалить элемент из Keychain: `CFDictionary` с атрибутами, описывающими элементы, которые нужно удалить. Любые элементы, соответствующие поисковому словарю, будут удалены навсегда, поэтому, если вы намерены удалить только один элемент, обязательно укажите его в своем запросе. Как всегда, мы можем использовать `NSDictionary` в Objective-C или в Swift, мы можем использовать `Dictionary` а затем бросать в `CFDictionary` .

Словарь запросов в этом контексте включает исключительно ключ класса, чтобы описать, что такое элемент, и атрибуты для описания информации об элементе. Включение

ограничений поиска, таких как `kSecMatchCaseInsensitive` , не допускается.

стриж

```
var dict = [String : AnyObject]()
dict[kSecClass as String] = kSecClassGenericPassword
// Label
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
```

И теперь мы можем просто удалить его:

стриж

```
let status = SecItemDelete(dict as CFDictionary)
```

`SecItemDelete` возвращает `OSStatus` . Коды результатов описаны здесь .

Keychain Добавить, обновить, удалить и найти операции с использованием одного файла.

Keychain.h

```
#import <Foundation/Foundation.h>
typedef void (^KeychainOperationBlock)(BOOL successfulOperation, NSData *data, OSStatus
status);

@interface Keychain : NSObject

-(id) initWithService:(NSString *) service_ withGroup:(NSString*)group_;

-(void)insertKey:(NSString *)key withData:(NSData *)data
withCompletion:(KeychainOperationBlock)completionBlock;
-(void)updateKey:(NSString*)key withData:(NSData*) data
withCompletion:(KeychainOperationBlock)completionBlock;
-(void)removeDataForKey:(NSString*)key
withCompletionBlock:(KeychainOperationBlock)completionBlock;
-(void)findDataForKey:(NSString*)key
withCompletionBlock:(KeychainOperationBlock)completionBlock;

@end
```

Keychain.m

```
#import "Keychain.h"
#import <Security/Security.h>

@implementation Keychain
```

```
{
    NSString * keychainService;
    NSString * keychainGroup;
}

-(id) initWithService:(NSString *)service withGroup:(NSString*)group
{
    self =[super init];
    if(self) {
        keychainService = [NSString stringWithString:service];
        if(group) {
            keychainGroup = [NSString stringWithString:group];
        }
    }

    return  self;
}

-(void)insertKey:(NSString *)key
        withData:(NSData *)data
  withCompletion:(KeychainOperationBlock)completionBlock
{
    NSMutableDictionary * dict =[self prepareDict:key];
    [dict setObject:data forKey:(__bridge id)kSecValueData];
    [dict setObject:keychainService forKey:(id)kSecAttrService];

    OSStatus status = SecItemAdd((__bridge CFDictionaryRef)dict, NULL);
    if(errSecSuccess != status) {
        DLog(@"Unable add item with key =%@ error:%d",key,(int)status);
        if (completionBlock) {
            completionBlock(errSecSuccess == status, nil, status);
        }
    }
    if (status == errSecDuplicateItem) {
        [self updateKey:key withData:data withCompletion:^(BOOL successfulOperation, NSData
*updateData, OSStatus updateStatus) {
            if (completionBlock) {
                completionBlock(successfulOperation, updateData, updateStatus);
            }
            DLog(@"Found duplication item -- updating key with data");
        }];
    }
}

-(void)findDataForKey:(NSString *)key
  withCompletionBlock:(KeychainOperationBlock)completionBlock
{
    NSMutableDictionary *dict = [self prepareDict:key];
    [dict setObject:(__bridge id)kSecMatchLimitOne forKey:(__bridge id)kSecMatchLimit];
    [dict setObject:keychainService forKey:(id)kSecAttrService];
    [dict setObject:(id)kCFBooleanTrue forKey:(__bridge id)kSecReturnData];
    CFTypeRef result = NULL;
    OSStatus status = SecItemCopyMatching((__bridge CFDictionaryRef)dict,&result);

    if( status != errSecSuccess) {
        DLog(@"Unable to fetch item for key %@ with error:%d",key,(int)status);
        if (completionBlock) {
            completionBlock(errSecSuccess == status, nil, status);
        }
    } else {
```

```
        if (completionBlock) {
            completionBlock(errSecSuccess == status, (__bridge NSData *)result, status);
        }
    }
}

-(void)updateKey:(NSString *)key
        withData:(NSData *)data
  withCompletion:(KeychainOperationBlock)completionBlock
{
    NSMutableDictionary * dictKey =[self prepareDict:key];

    NSMutableDictionary * dictUpdate =[[NSMutableDictionary alloc] init];
    [dictUpdate setObject:data forKey:(__bridge id)kSecValueData];
    [dictUpdate setObject:keychainService forKey:(id)kSecAttrService];
    OSStatus status = SecItemUpdate((__bridge CFDictionaryRef)dictKey, (__bridge
CFDictionaryRef)dictUpdate);
    if( status != errSecSuccess) {
        DLog(@"Unable to remove item for key %@ with error:%d",key,(int)status);
    }
    if (completionBlock) {
        completionBlock(errSecSuccess == status, nil, status);
    }
}

-(void)removeDataForKey:(NSString *)key
    withCompletionBlock:(KeychainOperationBlock)completionBlock {
    NSMutableDictionary *dict = [self prepareDict:key];
    OSStatus status = SecItemDelete((__bridge CFDictionaryRef)dict);
    if( status != errSecSuccess) {
        DLog(@"Unable to remove item for key %@ with error:%d",key,(int)status);
    }
    if (completionBlock) {
        completionBlock(errSecSuccess == status, nil, status);
    }
}

#pragma mark Internal methods

-(NSMutableDictionary*) prepareDict:(NSString *) key {

    NSMutableDictionary *dict = [[NSMutableDictionary alloc] init];
    [dict setObject:(__bridge id)kSecClassGenericPassword forKey:(__bridge id)kSecClass];

    NSData *encodedKey = [key dataUsingEncoding:NSUTF8StringEncoding];
    [dict setObject:encodedKey forKey:(__bridge id)kSecAttrGeneric];
    [dict setObject:encodedKey forKey:(__bridge id)kSecAttrAccount];
    [dict setObject:keychainService forKey:(__bridge id)kSecAttrService];
    [dict setObject:(__bridge id)kSecAttrAccessibleAlwaysThisDeviceOnly forKey:(__bridge
id)kSecAttrAccessible];

    //This is for sharing data across apps
    if(keychainGroup != nil) {
        [dict setObject:keychainGroup forKey:(__bridge id)kSecAttrAccessGroup];
    }

    return  dict;
}

@end
```

Контроль доступа Keychain (TouchID с возвратом пароля)

Keychain позволяет сохранять элементы со специальным атрибутом SecAccessControl, который позволит получить элемент из Keychain только после того, как пользователь будет аутентифицирован с помощью Touch ID (или кода доступа, если такой резерв разрешен). Приложение уведомляется только о том, была ли аутентификация успешной или нет, весь пользовательский интерфейс управляется iOS.

Сначала необходимо создать объект SecAccessControl:

стриж

```
let error: Unmanaged<CFError>?

guard let accessControl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly, .userPresence, &error) else {
    fatalError("Something went wrong")
}
```

Затем добавьте его в словарь с ключом kSecAttrAccessControl (который является взаимоисключающим с ключом kSecAttrAccessible, который вы использовали в других примерах):

стриж

```
var dictionary = [String : Any]()

dictionary[kSecClass as String] = kSecClassGenericPassword
dictionary[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
dictionary[kSecAttrAccount as String] = "My Name" as CFString
dictionary[kSecValueData as String] = "new_password!!".data(using: .utf8) as! CFData
dictionary[kSecAttrAccessControl as String] = accessControl
```

И сохраните его, как вы это делали раньше:

стриж

```
let lastResultCode = SecItemAdd(query as CFDictionary, nil)
```

Чтобы получить доступ к сохраненным данным, просто запросите Keychain для ключа. Служба Keychain Services представит пользователю диалог аутентификации и вернет данные или нуль в зависимости от того, был ли предоставлен подходящий отпечаток или совпадение кода доступа.

Необязательно, строка подсказки может быть указана:

стриж

```
var query = [String: Any]()

query[kSecClass as String] = kSecClassGenericPassword
query[kSecReturnData as String] = kCFBooleanTrue
query[kSecAttrAccount as String] = "My Name" as CFString
query[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
query[kSecUseOperationPrompt as String] = "Please put your fingers on that button" as CFString

var queryResult: AnyObject?
let status = withUnsafeMutablePointer(to: &queryResult) {
    SecItemCopyMatching(query as CFDictionary, UnsafeMutablePointer($0))
}
```

Обратите внимание, что `status` будет `err` если пользователь отклонил, отменил или отказал авторизацию.

стриж

```
if status == noErr {
    let password = String(data: queryResult as! Data, encoding: .utf8)!
    print("Password: \(password)")
} else {
    print("Authorization not passed")
}
```

Прочитайте Брелок онлайн: https://riptutorial.com/ru/ios/topic/6839/брелок

глава 121: Быстрая и объективная совместимость

Examples

Использование классов Objective-C в Swift

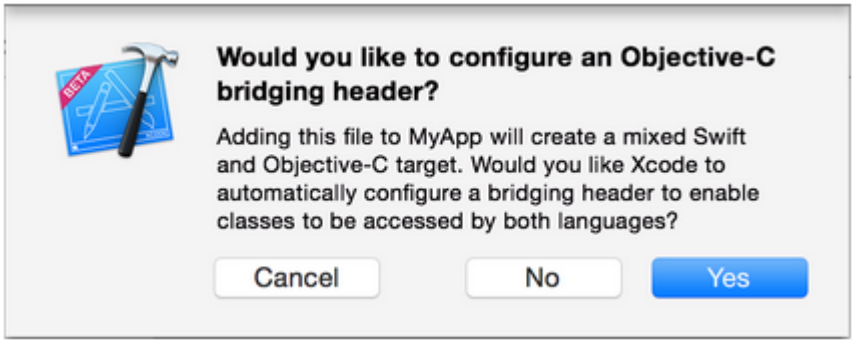
> Если у вас есть существующий класс, который вы хотите использовать, выполните **Шаг 2,** а затем перейдите к **Шагу 5** . (В некоторых случаях мне пришлось добавить явный `#import <Foundation/Foundation.h` в старый файл ObjC)

Шаг 1: Добавить реализацию Objective-C - .m

Добавьте файл `.m` в свой класс и назовите его `CustomObject.m`

Шаг 2: Добавить заголовок перемычки

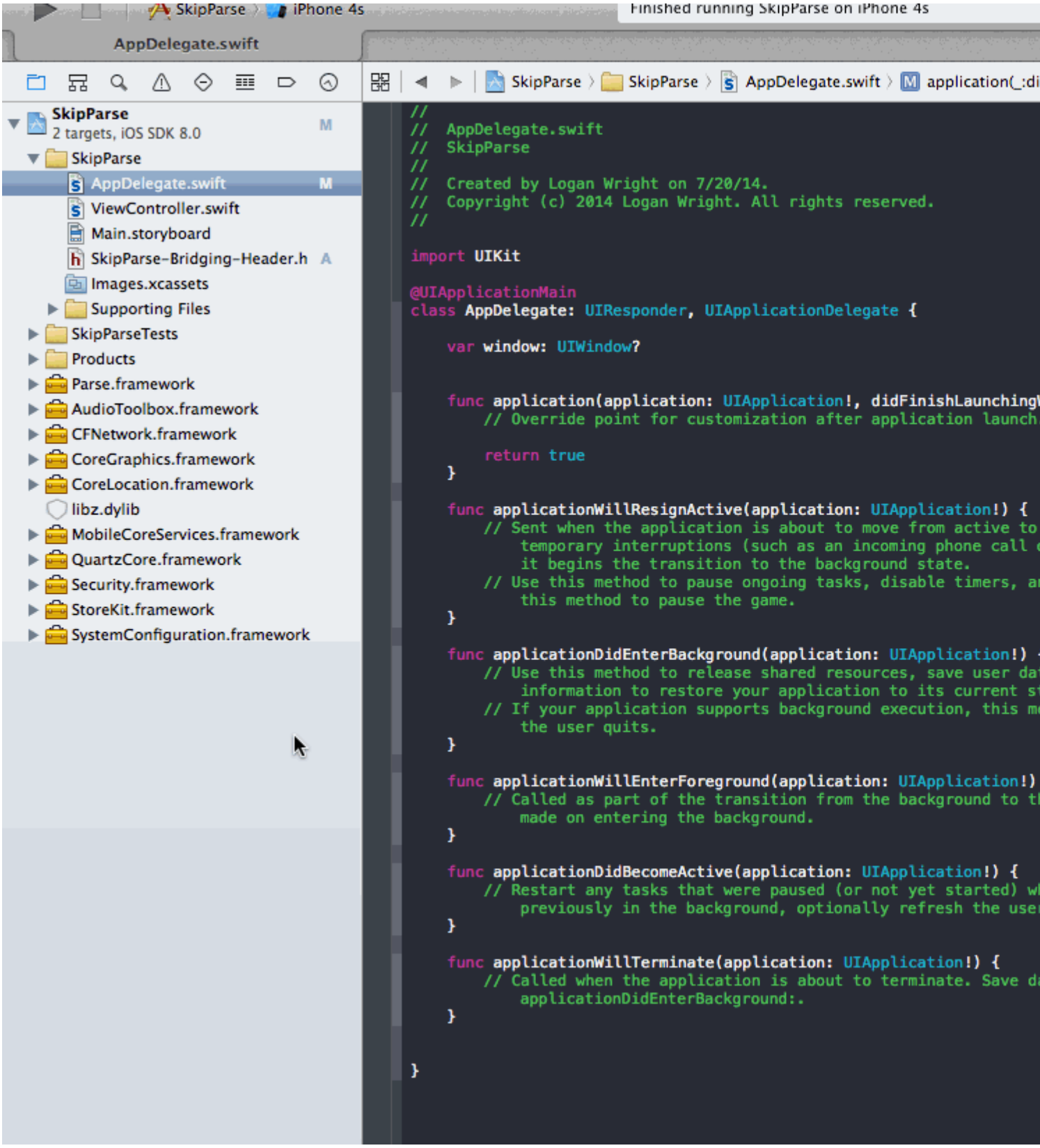
При добавлении вашего `.m` файла вы, вероятно, будете поражены подсказкой, которая выглядит так:



Нажмите **ДА** !

Если вы не увидели приглашение или случайно удалили заголовок моста, добавьте в проект новый файл `.h` и назовите его `<#YourProjectName#>-Bridging-Header.h`

В некоторых ситуациях, особенно при работе с инфраструктурами ObjC, вы не добавляете класс Objective-C явно, и Xcode не может найти компоновщик. В этом случае создайте свой `.h` файл с именем, указанным выше, затем убедитесь, что вы связали его путь в настройках проекта вашей цели следующим образом:



Заметка

Лучше всего связать свой проект с помощью макроса `$(SRCROOT)` чтобы при перемещении вашего проекта или работе с ним с помощью удаленного репо, он все равно будет работать. `$(SRCROOT)` можно рассматривать как каталог, содержащий ваш файл .xcodeproj. Это может выглядеть так:

`$(SRCROOT)/Folder/Folder/<#YourProjectName#>-Bridging-Header.h`

Шаг 3: Добавить заголовок Objective-C - .h

Добавьте еще один файл `.h` и назовите его `CustomObject.h`

Шаг 4: Создайте свой класс Objective-C

В `CustomObject.h`

```
#import <Foundation/Foundation.h>

@interface CustomObject : NSObject

@property (strong, nonatomic) id someProperty;

- (void) someMethod;

@end
```

В `CustomObject.m`

```
#import "CustomObject.h"

@implementation CustomObject

- (void) someMethod {
    NSLog(@"SomeMethod Ran");
}

@end
```

Шаг 5: добавьте класс в заголовок-заголовок

В `YourProject-Bridging-Header.h` :

```
#import "CustomObject.h"
```

Шаг 6: Используйте свой объект

В `SomeSwiftFile.swift` :

```
var instanceOfCustomObject: CustomObject = CustomObject()
instanceOfCustomObject.someProperty = "Hello World"
println(instanceOfCustomObject.someProperty)
instanceOfCustomObject.someMethod()
```

Нет необходимости явно импортировать, для чего нужен заголовок для моста.

Использование классов Swift в Objective-C

Шаг 1. Создание нового класса Swift.

Добавьте файл `.swift` в свой проект и назовите его `MySwiftObject.swift`

В `MySwiftObject.swift` :

```
import Foundation

class MySwiftObject : NSObject {

    var someProperty: AnyObject = "Some Initializer Val"

    init() {}

    func someFunction(someArg:AnyObject) -> String {
        var returnVal = "You sent me \(someArg)"
        return returnVal
    }

}
```

Шаг 2: импорт файлов Swift в класс ObjC

В `SomeRandomClass.m` :

```
#import "<#YourProjectName#>-Swift.h"
```

Файл: `<#YourProjectName#>-Swift.h` должен быть создан автоматически в вашем проекте, даже если вы его не видите.

Шаг 3: Используйте свой класс

```
MySwiftObject * myOb = [MySwiftObject new];
NSLog(@"MyOb.someProperty: %@", myOb.someProperty);
myOb.someProperty = @"Hello World";
NSLog(@"MyOb.someProperty: %@", myOb.someProperty);
NSString * retString = [myOb someFunction:@"Arg"];
NSLog(@"RetString: %@", retString);
```

Замечания:

1. CodeCompletion не вел себя так точно, как хотелось бы. В моей системе быстрый запуск w / cmd + r помог Swift найти код Objc и наоборот.

2. Если вы добавите файл `.swift` в более старый проект и получите ошибку: `dyld: Library not loaded: @rpath/libswift_stdlib_core.dylib` , попробуйте полностью перезапустить Xcode.

3. Хотя изначально было возможно использовать чистые классы Swift в Objective-C с помощью префикса `@objc` , после Swift 2.0 это невозможно. См. Историю изменений для

оригинального объяснения. Если эта функциональность будет повторно включена в будущих версиях Swift, ответ будет соответствующим образом обновлен.

Прочитайте Быстрая и объективная совместимость онлайн:
https://riptutorial.com/ru/ios/topic/1497/быстрая-и-объективная-совместимость

глава 122: Время выполнения в Objective-C

Examples

Использование связанных объектов

Связанные объекты полезны, когда вы хотите добавить функциональность к существующим классам, которые требуют состояния удержания.

Например, добавление индикатора активности в каждый UIView:

Реализация Objective-C

```
#import <objc/runtime.h>

static char ActivityIndicatorKey;

@implementation UIView (ActivityIndicator)

- (UIActivityIndicatorView *)activityIndicator {
    return (UIActivityIndicatorView *)objc_getAssociatedObject(self, &ActivityIndicatorKey);
}

- (void)setActivityIndicator: (UIActivityIndicatorView *)activityIndicator {
    objc_setAssociatedObject(self, &ActivityIndicatorKey, activityIndicator,
OBJC_ASSOCIATION_RETAIN_NONATOMIC);
}

- (void)showActivityIndicator {
    UIActivityIndicatorView *activityIndicator = [[UIActivityIndicatorView alloc]
initWithActivityIndicatorStyle: UIActivityIndicatorViewStyleGray];

    [self setActivityIndicator:activityIndicator];

    activityIndicator.center = self.center;
    activityIndicator.autoresizingMask = UIViewAutoresizingFlexibleTopMargin |
UIViewAutoresizingFlexibleLeftMargin | UIViewAutoresizingFlexibleRightMargin |
UIViewAutoresizingFlexibleBottomMargin;

    [activityIndicator startAnimating];

    [self addSubview: activityIndicator];
}

- (void)hideActivityIndicator {
    UIActivityIndicatorView * activityIndicator = [self activityIndicator];

    if (activityIndicator != nil) {
        [[self activityIndicator] removeFromSuperview];
    }
}

@end
```

Вы также можете получить доступ к объекту Objective-C через Swift:

Быстрый код

```
extension UIView {
    private struct AssociatedKeys {
        static var activityIndicator = "UIView.ActivityIndicatorView"
    }

    private var activityIndicatorView: UIActivityIndicatorView? {
        get {
            return objc_getAssociatedObject(self, &AssociatedKeys.activityIndicator) as?
UIActivityIndicatorView
        }
        set (activityIndicatorView) {
            objc_setAssociatedObject(self, &AssociatedKeys.activityIndicator,
activityIndicatorView, .OBJC_ASSOCIATION_RETAIN_NONATOMIC)
        }
    }

     func showActivityIndicator() {
        activityIndicatorView = UIActivityIndicatorView(activityIndicatorStyle: .gray)
        activityIndicatorView.center = center
        activityIndicatorView.autoresizingMask = [.flexibleLeftMargin, .flexibleRightMargin,
.flexibleTopMargin, .flexibleBottomMargin]

        activityIndicatorView.startAnimating()

        addSubview(activityIndicatorView)
    }

    func hideActivityIndicator() {
        activityIndicatorView.removeFromSuperview()
    }
}
```

Прочитайте Время выполнения в Objective-C онлайн: https://riptutorial.com/ru/ios/topic/10120/время-выполнения-в-objective-c

глава 123: Всплывающие уведомления

Синтаксис

- UIUserNotificationSettings.types: UIUserNotificationType // Битовая маска типов уведомлений, которые разрешено использовать вашему приложению
- UIUserNotificationSettings.categories: Установить // Зарегистрированные группы действий приложения

параметры

параметр	Описание
USERINFO	Словарь, содержащий информацию о удаленном уведомлении, потенциально включающий номер значка для значка приложения, звуковой сигнал оповещения, предупреждающее сообщение, идентификатор уведомления и пользовательские данные.

Examples

Регистрация устройства для Push-уведомлений

Чтобы зарегистрировать устройство для push-уведомлений, добавьте следующий код в файл `didFinishLaunchingWithOptions` методе `didFinishLaunchingWithOptions` :

стриж

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    // Override point for customization after application launch.
    if UIDevice.currentDevice().systemVersion.compare(v, options: .NumericSearch) ==
NSOrderedAscending {
        // Register for Push Notitications, if running iOS < 8
        if application.respondsToSelector("registerUserNotificationSettings:") {
            let types:UIUserNotificationType = (.Alert | .Badge | .Sound)
            let settings:UIUserNotificationSettings = UIUserNotificationSettings(forTypes:
types, categories: nil)

            application.registerUserNotificationSettings(settings)
            application.registerForRemoteNotifications()
        } else {
            // Register for Push Notifications before iOS 8
            application.registerForRemoteNotificationTypes(.Alert | .Badge | .Sound)
        }
    } else {
```

```
        var center = UNUserNotificationCenter.currentNotificationCenter()
        center.delegate = self
        center.requestAuthorizationWithOptions((UNAuthorizationOptionSound |
UNAuthorizationOptionAlert | UNAuthorizationOptionBadge)) {(granted: Bool, error: NSError) ->
Void in
            if !error {
                UIApplication.sharedApplication().registerForRemoteNotifications()
                // required to get the app to do anything at all about push notifications
                print("Push registration success.")
            } else {
                print("Push registration FAILED")
                print("ERROR: \(error.localizedFailureReason!) -
\(error.localizedDescription)")
                print("SUGGESTIONS: \(error.localizedRecoveryOptions) -
\(error.localizedRecoverySuggestion!)")
            })
    }

    return true
}
```

Objective-C

```
#define SYSTEM_VERSION_LESS_THAN(v) ([[[UIDevice currentDevice] systemVersion] compare:v
options:NSNumericSearch] == NSOrderedAscending)

if( SYSTEM_VERSION_LESS_THAN( @"10.0" ) )
{
        if ([application respondsToSelector:@selector(isRegisteredForRemoteNotifications)])
    {
        // iOS 8 Notifications
        [application registerUserNotificationSettings:[UIUserNotificationSettings
settingsForTypes:(UIUserNotificationTypeSound | UIUserNotificationTypeAlert |
UIUserNotificationTypeBadge) categories:nil]];

        [application registerForRemoteNotifications];
    }
    else
    {
        // iOS < 8 Notifications
        [application registerForRemoteNotificationTypes:
         (UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeAlert |
UIRemoteNotificationTypeSound)];
    }

}
else
{
    UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];
    center.delegate = self;
    [center requestAuthorizationWithOptions:(UNAuthorizationOptionSound |
UNAuthorizationOptionAlert | UNAuthorizationOptionBadge) completionHandler:^(BOOL granted,
NSError * _Nullable error)
     {
         if( !error )
         {
             [[UIApplication sharedApplication] registerForRemoteNotifications];  // required
to get the app to do anything at all about push notifications
             NSLog( @"Push registration success." );
```

```
         }
         else
         {
             NSLog( @"Push registration FAILED" );
             NSLog( @"ERROR: %@ - %@", error.localizedFailureReason,
error.localizedDescription );
             NSLog( @"SUGGESTIONS: %@ - %@", error.localizedRecoveryOptions,
error.localizedRecoverySuggestion );
         }
     }];
}


//to check if your App lunch from Push notification
//----------------------------------------------------
    //Handel Push notification
    if (launchOptions != nil)
    {
        // Here app will open from pushnotification
        //RemoteNotification
        NSDictionary* dictionary1 = [launchOptions
objectForKey:UIApplicationLaunchOptionsRemoteNotificationKey];
        //LocalNotification
        NSDictionary* dictionary2 = [launchOptions
objectForKey:UIApplicationLaunchOptionsLocalNotificationKey];
        if (dictionary1 != nil)
        {
            //RemoteNotification Payload
            NSLog(@"Launched from push notification: %@", dictionary1);
            //here handle your push notification
        }
        if (dictionary2 != nil)
        {
            NSLog(@"Launched from dictionary2dictionary2dictionary2 notification: %@",
dictionary2);
            double delayInSeconds = 7;
            dispatch_time_t popTime = dispatch_time(DISPATCH_TIME_NOW,
(int64_t)(delayInSeconds * NSEC_PER_SEC));
            dispatch_after(popTime, dispatch_get_main_queue(), ^(void){
                // [self addMessageFromRemoteNotification:dictionary2 updateUI:NO];
            });
        }

     }
     else
         {}
    //----------------------------------------------
```

Вышеприведенный код попытается связаться с сервером APN для получения токена устройства (предварительные запросы - это то, что APN включены в профиле инициализации iOS).

Как только он устанавливает надежное соединение с сервером APN, сервер предоставляет вам токен устройства.

После добавления вышеприведенного кода добавьте эти методы в класс `AppDelegate` :

стриж

```
func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    print("DEVICE TOKEN = \(deviceToken)")
}

func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    print(error)
}
```

Objective-C

```
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
NSString * deviceTokenString = [[[[deviceToken description]
                    stringByReplacingOccurrencesOfString: @"<" withString: @""]
                   stringByReplacingOccurrencesOfString: @">" withString: @""]
                  stringByReplacingOccurrencesOfString: @" " withString: @""];

NSLog(@"The generated device token string is : %@",deviceTokenString);
}

- (void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error
{
    NSLog(@"Failed to get token, error: %@", error.description);
}
```

Вышеупомянутые методы вызывают в соответствии с сценарием успешности регистрации или сбоя.

Вызов сценария успеха:

стриж

```
func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    print("DEVICE TOKEN = \(deviceToken)")
}
```

В Swift3:

```
@objc(userNotificationCenter:willPresentNotification:withCompletionHandler:) @available(iOS
10.0, *)
func userNotificationCenter(_ center: UNUserNotificationCenter, willPresent notification:
UNNotification, withCompletionHandler completionHandler: @escaping
(UNNotificationPresentationOptions) -> Void)
{
```

```
    //To show notifications in foreground.
   print("Userinfo2 \(notification.request.content.userInfo)")
}
```

Objective-C

```
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
if(application.applicationState == UIApplicationStateInactive) {
    NSLog(@"Inactive - the user has tapped in the notification when app was closed or in
background");
    //do some tasks
    [self handelPushNotification:userInfo];
}
else if (application.applicationState == UIApplicationStateBackground) {
    NSLog(@"application Background - notification has arrived when app was in background");
    [self handelPushNotification:userInfo];
}
else {
    NSLog(@"application Active - notication has arrived while app was opened");
    //Show an in-app banner
    //do tasks
}
}
```

Вызов сценария сбоя:

стриж

```
func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    print(error)
}
```

Objective-C

```
- (void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error
```

Заметка

Если ни один из вышеперечисленных методов не вызван, ваше устройство не может создать надежное соединение с сервером APN, что может быть связано с проблемами доступа в Интернет.

Проверка того, что ваше приложение уже зарегистрировано для Push

стриж

```
let isPushEnabled = UIApplication.sharedApplication().isRegisteredForRemoteNotifications()
```

Регистрация для (не интерактивного) Push-уведомления

Логику регистрации для push-уведомления рекомендуется добавлять в `AppDelegate.swift` поскольку функции обратного вызова (успех, сбой) будут называться их. Для регистрации выполните следующие действия:

```
let application = UIApplication.sharedApplication()
let settings = UIUserNotificationSettings(forTypes: [.Alert, .Badge, .Sound], categories: nil)
application.registerUserNotificationSettings(settings)
```

Затем будет вызываться функция обратного вызова `didRegisterUserNotificationSettings` , и в этом случае вы просто запускаете регистр следующим образом:

```
func application(application: UIApplication, didRegisterUserNotificationSettings
notificationSettings: UIUserNotificationSettings) {
    application.registerForRemoteNotifications()
}
```

И в этом случае будет отображаться системное предупреждение с запросом на получение уведомления о push-уведомлении. Будут вызываться одна из следующих функций обратного вызова:

```
func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    let tokenChars = UnsafePointer<CChar>(deviceToken.bytes)
    var tokenString = ""

    for i in 0..<deviceToken.length {
        tokenString += String(format: "%02.2hhx", arguments: [tokenChars[i]])
    }

    print("Push token: \(tokenString)")
}

func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    print("didFailToRegisterForRemoteNotificationsWithError: \(error)")

}
```

В очень редких случаях не вызываются функции отказа или неудачи. Это происходит, когда у вас проблемы с подключением к Интернету или недоступна песочница APNS. Система выполняет вызов API для APNS, чтобы сделать некоторую проверку, так как это

не приведет к тому, что ни одна из двух функций callbacks не будет вызвана. Посетите статус системы Apple, чтобы убедиться, что все в порядке.

Обращение с Push Push

Как только пользователь нажимает на push-уведомление, будет вызвана следующая функция обратного вызова. Вы можете разобрать JSON, чтобы получить какую-либо конкретную информацию, отправленную из бэкэнд, которая поможет вам в глубокой привязке:

стриж

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject
: AnyObject]) {
    print("Received notification: \(userInfo)")
}
```

Цель C

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification: (NSDictionary
*)userInfo
{
    NSLog(@"Received notification: %@", userInfo);
}
```

iOS 10

```
#define SYSTEM_VERSION_GREATER_THAN_OR_EQUAL_TO(v)  ([[[UIDevice currentDevice] systemVersion]
compare:v options:NSNumericSearch] != NSOrderedAscending)

-(void) application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary
*)userInfo fetchCompletionHandler:(void
(^)(UIBackgroundFetchResult))completionHandler
{
    // iOS 10 will handle notifications through other methods
    NSLog(@"Received notification: %@", userInfo);


    if( SYSTEM_VERSION_GREATER_THAN_OR_EQUAL_TO( @"10.0" ) )
    {
        NSLog( @"iOS version >= 10. Let NotificationCenter handle this one." );
        // set a member variable to tell the new delegate that this is background
        return;
    }
    NSLog( @"HANDLE PUSH, didReceiveRemoteNotification: %@", userInfo );

    // custom code to handle notification content

    if( [UIApplication sharedApplication].applicationState == UIApplicationStateInactive )
   {
       NSLog( @"INACTIVE" );
       completionHandler( UIBackgroundFetchResultNewData );
   }
   else if( [UIApplication sharedApplication].applicationState == UIApplicationStateBackground
```

```
)
    {
        NSLog( @"BACKGROUND" );
        completionHandler( UIBackgroundFetchResultNewData );
    }
    else
    {
        NSLog( @"FOREGROUND" );
        completionHandler( UIBackgroundFetchResultNewData );
    }
}

- (void)userNotificationCenter:(UNUserNotificationCenter *)center
   willPresentNotification:(UNNotification *)notification
     withCompletionHandler:(void (^)(UNNotificationPresentationOptions
options))completionHandler
{
    NSLog( @"Handle push from foreground" );
   // custom code to handle push while app is in the foreground
    NSLog(@"%@", notification.request.content.userInfo);
}

- (void)userNotificationCenter:(UNUserNotificationCenter *)center
didReceiveNotificationResponse:(UNNotificationResponse *)response
     withCompletionHandler:(void (^)())completionHandler
{

    NSLog( @"Handle push from background or closed" );
    // if you set a member variable in didReceiveRemoteNotification, you  will know if this is
from closed or background
    NSLog(@"%@", response.notification.request.content.userInfo);
}
```

Регистрация идентификатора приложения для использования с Push-уведомлениями

Вещи, которые нужно

- Платное членство в программе Apple Developer Program
- Действительный идентификатор и идентификатор приложения для вашего приложения (например, com.example.MyApp), который не используется нигде
- Доступ к developer.apple.com и членскому центру
- Устройство iOS для тестирования (поскольку Push Notifications не работают на Simulator)

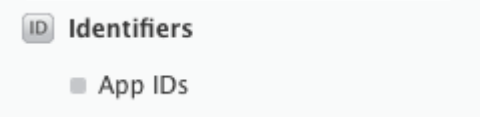
Включение доступа APN для идентификатора приложения в Apple

Developer Center

1 Войти в developer.apple.com Центр участников (ссылка на главной странице)

2- Перейдите в раздел «Сертификаты»

3- Выберите «Идентификатор приложения» с левой панели



4- Нажмите «+» вверху справа

5- Добавлен флажок Добавить идентификатор приложения с Push Notifications

6- Нажмите на созданный идентификатор приложения и выберите «Изменить».

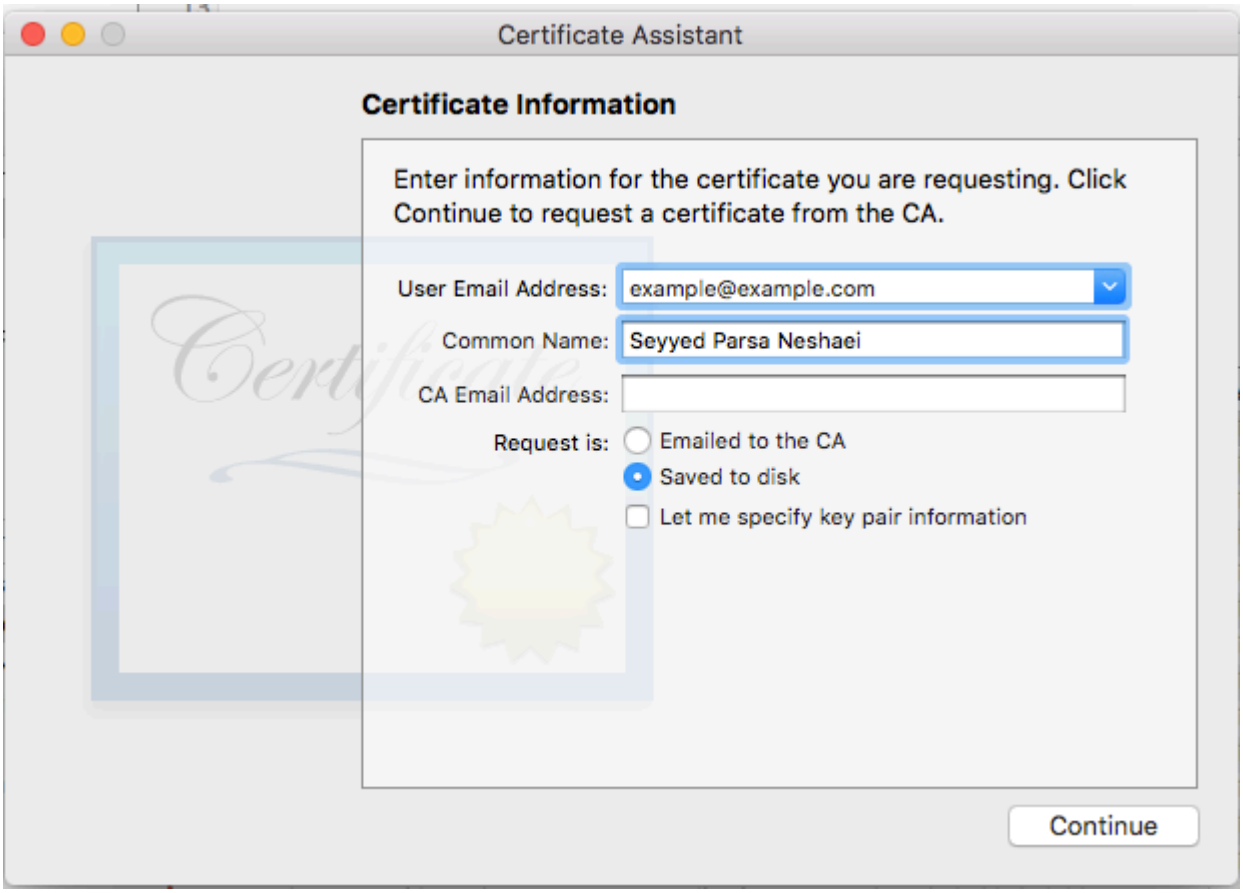
7 - Нажмите «Настроить» в панели «Push Push»

8 - Откройте приложение Access Keychain Access на вашем Mac

9- В меню «Доступ к цепочке ключей» выберите «Ассистент сертификата» -> «Запросить сертификат» из центра сертификации

10- Введите свою почту в первом текстовом поле

11- Введите свое имя во втором текстовом поле



12 - Оставьте адрес электронной почты CA пустым

13- Выберите «Сохранено на диск», а не «Отправлено по электронной почте» в CA

14 - Нажмите «Продолжить» и загрузите сгенерированный файл

15- Загрузите сгенерированный файл Apple и откройте его, пока доступ к Keychain Access открыт

Включение доступа APN в Xcode

1- Выберите свой проект

Вкладка «Свойства»

3- Найти Push-уведомления и включить его

4-Найти фоновый режим и включить его и проверить удаленные уведомления

Отмена регистрации от Push-уведомлений

Чтобы отменить регистрацию с удаленных уведомлений программно, вы можете использовать

Objective-C

```
[[UIApplication sharedApplication] unregisterForRemoteNotifications];
```

стриж

```
UIApplication.sharedApplication().unregisterForRemoteNotifications()
```

это похоже на настройку вашего телефона и ручное отключение уведомлений для приложения.

ПРИМЕЧАНИЕ. В редких случаях вам может понадобиться (например: когда ваше приложение больше не поддерживает push-уведомления)

Если вы просто хотите разрешить пользователю временно отключать уведомления. Вы должны реализовать метод удаления маркера устройства в базе данных вашего сервера. иначе, если вы только отключите уведомление локально на своем устройстве, ваш сервер по-прежнему будет отправлять сообщения.

Установка значка значка значка приложения

Используйте следующий фрагмент кода, чтобы установить номер значка из вашего приложения (предположим, что `someNumber` был объявлен ранее):

Objective-C

```
[UIApplication sharedApplication].applicationIconBadgeNumber = someNumber;
```

стриж

```
UIApplication.shared.applicationIconBadgeNumber = someNumber
```

Чтобы полностью *удалить* значок, просто установите `someNumber = 0` .

Тестирование push-уведомлений

Всегда полезно проверить, как работают push-уведомления даже до того, как у вас будет готовна ваша серверная часть, чтобы убедиться, что все правильно настроено на вашей стороне. Очень легко отправить себе push-уведомление, используя следующий PHP-скрипт.

1. Сохраните сценарий как файл (например, send_push.php) в той же папке, что и ваш сертификат (разработка или производство)
2. Отредактируйте его, чтобы поместить токен вашего устройства, пароль из сертификата

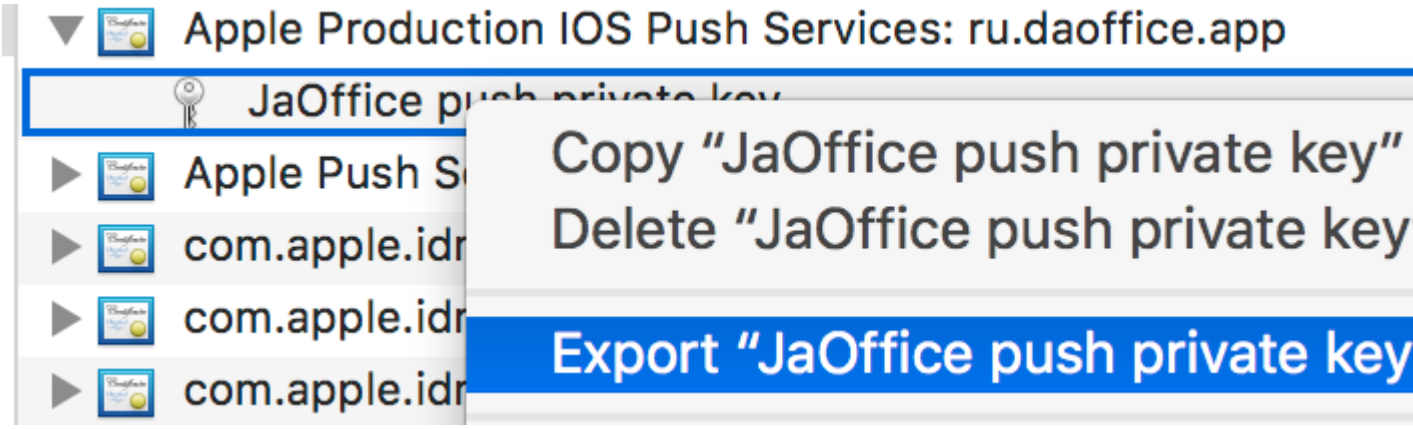
3. Выберите правильный путь для открытия соединения, dev_path или prod_path (здесь используется сценарий «Открыть соединение с APNS-сервером»)
4. cd в папку в терминале и запустить команду 'php send_push'
5. Получать уведомление на своем устройстве

```
<?php

// Put your device token here (without spaces):
$deviceToken = '20128697f872d7d39e48c4a61f50cb11d77789b39e6fc6b4cd7ec80582ed5229';
// Put your final pem cert name here. it is supposed to be in the same folder as this script
$cert_name = 'final_cert.pem';
// Put your private key's passphrase here:
$passphrase = '1234';

// sample point
$alert = 'Hello world!';
$event = 'new_incoming_message';

// You can choose either of the paths, depending on what kind of certificate you are using
$dev_path = 'ssl://gateway.sandbox.push.apple.com:2195';
$prod_path = 'ssl://gateway.push.apple.com:2195';

////////////////////////////////////////////////////////////////////////////////

$ctx = stream_context_create();
stream_context_set_option($ctx, 'ssl', 'local_cert', $cert_name);
stream_context_set_option($ctx, 'ssl', 'passphrase', $passphrase);

// Open a connection to the APNS server
$fp = stream_socket_client(
    $dev_path, $err,
    $errstr, 60, STREAM_CLIENT_CONNECT|STREAM_CLIENT_PERSISTENT, $ctx);

if (!$fp)
    exit("Failed to connect: $err $errstr" . PHP_EOL);

echo 'Connected to APNS' . PHP_EOL;

// Create the payload body
// it should be as short as possible
// if the notification doesnt get delivered that is most likely
// because the generated message is too long
$body['aps'] = array(
                     'alert' => $alert,
                     'sound' => 'default',
                     'event' => $event
                     );

// Encode the payload as JSON
$payload = json_encode($body);

// Build the binary notification
$msg = chr(0) . pack('n', 32) . pack('H*', $deviceToken) . pack('n', strlen($payload)) .
$payload;

// Send it to the server
$result = fwrite($fp, $msg, strlen($msg));

if (!$result)
```

```
    echo 'Message not delivered' . PHP_EOL;
else
    echo 'Message successfully delivered' . PHP_EOL;

// Close the connection to the server
fclose($fp);
```

Создание сертификата .pem из вашего .cer-файла для перехода к разработчику сервера

1. Сохранить aps.cer в папке
2. Откройте «Доступ к ключам» и экспортируйте ключ, находящийся под этим сертификатом, в файл .p12 (назовите его key.p12). Для этого щелкните правой кнопкой мыши и выберите «Экспорт». Сохраните его в той же папке, что и в шаге 1. При экспорте вам будет предложено ввести пароль. Сделайте что-нибудь и запомните.



3. cd в эту папку в терминале и выполнить следующие команды:
4. Преобразовать .cer в сертификат .pem

```
openssl x509 -in aps.cer -inform der -out aps.pem
```

5. Преобразуйте свой ключ в формат .pem. Чтобы открыть ключ, введите пароль, который вы его экспортировали, из брелка на шаге 2. Затем введите другой пароль, который защитит экспортированный файл. Вам будет предложено ввести его дважды для подтверждения.

```
openssl pkcs12 -nocerts -out key.pem -in key.p12
```

6. Объединение файлов в один файл

```
cat key.pem aps.pem > final_cert.pem
```

7. Конечный результат final_cert.pem. Передайте его разработчикам сервера с паролем с

шага 5, чтобы они могли использовать защищенный сертификат.

Прочитайте Всплывающие уведомления онлайн: https://riptutorial.com/ru/ios/topic/3492/всплывающие-уведомления

глава 124: Вырезать UIImage в круг

Examples

Вырезать изображение в круг - Цель C

import `#include <math.h>`

Код в viewDidLoad или loadView должен выглядеть чем-то вроде этого

```
- (void)loadView
{
[super loadView];
UIImageView *imageView=[[UIImageView alloc]initWithFrame:CGRectMake(0, 50, 320, 320)];
[self.view addSubview:imageView];
UIImage *image=[UIImage imageNamed:@"Dubai-Photos-Images-Travel-Tourist-Images-Pictures-
800x600.jpg"];
imageView.image=[self circularScaleAndCropImage:[UIImage imageNamed:@"Dubai-Photos-Images-
Travel-Tourist-Images-Pictures-800x600.jpg"] frame:CGRectMake(0, 0, 320, 320)];
}
```

Наконец, функция, выполняющая тяжелый подъемный `circularScaleAndCropImage` , как определено ниже

```
- (UIImage*)circularScaleAndCropImage:(UIImage*)image frame:(CGRect)frame {
    // This function returns a newImage, based on image, that has been:
    // - scaled to fit in (CGRect) rect
    // - and cropped within a circle of radius: rectWidth/2

    //Create the bitmap graphics context
    UIGraphicsBeginImageContextWithOptions(CGSizeMake(frame.size.width, frame.size.height),
NO, 0.0);
    CGContextRef context = UIGraphicsGetCurrentContext();

    //Get the width and heights
    CGFloat imageWidth = image.size.width;
    CGFloat imageHeight = image.size.height;
    CGFloat rectWidth = frame.size.width;
    CGFloat rectHeight = frame.size.height;

    //Calculate the scale factor
    CGFloat scaleFactorX = rectWidth/imageWidth;
    CGFloat scaleFactorY = rectHeight/imageHeight;

    //Calculate the centre of the circle
    CGFloat imageCentreX = rectWidth/2;
    CGFloat imageCentreY = rectHeight/2;

    // Create and CLIP to a CIRCULAR Path
    // (This could be replaced with any closed path if you want a different shaped clip)
    CGFloat radius = rectWidth/2;
    CGContextBeginPath (context);
    CGContextAddArc (context, imageCentreX, imageCentreY, radius, 0, 2*M_PI, 0);
    CGContextClosePath (context);
```

```
    CGContextClip (context);

    //Set the SCALE factor for the graphics context
    //All future draw calls will be scaled by this factor
    CGContextScaleCTM (context, scaleFactorX, scaleFactorY);

    // Draw the IMAGE
    CGRect myRect = CGRectMake(0, 0, imageWidth, imageHeight);
    [image drawInRect:myRect];

    UIImage *newImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();

    return newImage;
}
```

Пример SWIFT 3

```
 override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        let imageView = UIImageView(frame: CGRect(x: CGFloat(0), y: CGFloat(50), width:
CGFloat(320), height: CGFloat(320)))
        view.addSubview(imageView)
        let image = UIImage(named: "Dubai-Photos-Images-Travel-Tourist-Images-Pictures-
800x600.jpg")
        imageView.image = circularScaleAndCropImage(UIImage(named: "Dubai-Photos-Images-
Travel-Tourist-Images-Pictures-800x600.jpg")!, frame: CGRect(x: CGFloat(0), y: CGFloat(0),
width: CGFloat(100), height: CGFloat(100)))
    }
```

Наконец, функция, выполняющая тяжелый подъемный круговой шкал, как определено ниже

```
func circularScaleAndCropImage(_ image: UIImage, frame: CGRect) -> UIImage{
        // This function returns a newImage, based on image, that has been:
        // - scaled to fit in (CGRect) rect
        // - and cropped within a circle of radius: rectWidth/2
        //Create the bitmap graphics context
        UIGraphicsBeginImageContextWithOptions(CGSize(width: CGFloat(frame.size.width),
height: CGFloat(frame.size.height)), false, 0.0)
        let context: CGContext? = UIGraphicsGetCurrentContext()
        //Get the width and heights
        let imageWidth: CGFloat = image.size.width
        let imageHeight: CGFloat = image.size.height
        let rectWidth: CGFloat = frame.size.width
        let rectHeight: CGFloat = frame.size.height
        //Calculate the scale factor
        let scaleFactorX: CGFloat = rectWidth / imageWidth
        let scaleFactorY: CGFloat = rectHeight / imageHeight
        //Calculate the centre of the circle
        let imageCentreX: CGFloat = rectWidth / 2
        let imageCentreY: CGFloat = rectHeight / 2
        // Create and CLIP to a CIRCULAR Path
        // (This could be replaced with any closed path if you want a different shaped clip)
        let radius: CGFloat = rectWidth / 2
        context?.beginPath()
```

```
        context?.addArc(center: CGPoint(x: imageCentreX, y: imageCentreY), radius: radius,
startAngle: CGFloat(0), endAngle: CGFloat(2 * Float.pi), clockwise: false)
        context?.closePath()
        context?.clip()
        //Set the SCALE factor for the graphics context
        //All future draw calls will be scaled by this factor
        context?.scaleBy(x: scaleFactorX, y: scaleFactorY)
        // Draw the IMAGE
        let myRect = CGRect(x: CGFloat(0), y: CGFloat(0), width: imageWidth, height:
imageHeight)
        image.draw(in: myRect)
        let newImage: UIImage? = UIGraphicsGetImageFromCurrentImageContext()
        UIGraphicsEndImageContext()
        return newImage!
    }
```

Прочитайте Вырезать UIImage в круг онлайн: https://riptutorial.com/ru/ios/topic/7222/вырезать-uiimage-в-круг

глава 125: Глубокая привязка в iOS

замечания

Полезная документация Apple с примерами и пояснениями.

Examples

Открытие приложения на основе его схемы URL

Чтобы открыть приложение с определенной схемой URL `todolist://` :

Objective-C

```
NSURL *myURL = [NSURL URLWithString:@"todolist://there/is/something/to/do"];
[[UIApplication sharedApplication] openURL:myURL];
```

стриж

```
let stringURL = "todolist://there/is/something/to/do"
if let url = NSURL(string: stringURL) {
    UIApplication.shared().openURL(url)
}
```

HTML

```
<a href="todolist://there/is/something/to/do">New SMS Message</a>
```

> **Примечание.** Полезно проверить, можно ли открыть ссылку, чтобы в противном случае отобразить соответствующее сообщение пользователю. Это можно сделать с `canOpenURL:` метода `canOpenURL:`

Добавление схемы URL в собственное приложение

Предположим, вы работаете над приложением под названием `MyTasks` , и вы хотите разрешить входящие URL-адреса создавать новую задачу с заголовком и телом. URL-адрес, который вы проектируете, может выглядеть примерно так:

```
mytasks://create?title=hello&body=world
```

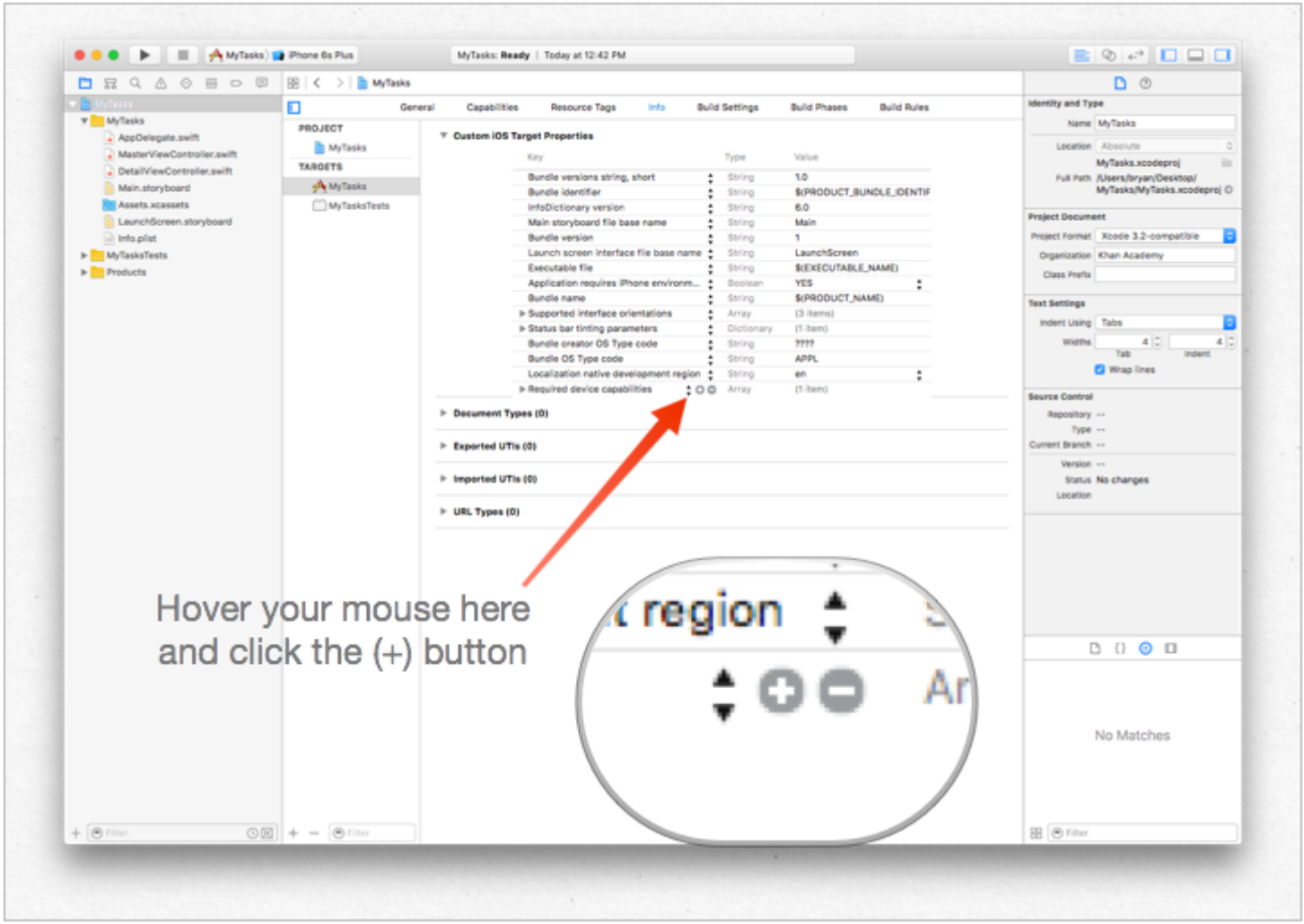
(Конечно, параметры `text` и `body` используются для заполнения нашей задачи, которую мы создаем!)

Вот «Большие шаги» для добавления этой схемы URL в ваш проект:

1. Зарегистрируйте URL-схему в файле `Info.plist` вашего приложения, чтобы система узнала, когда нужно перенаправить URL-адрес в ваше приложение.
2. Добавьте функцию в ваш `UIApplicationDelegate` который принимает и обрабатывает входящие URL-адреса.
3. Выполнять любую задачу при открытии этого URL-адреса.

Шаг 1. Зарегистрируйте схему URL в Info.plist:

Во-первых, нам нужно добавить запись «URL-адреса» в наш файл Info.plist. Нажмите кнопку (+) здесь:



... затем введите уникальный идентификатор для вашего приложения, а также схему URL, которую вы хотите использовать. Быть конкретными! Вы не хотите, чтобы схема URL конфликтует с реализацией другого приложения. Лучше быть слишком длинным здесь, чем слишком коротким!

Шаг второй: обрабатывайте URL-адрес в UIApplicationDelegate

Нам нужно реализовать `application:openURL:options:` на нашем `UIApplicationDelegate` . Мы проверим входящий `URL` и посмотрим, есть ли какое-либо действие, которое мы можем предпринять!

Одна из реализаций будет такой:

```
func application(app: UIApplication, openURL url: NSURL, options: [String : AnyObject]) ->
Bool {
    if url.scheme == "mytasks" && url.host == "create" {
        let title = // get the title out of the URL's query using a method of your choice
        let body = // get the title out of the URL's query using a method of your choice
        self.rootViewController.createTaskWithTitle(title, body: body)
        return true
    }

    return false
}
```

Шаг третий: выполните задачу в зависимости от URL-адреса.

Когда пользователь открывает ваше приложение через URL-адрес, они, вероятно, ожидали, *что* что- *то* произойдет. Возможно, это переход на часть контента, возможно, это создание нового элемента - в этом примере мы собираемся создать новую задачу в приложении!

В приведенном выше коде мы можем увидеть вызов `self.rootViewController.createTaskWithTitle(:body:)` - поэтому, если ваш `AppDelegate` имеет указатель на свой контроллер корневого представления, который правильно выполняет функцию, вы все настроены!

Настройка deeplink для вашего приложения

Настройка глубокой привязки для вашего приложения проста. Вам просто нужен

небольшой URL-адрес, который вы хотите открыть для своего приложения.

Выполните шаги, чтобы настроить глубокую привязку для вашего приложения.

1. Давайте создадим проект и назовите его DeepLinkPOC.
2. Теперь выберите цель проекта.
3. После выбора цели выберите вкладку «информация».
4. Прокрутите вниз до тех пор, пока не увидите параметр **URL-адресов**
5. Нажмите «+».
6. Вы увидите, что **схемы URL** добавляют строку, с помощью которой вы хотите открыть приложение. Давайте добавим « **DeepLinking** » в схемы URL.

Итак, чтобы открыть приложение, вы можете запустить его, введя **«DeepLinking: //»** в ваше сафари. Строка с глубокой связью имеет следующий формат.

```
[scheme]://[host]/[path]  --> DeepLinking://path/Page1
```

где, Схема: «Глубокая точка» Хост: путь «путь»: «Страница1»

Примечание . Даже если вы не добавите хост и путь, он запустит приложение, поэтому не беспокойтесь. Но вы можете добавить хост и путь для дополнительной перенаправления на определенную страницу после запуска приложения.

7. Теперь добавьте следующий метод в ваш appdelegate.

Swift:

```
 func application(application: UIApplication, openURL url: NSURL, sourceApplication: String?,
annotation: AnyObject) -> Bool
```

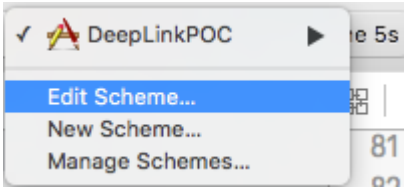
Objective-C:

```
-(BOOL)application:(UIApplication *)application
          openURL:(NSURL *)url
          sourceApplication:(NSString *)sourceApplication
          annotation:(id)annotation
```

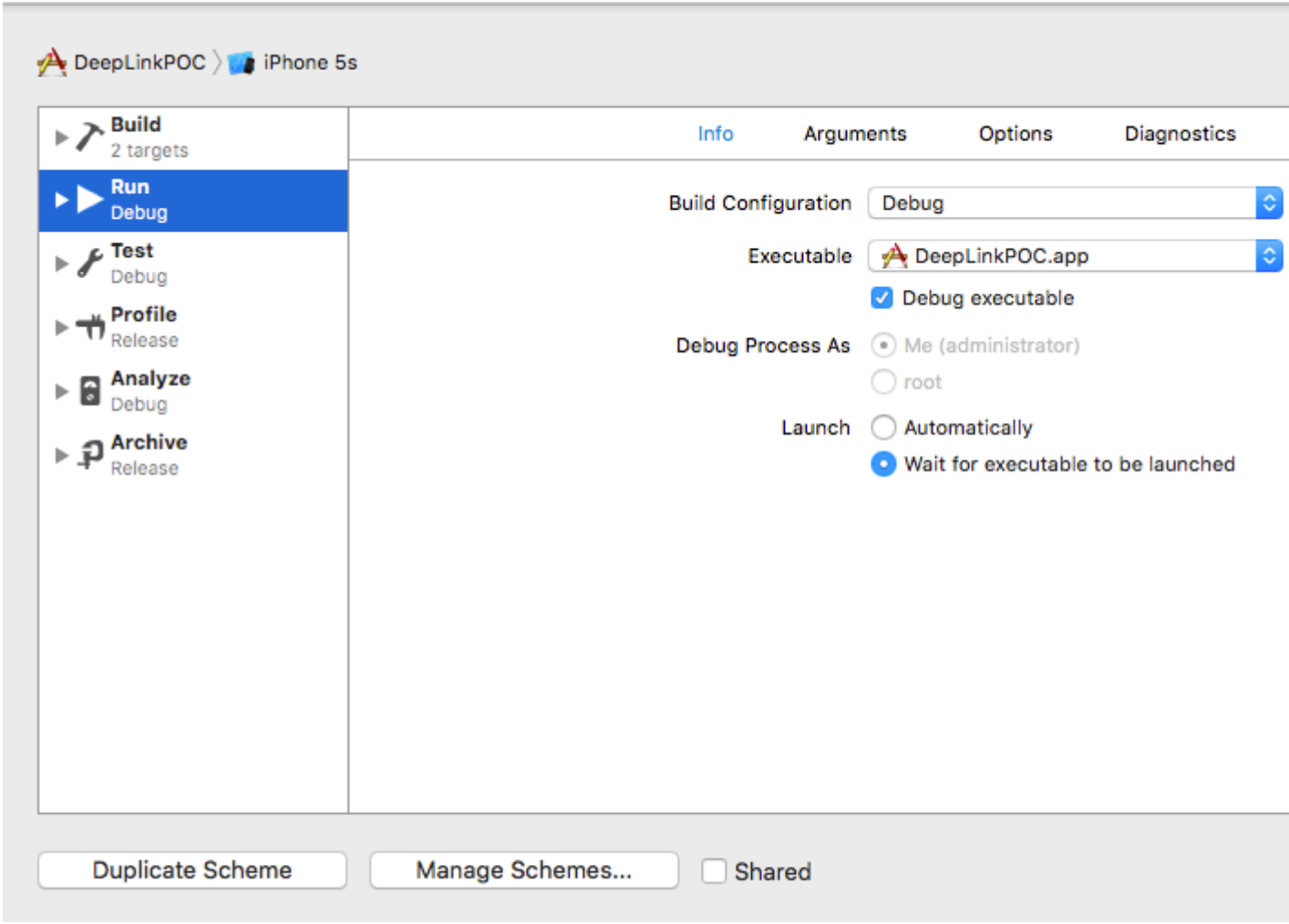
Вышеупомянутый метод вызывается всякий раз, когда ваше приложение запускается с использованием строки глубокой привязки, установленной для вашего приложения.

8. Настало время установить приложение, но подождите, прежде чем вы сразу перейдете к запуску button.Lets делает небольшое изменение в методе запуска приложения схемы.

- Выберите и отредактируйте свою схему как



- изменить его тип запуска и закрыть



9. Теперь нажмите кнопку «Выполнить» (если вы хотите добавить точку останова в ваши методы didFinishLaunchingWithOptions и openURL для наблюдения за значениями)

10. Появится сообщение «Ожидание запуска DeepLinkPOC (или вашего имени приложения)».

11. Откройте сафари и введите « **DeepLinking: //** » в строку поиска, на которой появится приглашение «открыть эту страницу в DeepLinkPOC», чтобы открыть приложение.

Надеюсь, вам нужно знать, как настроить глубокую привязку к вашему приложению :)

Прочитайте Глубокая привязка в iOS онлайн: https://riptutorial.com/ru/ios/topic/5173/глубокая-привязка-в-ios

глава 126: График (Coreplot)

Examples

Создание графиков с помощью CorePlot

Core Plot предоставляет podspec, поэтому вы можете использовать cocoapods в качестве менеджера библиотеки, который должен сделать установку и обновление намного проще

Установите cocoapods на вашу систему

В каталоге проекта добавьте текстовый файл в проект под названием Podfile, введя `pod init` в каталог проекта

В подпикселе добавьте строку «CorePlot», «~> 1,6»,

В терминале, cd в ваш каталог проекта и запустить pod install

Cocoapods создаст файл xcworkspace, который вы должны использовать для запуска вашего проекта (файл .xcodeproj не будет содержать библиотеки модулей)

Откройте .xcworkspace, сгенерированное CocoaPods

В файле ViewController.h

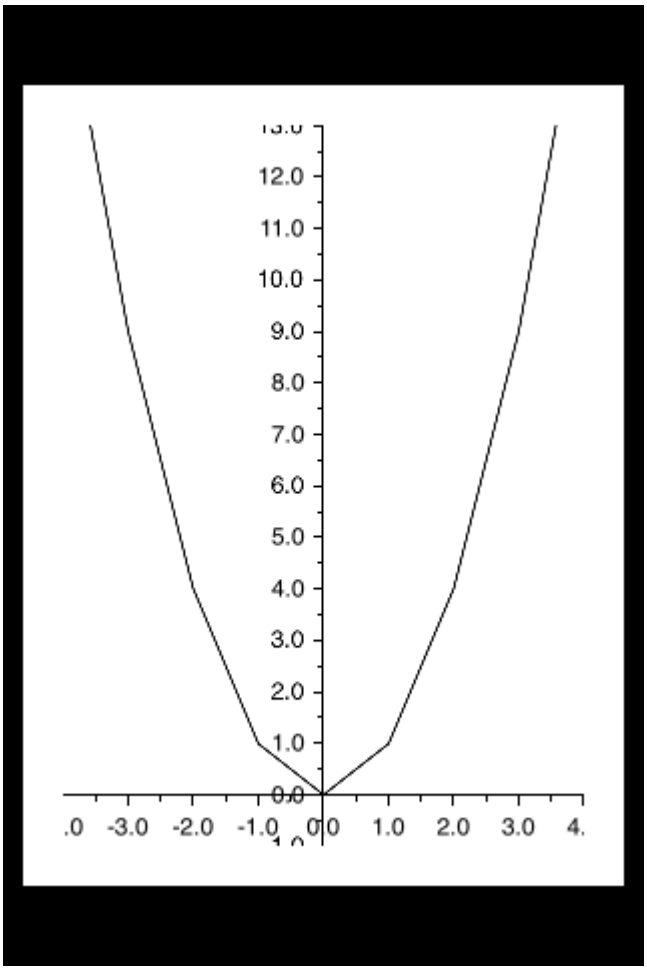
```
#import <CorePlot/ios/CorePlot.h>
//#import "CorePlot-CocoaTouch.h" or the above import statement
@interface ViewController : UIViewController<CPTPlotDataSource>
```

В файле ViewController.m

```
-(void)loadView
{
    [super loadView];
    // We need a hostview, you can create one in IB (and create an outlet) or just do this:
    CPTGraphHostingView* hostView = [[CPTGraphHostingView alloc] initWithFrame:CGRectMake(10,
40, 300, 400)];
    hostView.backgroundColor=[UIColor whiteColor];
    self.view.backgroundColor=[UIColor blackColor];
    [self.view addSubview: hostView];
    // Create a CPTGraph object and add to hostView
    CPTGraph* graph = [[CPTXYGraph alloc] initWithFrame:CGRectMake(10, 40, 300, 400)];
    hostView.hostedGraph = graph;
    // Get the (default) plotspace from the graph so we can set its x/y ranges
    CPTXYPlotSpace *plotSpace = (CPTXYPlotSpace *) graph.defaultPlotSpace;
    // Note that these CPTPlotRange are defined by START and LENGTH (not START and END) !!
    [plotSpace setYRange: [CPTPlotRange plotRangeWithLocation:CPTDecimalFromFloat( 0 )
length:CPTDecimalFromFloat( 20 )]];
    [plotSpace setXRange: [CPTPlotRange plotRangeWithLocation:CPTDecimalFromFloat( -4 )
length:CPTDecimalFromFloat( 8 )]];
```

```
    // Create the plot (we do not define actual x/y values yet, these will be supplied by the
datasource...)
    CPTScatterPlot* plot = [[CPTScatterPlot alloc] initWithFrame:CGRectZero];
    // Let's keep it simple and let this class act as datasource (therefore we implemtn
<CPTPlotDataSource>)
    plot.dataSource = self;
    // Finally, add the created plot to the default plot space of the CPTGraph object we
created before
    [graph addPlot:plot toPlotSpace:graph.defaultPlotSpace];
}
// This method is here because this class also functions as datasource for our graph
// Therefore this class implements the CPTPlotDataSource protocol
-(NSUInteger)numberOfRecordsForPlot:(CPTPlot *)plotnumberOfRecords
{
        return 9; // Our sample graph contains 9 'points'
}
// This method is here because this class also functions as datasource for our graph
// Therefore this class implements the CPTPlotDataSource protocol
-(NSNumber *)numberForPlot:(CPTPlot *)plot field:(NSUInteger)fieldEnum
recordIndex:(NSUInteger)index
{
    // We need to provide an X or Y (this method will be called for each) value for every
index
    int x = index - 4;
    // This method is actually called twice per point in the plot, one for the X and one for
the Y value
    if(fieldEnum == CPTScatterPlotFieldX)
    {
    // Return x value, which will, depending on index, be between -4 to 4
    return [NSNumber numberWithInt: x];
    } else
    {
    // Return y value, for this example we'll be plotting y = x * x
        return [NSNumber numberWithInt: x * x];
    }
}
```

Сгенерированный вывод приведен ниже:



Прочитайте График (Coreplot) онлайн: https://riptutorial.com/ru/ios/topic/7302/график--coreplot-

глава 127: десантный

Examples

десантный

Objective-C

Airdrop можно использовать из `UIActivityViewController` . Класс `UIActivityViewController` - это стандартный контроллер представлений, который предоставляет несколько стандартных сервисов, таких как копирование элементов в буфер обмена, совместное использование контента на сайтах социальных сетей, отправка сообщений через сообщения, AirDrop и некоторые сторонние приложения.

В этом случае мы отправим изображение через `UIActivityViewController`

```
UIImage *hatImage = [UIImage imageNamed:@"logo.png"];
if (hatImage)//checks if the image file is not nil
{
//Initialise a UIActivityViewController
UIActivityViewController *controller = [[UIActivityViewController alloc]
initWithActivityItems:@[hatImage] applicationActivities:nil];
//Excludes following options from the UIActivityViewController menu
NSArray *excludeActivities = @[UIActivityTypePostToWeibo,UIActivityTypePrint,
UIActivityTypeMail,UIActivityTypeMessage,UIActivityTypePostToTwitter,UIActivityTypePostToFacebook,


UIActivityTypeCopyToPasteboard,UIActivityTypeAssignToContact,

UIActivityTypeSaveToCameraRoll,UIActivityTypeAddToReadingList,

UIActivityTypePostToFlickr,UIActivityTypePostToVimeo,

UIActivityTypePostToTencentWeibo];
controller.excludedActivityTypes = excludeActivities;
[self presentViewController:controller animated:YES completion:nil];
}
```

стриж

```
if ((newImage) != nil)
        {
            let activityVC = UIActivityViewController(activityItems: [newImage],
applicationActivities: nil)
            activityVC.excludedActivityTypes =[UIActivityTypeAddToReadingList]
            self.presentViewController(activityVC, animated: true, completion: nil)

        }
```

Прочитайте десантный онлайн: https://riptutorial.com/ru/ios/topic/7360/десантный

глава 128: Динамика UIKit

Вступление

UIKit Dynamics - это полный физический движок реального мира, интегрированный в UIKit. Он позволяет создавать интерфейсы, которые чувствуют реальность, добавляя такие поведения, как гравитация, привязанности, столкновение и силы. Вы определяете физические черты, которые вы хотите, чтобы ваши элементы интерфейса принимались, а механизм динамики заботится обо всем остальном.

замечания

Импортная вещь, которую следует учитывать при использовании UIKit Dynamics, - это представления, которые позиционируются аниматором, не могут быть легко размещены другими распространенными методами компоновки iOS.

Новички в UIKit Dynamics часто борются с этим важным предостережением. Размещение ограничений на представление, которое также является элементом `UIDynamicBehavior` , вероятно, вызовет путаницу, как механизм автоматического макета, так и динамический движок аниматора в соответствующей позиции. Точно так же попытка установить кадр непосредственно из вида, контролируемого аниматором, как правило, приведет к дрожащей анимации и неожиданному размещению. Добавление представления в качестве элемента в `UIDynamicBehavior` означает, что аниматор возьмет на себя ответственность за позиционирование представления и, поскольку такие изменения позиций взгляда должны быть реализованы через аниматор.

Кадр представления, который обновляется динамическим аниматором, может быть установлен, но за ним следует сразу же отправить сообщение аниматору для обновления внутренней модели аниматора иерархии представлений. Например, если у меня есть `UILabel` , `label` который является элементом `UIGravityBehavior` я могу переместить его в верхнюю часть экрана, чтобы посмотреть, как он упадет снова, сказав:

стриж

```
label.frame = CGRect(x: 0.0, y: 0.0, width: label.intrinsicContentSize.width, height:
label.intrinsicContentSize.height)
dynamicAnimator.updateItem(usingCurrentState: label)
```

Objective-C

```
self.label.frame = CGRectMake(0.0, 0.0, self.label.intrinsicContentSize.width,
```

```
self.label.intrinsicContentSize.height);
[self.dynamicAnimator updateItemUsingCurrentState: self.label];
```

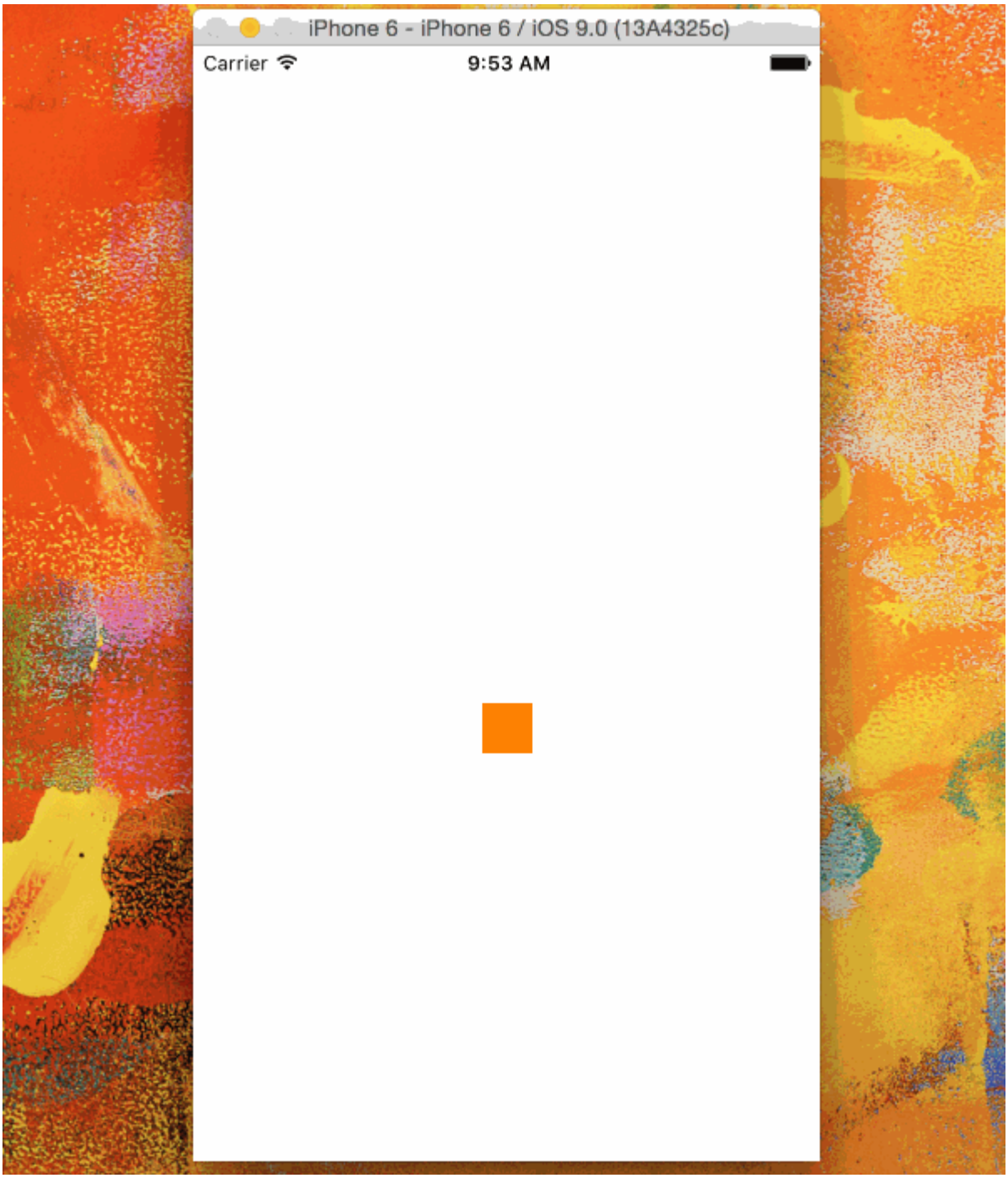
После этого аниматор применит гравитационное поведение от нового местоположения метки.

Другой распространенный метод - использовать `UIDynamicBehaviors` для размещения представлений. Например , если позиционирование вида под сенсорным случае желательно, создавая `UIAttachmentBehavior` и обновляют свой `anchorPoint` в любом `touchesMoved` или `UIGestureRecognizer` действия «ы является эффективной стратегией.

Examples

Падающая площадь

Давайте нарисуем квадрат посередине нашего обзора и сделаем его падающим на дно и остановимся на нижнем крае, коллизирующем с нижней границей экрана.




```
@IBOutlet var animationView: UIView!
var squareView:UIView!
var collision: UICollisionBehavior!
var animator: UIDynamicAnimator!
var gravity: UIGravityBehavior!

override func viewDidLoad() {
    super.viewDidLoad()
    let squareSize = CGSize(width: 30.0, height: 30.0)
    let centerPoint = CGPoint(x: self.animationView.bounds.midX - (squareSize.width/2), y:
self.animationView.bounds.midY - (squareSize.height/2))
    let frame = CGRect(origin: centerPoint, size: squareSize)
    squareView = UIView(frame: frame)
    squareView.backgroundColor = UIColor.orangeColor()
    animationView.addSubview(squareView)
    animator = UIDynamicAnimator(referenceView: view)
```

```
    gravity = UIGravityBehavior(items: [squareView])
    animator.addBehavior(gravity)
    collision = UICollisionBehavior(items: [square])
    collision.translatesReferenceBoundsIntoBoundary = true
    animator.addBehavior(collision)
}
```

Флик-взгляд на основе скорости жестов

В этом примере показано, как смотреть дорожку с панорамой и отходить в физическом режиме.

стриж

```
class ViewController: UIViewController
{
    // Adjust to change speed of view from flick
    let magnitudeMultiplier: CGFloat = 0.0008

    lazy var dynamicAnimator: UIDynamicAnimator =
    {
        let dynamicAnimator = UIDynamicAnimator(referenceView: self.view)
        return dynamicAnimator
    }()

    lazy var gravity: UIGravityBehavior =
    {
        let gravity = UIGravityBehavior(items: [self.orangeView])
        return gravity
    }()

    lazy var collision: UICollisionBehavior =
    {
        let collision = UICollisionBehavior(items: [self.orangeView])
        collision.translatesReferenceBoundsIntoBoundary = true
        return collision
    }()
```

```
    lazy var orangeView: UIView =
    {
        let widthHeight: CGFloat = 40.0
        let orangeView = UIView(frame: CGRect(x: 0.0, y: 0.0, width: widthHeight, height:
widthHeight))
        orangeView.backgroundColor = UIColor.orange
        self.view.addSubview(orangeView)
        return orangeView
    }()

    lazy var panGesture: UIPanGestureRecognizer =
    {
        let panGesture = UIPanGestureRecognizer(target: self, action:
#selector(self.handlePan(sender:)))
        return panGesture
    }()

    lazy var attachment: UIAttachmentBehavior =
    {
        let attachment = UIAttachmentBehavior(item: self.orangeView, attachedToAnchor: .zero)
        return attachment
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        dynamicAnimator.addBehavior(gravity)
        dynamicAnimator.addBehavior(collision)
        orangeView.addGestureRecognizer(panGesture)
    }

    override func viewDidLayoutSubviews()
    {
        super.viewDidLayoutSubviews()
        orangeView.center = view.center
        dynamicAnimator.updateItem(usingCurrentState: orangeView)
    }

    func handlePan(sender: UIPanGestureRecognizer)
    {
        let location = sender.location(in: view)
        let velocity = sender.velocity(in: view)
        let magnitude = sqrt((velocity.x * velocity.x) + (velocity.y * velocity.y))
        switch sender.state
        {
        case .began:
            attachment.anchorPoint = location
            dynamicAnimator.addBehavior(attachment)
        case .changed:
            attachment.anchorPoint = location
        case .cancelled, .ended, .failed, .possible:
            let push = UIPushBehavior(items: [self.orangeView], mode: .instantaneous)
            push.pushDirection = CGVector(dx: velocity.x, dy: velocity.y)
            push.magnitude = magnitude * magnitudeMultiplier
            dynamicAnimator.removeBehavior(attachment)
            dynamicAnimator.addBehavior(push)
        }
    }
}
```

Objective-C

```
@interface ViewController ()

@property (nonatomic, assign) CGFloat magnitudeMultiplier;
@property (nonatomic, strong) UIDynamicAnimator *dynamicAnimator;
@property (nonatomic, strong) UIGravityBehavior *gravity;
@property (nonatomic, strong) UICollisionBehavior *collision;
@property (nonatomic, strong) UIView *orangeView;
@property (nonatomic, strong) UIPanGestureRecognizer *panGesture;
@property (nonatomic, strong) UIAttachmentBehavior *attachment;

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.dynamicAnimator addBehavior:self.gravity];
    [self.dynamicAnimator addBehavior:self.collision];
    [self.orangeView addGestureRecognizer:self.panGesture];
    // Adjust to change speed of view from flick
    self.magnitudeMultiplier = 0.0008f;
}

- (void)viewDidLayoutSubviews
{
    [super viewDidLayoutSubviews];
    self.orangeView.center = self.view.center;
    [self.dynamicAnimator updateItemUsingCurrentState:self.orangeView];
}

- (void)handlePan:(UIPanGestureRecognizer *)sender
{
    CGPoint location = [sender locationInView:self.view];
    CGPoint velocity = [sender velocityInView:self.view];
    CGFloat magnitude = sqrt((velocity.x * velocity.x) + (velocity.y * velocity.y));
    if (sender.state == UIGestureRecognizerStateBegan)
    {
        self.attachment.anchorPoint = location;
        [self.dynamicAnimator addBehavior:self.attachment];
    }
    else if (sender.state == UIGestureRecognizerStateChanged)
    {
        self.attachment.anchorPoint = location;
    }
    else if (sender.state == UIGestureRecognizerStateCancelled ||
             sender.state == UIGestureRecognizerStateEnded ||
             sender.state == UIGestureRecognizerStateFailed ||
             sender.state == UIGestureRecognizerStatePossible)
    {
        UIPushBehavior *push = [[UIPushBehavior alloc] initWithItems:@[self.orangeView]
mode:UIPushBehaviorModeInstantaneous];
        push.pushDirection = CGVectorMake(velocity.x, velocity.y);
        push.magnitude = magnitude * self.magnitudeMultiplier;
        [self.dynamicAnimator removeBehavior:self.attachment];
        [self.dynamicAnimator addBehavior:push];
    }
}
```

```
#pragma mark - Lazy Init
- (UIDynamicAnimator *)dynamicAnimator
{
    if (!_dynamicAnimator)
    {
        _dynamicAnimator = [[UIDynamicAnimator alloc]initWithReferenceView:self.view];
    }
    return _dynamicAnimator;
}

- (UIGravityBehavior *)gravity
{
    if (!_gravity)
    {
        _gravity = [[UIGravityBehavior alloc]initWithItems:@[self.orangeView]];
    }
    return _gravity;
}

- (UICollisionBehavior *)collision
{
    if (!_collision)
    {
        _collision = [[UICollisionBehavior alloc]initWithItems:@[self.orangeView]];
        _collision.translatesReferenceBoundsIntoBoundary = YES;
    }
    return _collision;
}

- (UIView *)orangeView
{
    if (!_orangeView)
    {
        CGFloat widthHeight = 40.0f;
        _orangeView = [[UIView alloc]initWithFrame:CGRectMake(0.0, 0.0, widthHeight,
widthHeight)];
        _orangeView.backgroundColor = [UIColor orangeColor];
        [self.view addSubview:_orangeView];
    }
    return _orangeView;
}

- (UIPanGestureRecognizer *)panGesture
{
    if (!_panGesture)
    {
        _panGesture = [[UIPanGestureRecognizer alloc]initWithTarget:self
action:@selector(handlePan:)];
    }
    return _panGesture;
}

- (UIAttachmentBehavior *)attachment
{
    if (!_attachment)
    {
        _attachment = [[UIAttachmentBehavior alloc]initWithItem:self.orangeView
attachedToAnchor:CGPointZero];
    }
    return _attachment;
```

```
}

@end
```

Эффект «Sticky Corners» с использованием UIFieldBehaviors

В этом примере показано, как добиться эффекта, аналогичного FaceTime, было привлечено внимание к точке, когда она попадает в конкретную область, в этом случае две области сверху и снизу.

стриж

```
class ViewController: UIViewController
{
    lazy var dynamicAnimator: UIDynamicAnimator =
    {
        let dynamicAnimator = UIDynamicAnimator(referenceView: self.view)
        return dynamicAnimator
    }()

    lazy var collision: UICollisionBehavior =
    {
        let collision = UICollisionBehavior(items: [self.orangeView])
        collision.translatesReferenceBoundsIntoBoundary = true
        return collision
    }()

    lazy var fieldBehaviors: [UIFieldBehavior] =
    {
        var fieldBehaviors = [UIFieldBehavior]()
        for _ in 0 ..< 2
        {
            let field = UIFieldBehavior.springField()
            field.addItem(self.orangeView)
            fieldBehaviors.append(field)
        }
        return fieldBehaviors
    }()
```

```
    lazy var itemBehavior: UIDynamicItemBehavior =
    {
        let itemBehavior = UIDynamicItemBehavior(items: [self.orangeView])
        // Adjust these values to change the "stickiness" of the view
        itemBehavior.density = 0.01
        itemBehavior.resistance = 10
        itemBehavior.friction = 0.0
        itemBehavior.allowsRotation = false
        return itemBehavior
    }()

    lazy var orangeView: UIView =
    {
        let widthHeight: CGFloat = 40.0
        let orangeView = UIView(frame: CGRect(x: 0.0, y: 0.0, width: widthHeight, height:
widthHeight))
        orangeView.backgroundColor = UIColor.orange
        self.view.addSubview(orangeView)
        return orangeView
    }()

    lazy var panGesture: UIPanGestureRecognizer =
    {
        let panGesture = UIPanGestureRecognizer(target: self, action:
#selector(self.handlePan(sender:)))
        return panGesture
    }()

    lazy var attachment: UIAttachmentBehavior =
    {
        let attachment = UIAttachmentBehavior(item: self.orangeView, attachedToAnchor: .zero)
        return attachment
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        dynamicAnimator.addBehavior(collision)
        dynamicAnimator.addBehavior(itemBehavior)
        for field in fieldBehaviors
        {
            dynamicAnimator.addBehavior(field)
        }

        orangeView.addGestureRecognizer(panGesture)
    }

    override func viewDidLayoutSubviews()
    {
        super.viewDidLayoutSubviews()

        orangeView.center = view.center
        dynamicAnimator.updateItem(usingCurrentState: orangeView)

        for (index, field) in fieldBehaviors.enumerated()
        {
            field.position = CGPoint(x: view.bounds
                .midX, y:  view.bounds.height * (0.25 + 0.5 * CGFloat(index)))
            field.region = UIRegion(size: CGSize(width: view.bounds.width, height:
view.bounds.height * 0.5))
```

```
        }
    }

    func handlePan(sender: UIPanGestureRecognizer)
    {
        let location = sender.location(in: view)
        let velocity = sender.velocity(in: view)
        switch sender.state
        {
        case .began:
            attachment.anchorPoint = location
            dynamicAnimator.addBehavior(attachment)
        case .changed:
            attachment.anchorPoint = location
        case .cancelled, .ended, .failed, .possible:
            itemBehavior.addLinearVelocity(velocity, for: self.orangeView)
            dynamicAnimator.removeBehavior(attachment)
        }
    }
}
```

Objective-C

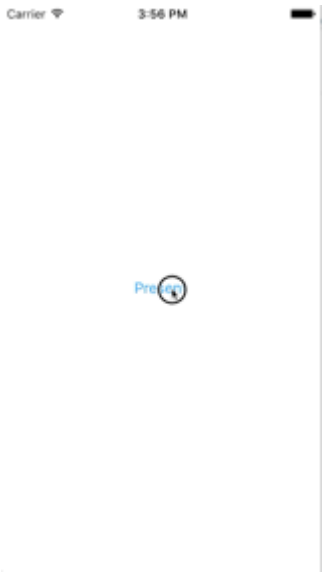
```
@interface ViewController ()

@property (nonatomic, strong) UIDynamicAnimator *dynamicAnimator;
@property (nonatomic, strong) UICollisionBehavior *collision;
@property (nonatomic, strong) UIAttachmentBehavior *attachment;
@property (nonatomic, strong) UIDynamicItemBehavior *itemBehavior;
@property (nonatomic, strong) NSArray <UIFieldBehavior *> *fieldBehaviors;
@property (nonatomic, strong) UIView *orangeView;
@property (nonatomic, strong) UIPanGestureRecognizer *panGesture;

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.dynamicAnimator addBehavior:self.collision];
    [self.dynamicAnimator addBehavior:self.itemBehavior];
    for (UIFieldBehavior *field in self.fieldBehaviors)
    {
        [self.dynamicAnimator addBehavior:field];
    }

    [self.orangeView addGestureRecognizer:self.panGesture];
}

- (void)viewDidLayoutSubviews
{
    [super viewDidLayoutSubviews];
    self.orangeView.center = self.view.center;
    [self.dynamicAnimator updateItemUsingCurrentState:self.orangeView];

    for (NSInteger i = 0; i < self.fieldBehaviors.count; i++)
    {
        UIFieldBehavior *field = self.fieldBehaviors[i];
```

```
        field.position = CGPointMake(CGRectGetMidX(self.view.bounds),
CGRectGetHeight(self.view.bounds) * (0.25f + 0.5f * i));
        field.region = [[UIRegion
alloc]initWithSize:CGSizeMake(CGRectGetWidth(self.view.bounds),
CGRectGetHeight(self.view.bounds) * 0.5)];
    }
}

- (void)handlePan:(UIPanGestureRecognizer *)sender
{
    CGPoint location = [sender locationInView:self.view];
    CGPoint velocity = [sender velocityInView:self.view];
    if (sender.state == UIGestureRecognizerStateBegan)
    {
        self.attachment.anchorPoint = location;
        [self.dynamicAnimator addBehavior:self.attachment];
    }
    else if (sender.state == UIGestureRecognizerStateChanged)
    {
        self.attachment.anchorPoint = location;
    }
    else if (sender.state == UIGestureRecognizerStateCancelled ||
             sender.state == UIGestureRecognizerStateEnded ||
             sender.state == UIGestureRecognizerStateFailed ||
             sender.state == UIGestureRecognizerStatePossible)
    {
        [self.itemBehavior addLinearVelocity:velocity forItem:self.orangeView];
        [self.dynamicAnimator removeBehavior:self.attachment];
    }
}

#pragma mark - Lazy Init
- (UIDynamicAnimator *)dynamicAnimator
{
    if (!_dynamicAnimator)
    {
        _dynamicAnimator = [[UIDynamicAnimator alloc]initWithReferenceView:self.view];
    }
    return _dynamicAnimator;
}

- (UICollisionBehavior *)collision
{
    if (!_collision)
    {
        _collision = [[UICollisionBehavior alloc]initWithItems:@[self.orangeView]];
        _collision.translatesReferenceBoundsIntoBoundary = YES;
    }
    return _collision;
}

- (NSArray <UIFieldBehavior *> *)fieldBehaviors
{
    if (!_fieldBehaviors)
    {
        NSMutableArray *fields = [[NSMutableArray alloc]init];
        for (NSInteger i =  0; i < 2; i++)
        {
            UIFieldBehavior *field = [UIFieldBehavior springField];
            [field addItem:self.orangeView];
            [fields addObject:field];
```

```
        }
        _fieldBehaviors = fields;
    }
    return _fieldBehaviors;
}

- (UIDynamicItemBehavior *)itemBehavior
{
    if (!_itemBehavior)
    {
        _itemBehavior = [[UIDynamicItemBehavior alloc]initWithItems:@[self.orangeView]];
        // Adjust these values to change the "stickiness" of the view
        _itemBehavior.density = 0.01;
        _itemBehavior.resistance = 10;
        _itemBehavior.friction = 0.0;
        _itemBehavior.allowsRotation = NO;
    }
    return _itemBehavior;
}

- (UIView *)orangeView
{
    if (!_orangeView)
    {
        CGFloat widthHeight = 40.0f;
        _orangeView = [[UIView alloc]initWithFrame:CGRectMake(0.0, 0.0, widthHeight,
widthHeight)];
        _orangeView.backgroundColor = [UIColor orangeColor];
        [self.view addSubview:_orangeView];
    }
    return _orangeView;
}

- (UIPanGestureRecognizer *)panGesture
{
    if (!_panGesture)
    {
        _panGesture = [[UIPanGestureRecognizer alloc]initWithTarget:self
action:@selector(handlePan:)];
    }
    return _panGesture;
}

- (UIAttachmentBehavior *)attachment
{
    if (!_attachment)
    {
        _attachment = [[UIAttachmentBehavior alloc]initWithItem:self.orangeView
attachedToAnchor:CGPointZero];
    }
    return _attachment;
}

@end
```

Для получения дополнительной информации о `UIFieldBehaviors` вы можете увидеть сессию WWDC 2015 года «Что нового в динамике UIKit и визуальных эффектах» и сопроводительный пример кода .

UIDynamicBehavior Driven Custom Transition



В этом примере показано, как создать настраиваемый переход презентации, который управляется составным `UIDynamicBehavior` . Мы можем начать с создания контроллера представления, который будет представлять модальный.

стриж

```
class PresentingViewController: UIViewController
{
    lazy var button: UIButton =
    {
        let button = UIButton()
        button.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(button)
        button.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive
            = true
        button.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        button.setTitle("Present", for: .normal)
        button.setTextColor(UIColor.blue, for: .normal)

        return button
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        button.addTarget(self, action: #selector(self.didPressPresent), for: .touchUpInside)
    }

    func didPressPresent()
    {
        let modal = ModalViewController()
        modal.view.frame = CGRect(x: 0.0, y: 0.0, width: 200.0, height: 200.0)
        modal.modalPresentationStyle = .custom
        modal.transitioningDelegate = modal
        self.present(modal, animated: true)
    }
```

```
}
```

Objective-C

```
@interface PresentingViewController ()
@property (nonatomic, strong) UIButton *button;
@end

@implementation PresentingViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.button addTarget:self action:@selector(didPressPresent)
forControlEvents:UIControlEventTouchUpInside];
}

- (void)didPressPresent
{
    ModalViewController *modal = [[ModalViewController alloc] init];
    modal.view.frame = CGRectMake(0.0, 0.0, 200.0, 200.0);
    modal.modalPresentationStyle = UIModalPresentationCustom;
    modal.transitioningDelegate = modal;
    [self presentViewController:modal animated:YES completion:nil];
}

- (UIButton *)button
{
    if (!_button)
    {
        _button = [[UIButton alloc] init];
        _button.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_button];
        [_button.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_button.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
        [_button setTitle:@"Present" forState:UIControlStateNormal];
        [_button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal];
    }
    return _button;
}

@end
```

Когда присутствует кнопка прослушиваются, мы создаем `ModalViewController` и установить его стиль презентации для `.custom` и установить его `transitionDelegate` себе. Это позволит нам продавать аниматор, который будет управлять своим модальным переходом. Мы также устанавливаем `modal` рамку вида по «s так будет меньше , чем на всем экране.

Давайте посмотрим на `ModalViewController` :

стриж

```
class ModalViewController: UIViewController
{
```

```
    lazy var button: UIButton =
    {
        let button = UIButton()
        button.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(button)
        button.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive
         = true
        button.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        button.setTitle("Dismiss", for: .normal)
        button.setTitleColor(.white, for: .normal)

        return button
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        button.addTarget(self, action: #selector(self.didPressDismiss), for: .touchUpInside)
        view.backgroundColor = .red
        view.layer.cornerRadius = 15.0
    }

    func didPressDismiss()
    {
        dismiss(animated: true)
    }
}

extension ModalViewController: UIViewControllerTransitioningDelegate
{
    func animationController(forPresented presented: UIViewController, presenting:
UIViewController, source: UIViewController) -> UIViewControllerAnimatedTransitioning?
    {
        return DropOutAnimator(duration: 1.5, isAppearing: true)
    }

    func animationController(forDismissed dismissed: UIViewController) ->
UIViewControllerAnimatedTransitioning?
    {
        return DropOutAnimator(duration: 4.0, isAppearing: false)
    }
}
```

Objective-C

```
@interface ModalViewController () <UIViewControllerTransitioningDelegate>
@property (nonatomic, strong) UIButton *button;
@end

@implementation ModalViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.button addTarget:self action:@selector(didPressPresent)
forControlEvents:UIControlEventTouchUpInside];
    self.view.backgroundColor = [UIColor redColor];
    self.view.layer.cornerRadius = 15.0f;
}
```

```
- (void)didPressPresent
{
    [self dismissViewControllerAnimated:YES completion:nil];
}

- (UIButton *)button
{
    if (!_button)
    {
        _button = [[UIButton alloc] init];
        _button.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_button];
        [_button.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_button.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
        [_button setTitle:@"Dismiss" forState:UIControlStateNormal];
        [_button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal];
    }
    return _button;
}

-
(id<UIViewControllerAnimatedTransitioning>)animationControllerForPresentedController:(UIViewController
*)presented presentingController:(UIViewController *)presenting
sourceController:(UIViewController *)source
{
    return [[DropOutAnimator alloc]initWithDuration: 1.5 appearing:YES];
}

-
(id<UIViewControllerAnimatedTransitioning>)animationControllerForDismissedController:(UIViewController
*)dismissed
{
    return [[DropOutAnimator alloc] initWithDuration:4.0 appearing:NO];
}

@end
```

Здесь мы создаем контроллер представления, который представлен. Кроме того, поскольку `ModalViewController` является собственным `transitioningDelegate` он также отвечает за вендинг объекта, который будет управлять своей переходной анимацией. Для нас это означает передачу экземпляра нашего составного подкласса `UIDynamicBehavior` .

Наш аниматор будет иметь два разных перехода: один для представления и один для увольнения. Для представления представление представления контроллера представления будет снижаться сверху. И для увольнения взгляд, кажется, качается от веревки, а затем выпадает. Поскольку `DropOutAnimator` соответствует `UIViewControllerAnimatedTransitioning` большая часть этой работы будет выполнена при реализации `func animateTransition(using transitionContext: UIViewControllerContextTransitioning)` .

стриж

```
class DropOutAnimator: UIDynamicBehavior
```

```
{
    let duration: TimeInterval
    let isAppearing: Bool

    var transitionContext: UIViewControllerContextTransitioning?
    var hasElapsedTimeExceededDuration = false
    var finishTime: TimeInterval = 0.0
    var collisionBehavior: UICollisionBehavior?
    var attachmentBehavior: UIAttachmentBehavior?
    var animator: UIDynamicAnimator?

    init(duration: TimeInterval = 1.0,  isAppearing: Bool)
    {
        self.duration = duration
        self.isAppearing = isAppearing
        super.init()
    }
}

extension DropOutAnimator: UIViewControllerAnimatedTransitioning
{
    func animateTransition(using transitionContext: UIViewControllerContextTransitioning)
    {
        // Get relevant views and view controllers from transitionContext
        guard let fromVC = transitionContext.viewController(forKey: .from),
              let toVC = transitionContext.viewController(forKey: .to),
              let fromView = fromVC.view,
              let toView = toVC.view else { return }

        let containerView = transitionContext.containerView
        let duration = self.transitionDuration(using: transitionContext)

        // Hold refrence to transitionContext to notify it of completion
        self.transitionContext = transitionContext

        // Create dynamic animator
        let animator = UIDynamicAnimator(referenceView: containerView)
        animator.delegate = self
        self.animator = animator

        // Presenting Animation
        if self.isAppearing
        {
            fromView.isUserInteractionEnabled = false

            // Position toView  just off-screen
            let fromViewInitialFrame = transitionContext.initialFrame(for: fromVC)
            var toViewInitialFrame = toView.frame
            toViewInitialFrame.origin.y -= toViewInitialFrame.height
            toViewInitialFrame.origin.x = fromViewInitialFrame.width * 0.5 -
toViewInitialFrame.width * 0.5
            toView.frame = toViewInitialFrame

            containerView.addSubview(toView)

            // Prevent rotation and adjust bounce
            let bodyBehavior = UIDynamicItemBehavior(items: [toView])
            bodyBehavior.elasticity = 0.7
            bodyBehavior.allowsRotation = false

            // Add gravity at exaggerated magnitude so animation doesn't seem slow
```

```
            let gravityBehavior = UIGravityBehavior(items: [toView])
            gravityBehavior.magnitude = 10.0

            // Set collision bounds to include off-screen view and have collision in center
            // where our final view should come to rest
            let collisionBehavior = UICollisionBehavior(items: [toView])
            let insets = UIEdgeInsets(top: toViewInitialFrame.minY, left: 0.0, bottom:
fromViewInitialFrame.height * 0.5 - toViewInitialFrame.height * 0.5, right: 0.0)
            collisionBehavior.setTranslatesReferenceBoundsIntoBoundary(with: insets)
            self.collisionBehavior = collisionBehavior

            // Keep track of finish time in case we need to end the animator befor the
animator pauses
            self.finishTime = duration + (self.animator?.elapsedTime ?? 0.0)

