EBook Gratuito

APPRENDIMENTO iOS

Free unaffiliated eBook created from
Stack Overflow contributors.

#ios

Sommario

Di

Capitolo 1: Inizia con iOS

Osservazioni

Gli appunti

1- Non è necessario un account sviluppatore Apple per iniziare a sviluppare app iOS. La documentazione e gli strumenti sono gratuiti per il download con il tuo ID Apple. Puoi anche firmare e installare app sui *tuoi dispositivi personali* usando lo stesso ID Apple. Se desideri distribuire o vendere app su App Store , devi iscrivere il Programma per gli sviluppatori Apple a partire da 99 USD (questo è il prezzo al momento della stesura e potrebbe cambiare). Questo aggiungerà anche incidenti di supporto a livello di codice e beta test per le tue app tramite TestFlight.

2- La creazione di un ID Apple senza carta di credito richiede una procedura breve . Se non ti dispiace associare un metodo di pagamento come parte della registrazione, vai a https://appleid.apple.com/

- Inizia a sviluppare app iOS (Swift)
- Aiuto Xcode (incluso Guida introduttiva)
- Download (incluso Xcode se non si desidera accedere all'AppStore)

Tag di overflow relativi allo stack

- xcode IDE di Apple (Integrated Development Environment) per lo sviluppo di app iOS e macOS
- swift-language Una delle principali lingue che è possibile utilizzare per sviluppare in iOS.
- linguaggio obiettivo-obiettivo Una delle principali lingue che è possibile utilizzare per sviluppare in iOS.
- cacao Un'API Apple per lo sviluppo in iOS e macOS.
- sprite-kit Per grafica animata 2D.
- core-data Per archiviare e recuperare i dati relazionali.

Versioni

Versione	Data di rilascio
iPhone OS 2	2008-07-11
iPhone OS 3	2009-06-17
iOS 4	2010-06-08

Versione	Data di rilascio
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

Creazione di un'applicazione di visualizzazione singola predefinita

Per sviluppare un'applicazione per iOS, dovresti iniziare con un'applicazione chiamata Xcode. Ci sono altri strumenti alternativi che puoi usare, ma Xcode è lo strumento ufficiale di Apple. Si noti, tuttavia, che funziona solo su macOS. L'ultima versione ufficiale è Xcode 8.3.3 con Xcode 9 (attualmente in beta) che dovrebbe essere rilasciato entro la fine dell'anno.

1. Avvia il tuo Mac e installa Xcode dall'App Store se non è già installato.

   (Se preferisci non utilizzare l'App Store o hai problemi, puoi anche scaricare Xcode dal sito

Web di Apple Developer , ma assicurati di selezionare la versione più recente e **non** una versione beta.)

2. Apri Xcode. Si aprirà la seguente finestra:

Welcome to Xc

Version 8.0 (8A218a)

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

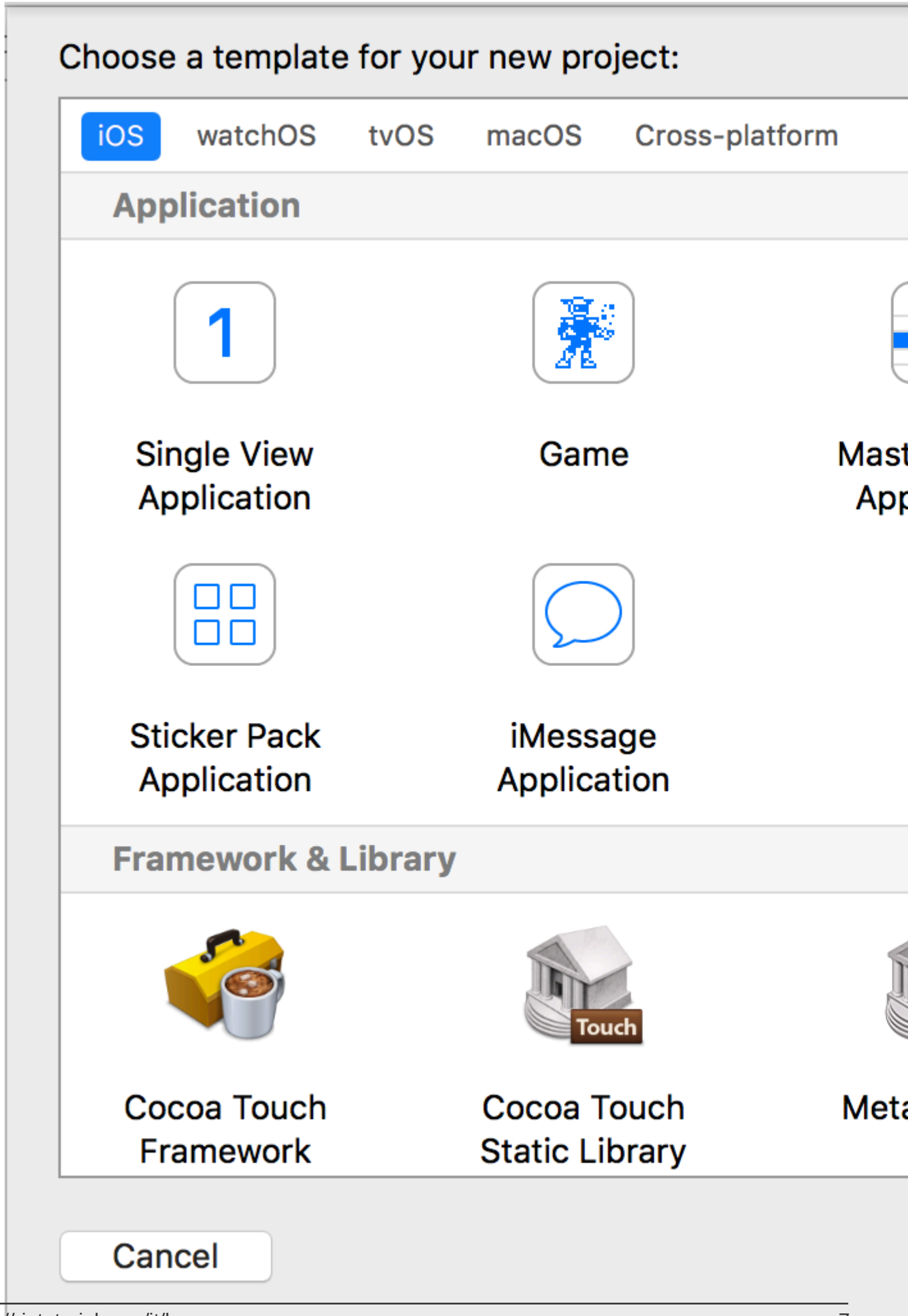
Create a new Xcode project
Create an app for iPhone, iPad, Mac, Ap

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

La finestra ti offre le seguenti opzioni:

- **Iniziare con un parco giochi: è** stato introdotto con il linguaggio Swift e Xcode 6. Si tratta di un'area interattiva che può essere utilizzata per scrivere piccoli pezzi di codice per controllare le modifiche al runtime. È un ottimo modo per far conoscere agli studenti Swift le nuove funzionalità di Swift.
  stato introdotto con il linguaggio Swift e Xcode 6. Si tratta di un'area interattiva che può essere utilizzata per scrivere piccoli pezzi di codice per controllare le modifiche al runtime. È un ottimo modo per far conoscere agli studenti Swift le nuove funzionalità di Swift.
- **Crea un nuovo progetto Xcode:** *scegli questa opzione* , che crea un nuovo progetto con configurazione predefinita.
- **Scopri un progetto esistente:** questo è usato per controllare un progetto da una posizione di repository, per esempio, controlla un progetto da SVN.

3. Seleziona la seconda opzione **Crea un nuovo progetto Xcode** e Xcode ti chiederà di eseguire alcune impostazioni iniziali del progetto:


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

Questa procedura guidata viene utilizzata per selezionare il modello del progetto. Ci sono 5 opzioni:

- **iOS:** utilizzato per creare app, librerie e framework iOS
- **watchOS:** utilizzato per creare app, librerie e framework watchOS
- **tvOS:** utilizzato per creare app, librerie e framework per TVOS
- **macOS:** utilizzato per creare app, librerie, framework, pacchetti, AppleScript, ecc. macOS
  utilizzato per creare app, librerie, framework, pacchetti, AppleScript, ecc. macOS
- **Multipiattaforma:** utilizzato per creare app multipiattaforma, modelli e contenuti di acquisto in-app

Puoi vedere che ci sono molti modelli diversi per la tua applicazione. Questi modelli sono utili per potenziare il tuo sviluppo; sono pre-costruiti con alcune impostazioni di base del progetto come interfacce dell'interfaccia utente e file di classe.

Qui, useremo la prima opzione, **iOS** .

1. Applicazione principale:

   Questo modello contiene un'interfaccia master e di dettaglio combinata: il master contiene oggetti correlati all'interfaccia di dettaglio. Selezionando gli oggetti nel master cambierà l'interfaccia dei dettagli. È possibile visualizzare l'interfaccia utente di questo tipo nelle applicazioni Impostazioni, Note e Contatti sull'iPad.

2. Applicazione basata su pagina:

   Questo modello viene utilizzato per creare l'applicazione basata su pagina. Le pagine sono viste diverse mantenute da un contenitore.

3. Applicazione vista singola:

   Questo è un normale modello di sviluppo dell'applicazione. Questo è utile per i principianti per imparare il flusso delle applicazioni.

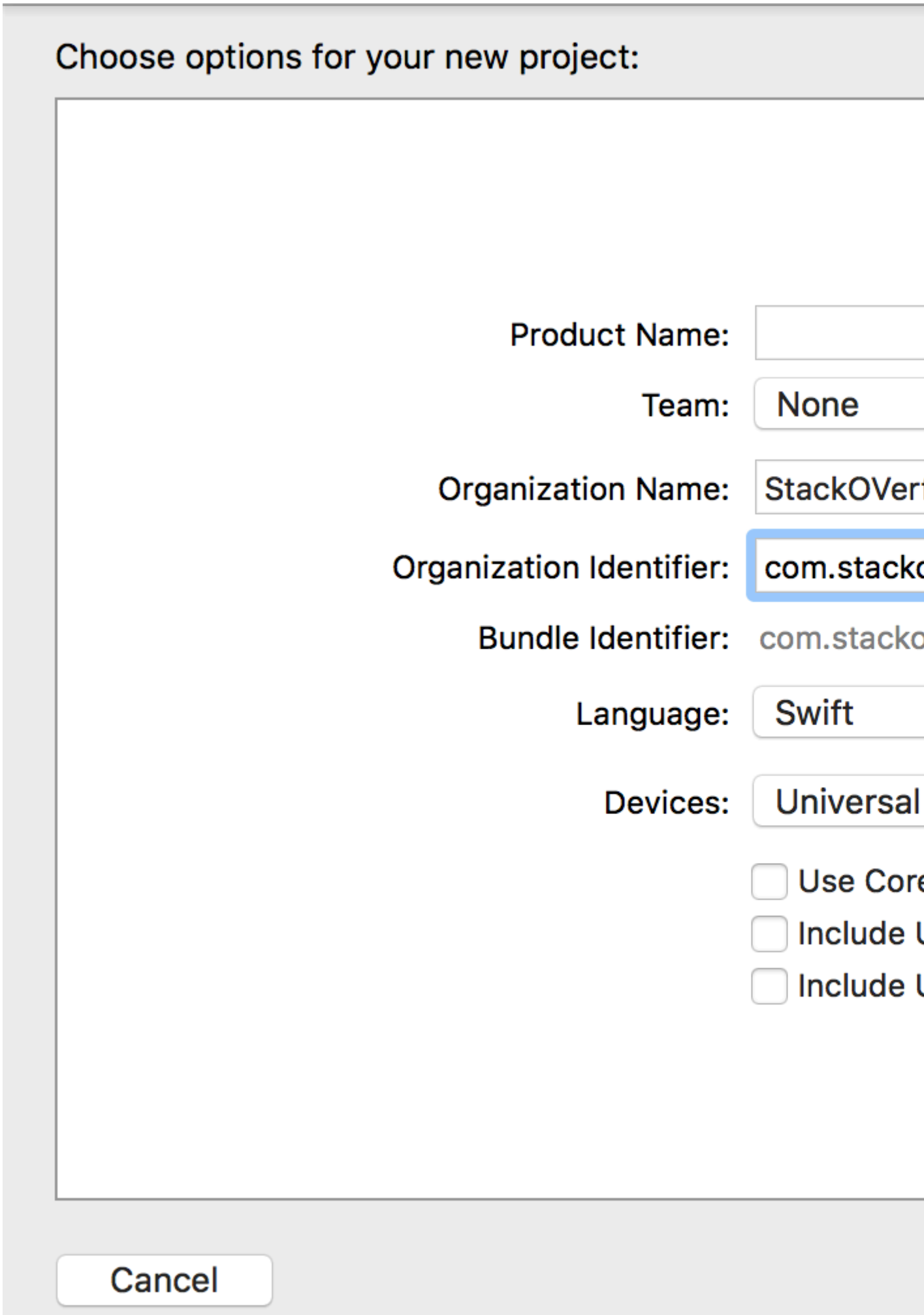
4. Applicazione a schede:

   Questo modello crea schede nella parte inferiore di un'applicazione. Ogni scheda ha un'interfaccia utente diversa e un flusso di navigazione diverso. Puoi vedere questo modello utilizzato in app come Orologio, iTunes Store, iBooks e App Store.

5. Gioco:

   Questo è un punto di partenza per lo sviluppo del gioco. Puoi andare oltre con le tecnologie di gioco come SceneKit, SpriteKit, OpenGL ES e Metal.

4. In questo esempio, inizieremo con l' **applicazione di visualizzazione singola**


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

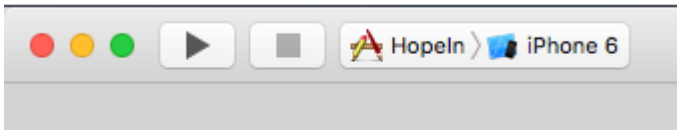
La procedura guidata ti aiuta a definire le proprietà del progetto:

- **Nome prodotto:** il nome del progetto / applicazione
- **Nome organizzazione:** il nome dell'organizzazione in cui sei coinvolto
- **Identificatore di organizzazione:** l'identificatore di organizzazione univoco utilizzato nell'identificatore del pacchetto. Si consiglia di seguire la notazione inversa del nome del dominio.
  l'identificatore di organizzazione univoco utilizzato nell'identificatore del pacchetto. Si consiglia di seguire la notazione inversa del nome del dominio.
- **Identificatore del pacchetto:** *questo campo è molto importante.* Si basa sul nome del progetto e sull'identificatore dell'organizzazione, scegli con saggezza. L'identificatore del bundle verrà utilizzato in futuro per installare l'applicazione su un dispositivo e caricare l'app su iTunes Connect (che è il luogo in cui carichiamo le app da pubblicare su App Store). È una chiave unica per identificare la tua applicazione.
- **Lingua:** il linguaggio di programmazione che si desidera utilizzare. Qui puoi cambiare da Objective-C a Swift se non è selezionato.
- **Dispositivi:** dispositivi supportati per la tua applicazione che possono essere modificati in seguito. Mostra iPhone, iPad e Universal. Le applicazioni universali supportano i dispositivi iPhone e iPad e si consiglia di selezionare questa opzione quando non è necessario eseguire l'app su un solo tipo di dispositivo.
- **Usa dati di base:** se desideri utilizzare il modello di dati di base nel progetto, contrassegnalo come selezionato e creerà un file per `.xcdatamodel` . Puoi anche aggiungere questo file in seguito se non lo sai in anticipo.
- **Includi test unitari:** configura l'obiettivo del test unitario e crea classi per il test dell'unità
- **Includi test dell'interfaccia utente:** configura il target del test dell'interfaccia utente e crea le classi per il test dell'interfaccia utente

Fare clic su **Avanti** e ti chiederà una posizione in cui si desidera creare la directory del progetto.
Fai clic su **Crea** e vedrai l'interfaccia utente Xcode con una configurazione di progetto già definita. Puoi vedere alcune classi e i file Storyboard.

Questo è un modello base per un'applicazione vista singola.

Nella parte in alto a sinistra della finestra, controlla che sia selezionato un simulatore (ad es. "IPhone 6" come mostrato qui) e quindi premi il pulsante RUN triangolare.



5. Una nuova applicazione si aprirà-Simulatore (questo potrebbe richiedere del tempo alla prima esecuzione e potrebbe essere necessario provare due volte se si vede un errore la prima volta). Questa applicazione ci fornisce la simulazione del dispositivo per le applicazioni create. Sembra quasi un vero dispositivo! Contiene alcune applicazioni come un vero dispositivo. Puoi simulare gli orientamenti, la posizione, i gesti di scuotimento, gli avvertimenti sulla memoria, la barra di stato della chiamata, il tocco con le dita, il blocco, il

riavvio, la casa, ecc.

Vedrai una semplice applicazione bianca perché non abbiamo ancora apportato alcuna modifica al modello.

Quindi inizia il tuo. è una corsa lunga e ci sono molte nuove opportunità che ti aspettano!

Se non sei sicuro di dove andare dopo, prova il tutorial " Jump Right In " di Apple. Hai già eseguito i primi passaggi, quindi sei pronto per iniziare.

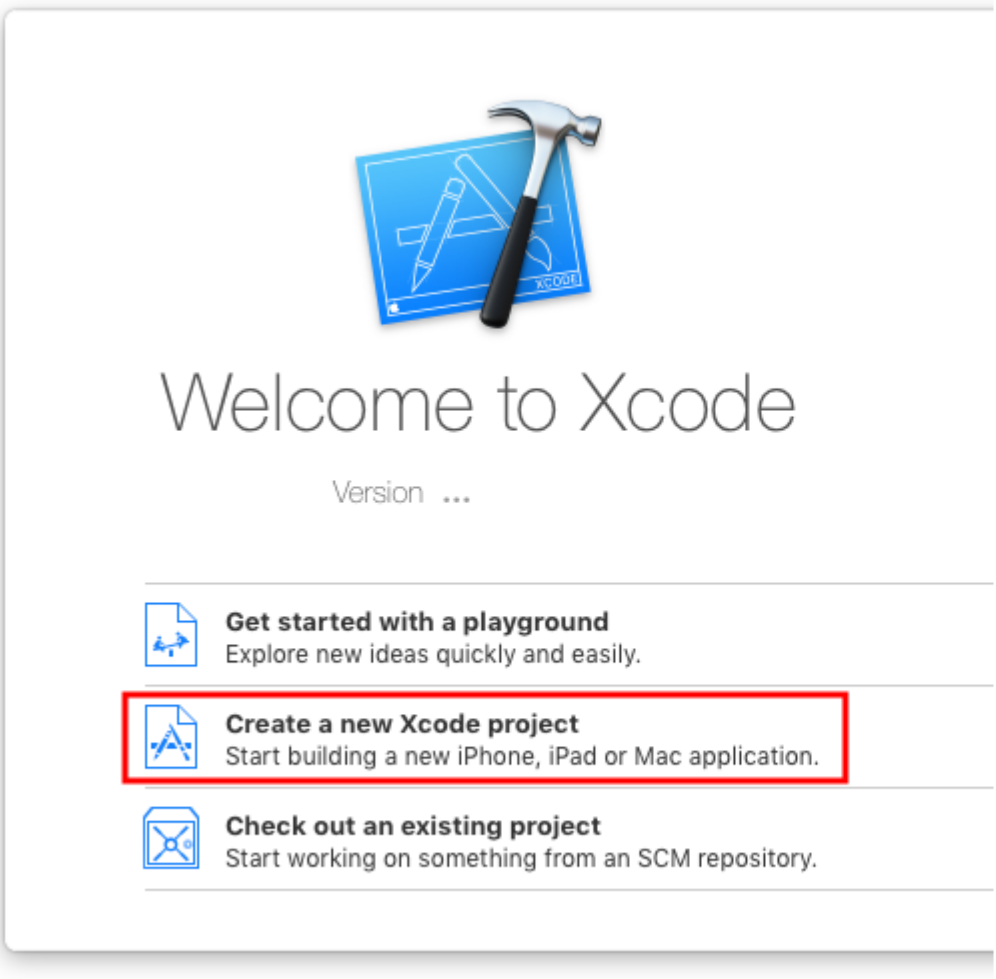
Ciao mondo

Dopo aver configurato Xcode, non è difficile ottenere il tuo primo iOS attivo e funzionante.
Nel seguente esempio:

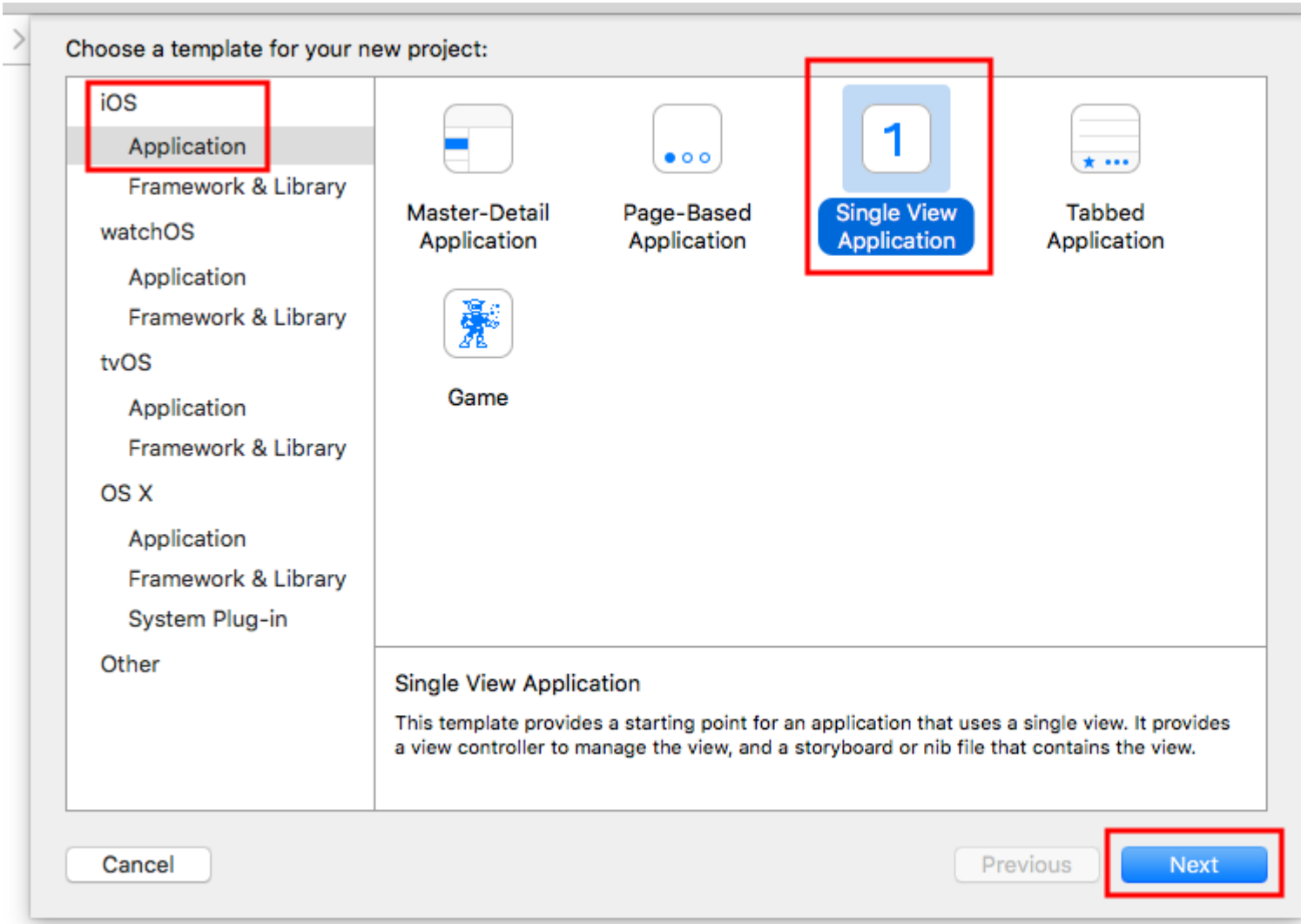
- Inizia un nuovo progetto
- Aggiungi un'etichetta
- Stampa del messaggio sulla console.
- Esegui nel simulatore

Iniziare un nuovo progetto

Quando viene visualizzata la schermata di benvenuto di Xcode, scegli **Crea un nuovo progetto Xcode** . In alternativa, puoi fare **File> Nuovo> Progetto ...** dal menu Xcode se già lo hai aperto.

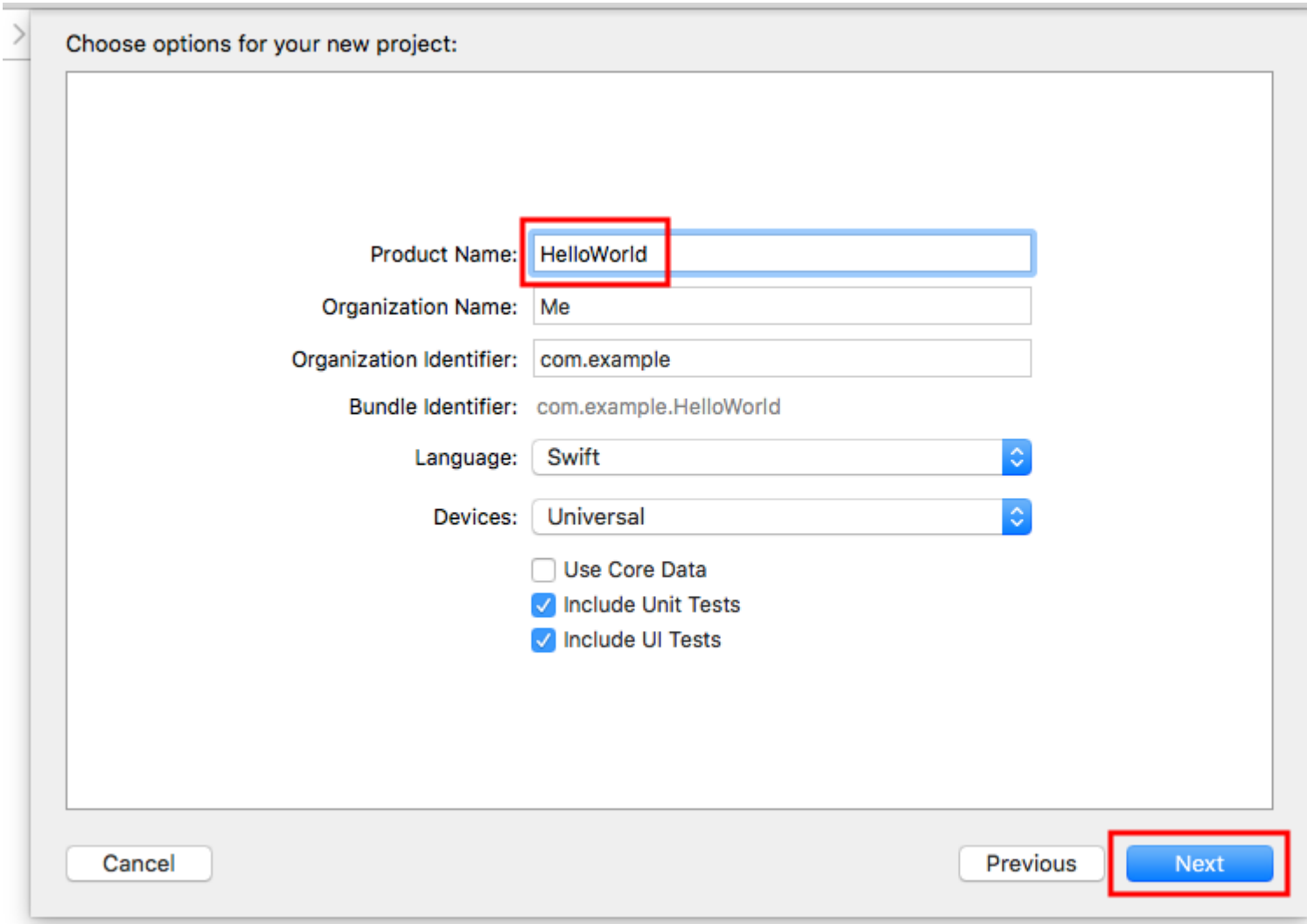


Scegli un'applicazione **Vista singola** e fai clic su **Avanti** .

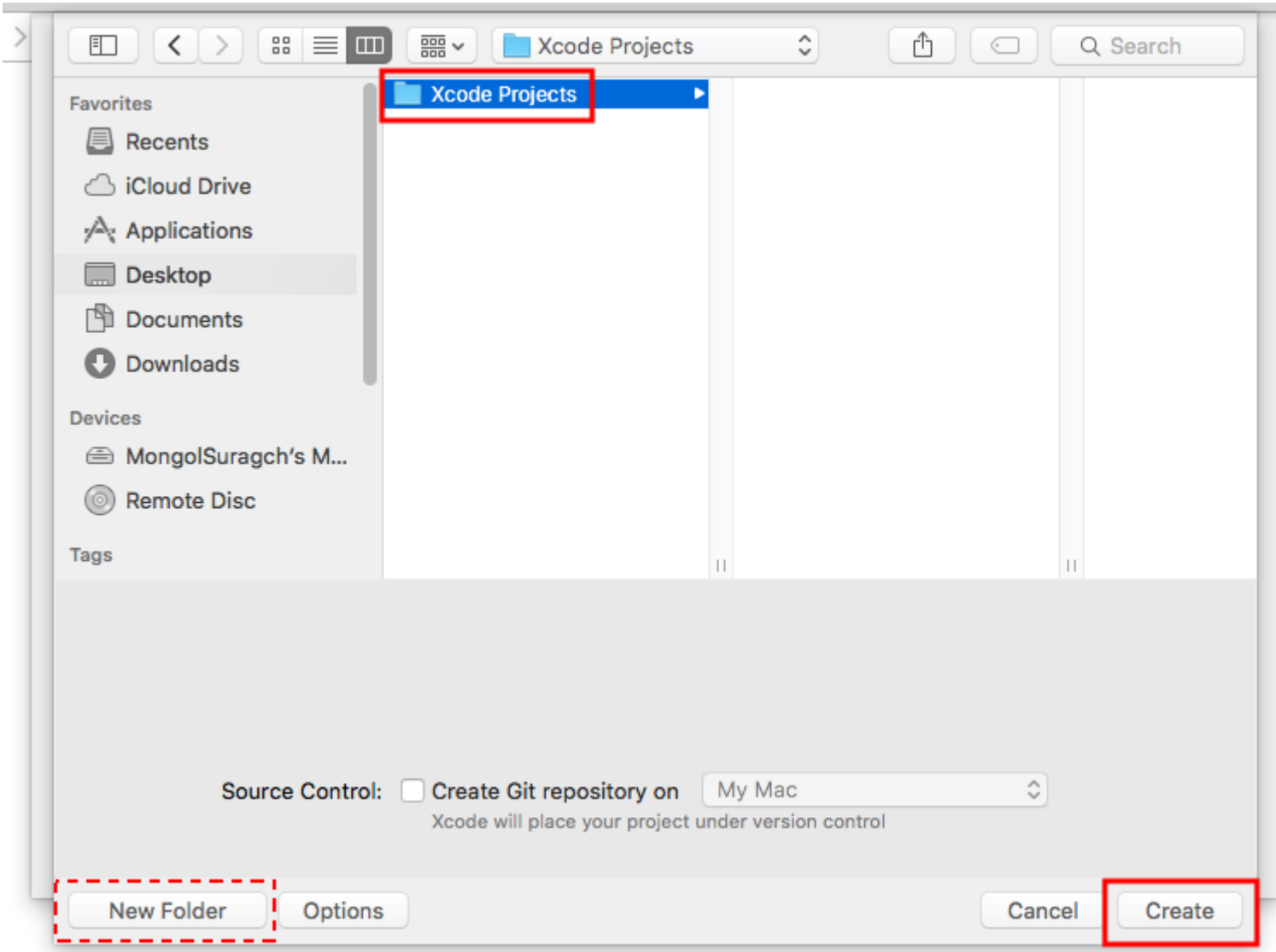


Scrivi "HelloWorld" per il **nome** del **prodotto** (o quello che vuoi veramente) e in **Lingua** , assicurati che sia selezionato **Swift** .

- **Universale** significa che la tua app funzionerà su iPhone e iPad.
- **Usa dati di base si** riferisce alla memorizzazione persistente dei dati, che non è necessaria nella nostra app Hello World.
- In questo esempio non eseguiremo **test di unità** o **test dell'interfaccia utente** , ma non ci farà male prendere l'abitudine di aggiungerli.



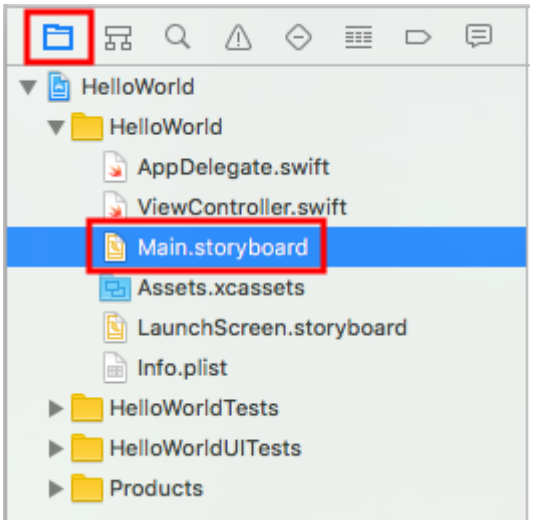
Scegli una cartella esistente o creane una nuova in cui salvare i tuoi progetti Xcode. Questo sarà il default in futuro. Ne abbiamo creato uno chiamato "Progetti Xcode". Quindi fare clic su **Crea** . È possibile selezionare il controllo del codice sorgente se lo si desidera (utilizzato durante la sincronizzazione con siti come GitHub), ma in questo esempio non ne avremo bisogno.



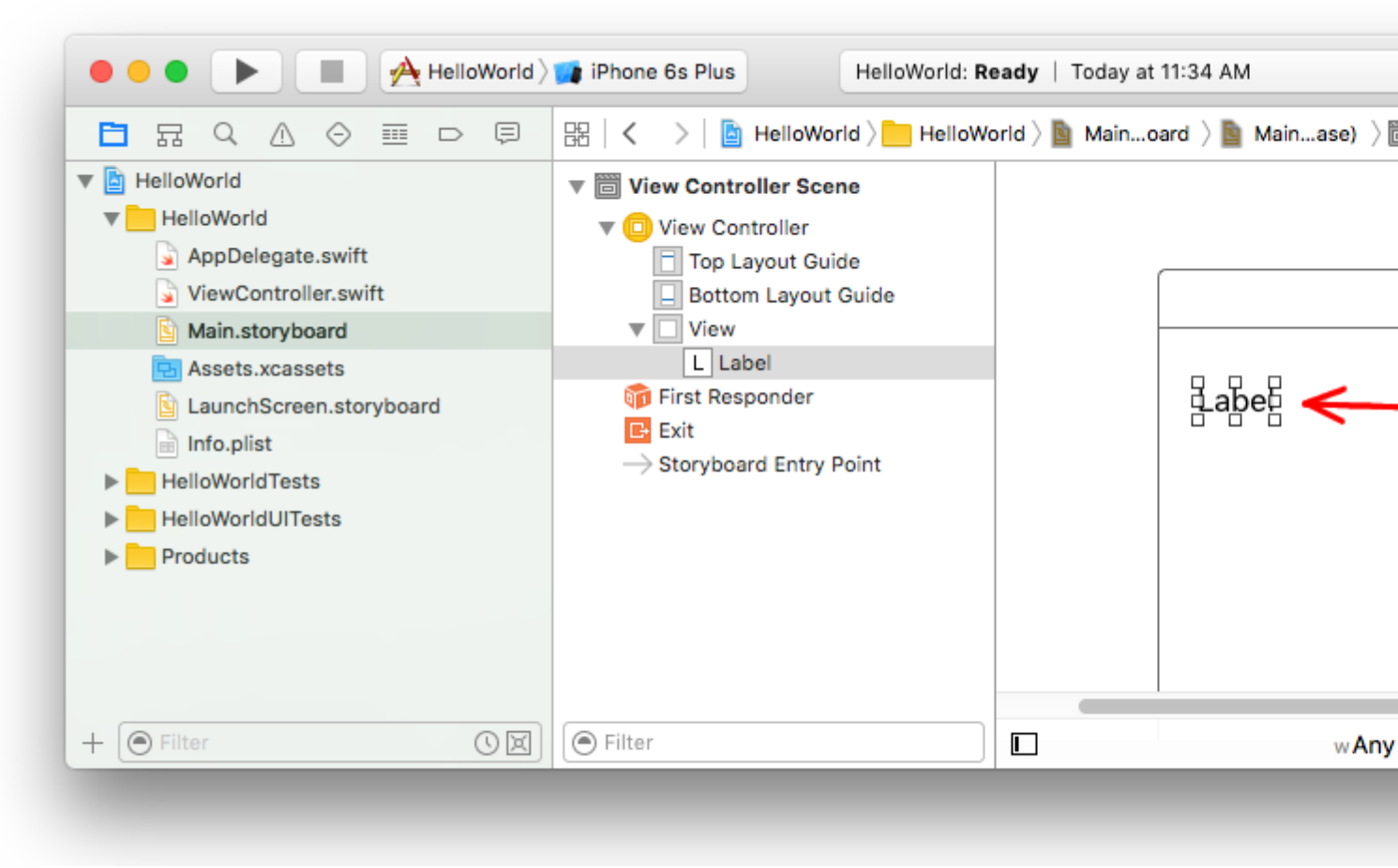
Aggiungere un'etichetta

Questa è la struttura del file di un progetto Xcode.

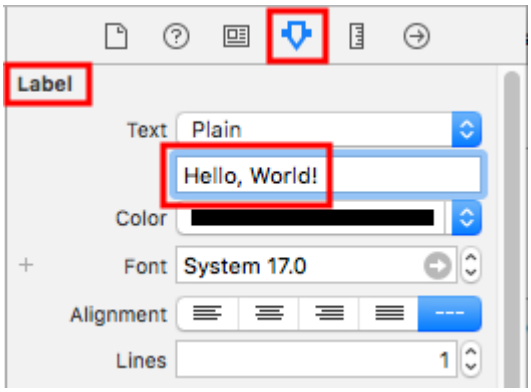
Seleziona *Main.storyboard* nel Project Navigator.



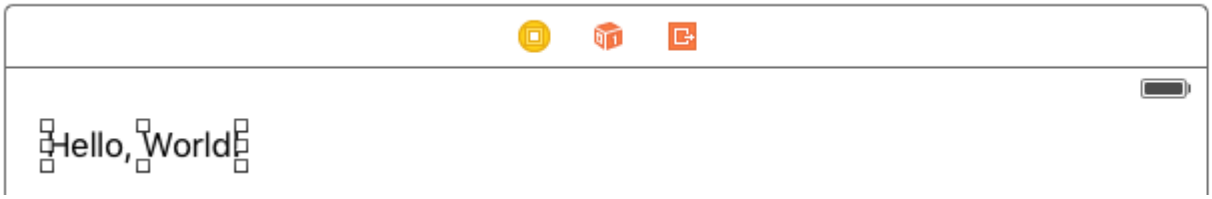
Digitare "etichetta" nel campo di ricerca della libreria di oggetti in basso a destra di Xcode. Quindi trascina la `UILabel` sullo storyboard Visualizza controller. Posizionalo generalmente nella zona dell'angolo in alto a sinistra.



Assicurati che l'etichetta sia selezionata nello storyboard e poi in **Impostazioni Attributi** , cambia il testo in "Ciao, Mondo!" Dovrai quindi ridimensionare e riposizionare l'etichetta sullo storyboard poiché la lunghezza del testo è più lunga.

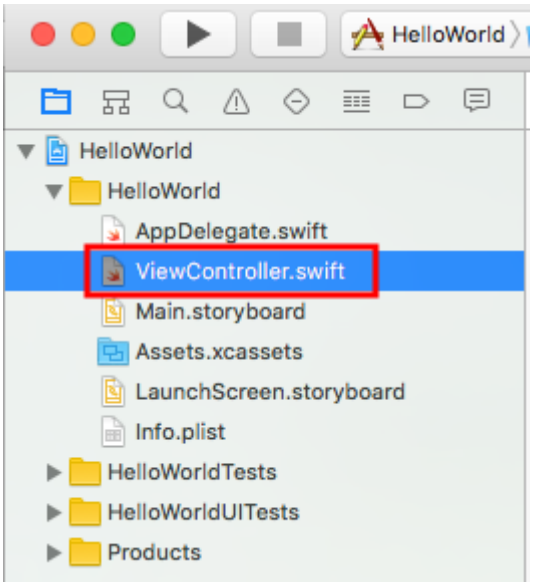


In alternativa, fai doppio clic sull'etichetta sullo storyboard per modificarlo come "Hello, World!". In ogni caso, lo storyboard dovrebbe assomigliare a questo:



Aggiungere codice

Seleziona *ViewController.swift* in Project Navigator.



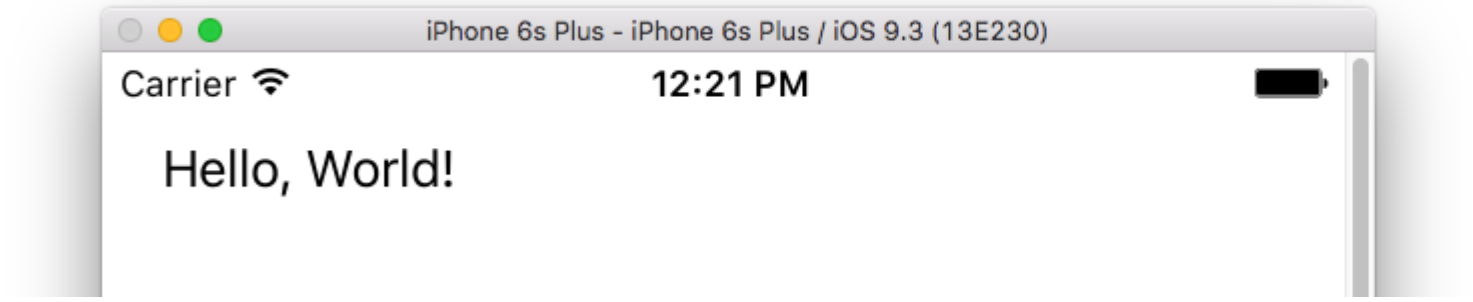
Aggiungi `print("Successfully created my first iOS application.")` `viewDidLoad()` metodo `viewDidLoad()` . Dovrebbe assomigliare a qualcosa di simile a questo.

```
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.
    }
}
```

Esecuzione dell'app nel simulatore

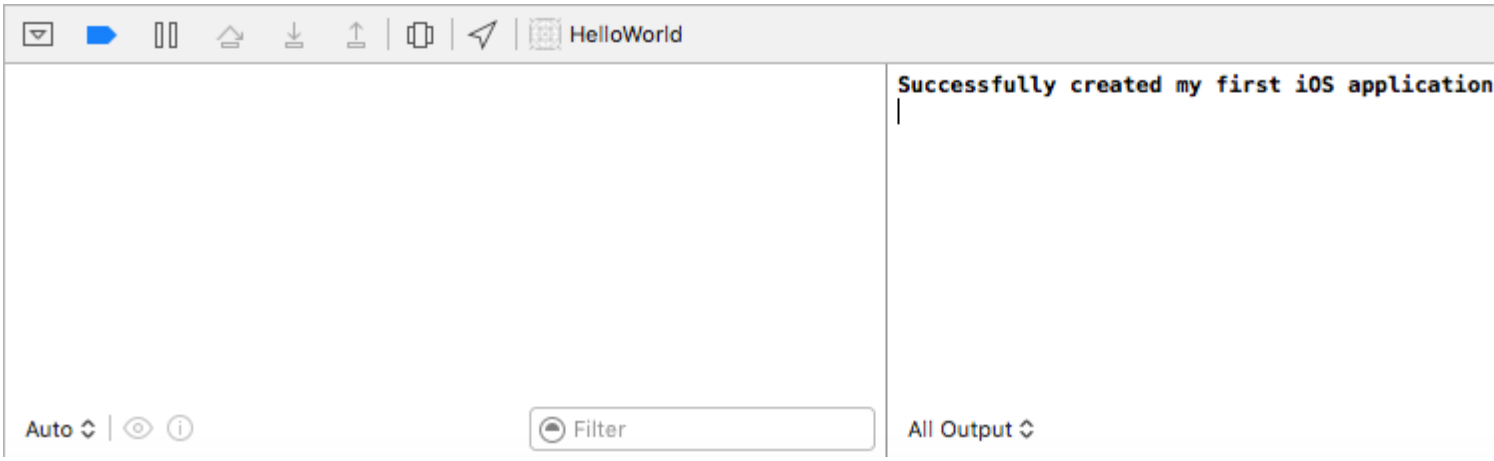
Premi il pulsante Esegui per creare ed eseguire l'app. In questo esempio, il dispositivo simulatore corrente (denominato "schema") è impostato su iPhone 6s Plus. Le versioni più recenti di Xcode imposteranno automaticamente i nuovi schemi. Puoi anche scegliere altri schemi facendo clic sul nome. Ci limiteremo a rispettare l'impostazione predefinita.

Il simulatore impiegherà del tempo per iniziare alla prima esecuzione. Una volta eseguito, dovrebbe apparire come questo:



Nel menu del simulatore, puoi scegliere **Finestra> Scala** per ridurla, oppure premere cmd + 1/2/3/4/5 per la scala 100% / 75% / 50% / 33% / 25% rispettivamente.

L'area di debug di Xcode (in basso) dovrebbe avere anche stampato "Creata con successo la mia prima applicazione iOS". alla console. "Ho creato con successo la mia prima applicazione iOS." messaggio è la stringa stampata a livello di **codice nella** parte **Aggiungi codice** .