            // Closure that is called after every "tick" of the animator
            // Check if we exceed duration
            self.action =
            { [weak self] in
                guard let strongSelf = self,
                  (strongSelf.animator?.elapsedTime ?? 0.0) >= strongSelf.finishTime else {
return }
                strongSelf.hasElapsedTimeExceededDuration = true
                strongSelf.animator?.removeBehavior(strongSelf)
            }

            // `DropOutAnimator` is a composit behavior, so add child behaviors to self
            self.addChildBehavior(collisionBehavior)
            self.addChildBehavior(bodyBehavior)
            self.addChildBehavior(gravityBehavior)

            // Add self to dynamic animator
            self.animator?.addBehavior(self)
        }
        // Dismissing Animation
        else
        {
            // Create allow rotation and have a elastic item
            let bodyBehavior = UIDynamicItemBehavior(items: [fromView])
            bodyBehavior.elasticity = 0.8
            bodyBehavior.angularResistance = 5.0
            bodyBehavior.allowsRotation = true

            // Create gravity with exaggerated magnitude
            let gravityBehavior = UIGravityBehavior(items: [fromView])
            gravityBehavior.magnitude = 10.0

            // Collision boundary is set to have a floor just below the bottom of the screen
            let collisionBehavior = UICollisionBehavior(items: [fromView])
            let insets = UIEdgeInsets(top: 0.0, left: -1000, bottom: -225, right: -1000)
            collisionBehavior.setTranslatesReferenceBoundsIntoBoundary(with: insets)
            self.collisionBehavior = collisionBehavior

            // Attachment behavior so view will have effect of hanging from a rope
            let offset = UIOffset(horizontal: 70.0, vertical: fromView.bounds.height * 0.5)
            var anchorPoint = CGPoint(x: fromView.bounds.maxX - 40.0, y: fromView.bounds.minY)
            anchorPoint = containerView.convert(anchorPoint, from: fromView)
            let attachmentBehavior = UIAttachmentBehavior(item: fromView, offsetFromCenter:
offset, attachedToAnchor: anchorPoint)
            attachmentBehavior.frequency = 3.0
            attachmentBehavior.damping = 3.0
```

```
            self.attachmentBehavior = attachmentBehavior

            // `DropOutAnimator` is a composit behavior, so add child behaviors to self
            self.addChildBehavior(collisionBehavior)
            self.addChildBehavior(bodyBehavior)
            self.addChildBehavior(gravityBehavior)
            self.addChildBehavior(attachmentBehavior)

            // Add self to dynamic animator
            self.animator?.addBehavior(self)

            // Animation has two parts part one is hanging from rope.
            // Part two is bouncying off-screen
            // Divide duration in two
            self.finishTime = (2.0 / 3.0) * duration + (self.animator?.elapsedTime ?? 0.0)

             // After every "tick" of animator check if past time limit
            self.action =
            { [weak self] in
                guard let strongSelf = self,
                  (strongSelf.animator?.elapsedTime ?? 0.0) >= strongSelf.finishTime else {
return }
                strongSelf.hasElapsedTimeExceededDuration = true
                strongSelf.animator?.removeBehavior(strongSelf)
            }
        }

    }

    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval
    {
        // Return the duration of the animation
        return self.duration
    }
}

extension DropOutAnimator: UIDynamicAnimatorDelegate
{
    func dynamicAnimatorDidPause(_ animator: UIDynamicAnimator)
    {
      // Animator has reached stasis
      if self.isAppearing
      {
            // Check if we are out of time
            if self.hasElapsedTimeExceededDuration
            {
                // Move to final positions
                let toView = self.transitionContext?.viewController(forKey: .to)?.view
                let containerView = self.transitionContext?.containerView
                toView?.center = containerView?.center ?? .zero
                self.hasElapsedTimeExceededDuration = false
            }

            // Clean up and call completion

self.transitionContext?.completeTransition(!(self.transitionContext?.transitionWasCancelled ??
false))
            self.childBehaviors.forEach { self.removeChildBehavior($0) }
            animator.removeAllBehaviors()
            self.transitionContext = nil
```

```
    }
    else
    {
        if let attachmentBehavior = self.attachmentBehavior
        {
            // If we have an attachment, we are at the end of part one and start part two.
            self.removeChildBehavior(attachmentBehavior)
            self.attachmentBehavior = nil
            animator.addBehavior(self)
            let duration = self.transitionDuration(using: self.transitionContext)
            self.finishTime = 1.0 / 3.0 * duration + animator.elapsedTime
        }
        else
        {
            // Clean up and call completion
            let fromView = self.transitionContext?.viewController(forKey: .from)?.view
            let toView = self.transitionContext?.viewController(forKey: .to)?.view
            fromView?.removeFromSuperview()
            toView?.isUserInteractionEnabled = true

self.transitionContext?.completeTransition(!(self.transitionContext?.transitionWasCancelled ??
false))
            self.childBehaviors.forEach { self.removeChildBehavior($0) }
            animator.removeAllBehaviors()
            self.transitionContext = nil
        }
    }
}
}
```

Objective-C

```
@interface ObjcDropOutAnimator() <UIDynamicAnimatorDelegate,
UIViewControllerAnimatedTransitioning>
@property (nonatomic, strong) id<UIViewControllerContextTransitioning> transitionContext;
@property (nonatomic, strong) UIDynamicAnimator *animator;
@property (nonatomic, assign) NSTimeInterval finishTime;
@property (nonatomic, assign) BOOL elapsedTimeExceededDuration;
@property (nonatomic, assign, getter=isAppearing) BOOL appearing;
@property (nonatomic, assign) NSTimeInterval duration;
@property (nonatomic, strong) UIAttachmentBehavior *attachBehavior;
@property (nonatomic, strong) UICollisionBehavior * collisionBehavior;

@end

@implementation ObjcDropOutAnimator

- (instancetype)initWithDuration:(NSTimeInterval)duration appearing:(BOOL)appearing
{
    self = [super init];
    if (self)
    {
        _duration = duration;
        _appearing = appearing;
    }
    return self;
}

- (void) animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
```

```
{
    // Get relevant views and view controllers from transitionContext
    UIViewController *fromVC = [transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *fromView = fromVC.view;
    UIView *toView = toVC.view;

    UIView *containerView = transitionContext.containerView;
    NSTimeInterval duration = [self transitionDuration:transitionContext];

    // Hold refrence to transitionContext to notify it of completion
    self.transitionContext = transitionContext;

    // Create dynamic animator
    UIDynamicAnimator *animator = [[UIDynamicAnimator
alloc]initWithReferenceView:containerView];
    animator.delegate = self;
    self.animator = animator;

    // Presenting Animation
    if (self.isAppearing)
    {
        fromView.userInteractionEnabled = NO;

        // Position toView  just above screen
        CGRect fromViewInitialFrame = [transitionContext
initialFrameForViewController:fromVC];
        CGRect toViewInitialFrame = toView.frame;
        toViewInitialFrame.origin.y -= CGRectGetHeight(toViewInitialFrame);
        toViewInitialFrame.origin.x = CGRectGetWidth(fromViewInitialFrame) * 0.5 -
CGRectGetWidth(toViewInitialFrame) * 0.5;
        toView.frame = toViewInitialFrame;

        [containerView addSubview:toView];

        // Prevent rotation and adjust bounce
        UIDynamicItemBehavior *bodyBehavior = [[UIDynamicItemBehavior
alloc]initWithItems:@[toView]];
        bodyBehavior.elasticity = 0.7;
        bodyBehavior.allowsRotation = NO;

        // Add gravity at exaggerated magnitude so animation doesn't seem slow
        UIGravityBehavior *gravityBehavior = [[UIGravityBehavior
alloc]initWithItems:@[toView]];
        gravityBehavior.magnitude = 10.0f;

        // Set collision bounds to include off-screen view and have collision floor in center
        // where our final view should come to rest
        UICollisionBehavior *collisionBehavior = [[UICollisionBehavior
alloc]initWithItems:@[toView]];
        UIEdgeInsets insets = UIEdgeInsetsMake(CGRectGetMinY(toViewInitialFrame), 0.0,
CGRectGetHeight(fromViewInitialFrame) * 0.5 - CGRectGetHeight(toViewInitialFrame) * 0.5, 0.0);
        [collisionBehavior setTranslatesReferenceBoundsIntoBoundaryWithInsets:insets];
        self.collisionBehavior = collisionBehavior;

        // Keep track of finish time in case we need to end the animator befor the animator
pauses
        self.finishTime = duration + self.animator.elapsedTime;
```

```
        // Closure that is called after every "tick" of the animator
        // Check if we exceed duration
        __weak ObjcDropOutAnimator *weakSelf = self;
        self.action = ^{
            __strong ObjcDropOutAnimator *strongSelf = weakSelf;
            if (strongSelf)
            {
                if (strongSelf.animator.elapsedTime >= strongSelf.finishTime)
                {
                    strongSelf.elapsedTimeExceededDuration = YES;
                    [strongSelf.animator removeBehavior:strongSelf];
                }
            }
        };

        // `DropOutAnimator` is a composit behavior, so add child behaviors to self
        [self addChildBehavior:collisionBehavior];
        [self addChildBehavior:bodyBehavior];
        [self addChildBehavior:gravityBehavior];

        // Add self to dynamic animator
        [self.animator addBehavior:self];
    }
    // Dismissing Animation
    else
    {
        // Allow rotation and have a elastic item
        UIDynamicItemBehavior *bodyBehavior = [[UIDynamicItemBehavior alloc]
initWithItems:@[fromView]];
        bodyBehavior.elasticity = 0.8;
        bodyBehavior.angularResistance = 5.0;
        bodyBehavior.allowsRotation = YES;

        // Create gravity with exaggerated magnitude
        UIGravityBehavior *gravityBehavior = [[UIGravityBehavior alloc]
initWithItems:@[fromView]];
        gravityBehavior.magnitude = 10.0f;

        // Collision boundary is set to have a floor just below the bottom of the screen
        UICollisionBehavior *collisionBehavior = [[UICollisionBehavior alloc]
initWithItems:@[fromView]];
        UIEdgeInsets insets = UIEdgeInsetsMake(0, -1000, -225, -1000);
        [collisionBehavior setTranslatesReferenceBoundsIntoBoundaryWithInsets:insets];
        self.collisionBehavior = collisionBehavior;

        // Attachment behavior so view will have effect of hanging from a rope
        UIOffset offset = UIOffsetMake(70, -(CGRectGetHeight(fromView.bounds) / 2.0));

        CGPoint anchorPoint = CGPointMake(CGRectGetMaxX(fromView.bounds) - 40,
                                          CGRectGetMinY(fromView.bounds));
        anchorPoint = [containerView convertPoint:anchorPoint fromView:fromView];
        UIAttachmentBehavior *attachBehavior = [[UIAttachmentBehavior alloc]
initWithItem:fromView offsetFromCenter:offset attachedToAnchor:anchorPoint];
        attachBehavior.frequency = 3.0;
        attachBehavior.damping = 0.3;
        attachBehavior.length = 40;
        self.attachBehavior = attachBehavior;

        // `DropOutAnimator` is a composit behavior, so add child behaviors to self
        [self addChildBehavior:collisionBehavior];
        [self addChildBehavior:bodyBehavior];
```

```
        [self addChildBehavior:gravityBehavior];
        [self addChildBehavior:attachBehavior];

        // Add self to dynamic animator
        [self.animator addBehavior:self];

        // Animation has two parts part one is hanging from rope.
        // Part two is bouncying off-screen
        // Divide duration in two
        self.finishTime = (2./3.) * duration + [self.animator elapsedTime];

        // After every "tick" of animator check if past time limit
        __weak ObjcDropOutAnimator *weakSelf = self;
        self.action = ^{
            __strong ObjcDropOutAnimator *strongSelf = weakSelf;
            if (strongSelf)
            {
                if ([strongSelf.animator elapsedTime] >= strongSelf.finishTime)
                {
                    strongSelf.elapsedTimeExceededDuration = YES;
                    [strongSelf.animator removeBehavior:strongSelf];
                }
            }
        };
    }
}

-
(NSTimeInterval)transitionDuration:(id<UIViewControllerContextTransitioning>)transitionContext
{
    return self.duration;
}

- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    // Animator has reached stasis
    if (self.isAppearing)
    {
        // Check if we are out of time
        if (self.elapsedTimeExceededDuration)
        {
            // Move to final positions
            UIView *toView = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey].view;
            UIView *containerView = [self.transitionContext containerView];
            toView.center = containerView.center;
            self.elapsedTimeExceededDuration = NO;
        }

        // Clean up and call completion
        [self.transitionContext completeTransition:![self.transitionContext
transitionWasCancelled]];
        for (UIDynamicBehavior *behavior in self.childBehaviors)
        {
            [self removeChildBehavior:behavior];
        }
        [animator removeAllBehaviors];
        self.transitionContext = nil;
    }
    // Dismissing
    else
```

```
    {
        if (self.attachBehavior)
        {
            // If we have an attachment, we are at the end of part one and start part two.
            [self removeChildBehavior:self.attachBehavior];
            self.attachBehavior = nil;
            [animator addBehavior:self];
            NSTimeInterval duration = [self transitionDuration:self.transitionContext];
            self.finishTime = 1./3. * duration + [animator elapsedTime];
        }
        else
        {
            // Clean up and call completion
            UIView *fromView = [self.transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey].view;
            UIView *toView = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey].view;
            [fromView removeFromSuperview];
            toView.userInteractionEnabled = YES;

            [self.transitionContext completeTransition:![self.transitionContext
transitionWasCancelled]];
            for (UIDynamicBehavior *behavior in self.childBehaviors)
            {
                [self removeChildBehavior:behavior];
            }
            [animator removeAllBehaviors];
            self.transitionContext = nil;
        }
    }
}
```

Поскольку составное поведение, `DropOutAnimator` , может сочетать несколько разных поведений, чтобы выполнять презентацию и отклонение анимаций. `DropOutAnimator` также показывает , как использовать `action` блок из поведения , чтобы проверить расположение его элементов, а также время , прошедшей технику , которая может быть использована для удаления представления , которые перемещаются закадровым или укоротить анимации , которые еще не достигли закупорки.

Для получения дополнительной информации 2013 сессия WWDC «Передовые методы с динамикой UIKit», а также SOLPresentingFun

Переход теней с физикой реального мира с использованием UIDynamicBehaviors

В этом примере показано, как сделать интерактивный переход с физикой реального мира, подобный экрану уведомлений iOS.

Начнем с того, что нам нужен контроллер представления, который будет отображаться в тени. Этот контроллер представления также будет действовать как наш `UIViewControllerTransitioningDelegate` для нашего представленного контроллера представлений и будет предлагать аниматоры для нашего перехода. Поэтому мы создадим экземпляры наших интерактивных аниматоров (один для представления, один для отклонения). Мы также создадим экземпляр контроллера тени, который в этом примере является просто контроллером представления с меткой. Поскольку мы хотим, чтобы один и тот же жест жестко управлял всем взаимодействием, мы передаем ссылки на контроллер представления представления и тень в наши интерактивные аниматоры.

стриж

```
class ViewController: UIViewController
{
    var presentingAnimator: ShadeAnimator!
    var dismissingAnimator: ShadeAnimator!
    let shadeVC = ShadeViewController()

    lazy var label: UILabel =
    {
        let label = UILabel()
        label.textColor = .blue
        label.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(label)
        label.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
        label.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        return label
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
```

```
        label.text = "Swipe Down From Top"
        presentingAnimator = ShadeAnimator(isAppearing: true, presentingVC: self, presentedVC:
shadeVC, transitionDelegate: self)
        dismissingAnimator = ShadeAnimator(isAppearing: false, presentingVC: self,
presentedVC: shadeVC, transitionDelegate: self)
    }
}
extension ViewController: UIViewControllerTransitioningDelegate
{
    func animationController(forPresented presented: UIViewController, presenting:
UIViewController, source: UIViewController) -> UIViewControllerAnimatedTransitioning?
    {
        return EmptyAnimator()
    }

    func animationController(forDismissed dismissed: UIViewController) ->
UIViewControllerAnimatedTransitioning?
    {
        return EmptyAnimator()
    }

    func interactionControllerForPresentation(using animator:
UIViewControllerAnimatedTransitioning) -> UIViewControllerInteractiveTransitioning?
    {
        return presentingAnimator
    }

    func interactionControllerForDismissal(using animator:
UIViewControllerAnimatedTransitioning) -> UIViewControllerInteractiveTransitioning?
    {
        return dismissingAnimator
    }
}
```

Objective-C

```
@interface ObjCViewController () <UIViewControllerTransitioningDelegate>
@property (nonatomic, strong) ShadeAnimator *presentingAnimator;
@property (nonatomic, strong) ShadeAnimator *dismissingAnimator;
@property (nonatomic, strong) UILabel *label;
@property (nonatomic, strong) ShadeViewController *shadeVC;
@end

@implementation ObjCViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.label.text = @"Swipe Down From Top";
    self.shadeVC = [[ShadeViewController alloc] init];
    self.presentingAnimator = [[ShadeAnimator alloc] initWithIsAppearing:YES presentingVC:self
presentedVC:self.shadeVC transitionDelegate:self];
    self.dismissingAnimator = [[ShadeAnimator alloc] initWithIsAppearing:NO presentingVC:self
presentedVC:self.shadeVC transitionDelegate:self];
}

- (UILabel *)label
{
    if (!_label)
```

```
    {
        _label = [[UILabel alloc] init];
        _label.textColor = [UIColor blueColor];
        _label.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_label];
        [_label.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_label.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
    }
    return _label;
}

#pragma mark - UIViewControllerTransitioningDelegate

-
(id<UIViewControllerAnimatedTransitioning>)animationControllerForPresentedController:(UIViewController
*)presented presentingController:(UIViewController *)presenting
sourceController:(UIViewController *)source
{
    return [[EmptyAnimator alloc] init];
}

-
(id<UIViewControllerAnimatedTransitioning>)animationControllerForDismissedController:(UIViewController
*)dismissed
{
    return [[EmptyAnimator alloc] init];
}

-
(id<UIViewControllerInteractiveTransitioning>)interactionControllerForPresentation:(id<UIViewControlle

{
    return self.presentingAnimator;
}

-
(id<UIViewControllerInteractiveTransitioning>)interactionControllerForDismissal:(id<UIViewControllerAn

{
    return self.dismissingAnimator;
}

@end
```

Мы хотим, чтобы только когда-либо хотели представить наш оттенок через интерактивный переход, но из-за того, как работает `UIViewControllerTransitioningDelegate` если мы не вернем обычный контроллер анимации, наш интерактивный контроллер никогда не будет использоваться. Из-за этого мы создаем класс `EmptyAnimator` который соответствует `UIViewControllerAnimatedTransitioning` .

стриж

```
class EmptyAnimator: NSObject
{

}
```

```
extension EmptyAnimator: UIViewControllerAnimatedTransitioning
{
    func animateTransition(using transitionContext: UIViewControllerContextTransitioning)
    {

    }

    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval
    {
        return 0.0
    }
}
```

Objective-C

```
@implementation EmptyAnimator

- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{

}

-
(NSTimeInterval)transitionDuration:(id<UIViewControllerContextTransitioning>)transitionContext
{
    return 0.0;
}

@end
```

Наконец, нам нужно создать `ShadeAnimator` который является подклассом `UIDynamicBehavior` который соответствует `UIViewControllerInteractiveTransitioning` .

стриж

```
class ShadeAnimator: UIDynamicBehavior
{
    // Whether we are presenting or dismissing
    let isAppearing: Bool

    // The view controller that is not the shade
    weak var presentingVC: UIViewController?

    // The view controller that is the shade
    weak var presentedVC: UIViewController?

    // The delegate will vend the animator
    weak var transitionDelegate: UIViewControllerTransitioningDelegate?

    // Feedback generator for haptics on collisions
    let impactFeedbackGenerator = UIImpactFeedbackGenerator(style: .light)

    // The context given to the animator at the start of the transition
    var transitionContext: UIViewControllerContextTransitioning?
```

```
    // Time limit of the dynamic part of the animation
    var finishTime: TimeInterval = 4.0

    // The Pan Gesture that drives the transition. Not using EdgePan because triggers
Notifications screen
    lazy var pan: UIPanGestureRecognizer =
    {
        let pan = UIPanGestureRecognizer(target: self, action:
#selector(self.handlePan(sender:)))
        return pan
    }()

    // The dynamic animator that we add `ShadeAnimator` to
    lazy var animator: UIDynamicAnimator! =
    {
        let animator = UIDynamicAnimator(referenceView: self.transitionContext!.containerView)
        return animator
    }()

    // init with all of our dependencies
    init(isAppearing: Bool, presentingVC: UIViewController, presentedVC: UIViewController,
transitionDelegate: UIViewControllerTransitioningDelegate)
    {
        self.isAppearing = isAppearing
        self.presentingVC = presentingVC
        self.presentedVC = presentedVC
        self.transitionDelegate = transitionDelegate
        super.init()
        self.impactFeedbackGenerator.prepare()

        if isAppearing
        {
            self.presentingVC?.view.addGestureRecognizer(pan)
        }
        else
        {
            self.presentedVC?.view.addGestureRecognizer(pan)
        }

    }

    // Setup and moves shade view controller to just above screen if appearing
    func setupViewsForTransition(with transitionContext: UIViewControllerContextTransitioning)
    {
        // Get relevant views and view controllers from transitionContext
        guard let fromVC = transitionContext.viewController(forKey: .from),
            let toVC = transitionContext.viewController(forKey: .to),
            let toView = toVC.view else { return }

        let containerView = transitionContext.containerView

        // Hold refrence to transitionContext to notify it of completion
        self.transitionContext = transitionContext
        if isAppearing
        {
            // Position toView  just off-screen
            let fromViewInitialFrame = transitionContext.initialFrame(for: fromVC)
            var toViewInitialFrame = toView.frame
            toViewInitialFrame.origin.y -= toViewInitialFrame.height
            toViewInitialFrame.origin.x = fromViewInitialFrame.width * 0.5 -
```

```
toViewInitialFrame.width * 0.5
            toView.frame = toViewInitialFrame

            containerView.addSubview(toView)
        }
        else
        {
            fromVC.view.addGestureRecognizer(pan)
        }
    }

    // Handles the entire interaction from presenting/dismissing to completion
    func handlePan(sender: UIPanGestureRecognizer)
    {
        let location = sender.location(in: transitionContext?.containerView)
        let velocity = sender.velocity(in: transitionContext?.containerView)
        let fromVC = transitionContext?.viewController(forKey: .from)
        let toVC = transitionContext?.viewController(forKey: .to)

        let touchStartHeight: CGFloat = 90.0
        let touchLocationFromBottom: CGFloat = 20.0

        switch sender.state
        {
        case .began:
            let beginLocation = sender.location(in: sender.view)
            if isAppearing
            {
                guard beginLocation.y <= touchStartHeight,
                      let presentedVC = self.presentedVC else { break }
                presentedVC.modalPresentationStyle = .custom
                presentedVC.transitioningDelegate = transitionDelegate
                presentingVC?.present(presentedVC, animated: true)
            }
            else
            {
                guard beginLocation.y >= (sender.view?.frame.height ?? 0.0) - touchStartHeight
else { break }
                presentedVC?.dismiss(animated: true)
            }
        case .changed:
            guard let view = isAppearing ? toVC?.view : fromVC?.view else { return }
            UIView.animate(withDuration: 0.2)
            {
                view.frame.origin.y = location.y - view.bounds.height +
touchLocationFromBottom
            }

            transitionContext?.updateInteractiveTransition(view.frame.maxY / view.frame.height
            )
        case .ended, .cancelled:
            guard let view = isAppearing ? toVC?.view : fromVC?.view else { return }
            let isCancelled = isAppearing ? (velocity.y < 0.5 || view.center.y < 0.0) :
(velocity.y > 0.5 || view.center.y > 0.0)
            addAttachmentBehavior(with: view, isCancelled: isCancelled)
            addCollisionBehavior(with: view)
            addItemBehavior(with: view)

            animator.addBehavior(self)
            animator.delegate = self
```

```
            self.action =
            { [weak self] in
                guard let strongSelf = self else { return }
                if strongSelf.animator.elapsedTime > strongSelf.finishTime
                {
                    strongSelf.animator.removeAllBehaviors()
                }
                else
                {
                    strongSelf.transitionContext?.updateInteractiveTransition(view.frame.maxY
/ view.frame.height
                    )
                }
            }
        default:
            break
        }
    }

    // Add collision behavior that causes bounce when finished
    func addCollisionBehavior(with view: UIView)
    {
        let collisionBehavior = UICollisionBehavior(items: [view])
        let insets = UIEdgeInsets(top: -view.bounds.height, left: 0.0, bottom: 0.0, right:
0.0)
        collisionBehavior.setTranslatesReferenceBoundsIntoBoundary(with: insets)
        collisionBehavior.collisionDelegate = self
        self.addChildBehavior(collisionBehavior)
    }

    // Add attachment behavior that pulls shade either to top or bottom
    func addAttachmentBehavior(with view: UIView, isCancelled: Bool)
    {
        let anchor: CGPoint
        switch (isAppearing, isCancelled)
        {
        case (true, true), (false, false):
            anchor = CGPoint(x: view.center.x, y: -view.frame.height)
        case (true, false), (false, true):
            anchor = CGPoint(x: view.center.x, y: view.frame.height)
        }
        let attachmentBehavior = UIAttachmentBehavior(item: view, attachedToAnchor: anchor)
        attachmentBehavior.damping = 0.1
        attachmentBehavior.frequency = 3.0
        attachmentBehavior.length = 0.5 * view.frame.height
        self.addChildBehavior(attachmentBehavior)
    }

    // Makes view more bouncy
    func addItemBehavior(with view: UIView)
    {
        let itemBehavior = UIDynamicItemBehavior(items: [view])
        itemBehavior.allowsRotation = false
        itemBehavior.elasticity = 0.6
        self.addChildBehavior(itemBehavior)
    }

}
extension ShadeAnimator: UIDynamicAnimatorDelegate
{
    // Determines transition has ended
```

```
    func dynamicAnimatorDidPause(_ animator: UIDynamicAnimator)
    {
        guard let transitionContext = self.transitionContext else { return }
        let fromVC = transitionContext.viewController(forKey: .from)
        let toVC = transitionContext.viewController(forKey: .to)
        guard let view = isAppearing ? toVC?.view : fromVC?.view else { return }
        switch (view.center.y < 0.0, isAppearing)
        {
        case (true, true), (true, false):
            view.removeFromSuperview()
            transitionContext.finishInteractiveTransition()
            transitionContext.completeTransition(!isAppearing)
        case (false, true):
            toVC?.view.frame = transitionContext.finalFrame(for: toVC!)
            transitionContext.finishInteractiveTransition()
            transitionContext.completeTransition(true)
        case (false, false):
            fromVC?.view.frame = transitionContext.initialFrame(for: fromVC!)
            transitionContext.cancelInteractiveTransition()
            transitionContext.completeTransition(false)
        }
        childBehaviors.forEach { removeChildBehavior($0) }
        animator.removeAllBehaviors()
        self.animator = nil
        self.transitionContext = nil
    }
}
extension ShadeAnimator: UICollisionBehaviorDelegate
{
    // Triggers haptics
    func collisionBehavior(_ behavior: UICollisionBehavior, beganContactFor item:
UIDynamicItem, withBoundaryIdentifier identifier: NSCopying?, at p: CGPoint)
    {
        guard p.y > 0.0 else { return }
        impactFeedbackGenerator.impactOccurred()
    }
}
extension ShadeAnimator: UIViewControllerInteractiveTransitioning
{
    // Starts transition
    func startInteractiveTransition(_ transitionContext: UIViewControllerContextTransitioning)
    {
        setupViewsForTransition(with: transitionContext)
    }
}
```

Objective-C

```
@interface ShadeAnimator() <UIDynamicAnimatorDelegate, UICollisionBehaviorDelegate>
@property (nonatomic, assign) BOOL isAppearing;
@property (nonatomic, weak) UIViewController *presentingVC;
@property (nonatomic, weak) UIViewController *presentedVC;
@property (nonatomic, weak) NSObject<UIViewControllerTransitioningDelegate>
*transitionDelegate;
@property (nonatomic, strong) UIImpactFeedbackGenerator *impactFeedbackGenerator;
@property (nonatomic, strong) id<UIViewControllerContextTransitioning> transitionContext;
@property (nonatomic, assign) NSTimeInterval finishTime;
@property (nonatomic, strong) UIPanGestureRecognizer *pan;
@property (nonatomic, strong) UIDynamicAnimator *animator;
```

```
@end

@implementation ShadeAnimator

- (instancetype)initWithIsAppearing:(BOOL)isAppearing presentingVC:(UIViewController
*)presentingVC presentedVC:(UIViewController *)presentedVC
transitionDelegate:(id<UIViewControllerTransitioningDelegate>)transitionDelegate
{
    self = [super init];
    if (self)
    {
        _isAppearing = isAppearing;
        _presentingVC = presentingVC;
        _presentedVC = presentedVC;
        _transitionDelegate = transitionDelegate;
        _impactFeedbackGenerator = [[UIImpactFeedbackGenerator
alloc]initWithStyle:UIImpactFeedbackStyleLight];
        [_impactFeedbackGenerator prepare];
        if (_isAppearing)
        {
            [_presentingVC.view addGestureRecognizer:self.pan];
        }
        else
        {
            [_presentedVC.view addGestureRecognizer:self.pan];
        }
    }
    return self;
}

#pragma mark - Lazy Init
- (UIPanGestureRecognizer *)pan
{
    if (!_pan)
    {
        _pan = [[UIPanGestureRecognizer alloc]initWithTarget:self
action:@selector(handlePan:)];
    }
    return _pan;
}

- (UIDynamicAnimator *)animator
{
    if (!_animator)
    {
        _animator = [[UIDynamicAnimator
alloc]initWithReferenceView:self.transitionContext.containerView];
    }
    return _animator;
}

#pragma mark - Setup
-
(void)setupViewForTransitionWithContext:(id<UIViewControllerContextTransitioning>)transitionContext

{
    UIViewController *fromVC = [transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *toView = toVC.view;
```

```
    UIView *containerView = transitionContext.containerView;
    self.transitionContext = transitionContext;
    if (self.isAppearing)
    {
        CGRect fromViewInitialFrame = [transitionContext
initialFrameForViewController:fromVC];
        CGRect toViewInitialFrame = toView.frame;
        toViewInitialFrame.origin.y -= CGRectGetHeight(toViewInitialFrame);
        toViewInitialFrame.origin.x = CGRectGetWidth(fromViewInitialFrame) * 0.5 -
CGRectGetWidth(toViewInitialFrame) * 0.5;

        [containerView addSubview:toView];
    }
    else
    {
        [fromVC.view addGestureRecognizer:self.pan];
    }
}

#pragma mark - Gesture
- (void)handlePan:(UIPanGestureRecognizer *)sender
{
    CGPoint location = [sender locationInView:self.transitionContext.containerView];
    CGPoint velocity = [sender velocityInView:self.transitionContext.containerView];
    UIViewController *fromVC = [self.transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];

    CGFloat touchStartHeight = 90.0;
    CGFloat touchLocationFromBottom = 20.0;

    if (sender.state == UIGestureRecognizerStateBegan)
    {
        CGPoint beginLocation = [sender locationInView:sender.view];
        if (self.isAppearing)
        {
            if (beginLocation.y <= touchStartHeight)
            {
                self.presentedVC.modalPresentationStyle = UIModalPresentationCustom;
                self.presentedVC.transitioningDelegate = self.transitionDelegate;
                [self.presentingVC presentViewController:self.presentedVC animated:YES
completion:nil];
            }
        }
        else
        {
            if (beginLocation.y >= [sender locationInView:sender.view].y - touchStartHeight)
            {
                [self.presentedVC dismissViewControllerAnimated:true completion:nil];
            }
        }
    }
    else if (sender.state == UIGestureRecognizerStateChanged)
    {
        UIView *view = self.isAppearing ? toVC.view : fromVC.view;
        [UIView animateWithDuration:0.2 animations:^{
            CGRect frame = view.frame;
            frame.origin.y = location.y - CGRectGetHeight(view.bounds) +
touchLocationFromBottom;
            view.frame = frame;
```

```
        }];
        [self.transitionContext updateInteractiveTransition:CGRectGetMaxY(view.frame) /
CGRectGetHeight(view.frame)];
    }
    else if (sender.state == UIGestureRecognizerStateEnded || sender.state ==
UIGestureRecognizerStateCancelled)
    {
        UIView *view = self.isAppearing ? toVC.view : fromVC.view;
        BOOL isCancelled = self.isAppearing ? (velocity.y < 0.5 || view.center.y < 0.0) :
(velocity.y > 0.5 || view.center.y > 0.0);
        [self addAttachmentBehaviorWithView:view isCancelled:isCancelled];
        [self addCollisionBehaviorWithView:view];
        [self addItemBehaviorWithView:view];

        [self.animator addBehavior:self];
        self.animator.delegate = self;

        __weak ShadeAnimator *weakSelf = self;
        self.action =
        ^{
            if (weakSelf.animator.elapsedTime > weakSelf.finishTime)
            {
                [weakSelf.animator removeAllBehaviors];
            }
            else
            {
                [weakSelf.transitionContext
updateInteractiveTransition:CGRectGetMaxY(view.frame) / CGRectGetHeight(view.frame)];
            }
        };
    }
}

#pragma mark - UIViewControllerInteractiveTransitioning
- (void)startInteractiveTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    [self setupViewForTransitionWithContext:transitionContext];
}

#pragma mark - Behaviors
- (void)addCollisionBehaviorWithView:(UIView *)view
{
    UICollisionBehavior *collisionBehavior = [[UICollisionBehavior
alloc]initWithItems:@[view]];
    UIEdgeInsets insets = UIEdgeInsetsMake(-CGRectGetHeight(view.bounds), 0.0, 0.0, 0.0);
    [collisionBehavior setTranslatesReferenceBoundsIntoBoundaryWithInsets:insets];
    collisionBehavior.collisionDelegate = self;
    [self addChildBehavior:collisionBehavior];
}

- (void)addItemBehaviorWithView:(UIView *)view
{
    UIDynamicItemBehavior *itemBehavior = [[UIDynamicItemBehavior
alloc]initWithItems:@[view]];
    itemBehavior.allowsRotation = NO;
    itemBehavior.elasticity = 0.6;
    [self addChildBehavior:itemBehavior];
}

- (void)addAttachmentBehaviorWithView:(UIView *)view isCancelled:(BOOL)isCancelled
{
```

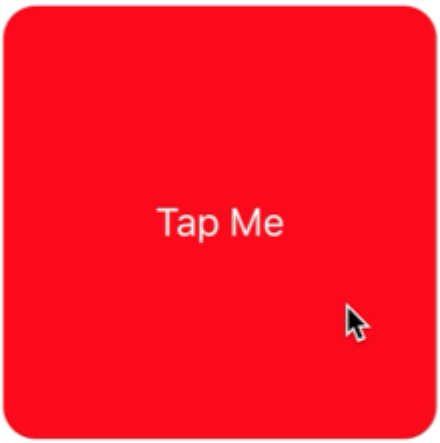
```
    CGPoint anchor;
    if ((self.isAppearing && isCancelled) || (!self.isAppearing && isCancelled))
    {
        anchor = CGPointMake(view.center.x, -CGRectGetHeight(view.frame));
    }
    else
    {
        anchor = CGPointMake(view.center.x, -CGRectGetHeight(view.frame));
    }
    UIAttachmentBehavior *attachmentBehavior = [[UIAttachmentBehavior alloc]initWithItem:view
attachedToAnchor:anchor];
    attachmentBehavior.damping = 0.1;
    attachmentBehavior.frequency = 3.0;
    attachmentBehavior.length = 0.5 * CGRectGetHeight(view.frame);
    [self addChildBehavior:attachmentBehavior];
}

#pragma mark - UICollisionBehaviorDelegate
- (void)collisionBehavior:(UICollisionBehavior *)behavior
beganContactForItem:(id<UIDynamicItem>)item withBoundaryIdentifier:(id<NSCopying>)identifier
atPoint:(CGPoint)p
{
    if (p.y > 0.0)
    {
        [self.impactFeedbackGenerator impactOccurred];
    }
}

#pragma mark - UIDynamicAnimatorDelegate
- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    UIViewController *fromVC = [self.transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *view = self.isAppearing ? toVC.view : fromVC.view;
    if (view.center.y < 0.0 && (self.isAppearing ||  !self.isAppearing))
    {
        [view removeFromSuperview];
        [self.transitionContext finishInteractiveTransition];
        [self.transitionContext completeTransition:!self.isAppearing];
    }
    else if (view.center.y >= 0.0 && self.isAppearing)
    {
        toVC.view.frame = [self.transitionContext finalFrameForViewController:toVC];
        [self.transitionContext finishInteractiveTransition];
        [self.transitionContext completeTransition:YES];
    }
    else
    {
        fromVC.view.frame = [self.transitionContext initialFrameForViewController:fromVC];
        [self.transitionContext cancelInteractiveTransition];
        [self.transitionContext completeTransition:NO];
    }
    for (UIDynamicBehavior *behavior in self.childBehaviors)
    {
        [self removeChildBehavior:behavior];
    }
    [animator removeAllBehaviors];
    self.animator = nil;
    self.transitionContext = nil;
```

```
}

@end
```

Аниматор запускает начало перехода, когда начинается жест панорамы. И просто перемещает представление по мере изменения жестов. Но когда жест заканчивается, когда `UIDynamicBehaviors` определяет, должен ли переход быть завершен или отменен. Для этого используется привязка и поведение при столкновении. Для получения дополнительной информации см. Сессию WWDC 2013 года «Передовые методы с динамикой UIKit» .

Изменения динамической анимации карты в границах

В этом примере показано, как настроить протокол `UIDynamicItem` для сопоставления изменений местоположения динамически анимированного представления с целью изменения ограничений для создания `UIButton` который расширяется и сжимается эластично.



Для начала нам нужно создать новый протокол, который реализует `UIDynamicItem` но также обладает настраиваемым и `UIDynamicItem` `bounds` .

стриж

```
protocol ResizableDynamicItem: UIDynamicItem
{
    var bounds: CGRect { set get }
}
extension UIView: ResizableDynamicItem {}
```

Objective-C

```
@protocol ResizableDynamicItem <UIDynamicItem>
@property (nonatomic, readwrite) CGRect bounds;
@end
```

Затем мы создадим объект-оболочку, который будет обертывать `UIDynamicItem` но будет отображать изменения центра на ширину и высоту элемента. Мы также предоставим переходы для `bounds` и `transform` базового элемента. Это приведет к любым изменениям динамического аниматора в центр. Значения x и y базового элемента будут применены к ширине и высоте элементов.

стриж

```
final class PositionToBoundsMapping: NSObject, UIDynamicItem
{
    var target: ResizableDynamicItem

    init(target: ResizableDynamicItem)
    {
        self.target = target
        super.init()
    }

    var bounds: CGRect
    {
        get
        {
            return self.target.bounds
        }
    }

    var center: CGPoint
    {
        get
        {
            return CGPoint(x: self.target.bounds.width, y: self.target.bounds.height)
        }

        set
        {
            self.target.bounds = CGRect(x: 0.0, y: 0.0, width: newValue.x, height: newValue.y)
        }
    }

    var transform: CGAffineTransform
    {
        get
        {
            return self.target.transform
        }

        set
        {
            self.target.transform = newValue
```

```
        }
    }
}
```

Objective-C

```
@interface PositionToBoundsMapping ()
@property (nonatomic, strong) id<ResizableDynamicItem> target;
@end

@implementation PositionToBoundsMapping

- (instancetype)initWithTarget:(id<ResizableDynamicItem>)target
{
    self = [super init];
    if (self)
    {
        _target = target;
    }
    return self;
}

- (CGRect)bounds
{
    return self.target.bounds;
}

- (CGPoint)center
{
    return CGPointMake(self.target.bounds.size.width, self.target.bounds.size.height);
}

- (void)setCenter:(CGPoint)center
{
    self.target.bounds = CGRectMake(0, 0, center.x, center.y);
}

- (CGAffineTransform)transform
{
    return self.target.transform;
}

- (void)setTransform:(CGAffineTransform)transform
{
    self.target.transform = transform;
}

@end
```

Наконец, мы создадим `UIViewController` , у которого будет кнопка. Когда кнопка нажата, мы создадим `PositionToBoundsMapping` с помощью кнопки в качестве обернутого динамического элемента. Мы создаем `UIAttachmentBehavior` для текущей позиции, а затем добавляем к нему мгновенное `UIPushBehavior` . Однако, поскольку мы отобразили изменения своих границ, кнопка не перемещается, а скорее растет и сжимается.

стриж

```
final class ViewController: UIViewController
{
    lazy var button: UIButton =
    {
        let button = UIButton(frame: CGRect(x: 0.0, y: 0.0, width: 300.0, height: 200.0))
        button.backgroundColor = .red
        button.layer.cornerRadius = 15.0
        button.setTitle("Tap Me", for: .normal)
        self.view.addSubview(button)
        return button
    }()

    var buttonBounds = CGRect.zero
    var animator: UIDynamicAnimator?

    override func viewDidLoad()
    {
        super.viewDidLoad()
        view.backgroundColor = .white
        button.addTarget(self, action: #selector(self.didPressButton(sender:)), for:
.touchUpInside)
        buttonBounds = button.bounds
    }

    override func viewDidLayoutSubviews()
    {
        super.viewDidLayoutSubviews()
        button.center = view.center
    }

    func didPressButton(sender: UIButton)
    {
        // Reset bounds so if button is press twice in a row, previous changes don't propogate
        button.bounds = buttonBounds
        let animator = UIDynamicAnimator(referenceView: view)

        // Create mapping
        let buttonBoundsDynamicItem = PositionToBoundsMapping(target: button)

        // Add Attachment behavior
        let attachmentBehavior = UIAttachmentBehavior(item: buttonBoundsDynamicItem,
attachedToAnchor: buttonBoundsDynamicItem.center)

        // Higher frequency faster oscillation
        attachmentBehavior.frequency = 2.0

        // Lower damping longer oscillation lasts
        attachmentBehavior.damping = 0.1
        animator.addBehavior(attachmentBehavior)

        let pushBehavior = UIPushBehavior(items: [buttonBoundsDynamicItem], mode:
.instantaneous)

        // Change angle to determine how much height/ width should change 45° means
heigh:width is 1:1
        pushBehavior.angle = .pi / 4.0
```

```
        // Larger magnitude means bigger change
        pushBehavior.magnitude = 30.0
        animator.addBehavior(pushBehavior)
        pushBehavior.active = true

        // Hold refrence so animator is not released
        self.animator = animator
    }
}
```

Objective-C

```
@interface ViewController ()
@property (nonatomic, strong) UIButton *button;
@property (nonatomic, assign) CGRect buttonBounds;
@property (nonatomic, strong) UIDynamicAnimator *animator;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.view.backgroundColor = [UIColor whiteColor];
    [self.button addTarget:self action:@selector(didTapButton:)
forControlEvents:UIControlEventTouchUpInside];
    self.buttonBounds = self.button.bounds;
}

- (void)viewDidLayoutSubviews
{
    [super viewDidLayoutSubviews];
    self.button.center = self.view.center;
}

- (UIButton *)button
{
    if (!_button)
    {
        _button = [[UIButton alloc]initWithFrame:CGRectMake(0.0, 0.0, 200.0, 200.0)];
        _button.backgroundColor = [UIColor redColor];
        _button.layer.cornerRadius = 15.0;
        [_button setTitle:@"Tap Me" forState:UIControlStateNormal];
        [self.view addSubview:_button];
    }
    return _button;
}

- (void)didTapButton:(id)sender
{
    self.button.bounds = self.buttonBounds;
    UIDynamicAnimator *animator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
    PositionToBoundsMapping *buttonBoundsDynamicItem = [[PositionToBoundsMapping
alloc]initWithTarget:sender];
    UIAttachmentBehavior *attachmentBehavior = [[UIAttachmentBehavior
alloc]initWithItem:buttonBoundsDynamicItem attachedToAnchor:buttonBoundsDynamicItem.center];
    [attachmentBehavior setFrequency:2.0];
    [attachmentBehavior setDamping:0.3];
    [animator addBehavior:attachmentBehavior];
```

```
    UIPushBehavior *pushBehavior = [[UIPushBehavior alloc]
initWithItems:@[buttonBoundsDynamicItem] mode:UIPushBehaviorModeInstantaneous];
    pushBehavior.angle = M_PI_4;
    pushBehavior.magnitude = 2.0;
    [animator addBehavior:pushBehavior];

    [pushBehavior setActive:TRUE];

    self.animator = animator;
}

@end
```

Для получения дополнительной информации см. Каталог UIKit Dynamics

Прочитайте Динамика UIKit онлайн: https://riptutorial.com/ru/ios/topic/9479/динамика-uikit

глава 129: Динамика UIKit с UICollectionView

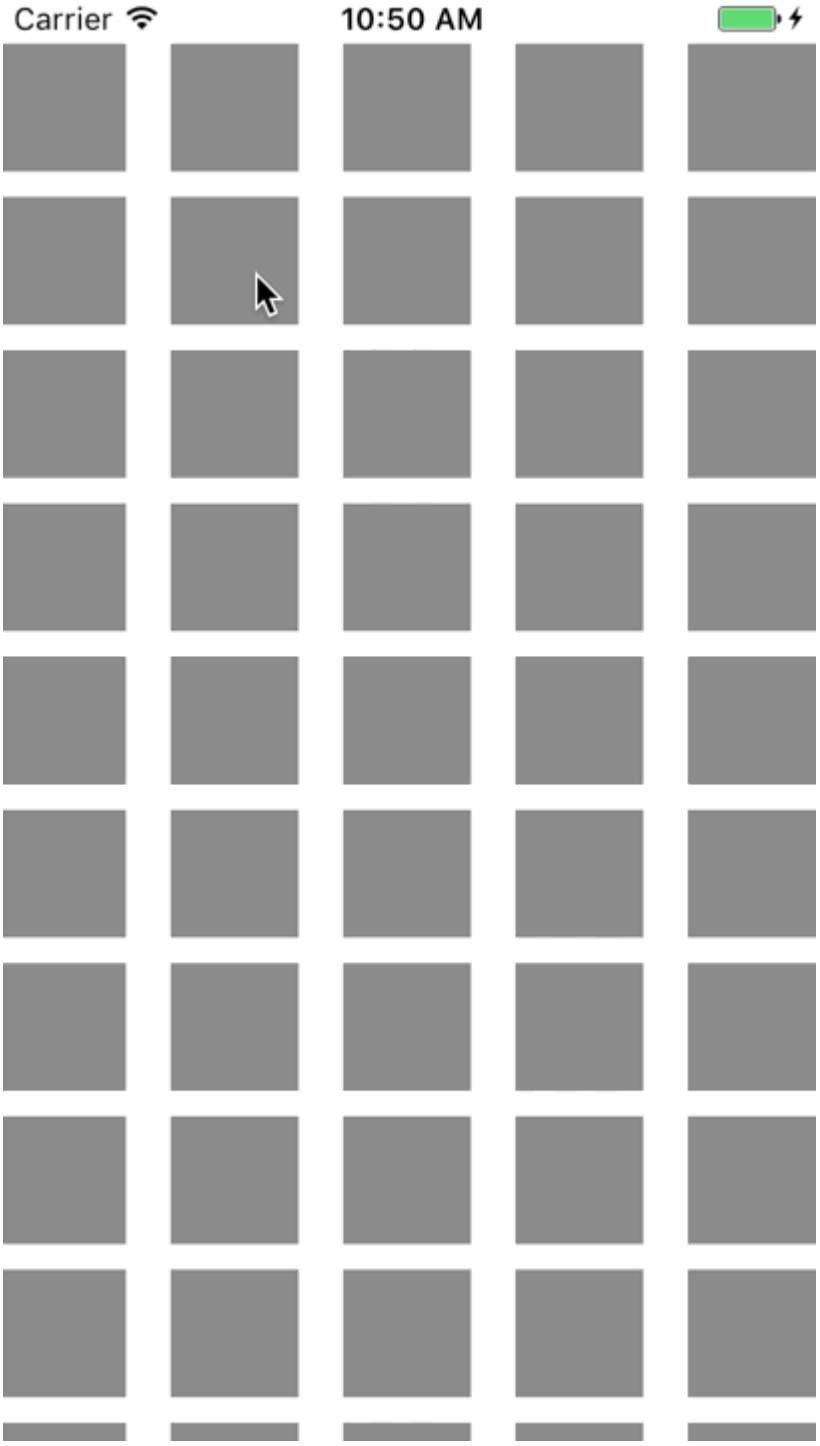
Вступление

UIKit Dynamics - это физический движок, интегрированный в UIKit. UIKit Dynamics предлагает набор API, который обеспечивает совместимость с `UICollectionView` и `UICollectionViewLayout`

Examples

Создание пользовательского поведения перетаскивания с помощью UIDynamicAnimator

В этом примере показано, как создать пользовательское поведение перетаскивания путем подкласса `UIDynamicBehavior` и подкласса `UICollectionViewFlowLayout` . В примере у нас есть `UICollectionView` который позволяет выбирать несколько элементов. Затем с длинным жестом нажатия эти предметы можно перетащить в упругую «пружинную» анимацию, управляемую `UIDynamicAnimator` .



Поведение перетаскивания создается путем объединения поведения низкого уровня, которое добавляет `UIAttachmentBehavior` к углам `UIDynamicItem` и поведения на высоком уровне, который управляет поведением на низком уровне для нескольких `UIDynamicItems` .

Мы можем начать с создания этого поведения на низком уровне, мы будем называть `RectangleAttachmentBehavior`

стриж

```
final class RectangleAttachmentBehavior: UIDynamicBehavior
{
    init(item: UIDynamicItem, point: CGPoint)
```

```
    {
        // Higher frequency more "ridged" formation
        let frequency: CGFloat = 8.0

        // Lower damping longer animation takes to come to rest
        let damping: CGFloat = 0.6

        super.init()

        // Attachment points are four corners of item
        let points = self.attachmentPoints(for: point)

        let attachmentBehaviors: [UIAttachmentBehavior] = points.map
        {
            let attachmentBehavior = UIAttachmentBehavior(item: item, attachedToAnchor: $0)
            attachmentBehavior.frequency = frequency
            attachmentBehavior.damping = damping
            return attachmentBehavior
        }

        attachmentBehaviors.forEach
        {
            addChildBehavior($0)
        }
    }

    func updateAttachmentLocation(with point: CGPoint)
    {
        // Update anchor points to new attachment points
        let points = self.attachmentPoints(for: point)
        let attachments = self.childBehaviors.flatMap { $0 as? UIAttachmentBehavior }
        let pairs = zip(points, attachments)
        pairs.forEach { $0.1.anchorPoint = $0.0 }
    }

    func attachmentPoints(for point: CGPoint) -> [CGPoint]
    {
        // Width and height should be close to the width and height of the item
        let width: CGFloat = 40.0
        let height: CGFloat = 40.0

        let topLeft = CGPoint(x: point.x - width * 0.5, y: point.y - height * 0.5)
        let topRight = CGPoint(x: point.x + width * 0.5, y: point.y - height * 0.5)
        let bottomLeft = CGPoint(x: point.x - width * 0.5, y: point.y + height * 0.5)
        let bottomRight = CGPoint(x: point.x + width * 0.5, y: point.y + height * 0.5)
        let points = [topLeft, topRight, bottomLeft, bottomRight]
        return points
    }
}
```

Objective-C

```
@implementation RectangleAttachmentBehavior

- (instancetype)initWithItem:(id<UIDynamicItem>)item point:(CGPoint)point
{
    CGFloat frequency = 8.0f;
    CGFloat damping = 0.6f;
    self = [super init];
```

```
    if (self)
    {
        NSArray <NSValue *> *pointValues = [self attachmentPointValuesForPoint:point];
        for (NSValue *value in pointValues)
        {
            UIAttachmentBehavior *attachment = [[UIAttachmentBehavior alloc]initWithItem:item
attachedToAnchor:[value CGPointValue]];
            attachment.frequency = frequency;
            attachment.damping = damping;
            [self addChildBehavior:attachment];
        }
    }
    return self;
}

- (void)updateAttachmentLocationWithPoint:(CGPoint)point
{
    NSArray <NSValue *> *pointValues = [self attachmentPointValuesForPoint:point];
    for (NSInteger i = 0; i < pointValues.count; i++)
    {
        NSValue *pointValue = pointValues[i];
        UIAttachmentBehavior *attachment = self.childBehaviors[i];
        attachment.anchorPoint = [pointValue CGPointValue];
    }
}

- (NSArray <NSValue *> *)attachmentPointValuesForPoint:(CGPoint)point
{
    CGFloat width = 40.0f;
    CGFloat height = 40.0f;

    CGPoint topLeft = CGPointMake(point.x - width * 0.5, point.y - height * 0.5);
    CGPoint topRight = CGPointMake(point.x + width * 0.5, point.y - height * 0.5);
    CGPoint bottomLeft = CGPointMake(point.x - width * 0.5, point.y + height * 0.5);
    CGPoint bottomRight = CGPointMake(point.x + width * 0.5, point.y + height * 0.5);

    NSArray <NSValue *> *pointValues = @[[NSValue valueWithCGPoint:topLeft], [NSValue
valueWithCGPoint:topRight], [NSValue valueWithCGPoint:bottomLeft], [NSValue
valueWithCGPoint:bottomRight]];
    return pointValues;
}

@end
```

Затем мы можем создать поведение на высоком уровне, которое объединит ряд `RectangleAttachmentBehavior` .

стриж

```
final class DragBehavior: UIDynamicBehavior
{
    init(items: [UIDynamicItem], point: CGPoint)
    {
        super.init()
        items.forEach
        {
            let rectAttachment = RectangleAttachmentBehavior(item: $0, point: point)
            self.addChildBehavior(rectAttachment)
```

```
        }
    }

    func updateDragLocation(with point: CGPoint)
    {
        // Tell low-level behaviors location has changed
        self.childBehaviors.flatMap { $0 as? RectangleAttachmentBehavior }.forEach {
$0.updateAttachmentLocation(with: point) }
    }
}
```

Objective-C

```
@implementation DragBehavior

- (instancetype)initWithItems:(NSArray <id<UIDynamicItem>> *)items point: (CGPoint)point
{
    self = [super init];
    if (self)
    {
        for (id<UIDynamicItem> item in items)
        {
            RectangleAttachmentBehavior *rectAttachment = [[RectangleAttachmentBehavior
alloc]initWithItem:item point:point];
            [self addChildBehavior:rectAttachment];
        }
    }
    return self;
}

- (void)updateDragLocationWithPoint:(CGPoint)point
{
    for (RectangleAttachmentBehavior *rectAttachment in self.childBehaviors)
    {
        [rectAttachment updateAttachmentLocationWithPoint:point];
    }
}

@end
```

Теперь с нашим поведением на месте, следующий шаг - добавить их в наш просмотр коллекции, когда. Поскольку обычно мы хотим стандартную схему сетки, мы можем подклассифицировать `UICollectionViewFlowLayout` и изменять атрибуты при перетаскивании. Мы делаем это главным образом за счет переопределения `layoutAttributesForElementsInRect` и использования `UIDynamicAnimator's` удобства `itemsInRect` .

стриж

```
final class DraggableLayout: UICollectionViewFlowLayout
{
    // Array that holds dragged index paths
    var indexPathsForDraggingElements: [IndexPath]?

    // The dynamic animator that will animate drag behavior
```

```
    var animator: UIDynamicAnimator?

    // Custom high-level behavior that dictates drag animation
    var dragBehavior: DragBehavior?

    // Where dragging starts so can return there once dragging ends
    var startDragPoint = CGPoint.zero

    // Bool to keep track if dragging has ended
    var isFinishedDragging = false


    // Method to inform layout that dragging has started
    func startDragging(indexPaths selectedIndexPaths: [IndexPath], from point: CGPoint)
    {
        indexPathsForDraggingElements = selectedIndexPaths
        animator = UIDynamicAnimator(collectionViewLayout: self)
        animator?.delegate = self

        // Get all of the draggable attributes but change zIndex so above other cells
        let draggableAttributes: [UICollectionViewLayoutAttributes] =
selectedIndexPaths.flatMap {
            let attribute = super.layoutAttributesForItem(at: $0)
            attribute?.zIndex = 1
            return attribute
        }

        startDragPoint = point

        // Add them to high-level behavior
        dragBehavior = DragBehavior(items: draggableAttributes, point: point)

        // Add high-level behavior to animator
        animator?.addBehavior(dragBehavior!)
    }

    func updateDragLocation(_ point: CGPoint)
    {
        // Tell high-level behavior that point has updated
        dragBehavior?.updateDragLocation(with: point)
    }

    func endDragging()
    {
        isFinishedDragging = true

        // Return high-level behavior to starting point
        dragBehavior?.updateDragLocation(with: startDragPoint)
    }

    func clearDraggedIndexPaths()
    {
        // Reset state for next drag event
        animator = nil
        indexPathsForDraggingElements = nil
        isFinishedDragging = false
    }

    override func layoutAttributesForElements(in rect: CGRect) ->
[UICollectionViewLayoutAttributes]?
    {
```

```
        let existingAttributes: [UICollectionViewLayoutAttributes] =
super.layoutAttributesForElements(in: rect) ?? []
        var allAttributes = [UICollectionViewLayoutAttributes]()

        // Get normal flow layout attributes for non-drag items
        for attributes in existingAttributes
        {
            if (indexPathsForDraggingElements?.contains(attributes.indexPath) ?? false) ==
false
            {
                allAttributes.append(attributes)
            }
        }

        // Add dragged item attributes by asking animator for them
        if let animator = self.animator
        {
            let animatorAttributes: [UICollectionViewLayoutAttributes] = animator.items(in:
rect).flatMap { $0 as? UICollectionViewLayoutAttributes }
            allAttributes.append(contentsOf: animatorAttributes)
        }
        return allAttributes
    }
}
extension DraggableLayout: UIDynamicAnimatorDelegate
{
    func dynamicAnimatorDidPause(_ animator: UIDynamicAnimator)
    {
        // Animator has paused and done dragging; reset state
        guard isFinishedDragging else { return }
        clearDraggedIndexPaths()
    }
}
```

Objective-C

```
@interface DraggableLayout () <UIDynamicAnimatorDelegate>
@property (nonatomic, strong) NSArray <NSIndexPath *> *indexPathsForDraggingElements;
@property (nonatomic, strong) UIDynamicAnimator *animator;
@property (nonatomic, assign) CGPoint startDragPoint;
@property (nonatomic, assign) BOOL finishedDragging;
@property (nonatomic, strong) DragBehavior *dragBehavior;
@end

@implementation DraggableLayout

- (void)startDraggingWithIndexPaths:(NSArray <NSIndexPath *> *)selectedIndexPaths
fromPoint:(CGPoint)point
{
    self.indexPathsForDraggingElements = selectedIndexPaths;
    self.animator = [[UIDynamicAnimator alloc]initWithCollectionViewLayout:self];
    self.animator.delegate = self;
    NSMutableArray *draggableAttributes = [[NSMutableArray
alloc]initWithCapacity:selectedIndexPaths.count];
    for (NSIndexPath *indexPath in selectedIndexPaths)
    {
        UICollectionViewLayoutAttributes *attributes = [super
layoutAttributesForItemAtIndexPath:indexPath];
        attributes.zIndex = 1;
```

```
        [draggableAttributes addObject:attributes];
    }
    self.startDragPoint = point;
    self.dragBehavior = [[DragBehavior alloc]initWithItems:draggableAttributes point:point];
    [self.animator addBehavior:self.dragBehavior];
}

- (void)updateDragLoactionWithPoint:(CGPoint)point
{
    [self.dragBehavior updateDragLocationWithPoint:point];
}

- (void)endDragging
{
    self.finishedDragging = YES;
    [self.dragBehavior updateDragLocationWithPoint:self.startDragPoint];
}

- (void)clearDraggedIndexPath
{
    self.animator = nil;
    self.indexPathsForDraggingElements = nil;
    self.finishedDragging = NO;
}

- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    if (self.finishedDragging)
    {
        [self clearDraggedIndexPath];
    }
}

- (NSArray<UICollectionViewLayoutAttributes *>
*)layoutAttributesForElementsInRect:(CGRect)rect
{
    NSArray *existingAttributes = [super layoutAttributesForElementsInRect:rect];
    NSMutableArray *allAttributes = [[NSMutableArray
alloc]initWithCapacity:existingAttributes.count];
    for (UICollectionViewLayoutAttributes *attributes in existingAttributes)
    {
        if (![self.indexPathsForDraggingElements containsObject:attributes.indexPath])
        {
            [allAttributes addObject:attributes];
        }
    }
    [allAttributes addObjectsFromArray:[self.animator itemsInRect:rect]];
    return allAttributes;
}

@end
```

Наконец, мы создадим контроллер вида, который создаст наш `UICollectionView` и обработает наш длинный жест прессы.

стриж

```
final class ViewController: UIViewController
```

```
{
    // Collection view that displays cells
    lazy var collectionView: UICollectionView =
    {
        let collectionView = UICollectionView(frame: .zero, collectionViewLayout:
DraggableLayout())
        collectionView.backgroundColor = .white
        collectionView.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(collectionView)
        collectionView.topAnchor.constraint(equalTo:
self.topLayoutGuide.bottomAnchor).isActive = true
        collectionView.leadingAnchor.constraint(equalTo: self.view.leadingAnchor).isActive =
true
        collectionView.trailingAnchor.constraint(equalTo: self.view.trailingAnchor).isActive =
true
        collectionView.bottomAnchor.constraint(equalTo:
self.bottomLayoutGuide.topAnchor).isActive = true

        return collectionView
    }()

    // Gesture that drives dragging
    lazy var longPress: UILongPressGestureRecognizer =
    {
        let longPress = UILongPressGestureRecognizer(target: self, action:
#selector(self.handleLongPress(sender:)))
        return longPress
    }()

    // Array that holds selected index paths
    var selectedIndexPaths = [IndexPath]()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        collectionView.delegate = self
        collectionView.dataSource = self
        collectionView.register(UICollectionViewCell.self, forCellWithReuseIdentifier: "Cell")
        collectionView.addGestureRecognizer(longPress)
    }

    func handleLongPress(sender: UILongPressGestureRecognizer)
    {
        guard let draggableLayout = collectionView.collectionViewLayout as? DraggableLayout
else { return }
        let location = sender.location(in: collectionView)
        switch sender.state
        {
        case .began:
            draggableLayout.startDragging(indexPaths: selectedIndexPaths, from: location)
        case .changed:
            draggableLayout.updateDragLocation(location)
        case .ended, .failed, .cancelled:
            draggableLayout.endDragging()
        case .possible:
            break
        }
    }
}
extension ViewController: UICollectionViewDelegate, UICollectionViewDataSource
{
```

```
    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section:
Int) -> Int
    {
        return 1000
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath:
IndexPath) -> UICollectionViewCell
    {
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "Cell", for:
indexPath)
        cell.backgroundColor = .gray
        if selectedIndexPaths.contains(indexPath) == true
        {
            cell.backgroundColor = .red
        }
        return cell
    }

    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath:
IndexPath)
    {
        // Bool that determines if cell is being selected or unselected
        let isSelected = !selectedIndexPaths.contains(indexPath)
        let cell = collectionView.cellForItem(at: indexPath)
        cell?.backgroundColor = isSelected ? .red : .gray
        if isSelected
        {
            selectedIndexPaths.append(indexPath)
        }
        else
        {
            selectedIndexPaths.remove(at: selectedIndexPaths.index(of: indexPath)!)
        }
    }
}
```

Objective-C

```
@interface ViewController () <UICollectionViewDelegate, UICollectionViewDataSource>
@property (nonatomic, strong) UICollectionView *collectionView;
@property (nonatomic, strong) UILongPressGestureRecognizer *longPress;
@property (nonatomic, strong) NSMutableArray <NSIndexPath *> *selectedIndexPaths;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.collectionView.delegate = self;
    self.collectionView.dataSource = self;
    [self.collectionView registerClass:[UICollectionViewCell class]
forCellWithReuseIdentifier:@"Cell"];
    [self.collectionView addGestureRecognizer:self.longPress];
    self.selectedIndexPaths = [[NSMutableArray alloc]init];
}

- (UICollectionView *)collectionView
```

```
{
    if (!_collectionView)
    {
        _collectionView = [[UICollectionView alloc]initWithFrame:CGRectZero
collectionViewLayout:[[DraggableLayout alloc]init]];
        _collectionView.backgroundColor = [UIColor whiteColor];
        _collectionView.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_collectionView];
        [_collectionView.topAnchor
constraintEqualToAnchor:self.topLayoutGuide.bottomAnchor].active = YES;
        [_collectionView.leadingAnchor constraintEqualToAnchor:self.view.leadingAnchor].active
= YES;
        [_collectionView.trailingAnchor
constraintEqualToAnchor:self.view.trailingAnchor].active = YES;
        [_collectionView.bottomAnchor
constraintEqualToAnchor:self.bottomLayoutGuide.topAnchor].active = YES;
    }
    return _collectionView;
}

- (UILongPressGestureRecognizer *)longPress
{
    if (!_longPress)
    {
        _longPress = [[UILongPressGestureRecognizer alloc]initWithTarget:self
action:@selector(handleLongPress:)];
    }
    return _longPress;
}

- (void)handleLongPress:(UILongPressGestureRecognizer *)sender
{
    DraggableLayout *draggableLayout = (DraggableLayout
*)self.collectionView.collectionViewLayout;
    CGPoint location = [sender locationInView:self.collectionView];
    if (sender.state == UIGestureRecognizerStateBegan)
    {
        [draggableLayout startDraggingWithIndexPaths:self.selectedIndexPaths
fromPoint:location];
    }
    else if(sender.state == UIGestureRecognizerStateChanged)
    {
        [draggableLayout updateDragLoactionWithPoint:location];
    }
    else if(sender.state == UIGestureRecognizerStateEnded ||  sender.state ==
UIGestureRecognizerStateCancelled || sender.state == UIGestureRecognizerStateFailed)
    {
        [draggableLayout endDragging];
    }
}

- (NSInteger)collectionView:(UICollectionView *)collectionView
numberOfItemsInSection:(NSInteger)section
{
    return 1000;
}

- (UICollectionViewCell *)collectionView:(UICollectionView *)collectionView
cellForItemAtIndexPath:(NSIndexPath *)indexPath
{
    UICollectionViewCell *cell = [collectionView
```

```
dequeueReusableCellWithReuseIdentifier:@"Cell" forIndexPath:indexPath];
    cell.backgroundColor = [UIColor grayColor];
    if ([self.selectedIndexPaths containsObject:indexPath])
    {
        cell.backgroundColor = [UIColor redColor];
    }
    return cell;
}

- (void)collectionView:(UICollectionView *)collectionView
didSelectItemAtIndexPath:(NSIndexPath *)indexPath
{
    BOOL isSelected = ![self.selectedIndexPaths containsObject:indexPath];
    UICollectionViewCell *cell = [collectionView cellForItemAtIndexPath:indexPath];
    if (isSelected)
    {
        cell.backgroundColor = [UIColor redColor];
        [self.selectedIndexPaths addObject:indexPath];
    }
    else
    {
        cell.backgroundColor = [UIColor grayColor];
        [self.selectedIndexPaths removeObject:indexPath];
    }
}

@end
```

Для получения дополнительной информации 2013 сессия WWDC «Передовые методы с динамикой UIKit»

Прочитайте Динамика UIKit с UICollectionView онлайн:
https://riptutorial.com/ru/ios/topic/10079/динамика-uikit-c-uicollectionview

глава 130: Динамический тип

замечания

```
// Content size category constants
UIContentSizeCategoryExtraSmall
UIContentSizeCategorySmall
UIContentSizeCategoryMedium
UIContentSizeCategoryLarge
UIContentSizeCategoryExtraLarge
UIContentSizeCategoryExtraExtraLarge
UIContentSizeCategoryExtraExtraExtraLarge

// Accessibility sizes
UIContentSizeCategoryAccessibilityMedium
UIContentSizeCategoryAccessibilityLarge
UIContentSizeCategoryAccessibilityExtraLarge
UIContentSizeCategoryAccessibilityExtraExtraLarge
UIContentSizeCategoryAccessibilityExtraExtraExtraLarge
```

Examples

Получить текущий размер контента

стриж

```
UIApplication.sharedApplication().preferredContentSizeCategory
```

Objective-C

```
[UIApplication sharedApplication].preferredContentSizeCategory;
```

Это возвращает константу категории размера контента или константу категории содержимого доступности.

Уведомление об изменении размера текста

Вы можете зарегистрироваться для уведомлений о том, когда изменяется размер текста устройства.

стриж

```
NSNotificationCenter.defaultCenter().addObserver(self, selector: #selector(updateFont), name:
name:UIContentSizeCategoryDidChangeNotification, object: nil)
```

Objective-C

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(updateFont)
name:UIContentSizeCategoryDidChangeNotification object:nil];
```

Объект `userInfo` оповещения содержит новый размер в `UIContentSizeCategoryNewValueKey` .

Соответствие размера шрифта динамического шрифта в WKWebView

WKWebView изменяет размеры шрифтов на веб-контенте, чтобы полноразмерная веб-страница соответствовала форм-фактору устройства. Если вы хотите, чтобы веб-текст как в портретном, так и в ландшафтном формате был похож по размеру на предпочтительный размер чтения пользователя, вам нужно явно установить его.

стриж

```
// build HTML header for dynamic type and responsive design
func buildHTMLHeader() -> String {

    // Get preferred dynamic type font sizes for html styles
    let bodySize = UIFont.preferredFont(forTextStyle: UIFontTextStyle.body).pointSize
    let h1Size = UIFont.preferredFont(forTextStyle: UIFontTextStyle.title1).pointSize
    let h2Size = UIFont.preferredFont(forTextStyle: UIFontTextStyle.title2).pointSize
    let h3Size = UIFont.preferredFont(forTextStyle: UIFontTextStyle.title3).pointSize

    // On iPad, landscape text is larger than preferred font size
    var portraitMultiplier = CGFloat(1.0)
    var landscapeMultiplier = CGFloat(0.5)

    // iPhone text is shrunken
    if UIDevice.current.model.range(of: "iPhone") != nil {
        portraitMultiplier = CGFloat(3.0)
        landscapeMultiplier = CGFloat(1.5)
    }

    // Start HTML header text
    let patternText = "<html> <head> <style> "

    // Match Dynamic Type for this page.
    + "body { background-color: \(backgroundColor);} "
    + "@media all and (orientation:portrait) {img {max-width: 90%; height: auto;} "
    + "p, li { font: -apple-system-body; font-family: Georgia, serif; font-
size:calc(\(bodySize * portraitMultiplier)px + 1.0vw); font-weight: normal; color:
\(fontColor) } "
    + "h1 { font: -apple-system-headine; font-family: Verdana, sans-serif; font-
size:calc(\(h1Size * portraitMultiplier)px + 1.0vw); font-weight: bold; color:
\(headFontColor) } "
    + "h2 { font: -apple-system-headine; font-family: Verdana, sans-serif; font-
```

```
size:calc(\(h2Size * portraitMultiplier)px + 1.0vw); font-weight: bold; color:
\(headFontColor) } "
    + "h3, h4 { font: -apple-system-headine; font-family: Verdana, sans-serif; font-
size:calc(\(h3Size * portraitMultiplier)px + 1.0vw); font-weight: bold; color:
\(headFontColor) } } "
    + "@media all and (orientation:landscape) {img {max-width: 65%; height: auto;}"
    + "p, li { font: -apple-system-body; font-family: Georgia, serif; font-
size:calc(\(bodySize * landscapeMultiplier)px + 1.0vw); font-weight: normal; color:
\(fontColor) }"
    + "h1 { font: -apple-system-headine; font-family: Verdana, sans-serif; font-
size:calc(\(h1Size * landscapeMultiplier)px + 1.0vw); font-weight: bold; color:
\(headFontColor) } "
    + "h2 { font: -apple-system-headine; font-family: Verdana, sans-serif; font-
size:calc(\(h2Size * landscapeMultiplier)px + 1.0vw); font-weight: bold; color:
\(headFontColor) } "
    + "h3, h4 { font: -apple-system-headine; font-family: Verdana, sans-serif; font-
size:calc(\(h3Size * landscapeMultiplier)px + 1.0vw); font-weight: bold; color:
\(headFontColor) } } </style>"
    + "</head><body>"
    + "<meta name=\"viewport\" content=\"width: device-width\">"

    return patternText
}
```

Изменение размера предпочтительного размера текста без уведомлений на iOS 10

`UILabel` , `UITextField` и `UITextView` классы имеют новое свойство, начиная с iOS 10, для автоматического изменения размера шрифта, когда пользователь меняет свой предпочтительный размер чтения под именем `adjustsFontForContentSizeCategory` .

стриж

```
@IBOutlet var label:UILabel!

if #available(iOS 10.0, *) {
    label.adjustsFontForContentSizeCategory = true
} else {
    // Observe for UIContentSizeCategoryDidChangeNotification and handle it manually
    // since the adjustsFontForContentSizeCategory property isn't available.
}
```

Прочитайте Динамический тип онлайн: https://riptutorial.com/ru/ios/topic/4466/динамический-тип

глава 131: Динамическое обновление UIStackView

Examples

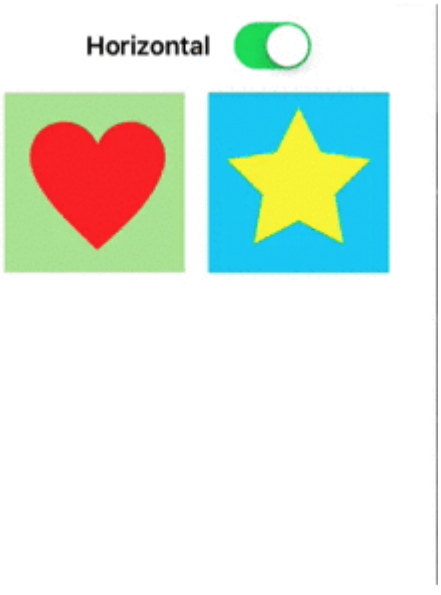
Подключите UISwitch к действию, которое мы можем активировать, переключая горизонтальную или вертикальную компоновку изображений

```
@IBAction func axisChange(sender: UISwitch) {
    UIView.animateWithDuration(1.0) {
        self.updateConstraintsForAxis()
    }
}
```

Функция updateConstraintForAxis просто устанавливает ось представления стека, содержащую два изображения:

```
private func updateConstraintsForAxis() {
    if (axisSwitch.on) {
        stackView.axis = .Horizontal
    } else {
        stackView.axis = .Vertical
    }
}
```

Анимированный gif ниже дает вам представление о том, как это выглядит:



Прочитайте Динамическое обновление UIStackView онлайн:
https://riptutorial.com/ru/ios/topic/5884/динамическое-обновление-uistackview

глава 132: ДОБАВЛЕНИЕ СВЕТОВОГО МОСТОВОГО ГОЛОВА

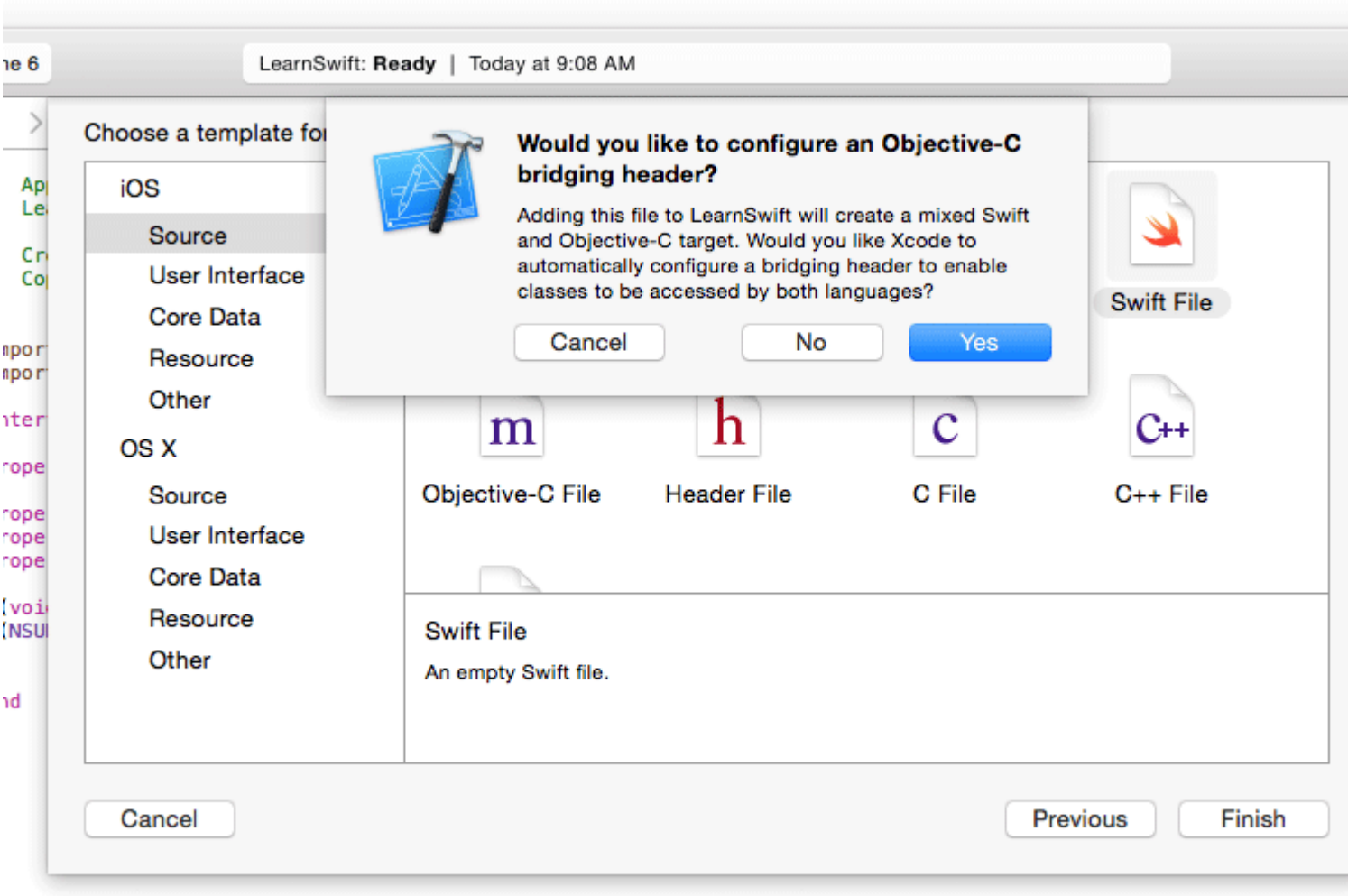
Examples

Как создать Swift Bridging Header вручную

- Добавьте новый файл в Xcode (File> New> File), затем выберите «Source» и нажмите «Header File».
- Назовите свой файл «YourProjectName-Bridging-Header.h». Пример. В моей станции приложения файл называется «Station-Bridging-Header».
- Создайте файл.
- Перейдите к настройкам сборки проекта и найдите раздел «Swift Compiler - Code Generation». Вы можете быстрее найти «Swift Compiler» в окне поиска, чтобы сузить результаты. Примечание. Если у вас нет раздела «Swift Compiler - Code Generation», это означает, что у вас, вероятно, еще нет классов Swift, добавленных в ваш проект. Добавьте файл Swift, затем повторите попытку.
- Рядом с «Objective-C Bridging Header» вам нужно будет добавить имя / путь вашего файла заголовка. Если ваш файл находится в корневой папке вашего проекта, просто укажите имя файла заголовка. Примеры: «ProjectName / ProjectName-Bridging-Header.h» или просто «ProjectName-Bridging-Header.h».
- Откройте свой недавно созданный заголовок моста и импортируйте классы Objective-C с помощью операторов #import. Любой класс, указанный в этом файле, будет доступен из ваших быстрых классов.

Xcode автоматически создается

Добавьте новый файл Swift в проект Xcode. Назовите его как хотите, и вы должны получить окно с предупреждением о том, хотите ли вы создать заголовок моста. Примечание. Если вы не получили приглашение добавить заголовок моста, вы, вероятно, отклонили это сообщение раньше, и вам придется добавить заголовок вручную (см. Ниже)



Прочитайте ДОБАВЛЕНИЕ СВЕТОВОГО МОСТОВОГО ГОЛОВА онлайн:
https://riptutorial.com/ru/ios/topic/10851/добавление-светового-мостового-голова

глава 133: доступность

Вступление

Доступность iOS позволяет пользователям с нарушениями слуха и визуальными нарушениями обращаться к iOS и вашему приложению, поддерживая различные функции, такие как VoiceOver, Voice Control, White on Black, Mono Audio, Speech to Text и т. Д. Обеспечение доступности в приложении iOS означает, что приложение можно использовать для всех.

Examples

Сделать доступным доступ

Отметьте свой подкласс `UIView` как доступный элемент, чтобы он отображался в VoiceOver.

```
myView.isAccessibilityElement = YES;
```

Убедитесь, что представление говорит о значимой метке, значении и подсказке. Apple предоставляет более подробную информацию о том, как выбирать хорошие описания в Руководстве по программированию специальных возможностей .

Рамка доступности

Кадр доступности используется VoiceOver для отслеживания ударов, рисования курсора VoiceOver и расчета того, где в сфокусированном элементе имитируется крана, когда пользователь дважды нажимает на экран. Обратите внимание, что кадр находится в координатах экрана!

```
myElement.accessibilityFrame = frameInScreenCoordinates;
```

Если ваши элементы или макеты экрана часто меняются, рассмотрите возможность переопределения - `accessibilityFrame` чтобы всегда предоставлять обновленный прямоугольник. Вычисление относительного кадра, относящегося к просмотру прокрутки, может быть подверженным ошибкам и утомительным. iOS 10 вводит новый API, чтобы сделать это проще: `accessibilityFrameInContainerSpace` .

Изменение экрана

VoiceOver отлично работает большую часть времени, легко читая вслух экраны, наполненные контентом и интуитивно следуя за пользователем. Увы, общее решение не является совершенным. Иногда только вы, разработчик приложения, знаете, где VoiceOver

должен быть сфокусирован на оптимальном опыте пользователя. К счастью, VoiceOver прислушивается к уведомлениям о доступности системы, чтобы узнать, где находится фокус. Чтобы переместить курсор VoiceOver вручную, опубликуйте экран с измененной информацией:

```
UIAccessibilityPostNotification(UIAccessibilityScreenChangedNotification, firstElement);
```

Когда это уведомление опубликовано, короткая серия тонов уведомляет пользователей об этом изменении. Второй параметр может быть либо следующим элементом для фокусировки, либо строкой, объявляющей об изменении. Публикуйте уведомление об изменении экрана только в том случае, если у него плохой опыт VoiceOver, и нет другого способа обхода проблемы. Перемещение курсора VoiceOver похоже на то, чтобы заглянуть на экран с видимым пользователем. Это может быть раздражающим и дезориентирующим, чтобы вестись таким образом.

Изменение макета

Во многих случаях содержимое на одном экране будет обновляться новым или другим контентом. Например, представьте форму, которая показывает дополнительные параметры, основанные на ответе пользователя на предыдущий вопрос. В этом случае уведомление «изменение макета» позволяет либо объявить об изменении, либо сосредоточиться на новом элементе. Это уведомление принимает те же параметры, что и уведомление об изменении экрана.

```
UIAccessibilityPostNotification(UIAccessibilityLayoutChangedNotification, firstElement);
```

Объявление

Объявления полезны для оповещения пользователей о событиях, которые не требуют какого-либо взаимодействия, например, «заблокирован экран» или «завершена загрузка». Используйте более конкретное объявление, чтобы уведомлять пользователей об изменениях экрана или более мелких изменениях макета.

```
UIAccessibilityPostNotification(UIAccessibilityAnnouncementNotification, @"The thing
happened!");
```

Элементы заказа

VoiceOver перемещается с верхнего левого угла в нижнее правое, независимо от иерархии представлений. Как правило, контент размещается на языках слева направо, так как зрячие люди склонны сканировать экран в «F-образном шаблоне». Пользователи VoiceOver будут ориентироваться так же, как и обычные пользователи. Предсказуемость и согласованность очень важны для доступности. Пожалуйста, воздержитесь от настроек, которые «улучшают» поведение по умолчанию (например, сначала заказывайте панель

вкладок в порядке выбора). Тем не менее, если вы получили отзывы о том, что порядок элементов в вашем приложении удивителен, есть несколько способов, которыми вы можете улучшить опыт.

Если VoiceOver должен читать подпункты представления один за другим, но это не так, вам может потребоваться указать VoiceOver, что элементы, содержащиеся в одном представлении, связаны. Вы можете сделать это, установив `shouldGroupAccessibiltyChildren` :

```
myView.shouldGroupAccessibilityChildren = YES;
```

Для поддержки сложных навигационных структур, которые охватывают несколько контейнеров или включают в себя интерфейсы, визуализированные без UIKit, рассмотрите возможность внедрения контейнерного протокола в родительском представлении.

Контейнер доступности

VoiceOver может перемещаться много приложений на прошивку , так как большинство `UIKit` классов реализуют `UIAccessibilityProtocol` . Функции, которые не представляют собой экранные элементы с использованием `UIView` , включая приложения, которые используют Core Graphics или Metal для выполнения чертежа, должны описывать эти элементы для доступности. Начиная с iOS 8.0, это можно сделать, назначив свойство `UIView` содержащее недоступные элементы:

```
myInaccessibleContainerView.accessibilityElements = @[elements, that, should, be, accessible];
```

Каждый объект в массиве может быть экземпляром `UIAccessibilityElement` или любым другим классом, который придерживается `UIAccessibilityProtocol` . Детские элементы должны быть возвращены в том порядке, в котором пользователь должен их перемещать. В качестве автора приложения вы можете использовать контейнеры доступности, чтобы переопределить порядок навигации по левому краю по умолчанию в нижнем правом углу навигации VoiceOver. Учитывая, что `UIView` реализует `UIAccessibilityProtocol` , вы можете комбинировать экземпляры `UIAccessibilityElement` и `UIView` в том же массиве элементов детской доступности. Обратите внимание: если вы назначаете элементы вручную, вам не нужно применять какие-либо методы протокола динамической доступности, хотя вам может потребоваться выдать уведомление об изменении экрана для элементов, которые будут обнаружены VoiceOver.

Модальный вид

Модальные представления полностью захватывают внимание пользователя до завершения задачи. iOS разъясняет это пользователям, затемняя и отключая все остальные контенты, когда видится модальное представление, такое как предупреждение или popover. Приложение, которое реализует пользовательский модальный интерфейс, должно указывать на VoiceOver, что этот взгляд заслуживает отдельного внимания пользователя,

установив `accessibilityViewIsModal` . Обратите внимание, что это свойство должно быть установлено только в представлении, содержащем модальный контент, а не в элементах, содержащихся в модальном представлении.

```
myModalView.accessibilityViewIsModal = YES;
```

Пометка вида как модального поощряет VoiceOver игнорировать взгляды братьев и сестер. Если после установки этого свойства вы обнаружите, что VoiceOver по-прежнему перемещает другие элементы в вашем приложении, попробуйте скрывать представления проблем до тех пор, пока модальный не отклонит.

Скрытие элементов

Большинство классов UIKit, включая UIView, придерживаются `UIAccessibilityProtocol` и возвращают правильные значения по умолчанию. Легко считать само собой разумеющимся, что `UIView` установленный для `UIView` также отсутствует в иерархии доступности и не будет перемещаться с помощью VoiceOver. Хотя этого поведения по умолчанию, как правило, достаточно, бывают случаи, когда представление будет присутствовать в иерархии представлений, но не видимо или судоходно. Например, набор кнопок может перекрываться другим видом, делая их невидимыми для наблюдаемого пользователя. Тем не менее VoiceOver по-прежнему будет пытаться перемещаться по ним, поскольку они технически не скрыты от `UIKit` и поэтому все еще присутствуют в иерархии доступности. В таких случаях вы должны намекнуть на VoiceOver, что родительское представление недоступно. Вы можете сделать это, явно скрывая представление от UIKit, установив скрытый, когда представление заходит на экран:

```
myViewFullofButtons.hidden = YES;
```

Кроме того, вы можете оставить родительский вид видимым и просто скрыть его дочерние элементы из иерархии доступности:

```
myViewFullofButtons.accessibilityElementsHidden = YES;
```

Временные представления - это другое место, которое вы хотите скрыть элементы из иерархии доступности, оставив их видимыми для пользователей. Например, вид, который появляется, когда вы нажимаете кнопку громкости, видится видимым пользователям, но не требует внимания, как это делает обычное предупреждение. Вы не хотите, чтобы VoiceOver прерывал пользователя и перемещал курсор от того, что они делали, чтобы объявить новый том, особенно учитывая, что регулировка громкости уже обеспечивает аудиальную обратную связь через звук щелчка. В подобных случаях вам нужно скрыть представление, используя `accessibilityElementsHidden` .

Прочитайте доступность онлайн: https://riptutorial.com/ru/ios/topic/773/доступность

глава 134: Загрузка изображений async

Examples

Самый простой способ

Самый простой способ создать это - использовать Alamofire и UIImageViewExtension . Нам нужен табличный вид с ячейкой, в которой есть ImageView, и позволяет называть его `imageView` .

В функции cellForRowAt: tableView мы загрузим изображение и установим его следующим образом:

```
let url = URL(string: "https://httpbin.org/image/png")!
let placeholderImage = UIImage(named: "placeholder")!

imageView.af_setImage(withURL: url, placeholderImage: placeholderImage)
```

URL-адрес должен указывать на изображение, которое вы хотите загрузить, и изображение placeHolder должно быть сохраненным изображением. Затем мы вызываем метод `af_setImage` на `imageView` который загружает изображение по указанному URL-адресу, и во время загрузки будет отображаться изображение-заполнитель. Как только изображение загрузится, отображается запрошенное изображение

Убедитесь, что ячейка все еще видна после загрузки

Иногда загрузка занимает больше времени, чем отображается ячейка. В этом случае может случиться так, что загруженное изображение отображается в неправильной ячейке. Чтобы исправить это, мы не можем использовать расширение UIImageView .

Мы все еще будем использовать Alamofire, но мы будем использовать обработчик завершения для отображения изображения.

В этом сценарии нам все еще нужен tableView с ячейкой, в которой есть imageView. В методе cellForRowAt: мы загрузим изображение со следующим кодом:

```
let placeholderImage = UIImage(named: "placeholder")!
imageView.image = placeholderImage

let url = URL(string: "https://httpbin.org/image/png")!

Alamofire.request(url!, method: .get).responseImage { response in
    guard let image = response.result.value else { return }

    if let updateCell = tableView.cellForRow(at: indexPath) {
        updateCell.imageView.image = image
    }
```

```
}
```

В этом примере мы сначала установили изображение на образ заполнителя. Затем мы загружаем изображение с помощью метода `request` Alamofire . Мы передаем url как первый аргумент, и поскольку мы просто хотим получить изображение, мы будем использовать HTTP-метод `.get` . Поскольку мы загружаем изображение, мы хотим, чтобы ответ был образом, поэтому мы используем метод `.responseImage` .

После загрузки изображения замыкание вызывает вызов, и в первую очередь мы гарантируем, что загруженное изображение действительно существует. Затем мы убеждаемся, что ячейка все еще видна, проверяя, что cellForRow (at: indexPath) не возвращает нуль. Если это ничего не происходит, если это не так, мы назначаем недавно загруженное изображение.

Этот последний оператор if гарантирует, что ячейка все еще будет видна, если пользователь уже прокрутил ячейку, а updateCell будет равен нулю, а оператор if возвращает nil. Это помогает нам предотвратить отображение неправильного изображения в ячейке.

Прочитайте Загрузка изображений async онлайн: https://riptutorial.com/ru/ios/topic/10793/загрузка-изображений-async

глава 135: Изменение размера UIImage

параметры

CGInterpolationQuality	Уровни качества интерполяции для рендеринга изображения.
Качество интерполяции - это параметр состояния графики	typedef enum CGInterpolationQuality CGInterpolationQuality;

Examples

Изменение размера изображения по размеру и качеству

```
- (UIImage *)drawImageBySize:(CGSize)size quality:(CGInterpolationQuality)quality
{
    UIGraphicsBeginImageContextWithOptions(size, NO, 0.0);
    CGContextRef context = UIGraphicsGetCurrentContext();
    CGContextSetInterpolationQuality(context, quality);
    [self drawInRect: CGRectMake (0, 0, size.width, size.height)];
    UIImage *resizedImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return resizedImage;
}
```

Прочитайте Изменение размера UIImage онлайн: https://riptutorial.com/ru/ios/topic/6422/изменение-размера-uiimage

глава 136: Изменить цвет строки состояния

Examples

Для баров статуса не-UINavigationBar

1. В настройке info.plist `View controller-based status bar appearance` на `YES`
2. В виду, что контроллеры, не содержащиеся в `UINavigationController` реализуют этот метод.

В Objective-C:

```
- (UIStatusBarStyle)preferredStatusBarStyle
{
    return UIStatusBarStyleLightContent;
}
```

В Свифт:

```
override func preferredStatusBarStyle() -> UIStatusBarStyle {
    return UIStatusBarStyle.LightContent
}
```

Для баров статуса UINavigationBar

Подкласс UINavigationController и затем переопределить эти методы:

В Objective-C:

```
- (UIStatusBarStyle)preferredStatusBarStyle
{
    return UIStatusBarStyleLightContent;
}
```

В Свифт:

```
override func preferredStatusBarStyle() -> UIStatusBarStyle {
    return .lightContent
}
```

Кроме того, вы можете установить `barStyle` в экземпляре `UINavigationBar` :

Цель C:

```
// e.g. in your view controller's viewDidLoad method:
self.navigationController.navigationBar.barStyle = UIBarStyleBlack;  // this will give you a
white status bar
```

стриж

```
// e.g. in your view controller's viewDidLoad method:
navigationController?.navigationBar.barStyle = .black // this will give you a white status bar
```

`UIBarStyle` по `default` : `black` , `blackOpaque` , `blackTranslucent` . Последние 3 должны дать вам строку состояния с белым текстом, только последние два указывают на непрозрачность панели.

Примечание. Вы по-прежнему можете изменить внешний вид панели навигации, как вам нравится.

Если вы не можете изменить код ViewController

Если вы используете библиотеку, содержащую (например) AwesomeViewController с неправильным цветом строки состояния, вы можете попробовать следующее:

```
  let awesomeViewController = AwesomeViewController()
  awesomeViewController.navigationBar.barStyle = .blackTranslucent // or other style
```

Для локализации ViewController

Если вы используете `UIViewControllerContainment` есть несколько других методов, на которые стоит обратить внимание.

Если вы хотите, чтобы child viewController контролировал представление строки состояния (то есть, если ребенок находится в верхней части экрана

в Свифт

```
class RootViewController: UIViewController {

    private let messageBarViewController = MessageBarViewController()

    override func childViewControllerForStatusBarStyle() -> UIViewController? {
        return messageBarViewController
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        //add child vc code here...

        setNeedsStatusBarAppearanceUpdate()
    }
}

class MessageBarViewController: UIViewController {

    override func preferredStatusBarStyle() -> UIStatusBarStyle {
        return .Default
```

```
    }
}
```

Изменение стиля строки состояния для всего приложения

SWIFT:

Шаг 1:

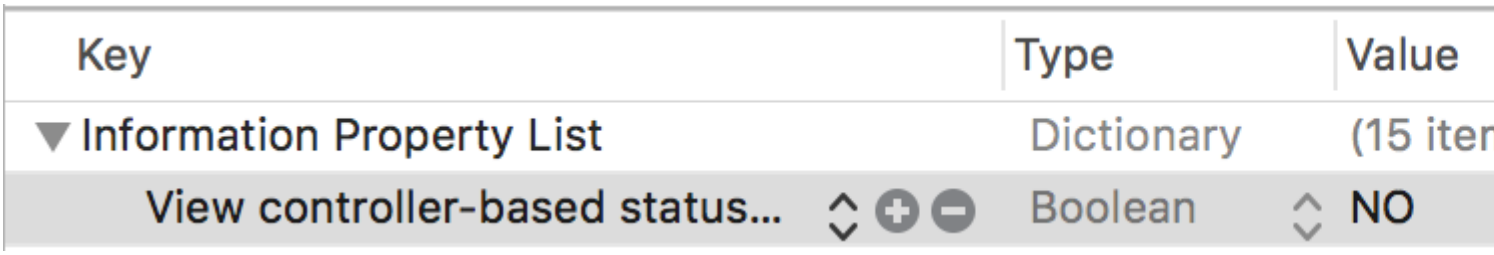
В вашем **Info.plist** добавьте следующий атрибут:

```
View controller-based status bar appearance
```

и установить его значение для

```
NO
```

как описано на изображении ниже:



Шаг 2:

В файле **AppDelegate.swift,** в `didFinishLaunchingWithOptions` методе, добавьте этот код:

```
UIApplication.shared.statusBarStyle = .lightContent
```

или же

```
UIApplication.shared.statusBarStyle = .default
```

- Опция **.lightContent** будет устанавливать цвет **statusBar** в белый для всего приложения.
- Параметр **.default** будет устанавливать цвет **statusBar** в исходный черный цвет для

всего приложения.

Objective-C:

Следуйте первому шагу от секции **SWIFT** . Затем добавьте этот код в файл **AppDelegate.m** :

```
[[UIApplication sharedApplication] setStatusBarStyle:UIStatusBarStyleLightContent];
```

или же

```
[[UIApplication sharedApplication] setStatusBarStyle:UIStatusBarStyleDefault];
```

Прочитайте Изменить цвет строки состояния онлайн: https://riptutorial.com/ru/ios/topic/378/изменить-цвет-строки-состояния

глава 137: имитатор

Вступление

iOS, watchOS и tvOS Simulators - отличные способы протестировать ваши приложения, не используя фактическое устройство. Здесь мы поговорим о работе с симуляторами.

замечания

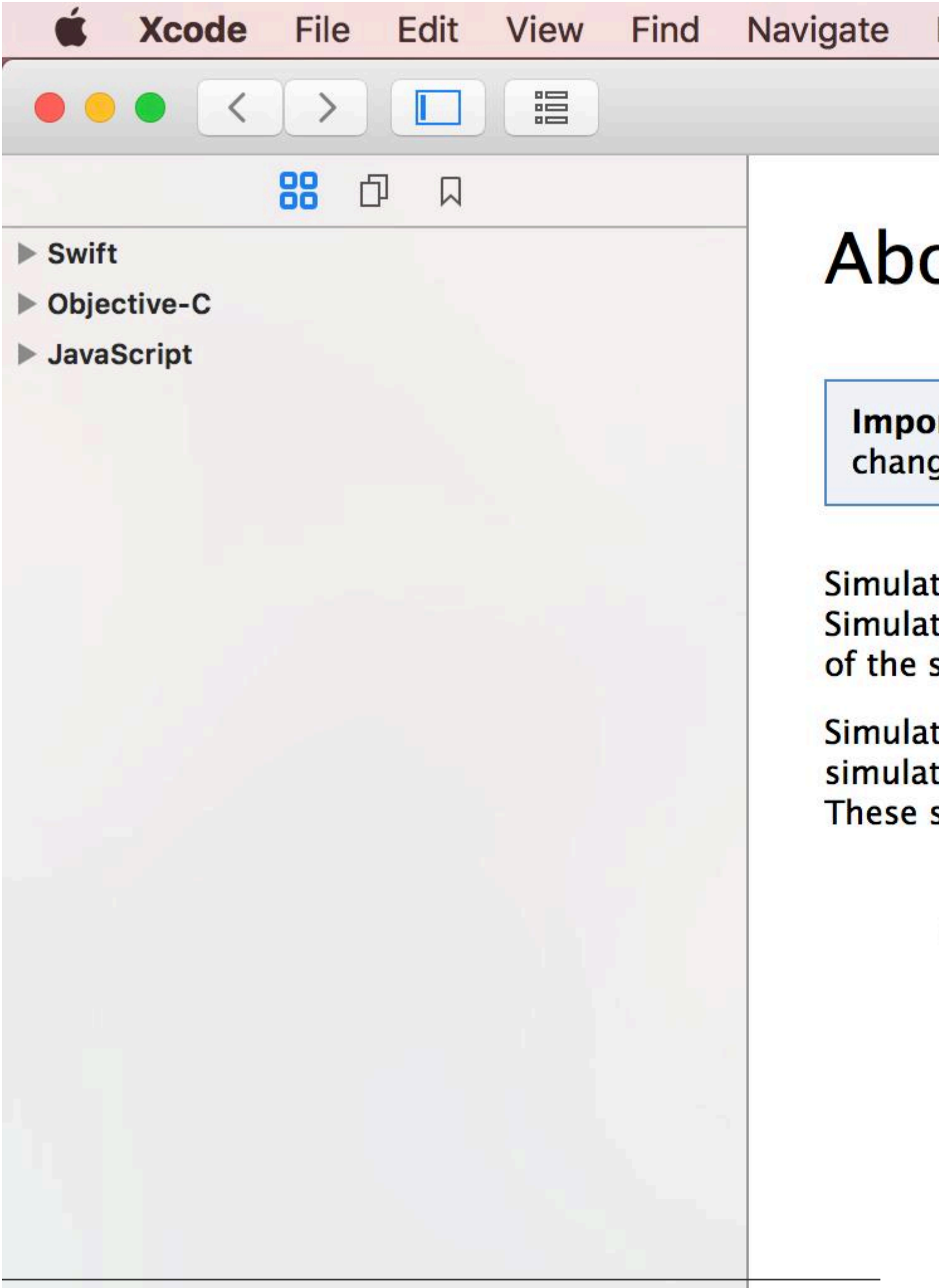
Различные типы тренажеров

- Симулятор iOS
- WatchOS Simulator
- Тренажер tvOS
- Симулятор сенсорного экрана

  Нет симулятора для macOS, потому что Xcode запускается на macOS, и, когда это необходимо, он запускает собственные приложения.