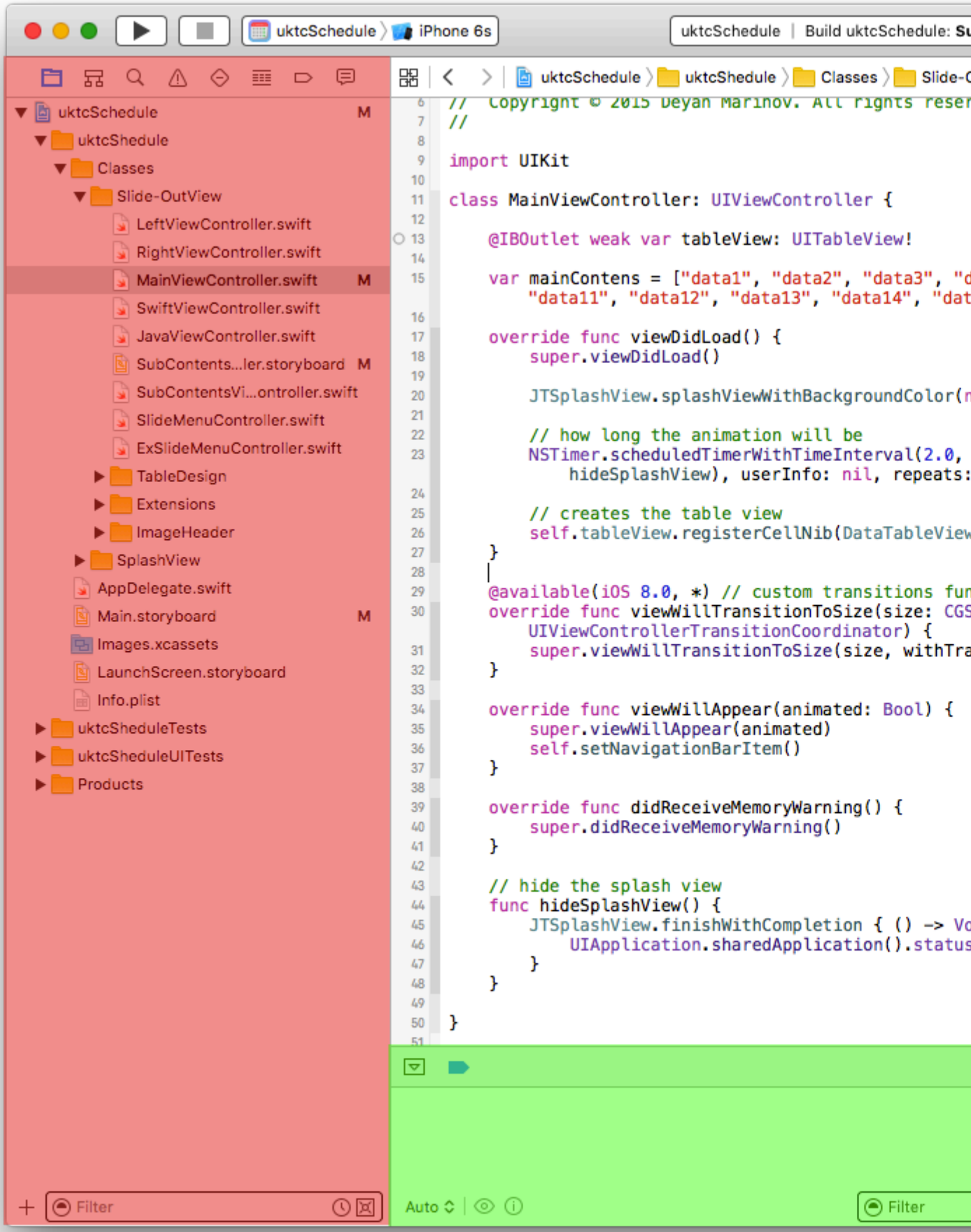


Andando avanti

Dovresti conoscere i vincoli del layout automatico dopo. Questi ti aiutano a posizionare i controlli sullo storyboard in modo che risultino belli su qualsiasi dimensione e orientamento del dispositivo.

Xcode Interface

In Xcode, hai tre aree separate di lavoro: i navigatori (in rosso), l'area di debug (in verde) e le utilità (in blu).


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

La finestra dell'area di lavoro include sempre l'area dell'editor. Quando selezioni un file nel tuo progetto, il suo contenuto appare nell'area dell'editor, dove Xcode apre il file in un editor appropriato. Ad esempio, nell'immagine sopra, l'area dell'editor MainViewController.swift, un file di codice rapido che viene selezionato nell'area Navigatore sulla sinistra della finestra dell'area di lavoro.

Area di navigazione

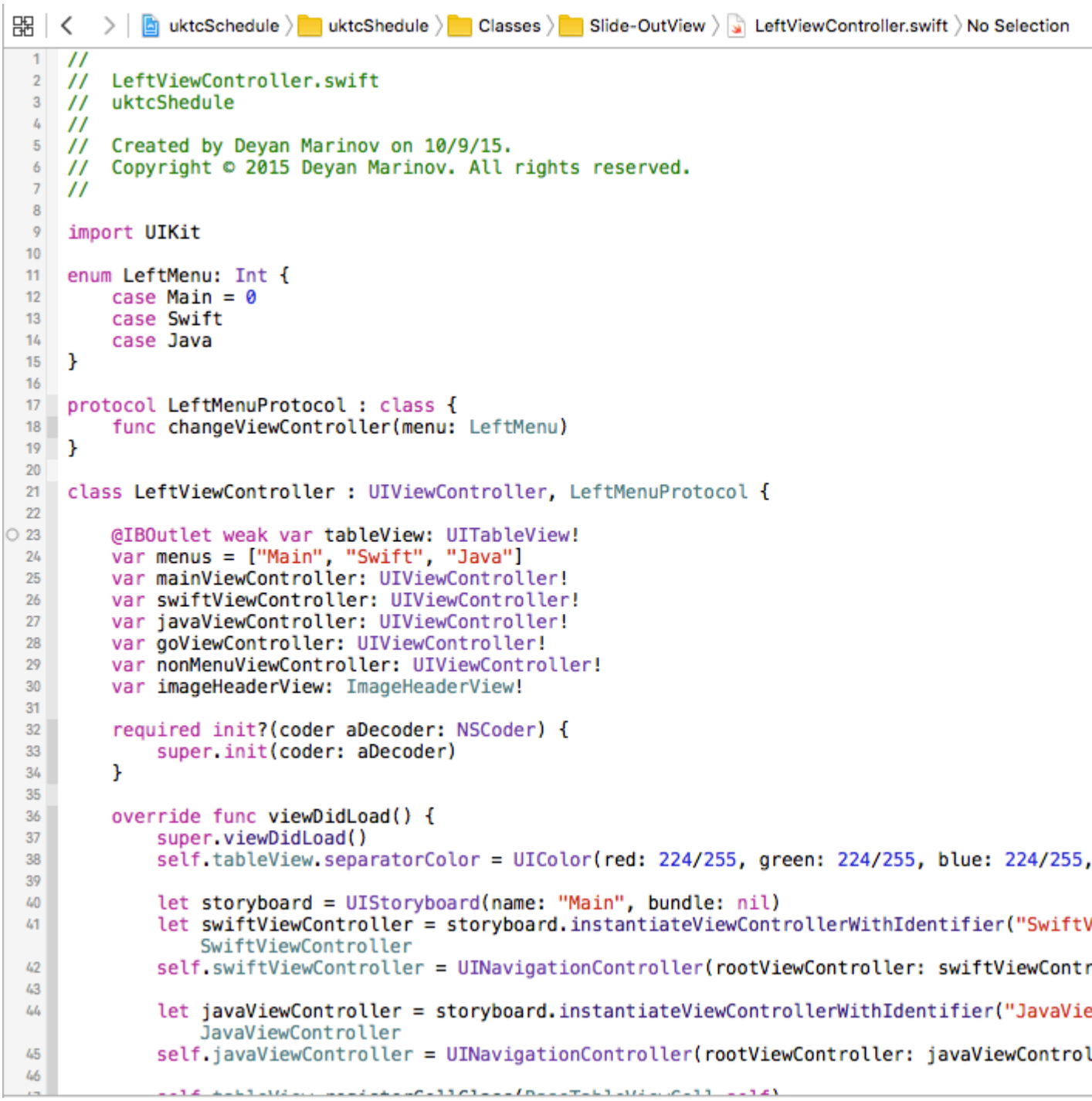
La finestra del navigatore contiene le seguenti otto opzioni:

- **Navigatore di progetto.** Aggiungi, elimina, raggruppa e gestisci in altro modo i file nel tuo progetto, oppure scegli un file per visualizzarne o modificarne i contenuti nell'area dell'editor.
- **Navigatore di simboli.** Sfoglia i simboli nel tuo progetto come un elenco o una gerarchia. I pulsanti a sinistra della barra dei filtri consentono di limitare i simboli visualizzati a una combinazione di sole classi e protocolli, solo simboli nel progetto o solo contenitori.
- **Trova navigatore** Usa opzioni di ricerca e filtri per trovare rapidamente qualsiasi stringa all'interno del tuo progetto.
- **Issue navigator.** Visualizza problemi come diagnostica, avvisi e errori rilevati durante l'apertura, l'analisi e la creazione del progetto.
- **Prova il navigatore.** Creare, gestire, eseguire e rivedere i test unitari.
- **Debug navigator.** Esaminare i thread in esecuzione e le informazioni sullo stack associate in un momento o momento specificato durante l'esecuzione del programma.
- **Navigatore Breakpoint.** Ottimizzare i breakpoint specificando caratteristiche come le condizioni di attivazione.
- **Segnala navigatore.** Visualizza la cronologia delle attività di compilazione, esecuzione, debug, integrazione continua e controllo del codice sorgente.

Gli editori

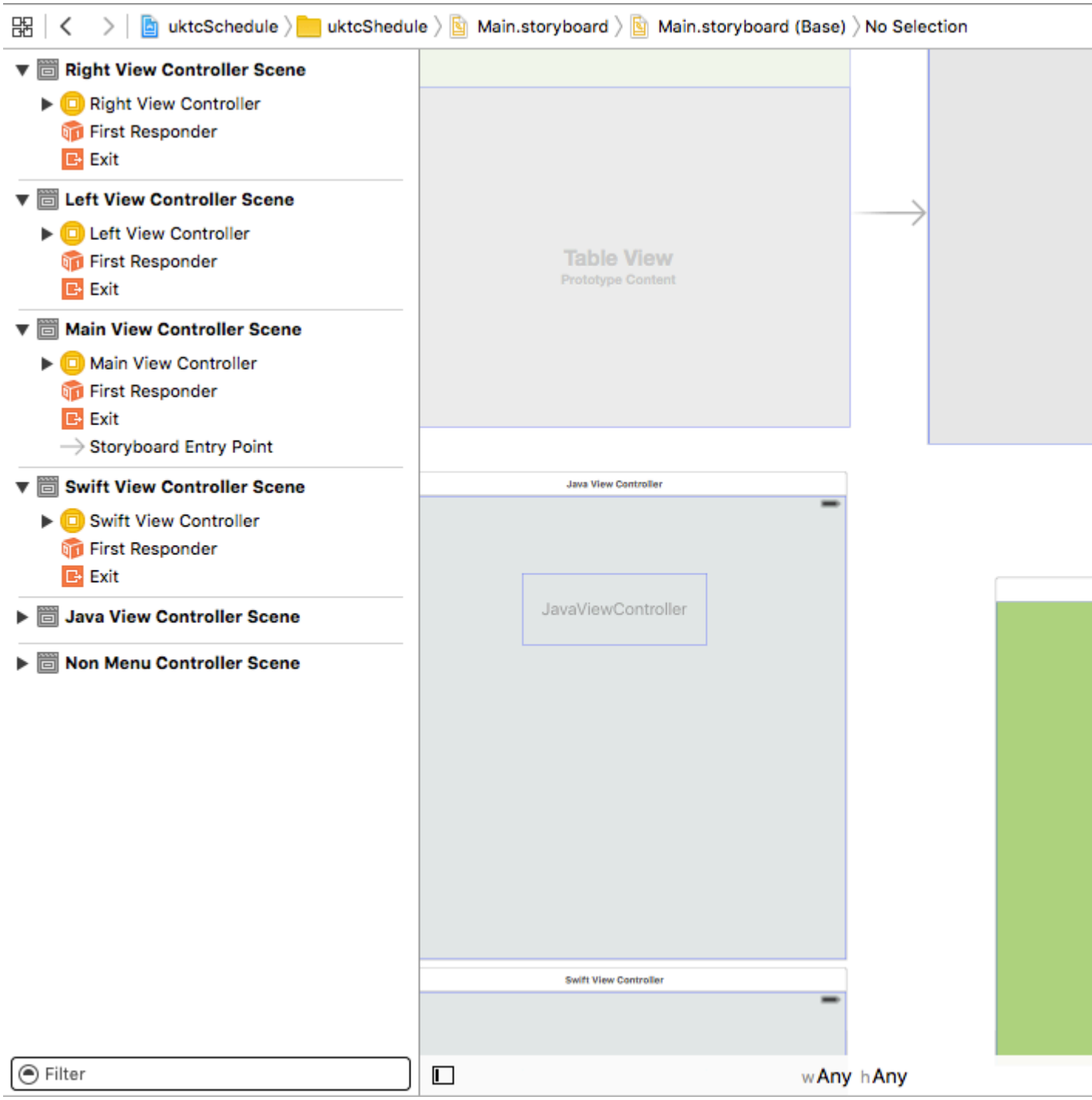
La maggior parte del lavoro di sviluppo in Xcode si verifica nell'area dell'editor, l'area principale che è sempre visibile all'interno della finestra dell'area di lavoro. Gli editor che usi più spesso sono:

- **Editor di sorgenti** Scrivi e modifica il codice sorgente.

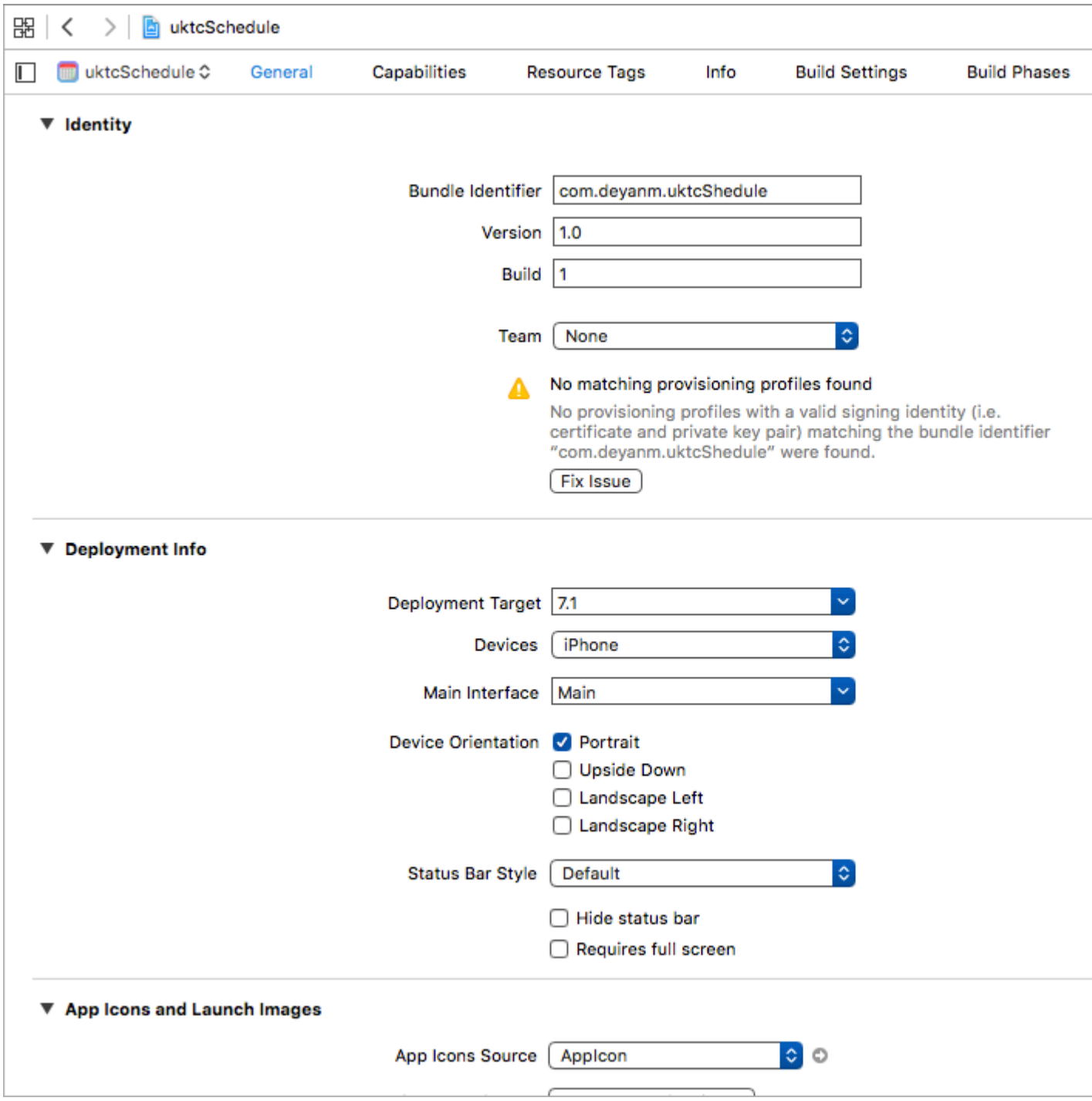



```swift
//
//  LeftViewController.swift
//  uktcShedule
//
//  Created by Deyan Marinov on 10/9/15.
//  Copyright © 2015 Deyan Marinov. All rights reserved.
//

import UIKit

enum LeftMenu: Int {
    case Main = 0
    case Swift
    case Java
}

protocol LeftMenuProtocol : class {
    func changeViewController(menu: LeftMenu)
}

class LeftViewController : UIViewController, LeftMenuProtocol {

    @IBOutlet weak var tableView: UITableView!
    var menus = ["Main", "Swift", "Java"]
    var mainViewController: UIViewController!
    var swiftViewController: UIViewController!
    var javaViewController: UIViewController!
    var goViewController: UIViewController!
    var nonMenuViewController: UIViewController!
    var imageHeaderView: ImageHeaderView!

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

    override func viewDidLoad() {
        super.viewDidLoad()
        self.tableView.separatorColor = UIColor(red: 224/255, green: 224/255, blue: 224/255,

        let storyboard = UIStoryboard(name: "Main", bundle: nil)
        let swiftViewController = storyboard.instantiateViewControllerWithIdentifier("SwiftV
            SwiftViewController
        self.swiftViewController = UINavigationController(rootViewController: swiftViewContr

        let javaViewController = storyboard.instantiateViewControllerWithIdentifier("JavaVie
            JavaViewController
        self.javaViewController = UINavigationController(rootViewController: javaViewControl
```

- **Interface Builder.** Crea e modifica graficamente i file dell'interfaccia utente.



- **Editor di progetti.** Visualizza e modifica la modalità di creazione delle tue app, ad esempio specificando le opzioni di compilazione, le architetture di destinazione e le autorizzazioni delle app.



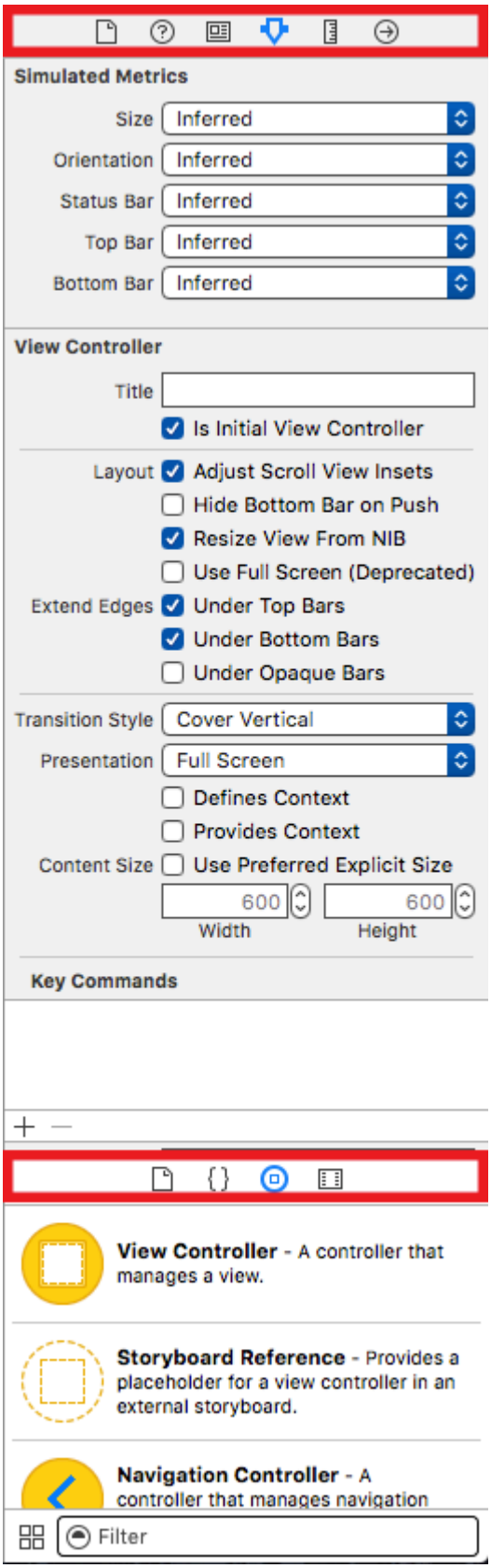
Configura l'area dell'editor per una determinata attività con i pulsanti di configurazione dell'editor sul lato destro della barra degli strumenti: 

- **Editor standard.** Riempie l'area dell'editor con i contenuti del file selezionato.
- **Assistente editore.** Presenta un riquadro di editor separato con contenuto logicamente correlato al contenuto nel riquadro dell'editor standard. Puoi anche modificare il contenuto.
- **Editor di versione.** Mostra le differenze tra il file selezionato in un riquadro e un'altra versione dello stesso file in un secondo riquadro. Questo editor funziona solo quando il tuo progetto è sotto il controllo del codice sorgente.

Risorse ed elementi nell'area delle utilità

L'area delle utilità all'estrema destra della finestra dell'area di lavoro consente di accedere rapidamente a queste risorse: Ispettori, per visualizzare e modificare le caratteristiche del file aperto in un editor Librerie di risorse pronte per l'uso nel progetto

Il pannello superiore dell'area delle utilità visualizza gli ispettori. Il riquadro in basso ti dà accesso alle librerie.



Il primo pannello (evidenziato in rosso) è la **barra dell'Inspector** , usalo per scegliere l'ispettore più adatto al tuo compito corrente. Due ispettori sono sempre visibili nella barra degli ispettori (sono disponibili ulteriori ispettori in alcuni editor):

- **Ispettore file.** Visualizza e gestisci i metadati per il file selezionato. In genere localizzerete gli storyboard e altri file multimediali e modificherete le impostazioni per i file dell'interfaccia

utente.
- **Aiuto rapido.** Visualizza i dettagli su un simbolo, un elemento dell'interfaccia o un'impostazione di generazione nel file. Ad esempio, Guida rapida visualizza una descrizione concisa di un metodo, dove e come viene dichiarato il metodo, il suo ambito, i parametri necessari e la sua disponibilità di piattaforma e architettura.

Utilizza la **barra Libreria** (la seconda evidenziata in rosso) per accedere a librerie di risorse pronte all'uso per il tuo progetto:

- **Modelli di file.** Modelli per tipi comuni di file e costrutti di codice.
- **Frammenti di codice.** Brevi pezzi di codice sorgente da utilizzare nel software, come dichiarazioni di classe, flussi di controllo, dichiarazioni di blocco e modelli per le tecnologie Apple comunemente utilizzate.
- **Oggetti.** Articoli per l'interfaccia utente della tua app.
- **Media.** File contenenti grafica, icone, file audio e simili.

  Per utilizzare una libreria, trascinala direttamente nell'area appropriata. Ad esempio, per utilizzare uno snippet di codice, trascinalo dalla libreria all'editor di origine; per creare un file sorgente da un modello di file, trascinare il relativo modello nel navigatore del progetto.

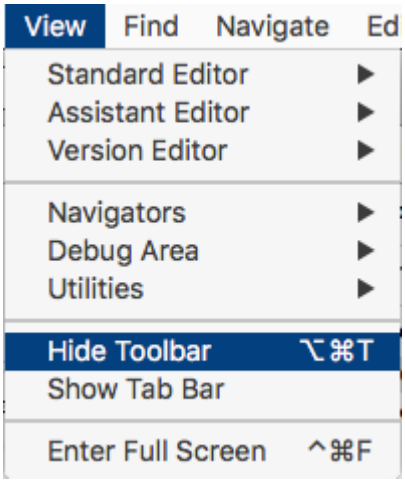
Per limitare gli elementi visualizzati in una libreria selezionata, digitare il testo pertinente nel campo di testo nella **barra Filtro** (il riquadro in basso). Ad esempio, digita "button" nel campo di testo per mostrare tutti i pulsanti nella libreria Objects.

Gestisci le attività con la barra degli strumenti dell'area di lavoro

La barra degli strumenti nella parte superiore della finestra dell'area di lavoro consente di accedere rapidamente ai comandi utilizzati più di frequente. Il **pulsante Esegui** crea e gestisce i tuoi prodotti. Il **pulsante Stop** termina il codice in esecuzione. Il **menu Schema** consente di configurare i prodotti che si desidera creare ed eseguire. Il **visualizzatore di attività** mostra lo stato di avanzamento delle attività attualmente in esecuzione visualizzando i messaggi di stato, lo stato di avanzamento e altre informazioni sul progetto.

I **pulsanti di configurazione dell'editor** (il primo gruppo di tre pulsanti) consentono di configurare l'area dell'editor e i **pulsanti di configurazione** dell'area di **lavoro** (il secondo gruppo di tre pulsanti) nascondono o mostrano le aree opzionali di navigatore, debug e utilità.

Il **menu Visualizza** include i comandi per nascondere o mostrare la barra degli strumenti.



Crea il tuo primo programma in Swift 3

Qui sto presentando come creare il primo programma di base in Swift 3. Per prima cosa è necessario avere una conoscenza di base della lingua di programmazione o non essere pronti per apprenderla dall'inizio.

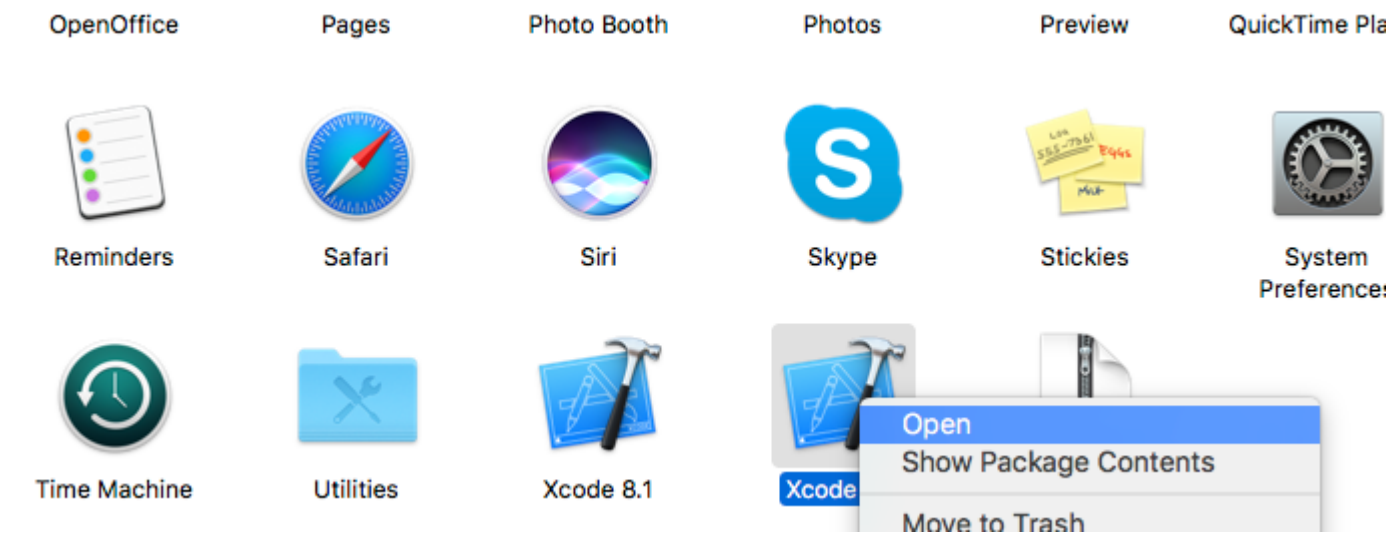
Requisiti per gli sviluppi:

1. MAC OS - Versione 10.11.6 o successive per il nuovo Xcode 8.2
2. Xcode - Versione 8.2 Documento Apple per l'introduzione di Xcode.

Xcode 8.2 ha nuove funzionalità linguistiche Swift 3 con nuovi APi compatibili con iOS 10.

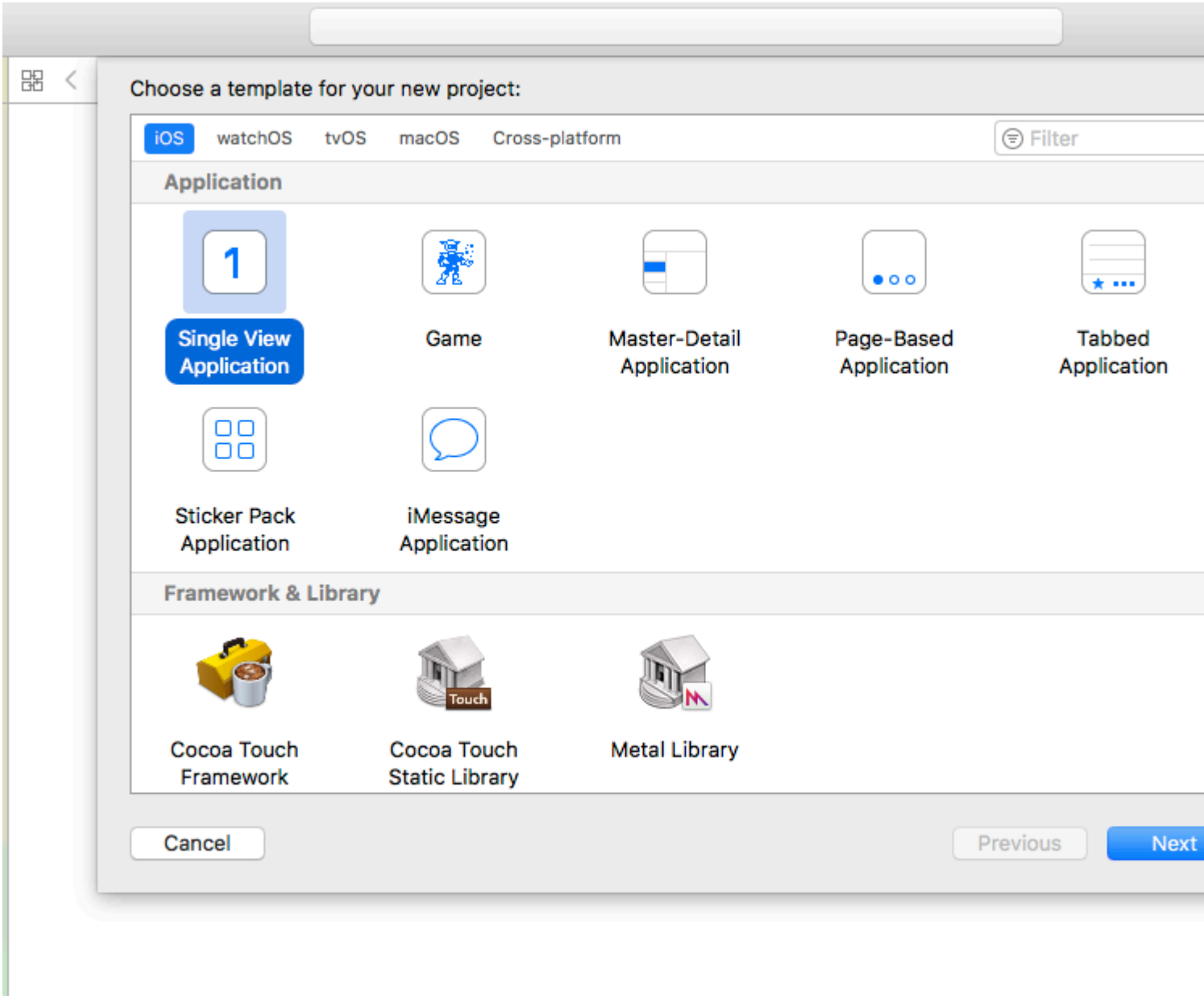
Crea il tuo primo programma

Prima vai su Applicazione e apri il tuo Xcode 8.2.



Dopodiché vedrai lo schermo

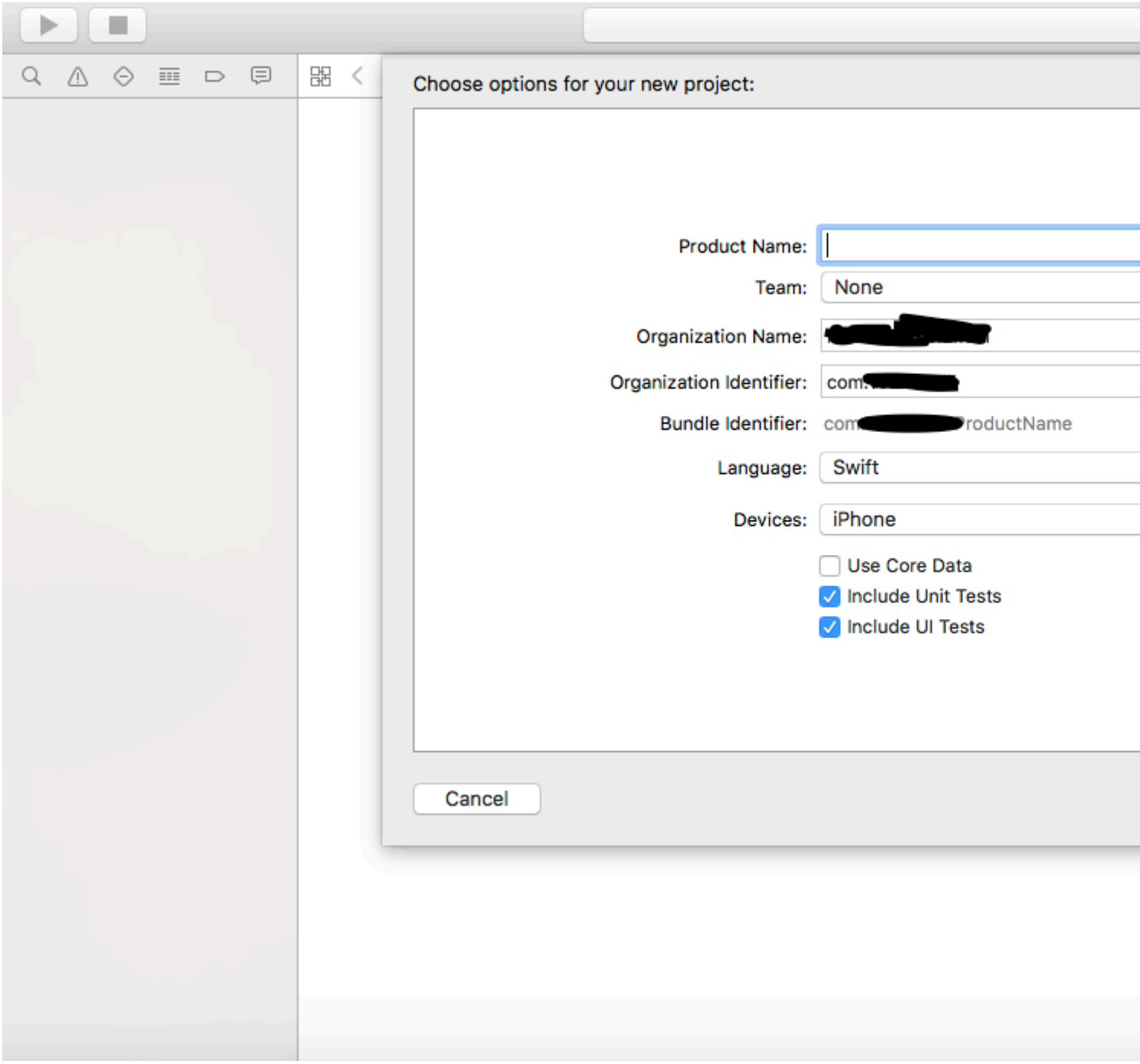
Quindi scegli Crea nuovo progetto e dopo vedrai la schermata successiva



Questa è anche una parte molto importante all'interno di Xcode per la selezione del nostro tipo di progetto. Dobbiamo scegliere il nostro progetto in base al tipo di sistema operativo. Ci sono cinque tipi di opzioni disponibili sul lato superiore:

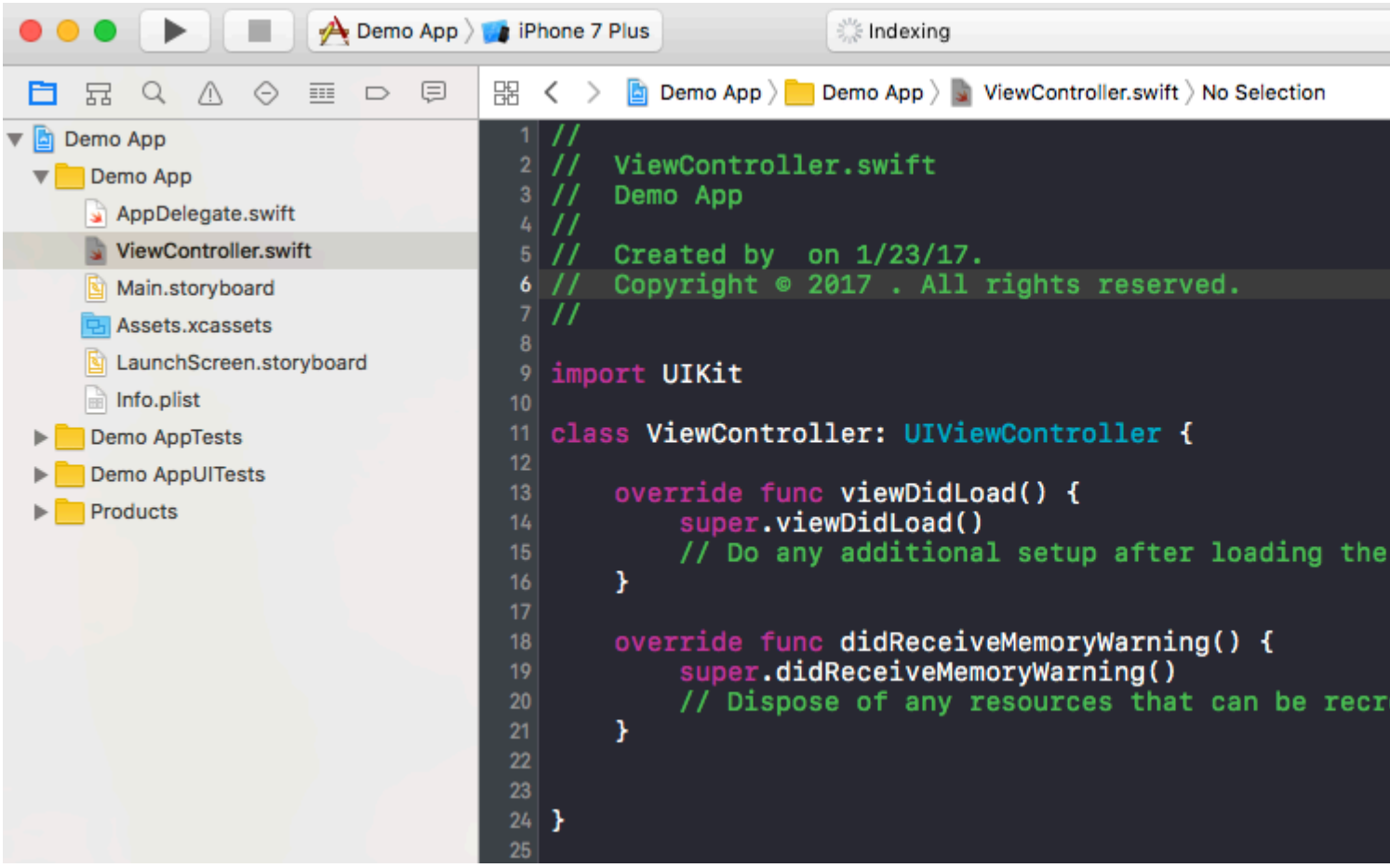
1. iOS
2. watchos
3. Mac OS
4. Cross-platform

Ora stiamo scegliendo la piattaforma iOS per lo sviluppo e la creazione di progetti di base con l'opzione di applicazione vista singola:



Quindi dobbiamo fornire il nome del prodotto, questo rappresenterà il nome del pacchetto e il nome dell'applicazione.

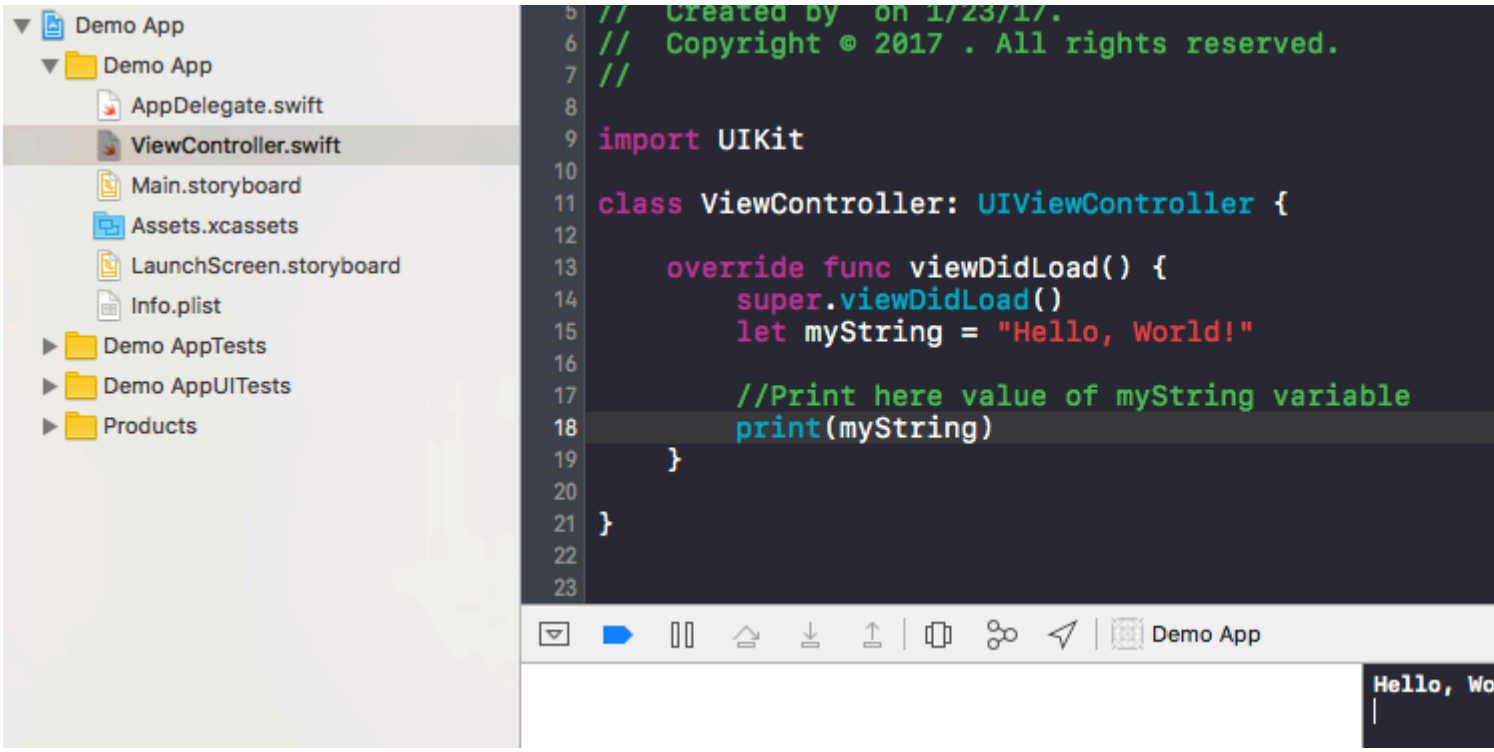
> Nome dell'applicazione che è possibile modificare in seguito secondo le vostre esigenze. Quindi dobbiamo fare clic su "Crea" e dopo che il tuo schermo sarà simile a questo qui sotto:



All'interno di questa classe è possibile vedere il nome del file è ViewController.swift e all'interno della classe il nome è anche ViewController che è ereditarietà della classe super UIViewController e infine stiamo creando la nostra prima variabile il cui nome è **myString** del tipo 'String'. Aggiungi quanto segue sotto 'super.viewDidLoad ()'

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

Stamperemo il contenuto di questa variabile. Per prima cosa, seleziona il tuo tipo di simulatore nella parte in alto a sinistra dello schermo e poi clicca sul pulsante "Esegui".



Successivamente, l'output verrà mostrato sul terminale che si trova in basso a destra.
Congratulazioni, questo è il tuo primo programma Hello World all'interno di Xcode.

Leggi Inizia con iOS online: https://riptutorial.com/it/ios/topic/191/inizia-con-ios

Capitolo 2: Accessibilità

introduzione

L'accessibilità in iOS consente agli utenti con disabilità uditive e disabilità visive di accedere a iOS e alla tua applicazione supportando varie funzionalità come VoiceOver, Controllo vocale, Bianco su nero, Mono Audio, Discorso al testo e così via. Fornire l'accessibilità nell'app per iOS significa rendere l'app utilizzabile per tutti.

Examples

Rendere accessibile una vista

Contrassegna la sottoclasse `UIView` come elemento accessibile in modo che sia visibile a VoiceOver.

```
myView.isAccessibilityElement = YES;
```

Assicurati che la vista contenga un'etichetta, un valore e un suggerimento significativi. Apple fornisce maggiori dettagli su come scegliere una buona descrizione nella Guida alla programmazione dell'accessibilità .

Cornice di accessibilità

Il frame di accessibilità viene utilizzato da VoiceOver per i tocchi di hit, disegnando il cursore VoiceOver e calcolando la posizione dell'elemento focalizzato per simulare un tocco quando l'utente tocca due volte lo schermo. Si noti che la cornice è in coordinate dello schermo!

```
myElement.accessibilityFrame = frameInScreenCoordinates;
```

Se i tuoi elementi o layout dello schermo cambiano spesso, considera l'override - `accessibilityFrame` per fornire sempre un rect aggiornato. Il calcolo della cornice relativa alla retinatura delle viewview della vista di scorrimento può essere soggetto a errori e noioso. iOS 10 introduce una nuova API per semplificare questo aspetto: `accessibilityFrameInContainerSpace` .

Cambia schermo

VoiceOver funziona alla grande la maggior parte del tempo, leggendo scherzosamente gli schermi ad alta voce pieni di contenuti e seguendo intuitivamente l'utente. Ahimè, nessuna soluzione generale è perfetta. A volte solo tu, lo sviluppatore dell'app, sai dove VoiceOver dovrebbe essere focalizzato per un'esperienza utente ottimale. Fortunatamente, VoiceOver ascolta le notifiche di accessibilità del sistema per trovare indizi su dove si trova l'attenzione. Per spostare manualmente il cursore VoiceOver, pubblica una notifica di modifica della schermata di accessibilità modificata:

```
UIAccessibilityPostNotification(UIAccessibilityScreenChangedNotification, firstElement);
```

Quando viene inviata questa notifica, una breve serie di toni notifica agli utenti la modifica. Il secondo parametro può essere l'elemento successivo da mettere a fuoco o una stringa che annuncia la modifica. Pubblica solo una notifica di cambio schermo se l'esperienza di VoiceOver è scarsa senza di essa e non esiste un'altra soluzione alternativa. Spostare il cursore VoiceOver è come colpire lo schermo di un utente avvistato. Può essere fastidioso e disorientante essere condotto in quel modo.

Modifica del layout

In molti casi, i contenuti all'interno di una singola schermata si aggiornano con contenuti nuovi o diversi. Ad esempio, immagina un modulo che rivela opzioni aggiuntive in base alla risposta dell'utente a una domanda precedente. In questo caso, una notifica di "cambio formato" ti consente di annunciare la modifica o concentrarsi su un nuovo elemento. Questa notifica accetta gli stessi parametri della notifica di modifica dello schermo.

```
UIAccessibilityPostNotification(UIAccessibilityLayoutChangedNotification, firstElement);
```

Annuncio

Gli annunci sono utili per avvisare gli utenti di eventi che non richiedono alcuna interazione, come "schermata bloccata" o "caricamento completato". Utilizzare un annuncio più specifico per notificare agli utenti le modifiche alle schermate o altre modifiche di layout minori.

```
UIAccessibilityPostNotification(UIAccessibilityAnnouncementNotification, @"The thing
happened!");
```

Ordinare gli elementi

VoiceOver passa da in alto a sinistra a in basso a destra, indipendentemente dalla gerarchia della vista. Questo di solito è il modo in cui il contenuto è disposto nelle lingue da sinistra a destra poiché gli individui vedenti tendono a scansionare lo schermo in un "modello a forma di F". Gli utenti di VoiceOver si aspettano di navigare allo stesso modo degli utenti tipici. La prevedibilità e la coerenza sono molto importanti per l'accessibilità. Si prega di astenersi dal fare personalizzazioni che "migliorano" il comportamento di default (ad esempio ordinando la barra delle schede prima nell'ordine di scorrimento). Detto questo, se hai ricevuto feedback sul fatto che l'ordine degli elementi nella tua app è sorprendente, ci sono un paio di modi in cui puoi migliorare l'esperienza.

Se VoiceOver deve leggere le sottoview di una vista una dopo l'altra ma non lo è, potrebbe essere necessario suggerire a VoiceOver che gli elementi contenuti in una singola vista sono correlati. Puoi farlo impostando `shouldGroupAccessibiltyChildren` :

```
myView.shouldGroupAccessibilityChildren = YES;
```

Per supportare strutture di navigazione complesse che si estendono su più contenitori o includere interfacce renderizzate senza UIKit, prendere in considerazione l'implementazione del protocollo contenitore nella visualizzazione padre.

Contenitore di accessibilità

VoiceOver può navigare su molte app su iOS perché la maggior `UIKit` classi `UIAccessibilityProtocol` implementa `UIAccessibilityProtocol` . Le funzionalità che non rappresentano elementi sullo schermo utilizzando `UIView` , incluse le app che utilizzano Core Graphics o Metal per eseguire il disegno, devono descrivere questi elementi per l'accessibilità. A partire da iOS 8.0, questo può essere fatto assegnando una proprietà su `UIView` contenente elementi inaccessibili:

```
myInaccessibleContainerView.accessibilityElements = @[elements, that, should, be, accessible];
```

Ogni oggetto nell'array può essere un'istanza di `UIAccessibilityElement` o qualsiasi altra classe che aderisce a `UIAccessibilityProtocol` . Gli elementi figli devono essere restituiti nell'ordine in cui l'utente deve navigarli. Come autore di un'applicazione, puoi utilizzare i contenitori di accessibilità per sostituire l'ordinamento predefinito in alto a sinistra in basso a destra della navigazione a scorrimento di VoiceOver. Dato che `UIView` implementa `UIAccessibilityProtocol` , è possibile combinare le istanze di `UIAccessibilityElement` e `UIView` nella stessa serie di elementi di accessibilità figlio. Tieni presente che se assegni manualmente gli elementi, non è necessario implementare alcun metodo di protocollo di accessibilità dinamica, anche se potrebbe essere necessario inviare una notifica di modifica dello schermo per gli elementi che verranno rilevati da VoiceOver.

Vista modale

Le visualizzazioni modali catturano completamente l'attenzione dell'utente fino al completamento di un'attività. iOS lo chiarisce agli utenti attenuando e disattivando tutti gli altri contenuti quando è visibile una vista modale, ad esempio un avviso o un popover. Un'app che implementa un'interfaccia modale personalizzata deve suggerire a VoiceOver che questa vista merita l'attenzione esclusiva dell'utente impostando `accessibilityViewIsModal` . Si noti che questa proprietà deve essere impostata solo nella vista che contiene il contenuto modale, non gli elementi contenuti in una vista modale.

```
myModalView.accessibilityViewIsModal = YES;
```

Tagging di una vista come modale incoraggia VoiceOver a ignorare le visualizzazioni di pari livello. Se, dopo aver impostato questa proprietà, trovi che VoiceOver continua a scorrere altri elementi nella tua app, prova a nascondere le visualizzazioni dei problemi finché la modale non viene chiusa.

Elementi nascosti

La maggior parte delle classi UIKit, incluso UIView, aderisce a `UIAccessibilityProtocol` e

restituisce valori corretti per impostazione predefinita. È facile dare per scontato che un `UIView` impostato su nascosto sia anche assente dalla gerarchia di accessibilità e non venga navigato da VoiceOver. Mentre questo comportamento predefinito è solitamente sufficiente, ci sono momenti in cui una vista sarà presente nella gerarchia della vista ma non visibile o navigabile. Ad esempio, una collezione di pulsanti può essere sovrapposta a un'altra vista, rendendola invisibile a un utente vedente. VoiceOver, comunque, cercherà comunque di navigare tra loro poiché tecnicamente non sono nascosti da `UIKit` e quindi sono ancora presenti nella gerarchia di accessibilità. In questi casi, devi indicare a VoiceOver che la vista principale non è accessibile. Puoi farlo nascondendo esplicitamente la vista da UIKit impostando nascosto quando la vista va fuori schermo:

```
myViewFullofButtons.hidden = YES;
```

In alternativa, puoi lasciare la vista principale visibile e nascondere semplicemente i suoi figli dalla gerarchia di accessibilità:

```
myViewFullofButtons.accessibilityElementsHidden = YES;
```

Le viste temporanee sono un altro punto in cui si desidera nascondere gli elementi della gerarchia di accessibilità lasciandoli visibili agli utenti. Ad esempio, la vista che si apre quando si preme il pulsante del volume è visibile agli utenti vedenti, ma non richiede attenzione come fa un normale avviso. Non vorresti che VoiceOver interrompesse l'utente e spostasse il cursore da qualsiasi cosa stessero facendo per annunciare il nuovo volume, specialmente dato che il volume di regolazione fornisce già un riscontro uditivo attraverso il clic che fa. In casi come questo, ti consigliamo di nascondere la vista usando `accessibilityElementsHidden` .

Leggi Accessibilità online: https://riptutorial.com/it/ios/topic/773/accessibilita

Capitolo 3: Acquisto in app

Examples

Single IAP in Swift 2

Dopo aver creato un IAP in iTunesConnect:

Nel controller di visualizzazione che desideri acquistare

```
import StoreKit
```

e aggiungere i delegati rilevanti

```
class ViewController: UIViewController, SKProductsRequestDelegate,
SKPaymentTransactionObserver {
```

dichiara una variabile con l'id prodotto da 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...")

   }

}
```

collegare un pulsante a una funzione per acquistare l'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");
    }

}
```

E qui ci sono alcuni metodi di supporto

```
func buyProduct(product: SKProduct){
    println("Sending the Payment Request to Apple");
    let payment = SKPayment(product: product)
    SKPaymentQueue.defaultQueue().addPayment(payment);

 }
```

i metodi delegati che devono essere dichiarati

```
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;
            }
        }
    }

}
```

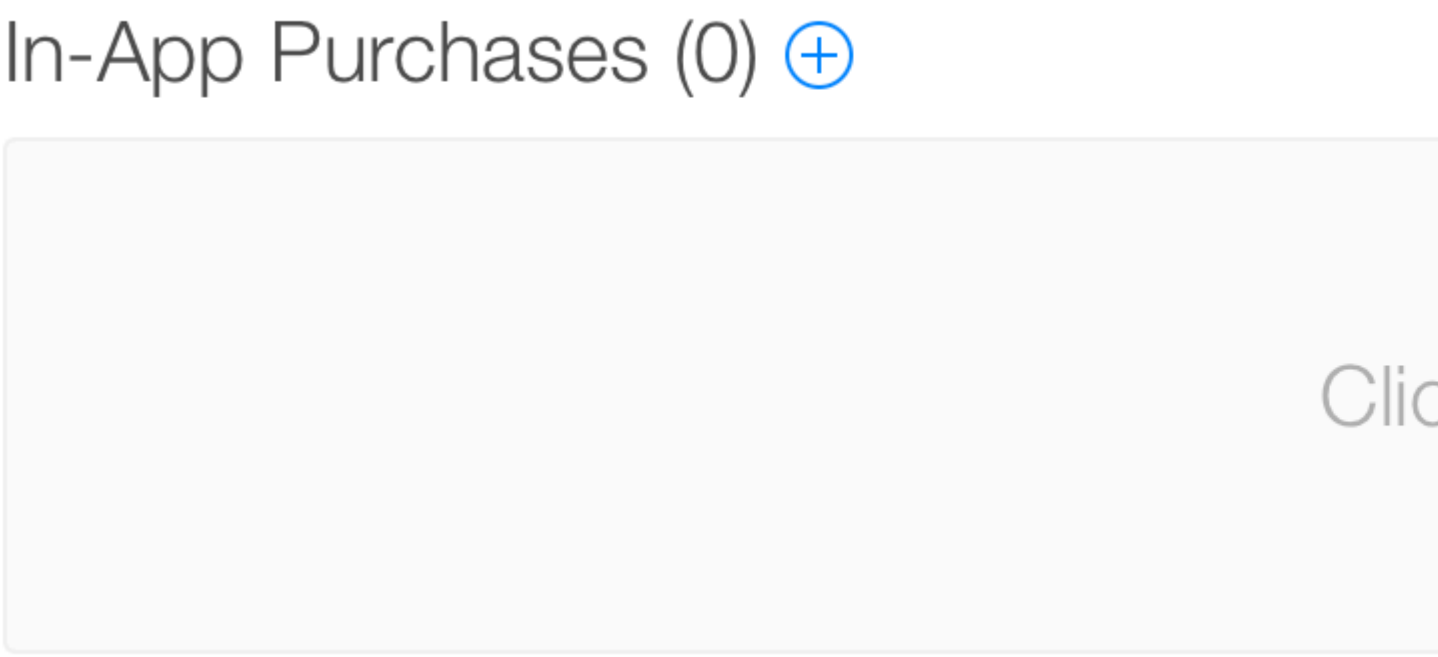
E poi il codice per ripristinare un non consumabile nell'acquisto di app

```
if (SKPaymentQueue.canMakePayments()) {
  SKPaymentQueue.defaultQueue().restoreCompletedTransactions()
}
```

Configura in iTunesConnect

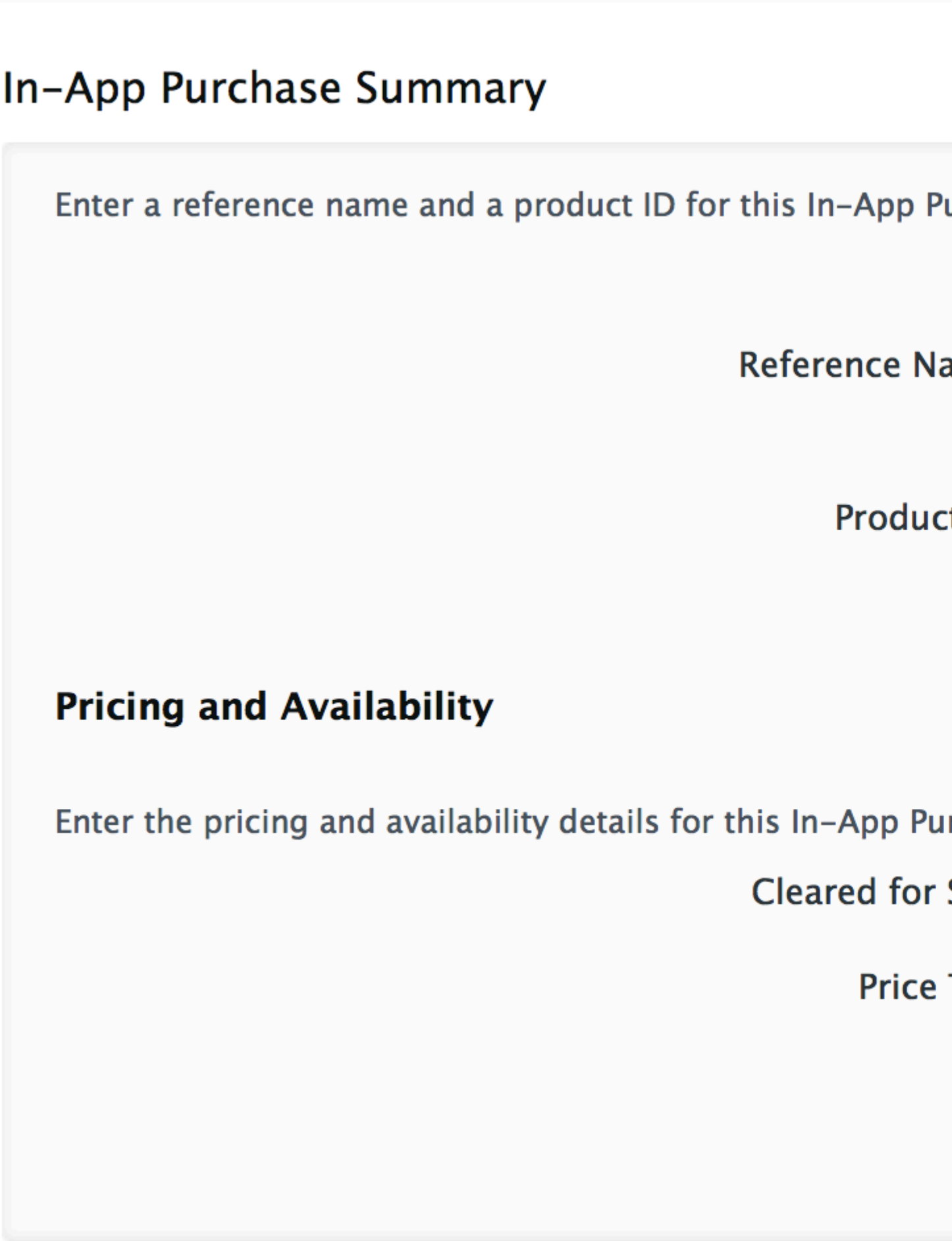
In iTunesConnect , seleziona l'app a cui desideri aggiungere un IAP.

Clicca sulle funzionalità e vedrai questo:



Fai clic sul segno più. Sarà quindi necessario selezionare il tipo di IAP che si desidera effettuare.

Quindi dovrai compilare tutte le informazioni per il tuo IAP.



In caso di problemi è possibile consultare la Guida alla configurazione IAP .

Capitolo 4: AFNetworking

Examples

Completamento del dispatching su un thread personalizzato

Ogni volta che viene utilizzato AFNetworking, la chiamata viene inviata su un thread personalizzato fornito da AFNetworking. Quando la chiamata ritorna al blocco di completamento, viene eseguita sul thread principale.

Questo esempio imposta un thread personalizzato che invia al blocco di completamento:

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];
```

Leggi AFNetworking online: https://riptutorial.com/it/ios/topic/3002/afnetworking

Capitolo 5: Aggiornamento dinamico di UIStackView

Examples

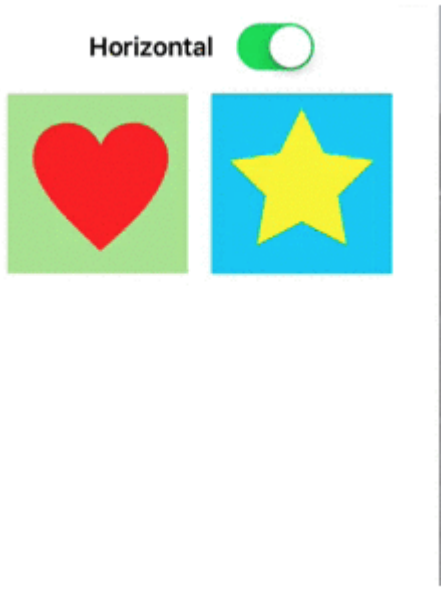
Collega l'UISwitch a un'azione che possiamo animare passando da un layout orizzontale o verticale delle viste dell'immagine

```
@IBAction func axisChange(sender: UISwitch) {
    UIView.animateWithDuration(1.0) {
        self.updateConstraintsForAxis()
    }
}
```

La funzione updateConstraintForAxis imposta solo l'asse della vista dello stack contenente le due viste dell'immagine:

```
private func updateConstraintsForAxis() {
    if (axisSwitch.on) {
        stackView.axis = .Horizontal
    } else {
        stackView.axis = .Vertical
    }
}
```

La gif animata di seguito ti dà un'idea di come appare:



Leggi Aggiornamento dinamico di UIStackView online:

https://riptutorial.com/it/ios/topic/5884/aggiornamento-dinamico-di-uistackview

Capitolo 6: AGGIUNTA DI UN'INTESTINA A BRIDGING SWIFT

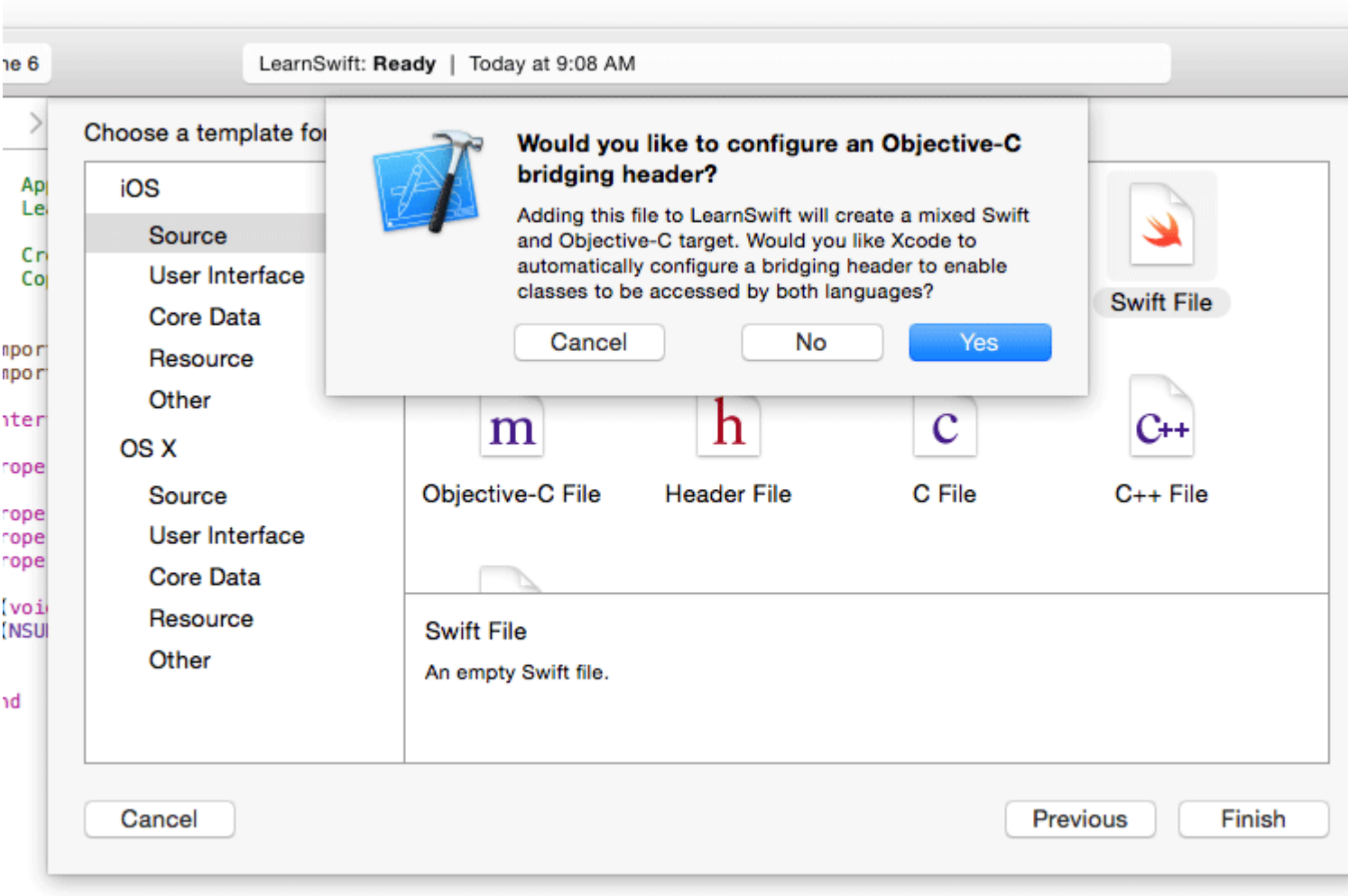
Examples

Come creare manualmente un'intestazione Bridging Bridging

- Aggiungi un nuovo file a Xcode (File> Nuovo> File), quindi seleziona "Sorgente" e fai clic su "File intestazione".
- Assegna un nome al file "YourProjectName-Bridging-Header.h". Esempio: nella mia app Station, il file è denominato "Station-Bridging-Header".
- Crea il file.
- Passare alle impostazioni di creazione del progetto e trovare la sezione "Swift Compiler - Generazione del codice". Potresti trovare più veloce digitare "Swift Compiler" nella casella di ricerca per restringere i risultati. Nota: se non hai una sezione "Swift Compiler - Generazione codice", significa che probabilmente non hai ancora aggiunto classi Swift al tuo progetto. Aggiungi un file Swift, quindi riprova.
- Accanto a "Objective-C Bridging Header" è necessario aggiungere il nome / percorso del file di intestazione. Se il tuo file si trova nella cartella principale del tuo progetto, inserisci semplicemente il nome del file di intestazione. Esempi: "ProjectName / ProjectName-Bridging-Header.h" o semplicemente "ProjectName-Bridging-Header.h".
- Apri la tua intestazione di bridging appena creata e importa le tue classi Objective-C usando le istruzioni #import. Qualsiasi classe elencata in questo file sarà accessibile dalle tue classi veloci.

Xcode crea automaticamente

Aggiungi un nuovo file Swift al tuo progetto Xcode. Chiamalo come ti pare e dovresti ricevere una casella di avviso che ti chiede se desideri creare un'intestazione di bridging. Nota: se non ricevi una richiesta per aggiungere un'intestazione di bridging, probabilmente hai rifiutato questo messaggio una volta prima e dovrai aggiungere l'intestazione manualmente (vedi sotto)



Leggi AGGIUNTA DI UN'INTESTINA A BRIDGING SWIFT online:
https://riptutorial.com/it/ios/topic/10851/aggiunta-di-un-intestina-a-bridging-swift

Capitolo 7: Airdrop

Examples

Airdrop

Objective-C

Airdrop può essere utilizzato da `UIActivityViewController` . La classe `UIActivityViewController` è un controller di visualizzazione standard che offre diversi servizi standard, come la copia degli articoli negli Appunti, la condivisione di contenuti su siti di social media, l'invio di articoli tramite Messaggi, AirDrop e alcune applicazioni di terze parti.

In questo caso, invieremo un'immagine tramite `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];
}
```

veloce

```
if ((newImage) != nil)
        {
            let activityVC = UIActivityViewController(activityItems: [newImage],
applicationActivities: nil)
            activityVC.excludedActivityTypes =[UIActivityTypeAddToReadingList]
            self.presentViewController(activityVC, animated: true, completion: nil)

        }
```

Leggi Airdrop online: https://riptutorial.com/it/ios/topic/7360/airdrop

Capitolo 8: Alamofire

Sintassi

- risposta()
- responseData ()
- responseString (encoding: NSStringEncoding)
- responseJSON (opzioni: NSJSONReadingOptions)
- responsePropertyList (opzioni: NSPropertyListReadOptions)

Parametri

Parametro	Dettagli
Metodo	`.OPTIONS, .GET, .HEAD, .POST, .PUT, .PATCH, .DELETE, .TRACE, .CONNECT`
URLString	`URLStringConvertible`
parametri	`[String: AnyObject]?`
codifica	`ParameterEncoding`
intestazioni	`[String: String]?`

Examples

Fare una richiesta

```
import Alamofire

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

Convalida automatica

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

Gestione delle risposte

```
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)")
         }
     }
```

Convalida manuale

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

Responsabile delle risposte

```
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)
      }
```

Gestori di risposta concatenati

```
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)")
     }
```

Leggi Alamofire online: https://riptutorial.com/it/ios/topic/1823/alamofire

Capitolo 9: API 10 riconoscimento vocale

Examples

Discorso al testo: Riconoscere la voce da un pacchetto conteneva la registrazione audio

```
//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()
}
```

Leggi API 10 riconoscimento vocale online: https://riptutorial.com/it/ios/topic/5986/api-10-riconoscimento-vocale

Capitolo 10: API di Google Places per iOS

Examples

Ottenere luoghi nelle vicinanze dalla posizione corrente

Prerequisiti

1. Installa i pod nel tuo progetto
2. Installa l'SDK di GooglePlaces
3. Abilita i servizi di localizzazione

Per prima cosa dobbiamo ottenere la posizione degli utenti ottenendo la loro longitudine e latitudine correnti.

1. Importa GooglePlaces e GooglePlacePicker

```
import GooglePlaces
import GooglePlacePicker
```

2. Aggiungere il protocollo `CLLOcationManagerDelegate`

```
class ViewController: UIViewController, CLLocationManagerDelegate {

}
```

3. crea il tuo CLLocationManager ()

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

4. Richiedi l'autorizzazione

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

5. Crea un pulsante per chiamare il metodo GooglePlacePicker

@IBAction func placePickerAction (mittente: 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")
            }
        })
    }
}
```

Leggi API di Google Places per iOS online: https://riptutorial.com/it/ios/topic/6908/api-di-google-places-per-ios

Capitolo 11: App Transport Security (ATS)

Parametri

Parametro	Dettagli
`NSAppTransportSecurity`	Configura ATS
`NSAllowsArbitraryLoads`	Impostare su `YES` per disabilitare ATS ovunque. In iOS 10 e versioni successive e macOS 10.12 e versioni successive, il valore di questa chiave viene ignorato se una delle seguenti chiavi è presente nel file Info.plist dell'app: NSAllowsArbitraryLoadsInMedia, NSAllowsArbitraryLoadsInWebContent, NSAllowsLocalNetworking
`NSAllowsArbitraryLoadsInMedia`	Impostare su `YES` per disabilitare ATS per i supporti caricati utilizzando le API dal framework AV Foundation. ***(iOS 10+, macOS 10.12+)***
`NSAllowsArbitraryLoadsInWebContent`	Impostare su `YES` per disabilitare ATS nelle viste Web dell'app (`WKWebView` , `UIWebView` , `WebView`) senza influire sulle connessioni NSURLSession. ***(iOS 10+, macOS 10.12+)***
`NSAllowsLocalNetworking`	Impostare su `YES` per disabilitare le connessioni ai domini non qualificati e ai domini .local. ***(iOS 10+, macOS 10.12+)***
`NSExceptionDomains`	Configura le eccezioni per domini specifici
`NSIncludesSubdomains`	Impostare su `YES` per applicare le eccezioni a tutti i sottodomini del dominio selezionato.
`NSRequiresCertificateTransparency`	Impostare su `YES` per richiedere che i timestamp validi, firmati con Certificate Transparency (CT), dai log CT noti, siano presentati per i certificati server (X.509) su un dominio. ***(iOS 10+, macOS 10.12+)***
`NSExceptionAllowsInsecureHTTPLoads`	Impostare su `YES` per consentire HTTP sul dominio selezionato.
`NSExceptionRequiresForwardSecrecy`	Predefinito a `YES` ; Impostare su `NO` per

Parametro	Dettagli
	disabilitare il segreto in avanti e accettare più codici.
`NSExceptionMinimumTLSVersion`	Il valore predefinito è `TLSv1.2` ; I valori possibili sono: `TLSv1.0` , `TLSv1.1` , `TLSv1.2`
`NSThirdPartyExceptionAllowsInsecureHTTPLoads`	Simile a `NSExceptionAllowsInsecureHTTPLoads` , ma per domini su cui non hai alcun controllo
`NSThirdPartyExceptionRequiresForwardSecrecy`	Simile a `NSExceptionRequiresForwardSecrecy` , ma per domini su cui non hai alcun controllo
`NSThirdPartyExceptionMinimumTLSVersion`	Simile a `NSExceptionMinimumTLSVersion` , ma per domini su cui non hai alcun controllo

Osservazioni

App Transport Security è una funzionalità di sicurezza in iOS e macOS. Impedisce alle app di stabilire connessioni non protette. Per impostazione predefinita, le app possono utilizzare solo connessioni HTTPS protette.

Se un'applicazione deve connettersi a un server tramite HTTP, è necessario definire le eccezioni in `Info.plist` . (vedi gli esempi per maggiori informazioni a riguardo)

Nota: nel 2017, Apple applicherà ATS. Ciò significa che non è più possibile caricare app con eccezioni ATS definite in `Info.plist` . Se è possibile fornire buoni argomenti, perché è necessario utilizzare HTTP, è possibile contattare Apple e potrebbero consentire di definire eccezioni. (Fonte: WWDC 2016 - Session 706)

Ulteriori informazioni sulla configurazione di App Transport Security sono disponibili nella documentazione di CocoaKeys .

Examples

Carica tutto il contenuto HTTP

Apple ha introdotto ATS con iOS 9 come nuova funzionalità di sicurezza per migliorare la privacy e la sicurezza tra app e servizi web. Per impostazione predefinita, ATS non riesce tutte le richieste non HTTPS. Mentre questo può essere davvero bello per gli ambienti di produzione, può essere fastidioso durante i test.

ATS è configurato nel file `Info.plist` della destinazione con il dizionario `NSAppTransportSecurity` (`App Transport Security Settings` nell'editor Xcode Info.plist). Per consentire tutto il contenuto HTTP, aggiungi il valore `Allow Arbitrary Loads` booleani `Allow Arbitrary Loads` (`NSAllowsArbitraryLoads`) e `NSAllowsArbitraryLoads` su `YES` . Questo non è consigliato per le app di

produzione e, se è richiesto il contenuto HTTP, si consiglia di abilitarlo selettivamente.

Caricare in modo selettivo il contenuto HTTP

Simile all'abilitazione di tutti i contenuti HTTP, tutte le configurazioni avvengono in `App Transport Security Settings` . Aggiungere il dizionario dei `Exception Domains` (`NSExceptionDomains`) alle impostazioni ATS di livello superiore.

Per ogni dominio, aggiungi un elemento del dizionario ai domini di eccezione, dove la chiave è il dominio in questione. Impostare `NSExceptionAllowsInsecureHTTPLoads` su `YES` per disabilitare il requisito HTTPS per quel dominio.

Gli endpoint richiedono SSL

Introdotto in iOS 9, tutti gli endpoint devono rispettare le specifiche HTTPS.
Qualsiasi endpoint che non utilizza SSL non riuscirà con un avviso nel log della console. Alla tua applicazione sembrerà che la connessione internet non sia riuscita.

Per configurare le eccezioni: inserire quanto segue nel file Info.plist:

1. Consentire particolare dominio (testdomain.com) **solo:**

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSExceptionDomains</key>
<dict>
    <key>testdomain.com</key>
    <dict>
        <key>NSIncludesSubdomains</key>
        <true/>
        <key>NSExceptionAllowsInsecureHTTPLoads</key>
        <true/>
    </dict>
</dict>
```

La chiave che consente tale comportamento è `NSExceptionAllowsInsecureHTTPLoads` . In questo caso, l'app consentirà solo la connessione HTTP al dominio menzionato (testdomain.com) e bloccherà tutte le altre connessioni HTTP.

La chiave `NSIncludesSubdomains` specifica che anche tutti i **sottodomini** del dominio menzionato (testdomain.com) dovrebbero essere consentiti.

2. Consenti qualsiasi dominio:

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

In questo caso, l'app consentirà la connessione HTTP a **qualsiasi** dominio. A partire dal 1 °

gennaio 2017, l'utilizzo di questo flag causerà una revisione approfondita di App Store e gli sviluppatori di app dovranno spiegare perché devono utilizzare questa eccezione in primo luogo. Le possibili spiegazioni includono:

- Un'applicazione che carica contenuti multimediali crittografati che non contengono informazioni personalizzate.
- Connessioni a dispositivi che non possono essere aggiornati per utilizzare connessioni sicure.
- Connessione a un server che è gestito da un'altra entità e non supporta connessioni sicure.

Leggi App Transport Security (ATS) online: https://riptutorial.com/it/ios/topic/5435/app-transport-security--ats-

Capitolo 12: AppDelegate

introduzione

AppDelegate è un protocollo che definisce i metodi che vengono chiamati dall'oggetto UIA singleton in risposta a eventi importanti nel corso della vita di un'app.

Normalmente utilizzato per eseguire attività all'avvio dell'applicazione (ambiente dell'app di configurazione, analisi (es .: Mixpanel / GoogleAnalytics / Crashlitics), stack DB ecc.) E arresto (es .: salva contesto DB), gestione richieste URL aperte e simili a livello di applicazione compiti.

Examples

Tutti gli stati dell'applicazione tramite i metodi AppDelegate

Per informazioni su come aggiornare o fare qualcosa prima che l'app venga resa disponibile all'utente, puoi utilizzare il metodo seguente.

AppDidFinishLaunching

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

Mentre App entra in primo piano:

```
- (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.
}
```

Quando l'avvio di app e anche lo sfondo in primo piano, colpiscono il metodo seguente:

```
- (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.
}
```

Mentre App entra in background:

```
- (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.
}
```

Mentre l'app si dimette attiva

```
- (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.
}
```

Mentre l'app termina:

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

AppDelegate Roles:

- AppDelegate contiene il `startup code` dell'app.
- Risponde ai `key changes` nello `state` della tua app. In particolare, risponde sia alle interruzioni temporanee che alle modifiche nello stato di esecuzione della tua app, come quando la tua app passa dal primo piano allo sfondo.
- `responds to notifications` provenienti dall'esterno dell'app, quali notifiche remote (note anche come notifiche push), avvisi di memoria insufficiente, notifiche di completamento del download e altro.
- `determines` se la `state preservation` e il `restoration` dovrebbero verificarsi e assiste nel processo di conservazione e restauro, se necessario.
- `responds to events` che `responds to events` come target l'app stessa e non sono specifici per le visualizzazioni o i controller della vista della tua app. Puoi utilizzarlo per archiviare gli oggetti dati centrali della tua app o qualsiasi contenuto che non dispone di un controller di visualizzazione proprietario.

Apertura di una risorsa specificata da URL

Chiede al delegato di aprire una risorsa specificata da un URL e fornisce un dizionario di opzioni di lancio.

Esempio di utilizzo:

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

```
    }
```

Gestione delle notifiche locali e remote

Esempio di utilizzo:

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

Registrazione:

```
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)
}
```

Gestione delle notifiche remote:

```
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
    )
}
```

Gestione delle notifiche locali:

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

Azione di gestione (deprecata):

```
func application(application: UIApplication, handleActionWithIdentifier identifier: String?,
forRemoteNotification userInfo: [NSObject : AnyObject],
```

```
                    completionHandler: () -> Void) {
    pushManager.handleInteractiveRemoteNotification(userInfo, actionIdentifier: identifier,
completion: completionHandler)
}
```

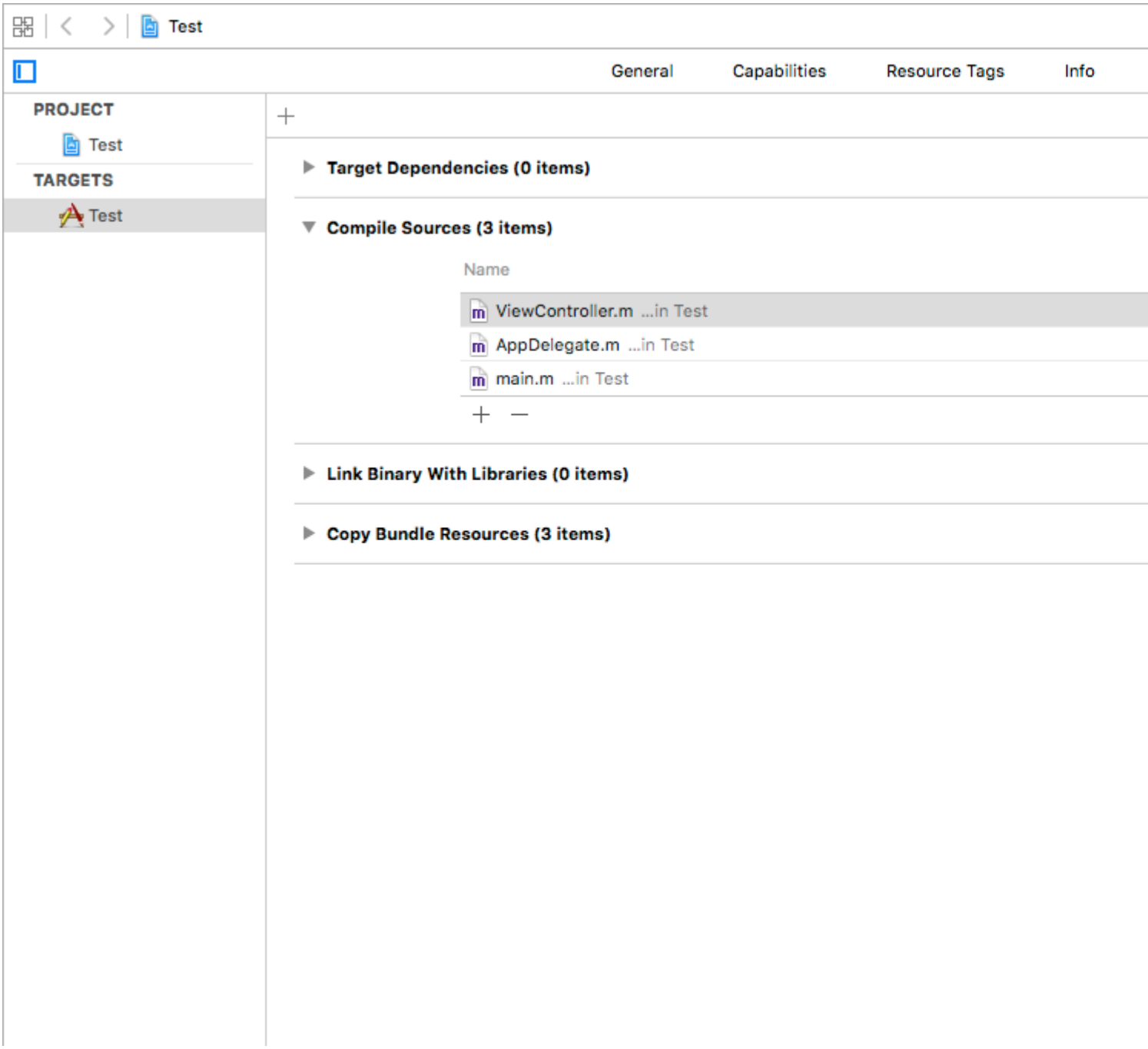
Leggi AppDelegate online: https://riptutorial.com/it/ios/topic/8740/appdelegate

Capitolo 13: ARC (conteggio di riferimento automatico)

Examples

Abilita / disabilita ARC su un file

ARC può essere disabilitato per singoli file aggiungendo il `-fno-objc-arc` compilatore `-fno-objc-arc` per ogni file. Viceversa può essere aggiunto in *Obiettivi* ▸ `Fasi di creazione` ▸ `Fonti di compilazione`



Leggi ARC (conteggio di riferimento automatico) online: https://riptutorial.com/it/ios/topic/4150/arc--conteggio-di-riferimento-automatico-

Capitolo 14: Architettura MVP

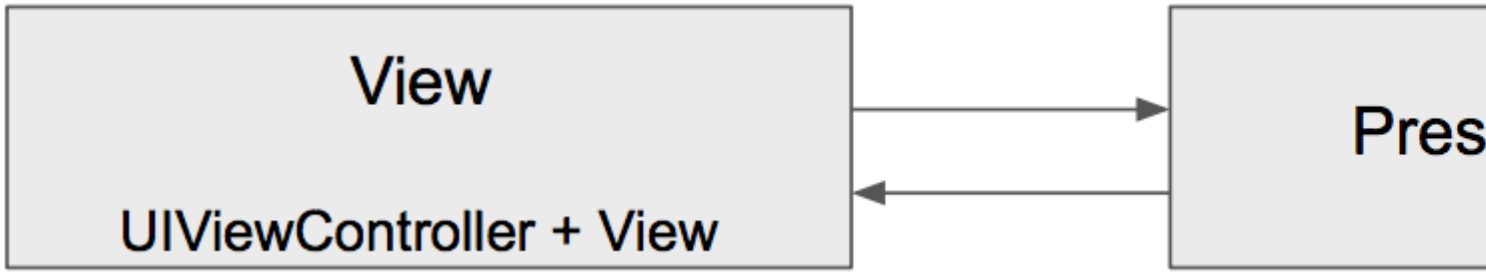
introduzione

MVP è un modello architettonico, una derivazione del Model-View-Controller. È rappresentato da tre componenti distinti: Modello, Visualizza e Presentatore. È stato progettato per facilitare i test unitari automatizzati e migliorare la separazione delle preoccupazioni nella logica di presentazione.

Negli esempi troverai un semplice progetto costruito pensando al pattern MVP.

Osservazioni

componenti:



- **Il modello** è un'interfaccia responsabile per i dati del dominio (da visualizzare o altrimenti applicata nella GUI)
- **La vista** è responsabile del livello di presentazione (GUI)
- **Presenter** è "l'uomo medio" tra Modello e Vista. Reagisce alle azioni dell'utente eseguite sulla vista, recupera i dati dal modello e li formatta per la visualizzazione nella vista

Funzioni del componente:

Modello	vista	Presentatore
Comunica con il livello DB	Rende i dati	Esegue query sul modello
Alzare gli eventi appropriati	Riceve gli eventi	Formatta i dati dal modello
	Logica di validazione molto semplice	Invia dati formattati alla vista
		Logica di convalida complessa

Differenze tra MVC e MVP :

- La vista in MVC è strettamente accoppiata al controller, la parte View dell'MVP è composta da UIViews e UIViewController
- MVP View è il più stupido possibile e non contiene quasi nessuna logica (come in MVVM), MVC View ha alcune logiche di business e può interrogare il Modello
- MVP View gestisce i gesti dell'utente e delega l'interazione al Presenter, in MVC il Gestore gestisce gesti e comandi Modello
- Il modello MVP supporta molto il test delle unità, MVC ha un supporto limitato
- MVC Controller ha molte dipendenze UIKit, MVP Presenter non ne ha

Professionisti:

- MVP rende UIViewController una parte del componente View è stupido, passivo e ... meno massiccio;]
- La maggior parte della logica aziendale è incapsulata a causa delle viste stupide, questo offre un'eccellente testabilità. È possibile introdurre oggetti fittizi per testare la parte del dominio.
- Le entità separate sono più facili da tenere a mente, le responsabilità sono chiaramente divise.

Contro

- Scriverò più codice.
- Barriera per sviluppatori inesperti o per coloro che non lavorano ancora con il modello.

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;
    }
}
```

Leggi Architettura MVP online: https://riptutorial.com/it/ios/topic/9467/architettura-mvp

Capitolo 15: attribuitoText in UILabel

introduzione

Il testo in stile attuale che viene visualizzato dall'etichetta.