Получать помощь

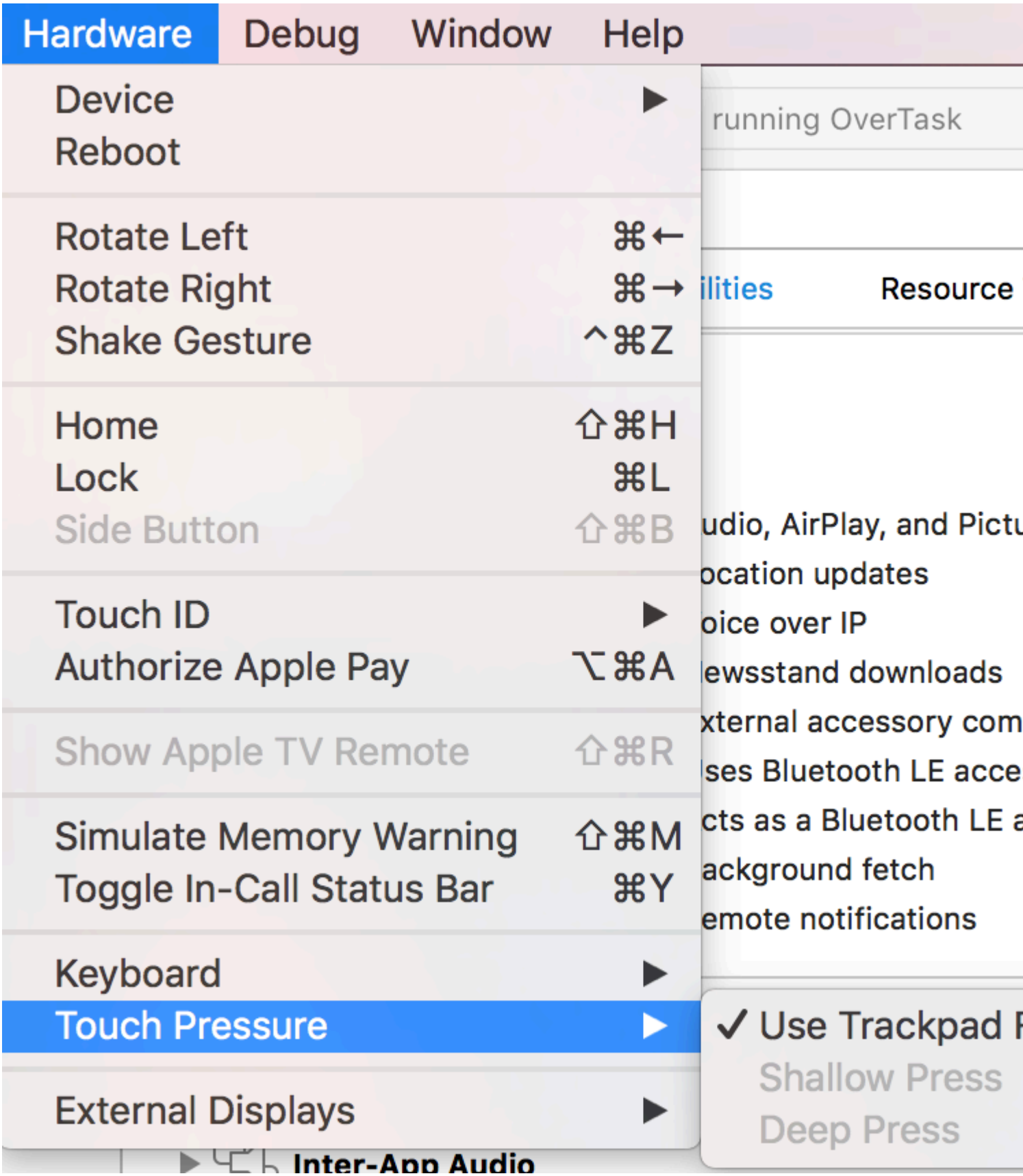
Вы всегда можете посетить справку Simulator в Help -> Справка Simulator:


Xcode
File
Edit
View
Find
Navigate
Swift
Objective-C
JavaScript
Abo
Impo
chang
Simulat
Simulat
of the s
Simulat
simulat
These s

Simulator (Watch)» будет запускать только watchOS.

Моделирование 3D / Force Touch

Перейдите к Hardware -> Touch Pressure:

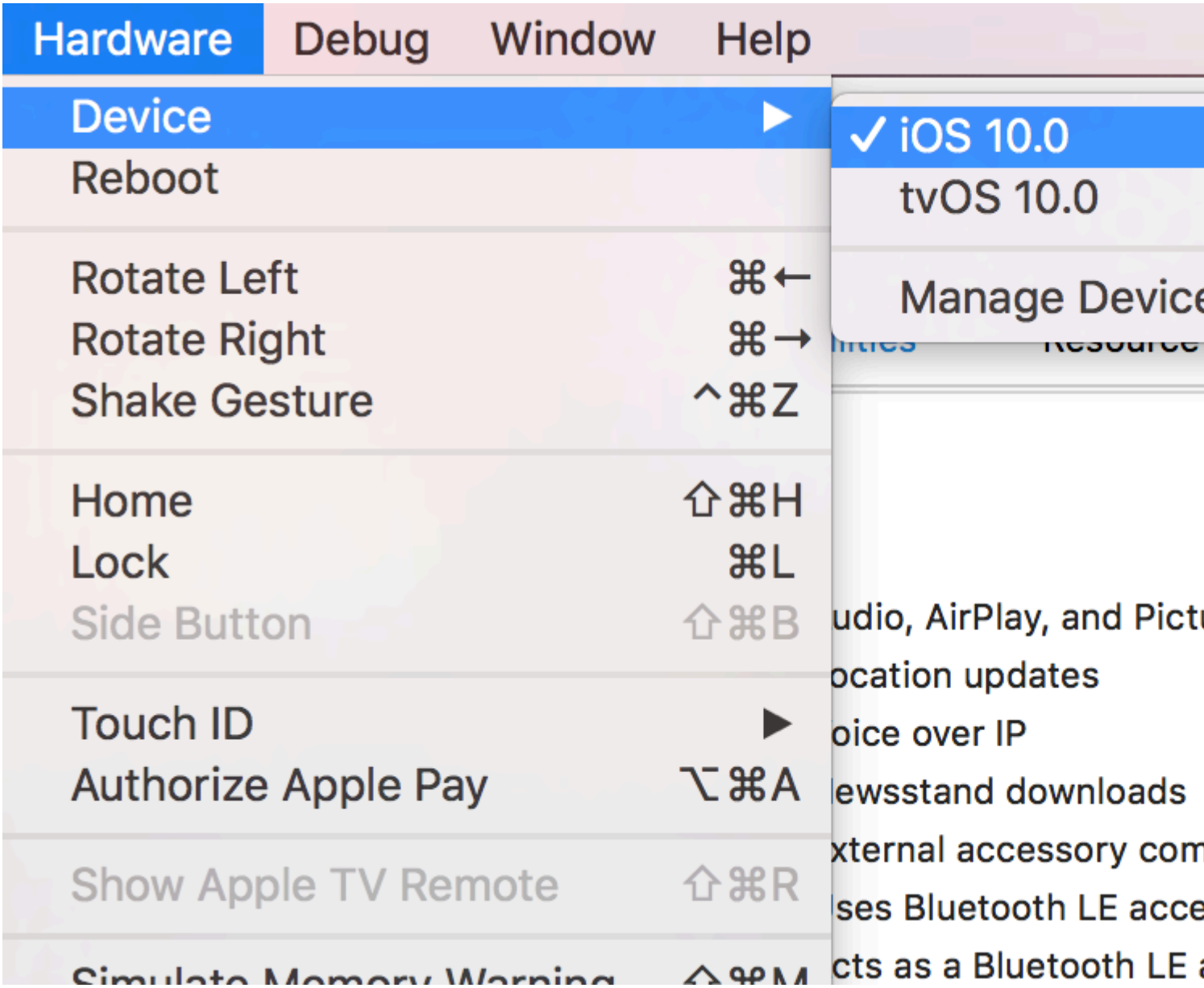


Чтобы имитировать нормальное нажатие, используйте Shift-Command-1 и глубокий, Shift-Command-2.

> Если у вашего MacBook есть трекпад Force Touch, вы можете использовать силу вашего трекпада для имитации 3D / Force Touch.

Изменить модель устройства

Перейдите в раздел «Оборудование -> Устройство:



Симулятор навигации

Главная кнопка

Вы должны использовать Shift-Command-H для анимации при закрытии приложения.

Замок

Чтобы заблокировать устройство, используйте Command-L.

вращение

Используйте команду со стрелками.

Прочитайте имитатор онлайн: https://riptutorial.com/ru/ios/topic/9177/имитатор

глава 138: Имитация местоположения с использованием файлов GPX iOS

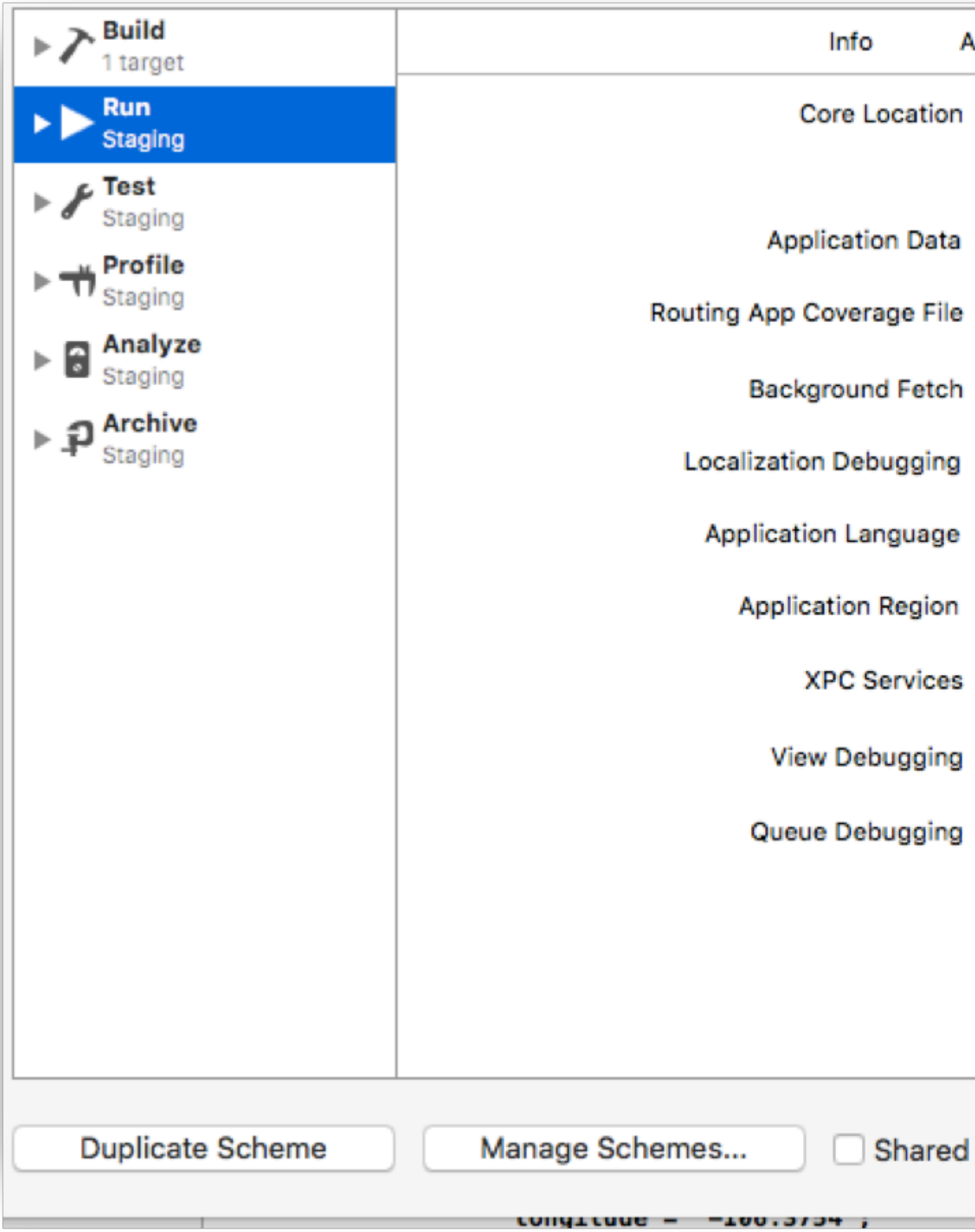
Examples

Ваш .gpx-файл: MPS_HQ.gpx

```
<gpx xmlns="http://www.topografix.com/GPX/1/1"
 xmlns:gpxx = "http://www.garmin.com/xmlschemas/GpxExtensions/v3"
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.topografix.com/GPX/1/1
 http://www.topografix.com/GPX/1/1/gpx.xsd
 http://www.garmin.com/xmlschemas/GpxExtensions/v3
 http://www8.garmin.com/xmlschemas/GpxExtensions/v3/GpxExtensionsv3.xsd"
 version="1.1"
 creator="gpx-poi.com">
 <wpt lat="38.9072" lon="77.0369">38.9072/-77.0369
 <time>2015-04-16T22:20:29Z</time>
  <name>Washington, DC</name>
  <extensions>
     <gpxx:WaypointExtension>
        <gpxx:Proximity>10</gpxx:Proximity>
        <gpxx:Address>
           <gpxx:StreetAddress>Washington DC</gpxx:StreetAddress>
           <gpxx:City>Washington</gpxx:City>
           <gpxx:State>DC</gpxx:State>
           <gpxx:Country>United States</gpxx:Country>
           <gpxx:PostalCode> 20005 </gpxx:PostalCode>
        </gpxx:Address>
     </gpxx:WaypointExtension>
  </extensions>
```

Чтобы установить это местоположение:

1. Перейдите в «Редактировать схему».
2. Выберите «Выполнить» -> «Параметры».
3. Установите флажок «Разрешить определение местоположения».
4. Выберите имя файла * .GPX в раскрывающемся списке «Расположение по умолчанию».



Прочитайте Имитация местоположения с использованием файлов GPX iOS онлайн: https://riptutorial.com/ru/ios/topic/9883/имитация-местоположения-с-использованием-файлов-

gpx-ios

глава 139: Инициализация идиом

Examples

Установите кортежи во избежание повторения кода

Избегайте повторения кода в конструкторах, устанавливая кортеж переменных с одним слоем:

```
class Contact: UIView
{
    private var message: UILabel
    private var phone: UITextView

    required init?(coder aDecoder: NSCoder) {
        (message, phone) = self.dynamicType.setUp()
        super.init(coder: aDecoder)
    }

    override func awakeFromNib() {
        (message, phone) = self.dynamicType.setUp()
        super.awakeFromNib()
    }

    override init(frame: CGRect) {
        (message, phone) = self.dynamicType.setUp()
        super.init(frame: frame)
    }

    private static func setUp(){
        let message = UILabel()  // ...
        let phone = UITextView() // ...
        return (message, phone)
    }
}
```

Инициализировать с помощью позиционных констант

```
let mySwitch: UISwitch = {
    view.addSubview($0)
    $0.addTarget(self, action: "action", forControlEvents: .TouchUpInside)
    return $0
}(UISwitch())
```

Инициализировать атрибуты в didSet

```
@IBOutlet weak var title: UILabel! {
  didSet {
    label.textColor = UIColor.redColor()
    label.font = UIFont.systemFontOfSize(20)
    label.backgroundColor = UIColor.blueColor()
```

```
  }
}
```

Также возможно установить значение и инициализировать его:

```
private var loginButton = UIButton() {
    didSet(oldValue) {
        loginButton.addTarget(self, action: #selector(LoginController.didClickLogin),
forControlEvents: .TouchUpInside)
    }
}
```

Групповые выходы в пользовательском NSObject

Переместите каждую розетку в NSObject. Затем перетащите объект из библиотеки в сцену контроллера раскадровки и закрепите там элементы.

```
class ContactFormStyle: NSObject
{
    @IBOutlet private weak var message: UILabel! {
      didSet {
        message.font = UIFont.systemFontOfSize(12)
        message.textColor = UIColor.blackColor()
      }
    }
}

class ContactFormVC: UIViewController
{
    @IBOutlet private var style: ContactFormStyle!
}
```

Инициализируйте с помощью

Это похоже на синтаксис примера, который инициализирует использование позиционных констант, но требует расширения `Then` от https://github.com/devxoul/Then (прилагается ниже).

```
let label = UILabel().then {
    $0.textAlignment = .Center
    $0.textColor = UIColor.blackColor(
    $0.text = "Hello, World!"
}
```

Расширение `Then` :

```
import Foundation

public protocol Then {}

extension Then
{
    public func then(@noescape block: inout Self -> Void) -> Self {
        var copy = self
```

```
        block(&copy)
        return copy
    }
}

extension NSObject: Then {}
```

Фабричный метод с блоком

```
internal func Init<Type>(value : Type, block: @noescape (object: Type) -> Void) -> Type
{
    block(object: value)
    return value
}
```

Использование:

```
Init(UILabel(frame: CGRect.zero)) {
    $0.backgroundColor = UIColor.blackColor()
}
```

Прочитайте Инициализация идиом онлайн: https://riptutorial.com/ru/ios/topic/3513/инициализация-идиом

глава 140: Интеграция SqlCipher

Вступление

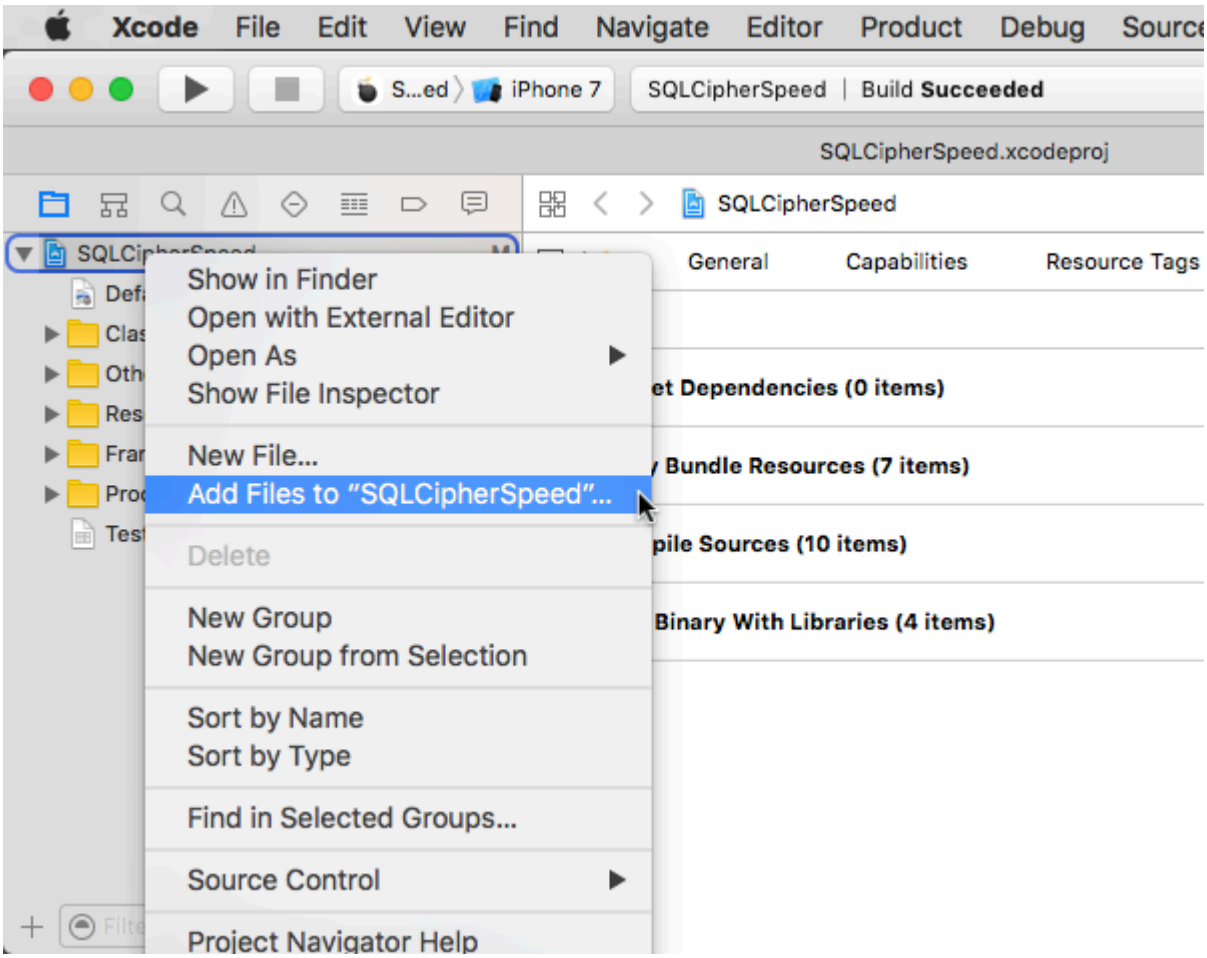
SQLite уже является популярным API для постоянного хранения данных в приложениях iOS, поэтому потенциал роста для разработчиков очевиден. Как программист вы работаете со стабильным, хорошо документированным API, который, как оказалось, имеет много хороших оберток, доступных в Objective-C, таких как FMDB и Encrypted Core Data. Все проблемы безопасности полностью отделены от кода приложения и управляются базовой структурой.

замечания

1. Откройте терминал, перейдите в корневой каталог вашего проекта и проверьте код проекта SQLCipher с помощью Git:

```
$ git clone https://github.com/sqlcipher/sqlcipher.git
```

2. Щелкните правой кнопкой мыши по проекту и выберите «Добавить файлы в« Мое приложение »» (ярлык будет меняться в зависимости от имени вашего приложения). Поскольку мы клонировали SQLCipher непосредственно в ту же папку, что и ваше приложение iOS, вы должны увидеть папку sqlcipher в корневой папке проекта. Откройте эту папку и выберите **sqlcipher.xcodeproj**

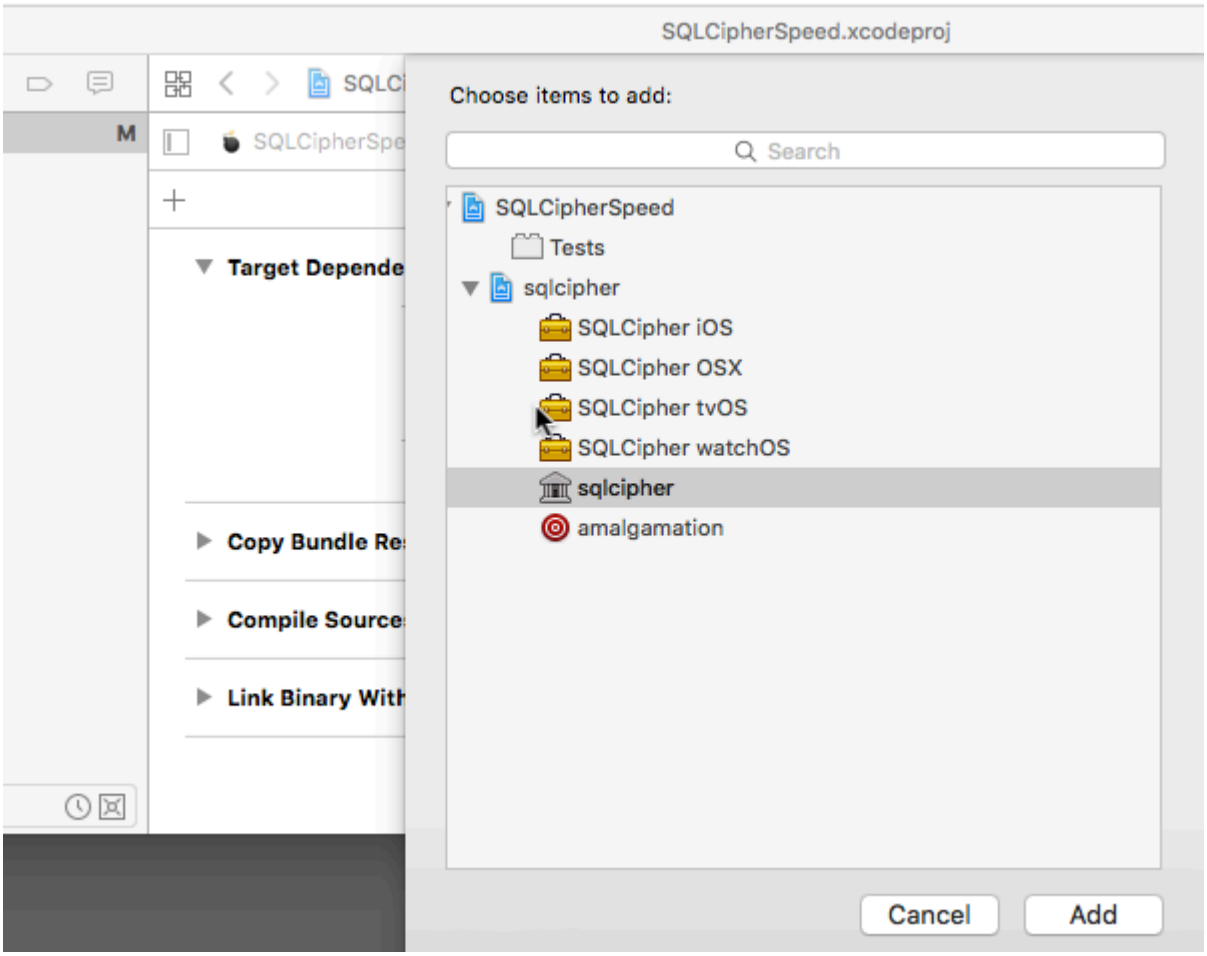


3. Выберите панель «Настройки сборки». В поле поиска введите «Пути поиска заголовков». Дважды щелкните поле под целевым столбцом и добавьте следующий путь: **$ (PROJECT_DIR) / sqlcipher / src**

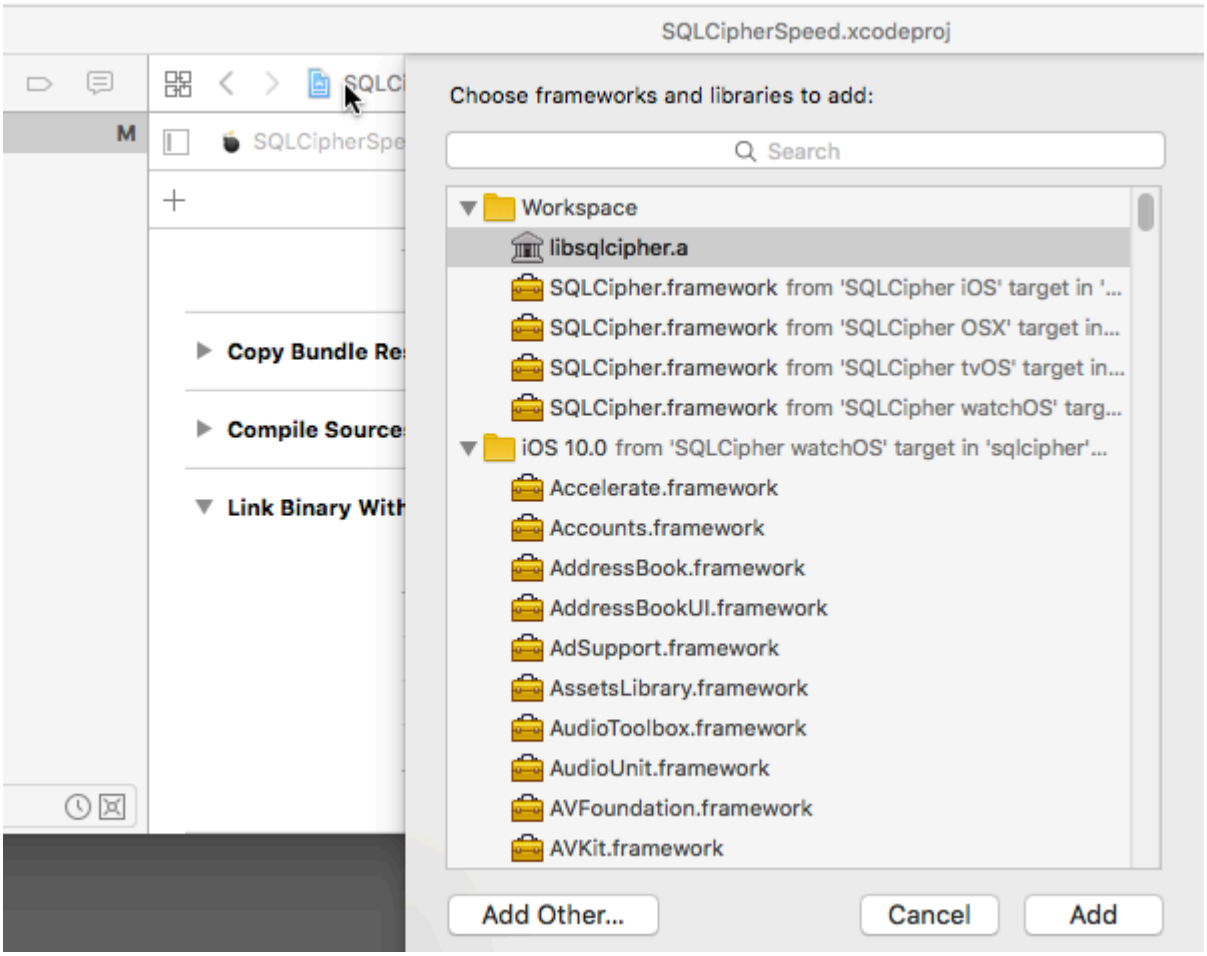
4. Начните вводить «Other Linker Flags» в поле поиска, пока не появится параметр, дважды щелкните его, чтобы изменить его, и добавьте следующее значение: **$ (BUILT_PRODUCTS_DIR) /libsqlcipher.a**

5. Начните вводить «Other C Flags» в поле поиска до появления параметра, дважды щелкните его, чтобы изменить его, а во всплывающем окне добавьте следующее значение: **-DSQLITE_HAS_CODEC**

6. Разверните «Зависимости целей» и нажмите кнопку «+» в конце списка. В открывшемся браузере выберите **цель** статической библиотеки **sqlcipher** :



7. Разверните ссылку «Бинарные ссылки» с библиотеками, нажмите кнопку «+» в конце списка и выберите библиотеку **libsqlcipher.a** .



8. Наконец, также в разделе «Связывание с библиотеками» добавьте **Security.framework** .

Examples

Интеграция кода:

Интеграция для открытия базы данных с использованием пароля.

```
-(void)checkAndOpenDB{
    sqlite3 *db;
    NSString *strPassword = @"password";

        if (sqlite3_open_v2([[databaseURL path] UTF8String], &db, SQLITE_OPEN_READWRITE |
SQLITE_OPEN_CREATE, NULL) == SQLITE_OK) {
            const char* key = [strPassword UTF8String];
            sqlite3_key(db, key, (int)strlen(key));
            if (sqlite3_exec(db1, (const char*) "SELECT count(*) FROM sqlite_master;", NULL,
NULL, NULL) == SQLITE_OK) {
                NSLog(@"Password is correct, or a new database has been initialized");
            } else {
                NSLog(@"Incorrect password!");
            }
            sqlite3_close(db);
        }
}
```

```
- (NSURL *)databaseURL
{
    NSArray *URLs = [[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
inDomains:NSUserDomainMask];
    NSURL *directoryURL = [URLs firstObject];
    NSURL *databaseURL = [directoryURL URLByAppendingPathComponent:@"database.sqlite"];
    return  databaseURL;
}
```

Прочитайте Интеграция SqlCipher онлайн: https://riptutorial.com/ru/ios/topic/9969/интеграция-sqlcipher

глава 141: Использование Image Aseets

Вступление

Атрибуты изображения используются для управления и организации различных типов изображений в нашем приложении iOS с использованием Xcode.

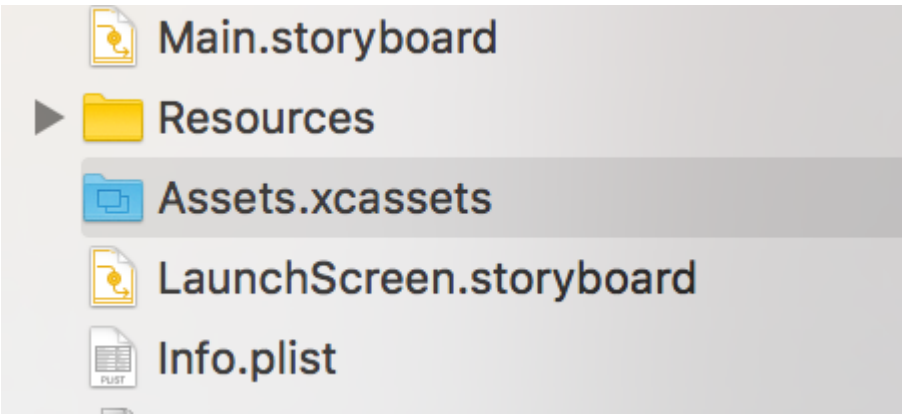
Этими активами могут быть **иконки приложений, запускать изображения, изображения, используемые во всем приложении, полноразмерные изображения, изображения в произвольном размере** и т. Д.

Examples

Значок приложения с использованием свойств изображения

Всякий раз, когда мы создаем новый проект в Xcode для нашего нового приложения, он дает нам различные встроенные классы, цели, тесты, файл `Assets.xcassets` и т. Д. Подобным же образом он также дает нам файл `Assets.xcassets` , который управляет всеми объектами изображения в нашем проект.

Вот как выглядит этот файл в файловом навигаторе:



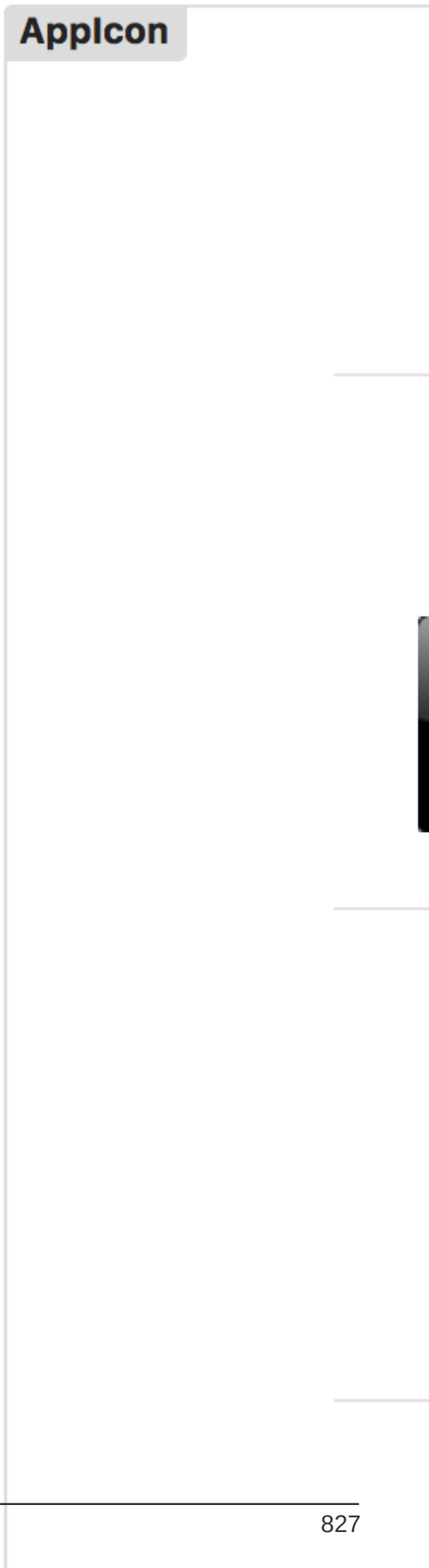
Если мы щелкнем по нему, это будет выглядеть так:


AppIcon

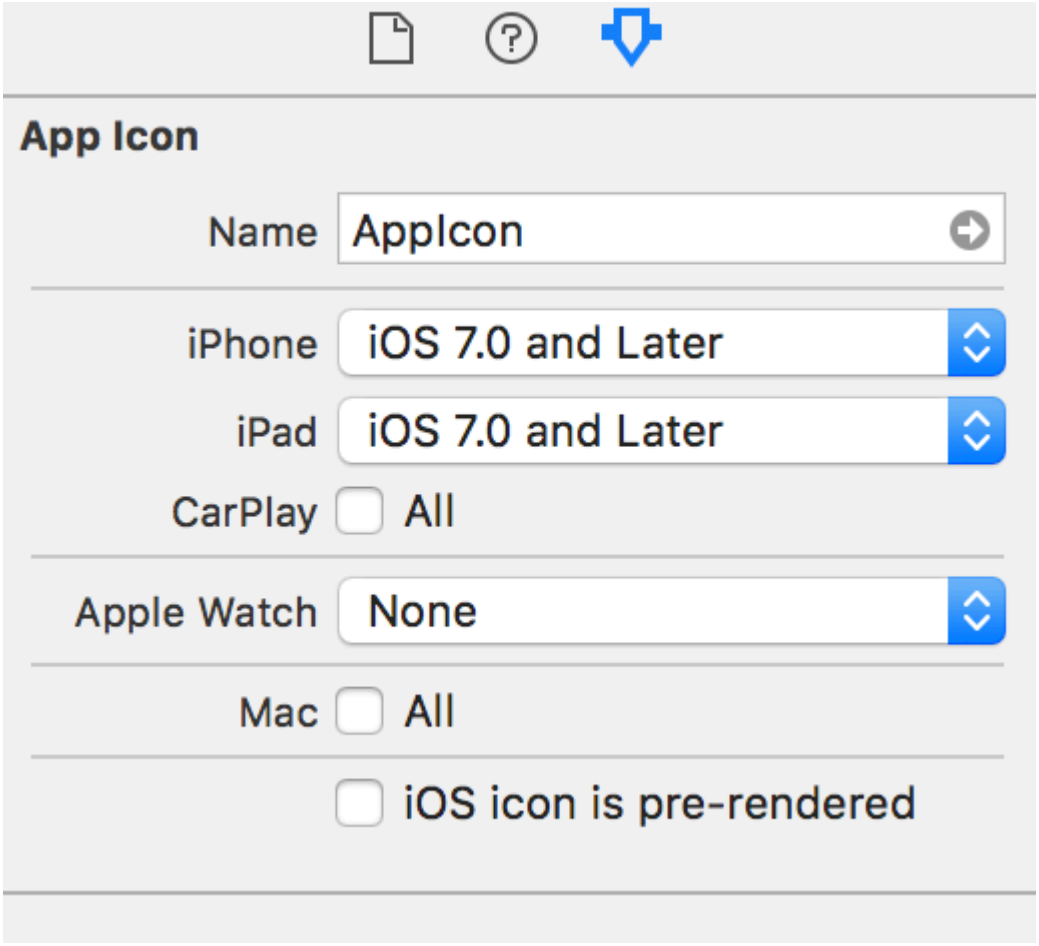

AppIcon

Нам просто нужно **перетащить** соответствующее изображение на каждый пустой квадратный блок. Каждый чернокожий скажет нам, какой размер должен быть таким, он написан чуть ниже.
После перетаскивания всех изображений во всех квадратах это будет выглядеть так:

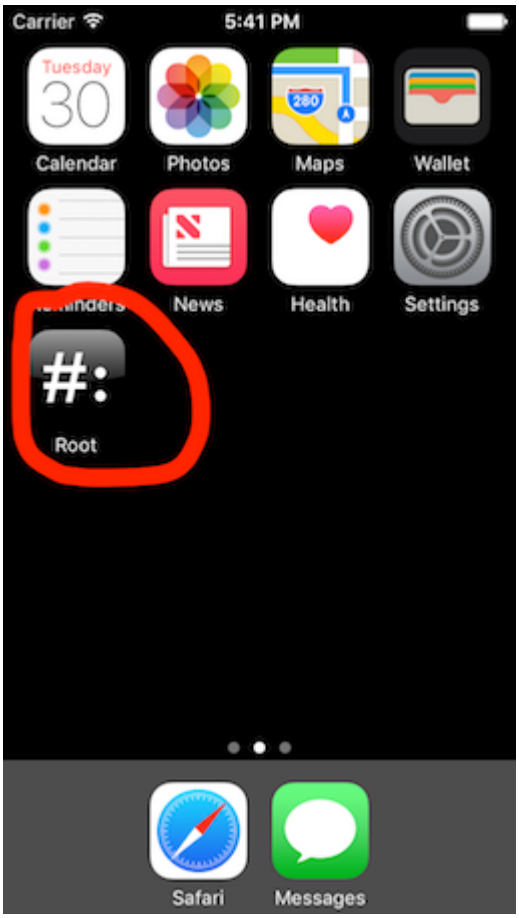

AppIcon


AppIcon

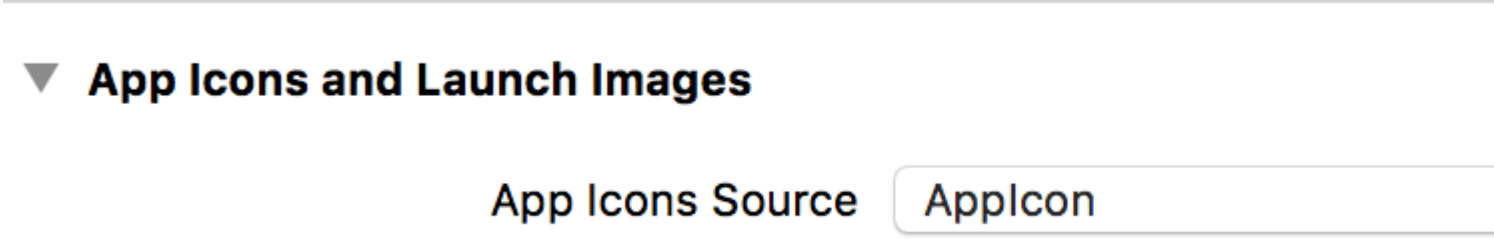
Utilities -> Attributes Inspector как:



Как только мы закончим это, просто запустите приложение, и у нас будет приятная иконка для приложения:



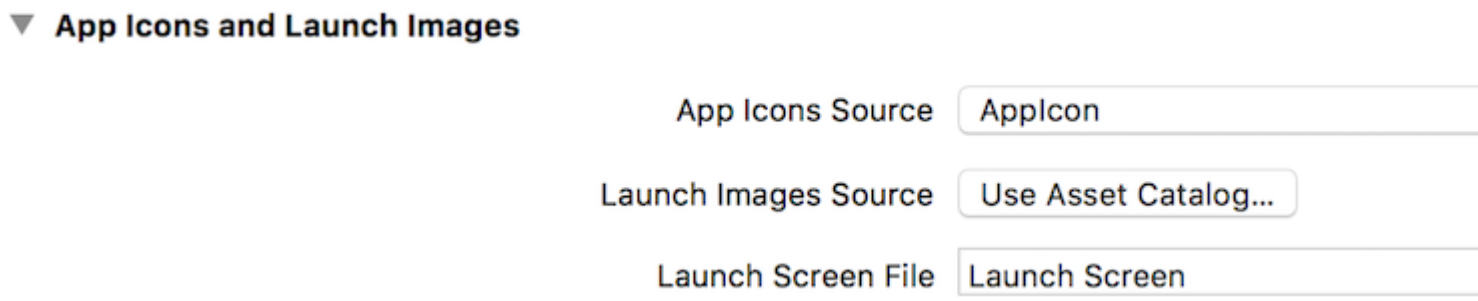
Он есть по умолчанию, но если это не так, убедитесь, что это параметры, как в Target-> General settings:



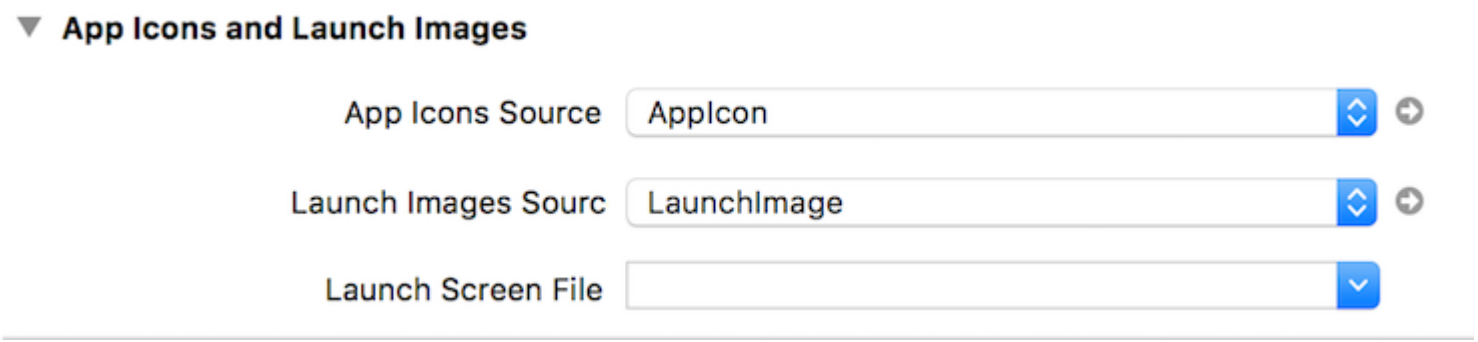
LaunchImage с использованием свойств изображения

Экран запуска - это экран, который появляется при запуске приложения и длится до появления первого экрана приложения.
Подробнее о Launch Screen и руководствах здесь .

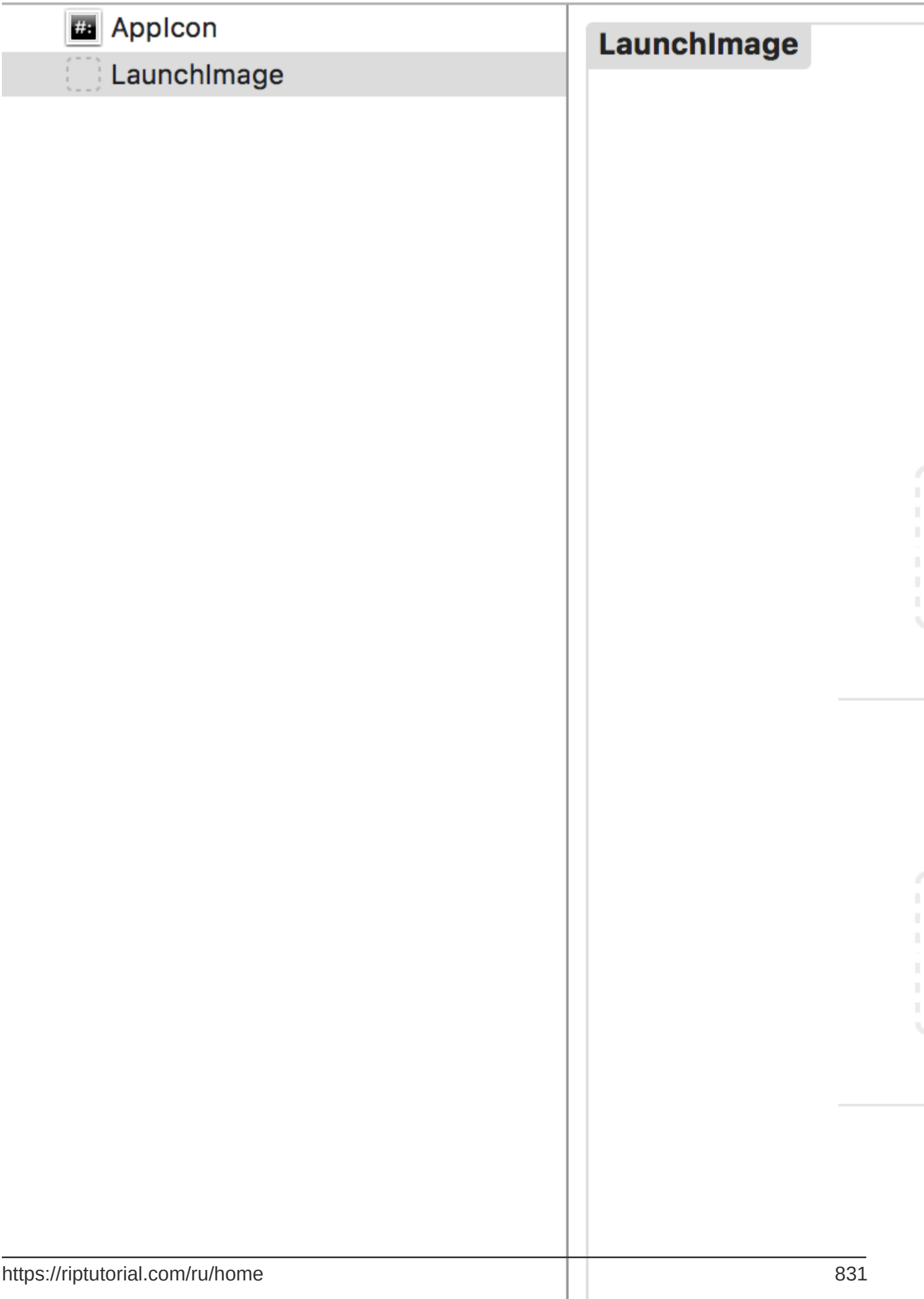
Подобно AppIcons, мы должны упомянуть в настройках проекта об использовании имиджевых ресурсов для изображения экрана запуска.
По умолчанию параметры проекта:



Мы должны изменить так:



Как только мы изменим эти настройки, Xcode попросит нас перейти к активам и автоматически создать файл LaunchImage в активах следующим образом:


AppIcon
LaunchImage
LaunchImage

iPads будут использовать `2x` изображения запуска путем масштабирования

2 **12.9 «iPad Pro** : нет квадрата для этого iPad, потому что этот iPad также будет использовать `2x iPad` изображения, масштабируя их

3 **Retina HD 5.5 ":** iPads должен иметь `1920x1080px` для портрета и `1080x1920px` для ландшафта, но Xcode даст waring, и запуск изображения на этих устройствах не будет отображаться

4 **SplitView:** поскольку мы используем `LaunchImage Asset` вместо `LaunchScreen XIB` , наше приложение не будет поддерживать `SplitView` на iPad и ландшафт 5.5 "iPhone

5 **Переустановите:** если наше приложение уже установлено на устройстве, и мы пытаемся запустить с этими недавно добавленными файлами образа запуска, то иногда устройство не будет показывать изображения запуска при запуске приложения. В этом случае просто удалите приложение с устройства, очистите + проект сборки и запустите его, он покажет новые снимки запуска

Прочитайте Использование Image Aseets онлайн: https://riptutorial.com/ru/ios/topic/10087/использование-image-aseets

глава 142: категории

замечания

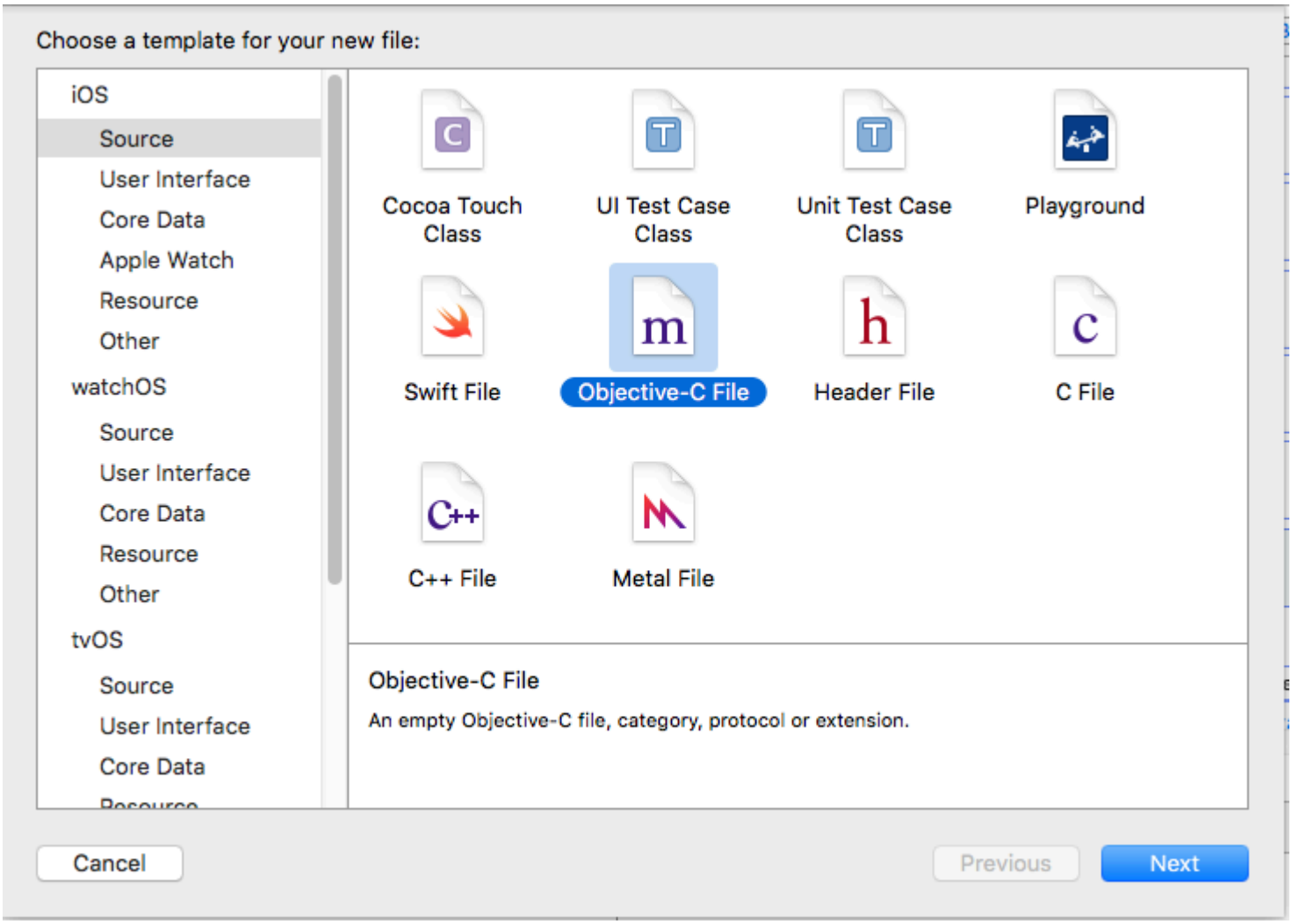
Категории могут использоваться для переопределения методов класса. Даже если метод действительно закрыт. Переопределенный метод недоступен из категории или где-либо еще. Поэтому важно, чтобы при добавлении методов к существующему классу эти методы уже не существовали.

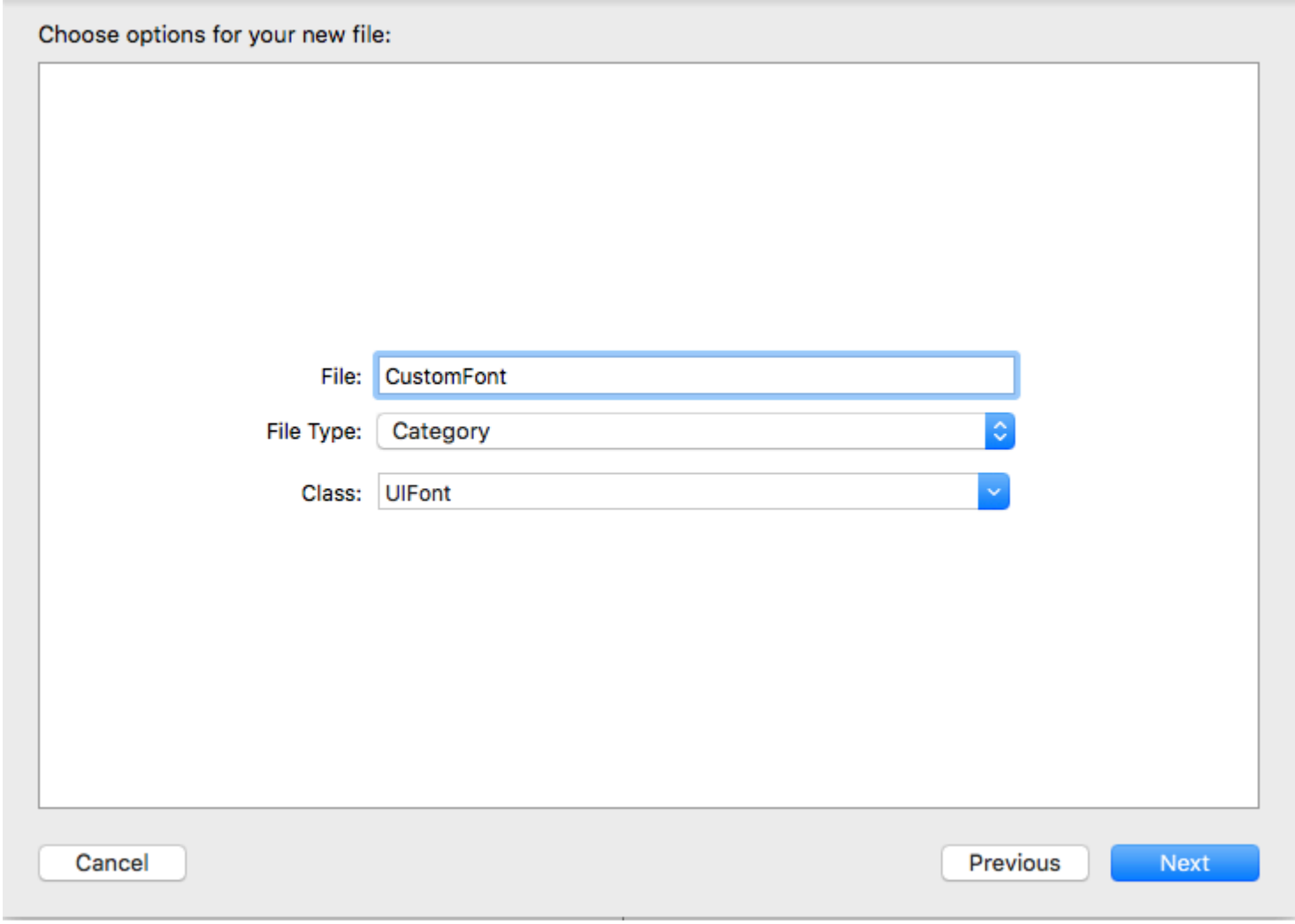
Examples

Создать категорию

Категории предоставляют возможность добавлять дополнительные функции к объекту без подкласса или изменения фактического объекта.

Например, мы хотим установить некоторые пользовательские шрифты. Позволяет создать категорию, добавляющую функциональность в класс `UIFont` . Откройте проект Xcode, нажмите «Файл» -> «Создать» -> «Файл» и выберите «Объектив-C-файл», нажмите «Далее», введите название своей категории, произнесите «CustomFont», выберите тип файла как категорию и класс как UIFont, затем нажмите «Далее», а затем «Создать. "


Choose a template for your new file:
iOS
Source
User Interface
Core Data
Apple Watch
Resource
Other
watchOS
Source
User Interface
Core Data
Resource
Other
tvOS
Source
User Interface
Core Data
Cocoa Touch Class
UI Test Case Class
Unit Test Case Class
Playground
Swift File
Objective-C File
Header File
C File
C++ File
Metal File
Objective-C File
An empty Objective-C file, category, protocol or extension.
Cancel
Previous
Next



Объявить метод категории: -

Нажмите «UIFont + CustomFonts.h», чтобы просмотреть заголовочный файл новой категории. Добавьте следующий код в интерфейс для объявления метода.

```
@interface UIFont (CustomFonts)

+(UIFont *)productSansRegularFontWithSize:(CGFloat)size;

@end
```

Теперь выполните метод категории: -

Нажмите «UIFont + CustomFonts.m», чтобы просмотреть файл реализации категории. Добавьте следующий код для создания метода, который будет устанавливать ProductSansRegular Font.

```
+(UIFont *)productSansRegularFontWithSize:(CGFloat)size{

    return [UIFont fontWithName:@"ProductSans-Regular" size:size];

}
```

Импортируйте свою категорию

```
#import "UIFont+CustomFonts.h"
```

Теперь установите шрифт Label

```
[self.label setFont:[UIFont productSansRegularFontWithSize:16.0]];
```

Прочитайте категории онлайн: https://riptutorial.com/ru/ios/topic/3633/категории

глава 143: Классы классов и адаптивность

замечания

Когда вы создаете адаптивные приложения, имейте в виду ограничения классов размеров: они являются *обобщениями* , а не конкретными руководствами для точных размеров пикселей или устройств. Никогда не пытайтесь определить, на каком устройстве работает ваше приложение, или находится ли он в режиме разделения экрана на основе классов размеров.

Вместо этого сделайте решения высокого уровня макета для класса размера и используйте Auto Layout для изменения точных рамок. (См. Также метод `viewWillTransition(to:with:)` для более точного уведомления о том, насколько большой будет представление контроллера после перехода.)

Examples

Коллекции персонажей

В приложении iOS ваш пользовательский интерфейс может принимать одну из нескольких общих форм и размеров. Они определяются с помощью **классов размера** , которые доступны через **коллекцию признаков** представления или представления контроллера.

Apple определяет два класса размера: **обычный** и **компактный** . Каждый из этих классов размеров доступен на обеих осях устройства (по **горизонтали** и по **вертикали**). Ваше приложение может существовать в любом из этих четырех состояний на протяжении всей его жизни. В качестве сокращения разработчики часто описывают комбинацию классов размеров, говоря или записывая два класса размера, сначала с горизонтальной осью: « Compact / Regular» описывает интерфейс, который является горизонтально компактным, но вертикально регулярным.

В вашем приложении используйте методы в протоколе UITraitEnvironment, чтобы проверить свой текущий класс размеров и отреагировать на изменения:

```
class MyViewController: UIViewController {
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        print("Horizontal size class: \(traitCollection.horizontalSizeClass)")
        print("Vertical size class: \(traitCollection.verticalSizeClass)")
    }

    override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
        super.traitCollectionDidChange(previousTraitCollection)
        print("Trait collection changed; size classes may be different.")
    }
```

```
}
```

Оба UIView и UIViewController соответствуют UITraitEnvironment, поэтому вы можете посмотреть свою текущую коллекцию признаков и обработать изменения в подклассах обоих.

Обновление автоматической компоновки с изменениями коллекции признаков

Создание **адаптивного** приложения, то есть реагирование на изменения класса размера путем изменения макета - часто требует большой помощи от системы автоматического макета. Одним из основных способов адаптации приложений является обновление активных ограничений автоматического макета при изменении класса размера представления.

Например, рассмотрите приложение, которое использует UIStackView для организации двух UILabels. Мы можем захотеть, чтобы эти ярлыки складывались друг на друга в горизонтально компактных средах, но сидели рядом друг с другом, когда у нас было немного больше места в горизонтально-обычных средах.

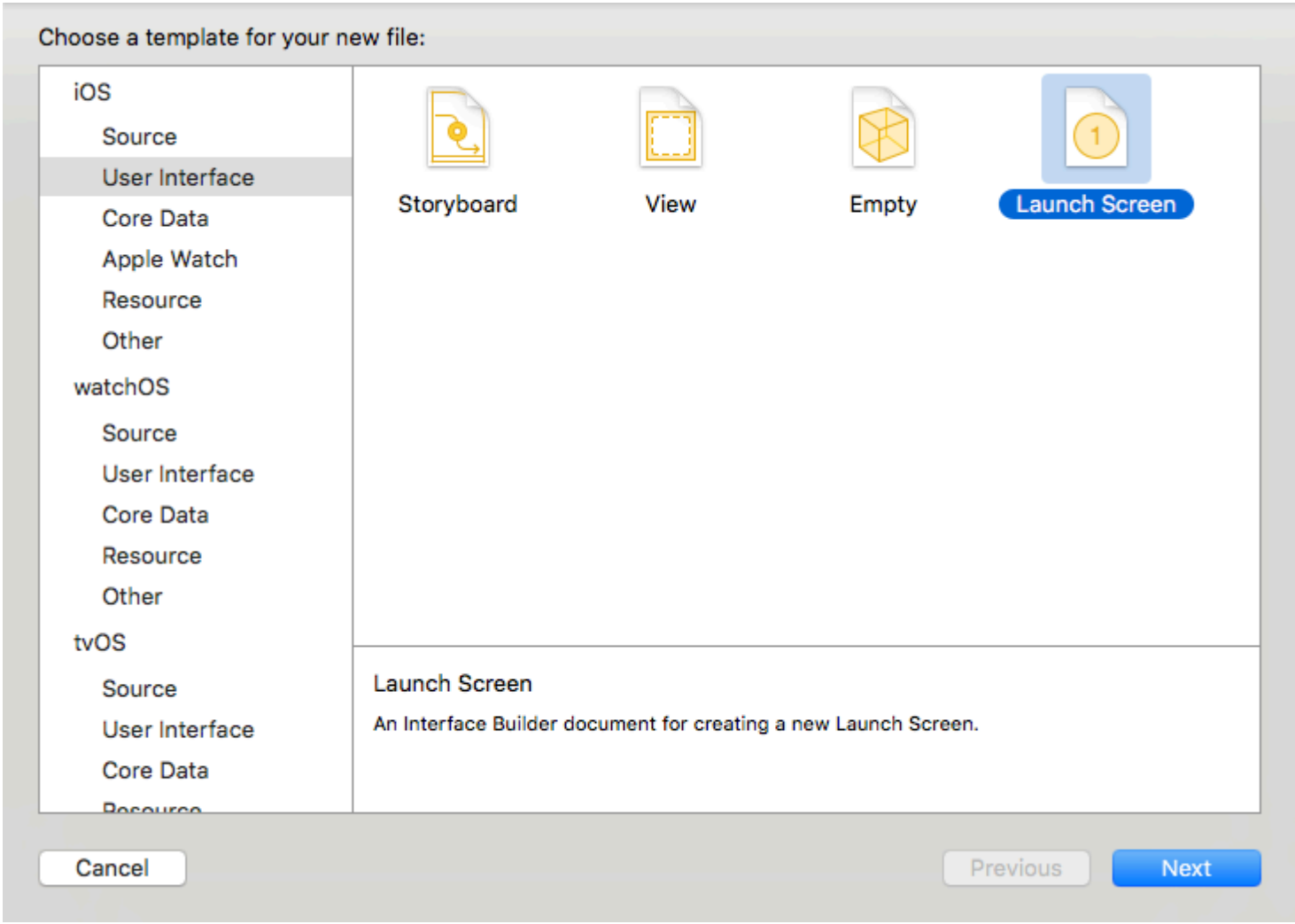
```
class ViewController: UIViewController {
    var stackView: UIStackView!

    override func viewDidLoad() {
        super.viewDidLoad()

        stackView = UIStackView()
        for text in ["foo", "bar"] {
            let label = UILabel()
            label.translatesAutoresizingMaskIntoConstraints = false
            label.text = text
            stackView.addArrangedSubview(label)
        }

        view.addSubview(stackView)
        stackView.translatesAutoresizingMaskIntoConstraints = false
        stackView.centerXAnchor.constraint(equalTo: view.centerXAnchor).isActive = true
        stackView.centerYAnchor.constraint(equalTo: view.centerYAnchor).isActive = true
    }

    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        updateAxis(forTraitCollection: traitCollection)
    }

    override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
        super.traitCollectionDidChange(previousTraitCollection)
        updateAxis(forTraitCollection: traitCollection)
    }

    private func updateAxis(forTraitCollection traitCollection: UITraitCollection) {
        switch traitCollection.horizontalSizeClass {
        case .regular:
```

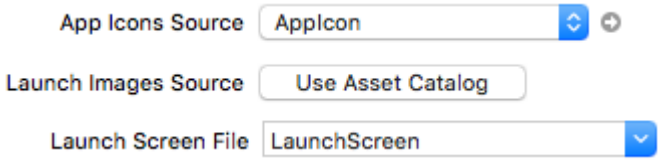
```
            stackView.axis = .horizontal
        case .compact:
            stackView.axis = .vertical
        case .unspecified:
            print("Unspecified size class!")
            stackView.axis = .horizontal
        }
    }
}
```

Поддержка многозадачности iOS на iPad

Ключевым элементом адаптивности в современном приложении iOS является поддержка многозадачности на iPad. По умолчанию приложения, созданные в Xcode 7 и новее, будут настроены на поддержку многозадачности: у них будет файл LaunchScreen.storyboard, который использует автоматический макет.

Самый простой способ для существующих приложений выбрать многозадачность - создать такую раскадровку, а затем установить ее как экран запуска проекта:





После того, как ваше приложение поддерживает многозадачность iPad, проверяйте существующие представления и контроллеры представления, чтобы убедиться, что они используют автоматический макет и могут поддерживать различные комбинации классов размеров.

Прочитайте Классы классов и адаптивность онлайн: https://riptutorial.com/ru/ios/topic/4628/классы-классов-и-адаптивность

глава 144: Классы классов и адаптивность

замечания

Для получения дополнительной информации (классы размера и адаптивность с помощью раскадровки) использования автоматической компоновки для адаптации в iOS мы можем следить за ссылкой на сайт разработчика Apple .

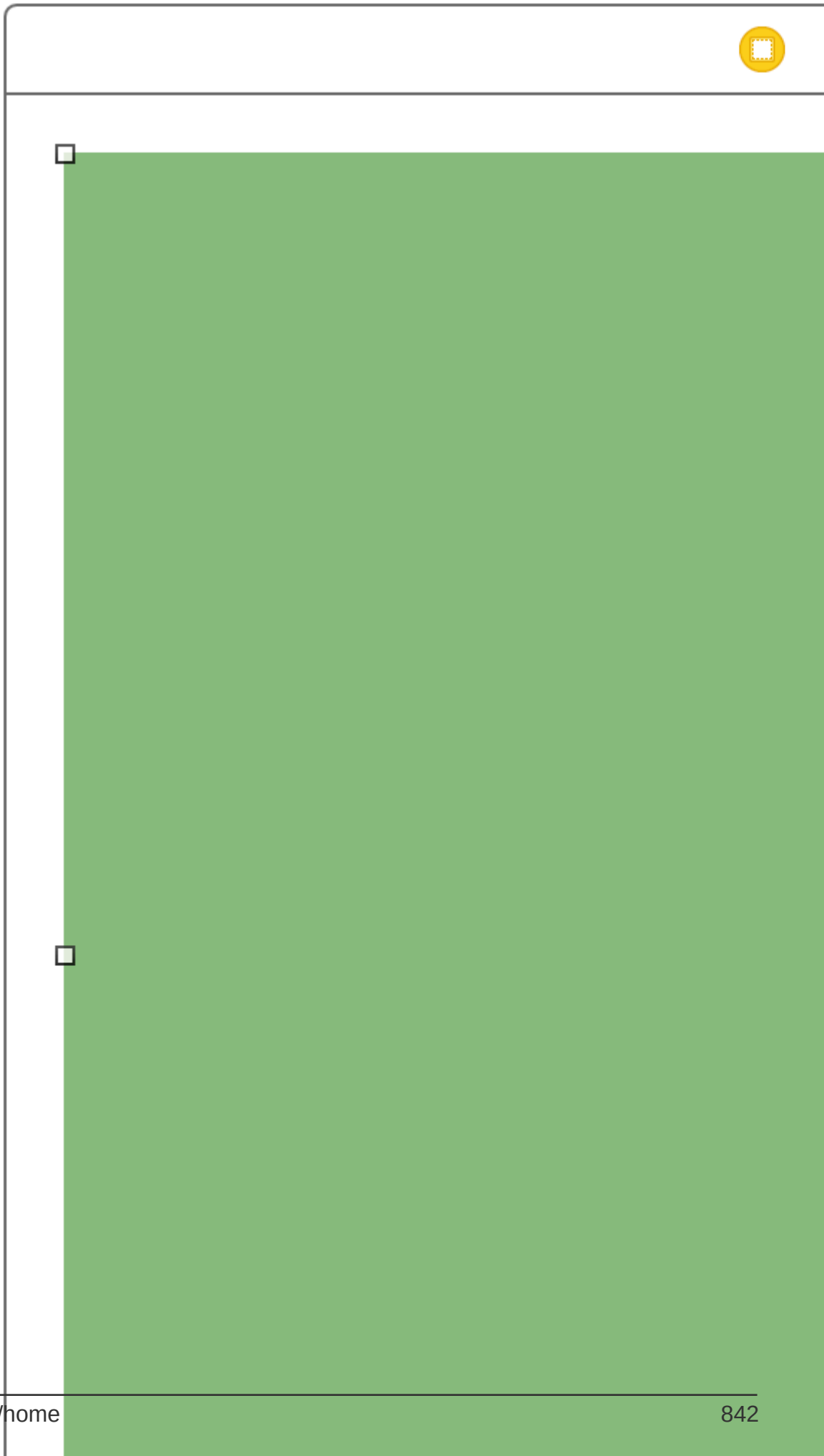
Мы также можем добавлять ограничения **Программно** с использованием **языка Visual Format,** как описано здесь на сайте разработчика Apple .

Examples

Классы размера и адаптивность с помощью раскадровки

Мы можем добавить адаптивность к любому подклассу `UIView` который мы добавляем в контроллер вида в файле nib.
Давайте рассмотрим пример добавления адаптивности с использованием классов размера в представление.

1. Добавьте представление в контроллер просмотра как:



классы-классов-и-адаптивность

глава 145: Кодируемый

Вступление

Codable добавлен с Xcode 9, iOS 11 и Swift 4. Codable используется для кодирования и декодирования ваших типов данных для совместимости с внешними представлениями, такими как JSON.

Codable для поддержки кодирования и декодирования объявляет совместимость с Codable, которая объединяет кодируемые и декодируемые протоколы. Этот процесс известен как способ кодирования ваших типов.

Examples

Использование Codable с JSONEncoder и JSONDecoder в Swift 4

Давайте возьмем пример со структурой фильма, здесь мы определили структуру как Codable. Итак, мы можем легко кодировать и декодировать его.

```
struct Movie: Codable {
    enum MovieGenere: String, Codable {
        case horror, skifi, comedy, adventure, animation
    }

    var name : String
    var moviesGenere : [MovieGenere]
    var rating : Int
}
```

Мы можем создать объект из фильма, например:

```
let upMovie = Movie(name: "Up", moviesGenere: [.comedy , .adventure, .animation], rating : 4)
```

UpMovie содержит имя «Up», а movieGenere - это комедия, приключение и анимация, в которой содержится 4 рейтинга из 5.

шифровать

JSONEncoder - это объект, который кодирует экземпляры типа данных как объекты JSON. JSONEncoder поддерживает объект Codable.

```
// Encode data
let jsonEncoder = JSONEncoder()
do {
    let jsonData = try jsonEncoder.encode(upMovie)
    let jsonString = String(data: jsonData, encoding: .utf8)
```

```
    print("JSON String : " + jsonString!)
}
catch {
}
```

JSONEncoder предоставит нам данные JSON, которые используются для извлечения строки JSON.

Строка вывода будет выглядеть так:

```
{
  "name": "Up",
  "moviesGenere": [
    "comedy",
    "adventure",
    "animation"
  ],
  "rating": 4
}
```

раскодировать

JSONDecoder - это объект, который декодирует экземпляры типа данных из объектов JSON. Мы можем вернуть объект из строки JSON.

```
do {
    // Decode data to object

    let jsonDecoder = JSONDecoder()
    let upMovie = try jsonDecoder.decode(Movie.self, from: jsonData)
    print("Rating : \(upMovie.name)")
    print("Rating : \(upMovie.rating)")
}
catch {
}
```

Дешифрируя JSONData, мы получим объект Movie назад. Таким образом, мы можем получить все значения, которые сохраняются в этом объекте.

Результат будет выглядеть следующим образом:

```
Name : Up
Rating : 4
```

Прочитайте Кодируемый онлайн: https://riptutorial.com/ru/ios/topic/10639/кодируемый

глава 146: Кэширование онлайн-изображений

Examples

AlamofireImage

Кэширование онлайновых изображений с использованием `AlamofireImage` . Он работает на вершине `Alamofire` в Свифте. Установите `AlamofireImage` с помощью `cocoapods`

```
pod 'AlamofireImage', '~> 3.1'
```

Настроить:

1. Импорт `AlamofireImage` и `Alamofire`
2. SetUp кэш изображения: `let imageCache = AutoPurgingImageCache( memoryCapacity: 111_111_111, preferredMemoryUsageAfterPurge: 90_000_000)`
3. Выполнение запроса и добавление изображения в кэш:

```
Alamofire.request(self.nameUrl[i]).responseImage { response in
                    if response.result.value != nil {
                        let image = UIImage(data: response.data!, scale: 1.0)!
                        imageCache.add(image, withIdentifier: self.nameUrl[i])
                    }
    }
```

4. Извлечение изображений из кеша:

```
if let image = imageCache.image(withIdentifier: self.nameUrl[self.a])
        {
            self.localImageView.image = image
        }
```

Для получения дополнительной информации перейдите по этой ссылке

Прочитайте Кэширование онлайн-изображений онлайн:
https://riptutorial.com/ru/ios/topic/9450/кэширование-онлайн-изображений

глава 147: Лидеры игр GameCenter

Examples

Лидеры игр GameCenter

Предпосылки:

1. Аккаунт Apple Developers
2. Настройка лидеров GameCenter с iTunesConnect

Настройка лидеров GameCenter:

1. Войдите в *iTunesConnect*
2. Перейдите в « *Мои приложения»* . Создайте приложение для своего проекта, затем перейдите в « *Особенности»* .
3. Нажмите на *Game Center*
4. Нажмите знак «плюс» рядом с «Лидербордами».
5. Выберите *Single Leaderboard* для типов Leaderboard.
6. Создайте *справочное имя Leaderboard* для справки.
7. Создайте *идентификатор Leaderboard* для вашего приложения, чтобы ссылаться на отчетность.
8. Установите формат оценки в *Integer*
9. Оценка результатов будет *лучшим показателем*
10. Нажмите « *Добавить язык»* и заполните записи.

Скопируйте свой `LeaderboardID` который вы создали, и перейдете к Xcode.

Работа с Xcode

Есть 4 функции, с которыми мы будем работать.

1. Импорт структуры и настройка протоколов
2. Проверка входа пользователя в GameCenter
3. Отчеты о результатах в GameCenter
4. Просмотр лидеров
5. Импорт GameKit `import GameKit` протоколов `GKGameCenterControllerDelegate`
6. Теперь мы хотим проверить, подписан ли пользователь в GameCenter

```
func authenticateLocalPlayer() {
```

```
        let localPlayer = GKLocalPlayer.localPlayer()
        localPlayer.authenticateHandler = { (viewController, error) -> Void in

            if viewController != nil {
                //If the user is not signed in to GameCenter, we make them sign in
                let vc:UIViewController = self.view!.window!.rootViewController!
                vc.presentViewController(viewController!, animated: true, completion: nil)

            } else {

                //Do something here if you want
            }
        }
    }
```

3. Теперь пользователь использует приложение, и внезапно у пользователя новый высокий балл, мы сообщаем о высоком балла, вызывая функцию ниже.

Функция ниже hols 2 параметра.

`Identifier` который определен как строка и используется для ввода вашего идентификатора руководителя, который вы создали в iTunesConnect.

`score` которая определяется как Int, которая будет оцениваться пользователями для iTunesConnect

```
func saveHighScore(identifier:String, score:Int) {

        if GKLocalPlayer.localPlayer().authenticated {

            let scoreReporter = GKScore(leaderboardIdentifier: identifier)

            scoreReporter.value = Int64(score)

            let scoreArray:[GKScore] = [scoreReporter]

            GKScore.reportScores(scoreArray, withCompletionHandler: {
                error -> Void in

                if error != nil {
                    print("Error")
                } else {


                }
            })
        }
    }
```

4. Теперь, если пользователь хочет просмотреть таблицы лидеров, вызовите функцию ниже

```
//This function will show GameCenter leaderboards and Achievements if you call this function.
    func showGameCenter() {
```

```
        let gameCenterViewController = GKGameCenterViewController()
        gameCenterViewController.gameCenterDelegate = self

        let vc:UIViewController = self.view!.window!.rootViewController!
        vc.presentViewController(gameCenterViewController, animated: true, completion:nil)

    }

    //This function closes gameCenter after showing.
    func gameCenterViewControllerDidFinish(gameCenterViewController:
GKGameCenterViewController) {

        gameCenterViewController.dismissViewControllerAnimated(true, completion: nil)
        self.gameCenterAchievements.removeAll()

    }
```

Прочитайте Лидеры игр GameCenter онлайн: https://riptutorial.com/ru/ios/topic/6720/лидеры-игр-gamecenter

глава 148: локализация

Вступление

Локализация - это функция, предоставляемая iOS, которая переводит ваше приложение на несколько языков. Для **локализации** необходима **интернационализация** . **Интернационализация** - это процесс создания приложения iOS для адаптации различных культур, языка и регионов.

Examples

Локализация в iOS

Создайте отдельный файл `Localizable.strings` для каждого языка. Правая сторона будет отличаться для каждого языка. Подумайте об этом как о ключе-значении:

```
"str" = "str-language";
```

Access str в Objective-C:

```
//Try to provide description on the localized string to be able to create a proper
documentation if needed
NSString *str = NSLocalizedString(@"string", @"description of the string");
```

Доступ к строке в Swift:

```
let str = NSLocalizedString("string", comment: "language");
```

Прочитайте локализация онлайн: https://riptutorial.com/ru/ios/topic/1579/локализация

глава 149: Многоадресные делегаты

Вступление

Шаблон для добавления возможностей многоадресной рассылки к существующим элементам управления iOS. Добавление многоадресной рассылки позволяет улучшить четкость и повторное использование кода.

Examples

Многоадресные делегаты для любых элементов управления

Пересылать сообщения одному объекту другому делегатам, многоадресная передача этих сообщений нескольким наблюдателям.

Шаг 1: - Создание `NSObject` **класс** `RRMulticastDelegate`

Шаг 2: - После выполнения кода в файле `RRMulticastDelegate.h`

```
#import <Foundation/Foundation.h>

@interface RRMulticastDelegate : NSObject

{
    //Handle multiple observers of delegate
    NSMutableArray* _delegates;
}

// Delegate method implementation to the list of observers
- (void)addDelegate:(id)delegate;
- (void)removeDelegate:(id)delegate;

// Get multiple delegates
-(NSArray *)delegatesObjects;

@end
```

Шаг 3: - После выполнения кода в файле `RRMulticastDelegate.m`

```
#import "RRMulticastDelegate.h"

@implementation RRMulticastDelegate

- (id)init
{
    if (self = [super init])
    {
        _delegates = [NSMutableArray array];
    }
    return self;
```

```
}

-(NSArray *)delegatesObjects
{
    return _delegates;
}

- (void)removeDelegate:(id)delegate
{
    if ([_delegates containsObject:delegate])
        [_delegates removeObject:delegate];
}

- (void)addDelegate:(id)delegate
{
    if (![_delegates containsObject:delegate])
        [_delegates addObject:delegate];
}

- (BOOL)respondsToSelector:(SEL)aSelector
{
    if ([super respondsToSelector:aSelector])
        return YES;

    // if any of the delegates respond to this selector, return YES
    for(id delegate in _delegates)
    {
        if (!delegate)
            continue;

        if ([delegate respondsToSelector:aSelector])
        {
            return YES;
        }
    }
    return NO;
}

- (NSMethodSignature *)methodSignatureForSelector:(SEL)aSelector
{
    // can this class create the sinature?
    NSMethodSignature* signature = [super methodSignatureForSelector:aSelector];

    // if not, try our delegates
    if (!signature)
    {
        for(id delegate in _delegates)
        {
            if (!delegate)
                continue;

            if ([delegate respondsToSelector:aSelector])
            {
                return [delegate methodSignatureForSelector:aSelector];
            }
        }
    }
    return signature;
}

- (void)forwardInvocation:(NSInvocation *)anInvocation
```

```
{
    // forward the invocation to every delegate
    for(id delegate in _delegates)
    {
        if (!delegate)
            continue;

        if ([delegate respondsToSelector:[anInvocation selector]])
        {
            [anInvocation invokeWithTarget:delegate];
        }
    }
}

@end
```

Шаг 4: - Создание `NSObject` **категории класса** `RRProperty`

Шаг 5: - После реализации кода в файле `NSObject+RRProperty.h`

```
#import <Foundation/Foundation.h>

#import "RRMulticastDelegate.h"

@interface NSObject (RRProperty)<UITextFieldDelegate,UITableViewDataSource>

-(void)setObject:(id)block forKey:(NSString *)key;
-(id)objectForKey:(NSString *)key;

#pragma mark - Multicast Delegate

- (RRMulticastDelegate *)multicastDelegate;
- (RRMulticastDelegate *)multicastDatasource;

-(void)addDelegate:(id)delegate;
-(void)addDataSource:(id)datasource;

@end
```

Шаг 6: - После реализации кода в файле `NSObject+RRProperty.m`

```
#import "NSObject+RRProperty.h"

#import <objc/message.h>
#import <objc/runtime.h>

#pragma GCC diagnostic ignored "-Wprotocol"

static NSString *const MULTICASTDELEGATE = @"MULTICASTDELEGATE";
static NSString *const MULTICASTDATASOURCE = @"MULTICASTDATASOURCE";

@implementation NSObject (RRProperty)

-(void)setObject:(id)block forKey:(NSString *)key
{
    objc_setAssociatedObject(self, (__bridge const void *)(key), block,
OBJC_ASSOCIATION_RETAIN);
}
```

```
-(id)objectForKey:(NSString *)key
{
    return objc_getAssociatedObject(self, (__bridge const void *)(key));
}

#pragma mark - Multicast Delegate

- (RRMulticastDelegate *)multicastDelegate
{
    id multicastDelegate = [self objectForKey:MULTICASTDELEGATE];
    if (multicastDelegate == nil) {
        multicastDelegate = [[RRMulticastDelegate alloc] init];

        [self setObject:multicastDelegate forKey:MULTICASTDELEGATE];
    }

    return multicastDelegate;
}

- (RRMulticastDelegate *)multicastDatasource
{
    id multicastDatasource = [self objectForKey:MULTICASTDATASOURCE];
    if (multicastDatasource == nil) {
        multicastDatasource = [[RRMulticastDelegate alloc] init];

        [self setObject:multicastDatasource forKey:MULTICASTDATASOURCE];
    }

    return multicastDatasource;
}

-(void)addDelegate:(id)delegate
{
    [self.multicastDelegate addDelegate:delegate];

    UITextField *text = (UITextField *) self;
    text.delegate = self.multicastDelegate;
}

-(void)addDataSource:(id)datasource
{
    [self.multicastDatasource addDelegate:datasource];
    UITableView *text = (UITableView *) self;
    text.dataSource = self.multicastDatasource;
}

@end
```

Наконец, вы можете использовать multicast dalegate для любых элементов управления ...

Для ...

Импортируйте свой класс `NSObject+RRProperty.h` файл `NSObject+RRProperty.h` чтобы получить доступ к его методам для **установки многоадресного делегирования** / **источника данных** .

```
UITextView *txtView = [[UITextView alloc]initWithFrame:txtframe];
```

```
[txtView addDelegate:self];

UITableView *tblView = [[UITableView alloc]initWithFrame:tblframe];
[tblView addDelegate:self];
[tblView addDataSource:self];
```

Прочитайте Многоадресные делегаты онлайн: https://riptutorial.com/ru/ios/topic/10081/многоадресные-делегаты

глава 150: Наблюдение за ключевыми значениями

замечания

KVC : - кодирование с ключом

Обычно переменные экземпляра получают доступ через свойства или аксессоры, но KVC дает другой способ доступа к переменным в виде строк. Таким образом, ваш класс действует как словарь, а ваше имя свойства, например, «возраст» становится ключевым, а значение, которое свойство сохраняет, становится значением для этого ключа.

```
For example, you have employee class with "age" property. Normally we access like this.
emp.age = @”20”;
NSString age = emp.age;

But KVC works like this:
[emp valueForKey:@"age"];
[emp setValue:@"25" forKey:@"age"];
```

KVO : - Наблюдатель по ключевым значениям

Механизм, посредством которого объекты уведомляются при изменении какого-либо свойства, называется KVO. Ex.: уведомление о клавиатуре

> Например, объект person заинтересован в получении уведомления, когда свойство AccountBalance изменено в объекте BankAccount. Чтобы достичь этого, Person Object должен зарегистрироваться в качестве наблюдателя в свойстве AccountAccount AccountBalance, отправив addObserver: forKeyPath: options: context: message.

Examples

Использование контекста для наблюдения KVO

```
-(void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object
change:(NSDictionary<NSString *,id> *)change context:(void *)context
```

Контекст важен, если вы отправляете свой класс для использования другими пользователями. Контекст позволяет наблюдателю вашего класса проверять, что его вызывающий наблюдатель.

Проблема с пропуском наблюдателя заключается в том, что если какой-то один подкласс вашего класса и зарегистрировать наблюдателя для одного и того же объекта, тот же

ключ и он не передает контекст, тогда суперкласс-наблюдатель можно назвать многократным.

Хорошим контекстом является переменная, уникальная и внутренняя для вашего использования.

Для дополнительной информации.

важность и хороший контекст

Наблюдение свойства подкласса NSObject

Большинство функций KVO и KVC уже реализованы по умолчанию во всех подклассах `NSObject` .

Чтобы начать наблюдать свойство с именем `firstName` объекта с именем `personObject` выполните это в классе наблюдения:

```
[personObject addObserver:self
               forKeyPath:@"firstName"
                  options:NSKeyValueObservingOptionNew
                  context:nil];
```

Объект, который `self` в вышеприведенном коде ссылается, затем получит сообщение `observeValueForKeyPath:ofObject:change:context:` при каждом изменении пути отслеживаемого ключа.

```
- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary<NSString *,id> *)change
                       context:(void *)context
{
    NSLog(@"new value of %@ is: %@", keyPath, change[NSKeyValueChangeNewKey]);
}
```

«Ключевой путь» - это термин KBK. Подклассы `NSObject` реализуют функциональность KVC по умолчанию.

Переменная экземпляра с именем `_firstName` будет доступна с помощью ключевого пути `@"firstName"` .

Метод getter с именем `firstName` будет вызываться при доступе к ключевому пути `@"firstName"` , независимо от `_firstName` переменной экземпляра `setFirstName` или `setFirstName` .

Прочитайте Наблюдение за ключевыми значениями онлайн:
https://riptutorial.com/ru/ios/topic/3493/наблюдение-за-ключевыми-значениями

глава 151: Настройка маяков с помощью CoreBluetooth

Вступление

Горячая запись и запись данных на устройство с низким энергопотреблением.

замечания

Некоторые важные моменты

- Никаких возможностей не требуется.
- iPhone хранит байты в формате Little Endian, поэтому проверьте, использует ли bluetooth-аксессуар Little Endian. Пример:
  - Intel CPU обычно использует немного endian.
  - Архитектура ARM была миниатюрной до версии 3, когда она стала широкоформатной.
- После однократной или пакетной операции соединение будет потеряно, поэтому перед продолжением необходимо снова подключиться.

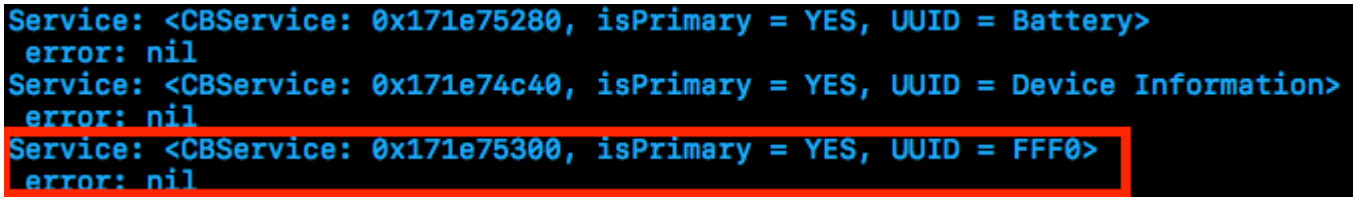
Сканировать для SERVICE UUID

```
func SearchBLE(){
    cb_manager.scanForPeripherals(withServices:[service_uuid], options: nil)
    StopSearchBLE()
}
```

Как открыть SERVICE UUID без документации

```
func centralManager(_ central: CBCentralManager, didConnect peripheral:
CBPeripheral) {
        peripheral.delegate = self
        peripheral.discoverServices(nil)
}

func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
    for service in peripheral.services! {
        print("Service: \(service)\n error: \(error)")
    }
}
```

- DiscoverServices (nil) - NIL означает, что все службы будут возвращены, что не является хорошим вариантом. (READ Remarks 3)
- Если вы еще не обнаружили, что SERVICE UUID запускает ваш код и ищет консоль



- Я обнаружил, что у меня есть 3 службы: Аккумулятор, Информация об устройстве (Firmware) и FFF0
- Этот сервис uuid не является стандартным, список со стандартами можно найти здесь
- FFF0 - это SERVICE UUID в этом случае

Преобразование данных в UInt16 и наоборот

Добавьте эти расширения в свой класс

```
protocol DataConvertible {
    init?(data: Data)
    var data: Data { get }
}

extension DataConvertible {

    init?(data: Data) {
        guard data.count == MemoryLayout<Self>.size else { return nil }
        self = data.withUnsafeBytes { $0.pointee }
    }

    var data: Data {
        var value = self
        return Data(buffer: UnsafeBufferPointer(start: &value, count: 1))
    }
}
extension UInt16 : DataConvertible {
    init?(data: Data) {
        guard data.count == MemoryLayout<UInt16>.size else { return nil }
        self = data.withUnsafeBytes { $0.pointee }
    }
    var data: Data {
        var value = CFSwapInt16HostToBig(self)
        return Data(buffer: UnsafeBufferPointer(start: &value, count: 1))
    }
}
```

Examples

Отображение имен всех Bluetooth Low Energy (BLE)

- В этом примере у меня есть контролируемая комната с одним устройством BLE.
- Ваш класс должен расширить CBCentralManagerDelegate.
- Внедрите метод: centralManagerDidUpdateState (_ central: CBCentralManager).
- Используйте глобальную очередь, чтобы не затормозить экран во время поиска устройства.
- Выполните активацию CBCentralManager и дождитесь ответа callManagerDidUpdateState.

```
class BLEController: CBCentralManagerDelegate{

var cb_manager: CBCentralManager!
var bles : [CBPeripheral] = []

    override func viewDidLoad() {
        super.viewDidLoad()
        cb_manager = CBCentralManager(delegate: self, queue: DispatchQueue.global())
    }

    func centralManagerDidUpdateState(_ central: CBCentralManager) {
        print("UPDATE STATE - \(central)")
    }
}
```

Обратный вызов centralManagerDidUpdateState указывает, что CoreBluetooth готов, поэтому вы можете искать BLE сейчас. Обновите код centralManagerDidUpdateState для поиска всех устройств BLE, когда они будут готовы.

```
func centralManagerDidUpdateState(_ central: CBCentralManager) {
    print("UPDATE STATE - \(central)")
    SearchBLE()
}

func SearchBLE(){
    cb_manager.scanForPeripherals(withServices: nil, options: nil)
    StopSearchBLE()
}

func StopSearchBLE() {
    let when = DispatchTime.now() + 5 // change 5 to desired number of seconds
    DispatchQueue.main.asyncAfter(deadline: when) {
        self.cb_manager.stopScan()
    }
}
```

- SearchBLE () выполняет поиск устройств BLE и прекращает поиск после 5 секунд
- cb_manager.scanForPeripherals (withServices: nil, options: nil) ищет все BLE в диапазоне с вами.
- StopSearchBLE () остановит поиск после 5 секунд.
- Каждый найденный BLE будет callback func centralManager (_ central: CBCentralManager, didDiscover периферийное: CBPeripheral, advertData: [String: Any], rssi RSSI: NSNumber)

```
func centralManager(_ central: CBCentralManager, didDiscover peripheral:

    CBPeripheral, advertisementData: [String : Any], rssi RSSI: NSNumber) {
    guard let name = peripheral.name else {
        return
    }
    print(name)
    bles.append(peripheral)
}
```

Подключить и прочитать главное значение

- Я в контролируемой комнате с одним маяком, который использует протокол IBEACON.
- BLEController необходимо расширить CBPeripheralDelegate
- Я буду использовать первый BLE для подключения после остановки поиска.
- Измените метод StopSearchBLE ()

```
class BLEController: CBCentralManagerDelegate, CBPeripheralDelegate{
//...
    func StopSearchMiniewBeacon() {
        let when = DispatchTime.now() + 5 // change 2 to desired number of seconds
        DispatchQueue.main.asyncAfter(deadline: when) {
            self.cb_manager.stopScan()
            self.cb_manager.connect(bles.first)
        }
    }
/...
}
```

- В документации вашего устройства BLE вы должны искать SERVICE UUID и MAJOR UUID ХАРАКТЕРИСТИКА

```
var service_uuid =  CBUUID(string: "0000fff0-0000-1000-8000-00805f9b34fb")
var major_uuid =  CBUUID(string: "0000fff2-0000-1000-8000-00805f9b34fb")
func centralManager(_ central: CBCentralManager, didConnect peripheral:
CBPeripheral) {
    peripheral.delegate = self
    peripheral.discoverServices([service_uuid])
}

func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
    print("Service: \(service)\n error: \(error)")
    peripheral.discoverCharacteristics([major_uuid], for: (peripheral.services?[0])!)
}
```

- Создайте переменную 'service_uuid' и 'major_uuid', как показано выше. '-0000-1000-8000-00805f9b34fb' является частью стандарта. 'fff0' - мой СЕРВИСНЫЙ UUID, 'fff2' - моя главная характеристика UUID, и '0000' необходимо заполнить блок размером 4 байта uuid 1º.
- discoverCharacteristics ([major_uuid], для: (peripheral.services?[0])!) будет получать основные характеристики с моего gatt-сервера устройства, и на данный момент он будет иметь значение NIL.

- (Peripheral.services?[0])! - 0 beacuse вернет одно значение после того, как я сделал периферийное. DiscoverServices ([service_uuid])

```
func peripheral(_ peripheral: CBPeripheral, didDiscoverCharacteristicsFor service: CBService,
error: Error?) {
    for characteristic in service.characteristics! {
        print("Characteristic: \(characteristic)\n error: \(error)")
        if(characteristic.uuid.uuidString == "FFF2"){
            peripheral.readValue(for: characteristic)
        }
    }
}

func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBCharacteristic, error: Error?) {
    print("Characteristic read: \(characteristic)\n error: \(error)")
    let major = UInt16.init(bigEndian: UInt16(data: characteristic.value!)!)
    print("major: \(major)")
}
```

- Характеристическое значение будет читаемым только после вызова peripheral.readValue (для: характеристики)
- readValue приведет к периферийной функции func (_ периферийная: CBPeripheral, didUpdateValueFor характеристика: CBCharacteristic, error: Error?) со значением в типе данных.

Напишите основное значение

- Вам необходимо открыть службы и характеристики
- Перед тем, как написать над ним, вам не нужно считывать значение с характеристики.
- будет продолжен для этого примера после значения чтения. Изменить func периферийное (_ периферийное: CBPeripheral, didUpdateValueДля характеристики: CBCharacteristic, ошибка: ошибка?)
- Добавьте переменную new_major и reset_characteristic

```
var reset_characteristic : CBCharacteristic!
func peripheral(_ peripheral: CBPeripheral, didDiscoverCharacteristicsFor service: CBService,
error: Error?) {
    for characteristic in service.characteristics! {
        print("Characteristic: \(characteristic)\n error: \(error)")
        if(characteristic.uuid.uuidString == "FFF2"){
            peripheral.readValue(for: characteristic)
        }
        if(characteristic.uuid.uuidString == "FFFF"){
            reset_characteristic = characteristic
        }
    }
}
let new_major : UInt16 = 100
func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBCharacteristic, error: Error?) {
    print("Characteristic read: \(characteristic)\n error: \(error)")
```

```
    let major = UInt16.init(bigEndian: UInt16(data: characteristic.value!)!)
    print("major: \(major)")
    peripheral.writeValue(new_major.data, for: characteristic, type:
CBCharacteristicWriteType.withResponse)
}
```

- iPhone by deafult отправит и получит байты в формате Little Endian, но мое устройство MINEW witch чипсет NRF51822 имеет ARM archteture и нуждается в байтах в формате Big Endian, поэтому мне нужно его поменять.
- Документация BLE Device сообщит, какой тип ввода и вывода будет иметь каждый признак, и если вы можете прочитать его, как указано выше (CBCharacteristicWriteType.withResponse).

```
func peripheral(_ peripheral: CBPeripheral, didWriteValueFor characteristic: CBCharacteristic,
error: Error?) {
    print("Characteristic write: \(characteristic)\n error: \(error)")
    if(characteristic.uuid.uuidString == "FFF2"){
            print("Resetting")
            peripheral.writeValue("minew123".data(using: String.Encoding.utf8)!, for:
reset_characteristic, type: CBCharacteristicWriteType.withResponse)
        }
    if(characteristic.uuid.uuidString == "FFFF"){
        print("Reboot finish")
        cb_manager.cancelPeripheralConnection(peripheral)
    }
}
```

- Чтобы обновить информацию о gatt-сервере, вам необходимо перезагрузить его программным способом или сохранить данные на нем, выключить и включить вручную.
- FFFF характерен для этого устройства.
- «minew123» - это пароль по умолчанию для перезагрузки. о сохранение информации в этом случае.
- запустите приложение и наблюдайте за консолью за любую ошибку, я надеюсь, что нет, но вы еще не увидите новое значение.

```
func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBCharacteristic, error: Error?) {
    print("Characteristic read: \(characteristic)\n error: \(error)")
    let major = UInt16.init(bigEndian: UInt16(data: characteristic.value!)!)
    print("major: \(major)")
    //peripheral.writeValue(new_major.data, for: characteristic, type:
CBCharacteristicWriteType.withResponse)
```

}

- Последний шаг - прокомментировать последнюю строку в методе didUpdateValueFor и повторно запустить приложение, теперь вы получите новое значение.

Прочитайте Настройка маяков с помощью CoreBluetooth онлайн:
https://riptutorial.com/ru/ios/topic/9488/настройка-маяков-с-помощью-corebluetooth

глава 152: область

замечания

Добавление нового RLMObject к существующему царству - схема и миграция

Добавление новых классов моделей в Realm не требует миграции или изменения схемы; только внося изменения в существующее Царство.

Examples

Класс базовой модели RLMObject с основным ключом - Objective-C

Пример базового класса модели RLMObject, который использует первичный ключ и некоторые общие свойства по умолчанию. Затем подклассы могут задавать метаданные, соответствующие их потребностям.

```
@interface BaseModel : RLMObject

@property NSString *uuid;
@property NSString *metadata;

@end

@implementation BaseModel

+ (NSString *)primaryKey
{
    return @"uuid";
}

+ (NSDictionary *)defaultPropertyValues
{
    NSMutableDictionary *defaultPropertyValues = [NSMutableDictionary
dictionaryWithDictionary:[super defaultPropertyValues]];
    NSString *uuid = [[NSUUID UUID] UUIDString];
    [defaultPropertyValues setValue:@"" forKey:@"metadata"];
    [defaultPropertyValues setValue:uuid forKey:@"uuid"];
    return defaultPropertyValues;
}

+ (NSArray *)ignoredProperties
{
    return @[];
}

@end
```

Прочитайте область онлайн: https://riptutorial.com/ru/ios/topic/4084/область

глава 153: Обмен сообщениями FCM в Swift

замечания

FCM: https://firebase.google.com/docs/cloud-messaging/ios/client

Examples

Инициализировать FCM в Swift

следуйте ниже, чтобы добавить FCM в ваш быстрый проект

1- Если у вас еще нет проекта Xcode, создайте его сейчас. Создайте подфайл, если у вас его нет:

```
$ cd каталог вашего проекта
$ pod init
```

2- Добавьте стручки, которые вы хотите установить. Вы можете включить Pod в свой подфайл так:

```
pod 'Firebase / Core'
pod 'Firebase / Messaging'
```

3- Установите контейнеры и откройте файл .xcworkspace, чтобы увидеть проект в Xcode.

```
$ pod install
$ open your-project.xcworkspace
```

4- Загрузите файл GoogleService-Info.plist из plist и включите его в свое приложение.