È possibile aggiungere testo `HTML` in `UILabel` utilizzando la proprietà attributeText o il testo singolo `UILabel` personalizzato con proprietà diverse

Examples

Testo HTML in 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;
```

Imposta proprietà diverse su testo nella singola UILabel

Il primo passaggio da eseguire è la creazione di un oggetto `NSMutableAttributedString` . Il motivo per cui creiamo `NSMutableAttributedString` anziché `NSAttributedString` è perché ci consente di aggiungere una stringa ad esso.

```
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];
```

Produzione :

Leggi attribuitoText in UILabel online: https://riptutorial.com/it/ios/topic/10927/attribuitotext-in-uilabel

Capitolo 16: AutoLayout di UIScrollView

Examples

ScrollableController

Quando si utilizza l'Autolayout con un `UIScrollView` , NON viene ridimensionato in modo appropriato a seconda delle dimensioni del suo contenuto o delle sue sottoview.

Al fine di ottenere un `UIScrollView` per scorrere automaticamente quando le informazioni diventano troppo grande per entrare nella zona visibile, abbiamo bisogno di aggiungere un `ContentView` e alcuni vincoli che consentono al `UIScrollView` di determinare le dimensioni del suo contenuto e la sua larghezza e altezza in suo genitore vista.

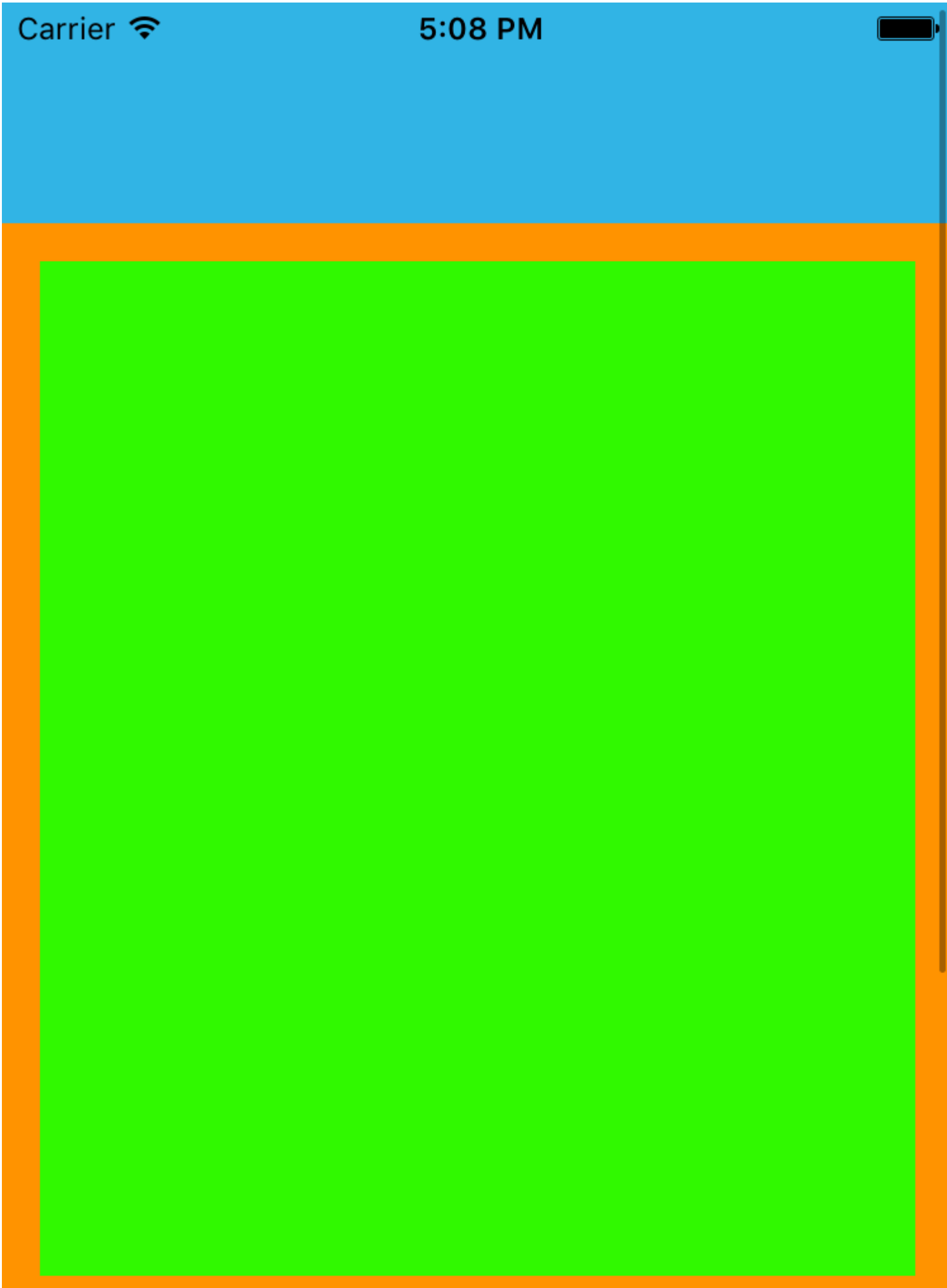
```
import Foundation
import UIKit

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
    }
}
```

Ora possiamo vedere che greenView (400 altezza) + whiteView (altezza 500) è più grande del nostro schermo. Ciò farà sì che il contenuto di ScrollViewSize cresca per adattarsi a ENTRAMBI le viste, permettendogli di scorrere verticalmente.

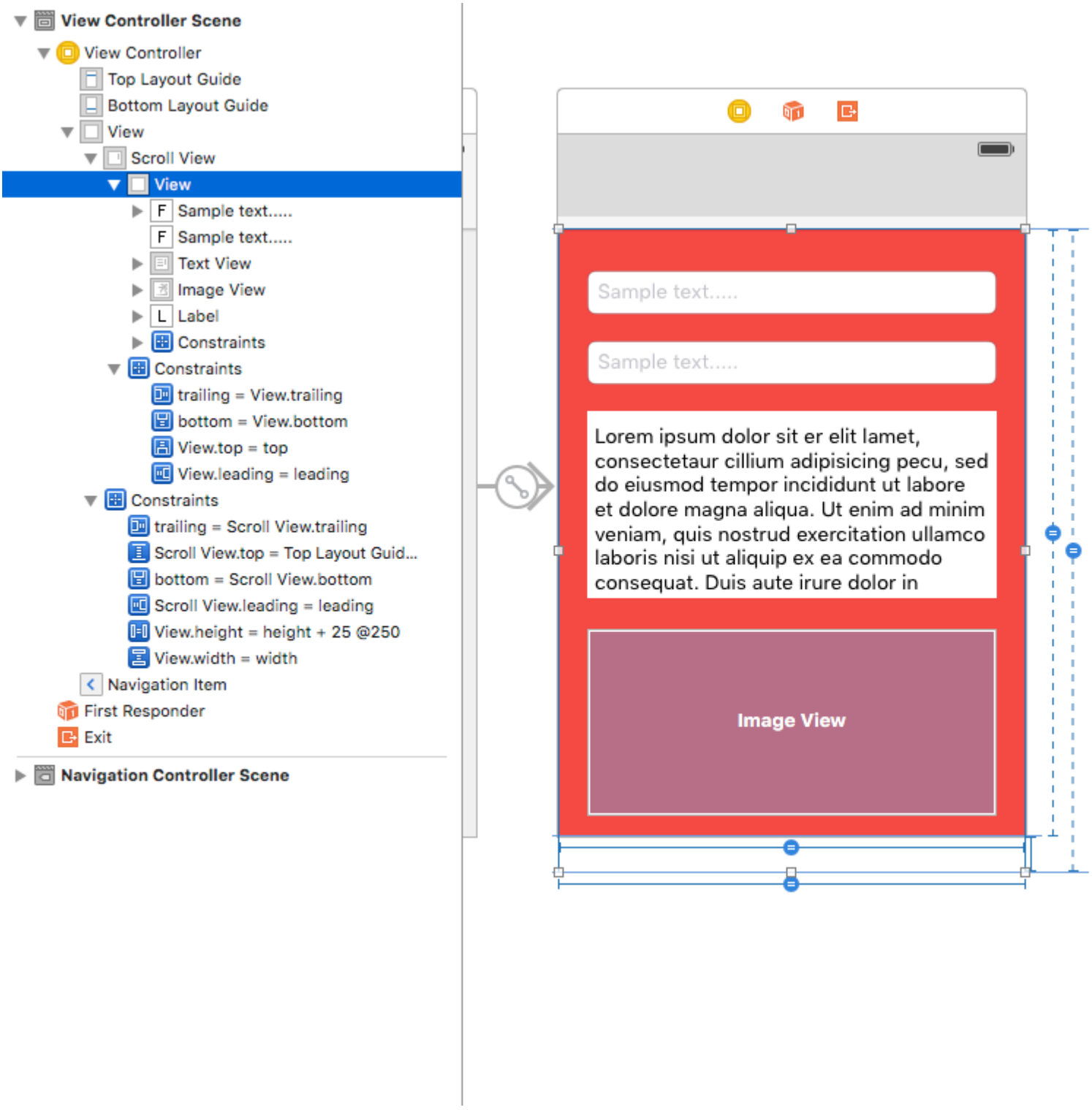
Abbiamo disabilitato lo scorrimento orizzontale usando il vincolo di `EqualWidth` su `contentView` e `self.view`

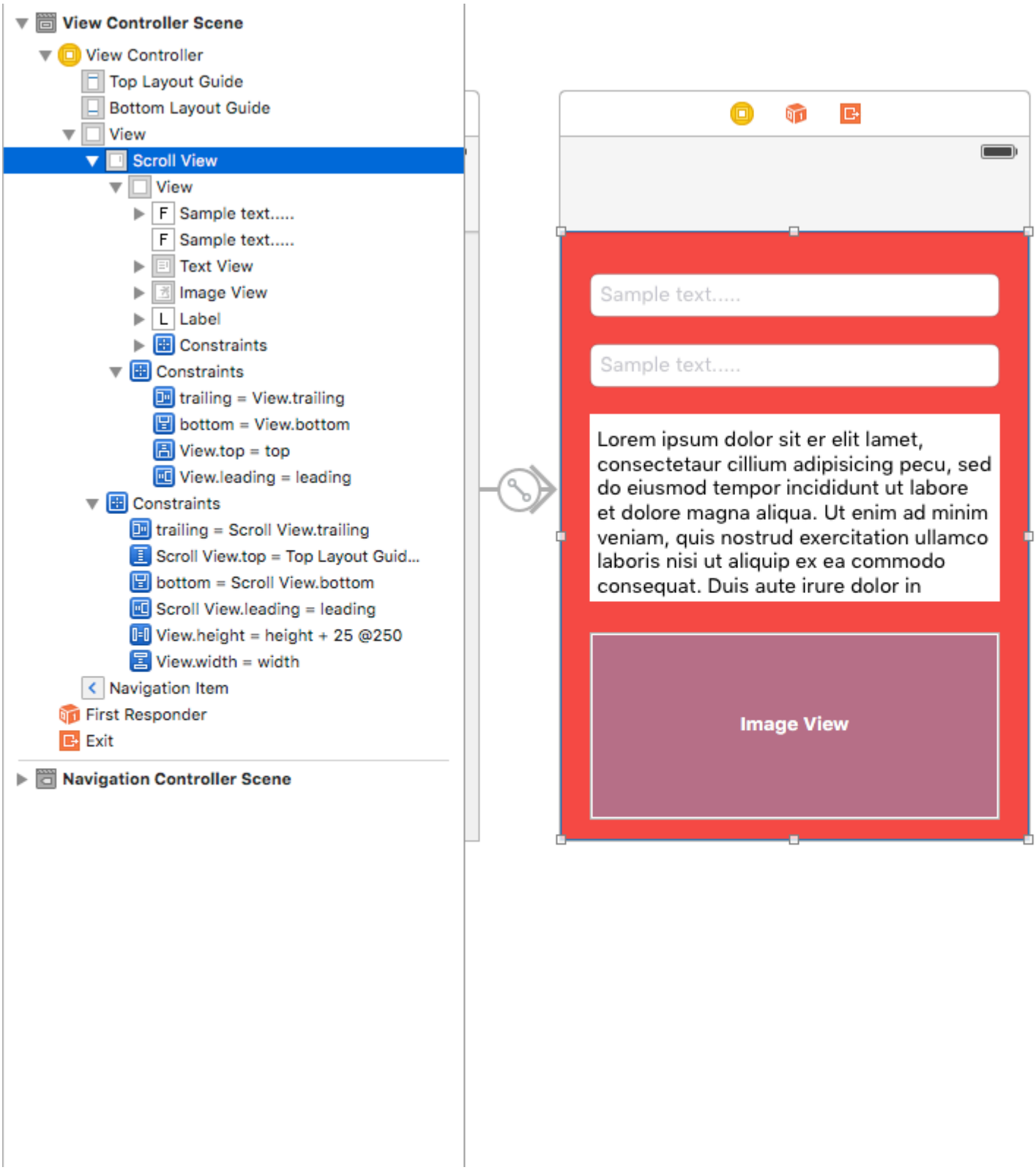

Carrier
5:08 PM

2. Aggiungi uguale altezza, larghezza uguale alla vista principale (cioè che contiene scrollview). Per uguale altezza impostare la priorità su bassa. (Questo è il passo importante per l'impostazione della dimensione del contenuto).

3. L'altezza di questa vista del contenuto sarà in base al numero di visualizzazioni aggiunte alla vista. diciamo che se hai aggiunto l'ultima visualizzazione è un'etichetta e la sua posizione Y è 420 e l'altezza è 20, quindi la visualizzazione del contenuto sarà 440.

Passaggio 3: aggiungere i vincoli a tutte le viste aggiunte nella visualizzazione del contenuto in base alle proprie esigenze.





Leggi AutoLayout di UIScrollView online: https://riptutorial.com/it/ios/topic/4671/autolayout-di-uiscrollview

Capitolo 17: AVPlayer e AVPlayerViewController

Osservazioni

importare AVKit, importare AVFoundation.

Examples

Riproduzione di contenuti multimediali con 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];
```

veloce

```
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()
}
```

Riproduzione di contenuti multimediali con AVPlayer e AVPlayerLayer

Obiettivo 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];
```

veloce

```
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()
```

Esempio di AVPlayer

AVPlayer * avPlayer = [AVPlayer playerWithURL: [NSURL URLWithString: @ "IL TUO 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
```

Leggi AVPlayer e AVPlayerViewController online: https://riptutorial.com/it/ios/topic/5092/avplayer-e-avplayerviewcontroller

Capitolo 18: AVSpeechSynthesizer

Sintassi

- AVSpeechSynthesizer () // Crea un sintetizzatore vocale
- speaker.speakUtterance (speech) // Converte il testo in sintesi vocale

Parametri

Parametro	Dettagli
altoparlante	Oggetto AVSpeechSynthesizer
discorso	Oggetto AVSpeechUtterance

Examples

Creazione di un testo di base per la sintesi vocale

Usa il `speakUtterance:` metodo di `AVSpeechSynthesizer` per convertire il testo in parlato. È necessario passare un oggetto `AVSpeechUtterance` a questo metodo, che contiene il testo che si desidera venga pronunciato.

Obiettivo C

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

veloce

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

Leggi AVSpeechSynthesizer online: https://riptutorial.com/it/ios/topic/1526/avspeechsynthesizer

Capitolo 19: Barra di navigazione

Examples

Personalizza l'aspetto della barra di navigazione predefinita.

```
// 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]];
```

Esempio 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
```

Leggi Barra di navigazione online: https://riptutorial.com/it/ios/topic/7066/barra-di-navigazione

Capitolo 20: Bloccare

Sintassi

- Come variabile:

  returnType (^ blockName) (parameterTypes) = ^ returnType (parametri) {...};

- Come una proprietà:

  @property (nonatomic, copy) returnType (^ blockName) (parameterTypes);

- Come parametro del metodo:

  - (void) methodWithBlock: (returnType (^) (parameterTypes)) blockName;

- Come typedef:

  typedef returnType (^ TypeName) (parameterTypes);

  TypeName blockName = ^ returnType (parameters) {...};

Examples

Animazioni UIView

```
[UIView animateWithDuration:1.0
    animations:^{
        someView.alpha = 0;
        otherView.alpha = 1;
    }
    completion:^(BOOL finished) {
    [someView removeFromSuperview];
}];
```

Il carattere carato "^" definisce un blocco. Ad esempio, `^{ … }` è un blocco. Più specificamente, è un blocco che restituisce "void" e non accetta argomenti. È equivalente a un metodo come: "- (void) something;" ma non esiste un nome intrinseco associato al blocco di codice.

Definire un blocco che possa accettare argomenti funziona in modo molto simile. Per fornire un argomento a un blocco, si definisce il blocco in questo modo: ^ *(BOOL someArg, NSString someStr) {...}* *. Quando si utilizzano chiamate API che supportano i blocchi, si scriveranno blocchi simili a questo, in particolare per i blocchi di animazione o i blocchi NSURLConnection, come mostrato nell'esempio precedente.

Blocco di completamento personalizzato per metodi personalizzati

1- Definisci il tuo blocco personalizzato

```
typedef void(^myCustomCompletion)(BOOL);
```

2- Creare un metodo personalizzato che accetta il blocco di completamento personalizzato come parametro.

```
-(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- Come utilizzare il blocco nel metodo

```
[self customMethodName:^(BOOL finished) {
if(finished){
    NSLog(@"success");
}
}];
```

Modifica la variabile catturata

Il blocco catturerà le variabili che sono apparse nello stesso ambito lessicale. Normalmente queste variabili sono catturate come valore "const":

```
int val = 10;
void (^blk)(void) = ^{
    val = 20; // Error! val is a constant value and cannot be modified!
};
```

Per modificare la variabile, è necessario utilizzare il modificatore del tipo di memoria __block.

```
__block int val = 10;
void (^blk)(void) = ^{
    val = 20; // Correct! val now can be modified as an ordinary variable.
};
```

Leggi Bloccare online: https://riptutorial.com/it/ios/topic/6888/bloccare

Capitolo 21: Build Simulator

introduzione

Dove trovare la build del simulatore?

Vai a ~ / Libreria / Sviluppatore / CoreSimulator / Dispositivi /

Troverai directory con nomi alfanumerici

quindi fare clic su una delle directory e effettuare la seguente selezione

Dati / Contenitori / Bundle / Applicazioni /

Di nuovo troverai le directory con nomi alfanumerici se fai clic su quello che troverai

Costruisci il simulatore laggiù

Nota:

L'installazione del dispositivo iOS sul simulatore non funzionerà.

Il simulatore iPhone utilizza i costruttori di simulatore per iPad i386 architettura usa x8

Examples

Installazione manuale del build sul simulatore

```
xcrun simctl install booted *.app
```

Leggi Build Simulator online: https://riptutorial.com/it/ios/topic/9813/build-simulator

Capitolo 22: CAAnimation

Osservazioni

CAAnimation è una classe di animazione astratta. Fornisce il supporto di base per i protocolli **CAMediaTiming** e **CAAction** . Per animare i livelli Core Animation o gli oggetti Scene Kit, creare istanze delle sottoclassi concrete **CABasicAnimation** , **CAKeyframeAnimation** , **CAAnimationGroup** o **CATransition** .

Examples

Animare una vista da una posizione all'altra.

Objective-C

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

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

veloce

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

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

La vista si sposterà da 0 a 320 orizzontalmente. se vuoi spostare la vista in verticale, basta sostituire keypath in questo modo:

```
"position.y"
```

Visualizza animazione - 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"];
```

SWIFT

```
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")
```

Giri la vista

```
 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"];
```

Shake View

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")
```

Push View Animation

Obiettivo 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"];
```

veloce

```
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")
```

Leggi CAAnimation online: https://riptutorial.com/it/ios/topic/981/caanimation

Capitolo 23: Cache immagini online

Examples

AlamofireImage

Memorizzazione nella cache delle immagini online tramite `AlamofireImage` . Funziona su `Alamofire` in Swift. Installa `AlamofireImage` usando `cocoapods`

```
pod 'AlamofireImage', '~> 3.1'
```

Impostare:

1. Importa `AlamofireImage` e `Alamofire`
2. Impostare la cache dell'immagine: `let imageCache = AutoPurgingImageCache( memoryCapacity: 111_111_111, preferredMemoryUsageAfterPurge: 90_000_000)`
3. Esecuzione di una richiesta e aggiunta dell'immagine alla cache:

```
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. Recupera immagini dalla cache:

```
if let image = imageCache.image(withIdentifier: self.nameUrl[self.a])
        {
            self.localImageView.image = image
        }
```

Per maggiori informazioni segui questo link

Leggi Cache immagini online online: https://riptutorial.com/it/ios/topic/9450/cache-immagini-online

Capitolo 24: CAGradientLayer

Sintassi

- CAGradientLayer () // Restituisce un oggetto CALayer inizializzato.
- CAGradientLayer (layer: layer) // Override per copiare o inizializzare i campi personalizzati del layer specificato.

Parametri

Parametro	Dettagli
colore	Una serie di oggetti `CGColorRef` che definiscono il colore di ciascun gradiente. Animatable.
posizioni	Una matrice opzionale di oggetti `NSNumber` che definisce la posizione di ciascun arresto del gradiente. Animatable.
endPoint	Il punto finale del gradiente quando disegnato nello spazio delle coordinate del livello. Animatable.
punto di partenza	Il punto iniziale del gradiente quando viene disegnato nello spazio delle coordinate del livello. Animatable.
genere	Stile del gradiente disegnato dal livello. Il valore predefinito è `kCAGradientLayerAxial` .

Osservazioni

- Utilizzare `startPoint` e `endPoint` per modificare l'orientamento di `CAGradientLayer` .
- Usa le `locations` per influenzare la diffusione / le posizioni dei colori.

Examples

Creazione di un 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)
```

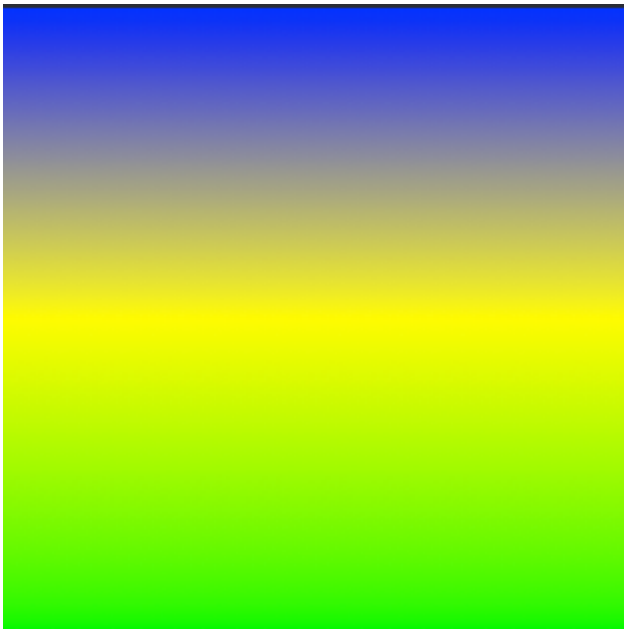
Risultato:

Creazione di un CGGradientLayer con più colori.

```
// 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)
```

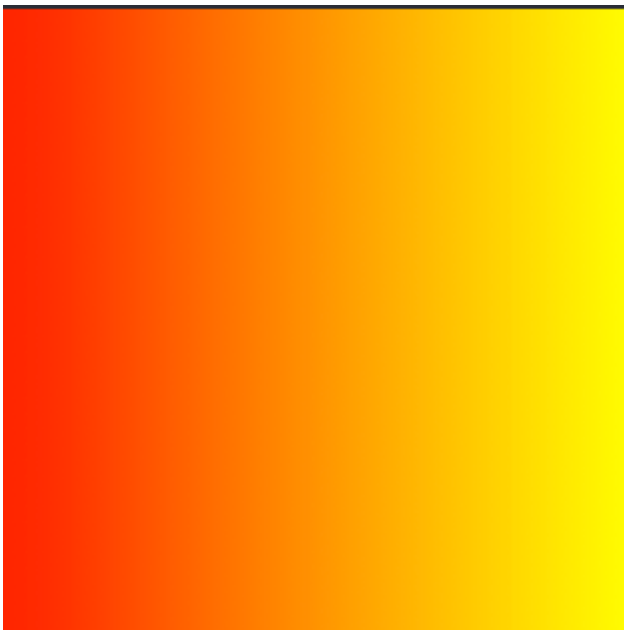
Risultato:



Creazione di un CAGradientLayer orizzontale.

```
// 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)
```

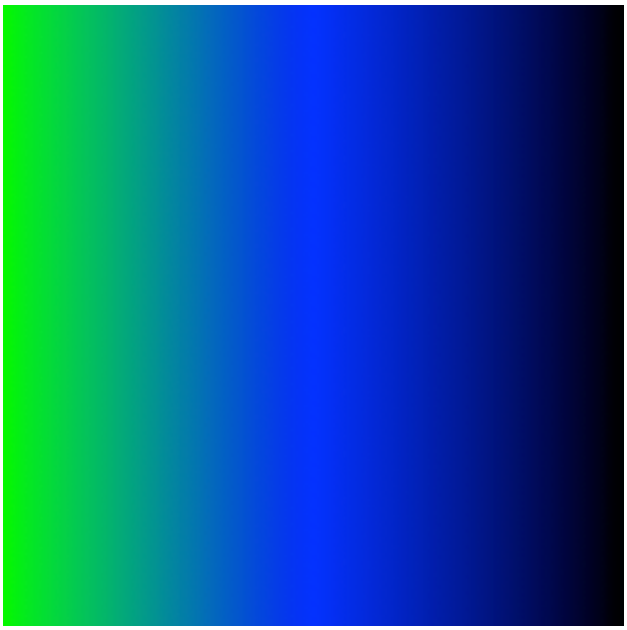
Risultato:



Creazione di un CAGradientLayer orizzontale con più colori.

```
// 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)
```

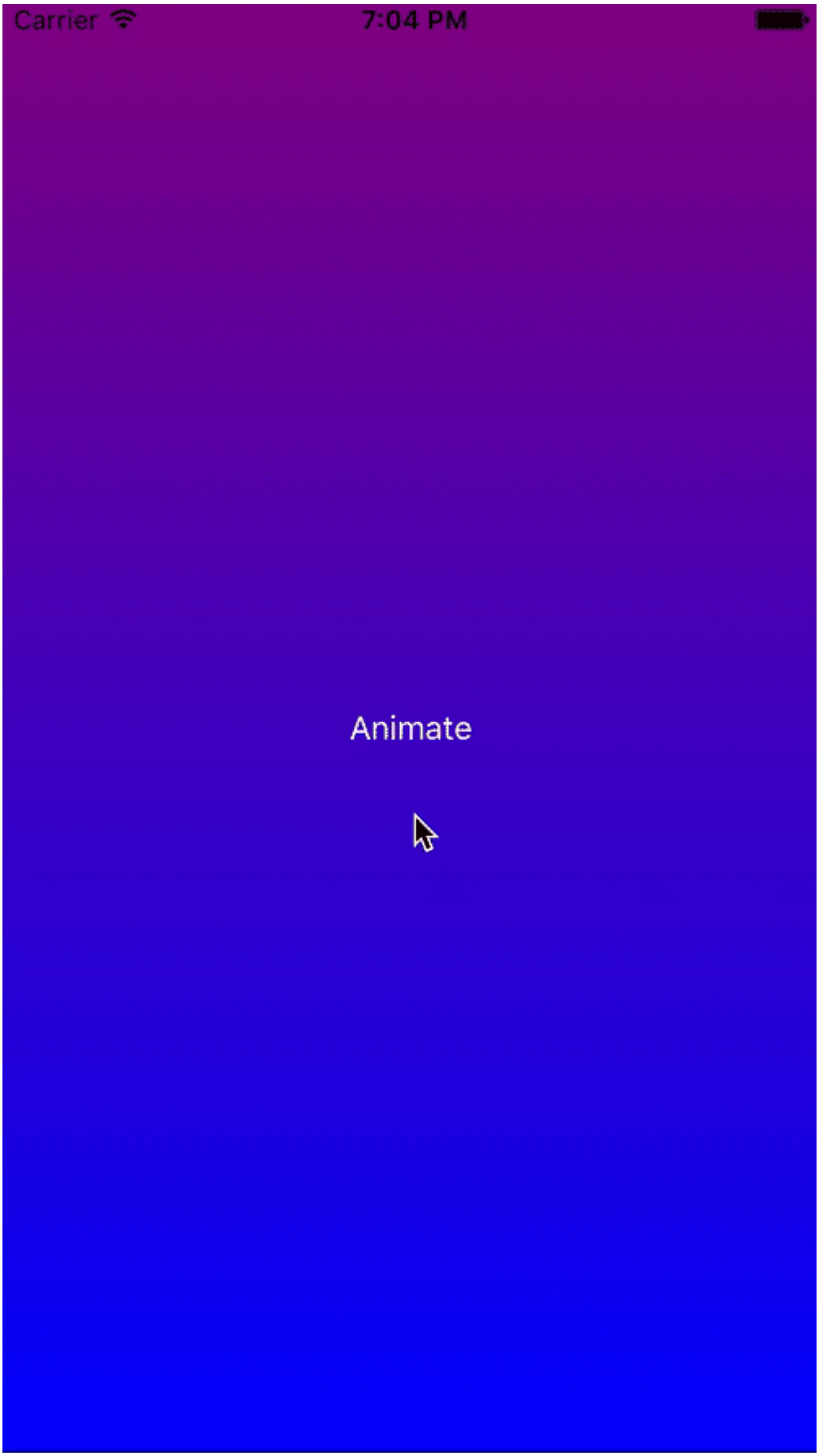
Risultato:



Animazione di un cambiamento di colore in 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")
```

Risultato:



Leggi CAGradientLayer online: https://riptutorial.com/it/ios/topic/1190/cagradientlayer

Capitolo 25: CALayer

Examples

Creazione di un CALayer

Puoi creare un CALayer e impostarne la cornice in questo modo:

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);
```

È quindi possibile aggiungerlo come sottolivello a un CALayer esistente:

Swift:

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

Objective-C:

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

Nota:

Per fare questo è necessario includere il framework QuartzCore.

Swift:

```
 @import QuartzCore
```

Objective-C

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

Creazione di particelle con CAEmitterLayer

La classe **CAEmitterLayer** fornisce un sistema di emittenti di particelle per Core Animation. Le particelle sono definite da istanze di **CAEmitterCell** .

Le particelle sono disegnate sopra il colore e il bordo dello sfondo del livello.

```
        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)
```

Vista emettitore con immagine personalizzata

Ad esempio creeremo una vista che contiene il livello di emettitore e anima le particelle.

```
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
    }
}
```

È necessario aggiungere l'immagine "confetti" o definire rect con **confetti.contentsRect**

Come aggiungere una UIImage a un CALayer

È possibile aggiungere un'immagine a di una vista `layer` semplicemente utilizzando il suo `contents` di proprietà:

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

- Si noti che `UIImage` deve essere convertito in un `CGImage` .

Se desideri aggiungere l'immagine nel proprio livello, puoi farlo in questo modo:

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

Modifica l'aspetto

Il codice precedente produce una vista come questa. L'azzurro è l' `UIView` e la stella blu scuro è `UIImage` .

Come puoi vedere, però, sembra pixelato. Questo perché `UIImage` è più piccolo di `UIView` quindi viene ridimensionato per riempire la vista, che è l'impostazione predefinita e non specifica altro.

Gli esempi seguenti mostrano variazioni sulla proprietà `contentsGravity` del `contentsGravity` del layer. Il codice si presenta così:

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

In iOS, potresti voler impostare la proprietà `geometryFlipped` su `true` se stai facendo qualcosa con gravità superiore o inferiore, altrimenti sarà l'opposto di quello che ti aspetti. (Solo la gravità viene ruotata verticalmente, non il rendering del contenuto. Se riscontri problemi con il contenuto capovolto, consulta questa risposta sull'overflow dello stack .)

Ci sono due `UIView` esempi indicati per ogni `contentsGravity` impostazione, una vista è maggiore della `UIImage` e l'altro è più piccolo. In questo modo puoi vedere gli effetti del ridimensionamento e della gravità.

`kCAGravityResize`

Questo è l'impostazione predefinita.

`kCAGravityResizeAspect`

kCAGravityResizeAspectFill

kCAGravityCenter

kCAGravityTop

kCAGravityBottom

kCAGravityLeft

kCAGravityRight

kCAGravityTopLeft

kCAGravityTopRight

kCAGravityBottomLeft

kCAGravityBottomRight

Relazionato

- Proprietà della modalità Contenuto di una vista
- Disegnare una `UIImage` in `drawRect` con `CGContextDrawImage`
- Esercitazione su CALayer: come iniziare

Gli appunti

- Questo esempio proviene originariamente da questa risposta di Overflow dello stack .

Aggiungere trasformazioni a un CALayer (tradurre, ruotare, ridimensionare)

Nozioni di base

Ci sono un certo numero di trasformazioni diverse che puoi fare su un livello, ma quelle di base sono

- tradurre (spostare)
- scala
- ruotare

Per eseguire trasformazioni su un `CALayer` , impostare la proprietà di `transform` del livello su un tipo `CATransform3D` . Ad esempio, per tradurre un livello, si dovrebbe fare qualcosa del genere:

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

La parola `Make` è usata nel nome per creare la trasformazione iniziale: CATransform3D **Make** Translation. Le trasformazioni successive che vengono applicate omettono il `Make` . Vedi, per esempio, questa rotazione seguita da una traduzione:

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

Ora che abbiamo le basi su come realizzare una trasformazione, diamo un'occhiata ad alcuni esempi su come fare ognuno di essi. Prima, però, mostrerò come ho impostato il progetto nel caso in cui vogliate giocare con esso.

Impostare

Per gli esempi che seguono ho impostato un'applicazione di visualizzazione singola e aggiunto uno `UIView` con uno sfondo blu chiaro allo storyboard. Ho collegato la vista al controller della vista con il seguente codice:

```
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 ...


    }

}
```

Esistono molti tipi diversi di `CALayer` , ma ho scelto di utilizzare `CATextLayer` modo che le trasformazioni siano visivamente più chiare.

Tradurre

La trasformazione di traduzione sposta il livello. La sintassi di base è

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

dove `tx` è il cambiamento nelle coordinate x, `ty` è il cambiamento in y, e `tz` è il cambiamento in z.

Esempio

In iOS l'origine del sistema di coordinate è in alto a sinistra, quindi se volessimo spostare il livello di 90 punti verso destra e 50 punti verso il basso, faremo quanto segue:

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

Gli appunti

- Ricorda che puoi incollarlo nel metodo `transformExample()` nel codice progetto precedente.
- Dato che stiamo andando a trattare due dimensioni qui, `tz` è impostato a `0` .
- La linea rossa nell'immagine qui sopra va dal centro della posizione originale al centro della nuova posizione. Questo perché le trasformazioni vengono eseguite in relazione al punto di ancoraggio e il punto di ancoraggio di default si trova al centro del livello.

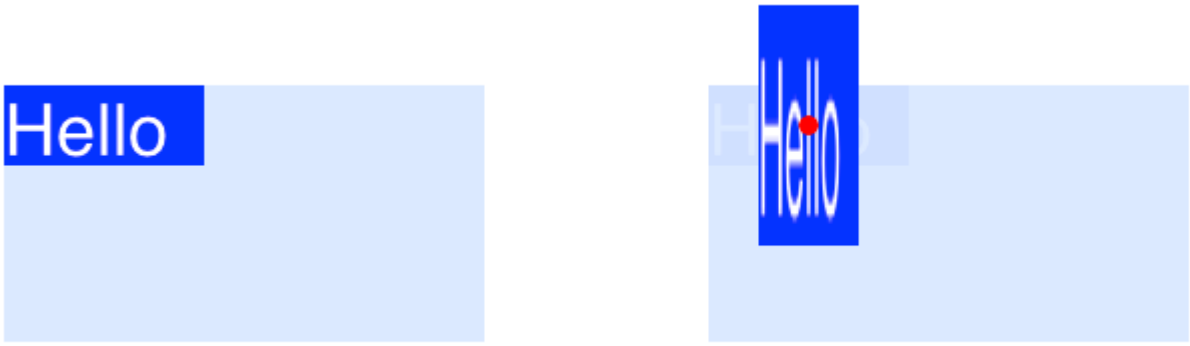
Scala

La scala trasforma tratti o schiaccia il livello. La sintassi di base è

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

dove `sx` , `sy` e `sz` sono i numeri con cui ridimensionare (moltiplicare) le coordinate x, y e z rispettivamente.

Esempio



Se volessimo metà della larghezza e triplicare l'altezza, faremmo quanto segue

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

Gli appunti

- Dato che stiamo lavorando solo in due dimensioni, moltiplichiamo le coordinate z per 1.0 per lasciarle inalterate.
- Il punto rosso nell'immagine sopra rappresenta il punto di ancoraggio. Nota come il ridimensionamento è fatto in relazione al punto di ancoraggio. Cioè, tutto è teso verso o lontano dal punto di ancoraggio.

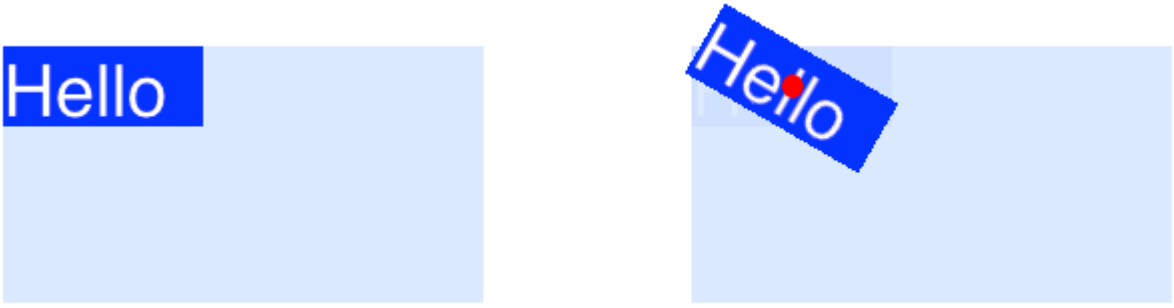
Ruotare

La trasformazione di rotazione ruota lo strato attorno al punto di ancoraggio (il centro del livello di default). La sintassi di base è

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

dove `angle` è l'angolo in radianti che lo strato deve essere ruotato e `x` , `y` e `z` sono gli assi su cui ruotare. Impostando un asse su 0 si annulla una rotazione attorno a quel particolare asse.

Esempio



Se volessimo ruotare uno strato in senso orario di 30 gradi, faremmo quanto segue:

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

Gli appunti

- Dato che stiamo lavorando in due dimensioni, vogliamo solo che il piano xy sia ruotato attorno all'asse z. Così abbiamo impostato `x` ed `y` di `0.0` e impostare `z` a `1.0` .
- Questo ha ruotato lo strato in senso orario. Potremmo aver ruotato in senso antiorario impostando `z` su `-1.0` .
- Il punto rosso indica dove si trova il punto di ancoraggio. La rotazione viene eseguita attorno al punto di ancoraggio.

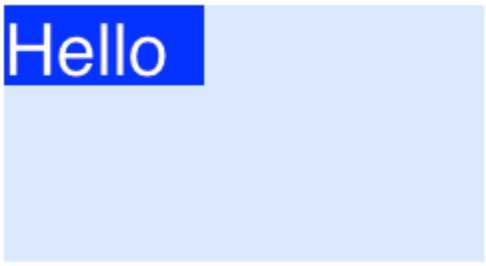
Trasformazioni multiple

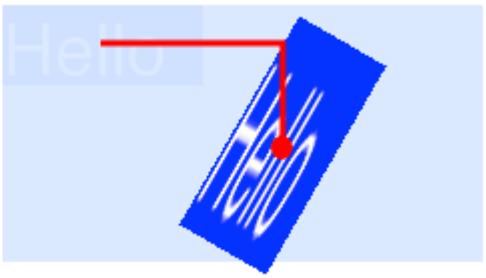
Per combinare più trasformazioni, potremmo usare la concatinazione come questa

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

Tuttavia, faremo solo uno dopo l'altro. La prima trasformazione utilizzerà la `Make` nel suo nome. Le seguenti trasformazioni non useranno `Make` , ma prenderanno la trasformazione precedente come parametro.

Esempio





Questa volta combiniamo tutte e tre le trasformazioni precedenti.

```
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
```

Gli appunti

- L'ordine in cui le trasformazioni sono fatte in materia.
- Tutto è stato fatto in relazione al punto di ancoraggio (punto rosso).

Una nota su punto di ancoraggio e posizione

Abbiamo fatto tutte le nostre trasformazioni sopra senza cambiare il punto di ancoraggio. A volte è necessario cambiarlo, però, come se si volesse ruotare attorno ad un altro punto oltre al centro. Tuttavia, questo può essere un po 'complicato.

Il punto di ancoraggio e la posizione sono entrambi nello stesso punto. Il punto di ancoraggio è espresso come unità del sistema di coordinate del livello (il valore predefinito è `0.5, 0.5`) e la posizione è espressa nel sistema di coordinate del superlayer. Possono essere impostati in questo modo

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

Se si imposta solo il punto di ancoraggio senza modificare la posizione, la cornice cambia in modo tale che la posizione si trovi nel punto giusto. O più precisamente, la cornice viene ricalcolata in base al nuovo punto di ancoraggio e alla vecchia posizione. Questo di solito dà risultati inaspettati. I seguenti due articoli hanno un'ottima discussione su questo.

- Informazioni sul punto di ancoraggio
- Traduci ruotare tradurre?

Guarda anche

- Bordo, angoli arrotondati e ombra su un `CALayer`
- Utilizzo di un bordo con un percorso di Bézier per un livello

Questo esempio deriva originariamente da questo esempio di overflow dello stack .

Disabilita animazioni

`CALayer` animazioni delle proprietà `CALayer` sono abilitate per impostazione predefinita. Quando questo non è desiderabile, possono essere disabilitati come segue.

veloce

```
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];
```

Angoli arrotondati

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

Shadows

Puoi usare 5 proprietà su ogni livello per configurare le tue ombre:

- `shadowOffset` : questa proprietà sposta l'ombra a sinistra / a destra o su / giù

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

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

- `shadowColor` : imposta il colore della tua ombra

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

- `shadowOpacity` : questa è l'opacità dell'ombra, da `0` a `1`

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

- `shadowRadius` : questo è il raggio di sfocatura (equivalente alla proprietà sfocatura di Sketch o Photoshop)

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

- `shadowPath` : questa è una proprietà importante per le prestazioni, quando unset di iOS non fa `shadowPath` l'ombra sul canale alfa della vista, che può richiedere prestazioni elevate con un PNG complesso con alfa. Questa proprietà ti consente di forzare una forma per la tua ombra ed essere più performante a causa di essa.

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
```

Leggi CALayer online: https://riptutorial.com/it/ios/topic/1462/calayer

Capitolo 26: Cambia colore barra di stato

Examples

Per barre di stato non UINavigationBar

1. In info.plist set `View controller-based status bar appearance YES`
2. Nella vista i controller non contenuti da `UINavigationController` implementano questo metodo.

In Objective-C:

```
- (UIStatusBarStyle)preferredStatusBarStyle
{
    return UIStatusBarStyleLightContent;
}
```

In Swift:

```
override func preferredStatusBarStyle() -> UIStatusBarStyle {
    return UIStatusBarStyle.LightContent
}
```

Per le barre di stato di UINavigationBar

Sottoclassi UINavigationController e quindi esegui l'override di questi metodi:

In Objective-C:

```
- (UIStatusBarStyle)preferredStatusBarStyle
{
    return UIStatusBarStyleLightContent;
}
```

In Swift:

```
override func preferredStatusBarStyle() -> UIStatusBarStyle {
    return .lightContent
}
```

In alternativa, puoi impostare `barStyle` nell'istanza `UINavigationBar` :

Obiettivo C:

```
// e.g. in your view controller's viewDidLoad method:
self.navigationController.navigationBar.barStyle = UIBarStyleBlack;  // this will give you a
white status bar
```

veloce

```
// e.g. in your view controller's viewDidLoad method:
navigationController?.navigationBar.barStyle = .black // this will give you a white status bar
```

`UIBarStyle` opzioni di `UIBarStyle` sono `default` , `black` , `blackOpaque` , `blackTranslucent` . Quest'ultimo 3 dovrebbe darti una barra di stato con testo bianco, solo gli ultimi due specificano l'opacità della barra.

Nota: puoi comunque modificare l'aspetto della barra di navigazione a tuo piacimento.

Se non riesci a cambiare il codice di ViewController

Se stai usando una libreria che contiene (ad esempio) AwesomeViewController con un colore della barra di stato sbagliato puoi provare questo:

```
  let awesomeViewController = AwesomeViewController()
  awesomeViewController.navigationBar.barStyle = .blackTranslucent // or other style
```

Per il contenimento del ViewController

Se stai utilizzando `UIViewControllerContainment` ci sono alcuni altri metodi che vale la pena guardare.