5- Загрузить сертификат APN для Firebase. APN Cert

6 - добавьте «import Firebase» в файл appDelegate проекта

7 - добавьте этот «FIRApp.configure ()» в свое приложение: didFinishLaunchingWithOptions »

8-регистр для удаленного уведомления

```
if #available(iOS 10.0, *) {
let authOptions : UNAuthorizationOptions = [.Alert, .Badge, .Sound]
UNUserNotificationCenter.currentNotificationCenter().requestAuthorizationWithOptions(
  authOptions,
  completionHandler: {_,_ in })

// For iOS 10 display notification (sent via APNS)
UNUserNotificationCenter.currentNotificationCenter().delegate = self
```

```
  // For iOS 10 data message (sent via FCM)
  FIRMessaging.messaging().remoteMessageDelegate = self

} else {
  let settings: UIUserNotificationSettings =
  UIUserNotificationSettings(forTypes: [.Alert, .Badge, .Sound], categories: nil)
  application.registerUserNotificationSettings(settings)
}

application.registerForRemoteNotifications()
```

9 - для использования токена регистра

```
let token = FIRInstanceID.instanceID().token()!
```

10-, и если вы хотите, чтобы монитор для изменения токена использовал ниже код в файле appDelegate

```
func tokenRefreshNotification(notification: NSNotification) {
if let refreshedToken = FIRInstanceID.instanceID().token() {
    print("InstanceID token: \(refreshedToken)")
  }

  // Connect to FCM since connection may have failed when attempted before having a token.
  connectToFcm()
}
```

11- для получения сообщения из fcm добавить код ниже в appDelegate

```
func connectToFcm() {
  FIRMessaging.messaging().connectWithCompletion { (error) in
    if (error != nil) {
      print("Unable to connect with FCM. \(error)")
    } else {
      print("Connected to FCM.")
    }
  }
}
```

12- и для отключения

```
func applicationDidEnterBackground(application: UIApplication) {
  FIRMessaging.messaging().disconnect()
  print("Disconnected from FCM.")
}
```

в вашем приложении appDelegate.

завершение инициализации и клиент, готовый получить сообщение с панели fcm или отправить маркером с стороннего сервера

Прочитайте Обмен сообщениями FCM в Swift онлайн: https://riptutorial.com/ru/ios/topic/7326/обмен-сообщениями-fcm-в-swift

глава 154: Обнаружение лица с использованием CoreImage / OpenCV

Examples

Обнаружение лиц и функций

Objective-C

Импортируйте следующее в ViewController

```
#import <CoreImage/CoreImage.h>
#import <CoreImage/CoreImage.h>
#import <QuartzCore/QuartzCore.h>
```

Вызовите функцию

```
[self faceDetector];
```

Определение функции:

```
-(void)faceDetector
{
    // Load the picture for face detection
    UIImageView* image = [[UIImageView alloc] initWithImage:[UIImage
imageNamed:@"download.jpeg"]];

    // Draw the face detection image
    [self.view addSubview:image];

    // Execute the method used to markFaces in background
    [self performSelectorInBackground:@selector(markFaces:) withObject:image];

    // flip image on y-axis to match coordinate system used by core image
    [image setTransform:CGAffineTransformMakeScale(1, -1)];

    // flip the entire window to make everything right side up
    [self.view setTransform:CGAffineTransformMakeScale(1, -1)];


}
```

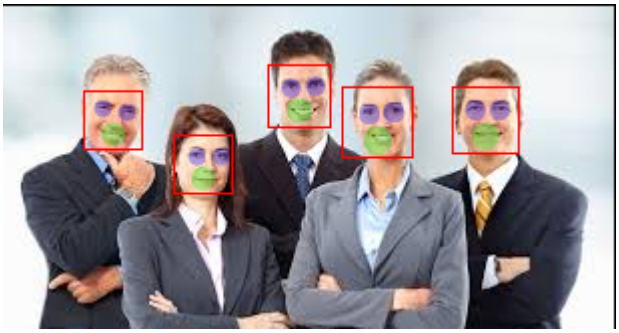
Функция Mark Face

```
//Adds face squares and color masks to eyes and mouth
-(void)markFaces:(UIImageView *)facePicture
{
    // draw a CI image with the previously loaded face detection picture
    CIImage* image = [CIImage imageWithCGImage:facePicture.image.CGImage];
```

```
    // create a face detector - since speed is not an issue we'll use a high accuracy
    // detector
    CIDetector* detector = [CIDetector detectorOfType:CIDetectorTypeFace
                                              context:nil options:[NSDictionary
dictionaryWithObject:CIDetectorAccuracyHigh forKey:CIDetectorAccuracy]];

    // create an array containing all the detected faces from the detector
    NSArray* features = [detector featuresInImage:image];
    NSLog(@"Number of faces %d",[features count]);

    // we'll iterate through every detected face.  CIFaceFeature provides us
    // with the width for the entire face, and the coordinates of each eye
    // and the mouth if detected.  Also provided are BOOL's for the eye's and
    // mouth so we can check if they already exist.
    //    for (features in image)
    //    {
    for(CIFaceFeature* faceFeature in features)
    {
        // get the width of the face
        CGFloat faceWidth = faceFeature.bounds.size.width;

        // create a UIView using the bounds of the face
        UIView* faceView = [[UIView alloc] initWithFrame:faceFeature.bounds];

        // add a border around the newly created UIView
        faceView.layer.borderWidth = 1;
        faceView.layer.borderColor = [[UIColor redColor] CGColor];

        // add the new view to create a box around the face
        [self.view addSubview:faceView];

        if(faceFeature.hasLeftEyePosition)
        {
            // create a UIView with a size based on the width of the face
            UIView* leftEyeView = [[UIView alloc]
initWithFrame:CGRectMake(faceFeature.leftEyePosition.x-faceWidth*0.15,
faceFeature.leftEyePosition.y-faceWidth*0.15, faceWidth*0.3, faceWidth*0.3)];
            // change the background color of the eye view
            [leftEyeView setBackgroundColor:[[UIColor blueColor]
colorWithAlphaComponent:0.3]];
            // set the position of the leftEyeView based on the face
            [leftEyeView setCenter:faceFeature.leftEyePosition];
            // round the corners
            leftEyeView.layer.cornerRadius = faceWidth*0.15;
            // add the view to the window
            [self.view addSubview:leftEyeView];
        }

        if(faceFeature.hasRightEyePosition)
        {
            // create a UIView with a size based on the width of the face
            UIView* leftEye = [[UIView alloc]
initWithFrame:CGRectMake(faceFeature.rightEyePosition.x-faceWidth*0.15,
faceFeature.rightEyePosition.y-faceWidth*0.15, faceWidth*0.3, faceWidth*0.3)];
            // change the background color of the eye view
            [leftEye setBackgroundColor:[[UIColor blueColor] colorWithAlphaComponent:0.3]];
            // set the position of the rightEyeView based on the face
            [leftEye setCenter:faceFeature.rightEyePosition];
            // round the corners
            leftEye.layer.cornerRadius = faceWidth*0.15;
```

```
            // add the new view to the window
            [self.view addSubview:leftEye];
        }

        if(faceFeature.hasMouthPosition)
        {
            // create a UIView with a size based on the width of the face
            UIView* mouth = [[UIView alloc]
initWithFrame:CGRectMake(faceFeature.mouthPosition.x-faceWidth*0.2,
faceFeature.mouthPosition.y-faceWidth*0.2, faceWidth*0.4, faceWidth*0.4)];
            // change the background color for the mouth to green
            [mouth setBackgroundColor:[[UIColor greenColor] colorWithAlphaComponent:0.3]];
            // set the position of the mouthView based on the face
            [mouth setCenter:faceFeature.mouthPosition];
            // round the corners
            mouth.layer.cornerRadius = faceWidth*0.2;
            // add the new view to the window
            [self.view addSubview:mouth];
        }
    }

    // }
}
```

Симулятор ScreenShot для функции



Прочитайте Обнаружение лица с использованием CoreImage / OpenCV онлайн: https://riptutorial.com/ru/ios/topic/7298/обнаружение-лица-с-использованием-coreimage---opencv

глава 155: Обработка URL-адресов

Синтаксис

1. // Метод **canOpenURL** проверяет, есть ли какое-либо приложение, которое может обрабатывать указанную схему URL.
2. // Swift

   UApplication.sharedApplication (). CanOpenURL (_ aUrl: NSURL)
3. // Объектив-C

   [[UAppication sharedApplication] canOpenURL: (NSURL *) aUrl];
4. // метод **openURL** пытается открыть ресурс, расположенный по URL-адресу. YES / true, если он был открыт иначе NO / false.
5. // Swift

   UIApplication.sharedApplication (). OpenURL (_ aUrl: NSURL)
6. // Объектив-C

   [[UAppication sharedApplication] openURL: (NSURL *) aUrl];

параметры

параметр	Имея в виду
aUrl	экземпляр NSURL, в котором хранится встроенная или настраиваемая строка схемы

замечания

В iOS9 и выше ваше приложение должно указать любые схемы URL, которые он хочет запросить. Это делается путем добавления `LSApplicationQueriesSchemes` к Info.plist

---

iOS имеет встроенную поддержку схем `tel` , `http` / `https` , `sms` , `mailto` , `facetime` . Он также поддерживает HTTP-адреса для приложений `Youtube` , `Maps` и `iTunes` .

Примеры встроенных схем URL:

тел : `tel://123456890` **или** `tel:123456890`

http : `http://www.google.com`

facetime : `facetime://azimov@demo.com`

mailto : `mailto://azimov@demo.com`

смс : `sms://123456890` **или** `sms:123456890`

Youtube : `https://www.youtube.com/watch?v=-eCaif2QKfA`

Карты :

- Использование адреса:
  `http://maps.apple.com/?address=1,Infinite+Loop,Cupertino,California`
- Использование координат: `http://maps.apple.com/?ll=46.683155557,6.683155557`

iTunes : `https://itunes.apple.com/us/artist/randy-newman/id200900`

Примечание . Не все специальные символы поддерживаются в `tel` схеме (например, `*` или `#`). Это делается из-за проблем с безопасностью, чтобы запретить пользователям несанкционированную переадресацию вызовов, поэтому в этом случае приложение « `Phone` не будет открыто.

Examples

Использование встроенной схемы URL для открытия приложения «Почта»

Swift:

```
if let url = URL(string: "mailto://azimov@demo.com") {
    if UIApplication.shared.canOpenURL(url) {
        UIApplication.shared.openURL(url)
    } else {
        print("Cannot open URL")
    }
}
```

Objective-C:

```
NSURL *url = [NSURL URLWithString:@"mailto://azimov@demo.com"];
if ([[UIApplication sharedApplication] canOpenURL:url]) {
    [[UIApplication sharedApplication] openURL:url];
} else {
```

```
    NSLog(@"Cannot open URL");
}
```

Схемы URL-адресов Apple

Это схемы URL, поддерживаемые родными приложениями в iOS, OS X и watchOS 2 и более поздних версиях.

Открытие ссылки в Safari:

Objective-C

```
NSString *stringURL = @"http://stackoverflow.com/";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Swift:

```
let stringURL = "http://stackoverflow.com/"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

Запуск телефонного разговора

Objective-C

```
NSString *stringURL = @"tel:1-408-555-5555";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Swift:

```
let stringURL = "tel:1-408-555-5555"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

HTML

```
<a href="tel:1-408-555-5555">1-408-555-5555</a>
```

Запуск сеанса FaceTime

Objective-C

```
NSString *stringURL = @"facetime:14085551234";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Swift:

```
let stringURL = "facetime:14085551234"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

HTML

```
<a href="facetime:14085551234">Connect using FaceTime</a>
<a href="facetime:user@example.com">Connect using FaceTime</a>
```

Открытие сообщений Приложение для составления смс получателю:

Objective-C

```
NSString *stringURL = @"sms:1-408-555-1212";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Swift:

```
let stringURL = "sms:1-408-555-1212"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

HTML

```
<a href="sms:">Launch Messages App</a>
<a href="sms:1-408-555-1212">New SMS Message</a>
```

Открытие приложения Mail для составления письма получателю:

Objective-C

```
NSString *stringURL = @"mailto:foo@example.com";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Swift:

```
let stringURL = "mailto:foo@example.com"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

HTML

```
<a href="mailto:frank@wwdcdemo.example.com">John Frank</a>
```

Вы также можете указать поле темы, сообщение и несколько получателей в полях To, Cc и Bcc. (В iOS атрибут from игнорируется.) В следующем примере показан URL-адрес mailto, который включает в себя несколько разных атрибутов:

```
mailto:foo@example.com?cc=bar@example.com&subject=Greetings%20from%20Cupertino!&body=Wish%20you%20were
```

Примечание. Диалоговое окно создания электронной почты также может быть представлено в приложении с помощью `MFMailComposeViewController` .

Прочитайте Обработка URL-адресов онлайн: https://riptutorial.com/ru/ios/topic/3646/обработка-url-адресов

глава 156: Основная графика

Examples

Создание основного графического контекста

Контекст основной графики

Контекст Core Graphics - это холст, который мы можем рисовать в нем и задавать некоторые свойства, такие как толщина линии.

Создание контекста

Чтобы создать контекст, мы используем `UIGraphicsBeginImageContextWithOptions()` C. Затем, когда мы закончили рисование, мы просто вызываем `UIGraphicsEndImageContext()` чтобы закончить контекст:

стриж

```
let size = CGSize(width: 256, height: 256)

UIGraphicsBeginImageContextWithOptions(size, false, 0)

let context = UIGraphicsGetCurrentContext()

// drawing code here

UIGraphicsEndImageContext()
```

Objective-C

```
CGSize size = [CGSize width:256 height:256];

UIGraphicsBeginImageContextWithOptions(size, NO, 0);

CGContext *context = UIGraphicsGetCurrentContext();

// drawing code here

UIGraphicsEndImageContext();
```

В приведенном выше коде мы передали 3 параметра функции `UIGraphicsBeginImageContextWithOptions()` :

1. Объект `CGSize` который сохраняет весь размер контекста (холст)
2. Логическое значение, которое, если оно истинно, контекст будет непрозрачным
3. Целочисленное значение, которое устанавливает масштаб (1 для не-сетчатки, 2 для сетчатки и 3 для сетчатых экранов HD). Если установлено значение 0, система автоматически обрабатывает масштаб на основе целевого устройства.

Представление протяжного холста для пользователя

стриж

```
let image = UIGraphicsGetImageFromCurrentImageContext()
imageView.image = image //assuming imageView is a valid UIImageView object
```

Objective-C

```
UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
imageView.image = image; //assuming imageView is a valid UIImageView object
```

Прочитайте Основная графика онлайн: https://riptutorial.com/ru/ios/topic/5530/основная-графика

глава 157: Основные данные

Вступление

Core Data - это модельный уровень вашего приложения в самом широком смысле. Это модель в модели Model-View-Controller, которая пронизывает IOS SDK.

Core Data - это не база данных вашего приложения, а также API для хранения данных в базе данных. Core Data - это структура, управляющая графом объектов. Это так просто. Основные данные могут сохранять этот граф объектов, записывая его на диск, но это не основная цель структуры.

Examples

Операции с основными данными

Получить контекст:

```
NSManagedObjectContext *context = ((AppDelegate*)[[UIApplication sharedApplication]
delegate]).persistentContainer.viewContext;
```

Для получения данных:

```
NSFetchRequest<EntityName *> *fetchRequest = [EntityName fetchRequest];
NSError *error ;
NSArray *resultArray= [context executeFetchRequest:fetchRequest error:&error];
```

Чтобы получить данные с сортировкой:

```
NSFetchRequest<EntityName *> *fetchRequest = [EntityName fetchRequest];
NSSortDescriptor *sortDescriptor = [NSSortDescriptor sortDescriptorWithKey:@"someKey"
ascending:YES];
fetchRequest.sortDescriptors = @[sortDescriptor];
NSError *error ;
NSArray *resultArray= [context executeFetchRequest:fetchRequest error:&error];
```

Чтобы добавить данные:

```
NSManagedObject *entityNameObj = [NSEntityDescription
insertNewObjectForEntityForName:@"EntityName" inManagedObjectContext:context];
[entityNameObj setValue:@"someValue" forKey:@"someKey"];
```

Чтобы сохранить контекст:

```
[((AppDelegate*)[[UIApplication sharedApplication] delegate]) saveContext];
```

Прочитайте Основные данные онлайн: https://riptutorial.com/ru/ios/topic/9489/основные-данные

глава 158: Основные текстовые файлы ввода / вывода

Examples

Чтение и запись из папки «Документы»

Swift 3

```
import UIKit

// Save String to file
let fileName = "TextFile"
let documentDirectory = try FileManager.default.urlForDirectory(.documentDirectory, in:
.userDomainMask, appropriateFor: nil, create: true)

var fileURL = try
documentDirectory.appendingPathComponent(fileName).appendingPathExtension("txt")

print("FilePath: \(fileURL.path)")

var toFileString = "Text to write"
do {
  // Write to file
  try toFileString.writeToURL(fileURL, atomically: true, encoding: NSUTF8StringEncoding)
} catch let error as NSError {
  print("Failed writing to URL: \(fileURL), Error:\(error.localizedDescription)")
}

// Reading
var fromFileString = ""
do {
  fromFileString = try String(contentsOfURL: fileURL)
} catch let error as NSError {
  print("Failed reading from URL: \(fileURL), Error: " + error.localizedDescription)
}
print("Text input from file: \(fromFileString)")
```

Swift 2

```
import UIKit

// Save String to file
let fileName = "TextFile"
let DocumentDirectoryURL = try!
NSFileManager.defaultManager().URLForDirectory(.DocumentDirectory, inDomain: .UserDomainMask,
appropriateForURL: nil, create: true)

let fileURL =
DocumentDirectoryURL.URLByAppendingPathComponent(fileName).URLByAppendingPathExtension("txt")
print("FilePath: \(fileURL.path)")

var toFileString = "Text to write"
```

```
do {
  // Write to file
  try toFileString.writeToURL(fileURL, atomically: true, encoding: NSUTF8StringEncoding)
} catch let error as NSError {
  print("Failed writing to URL: \(fileURL), Error:\(error.localizedDescription)")
}

// Reading
var fromFileString = ""
do {
  fromFileString = try String(contentsOfURL: fileURL)
} catch let error as NSError {
  print("Failed reading from URL: \(fileURL), Error: " + error.localizedDescription)
}
print("Text input from file: \(fromFileString)")
```

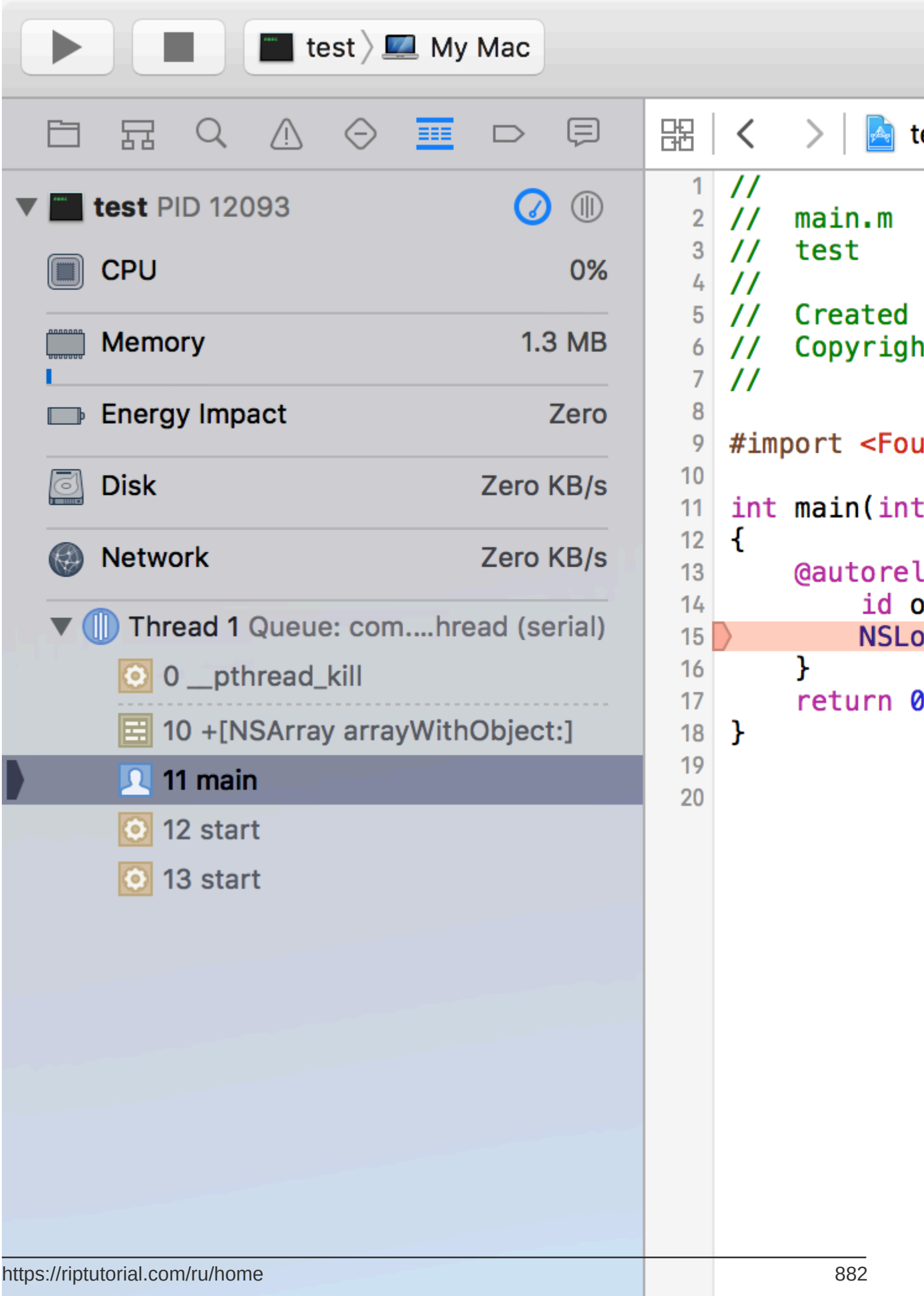
Прочитайте Основные текстовые файлы ввода / вывода онлайн:
https://riptutorial.com/ru/ios/topic/8892/основные-текстовые-файлы-ввода---вывода

глава 159: Отладка сбоев

Examples

Поиск информации о сбое

Когда ваше приложение выйдет из строя, Xcode войдет в отладчик и покажет вам больше информации о сбое:


test
My Mac
test PID 12093
CPU
0%
Memory
1.3 MB
Energy Impact
Zero
Disk
Zero KB/s
Network
Zero KB/s
Thread 1 Queue: com....hread (serial)
0 __pthread_kill
10 +[NSArray arrayWithObject:]
11 main
12 start
13 start
//
// main.m
// test
//
// Created
// Copyrigh
//
#import <Fou
int main(int
{
@autorel
id o
NSLo
}
return 0
}

вы получите текстовое представление трассировки стека, которое вы можете скопировать и вставить:

```
(lldb) bt
* thread #1: tid = 0x3aaec5, 0x00007fff91055f06 libsystem_kernel.dylib`__pthread_kill + 10,
queue = 'com.apple.main-thread', stop reason = signal SIGABRT
    frame #0: 0x00007fff91055f06 libsystem_kernel.dylib`__pthread_kill + 10
    frame #1: 0x000000010008142d libsystem_pthread.dylib`pthread_kill + 90
    frame #2: 0x00007fff96dc76e7 libsystem_c.dylib`abort + 129
    frame #3: 0x00007fff8973bf81 libc++abi.dylib`abort_message + 257
    frame #4: 0x00007fff89761a47 libc++abi.dylib`default_terminate_handler() + 267
    frame #5: 0x00007fff94f636ae libobjc.A.dylib`_objc_terminate() + 103
    frame #6: 0x00007fff8975f19e libc++abi.dylib`std::__terminate(void (*)()) + 8
    frame #7: 0x00007fff8975ec12 libc++abi.dylib`__cxa_throw + 121
    frame #8: 0x00007fff94f6108c libobjc.A.dylib`objc_exception_throw + 318
    frame #9: 0x00007fff8d067372 CoreFoundation`-[__NSPlaceholderArray initWithObjects:count:]
+ 290
    frame #10: 0x00007fff8d0eaa1f CoreFoundation`+[NSArray arrayWithObject:] + 47
  * frame #11: 0x0000000100001b54 test`main(argc=1, argv=0x00007fff5fbff808) + 68 at main.m:15
    frame #12: 0x00007fff8bea05ad libdyld.dylib`start + 1
    frame #13: 0x00007fff8bea05ad libdyld.dylib`start + 1
```

Отладка сбоев SIGABRT и EXC_BAD_INSTRUCTION

`SIGABRT` или EXC_BAD_INSTRUCTION обычно означает, что приложение разбилось само намеренно, потому что некоторая проверка завершилась неудачно. Они должны записать сообщение на консоль отладчика с дополнительной информацией; проверьте там дополнительную информацию.

Многие `SIGABRT` вызваны неотображаемыми исключениями Objective-C. Существует *множество* причин, по которым могут быть исключены исключения, и они *всегда* будут записывать на консоль много полезной информации.

- `NSInvalidArgumentException` , что означает, что приложение передало недопустимый аргумент методу
- `NSRangeException` , что означает, что приложение попыталось получить доступ к индексу вне пределов объекта, например `NSArray` или `NSString`
- `NSInternalInconsistencyException` означает, что объект обнаружен в неожиданном состоянии.
- `NSUnknownKeyException` обычно означает, что у вас плохое соединение в XIB. Попробуйте некоторые ответы на этот вопрос .

Отладка EXC_BAD_ACCESS

`EXC_BAD_ACCESS` означает, что процесс пытался получить доступ к памяти недействительным способом, например разыменование указателя `NULL` или запись в постоянную память. Это самый трудный вид сбоя для отладки, потому что он обычно не имеет сообщения об ошибке, и некоторые сбои могут быть *очень* трудными для воспроизведения и / или встречаются в коде, полностью не связанном с проблемой. Эта ошибка встречается очень

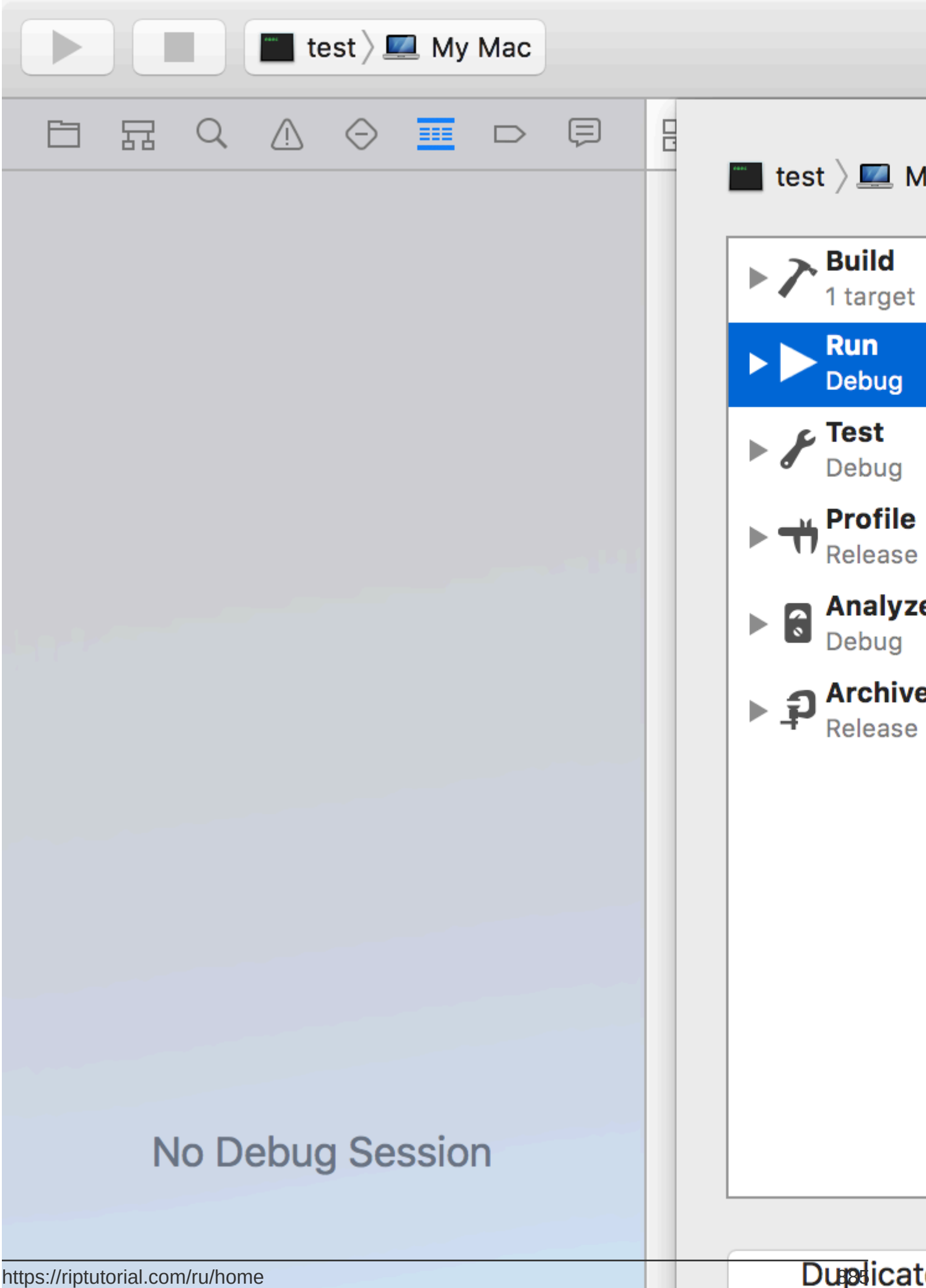
редко в Swift, но если это происходит, вы можете часто получать более легкие отладки, уменьшая оптимизацию компилятора.

Большинство ошибок `EXC_BAD_ACCESS` вызваны попыткой разыменовать указатель `NULL` . Если это так, адрес, указанный в красной стрелке, обычно будет шестнадцатеричным числом, которое ниже обычного адреса памяти, часто `0x0` . Установите контрольные точки в отладчике или добавьте случайные инструкции `printf` / `NSLog` чтобы узнать, почему этот указатель имеет `NULL` .

`EXC_BAD_ACCESS` который происходит менее надежно или вообще не имеет смысла, может быть результатом проблемы управления памятью. Общие проблемы, которые могут вызвать это:

- Использование памяти, которая была освобождена
- Попытка написать за конец массива C или другого типа буфера
- Использование указателя, который не был инициализирован

В разделе «Диагностика» редактора схем Xcode содержит несколько полезных инструментов для устранения проблем с памятью:


test
My Mac
test
Build
1 target
Run
Debug
Test
Debug
Profile
Release
Analyze
Debug
Archive
Release
No Debug Session

добавляет множество проверок, которые остановят приложение, когда возникнут проблемы с памятью, и предоставит полезное сообщение об ошибке, подробно описывающее, что произошло. Объекты Zombie обнаруживают проблемы с освобожденными объектами Objective-C, но вы не должны получать эти проблемы с включенным автоматическим подсчетом ссылок.

Прочитайте Отладка сбоев онлайн: https://riptutorial.com/ru/ios/topic/4745/отладка-сбоев

глава 160: Оценка заявки / запроса заявки

Вступление

Теперь, начиная с iOS 10.3, нет необходимости перемещать приложение в магазин apple для оценки / обзора. Apple внедрила класс SKStoreReviewController в инфраструктуру storekit. В котором разработчику просто нужно вызвать метод класса queryReview () класса SKStoreReviewController, и система обрабатывает весь процесс для вас.

Кроме того, вы можете продолжать включать постоянную ссылку в настройках или экранах конфигурации вашего приложения, которые глубоко связаны с вашей страницей продукта в App Store. Для автоматического отключения

Examples

Оценить / просмотреть приложение iOS

Просто введите ниже одного кода строки, где вы хотите, чтобы пользователь оценивал / просматривал ваше приложение.

SKStoreReviewController.requestReview ()

Прочитайте Оценка заявки / запроса заявки онлайн: https://riptutorial.com/ru/ios/topic/9678/оценка-заявки---запроса-заявки

глава 161: Панель навигации

Examples

Настроить внешний вид панели навигации по умолчанию.

```
// Default UINavigationBar appearance throughout the app
[[UINavigationBar appearance] setTitleTextAttributes:@{NSForegroundColorAttributeName:
[UIColor whiteColor],
                                                       NSFontAttributeName : [UIFont
fontWithName:@"HelveticaNeue-CondensedBold" size:17],
                                                       }];

[[UINavigationBar appearance] setTintColor:[UIColor whiteColor]];
[[UINavigationBar appearance] setBarTintColor:[UIColor KNGRed]];
[[UINavigationBar appearance] setTranslucent:NO];
[[UINavigationBar appearance] setBarStyle:UIBarStyleBlack];
[[UIBarButtonItem appearanceWhenContainedIn: [UISearchBar class], nil] setTintColor:[UIColor
KNGGray]];
```

Пример SWIFT

```
navigationController?.navigationBar.titleTextAttributes = [NSForegroundColorAttributeName:
UIColor.white, NSFontAttributeName:UIFont(name: "HelveticaNeue-CondensedBold", size: 17)!,]
navigationController?.navigationBar.tintColor = .white
navigationController?.navigationBar.barTintColor = .red
navigationController?.navigationBar.isTranslucent = false
navigationController?.navigationBar.barStyle = .black
```

Прочитайте Панель навигации онлайн: https://riptutorial.com/ru/ios/topic/7066/панель-навигации

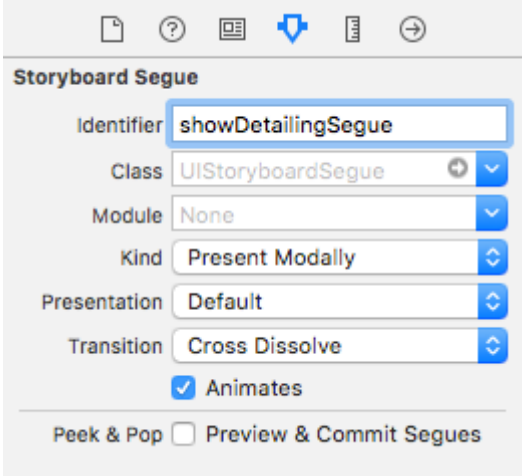
глава 162: Передача данных между контроллерами просмотра

Examples

Использование Segues (передача данных вперед)

Чтобы передать данные с текущего контроллера представлений на следующий новый контроллер представления (а не на предыдущий контроллер представления) с использованием segues, сначала создайте segue с идентификатором в соответствующей раскадровке. Переопределите метод `prepareForSegue` текущего контроллера. Внутри метода проверьте, что вы только что создали его идентификатор. Передайте контроллер представления назначения и передайте ему данные, установив свойства на панели управления снизу.

Установка идентификатора для сегмента:



Сегменты могут выполняться программно или с помощью события действия кнопки, установленного в раскадровке, с помощью Ctrl + перетаскивание до контроллера представления цели. Вы можете при необходимости запросить segue, если необходимо, используя идентификатор segue в контроллере представления:

Objective-C

```
- (void)showDetail {
    [self performSegueWithIdentifier:@"showDetailingSegue" sender:self];
}
```

стриж

```
func showDetail() {
```

```
    self.performSegue(withIdentifier: "showDetailingSegue", sender: self)
}
```

Вы можете настроить полезную нагрузку `prepareForSegue` в переопределенной версии метода `prepareForSegue` . Вы можете установить требуемые свойства перед загрузкой контроллера точки назначения.

Objective-C

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if([segue.identifier isEqualToString:@"showDetailingSegue"]){
        DetailViewController *controller = (DetailViewController
*)segue.destinationViewController;
        controller.isDetailingEnabled = YES;
    }
}
```

стриж

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "showDetailingSegue" {
        let controller = segue.destinationViewController as! DetailViewController
        controller.isDetailingEnabled = true
    }
}
```

`DetailViewController` - это имя второго контроллера представления, а `isDetailingEnabled` - общедоступная переменная в этом контроллере представления.

Чтобы расширить этот шаблон, вы можете обработать открытый метод `DetailViewController` в качестве псевдоинициализатора, чтобы помочь инициализировать любые требуемые переменные. Это будет переменные документа, которые должны быть установлены на `DetailViewController` без необходимости читать через его исходный код. Это также удобное место для установки значений по умолчанию.

Objective-C

```
- (void)initVC:(BOOL *)isDetailingEnabled {
    self.isDetailingEnabled = isDetailingEnabled
}
```

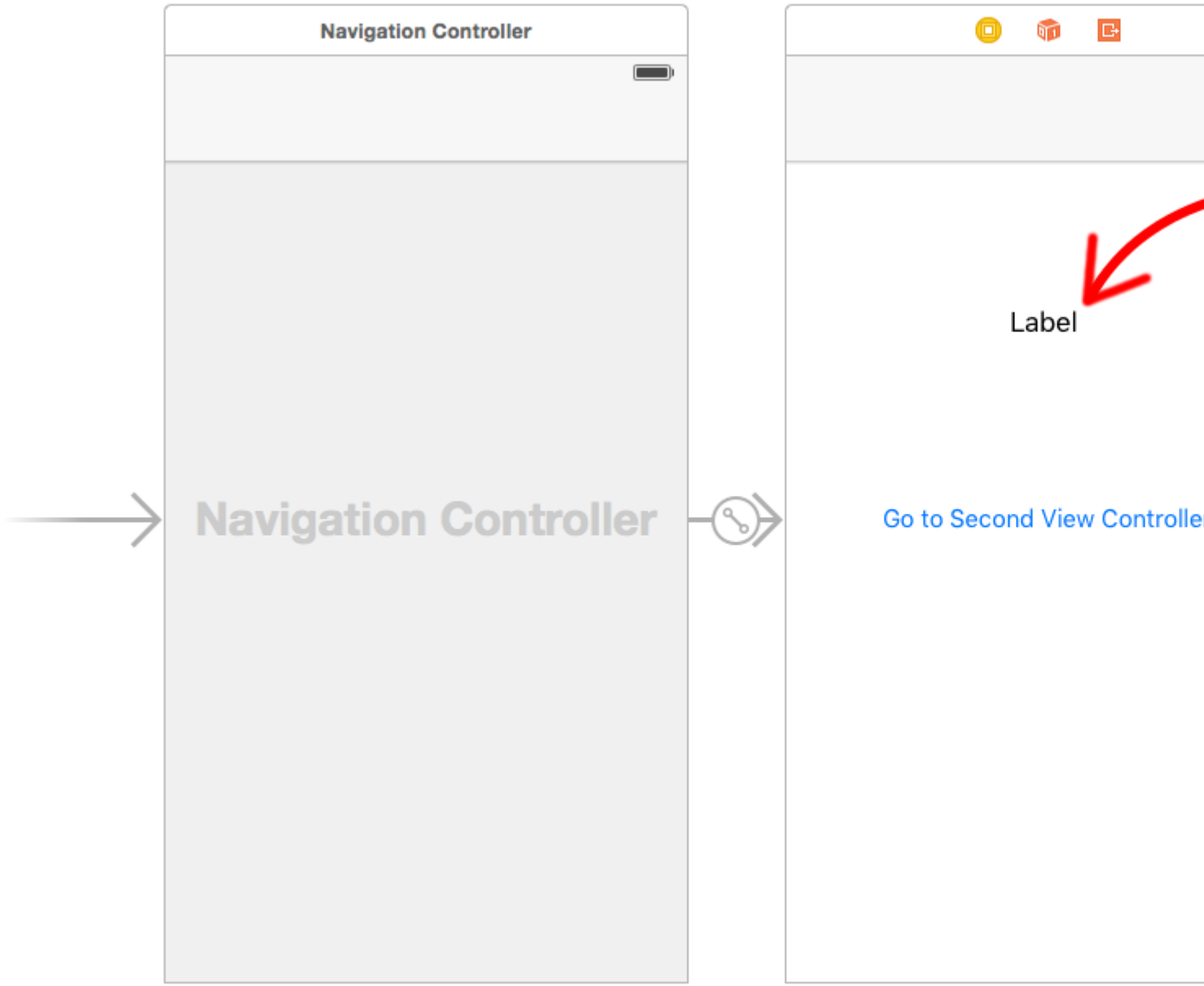
стриж

```
func initVC(isDetailingEnabled: Bool) {
    self.isDetailingEnabled = isDetailingEnabled
}
```

Использование шаблона делегата (передача данных назад)

Чтобы передать данные с текущего контроллера представления обратно на предыдущий

контроллер просмотра, вы можете использовать шаблон делегата.



В этом примере предполагается, что вы сделали segue в Interface Builder и установили идентификатор `showSecondViewController` для `showSecondViewController` . Выходы и действия также должны быть подключены к именам в следующем коде.

Контроллер первого взгляда

Код для первого контроллера просмотра

стриж

```
class FirstViewController: UIViewController, DataEnteredDelegate {

    @IBOutlet weak var label: UILabel!

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
```

```
        if segue.identifier == "showSecondViewController", let secondViewController =
segue.destinationViewController as? SecondViewController  {
            secondViewController.delegate = self
        }
    }

    // required method of our custom DataEnteredDelegate protocol
    func userDidEnterInformation(info: String) {
        label.text = info
        navigationController?.popViewControllerAnimated(true)
    }
}
```

Objective-C

```
@interface FirstViewController : UIViewController <DataEnteredDelegate>
@property (weak, nonatomic) IBOutlet UILabel *label;
@end

@implementation FirstViewController
- (void)viewDidLoad {
    [super viewDidLoad];
}
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    SecondViewController *secondViewController = segue.destinationViewController;
    secondViewController.delegate = self;

}
-(void)userDidEnterInformation:(NSString *)info {
    _label.text = info
    [self.navigationController popViewControllerAnimated:YES];
}
@end
```

Обратите внимание на использование нашего пользовательского протокола `DataEnteredDelegate` .

Второй контроллер и протокол просмотра

Код для второго контроллера просмотра

стриж

```
// protocol used for sending data back
protocol DataEnteredDelegate: class {
    func userDidEnterInformation(info: String)
}

class SecondViewController: UIViewController {

    // making this a weak variable so that it won't create a strong reference cycle
    weak var delegate: DataEnteredDelegate?

    @IBOutlet weak var textField: UITextField!
```

```
    @IBAction func sendTextBackButton(sender: AnyObject) {

        // call this method on whichever class implements our delegate protocol (the first
view controller)
        delegate?.userDidEnterInformation(textField.text ?? "")
    }
}
```

Objective-C

```
@protocol DataEnteredDelegate <NSObject>
-(void)userDidEnterInformation:(NSString *)info;
@end

@interface SecondViewController : UIViewController
@property (nonatomic) id <DataEnteredDelegate> delegate;
@property (weak, nonatomic) IBOutlet UITextField *textField;
@end

@implementation SecondViewController
- (void)viewDidLoad {
    [super viewDidLoad];
}

- (IBAction) sendTextBackButton:(id)sender{
    [_delegate userDidEnterInformation:textField.text];
}
@end
```

Обратите внимание, что `protocol` находится вне класса View Controller.

Передача данных назад, используя разматывание

В отличие от segue, который позволяет передавать данные «вперед» с текущего контроллера точки зрения на контроллер точки назначения:

(VC1) **->** *(VC2)*

Используя «разматывать», вы можете сделать обратное, передать данные с адресата или текущего контроллера представления на свой контроллер представления:

(VC1) **<-** *(VC2)*

ПРИМЕЧАНИЕ . Обратите внимание, что с помощью функции размотки вы можете сначала передать данные, а затем вывести на экран текущий контроллер (VC2).

Вот как это сделать:

Во-первых, вам нужно будет добавить следующее объявление в представлении контроллера представления (VC1), который является контроллером представления, который мы хотим передать данным:

```
@IBAction func unwindToPresentingViewController(segue:UIStoryboardSegue)
```

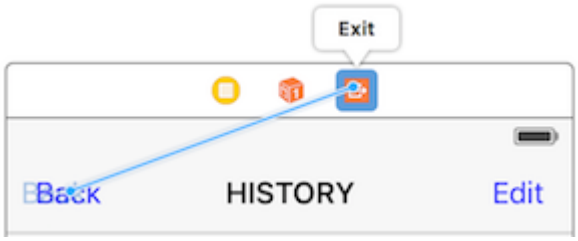
Важно использовать префикс `unwind` , это «сообщает» Xcode, что это метод разматывания, дающий вам возможность использовать его и в раскадровке.

После этого вам нужно будет реализовать метод, он выглядит почти так же, как фактический segue:

```
@IBAction func unwindToPresentingViewController(segue:UIStoryboardSegue)
{
    if segue.identifier == "YourCustomIdentifer"
    {
        if let VC2 = segue.sourceViewController as? VC2
        {
            //    Your custom code in here to access VC2 class member
        }
```

Теперь у вас есть 2 варианта вызова вызовов для разговора:

1. Вы можете «жестко закодировать» вызывать: `self.performSegueWithIdentifier("YourCustomIdentifier", sender: self)` который будет раскручивать вас, когда вы будете `performSegueWithIdentifier` .
2. Вы можете связать метод разматывания с помощью `storyboard` с объектом «Выход»: ctrl + перетащить кнопку, которую вы хотите вызвать метод разматывания, к объекту «Выход»:



Отпустите, и у вас будет возможность выбрать свой способ раскрутки:

Передача данных с помощью закрытий (передача данных назад)

Вместо использования **шаблона делегирования** , который разделяет реализацию в разных `UIViewController` класса `UIViewController` , вы даже можете использовать `closures` для передачи данных назад и вперед. Предполагая, что вы используете `UIStoryboardSegue` , в

методе `prepareForSegue` вы можете легко настроить новый контроллер за один шаг

```
final class DestinationViewController: UIViewController {
    var onCompletion: ((success: Bool) -> ())?

    @IBAction func someButtonTapped(sender: AnyObject?) {
        onCompletion?(success: true)
    }
}

final class MyViewController: UIViewController {
    override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {

        guard let destinationController = segue.destinationViewController as?
DestinationViewController else { return }

        destinationController.onCompletion = { success in
            // this will be executed when `someButtonTapped(_:)` will be called
            print(success)
        }
    }
}
```

Это пример использования, и лучше использовать в синтаксисе блока Swift, Objective-C не так просто сделать код более удобочитаемым

Использование закрытия обратного вызова (блока), передающего данные назад

эта тема является классической проблемой в iOS-разработке, и ее решение различно, как показано в другом примере. В этом примере я покажу еще одно ежедневное общее использование: передача данных с использованием `closure` путем адаптации примера `delegate pattern` на этой странице в `closure` обратного вызова!

одна вещь, что этот метод превосходит `delegate pattern` вместо того, чтобы разбить код настройки в двух разных местах (посмотрите пример делегата на этой странице, `prepareForSegue` , `userDidEnterInformation`), а `userDidEnterInformation` их вместе (только в `prepareForSegue` , я покажу его)

Начать с контроллера второго просмотра

мы должны выяснить, как использовать обратный вызов, тогда мы можем написать его, поэтому мы начинаем с второго контроллера представления, так как там мы используем обратный вызов: когда мы получили новый ввод текста, мы вызываем наш обратный вызов, **используя параметр обратного вызова** в качестве среды для передачи данных обратно в первый ViewController, обратите внимание, что я сказал, используя параметр обратного вызова, это очень важно, новички (как и я) всегда упускают из виду это и не знают, где начать правильно писать обратный вызов

поэтому в этом случае мы знаем, что наш обратный вызов принимает только один

параметр: текст и его тип - `String` , давайте объявим его и сделаем это свойство, так как нам нужно заполнить наш первый контроллер представления

Я просто комментирую всю часть `delegate` и держу его для сравнения

```
class SecondViewController: UIViewController {

    //weak var delegate: DataEnteredDelegate? = nil
    var callback: ((String?)->())?

    @IBOutlet weak var textField: UITextField!

    @IBAction func sendTextBackButton(sender: AnyObject) {

        //delegate?.userDidEnterInformation(textField.text!)
        callback?(input.text)

        self.navigationController?.popViewControllerAnimated(true)
    }
}
```

Завершить контроллер первого вида

все, что вам нужно сделать, это передать закрытие обратного вызова, и мы закончили, закрытие сделает для нас будущую работу, поскольку мы уже установили его во втором контроллере представления

посмотрите, как сделать наш код короче по сравнению с `delegate pattern`

```
//no more DataEnteredDelegate
class FirstViewController: UIViewController {

    @IBOutlet weak var label: UILabel!

    override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
        if segue.identifier == "showSecondViewController" {
            let secondViewController = segue.destinationViewController as!
SecondViewController
            //secondViewController.delegate = self
            secondViewController.callback = { text in self.label.text = text }
        }
    }

    // required method of our custom DataEnteredDelegate protocol
    //func userDidEnterInformation(info: String) {
    //    label.text = info
    //}
}
```

и в последнем случае кто-то из вас будет смущен тем, что мы просто передаем данные (закрытие в этом случае) только одним способом: от первого контроллера представления до второго, без прямого возврата со второго контроллера представления, как мы можем считать его инструментом коммуникации? возможно, вам действительно нужно запустить его и доказать это самостоятельно, все, что я скажу, это **параметр** , это **параметр**

закрытия обратного вызова, который передает данные обратно!

Назначая свойство (передача данных вперед)

Вы можете передавать данные напрямую, назначая свойство следующего контроллера представлений перед тем, как нажать или представить его.

```
class FirstViewController: UIViewController {

    func openSecondViewController() {

        // Here we initialize SecondViewController and set the id property to 492
        let secondViewController = SecondViewController()
        secondViewController.id = 492

        // Once it was assign we now push or present the view controller
        present(secondViewController, animated: true, completion: nil)
    }

}

class SecondViewController: UIViewController {

    var id: Int?

    override func viewDidLoad() {
        super.viewDidLoad()

        // Here we unwrapped the id and will get the data from the previous view controller.
        if let id = id {
            print("Id was set: \(id)")
        }
    }
}
```

Прочитайте Передача данных между контроллерами просмотра онлайн:
https://riptutorial.com/ru/ios/topic/434/передача-данных-между-контроллерами-просмотра

глава 163: Передача данных между контроллерами просмотра (с помощью MessageBox-Concept)

Вступление

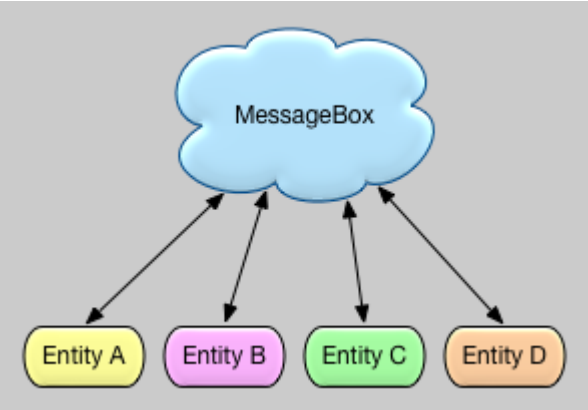
MessageBox - это простая концепция для развязывания сущностей.

Например, объект A может поместить сообщение, которое может распознать объект B всякий раз, когда он подходит.

Контроллер вида хотел бы поговорить с другим контроллером представления, но вы не хотите создавать сильные или слабые отношения.

Examples

Простой пример использования




```
let messageBox:MessageBox = MessageBox()

// set
messageBox.setObject("TestObject1", forKey:"TestKey1")

// get
// but don't remove it, keep it stored, so that it can still be retrieved later
let someObject:String = messageBox.getObject(forKey:"TestKey1", removeIfFound:false)

// get
// and remove it
let someObject:String = messageBox.getObject(forKey:"TestKey1", removeIfFound:true)
```

Прочитайте Передача данных между контроллерами просмотра (с помощью MessageBox-Concept) онлайн: https://riptutorial.com/ru/ios/topic/9118/передача-данных-между-

контроллерами-просмотра--с-помощью-messagebox-concept-

глава 164: перетекает

Examples

Обзор

Из документации Apple:

> Объект UIStoryboardSegue отвечает за **выполнение визуального перехода между двумя контроллерами представлений** . Кроме того, объекты segue используются для подготовки к переходу от одного контроллера представления к другому. **Объекты Segue содержат информацию о контроллерах представлений, участвующих в переходе** . Когда срабатывает segue, но до появления визуального перехода среда исполнения раскадровки вызывает метод prepareForSegue: sender: текущий метод управления представлением, чтобы он мог передавать любые необходимые данные контроллеру представления, который должен отображаться.

Атрибуты

стриж

```
sourceViewController: UIViewController {get}
destinationViewController: UIViewController {get}
identifier: String? {get}
```

Рекомендации:

- Справочник по классу UIViewController
- Справочник по классу UIStoryBoardSegue

Подготовка контроллера вида до запуска Segue

Подготовка к обучению :

```
func prepareForSegue(_ segue:UIStoryboardSegue, sender sender:AnyObject?)
```

Уведомляет диспетчер представлений о необходимости выполнить сеанс segue

параметры

segue : объект segue.

отправитель : объект, который инициализировал segue.

Пример в Swift

Выполните задачу, если идентификатором сегмента является «SomeSpecificIdentifier»

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
    if segue.identifier == "SomeSpecificIdentifier" {
        //- Do specific task
    }
}
```

Решите, нужно ли выполнять вызов Segue.

ShouldPerformSegueWithIdentifier :

```
func shouldPerformSegueWithIdentifier(_ identifier:String, sender sender:AnyObject?) -> Bool
```

Определяет, должен ли выполняться сеанс с указанным идентификатором.

параметры

Идентификатор : строка, которая идентифицирует инициированный сеанс

Отправитель : объект, который инициализировал segue.

Пример в Swift

Выполняйте только segue, если идентификатор «SomeSpecificIdentifier»

```
override func shouldPerformSegueWithIdentifier(identifier:String, sender:AnyObject?) -> Bool {
    if identifier == "SomeSpecificIdentifier" {
        return true
    }
    return false
}
```

Использование Segues для перемещения назад в стеке навигации

Разморозить Segues

Unwind Segues дает вам возможность «размотать» стек навигации и указать

место назначения, куда нужно вернуться. Подпись этой функции является ключом к распознанию интерфейса Interface Builder. **Он должен иметь возвращаемое значение IBAction и принимать один параметр UIStoryboardSegue** . Имя функции не имеет значения. Фактически, функция даже не должна ничего делать. Это только как маркер, которым UIViewController является пунктом назначения Unwind Segue. [Источник] [1]

Требуемая подпись разматывания

Цель C:

```
-(IBAction)prepareForUnwind:(UIStoryboardSegue *)segue {
}
```

Swift:

```
@IBAction func prepareForUnwind(segue: UIStoryboardSegue) {
}
```

Программный запуск Trigger Segue

PerformSegueWithIdentifier:

```
 func performSegueWithIdentifier(_ identifier:String, sender sender:AnyObject?)
```

Инициирует segue с указанным идентификатором из файла раскадровки текущего контроллера

параметры

Идентификатор : строка, которая идентифицирует инициированный сеанс

Отправитель : объект, который будет инициировать сеанс.

Пример в Swift

Выполнение segue с идентификатором «SomeSpecificIdentifier» из выбора строки таблицы:

```
func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
    performSegueWithIdentifier("SomeSpecificIdentifier", sender: indexPath.item)
}
```

Прочитайте перетекает онлайн: https://riptutorial.com/ru/ios/topic/5575/перетекает

глава 165: Подписание кода

Examples

Профилирование профилей

Чтобы создать файл IPA в XCode, вам необходимо подписать приложение с сертификатом и профилем подготовки. Их можно создать на странице https://developer.apple.com/account/ios/profile/create.

Типы профилей предоставления

Профилирование профилей разделено на два типа: «Разработка и распределение»:

развитие

- Разработка приложений iOS / разработка приложений tvOS - используется в разработке для установки вашего приложения на тестовое устройство.

распределение

- App Store / tvOS App Store - используется для подписи приложения для загрузки приложения.
- В доме - используется для распространения вашего приложения на предприятиях, в устройствах вашего бизнеса.
- Ad Hoc / tvOS Ad Hoc - используется для распространения вашего приложения на ограниченном количестве определенных устройств (например, вам нужно знать UDID устройств, к которым вы хотите установить приложение).

Прочитайте Подписание кода онлайн: https://riptutorial.com/ru/ios/topic/6055/подписание-кода

глава 166: Покупки в приложении

Examples

Единый IAP в Swift 2

После создания IAP в iTunesConnect:

В контроллере представления, который вы хотите купить в

```
import StoreKit
```

и добавить соответствующих делегатов

```
class ViewController: UIViewController, SKProductsRequestDelegate,
SKPaymentTransactionObserver {
```

объявить переменную с идентификатором продукта из iTunesConnect

```
var product_id: NSString?

override func viewDidLoad() {


    product_id = "YOUR_PRODUCT_ID"
    super.viewDidLoad()
    SKPaymentQueue.defaultQueue().addTransactionObserver(self)

   //Check if product is purchased
   if (NSUserDefaults.standardUserDefaults().boolForKey("purchased")){

       // Hide ads
       adView.hidden = true

   } else {
       print("Should show ads...")

   }

}
```

подключите кнопку к функции для покупки IAP

```
@IBAction func unlockAction(sender: AnyObject) {

   print("About to fetch the product...")

// Can make payments
if (SKPaymentQueue.canMakePayments())
    {
        let productID:NSSet = NSSet(object: self.product_id!);
```

```
        let productsRequest:SKProductsRequest = SKProductsRequest(productIdentifiers:
productID as! Set<NSString>);
        productsRequest.delegate = self;
        productsRequest.start();
        println("Fetching Products");
    }else{
        print("Can't make purchases");
    }

}
```

И вот некоторые вспомогательные методы

```
func buyProduct(product: SKProduct){
    println("Sending the Payment Request to Apple");
    let payment = SKPayment(product: product)
    SKPaymentQueue.defaultQueue().addPayment(payment);

 }
```

методы делегата, которые должны быть объявлены

```
func productsRequest (request: SKProductsRequest, didReceiveResponse response:
SKProductsResponse) {

    let count : Int = response.products.count
    if (count>0) {
        var validProduct: SKProduct = response.products[0] as SKProduct
        if (validProduct.productIdentifier == self.product_id) {
            print(validProduct.localizedTitle)
            print(validProduct.localizedDescription)
            print(validProduct.price)
            buyProduct(validProduct);
        } else {
            print(validProduct.productIdentifier)
        }
    } else {
        print("nothing")
    }
}


func request(request: SKRequest!, didFailWithError error: NSError!) {
    print("Error Fetching product information");
}

    func paymentQueue(_ queue: SKPaymentQueue,
updatedTransactions transactions: [SKPaymentTransaction])

{
    print("Received Payment Transaction Response from Apple");

    for transaction:AnyObject in transactions {
        if let trans:SKPaymentTransaction = transaction as? SKPaymentTransaction{
            switch trans.transactionState {
            case .Purchased:
                print("Product Purchased");
                SKPaymentQueue.defaultQueue().finishTransaction(transaction as!
```

```
SKPaymentTransaction)
                // Handle the purchase
                NSUserDefaults.standardUserDefaults().setBool(true , forKey: "purchased")
                adView.hidden = true
                break;
            case .Failed:
                print("Purchased Failed");
                SKPaymentQueue.defaultQueue().finishTransaction(transaction as!
SKPaymentTransaction)
                break;



            case .Restored:
                print("Already Purchased");
               SKPaymentQueue.defaultQueue().restoreCompletedTransactions()


                 // Handle the purchase
                    NSUserDefaults.standardUserDefaults().setBool(true , forKey: "purchased")
                    adView.hidden = true
                    break;
            default:
                break;
            }
        }
    }

}
```

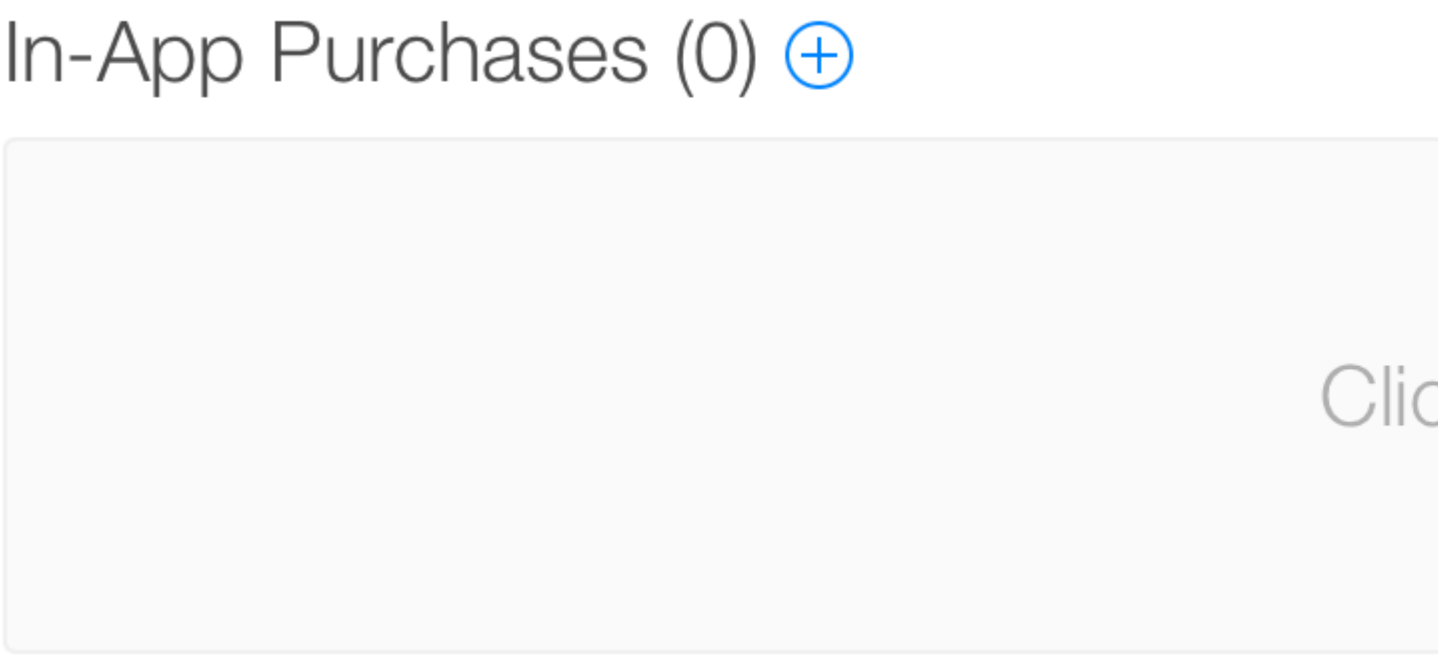
И тогда код для восстановления нерасходуемого в приложении покупки

```
if (SKPaymentQueue.canMakePayments()) {
  SKPaymentQueue.defaultQueue().restoreCompletedTransactions()
}
```

Настройка в iTunesConnect

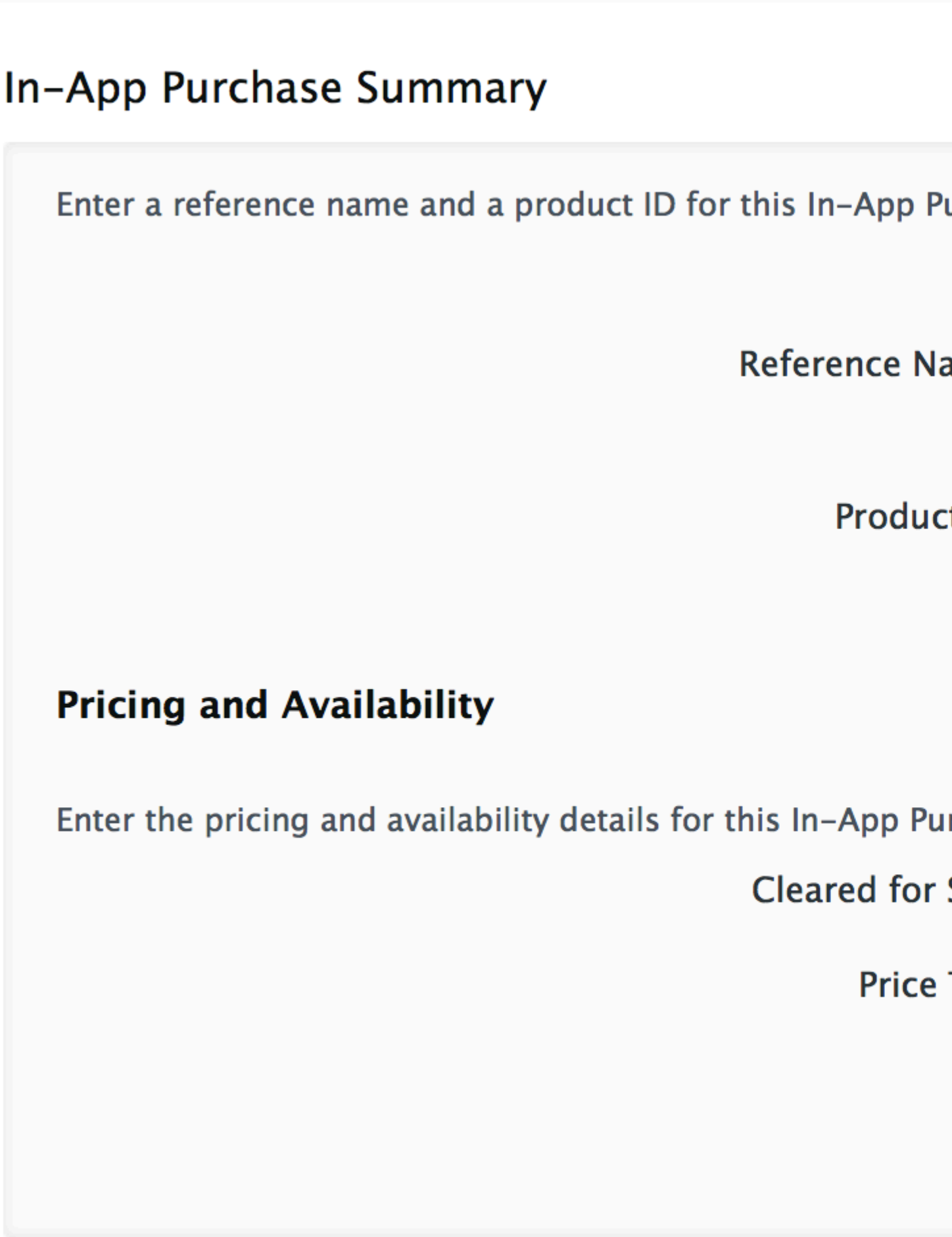
В iTunesConnect выберите приложение, к которому вы хотите добавить IAP.

Нажмите на функции, и вы увидите следующее:



Нажмите плюс. Затем вам нужно будет выбрать тип IAP, который вы хотите создать.

Затем вам нужно будет заполнить всю информацию для вашего IAP.


In-App Purchase Summary
Enter a reference name and a product ID for this In-App Pu
Reference Na
Product
Pricing and Availability
Enter the pricing and availability details for this In-App Pur
Cleared for
Price

IAP .

Большинство основных шагов для покупки / подписки пользователя на IAP

Предполагая, что вы знаете `productID` :

Первый

```
import StoreKit
```

Затем в вашем коде

```
let productID: Set = ["premium"]
let request = SKProductsRequest(productIdentifiers: productID)
request.delegate = self
request.start()
```

и в `SKProductsRequestDelegate` :

```
func productsRequest(request: SKProductsRequest, didReceiveResponse response:
SKProductsResponse) {
    if response.products.count > 0 {
        let product = response.products[0]
        let payment = SKPayment(product: product)
        SKPaymentQueue.defaultQueue().addPayment(payment)
    }
}
```

Прочитайте Покупки в приложении онлайн: https://riptutorial.com/ru/ios/topic/2549/покупки-в-приложении

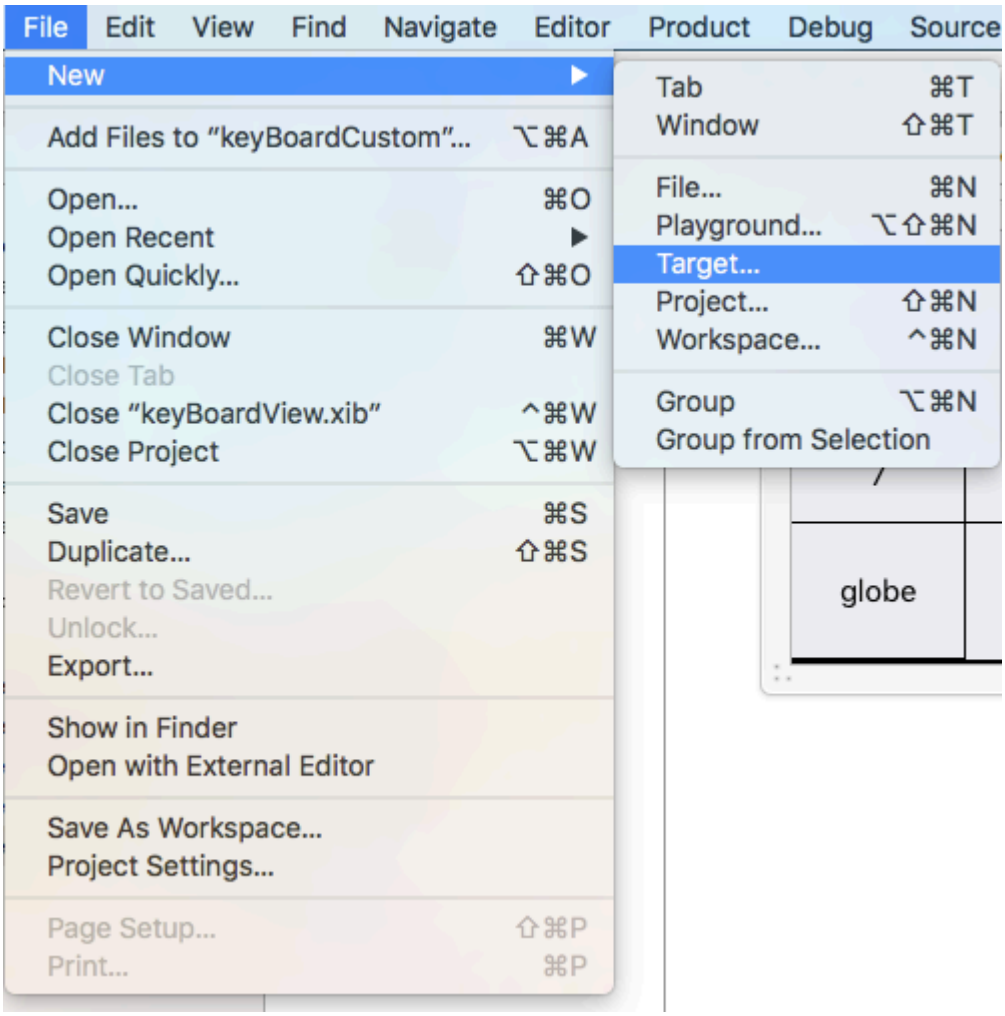
глава 167: Пользовательская клавиатура

Examples

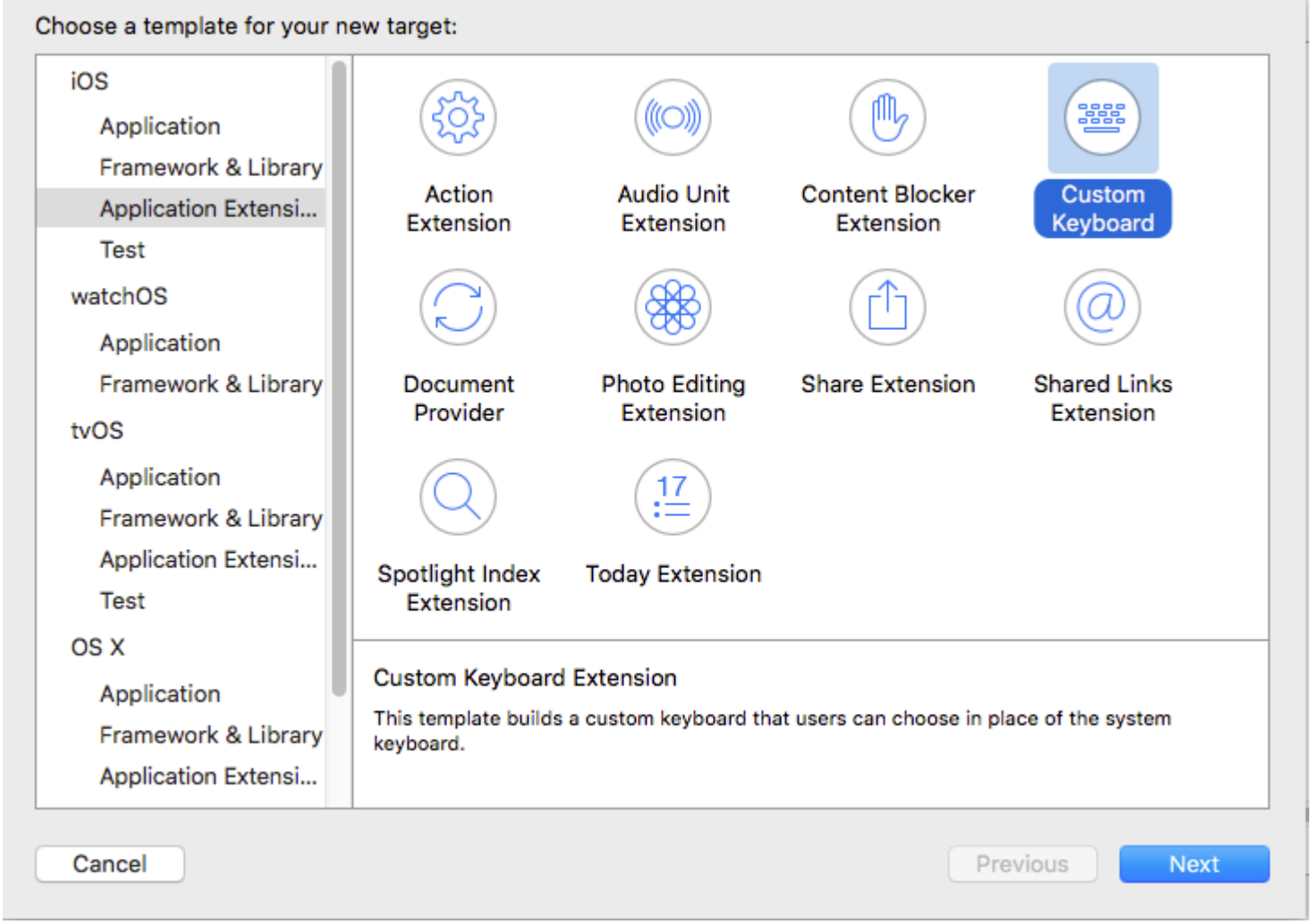
Пример пользовательского KeyBoard

Объектив-С и Xib

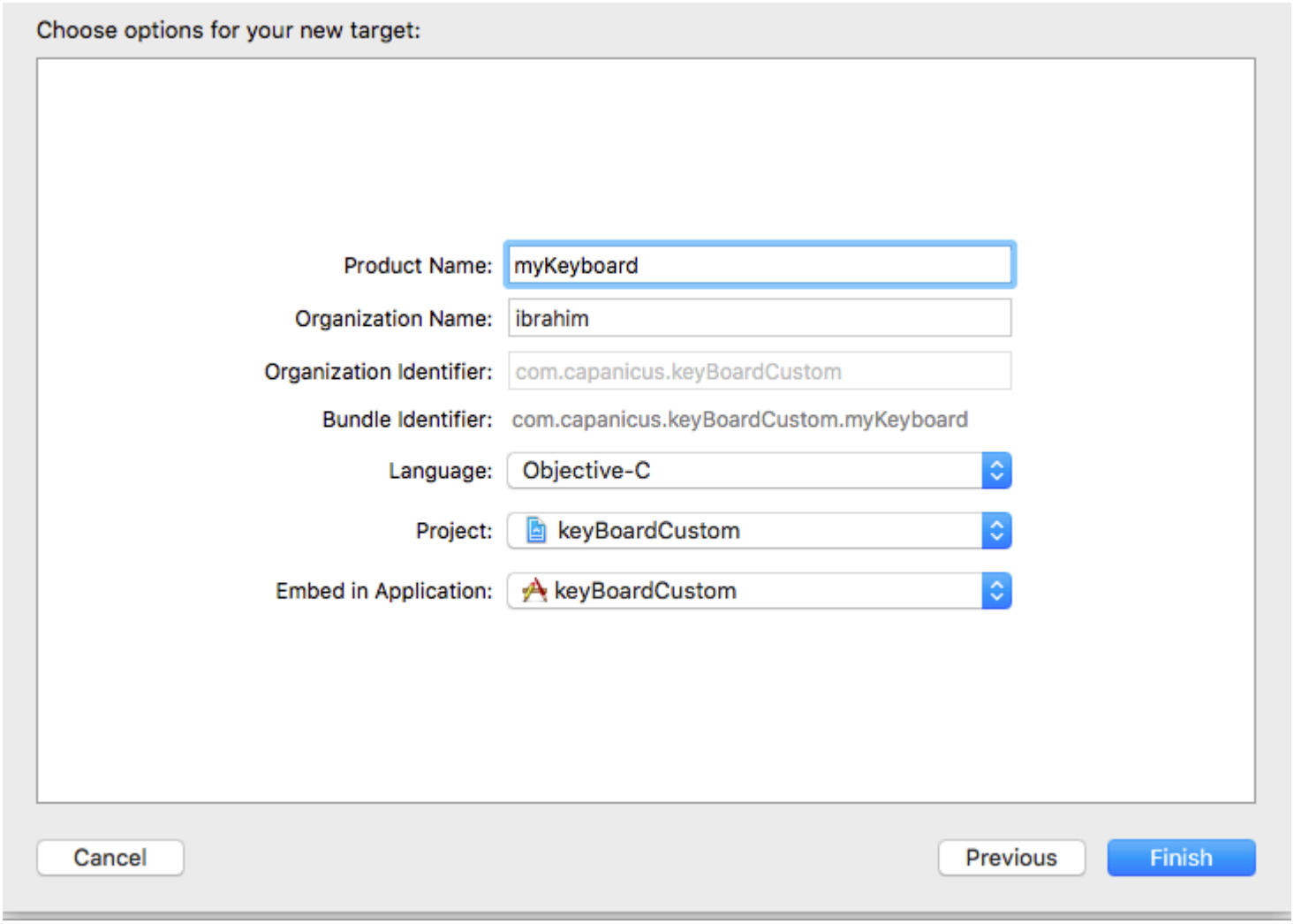
Добавить цель в существующий проект XCode



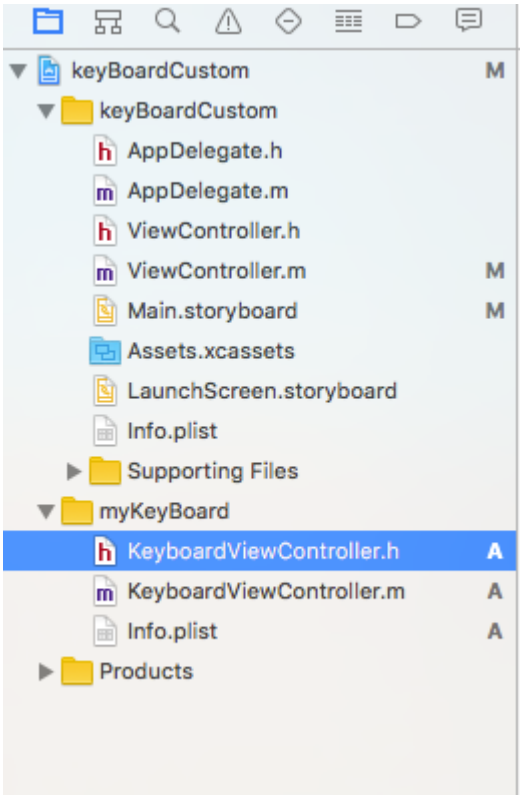
В Add Target выберите Custom KeyBoard



Добавьте цель следующим образом:

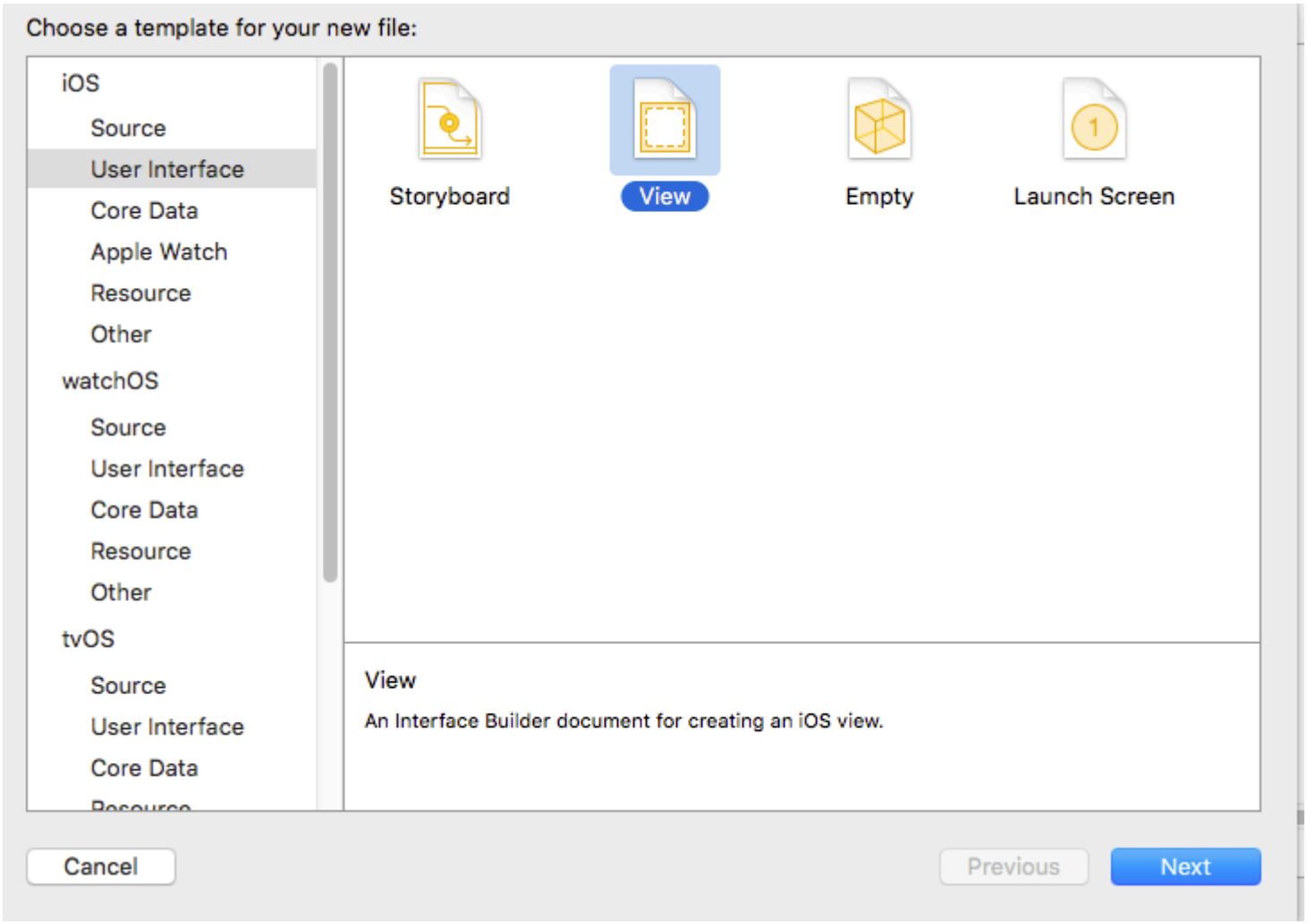


Каталог файлов проекта должен выглядеть примерно так:

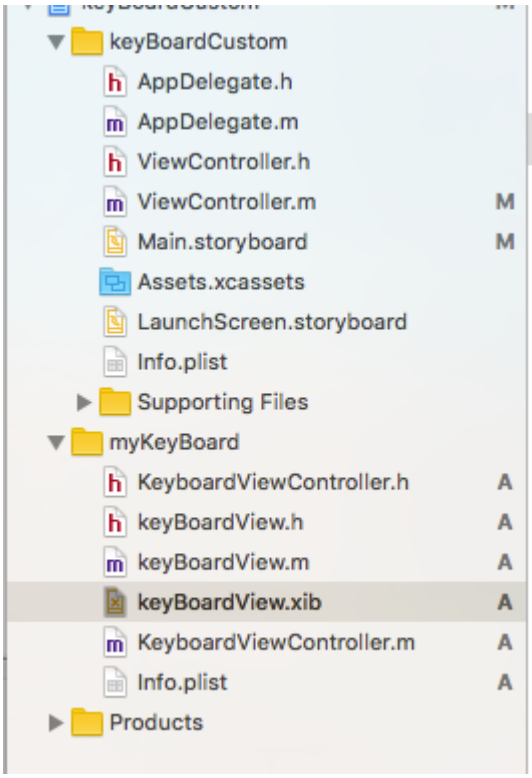


Здесь myKeyBoard - это имя добавленной цели

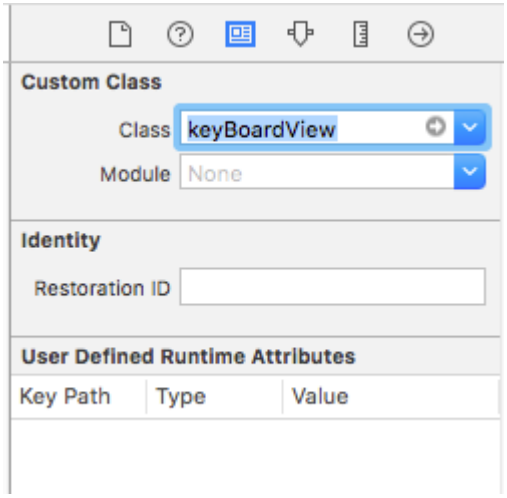
Добавьте новый файл Cocoatouch типа UIView и добавьте файл интерфейса



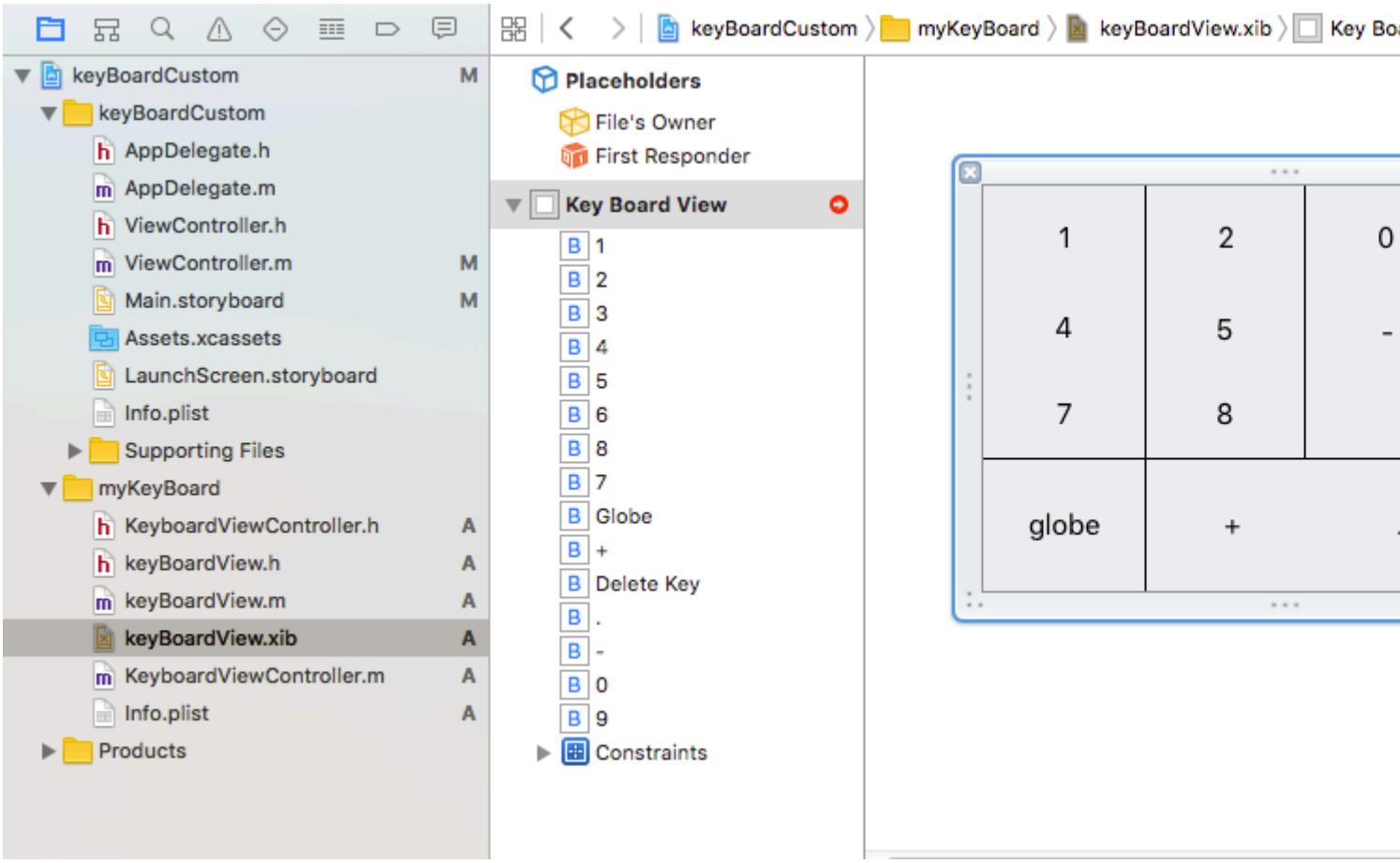
Наконец, ваш каталог проекта должен выглядеть так:



сделать `keyBoardView.xib` подклассом `keyBoardView`



Сделать интерфейс в файле `keyBoardView.xib`



Сделать `keyBoardView.xib` K `keyBoardView.h` `keyBoardView.xib` B `keyBoardView.h`

`keyBoardView.h` должен выглядеть так:

```
#import <UIKit/UIKit.h>

@interface keyBoardView : UIView

@property (weak, nonatomic) IBOutlet UIButton *deleteKey;
//IBOutlet for the delete Key
@property (weak, nonatomic) IBOutlet UIButton *globe;
//Outlet for the key with title globe which changes the keyboard type
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *keys;
//Contains a colloection of all the keys '0 to 9' '+' '-' and '.'

@end
```

В файле `keyBoardViewController.h` импортируйте `#import "keyBoardView.h"`

Объявить свойство клавиатуры `@property (strong, nonatomic)keyBoardView *keyboard;`

Прокомментируйте

```
@property (nonatomic, strong) UIButton *nextKeyboardButton and all the code associated with it
```

Функция viewDidLoad () функции KeyboardViewController.m должна выглядеть так:

```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.keyboard=[[[NSBundle mainBundle]loadNibNamed:@"keyBoardView" owner:nil
options:nil]objectAtIndex:0];
      self.inputView=self.keyboard;
    [self addGestureToKeyboard];

    // Perform custom UI setup here
//    self.nextKeyboardButton = [UIButton buttonWithType:UIButtonTypeSystem];
//
//    [self.nextKeyboardButton setTitle:NSLocalizedString(@"Next Keyboard", @"Title for 'Next
Keyboard' button") forState:UIControlStateNormal];
//    [self.nextKeyboardButton sizeToFit];
//    self.nextKeyboardButton.translatesAutoresizingMaskIntoConstraints = NO;
//
//    [self.nextKeyboardButton addTarget:self action:@selector(advanceToNextInputMode)
forControlEvents:UIControlEventTouchUpInside];
//
//    [self.view addSubview:self.nextKeyboardButton];
//
//    [self.nextKeyboardButton.leftAnchor constraintEqualToAnchor:self.view.leftAnchor].active
= YES;
//    [self.nextKeyboardButton.bottomAnchor
constraintEqualToAnchor:self.view.bottomAnchor].active = YES;
}
```

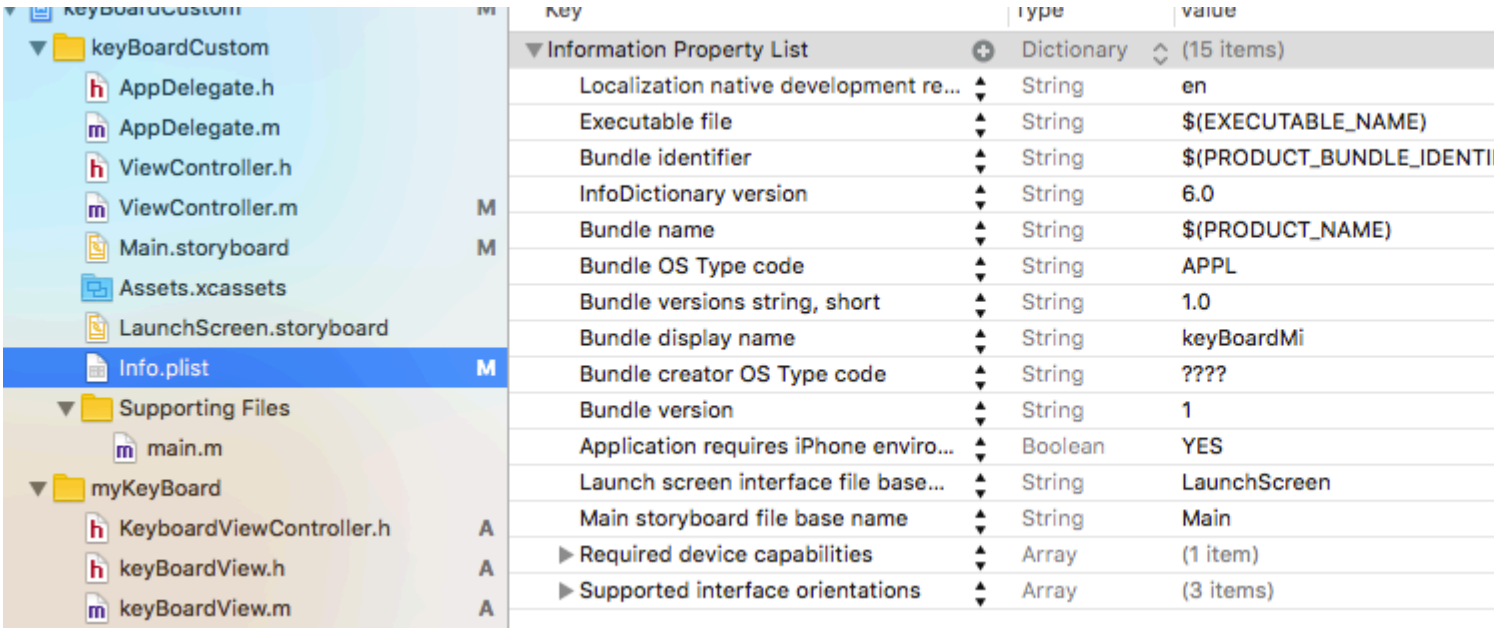
Функции `addGestureToKeyboard` , `pressDeleteKey` , `keyPressed` определены ниже

```
-(void) addGestureToKeyboard
{
    [self.keyboard.deleteKey addTarget:self action:@selector(pressDeleteKey)
forControlEvents:UIControlEventTouchUpInside];
    [self.keyboard.globe addTarget:self action:@selector(advanceToNextInputMode)
forControlEvents:UIControlEventTouchUpInside];

    for (UIButton *key in self.keyboard.keys)
    {
        [key addTarget:self action:@selector(keyPressed:)
forControlEvents:UIControlEventTouchUpInside];
    }


}
-(void) pressDeleteKey
{
    [self.textDocumentProxy deleteBackward];
}

-(void)keyPressed:(UIButton *)key
{
    [self.textDocumentProxy insertText:[key currentTitle]];
}
```

Запустите основное приложение и перейдите в Настройки-> Общие-> Клавиатура-> Добавить новую клавиатуру-> и добавьте клавиатуру с клавиатуры третьей стороны (Отображаемая клавиатураName будет keyBoardCustom)

Имя клавиатуры можно изменить, добавив ключ, называемый `Bundle display name` и в Value

String Value введите нужное имя для клавиатуры основного проекта.



Вы также можете посмотреть это видео Youtube

Прочитайте Пользовательская клавиатура онлайн: https://riptutorial.com/ru/ios/topic/7358/пользовательская-клавиатура

глава 168: Пользовательские UIViews из XIB-файлов

замечания

От Apple: создание пользовательского представления, которое визуализирует в интерфейсе Builder

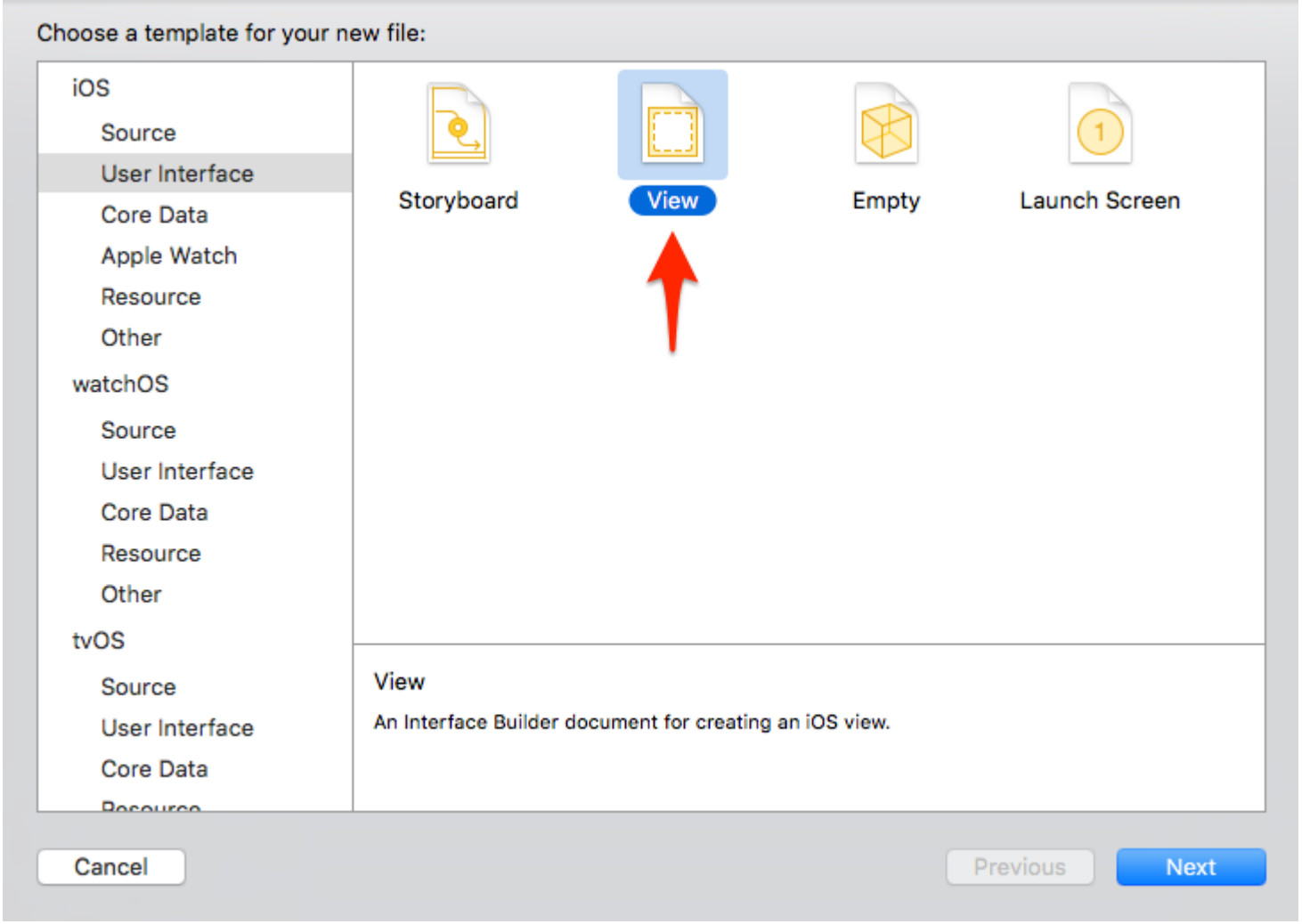
- Примечание. Имейте в виду, что если вы используете необычные «пользовательские» шрифты в ваших XIB-элементах (такие как UILabel, UITextField и т. Д.), Тогда начальное время загрузки вашего XIB будет больше в зависимости от выбранного шрифта и версии системы.

Examples

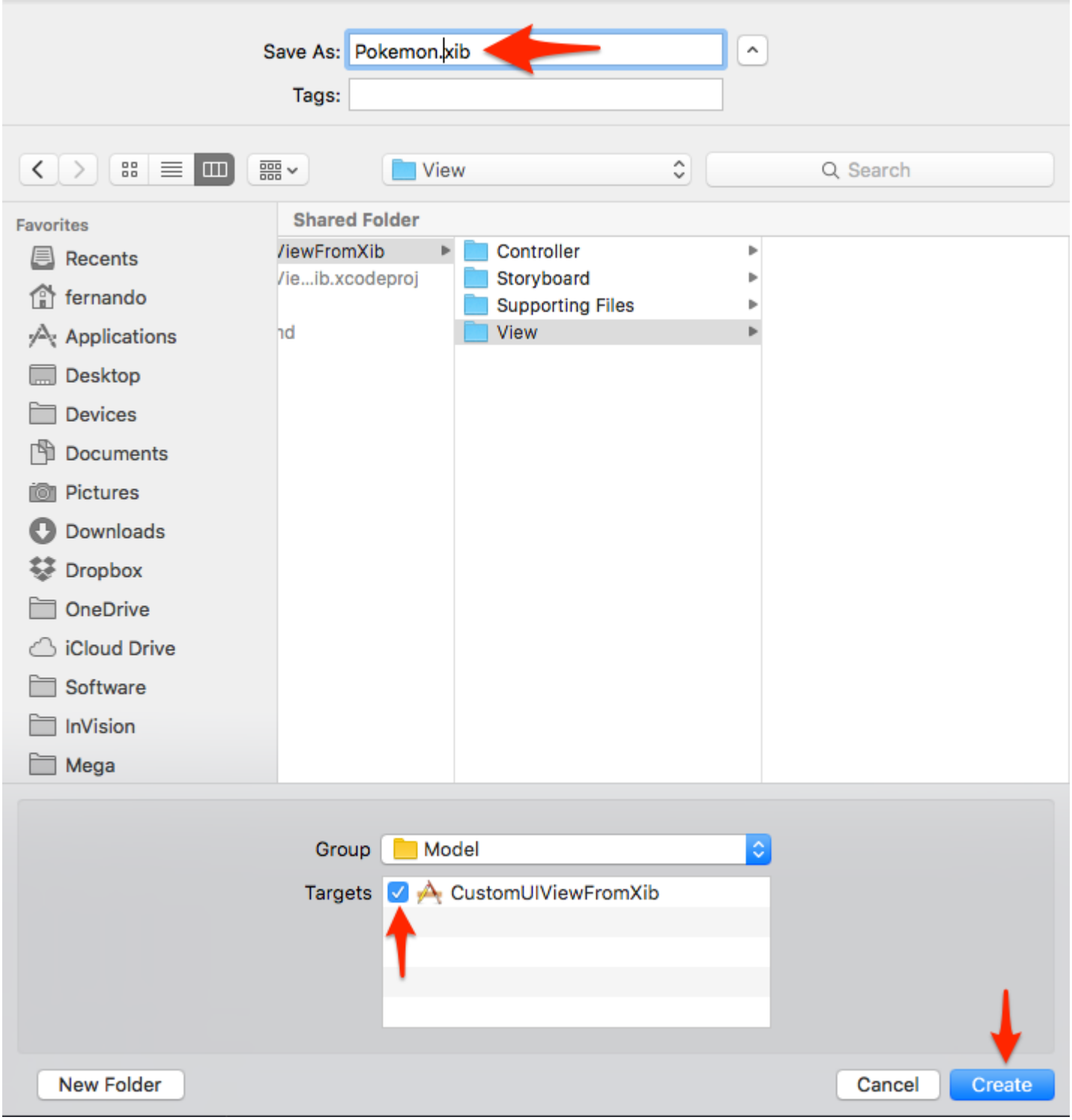
Элементы проводки

Создание файла XIB

Панель меню Xcode> Файл> Создать> Файл.
Выберите iOS, Пользовательский интерфейс, а затем «Вид»:



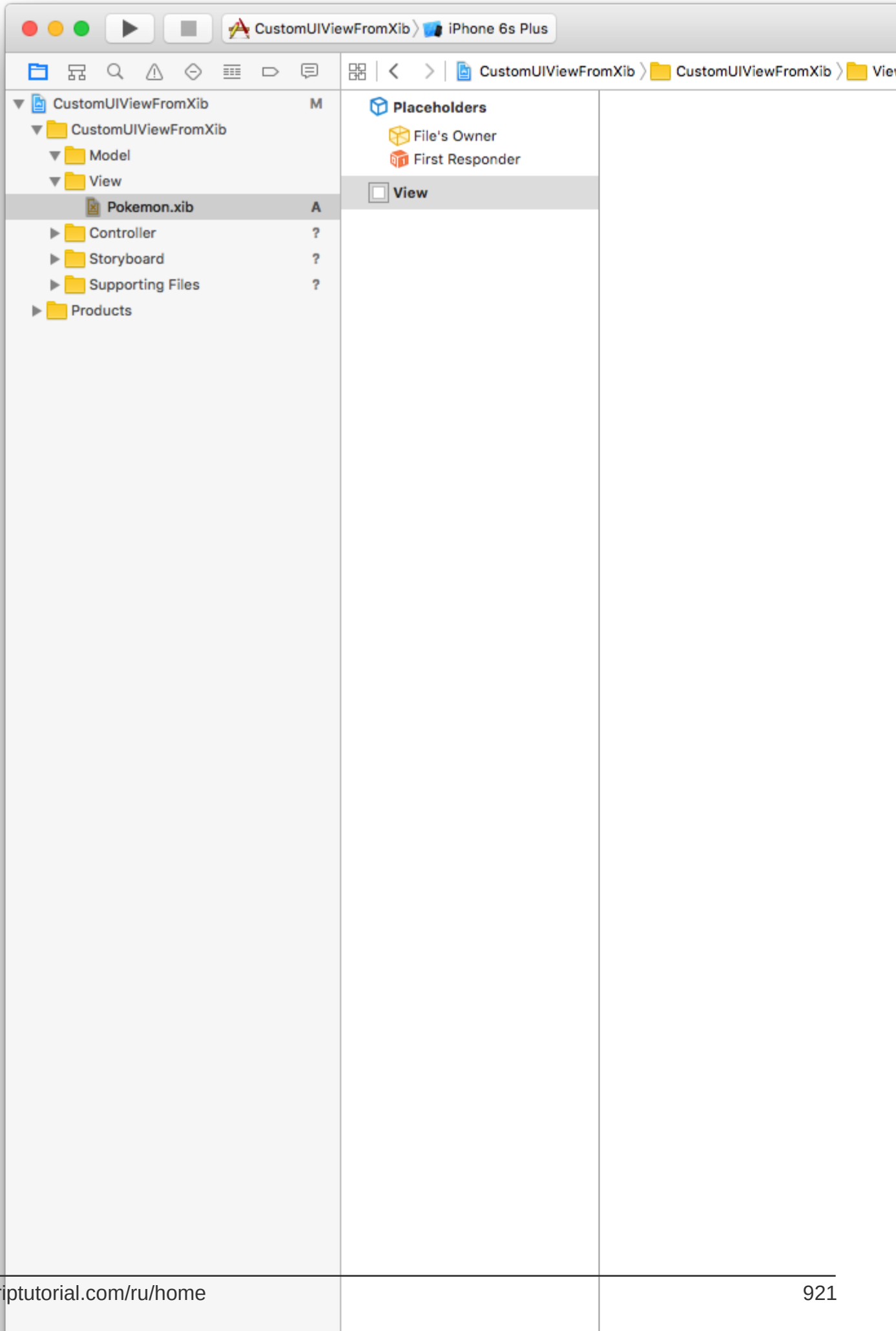
Дайте вашему XIB имя (да, мы делаем пример Pokemon).
Не забудьте проверить свою цель и нажать «Создать».



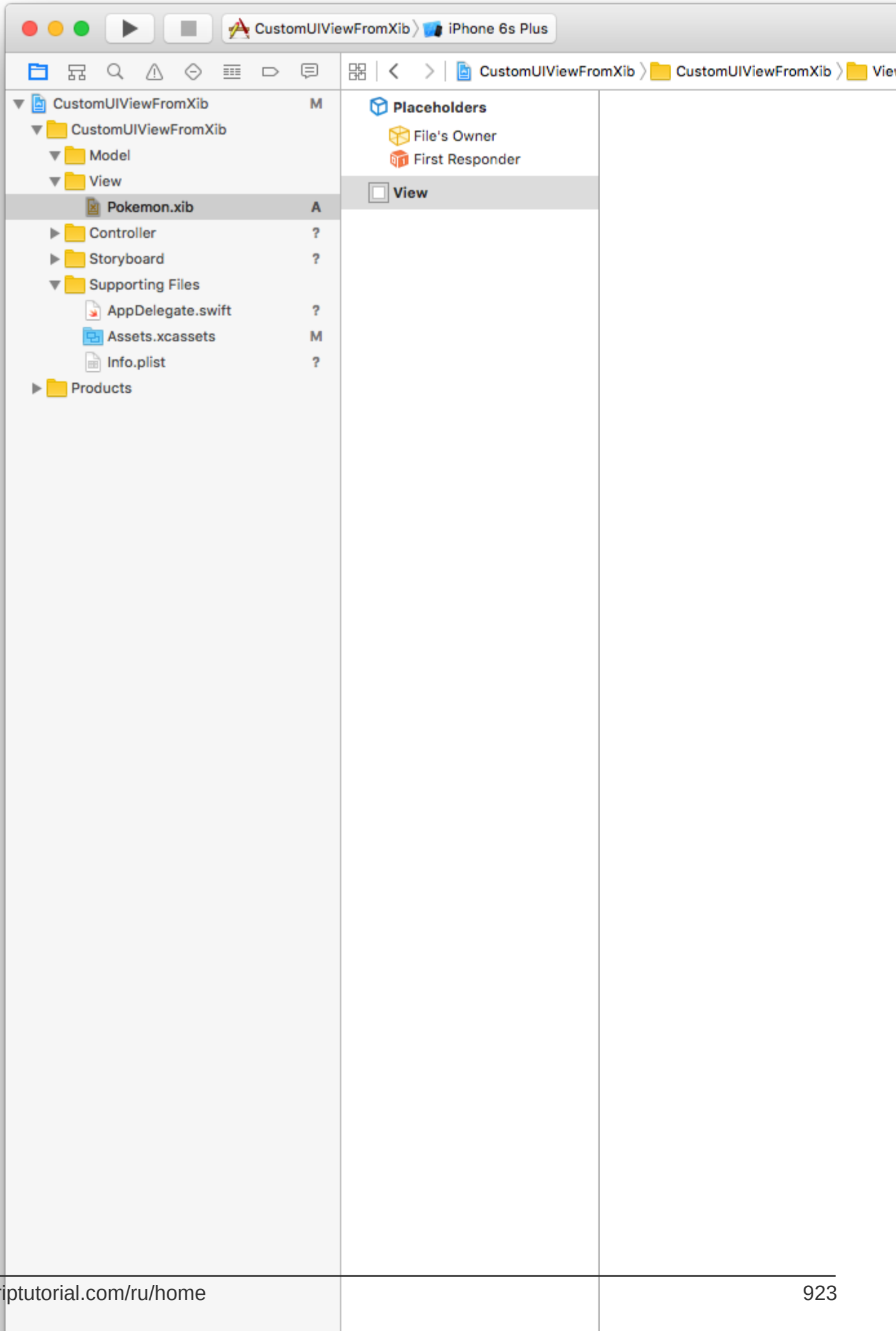
Создайте свой взгляд

Чтобы упростить задачу, установите:

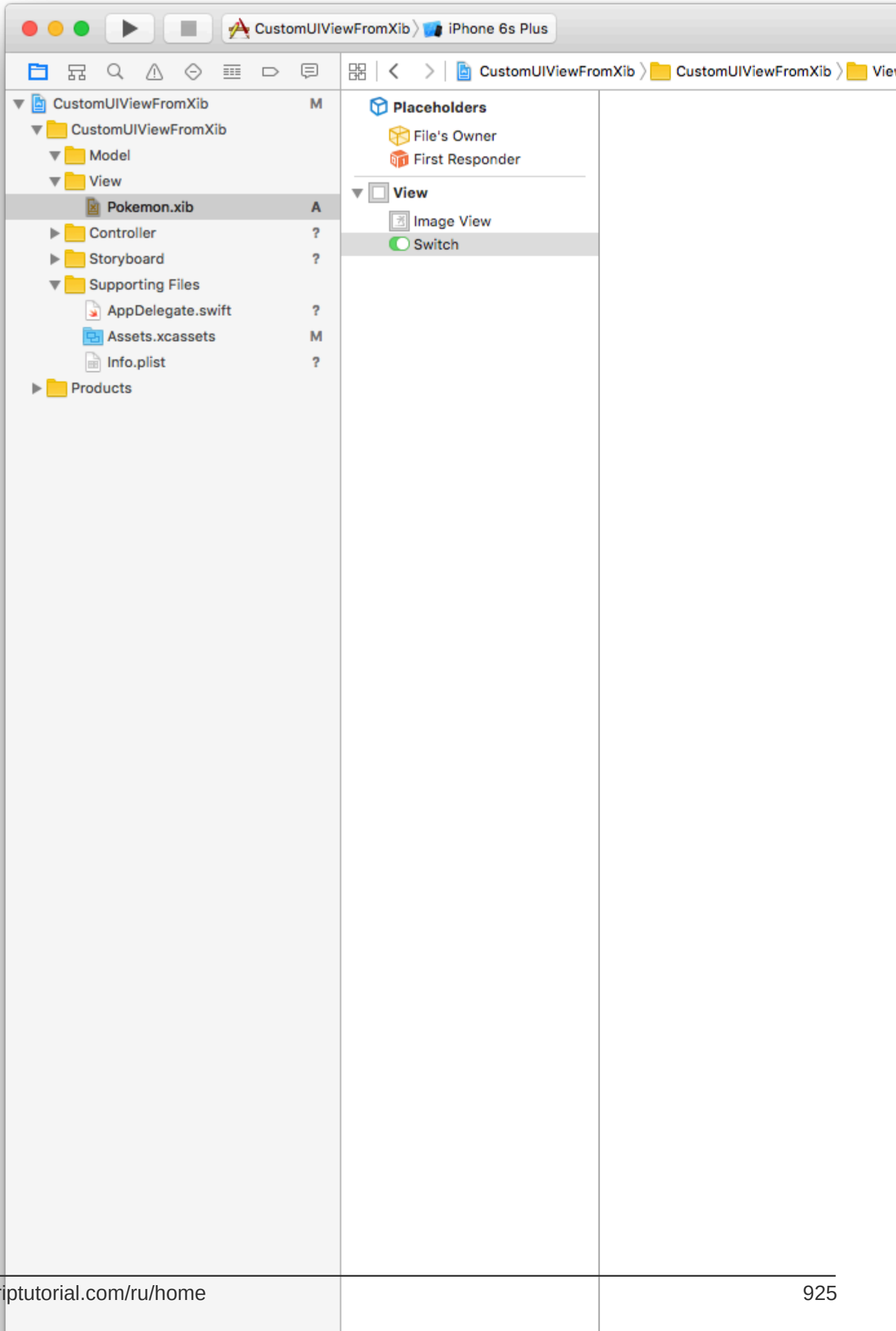
- Размер: Freeform
- Строка состояния: нет
- Верхняя панель: нет
- Нижняя панель: нет


CustomUIViewFromXib
iPhone 6s Plus
CustomUIViewFromXib
CustomUIViewFromXib
CustomUIViewFromXib
M
CustomUIViewFromXib
Model
View
Pokemon.xib
A
Controller
?
Storyboard
?
Supporting Files
?
Products
Placeholders
File's Owner
First Responder
View

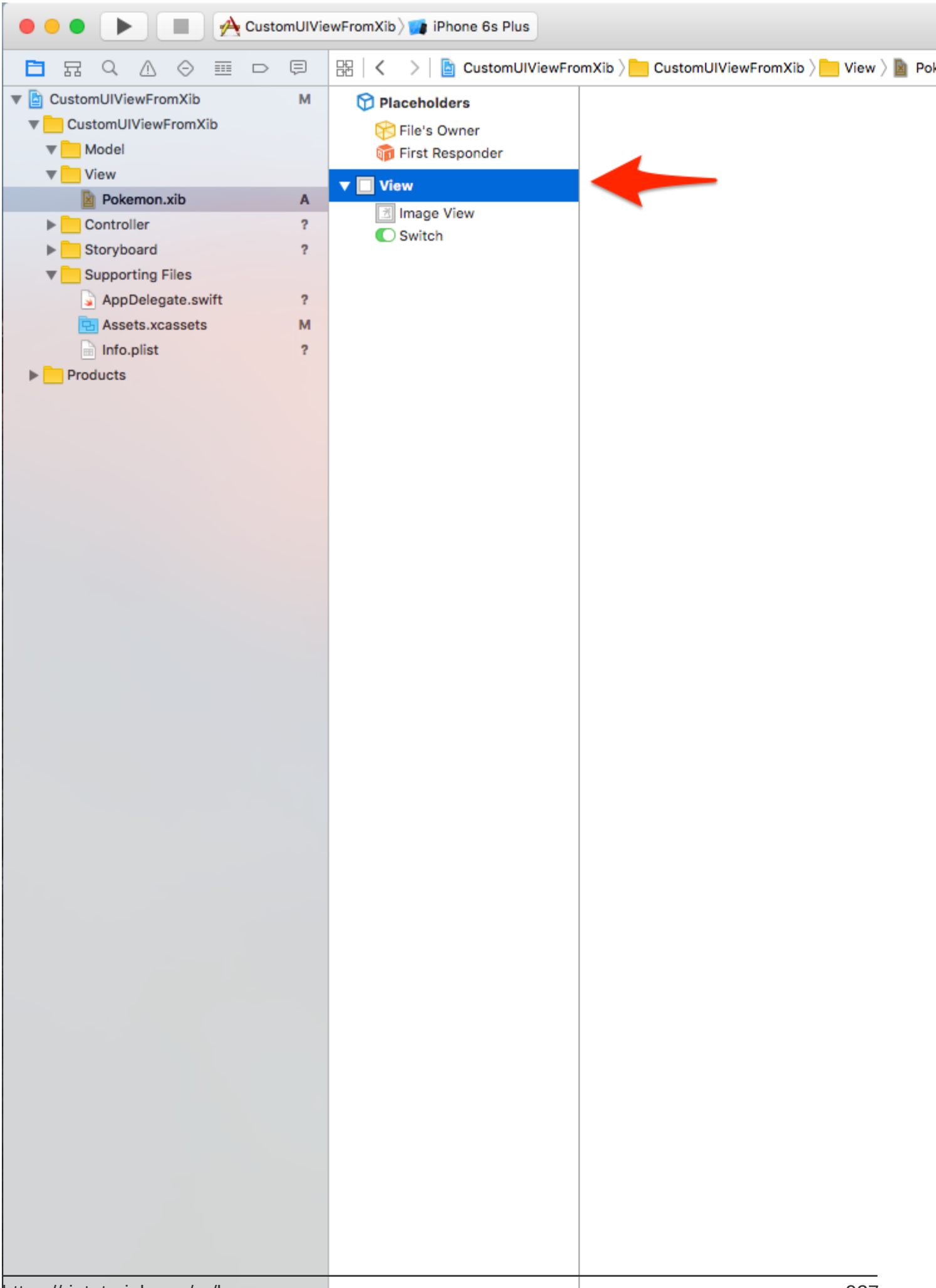
В этом примере мы будем использовать ширину 321 и высоту 256.


CustomUIViewFromXib
iPhone 6s Plus
CustomUIViewFromXib
CustomUIViewFromXib
CustomUIViewFromXib M
CustomUIViewFromXib
Model
View
Pokemon.xib A
Controller ?
Storyboard ?
Supporting Files
AppDelegate.swift ?
Assets.xcassets M
Info.plist ?
Products
Placeholders
File's Owner
First Responder
View

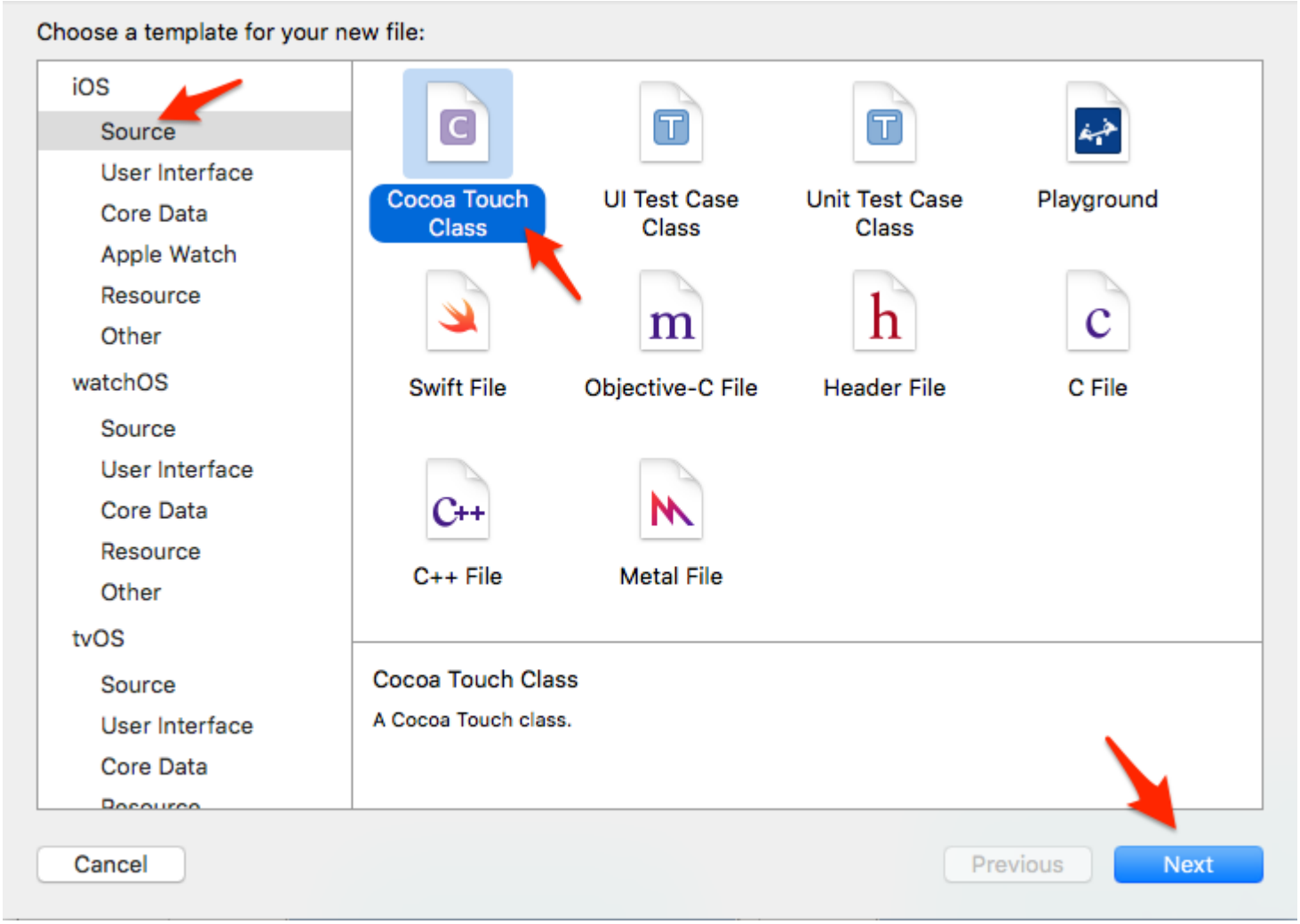
Здесь мы добавим **изображение** (256x256) и **переключатель** .


CustomUIViewFromXib
iPhone 6s Plus
CustomUIViewFromXib
CustomUIViewFromXib
CustomUIViewFromXib M
CustomUIViewFromXib
Model
View
Pokemon.xib A
Controller ?
Storyboard ?
Supporting Files
AppDelegate.swift ?
Assets.xcassets M
Info.plist ?
Products
Placeholders
File's Owner
First Responder
View
Image View
Switch

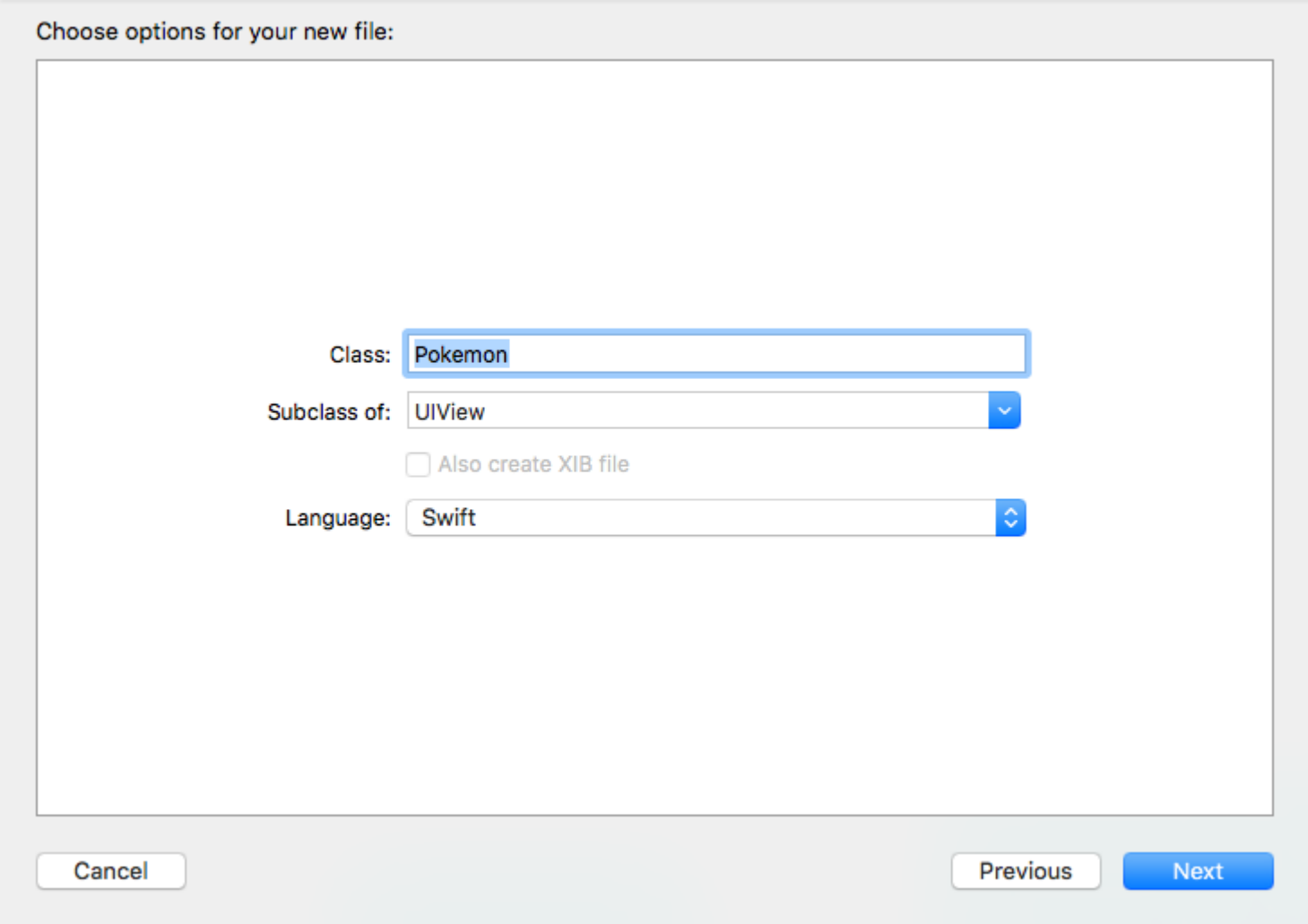
нажав «Разрешить автоматические макеты» (внизу справа) и выбрав «Добавить отсутствующие ограничения» в разделе «Все представления».


CustomUIViewFromXib
iPhone 6s Plus
CustomUIViewFromXib
CustomUIViewFromXib
View
CustomUIViewFromXib
M
CustomUIViewFromXib
Model
View
Pokemon.xib
A
Controller
?
Storyboard
?
Supporting Files
AppDelegate.swift
?
Assets.xcassets
M
Info.plist
?
Products
Placeholders
File's Owner
First Responder
View
Image View
Switch

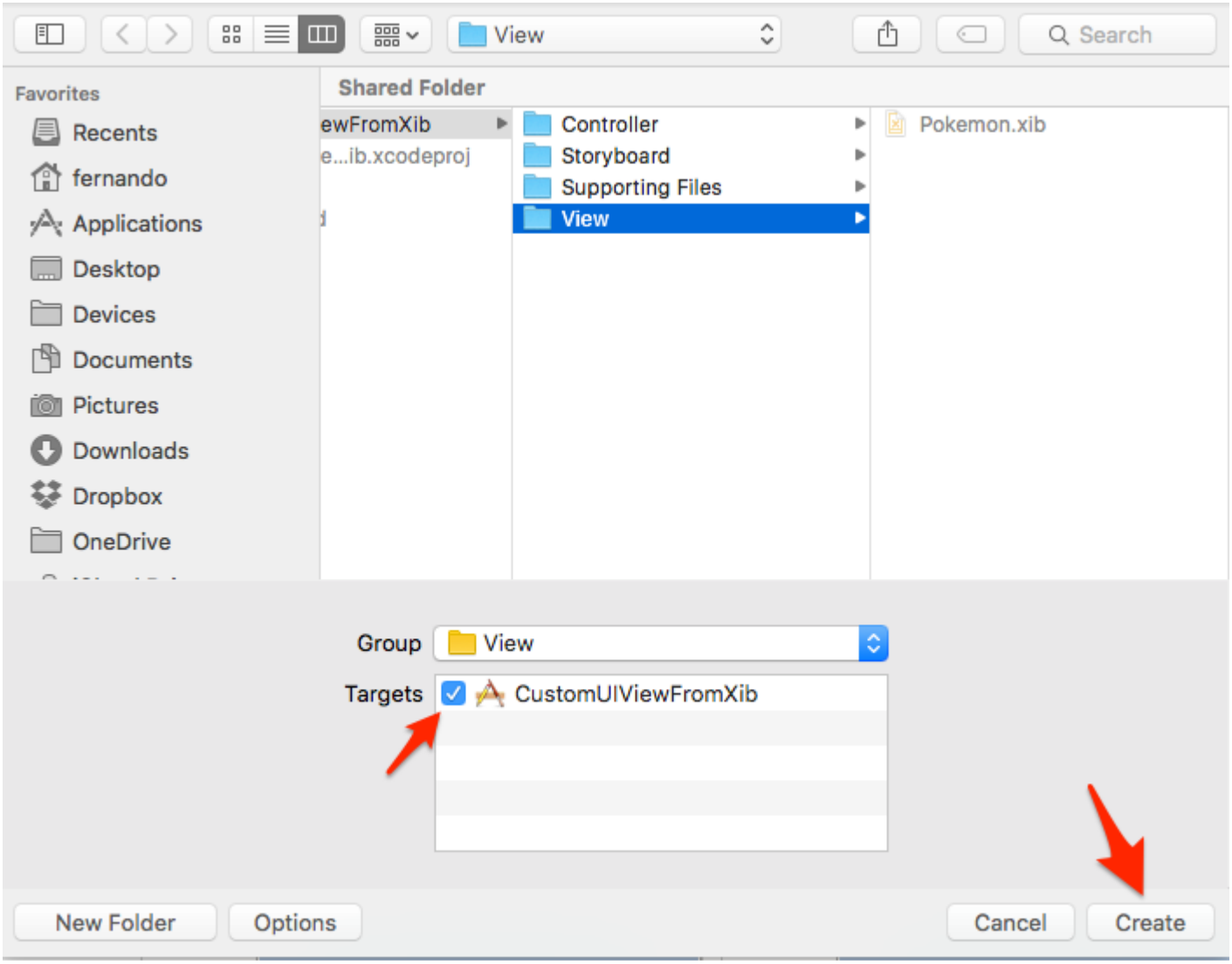
Панель меню Xcode> Файл> Создать> Файл.
Выберите iOS / Source / Cocoa Touch Class. Нажмите «Далее».



Дайте классу имя, которое должно быть тем же именем, что и файл XIB (Pokemon).
Выберите UIView в качестве типа подкласса, а затем нажмите «Далее».

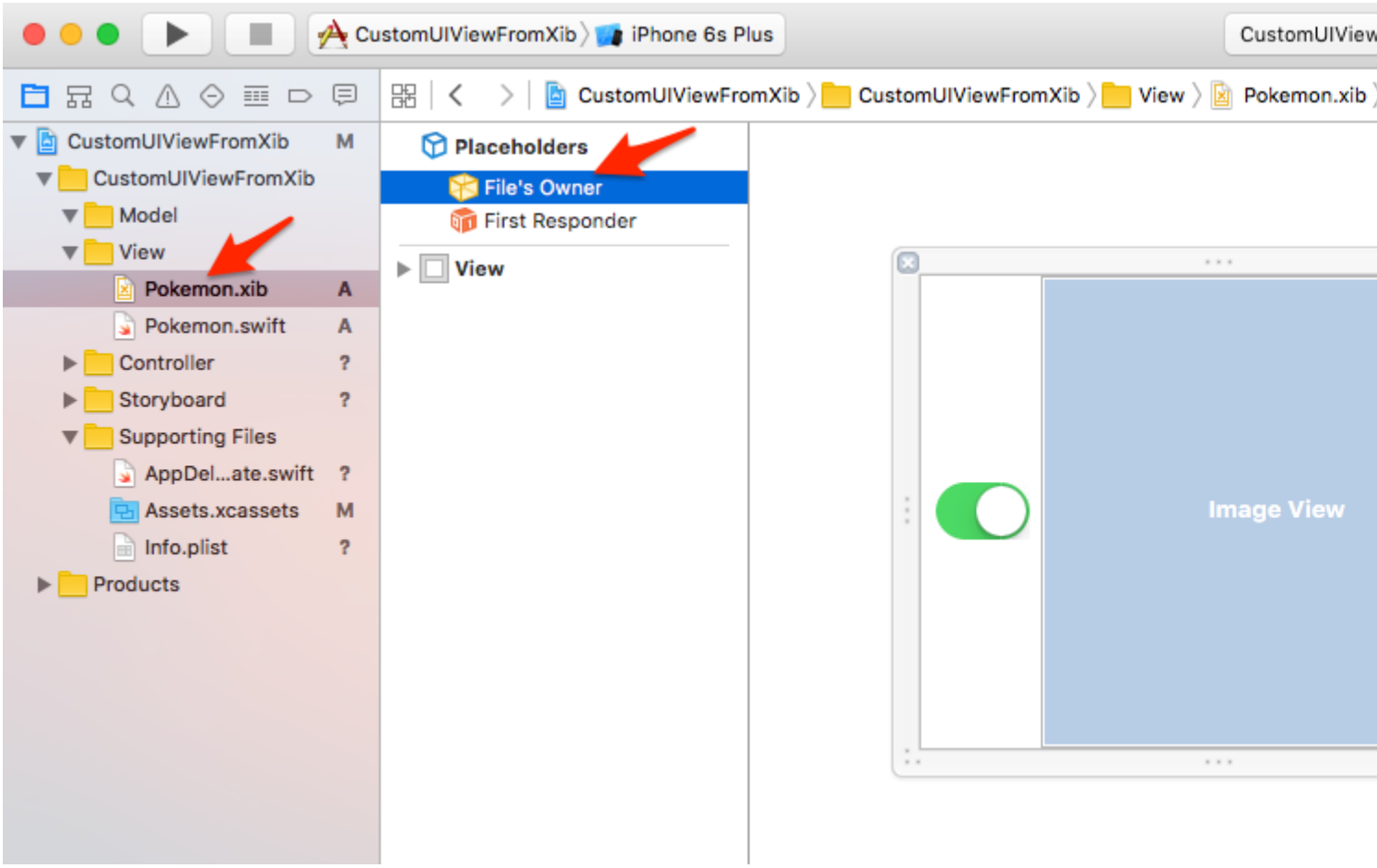


В следующем окне выберите цель и нажмите «Создать».



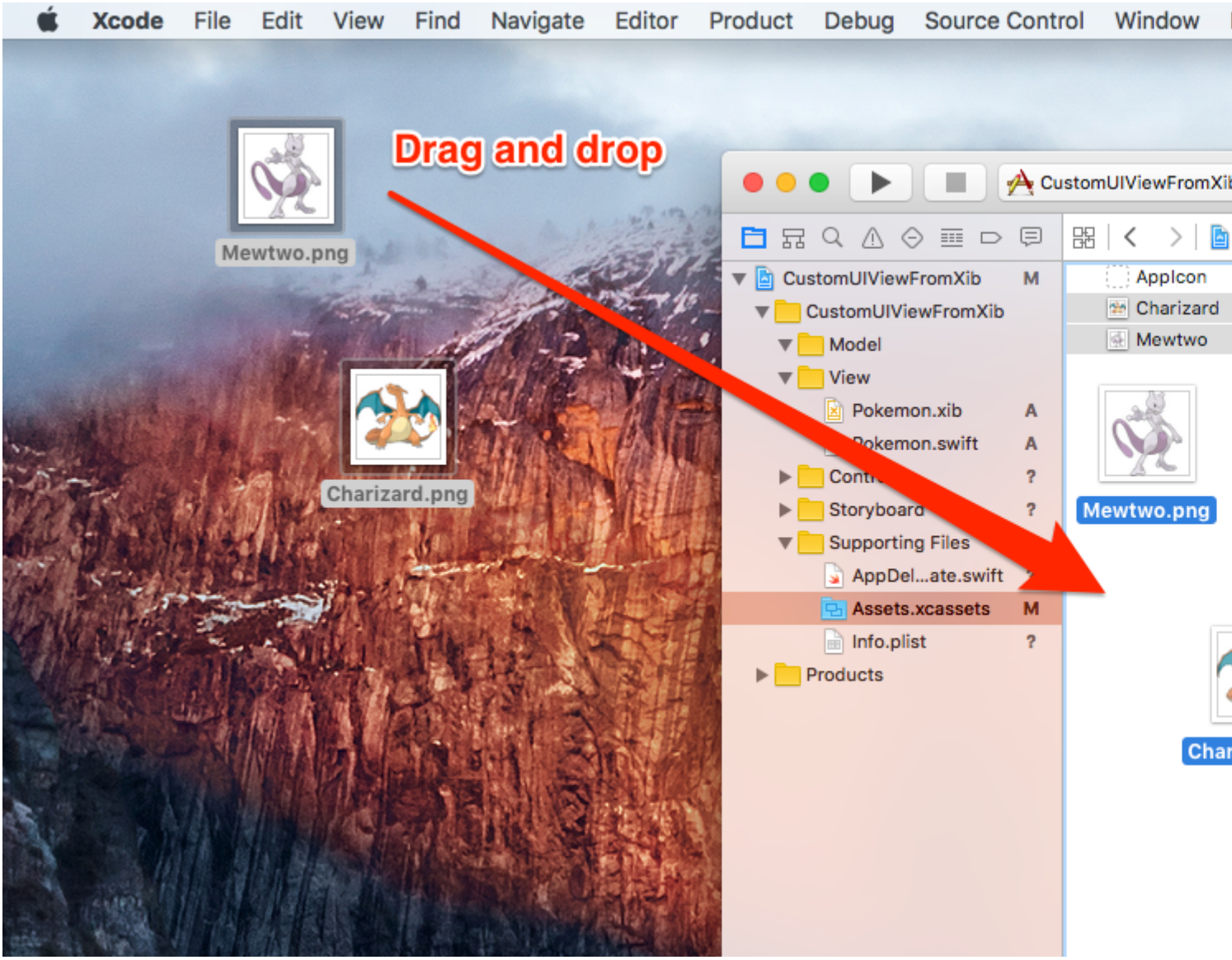
Подключите Pokemon.xib к Pokemon.swift с помощью атрибута «Владелец файла»

Нажмите на файл Pokemon.xib в Xcode.
Нажмите на кнопку «Владелец файла».
В «Инспекторе идентификации» (вверху справа) установите класс в наш недавно созданный файл Pokemon.swift.



Покемонов !!!

Да! Перетащите некоторые покемоны в свой проект, чтобы завершить нашу «инфраструктуру».
Здесь мы добавляем два файла PGN, 256x256, прозрачный.



Покажите мне код.

Хорошо, хорошо.
Время добавить код в наш класс Pokemon.swift.

На самом деле это довольно просто:

1. Внедрить необходимые инициализаторы
2. Загрузите файл XIB
3. Настройте представление, которое отобразит XIB-файл
4. Показать приведенный выше вид

Добавьте следующий код в класс Pokemon.swift:

```
import UIKit

class Pokemon: UIView {

    // MARK: - Initializers
```

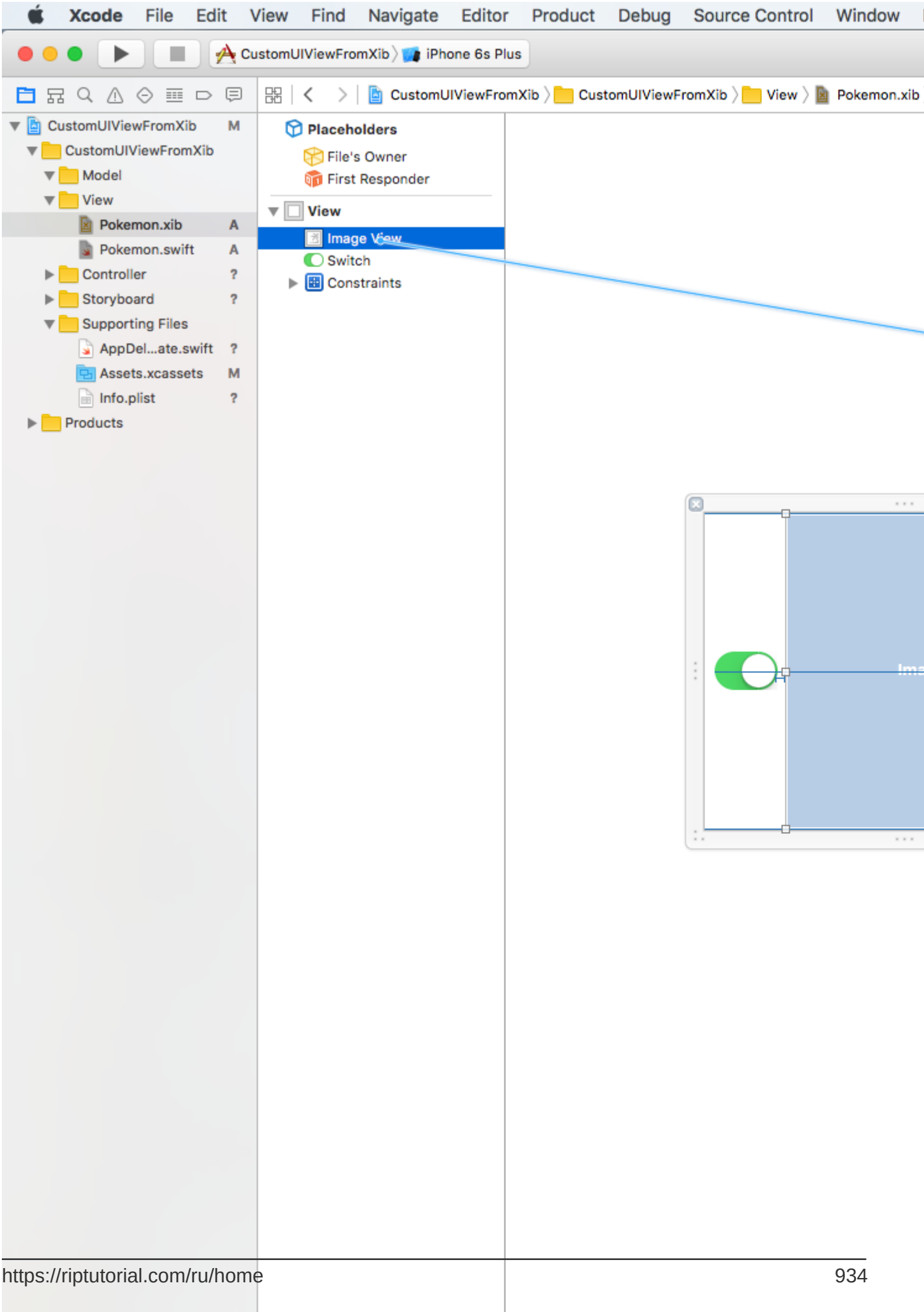
```
    override init(frame: CGRect) {
        super.init(frame: frame)
        setupView()
    }

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setupView()
    }

    // MARK: - Private Helper Methods

    // Performs the initial setup.
    private func setupView() {
        let view = viewFromNibForClass()
        view.frame = bounds

        // Auto-layout stuff.
        view.autoresizingMask = [
            UIViewAutoresizing.flexibleWidth,
            UIViewAutoresizing.flexibleHeight
        ]

        // Show the view.
        addSubview(view)
    }

    // Loads a XIB file into a view and returns this view.
    private func viewFromNibForClass() -> UIView {

        let bundle = Bundle(for: type(of: self))
        let nib = UINib(nibName: String(describing: type(of: self)), bundle: bundle)
        let view = nib.instantiate(withOwner: self, options: nil).first as! UIView

        /* Usage for swift < 3.x
        let bundle = NSBundle(forClass: self.dynamicType)
        let nib = UINib(nibName: String(self.dynamicType), bundle: bundle)
        let view = nib.instantiateWithOwner(self, options: nil)[0] as! UIView
        */

        return view
    }
}
```

@IBDesignable и @IBInspectable

Добавив `@IBDesignable` к вашему классу, вы можете сделать это для рендеринга в Interface Builder.
Добавляя `@IBInspectable` к свойствам вашего класса, вы можете увидеть изменения пользовательских представлений в Interface Builder, как только вы измените эти свойства.

Давайте сделаем `Image View` нашего пользовательского представления «Inspectable».

Сначала подключите `Image View` из файла Pokemon.xib к классу Pokemon.swift.

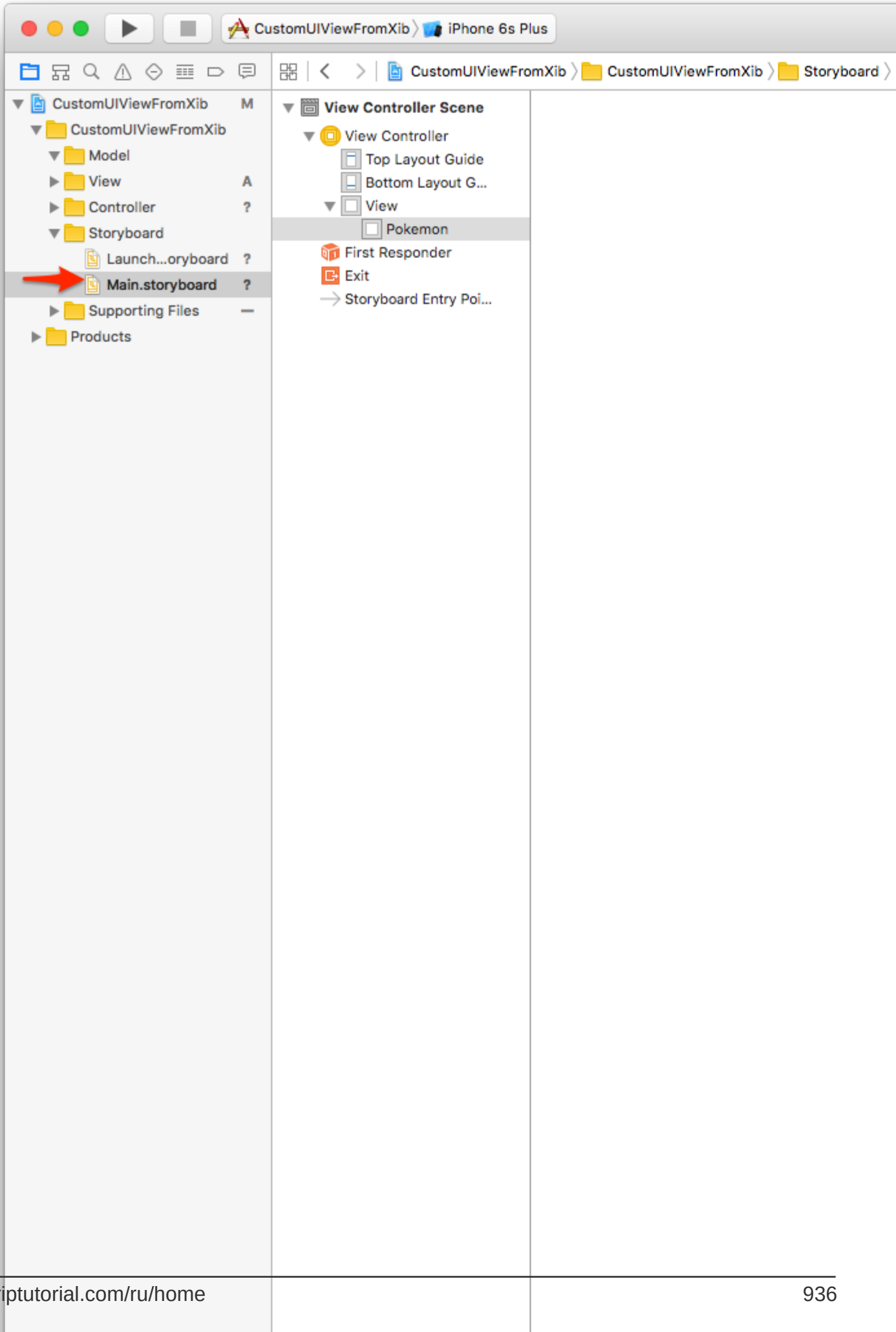

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window
CustomUIViewFromXib iPhone 6s Plus
CustomUIViewFromXib CustomUIViewFromXib View Pokemon.xib
CustomUIViewFromXib M
CustomUIViewFromXib
Model
View
Pokemon.xib A
Pokemon.swift A
Controller ?
Storyboard ?
Supporting Files
AppDel...ate.swift ?
Assets.xcassets M
Info.plist ?
Products
Placeholders
File's Owner
First Responder
View
Image View
Switch
Constraints

перед именем класса):

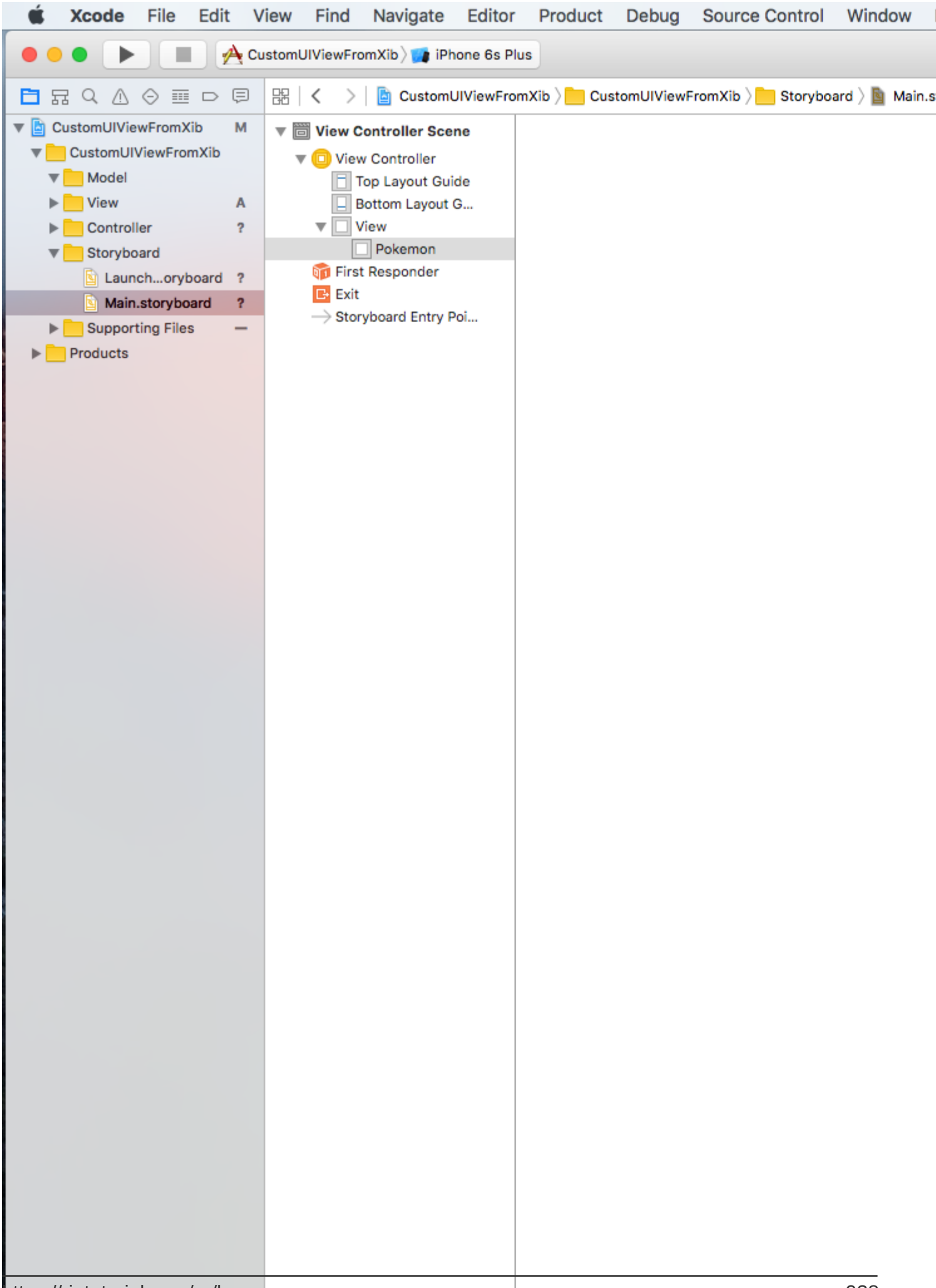
```
@IBDesignable class Pokemon: UIView {

    // MARK: - Properties

    @IBOutlet weak var imageView: UIImageView!

    @IBInspectable var image: UIImage? {
        get {
            return imageView.image
        }
        set(image) {
            imageView.image = image
        }
    }

    // MARK: - Initializers
    ...
```

Использование пользовательских представлений

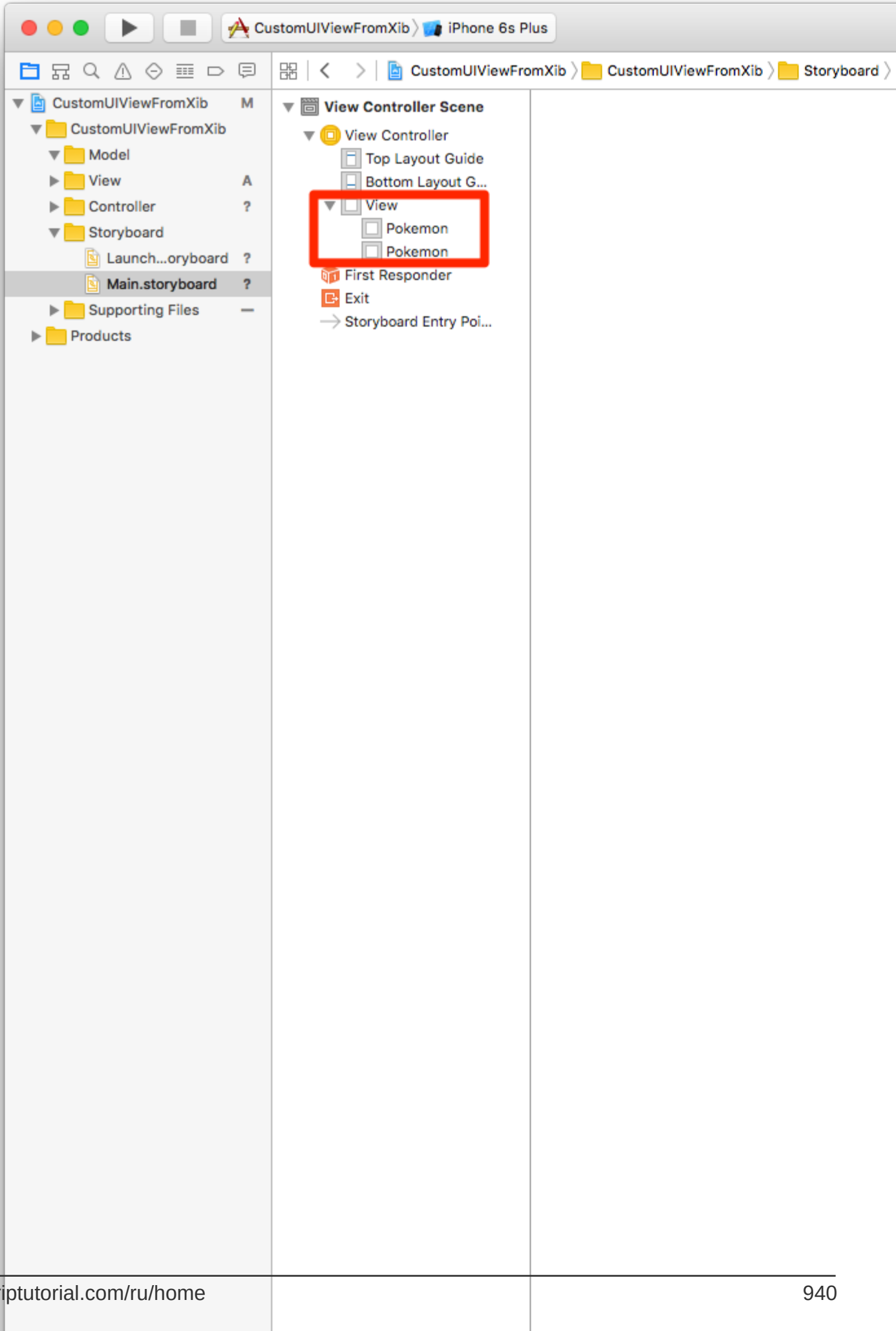
Попав в основной файл раскадровки, перетащите UIView в него.
Измените размер представления, скажем, 200x200. Централизация.
Перейдите к инспектору идентификации (вверху справа) и установите класс «Покемон».


CustomUIViewFromXib
iPhone 6s Plus
CustomUIViewFromXib
CustomUIViewFromXib
Storyboard
CustomUIViewFromXib
M
CustomUIViewFromXib
Model
View
A
Controller
?
Storyboard
Launch...oryboard
?
Main.storyboard
?
Supporting Files
Products
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout G...
View
Pokemon
First Responder
Exit
Storyboard Entry Poi...

перейдите к Инспектору атрибутов (вверху справа) и выберите один из изображений Pokemon, которые вы ранее добавили, используя удивительное `@IBInspectable` свойства `@IBInspectable` .


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window
CustomUIViewFromXib iPhone 6s Plus
CustomUIViewFromXib CustomUIViewFromXib Storyboard Main.s
CustomUIViewFromXib M
CustomUIViewFromXib
Model
View A
Controller ?
Storyboard
Launch...oryboard ?
Main.storyboard ?
Supporting Files
Products
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout G...
View
Pokemon
First Responder
Exit
Storyboard Entry Poi...

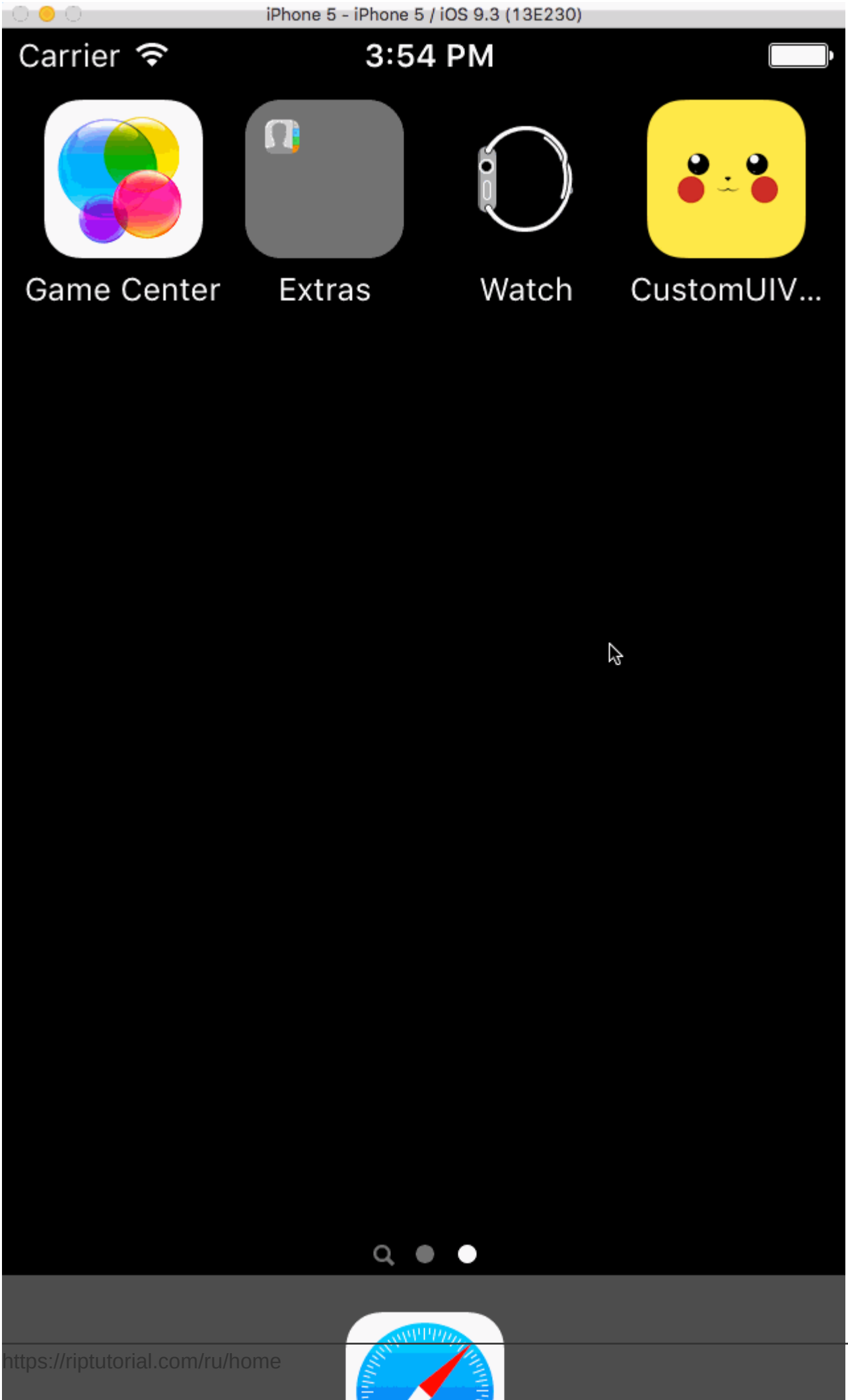
Дайте ему другой размер, скажем 150x150.
Выберите другое изображение Pokemon, соблюдайте:


CustomUIViewFromXib
iPhone 6s Plus
CustomUIViewFromXib
CustomUIViewFromXib
Storyboard
CustomUIViewFromXib
M
CustomUIViewFromXib
Model
View
A
Controller
?
Storyboard
Launch...oryboard
?
Main.storyboard
?
Supporting Files
Products
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout G...
View
Pokemon
Pokemon
First Responder
Exit
Storyboard Entry Poi...

Вызвать действие вроде `switchTapped` .
Добавьте к нему следующий код:

```
// MARK: - Actions

@IBAction func switchTapped(sender: UISwitch) {
    imageView.alpha = sender.on ? 1.0 : 0.2
}

// MARK: - Initializers
...
```

Конечный результат:


iPhone 5 - iPhone 5 / iOS 9.3 (13E230)
Carrier
3:54 PM
Game Center
Extras
Watch
CustomUIV...

Теперь вы можете создавать сложные пользовательские представления и использовать их в любом месте.
Это повысит производительность при изоляции кода в автономных элементах пользовательского интерфейса.

Окончательный проект можно клонировать в Github.
(**Обновлено до Swift 3.1**)

Как сделать пользовательский повторно используемый UIView с помощью XIB

В следующем примере показаны шаги, связанные с инициализацией представления из XIB.

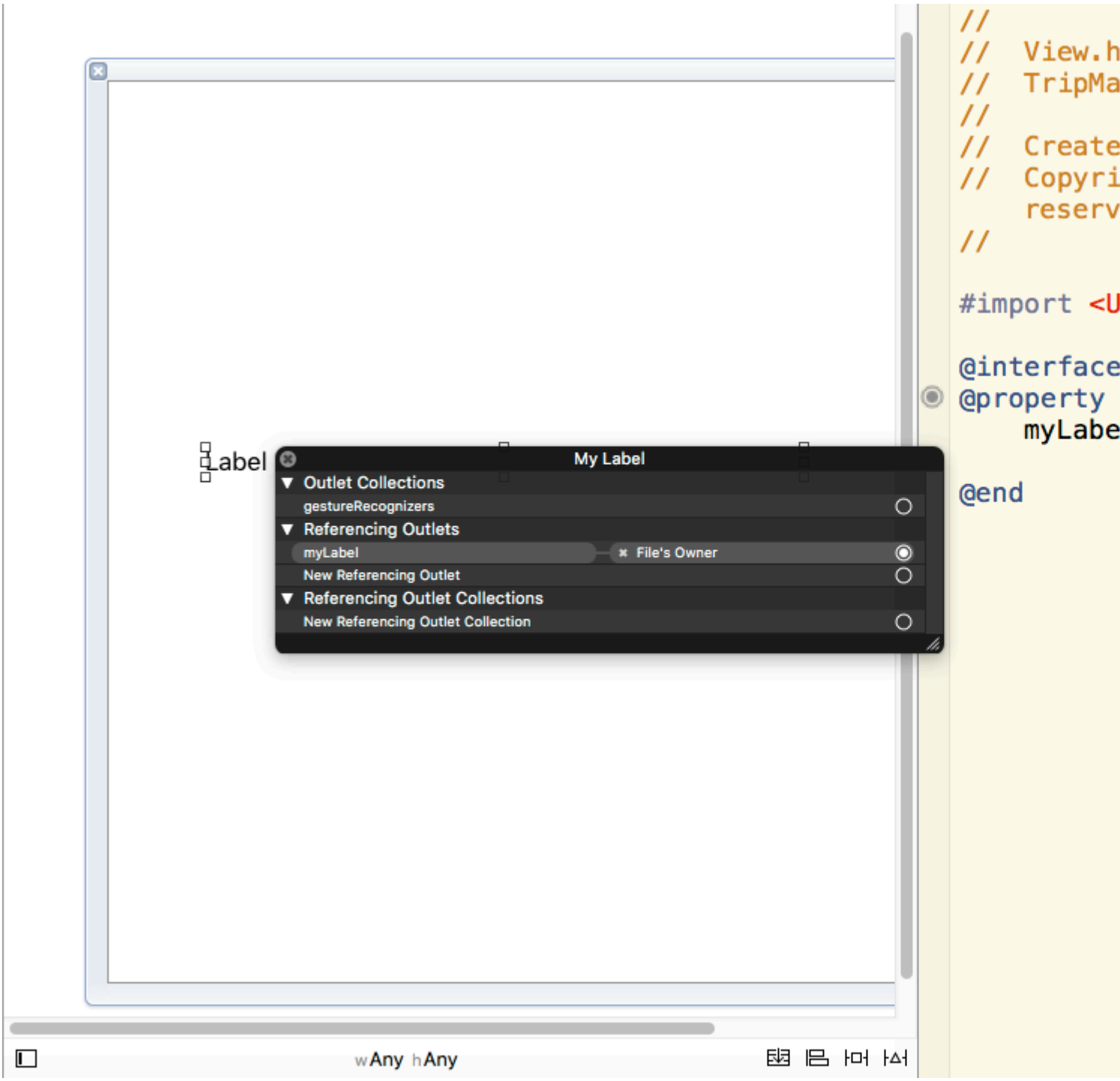
Это не сложная операция, но необходимо выполнить точные шаги, чтобы сделать это правильно в первый раз, избегая исключений.

Как работает loadNibNamed

Основные шаги:

1. Создать XIB
2. Создать класс .h и .m
3. Определение точек в .h
4. Подключите розетки между .h и XIB

См. Прилагаемый скриншот:



5. Вызывать loadNibNamed внутри функции initWithCoder файла .m. Это необходимо для того, чтобы вы могли напрямую разместить объект UIView в файл раскадровки / родительского UIView XIB и определить его как свое собственное представление. При загрузке раскадровки / родительского XIB другого кода инициализации не требуется. Пользовательский вид может быть добавлен к другим представлениям, как и другие встроенные объекты Objective C view, заданные в XCode.

Прочитайте Пользовательские UIViews из XIB-файлов онлайн:
https://riptutorial.com/ru/ios/topic/1362/пользовательские-uiviews-из-xib-файлов

глава 169: Пользовательские методы выбора UITableViewCells

Вступление

Продвигайте способы управления выборами UITableViewCell. Примеры, когда simple `didSelect...` form `UITableViewDelegate` недостаточно для достижения чего-то.

Examples

Различие между одиночным и двойным выбором в строке.

Пример реализации, который дает возможность определить, является ли пользователь одиночным или двойным нажатием на UITableViewCell.

```
override func viewDidLoad() {
    viewDidLoad()

    let doubleTapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleDoubleTap(sender:)))
    doubleTapGestureRecognizer.numberOfTapsRequired = 2
    tableView.addGestureRecognizer(doubleTapGestureRecognizer)

    let tapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleTapGesture(sender:)))
    tapGestureRecognizer.numberOfTapsRequired = 1
    tapGestureRecognizer.require(toFail: doubleTapGestureRecognizer)
    tableView.addGestureRecognizer(tapGestureRecognizer)
}

func handleTapGesture(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}

func handleDoubleTap(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}
```

Прочитайте Пользовательские методы выбора UITableViewCells онлайн:
https://riptutorial.com/ru/ios/topic/9961/пользовательские-методы-выбора-uitableviewcells

глава 170: Пользовательские методы выбора UITableViewCells

Examples

Различие между одиночным и двойным выбором в строке.

Пример реализации UITableView, который позволяет определить, была ли ячейка использована одно или два раза.

```
override func viewDidLoad() {
    viewDidLoad()

    let doubleTapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleDoubleTap(sender:)))
    doubleTapGestureRecognizer.numberOfTapsRequired = 2
    tableView.addGestureRecognizer(doubleTapGestureRecognizer)

    let tapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleTapGesture(sender:)))
    tapGestureRecognizer.numberOfTapsRequired = 1
    tapGestureRecognizer.require(toFail: doubleTapGestureRecognizer)
    tableView.addGestureRecognizer(tapGestureRecognizer)
}

func handleTapGesture(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}

func handleDoubleTap(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}
```

Прочитайте Пользовательские методы выбора UITableViewCells онлайн: https://riptutorial.com/ru/ios/topic/9962/пользовательские-методы-выбора-uitableviewcells

глава 171: Пользовательские шрифты

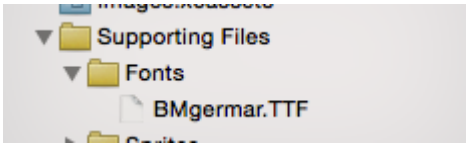
Examples

Встраивание пользовательских шрифтов

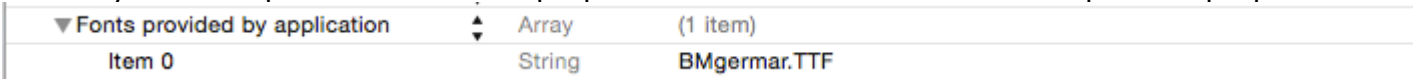
> **Поддержка пользовательских шрифтов**
> Приложения, которые хотят использовать пользовательские шрифты, теперь могут включать эти шрифты в свой пакет приложений и регистрировать эти шрифты с помощью системы, включая ключ UIAppFonts в свой файл Info.plist. Значение этого ключа представляет собой массив строк, идентифицирующих файлы шрифтов в пакете приложения. Когда система видит ключ, он загружает указанные шрифты и делает их доступными для приложения.

После того, как шрифты были установлены в `Info.plist` , вы можете использовать свои пользовательские шрифты как любой другой шрифт в IB или программно.

1. Перетащите свой шрифт в папку Xcode Supporting Files. Не забудьте отметить свое приложение в разделе «Добавить в целевые объекты». С этого момента вы можете использовать этот шрифт в IB и выбрать его из палитры шрифтов.



2. Чтобы этот шрифт был доступен на устройстве, откройте `Info.plist` и добавьте `Fonts provided by application key` (UIAppFonts). Добавьте имя шрифта в качестве значения в клавишу Item 0. Примечание. Имя шрифта может отличаться от имени файла шрифта.



3. Получить пользовательское добавленное имя шрифта, используя нижеприведенный фрагмент

[*Swift 3*]

```
for family in UIFont.familyNames {
            print("\(family)")

            for name in UIFont.fontNames(forFamilyName: family) {
                print("   \(name)")
            }
        }
```

[*Цель - C*]

```
for (NSString *familyName in [UIFont familyNames]){
        NSLog(@"Family name: %@", familyName);
        for (NSString *fontName in [UIFont fontNamesForFamilyName:familyName]) {
            NSLog(@"--Font name: %@", fontName);
        }
    }
```

Пользовательские шрифты с раскадровки

Пользовательские шрифты для компонентов пользовательского интерфейса из раскадровки можно легко достичь с помощью пользовательских атрибутов Runtime в раскадровке и категориях .

Преимущества таковы,

- Нет необходимости определять выходные точки для элемента ui
- Нет необходимости устанавливать шрифт для элементов программно.

  Шаги, чтобы следовать

  1. **Файл шрифта:** добавьте файл шрифта (.ttf) в пакет приложений и добавьте запись для шрифта в Info.plist в разделе ***Шрифт, предоставляемый приложением,*** как в этой документации пользовательских шрифтов.

  2. **Определите категории:** добавьте файл, подобный ***UIKit + IBExtensions,*** и добавьте категории элементов UI, такие как UILabel, UIButton и т. Д., Для которых вы хотите установить собственный шрифт. Все категории будут иметь настраиваемое свойство say **fontName** . Это позже будет использоваться из раскадровки для установки пользовательского шрифта (как на шаге 4).

UIKit + IBExtensions.h

```
#import <UIKit/UIKit.h>

//Category extension for UILabel
@interface UILabel (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end

// Category extension for UITextField
@interface UITextField (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end

// Category extension for UIButton
```

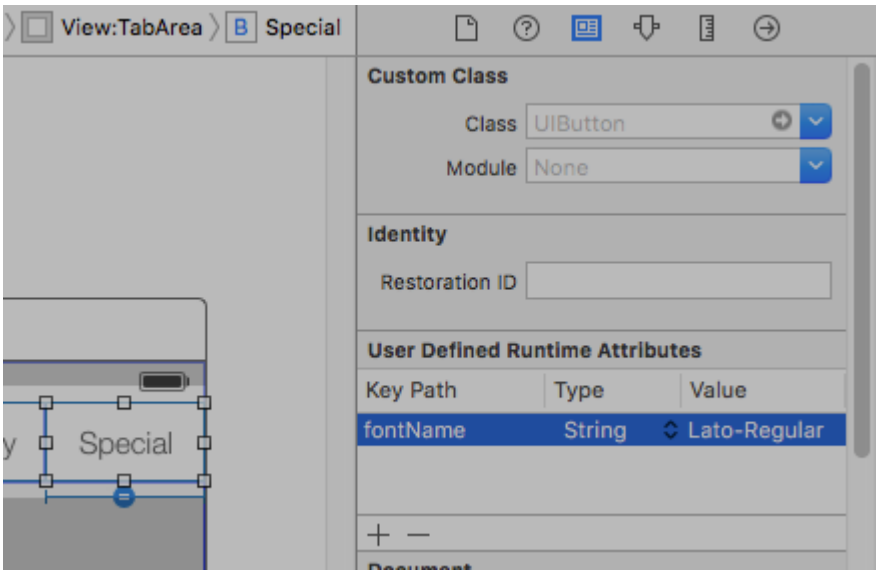
```
@interface UIButton (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end
```

3. **Getters and Setters:** Определите getters и seters для свойства fontName по каждой добавленной категории.

UIKit + IBExtensions.m

```
#import "UIKit+IBExtensions.h"

@implementation UILabel (IBExtensions)

- (NSString *)fontName {
    return self.font.fontName;
}

- (void)setFontName:(NSString *)fontName {
    self.font = [UIFont fontWithName:fontName size:self.font.pointSize];
}
@end

@implementation UITextField (IBExtensions)

- (NSString *)fontName {
    return self.font.fontName;
}

- (void)setFontName:(NSString *)fontName {
    self.font = [UIFont fontWithName:fontName size:self.font.pointSize];
}
@end

@implementation UIButton (IBExtensions)

- (NSString *)fontName {
    return self.titleLabel.font.fontName;
}

- (void)setFontName:(NSString *)fontName{
    self.titleLabel.font = [UIFont fontWithName:fontName size:self.titleLabel.font.pointSize];
}
@end
```

4. **Установка шрифта в раскадровке:** добавьте запись в User Defined Runtime Attributes с ***именем fontName*** как keyPath и ваше имя ***пользовательского шрифта*** как значение с типом как String, как показано.



При этом будет установлен ваш пользовательский шрифт во время работы приложения.

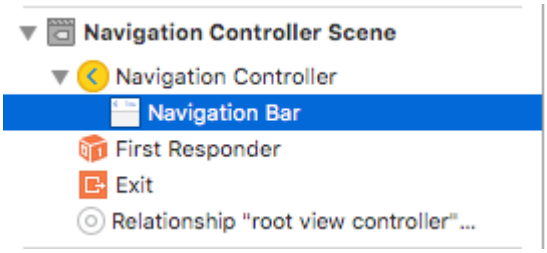
Заметки:

- Lato-Regular - это специальный шрифт, который я использовал.
- То же имя в файле ***.ttf,*** добавленном в комплект, должно использоваться без расширения в раскадровке.
- Размер шрифта будет таким же, как он определен в инспекторе атрибутов элемента интерфейса.

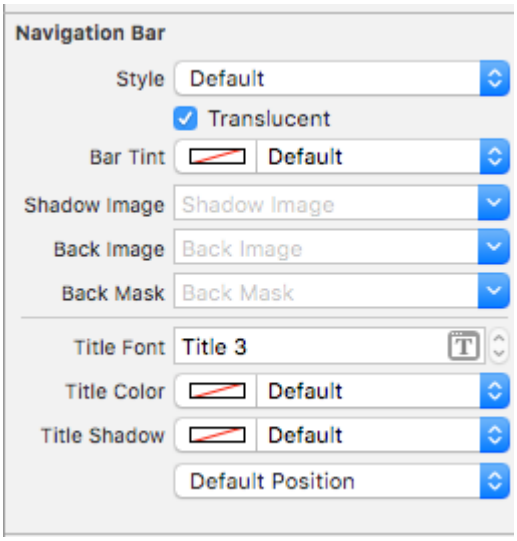
Применение пользовательских шрифтов для управления в раскадровке

В следующем примере показано, как применять пользовательские шрифты к панели навигации и включает исправления для некоторых причудливых действий, обнаруженных в Xcode. Также можно использовать пользовательские шрифты для **любых других UIControls,** таких как **UILabels** , **UIButtons** и т. Д., Используя инспектор атрибутов после добавления пользовательского шрифта в проект. Обратите внимание на внешние ссылки на рабочие образцы и видео внизу.

1. Выберите свою навигационную панель в своем навигационном контроллере



2. Изменение шрифта заголовка в инспекторе атрибутов

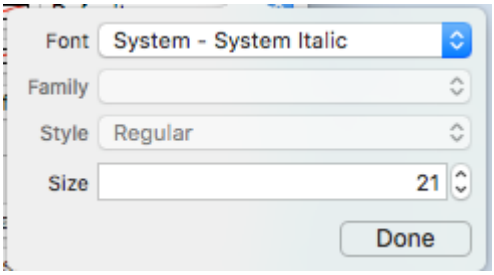


(Вам, скорее всего, нужно будет переключить оттенок панели для панели навигации, прежде чем Xcode возьмет новый шрифт)

Примечания (Предостережения)

Убедитесь, что это работает на Xcode 7.1.1+. (**См. Примеры ниже**)

1. Вам нужно переключить оттенок навигационной панели до того, как шрифт вступит в силу (кажется, что ошибка в Xcode, вы можете переключить его обратно на значение по умолчанию, и шрифт будет зависеть)
2. Если вы выберете системный шрифт ~ Обязательно убедитесь, что размер не равен 0.0 (иначе новый шрифт будет проигнорирован)



3. Кажется, что это работает без проблем, когда только один NavBar находится в иерархии представлений. Похоже, что вторичные NavBars в одном стеке игнорируются. (Обратите внимание: если вы покажете навигационную навигационную панель главного навигатора, все остальные пользовательские настройки NavBar будут проигнорированы).

Gotchas (deux)

Некоторые из них повторяются, что означает, что они, скорее всего, заслуживают внимания.

1. Иногда раскадровка xml становится коррумпированной. Это требует, чтобы вы

просмотрели структуру в Storyboard в качестве режима исходного кода (щелкните правой кнопкой мыши файл раскадровки> Открыть как ...)
2. В некоторых случаях тег navigationItem, связанный с пользовательским атрибутом runtime, был установлен как дочерний элемент xml тега представления вместо тега контроллера представления. Если это так, удалите его между метками для правильной работы.
3. Переключите NavBar Tint, чтобы гарантировать, что пользовательский шрифт используется.
4. Проверьте параметр размера шрифта, если не используется динамический стиль шрифта
5. Иерархия просмотра переопределит настройки. Кажется, что один шрифт за стек возможен.

Результат

образцы

- Видео, показывающее несколько шрифтов в расширенном проекте
- Простая загрузка источника
- Продвинутая загрузка проекта ~ Показывает несколько шрифтов NavBar и настраиваемый шрифт
- Видео, показывающее несколько шрифтов и пользовательские шрифты

Обработка пользовательских шрифтов

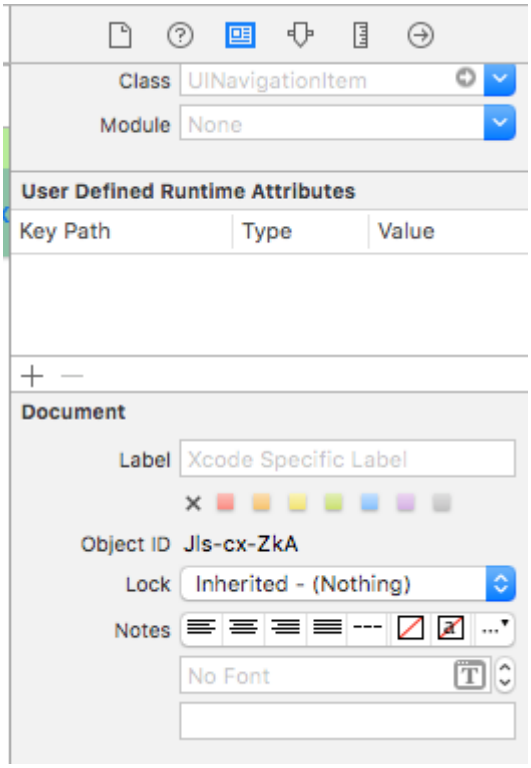
Примечание ~ Хороший контрольный список можно найти на веб-сайте Code With Chris, и вы можете увидеть образец проекта загрузки.

Если у вас есть собственный шрифт и вы хотите использовать его в своем раскадровке, тогда есть достойный набор ответов на следующий SO-вопрос . Один ответ определяет эти шаги.

1. Получите пользовательский файл шрифта (.ttf, .ttc)
2. Импортируйте файлы шрифтов в проект Xcode
3. В app-info.plist добавьте ключ с именем Fonts, предоставляемый приложением. Это тип массива, добавьте все имена файлов шрифтов в массив, обратите внимание: включая расширение файла.
4. В раскадровке на навигационной панели перейдите к Инспектору атрибутов, щелкните правой кнопкой мыши в области выбора шрифта. На всплывающей панели выберите «Шрифт» для «Пользовательский» и выберите «Семейство» вложенного имени шрифта.

Обычное обходное решение для шрифтов

Таким образом, Xcode, естественно, выглядит так, как будто он может обрабатывать пользовательские шрифты в UINavigationItem, но эта функция просто не обновляется должным образом (выбранный шрифт игнорируется).



Чтобы обойти это:

Один из способов - исправить использование раскадровки и добавить строку кода: сначала добавьте UIView (UIButton, UILabel или какой-либо другой подкласс UIView) в контроллер просмотра (а не элемент навигации ... Xcode в настоящее время не позволяет тот). После того, как вы добавите элемент управления, вы можете изменить шрифт в раскадровке и добавить ссылку в качестве выхода на ваш контроллер просмотра. Просто присвойте это представление UINavigationItem.titleView. Вы также можете указать текстовое имя в коде. Сообщенная ошибка (23600285).

```
@IBOutlet var customFontTitleView: UIButton!

//Sometime later...
self.navigationItem.titleView = customFontTitleView
```

Примечание. Этот пример получен из ответа, который я разместил на SO (здесь).

Прочитайте Пользовательские шрифты онлайн: https://riptutorial.com/ru/ios/topic/1504/пользовательские-шрифты

глава 172: Пользовательский UITextField

Вступление

Используя пользовательский UITextField, мы можем манипулировать поведением текстового поля!

Examples

Пользовательский UITextField для фильтрации входного текста

Вот пример пользовательского `UITextField` который принимает только числовой текст и отбрасывает все остальные.

ПРИМЕЧАНИЕ. Для iPhone это легко сделать с помощью клавиатуры типа Number, но для iPad нет клавиатуры с номерами

```
class NumberTextField: UITextField {

required init(coder aDecoder: NSCoder) {
    super.init(coder: aDecoder)
    registerForTextFieldNotifications()
}

override init(frame: CGRect) {
    super.init(frame: frame)
}

override func awakeFromNib() {
    super.awakeFromNib()
    keyboardType = .numberPad//useful for iPhone only
}

private func registerForTextFieldNotifications() {
    NotificationCenter.default.addObserver(self, selector:
#selector(NumberTextField.textDidChange), name: NSNotification.Name(rawValue:
"UITextFieldTextDidChangeNotification"), object: self)
}

deinit {
    NotificationCenter.default.removeObserver(self)
}

func textDidChange() {
    text = filteredText()
}
private func filteredText() -> String {
    let inverseSet = CharacterSet(charactersIn:"0123456789").inverted
    let components = text!.components(separatedBy: inverseSet)
    return components.joined(separator: "")
}
}
```

Итак, везде, где мы хотим, чтобы текстовое поле принимало только числа в качестве входного текста, мы можем использовать этот пользовательский UITextField

Пользовательский UITextField запрещает все действия, такие как копирование, вставка и т. Д.

Если мы хотим отключить все действия, такие как Copy, Paste, Replace, Select и т. Д. Из `UITextField` мы можем использовать следующее пользовательское текстовое поле:

```
class CustomTextField: UITextField {

var enableLongPressActions = false

required init(coder aDecoder: NSCoder) {
    super.init(coder: aDecoder)!
}

override init(frame: CGRect) {
    super.init(frame: frame)
}

override func canPerformAction(_ action: Selector, withSender sender: Any?) -> Bool {
    return enableLongPressActions
}
}
```

Используя свойство `enableLongPressActions` , мы можем активировать все действия в любое время позже, если это необходимо.

Прочитайте Пользовательский UITextField онлайн: https://riptutorial.com/ru/ios/topic/9997/пользовательский-uitextfield

глава 173: Преобразование HTML в строку NSAttributed и наоборот

Examples

Объективный код C для преобразования строки HTML в NSAttributedString и наоборот

Преобразование HTML в NSAttributedString Код: -

```
 //HTML String
 NSString *htmlString=[[NSString alloc]initWithFormat:@"<!DOCTYPE html><html><body><h1>My
First Heading</h1><p>My first paragraph.</p></body></html>"];
//Converting HTML string with UTF-8 encoding to NSAttributedString
 NSAttributedString *attributedString = [[NSAttributedString alloc]
                                         initWithData: [htmlString
dataUsingEncoding:NSUnicodeStringEncoding]
                                         options: @{ NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType }
                                         documentAttributes: nil
                                         error: nil ];
```

NSAttributedString to HTML Conversion: -

```
 //Dictionary to hold all the attributes of NSAttributed String
 NSDictionary *documentAttributes = @{NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType};
 //Saving the NSAttributedString with all its attributes as a NSData Entity
 NSData *htmlData = [attributedString dataFromRange:NSMakeRange(0, attributedString.length)
documentAttributes:documentAttributes error:NULL];
 //Convert the NSData into HTML String with UTF-8 Encoding
 NSString *htmlString = [[NSString alloc] initWithData:htmlData
encoding:NSUTF8StringEncoding];
```

Прочитайте Преобразование HTML в строку NSAttributed и наоборот онлайн:
https://riptutorial.com/ru/ios/topic/7225/преобразование-html-в-строку-nsattributed-и-наоборот

глава 174: Преобразование NSAttributedString в UIImage

Examples

NSAttributedString для UIImage Conversion

Objective-C

```
 NSMutableAttributedString *str = [[NSMutableAttributedString alloc] initWithString:@"Hello.
That is a test attributed string."];
 [str addAttribute:NSBackgroundColorAttributeName value:[UIColor yellowColor]
range:NSMakeRange(3,5)];
 [str addAttribute:NSForegroundColorAttributeName value:[UIColor greenColor]
range:NSMakeRange(10,7)];
 [str addAttribute:NSFontAttributeName value:[UIFont fontWithName:@"HelveticaNeue-Bold"
size:20.0] range:NSMakeRange(20, 10)];
 UIImage *customImage = [self imageFromAttributedString:str];
```

Функция `imageFromAttributedString` имеет следующие значения:

```
- (UIImage *)imageFromAttributedString:(NSAttributedString *)text
{
    UIGraphicsBeginImageContextWithOptions(text.size, NO, 0.0);

    // draw in context
    [text drawAtPoint:CGPointMake(0.0, 0.0)];

    // transfer image
    UIImage *image = [UIGraphicsGetImageFromCurrentImageContext()
imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
    UIGraphicsEndImageContext();

    return image;
}
```

Прочитайте Преобразование NSAttributedString в UIImage онлайн:
https://riptutorial.com/ru/ios/topic/7242/преобразование-nsattributedstring-в-uiimage

глава 175: Проверка версии iOS

Examples

iOS 8 и более поздние версии

Swift 3:

```
let minimumVersion = OperatingSystemVersion(majorVersion: 8, minorVersion: 1, patchVersion: 2)
if ProcessInfo().isOperatingSystemAtLeast(minimumVersion) {
    //current version is >= (8.1.2)
} else {
    //current version is < (8.1.2)
}
```

Сравнить версии

```
let minimumVersionString = "3.1.3"
let versionComparison = UIDevice.current.systemVersion.compare(minimumVersionString, options:
.numeric)
switch versionComparison {
    case .orderedSame, .orderedDescending:
        //current version is >= (3.1.3)
        break
    case .orderedAscending:
        //current version is < (3.1.3)
        fallthrough
    default:
        break;
}
```

Objective-C

```
NSString *version = @"3.1.3";
NSString *currentVersion = @"3.1.1";
NSComparisonResult result = [currentVersion compare:version options:NSNumericSearch];
switch(result){
  case: NSOrderedAscending:
        //less than the current version
  break;
  case: NSOrderedDescending:
  case: NSOrderedSame:
       // equal or greater than the current version
  break;
}
```

Swift 2.0 и более поздние версии

```
if #available(iOS 9, *) {
  // iOS 9
} else {
  // iOS 8 or earlier
}
```

Версия устройства iOS

Это даст текущую версию системы.

Objective-C

```
NSString *version = [[UIDevice currentDevice] systemVersion]
```

стриж

```
let version = UIDevice.currentDevice().systemVersion
```

Swift 3

```
let version = UIDevice.current.systemVersion
```

Прочитайте Проверка версии iOS онлайн: https://riptutorial.com/ru/ios/topic/2194/проверка-версии-ios

глава 176: Проверка сетевого подключения

замечания

Исходный код для `Reachability.h` и `Reachability.m` можно найти на сайте документации разработчика Apple.

Предостережения

В отличие от других платформ, Apple еще не предоставила стандартный набор API для определения состояния сети устройства iOS и предлагает только эти примеры кода, приведенные выше. Исходный файл изменяется со временем, но после импорта в проект приложения они редко обновляются разработчиками.

По этой причине большинство разработчиков приложений склонны использовать одну из многочисленных поддерживаемых библиотек Github / Cocoapod для достижения достижимости.

Apple также рекомендует, чтобы для запросов, сделанных по просьбе пользователя, вы всегда пытались подключиться *первым* , прежде чем использовать Reachability / SCNetworkReachability, чтобы диагностировать отказ или дождаться возврата соединения .

Examples

Создание слушателя по достижимости

Класс Apple Reachability периодически проверяет статус сети и предупреждает наблюдателей об изменениях.

```
Reachability *internetReachability = [Reachability reachabilityForInternetConnection];
[internetReachability startNotifier];
```

Добавить наблюдателя в сетевые изменения

`Reachability` использует сообщения `NSNotification` для предупреждения наблюдателей при изменении состояния сети. Ваш класс должен стать наблюдателем.

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(reachabilityChanged:) name:kReachabilityChangedNotification object:nil];
```

В другом месте вашего класса реализуйте подпись метода

```
- (void) reachabilityChanged:(NSNotification *)note {
    //code which reacts to network changes
}
```

Предупреждение, когда сеть становится недоступной

```
- (void)reachabilityChanged:(NSNotification *)note {
    Reachability* reachability = [note object];
    NetworkStatus netStatus = [reachability currentReachabilityStatus];

    if (netStatus == NotReachable) {
        NSLog(@"Network unavailable");
    }
}
```

Предупреждение, когда соединение становится WIFI или сотовой сетью

```
- (void)reachabilityChanged:(NSNotification *)note {
    Reachability* reachability = [note object];
    NetworkStatus netStatus = [reachability currentReachabilityStatus];

    switch (netStatus) {
        case NotReachable:
            NSLog(@"Network unavailable");
            break;
        case ReachableViaWWAN:
            NSLog(@"Network is cellular");
            break;
        case ReachableViaWiFi:
            NSLog(@"Network is WIFI");
            break;
    }
}
```

Проверьте, подключен ли к сети

стриж

```
import SystemConfiguration

/// Class helps to code reuse in handling internet network connections.
class NetworkHelper {

    /**
     Verify if the device is connected to internet network.
     - returns:          true if is connected to any internet network, false if is not
     connected to any internet network.
     */
   class func isConnectedToNetwork() -> Bool {
       var zeroAddress = sockaddr_in()

       zeroAddress.sin_len = UInt8(sizeofValue(zeroAddress))
```

```
        zeroAddress.sin_family = sa_family_t(AF_INET)

        let defaultRouteReachability = withUnsafePointer(&zeroAddress) {
            SCNetworkReachabilityCreateWithAddress(nil, UnsafePointer($0))
        }

        var flags = SCNetworkReachabilityFlags()

        if !SCNetworkReachabilityGetFlags(defaultRouteReachability!, &flags) {
            return false
        }

        let isReachable = (flags.rawValue & UInt32(kSCNetworkFlagsReachable)) != 0
        let needsConnection = (flags.rawValue & UInt32(kSCNetworkFlagsConnectionRequired)) != 0

       return (isReachable && !needsConnection)
    }
}



if NetworkHelper.isConnectedToNetwork() {
    // Is connected to network
}
```

Objective-C:

мы можем проверить сетевое подключение в нескольких строках кода как:

```
-(BOOL)isConntectedToNetwork
{
    Reachability *networkReachability = [Reachability reachabilityForInternetConnection];
    NetworkStatus networkStatus = [networkReachability currentReachabilityStatus];
    if (networkStatus == NotReachable)
    {
        NSLog(@"There IS NO internet connection");
        return false;
    } else
    {
        NSLog(@"There IS internet connection");
        return true;
    }
}
```

Прочитайте Проверка сетевого подключения онлайн: https://riptutorial.com/ru/ios/topic/704/проверка-сетевого-подключения

глава 177: Профиль с инструментами

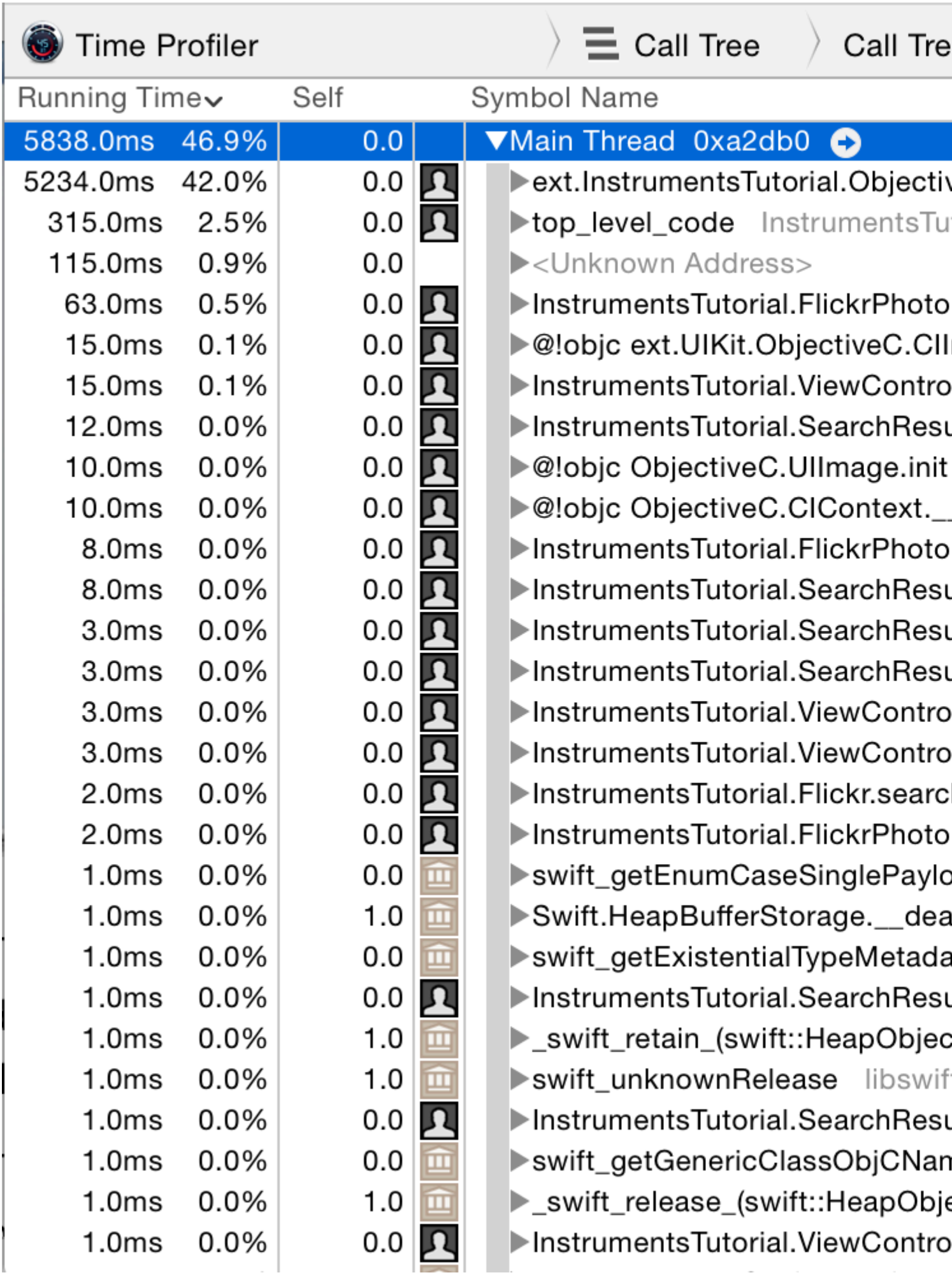
Вступление

Xcode включает приложение настройки производительности под названием «Инструменты», которое можно использовать для профилирования вашего приложения, используя всевозможные различные показатели. У них есть инструменты для проверки использования ЦП, использования памяти, утечек, активности файлов / сетей и использования энергии, просто чтобы назвать несколько. Очень легко начать профилирование вашего приложения с Xcode, но иногда не так легко понять, что вы видите, когда оно профилируется, что отталкивает некоторых разработчиков от возможности использовать этот инструмент в полном объеме.

Examples

Профайлер времени

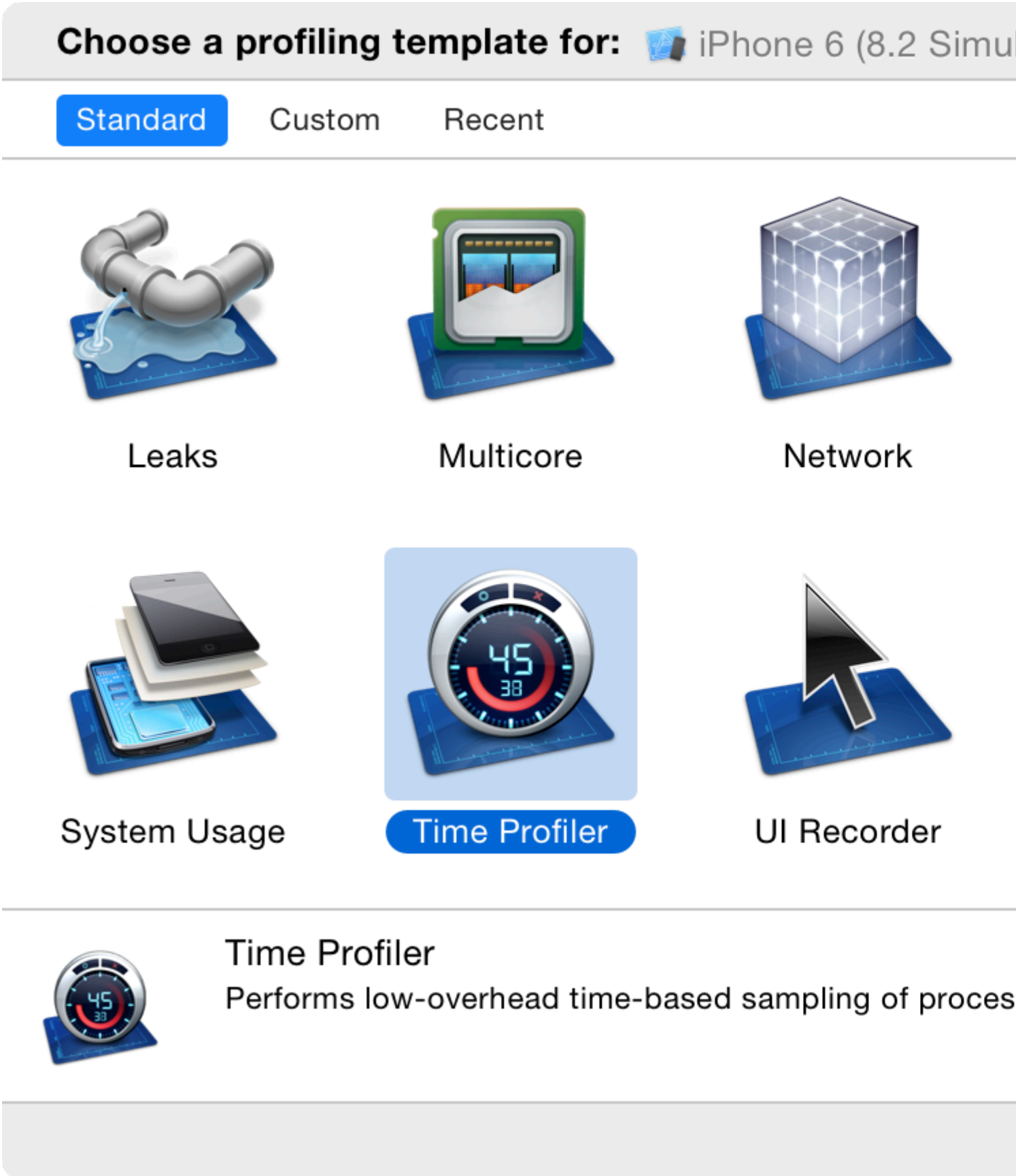
Первым инструментом, на который вы будете смотреть, является `Time Profiler` . Через измеренные интервалы инструменты прекратят выполнение программы и проведут трассировку стека на каждом запущенном потоке. Подумайте об этом, нажав кнопку паузы в отладчике Xcode. Вот предварительный просмотр Time Profiler:



На этом экране отображается `Call Tree . Call Tree`

показывает количество времени, затрачиваемого на выполнение различными способами в приложении. Каждая строка отличается от метода выполнения программы. Время, затрачиваемое в каждом методе, может быть определено из количества остановок профайлера в каждом методе. Например, если **100** выборок выполняются с **интервалом** в **1 миллисекунду** , а определенный метод находится в верхней части стека в 10 образцах, то вы можете сделать вывод, что примерно **10%** общего времени выполнения - **10 миллисекунд** - было потрачено в этом методе. Это довольно грубое приближение, но оно работает!