Quando vuoi un bambino viewController per controllare la presentazione della barra di stato (cioè se il bambino è posizionato nella parte superiore dello schermo

in Swift

```
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
    }
}
```

Modifica dello stile della barra di stato per l'intera applicazione

SWIFT:

Passo 1:

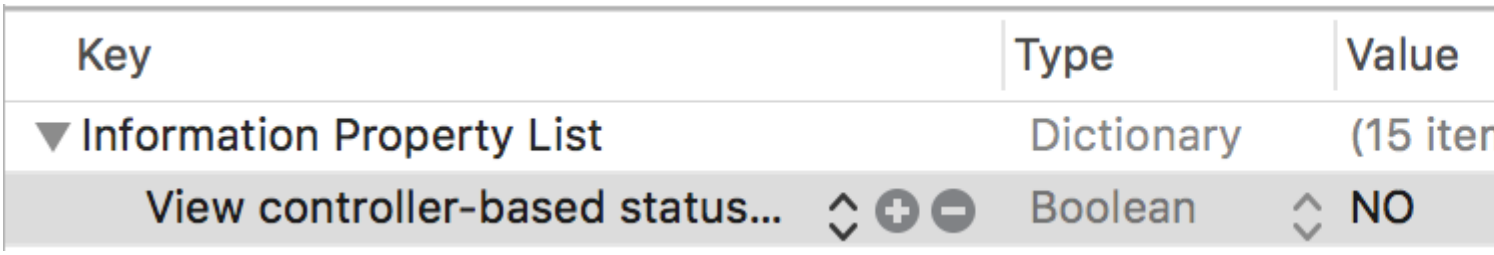
Nel tuo **Info.plist** aggiungi il seguente attributo:

```
View controller-based status bar appearance
```

e impostare il suo valore a

```
NO
```

come descritto nell'immagine qui sotto:



Passo 2:

Nel tuo file **AppDelegate.swift** , nel metodo `didFinishLaunchingWithOptions` , aggiungi questo codice:

```
UIApplication.shared.statusBarStyle = .lightContent
```

o

```
UIApplication.shared.statusBarStyle = .default
```

- L'opzione **.lightContent** imposterà il colore dello **statusBar** su bianco, per l'intera app.
- L'opzione **.default** imposterà il colore dello **statusBar** sul colore nero originale, per l'intera app.

Objective-C:

Segui il primo passaggio dalla sezione **SWIFT** . Quindi aggiungere questo codice al file **AppDelegate.m** :

```
[[UIApplication sharedApplication] setStatusBarStyle:UIStatusBarStyleLightContent];
```

o

```
[[UIApplication sharedApplication] setStatusBarStyle:UIStatusBarStyleDefault];
```

Leggi Cambia colore barra di stato online: https://riptutorial.com/it/ios/topic/378/cambia-colore-barra-di-stato

Capitolo 27: Caratteri personalizzati

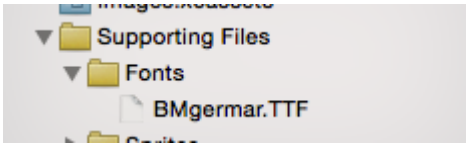
Examples

Incorporamento di caratteri personalizzati

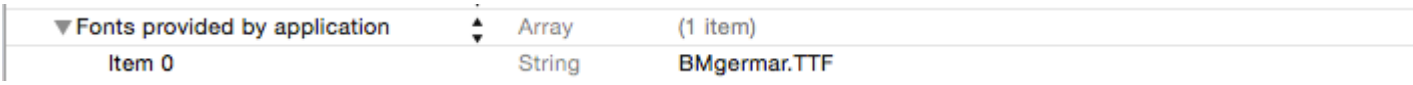
> **Supporto font personalizzato**
> Le applicazioni che desiderano utilizzare caratteri personalizzati possono ora includere quei tipi di carattere nel loro pacchetto di applicazioni e registrare tali caratteri con il sistema includendo la chiave UIAppFonts nel loro file Info.plist. Il valore di questa chiave è una serie di stringhe che identificano i file di caratteri nel pacchetto dell'applicazione. Quando il sistema vede la chiave, carica i caratteri specificati e li rende disponibili per l'applicazione.

Una volta che i font sono stati impostati in `Info.plist` , puoi usare i tuoi font personalizzati come qualsiasi altro font in IB o programmaticamente.

1. Trascina e rilascia il tuo font nella cartella Xcode Supporting Files. Non dimenticare di contrassegnare la tua app nella sezione "Aggiungi agli obiettivi". Da questo momento puoi usare questo font in IB e sceglierlo dal pallet dei font.



2. Per rendere questo carattere disponibile sul dispositivo, apri `Info.plist` e aggiungi `Fonts provided by application key` (UIAppFonts). Aggiungi il nome del font come valore alla chiave Item 0. Nota: il nome del carattere può variare dal nome del file del font.



3. Ottieni il nome del carattere personalizzato aggiunto utilizzando lo snippet sottostante

[*Swift 3*]

```
for family in UIFont.familyNames {
            print("\(family)")

            for name in UIFont.fontNames(forFamilyName: family) {
                print("   \(name)")
            }
        }
```

[*Obiettivo - C*]

```
for (NSString *familyName in [UIFont familyNames]){
        NSLog(@"Family name: %@", familyName);
        for (NSString *fontName in [UIFont fontNamesForFamilyName:familyName]) {
```

```
            NSLog(@"--Font name: %@", fontName);
        }
    }
```

Caratteri personalizzati con Storyboard

I caratteri personalizzati per i componenti dell'interfaccia utente dallo storyboard possono essere facilmente raggiunti con attributi definiti dall'utente in storyboard e categorie .

I vantaggi sono come,

- Non è necessario definire punti vendita per l'elemento ui
- Non è necessario impostare il font per gli elementi in modo programmatico.

  Passi da seguire

  1. **File Font:** aggiungi il file Font (.ttf) al pacchetto di applicazioni e aggiungi la voce per il font in Info.plist sotto ***Font fornito dall'applicazione*** come in questa documentazione di caratteri personalizzati.

  2. **Definisci categorie:** aggiungi un file come ***UIKit + IBExtensions*** e aggiungi le categorie per gli elementi dell'interfaccia utente come UILabel, UIButton ecc. Per i quali desideri impostare il carattere personalizzato. Tutte le categorie avranno una proprietà personalizzata come **fontName** . Questo verrà utilizzato dallo storyboard in seguito per l'impostazione del carattere personalizzato (come nel passaggio 4).

UIKit + IBExtensions.h

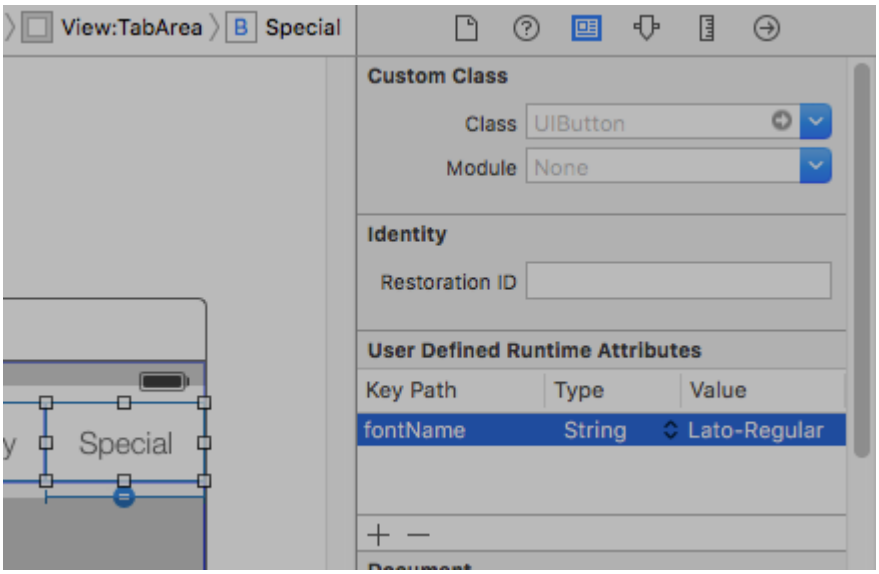
```
#import <UIKit/UIKit.h>

//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. **Getter e setter:** Definisci getter e setter per la proprietà fontName verso ogni categoria aggiunta.

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. **Impostazione del carattere nello storyboard:** aggiungere una voce in Attributi di runtime definiti dall'utente con ***fontName*** come keyPath e il ***nome del carattere personalizzato*** come valore con il tipo String come mostrato.



Questo imposterà il tuo font personalizzato durante l'esecuzione dell'app.

Gli appunti:

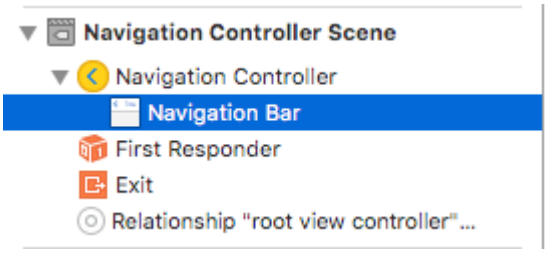
- Lato-Regular è il carattere personalizzato che ho usato.
- Lo stesso nome nel file ***.ttf*** aggiunto in bundle dovrebbe essere usato senza estensione nello storyboard.
- La dimensione del carattere sarà uguale a quella definita nell'ispettore degli attributi dell'elemento dell'interfaccia utente.

Applicazione di caratteri personalizzati ai controlli all'interno di uno storyboard

L'esempio seguente mostra come applicare caratteri personalizzati a una barra di navigazione e include correzioni per alcuni comportamenti bizzarri trovati in Xcode. Uno può anche applicare i caratteri personalizzati a **qualsiasi altro UIControls** come **UILabels** , **UIButtons** e altro utilizzando la **finestra di** ispezione attributi dopo che il carattere personalizzato è stato aggiunto al progetto. Si prega di notare i collegamenti esterni a campioni di lavoro e video nella parte inferiore.

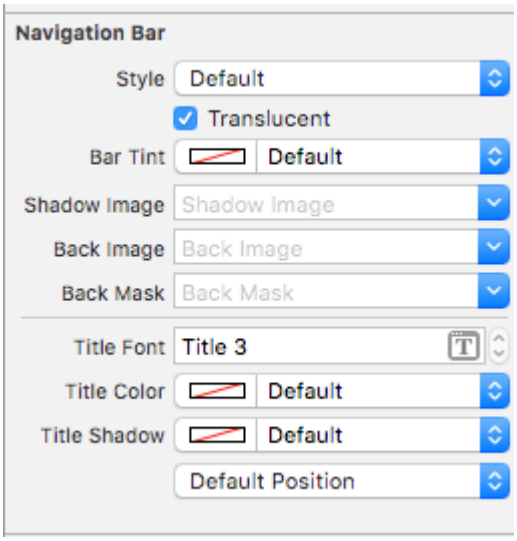
1. Seleziona la barra di navigazione all'interno del tuo controller di navigazione

---



2. Cambia il carattere del titolo nell'Inspector degli attributi

---

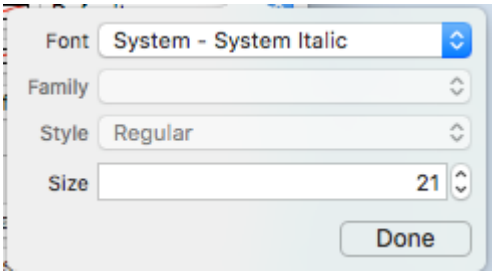


(Probabilmente dovrai cambiare la Tinta della barra per la barra di navigazione prima che Xcode prenda il nuovo carattere)

Note (caveat)

Verificato che questo funziona su Xcode 7.1.1+. (**Vedi gli esempi sotto**)

1. È necessario attivare la tinta della barra di navigazione prima che il font abbia effetto (sembra un bug in Xcode, puoi reimpostarlo su default e il font si bloccherà)
2. Se scegli un font di sistema ~ Assicurati che la dimensione non sia 0.0 (altrimenti il nuovo font sarà ignorato)



3. Sembra che questo funzioni senza problemi quando solo una NavBar si trova nella gerarchia della vista. Sembra che i NavBars secondari nello stesso stack siano ignorati. (Si noti che se si visualizza la barra di navigazione del controller di navigazione principale tutte le altre impostazioni personalizzate di navBar vengono ignorate).

Gotchas (deux)

Alcuni di questi vengono ripetuti, il che significa che è molto probabile che ne valga la pena.

1. A volte lo xml dello storyboard si corrompe. Ciò richiede che rivedi la struttura nello Storyboard come modalità Codice sorgente (fai clic con il pulsante destro del mouse sul file storyboard> Apri come ...)
2. In alcuni casi, il tag navigationItem associato all'attributo runtime definito dall'utente è stato impostato come figlio xml del tag view anziché del tag controller vista. In tal caso, rimuoverlo

tra i tag per il corretto funzionamento.
3. Attiva / disattiva la tinta della barra laterale per garantire che venga utilizzato il carattere personalizzato.
4. Verificare il parametro di dimensione del carattere a meno che non si utilizzi uno stile di carattere dinamico
5. La gerarchia della vista sovrascriverà le impostazioni. Sembra che sia possibile un font per stack.

Risultato

Campioni

- Video che mostra più font nel progetto avanzato
- Download semplice sorgente
- Download di progetti avanzati ~ Mostra più font NavBar e una soluzione personalizzata per i font
- Video che mostra più font e caratteri personalizzati

Gestione di caratteri personalizzati

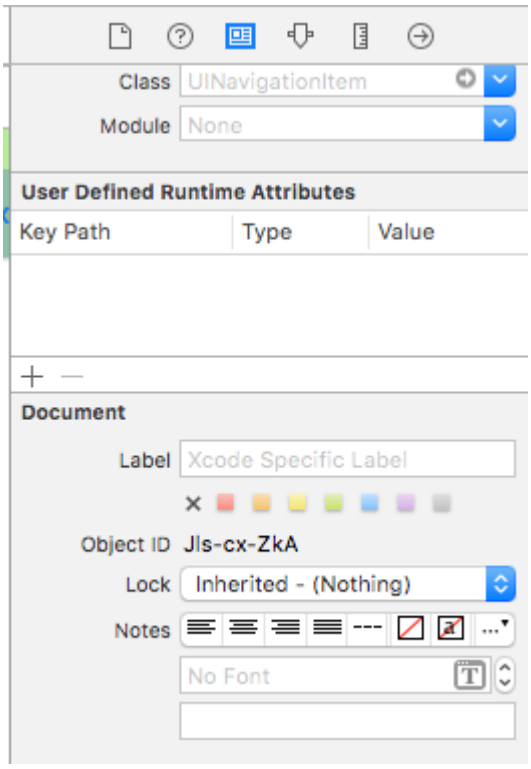
Nota ~ Una buona lista di controllo può essere trovata dal sito Web di Code With Chris e puoi vedere il progetto di download di esempio.

Se hai il tuo font e vuoi usarlo nello storyboard, allora c'è una serie decente di risposte sulla seguente SO Question . Una risposta identifica questi passaggi.

1. Ottieni file di font personalizzati (.ttf, .ttc)
2. Importa i file dei font sul tuo progetto Xcode
3. Nell'app-info.plist, aggiungi una chiave denominata Fonts fornita dall'applicazione. È un tipo di array, aggiungi tutti i nomi dei file dei font all'array, nota: inclusa l'estensione del file.
4. Nello storyboard, sulla barra di navigazione, vai a Impostazioni attributi, fai clic sul pulsante icona destra dell'area di selezione Carattere. Nel pannello popup, scegli Carattere su Personalizzato e scegli il nome di carattere Famiglia incorporata.

Soluzione per i caratteri personalizzati

Quindi Xcode sembra naturalmente in grado di gestire i caratteri personalizzati su UINavigationItem ma questa funzione non si aggiorna correttamente (il font selezionato viene ignorato).



Per risolvere questo problema:

Un modo è quello di correggere usando lo storyboard e aggiungendo una riga di codice: prima aggiungi un UIView (UIButton, UILabel o qualche altra sottoclasse UIView) al View Controller (non l'elemento di navigazione ... Xcode non sta attualmente permettendo di farlo quello). Dopo aver aggiunto il controllo, puoi modificare il carattere nello storyboard e aggiungere un riferimento come uscita al View Controller. Assegna semplicemente quella vista a UINavigationItem.titleView. È anche possibile impostare il nome del testo nel codice, se necessario. Segnalato bug (23600285).

```
@IBOutlet var customFontTitleView: UIButton!

//Sometime later...
self.navigationItem.titleView = customFontTitleView
```

Nota - Questo esempio è derivato da una risposta che ho postato su SO (qui).

Leggi Caratteri personalizzati online: https://riptutorial.com/it/ios/topic/1504/caratteri-personalizzati

Capitolo 28: Carica immagini asincrone

Examples

Modo più semplice

Il modo più semplice per crearlo è usare Alamofire e la sua UIImageViewExtension . Quello di cui abbiamo bisogno è una vista tabella con una cella che abbia un imageView al suo interno e che la chiami `imageView` .

Nella cellaForRowAt: funzione di tableView dovremmo scaricare l'immagine e impostarla nel seguente modo:

```
let url = URL(string: "https://httpbin.org/image/png")!
let placeholderImage = UIImage(named: "placeholder")!

imageView.af_setImage(withURL: url, placeholderImage: placeholderImage)
```

L'url dovrebbe puntare all'immagine che vuoi scaricare e l'immagine placeHolder dovrebbe essere un'immagine memorizzata. Chiamiamo quindi il metodo `af_setImage` su `imageView` che scarica l'immagine `imageView` specificato e durante il download verrà mostrata l'immagine del segnaposto. Non appena viene scaricata l'immagine, viene visualizzata l'immagine richiesta

Verifica che la cella sia ancora visibile dopo il download

A volte il download richiede più tempo rispetto a quando la cella viene visualizzata. In questo caso può succedere che l'immagine scaricata sia mostrata nella cella sbagliata. Per risolvere questo problema non possiamo usare l' estensione UIImageView .

Useremo comunque Alamofire, tuttavia useremo il gestore di completamento per visualizzare l'immagine.

In questo scenario abbiamo ancora bisogno di una tabellaView con una cella che ha un imageView in esso. Nel cellForRowAt: metodo dovremmo scaricare l'immagine con il seguente codice:

```
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
    }
}
```

In questo esempio, per prima cosa impostiamo l'immagine sull'immagine segnaposto. Successivamente scarichiamo l'immagine con il metodo di `request` di Alamofire . Passiamo l'url come primo argomento e dal momento che vogliamo solo ottenere l'immagine useremo il metodo HTTP `.get` . Dal momento che stiamo scaricando un'immagine vogliamo che la risposta sia un'immagine, quindi usiamo il metodo `.responseImage` .

Dopo che l'immagine è stata scaricata, la chiusura viene chiamata e prima di tutto ci assicuriamo che l'immagine scaricata esista effettivamente. Quindi ci assicuriamo che la cella sia ancora visibile controllando che cellForRow (a: indexPath) non restituisca nulla. Se non succede nulla, se non lo facciamo assegniamo l'immagine scaricata di recente.

Quest'ultima istruzione if garantisce che la cella sia ancora visibile se l'utente ha già fatto scorrere la cella, l'updateCell sarà nullo e l'istruzione if restituirà zero. Questo ci aiuta a prevenire la visualizzazione dell'immagine sbagliata in una cella.

Leggi Carica immagini asincrone online: https://riptutorial.com/it/ios/topic/10793/carica-immagini-asincrone

Capitolo 29: CAShapeLayer

Sintassi

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

Osservazioni

La classe CAShapeLayer disegna una spline Bezier cubica nel suo spazio di coordinate. La forma è composta tra il contenuto del livello e il suo primo sottolivello.

Examples

Funzionamento base CAShapeLayer

UIBezierPath che utilizza per creare un percorso circolare 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
```

Rimuovi ShapeLayer

Mantieni un riferimento a quel livello forma. Ad esempio, potresti avere una proprietà currentShapeLayer: Ora che hai un riferimento, puoi rimuovere facilmente il livello:

Tipo 1:

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

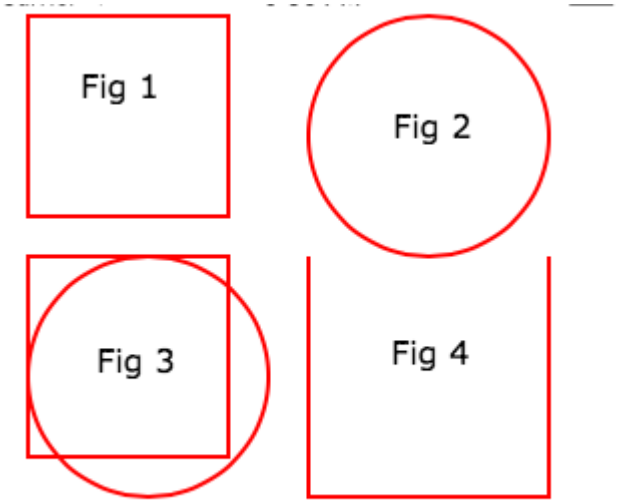
Tipo 2:

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

Altra operazione

```
 //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)];
```

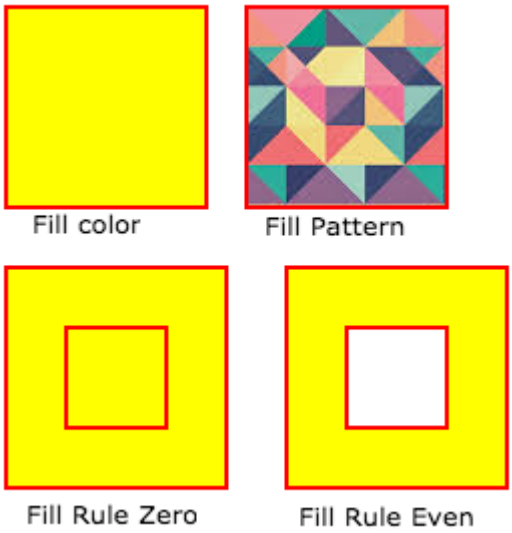
```
[linePath addLineToPoint:CGPointMake(120 , 0)];
shapeLayer.path = linePath.CGPath;
[[self.view layer] addSublayer:shapeLayer];
```



Riempi i concetti // Riempi colore

```
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];
```



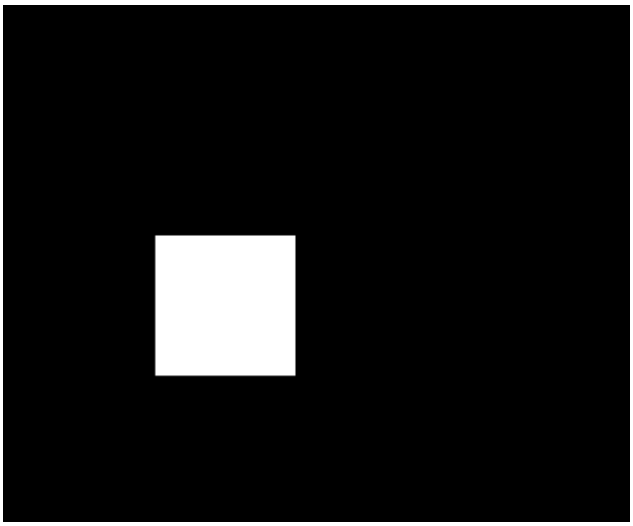
Elencato le proprietà di stile di accesso

```
   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.
```

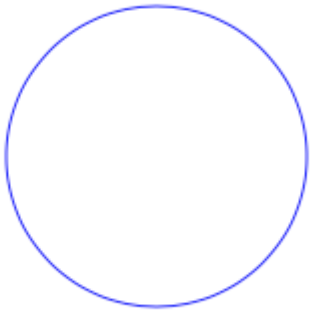
Disegna rettangolo

```
 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;
```



Disegna il cerchio

```
 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];
```



Animazione CAShapeLayer

```
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"];
```

Leggi CAShapeLayer online: https://riptutorial.com/it/ios/topic/3575/cashapelayer

Capitolo 30: categorie

Osservazioni

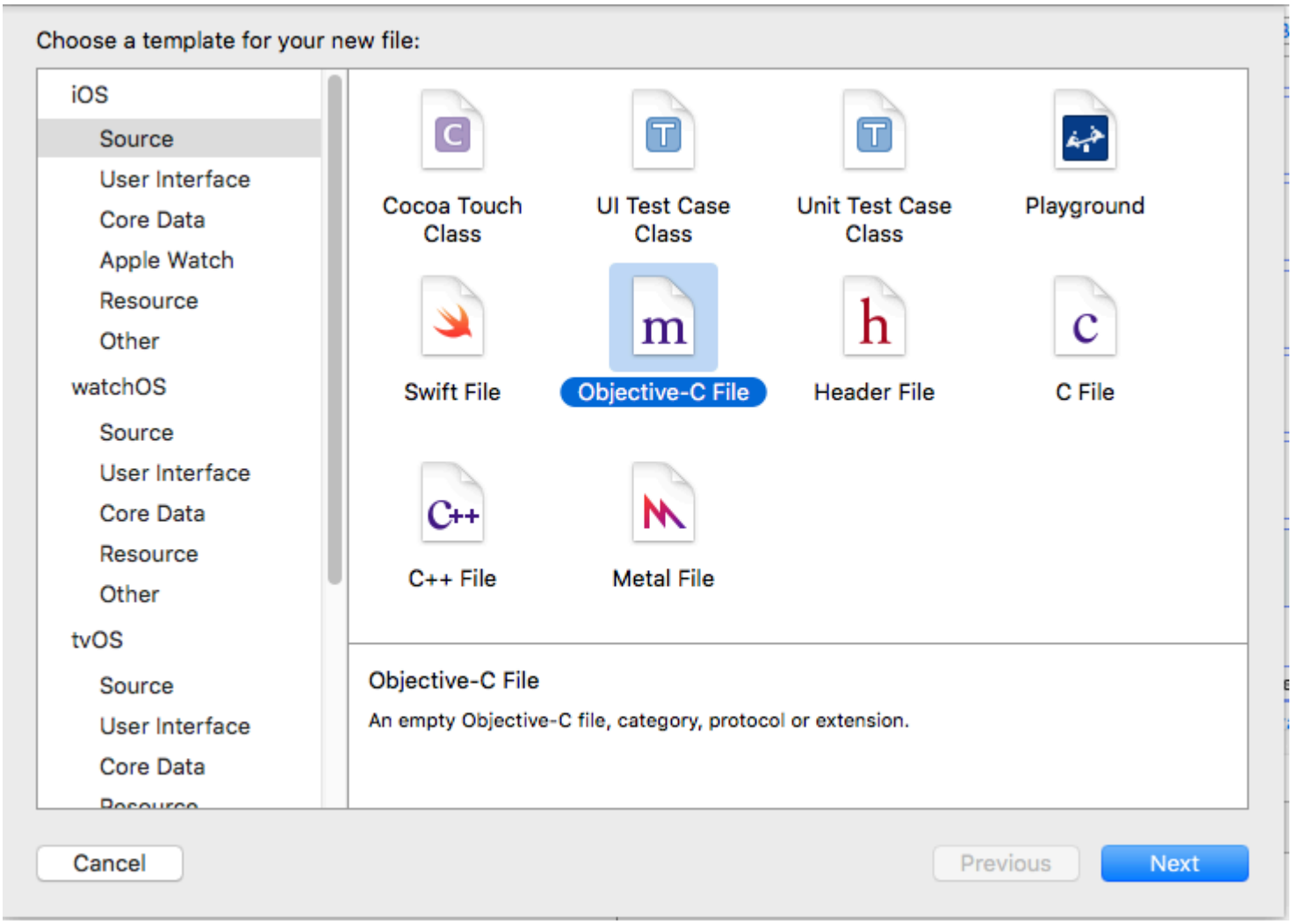
Le categorie possono essere utilizzate per sovrascrivere i metodi di una classe. Anche se il metodo è in realtà privato. Non è possibile accedere al metodo sottoposto a override dalla categoria o da qualsiasi altra parte. Quindi è importante assicurarsi che quando si aggiungono metodi a una classe esistente, tali metodi non esistano già.

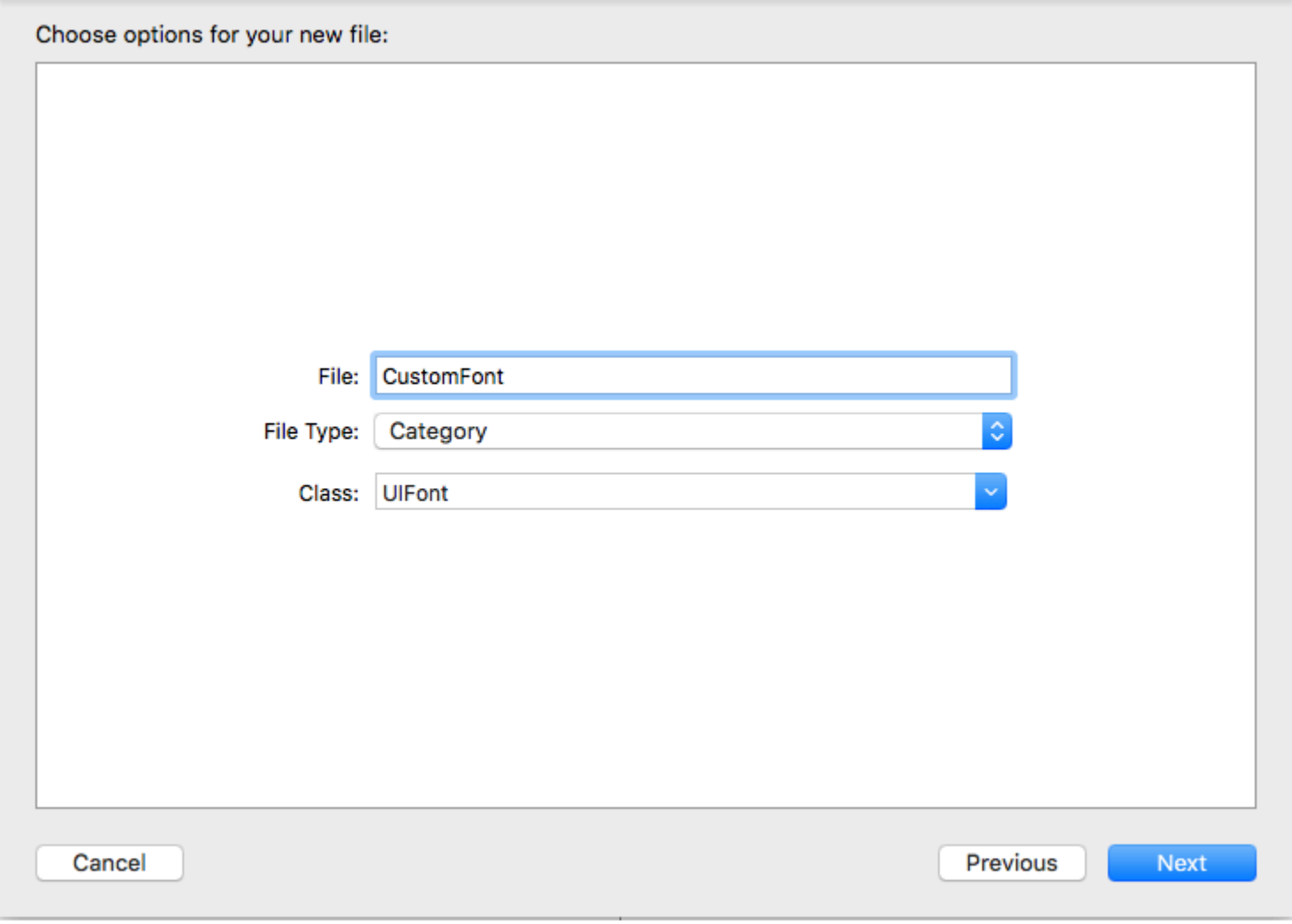
Examples

Crea una categoria

Le categorie offrono la possibilità di aggiungere alcune funzionalità extra a un oggetto senza creare sottoclassi o modificare l'oggetto reale.

Ad esempio vogliamo impostare alcuni caratteri personalizzati. Consente di creare una categoria che aggiunge funzionalità alla classe `UIFont` . Apri il tuo progetto Xcode, fai clic su File -> Nuovo -> File e scegli il file Objective-C, fai clic su Avanti inserisci il nome della tua categoria dì "CustomFont" scegli il tipo di file come Categoria e Classe come UIFont, quindi fai clic su "Avanti" seguito da "Crea". "


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



Dichiarare il metodo di categoria: -

Fai clic su "UIFont + CustomFonts.h" per visualizzare il file di intestazione della nuova categoria. Aggiungere il seguente codice all'interfaccia per dichiarare il metodo.

```
@interface UIFont (CustomFonts)

+(UIFont *)productSansRegularFontWithSize:(CGFloat)size;

@end
```

Ora implementare il metodo di categoria: -

Fai clic su "UIFont + CustomFonts.m" per visualizzare il file di implementazione della categoria. Aggiungere il seguente codice per creare un metodo che imposterà il carattere ProductSansRegular.

```
+(UIFont *)productSansRegularFontWithSize:(CGFloat)size{

    return [UIFont fontWithName:@"ProductSans-Regular" size:size];

}
```

Importa la tua categoria

```
#import "UIFont+CustomFonts.h"
```

Ora imposta il font Label

```
[self.label setFont:[UIFont productSansRegularFontWithSize:16.0]];
```

Leggi categorie online: https://riptutorial.com/it/ios/topic/3633/categorie

Capitolo 31: Chain Blocks in una coda (con MKBlockQueue)

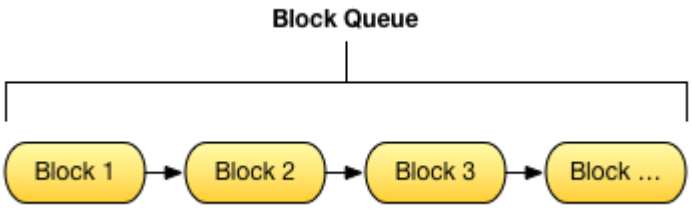
introduzione

MKBlockQueue ti permette di creare una catena di blocchi ed eseguirli uno dopo l'altro in una coda. Rispetto a NSOperation, con MKBlockQueue ti decidi quando un blocco è completo e quando vuoi che la coda continui. Puoi anche passare i dati da un blocco a quello successivo.

https://github.com/MKGitHub/MKBlockQueue

Examples

Codice di esempio




```
// 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)
```

Leggi Chain Blocks in una coda (con MKBlockQueue) online:
https://riptutorial.com/it/ios/topic/9122/chain-blocks-in-una-coda--con-mkblockqueue-

Capitolo 32: Classi di dimensioni e adattabilità

Osservazioni

Mentre crei app adattive, tieni a mente i limiti delle classi di dimensioni: sono *generalizzazioni* , non guide specifiche per dimensioni o dispositivi di pixel esatti. Non tentare mai di determinare su quale dispositivo è in esecuzione la tua app o se è in modalità a schermo diviso, in base alle classi di dimensioni.

Invece, prendi decisioni di layout di alto livello sulla classe di dimensioni e usa Layout automatico per cambiare i fotogrammi di una vista precisa. (Vedi anche il metodo UIViewController `viewWillTransition(to:with:)` per una notifica più precisa di quanto grande sarà la vista di un controller dopo una transizione.)

Examples

Collezioni di tratti

In un'app iOS, l'interfaccia utente può assumere una delle diverse forme e dimensioni generali. Questi sono definiti utilizzando le **classi di dimensioni** , che sono disponibili attraverso una **collezione di tratti di** visualizzazione o vista del controllore.

Apple definisce due classi di dimensioni: **regolari** e **compatte** . Ciascuna di queste classi dimensionali è disponibile su entrambi gli assi del dispositivo (**orizzontale** e **verticale**). La tua app potrebbe esistere in questi quattro stati per tutta la sua durata. Come una scorciatoia, gli sviluppatori spesso descrivono una combinazione di classi di dimensioni dicendo o scrivendo le due classi di dimensioni, con l'asse orizzontale per primo: "Compatto / Regolare" descrive un'interfaccia che è orizzontale compatta ma verticale regolare.

Nella tua app, utilizza i metodi sul protocollo UITraitEnvironment per verificare la tua classe di dimensioni corrente e rispondere alle modifiche:

```
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.")
    }
}
```

Sia UIView che UIViewController sono conformi a UITraitEnvironment, quindi puoi guardare la tua

attuale raccolta di tratti e gestire le modifiche in sottoclassi di entrambi.

Aggiornamento automatico del layout con le modifiche alle raccolte di tratti

Apportare un'app **adattiva** , ovvero rispondere alle modifiche delle classi di dimensioni modificando il layout, spesso richiede molto aiuto dal sistema di layout automatico. Uno dei modi principali in cui le app diventano adattive è l'aggiornamento dei vincoli del layout automatico attivo quando la classe di dimensioni di una vista cambia.

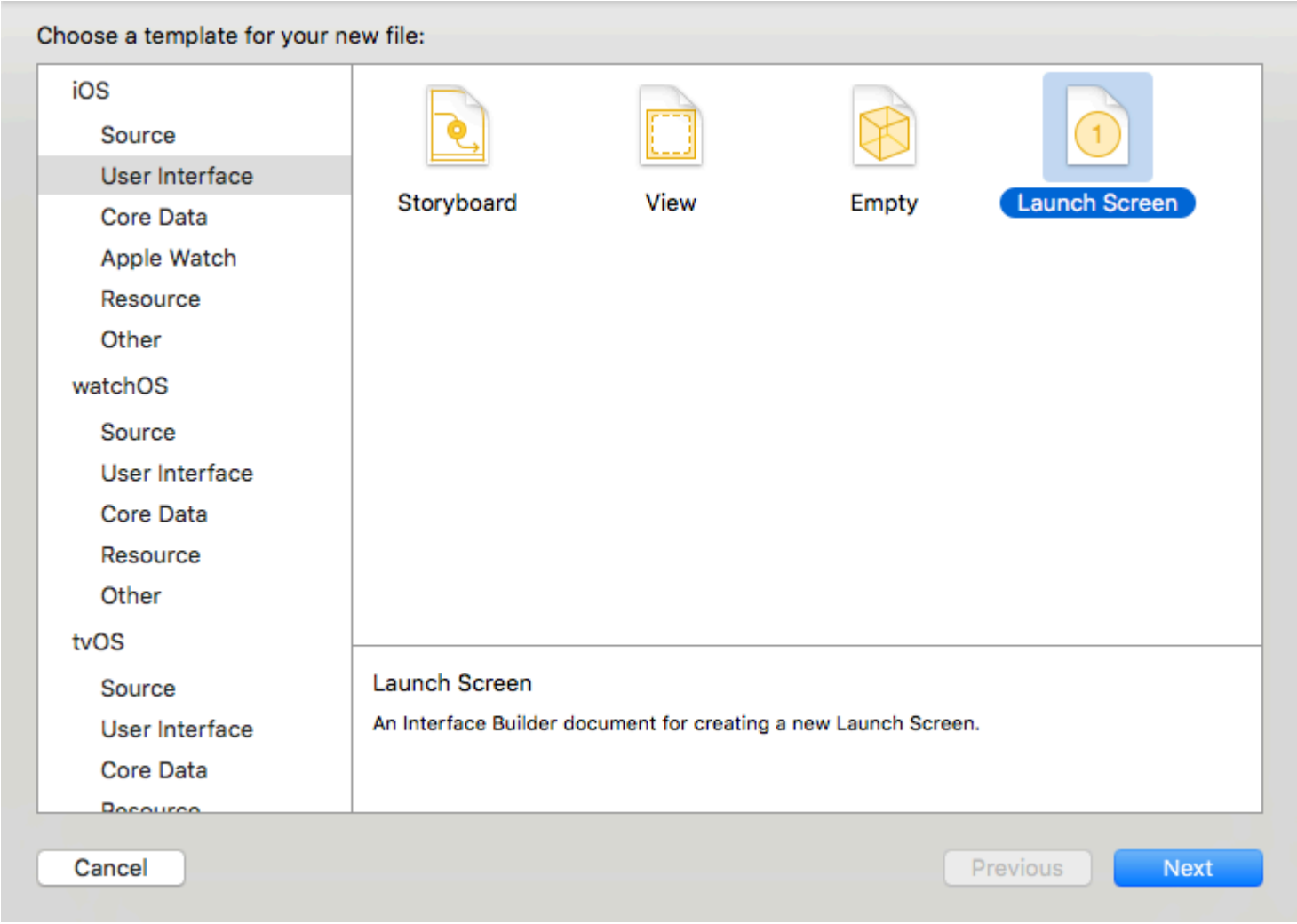
Ad esempio, considera un'app che utilizza UIStackView per organizzare due UILabels. Potremmo desiderare che queste etichette si sovrappongano l'una all'altra in ambienti orizzontali compatti, ma si siedono uno accanto all'altro quando abbiamo un po 'più di spazio in ambienti orizzontali regolari.

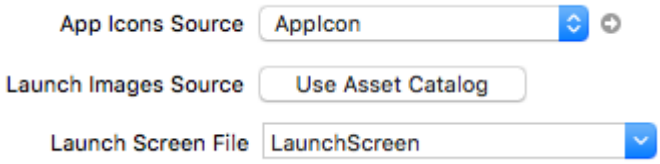
```
class ViewController: UIViewController {
    var stackView: UIStackView!

    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
        }
    }
}
```

Supporto di iOS Multitasking su iPad

Un elemento chiave di adattamento in una moderna app iOS è il supporto del multitasking su iPad. Per impostazione predefinita, le app create in Xcode 7 e versioni successive saranno configurate per supportare il multitasking: avranno un file LaunchScreen.storyboard che utilizza Auto Layout.

Il modo più semplice per le app esistenti di optare per il multitasking è creare uno storyboard di questo tipo, quindi impostarlo come schermata di avvio del progetto:





Una volta che l'app supporta il multitasking di iPad, controlla le viste esistenti e visualizza i controller per assicurarti che utilizzino il layout automatico e supportino una varietà di combinazioni di classi di dimensioni.

Leggi Classi di dimensioni e adattabilità online: https://riptutorial.com/it/ios/topic/4628/classi-di-dimensioni-e-adattabilita

Capitolo 33: Classi di dimensioni e adattabilità

Osservazioni

Per ulteriori dettagli (Classi di dimensioni e Adattabilità tramite Storyboard) dell'utilizzo del layout automatico per l'adattabilità in iOS, possiamo seguire il link al sito dello sviluppatore Apple .

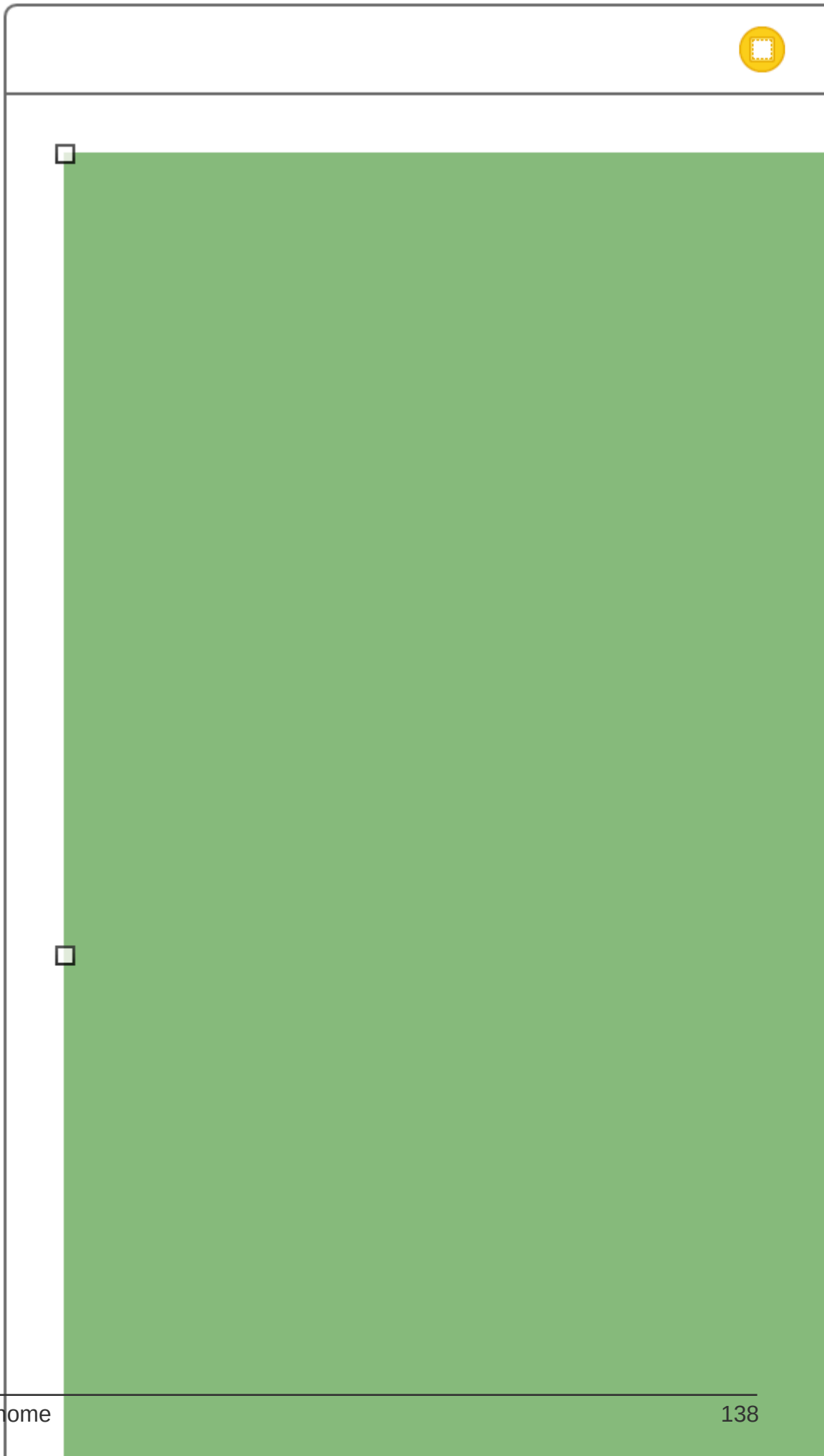
Possiamo anche aggiungere vincoli **Programatically** che utilizzano **Visual Formato Lingua** come descritto qui a sito degli sviluppatori di Apple .