`Xcode's` строке меню `Xcode's` выберите `Product\Profile` или `press ⌘I` . Это создаст приложение и запустит Инструменты. Вам будет предложено окно выбора, которое выглядит так:



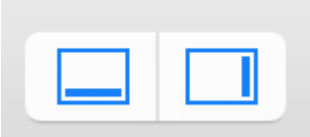
Это все разные шаблоны, которые поставляются с инструментами.

Выберите инструмент `Time Profiler` и нажмите «Выбрать». Это откроет новый документ «Инструменты». Нажмите красную **кнопку записи** в левом верхнем углу, чтобы начать запись и запустить приложение. Вам может быть предложено ввести пароль для

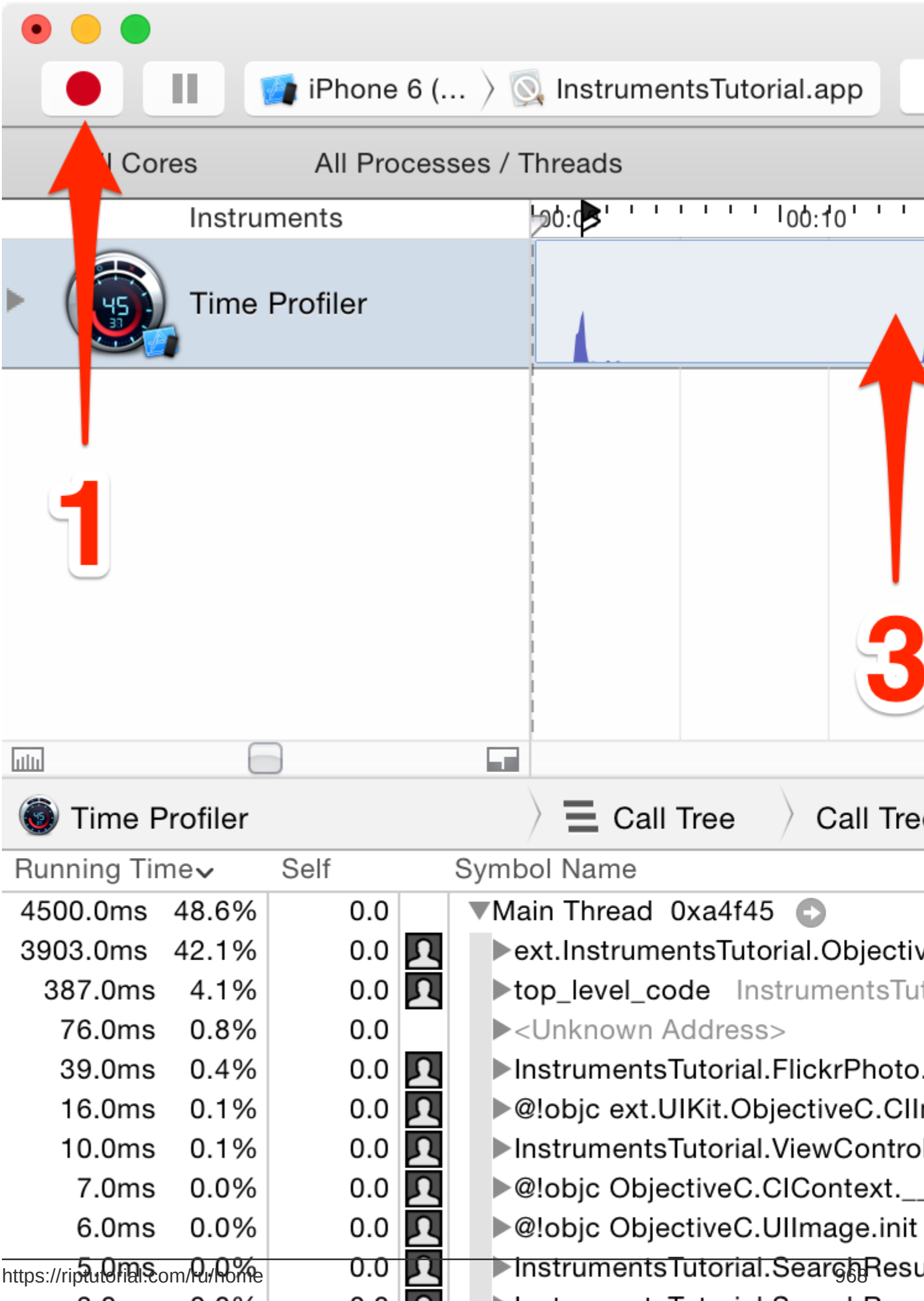
авторизации **инструментов** для анализа других процессов - не бойтесь, это безопасно здесь! В **окне «Инструменты»** вы можете увидеть отсчет времени и небольшую стрелку, перемещающуюся слева направо над **графиком** в центре экрана. Это означает, что приложение запущено.

Теперь начните использовать приложение. Найдите некоторые изображения и разверните их в один или несколько результатов поиска. Вероятно, вы заметили, что вхождение в результат поиска утомительно медленное, и прокрутка списка результатов поиска также невероятно раздражает - это ужасно неуклюжее приложение!

Ну, тебе повезло, потому что ты собираешься это исправить! Тем не менее, вы сначала получите быстрый контроль над тем, что вы смотрите на **инструменты** . Во-первых, убедитесь, что в селекторе представлений в правой части панели инструментов выбраны оба варианта:



Это обеспечит открытость всех панелей. Изучите скриншот ниже и объяснение каждого раздела под ним:


iPhone 6 (...
InstrumentsTutorial.app
All Processes / Threads
Instruments
00:10
Time Profiler
1
3
Time Profiler
Call Tree
Running Time
Self
Symbol Name
4500.0ms 48.6% 0.0 Main Thread 0xa4f45
3903.0ms 42.1% 0.0 ext.InstrumentsTutorial.Objectiv
387.0ms 4.1% 0.0 top_level_code InstrumentsTut
76.0ms 0.8% 0.0 <Unknown Address>
39.0ms 0.4% 0.0 InstrumentsTutorial.FlickrPhoto.
16.0ms 0.1% 0.0 @!objc ext.UIKit.ObjectiveC.CIIr
10.0ms 0.1% 0.0 InstrumentsTutorial.ViewControl
7.0ms 0.0% 0.0 @!objc ObjectiveC.CIContext.__
6.0ms 0.0% 0.0 @!objc ObjectiveC.UIImage.init

Красная кнопка «запись» остановится и запустит приложение, которое сейчас профилируется при нажатии (он переключается между значком записи и остановки). Кнопка паузы делает именно то, что вы ожидаете, и приостанавливает текущее выполнение приложения.

2. Это таймер запуска. Таймер подсчитывает, сколько времени работает профилирование приложения, и сколько раз оно было запущено. Если вы остановите и затем перезапустите приложение, используя элементы управления записью, это запустит новый прогон, и на дисплее появится «Run 2 of 2».

3. Это называется дорожкой. В случае выбранного шаблона Time Profiler существует только один инструмент, поэтому есть только один трек. Вы узнаете больше о специфике графика, показанного здесь позже в учебнике.

4. Это панель деталей. Он показывает основную информацию о конкретном инструменте, который вы используете. В этом случае он показывает методы, которые являются «самыми горячими», то есть теми, которые использовали наибольшее время процессора. Если вы нажмете на верхнюю панель, которая говорит «Дерево вызовов» (левая рука) и выберите «Пример списка», вам будет предоставлен другой вид данных. В этом представлении отображается каждый отдельный образец. Нажмите несколько выборок, и вы увидите, что полученная трассировка стека отображается в инспекторе расширенной детали.

5. Это панель инспекторов. Существует три инспектора: настройки записи, настройки дисплея и расширенная деталь. Вскоре вы узнаете больше о некоторых из этих вариантов.

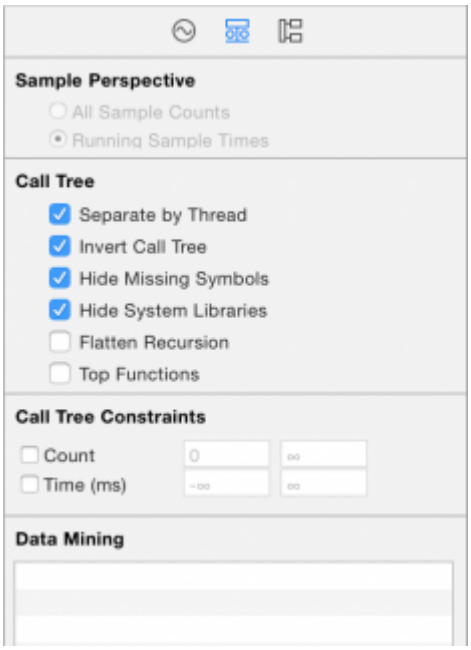
Глубокое сверление

Выполните поиск изображений и сверьтесь с результатами. Мне лично нравится искать «собаку», но выбирайте все, что пожелаете, - вы можете быть одним из тех людей кошки!

Теперь прокрутите список вверх и вниз по списку несколько раз, чтобы у вас было хорошее количество данных в `Time Profiler` . Вы должны заметить числа в середине изменения экрана и заполнения **графа** ; это говорит о том, что используются **циклы процессора** .

Вы действительно не ожидали бы, что какой-либо пользовательский интерфейс будет таким же неуклюжим, как если бы этот `table view` не был готов к отправке, пока он не прокрутится, как масло! Чтобы помочь выявить проблему, вам нужно установить некоторые параметры.

С правой стороны выберите **Инспектор настроек дисплея** `(or press ⌘+2)` . В **инспекторе** в разделе `Call Tree` выберите «Разделить по **потоку»** , «Инвертировать `Call Tree` , «Скрыть отсутствующие символы» и «Скрыть системные библиотеки». Это будет выглядеть так:



Вот что каждый параметр делает с данными, отображаемыми в таблице слева:

Отдельно по потоку **:** каждый поток следует рассматривать отдельно. Это позволяет понять, какие потоки отвечают за наибольшее количество использования **ЦП** .

Инвертировать дерево вызовов. С помощью этой опции `stack trace` рассматривается сверху вниз. Обычно это то, что вы хотите, так как вы хотите увидеть самые глубокие методы, в которых **процессор** проводит свое время.

Скрыть отсутствующие символы: если файл `dSYM` не найден для вашего приложения или `system framework` , вместо того, чтобы видеть имена (символы) в таблице, вы увидите только шестнадцатеричные значения, соответствующие адресам внутри двоичного файла. Если эта опция выбрана, отображаются только полностью разрешенные символы, а неразрешенные **шестнадцатеричные** значения скрыты. Это помогает декларировать представленные данные.

Скрыть системные библиотеки. Когда этот параметр выбран, отображаются только символы из вашего собственного приложения. Часто бывает полезно выбрать эту опцию, так как обычно вам остается только заботиться о том, где **процессор** проводит время в своем собственном коде - вы не можете многое сделать о том, сколько **CPU** используются `system libraries` !

Flatten Recursion: эта опция рассматривает **рекурсивные** функции (те, которые называют себя) как одна запись в каждой `stack trace` , а не несколько.

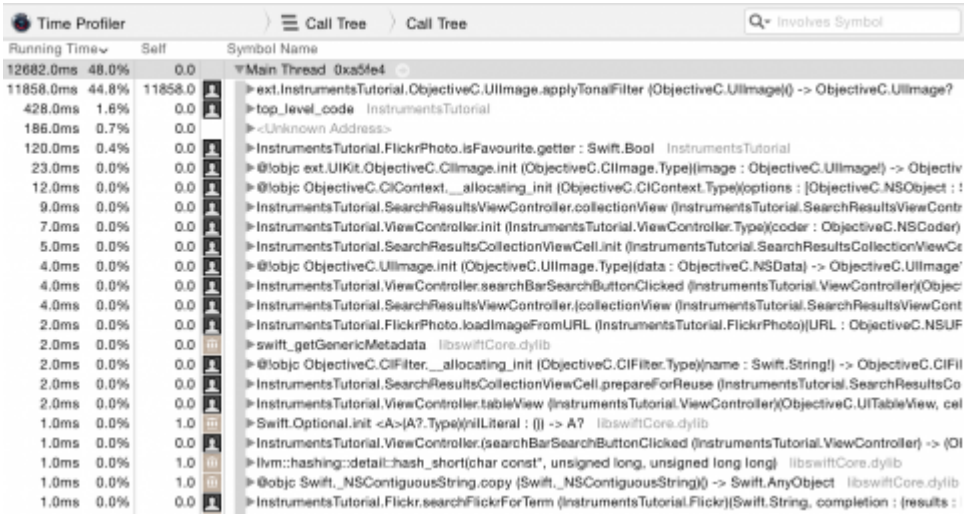
Верхние функции. Включение этого означает, что `Instruments` учитывают общее время, затрачиваемое на функцию, как сумму времени непосредственно внутри этой функции, а также время, затрачиваемое на функции, вызываемые этой функцией.

Поэтому, если функция A вызывает B, тогда время A сообщается как время, проведенное в

A PLUS, время, проведенное в B. Это может быть действительно полезно, поскольку оно позволяет вам выбирать наибольшее значение времени при каждом сходе в стек вызовов, обнуление в ваших наиболее трудоемких методах.

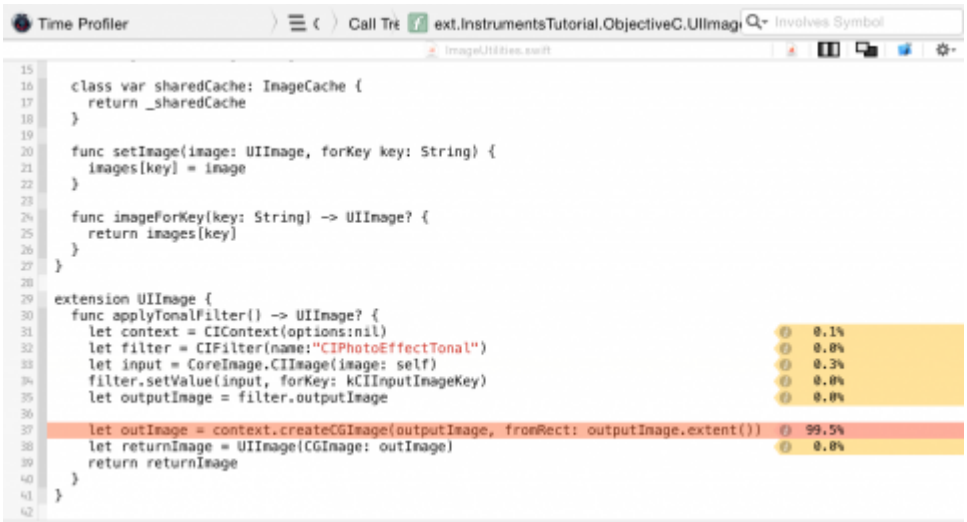
Если вы используете приложение `Objective-C` , есть опция Show **Obj-C Only** : если это выбрано, то отображаются только методы `Objective-C` , а не любые функции `C` или `C++` . В вашей программе их нет, но если вы смотрите на приложение `OpenGL` , у него может быть, например, `C++` .

Хотя некоторые значения могут несколько отличаться, порядок записей должен быть похож на таблицу ниже, если вы включили следующие опции:



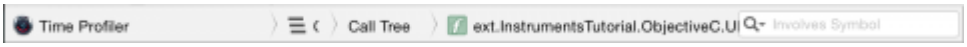
Ну, это, конечно, выглядит не слишком хорошо. Подавляющее большинство времени проводится в методе, который применяет фильтр «тональный» к фотографиям миниатюр. Это не должно быть для вас шоком, так как загрузка и прокрутка таблицы были самыми неуклюжими частями пользовательского интерфейса, и именно поэтому ячейки таблицы постоянно обновляются.

Чтобы узнать больше о том, что происходит внутри этого метода, дважды щелкните его строку в таблице. Это приведет к следующему представлению:

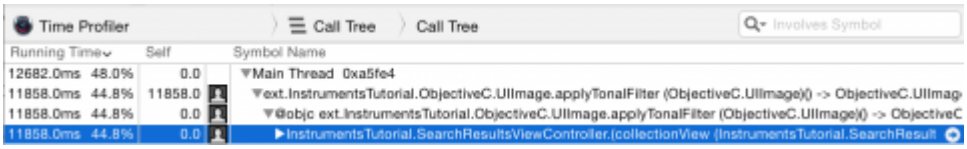


Ну, это интересно, не так ли! `applyTonalFilter()` - это метод, добавленный в `UIImage` в расширение, и почти **100** % времени, затраченного на него, тратится на создание CGImage после применения фильтра изображения.

Существует не так много, что можно сделать, чтобы ускорить это: создание изображения - довольно интенсивный процесс и занимает столько времени, сколько потребуется. Давайте попробуем отступить и посмотреть, откуда `applyTonalFilter()` . **Нажмите «** `Call Tree` на дорожке с сухарями вверху окна кода, чтобы вернуться к предыдущему экрану:



Теперь щелкните маленькую стрелку слева от строки applyTonalFilter в верхней части таблицы. Это откроет Дерево вызовов, чтобы отобразить вызывающего абонента applyTonalFilter. Возможно, вам придется развернуть следующую строку; при профилировании Swift иногда иногда появляются повторяющиеся строки в Дереве вызовов с префиксом @objc. Вы заинтересованы в первой строке с префиксом целевого имени вашего приложения (InstructionsTutorial):

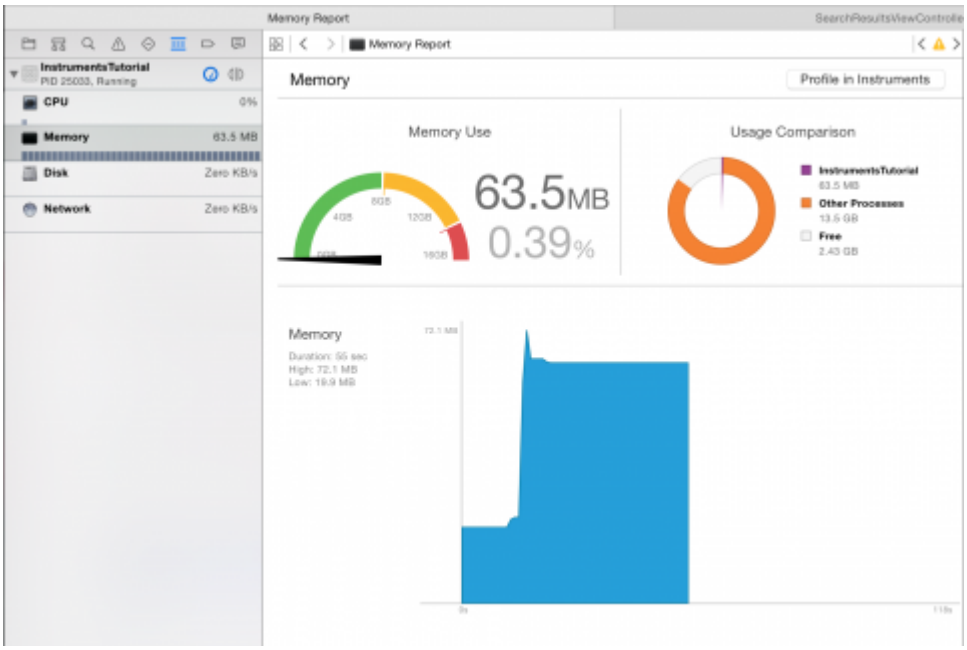


В этом случае эта строка относится к `cellForItemAtIndexPath` коллекции результатов. Дважды щелкните строку, чтобы увидеть связанный код из проекта.

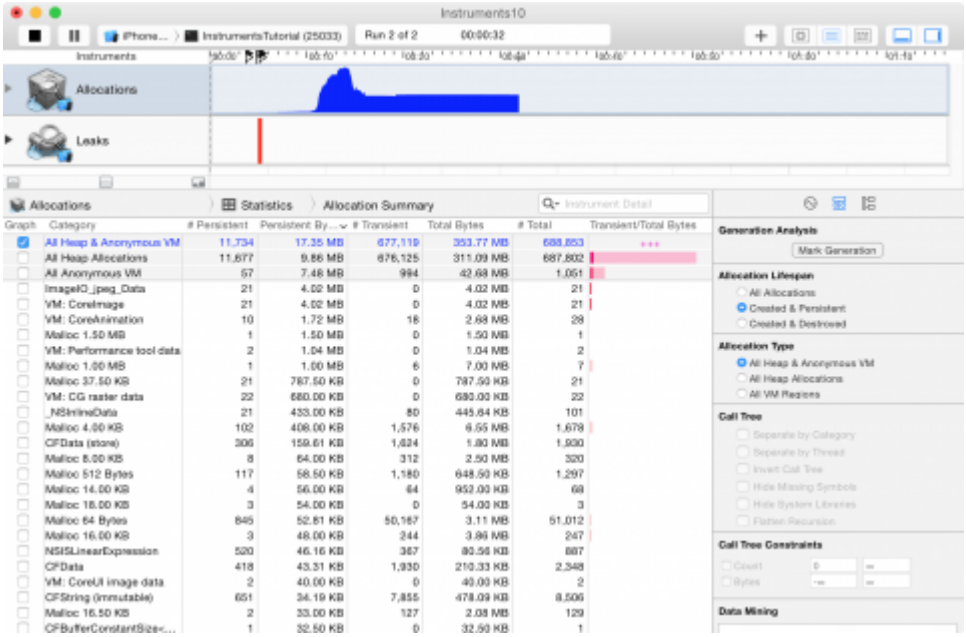
Теперь вы можете понять, в чем проблема. Метод применения тонального фильтра занимает много времени, и он вызывается непосредственно из cellForItemAtIndexPath, который будет блокировать `main thread` (и, следовательно, весь пользовательский интерфейс) каждый раз, когда он запрашивает отфильтрованное изображение.

ассигнования

Есть подробная информация обо всех создаваемых **объектах** и памяти, которая их поддерживает; он также показывает, что вы `retain counts` каждого объекта. Чтобы снова начать работу с новым `instruments profile` , закройте приложение «Инструменты». На этот раз создайте и запустите приложение и откройте Debug Navigator в области Navigators. Затем нажмите « **Память»,** чтобы отобразить графики использования памяти в главном окне:



Эти графики полезны для быстрого представления о том, как работает ваше приложение. Но вам понадобится немного больше энергии. Нажмите кнопку « `Profile in Instruments` , а затем «Передача», чтобы включить эту сессию в « **Инструменты**» . Инструмент **Allocations** запустится автоматически.



На этот раз вы заметите два трека. Один называется Allocations, а один называется Leaks. Дорожка Allocations будет подробно обсуждена позже; дорожка утечек, как правило, более полезна в Objective-C и не будет рассмотрена в этом уроке. Итак, какую ошибку вы собираетесь отслеживать дальше? В проекте есть что-то скрытое, что вы, вероятно, не знаете, есть. Вероятно, вы слышали о утечке памяти. Но то, что вы, возможно, не знаете, состоит в том, что на самом деле существуют два типа утечек:

Истинные утечки памяти - это то, где объект больше не ссылается ни на что, но все еще выделяется - это означает, что память никогда не может быть повторно использована.

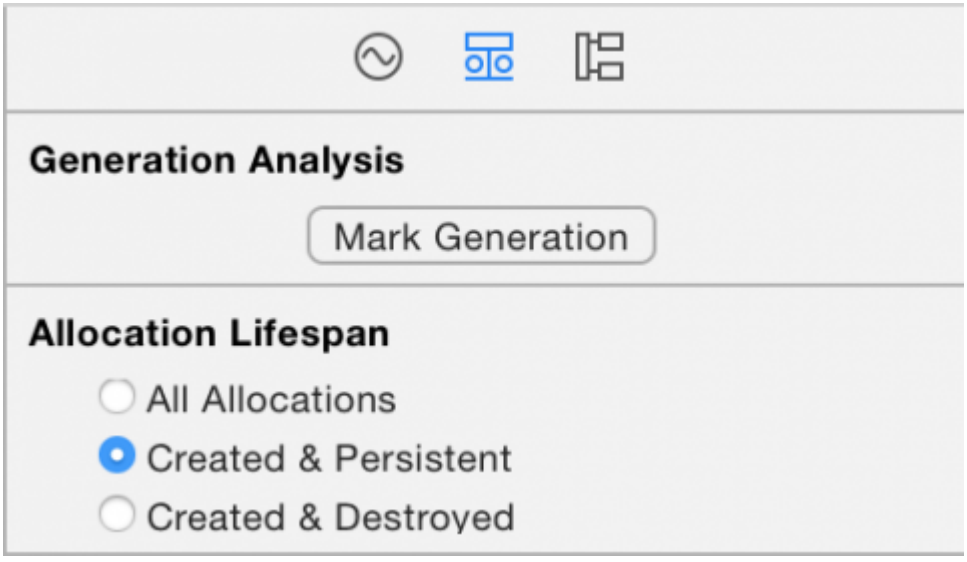
Даже когда Swift и `ARC` помогают управлять памятью, наиболее распространенным видом утечки памяти является `retain cycle or strong reference cycle` . Это когда два объекта содержат сильные ссылки друг на друга, так что каждый объект удерживает другой от освобождения. Это означает, что их память никогда не выпускается!

Неограниченный рост памяти - это то место, где память продолжает выделяться и никогда не предоставляется шанс быть **освобожденной** . Если это будет продолжаться вечно, то в какой-то момент `system's memory` будет заполнена, и у вас будет большая проблема с памятью на ваших руках. В iOS это означает, что приложение будет убито системой.

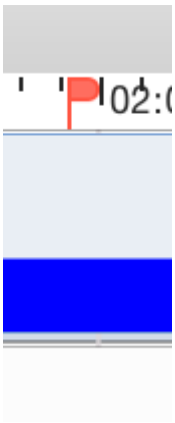
С помощью **инструмента** Allocations, запущенного в приложении, выполните пять различных поисков в приложении, но пока не сверните их в результаты. Убедитесь, что результаты поиска имеют некоторые результаты! Теперь пусть приложение немного успокоится, подождав несколько секунд.

Вы должны были заметить, что **график** на дорожке Allocations растет. Это говорит вам, что память распределяется. Именно эта функция поможет вам найти `unbounded memory growth` .

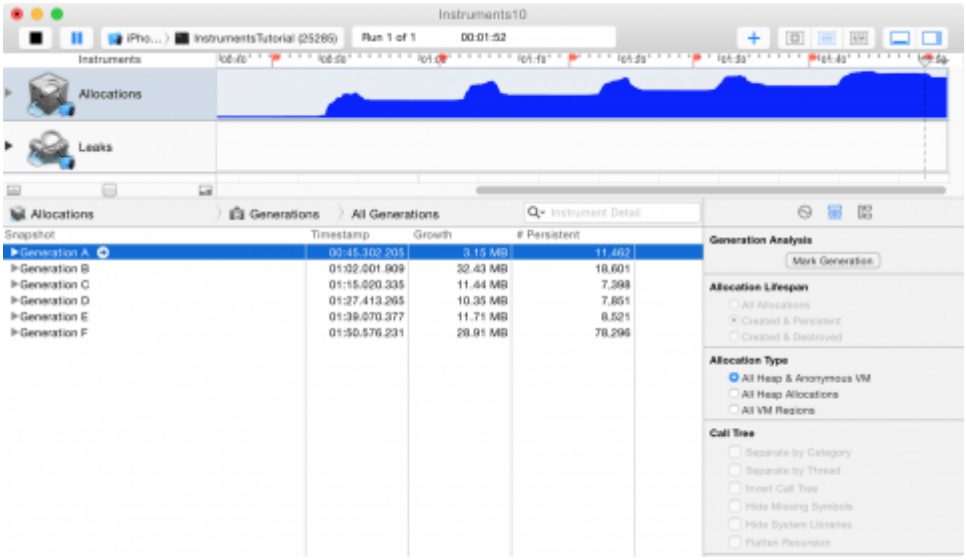
То, что вы собираетесь выполнять, - это `generation analysis` . Для этого нажмите кнопку Mark Generation. Вы найдете кнопку в верхней части инспектора настроек дисплея:



Нажмите его, и в треке появится красный флаг, например:



Цель `generation analysis` состоит в том, чтобы выполнить действие несколько раз и посмотреть, растет ли память `unbounded fashion` . **Просверлите** поиск, подождите несколько секунд, чтобы изображения загрузились, а затем вернитесь на главную страницу. Затем отметьте поколение снова. Повторите это для разных поисков. После **сверления** нескольких поисковых систем **инструменты** будут выглядеть так:



На данный момент вы должны быть подозрительными. Обратите внимание на то, как синий график растет с каждым поиском, который вы **пробуете** . Ну, это конечно не хорошо. Но подождите, как насчет `memory warnings?` Вы знаете об этом, верно? `Memory warnings` - это способ iOS сообщать приложениям, что в отделении памяти все тесно, и вам нужно очистить память.

Возможно, этот рост связан не только с вашим приложением; это может быть что-то в глубине `UIKit` которое держится за память. Дайте системным фреймворкам и вашему приложению возможность очистить **память** перед тем, как указывать палец на один.

Имитируйте `memory warning` о `memory warning` , выбрав «Панель `Instrument\Simulate Memory Warning` в меню инструментов» или « `Hardware\Simulate Memory Warning` в строке меню `simulator's` . Вы заметите, что использование памяти немного уменьшилось, или, возможно, совсем нет. Конечно, не вернемся туда, где это должно быть. Таким образом, все еще существует **неограниченный рост памяти** .

Причиной маркировки поколения после каждой итерации сверления в поиске является то, что вы можете видеть, какая **память** была распределена между каждым поколением. Взгляните на панель подробностей, и вы увидите кучу поколений.

Прочитайте Профиль с инструментами онлайн: https://riptutorial.com/ru/ios/topic/9629/профиль-с-инструментами

глава 178: Процесс подачи заявки

Вступление

В этом руководстве описаны все необходимые шаги, необходимые для загрузки приложения iOS в App Store.

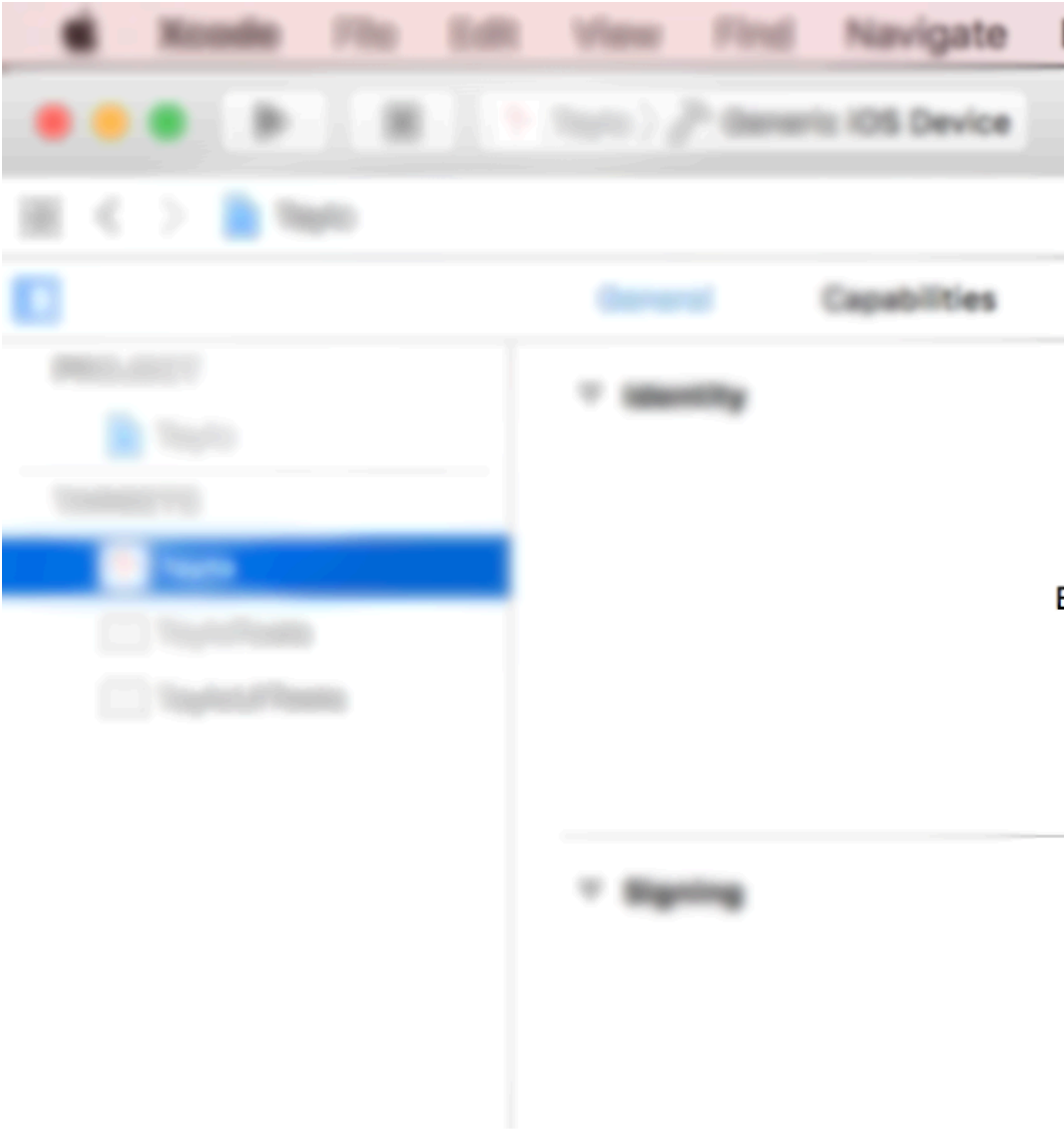
Examples

Настройка профилей настройки

В предыдущих версиях настройка профилей подготовки выполнялась вручную. Вы создаете профиль распространения, загружаете его, а затем распространяете свое приложение. Это должно было быть сделано для каждой машины разработки, которая занимала много времени. Однако в большинстве случаев в настоящее время Xcode 8 выполнит большую часть этой работы для вас. Убедитесь, что вы вошли в систему с учетной записью, которая будет использоваться для распространения приложения, а затем просто выберите «Автоматическое управление подписанием кода» в «Цели» -> «Общие».

Архивный код

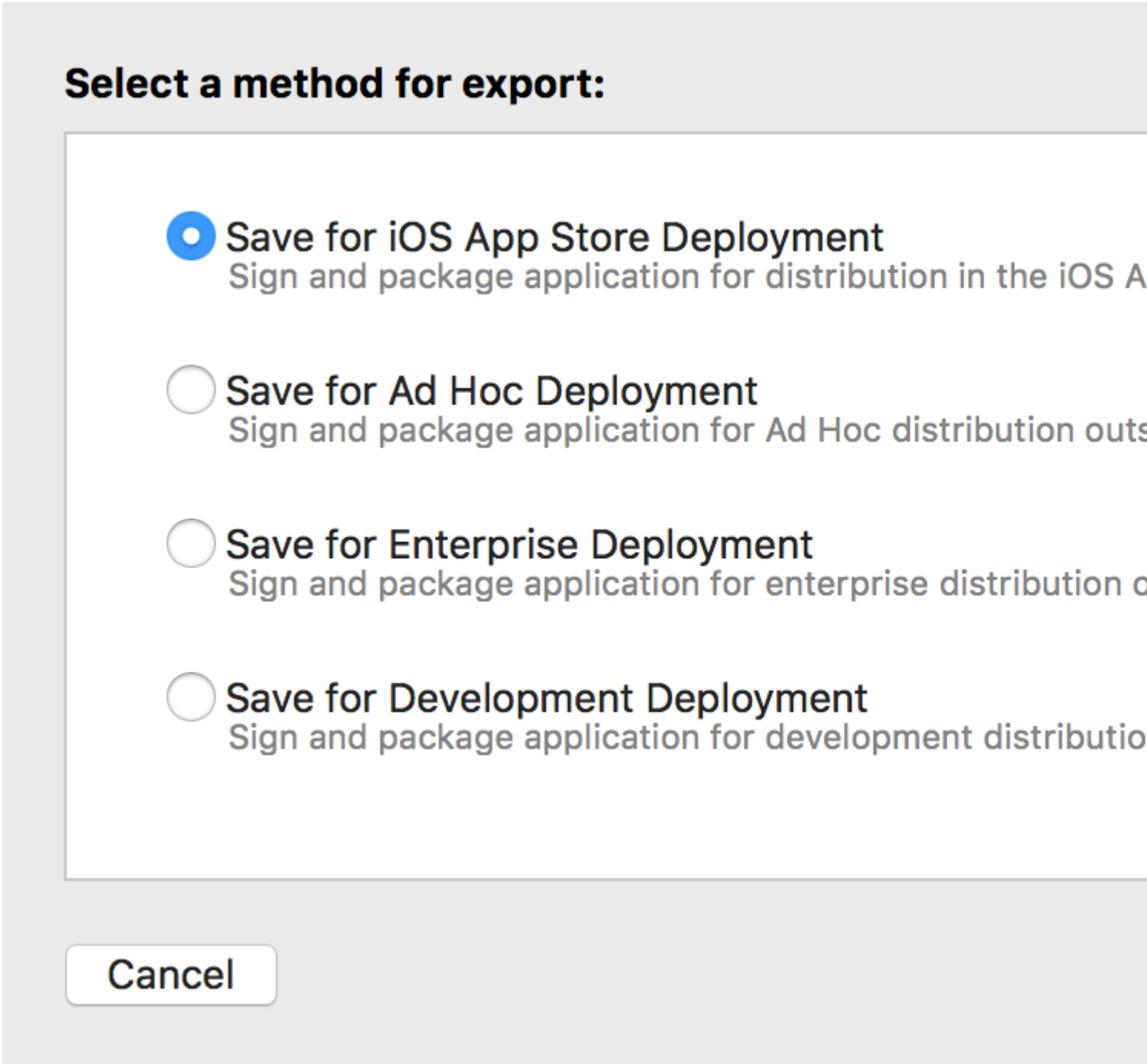
После того как все профили обеспечения настроены, следующим шагом в процессе подачи приложения является архивирование вашего кода. Из раскрывающегося списка устройств и симуляторов выберите опцию «Generic iOS device». Затем в меню «Продукт» выберите опцию «Архив».



В случае, если представление представляет собой обновление существующего приложения в хранилище, убедитесь, что номер сборки выше текущего и номер версии отличается. Например, текущее приложение имеет номер сборки 30 и версию 1.0. Следующее обновление должно содержать хотя бы номер 31 и версию 1.0.1. В большинстве случаев вы должны добавить третью цифру в свою версию в случае некоторых срочных исправлений ошибок или небольших патчей, вторая десятичная запятая в основном зарезервирована для обновлений функций, в то время как первая десятичная цифра увеличивается в случае значительного обновления приложения.

Экспорт файла IPA

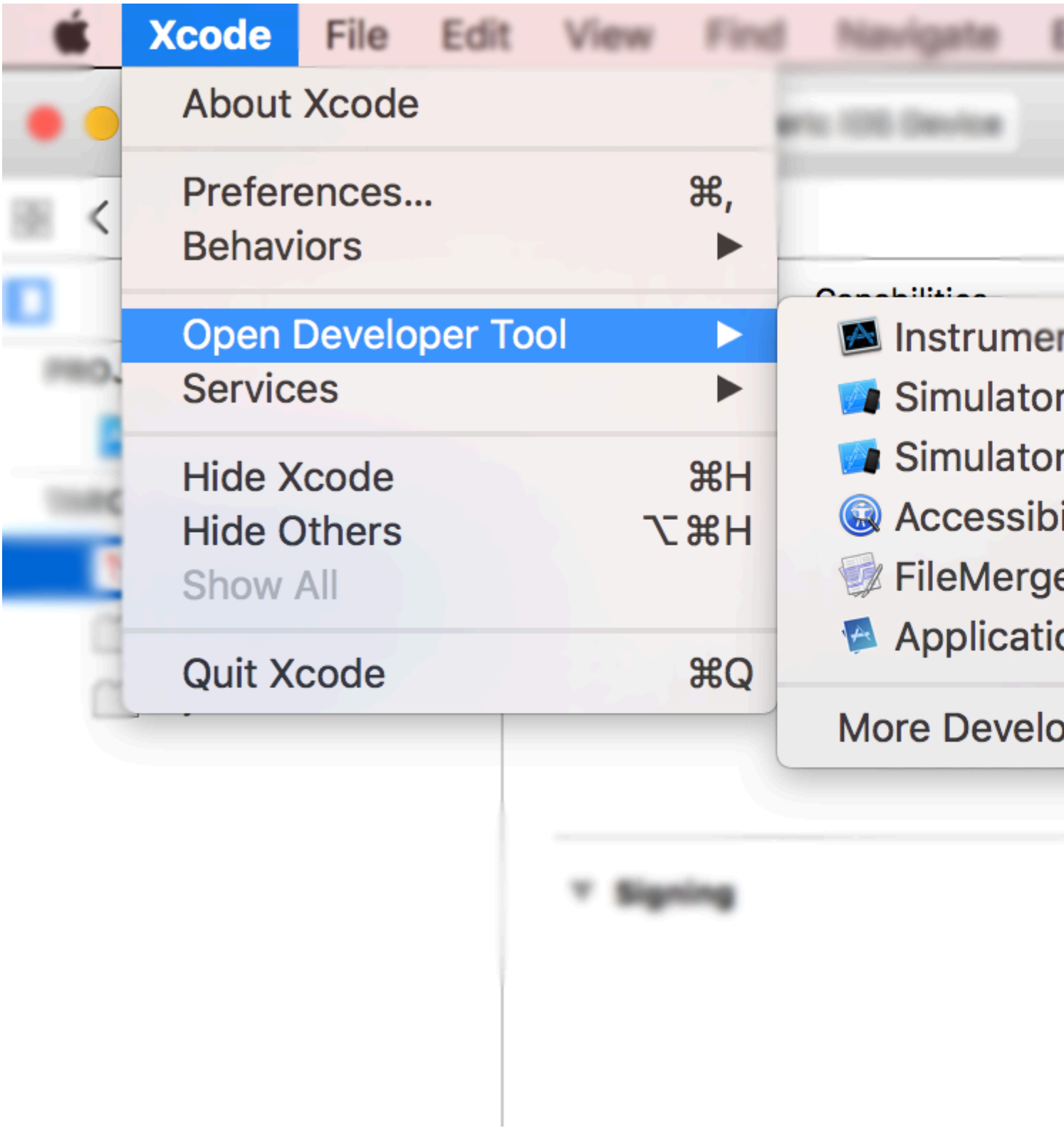
Как только это будет сделано, вы сможете найти свой архив в организаторе Xcode. Здесь все ваши предыдущие версии и архивные сборки сохраняются и организованы, если вы их не удаляете. Вы сразу заметите большую синюю кнопку с надписью «Загрузить в App Store ...», однако в 9/10 случаях это не будет работать по различным причинам (главным образом, ошибки Xcode). Обходной путь заключается в том, чтобы экспортировать ваш архив и загрузить его с помощью другого инструмента Xcode под названием Application Loader. Однако, поскольку загрузчик приложений загружает файлы IPA в App Store, архив необходимо экспортировать в правильный формат. Это тривиальная задача, которая может занять полчаса. Нажмите кнопку «Экспорт» на правой боковой панели.



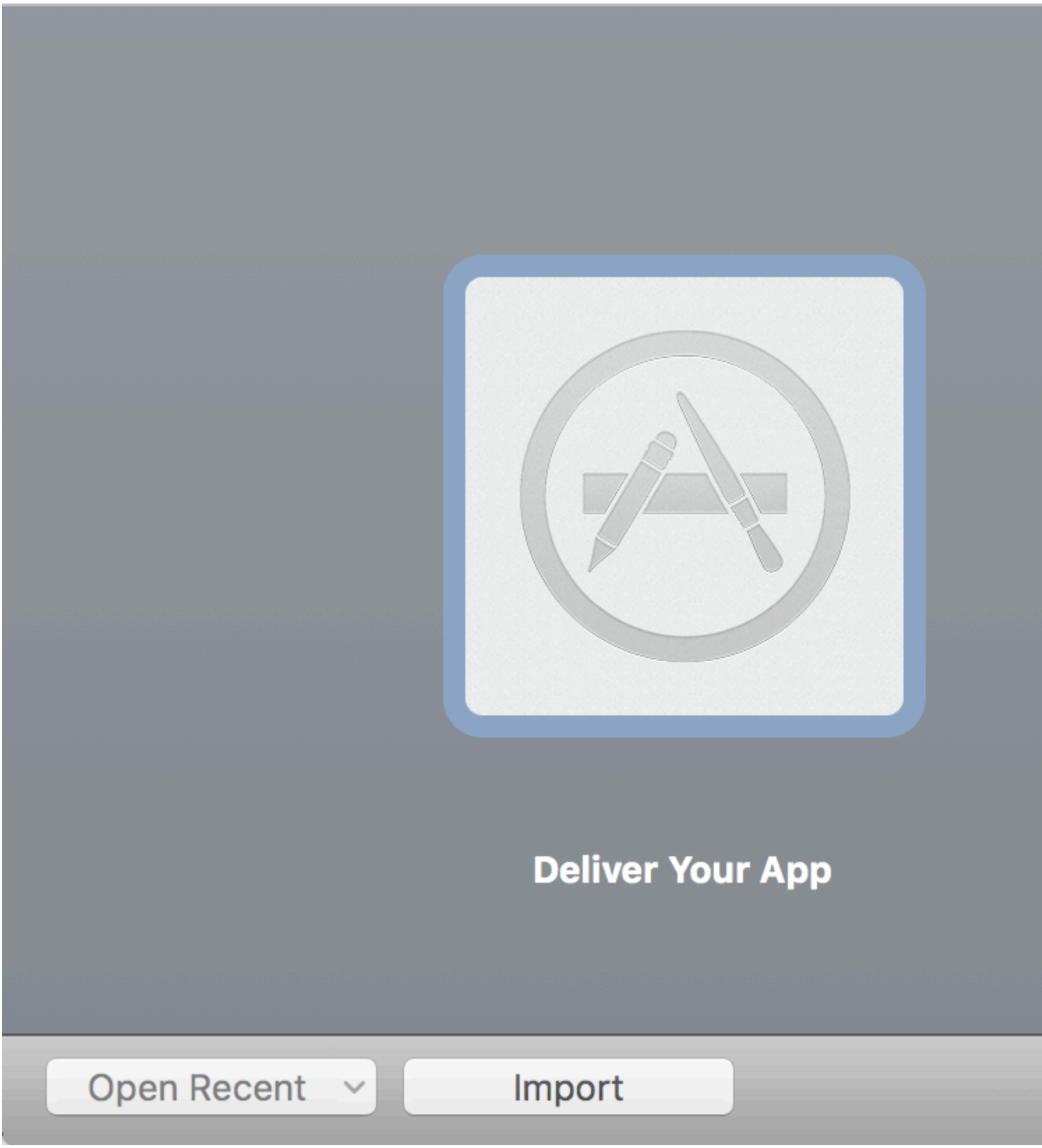
Если вы загружаете приложение в App Store, выберите первый вариант и нажмите «Далее». Войдите в систему и подтвердите свой код еще раз и возьмите чашку кофе. Как только процесс экспорта будет выполнен, вам будет предложено сохранить сгенерированный файл IPA. Обычно рабочий стол - самый удобный выбор.

Загрузить файл IPA с помощью Application Loader

После создания файла IPA откройте Xcode, перейдите к инструментам разработчика и откройте Application Loader.



Если в вашем Xcode имеется несколько учетных записей, вам будет предложено выбрать. Естественно, выберите тот, который вы использовали для подписи кода на первом шаге. Выберите «Доставить приложение» и загрузите код. После завершения загрузки может потребоваться до одного часа, чтобы появиться в списке сборки в iTunes Connect.



Прочитайте Процесс подачи заявки онлайн: https://riptutorial.com/ru/ios/topic/8765/процесс-подачи-заявки

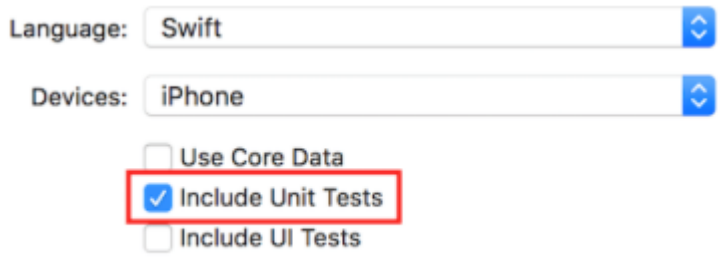
глава 179: Рамка XCTest - Единичное тестирование

Examples

Добавление тестовых файлов в проект Xcode

При создании проекта

В диалоговом окне создания проекта вы должны проверить «Включить тесты единиц».



После создания проекта

Если вы пропустили проверку этого элемента во время создания проекта, вы всегда можете добавить тестовые файлы позже. Для этого:

1- Перейдите к настройкам вашего проекта в Xcode

2- Перейдите в раздел «Цели»

3- Нажмите «Добавить цель»

4- В разделе «Другое» выберите «Пакет тестового тестирования сенсорного блока какао»,

В конце вы должны иметь файл с именем `[Your app name]Tests.swift` . В Objective-C у вас должно быть два файла с именем `[Your app name]Tests.h` и `[Your app name]Tests.m` .

`[Your app name]Tests.swift or .m` будет включать по умолчанию:

- `XCTest` модуля `XCTest`
- А `[Your app name]Tests` `XCTestCase` класс, который расширяет `XCTestCase`
- `setUp` , `tearDown` , `testExample` , `testPerformanceExample` методы

стриж

```
import XCTest

class MyProjectTests: XCTestCase {

override func setUp() {
    super.setUp()
    // Put setup code here. This method is called before the invocation of each test method in
the class.
}

override func tearDown() {
    // Put teardown code here. This method is called after the invocation of each test method
in the class.
    super.tearDown()
}

func testExample() {
    // This is an example of a functional test case.
    // Use XCTAssert and related functions to verify your tests produce the correct results.

}

func testPerformanceExample() {
    // This is an example of a performance test case.
    self.measure {
        // Put the code you want to measure the time of here.
    }
}

}
```

Objective-C

```
#import <XCTest/XCTest.h>

@interface MyProjectTests : XCTestCase

@end

@implementation MyProjectTests

- (void)setUp {
    [super setUp];
// Put setup code here. This method is called before the invocation of each test method in the
class.
}

- (void)tearDown {
// Put teardown code here. This method is called after the invocation of each test method in
the class.
    [super tearDown];
}

- (void)testExample {
// This is an example of a functional test case.
```

```
// Use XCTAssert and related functions to verify your tests produce the correct results.
}

- (void)testPerformanceExample {
// This is an example of a performance test case.
    [self measureBlock:^{
    // Put the code you want to measure the time of here.
    }];
}

@end
```

Добавление Storyboard и View Controller в качестве экземпляров для проверки файла

Чтобы начать работу с модульного тестирования, который будет выполнен в файле тестов, и будет тестировать контроллер просмотра и раскадровки, мы должны ввести эти два файла в тестовый файл.

Определение контроллера просмотра

стриж

```
var viewController : ViewController!
```

Представляем раскадровку и инициализацию контроллера просмотра

Добавьте этот код в метод `setUp()` :

стриж

```
let storyboard = UIStoryboard(name: "Main", bundle: nil)
viewController = storyboard.instantiateInitialViewController() as! ViewController
```

Objective-C

```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:"Main" bundle:nil];
viewController = (ViewController *) [storyboard instantiateInitialViewController];
```

Таким образом, вы можете написать методы тестирования, и они будут знать, где можно проверить наличие ошибок. В этом случае есть View Controller и Storyboard.

Добавление методов тестирования

По словам Apple:

> # Методы испытаний
>
> Метод тестирования - это метод экземпляра тестового класса, который начинается с теста префикса, не принимает параметров и возвращает void, например, (void) testColorIsRed (). Метод тестирования реализует код в вашем проекте и, если этот код не дает ожидаемого результата, сбои отчетов используют набор API-интерфейсов утверждения. Например, возвращаемое значение функции может сравниваться с ожидаемым значением или ваш тест может утверждать, что неправильное использование метода в одном из ваших классов вызывает исключение.

Поэтому мы добавляем тестовый метод, используя «тест» в качестве префикса метода, например:

стриж

```
func testSomething() {

}
```

Objective-C

```
- (void)testSomething {

}
```

Чтобы действительно проверить результаты, мы используем `XCTAssert()` , который принимает логическое выражение, и если true, помечает тест как преуспевающий, иначе он будет отмечать его как неудачный.

Скажем, у нас есть метод в классе View Controller, называемый `sum()` который вычисляет сумму двух чисел. Чтобы проверить это, мы используем этот метод:

стриж

```
func testSum(){
    let result = viewController.sum(4, and: 5)
    XCTAssertEqual(result, 9)
}
```

Objective-C

```
- (void)testSum {
    int result = [viewController sum:4 and:5];
    XCTAssertEqual(result, 9);
}
```

> # Заметка
>
> По умолчанию вы не можете получить доступ к метке, текстовому полю или другим элементам пользовательского интерфейса класса View Controller из тестового класса, если они сначала сделаны в файле Storyboard. Это связано с тем, что они инициализируются в `loadView()` класса View Controller, и это не будет вызываться при тестировании. Лучший способ вызвать `loadView()` и все другие необходимые методы - это доступ к свойству `view` нашего свойства `viewController` . Вы должны добавить эту строку перед тестированием элементов пользовательского интерфейса:

```
XCTAssertNotNil(viewController.view)
```

Начать тестирование

Тестирование определенного метода

Чтобы протестировать определенный метод, щелкните квадрат рядом с определением метода.

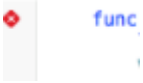
Тестирование всех методов

Чтобы проверить все методы, щелкните квадрат рядом с определением класса.

См. Результат тестирования

Если рядом с определением есть зеленый флажок, тест прошел успешно.

Если рядом с определением есть красный крест, тест не сработал.



Выполнение всех тестов

```
Product -> Test OR Cmd + U
```

Он будет запускать все тесты из всех тестовых целей!

Импортируйте модуль, который можно протестировать

Классы, структуры, перечисления и все их методы являются `internal` по умолчанию. Это означает, что к ним можно получить доступ только из одного модуля. Тесты находятся в другой цели, это означает, что они находятся в другом модуле. Чтобы получить доступ к методу, который вы хотите протестировать, вам необходимо импортировать модуль для тестирования с использованием `@testable` слова `@testable` .

Предположим, у нас есть основной модуль под названием `ToDo` и мы хотим написать для него тесты. Мы будем импортировать этот модуль следующим образом:

```
@testable import ToDo
```

Все методы тестирования в файле с этим оператором импорта теперь могут обращаться ко всем `internal` классам, структурам, перечислениям и всем их `internal` методам модуля `ToDo` .

Вы никогда не должны добавлять файлы с элементами, которые хотите протестировать, в тестовую цель, поскольку это может привести к затруднению отладки ошибок.

Загрузка и появление триггера

Просмотр загрузки

В тесте для контроллера вида вы иногда должны запускать выполнение `loadView()` или `viewDidLoad()` . Это можно сделать, обратившись к представлению. Допустим, у вас есть экземпляр контроллера просмотра в вашем тесте под названием `sut` (система под тестированием), тогда код будет выглядеть так:

```
XCTAssertNotNil(sut.view)
```

Внешний вид

Вы также можете запускать методы `viewWillAppear(_:)` и `viewDidAppear(_:)` путем добавления следующего кода:

```
sut.beginAppearanceTransition(true, animated: true)
sut.endAppearanceTransition()
```

Написание тестового класса

```
import XCTest
@testable import PersonApp

class PersonTests: XCTestCase {
    func test_completeName() {
        let person = Person(firstName: "Josh", lastName: "Brown")
        XCTAssertEqual(person.completeName(), "Josh Brown")
    }
}
```

Теперь давайте обсудим, что здесь происходит. Линия `import XCTest` позволит нам расширить `XCTestCase` и использовать `XCTAssertEqual` (среди других утверждений). Расширение `XCTestCase` и префикс нашего тестового имени с помощью `test` гарантируют, что Xcode автоматически **выполнит** этот тест при выполнении тестов в проекте (**U** или **Product** > **Test**). `@testable import PersonApp` линия `@testable import PersonApp` импортирует нашу цель `PersonApp` чтобы мы могли тестировать и использовать из нее классы, такие как `Person` в нашем примере выше. И, наконец, наш `XCTAssertEqual` гарантирует, что `person.completeName()` будет равно строке `"Josh Brown"` .

Прочитайте Рамка XCTest - Единичное тестирование онлайн:
https://riptutorial.com/ru/ios/topic/5075/рамка-xctest---единичное-тестирование

глава 180: Раскадровка

Вступление

Как правило, диспетчеры представлений в раскадровке создаются и создаются автоматически в ответ на действия, определенные в самом раскадровке. Тем не менее, вы можете использовать объект раскадровки, чтобы создать экземпляр исходного контроллера представления в файле раскадровки или создать экземпляр других контроллеров представлений, которые вы хотите представить программно. Ниже вы найдете примеры обоих вариантов использования.

Examples

инициализировать

```
//Swift
let storyboard = UIStoryboard(name: "Main", bundle: NSBundle.mainBundle())

//Objective-c
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:[NSBundle
mainBundle]];
```

Извлечь начальный ViewController

```
//Swift
let initialScreen = storyboard.instantiateInitialViewController()

//Objective-c
UIViewController *initailScreen = [storyboard instantiateInitialViewController];
```

Fetch ViewController

```
//Swift
let viewController = storyboard.instantiateViewControllerWithIdentifier("identifier")

//Objective-c
UIViewController *viewController = [storyboard
instantiateViewControllerWithIdentifier:@"identifier"];
```

Прочитайте Раскадровка онлайн: https://riptutorial.com/ru/ios/topic/3514/раскадровка

глава 181: Расположение ядра

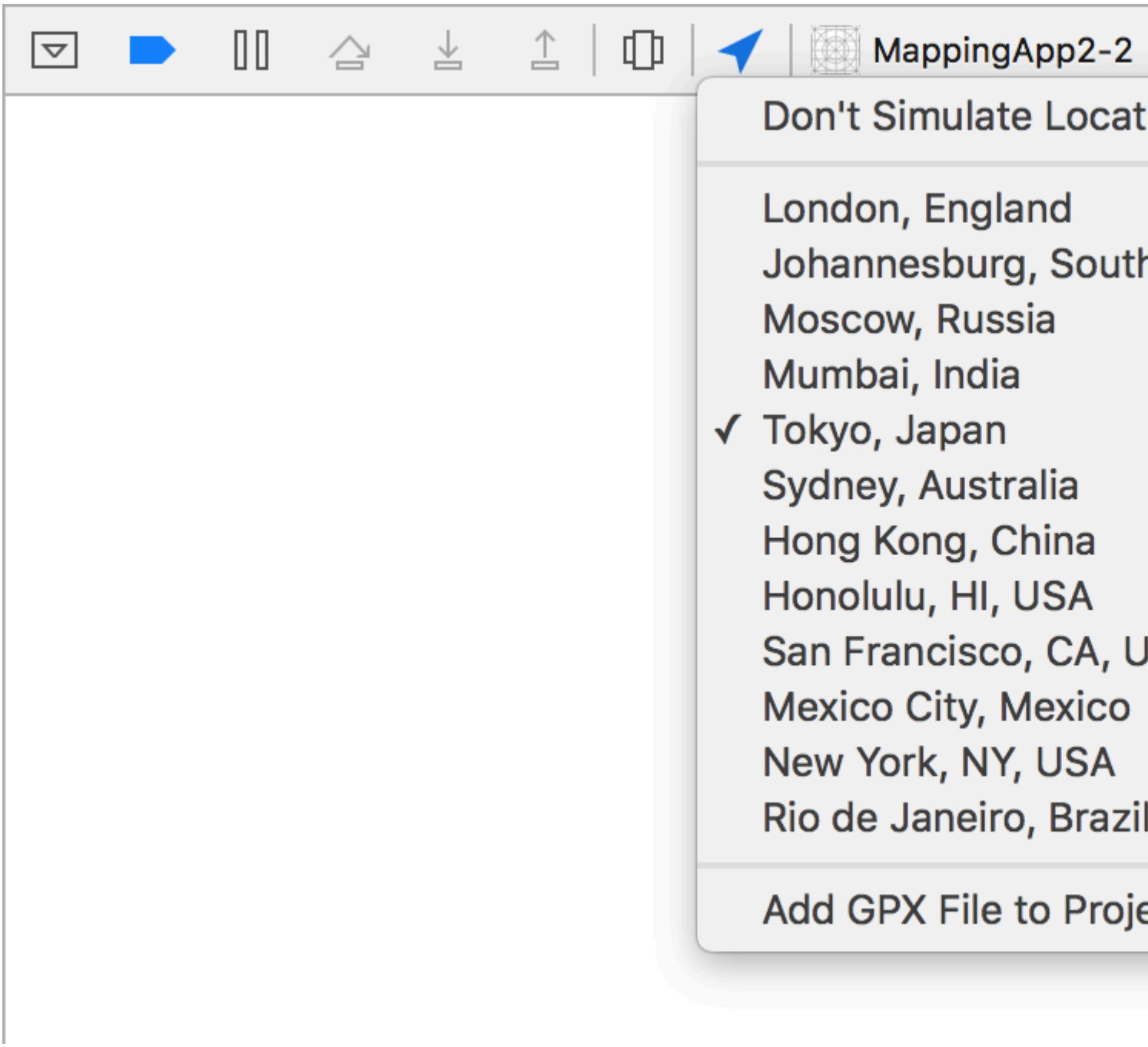
Синтаксис

1. desiredAccuracy
2. distanceFilter
3. requestLocation ()
4. startUpdatingLocation ()
5. allowDeferredLocationUpdates (untilTraveled: тайм-аут :)
6. startMonitoringSignificantLocationChanges ()
7. allowDeferredLocationUpdates (untilTraveled: тайм-аут :)
8. authorizedAlways
9. authorizedWhenInUse
10. locationManager (_: didChangeAuthorization :)

замечания

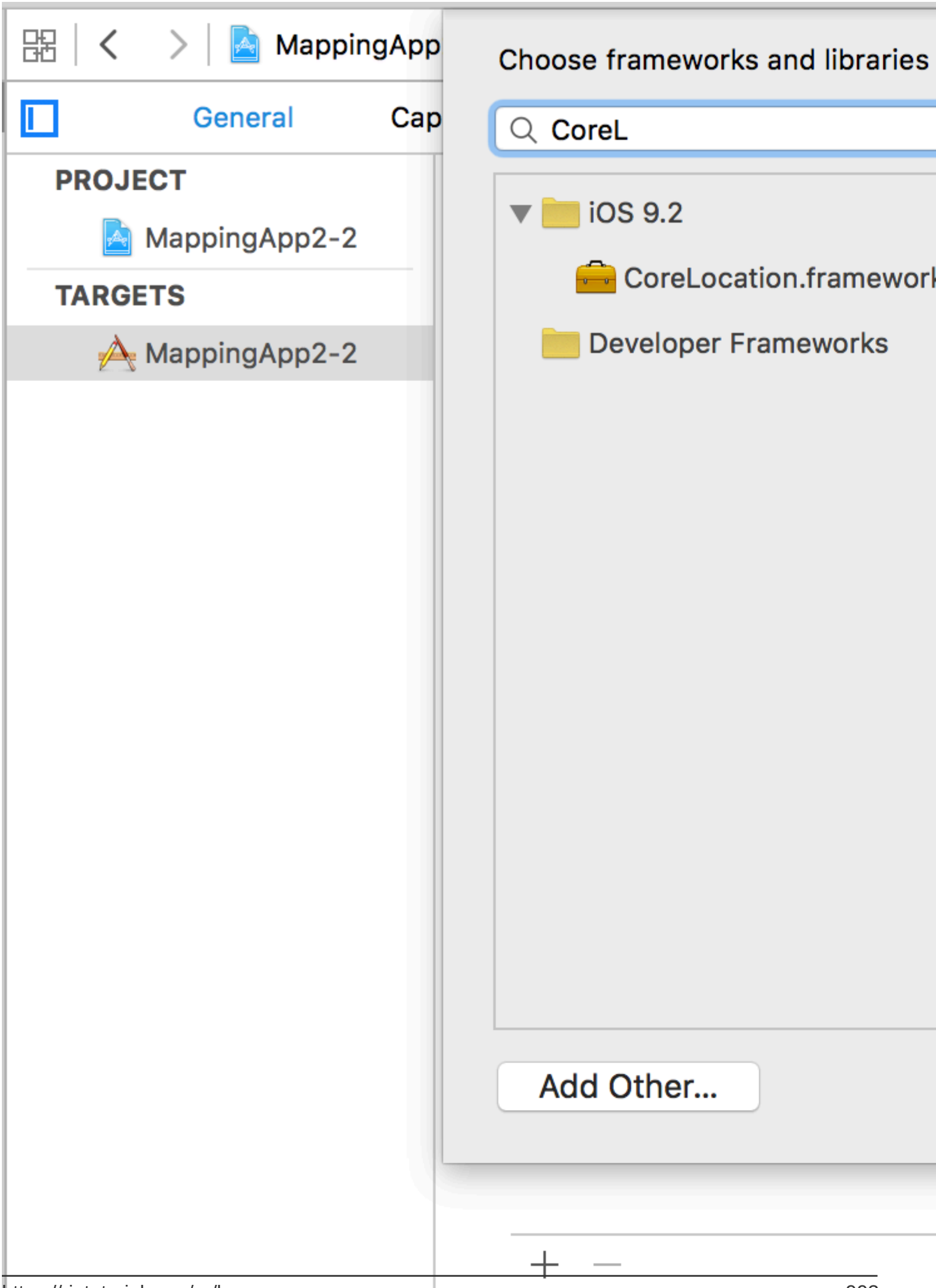
Имитировать местоположение в Runtime

1. Запустите приложение из Xcode.
2. В панели отладки нажмите кнопку «Имитировать местоположение».
3. Выберите местоположение из меню.



Examples

Link CoreLocation Framework


MappingApp
General
Cap
PROJECT
MappingApp2-2
TARGETS
MappingApp2-2
Choose frameworks and libraries
CoreL
iOS 9.2
CoreLocation.framework
Developer Frameworks
Add Other...

CoreLocation.

```
//Swift
import CoreLocation

//Objective-C
#import <CoreLocation/CoreLocation.h>
```

Запросить разрешение на использование служб местоположения

Проверьте статус авторизации приложения с помощью:

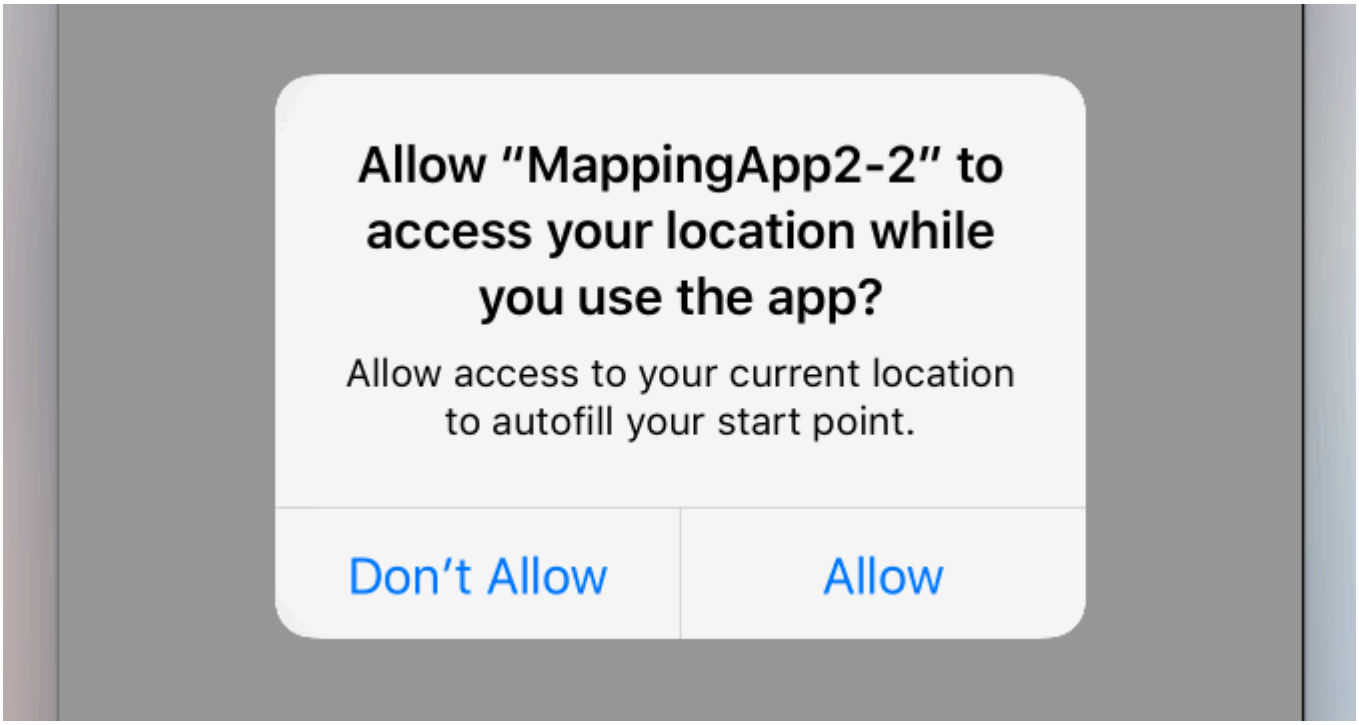
```
//Swift
let status: CLAuthorizationStatus = CLLocationManager.authorizationStatus()

//Objective-C
CLAuthorizationStatus status = [CLLocationManager authorizationStatus];
```

Проверьте статус на следующие константы:

```
//Swift
switch status {
case .NotDetermined:
    // Do stuff
case .AuthorizedAlways:
    // Do stuff
case .AuthorizedWhenInUse:
    // Do stuff
case .Restricted:
    // Do stuff
case .Denied:
    // Do stuff
}

//Objective-C
switch (status) {
    case kCLAuthorizationStatusNotDetermined:

        //The user hasn't yet chosen whether your app can use location services or not.

        break;

    case kCLAuthorizationStatusAuthorizedAlways:

        //The user has let your app use location services all the time, even if the app is in
the background.

        break;

    case kCLAuthorizationStatusAuthorizedWhenInUse:

        //The user has let your app use location services only when the app is in the
foreground.

        break;

    case kCLAuthorizationStatusRestricted:
```

```
        //The user can't choose whether or not your app can use location services or not, this
could be due to parental controls for example.

        break;

    case kCLAuthorizationStatusDenied:

        //The user has chosen to not let your app use location services.

        break;

    default:
        break;
}
```

Получение разрешения на доступ к местоположению во время работы приложения



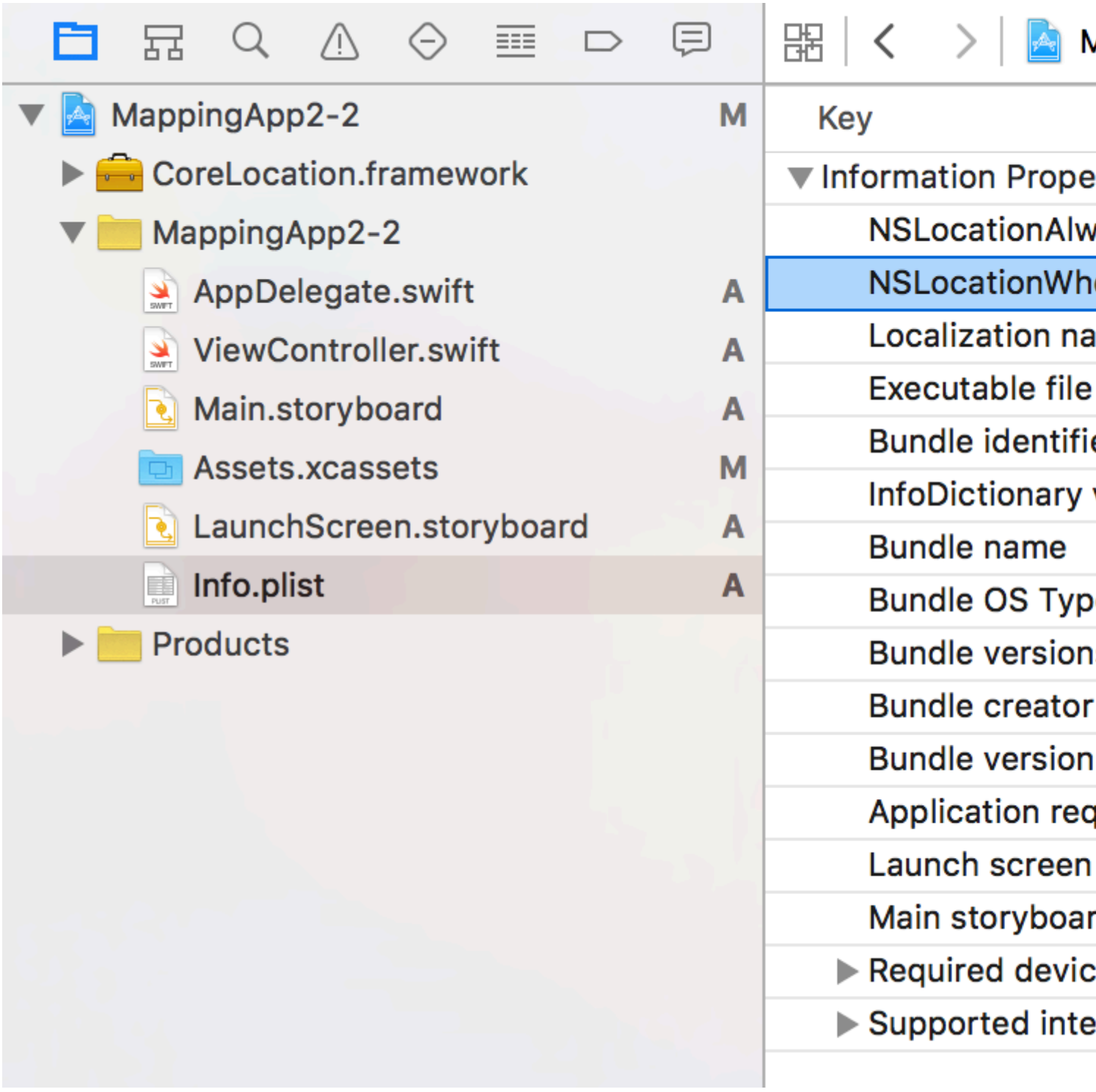
`viewDidLoad` способ - инициализировать диспетчер местоположений как свойство вашего контроллера корневого представления и поместить запрос разрешения в свой `viewDidLoad` .

Это вызывает контроллер предупреждений, который запрашивает разрешение:

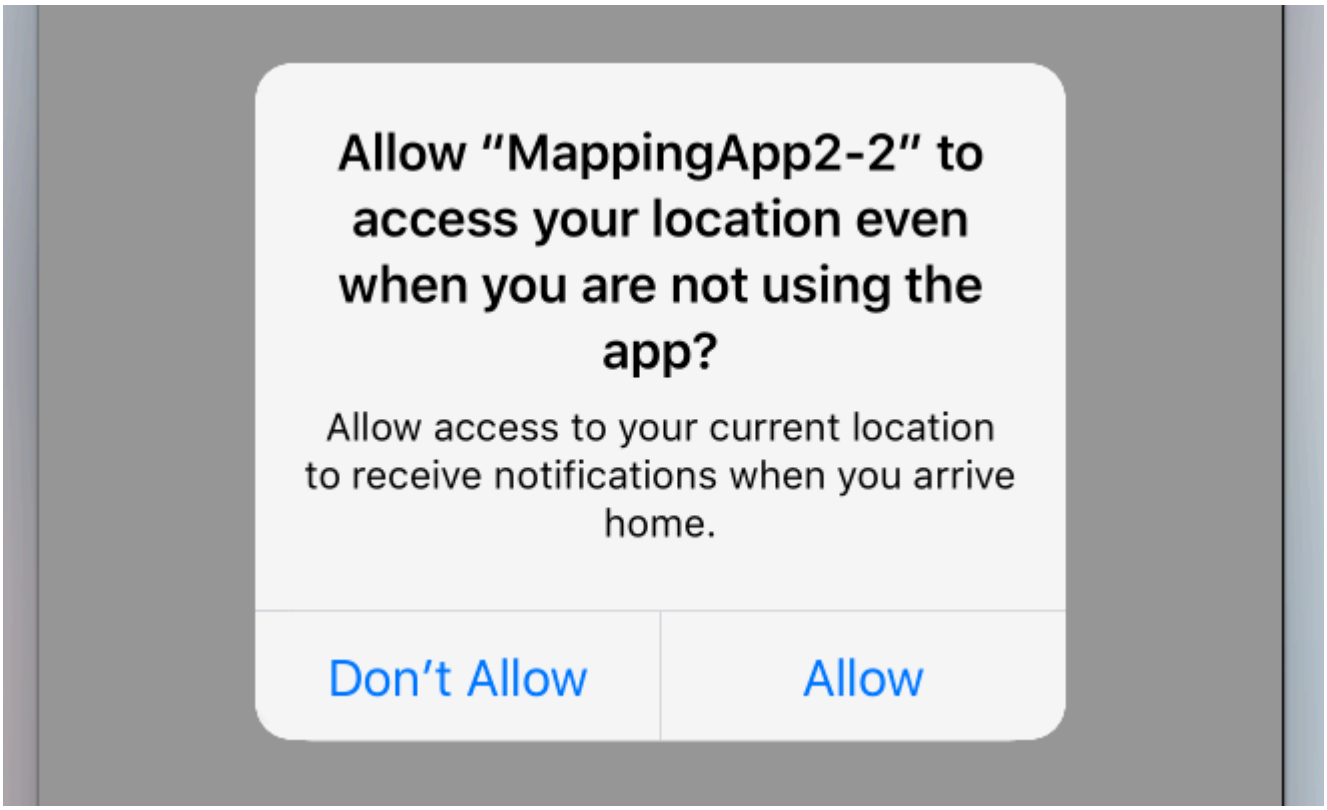
```
//Swift
let locationManager = CLLocationManager()
locationManager.requestWhenInUseAuthorization()
```

```
//Objective-C
CLLocationManager *locationManager = [[CLLocationManager alloc] init];
[locationManager requestWhenInUseAuthorization];
```

Добавьте ключ **NSLocationWhenInUseUsageDescription** в свой *Info.plist* . Значение будет использоваться в метке `message` контроллера предупреждения.



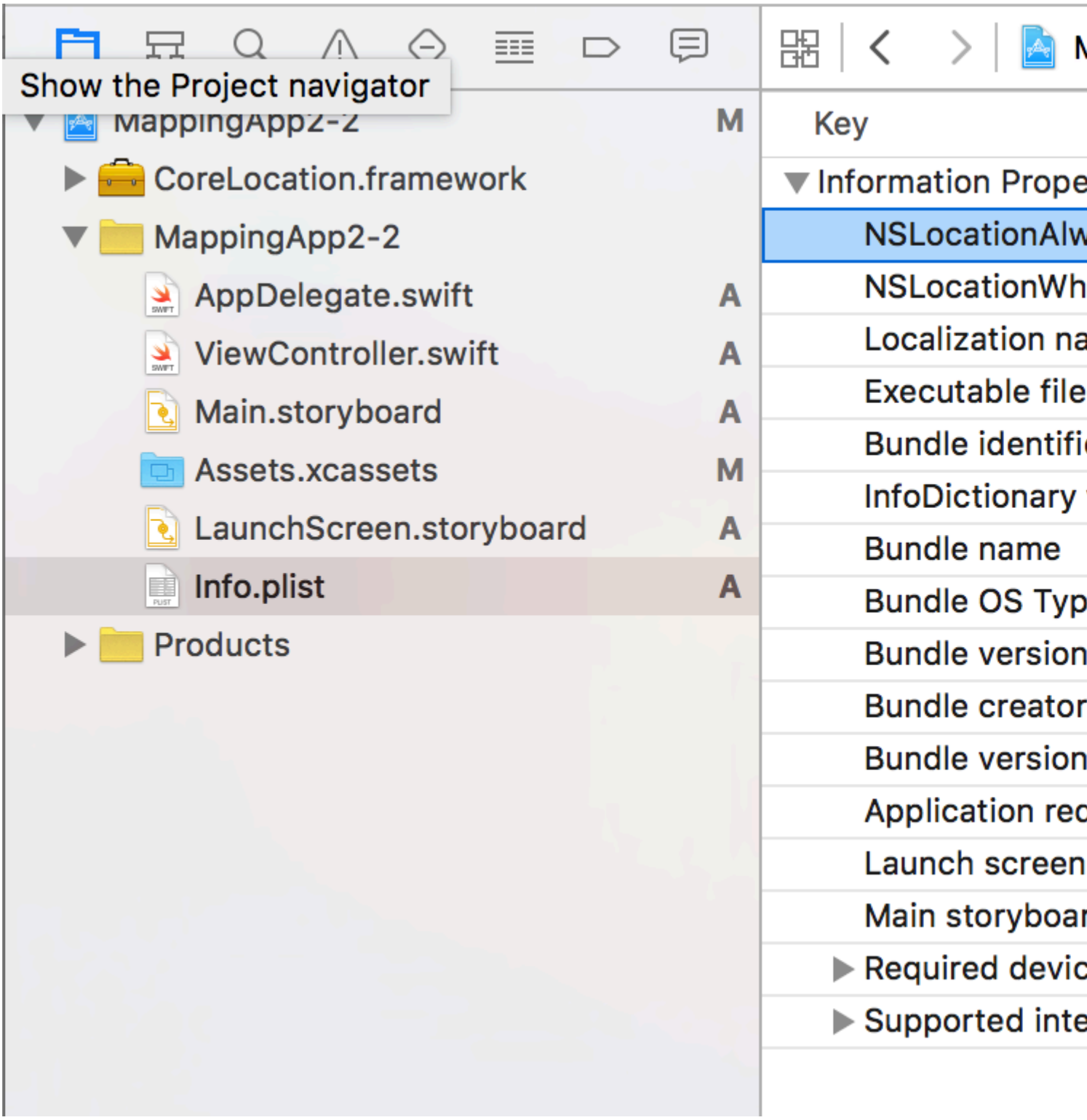
Получение разрешения на обслуживание



Чтобы запросить разрешение на использование служб определения местоположения, даже если приложение неактивно, вместо этого используйте следующий вызов:

```
//Swift
locationManager.requestAlwaysAuthorization()

//Objective-C
[locationManager requestAlwaysAuthorization];
```

Затем добавьте ключ **NSLocationAlwaysUsageDescription** в свой *Info.plist* . Опять же, значение будет использоваться в метке `message` контроллера предупреждения.



Добавить собственное пользовательское местоположение с помощью файла GPX

Для проверки сервисов определения местоположения нам нужно настоящее устройство, но для целей тестирования мы также можем использовать симулятор и добавить наше собственное местоположение, выполнив следующие шаги:

- добавьте новый GPX-файл в свой проект.
- в файле GPX добавьте путевые точки, например

```
<?xml version="1.0"?>
<gpx version="1.1" creator="Xcode">
<!--
        Provide one or more waypoints containing a latitude/longitude pair. If you provide one
        waypoint, Xcode will simulate that specific location. If you provide multiple
waypoints,
        Xcode will simulate a route visitng each waypoint.
 -->
<wpt lat="52.599878" lon="4.702029">
     <name>location name (eg. Florida)</name>
</wpt>
```

- затем перейдите к продукту -> Схема -> Изменить схему и в RUN укажите местоположение по умолчанию в качестве имени вашего GPX-файла.

Услуги по размещению в фоновом режиме

Чтобы использовать стандартные службы определения местоположения, когда приложение находится в фоновом режиме, сначала необходимо включить `Background Modes` режим на вкладке «Возможности» целевых настроек и выбрать « `Location updates` .

Или добавьте его непосредственно в Info.plist.

```
<key>NSLocationAlwaysUsageDescription</key>
<string>I want to get your location Information in background</string>

<key>UIBackgroundModes</key>
<array>
    <string>location</string>
</array>
```

Затем вам нужно настроить CLLocationManager

Цель C

```
//The Location Manager must have a strong reference to it.
_locationManager = [[CLLocationManager alloc] init];
_locationManager.delegate = self;

//Request Always authorization (iOS8+)
if ([_locationManager respondsToSelector:@selector(requestAlwaysAuthorization)]) {
    [_locationManager requestAlwaysAuthorization];
}

//Allow location updates in the background (iOS9+)
if ([_locationManager respondsToSelector:@selector(allowsBackgroundLocationUpdates)]) {
    _locationManager.allowsBackgroundLocationUpdates = YES;
}

[_locationManager startUpdatingLocation];
```

стриж

```
self.locationManager.delegate = self

if #available (iOS 8.0,*) {
    self.locationManager.requestAlwaysAuthorization()
}

if #available (iOS 9.0,*) {
    self.locationManager.allowsBackgroundLocationUpdates = true
}

self.locationManager.startUpdatingLocation()
```

Прочитайте Расположение ядра онлайн: https://riptutorial.com/ru/ios/topic/2937/расположение-ядра

глава 182: Расширение для расширенного Push-уведомления - iOS 10.

Вступление

iOS 10 предоставил нам `UserNotifications.framework` , новый API для локальных / удаленных уведомлений. Он предлагает просмотр вложений в медиа или ответы на сообщения прямо из уведомления.

Содержание уведомлений состоит из: названия, субтитров, тела и вложения. Приложение может содержать изображения / gifs / videos до 50 мб.

Examples

Расширение содержимого уведомлений

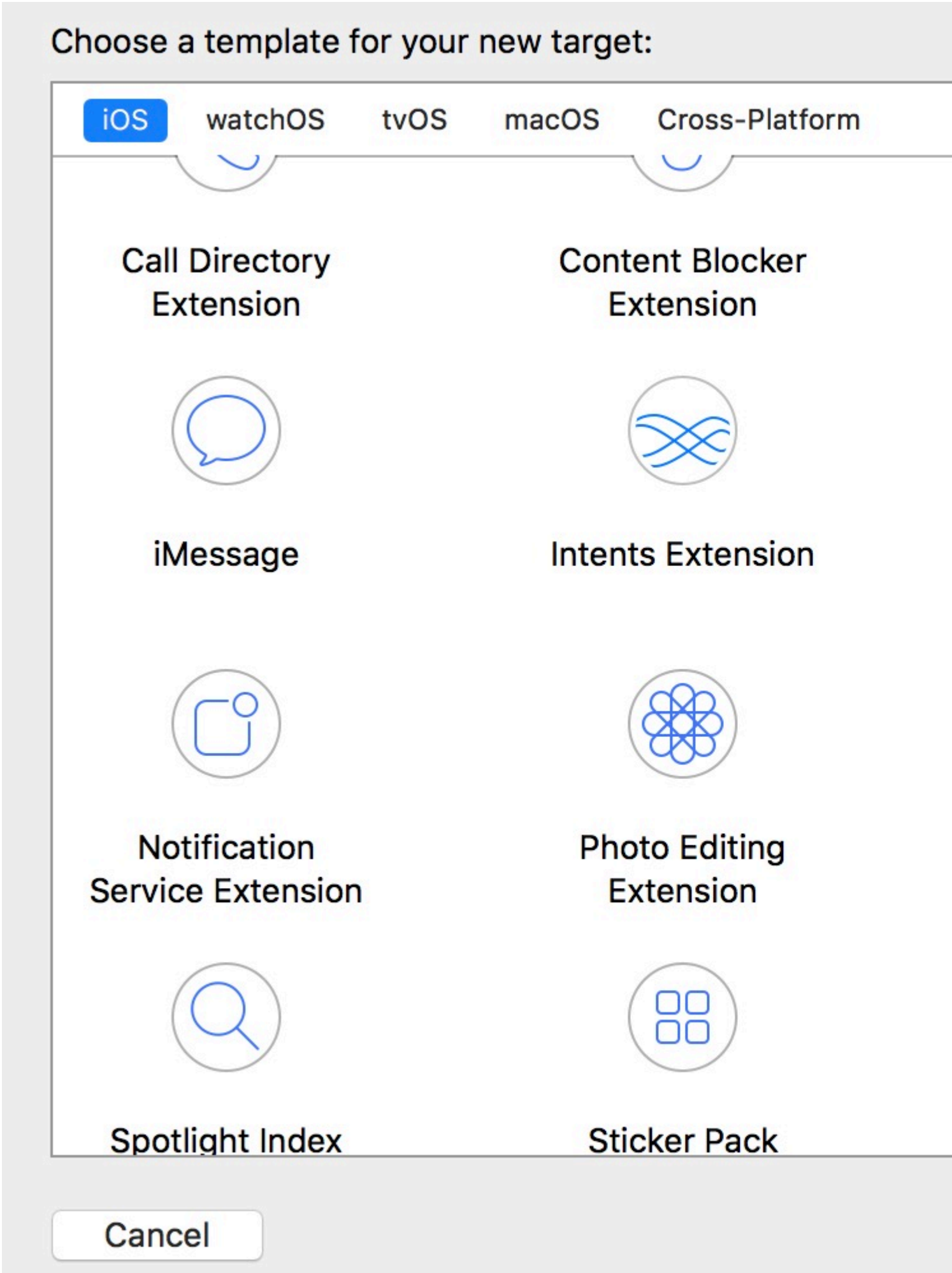
Зачем нам это нужно?

Расширение содержимого помогает нам создавать пользовательский интерфейс при расширении уведомлений.

Вы используете эту структуру для определения расширения, которое получает данные уведомления и предоставляет соответствующее визуальное представление. Ваше расширение также может реагировать на пользовательские действия, связанные с этими уведомлениями.

Реализация

1. В окне xCode `Navigator` перейдите в раздел `Targets` . Нажмите `Add New Target` .
2. Выберите шаблон `Notification Content Extension` :



3. В файле `info.plist` укажите идентификатор ключа `UNNotificationExtensionCategory` :

система отображает только ваш пользовательский контроллер представления в интерфейсе уведомления. Если установлено значение НЕТ, система отображает контент уведомления по умолчанию в дополнение к содержимому вашего контроллера.

`UNNotificationExtensionOverridesDefaultTitle` (необязательно)

Значение этого ключа является логическим. Если установлено значение true, система использует свойство title вашего контроллера представления как название уведомления. Когда установлено значение false, система устанавливает название уведомления на имя вашего приложения. Если вы не укажете этот ключ, значение по умолчанию будет установлено на false.

4. Создание пользовательского представления в файле `NotificationViewController.swift`
5. Добавьте новый `category key` и установите его значение в том, что мы ввели в Info.plist (шаг 3):

От себя:

```
{
 aps: {
 alert: { … },
 category: 'io.swifting.notification-category'
 }
}
```

Местный:

```
let mutableNotificationContent = UNMutableNotificationContent()
mutableNotificationContent.category = "io.swifting.notification-category"
mutableNotificationContent.title = "Swifting.io Notifications"
mutableNotificationContent.subtitle = "Swifting.io presents"
mutableNotificationContent.body = "Custom notifications"
```

Также ознакомьтесь с официальной ссылкой API:
https://developer.apple.com/reference/usernotificationsui/unnotificationcontentextension?utm_source=swi

Прочитайте Расширение для расширенного Push-уведомления - iOS 10. онлайн:
https://riptutorial.com/ru/ios/topic/9501/расширение-для-расширенного-push-уведомления---ios-10-

глава 183: Режимы фона

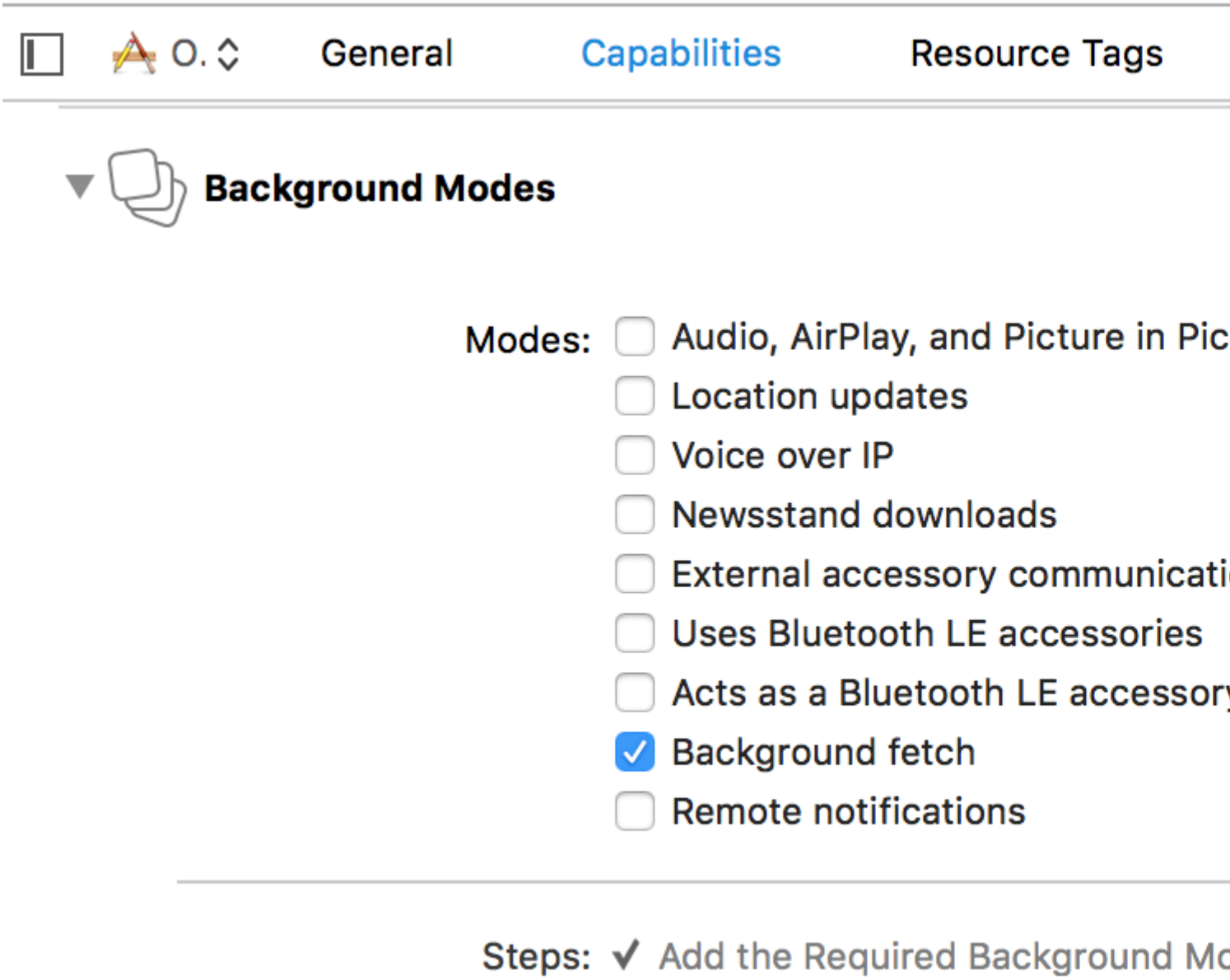
Вступление

Быть отзывчивым - это потребность в каждом приложении. Пользователи хотят иметь приложения, у которых есть готовый контент, когда они его открывают, поэтому разработчики должны использовать Фоновые режимы, чтобы сделать их приложения более удобными для пользователя.

Examples

Включение возможности фоновых режимов

1. Перейдите в Xcode и откройте свой проект.
2. В своей целевой программе перейдите на вкладку «Возможности».
3. Включите фоновый режим.



Фоновая выборка

Фоновая выборка - это новый режим, позволяющий вашему приложению всегда быть в курсе новейшей информации, минимизируя влияние на батарею. С этой возможностью вы можете загружать фиды с фиксированными временными интервалами.

Для начала:

1- Проверить фоновый выбор в окне возможностей в Xcode.

2- В `application(_:didFinishLaunchingWithOptions:)` в приложении `AppDelegate` добавьте:

стриж

```
UIApplication.shared.setMinimumBackgroundFetchInterval(UIApplicationBackgroundFetchIntervalMinimum)
```

Objective-C

```
[[UIApplication shared]
setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMinimum]
```

> Вместо `UIApplicationBackgroundFetchIntervalMinimum` вы можете использовать любое значение `CGFloat` для установки интервалов выборки.

3- Вы должны реализовать `application(_:performFetchWithCompletionHandler:)` . Добавьте это в свой `AppDelegate` :

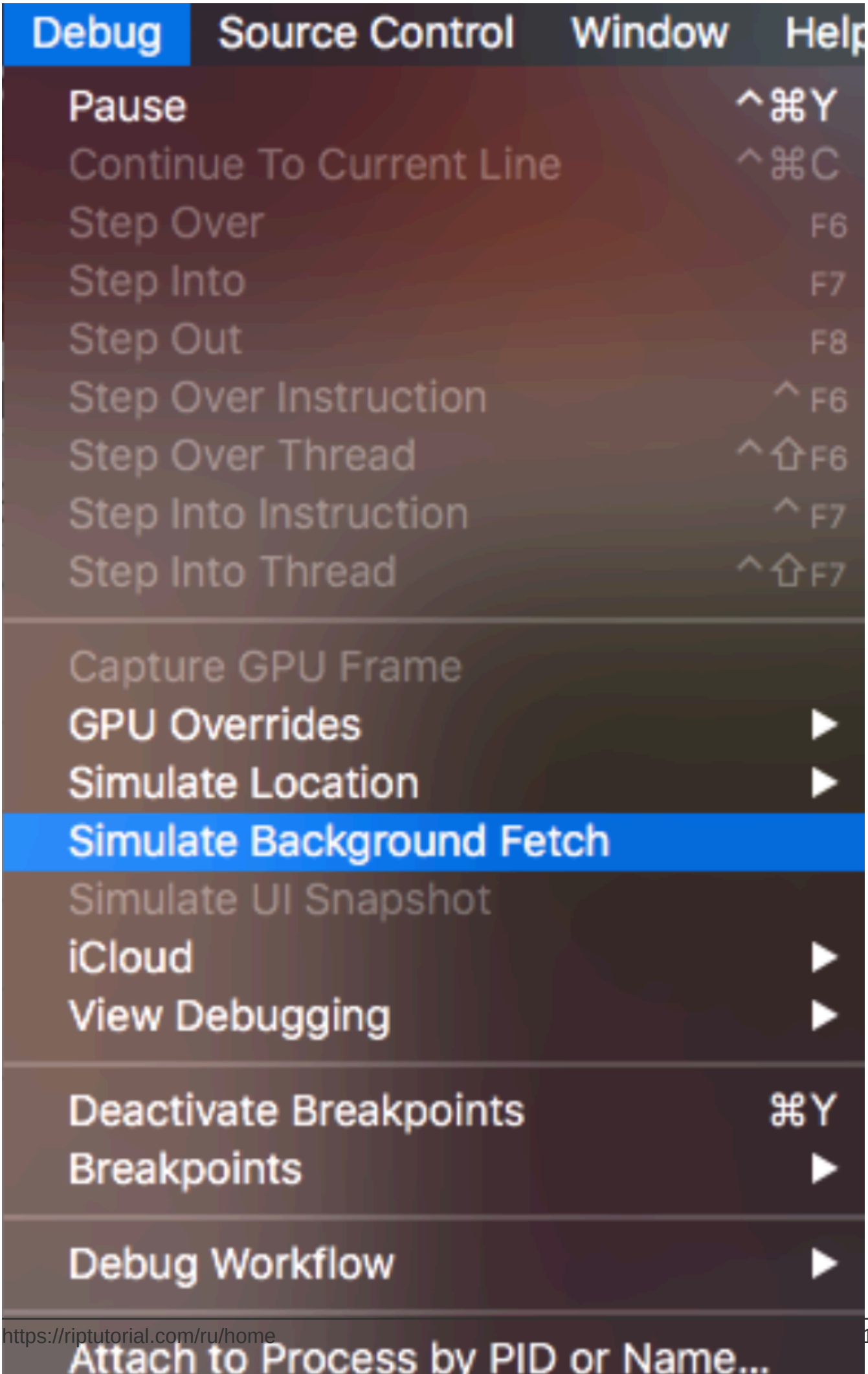
стриж

```
func application(_ application: UIApplication, performFetchWithCompletionHandler
completionHandler: @escaping (UIBackgroundFetchResult) -> Void) {
    // your code here
}
```

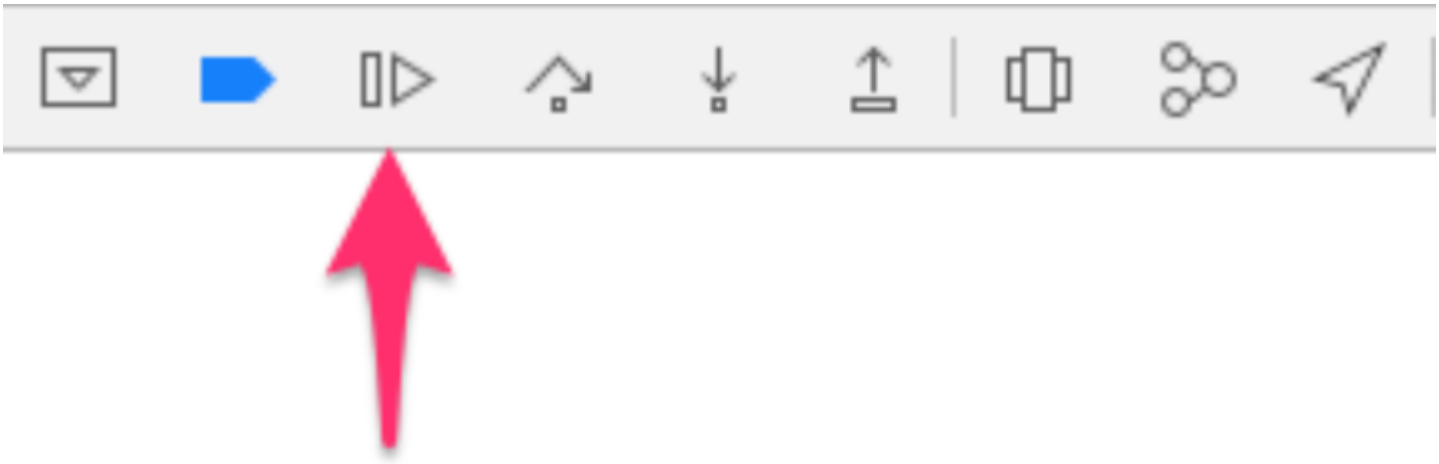
Тестирование фоновой выборки

1- Запустите приложение на реальном устройстве и присоедините его к отладчику Xcode.