Examples

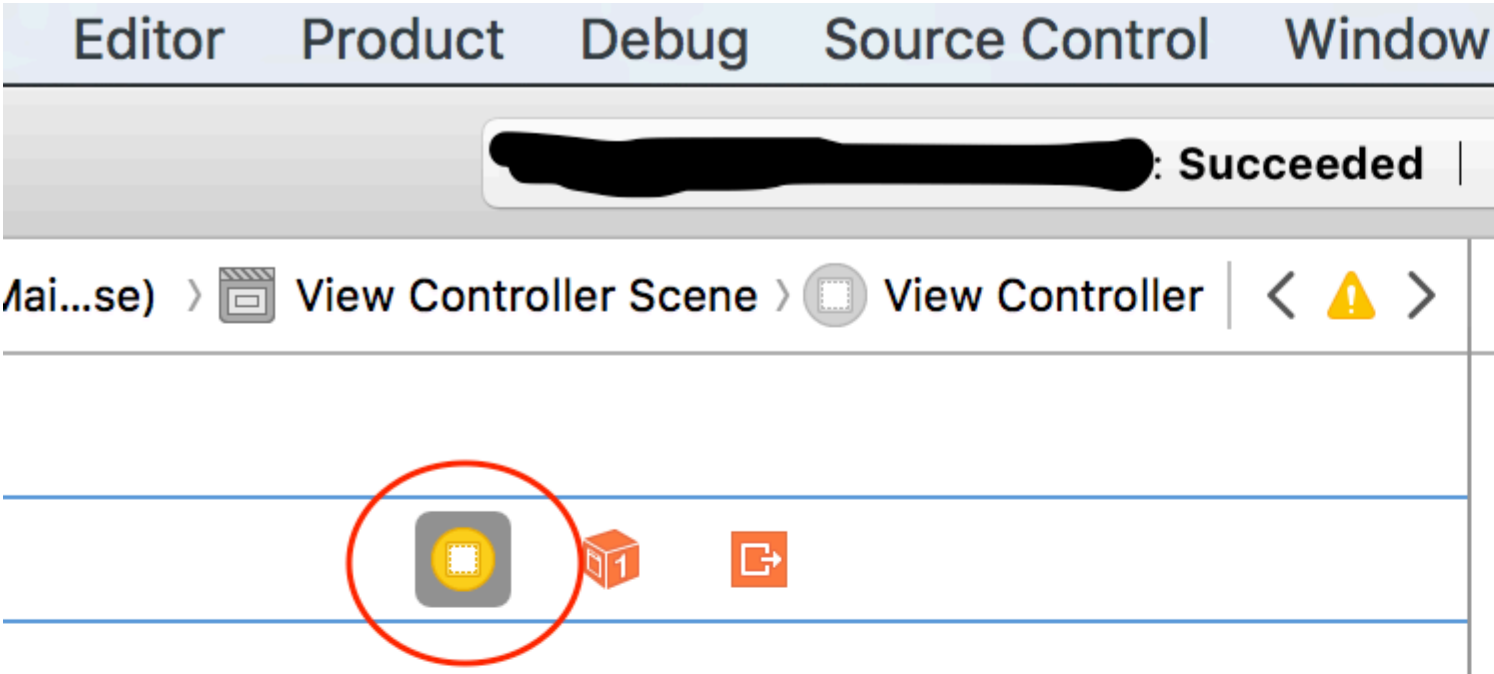
Classi di dimensioni e adattabilità attraverso Storyboard

Possiamo aggiungere l'adattabilità a qualsiasi sottoclasse di `UIView` che aggiungiamo sul controller di visualizzazione nel file di pennino.
Prendiamo un esempio di aggiunta di adattività utilizzando le classi di dimensioni per una vista.

1. Aggiungi una vista sul controller della vista come:



Assistente Editor come;


Editor
Product
Debug
Source Control
Window
[redacted]: Succeeded
Mai...se)
View Controller Scene
View Controller

https://riptutorial.com/it/ios/topic/6424/classi-di-dimensioni-e-adattabilita

Capitolo 34: Classifiche di GameCenter

Examples

Classifiche di GameCenter

Prerequisiti:

1. Account per sviluppatori Apple
2. Imposta le classifiche GameCenter con iTunesConnect

Impostazione delle classifiche di GameCenter:

1. Accedi a *iTunesConnect*
2. Vai a Le *mie app* . Crea un'app per il tuo progetto quindi vai a *Funzionalità* .
3. Clicca su *Game Center*
4. Fai clic sul segno più accanto a Classifiche.
5. Scegli *Single Leaderboard* per i tipi di classifica.
6. Crea un *nome di riferimento della classifica* per il tuo riferimento.
7. Crea un *ID classifica* per la tua app a cui fare riferimento quando segnali i punteggi.
8. Imposta il formato del punteggio su *Integer*
9. L'invio del punteggio sarà il *miglior punteggio*
10. Fai clic su *Aggiungi lingua* e riempi le voci.

Copia il tuo `LeaderboardID` che hai creato e passa a Xcode.

Lavorare con Xcode

Ci sono 4 funzioni con le quali lavoreremo.

1. Importazione del framework e impostazione dei protocolli

2. Verifica se l'utente ha effettuato l'accesso a GameCenter

3. Segnalazione dei punteggi a GameCenter

4. Visualizzazione delle classifiche

5. Importa `import GameKit` Protocolli `GKGameCenterControllerDelegate`

6. Ora vogliamo verificare se l'utente ha effettuato l'accesso a 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. Ora l'utente sta utilizzando l'app e improvvisamente l'utente ha un nuovo punteggio elevato, riportiamo il punteggio più alto chiamando la funzione seguente.

La funzione sotto hols 2 parametri.

`Identifier` che è definito come una stringa e utilizzato per inserire il leaderboardID che hai creato in iTunesConnect.

`score` che è definito come Int che sarà il punteggio degli utenti da inviare a 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. Ora se l'utente vuole vedere le classifiche, chiama la funzione qui sotto

```
//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()

    }
```

Leggi Classifiche di GameCenter online: https://riptutorial.com/it/ios/topic/6720/classifiche-di-gamecenter

Capitolo 35: CLLocation

Examples

Filtro a distanza usando

Esempio :

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

Ad esempio, nel codice di esempio sopra riportato, le modifiche di posizione inferiori a 5 metri non verranno inviate al callback, ma saranno ignorate.

Ottieni posizione utente con CLLocationManager

1 - Includi il CoreLocation.framework nel tuo progetto; questo si ottiene facendo clic su:

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

Fare clic sul pulsante (+), cercare CoreLocation.framework e fare clic su Aggiungi.

2- Modificare il file info.plist per chiedere il permesso di utilizzare la posizione dell'utente aprendolo come codice sorgente. Aggiungi una delle seguenti chiavi: coppia di valori sotto il tag per chiedere l'utilizzo della posizione dell'utente mentre l'applicazione è in uso:

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

3- importare CoreLocation sul ViewController che lo utilizzerà.

```
import CoreLocation
```

4- Assicurati che ViewController sia conforme al protocollo CLLocationManagerDelagate

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

Dopo questi passaggi, possiamo creare un oggetto CLLocationManager come variabile di istanza e utilizzarlo nel ViewController.

```
var manager:CLLocationManager!
```

Non usiamo 'let' qui perché modificheremo il gestore per specificare il suo delegato, la distanza minima prima dell'evento di aggiornamento e la sua precisione

```
//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()

  }
```

Ora per ottenere l'accesso agli aggiornamenti di posizione, possiamo implementare la funzione di sotto che è chiamata tempo straordinario in cui viene raggiunto distanceFilter.

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

Il parametro locations è una matrice di oggetti CLLocation che rappresentano la posizione effettiva del dispositivo. Da questi oggetti, è possibile accedere ai seguenti attributi: `coordinate`,`altitude`, `floor`, `horizontalAccuracy`, `verticalAccuracy`, `timestamp`, `description`, `course`, `speed` e una `distance(from:)` funzione `distance(from:)` che misura la distanza tra due posizioni.

Nota: mentre si richiede l'autorizzazione per la posizione, esistono due diversi tipi di autorizzazione.

> L'autorizzazione "Quando in uso" autorizza l'app a ricevere la tua posizione solo quando l'app è in uso o in primo piano.
>
> Autorizzazione "sempre", fornisce le autorizzazioni in background dell'app che possono portare a ridurre la durata della batteria nel caso in cui l'app sia chiusa.
>
> Il file Plist deve essere regolato secondo necessità.

Leggi CLLocation online: https://riptutorial.com/it/ios/topic/2002/cllocation

Capitolo 36: CloudKit

Osservazioni

Tipi supportati

- NSData
- NSDate (data)
- NSNumber (Int / Double)
- NSString (String)
- NSArray (matrice)
- CLLocation
- CKReference
- CKAsset

Più dettagli

CloudKit Dashboard

Examples

Registrazione dell'app da utilizzare con CloudKit

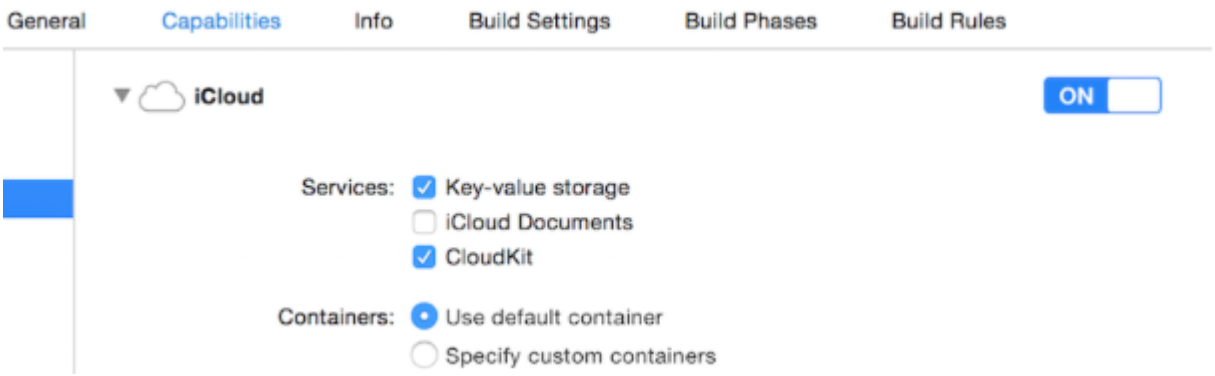
Quello di cui hai bisogno è ottenere un file di diritti in modo che l'app possa accedere a iCloud e scrivere record usando CloudKit.

Segui i passaggi per concedere l'accesso a iCloud dalla tua app:

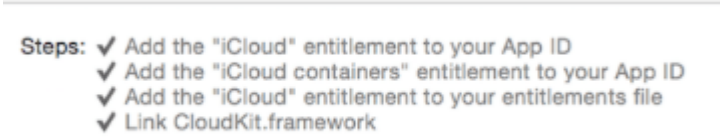
1- Seleziona il progetto nel Navigatore progetto, quindi apri la scheda Generale.

2- Nella sezione Identità, imposta l'ID Apple dello sviluppatore nel menu a discesa Team. (Se non è disponibile, aggiungilo nel menu Xcode -> Preferenze -> Account.

3- Vai alla scheda Capacità nelle proprietà del progetto e attiva iCloud. Quindi, seleziona "Memorizzazione valore-chiave" e "CloudKit".



4- Assicurati che questi elementi siano spuntati:



Se tutti gli elementi sono selezionati, la tua app è pronta per l'uso di CloudKit.

Utilizzo di CloudKit Dashboard

Tutti i record creati utilizzando il codice relativo a CloudKit possono essere visualizzati in anteprima, modificati e persino rimossi in CloudKit Dashboard. Per accedere a CloudKit Dashboard, vai qui .

Ci sono diverse parti nel dashboard:

- Tipi di record (che saranno discussi in seguito)
- Ruoli di sicurezza (in cui è possibile impostare i database come pubblici o privati)
- Tipi di abbonamento (che la tua app potrebbe registrare per le notifiche push Apple (APN) per avvisarti quando viene modificato un record)

Registra i tipi

Qui, si ottiene un elenco di tutti i tipi di record esistenti nell'app. Quando apri CloudKit Dashboard per la prima volta su un'app, c'è un tipo di record chiamato Utenti lì, che puoi utilizzare o semplicemente cancellarlo e utilizzarne uno tuo.

In questa pagina è possibile inserire manualmente i dati digitati. Naturalmente, nella maggior parte dei casi questo non ha senso, perché iOS SDK può gestirlo in modo migliore rispetto al dashboard, ma la funzionalità è anche presente se preferisci. Il maggior uso di questa pagina è per l'anteprima dei tipi.

Salvataggio dei dati su CloudKit

Per salvare la data su CloudKit, dobbiamo fare:

- Un `CKRecordID` (la chiave del tuo record unico)
- Un `CKRecord` (che include dati)

Fare una chiave di registrazione

Per garantire che ogni nuovo identificatore di record sia unico, utilizziamo il *timestamp* attuale, che è unico. Otteniamo il timestamp usando `NSDate` metodo s' `timeIntervalSinceReferenceDate()` . È in forma di ###. ### (# sono numeri), che useremo la parte intera. Per fare ciò, dividiamo la stringa:

veloce

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

Fare il record

Per rendere il record, dovremmo specificare il tipo di record (spiegato in Utilizzo di CloudKit Dashboard) come Utenti, l'ID come la cosa che abbiamo appena creato e i dati. Qui, aggiungeremo un testo di esempio, un'immagine e la data corrente al record:

veloce

```
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"];
```

> ## Nota
>
> Qui, non abbiamo aggiunto `UIImage` direttamente al record, perché come menzionato in Note, il formato dell'immagine non è direttamente supportato in CloudKit, quindi abbiamo convertito `UIImage` in `CKAsset` .

Accesso al contenitore

veloce

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

Salvataggio dei record nel database CloudKit

veloce

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

Leggi CloudKit online: https://riptutorial.com/it/ios/topic/4946/cloudkit

Capitolo 37: codificabile

introduzione

Il codice è aggiunto con Xcode 9, iOS 11 e Swift 4. Codable viene utilizzato per rendere i tuoi tipi di dati codificabili e decodificabili per la compatibilità con rappresentazioni esterne come JSON.

Uso codificabile per supportare sia la codifica che la decodifica, dichiarare la conformità a Codable, che combina i protocolli Encodable e Decodable. Questo processo è noto per rendere i tuoi tipi codificabili.

Examples

Utilizzo di Codifica con JSONEncoder e JSONDecoder in Swift 4

Prendiamo un esempio con Structure of Movie, qui abbiamo definito la struttura come Codable. Quindi, possiamo codificarlo e decodificarlo facilmente.

```
struct Movie: Codable {
    enum MovieGenere: String, Codable {
        case horror, skifi, comedy, adventure, animation
    }

    var name : String
    var moviesGenere : [MovieGenere]
    var rating : Int
}
```

Possiamo creare un oggetto dal film come:

```
let upMovie = Movie(name: "Up", moviesGenere: [.comedy , .adventure, .animation], rating : 4)
```

L'upMovie contiene il nome "Up" ed è movieGenere è una commedia, un'avventura e una strega d'animazione che contiene 4 valutazioni su 5.

Codificare

JSONEncoder è un oggetto che codifica le istanze di un tipo di dati come oggetti JSON. JSONEncoder supporta l'oggetto codificabile.

```
// 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 ci fornirà i dati JSON che vengono utilizzati per recuperare la stringa JSON.

La stringa di output sarà simile a:

```
{
  "name": "Up",
  "moviesGenere": [
    "comedy",
    "adventure",
    "animation"
  ],
  "rating": 4
}
```

Decodificare

JSONDecoder è un oggetto che decodifica le istanze di un tipo di dati dagli oggetti JSON. Possiamo recuperare l'oggetto dalla stringa 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 {
}
```

Decodificando la JSONData riceviamo indietro l'oggetto Movie. Quindi possiamo ottenere tutti i valori che vengono salvati in quell'oggetto.

L'output sarà come:

```
Name : Up
Rating : 4
```

Leggi codificabile online: https://riptutorial.com/it/ios/topic/10639/codificabile

Capitolo 38: Compressione del contenuto / compressione dei contenuti in Autolayout

Osservazioni

Contenuto Resistenza alla compressione Priorità

> Questo valore determina la resistenza di una vista a essere compresso o ridotto. Un valore più elevato indica che la vista sarà meno probabile che sia compresso e con maggiori probabilità di rimanere invariata.

Priorità di abbronzatura del contenuto

> Questo valore determina la resistenza di una vista all'espansione. Puoi immaginare "abbraccio" qui per indicare "taglia per adattarsi" - i limiti della vista si "abbracceranno" o si avvicineranno alla dimensione intrinseca del contenuto. Un valore più alto indica che la vista avrà meno probabilità di crescere e più probabilmente di rimanere la stessa.

Examples

Definizione: dimensione del contenuto intrinseco

Prima del layout automatico, dovevi sempre dire ai pulsanti e ad altri controlli quanto dovrebbero essere grandi, impostando le loro proprietà di frame o bound o ridimensionandole in Interface Builder. Ma si scopre che la maggior parte dei controlli è perfettamente in grado di determinare la quantità di spazio di cui hanno bisogno, in base al loro contenuto.

Un'etichetta sa quanto è ampia e alta perché conosce la lunghezza del testo che è stato impostato su di essa, così come la dimensione del carattere per quel testo. Allo stesso modo per un **pulsante** , che potrebbe combinare il testo con un'immagine di sfondo e un po 'di padding.

Lo stesso vale per i controlli segmentati, le barre di avanzamento e la maggior parte degli altri controlli, sebbene alcuni possano avere solo un'altezza predeterminata ma una larghezza sconosciuta.

Questo è noto come dimensione intrinseca del contenuto ed è un concetto importante in Layout automatico. Layout automatico chiede ai tuoi controlli quanto sono grandi e devono essere disposti sullo schermo in base a tali informazioni.

Di solito si desidera utilizzare la `intrinsic content size` , ma ci sono alcuni casi in cui potresti non voler farlo. È possibile evitare ciò impostando un vincolo di larghezza o altezza esplicito su un controllo.

Immagina cosa succede quando imposti un'immagine su UIImageView se quell'immagine è molto

più grande dello schermo. Di solito vuoi dare a viste di immagini una larghezza e un'altezza fissa e ridimensionare il contenuto, a meno che tu non voglia che la vista ridimensioni alle dimensioni dell'immagine.

Riferimento: https://www.raywenderlich.com/115444/auto-layout-tutorial-in-ios-9-part-2- vincoli

Leggi Compressione del contenuto / compressione dei contenuti in Autolayout online: https://riptutorial.com/it/ios/topic/6899/compressione-del-contenuto---compressione-dei-contenuti-in-autolayout

Capitolo 39: Concorrenza

introduzione

Argomento correlato: Grand Central Dispatch

Sintassi

- dispatch_async: esegue un blocco di codice in una coda separata e non interrompe la coda corrente. Se la coda è su un thread diverso da quello su cui è stato chiamato dispatch_async, il codice nel blocco verrà eseguito mentre il codice dopo l'esecuzione di dispatch_async
- dispatch_sync - Esegue un blocco di codice in una coda separata, e *si* ferma la coda corrente. Se la coda è su un thread diverso da quello su cui è stato chiamato dispatch_async, verrà eseguito il codice nel blocco e l'esecuzione sul thread in cui è stato chiamato il metodo riprenderà solo dopo il completamento

Parametri

coda	**La coda in cui verrà eseguito il codice nel blocco di invio. Una *coda* è simile (ma non esattamente uguale a) a un thread; il codice in code diverse può essere eseguito in parallelo. Usa `dispatch_get_main_queue` per ottenere la coda per il thread principale Per creare una nuova coda, che a sua volta crea un nuovo thread, usa `dispatch_queue_create("QUEUE_NAME", DISPATCH_QUEUE_CONCURRENT)` . Il primo parametro è il nome della coda, che viene visualizzato nel debugger se si mette in pausa mentre il blocco è ancora in esecuzione. Il secondo parametro non ha importanza se non si desidera utilizzare la stessa coda per più chiamate `dispatch_async` o `dispatch_sync` . Descrive cosa succede quando un altro blocco viene inserito nella stessa coda; `DISPATCH_QUEUE_CONCURRENT` farà sì che entrambi i blocchi vengano eseguiti allo stesso tempo, mentre `DISPATCH_QUEUE_SERIAL` farà in modo che il secondo blocco attenda che il primo blocco finisca**
bloccare	Il codice in questo blocco verrà eseguito nella coda della `queue` ; inserisci il codice che vuoi eseguire sulla coda separata qui. Un consiglio utile: se stai scrivendo questo in Xcode e l'argomento del blocco ha il contorno blu attorno ad esso, fai doppio clic sull'argomento e Xcode creerà automaticamente un blocco vuoto (questo vale per tutti gli argomenti dei blocchi in qualsiasi funzione o metodo)

Osservazioni

Ogni volta che fai qualcosa su un thread separato, cosa che succede quando usi le code, è importante mantenere la sicurezza dei thread. Alcuni metodi, in particolare quelli per `UIView` ,

potrebbero non funzionare e / o bloccarsi su thread diversi dal thread principale. Inoltre, assicurati di non modificare nulla (variabili, proprietà, ecc.) Che viene utilizzato anche sul thread principale, a meno che tu non stia tenendo conto di questo cambiamento

Examples

Esecuzione simultanea del codice: esecuzione di codice durante l'esecuzione di altro codice

Dite che vuoi eseguire in azione (in questo caso, registrando "Foo"), mentre fai qualcos'altro (registrando "Bar"). Normalmente, se non si utilizza la concorrenza, una di queste azioni sarà completamente eseguita e l'altra esecuzione verrà eseguita solo dopo che è stata completata. Ma con la concorrenza, puoi eseguire entrambe le azioni contemporaneamente:

```
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);
}
```

Questo registrerà "Foo" 100 volte, fermandosi per 100ms ogni volta che registra, ma farà tutto questo su un thread separato. Mentre viene registrato `Foo` , "Bar" verrà anche registrato a intervalli di 50 ms, allo stesso tempo. Dovresti idealmente vedere un output con "Foo" e "Bars" mescolati insieme

Esecuzione sul thread principale

Quando si eseguono attività in modo asincrono, in genere diventa necessario assicurarsi che una parte di codice venga eseguita sul thread principale. Ad esempio potresti voler colpire un'API REST in modo asincrono, ma inserire il risultato in una UILabel sullo schermo. Prima di aggiornare UILabel devi assicurarti che il tuo codice sia eseguito sul thread principale:

```
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 = //....
    });
}
```

Ogni volta che aggiorni le visualizzazioni sullo schermo, assicurati sempre di farlo sul thread principale, altrimenti potrebbe verificarsi un comportamento non definito.

Gruppo di spedizione - in attesa di altri thread completati.

```
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");
});
```

Aggiornamento 1. Versione 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")
}
```

Leggi Concorrenza online: https://riptutorial.com/it/ios/topic/1090/concorrenza

Capitolo 40: Configura i beacon con CoreBluetooth

introduzione

Caldo da leggere e scrivere dati su un dispositivo a basso consumo di bluetooth.

Osservazioni

Alcuni punti importanti

- Non sono necessarie capacità.
- I byte del negozio di iPhone in formato Little Endian, quindi controlla se l'accessorio Bluetooth usa anche Little Endian. Esempio:
  - CPU Intel generalmente usa little endian.
  - L'architettura ARM era little-endian prima della versione 3 quando divenne big-endian.
- Dopo un'operazione singola o batch, la connessione andrà persa, quindi è necessario riconnettersi prima di continuare.

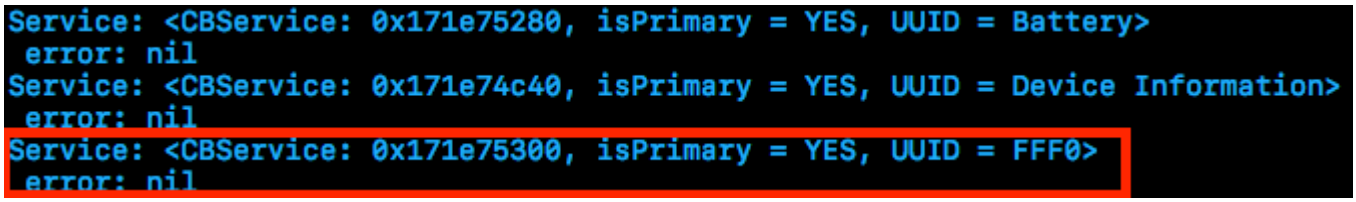
Cerca UUID SERVICE

```
func SearchBLE(){
    cb_manager.scanForPeripherals(withServices:[service_uuid], options: nil)
    StopSearchBLE()
}
```

Come scoprire SERVICE UUID senza documentazione

```
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 significa che tutti i servizi saranno restituiti, che non è una buona opzione. (LEGGI Note 3)
- Se non hai trovato l'UUID SERVICE, esegui il tuo codice e cerca nella console



- Ho trovato 3 servizi: batteria, informazioni sul dispositivo (firmware) e FFF0
- Questo servizio uuid non è uno standard, una lista con gli standard può trovare qui
- FFF0 è l'UUID di SERVIZIO in questo caso

Converti i dati in UInt16 e viceversa

Aggiungi queste estensioni alla tua classe

```
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

Visualizzazione dei nomi di tutti i Bluetooth Low Energy (BLE)

- Per questo esempio ho una stanza controllata con una sola abilitazione del dispositivo BLE.
- La tua classe dovrebbe estendere CBCentralManagerDelegate.
- Implementare il metodo: centralManagerDidUpdateState (_ centrale: CBCentralManager).
- Utilizzare la coda globale per non congelare lo schermo durante la ricerca di un dispositivo.
- Istanziare CBCentralManager e attendere la risposta di callback centralManagerDidUpdateState.

```
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)")
    }
}
```

La richiamata su centralManagerDidUpdateState indica che CoreBluetooth è pronto, quindi puoi cercare BLE ora. Aggiorna il codice centralManagerDidUpdateState per cercare tutto il dispositivo BLE quando è pronto.

```
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 () cerca dispositivi BLE e interrompe la ricerca dopo 5 secondi
- cb_manager.scanForPeripherals (withServices: nil, options: nil) cerca ogni BLE in campo con te.
- StopSearchBLE () interromperà la ricerca dopo 5 secondi.
- Ogni BLE trovato eseguirà callback func centralManager (_ central: CBCentralManager, didDiscover periferica: CBPeripheral, advertisementData: [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)
}
```

Connetti e leggi il valore maggiore

- Sono in una stanza controllata con un singolo segnale di soccorso che utilizza il protocollo IBEACON.
- BLEController deve estendere CBPeripheralDelegate
- Userò il primo BLE da connettere dopo che la ricerca si è fermata.
- Modifica il metodo 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)
        }
    }
/...
}
```

- Nel documentario del tuo dispositivo BLE, dovresti cercare l'UUID di SERVIZIO e l'UUIDO MAGGIORE CARATTERISTICO

```
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])!)
}
```

- Crea una variabile "service_uuid" e "major_uuid" come sopra. '-0000-1000-8000-00805f9b34fb' fa parte dello standard. 'fff0' è il mio UUID di SERVIZIO, 'fff2' è la mia caratteristica MAUOR UUID e '0000' sono necessari per riempire i 4 byte del blocco uuid 1º.
- discoverCharacteristics ([major_uuid], per: (peripheral.services?[0])!) otterrà le principali caratteristiche dal mio dispositivo gatt server e avrà NIL come valore per ora.
- (Peripheral.services?[0])! - 0 beacuse restituirà un singolo valore una volta eseguito periferal.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)")
}
```

- Il valore caratteristico sarà leggibile solo dopo aver chiamato peripheral.readValue (per: caratteristica)
- readValue risulterà nella periferica func (_ periferica: CBPeripheral, didUpdateValueFor caratteristica: CBCharacteristic, errore: errore?) con valore nel tipo di dati.

Scrivi un valore importante

- Hai bisogno di scoprire i servizi e le caratteristiche
- Non è necessario leggere il valore dalla caratteristica prima di scriverla.
- continuerà per, per questo esempio, dopo il valore letto. Modifica periferica func (_ periferica: CBPeripheral, didUpdateValueFor caratteristica: caratteristica CBC, errore: errore?)
- Aggiungi una variabile new_major e 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 di deafult invierà e riceverà byte in formato Little Endian, ma il mio dispositivo MINEW chipset NRF51822 ha archteture ARM e necessita di byte in formato Big Endian, quindi devo scambiarlo.
- La documentazione del dispositivo BLE dirà quale tipo di input e output avrà ciascuna caratteristica e se è possibile leggerla come sopra (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)
    }
}
```

- Per aggiornare le informazioni di un server Gatt, è necessario riavviarlo a livello di codice o salvare i dati su di esso e spegnere e accendere manualmente.
- FFFF è caratteristico che lo fa in questo dispositivo.
- 'minew123' è la password predefinita per il riavvio o salva le informazioni in questo caso.
- esegui la tua app e tieni sotto controllo la console per qualsiasi errore, spero nessuno, ma non vedrai ancora il nuovo valore.

```
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)
```

}

- L'ultimo passaggio consiste nel commentare l'ultima riga nel metodo didUpdateValueFor e rieseguire l'app, ora si avrà il nuovo valore.

Leggi Configura i beacon con CoreBluetooth online:
https://riptutorial.com/it/ios/topic/9488/configura-i-beacon-con-corebluetooth

Capitolo 41: Controllo della versione di iOS

Examples

iOS 8 e versioni successive

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)
}
```

Confronta le versioni

```
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 e versioni successive

```
if #available(iOS 9, *) {
```

```
  // iOS 9
} else {
  // iOS 8 or earlier
}
```

Versione iOS del dispositivo

Questo darà la versione attuale del sistema.

Objective-C

```
NSString *version = [[UIDevice currentDevice] systemVersion]
```

veloce

```
let version = UIDevice.currentDevice().systemVersion
```

Swift 3

```
let version = UIDevice.current.systemVersion
```

Leggi Controllo della versione di iOS online: https://riptutorial.com/it/ios/topic/2194/controllo-della-versione-di-ios

Capitolo 42: Converti HTML in stringa NSAtribuita e viceversa

Examples

Codice oggettivo C per convertire la stringa HTML in NSAttributedString e Vice Versa

Codice di conversione da HTML a NSAttributoString: -

```
 //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 ];
```

Conversione NSAattributedString in HTML: -

```
 //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];
```

Leggi Converti HTML in stringa NSAtribuita e viceversa online:
https://riptutorial.com/it/ios/topic/7225/converti-html-in-stringa-nsatribuita-e-viceversa

Capitolo 43: Converti NSAttributedString in UIImage

Examples

NSAttributoString a 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];
```

La funzione `imageFromAttributedString` è come definito di seguito:

```
- (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;
}
```

Leggi Converti NSAttributedString in UIImage online:
https://riptutorial.com/it/ios/topic/7242/converti-nsattributedstring-in-uiimage

Capitolo 44: Core Graphics

Examples

Creazione di un contesto di grafica principale

Contesto grafico core

Un contesto Core Graphics è una tela che possiamo disegnare in essa e impostare alcune proprietà come lo spessore della linea.

Fare un contesto

Per creare un contesto, usiamo la funzione C di `UIGraphicsBeginImageContextWithOptions()` . Quindi, quando abbiamo finito con il disegno, chiamiamo semplicemente `UIGraphicsEndImageContext()` per terminare il contesto:

veloce

```
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();
```

Nel codice sopra, abbiamo passato 3 parametri alla funzione `UIGraphicsBeginImageContextWithOptions()` :

1. Un oggetto `CGSize` che memorizza l'intera dimensione del contesto (la tela)

2. Un valore booleano che, se è vero, il contesto sarà opaco

3. Un valore intero che imposta la scala (1 per non retina, 2 per retina e 3 per retina HD schermate). Se impostato su 0, il sistema gestisce automaticamente la scala in base al dispositivo di destinazione.

Presentazione della tela disegnata all'utente

veloce

```
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
```

Leggi Core Graphics online: https://riptutorial.com/it/ios/topic/5530/core-graphics

Capitolo 45: Core Location

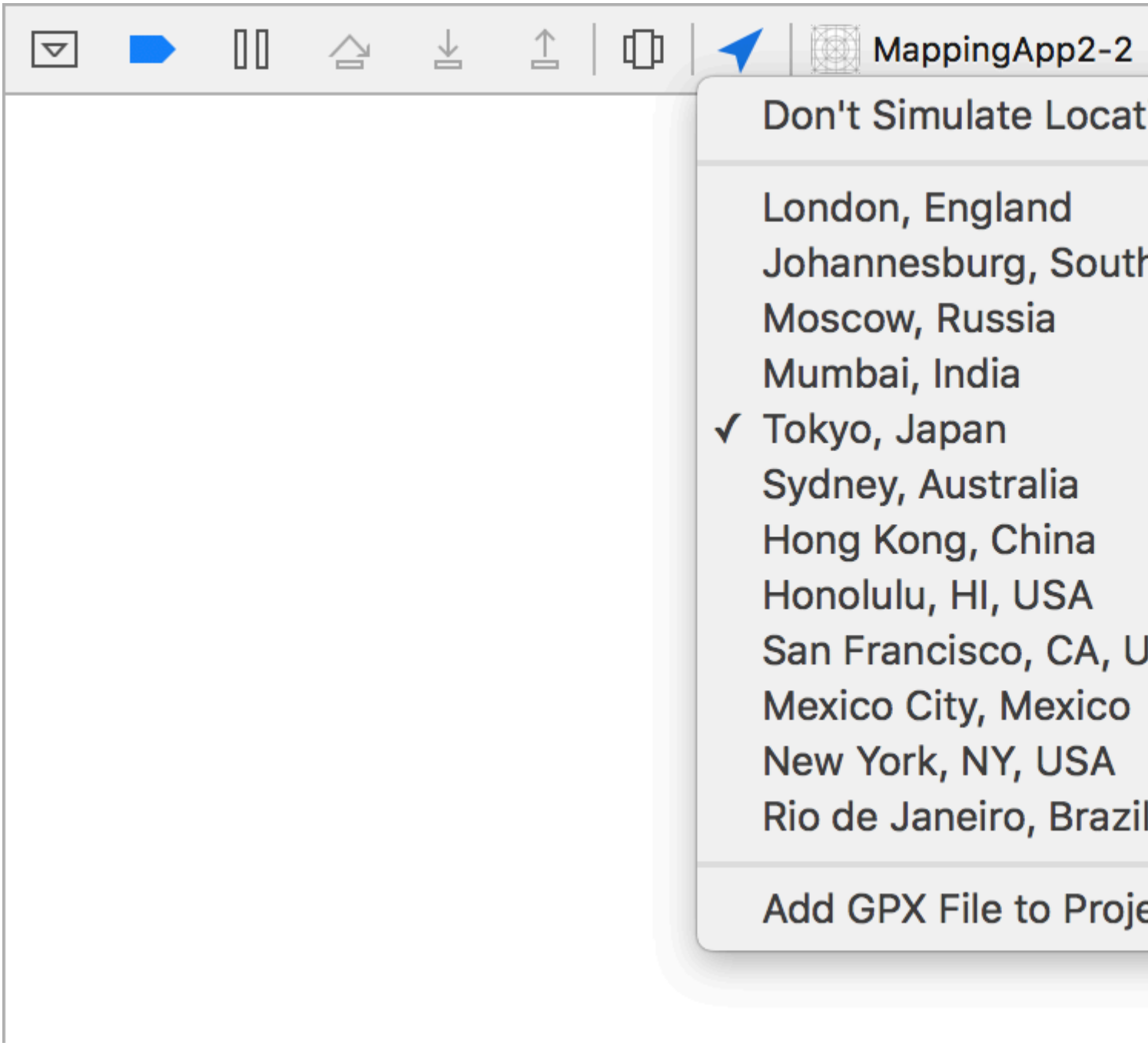
Sintassi

1. desiredAccuracy
2. distanceFilter
3. requestLocation ()
4. startUpdatingLocation ()
5. allowDeferredLocationUpdates (untilTraveled: timeout :)
6. startMonitoringSignificantLocationChanges ()
7. allowDeferredLocationUpdates (untilTraveled: timeout :)
8. authorizedAlways
9. authorizedWhenInUse
10. locationManager (_: didChangeAuthorization :)

Osservazioni

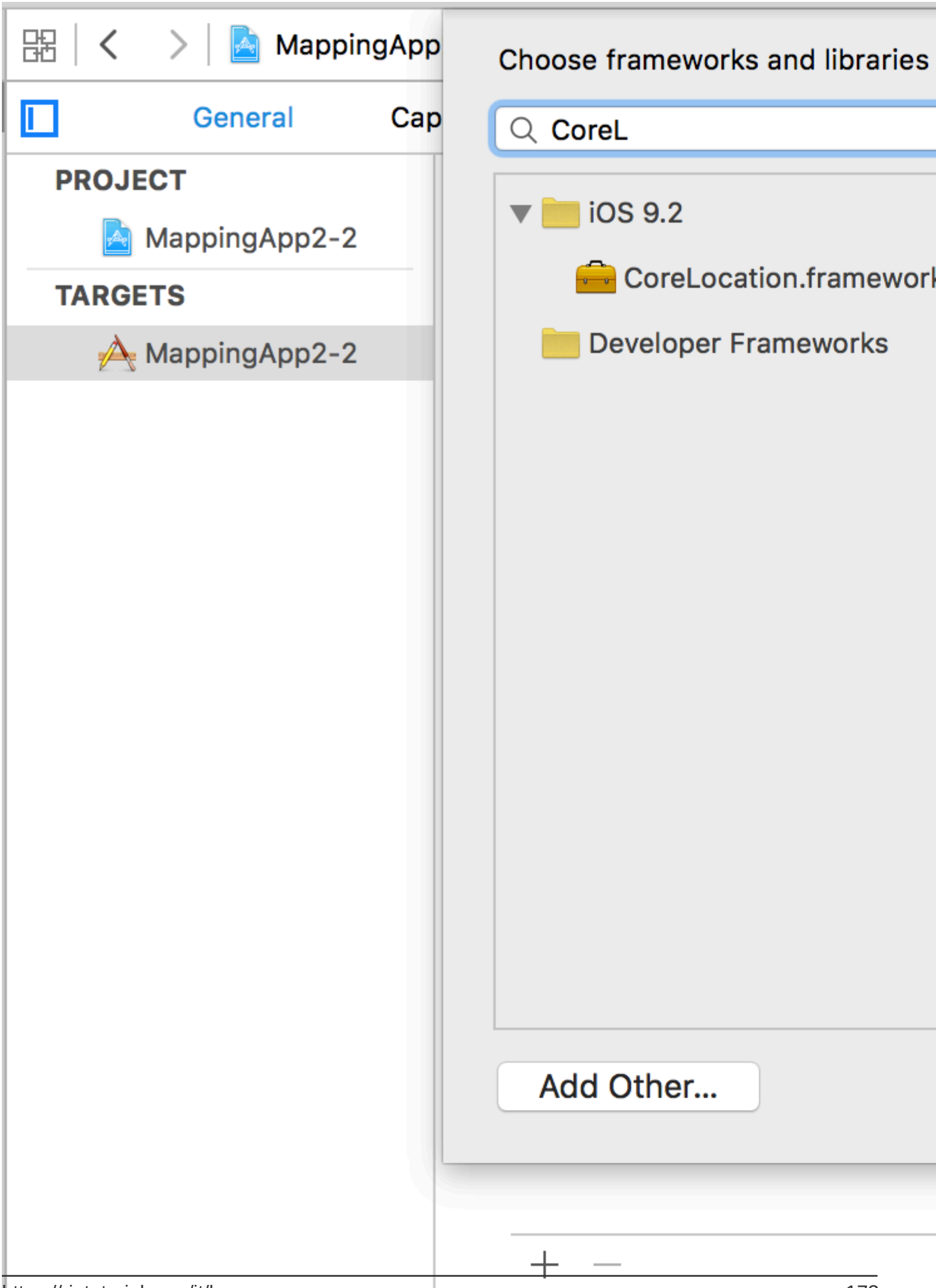
Simula una posizione in fase di esecuzione

1. Esegui l'app da Xcode.
2. Nella barra di debug, fai clic sul pulsante "Simula posizione".
3. Scegli una posizione dal menu.

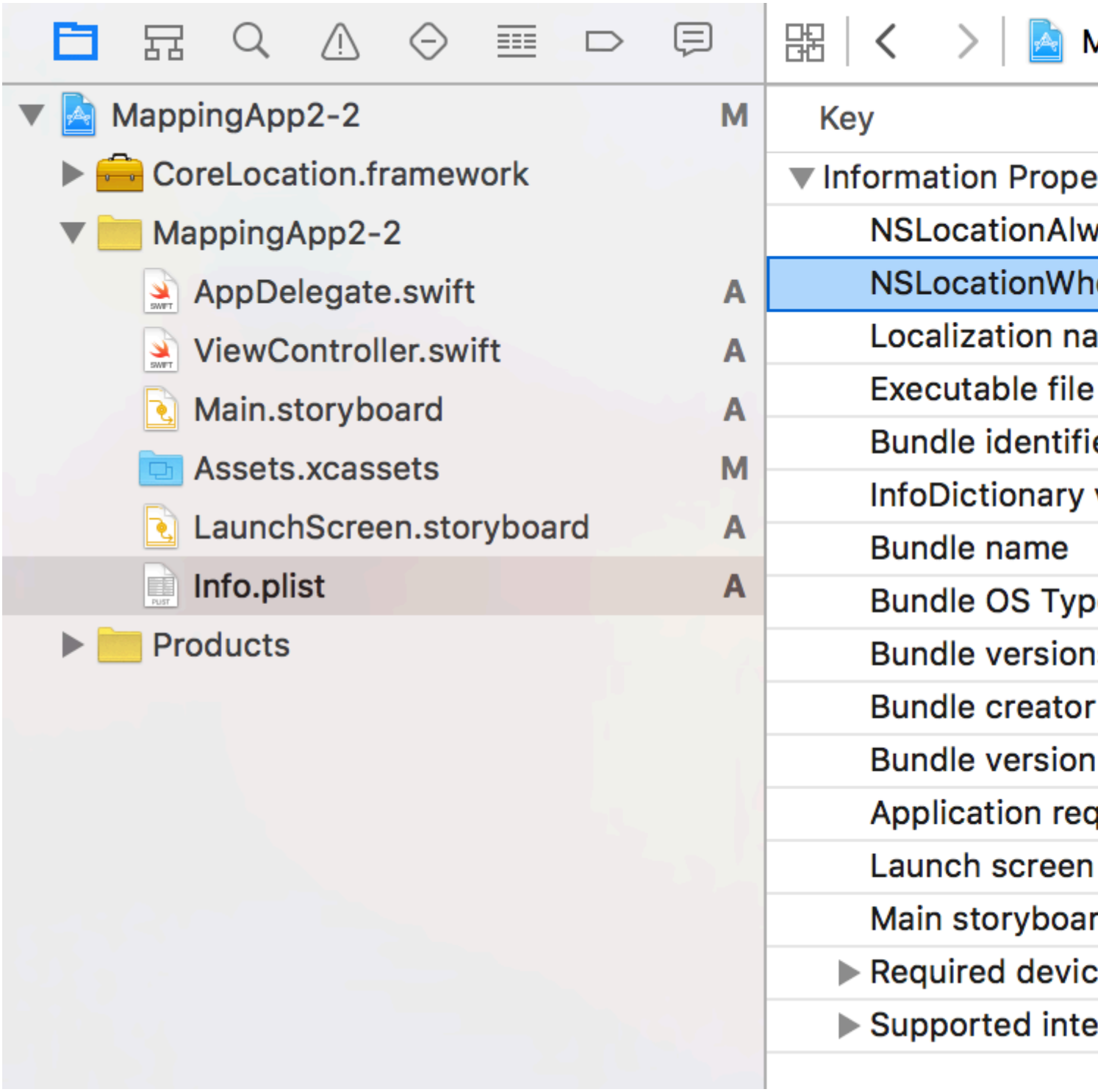


Examples

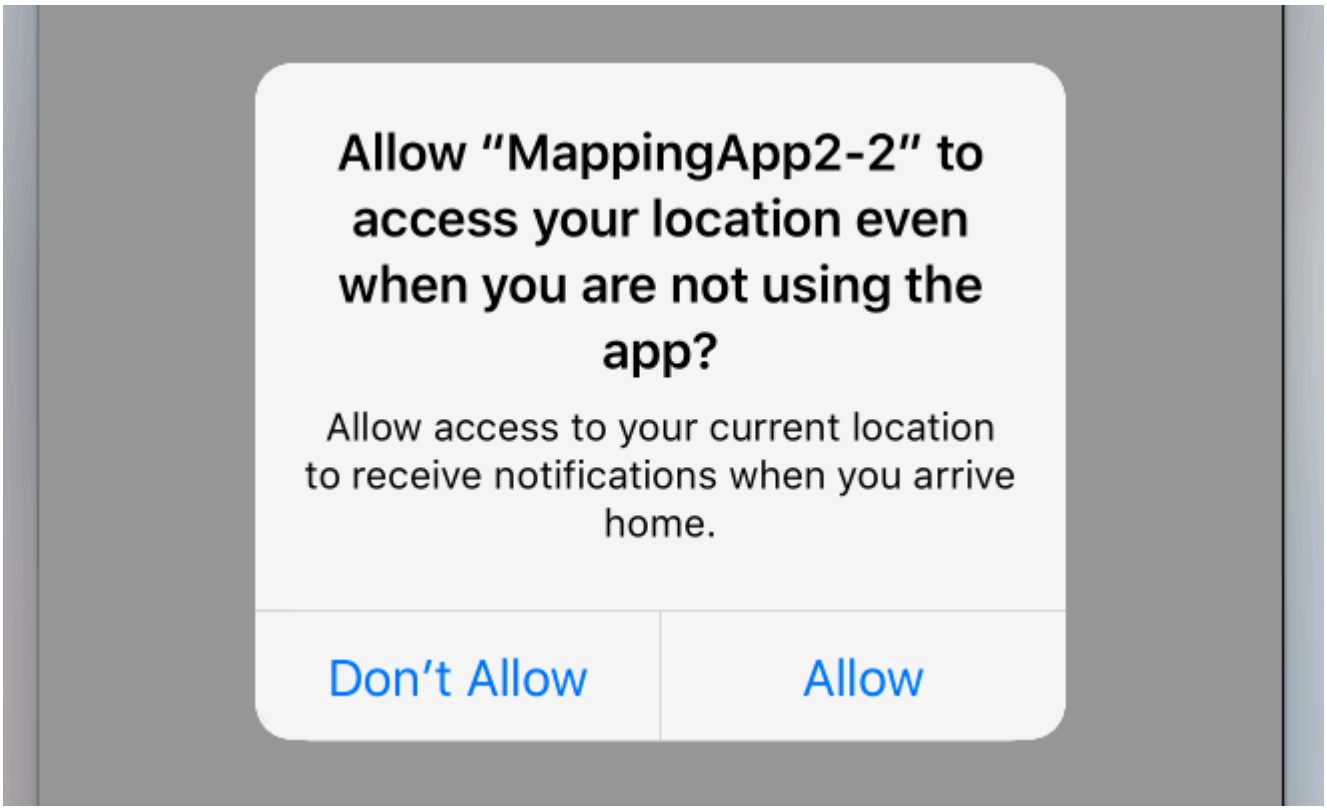
Link CoreLocation Framework


MappingApp
General
Cap
PROJECT
MappingApp2-2
TARGETS
MappingApp2-2
Choose frameworks and libraries
CoreL
iOS 9.2
CoreLocation.framework
Developer Frameworks
Add Other...

. Il valore verrà utilizzato nell'etichetta del `message` del controller di `message` .



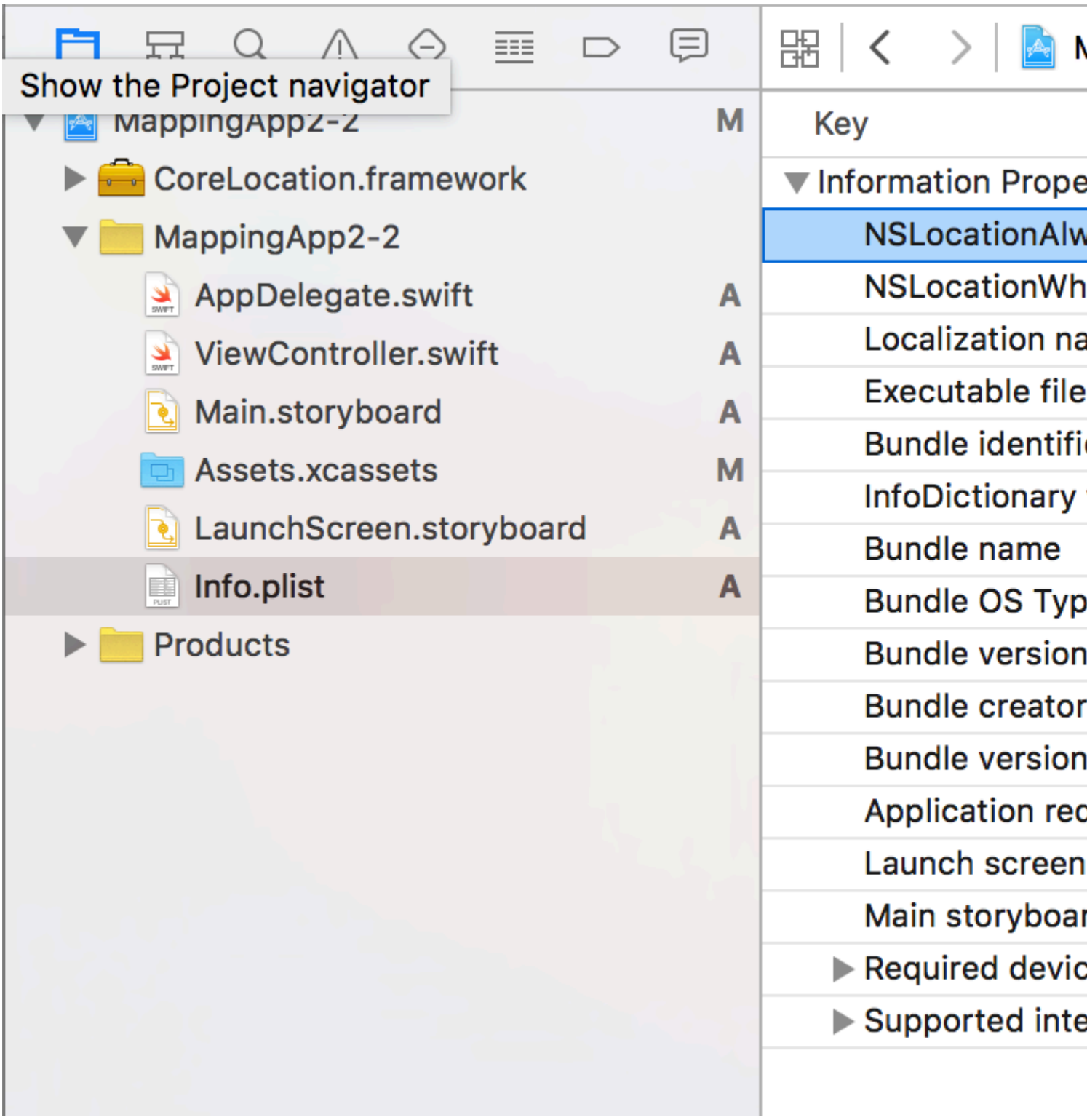
Ottenere sempre l'autorizzazione del servizio di localizzazione



Per chiedere il permesso di utilizzare i servizi di localizzazione anche quando l'app non è attiva, utilizzare invece la seguente chiamata:

```
//Swift
locationManager.requestAlwaysAuthorization()

//Objective-C
[locationManager requestAlwaysAuthorization];
```

Quindi aggiungi la chiave **NSLocationAlwaysUsageDescription** al tuo *Info.plist* . Anche in questo caso, il valore verrà utilizzato nell'etichetta del `message` del controller di `message` .



Aggiungi la tua posizione personalizzata usando il file GPX

Per verificare i servizi di localizzazione, abbiamo bisogno di un dispositivo reale, ma a scopo di test possiamo anche usare il simulatore e aggiungere la nostra posizione seguendo i seguenti passaggi:

- aggiungi un nuovo file GPX nel tuo progetto.
- nel file GPX aggiungi waypoint come

```
<?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>
```

- quindi vai al prodotto -> Schema -> Modifica schema e in RUN imposta la posizione predefinita come nome del tuo file GPX.

Servizi di localizzazione in background

Per utilizzare i servizi di localizzazione standard mentre l'applicazione è in background, è necessario innanzitutto attivare le `Background Modes` nella scheda Funzionalità delle impostazioni di destinazione e selezionare `Location updates` .

Oppure, aggiungilo direttamente a Info.plist.

```
<key>NSLocationAlwaysUsageDescription</key>
<string>I want to get your location Information in background</string>

<key>UIBackgroundModes</key>
<array>
    <string>location</string>
</array>
```

Quindi è necessario configurare CLLocationManager

Obiettivo 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];
```

veloce

```
self.locationManager.delegate = self
```

```
if #available (iOS 8.0,*) {
    self.locationManager.requestAlwaysAuthorization()
}

if #available (iOS 9.0,*) {
    self.locationManager.allowsBackgroundLocationUpdates = true
}

self.locationManager.startUpdatingLocation()
```

Leggi Core Location online: https://riptutorial.com/it/ios/topic/2937/core-location

Capitolo 46: Core Motion

Examples

Accedere al barometro per ottenere l'altitudine relativa

veloce

Importa la libreria Core Motion:

```
import CoreMotion
```

Successivamente, dobbiamo creare un oggetto `CMAltimeter` , ma una trappola comune è crearla in `viewDidLoad()` . Se fatto in questo modo, l'altimetro non sarà accessibile quando abbiamo bisogno di chiamare un metodo su di esso. Tuttavia, vai avanti e crea il tuo oggetto `CMAltimeter` subito prima di `viewDidLoad()` :

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

Adesso:

1. Abbiamo bisogno di controllare se `relativeAltitude` è anche disponibile con il seguente metodo: `CMAltimeter.isRelativeAltitudeAvailable` .
2. Se ciò restituisce `true` , puoi iniziare a monitorare il cambio di altitudine con `startRelativeAltitudeUpdatesToQueue`
3. Se non ci sono errori, dovresti essere in grado di recuperare i dati dalle proprietà `relativeAltitude` e pressione.

Di seguito è riportata la definizione di un'azione del pulsante per iniziare il monitoraggio con il nostro barometro.

```
@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)")
        }
    })
}
```

Leggi Core Motion online: https://riptutorial.com/it/ios/topic/7636/core-motion

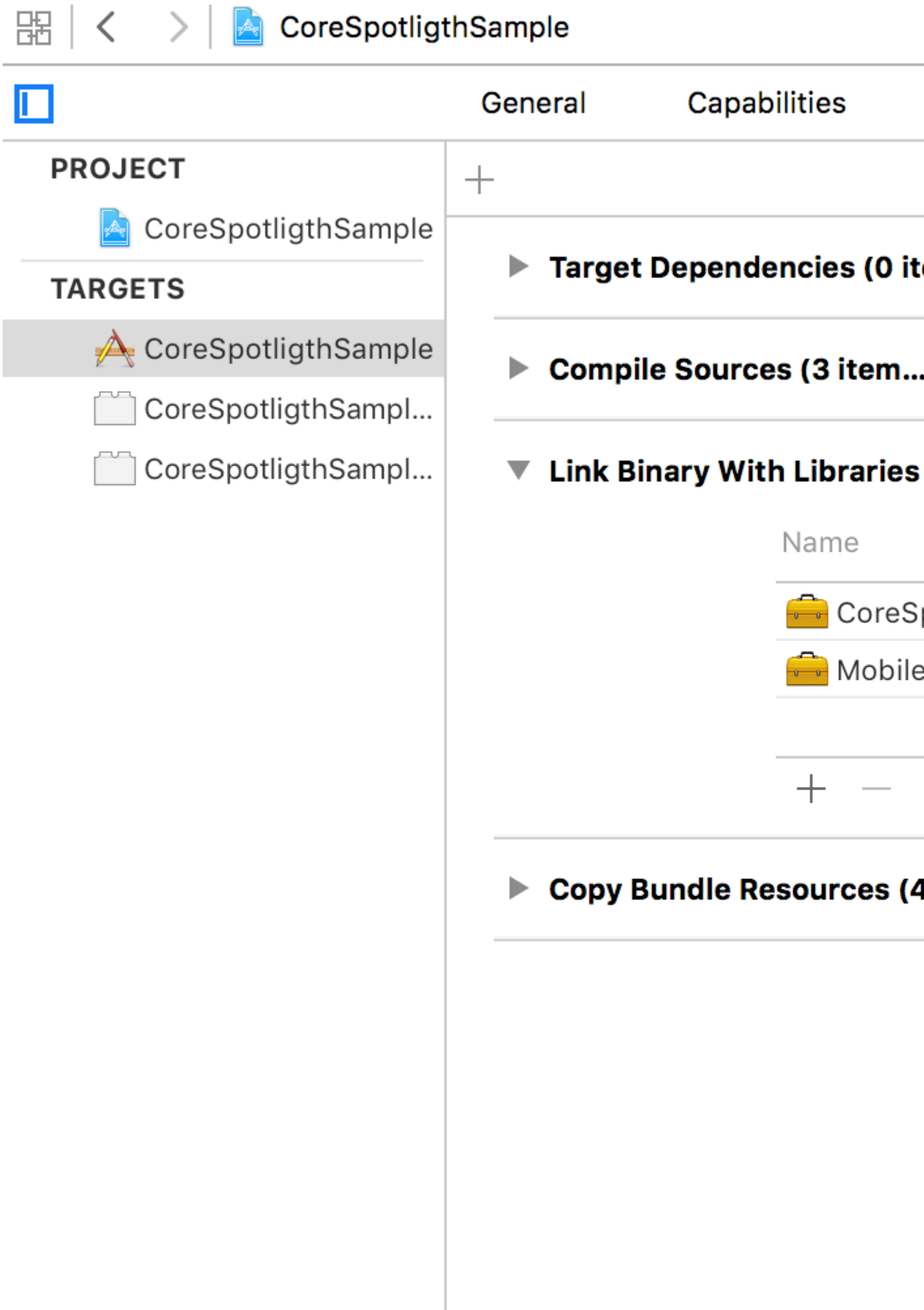
Capitolo 47: Core SpotLight in iOS

Examples

Core-Spotlight

Objective-C

1. Crea un nuovo progetto iOS e aggiungi il *framework CoreSpotlight* e *MobileCoreServices* al tuo progetto.


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. Creare il CSSearchableItem effettivo e associare uniqueIdentifier, domainIdentifier e attributeSet. Infine indicizza CSSearchableItem usando [[CSSearchableIndex defaultSearchableIndex] ...] come mostrato di seguito.

```
//
//  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, prova l'indice!

Capitolo 48: Corsia di sorpasso

Examples

strumenti fastlane

fastlane è uno strumento di automazione di build open source per Android e iOS per sviluppatori. Riduce il tempo di generazione della build. È uno strumento a riga di comando che usa Ruby , quindi hai bisogno di Ruby sul tuo computer. La maggior parte dei Mac ha già installato Ruby di default.

Installare fastlane

1. Apri un terminale.
2. Esegui `sudo gem install fastlane --verbose`
3. Se non hai ancora installato gli strumenti da riga di comando Xcode, esegui `xcode-select --install` per installarli
4. Ora, `cd` nella cartella del progetto (digita `cd` [con lo spazio alla fine] e trascina la cartella del progetto nel terminale)
5. Esegui `fastlane init` per ottenere l'installazione fastlane.
6. Ora puoi usare tutti gli strumenti di Fastlane:

Strumenti iOS

- consegna : carica screenshot, metadati e la tua app sull'App Store
- istantanea : automatizza l'acquisizione di screenshot localizzati della tua app iOS su ogni dispositivo
- frameit : inserisci rapidamente gli screenshot nei frame dei dispositivi corretti
- pem : genera e rinnova automaticamente i profili di notifica push
- sospiro : perché preferiresti passare il tuo tempo a costruire cose che a combattere il provisioning
- produce : crea nuove app iOS su iTunes Connect e Dev Portal utilizzando la riga di comando
- cert : crea e conserva automaticamente certificati di firma del codice iOS
- palestra : costruire le tue app iOS non è mai stato così facile
- corrispondenza : sincronizza facilmente i tuoi certificati e i profili del tuo team utilizzando Git
- scansione : il modo più semplice per eseguire test per le tue app iOS e Mac
- astronave : Ruby library per accedere a Apple Dev Center e iTunes Connect

iOS TestFlight Tools

- pilota : il modo migliore per gestire i tester TestFlight e le build dal tuo terminale
- imbarco : il modo più semplice per invitare i beta tester TestFlight

Strumenti Android

- fornitura : carica la tua app Android e i relativi metadati su Google Play
- screengrab : automatizza l'acquisizione di screenshot localizzati della tua app Android su ogni dispositivo

Leggi Corsia di sorpasso online: https://riptutorial.com/it/ios/topic/3574/corsia-di-sorpasso

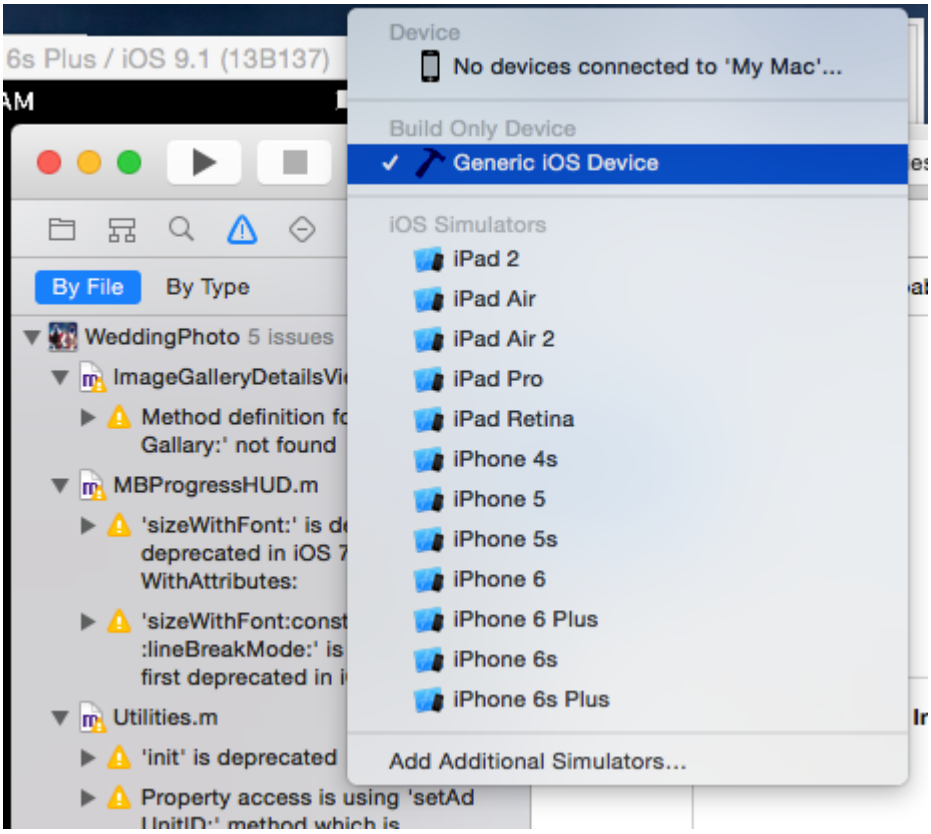
Capitolo 49: Crea il file .ipa da caricare su appstore con Applicationloader

Examples

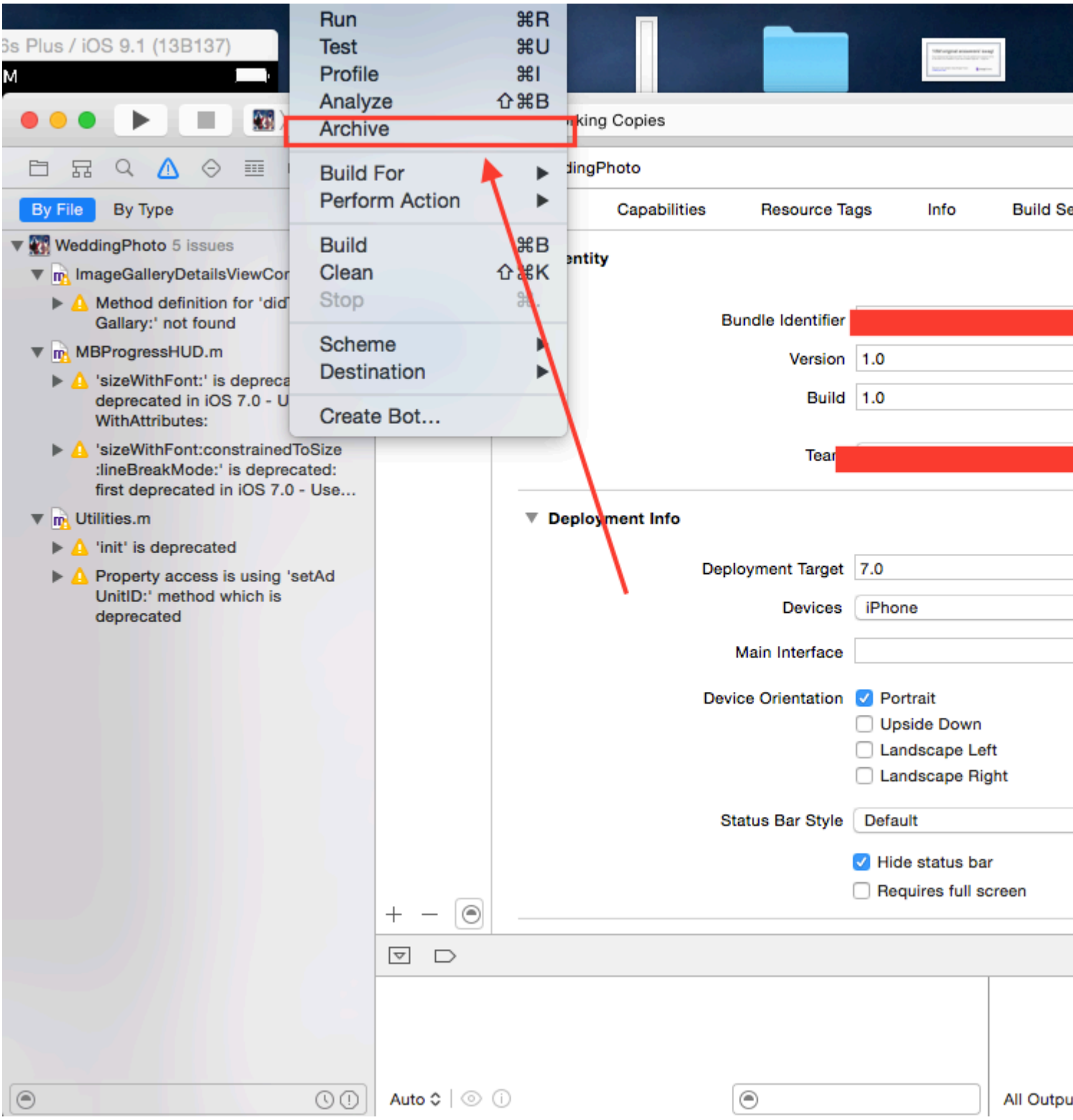
crea il file .ipa per caricare l'app nell'appstore con Application Loader

Se si desidera caricare il file .ipa su itunesconnect **senza integrare l'account sviluppatore in Xcode** e si desidera utilizzare il **caricatore applicazioni** . quindi puoi **generare .ipa con iTunes** .

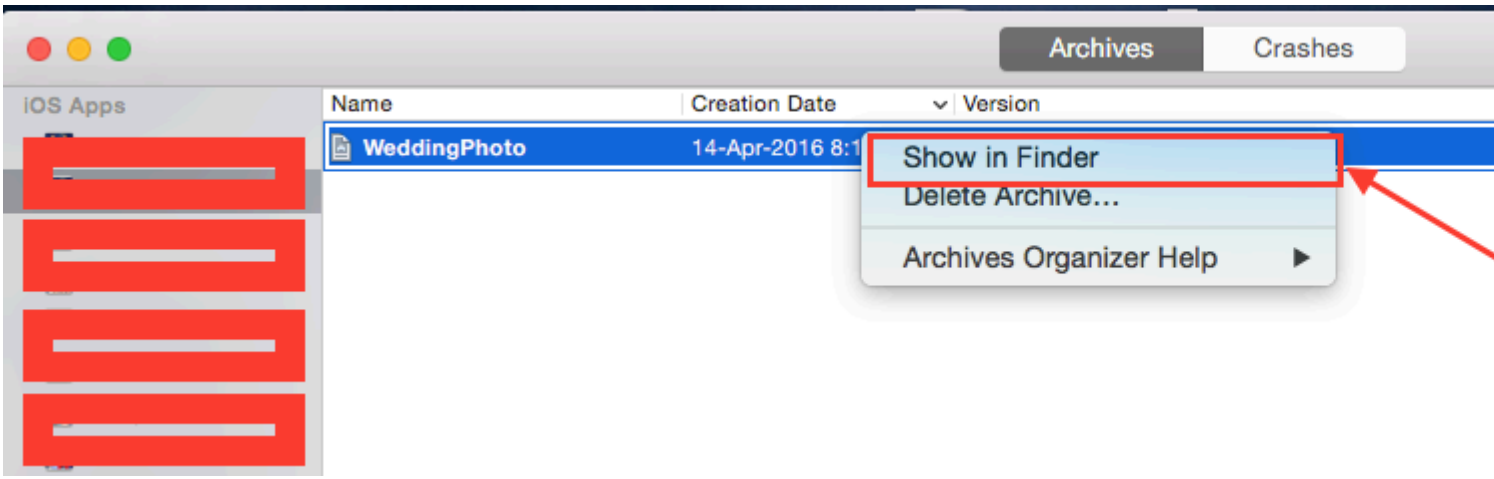
Passo 1: - Seleziona il dispositivo al posto del simulatore.



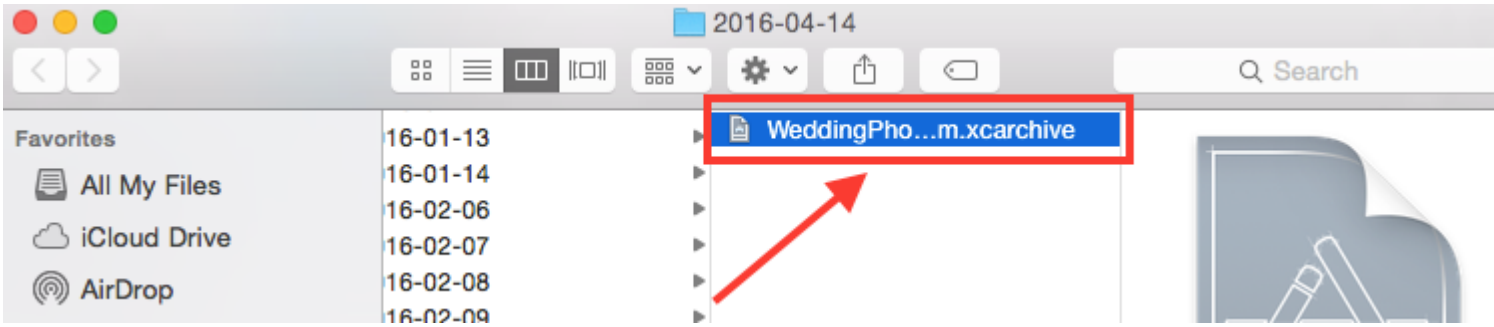
Passaggio 2: - Vai a Prodotto -> seleziona Archivio



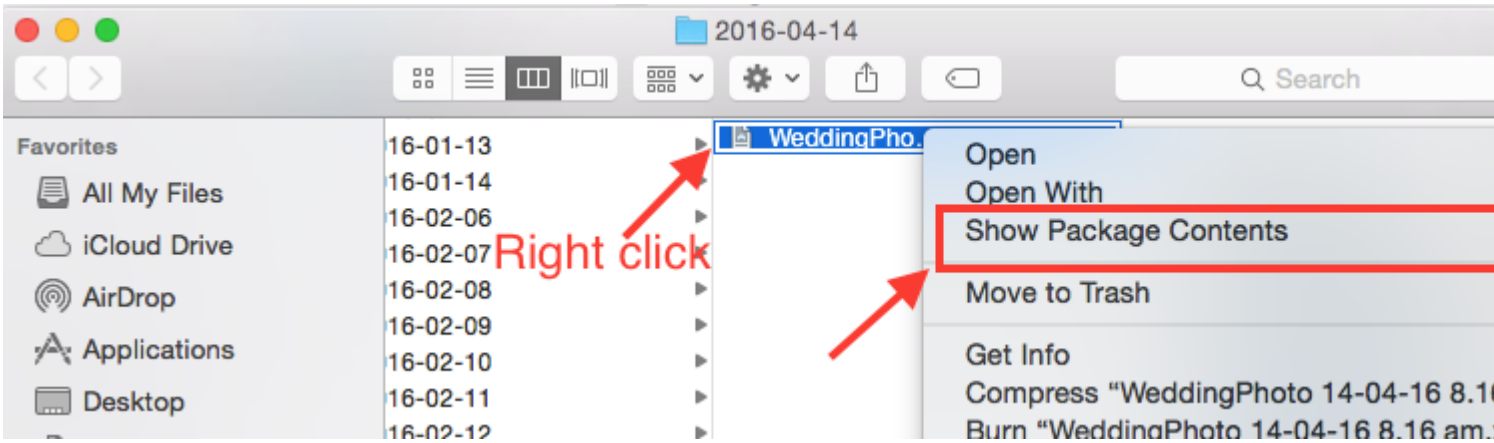
Passaggio 3: - Dopo aver completato il processo, fare clic con il pulsante destro del mouse su Archivio -> e selezionare Mostra nel Finder



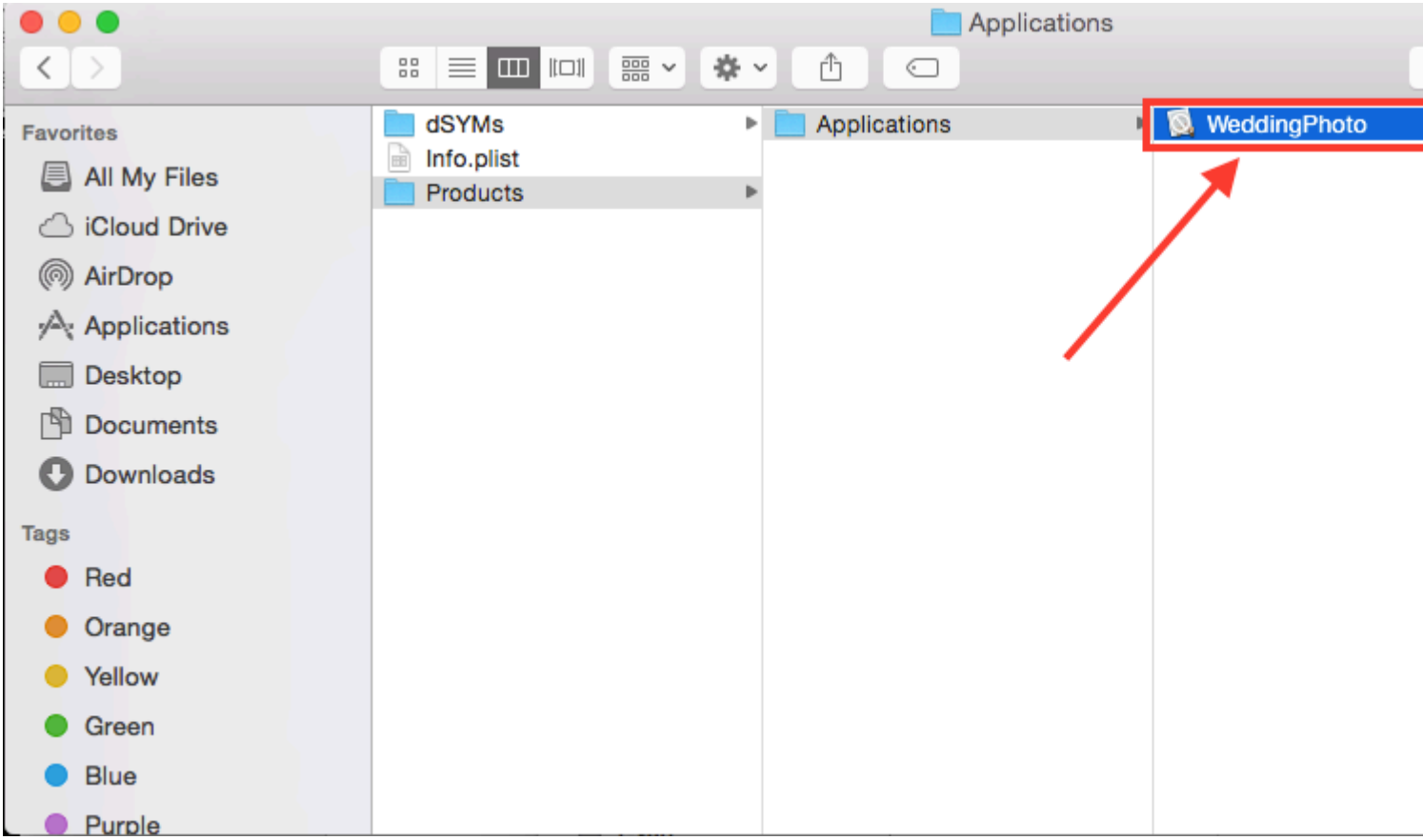
Passo 4: - quando si fa clic su Mostra nel Finder si reindirizza alla cartella Archivio, appare come questo



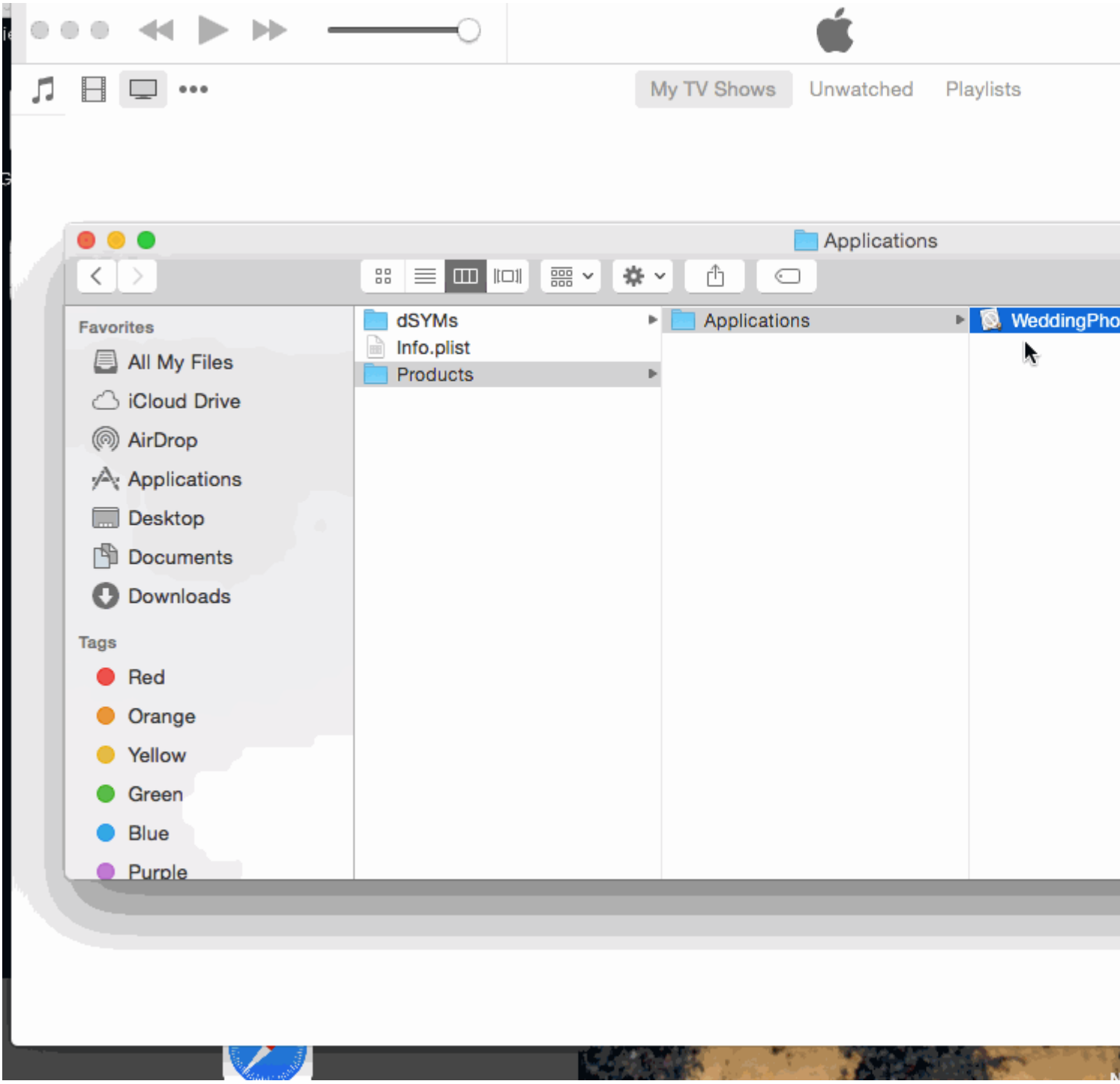
Passaggio 5: - Fare clic con il pulsante destro del mouse sul file .xarchive -> selezionare Mostra nell'opzione di ricerca.



Passo 6: - Vai alla cartella del prodotto -> Cartella dell'applicazione -> Troverai il tuo nomeproject.app



Passo 7: - Ora per convertire .app in .ipa basta trascinare e rilasciare in iTunes. controlla sotto l'immagine,



Passo 8: - Ora metti questo file .ipa in un posto sicuro e usalo quando carichi con il caricatore di applicazioni.

Nota: - se vuoi sapere come caricare l'app con il caricatore dell'applicazione, controlla questo,

Carica l'app con Loader dell'applicazione

MODIFICARE :-

ATTENZIONE: - Non fare .ipa cambiando l'estensione da .aap a .zip e .zip a .ipa.

Ho visto in molte risposte che, hanno suggerito di comprimere il file .app e quindi modificare l'estensione da .zip a .ipa. Non sta funzionando ora. Con questo metodo otterrai errore come,

IPA non è valido, non include una directory di payload.

Leggi Crea il file .ipa da caricare su appstore con Applicationloader online: https://riptutorial.com/it/ios/topic/6119/crea-il-file--ipa-da-caricare-su-appstore-con-applicationloader

Capitolo 50: Crea un framework personalizzato in iOS

Examples

Crea Framework in Swift

segui questi passaggi per creare Custom Framework in Swift-IOS:

1. Crea un nuovo progetto. In Xcode
2. Scegli iOS / Framework e Libreria / Cocoa Touch Framework per creare un nuovo framework
3. fare clic su Avanti e impostare productName
4. fai clic su Avanti e scegli la directory per creare il Progetto lì
5. aggiungi codice e risorse al progetto creato

Framework creato con successo

per aggiungere un framework creato a un altro progetto, per prima cosa è necessario creare uno spazio di lavoro
aggiungere "progetto target" e "progetto quadro" all'area di lavoro, quindi:

1. vai alla scheda generale del progetto di destinazione
2. trascinare il file "* .framework" nella cartella del prodotto del progetto quadro nella sezione "Binari incorporati"
3. per utilizzare in qualsiasi ViewController o classe basta importare framework in ogni file

Leggi Crea un framework personalizzato in iOS online: https://riptutorial.com/it/ios/topic/7331/crea-un-framework-personalizzato-in-ios

Capitolo 51: Crea un video dalle immagini

introduzione

Crea un video da immagini usando AVFoundation

Examples

Crea video da UIImages

Prima di tutto è necessario creare `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` richiede almeno un input di asset writer.

```
 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
 }
```

Per aggiungere `CVPixelBufferRef` a `AVAssetWriterInput` è necessario creare `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];
```

Ora possiamo iniziare a scrivere

```
[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
        }
    }];
}];
```

Ecco una funzione per ottenere `CVPixelBufferRef` da `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;
}
```

Leggi Crea un video dalle immagini online: https://riptutorial.com/it/ios/topic/10607/crea-un-video-dalle-immagini

Capitolo 52: Creazione di PDF in iOS

Examples

Crea PDF

```
UIGraphicsBeginPDFContextToFile(fileName, CGRectZero, nil);

UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 612, 792), nil);

[self drawText];

UIGraphicsEndPDFContext();
```

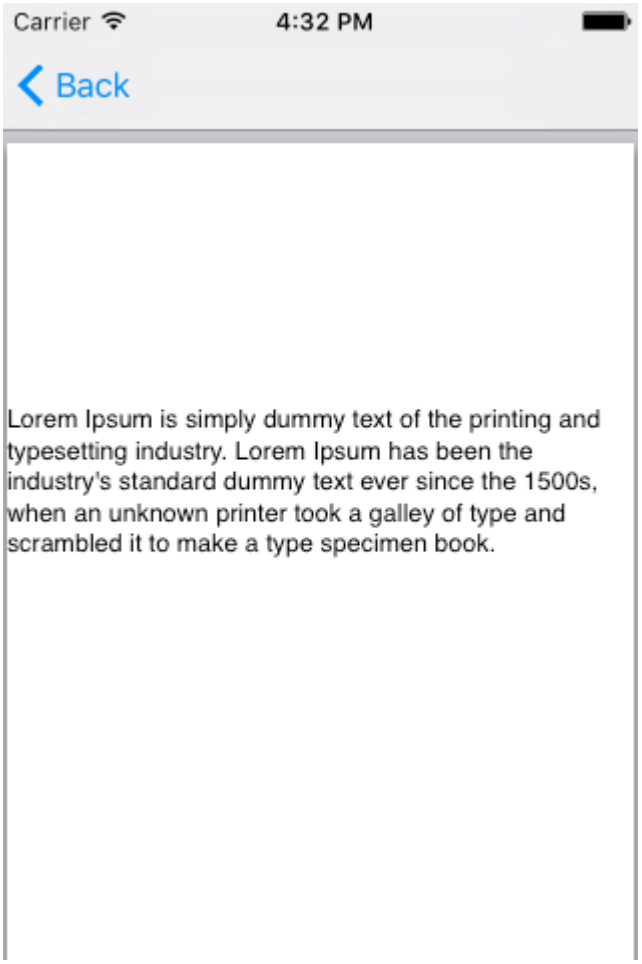
fileName è il file del documento in cui si intende aggiungere o allegare

```
 NSString* temporaryFile = @"firstIOS.PDF";
    NSArray *arrayPaths =
    NSSearchPathForDirectoriesInDomains(
                                        NSDocumentDirectory,
                                        NSUserDomainMask,
                                        YES);

    NSString *path = [arrayPaths objectAtIndex:0];

    NSString* fileName = [path stringByAppendingPathComponent:fileName];
```

Dove **drawText** è

```
   (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);
}
```



Mostra 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];
```

PDF a più pagine

```
 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();
```

Crea PDF da qualsiasi documento Microsoft caricato in UIWebview

```
#define kPaperSizeA4 CGSizeMake(595.2,841.8)
```

Innanzitutto implementare il protocollo 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
```

Quindi, chiama il metodo seguente dopo il caricamento terminato del documento in `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");
    }
});}
```

Leggi Creazione di PDF in iOS online: https://riptutorial.com/it/ios/topic/2416/creazione-di-pdf-in-ios

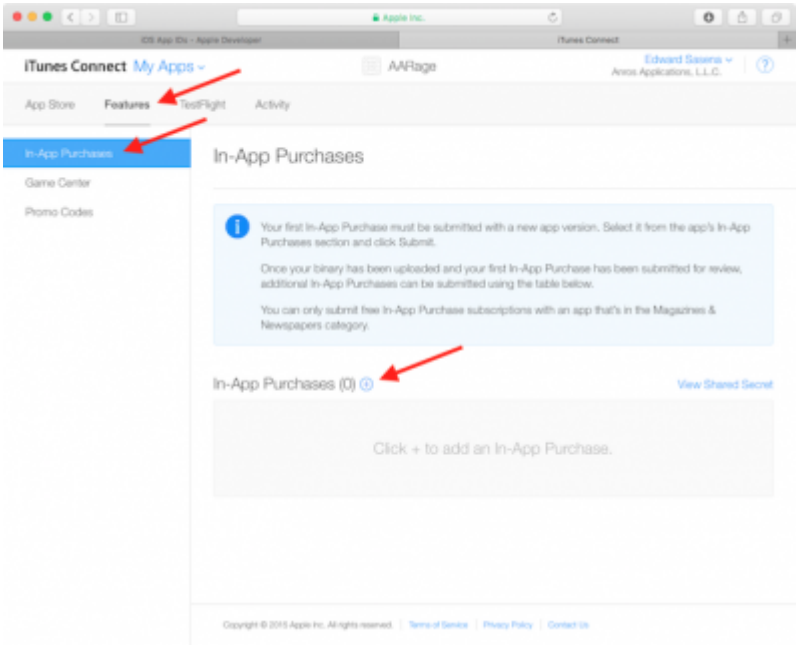
Capitolo 53: Creazione di un ID app

Examples

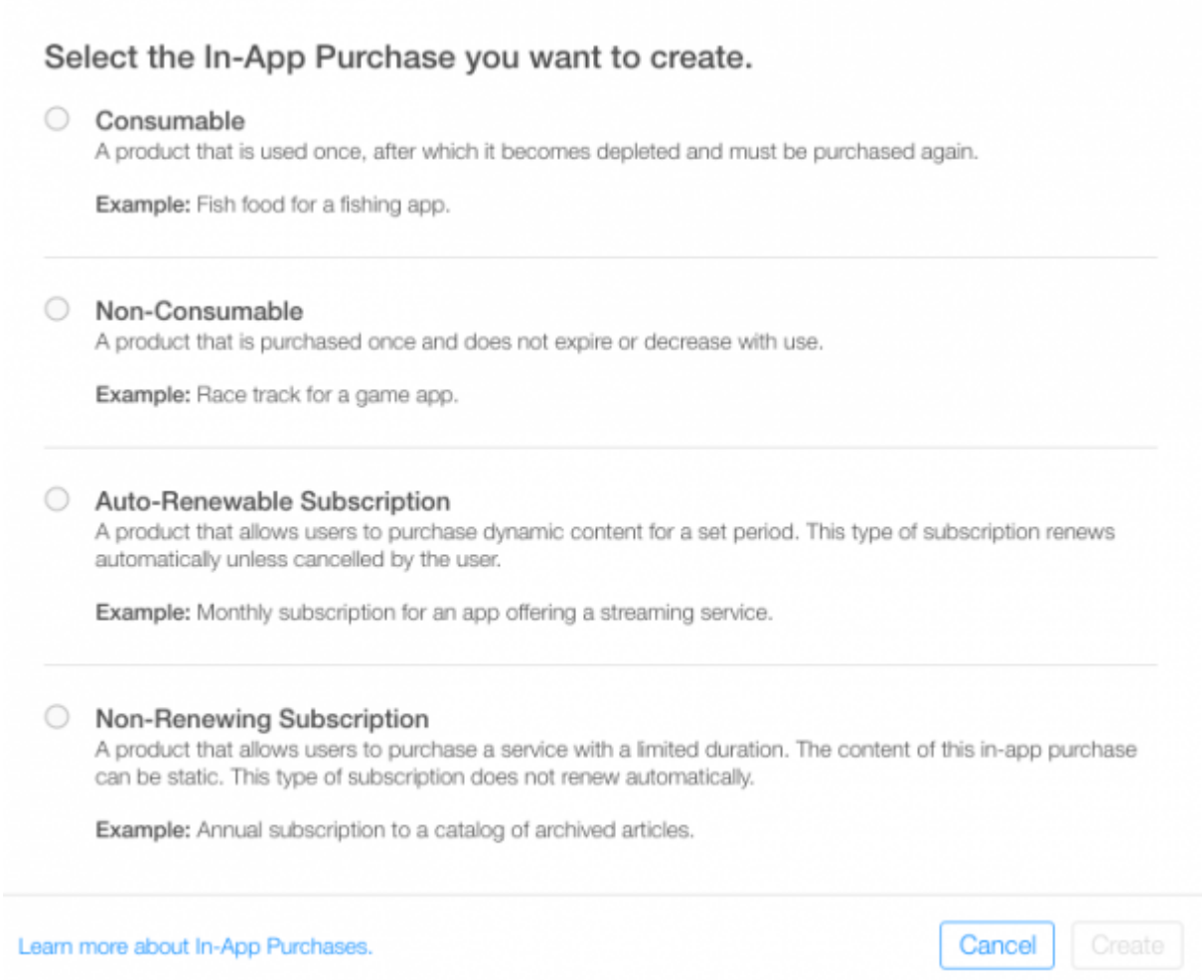
Creazione di prodotti di acquisto in-app

- Quando offri IAP all'interno di un'app, devi innanzitutto aggiungere una voce per ogni singolo acquisto all'interno di iTunes Connect. Se hai mai elencato un'app in vendita nel negozio, è un processo simile e include cose come la scelta di un livello di prezzo per l'acquisto. Quando l'utente effettua un acquisto, l'App Store gestisce il complesso processo di addebito sull'account iTunes dell'utente. Ci sono molti tipi diversi di IAP che puoi aggiungere:
    - **Consumabili** : possono essere acquistati più di una volta e possono essere esauriti. Queste sono cose come vite extra, valuta di gioco, power-up temporanei e simili.
    - **Non consumabile** : qualcosa che acquisti una sola volta e prevedi di avere in modo permanente livelli extra e contenuti sbloccabili.
    - **Sottoscrizione non rinnovata** : contenuto disponibile per un determinato periodo di tempo.
    - **Abbonamento con rinnovo automatico** : un abbonamento ripetuto come un abbonamento mensile raywenderlich.com.