2- В меню «Отладка» выберите « **Имитировать фоновый** выбор»:


Debug
Source Control
Window
Help
Pause ^⌘Y
Continue To Current Line ^⌘C
Step Over F6
Step Into F7
Step Out F8
Step Over Instruction ^F6
Step Over Thread ^⇧F6
Step Into Instruction ^F7
Step Into Thread ^⇧F7
Capture GPU Frame
GPU Overrides
Simulate Location
Simulate Background Fetch
Simulate UI Snapshot
iCloud
View Debugging
Deactivate Breakpoints ⌘Y
Breakpoints
Debug Workflow
Attach to Process by PID or Name...

Просто нажмите кнопку «Продолжить», чтобы приложение выполнило фоновое извлечение.



Теперь вы увидите, что данные получены и готовы для вас.

Фоновый звук

По умолчанию при потоковой передаче звука, выйдя из приложения, он остановится, но вы можете предотвратить это, включив первый флажок на странице «Возможности фона» в Xcode.

iOS автоматически обработает это для вас, и вам не нужно писать какой-либо код!

Прочитайте Режимы фона онлайн: https://riptutorial.com/ru/ios/topic/9178/режимы-фона

глава 184: Руководство по выбору лучших шаблонов архитектуры iOS

Вступление

Демистификация MVC, MVP, MVVM и VIPER или любых других шаблонов проектирования для выбора наилучшего подхода к созданию приложения

Examples

Шаблон MVC

```
import UIKit

struct Person { // Model
    let firstName: String
    let lastName: String
}

class GreetingViewController : UIViewController { // View + Controller
    var person: Person!
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self, action: "didTapButton:", forControlEvents:
.TouchUpInside)
    }

    func didTapButton(button: UIButton) {
        let greeting = "Hello" + " " + self.person.firstName + " " + self.person.lastName
        self.greetingLabel.text = greeting

    }
    // layout code goes here
}
// Assembling of MVC
let model = Person(firstName: "David", lastName: "Blaine")
let view = GreetingViewController()
view.person = model
```

Шаблоны MVP

```
import UIKit

struct Person { // Model
    let firstName: String
    let lastName: String
}
```

```
protocol GreetingView: class {
    func setGreeting(greeting: String)
}

protocol GreetingViewPresenter {
    init(view: GreetingView, person: Person)
    func showGreeting()
}

class GreetingPresenter : GreetingViewPresenter {
    unowned let view: GreetingView
    let person: Person
    required init(view: GreetingView, person: Person) {
        self.view = view
        self.person = person
    }
    func showGreeting() {
        let greeting = "Hello" + " " + self.person.firstName + " " + self.person.lastName
        self.view.setGreeting(greeting)
    }
}

class GreetingViewController : UIViewController, GreetingView {
    var presenter: GreetingViewPresenter!
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self, action: "didTapButton:", forControlEvents:
.TouchUpInside)
    }

    func didTapButton(button: UIButton) {
        self.presenter.showGreeting()
    }

    func setGreeting(greeting: String) {
        self.greetingLabel.text = greeting
    }

    // layout code goes here
}
// Assembling of MVP
let model = Person(firstName: "David", lastName: "Blaine")
let view = GreetingViewController()
let presenter = GreetingPresenter(view: view, person: model)
view.presenter = presenter
```

Шаблон MVVM

```
import UIKit

struct Person { // Model
    let firstName: String
    let lastName: String
}
```

```
protocol GreetingViewModelProtocol: class {
    var greeting: String? { get }
    var greetingDidChange: ((GreetingViewModelProtocol) -> ())? { get set } // function to
call when greeting did change
    init(person: Person)
    func showGreeting()
}

class GreetingViewModel : GreetingViewModelProtocol {
    let person: Person
    var greeting: String? {
        didSet {
            self.greetingDidChange?(self)
        }
    }
    var greetingDidChange: ((GreetingViewModelProtocol) -> ())?
    required init(person: Person) {
        self.person = person
    }
    func showGreeting() {
        self.greeting = "Hello" + " " + self.person.firstName + " " + self.person.lastName
    }
}

class GreetingViewController : UIViewController {
    var viewModel: GreetingViewModelProtocol! {
        didSet {
            self.viewModel.greetingDidChange = { [unowned self] viewModel in
                self.greetingLabel.text = viewModel.greeting
            }
        }
    }
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self.viewModel, action: "showGreeting",
forControlEvents: .TouchUpInside)
    }
    // layout code goes here
}
// Assembling of MVVM
let model = Person(firstName: "David", lastName: "Blaine")
let viewModel = GreetingViewModel(person: model)
let view = GreetingViewController()
view.viewModel = viewModel
```

Шаблон VIPER

```
import UIKit

struct Person { // Entity (usually more complex e.g. NSManagedObject)
    let firstName: String
    let lastName: String
}

struct GreetingData { // Transport data structure (not Entity)
    let greeting: String
```

```
    let subject: String
}

protocol GreetingProvider {
    func provideGreetingData()
}

protocol GreetingOutput: class {
    func receiveGreetingData(greetingData: GreetingData)
}

class GreetingInteractor : GreetingProvider {
    weak var output: GreetingOutput!

    func provideGreetingData() {
        let person = Person(firstName: "David", lastName: "Blaine") // usually comes from data
access layer
        let subject = person.firstName + " " + person.lastName
        let greeting = GreetingData(greeting: "Hello", subject: subject)
        self.output.receiveGreetingData(greeting)
    }
}

protocol GreetingViewEventHandler {
    func didTapShowGreetingButton()
}

protocol GreetingView: class {
    func setGreeting(greeting: String)
}

class GreetingPresenter : GreetingOutput, GreetingViewEventHandler {
    weak var view: GreetingView!
    var greetingProvider: GreetingProvider!

    func didTapShowGreetingButton() {
        self.greetingProvider.provideGreetingData()
    }

    func receiveGreetingData(greetingData: GreetingData) {
        let greeting = greetingData.greeting + " " + greetingData.subject
        self.view.setGreeting(greeting)
    }
}

class GreetingViewController : UIViewController, GreetingView {
    var eventHandler: GreetingViewEventHandler!
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self, action: "didTapButton:", forControlEvents:
.TouchUpInside)
    }

    func didTapButton(button: UIButton) {
        self.eventHandler.didTapShowGreetingButton()
    }

    func setGreeting(greeting: String) {
```

```
        self.greetingLabel.text = greeting
    }

    // layout code goes here
}
// Assembling of VIPER module, without Router
let view = GreetingViewController()
let presenter = GreetingPresenter()
let interactor = GreetingInteractor()
view.eventHandler = presenter
presenter.view = view
presenter.greetingProvider = interactor
interactor.output = presenter
```

Прочитайте Руководство по выбору лучших шаблонов архитектуры iOS онлайн: https://riptutorial.com/ru/ios/topic/10029/руководство-по-выбору-лучших-шаблонов-архитектуры-ios

глава 185: Связанные объекты Objective-C

Вступление

Впервые представленный в iOS 3.1 как часть среды выполнения Objective-C, связанные объекты предоставляют способ добавления переменных экземпляра в существующий объект класса (w \ o subclassing).

Это означает, что вы сможете присоединить любой объект к любому другому объекту без подкласса.

Синтаксис

- void objc_setAssociatedObject (id object, void * key, id value, objc_AssociationPolicy policy)
- id objc_getAssociatedObject (id object, void * key)
- void objc_removeAssociatedObjects (объект id)

параметры

Param	подробности
объект	Существующий объект, который вы хотите изменить
ключ	В основном это может быть любой указатель, который имеет постоянный адрес памяти, но хорошей практикой является использование здесь вычисленного свойства (getter)
значение	Объект, который вы хотите добавить
политика	Политика памяти для этого нового `value` т. Е. Должна быть сохранена / назначена, скопирована и т. Д. Точно так же, как любое другое свойство, которое вы объявили

замечания

Подробнее здесь:

NSHipster

@kostiakoval

kingscocoa

Examples

Пример базового ассоциированного объекта

Предположим, нам нужно добавить объект NSString к `SomeClass` (мы не можем подкласса).

В этом примере мы не только создаем связанный объект, но и переносим его в вычисленное свойство в категории для дополнительной аккуратности

```
#import <objc/runtime.h>

@interface SomeClass (MyCategory)
// This is the property wrapping the associated object. below we implement the setter and
getter which actually utilize the object association
@property (nonatomic, retain) NSString *associated;
@end

@implementation SomeClass (MyCategory)

- (void)setAssociated:(NSString *)object {
    objc_setAssociatedObject(self, @selector(associated), object,
                             OBJC_ASSOCIATION_RETAIN_NONATOMIC);
}

- (NSString *)associated {
    return objc_getAssociatedObject(self, @selector(associated));
}
```

Теперь было бы так просто, как использовать свойство

```
SomeClass *instance = [SomeClass alloc] init];
instance.associated = @"this property is an associated object under the hood";
```

Прочитайте Связанные объекты Objective-C онлайн: https://riptutorial.com/ru/ios/topic/9102/связанные-объекты-objective-c

глава 186: Сделайте округлые углы UIView

Examples

Объективный код C, чтобы сделать выбранный угол UiView округленным

Сначала **импортируйте** `#import <QuartzCore/QuartzCore.h>` в свой класс ViewController. Вот как я установил свой код в коде

```
UIView *view1=[[UIView alloc]init];
view1.backgroundColor=[UIColor colorWithRed:255/255.0 green:193/255.0 blue:72/255.0
alpha:1.0];
CGRect view1Frame = view1.frame;
view1Frame.size.width = SCREEN_WIDTH*0.97;
view1Frame.size.height = SCREEN_HEIGHT*0.2158;
view1Frame.origin.x = 0;
view1Frame.origin.y = 0.1422*SCREEN_HEIGHT-10;
view1.frame = view1Frame;
[self setMaskTo:view1 byRoundingCorners:UIRectCornerBottomRight|UIRectCornerTopRight];
[self.view addSubview:view1];
```

Вот функция, которая делает тяжелый подъем и округляет выбранные края, которые являются нижним правым и верхним правым краем в нашем случае

```
- (void)setMaskTo:(UIView*)view byRoundingCorners:(UIRectCorner)corners
{
    UIBezierPath *rounded = [UIBezierPath bezierPathWithRoundedRect:view.bounds
                                                  byRoundingCorners:corners
                                                        cornerRadii:CGSizeMake(20.0, 20.0)];
    CAShapeLayer *shape = [[CAShapeLayer alloc] init];
    [shape setPath:rounded.CGPath];
    view.layer.mask = shape;
}
```

Прочитайте Сделайте округлые углы UIView онлайн: https://riptutorial.com/ru/ios/topic/7224/сделайте-округлые-углы-uiview

глава 187: Сжатие содержимого / сжатие содержимого в автозапуске

замечания

Приоритет сжатия содержимого

> Это значение определяет, насколько устойчивым считается сжатие или сжатие. Более высокое значение здесь означает, что представление будет с меньшей вероятностью сжиматься и, скорее всего, останется прежним.

Приоритет обхода контента

> Это значение определяет, насколько устойчивым является просмотр для расширения. Вы можете себе представить, что «обнимать» здесь означает «размер для соответствия» - границы представления «обнимаются» или близки к размеру внутреннего содержимого. Более высокое значение здесь означает, что представление будет менее вероятно расти и, скорее всего, останется прежним.

Examples

Определение: Внутренний размер содержимого

Перед авторазложением вам всегда приходилось указывать кнопкам и другим элементам управления, насколько они велики, либо устанавливая их свойства рамки или границ, либо изменяя их размеры в интерфейсе Builder. Но оказывается, что большинство элементов управления отлично способны определить, сколько места им нужно, основываясь на их содержании.

Ярлык знает, насколько он широк и высок, потому что он знает длину текста, который был установлен на нем, а также размер шрифта для этого текста. Аналогично для **кнопки** , которая может сочетать текст с фоновым изображением и некоторым дополнением.

То же самое верно для сегментированных элементов управления, индикаторов выполнения и большинства других элементов управления, хотя некоторые могут иметь только заданную высоту, но неизвестную ширину.

Это известно как собственный размер содержимого, и это важная концепция в Auto Layout. Auto Layout запрашивает у вашего контроля, насколько большой он должен быть, и выставляет экран на основе этой информации.

Обычно вы хотите использовать `intrinsic content size` , но есть случаи, когда вы не можете этого делать. Это можно предотвратить, установив явное ограничение ширины или высоты элемента управления.

Представьте, что происходит, когда вы устанавливаете изображение в UIImageView, если изображение намного больше экрана. Обычно вы хотите, чтобы изображения отображали фиксированную ширину и высоту и масштабировали контент, если вы не хотите, чтобы представление изменялось до размеров изображения.

Ссылка: https://www.raywenderlich.com/115444/auto-layout-tutorial-in-ios-9-part-2-constraints

Прочитайте Сжатие содержимого / сжатие содержимого в автозапуске онлайн: https://riptutorial.com/ru/ios/topic/6899/сжатие-содержимого---сжатие-содержимого-в-автозапуске

глава 188: Симулятор строит

Вступление

Где найти сборку симулятора?

Перейти к ~ / Библиотека / Разработчик / CoreSimulator / Devices /

Вы найдете каталоги с алфавитно-цифровыми именами

затем нажмите на один из каталогов и сделайте следующий выбор.

Данные / Контейнеры / Bundle / Application /

Снова вы найдете каталоги с буквенно-цифровыми именами, если вы нажмете на них, которые вы найдете

Симулятор строит там

Замечания:

Установка устройства iOS на симулятор не будет работать.

Симулятор iPhone использует архитектуру i386. Создается симулятор iPad. Использует x8

Examples

Установка сборки вручную на тренажере

```
xcrun simctl install booted *.app
```

Прочитайте Симулятор строит онлайн: https://riptutorial.com/ru/ios/topic/9813/симулятор-строит

глава 189: Сканер QR-кода

Вступление

QR (Quick Response) - это двумерные штрих-коды, которые широко используются на машиночитаемых оптических меток. iOS действительно предоставляет возможность читать QR-коды, используя систему `AVFoundation` от iOS 7 и далее. Эта структура предоставляет набор API для настройки / открытия камеры и чтения QR-кодов из канала камеры.

Examples

Проверка UIViewController для QR и отображение видеовхода

```
import AVFoundation
class QRScannerViewController: UIViewController,
     AVCaptureMetadataOutputObjectsDelegate {

    func viewDidLoad() {
        self.initCaptureSession()
    }

    private func initCaptureSession() {
        let captureDevice = AVCaptureDevice
            .defaultDevice(withMediaType: AVMediaTypeVideo)
        do {
            let input = try AVCaptureDeviceInput(device: captureDevice)
            let captureMetadataOutput = AVCaptureMetadataOutput()
            self.captureSession?.addOutput(captureMetadataOutput)
            captureMetadataOutput.setMetadataObjectsDelegate(self,
                 queue: DispatchQueue.main)
            captureMetadataOutput
                .metadataObjectTypes = [AVMetadataObjectTypeQRCode]

            self.videoPreviewLayer =
                AVCaptureVideoPreviewLayer(session: self.captureSession)
            self.videoPreviewLayer?
                .videoGravity = AVLayerVideoGravityResizeAspectFill
            self.videoPreviewLayer?.frame =
                self.view.layer.bounds

            self._viewController?.view.layer
                .addSublayer(videoPreviewLayer!)
            self.captureSession?.startRunning()
        } catch {
            //TODO: handle input open error
        }
    }
    private func dismissCaptureSession() {
        if let running = self.captureSession?.isRunning, running {
            self.captureSession?.stopRunning()
        }
        self.captureSession = nil
        self.videoPreviewLayer?.removeFromSuperLayer()
```

```
        self.videoPreviewLayer = nil
    }

    func captureOutput(_ captureOutput: AVCaptureOutput,
        didOutputMetadataObjects metadataObjects: [Any]!,
        from connection: AVCaptureConnection) {
        guard metadataObjects != nil && metadataObjects.count != 0 else {
            //Nothing captured
            return
        }

        if let metadataObj =
            metadataObjects[0] as? AVMetadataMachineReadableCodeObject {
            guard metadataObj.type == AVMetadataObjectTypeQRCode else {
                return
            }

            let barCodeObject = videoPreviewLayer?
                .transformedMetadataObject(for:
                    metadataObj as AVMetadataMachineReadableCodeObject)
                 as! AVMetadataMachineReadableCodeObject

            if let qrValue = metadataObj.stringValue {
                self.handleQRRead(value: qrValue)
            }
        }
    }

    private handleQRRead(value: String) {
        //TODO: Handle the read qr
    }
    private captureSession: AVCaptureSession?
    private videoPreviewLayer: AVCaptureVideo
}
```

`handleQRRead` - будет вызываться при успешном сканировании `initCaptureSession` - инициализировать сканирование для QR и ввода с камеры `dismissCaptureSession` - скрыть ввод камеры и остановить сканирование

Сканирование QR-кода с помощью системы AVFoudation

До iOS 7, когда вы хотите отсканировать QR-код, нам, возможно, придется полагаться на сторонние структуры или библиотеки, такие как zBar или zXing . Но Apple представила `AVCaptureMetaDataOutput` из iOS 7 для чтения штрих-кодов.

Чтобы прочитать QR-код, используя `AVFoundation` нам нужно настроить / создать `AVCaptureSession` и использовать `captureOutput:didOutputMetadataObjects:fromConnection:` delegate.

Шаг 1

Импортируйте `AVFoundation` и подтвердите в `AVCaptureMetadataOutputObjectsDelegate` протокол

```
import AVFoundation
class ViewController: UIViewController, AVCaptureMetadataOutputObjectsDelegate
```

Шаг 2

Чтение QR-кода полностью основано на захвате видео. Таким образом, для захвата непрерывного видео создайте `AVCaptureSession` и настройте вход и выход устройства. Добавьте приведенный ниже код в виде контроллера `viewDidLoad`

```
// Create an instance of the AVCaptureDevice and provide the video as the media type
parameter.
let captureDevice = AVCaptureDevice.defaultDevice(withMediaType: AVMediaTypeVideo)

do {
    // Create an instance of the AVCaptureDeviceInput class using the device object and
intialise capture session
    let input = try AVCaptureDeviceInput(device: captureDevice)
    captureSession = AVCaptureSession()
    captureSession?.addInput(input)

    // Create a instance of AVCaptureMetadataOutput object and set it as the output device the
capture session.
    let captureMetadataOutput = AVCaptureMetadataOutput()
    captureSession?.addOutput(captureMetadataOutput)
    // Set delegate with a default dispatch queue
    captureMetadataOutput.setMetadataObjectsDelegate(self, queue: DispatchQueue.main)
    //set meta data object type as QR code, here we can add more then one type as well
    captureMetadataOutput.metadataObjectTypes = [AVMetadataObjectTypeQRCode]


    // Initialize the video preview layer and add it as a sublayer to the viewcontroller
view's layer.
    videoPreviewLayer = AVCaptureVideoPreviewLayer(session: captureSession)
    videoPreviewLayer?.videoGravity = AVLayerVideoGravityResizeAspectFill
    videoPreviewLayer?.frame = view.layer.bounds
    view.layer.addSublayer(videoPreviewLayer!)

    // Start capture session.
    captureSession?.startRunning()
} catch {
    // If any error occurs, let the user know. For the example purpose just print out the
error
    print(error)
    return
}
```

Шаг 3

Внедрить `AVCaptureMetadataOutputObjectsDelegate` делегата `AVCaptureMetadataOutputObjectsDelegate` для чтения QR-кода

```
func captureOutput(_ captureOutput: AVCaptureOutput!, didOutputMetadataObjects
```

```
metadataObjects: [Any]!, from connection: AVCaptureConnection!) {

    // Check if the metadataObjects array contains at least one object. If not no QR code is
in our video capture
    if metadataObjects == nil || metadataObjects.count == 0 {
        // NO QR code is being detected.
        return
    }

    // Get the metadata object and cast it to `AVMetadataMachineReadableCodeObject`
    let metadataObj = metadataObjects[0] as! AVMetadataMachineReadableCodeObject

    if metadataObj.type == AVMetadataObjectTypeQRCode {
        // If the found metadata is equal to the QR code metadata then get the string value
from meta data
        let barCodeObject = videoPreviewLayer?.transformedMetadataObject(for: metadataObj)

        if metadataObj.stringValue != nil {
           // metadataObj.stringValue is our QR code
        }
    }
}
```

здесь объект метаданных может предоставить вам также QR-код, который читается на канале камеры. Чтобы получить оценки, просто передайте объект метаданных в метод `transformedMetadataObject videoPreviewLayer` ниже.

```
let barCodeObject = videoPreviewLayer?.transformedMetadataObject(for: metadataObj)
        qrCodeFrameView?.frame = barCodeObject!.bounds
```

Прочитайте Сканер QR-кода онлайн: https://riptutorial.com/ru/ios/topic/7963/сканер-qr-кода

глава 190: Снимок UIView

Examples

Получение снимка

```
- (UIImage *)getSnapshot
{
UIScreen *screen = [UIScreen mainScreen];
CGRect bounds = [self.view bounds];
UIGraphicsBeginImageContextWithOptions(bounds.size, false, screen.scale);
CGContextRef context = UIGraphicsGetCurrentContext();
CGContextSetInterpolationQuality(context, kCGInterpolationHigh);
[self.view drawViewHierarchyInRect:bounds afterScreenUpdates:YES];
UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
return image;
}
```

стриж

```
var screenshot: UIImage
{
UIGraphicsBeginImageContext(self.bounds.size);
let context = UIGraphicsGetCurrentContext();
self.layer.render(in: context)
let screenShot = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
return screenShot
}
```

Снимок с подвью с другой разметкой и текстом

- Поддержка портрета и пейзажа обоих типов изображения
- Рисование и другие подвид могут быть объединены в моем случае. Я добавляю ярлык для рисования

```
{
    CGSize fullSize = getImageForEdit.size;
    CGSize sizeInView = AVMakeRectWithAspectRatioInsideRect(imgViewFake.image.size,
imgViewFake.bounds).size;
    CGFloat orgScale = orgScale = fullSize.width/sizeInView.width;
    CGSize newSize = CGSizeMake(orgScale * img.image.size.width, orgScale *
img.image.size.height);
    if(newSize.width <= fullSize.width && newSize.height <= fullSize.height){
        newSize = fullSize;
    }
    CGRect offsetRect;
    if (getImageForEdit.size.height > getImageForEdit.size.width){
        CGFloat scale = newSize.height/fullSize.height;
        CGFloat offset = (newSize.width - fullSize.width*scale)/2;
```

```
        offsetRect = CGRectMake(offset, 0, newSize.width-offset*2, newSize.height);
    }
    else{
        CGFloat scale = newSize.width/fullSize.width;
        CGFloat offset = (newSize.height - fullSize.height*scale)/2;
        offsetRect = CGRectMake(0, offset, newSize.width, newSize.height-offset*2);
    }
    UIGraphicsBeginImageContextWithOptions(newSize, NO, getImageForEdit.scale);
    [getImageForEdit drawAtPoint:offsetRect.origin];
    //        [img.image drawInRect:CGRectMake(0,0,newSize.width,newSize.height)];
    CGFloat oldScale = img.contentScaleFactor;
    img.contentScaleFactor = getImageForEdit.scale;
    [img drawViewHierarchyInRect:CGRectMake(0, 0, newSize.width, newSize.height)
afterScreenUpdates:YES];
    img.contentScaleFactor = oldScale;
    UIImage *combImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    imageData = UIImageJPEGRepresentation(combImage, 1);
}
```

Прочитайте Снимок UIView онлайн: https://riptutorial.com/ru/ios/topic/4622/снимок-uiview

глава 191: совпадение

Вступление

Связанная тема: Grand Central Dispatch

Синтаксис

- dispatch_async - запускает блок кода в отдельной очереди и не останавливает текущую очередь. Если очередь находится в другом потоке, чем тот, на который был вызван метод dispatch_async, код в блоке будет работать во время кода после того, как dispatch_async также будет запущен
- dispatch_sync - Выполняет блок кода в отдельной очереди, и *не* остановить текущую очередь. Если очередь находится в другом потоке, чем тот, на который был вызван метод dispatch_async, будет выполняться код в блоке, а выполнение в потоке, где был вызван метод, будет возобновляться только после завершения

параметры

очередь	**Очередь, в которой будет выполняться код в блоке отправки. *Очередь* похожа (но не совсем такая же, как) на поток; код в разных очередях может выполняться параллельно. Используйте `dispatch_get_main_queue` чтобы получить очередь для основного потока. Чтобы создать новую очередь, которая, в свою очередь, создает новый поток, используйте `dispatch_queue_create("QUEUE_NAME", DISPATCH_QUEUE_CONCURRENT)` . Первый параметр - это имя очереди, которое отображается в отладчике, если вы приостановили пока блок все еще работает. Второй параметр не имеет значения, если вы не хотите использовать одну и ту же очередь для нескольких вызовов `dispatch_async` или `dispatch_sync` . В нем описывается, что происходит, когда другой блок помещается в одну очередь; `DISPATCH_QUEUE_CONCURRENT` вызовет запуск обоих блоков одновременно, в то время как `DISPATCH_QUEUE_SERIAL` заставит второй блок ждать завершения первого блока**
блок	Код в этом блоке будет запущен в очереди `queue` ; введите код, который вы хотите запустить в отдельной очереди. Полезный совет: если вы пишете это в Xcode, а аргумент блока имеет синий контур вокруг него, дважды щелкните по аргументу, и Xcode автоматически сделает пустой блок (это относится ко всем блочным аргументам в любой функции или методе)

замечания

Всякий раз, когда вы делаете что-то в отдельном потоке, который возникает при использовании очередей, важно поддерживать безопасность потоков. Некоторые методы, в частности, для `UIView` s, могут не работать и / или сбой в потоках, отличных от основного потока. Кроме того, убедитесь, что вы ничего не меняете (переменные, свойства и т. Д.), Которые также используются в основном потоке, если вы не учитываете это изменение

Examples

Запуск кода одновременно - Запуск кода при запуске другого кода

Скажите, что вы хотите выполнить в действии (в этом случае записывать «Foo»), делая что-то еще (записывая «Bar»). Обычно, если вы не используете параллелизм, одно из этих действий будет полностью выполнено, а другой запуск будет выполняться только после того, как он будет полностью завершен. Но с параллелизмом вы можете одновременно запускать оба действия:

```
dispatch_async(dispatch_queue_create("Foo", DISPATCH_QUEUE_CONCURRENT), ^{
    for (int i = 0; i < 100; i++) {
        NSLog(@"Foo");
        usleep(100000);
    }
});

for (int i = 0; i < 100; i++) {
    NSLog(@"Bar");
    usleep(50000);
}
```

Это будет записывать «Foo» 100 раз, приостанавливаясь на 100 мсек каждый раз, когда он регистрируется, но он будет делать все это в отдельном потоке. В то время как `Foo` регистрируется, «Bar» также будет регистрироваться через 50 мс в одно и то же время. В идеале вы должны увидеть результат с сочетаниями «Foo» и «Bars»

Выполнение основной темы

При асинхронном выполнении задач обычно возникает необходимость обеспечить выполнение части кода в основном потоке. Например, вам может понадобиться асинхронно удалять REST API, но поместить результат в UILabel на экране. Перед обновлением UILabel вы должны убедиться, что ваш код запущен в основном потоке:

```
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
    //Perform expensive tasks
    //...

    //Now before updating the UI, ensure we are back on the main thread
```

```
    dispatch_async(dispatch_get_main_queue(), ^{
        label.text = //....
    });
}
```

Всякий раз, когда вы обновляете представления на экране, всегда убедитесь, что вы делаете это в основном потоке, иначе может возникнуть неопределенное поведение.

Диспетчерская группа - ждет завершения других потоков.

```
dispatch_group_t preapreWaitingGroup = dispatch_group_create();

dispatch_group_enter(preapreWaitingGroup);
[self doAsynchronousTaskWithComplete:^(id someResults, NSError *error) {
    // Notify that this task has been completed.
    dispatch_group_leave(preapreWaitingGroup);
}]

dispatch_group_enter(preapreWaitingGroup);
[self doOtherAsynchronousTaskWithComplete:^(id someResults, NSError *error) {
    dispatch_group_leave(preapreWaitingGroup);
}]

dispatch_group_notify(preapreWaitingGroup, dispatch_get_main_queue(), ^{
    // This block will be executed once all above threads completed and call
dispatch_group_leave
    NSLog(@"Prepare completed. I'm readyyyy");
});
```

Обновление 1. Версия Swift 3.

```
let prepareGroup = DispatchGroup()
prepareGroup.enter()
doAsynchronousTaskWithComplete() { (someResults, error) in
    // Notify that this task has been completed.
    prepareGroup.leave()
}

prepareGroup.enter()
doOtherAsynchronousTaskWithComplete() { (someResults, error) in
    // Notify that this task has been completed.
    prepareGroup.leave()
}

prepareGroup.notify(queue: DispatchQueue.main) {
    // This block will be executed once all above threads completed and call
dispatch_group_leave
    print("Prepare completed. I'm readyyyy")
}
```

Прочитайте совпадение онлайн: https://riptutorial.com/ru/ios/topic/1090/совпадение

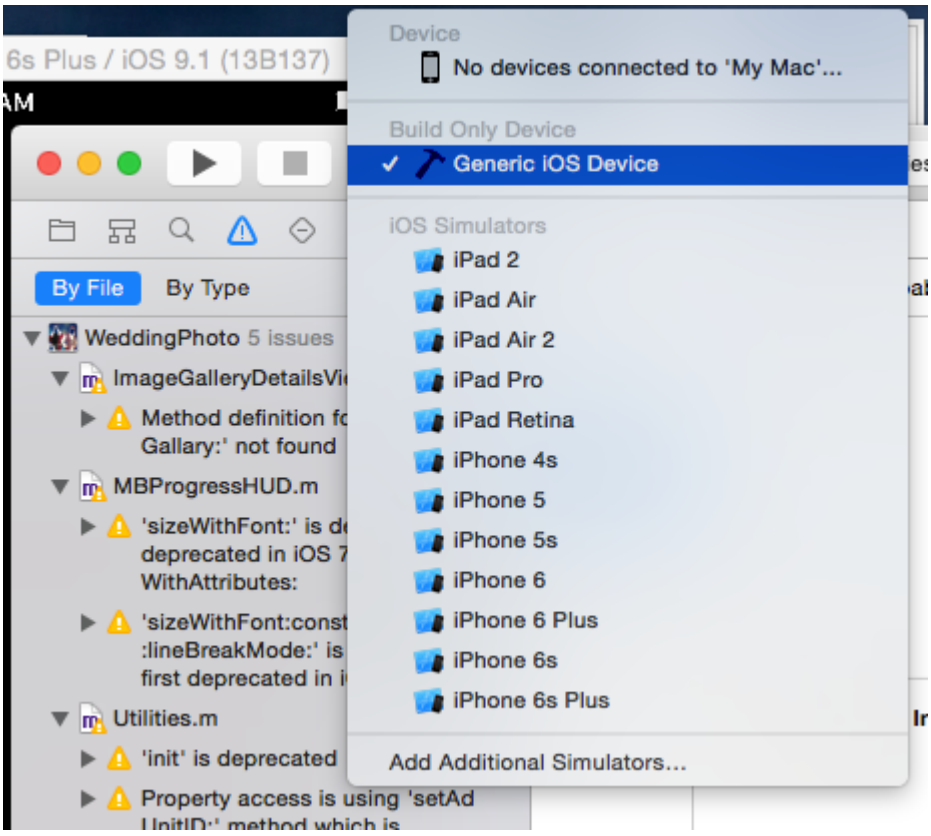
глава 192: Создайте файл .ipa для загрузки в appstore с помощью Applicationloader

Examples

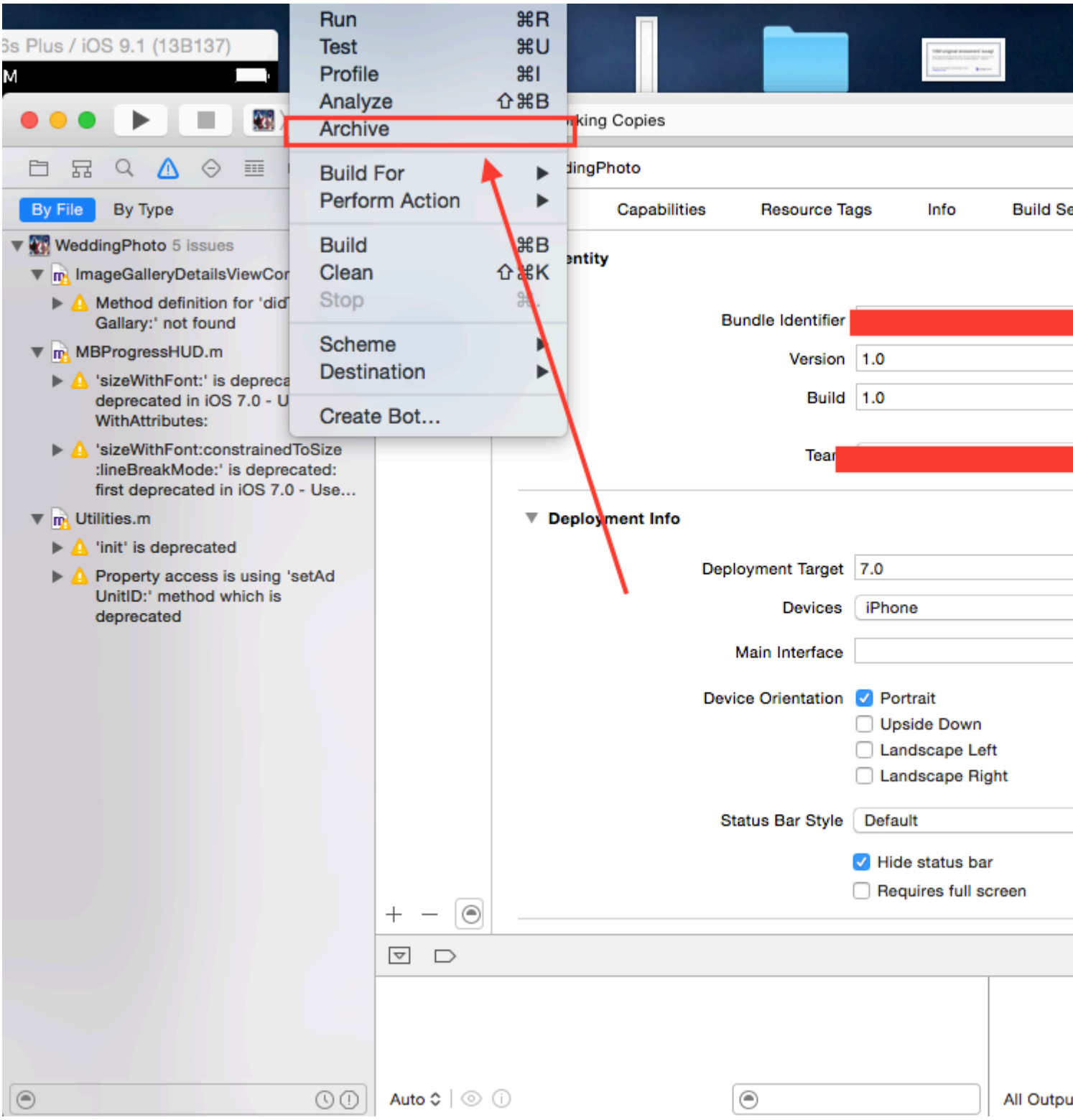
создать файл .ipa для загрузки приложения в appstore с помощью Application Loader

Если вы хотите загрузить файл .ipa в itunesconnect **без интеграции учетной записи разработчика в Xcode,** и вы хотите использовать **загрузчик приложений** . то вы можете **сгенерировать .ipa с iTunes** .

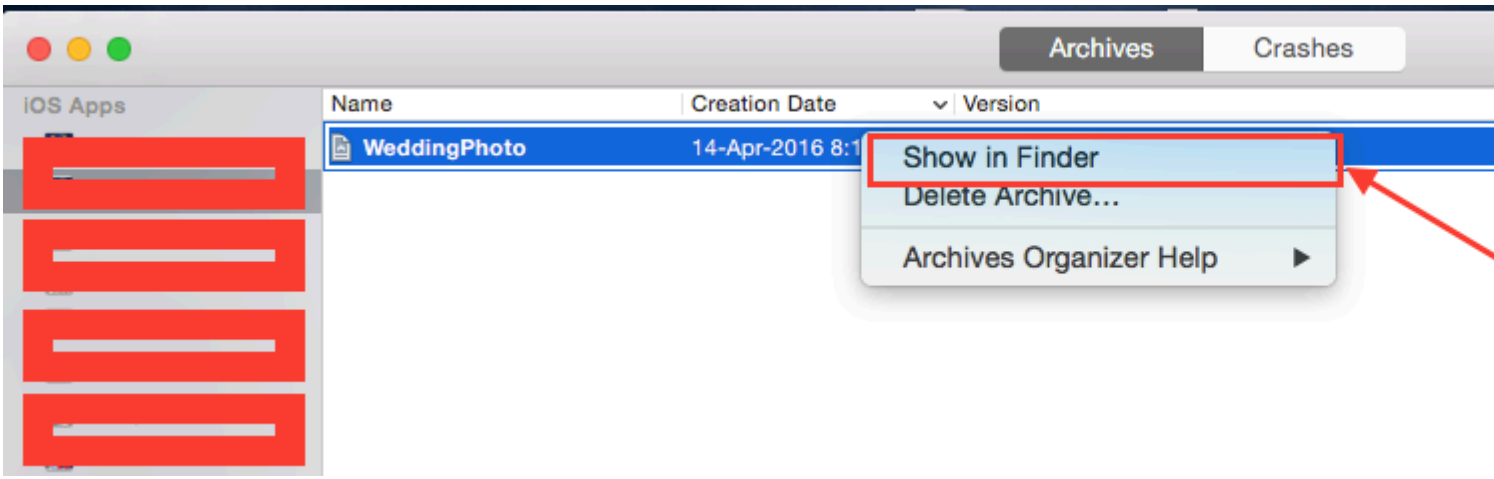
Шаг 1: - Выберите устройство на месте симулятора.



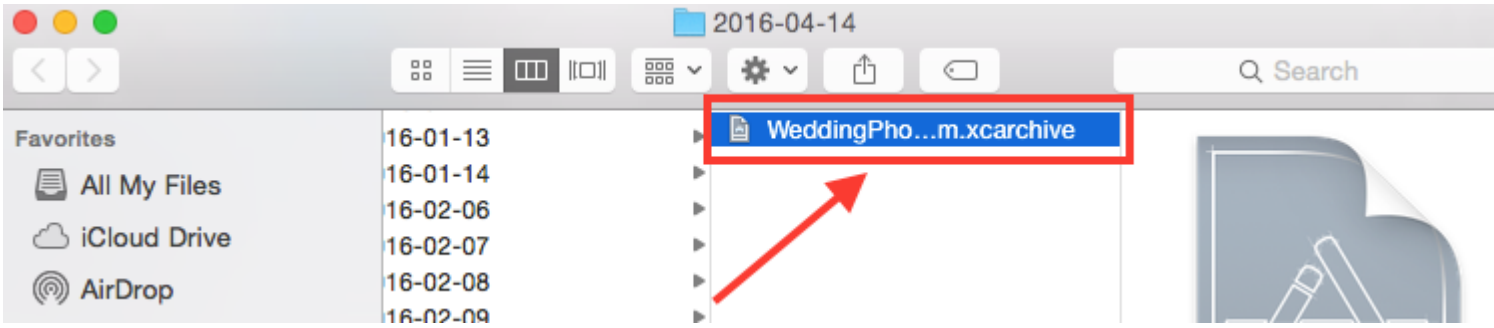
Шаг 2: - Перейти к продукту -> выбрать архив



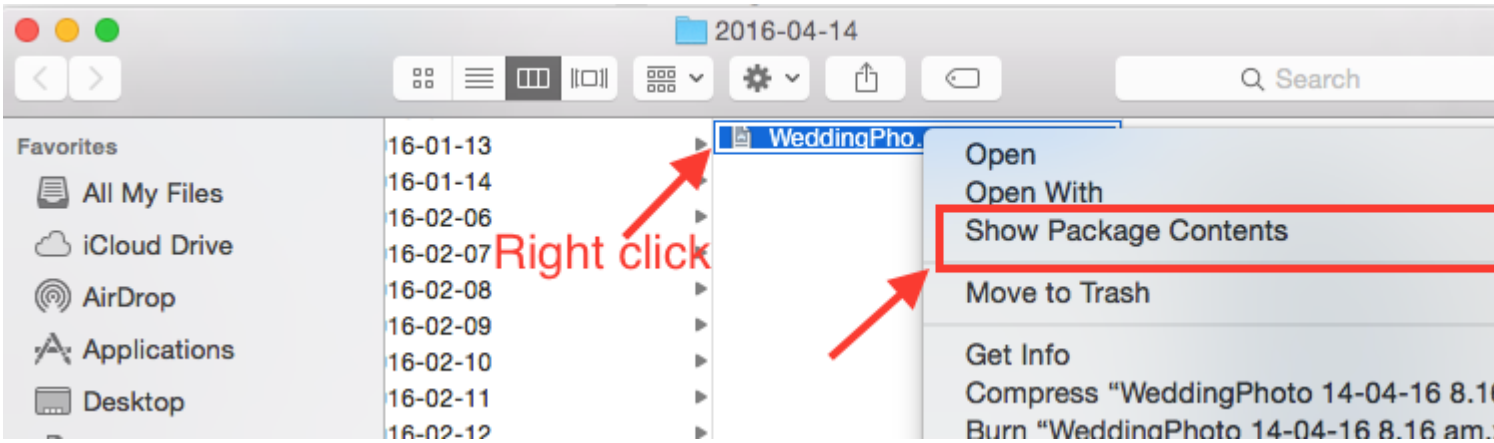
Шаг 3: После успешного завершения процесса щелкните правой кнопкой мыши на вашем архиве -> и выберите шоу в Finder



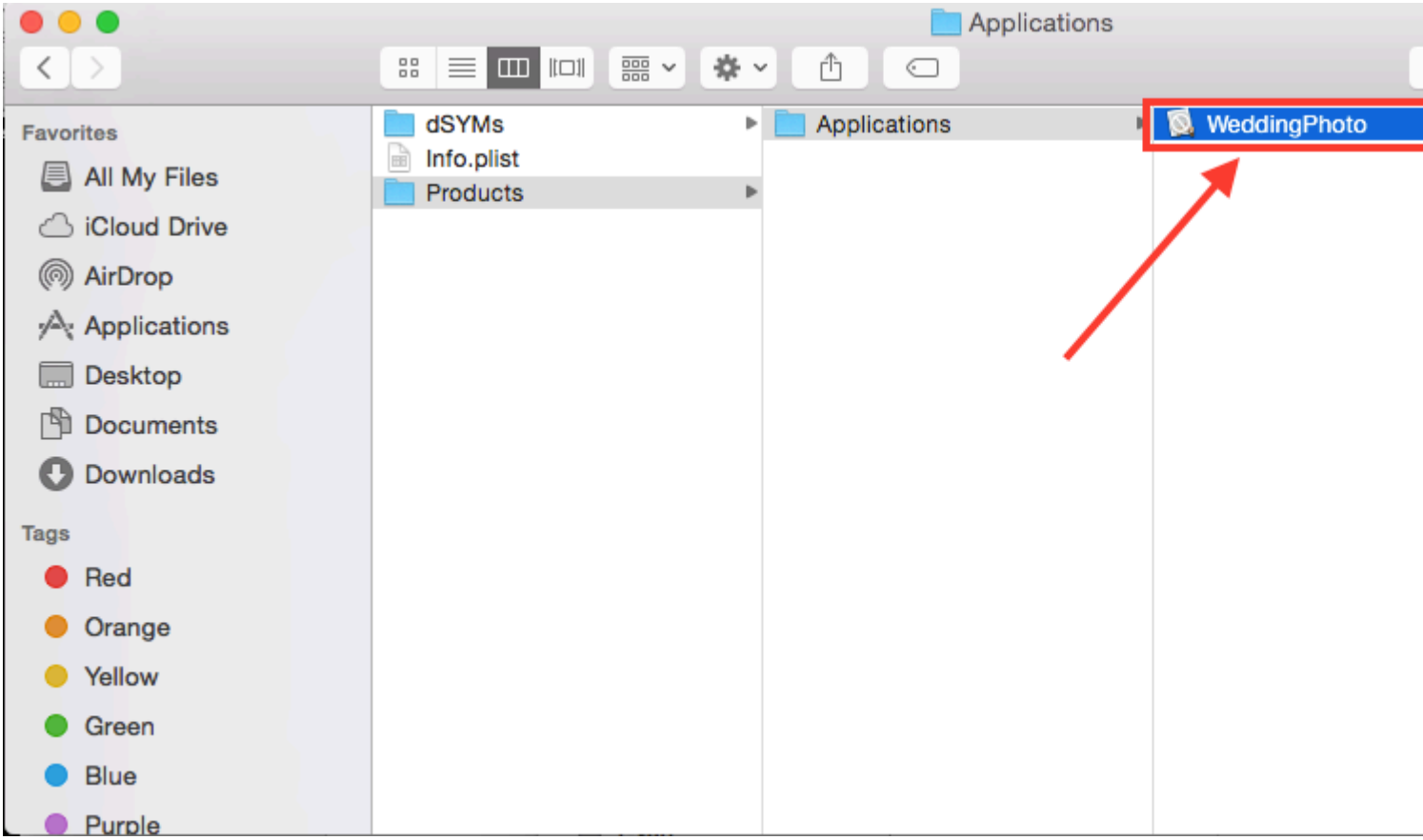
Шаг 4: - когда вы нажимаете на шоу в finder, вы переадресовываете его в папку Archive, выглядит так:



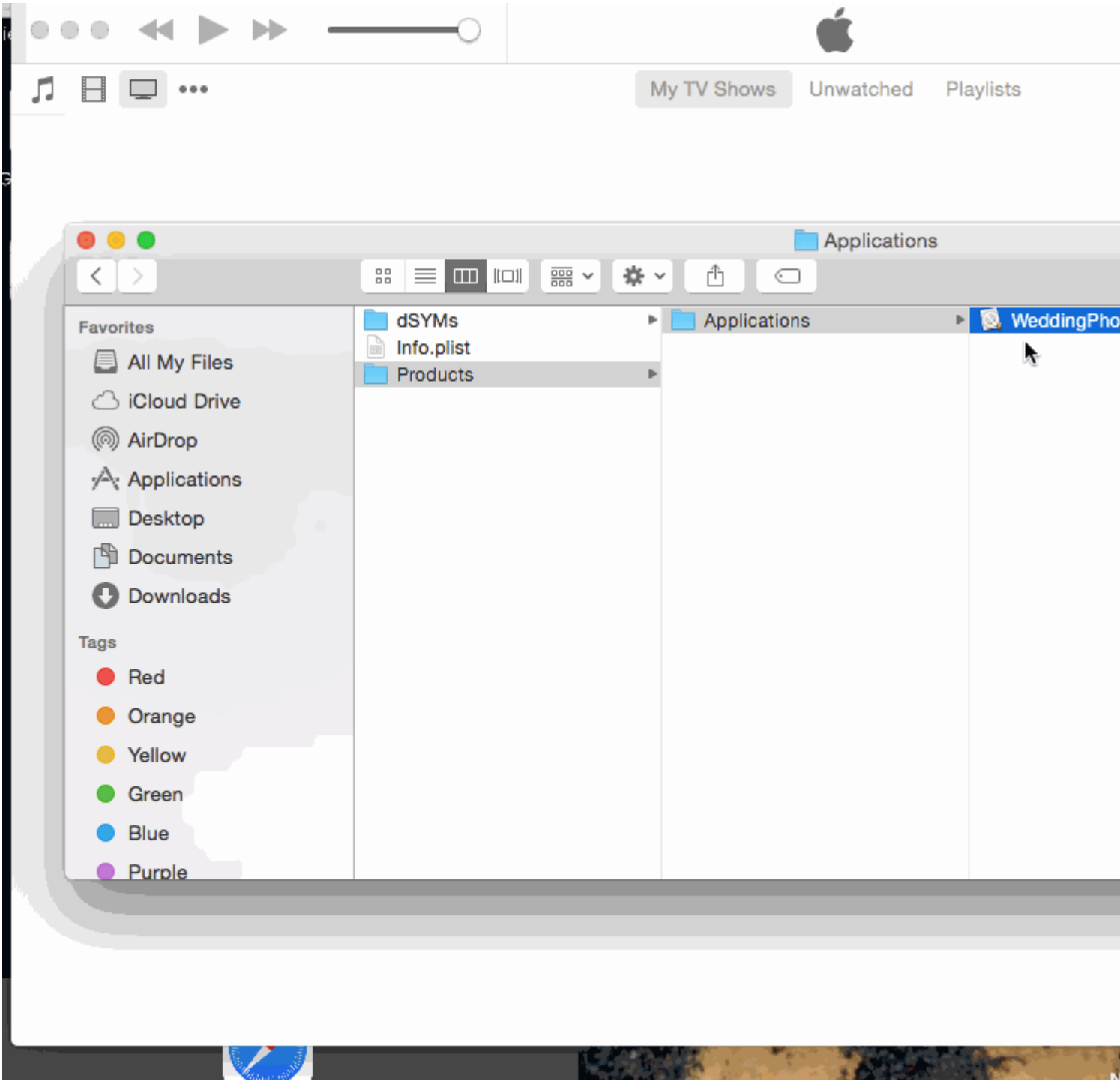
Шаг 5: - Щелкните правой кнопкой мыши на .xarchive file -> выберите «Показать в опции поиска».



Шаг 6: - Перейдите в папку продуктов -> Папка приложения -> Вы найдете свой проект project.app



Шаг 7: - Теперь для преобразования .app в .ipa просто перетащите его в itunes. проверьте изображение ниже,



Шаг 8: - Теперь поместите этот .ipa-файл в безопасное место и используйте при загрузке с загрузчиком приложений.

Примечание. Если вы хотите узнать, как загрузить приложение с загрузчиком приложений, проверьте это,

Загрузить приложение с приложением Loader

РЕДАКТИРОВАТЬ :-

ПРЕДУПРЕЖДЕНИЕ: - Не делайте .ipa с изменением расширения с .aap на .zip и .zip на . ipa.

Я видел во многих ответах, что они предлагают сжать файл .app, а затем изменить расширение с .zip на .ipa. Сейчас он не работает. С помощью этого метода вы получите ошибку, например,

> IPA недействителен, он не содержит каталог полезной нагрузки.

Прочитайте Создайте файл .ipa для загрузки в appstore с помощью Applicationloader онлайн: https://riptutorial.com/ru/ios/topic/6119/создайте-файл--ipa-для-загрузки-в-appstore-с-помощью-applicationloader

глава 193: Создание PDF в iOS

Examples

Создать PDF-файл

```
UIGraphicsBeginPDFContextToFile(fileName, CGRectZero, nil);

UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 612, 792), nil);

[self drawText];

UIGraphicsEndPDFContext();
```

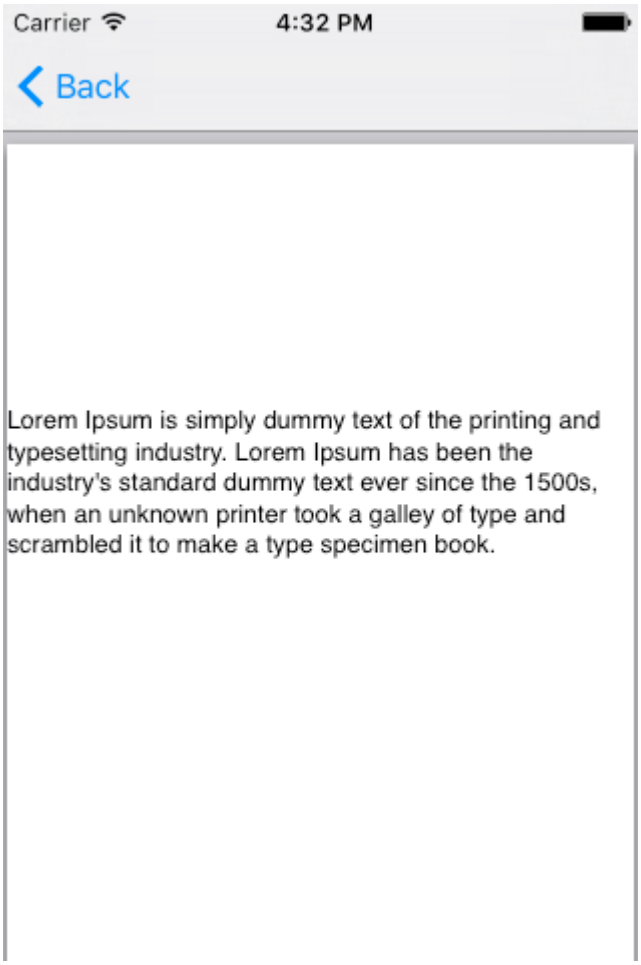
fileName - это файл документа, в который вы собираетесь добавлять или прикреплять

```
 NSString* temporaryFile = @"firstIOS.PDF";
    NSArray *arrayPaths =
    NSSearchPathForDirectoriesInDomains(
                                        NSDocumentDirectory,
                                        NSUserDomainMask,
                                        YES);

    NSString *path = [arrayPaths objectAtIndex:0];

    NSString* fileName = [path stringByAppendingPathComponent:fileName];
```

Где **drawText** является

```
   (void)drawText
{
    NSString* textToDraw = @"Lorem Ipsum is simply dummy text of the printing and typesetting
industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
an unknown printer took a galley of type and scrambled it to make a type specimen book.";

    CFStringRef stringRef = (__bridge CFStringRef)textToDraw;

    CFAttributedStringRef currentText = CFAttributedStringCreate(NULL, stringRef, NULL);

    CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedString(currentText);

    CGRect frameRect = CGRectMake(0, 0, 300, 100);

    CGMutablePathRef framePath = CGPathCreateMutable();

    CGPathAddRect(framePath, NULL, frameRect);

    CFRange currentRange = CFRangeMake(0, 0);

    CTFrameRef frameRef = CTFramesetterCreateFrame(framesetter, currentRange, framePath,
NULL);
    CGPathRelease(framePath);
```

```
    CGContextRef currentContext = UIGraphicsGetCurrentContext();


    CGContextSetTextMatrix(currentContext, CGAffineTransformIdentity);


    CGContextTranslateCTM(currentContext, 0, 450);

    CGContextScaleCTM(currentContext, 2, -2);

    CTFrameDraw(frameRef, currentContext);

    CFRelease(frameRef);

    CFRelease(stringRef);

    CFRelease(framesetter);
}
```



Показать PDF

```
  NSString* fileName = @"firstIOS.PDF";


NSArray *arrayPaths =
NSSearchPathForDirectoriesInDomains(
                                    NSDocumentDirectory,
                                    NSUserDomainMask,
```

```
                                    YES);

NSString *path = [arrayPaths objectAtIndex:0];

NSString* pdfFileName = [path stringByAppendingPathComponent:fileName];

UIWebView* webView = [[UIWebView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];

NSURL *url = [NSURL fileURLWithPath:pdfFileName];

NSURLRequest *request = [NSURLRequest requestWithURL:url];


[webView setScalesPageToFit:YES];

[webView loadRequest:request];

[self.view addSubview:webView];
```

Многостраничный PDF-файл

```
 UIGraphicsBeginPDFContextToFile(fileName, CGRectZero, nil);

    UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 600, 792), nil);

    UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 600, 792), nil);

    UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 600, 792), nil);

    UIGraphicsEndPDFContext();
```

Создайте PDF-документ из любого документа Microsoft, загруженного в UIWebview

```
#define kPaperSizeA4 CGSizeMake(595.2,841.8)
```

Прежде всего реализуйте протокол UIPrintPageRenderer

```
@interface UIPrintPageRenderer (PDF)

- (NSData*) printToPDF;

@end

@implementation UIPrintPageRenderer (PDF)

- (NSData*) printToPDF
{
    NSMutableData *pdfData = [NSMutableData data];
    UIGraphicsBeginPDFContextToData( pdfData, self.paperRect, nil );
    [self prepareForDrawingPages: NSMakeRange(0, self.numberOfPages)];
    CGRect bounds = UIGraphicsGetPDFContextBounds();
    for ( int i = 0 ; i < self.numberOfPages ; i++ )
    {
        UIGraphicsBeginPDFPage();
```

```
        [self drawPageAtIndex: i inRect: bounds];
    }
    UIGraphicsEndPDFContext();
    return pdfData;
}
@end
```

Затем вызовите метод ниже после завершения загрузки документа в `UIWebView`

```
-(void)createPDF:(UIWebView *)webView {

UIPrintPageRenderer *render = [[UIPrintPageRenderer alloc] init];
[render addPrintFormatter:webView.viewPrintFormatter startingAtPageAtIndex:0];

float padding = 10.0f;
CGRect paperRect = CGRectMake(0, 0, kPaperSizeA4.width, kPaperSizeA4.height);
CGRect printableRect = CGRectMake(padding, padding, kPaperSizeA4.width-(padding * 2),
kPaperSizeA4.height-(padding * 2));

[render setValue:[NSValue valueWithCGRect:paperRect] forKey:@"paperRect"];
[render setValue:[NSValue valueWithCGRect:printableRect] forKey:@"printableRect"];

NSData *pdfData = [render printToPDF];

dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{

    if (pdfData) {
        [pdfData writeToFile:directoryPath atomically: YES];
    }
    else
    {
        NSLog(@"PDF couldnot be created");
    }
});}
```

Прочитайте Создание PDF в iOS онлайн: https://riptutorial.com/ru/ios/topic/2416/создание-pdf-в-ios

глава 194: Создание видео с изображений

Вступление

Создайте видео с изображений с помощью AVFoundation

Examples

Создание видео с UIImages

Прежде всего вам нужно создать `AVAssetWriter`

```
NSError *error = nil;
NSURL *outputURL = <#NSURL object representing the URL where you want to save the video#>;
AVAssetWriter *assetWriter = [AVAssetWriter assetWriterWithURL:outputURL
fileType:AVFileTypeQuickTimeMovie error:&error];
if (!assetWriter) {
    // handle error
}
```

`AVAssetWriter` требуется, по крайней мере, один вход записи записи.

```
 NSDictionary *writerInputParams = [NSDictionary dictionaryWithObjectsAndKeys:
                                                AVVideoCodecH264, AVVideoCodecKey,
                                                [NSNumber numberWithInt:renderSize.width],
AVVideoWidthKey,
                                                [NSNumber numberWithInt:renderSize.height],
AVVideoHeightKey,
                                                AVVideoScalingModeResizeAspectFill,
AVVideoScalingModeKey,
                                                nil];

 AVAssetWriterInput *assetWriterInput = [AVAssetWriterInput
assetWriterInputWithMediaType:AVMediaTypeVideo outputSettings:writerInputParams];
 if ([assetWriter canAddInput:assetWriterInput]) {
     [assetWriter addInput:assetWriterInput];
 } else {
     // show error message
 }
```

Чтобы добавить `CVPixelBufferRef` в `AVAssetWriterInput` нам нужно создать `AVAssetWriterInputPixelBufferAdaptor`

```
NSDictionary *attributes = [NSDictionary dictionaryWithObjectsAndKeys:
                            [NSNumber numberWithUnsignedInt:kCVPixelFormatType_32ARGB],
(NSString*)kCVPixelBufferPixelFormatTypeKey,
                            [NSNumber numberWithBool:YES], (NSString
*)kCVPixelBufferCGImageCompatibilityKey,
                            [NSNumber numberWithBool:YES], (NSString
*)kCVPixelBufferCGBitmapContextCompatibilityKey,
                           nil];
```

```
AVAssetWriterInputPixelBufferAdaptor *writerAdaptor = [AVAssetWriterInputPixelBufferAdaptor
assetWriterInputPixelBufferAdaptorWithAssetWriterInput:assetWriterInput
sourcePixelBufferAttributes:attributes];
```

Теперь мы можем начать писать

```
[assetWriter startWriting];
[assetWriter startSessionAtSourceTime:kCMTimeZero];
[assetWriterInput requestMediaDataWhenReadyOnQueue:exportingQueue usingBlock:^{
    for (int i = 0; i < images.count; ++i) {
        while (![assetWriterInput isReadyForMoreMediaData]) {
            [NSThread sleepForTimeInterval:0.01];
            // can check for attempts not to create an infinite loop
        }

        UIImage *uIImage = images[i];

        CVPixelBufferRef buffer = NULL;
        CVReturn err = PixelBufferCreateFromImage(uIImage.CGImage, &buffer);
        if (err) {
            // handle error
        }

        // frame duration is duration of single image in seconds
        CMTime presentationTime = CMTimeMakeWithSeconds(i * frameDuration, 1000000);

        [writerAdaptor appendPixelBuffer:buffer withPresentationTime:presentationTime];

        CVPixelBufferRelease(buffer);
    }

    [assetWriterInput markAsFinished];
    [assetWriter finishWritingWithCompletionHandler:^{
        if (assetWriter.error) {
            // show error message
        } else {
            // outputURL
        }
    }];
}];
```

Вот функция получения `CVPixelBufferRef` из `CGImageRef`

```
CVReturn PixelBufferCreateFromImage(CGImageRef imageRef, CVPixelBufferRef *outBuffer) {
    CIContext *context = [CIContext context];
    CIImage *ciImage = [CIImage imageWithCGImage:imageRef];

    NSDictionary *attributes = [NSDictionary dictionaryWithObjectsAndKeys:
                                [NSNumber numberWithBool:YES], (NSString
*)kCVPixelBufferCGBitmapContextCompatibilityKey,
                                [NSNumber numberWithBool:YES], (NSString
*)kCVPixelBufferCGImageCompatibilityKey
                                ,nil];

    CVReturn err = CVPixelBufferCreate(kCFAllocatorDefault, CGImageGetWidth(imageRef),
CGImageGetHeight(imageRef), kCVPixelFormatType_32ARGB, (__bridge CFDictionaryRef
_Nullable)(attributes), outBuffer);
    if (err) {
```

```
        return err;
    }

    if (outBuffer) {
        [context render:ciImage toCVPixelBuffer:*outBuffer];
    }

    return kCVReturnSuccess;
}
```

Прочитайте Создание видео с изображений онлайн: https://riptutorial.com/ru/ios/topic/10607/создание-видео-с-изображений

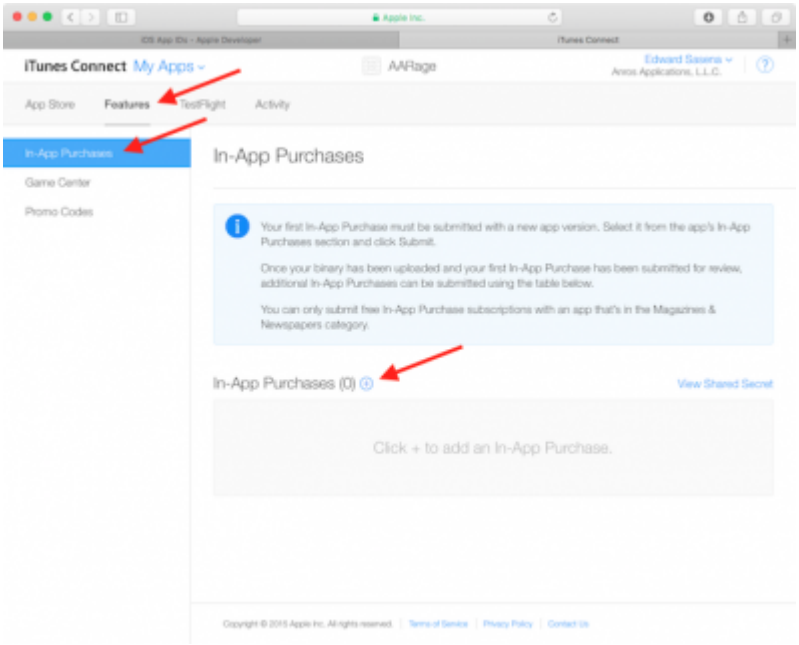
глава 195: Создание идентификатора приложения

Examples

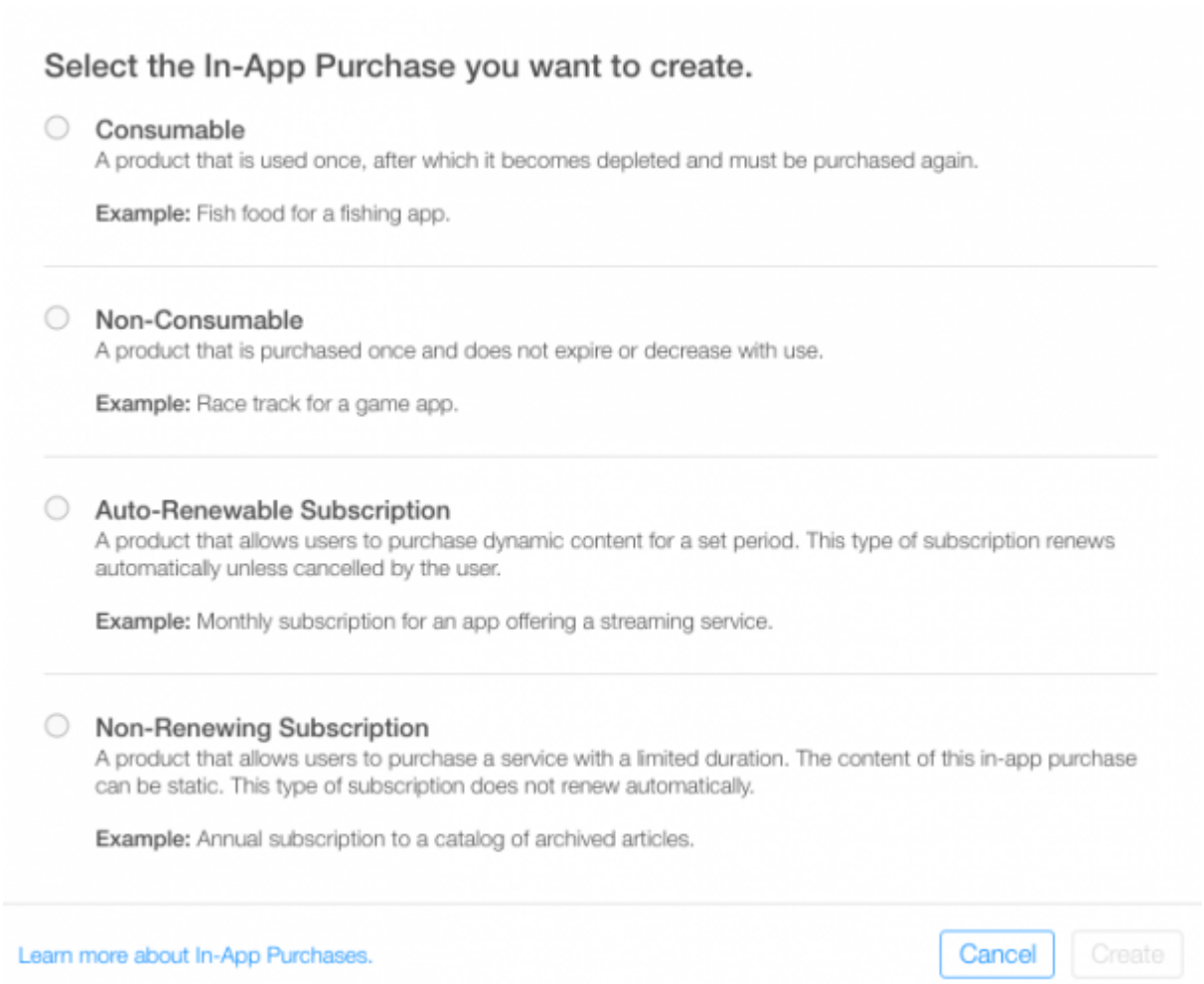
Создание продуктов для покупок в приложениях

- Предлагая IAP в приложении, вы должны сначала добавить запись для каждой отдельной покупки в iTunes Connect. Если вы когда-либо посещали приложение для продажи в магазине, это аналогичный процесс и включает такие вещи, как выбор уровня цен для покупки. Когда пользователь совершает покупку, App Store обрабатывает сложный процесс зарядки учетной записи пользователя iTunes. Есть целый набор различных типов IAP, которые вы можете добавить:
    - **Расходуемые** : их можно купить более одного раза и их можно использовать. Это такие вещи, как дополнительные жизни, внутриигровая валюта, временные бонусы и т. П.
    - **Non-Consumable** : то, что вы покупаете один раз, и ожидаете, что будете постоянно, например, дополнительные уровни и разблокируемый контент.
    - **Подписка на продление подписки** : контент, доступный в течение фиксированного периода времени.
    - **Подписка на автоматическое обновление** : повторяющаяся подписка, такая как ежемесячная подписка на raywenderlich.com.

Вы можете предлагать покупки в приложениях только для цифровых товаров, а не для физических товаров или услуг. Для получения дополнительной информации обо всем этом, ознакомьтесь с полной документацией Apple о создании продуктов для покупок в приложении. Теперь, просматривая запись своего приложения в iTunes Connect, перейдите на вкладку «Возможности» и выберите «Покупки в приложении». Чтобы добавить новый продукт IAP, нажмите + справа от покупки в приложении.



Появится следующее диалоговое окно:



Когда пользователь приобретает ярость комикса в вашем приложении, вы хотите, чтобы у них всегда был доступ к нему, поэтому выберите «Нерасходуемые» и нажмите «Создать». Затем заполните данные для IAP следующим образом:

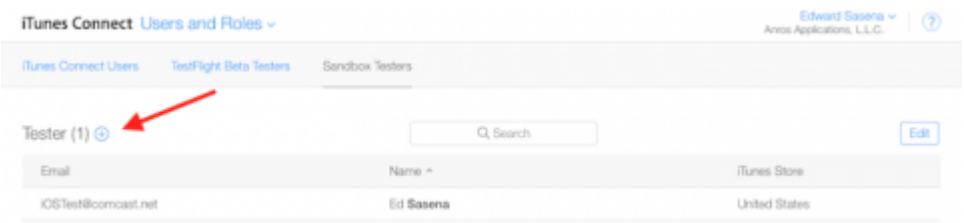
- **Имя ссылки** : псевдоним, идентифицирующий IAP в iTunes Connect. Это имя не отображается нигде в приложении. Название комикса, которое вы будете разблокировать с этой покупкой, - **«Girlfriend of Drummer»** , поэтому введите здесь.
- **Идентификатор продукта** : это уникальная строка, идентифицирующая IAP. Обычно лучше всего начинать с идентификатора Bundle, а затем добавлять уникальное имя, специфичное для этого приобретаемого предмета. Для этого урока убедитесь, что вы добавили «GirlfriendOfDrummerRage», так как это будет использоваться позже в приложении для поиска комикса, чтобы разблокировать. Итак, например: com.theNameYouPickedEarlier.Rage.GirlFriendOfDrummerRage.
- **Снято для продажи** : Включает или отключает продажу IAP. Вы хотите включить его!
- **Ценовой уровень** : стоимость IAP. Выберите Уровень 1.

Теперь прокрутите вниз до раздела «Локализация» и обратите внимание, что для английского (США) есть запись по умолчанию. Введите «Girlfriend of Drummer» как для отображаемого имени, так и для описания. Нажмите «Сохранить». Большой! Вы создали свой первый продукт IAP.

Перед тем, как вы сможете вникать в какой-то код, требуется еще один шаг. При тестировании покупок в приложении в приложении для разработки приложения Apple предоставляет тестовую среду, которая позволяет вам «покупать» ваши продукты IAP без создания финансовых транзакций.

Создание пользователя Sandbox

В iTunes Connect нажмите iTunes Connect в левом верхнем углу окна, чтобы вернуться в главное меню. Выберите «Пользователи и роли», затем щелкните вкладку «Тестеры песочников». Нажмите + рядом с заголовком «Тестер».



Заполните информацию и нажмите «Сохранить», когда вы закончите. Вы можете составить имя и фамилию для своего тестового пользователя, но выбранный адрес электронной почты должен быть настоящим адресом электронной почты, так как проверка будет отправлена по адресу Apple. После того, как вы получите это письмо, не забудьте нажать ссылку в нем, чтобы подтвердить свой адрес. Введенный вами адрес электронной почты также НЕ должен быть связан с учетной записью Apple ID. Подсказка: если у вас есть учетная запись gmail, вы можете просто использовать псевдоним адреса вместо создания новой учетной записи

Прочитайте Создание идентификатора приложения онлайн:
https://riptutorial.com/ru/ios/topic/10854/создание-идентификатора-приложения

глава 196: Создание пользовательской структуры в iOS

Examples

Создать структуру в Swift

выполните следующие действия для создания Custom Framework в Swift-IOS:

1. Создайте новый проект. В Xcode
2. Выберите iOS / Framework & Library / Cocoa Touch Framework для создания новой структуры
3. нажмите «Далее» и задайте имя продукта
4. нажмите далее и выберите каталог, чтобы создать проект там
5. добавить код и ресурсы в созданный проект

Рамка создана успешно

добавить созданный фреймворк в другой проект, сначала создать рабочую область добавить «целевой проект» и «рамочный проект» в рабочее пространство, а затем:

1. перейдите на общую вкладку целевого проекта
2. перетащите файл «* .framework» в папку продукта проекта framework в раздел «Встроенные двоичные файлы»
3. для использования в любом ViewController или классе просто импортируйте фреймворк в каждом файле

Прочитайте Создание пользовательской структуры в iOS онлайн:
https://riptutorial.com/ru/ios/topic/7331/создание-пользовательской-структуры-в-ios

глава 197: Структура контактов

замечания

Полезные ссылки

- Документация Apple
- Связанные вопросы о переполнении стека
- WWDC15 сеансовое видео

Examples

Авторизация доступа к контактам

Импорт структуры

стриж

```
import Contacts
```

Objective-C

```
#import <Contacts/Contacts.h>
```

Проверка доступности

стриж

```
switch CNContactStore.authorizationStatusForEntityType(CNEntityType.Contacts){
case .Authorized: //access contacts
case .Denied, .NotDetermined: //request permission
default: break
}
```

Objective-C

```
switch ([CNContactStore authorizationStatusForEntityType:CNEntityType.Contacts]){
case CNAuthorizationStatus.Authorized:
    //access contacts
    break;
case CNAuthorizationStatus.Denied:
    //request permission
    break;
case CNAuthorizationStatus.NotDetermined:
    //request permission
    break;
}
```

Запрос разрешения

стриж

```
var contactStore = CKContactStore()
contactStore.requestAccessForEntityType(CKEntityType.Contacts, completionHandler: { (ok, _) ->
Void in
    if access{
        //access contacts
    }
}
```

Доступ к контактам

Применение фильтра

Чтобы получить доступ к контактам, мы должны применить фильтр типа `NSPredicate` к нашей переменной contactStore, которую мы определили в примере авторизации контакта. Например, здесь мы хотим отсортировать контакты с совпадением имен с нашими:

стриж

```
let predicate = CNContact.predicateForContactsMatchingName("Some Name")
```

Objective-C

```
NSPredicate *predicate = [CNContact predicateForContactsMatchingName:@"Some Name"];
```

Указание ключей для извлечения

Здесь мы хотим получить имя, фамилию и профиль контакта.

стриж

```
let keys = [CNContactGivenNameKey, CNContactFamilyNameKey, CNContactImageDataKey]
```

Получение контактов

стриж

```
do {
    let contacts = try contactStore.unifiedContactsMatchingPredicate(predicate, keysToFetch:
keys)
} catch let error as NSError {
    //...
}
```

Доступ к контактным данным

стриж

```
print(contacts[0].givenName)
print(contacts[1].familyName)
let image = contacts[2].imageData
```

Добавление контакта

стриж

```
import Contacts

// Creating a mutable object to add to the contact
let contact = CNMutableContact()

contact.imageData = NSData() // The profile picture as a NSData object

contact.givenName = "John"
contact.familyName = "Appleseed"

let homeEmail = CNLabeledValue(label:CNLabelHome, value:"john@example.com")
let workEmail = CNLabeledValue(label:CNLabelWork, value:"j.appleseed@icloud.com")
contact.emailAddresses = [homeEmail, workEmail]

contact.phoneNumbers = [CNLabeledValue(
    label:CNLabelPhoneNumberiPhone,
    value:CNPhoneNumber(stringValue:"(408) 555-0126"))]
```

```
let homeAddress = CNMutablePostalAddress()
homeAddress.street = "1 Infinite Loop"
homeAddress.city = "Cupertino"
homeAddress.state = "CA"
homeAddress.postalCode = "95014"
contact.postalAddresses = [CNLabeledValue(label:CNLabelHome, value:homeAddress)]

let birthday = NSDateComponents()
birthday.day = 1
birthday.month = 4
birthday.year = 1988  // You can omit the year value for a yearless birthday
contact.birthday = birthday

// Saving the newly created contact
let store = CNContactStore()
let saveRequest = CNSaveRequest()
saveRequest.addContact(contact, toContainerWithIdentifier:nil)
try! store.executeSaveRequest(saveRequest)
```

Прочитайте Структура контактов онлайн: https://riptutorial.com/ru/ios/topic/5872/структура-контактов

глава 198: Текст UILabel подчеркивает

Examples

Подчеркивание текста в UILabel с использованием Objective C

```
UILabel *label=[[UILabel alloc]initWithFrame:CGRectMake(0, 0, 320, 480)];
label.backgroundColor=[UIColor lightGrayColor];
NSMutableAttributedString *attributedString;
attributedString = [[NSMutableAttributedString alloc] initWithString:@"Apply Underlining"];
[attributedString addAttribute:NSUnderlineStyleAttributeName value:@1 range:NSMakeRange(0,
[attributedString length])];
[label setAttributedText:attributedString];
```

Подчеркивание текста в UILabel с использованием Swift

```
 let label = UILabel.init(frame: CGRect(x: 0, y:0, width: 100, height: 40))
 label.backgroundColor = .lightGray
 let attributedString = NSMutableAttributedString.init(string: "Apply UnderLining")
 attributedString.addAttribute(NSUnderlineStyleAttributeName, value: 1, range:
NSRange.init(location: 0, length: attributedString.length))
 label.attributedText = attributedString
```

Прочитайте Текст UILabel подчеркивает онлайн: https://riptutorial.com/ru/ios/topic/7219/текст-uilabel-подчеркивает

глава 199: Тестирование пользовательского интерфейса

Синтаксис

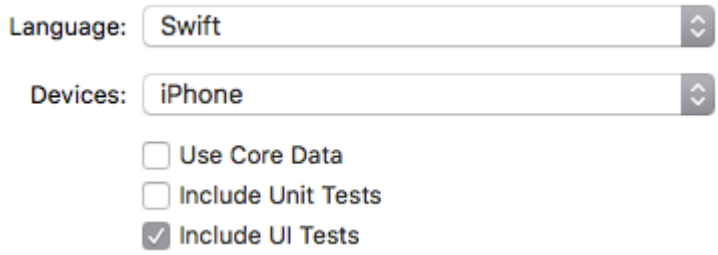
- XCUIApplication () // Прокси для приложения. Информация, идентифицирующая приложение, указана в целевых настройках Xcode как «Target Application».
- XCUIElement () // Элемент пользовательского интерфейса в приложении.

Examples

Добавление тестовых файлов в проект Xcode

При создании проекта

Вы должны проверить «Включить тесты пользовательского интерфейса» в диалоговом окне создания проекта.

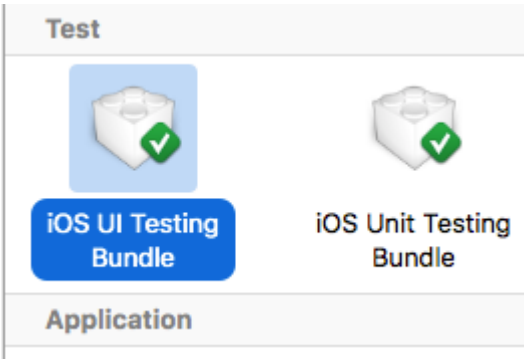


После создания проекта

Если вы пропустили проверку `UI target` при создании проекта, вы всегда можете добавить тестовый результат позже.

Setps:

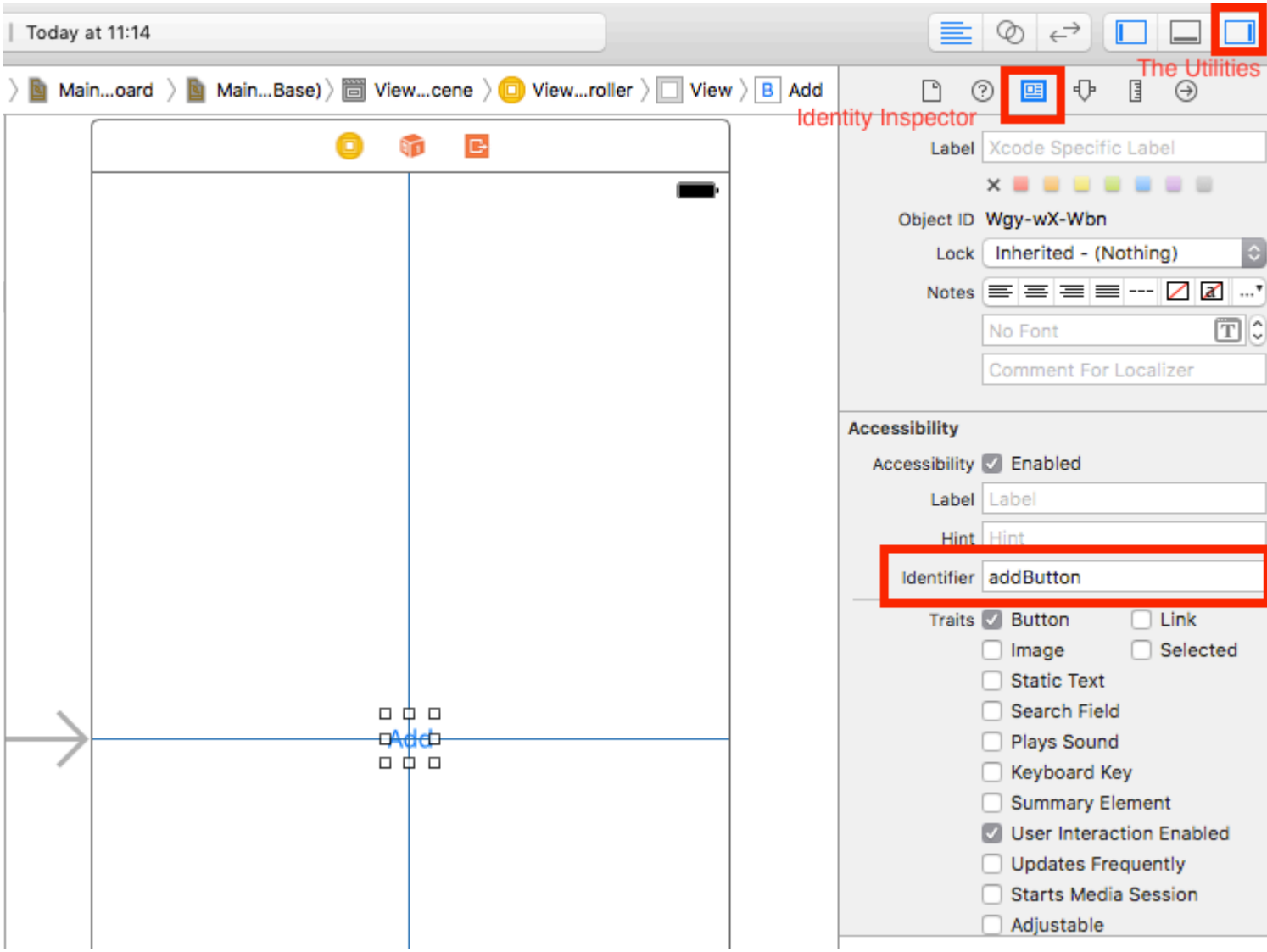
- Пока проект открыт, перейдите в `File` -> `New` -> `Target`
- Найдите `iOS UI Testing Bundle`



Идентификатор доступности

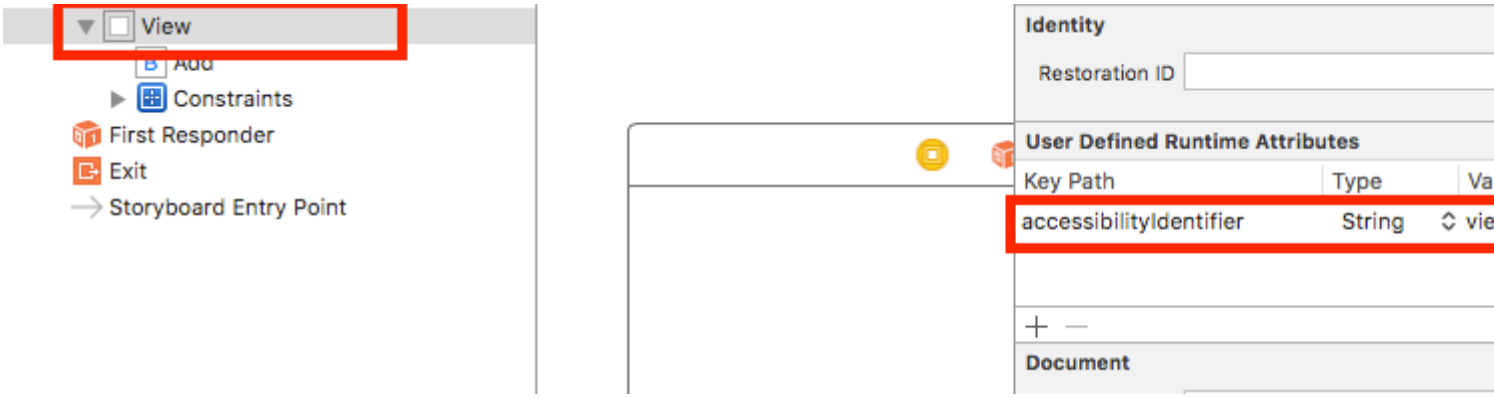
Когда доступность доступна в Утилитах

- Выберите `storyboard` .
- Развернуть `the Utilities`
- Выберите `Identity Inspector`
- Выберите свой элемент в раскадровке
- Добавить новый идентификатор доступности (в примере `addButton`)



Когда функция доступности отключена в утилитах

- Выберите `storyboard` .
- Развернуть `the Utilities`
- Выберите `Identity Inspector`
- Выберите свой элемент в раскадровке
- Добавить атрибут в `User Defined Runtime Attributes`
- Тип `Key Path` - `accessibilityIdentifier`
- Для `Type` - `String
- Для `Value` - новый идентификатор доступности для элемента (например , в `view`)



Настройка в файле UITest

```
import XCTest

class StackOverFlowUITests: XCTestCase {

    private let app = XCUIApplication()

    //Views

    private var view: XCUIElement!

    //Buttons

    private var addButton: XCUIElement!


    override func setUp() {
        super.setUp()

        app.launch()

        //Views

        view = app.otherElements["view"]

        //Buttons

        addButton = app.buttons["addButton"]
    }
```

```
    func testMyApp() {

        addButton.tap()
        view.tap()
    }
}
```

В `[ ]` добавить идентификатор доступности для элемента.

UIView, UIImageView, UIScrollView

```
let imageView = app.images["imageView"]
let scrollView = app.scrollViews["scrollView"]
let view = app.otherElements["view"]
```

UILabel

```
let label = app.staticTexts["label"]
```

UIStackView

```
let stackView = app.otherElements["stackView"]
```

UITableView

```
let tableView = app.tables["tableView"]
```

UITableViewCell

```
let tableViewCell = tableView.cells["tableViewCell"]
```

Элементы UITableViewCell

```
let tableViewCellButton = tableView.cells.element(boundBy: 0).buttons["button"]
```

UICollectionView

```
let collectionView = app.collectionViews["collectionView"]
```

UIButton, UIBarButtonItem

```
let button = app.buttons["button"]
let barButtonItem = app.buttons["barButtonItem"]
```

UITextField

- нормальный UITextField

```
let textField = app.textFields["textField"]
```

- пароль UITextField

```
let passwordTextField = app.secureTextFields["passwordTextField"]
```

UITextView

```
let textView = app.textViews["textView"]
```

UISwitch

```
let switch = app.switches["switch"]
```

Оповещения

```
let alert = app.alerts["About yourself"] // Title of presented alert
```

Отключить анимацию во время тестирования пользовательского интерфейса

В тесте вы можете отключить анимацию, добавив в `setUp` :

```
    app.launchEnvironment = ["animations": "0"]
```

Где `app` является экземпляром XCUIApplication.

Запуск и завершение приложения во время выполнения

Заявка на обед для тестирования

```
override func setUp() {
    super.setUp()

    let app = XCUIApplication()
```

```
    app.launch()
}
```

Завершение заявки

```
func testStacOverFlowApp() {

    app.terminate()
}
```

Поворот устройств

Устройство можно повернуть, изменив `orientation` в `XCUIDevice.shared().orientation` :

```
XCUIDevice.shared().orientation = .landscapeLeft
XCUIDevice.shared().orientation = .portrait
```

Прочитайте Тестирование пользовательского интерфейса онлайн:
https://riptutorial.com/ru/ios/topic/7526/тестирование-пользовательского-интерфейса

глава 200: Универсальные ссылки

замечания

1. Когда вы поддерживаете универсальные ссылки, пользователи iOS 9 могут использовать ссылку на ваш сайт и плавно перенаправляться в установленное приложение, не переходя через Safari. Если ваше приложение не установлено, нажмите ссылку на свой сайт, чтобы открыть свой сайт в Safari.
2. Как правило, любая поддерживаемая ссылка, нажатая в Safari, или в случаях `UIWebView` / `WKWebView` должна открыть приложение.
3. Для iOS 9.2 и менее это будет работать только на устройстве. iOS 9.3 также поддерживает симулятор.
4. iOS запоминает выбор пользователя при открытии Universal Links. Если они нажали верхнюю правую палитру, чтобы открыть ссылку в Safari, все дальнейшие клики переведут их в Safari, а не в приложение. Они могут вернуться к открытию приложения по умолчанию, выбрав «Открыть» в баннере приложений на веб-сайте.

Examples

Сервер установки

Вам нужно иметь сервер, работающий в Интернете. Чтобы безопасно связать приложение iOS с сервером, Apple требует, чтобы вы предоставили файл конфигурации, называемый `apple-app-site-association` . Это `JSON` файл, который описывает домен и поддерживаемые маршруты.

Файл `apple-app-site-association` должен быть доступен через `HTTPS` без каких-либо переадресаций в *https: // {domain} / apple-app-site-association* .

Файл выглядит следующим образом:

```
{
"applinks": {
    "apps": [ ],
    "details": [
        {
            "appID": "{app_prefix}.{app_identifier}",
            "paths": [ "/path/to/content", "/path/to/other/*", "NOT /path/to/exclude" ]
        },
        {
            "appID": "TeamID.BundleID2",
            "paths": [ "*" ]
        }
    ]
}
}
```

ПРИМЕЧАНИЕ. *Не* `.json` *к имени файла* `apple-app-site-association` *.*

Ключи следующие:
`apps` : Должен иметь пустой массив как его значение, и он должен присутствовать. Именно так Apple хочет этого.
`details` : Является массивом словарей, по одному для каждого приложения iOS, поддерживаемого веб-сайтом. Каждый словарь содержит информацию о приложении, команде и идентификаторах пакетов.

Существует три способа определения путей:
`Static` : весь поддерживаемый путь жестко запрограммирован для идентификации конкретной ссылки, например / static / terms
`Wildcards` : A * может использоваться для соответствия динамическим путям, например / books / * может соответствовать пути к любой странице автора. ? внутри конкретных компонентов пути, например, книги / 1? может использоваться для соответствия любым книгам, чей идентификатор начинается с 1.
`Exclusions` : предварительный путь с НЕ исключает, что этот путь не согласован.

Важен порядок, в котором упоминаются пути в массиве. Более ранние индексы имеют более высокий приоритет. Когда путь совпадает, оценка останавливается, а другие пути игнорируются. Каждый путь чувствителен к регистру.

#Website Code

Код сайта можно найти в разделе gh-pages на https://github.com/vineetchoudhary/iOS-Universal-Links/tree/gh-pages

Поддержка нескольких доменов

Каждому домену, поддерживаемому в приложении, необходимо предоставить свой собственный файл ассоциации apple-app-site-association. Если контент, обслуживаемый каждым доменом, отличается, содержимое файла также будет изменяться для поддержки соответствующих путей. В противном случае можно использовать один и тот же файл, но он должен быть доступен в каждом поддерживаемом домене.

Подписание файла ассоциации приложений

Примечание . *Вы можете пропустить эту часть, если ваш сервер использует* `HTTPS` *для обслуживания контента и перехода к руководству по настройке приложения.*

Если ваше приложение предназначено для iOS 9, а ваш сервер использует `HTTPS` для обслуживания контента, вам не нужно подписывать файл. Если нет (например, при поддержке Handoff на iOS 8), он должен быть подписан с использованием сертификата `SSL` из признанного центра сертификации.

Примечание . Это не сертификат, предоставленный Apple для отправки вашего приложения в App Store. Он должен быть предоставлен сторонним производителем, и рекомендуется использовать тот же сертификат, который вы используете для своего `HTTPS` сервера (хотя это не обязательно).

Чтобы подписать файл, сначала создайте и сохраните простую .txt-версию. Затем в терминале выполните следующую команду:

```
cat <unsigned_file>.txt | openssl smime -sign -inkey example.com.key -signer example.com.pem -certfile intermediate.pem -noattr -nodetach -outform DER > apple-app-site-association
```

Это приведет к выводу подписанного файла в текущий каталог. `example.com.key` , `example.com.pem` и `intermediate.pem` - это файлы, которые предоставили вам ваш сертифицирующий орган.

Примечание . Если файл неподписан, он должен иметь `Content-Type` of `application/json` . В противном случае это должно быть `application/pkcs7-mime` .

Подтвердите свой сервер с помощью инструмента проверки приложений Apple App Проверьте свою веб-страницу для API поиска IOS 9. Введите URL-адрес, и Applebot сканирует вашу веб-страницу и покажет, как вы можете оптимизировать наилучшие результаты. Https://search.developer.apple.com/appsearch-validation-tool/

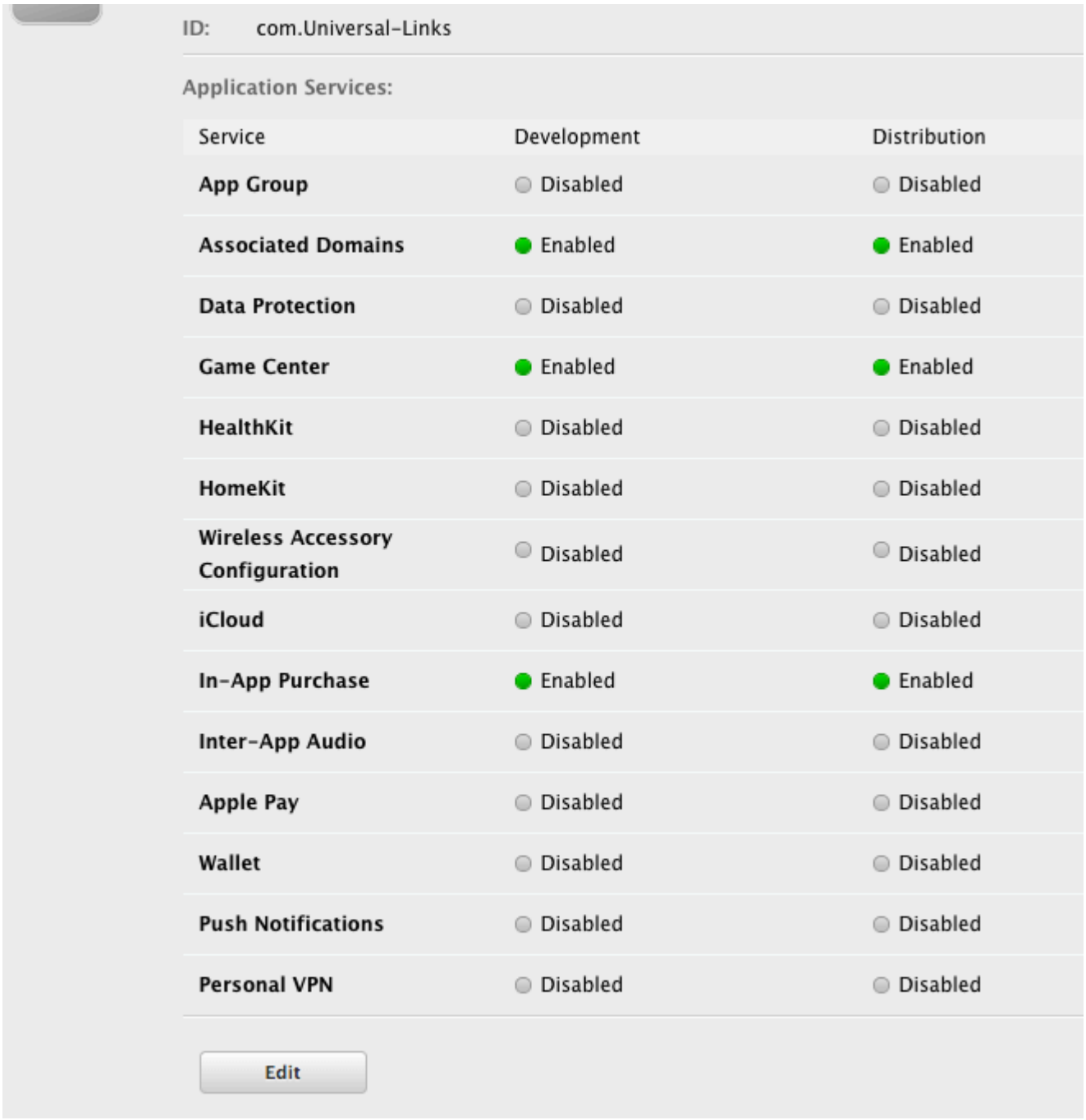
Настройка приложения iOS (включение универсальных ссылок)

Настройка на стороне приложения требует двух вещей:

1. Конфигурирование права приложения и включение универсальных ссылок путем включения в проект функции Связанные домены.
2. Обработка входящих ссылок в приложении `AppDelegate` .

1. Настройка права приложения и включение универсальных ссылок.

Первым шагом в настройке прав вашего приложения является включение его для вашего идентификатора приложения. Сделайте это в Центре разработчиков Apple Developer. Нажмите «Сертификаты», «Идентификаторы» и «Профили», а затем «Идентификаторы». Выберите свой идентификатор приложения (при необходимости создайте его), нажмите «Изменить» и включите право Ассоциированных доменов.



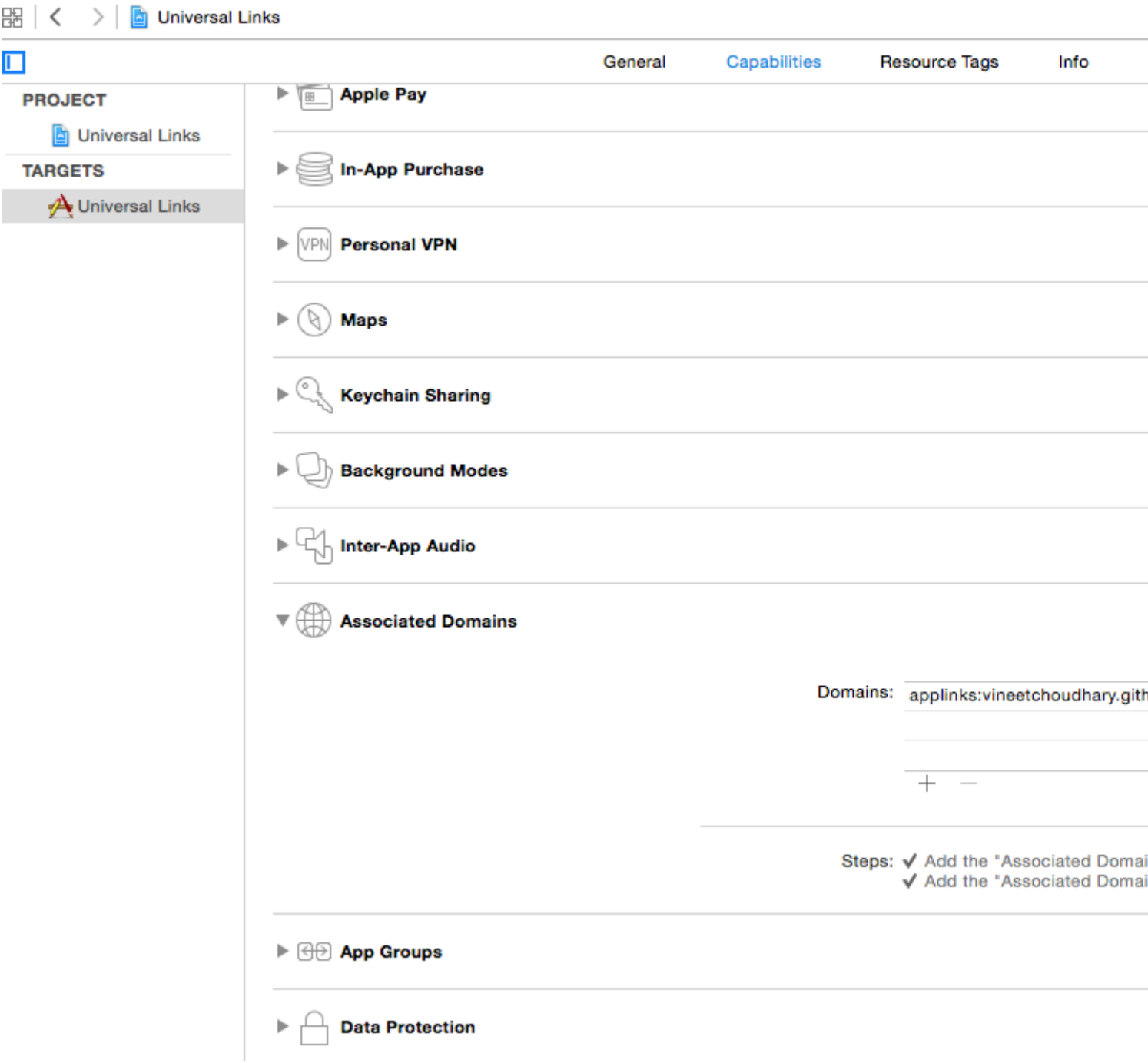
Затем получите префикс и суффикс идентификатора приложения, щелкнув соответствующий идентификатор приложения.

Префикс и суффикс идентификатора приложения должны совпадать с приложением в файле ассоциации apple-app-site-association.

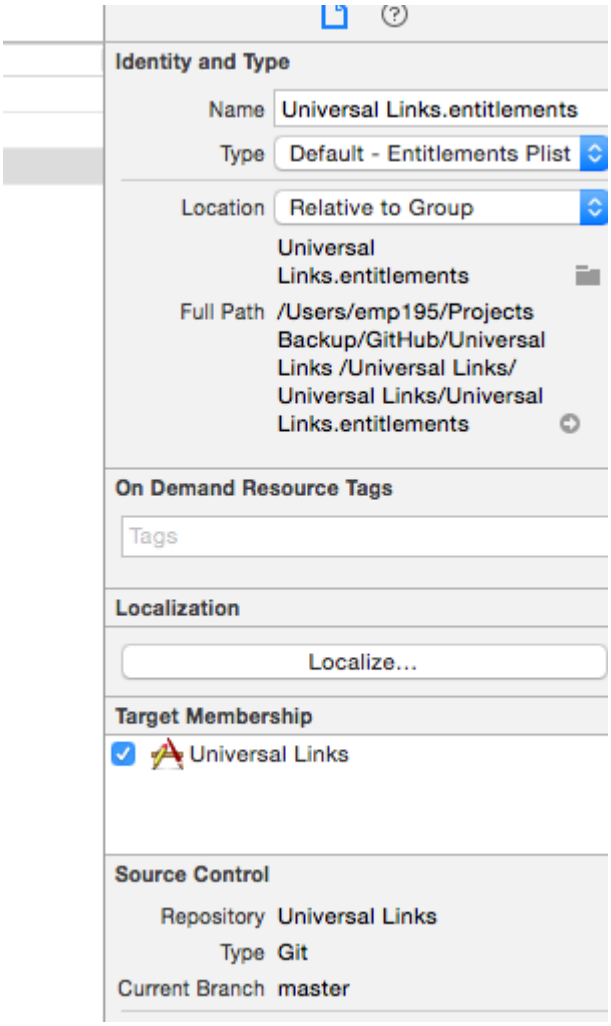
Затем в `Xcode` выберите целевую аудиторию приложения, нажмите «Возможности» и переключите соответствующие домены на «Вкл.». Добавьте запись для каждого домена, поддерживаемого вашим приложением, с префиксом **ссылок на приложения:**

Например, **приложения: YourCustomDomainName.com**

Что похоже на пример приложения:



Примечание . Убедитесь, что вы выбрали ту же команду и ввели тот же идентификатор пакета, что и зарегистрированный идентификатор приложения в Центре-члене. Также убедитесь, что файл прав включен в Xcode, выбирая файл и в File Inspector, убедитесь, что ваша цель проверена.



---

2. Обработка входящих ссылок в приложении AppDelegate

Все переадресации из Safari в приложение для универсальных ссылок идут по методу ниже в классе AppDelegate приложения. Вы анализируете этот URL, чтобы определить правильное действие в приложении.

```
[UIApplicationDelegate application: continueUserActivity: restorationHandler:]
```

Objective-C

```
-(BOOL)application:(UIApplication *)application continueUserActivity:(NSUserActivity
*)userActivity restorationHandler:(void (^)(NSArray * _Nullable))restorationHandler{
    ///Checking whether the activity was from a web page redirect to the app.
    if ([userActivity.activityType isEqualToString: NSUserActivityTypeBrowsingWeb]) {
        ///Getting the URL from the UserActivty Object.
        NSURL *url = userActivity.webpageURL;
        UIStoryboard *storyBoard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
        UINavigationController *navigationController = (UINavigationController
*)_window.rootViewController;
        if ([url.pathComponents containsObject:@"home"]) {
            [navigationController pushViewController:[storyBoard
instantiateViewControllerWithIdentifier:@"HomeScreenId"] animated:YES];
        }else if ([url.pathComponents containsObject:@"about"]){
```

```
            [navigationController pushViewController:[storyBoard
instantiateViewControllerWithIdentifier:@"AboutScreenId"] animated:YES];
        }
    }
    return YES;
}
```

Быстро:

```
func application(application: UIApplication, continueUserActivity userActivity:
NSUserActivity, restorationHandler: ([AnyObject]?) -> Void) -> Bool {
      if userActivity.activityType == NSUserActivityTypeBrowsingWeb {
          let url = userActivity.webpageURL!
          //handle url
      }
      return true
  }
```

Код приложения iOS

Код приложения можно найти здесь .

Прочитайте Универсальные ссылки онлайн: https://riptutorial.com/ru/ios/topic/2362/универсальные-ссылки

глава 201: Управление клавиатурой

Examples

Прокрутка UIScrollView / UITableView при отображении клавиатуры

Существует несколько подходов:

1. Вы можете подписаться на уведомления о событиях на клавиатуре и вручную изменить смещение:

```
//Swift 2.0+
override func viewDidLoad() {
    super.viewDidLoad()

    NSNotificationCenter.defaultCenter().addObserver(self, selector:
#selector(YourVCClassName.keyboardWillShow(_:)), name: UIKeyboardWillShowNotification, object:
nil)
    NSNotificationCenter.defaultCenter().addObserver(self, selector:
#selector(YourVCClassName.keyboardWillHide(_:)), name: UIKeyboardWillHideNotification, object:
nil)
}

func keyboardWillShow(notification: NSNotification) {
    if let userInfo = notification.userInfo {
        if let keyboardHeight =
userInfo[UIKeyboardFrameEndUserInfoKey]?.CGRectValue.size.height {
            tableView.contentInset = UIEdgeInsetsMake(0, 0, keyboardHeight, 0)
        }
    }
}

func keyboardWillHide(notification: NSNotification) {
    tableView.contentInset = UIEdgeInsetsMake(0, 0, 0, 0)
}

//Objective-C
- (void)viewDidLoad {

   [super viewDidLoad];


    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillShow:) name:UIKeyboardWillShowNotification object:nil];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillHide:) name:UIKeyboardWillHideNotification object:nil];

}

- (void)keyboardWillShow:(NSNotification *)notification {

    NSDictionary *userInfo = [notification userInfo];

    if (userInfo) {
```

```
        CGRect keyboardEndFrame;
        [[userInfo objectForKey:UIKeyboardFrameEndUserInfoKey] getValue:&keyboardEndFrame];
        tableView.contentInset = UIEdgeInsetsMake(0, 0, keyboardEndFrame.size.height, 0);

    }

}

- (void)keyboardWillHide:(NSNotification *)notification {

    tableView.contentInset = UIEdgeInsetsMake(0, 0, 0, 0);

}
```

2. Или используйте готовые решения, такие как TPKeyboardAvoidingTableView или TPKeyboardAvoidingScrollView https://github.com/michaeltyson/TPKeyboardAvoiding

Отключите клавиатуру с краном

Если вы хотите скрыть клавиатуру, коснитесь ее наружу, можно использовать этот хакерский трюк (работает только с Objective-C):

```
- (void)viewDidLoad {
    [super viewDidLoad];

    // dismiss keyboard when tap outside a text field
    UITapGestureRecognizer *tapGestureRecognizer = [[UITapGestureRecognizer alloc]
initWithTarget:self.view action:@selector(endEditing:)];
    [tapGestureRecognizer setCancelsTouchesInView:NO];
    [self.view addGestureRecognizer:tapGestureRecognizer];
}
```

для Swift будет немного больше кода:

```
override func viewDidLoad() {
    super.viewDidLoad()

    // dismiss keyboard when tap outside a text field
    let tapGestureRecognizer: UITapGestureRecognizer = UITapGestureRecognizer(target: self,
action: #selector(YourVCName.dismissKeyboard))
    view.addGestureRecognizer(tapGestureRecognizer)
}

//Calls this function when the tap is recognized.
func dismissKeyboard() {
    //Causes the view (or one of its embedded text fields) to resign the first responder
status.
    view.endEditing(true)
}
```

Другой пример Swift 3 / iOS 10

```
class vc: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
```

```
        // Do any additional setup after loading the view, typically from a nib.

        txtSomeField.delegate = self
    }
}

extension vc: UITextFieldDelegate {
    //Hide the keyboard for any text field when the UI is touched outside of the keyboard.
    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)
    {
        self.view.endEditing(true) //Hide the keyboard
    }
}
```

Создание пользовательской встроенной клавиатуры

Это базовая клавиатура в приложении. Тот же метод можно использовать для создания любой раскладки клавиатуры. Вот основные вещи, которые нужно сделать:

- Создайте раскладку клавиатуры в файле .xib, владельцем которого является класс Swift или Objective-C, который является подклассом `UIView` .
- Скажите `UITextField` чтобы использовать пользовательскую клавиатуру.
- Используйте делегат для связи между клавиатурой и основным контроллером.

Создайте файл раскладки клавиатуры .xib.

- В Xcode перейдите в **File> New> File ...> iOS> User Interface> View,** чтобы создать файл .xib.
- Я назвал мой Keyboard.xib
- Добавьте нужные вам кнопки.
- Используйте ограничения автоматической компоновки, чтобы независимо от размера клавиатуры, кнопки будут соответственно изменять размер.
- Установите владельца файла (а не корневой режим) как класс `Keyboard` . Это общий источник ошибок. Вы создадите этот класс на следующем шаге. См. Примечание в конце.

Создайте файл .clift UIView для подкласса

- В Xcode перейдите в **File> New> File ...> iOS> Source> Cocoa Touch Class,** чтобы создать **класс** Swift или Objective-C. Выберите `UIView` как суперкласс для вновь созданного класса
- Я назвал мой `Keyboard.swift` (класс `Keyboard` в Objective-C)
- Добавьте следующий код для Swift:

```
import UIKit

// The view controller will adopt this protocol (delegate)
// and thus must contain the keyWasTapped method
protocol KeyboardDelegate: class {
    func keyWasTapped(character: String)
}

class Keyboard: UIView {

    // This variable will be set as the view controller so that
    // the keyboard can send messages to the view controller.
    weak var delegate: KeyboardDelegate?

    // MARK:- keyboard initialization

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        initializeSubviews()
    }

    override init(frame: CGRect) {
        super.init(frame: frame)
        initializeSubviews()
    }

    func initializeSubviews() {
```

```
          let xibFileName = "Keyboard" // xib extention not included
          let view = NSBundle.mainBundle().loadNibNamed(xibFileName, owner: self,
options: nil)[0] as! UIView
          self.addSubview(view)
          view.frame = self.bounds
      }

      // MARK:- Button actions from .xib file

      @IBAction func keyTapped(sender: UIButton) {
          // When a button is tapped, send that information to the
          // delegate (ie, the view controller)
          self.delegate?.keyWasTapped(sender.titleLabel!.text!) // could alternatively
send a tag value
      }

  }
```

- Добавьте следующий код для Objective-C:

Файл Keyboard.h

```
#import <UIKit/UIKit.h>

// The view controller will adopt this protocol (delegate)
// and thus must contain the keyWasTapped method
@protocol KeyboardDelegate<NSObject>
- (void)keyWasTapped:(NSString *)character;
@end

@interface Keyboard : UIView
@property (nonatomic, weak) id<KeyboardDelegate> delegate;
@end
```

Файл Keyboard.m

```
#import "Keyboard.h"

@implementation Keyboard

- (id)initWithCoder:(NSCoder *)aDecoder {
    self = [super initWithCoder:aDecoder];
    [self initializeSubviews];
    return self;
}

- (id)initWithFrame:(CGRect)frame {
    self = [super initWithFrame:frame];
    [self initializeSubviews];
    return self;
}

- (void)initializeSubviews {
    NSString *xibFileName = @"Keyboard"; // xib extention not included
    UIView *view = [[[NSBundle mainBundle] loadNibNamed:xibFileName owner:self
options:nil] firstObject];
    [self addSubview:view];
```

```
    view.frame = self.bounds;
}

// MARK:- Button actions from .xib file

-(IBAction)keyTapped:(UIButton *)sender {
    // When a button is tapped, send that information to the
    // delegate (ie, the view controller)
    [self.delegate keyWasTapped:sender.titleLabel.text]; // could alternatively send a
tag value
}

@end
```

- Управляйте перетаскиванием с кнопок на обратный вызов кнопки в файле `@IBAction` метод `@IBAction` в владельце Swift или Objective-C, чтобы связать их все.
- Обратите внимание, что протокол и код делегирования. См. Этот ответ для простого объяснения того, как работают делегаты.

Настройка контроллера просмотра

- Добавьте `UITextField` в свою основную раскадровку и подключите ее к контроллеру своего вида с помощью `IBOutlet` . Назовите его `textField` .

- Используйте следующий код для контроллера просмотра в Swift:

```
  import UIKit

  class ViewController: UIViewController, KeyboardDelegate {

      @IBOutlet weak var textField: UITextField!

      override func viewDidLoad() {
          super.viewDidLoad()

          // initialize custom keyboard
          let keyboardView = Keyboard(frame: CGRect(x: 0, y: 0, width: 0, height: 300))
          keyboardView.delegate = self // the view controller will be notified by the
keyboard whenever a key is tapped

          // replace system keyboard with custom keyboard
          textField.inputView = keyboardView
      }

      // required method for keyboard delegate protocol
      func keyWasTapped(character: String) {
          textField.insertText(character)
      }
  }
```

- Используйте следующий код для Objective-C:

  Файл .h

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

@end
```

Файл .m

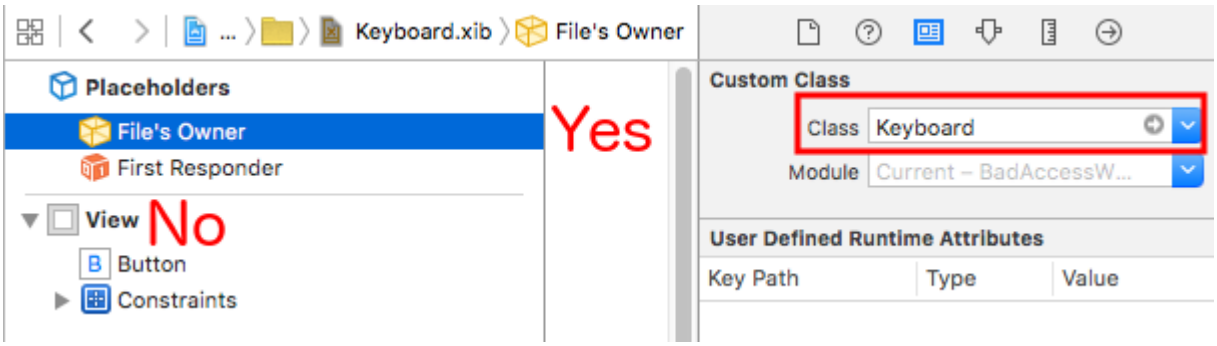
```
#import "ViewController.h"
#import "Keyboard.h"

@interface ViewController ()<KeyboardDelegate>

@property (nonatomic, weak) IBOutlet UITextField *textField;

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // initialize custom keyboard
    Keyboard *keyboardView = [[Keyboard alloc] initWithFrame:CGRectMake(0, 0, 0, 300)];
    keyboardView.delegate = self; // the view controller will be notified by the keyboard
whenever a key is tapped

    // replace system keyboard with custom keyboard
    self.textField.inputView = keyboardView;
}

- (void)keyWasTapped:(NSString *)character {
    [self.textField insertText:character];
}

@end
```

- Обратите внимание, что контроллер просмотра использует протокол `KeyboardDelegate` , который мы определили выше.

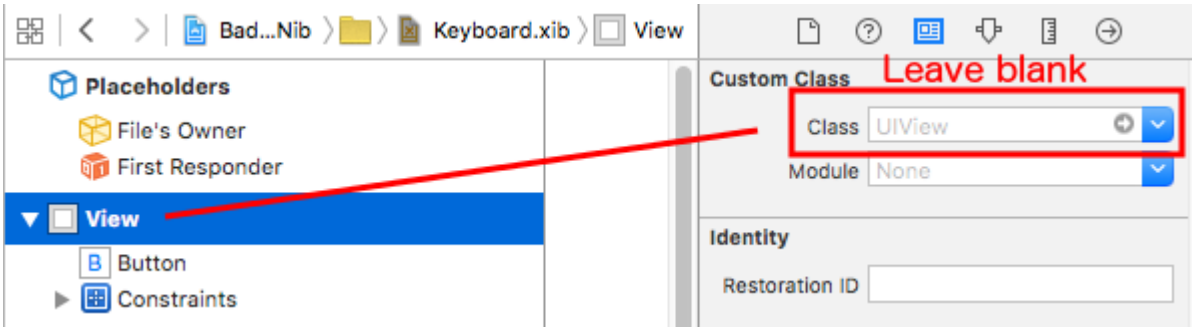
Общая ошибка

Если вы получаете ошибку `EXC_BAD_ACCESS` , это, вероятно, потому, что вы устанавливаете пользовательский класс представления как `Keyboard` а не для владельца файла nib.

Выберите `Keyboard.nib` а затем выберите «Владелец файла».



Убедитесь, что пользовательский класс для корневого представления пуст.



Заметки

Этот пример исходит из этого ответа на переполнение стека .

Управление клавиатурой с помощью Singleton + Delegate

Когда я впервые начал управлять клавиатурой, я бы использовал отдельные уведомления в каждом ViewController.

Метод уведомления (с использованием NSNotification):

```
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        NSNotificationCenter.defaultCenter().addObserver(self, selector:
#selector(ViewController.keyboardNotification(_:)), name:
UIKeyboardWillChangeFrameNotification, object: nil)
    }

    func keyboardNotification(notification: NSNotification) {
        guard let userInfo = notification.userInfo else { return }

        let endFrame = (userInfo[UIKeyboardFrameEndUserInfoKey] as? NSValue)?.CGRectValue()
        let duration: NSTimeInterval = (userInfo[UIKeyboardAnimationDurationUserInfoKey] as?
NSNumber)?.doubleValue ?? 0
        let animationCurveRawNSN = userInfo[UIKeyboardAnimationCurveUserInfoKey] as? NSNumber
        let animationCurveRaw = animationCurveRawNSN?.unsignedLongValue ??
UIViewAnimationOptions.CurveEaseOut.rawValue
        let animationCurve: UIViewAnimationOptions = UIViewAnimationOptions(rawValue:
animationCurveRaw)
```

```
        if endFrame?.origin.y >= UIScreen.mainScreen().bounds.size.height {
            lowerViewBottomConstraint.constant = 0
        } else {
            lowerViewBottomConstraint.constant = endFrame?.size.height ?? 0.0
        }
        view.animateConstraintWithDuration(duration, delay: NSTimeInterval(0), options:
animationCurve, completion: nil)
    }
}
```

Моя проблема заключалась в том, что я снова и снова записывал этот код для каждого ViewController. После нескольких экспериментов, проведенных с использованием шаблона Singleton + Delegate, я смог повторно использовать кучу кода и организовать все управление клавиатурой в одном месте!

Метод Singleton + Delegate:

```
protocol KeyboardManagerDelegate: class {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions)
}

class KeyboardManager {

    weak var delegate: KeyboardManagerDelegate?

    class var sharedInstance: KeyboardManager {
        struct Singleton {
            static let instance = KeyboardManager()
        }
        return Singleton.instance
    }

    init() {
        NSNotificationCenter.defaultCenter().addObserver(self, selector:
#selector(KeyboardManager.keyboardWillChangeFrameNotification(_:)), name:
UIKeyboardWillChangeFrameNotification, object: nil)
    }

    @objc func keyboardWillChangeFrameNotification(notification: NSNotification) {
        guard let userInfo = notification.userInfo else { return }

        let endFrame = (userInfo[UIKeyboardFrameEndUserInfoKey] as? NSValue)?.CGRectValue()
        let duration: NSTimeInterval = (userInfo[UIKeyboardAnimationDurationUserInfoKey] as?
NSNumber)?.doubleValue ?? 0
        let animationCurveRawNSN = userInfo[UIKeyboardAnimationCurveUserInfoKey] as? NSNumber
        let animationCurveRaw = animationCurveRawNSN?.unsignedLongValue ??
UIViewAnimationOptions.CurveEaseOut.rawValue
        let animationCurve: UIViewAnimationOptions = UIViewAnimationOptions(rawValue:
animationCurveRaw)

        delegate?.keyboardWillChangeFrame(endFrame, duration: duration, animationCurve:
animationCurve)
    }
}
```

Теперь, когда я хочу управлять клавиатурой из ViewController, все, что мне нужно сделать,

это установить делегат на этот ViewController и реализовать любые методы делегата.

```
class ViewController: UIViewController {
    override func viewWillAppear(animated: Bool) {
        super.viewWillAppear(animated)
        KeyboardManager.sharedInstance.delegate = self
    }
}

// MARK: - Keyboard Manager

extension ViewController: KeyboardManagerDelegate {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions) {
        if endFrame?.origin.y >= UIScreen.mainScreen().bounds.size.height {
           lowerViewBottomConstraint.constant = 0
        } else {
            lowerViewBottomConstraint.constant = (endFrame?.size.height ?? 0.0)
        }
        view.animateConstraintWithDuration(duration, delay: NSTimeInterval(0), options:
animationCurve, completion: nil)
    }
}
```

Этот метод очень настраиваемый тоже! Предположим, мы хотим добавить функциональность для `UIKeyboardWillHideNotification` . Это так же просто, как добавить метод к нашему `KeyboardManagerDelegate` .

`KeyboardManagerDelegate` с `UIKeyboardWillHideNotification` :

```
protocol KeyboardManagerDelegate: class {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions)
    func keyboardWillHide(notificationUserInfo: [NSObject: AnyObject])
}

class KeyboardManager {
    init() {
        NSNotificationCenter.defaultCenter().addObserver(self, selector:
#selector(KeyboardManager.keyboardWillChangeFrameNotification(_:)), name:
UIKeyboardWillChangeFrameNotification, object: nil)
        NSNotificationCenter.defaultCenter().addObserver(self, selector:
#selector(KeyboardManager.keyboardWillHide(_:)), name: UIKeyboardWillHideNotification, object:
nil)
    }

    func keyboardWillHide(notification: NSNotification) {
        guard let userInfo = notification.userInfo else { return }
        delegate?.keyboardWillHide(userInfo)
    }
}
```

Скажем, мы хотим только реализовать `func keyboardWillHide(notificationUserInfo: [NSObject: AnyObject])` в одном ViewController. Мы также можем сделать этот метод опционным.

```
typealias KeyboardManagerDelegate = protocol<KeyboardManagerModel,
```

```
KeyboardManagerConfigureable>

protocol KeyboardManagerModel: class {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions)
}

@objc protocol KeyboardManagerConfigureable {
    optional func keyboardWillHide(userInfo: [NSObject: AnyObject])
}
```

* Обратите внимание, что этот шаблон помогает избежать чрезмерного использования `@objc` . См. Http://www.jessesquires.com/avoiding-objc-in-swift/ для получения более подробной информации!

В целом, я нашел, что использование Singleton + Delegate для управления клавиатурой является более эффективным и простым в использовании, чем использование уведомлений

Перемещение вверх или вниз, когда клавиатура присутствует

Примечание. Это работает только для встроенной клавиатуры, предоставляемой iOS

SWIFT:

Чтобы представление **UIViewController** увеличивало начало кадра при его представлении и уменьшало его при его скрытии, добавьте в свой класс следующие функции:

```
func keyboardWillShow(notification: NSNotification) {

    if let keyboardSize = (notification.userInfo?[UIKeyboardFrameBeginUserInfoKey] as?
NSValue)?.cgRectValue {
        if self.view.frame.origin.y == 0{
            self.view.frame.origin.y -= keyboardSize.height
        }
    }

}

func keyboardWillHide(notification: NSNotification) {
    if let keyboardSize = (notification.userInfo?[UIKeyboardFrameBeginUserInfoKey] as?
NSValue)?.cgRectValue {
        if self.view.frame.origin.y != 0{
            self.view.frame.origin.y += keyboardSize.height
        }
    }
}
```

А в `viewDidLoad()` вашего класса добавьте следующих наблюдателей:

```
NotificationCenter.default.addObserver(self, selector: #selector(Login.keyboardWillShow),
name: NSNotification.Name.UIKeyboardWillShow, object: nil)
NotificationCenter.default.addObserver(self, selector: #selector(Login.keyboardWillHide),
name: NSNotification.Name.UIKeyboardWillHide, object: nil)
```

И это будет работать для любого размера экрана, используя свойство высоты клавиатуры.

Objective-C:

Чтобы сделать то же самое в Objective-C, этот код можно использовать:

```
- (void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillShow:) name:UIKeyboardWillShowNotification object:nil];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillHide:) name:UIKeyboardWillHideNotification object:nil];
}

- (void)viewWillDisappear:(BOOL)animated {
    [super viewWillDisappear:animated];
    [[NSNotificationCenter defaultCenter] removeObserver:self
name:UIKeyboardWillShowNotification object:nil];
    [[NSNotificationCenter defaultCenter] removeObserver:self
name:UIKeyboardWillHideNotification object:nil];
}

- (void)keyboardWillShow:(NSNotification *)notification
{
    CGSize keyboardSize = [[[notification userInfo]
objectForKey:UIKeyboardFrameBeginUserInfoKey] CGRectValue].size;

    [UIView animateWithDuration:0.3 animations:^{
        CGRect f = self.view.frame;
        f.origin.y = -keyboardSize.height;
        self.view.frame = f;
    }];
}

-(void)keyboardWillHide:(NSNotification *)notification
{
    [UIView animateWithDuration:0.3 animations:^{
        CGRect f = self.view.frame;
        f.origin.y = 0.0f;
        self.view.frame = f;
    }];
}
```

Прочитайте Управление клавиатурой онлайн: https://riptutorial.com/ru/ios/topic/436/управление-клавиатурой

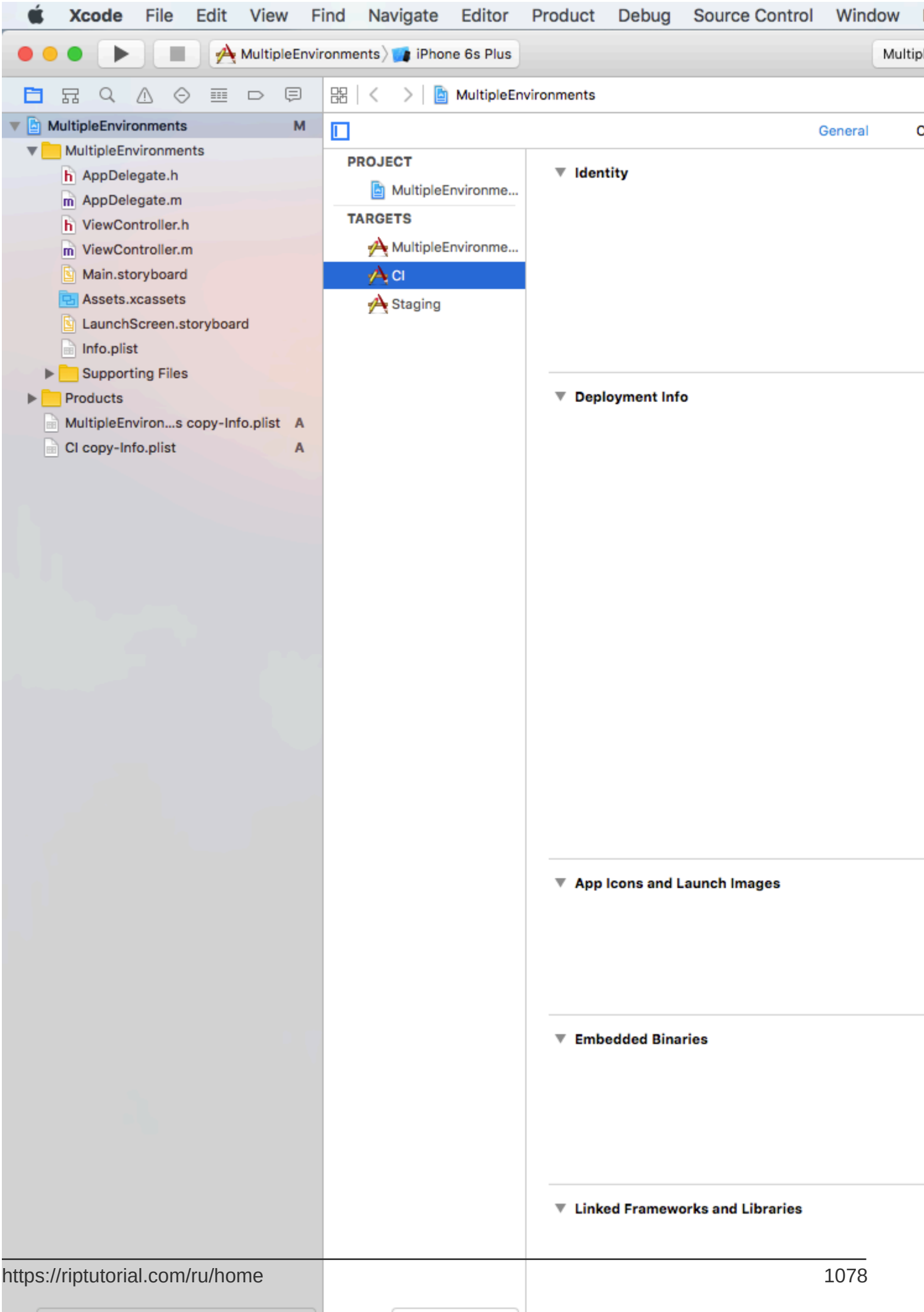
глава 202: Управление несколькими средами с помощью макроса

Examples

Управление несколькими средами с использованием нескольких целей и макросов

Например, у нас есть две среды: CI - Staging и хотите добавить некоторые настройки для каждой среды. Здесь я попытаюсь настроить URL-адрес сервера, имя приложения.

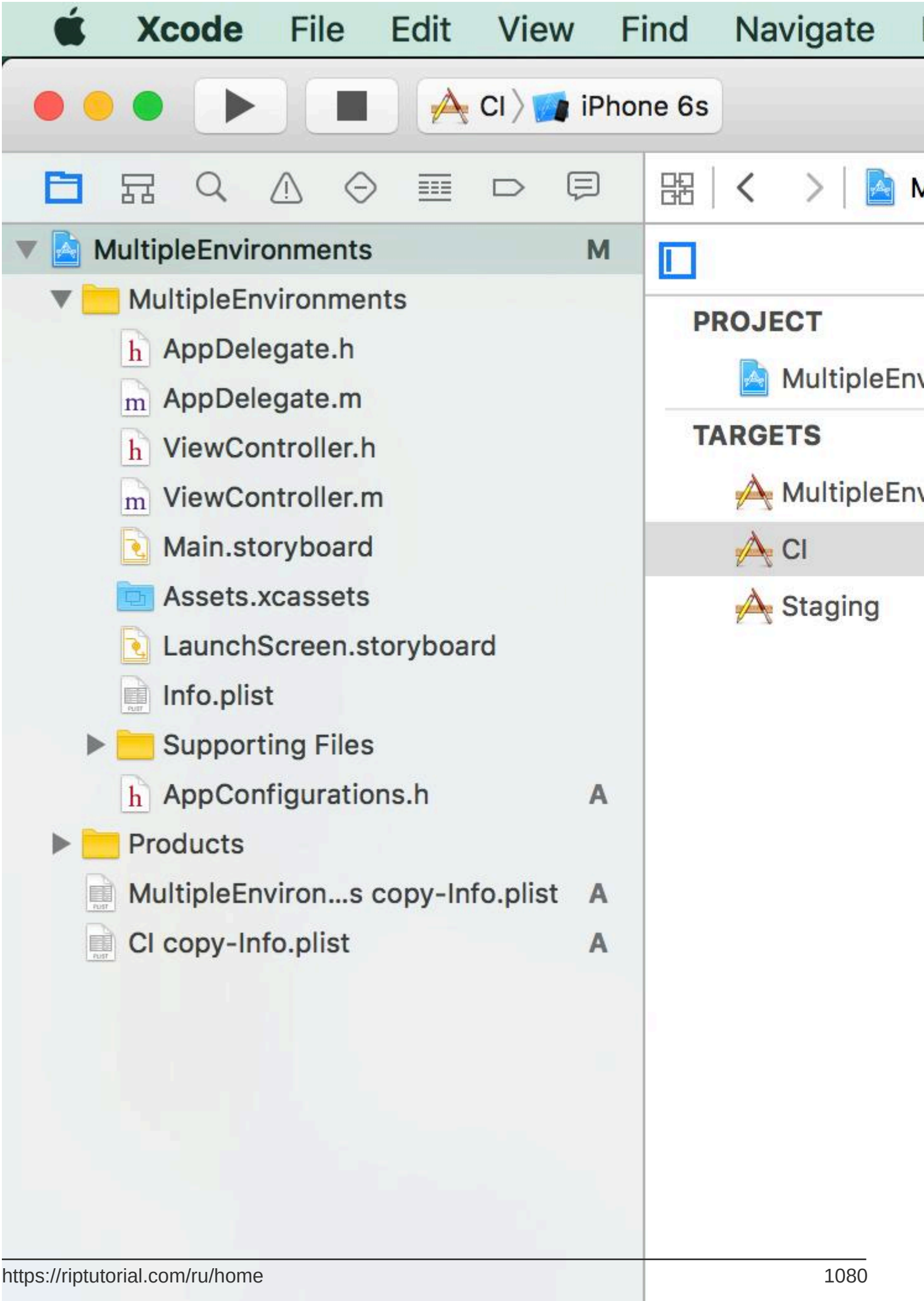
Во-первых, мы создаем две цели для двух сред, дублируя главную цель:


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window
MultipleEnvironments iPhone 6s Plus
MultipleEnvironments
MultipleEnvironments M
MultipleEnvironments
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Supporting Files
Products
MultipleEnviron...s copy-Info.plist A
CI copy-Info.plist A
General
PROJECT
MultipleEnvironme...
TARGETS
MultipleEnvironme...
CI
Staging
Identity
Deployment Info
App Icons and Launch Images
Embedded Binaries
Linked Frameworks and Libraries

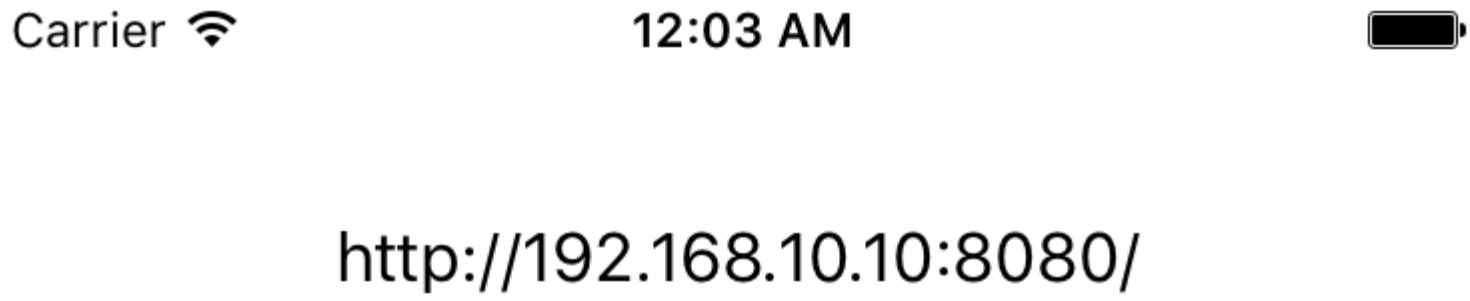
MultipleEnvironment), SERVER_URL: http://192.168.10.10:8080/

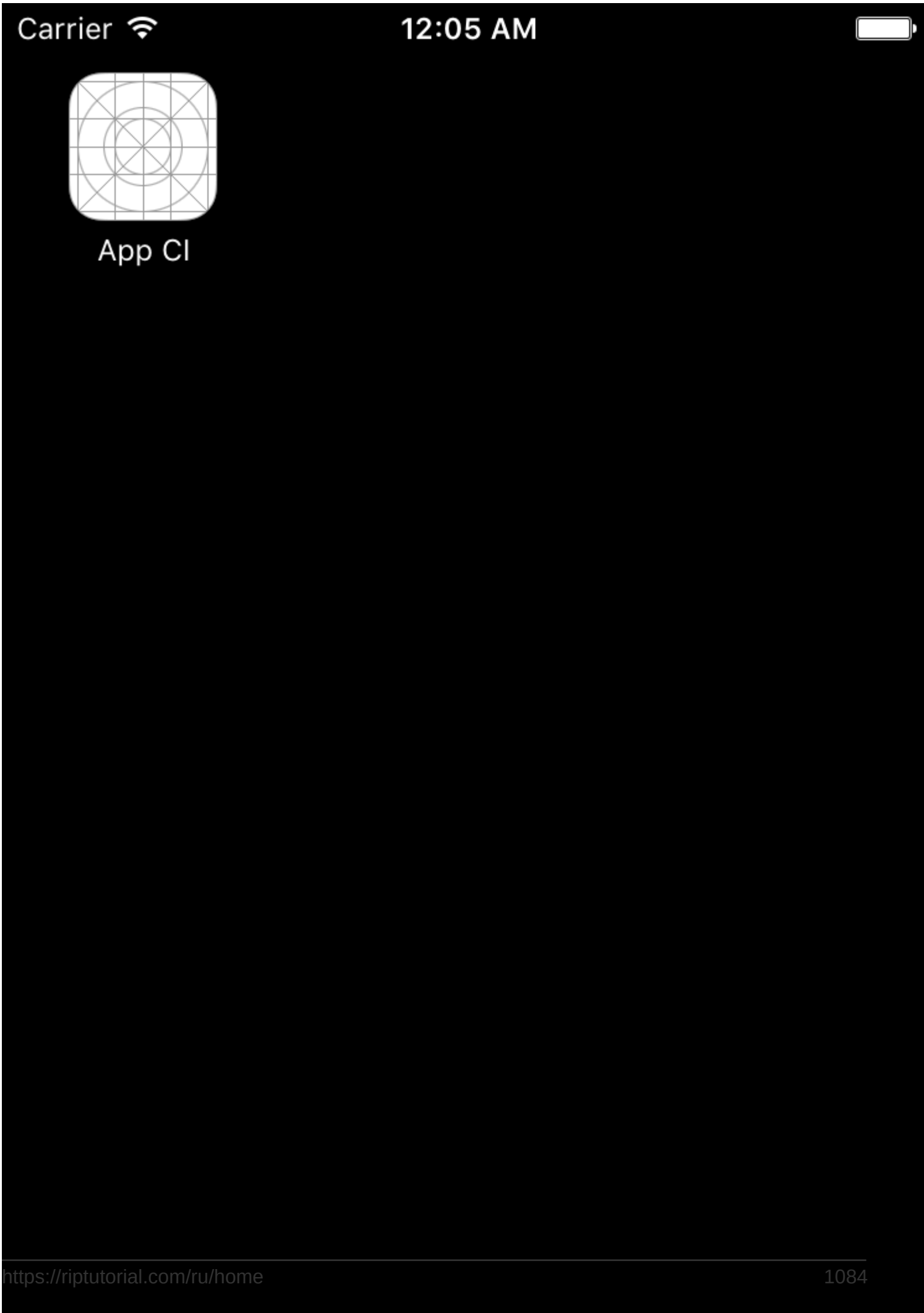
- Если мы запускаем / архивируем с использованием цели CI, SERVER_URL - http://ci.api.example.com/
- Если мы запускаем / архивируем с использованием цели STAGING, SERVER_URL - http://stg.api.example.com/

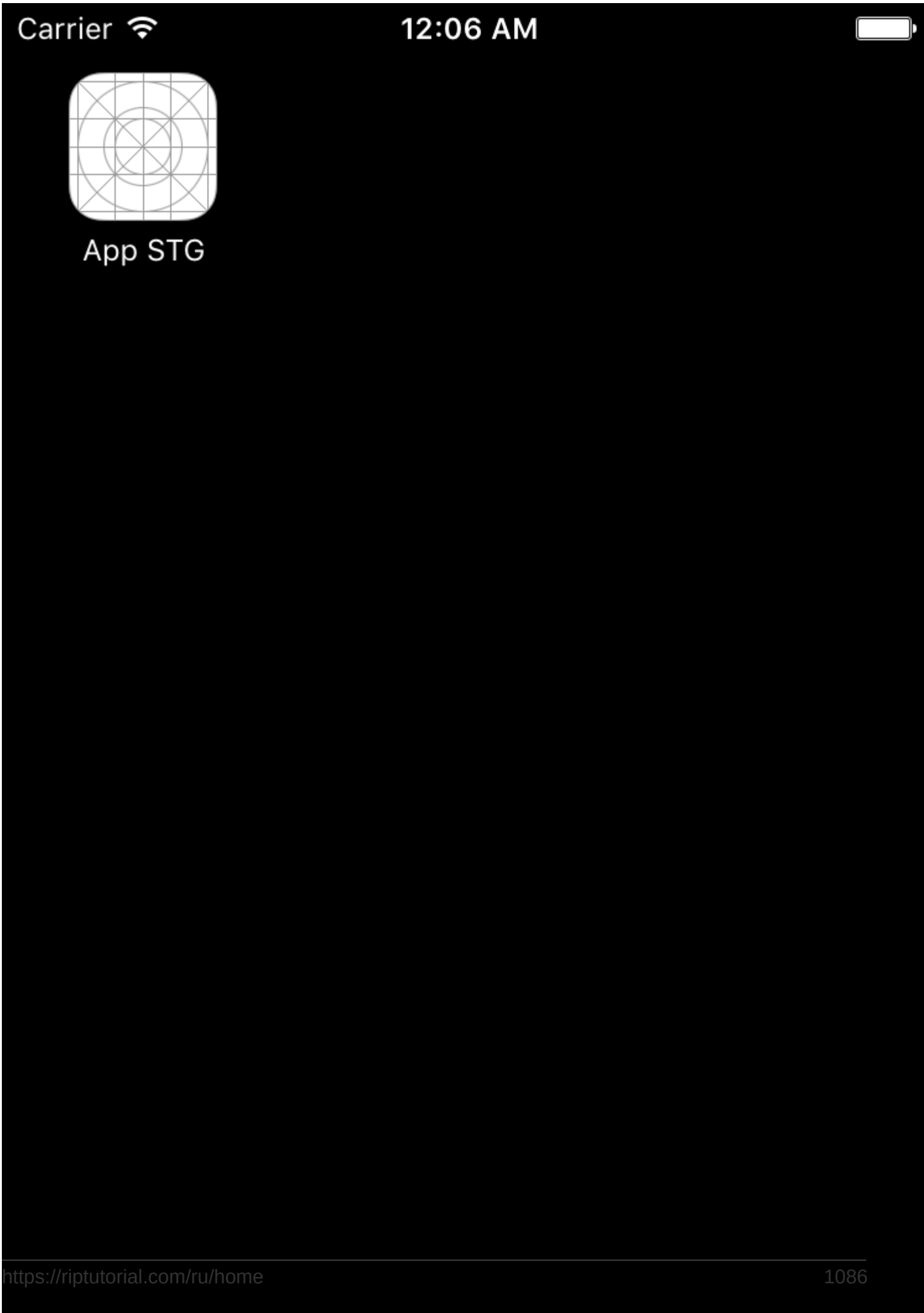
Если вы хотите выполнить дополнительную настройку, например: Изменить имя приложения для каждой цели:


Xcode File Edit View Find Navigate
CI iPhone 6s
MultipleEnvironments M
MultipleEnvironments
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Supporting Files
AppConfigurations.h A
Products
MultipleEnviron...s copy-Info.plist A
CI copy-Info.plist A
PROJECT
MultipleEnv
TARGETS
MultipleEnv
CI
Staging

MultipleEnvironment)


Carrier
12:03 AM
http://192.168.10.10:8080/


Carrier
12:05 AM
App CI


Carrier
12:06 AM
App STG

и имени приложения изменяется для каждой цели :)

Прочитайте Управление несколькими средами с помощью макроса онлайн:
https://riptutorial.com/ru/ios/topic/6849/управление-несколькими-средами-с-помощью-макроса

глава 203: Услуги Safari

Examples

Внедрение SFSafariViewControllerDelegate

Вы должны реализовать `SFSafariViewControllerDelegate` чтобы ваш класс был уведомлен, когда пользователь нажимает кнопку «Готово» на SafariViewController, и вы также можете ее отклонить.

Сначала объявите свой класс для реализации протокола.

```
class MyClass: SFSafariViewControllerDelegate {

}
```

Реализовать метод делегата для уведомления об увольнении.

```
func safariViewControllerDidFinish(controller: SFSafariViewController) {
    // Dismiss the SafariViewController when done
    controller.dismissViewControllerAnimated(true, completion: nil)
}
```

Не забудьте указать свой класс как делегата SafariViewController.

```
let safariVC = SFSafariViewController(URL: yourURL)
safariVC.delegate = self
```

Дополнительные методы делегирования, которые вы можете реализовать:

```
// Called when the initial URL load is complete.
safariViewController(_ controller: SFSafariViewController, didCompleteInitialLoad
didLoadSuccessfully: Bool) { }

// Called when the user taps an Action button.
safariViewController(_ controller: SFSafariViewController, activityItemsFor URL: URL, title:
String?) -> [UIActivity] { }
```

Добавить элементы в список просмотра Safari

Вы можете добавлять элементы в список пользователя для чтения в Safari, вызвав `addItem` метод на `SSReadingList` одноплодного.

```
let readingList = SSReadingList.default()
readingList?.addItem(with: yourURL, title: "optional title", previewText: "optional preview
text")
```

Список Чтения по умолчанию может быть равен `nil` если доступ к списку чтения не разрешен.

Кроме того, вы можете проверить, поддерживает ли список чтения URL-адрес, вызывая `supportsURL` .

```
SSReadingList.default().supportsURL(URL(string: "https://example.com")!)
```

Это вернет либо `true` либо `false` указывая, поддерживается ли данный URL в списке чтения Safari. Используйте это, например, чтобы определить, показывать ли кнопку для добавления URL-адреса в список чтения.

Открыть URL-адрес с помощью SafariViewController

Не забудьте сначала импортировать необходимые рамки.

```
import SafariServices
//Objective-C
@import SafariServices;
```

`SafariViewController` экземпляр экземпляра `SafariViewController` .

```
let safariVC = SFSafariViewController(URL: URL(string: "your_url")!)
//Objective-C
@import SafariServices;
NSURL *URL = [NSURL URLWithString:[NSString stringWithFormat:@"http://www.google.com"]];
SFSafariViewController *sfvc = [[SFSafariViewController alloc] initWithURL:URL];
```

При желании вы также можете указать SafariViewController, чтобы войти в режим чтения, если это возможно, как только это будет сделана загрузка.

```
let safariVC = SFSafariViewController(URL: URL(string: "your_url")!, entersReaderIfAvailable:
true)
//Objective-C
NSURL *URL = [NSURL URLWithString:[NSString stringWithFormat:@"http://www.google.com"]];
SFSafariViewController *sfvc = [[SFSafariViewController alloc] initWithURL:URL
entersReaderIfAvailable:YES];
```

Представить контроллер вида.

```
present(safariVC, animated: true, completion: nil)
//Objective-C
[self presentViewController:sfvc animated:YES completion:nil];
```

Прочитайте Услуги Safari онлайн: https://riptutorial.com/ru/ios/topic/1371/услуги-safari

глава 204: Установить фон обзора

Examples

Установить фон просмотра

Цель C:

```
view.backgroundColor = [UIColor redColor];
```

Swift:

```
view.backgroundColor! = UIColor.redColor()
```

Swift 3

```
view.backgroundColor = UIColor.redColor
```

Заполнить фоновое изображение UIView

Objective-C

```
 UIGraphicsBeginImageContext(self.view.frame.size);
 [[UIImage imageNamed:@"image.png"] drawInRect:self.view.bounds];
 UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
 UIGraphicsEndImageContext();
 self.view.backgroundColor = [UIColor colorWithPatternImage:image];
```

Задать фокус с изображением

```
self.view.backgroundColor = [UIColor colorWithPatternImage:[UIImage
imageNamed:@"Background.png"]];
```

Создание фонового изображения градиента

Чтобы создать фон с градиентом, вы можете использовать класс CAGradientLayer :

Swift 3.1:

```
func createGradient() {
    let caLayer = CAGradientLayer()
    caLayer.colors = [UIColor.white, UIColor.green, UIColor.blue]
    caLayer.locations = [0, 0.5, 1]
    caLayer.bounds = self.bounds
    self.layer.addSublayer(caLayer)
}
```

Это можно вызвать в viewDidLoad () следующим образом:

```
override func viewDidLoad() {
    super.viewDidLoad()
    createGradient()
}
```

Переменные местоположения и границ CAGradientLayer могут принимать несколько значений, чтобы создать слой градиента с тем, сколько цветов вам нужно. Из документации:

> По умолчанию цвета распределяются равномерно по всему слою, но вы можете дополнительно указать местоположения для управления положением цвета через градиент.

Прочитайте Установить фон обзора онлайн: https://riptutorial.com/ru/ios/topic/6854/установить-фон-обзора

глава 205: Участник UITextField

Examples

UITextField - ограничение текстового поля на определенные символы

Если вы хотите выполнить проверку входных данных вашего текстового поля, используйте следующий фрагмент кода:

```
// MARK: - UITextFieldDelegate

let allowedCharacters =
CharacterSet(charactersIn:"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvxyz").inverted


func textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange,
replacementString string: String) -> Bool {

    let components = string.components(separatedBy: allowedCharacters)
    let filtered = components.joined(separator: "")

    if string == filtered {

        return true

    } else {

        return false
    }
}
```

Objective-C

```
#define ACCEPTABLE_CHARACTERS @"0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

- (BOOL)textField:(UITextField *)textField shouldChangeCharactersInRange:(NSRange)range
replacementString:(NSString *)string
{
      NSCharacterSet *cs = [[NSCharacterSet
characterSetWithCharactersInString:ACCEPTABLE_CHARACTERS] invertedSet];

      NSString *filtered = [[string componentsSeparatedByCharactersInSet:cs]
componentsJoinedByString:@""];

      return [string isEqualToString:filtered];
}
```

Кроме того, вы также можете использовать наборы символов, предоставляемые apple для проверки:

Взгляните на https://developer.apple.com/reference/foundation/nscharacterset

```
let allowedCharacters = CharacterSet.alphanumerics.inverted
let allowedCharacters = CharacterSet.capitalizedLetters.inverted
```

Найти следующий тег и управлять клавиатурой

Текстовое поле вызывает разные методы делегатов (только если заданы делегаты). Один из методов делегата, вызванный текстовым полем, - * **- (BOOL) textFieldShouldReturn: (UITextField *) textField***

Этот метод вызывается всякий раз, когда пользователи нажимают кнопку возврата. Используя этот метод, мы можем реализовать любое пользовательское поведение.

Например,

> В приведенном ниже примере следующий ответчик будет обнаружен на основе тега и управления клавиатурой. Здесь 20 - константа, так как тег, назначенный текстовому полю, подобен этому 50,70,90 и т. Д.

Здесь, при поиске нового объекта текстового поля в качестве ответчика, он сделает текущее текстовое поле новым ответчиком и открытой клавиатурой соответственно.

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField {

                NSInteger nextTag = textField.tag+20;
                // Try to find next responder
                UIResponder *nextResponder = [textField.superview viewWithTag:nextTag];
                if (nextResponder)
                {
                    // Found next responder, so set it.
                    [nextResponder becomeFirstResponder];
                }
                else
                {
                    // Not found, so remove keyboard.
                    [textField resignFirstResponder];
                }
                return YES;
            }
```

Действия, когда пользователь начал / закончил взаимодействие с текстовым полем

Для Swift 3.1:

В первом примере можно увидеть, как вы будете перехватывать пользователя, взаимодействующего с текстовым полем во время записи. Аналогично, в UITextFieldDelegate есть методы, которые вызывается , когда пользователь начал и закончил свое взаимодействие с TextField.

Чтобы иметь доступ к этим методам, вам необходимо соответствовать протоколу

UITextFieldDelegate , и для каждого текстового поля , о котором вы хотите получить уведомление, назначьте родительский класс в качестве делегата:

```
class SomeClass: UITextFieldDelegate {

    @IBOutlet var textField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        textField.delegate = self
    }

}
```

Теперь вы сможете реализовать все методы UITextFieldDelegate.

Чтобы получать уведомление, когда пользователь начал редактировать текстовое поле, вы можете реализовать метод textFieldDidBeginEditing (_ :) :

```
func textFieldDidBeginEditing(_ textField: UITextField) {
    // now you can perform some action
    // if you have multiple textfields in a class,
    // you can compare them here to handle each one separately
    if textField == emailTextField {
        // e.g. validate email
    }
    else if textField == passwordTextField {
        // e.g. validate password
    }
}
```

Точно так же, будучи уведомленным, если пользователь закончил взаимодействие с текстовым полем , вы можете использовать метод textFieldDidEndEditing (_ :) :

```
func textFieldDidEndEditing(_ textField: UITextField) {
    // now you can perform some action
    // if you have multiple textfields in a class,
    // you can compare them here to handle each one separately
    if textField == emailTextField {
        // e.g. validate email
    }
    else if textField == passwordTextField {
        // e.g. validate password
    }
}
```

Если вы хотите контролировать, нужно ли TextField начинать / заканчивать редактирование, методы textFieldShouldBeginEditing (_ :) и textFieldShouldEndEditing (_ :) могут использоваться путем возврата true / false на основе вашей необходимой логики.

Прочитайте Участник UITextField онлайн: https://riptutorial.com/ru/ios/topic/7185/участник-uitextfield

глава 206: Учебник AirPrint в iOS

Examples

Печать AirPrint Banner Text

Objective-C

Добавление делегата и текстового форматирования в файл `ViewController.h`

```
@interface ViewController : UIViewController <UIPrintInteractionControllerDelegate> {
    UISimpleTextPrintFormatter *_textFormatter;
}
```

В файле `ViewController.m` определите следующие константы

```
#define DefaultFontSize 48
#define PaddingFactor 0.1f
```

Функция, которая печатает текст, выглядит следующим образом:

```
-(IBAction)print:(id)sender;
{
    /* Get the UIPrintInteractionController, which is a shared object */
    UIPrintInteractionController *controller = [UIPrintInteractionController
sharedPrintController];
    if(!controller){
        NSLog(@"Couldn't get shared UIPrintInteractionController!");
        return;
    }

    /* Set this object as delegate so you can  use the
printInteractionController:cutLengthForPaper: delegate */
    controller.delegate = self;

    UIPrintInfo *printInfo = [UIPrintInfo printInfo];
    printInfo.outputType = UIPrintInfoOutputGeneral;

    /* Use landscape orientation for a banner so the text  print along the long side of the
paper. */
    printInfo.orientation = UIPrintInfoOrientationLandscape;

    printInfo.jobName = self.textField.text;
    controller.printInfo = printInfo;

    /* Create the UISimpleTextPrintFormatter with the text supplied by the user in the text
field */
    _textFormatter = [[UISimpleTextPrintFormatter alloc] initWithText:self.textField.text];

    /* Set the text formatter's color and font properties based on what the user chose */
    _textFormatter.color = [self chosenColor];
    _textFormatter.font = [self chosenFontWithSize:DefaultFontSize];
```

```
    /* Set this UISimpleTextPrintFormatter on the controller */
    controller.printFormatter = _textFormatter;

    /* Set up a completion handler block.  If the print job has an error before spooling, this
is where it's handled. */
    void (^completionHandler)(UIPrintInteractionController *, BOOL, NSError *) =
^(UIPrintInteractionController *printController, BOOL completed, NSError *error) {
        if(completed && error)
            NSLog( @"Printing failed due to error in domain %@ with error code %lu. Localized
description: %@, and failure reason: %@", error.domain, (long)error.code,
error.localizedDescription, error.localizedFailureReason );
    };

    if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad)
        [controller presentFromRect:self.printButton.frame inView:self.view animated:YES
completionHandler:completionHandler];
    else
        [controller presentAnimated:YES completionHandler:completionHandler];  // iPhone
}
```

Функция делегата, которая настраивает страницу печати: -

```
- (CGFloat)printInteractionController:(UIPrintInteractionController
*)printInteractionController cutLengthForPaper:(UIPrintPaper *)paper {

    /* Create a font with arbitrary size so that you can calculate the approximate
        font points per screen point for the height of the text. */
    UIFont *font = _textFormatter.font;
    CGSize size = [self.textField.text sizeWithAttributes:@{NSFontAttributeName: font}];

    float approximateFontPointPerScreenPoint = font.pointSize / size.height;

    /* Create a new font using a size  that will fill the width of the paper */
    font = [self chosenFontWithSize: paper.printableRect.size.width *
approximateFontPointPerScreenPoint];

    /* Calculate the height and width of the text with the final font size */
    CGSize finalTextSize = [self.textField.text sizeWithAttributes:@{NSFontAttributeName:
font}];

    /* Set the UISimpleTextFormatter font to the font with the size calculated */
    _textFormatter.font = font;

    /* Calculate the margins of the roll. Roll printers may have unprintable areas
        before and after the cut.  We must add this to our cut length to ensure the
        printable area has enough room for our text. */
    CGFloat lengthOfMargins = paper.paperSize.height - paper.printableRect.size.height;

    /* The cut length is the width of the text, plus margins, plus some padding */
    return finalTextSize.width + lengthOfMargins + paper.printableRect.size.width *
PaddingFactor;
}
```

Прочитайте Учебник AirPrint в iOS онлайн: https://riptutorial.com/ru/ios/topic/7395/учебник-airprint-в-ios

глава 207: Фильтры CoreImage

Examples

Пример фильтра основного изображения

Objective-C

Просто зарегистрируйтесь, чтобы узнать, как использовать определенный фильтр

```
 NSArray *properties = [CIFilter filterNamesInCategory:kCICategoryBuiltIn];

 for (NSString *filterName in properties)
 {
 CIFilter *fltr = [CIFilter filterWithName:filterName];
 NSLog(@"%@", [fltr attributes]);
 }
```

В случае CISepiaTone системный журнал выглядит следующим образом

```
CIAttributeFilterDisplayName = "Sepia Tone";
CIAttributeFilterName = CISepiaTone;
    CIAttributeReferenceDocumentation = "http://developer.apple.com/cgi-
bin/apple_ref.cgi?apple_ref=//apple_ref/doc/filter/ci/CISepiaTone";
    inputImage =     {
        CIAttributeClass = CIImage;
        CIAttributeDescription = "The image to use as an input image. For filters that also
use a background image, this is the foreground image.";
        CIAttributeDisplayName = Image;
        CIAttributeType = CIAttributeTypeImage;
    };
    inputIntensity =     {
        CIAttributeClass = NSNumber;
        CIAttributeDefault = 1;
        CIAttributeDescription = "The intensity of the sepia effect. A value of 1.0 creates a
monochrome sepia image. A value of 0.0 has no effect on the image.";
        CIAttributeDisplayName = Intensity;
        CIAttributeIdentity = 0;
        CIAttributeMin = 0;
        CIAttributeSliderMax = 1;
        CIAttributeSliderMin = 0;
        CIAttributeType = CIAttributeTypeScalar;
    };
}
```

Используя приведенный выше системный журнал, мы устанавливаем фильтр следующим образом:

```
    CIImage *beginImage = [CIImage imageWithCGImage:[myImageView.image CGImage]];
    CIContext *context = [CIContext contextWithOptions:nil];
    //select Filter Name and Intensity
    CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone" keysAndValues:
```

```
kCIInputImageKey, beginImage, @"inputIntensity", [NSNumber numberWithFloat:0.8], nil];
    CIImage *outputImage = [filter outputImage];

    CGImageRef cgimg = [context createCGImage:outputImage fromRect:[outputImage extent]];
    UIImage *newImg = [UIImage imageWithCGImage:cgimg];

    [myImageView1 setImage:newImg];

    CGImageRelease(cgimg);
```

Изображение, сгенерированное вышеуказанным кодом

Альтернативный способ настройки фильтра

```
UIImageView *imageView1=[[UIImageView alloc]initWithFrame:CGRectMake(0, 0,
self.view.frame.size.width, self.view.frame.size.height/2)];
UIImageView *imageView2=[[UIImageView alloc]initWithFrame:CGRectMake(0,
self.view.frame.size.height/2, self.view.frame.size.width, self.view.frame.size.height/2)];
imageView1.image=[UIImage imageNamed:@"image.png"];

CIImage *beginImage = [CIImage imageWithCGImage:[imageView1.image CGImage]];
CIContext *context = [CIContext contextWithOptions:nil];
//select Filter Name and Intensity
```

```
CIFilter *filter = [CIFilter filterWithName:@"CIColorPosterize"];
[filter setValue:beginImage forKey:kCIInputImageKey];
[filter setValue:[NSNumber numberWithFloat:8.0] forKey:@"inputLevels"];
CIImage *outputImage = [filter outputImage];

CGImageRef cgimg = [context createCGImage:outputImage fromRect:[outputImage extent]];
UIImage *newImg = [UIImage imageWithCGImage:cgimg];

[imageView2 setImage:newImg];

CGImageRelease(cgimg);
[self.view addSubview:imageView1];
[self.view addSubview:imageView2];
```

Изображение сгенерировано с помощью этого кода

Все доступные фильтры:

```
/* CIAccordionFoldTransition,
    CIAdditionCompositing,
    CIAffineClamp,
    CIAffineTile,
    CIAffineTransform,
    CIAreaAverage,
    CIAreaHistogram,
    CIAreaMaximum,
    CIAreaMaximumAlpha,
```

```
    CIAreaMinimum,
    CIAreaMinimumAlpha,
    CIAztecCodeGenerator,
    CIBarsSwipeTransition,
    CIBlendWithAlphaMask,
    CIBlendWithMask,
    CIBloom,
    CIBoxBlur,
    CIBumpDistortion,
    CIBumpDistortionLinear,
    CICheckerboardGenerator,
    CICircleSplashDistortion,
    CICircularScreen,
    CICircularWrap,
    CICMYKHalftone,
    CICode128BarcodeGenerator,
    CIColorBlendMode,
    CIColorBurnBlendMode,
    CIColorClamp,
    CIColorControls,
    CIColorCrossPolynomial,
    CIColorCube,
    CIColorCubeWithColorSpace,
    CIColorDodgeBlendMode,
    CIColorInvert,
    CIColorMap,
    CIColorMatrix,
    CIColorMonochrome,
    CIColorPolynomial,
    CIColorPosterize,
    CIColumnAverage,
    CIComicEffect,
    CIConstantColorGenerator,
    CIConvolution3X3,
    CIConvolution5X5,
    CIConvolution7X7,
    CIConvolution9Horizontal,
    CIConvolution9Vertical,
    CICopyMachineTransition,
    CICrop,
    CICrystallize,
    CIDarkenBlendMode,
    CIDepthOfField,
    CIDifferenceBlendMode,
    CIDiscBlur,
    CIDisintegrateWithMaskTransition,
    CIDisplacementDistortion,
    CIDissolveTransition,
    CIDivideBlendMode,
    CIDotScreen,
    CIDroste,
    CIEdges,
    CIEdgeWork,
    CIEightfoldReflectedTile,
    CIExclusionBlendMode,
    CIExposureAdjust,
    CIFalseColor,
    CIFlashTransition,
    CIFourfoldReflectedTile,
    CIFourfoldRotatedTile,
    CIFourfoldTranslatedTile,
```

```
    CIGammaAdjust,
    CIGaussianBlur,
    CIGaussianGradient,
    CIGlassDistortion,
    CIGlassLozenge,
    CIGlideReflectedTile,
    CIGloom,
    CIHardLightBlendMode,
    CIHatchedScreen,
    CIHeightFieldFromMask,
    CIHexagonalPixellate,
    CIHighlightShadowAdjust,
    CIHistogramDisplayFilter,
    CIHoleDistortion,
    CIHueAdjust,
    CIHueBlendMode,
    CIKaleidoscope,
    CILanczosScaleTransform,
    CILenticularHaloGenerator,
    CILightenBlendMode,
    CILightTunnel,
    CILinearBurnBlendMode,
    CILinearDodgeBlendMode,
    CILinearGradient,
    CILinearToSRGBToneCurve,
    CILineOverlay,
    CILineScreen,
    CILuminosityBlendMode,
    CIMaskedVariableBlur,
    CIMaskToAlpha,
    CIMaximumComponent,
    CIMaximumCompositing,
    CIMedianFilter,
    CIMinimumComponent,
    CIMinimumCompositing,
    CIModTransition,
    CIMotionBlur,
    CIMultiplyBlendMode,
    CIMultiplyCompositing,
    CINoiseReduction,
    CIOpTile,
    CIOverlayBlendMode,
    CIPageCurlTransition,
    CIPageCurlWithShadowTransition,
    CIParallelogramTile,
    CIPDF417BarcodeGenerator,
    CIPerspectiveCorrection,
    CIPerspectiveTile,
    CIPerspectiveTransform,
    CIPerspectiveTransformWithExtent,
    CIPhotoEffectChrome,
    CIPhotoEffectFade,
    CIPhotoEffectInstant,
    CIPhotoEffectMono,
    CIPhotoEffectNoir,
    CIPhotoEffectProcess,
    CIPhotoEffectTonal,
    CIPhotoEffectTransfer,
    CIPinchDistortion,
    CIPinLightBlendMode,
    CIPixellate,
```

```
    CIPointillize,
    CIQRCodeGenerator,
    CIRadialGradient,
    CIRandomGenerator,
    CIRippleTransition,
    CIRowAverage,
    CISaturationBlendMode,
    CIScreenBlendMode,
    CISepiaTone,
    CIShadedMaterial,
    CISharpenLuminance,
    CISixfoldReflectedTile,
    CISixfoldRotatedTile,
    CISmoothLinearGradient,
    CISoftLightBlendMode,
    CISourceAtopCompositing,
    CISourceInCompositing,
    CISourceOutCompositing,
    CISourceOverCompositing,
    CISpotColor,
    CISpotLight,
    CISRGBToneCurveToLinear,
    CIStarShineGenerator,
    CIStraightenFilter,
    CIStretchCrop,
    CIStripesGenerator,
    CISubtractBlendMode,
    CISunbeamsGenerator,
    CISwipeTransition,
    CITemperatureAndTint,
    CIToneCurve,
    CITorusLensDistortion,
    CITriangleKaleidoscope,
    CITriangleTile,
    CITwelvefoldReflectedTile,
    CITwirlDistortion,
    CIUnsharpMask,
    CIVibrance,
    CIVignette,
    CIVignetteEffect,
    CIVortexDistortion,
    CIWhitePointAdjust,
    CIZoomBlur*/
```

Прочитайте Фильтры CoreImage онлайн: https://riptutorial.com/ru/ios/topic/7278/фильтры-coreimage

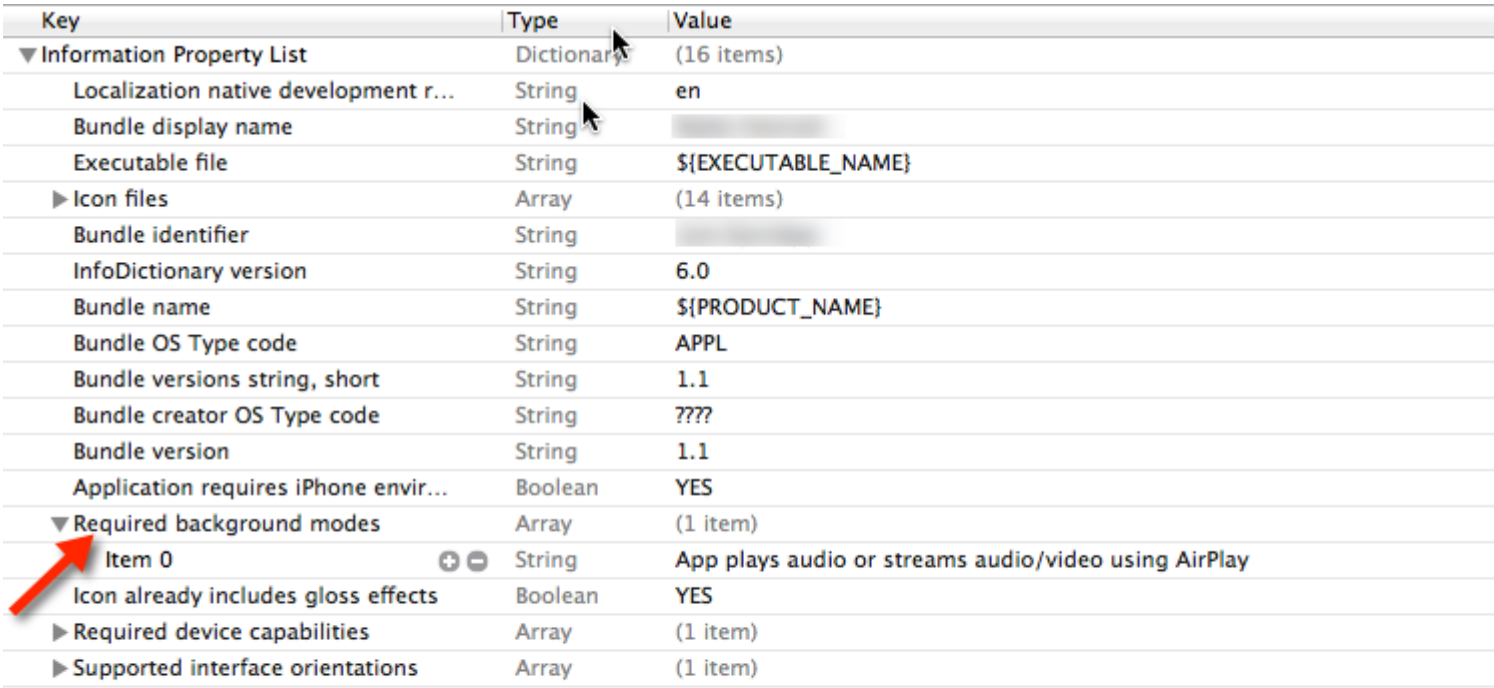
глава 208: Фоновые режимы и события

Examples

Воспроизведение аудиоизображения

Добавьте ключ с именем « **Требуемые фоновые режимы**» в файле свойств (.plist).

как показано ниже.



И добавьте следующий код в

AppDelegate.h

```
#import <AVFoundation/AVFoundation.h>
#import <AudioToolbox/AudioToolbox.h>
```

AppDelegate.m

в приложении didFinishLaunchingWithOptions

```
[[AVAudioSession sharedInstance] setDelegate:self];
[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPlayback error:nil];
[[AVAudioSession sharedInstance] setActive:YES error:nil];
```

```
[[UIApplication sharedApplication] beginReceivingRemoteControlEvents];

UInt32 size = sizeof(CFStringRef);
CFStringRef route;
AudioSessionGetProperty(kAudioSessionProperty_AudioRoute, &size, &route);
NSLog(@"route = %@", route);
```

Если вам нужны изменения в соответствии с событиями, вы должны добавить следующий код в AppDelegate.m

```
- (void)remoteControlReceivedWithEvent:(UIEvent *)theEvent {

    if (theEvent.type == UIEventTypeRemoteControl)    {
        switch(theEvent.subtype)        {
            case UIEventSubtypeRemoteControlPlay:
                [[NSNotificationCenter defaultCenter] postNotificationName:@"TogglePlayPause"
object:nil];
                break;
            case UIEventSubtypeRemoteControlPause:
                [[NSNotificationCenter defaultCenter] postNotificationName:@"TogglePlayPause"
object:nil];
                break;
            case UIEventSubtypeRemoteControlStop:
                break;
            case UIEventSubtypeRemoteControlTogglePlayPause:
                [[NSNotificationCenter defaultCenter] postNotificationName:@"TogglePlayPause"
object:nil];
                break;
            default:
                return;
        }
    }
}
```

На основании уведомления должны работать над этим.

Прочитайте Фоновые режимы и события онлайн: https://riptutorial.com/ru/ios/topic/3515/фоновые-режимы-и-события

глава 209: Цеповые блоки в очереди (с MKBlockQueue)

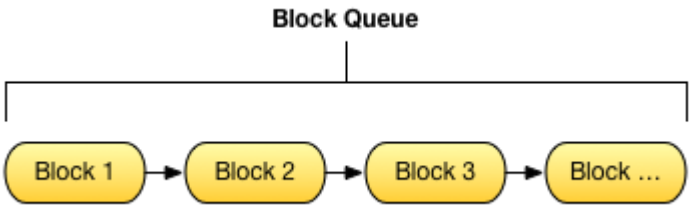
Вступление

MKBlockQueue позволяет создавать цепочку блоков и выполнять их один за другим в очереди. По сравнению с NSOperation, с MKBlockQueue вы сами решаете, когда блок завершен, и когда вы хотите, чтобы очередь продолжалась. Вы также можете передавать данные с одного блока на другой.

https://github.com/MKGitHub/MKBlockQueue

Examples

Пример кода




```
// create the dictionary that will be sent to the blocks
var myDictionary:Dictionary<String, Any> = Dictionary<String, Any>()
myDictionary["InitialKey"] = "InitialValue"

// create block queue
let myBlockQueue:MKBlockQueue = MKBlockQueue()

// block 1
let b1:MKBlockQueueBlockType =
{
    (blockQueueObserver:MKBlockQueueObserver, dictionary:inout Dictionary<String, Any>) in

    print("Block 1 started with dictionary: \(dictionary)")
    dictionary["Block1Key"] = "Block1Value"

    // tell this block is now completed
    blockQueueObserver.blockCompleted(with:&dictionary)

}

// block 2
let b2:MKBlockQueueBlockType =
{
    (blockQueueObserver:MKBlockQueueObserver, dictionary:inout Dictionary<String, Any>) in

    var copyOfDictionary:Dictionary<String, Any> = dictionary
```

```
    // test calling on main thread, async, with delay
    DispatchQueue.main.asyncAfter(deadline:(.now() + .seconds(1)), execute:
    {
        print("Block 2 started with dictionary: \(copyOfDictionary)")

        copyOfDictionary["Block2Key"] = "Block2Value"

        // tell this block is now completed
        blockQueueObserver.blockCompleted(with:&copyOfDictionary)
    })
}

// block 3
let b3:MKBlockQueueBlockType =
{
    (blockQueueObserver:MKBlockQueueObserver, dictionary:inout Dictionary<String, Any>) in

    var copyOfDictionary:Dictionary<String, Any> = dictionary

    // test calling on global background queue, async, with delay
    DispatchQueue.global(qos:.background).asyncAfter(deadline:(.now() + .seconds(1)), execute:
    {
        print("Block 3 started with dictionary: \(copyOfDictionary)")

        copyOfDictionary["Block3Key"] = "Block3Value"

        // tell this block is now completed
        blockQueueObserver.blockCompleted(with:&copyOfDictionary)
    })
}

// add blocks to the queue
myBlockQueue.addBlock(b1)
myBlockQueue.addBlock(b2)
myBlockQueue.addBlock(b3)

// add queue completion block for the queue
myBlockQueue.queueCompletedBlock(
{
    (dictionary:Dictionary<String, Any>) in
    print("Queue completed with dictionary: \(dictionary)")
})

// run queue
print("Queue starting with dictionary: \(myDictionary)")
myBlockQueue.run(with:&myDictionary)
```

Прочитайте Цеповые блоки в очереди (с MKBlockQueue) онлайн:
https://riptutorial.com/ru/ios/topic/9122/цеповые-блоки-в-очереди--с-mkblockqueue-

глава 210: Широкий спектр приложений

Examples

Получить самый высокий UIViewController

Общий подход к получению самого верхнего `UIViewController` заключается в получении RootViewController вашего активного `UIWindow` . Я написал расширение для этого:

```
extension UIApplication {

func topViewController(_ base: UIViewController? =
UIApplication.shared.keyWindow?.rootViewController) -> UIViewController {

    if let nav = base as? UINavigationController {
        return topViewController(nav.visibleViewController)
    }

    if let tab = base as? UITabBarController {
        if let selected = tab.selectedViewController {
            return topViewController(selected)
        }
    }

    if let presented = base?.presentedViewController {
        return topViewController(presented)
    }

    return base!
}
```

Перехват системных событий

Используя NotificationCenter iOS, который может быть очень мощным, вы можете перехватить определенные события приложения:

```
NotificationCenter.default.addObserver(
        self,
        selector: #selector(ViewController.do(_:)),
        name: NSNotification.Name.UIApplicationDidBecomeActive,
        object: nil)
```

Вы можете зарегистрироваться для большего количества событий, просто взгляните на https://developer.apple.com/reference/foundation/nsnotification.name .

Прочитайте Широкий спектр приложений онлайн: https://riptutorial.com/ru/ios/topic/7188/широкий-спектр-приложений

кредиты

S. No	Главы	Contributors
1	Начало работы с iOS	Ali Beadle, Allan Burleson, Anand Nimje, Anatoliy, Ashutosh Dave, Bhadresh Kathiriya, bjtitus, Blachshma, bmike, Charlie H, Cin316, Community, Dair, dan, deyanm , Efraim Weiss, Erik Godard, FelixSFD, Fogmeister, Hudson Taylor, Irfan, J F, Jack Ngai, James, Josh Brown, jrf, Kampai, Kevin, Losiowaty, M. Galban, Maddyゾゾ, Matthew Cawley, Md. Ibrahim Hassan, Midhun MP, Miguel Cabezas, Muhammad Zohaib Ehsan, Pro Q, PSN, RamenChef, Sam Fischer, Seyyed Parsa Neshaei, shim, Skeleton Bow, Stephen Leppik, Steve Moser, Suragch, SuzGupta, The_Curry_Man, ThrowingSpoon, Undo, user3480295, user6939352, Vignan
2	3D Touch	4444, Harshal Bhavsar, LinusGeffarth, Md. Ibrahim Hassan, Onur Tuna, Stephen Leppik, tobeiosdeveloper
3	AFNetworking	4444, Mayank Patel, OhadM, Ruby
4	Alamofire	Alex Koshy, Josh Caswell, Sour LeangChhean, yogesh wadhwa
5	API iOS Google Places	Cyril Ivar Garcia, Vignan
6	API распознавания речи IOS 10	rohit90, Stephen Leppik
7	AppDelegate	CodeChanger, Oleh Zayats, Saumil Shah
8	ARC (автоматический подсчет ссылок)	4444, Irfan, John Militer, Ketan P, Tricertops
9	attribitedText в UILabel	vp2698
10	AVPlayer и AVPlayerViewController	Bonnie, Chirag Desai, Gazi Alankus, Harshal Bhavsar, Konda Yadav, Stephen Leppik
11	AVSpeechSynthesizer	Ali Beadle, Bhumit Mehta, Harshal Bhavsar, Midhun MP, Stephen Leppik
12	AWS SDK	OhadM
13	CAAnimation	Bhavin Ramani, James P, Mr. Xcoder, Narendra Pandey,

		Rahul, Rob, Undo
14	CAGradientLayer	Bhavin Ramani, Harshal Bhavsar, Sam Fischer, Stephen Leppik, Undo
15	CALayer	Alistra, Dunja Lalic, HariKrishnan.P, Harshal Bhavsar, ignotusverum, iOS BadBoy, Kamil Harasimowicz, Luiz Henrique Guimaraes, Stephen Leppik, Suragch, Viktor Simkó, william205
16	Carthage iOS Setup	Md. Ibrahim Hassan
17	CAShapeLayer	Filip Radelic, HariKrishnan.P, Harshal Bhavsar, Narendra Pandey, Stephen Leppik
18	CGContext Reference	4444, Narendra Pandey
19	CLLocation	amar, Duly Kinsky, FelixSFD, Siddharth Sunil, Sujania, That lazy iOS Guy , void, Zee
20	CloudKit	Seyyed Parsa Neshaei
21	Core Motion	Md. Ibrahim Hassan, RamenChef
22	Core SpotLight в iOS	Md. Ibrahim Hassan
23	CTCallCenter	MANI, Md. Ibrahim Hassan, OhadM
24	CydiaSubstrate tweak	gkpln3
25	DispatchGroup	Brandon, Fonix
26	EventKit	Seyyed Parsa Neshaei
27	FacebookSDK	Brian, Harshal Bhavsar, Irfan, Mehul Chuahan, OhadM, Ravi Prakash Verma, Stephen Leppik
28	Fastlane	J F, KrauseFx, SM18, tharkay
29	FileHandle	Nikhlesh Bagdiya
30	GameplayKit	BennX, Seyyed Parsa Neshaei
31	GCD (Grand Central Dispatch)	Andrea Antonioni, DS Dharma, Fonix, Md. Ibrahim Hassan, skyline75489
32	Healthkit	Md. Ibrahim Hassan
33	IBeacon	amar, Arefly, Harshal Bhavsar, Stephen Leppik
34	IBOutlets	Fabio, SharkbaitWhohaha

35	iOS - Внедрение XMPP с платформой Робби Хэнсона	Saheb Roy
36	iOS TTS	Ali Abbas, Stephen Leppik
37	MKDistanceFormatter	Harshal Bhavsar, Md. Ibrahim Hassan, Stephen Leppik, Undo
38	MKMapView	Arnon Rodrigues, Brian, FelixSFD, Harshal Bhavsar, Kosuke Ogawa, Mahesh, Mehul Thakkar, Ortwin Gentz, Reinier Melian, Stephen Leppik
39	ModelPresentationStyles	Dishant Kapadiya
40	MPMediaPickerDelegate	FelixSFD, George Lee
41	MPVolumeView	lostAtSeaJoshua
42	MVVM	JPetric
43	MyLayout	
44	NSArray	Krunal, user5553647
45	NSAttributedString	Bhavin Ramani, Harshal Bhavsar, Jinhuan Li, Kirit Modi, Luiz Henrique Guimaraes, Mansi Panchal, Stephen Leppik, Tim, Tim Ebenezer, Undo
46	NSBundle	wdywayne
47	NSData	Felipe Cypriano, maxkonovalov, Seyyed Parsa Neshaei
48	NSDate	Bonnie, Charles, dasdom, Dunja Lalic, ERbittuu, FelixSFD, Harshal Bhavsar, Jon Snow, Josh Caswell, lostAtSeaJoshua, maxkonovalov, Mehul Thakkar, NSNoob, Nykholas, OhadM, Sally, Samuel Teferra, Sandy, Seyyed Parsa Neshaei, Stephen Leppik, tharkay, tobeiosdeveloper
49	NSHTTPCookieStorage	balagurubaran
50	NSInvocation	Md. Ibrahim Hassan
51	NSNotificationCenter	Alex Kallam, Alex Koshy, Anand Nimje, Bence Pattogato, Bright Future, Ichthyocentaurs, Jacopo Penzo, James P, Kirit Modi, Tarun Seera
52	NSPredicate	Brendon Roberto, Joshua, Mehul Chuahan

53	NSTimer	AJ9, James P, Maddyゾゾ, Samuel Teferra, tfrank377, That lazy iOS Guy , Undo, william205
54	NSURL	Adnan Aftab, ApolloSoftware, tharkay
55	NSURLConnection	byJeevan
56	NSURLSession	bluey31, dasdom, dgatwood, Duly Kinsky, Harshal Bhavsar, Narendra Pandey, Otávio, R P, sage444, Stephen Leppik
57	NSUserActivity	Samuel Spencer
58	NSUserDefaults	Anand Nimje, Emptyless, Harshal Bhavsar, Husein Behboodi Rad, J F, James P, Josh Caswell, Kirit Modi, Mr. Xcoder, Roland Keesom, Seyyed Parsa Neshaei, user3760892, william205
59	OpenGL	Fonix
60	plist iOS	SNarula
61	Sirikit	Seyyed Parsa Neshaei
62	SLComposeViewController	Md. Ibrahim Hassan
63	StoreKit	askielboe
64	Swift: изменение rootViewController в AppDelegate для представления основного или входного / входного потока	cleverbit
65	SWRevealViewController	Reinier Melian, tharkay
66	UIActivityViewController	Amandeep, Harshal Bhavsar, Stephen Leppik, Vivek Molkar
67	UIAlertController	Andrii Chernenko, Arefly, Bhavin Ramani, FelixSFD, Harshal Bhavsar, Irfan, juliand665, Kirit Modi, Muhammad Zohaib Ehsan, Narendra Pandey, Nikita Kurtin, NSNoob, pableiros, Senseful, Seyyed Parsa Neshaei, shim, Stephen Leppik, Sunil Sharma, Suragch, user3480295
68	UIAppearance	azimov, Harshal Bhavsar, Stephen Leppik, Undo
69	UIBarButtonItem	Ahmed Khalaf, Dunja Lalic, hgwhittle, Suragch, william205

70	UIBezierPath	Bean, Igor Bidiniuc, Suragch, Teja Nandamuri
71	UIButton	Aleksei Minaev, Arefly, dasdom, ddb, Fabio Berger, FelixSFD, fredpi, James, James P, Jojodmo, Joshua, mattblessed, Mr. Xcoder, mtso, Nate Lee, NSNoob, P. Pawluś, Quantm, RamenChef, Roland Keesom, Sachin S P, tharkay, Viktor Simkó, william205, WMios
72	UICollectionView	Adam Eberbach, AJ9, Alex Koshy, Anand Nimje, Anh Pham, Bhavin Ramani, Bhumit Mehta, Brian, Dalija Prasnikar, ddb, Dima Deplov, Harshal Bhavsar, Kevin DiTraglia, Koushik, Mark, Rodrigo de Santiago, Stephen Leppik, Suragch, Undo
73	UIColor	Amanpreet, Anh Pham, Avineet Gupta, Brett Ponder, Cin316, Community, dasdom, DeyaEldeen, Douglas Hill, Elias Datler, Fabio Berger, FelixSFD, Gary Riches, Harshal Bhavsar, Honey, ing0, iphonic, Irfan, JAL, Jaleel Nazir, Jojodmo, Luca D'Alberti, maxkonovalov, mtso, nielsbot, NSNoob, pableiros, Reinier Melian, Rex, Sally, Samer Murad, Sandy, shim, The_Curry_Man, Tommie C., Viktor Simkó, WMios, Yagnesh Dobariya
74	UIControl - Обработка событий с помощью блоков	Brandon
75	UIDatePicker	Pavel Gatilov
76	UIDevice	Bhavin Ramani, FelixSFD, Md. Ibrahim Hassan, Mehul Chuahan, Nef10, pableiros, Ramkumar chintala
77	UIFeedbackGenerator	beyowulf
78	UIFont	Mr. Xcoder
79	UIGestureRecognizer	Adam Preble, dannyzlo, Dunja Lalic, Harshal Bhavsar, John Leonardo, Josh Caswell, Md. Ibrahim Hassan, Ruby , Stephen Leppik, Sujania, Suragch, Undo
80	UIImage	Adrian Schönig, Alexander Tkachenko, Bean, Bhavin Ramani, Dipen Panchasara, Dunja Lalic, Emptyless, FelixSFD, Harshal Bhavsar, Heberti Almeida, Jimmy James, Mahmoud Adam, maxkonovalov, Md. Ibrahim Hassan, Muhammad Zeeshan, RamenChef, Reinier Melian, Rex, rob180, Ronak Chaniyara, sage444, Sandy, Seyyed Parsa Neshaei, Sujania, Sunil Sharma, The_Curry_Man, user3480295, Vineet Choudhary

81	UIImagePickerController	Brian, stonybrooklyn, william205
82	UIImageView	Adam Eberbach, Anh Pham, Bean, Caleb Kleveter, DeyaEldeen, Dunja Lalic, FelixSFD, il Malvagio Dottor Prosciutto, Irfan, Joshua, mattblessed, Md. Ibrahim Hassan, njuri, Quantm, Reinier Melian, Rex, Rob, Samuel Spencer, Sunil Sharma, Suragch, william205
83	UILabel	4oby, Akilan Arasu, Alex Koshy, alvarolopez, Andres Canella, Andrii Chernenko, Anh Pham, Ashwin Ramaswami, AstroCB, Barlow Tucker, bentford, Bhumit Mehta, Brian, byJeevan, Caleb Kleveter, Chathuranga Silva, Chris Brandsma, Cin316, Code.Warrior, Community, Daniel Bocksteger, Daniel Stradowski, danshevluk, dasdom, ddb, DeyaEldeen, Dunja Lalic, Eric, Erwin, esthepiking, Fabio Berger, Fahim Parkar, Felix, FelixSFD, Franck Dernoncourt, gadu, ggrana, GingerHead, gvuksic, HaemEternal, hankide, Hans Sjunnesson, Harshal Bhavsar, Hossam Ghareeb, idobn, Imanou Petit, iOS BadBoy, iphonic, Irfan, J F, Jacky, Jacobanks, johnpenning, Jojodmo, Josh Brown, Joshua, Joshua J. McKinnon, jtbandes, juanjo, kabiroberai, Kai Engelhardt, KANGKANG, Khanh Nguyen, Kireyin, leni, Luca D'Alberti, lufritz, Lukas, Luke Patterson, Lumialxk, Mad Burea, Mahmoud Adam, Md. Ibrahim Hassan, Moshe, Nadzeya, Narendra Pandey, Nathan Levitt, Nirav D, njuri, noelicus, NSNoob, Ollie, Quantm, Radagast the Brown, Rahul Vyas, RamenChef, ramsserio, rfarry, sage444, Scotow, Seyyed Parsa Neshaei, Shahabuddin Vansiwala, solidcell, Sravan, stackptr, Sunil Sharma, Suragch, sushant jagtap, TDM, tharkay, The_Curry_Man, Tibor Molnár, Tyler, Undo, user3480295, vasili111, Vignan, Viktor Simkó, william205, WMios, Yagnesh Dobariya
84	UILocalNotification	Bhumit Mehta, Brian, Byte1518, D4ttatraya, David, ElonChan, Harshal Bhavsar, hgwhittle, kamwysoc, KrishnaCA, rajesh sukumaran, Rex, Samuel Spencer, themathsrobot, tktsubota, william205, Wolverine, Xenon
85	UINavigationController	dasdom, Oleh Zayats, sage444, Suragch, william205, WMios
86	UIPageViewController	azimov, Bright Future, Harshal Bhavsar, Mayuri R Talaviya, Stephen Leppik, stonybrooklyn, Victor M
87	UIPheonix - простой, гибкий, динамичный и	StackUnderflow

	масштабируемый интерфейс пользовательского интерфейса	
88	UIPickerView	FelixSFD, Hasintha Janka, MCMatan, Md. Ibrahim Hassan, Moritz, NinjaDeveloper
89	UIRefreshControl TableView	Md. Ibrahim Hassan, Mohammad Rana
90	UIScrollView	Bhavin Ramani, LinusGeffarth, maxkonovalov, Rex, sanman, Sujania, Sunil Sharma, Suragch, tharkay, torinpitchers
91	UIScrollView AutoLayout	Aaron, Brandon, Shrikant K
92	UIScrollView с дочерним элементом StackView	mourodrigo
93	UISearchController	Harshal Bhavsar, Mehul Chuahan, mtso, Stephen Leppik, Tarvo Mäesepp
94	UISegmentedControl	Kamil Harasimowicz
95	UISlider	Andreas, Md. Ibrahim Hassan
96	UISplitViewController	Cerbrus, Koushik
97	UIStackView	Anuj Joshi, danshevluk, Harshal Bhavsar, Kof, Lior Pollak , Sally, sasquatch, Stephen Leppik, william205
98	UIStoryboard	Adriana Carelli, Mr. Xcoder, Vignan
99	UISwitch	Bhavin Ramani, FelixSFD, Md. Ibrahim Hassan, Mr. Xcoder, RamenChef, Sujay
100	UITabBarController	Alexi, Anand Nimje, Cristina, Mehul Chuahan, Quantm, Srinija
101	UITableView	AJ9, Alex Koshy, Andres Kievsky, Anh Pham, animuson, Bean, Brendon Roberto, Brian, dasdom, DeyaEldeen, Dima Deplov, Dunja Lalic, Erik Godard, Glorfindel, Harshal Bhavsar, Jojodmo, Kof, Luca D'Alberti, Luis, Meng Zhang, Nathan, Nirav Bhatt, Nirav D, RamenChef, Rex, RodolfoAntonici, Ruby, Samuel Spencer, Seslyn, simple_code, Srinija, Steve Moser, Sujania, Sujay, Suragch, Tamarous, user3480295

102	UITableViewCell	Rahul
103	UITableViewController	Aju
104	UITextField	Alex Koshy, Ali Elsokary, Ashvinkumar, Duly Kinsky, Fabio Berger, FelixSFD, J F, Joshua, Kof, Luiz Henrique Guimaraes, Maddyゾゾ, P. Pawluś, RamenChef, Reinier Melian, Ruby, samwize, sasquatch, shim, SourabhV, Suragch, sushant jagtap, tharkay, william205, WMios
105	UITextView	Anh Pham, animuson, Bole Tzar, Bright Future, Cris , Dunja Lalic, Eonil, gadu, Harshal Bhavsar, Hejazi, Md. Ibrahim Hassan, njuri, Roland Keesom, Ruby, Suragch, sushant jagtap, william205, WMios
106	UIView	Adam Preble, alaphao, Anh Pham, Caleb Kleveter, Community, Cory Wilhite, D4ttatraya, ddb, DeyaEldeen, Douglas Starnes, hgwhittle, iphonic, Irfan, James, Jojodmo, Jota, Kotha Sai Ram, Luca D'Alberti, maxkonovalov, Md. Ibrahim Hassan, muazhud, Narendra Pandey, Nikhil Manapure, NSNoob, pableiros, pckill, Peter DeWeese, Rahul Vyas, sasquatch, shallowThought , Sunil Sharma, That lazy iOS Guy , The_Curry_Man, Viktor Simkó, william205
107	UIViewController	dasdom, Dunja Lalic, shim, Suragch, tassinari, william205
108	UIWebView	Allan Burleson, dchar4life80X, iOS BadBoy, J F, Julian135, KANGKANG, Kevin DiTraglia, maxkonovalov, Md. Ibrahim Hassan, Ortwin Gentz, Ramkumar chintala, Sunil Sharma
109	UUID (универсальный уникальный идентификатор)	Anand Nimje, FelixSFD, Harshal Bhavsar, James P, Mehul Chuahan, Rahul Vyas, Seyyed Parsa Neshaei, shim, Stephen Leppik, sushant jagtap
110	WCSessionDelegate	pkc456
111	WKWebView	Brandon, byJeevan, Mahmoud Adam, Yevhen Dubinin
112	Xcode Build & Archive из командной строки	Kyle Decot, Shardul
113	Автоматическая компоновка	alaphao, amar, Anuj Joshi, Bean, Bhumit Mehta, BlackDeveraux, dasdom, Dennis, Dima Deplov, Dinesh Raja, Đông An, Harshal Bhavsar, Hasintha Janka, Irfan, Jano, juanjo, keithbhunter, Mahesh, Mert Buran, Mr. Xcoder, NSNoob, ozgur, Pärserk, Rajesh, Sally, Sandy, Stephen Leppik, Suragch, Undo, user3480295, Vignan

114	Архитектура MVP	Oleh Zayats
115	Безопасность	D4ttatraya
116	Безопасность транспорта приложений (ATS)	breakingobstacles, D4ttatraya, esthepiking, FelixSFD, Mehul Chuahan, nathan
117	блок	4444, animuson, Joshua, Mehul Chuahan, Ruby, Tamarous, user459460
118	Богатые уведомления	Koushik
119	Брелок	abjurato, avojak, Matthew Seaman, Mehul Chuahan
120	Быстрая и объективная совместимость	Harshal Bhavsar, njuri, Stephen Leppik
121	Время выполнения в Objective-C	halil_g
122	Всплывающие уведомления	Amanpreet, Anh Pham, Ashish Kakkad, Bhadresh Kathiriya, BloodWoork, Bonnie, Honey, Hossam Ghareeb , iOS BadBoy, J F, Patrick Beard, Pavel Gurov, sanman, Seyyed Parsa Neshaei, tilo
123	Вырезать UIImage в круг	Md. Ibrahim Hassan
124	Глубокая привязка в iOS	bryanjclark, Dunja Lalic, FelixSFD, sanman
125	График (Coreplot)	MarmiK, Md. Ibrahim Hassan
126	десантный	FelixSFD, Md. Ibrahim Hassan
127	Динамика UIKit	beyowulf, Mark Stewart, Md. Ibrahim Hassan
128	Динамика UIKit с UICollectionView	beyowulf
129	Динамический тип	Alvin Abia, H. M. Madrone, Harshal Bhavsar, James P, Stephen Leppik
130	Динамическое обновление UIStackView	Harshal Bhavsar, Rahul, Stephen Leppik
131	ДОБАВЛЕНИЕ СВЕТОВОГО МОСТОВОГО ГОЛОВА	yogesh wadhwa

132	доступность	Harshal Bhavsar, Justin, Ruby, Stephen Leppik, Zev Eisenberg
133	Загрузка изображений async	J.Paravicini
134	Изменение размера UIImage	Rahul
135	Изменить цвет строки состояния	Alex Rouse, danshevluk, Harshal Bhavsar, Mr. Xcoder, shim, Stephen Leppik, Steve Moser, william205, WMios
136	имитатор	Seyyed Parsa Neshaei
137	Имитация местоположения с использованием файлов GPX iOS	Uma
138	Инициализация идиом	Jano
139	Интеграция SqlCipher	Nirav
140	Использование Image Aseets	D4ttatraya
141	категории	Faran Ghani, simple_code
142	Классы классов и адаптивность	Tim
143	Кодируемый	Ashish Kakkad
144	Кэширование онлайн-изображений	Md. Ibrahim Hassan
145	Лидеры игр GameCenter	4444, Cyril Ivar Garcia, Harshal Bhavsar, Stephen Leppik
146	локализация	4444, animuson, Joshua, Ruby, WMios
147	Многоадресные делегаты	Rahul
148	Наблюдение за ключевыми значениями	D4ttatraya, Harshal Bhavsar, Mehul Chuahan, Mihriban Minaz, Mithrandir, Muhammad Zohaib Ehsan, Pärserk, sanman
149	Настройка маяков с	Beto Caldas

	помощью CoreBluetooth	
150	область	subv3rsion
151	Обмен сообщениями FCM в Swift	Saeed-rz
152	Обнаружение лица с использованием CoreImage / OpenCV	Md. Ibrahim Hassan
153	Обработка URL-адресов	azimov, Brian, Dunja Lalic, Harshal Bhavsar, James P, Stephen Leppik
154	Основная графика	Dunja Lalic, Josh Caswell, Seyyed Parsa Neshaei, Sunil Sharma, Unheilig
155	Основные данные	Ankit chauhan, Md. Ibrahim Hassan
156	Основные текстовые файлы ввода / вывода	Idan
157	Отладка сбоев	NobodyNada
158	Оценка заявки / запроса заявки	Abhijit
159	Панель навигации	Md. Ibrahim Hassan, Mehul Chuahan
160	Передача данных между контроллерами просмотра	Arulkumar, Ashish Kakkad, BorisE, Bright Future, Dima Deplov, dispute, FelixSFD, Honey, ignotusverum, Irfan, Jake Runzer, juanjo, Kasun Randika, Kendall Lister, Kyle KIM, Luca D'Alberti, muazhud, OhadM, RamenChef, rustproofFish, salabaha, StackUnderflow, Steve Moser, Suragch, Tamarous, timbroder, Undo, WMios, Yagnesh Dobariya
161	Передача данных между контроллерами просмотра (с помощью MessageBox-Concept)	StackUnderflow
162	перетекает	Daniel Ormeño
163	Подписание кода	HaemEternal
164	Покупки в приложении	Cyril Ivar Garcia, Martin, rigdonmr, WMios

165	Пользовательская клавиатура	Md. Ibrahim Hassan
166	Пользовательские UIViews из XIB-файлов	backslash-f, Code.Warrior, Harshal Bhavsar, idocode, Nirav Bhatt, Sharpkits Innovations, Stephen Leppik
167	Пользовательские методы выбора UITableViewCells	Kamil Harasimowicz
168	Пользовательские шрифты	Alexi, Dima Deplov, Harshal Bhavsar, Maddyツ, njuri, Stephen Leppik, Tommie C.
169	Пользовательский UITextField	D4ttatraya
170	Преобразование HTML в строку NSAttributed и наоборот	Md. Ibrahim Hassan
171	Преобразование NSAttributedString в UIImage	Md. Ibrahim Hassan
172	Проверка версии iOS	Bhavin Ramani, byJeevan, James P, Joshua, njuri, Samuel Teferra, Sandy
173	Проверка сетевого подключения	ajmccall, breakingobstacles, Mick MacCallum, pableiros, sushant jagtap
174	Профиль с инструментами	Vinod Kumar
175	Процесс подачи заявки	Nermin Sehic
176	Рамка XCTest - Единичное тестирование	D4ttatraya, dasdom, Jan ATAC, Josh Brown, msohng, Raphael Silva, Seyyed Parsa Neshaei, Tarun Seera
177	Раскадровка	Harshal Bhavsar, Kirit Vaghela, Stephen Leppik, Tommie C.
178	Расположение ядра	Harshal Bhavsar, Mayuri R Talaviya, Mehul Chuahan, mtso, quant24, Stephen Leppik, sushant jagtap, william205
179	Расширение для расширенного Push-уведомления - iOS 10.	Oleh Zayats

180	Режимы фона	Seyyed Parsa Neshaei
181	Руководство по выбору лучших шаблонов архитектуры iOS	Phani Sai
182	Связанные объекты Objective-C	Noam
183	Сделайте округлые углы UIView	Md. Ibrahim Hassan
184	Сжатие содержимого / сжатие содержимого в автозапуске	Mehul Chuahan
185	Симулятор строит	Durai Amuthan.H
186	Сканер QR-кода	Bluewings, Efraim Weiss
187	Снимок UIView	Bright Future, Darshit Shah, Md. Ibrahim Hassan, SpaceDog
188	совпадение	Doc, Fonix, Juan Campa, Kevin DiTraglia, Tien
189	Создайте файл .ipa для загрузки в appstore с помощью Applicationloader	Anuj Joshi
190	Создание PDF в iOS	Mansi Panchal, Narendra Pandey
191	Создание видео с изображений	Tiko
192	Создание идентификатора приложения	yogesh wadhwa
193	Создание пользовательской структуры в iOS	Saeed-rz
194	Структура контактов	Md. Ibrahim Hassan, Seyyed Parsa Neshaei
195	Текст UILabel подчеркивает	Md. Ibrahim Hassan

196	Тестирование пользовательского интерфейса	P. Pawluś
197	Универсальные ссылки	Harshal Bhavsar, Irfan, satheeshwaran, Stephen Leppik, Vineet Choudhary
198	Управление клавиатурой	Alexander Tkachenko, Greg, Harshal Bhavsar, Mr. Xcoder, Richard Ash, Shog9, slxl, Stephen Leppik, Steve Moser, Suragch, V1P3R, william205, WMios
199	Управление несколькими средами с помощью макроса	Tien
200	Услуги Safari	Arnon Rodrigues, Harshal Bhavsar, Kilian Koeltzsch, Md. Ibrahim Hassan, Stephen Leppik
201	Установить фон обзора	Adriana Carelli, Andreas, Bhadresh Kathiriya, Harshal Bhavsar, Md. Ibrahim Hassan, user459460
202	Участник UITextField	Andreas, animuson, Md. Ibrahim Hassan, midori, Ruby
203	Учебник AirPrint в iOS	Md. Ibrahim Hassan
204	Фильтры CoreImage	Md. Ibrahim Hassan
205	Фоновые режимы и события	Ashish Kakkad
206	Цеповые блоки в очереди (с MKBlockQueue)	StackUnderflow
207	Широкий спектр приложений	midori