Puoi offrire acquisti in-app solo per articoli digitali e non per beni o servizi fisici. Per ulteriori informazioni su tutto ciò, consultare la documentazione completa di Apple sulla creazione di prodotti di acquisto in-app. Ora, mentre visualizzi la voce della tua app in iTunes Connect, fai clic sulla scheda Funzioni e poi seleziona Acquisti in-app. Per aggiungere un nuovo prodotto IAP, fai clic su + a destra di Acquisti in-app.



Apparirà la seguente finestra di dialogo:



Quando un utente acquista un fumetto rabbia nella tua app, vorrai che abbia sempre accesso ad esso, quindi seleziona Non consumabile e fai clic su Crea. Quindi, compila i dettagli per l'IAP come segue:

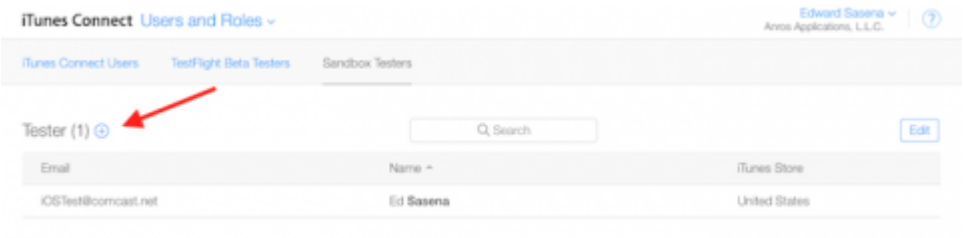
- **Nome di riferimento** : un nickname che identifica l'IAP all'interno di iTunes Connect. Questo nome non appare in nessuna parte dell'app. Il titolo del fumetto che sbloccherai con questo acquisto è **"Girlfriend of Drummer"** , quindi entra qui.
- **ID prodotto** : si tratta di una stringa univoca che identifica lo IAP. Di solito è meglio iniziare con l'ID bundle e quindi aggiungere un nome univoco specifico per questo oggetto acquistabile. Per questo tutorial, assicurati di aggiungere "GirlfriendOfDrummerRage", in quanto verrà utilizzato in seguito all'interno dell'applicazione per cercare il fumetto da sbloccare. Quindi, ad esempio: com.theNameYouPickedEarlier.Rage.GirlFriendOfDrummerRage.
- **Cleared for Sale** : abilita o disabilita la vendita dello IAP. Vuoi abilitarlo!
- **Livello di prezzo** : il costo dello IAP. Scegli il livello 1.

Ora scorri verso il basso fino alla sezione Localizzazioni e nota che esiste una voce predefinita per l'inglese (Stati Uniti). Inserisci "Girlfriend of Drummer" sia per il Nome visualizzato che per la Descrizione. Fai clic su Salva. Grande! Hai creato il tuo primo prodotto IAP.

È necessario un ulteriore passaggio prima di poter approfondire il codice. Durante il test degli acquisti in-app in una build di sviluppo di un'app, Apple fornisce un ambiente di test che consente di "acquistare" i prodotti IAP senza creare transazioni finanziarie.

Creazione di un utente Sandbox

In iTunes Connect, fai clic su iTunes Connect nell'angolo in alto a sinistra della finestra per tornare al menu principale. Selezionare Utenti e ruoli, quindi fare clic sulla scheda Sandbox Tester. Fai clic su + accanto al titolo "Tester".



Compila le informazioni e fai clic su Salva quando hai finito. Puoi creare un nome e un cognome per il tuo utente di prova, ma l'indirizzo email scelto deve essere un indirizzo email reale, poiché una verifica verrà inviata all'indirizzo di Apple. Dopo aver ricevuto quell'e-mail, assicurati di fare clic sul link al suo interno per verificare il tuo indirizzo. Anche l'indirizzo email che inserisci NON dovrebbe essere già associato a un account ID Apple. Suggerimento: se hai un account Gmail, puoi semplicemente usare un alias di indirizzo invece di dover creare un nuovo account

Leggi Creazione di un ID app online: https://riptutorial.com/it/ios/topic/10854/creazione-di-un-id-app

Capitolo 54: CTCallCenter

Examples

Intercettazione delle chiamate dalla tua app anche dallo sfondo

Dalla documentazione di Apple:

> Utilizzare la classe CTCallCenter per ottenere un elenco di chiamate cellulari correnti e per rispondere alle modifiche di stato per le chiamate, ad esempio da uno stato di composizione a uno stato connesso. Tali cambiamenti di stato sono noti come eventi di chiamata cellulare.

Lo scopo di CTCallCenter è di dare allo sviluppatore la possibilità di mettere in pausa il suo stato di app durante una chiamata per dare all'utente la migliore esperienza.

Objective-C:

In primo luogo, definiremo un nuovo membro della classe all'interno della classe che intendiamo gestire le intercettazioni:

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

All'interno della nostra classe init (costruttore) assegneremo nuova memoria per il nostro membro della classe:

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

Successivamente, invocheremo il nostro nuovo metodo che gestisce effettivamente le intercettazioni:

```
- (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");
    }
 }];
}
```

È così, se l'utente userà la tua app e riceverà una telefonata, potresti intercettare questa chiamata e gestire l'app per uno stato di salvataggio.

Vale la pena ricordare che ci sono 4 stati di chiamata che puoi intercettare:

```
CTCallStateDialing
CTCallStateIncoming
CTCallStateConnected
CTCallStateDisconnected
```

Swift:

Definisci il tuo membro della classe nella classe rilevante e definiscilo:

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

Che cosa succederà se la tua app è in background e devi intercettare le chiamate mentre l'app è in background?

Ad esempio, se sviluppi un'app **aziendale** , puoi semplicemente aggiungere 2 funzionalità (VOIP e recupero in background) nella scheda Funzionalità:

Il tuo obiettivo di progetto -> Funzionalità -> Modalità di sfondo -> contrassegna Voice over IP e recupero in background

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
    }
}
```

Leggi CTCallCenter online: https://riptutorial.com/it/ios/topic/3007/ctcallcenter

Capitolo 55: CydiaSubstrate tweak

introduzione

Scopri come creare modifiche al substrato Cydia per iPhone jailbroken.

Queste modifiche ti permetteranno di modificare il comportamento del sistema operativo per agire nel modo che preferisci.

Osservazioni

Installare Theos

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

Examples

Crea nuovo tweak usando Theos

Usa nic per creare un nuovo progetto

Inserisci questo comando nel tuo terminale

```
$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):
```

Scegli il modello `[11.] iphone/tweak`

Compila i dettagli e otterrai i seguenti file creati:

```
-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
```

Sostituisci il metodo di salvataggio degli screenshot iOS

apri il file `Tweak.xm` usando il tuo editor di codice preferito.

agganciare a un determinato metodo dal sistema operativo.

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

Nota puoi scegliere se la funzione originale debba essere chiamata, ad esempio:

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

sovrascriverà la funzione senza chiamare quella originale, in modo da non salvare gli screenshot.

Leggi CydiaSubstrate tweak online: https://riptutorial.com/it/ios/topic/10533/cydiasubstrate-tweak

Capitolo 56: Dati principali

introduzione

Core Data è il livello del modello della tua applicazione nel senso più ampio possibile. È il modello nel modello Model-View-Controller che permea l'iOS SDK.

Core Data non è il database della tua applicazione né è un'API per la permanenza dei dati in un database. Core Data è un framework che gestisce un oggetto grafico. E 'così semplice. I Core Data possono persistere quel grafico dell'oggetto scrivendolo su disco, ma questo non è l'obiettivo principale del framework.

Examples

Operazioni sui dati principali

Per ottenere il contesto:

```
NSManagedObjectContext *context = ((AppDelegate*)[[UIApplication sharedApplication]
delegate]).persistentContainer.viewContext;
```

Per recuperare i dati:

```
NSFetchRequest<EntityName *> *fetchRequest = [EntityName fetchRequest];
NSError *error ;
NSArray *resultArray= [context executeFetchRequest:fetchRequest error:&error];
```

Per recuperare i dati con l'ordinamento:

```
NSFetchRequest<EntityName *> *fetchRequest = [EntityName fetchRequest];
NSSortDescriptor *sortDescriptor = [NSSortDescriptor sortDescriptorWithKey:@"someKey"
ascending:YES];
fetchRequest.sortDescriptors = @[sortDescriptor];
NSError *error ;
NSArray *resultArray= [context executeFetchRequest:fetchRequest error:&error];
```

Per aggiungere dati:

```
NSManagedObject *entityNameObj = [NSEntityDescription
insertNewObjectForEntityForName:@"EntityName" inManagedObjectContext:context];
[entityNameObj setValue:@"someValue" forKey:@"someKey"];
```

Per salvare il contesto:

```
[((AppDelegate*)[[UIApplication sharedApplication] delegate]) saveContext];
```

Leggi Dati principali online: https://riptutorial.com/it/ios/topic/9489/dati-principali

Capitolo 57: Deep linking in iOS

Osservazioni

Documentazione Apple utile con esempi e chiarimenti.

Examples

Aprire un'app basata sul suo schema URL

Per aprire un'app con schema URL definito `todolist://` :

Objective-C

```
NSURL *myURL = [NSURL URLWithString:@"todolist://there/is/something/to/do"];
[[UIApplication sharedApplication] openURL:myURL];
```

veloce

```
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>
```

> **Nota:** è utile verificare se il collegamento può essere aperto per visualizzare altrimenti un messaggio appropriato all'utente. Questo può essere fatto usando `canOpenURL:` method.

Aggiunta di uno schema URL alla tua app

Supponiamo che tu stia lavorando su un'app chiamata `MyTasks` e desideri consentire agli URL in entrata di creare una nuova attività con un titolo e un corpo. L'URL che stai progettando potrebbe essere simile a questo:

```
mytasks://create?title=hello&body=world
```

(Naturalmente, i parametri del `text` e del `body` sono usati per popolare il nostro compito che stiamo creando!)

Ecco i grandi passaggi per aggiungere questo schema URL al tuo progetto:

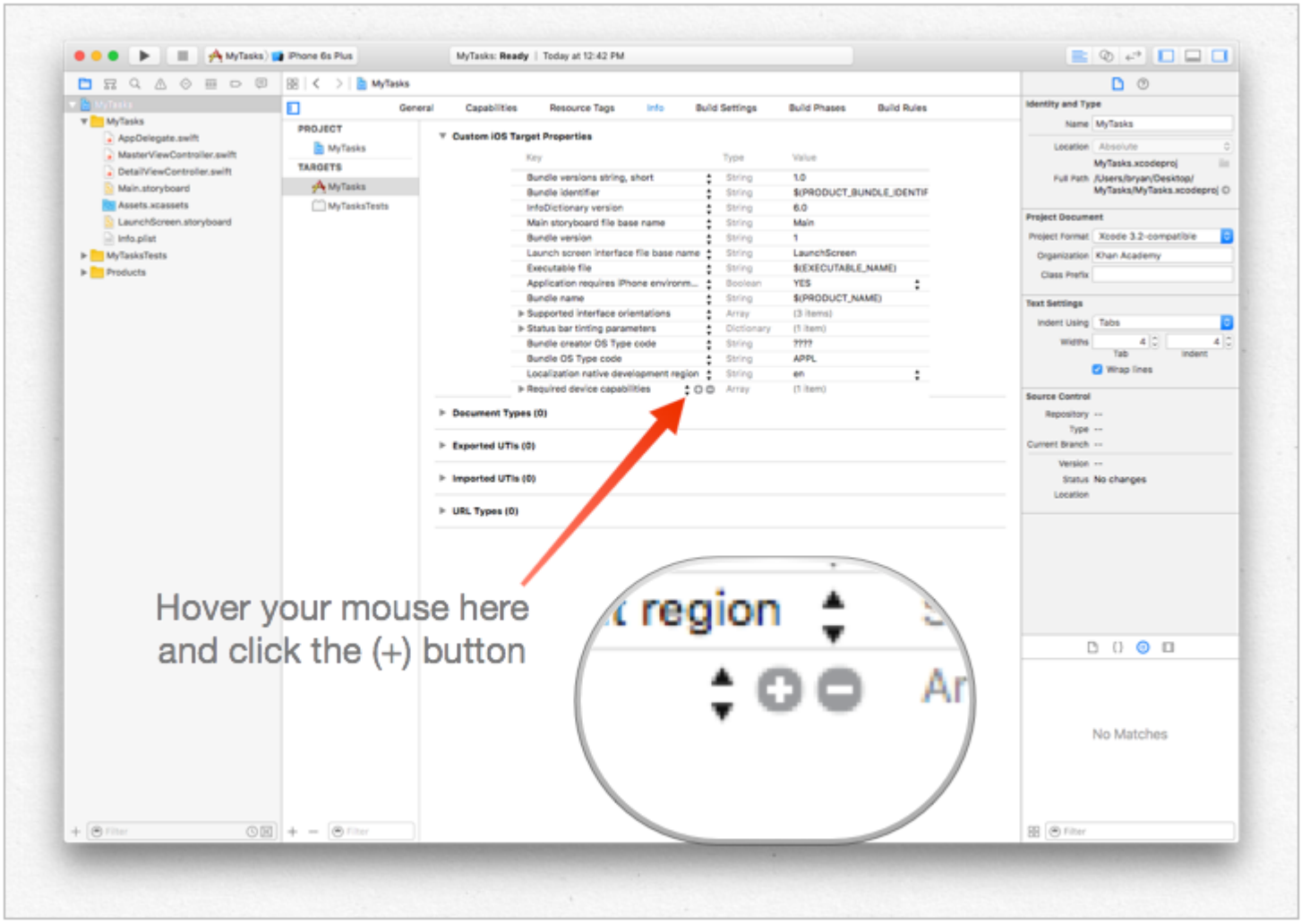
1. Registra uno schema URL nel file `Info.plist` della tua app, in modo che il sistema sappia

quando indirizzare un URL alla tua app.

2. Aggiungi una funzione al tuo `UIApplicationDelegate` che accetta e gestisce gli URL in arrivo.
3. Esegui qualsiasi attività che deve essere eseguita quando viene aperto quell'URL.

Fase uno: registrare uno schema URL in Info.plist:

Per prima cosa, dobbiamo aggiungere una voce "Tipi di URL" al nostro file Info.plist. Fai clic sul pulsante (+) qui:



... quindi inserisci un identificativo univoco per la tua app, nonché lo schema URL che desideri utilizzare. Sii specifico! Non vuoi che lo schema URL sia in conflitto con l'implementazione di un'altra app. Meglio essere troppo lungo qui che troppo breve!

Passaggio 2: gestire l'URL in UIApplicationDelegate

Dobbiamo implementare l' `application:openURL:options:` sul nostro `UIApplicationDelegate` . Ispezioneremo l' `URL` arrivo e vediamo se c'è un'azione da intraprendere!

Un'implementazione sarebbe questa:

```
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
}
```

Passaggio 3: eseguire un'attività in base all'URL.

Quando un utente apre la tua app tramite un URL, probabilmente si aspettava *che* succedesse *qualcosa* . Forse sta navigando verso un contenuto, forse sta creando un nuovo elemento: in questo esempio, creeremo una nuova attività nell'app!

Nel codice sopra, possiamo vedere una chiamata a `self.rootViewController.createTaskWithTitle(:body:)` - così, supponendo che il tuo `AppDelegate` abbia un puntatore al suo controller di visualizzazione radice che implementa correttamente la funzione, sei pronto!

Impostazione di deeplink per la tua app

La configurazione del deep linking per la tua app è semplice. Basta un piccolo url con cui vuoi aprire la tua app.

Segui i passaggi per configurare il deep-linking per la tua app.

1. Consente di creare un progetto e denominarlo DeepLinkPOC.
2. Ora seleziona il tuo obiettivo di progetto.
3. Dopo aver selezionato il bersaglio, seleziona la scheda "Informazioni".
4. Scorri verso il basso fino a quando non vedi un'opzione di **Tipi** di **URL**

5. Fai clic sull'opzione "+".

6. Vedrai che gli **schemi URL** aggiungono una stringa tramite la quale vuoi aprire l'app. Aggiunge " **DeepLinking** " negli schemi URL.

Quindi, per aprire la tua app puoi avviarla digitando **"DeepLinking: //"** nel tuo safari. La tua stringa di deep-linking ha il seguente formato.

```
[scheme]://[host]/[path]  --> DeepLinking://path/Page1
```

dove, Scheme: Host "DeepLinking": percorso "percorso": "Pagina1"

Nota : anche se non aggiungere l'host e il percorso, verrà avviata l'app, quindi non preoccuparti. È possibile aggiungere host e percorso per reindirizzare ulteriormente a una determinata pagina dopo l'avvio dell'applicazione.

7. Ora aggiungi il seguente metodo al tuo accountdelegate.

Swift:

```
 func application(application: UIApplication, openURL url: NSURL, sourceApplication: String?,
annotation: AnyObject) -> Bool
```

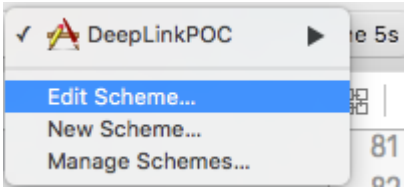
Objective-C:

```
-(BOOL)application:(UIApplication *)application
          openURL:(NSURL *)url
          sourceApplication:(NSString *)sourceApplication
          annotation:(id)annotation
```

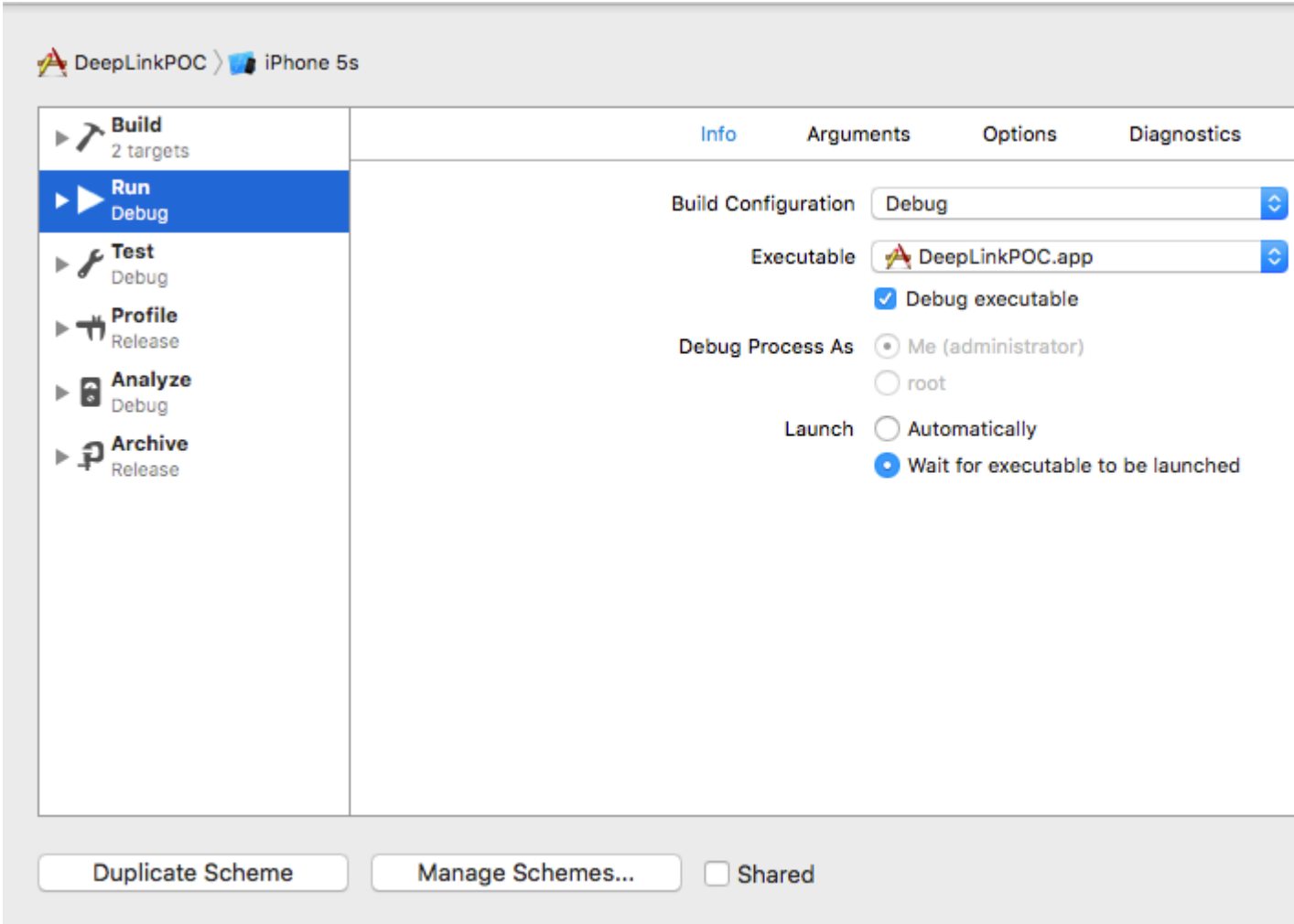
Il metodo precedente viene chiamato ogni volta che la tua app viene avviata utilizzando una stringa di deep linking impostata per la tua app.

8. Ora è il momento di installare la tua app, ma attendi prima di saltare direttamente al pulsante di esecuzione. Effettua un piccolo cambiamento nel metodo di avvio dell'app dello schema.

- Seleziona e modifica il tuo schema come



- cambia il suo tipo di avvio e chiudi



9. Ora fai clic sul pulsante Esegui (se vuoi, puoi aggiungere breakpoint ai tuoi metodi didFinishLaunchingWithOptions e openURL per osservare i valori)

10. Verrà visualizzato un messaggio "In attesa di DeepLinkPOC (o il nome della tua app) da avviare".

11. Apri Safari e digita " **DeepLinking: //** " nella barra di ricerca per visualizzare il prompt "apri questa pagina in DeepLinkPOC" fai clic su Apri per avviare la tua app.

Spero che tu sappia come impostare il deep-linking per la tua app :)

Leggi Deep linking in iOS online: https://riptutorial.com/it/ios/topic/5173/deep-linking-in-ios

Capitolo 58: Delegati multicast

introduzione

Modello per l'aggiunta di funzionalità di multicasting ai controlli iOS esistenti. L'aggiunta del multicasting consente una maggiore chiarezza e riutilizzo del codice.

Examples

Delegati multicast per eventuali controlli

Inoltra messaggi di un oggetto a un altro dai delegati, trasmettendo in multicasting questi messaggi a più osservatori.

Passo 1: - Crea una **classe** `NSObject` di `RRMulticastDelegate`

Passaggio 2: - Implementazione del codice seguente nel file `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
```

Passaggio 3: - Implementazione del codice seguente nel file `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
```

Passaggio 4: - Creare una **classe** di **categoria** `NSObject` di `RRProperty`

Passaggio 5: - Implementazione del codice seguente nel file `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
```

Passaggio 6: - Implementazione del codice seguente nel file `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
```

Finalmente si usa multicast dalegate per qualsiasi controllo ...

Per esempio ...

Importa la classe viewcontroller nel file `NSObject+RRProperty.h` per accedere ai suoi metodi per **impostare il delegato / origine dati multicast** .

```
UITextView *txtView = [[UITextView alloc]initWithFrame:txtframe];
[txtView addDelegate:self];

UITableView *tblView = [[UITableView alloc]initWithFrame:tblframe];
[tblView addDelegate:self];
[tblView addDataSource:self];
```

Leggi Delegati multicast online: https://riptutorial.com/it/ios/topic/10081/delegati-multicast

Capitolo 59: DispatchGroup

introduzione

Argomenti correlati:

Grand Central Dispatch

Concorrenza

Examples

introduzione

Supponiamo che tu abbia più thread in esecuzione. Ogni thread sta facendo un compito. Si desidera ricevere una notifica sul mainThread OPPURE su un altro thread, una volta completati tutti i thread delle attività.

La soluzione più semplice a tale problema è un `DispatchGroup` .

Quando si utilizza un gruppo `DispatchGroup` , per ogni richiesta, si `enter` nel gruppo e per ogni richiesta completata si `leave` dal gruppo.

Quando non ci sono più richieste nel gruppo, sarai `notify` (notificato).

Uso:

```
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()
    }
}
```

Con quanto sopra, non è necessario `wait` all'infinito finché tutte le attività sono state completate. È possibile visualizzare un caricatore PRIMA di iniziare tutte le attività e chiudere il caricatore DOPO che tutte le attività sono state completate. In questo modo, il tuo thread principale non viene bloccato e il tuo codice rimane pulito.

Supponiamo ora che vogliate anche `ordered` i compiti o aggiungere le loro risposte a un array in sequenza. Potresti fare quanto segue:

```
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()
    }
}
```

Gli appunti

Ogni voce deve avere un'uscita in un `DispatchGroup` . Se ti dimentichi di `leave` dopo essere `entering` , ti stai preparando. Non sarai mai informato quando le attività saranno completate.

L'importo di `enter` deve essere uguale all'importo del `leave` .

Leggi DispatchGroup online: https://riptutorial.com/it/ios/topic/4624/dispatchgroup

Capitolo 60: Estensione per Rich Push Notification - iOS 10.

introduzione

iOS 10 ci ha fornito `UserNotifications.framework` , la nuova API per le notifiche locali / remote. Offre la visualizzazione degli allegati multimediali o risponde ai messaggi direttamente dalla notifica.

Il contenuto della notifica è composto da: titolo, sottotitolo, corpo e allegato. L'allegato può contenere immagini / gif / video fino a 50 mb.

Examples

Estensione del contenuto delle notifiche

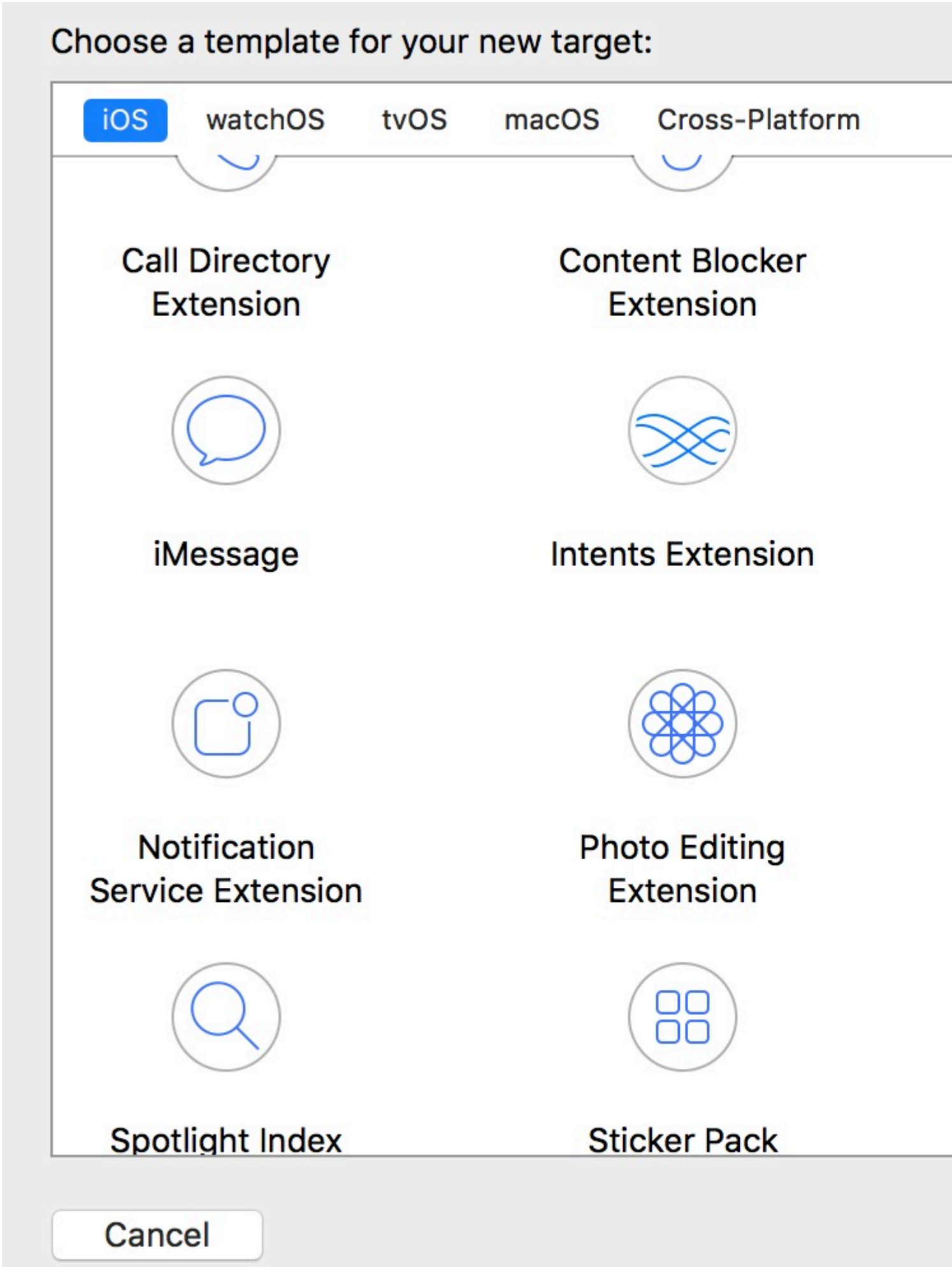
Perchè ne abbiamo bisogno?

L'estensione del contenuto ci aiuta a creare un'interfaccia utente personalizzata al momento della notifica.

Si utilizza questo framework per definire un'estensione che riceve i dati di notifica e fornisce la rappresentazione visiva corrispondente. La tua estensione può anche rispondere alle azioni personalizzate associate a tali notifiche.

Implementazione

1. Nella finestra di xCode `Navigator` vai alla sezione `Targets` . Premi `Add New Target` .
2. Selezionare il modello di `Notification Content Extension` :



3. Nel file `info.plist` , impostare l'identificatore per la chiave `UNNotificationExtensionCategory` :

5. Aggiungi una nuova `category key` e imposta il valore su ciò che abbiamo digitato nel file Info.plist (passaggio 3):
   e imposta il valore su ciò che abbiamo digitato nel file Info.plist (passaggio 3):

Spingere:

```
{
 aps: {
 alert: { … },
 category: 'io.swifting.notification-category'
 }
}
```

Locale:

```
let mutableNotificationContent = UNMutableNotificationContent()
mutableNotificationContent.category = "io.swifting.notification-category"
mutableNotificationContent.title = "Swifting.io Notifications"
mutableNotificationContent.subtitle = "Swifting.io presents"
mutableNotificationContent.body = "Custom notifications"
```

Controlla anche il riferimento API ufficiale:
https://developer.apple.com/reference/usernotificationsui/unnotificationcontentextension?utm_source=swi

Leggi Estensione per Rich Push Notification - iOS 10. online:
https://riptutorial.com/it/ios/topic/9501/estensione-per-rich-push-notification---ios-10-

Capitolo 61: EventKit

Examples

Richiesta di autorizzazione

La tua app non può accedere ai tuoi promemoria e al tuo calendario senza autorizzazione. Invece, deve mostrare un avviso all'utente, chiedendogli di concedere l'accesso agli eventi per l'app.

Per iniziare, importa il framework `EventKit` :

veloce

```
import EventKit
```

Objective-C

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

Creare un `EKEventStore`

Quindi, creiamo un oggetto `EKEventStore` . Questo è l'oggetto da cui possiamo accedere ai dati del calendario e dei promemoria:

veloce

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

Objective-C

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

> # Nota
>
> Creare un oggetto `EKEventStore` ogni volta che è necessario accedere al calendario non è efficiente. Prova a farlo una volta e usalo ovunque nel tuo codice.

Controllando la disponibilità

La disponibilità ha tre diversi stati: Autorizzato, Rifiutato e Non determinato. Non determinato significa che l'app deve concedere l'accesso.

Per verificare la disponibilità, utilizziamo il metodo `authorizationStatusForEntityType()` dell'oggetto `EKEventStore` :

veloce

```
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;
}
```

Richiesta di autorizzazione

Inserisci il seguente codice nel caso `NotDetermined` :

veloce

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

Accesso a diversi tipi di calendari

Accesso alla serie di calendari

Per accedere alla matrice di `EKCalendar` s, usiamo il metodo `calendarsForEntityType` :

veloce

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

Iterare attraverso i calendari

Basta usare un ciclo `for` semplice:

veloce

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

Accesso al titolo e al colore del calendario

veloce

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

Objective-C

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

Aggiungere un evento

Creare l'oggetto evento

veloce

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

Objective-C

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

Impostazione calendario, titolo e date correlati

veloce

```
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
```

Aggiunta di eventi al calendario

veloce

```
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
}
```

Leggi EventKit online: https://riptutorial.com/it/ios/topic/5854/eventkit

Capitolo 62: FacebookSDK

Examples

Integrazione con FacebookSDK

Passaggio 1: installare l'SDK

È possibile installare l'SDK manualmente o tramite `CocoaPods` . Quest'ultima opzione è altamente raccomandata.

Metti queste righe in `Podfile` :

```
target 'MyApp' do
  use_frameworks!

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

Esegui l' `pod install` nel terminale e apri `.xcworkspace` invece di `.xcodeproj` .

`FBSDKLoginKit` e `FBSDKShareKit` sono opzionali. Potresti o non potresti averne bisogno.

Passaggio 2: crea un'app su Facebook

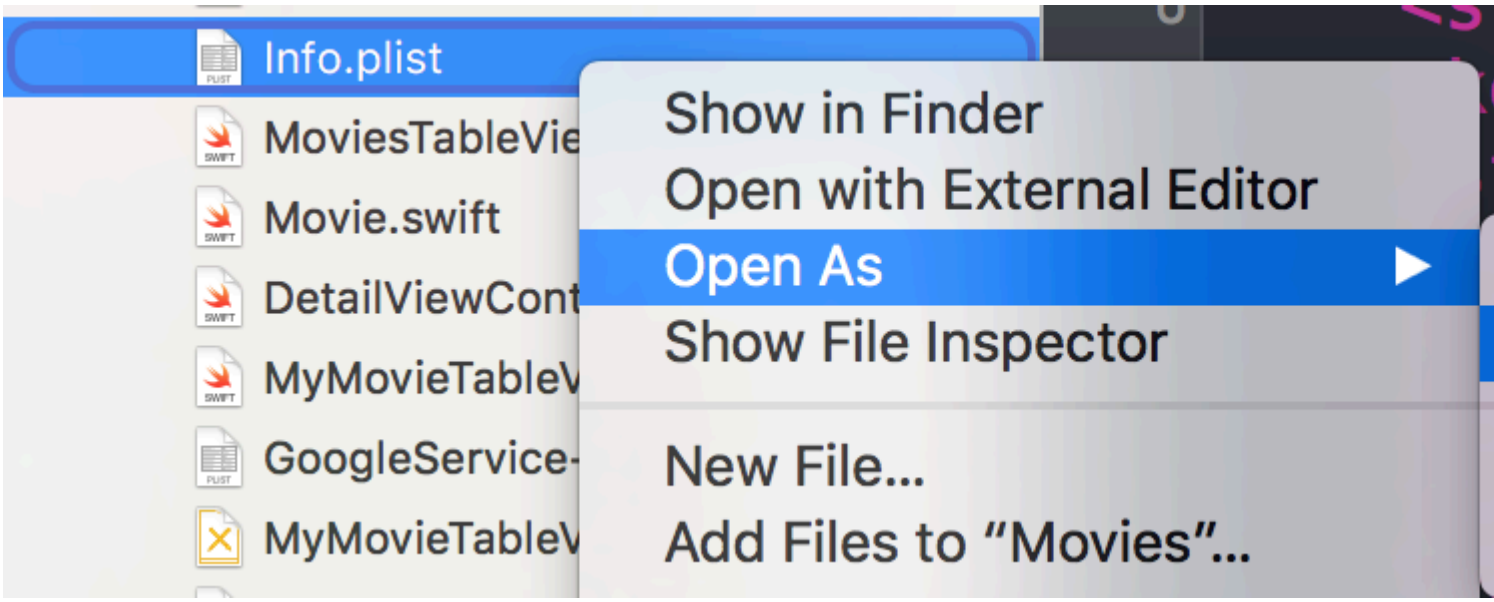
Vai a Avvio rapido - Facebook per sviluppatori per creare un'app.

Facebook ti chiederà di scaricare l'SDK dopo aver creato l'app. Puoi saltare questa parte se hai già installato l'SDK tramite CocoaPods.

Passaggio 3: modifica `.plist`

un. Per rendere la tua app in grado di "comunicare" con Facebook, devi inserire alcune impostazioni nel tuo file `.plist` . Facebook ti darà lo snippet personalizzato nella pagina Avvio rapido.

b. Modifica il tuo file `.plist` come codice sorgente.



c. Incolla lo snippet personalizzato nel codice sorgente. **Stai attento!** Lo snippet deve essere esattamente il figlio del tag `<dict>` . Il tuo codice sorgente dovrebbe essere qualcosa del tipo:

```
<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>
```

Se incolli lo snippet in un posto sbagliato, ti imbatterai in problemi.

Passaggio 4: comunica a Facebook l'identificatore del gruppo nella pagina Avvio rapido.

=> Come ottenere l'identificatore del gruppo

Passaggio 5: modifica il tuo `AppDelegate.swift`

un.

```
import FBSDKCoreKit
```

b.

```
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)
}
```

Crea il tuo pulsante personalizzato "Accedi con Facebook"

A volte vogliamo progettare la nostra interfaccia utente per il pulsante "Accedi con Facebook" invece del pulsante originale fornito con FacebookSDK.

1. Nello storyboard, trascina il tuo UIButton e impostalo come vuoi tu.
2. Ctrl + trascina il pulsante sul controller di visualizzazione come IBAction.
3. **All'interno** del metodo IBAction avrai simulato un tap sul pulsante Facebook reale come segue:

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];
```

Hai finito.

Recupero dei dati dell'utente di Facebook

Dopo che l'utente ha effettuato l'accesso a Facebook nella tua app, ora è il momento di recuperare i dati richiesti su `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
                        }
                    }
                }
            }
        }
    })
}
```

Leggi FacebookSDK online: https://riptutorial.com/it/ios/topic/2972/facebooksdk

Capitolo 63: FileHandle

introduzione

Leggi il file in blocchi dalla directory del documento

Examples

Leggi il file dalla directory del documento in blocchi

Ottengo il percorso del file dalla directory del documento e leggo quel file in blocchi di 1024 e salva (aggiungi) all'oggetto `NSMutableData` oppure puoi scrivere direttamente sul socket.

```
// 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)")
        }
    }
}
```

Dopo aver completato la lettura del file, otterrai i dati del file nella variabile `ReadData` Qui outputFileHandle è un oggetto di `FileHandle`

```
var outputFileHandle:FileHandle?
```

Leggi FileHandle online: https://riptutorial.com/it/ios/topic/10665/filehandle

Capitolo 64: Filtri CoreImage

Examples

Esempio di filtro immagine di base

Objective-C

Basta loggare per vedere come usare un particolare filtro

```
 NSArray *properties = [CIFilter filterNamesInCategory:kCICategoryBuiltIn];

 for (NSString *filterName in properties)
 {
 CIFilter *fltr = [CIFilter filterWithName:filterName];
 NSLog(@"%@", [fltr attributes]);
 }
```

Nel caso di CISepiaTone il registro di sistema è il seguente

```
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;
    };
}
```

Utilizzando il registro di sistema sopra abbiamo impostato il filtro come di seguito:

```
    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);
```

Immagine generata dal codice precedente

Un modo alternativo per impostare un filtro

```
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];
```

Immagine generata da questo codice

Tutti i filtri disponibili sono i seguenti

```
/* 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*/
```

Leggi Filtri CoreImage online: https://riptutorial.com/it/ios/topic/7278/filtri-coreimage

Capitolo 65: Firma del codice

Examples

Profili di provisioning

Per creare un file IPA in XCode, è necessario firmare l'applicazione con un certificato e un profilo di provisioning. Questi possono essere creati su https://developer.apple.com/account/ios/profile/create

Tipi di profilo di provisioning

I profili di provisioning sono suddivisi in due tipi: Sviluppo e Distribuzione:

Sviluppo

- Sviluppo app iOS / Sviluppo app tvOS - utilizzato in fase di sviluppo per installare l'app su un dispositivo di test.

Distribuzione

- App Store / tvOS App Store: utilizzato per firmare la tua applicazione per il caricamento dell'app store.
- In House - Usato per la distribuzione aziendale della tua app, ai dispositivi all'interno della tua azienda.
- Ad Hoc / tvOS Ad Hoc - Utilizzato per distribuire l'app su un numero limitato di dispositivi specifici (ad esempio, è necessario conoscere gli UDID dei dispositivi su cui si desidera installare l'app).

Leggi Firma del codice online: https://riptutorial.com/it/ios/topic/6055/firma-del-codice

Capitolo 66: GameplayKit

Examples

Generare numeri casuali

Anche se `GameplayKit` (introdotto con iOS 9 SDK) riguarda l'implementazione della logica di gioco, potrebbe anche essere utilizzato per generare numeri casuali, il che è molto utile in app e giochi.

Oltre al `GKRandomSource.sharedRandom` che viene utilizzato nei seguenti capitoli ci sono tre tipi aggiuntivi di `GKRandomSource` 'out of the box.

- **GKARC4RandomSource** Che utilizza l'algoritmo ARC4
- **GKLinearCongruentialRandomSource** Che è un veloce ma non così casuale `GKRandomSource`
- **GKMersenneTwisterRandomSource** Che implementa un algoritmo MersenneTwister. È più lento ma più casuale.

Nel capitolo seguente usiamo solo il metodo `nextInt()` di `GKRandomSource` . In aggiunta a questo c'è il `nextBool() -> Bool` e il `nextUniform() -> Float`

Generazione

Innanzitutto, importa `GameplayKit` :

veloce

```
import GameplayKit
```

Objective-C

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

Quindi, per generare un numero casuale, utilizzare questo codice:

veloce

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

Objective-C

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

Nota

La funzione nextInt (), se usata senza parametri, restituirà un numero casuale compreso tra -2,147,483,648 e 2.147.483.647, inclusi se stessi, quindi non siamo sicuri che sia sempre un numero positivo o diverso da zero.

Generando un numero da 0 a n

Per ottenere ciò, devi dare n al metodo `nextIntWithUpperBound()` :

veloce

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

Objective-C

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

Questo codice ci darà un numero compreso tra 0 e 10, inclusi se stessi.

Generare un numero da m a n

Per fare questo si crea un oggetto `GKRandomDistribution` con un `GKRandomSource` e si passa ai limiti. Una `GKRandomDistribution` può essere utilizzata per modificare il comportamento di distribuzione come `GKGaussianDistribution` o `GKShuffledDistribution` .

Successivamente l'oggetto può essere utilizzato come ogni normale `GKRandomSource` poiché implementa anche il protocollo `GKRandom` .

veloce

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

Obiettivo-C *obsoleto*

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

Ad esempio, per generare un numero casuale compreso tra 3 e 10, si utilizza questo codice:

veloce

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

Obiettivo-C *obsoleto*

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

GKEntity e GKComponent

Un'entità rappresenta un oggetto di un gioco come una figura del giocatore o una figura nemica. Poiché questo oggetto non fa molto senza braccia e gambe, possiamo aggiungere i componenti a questo. Per creare questo sistema, Apple ha le classi `GKEntity` e `GKComponent` .

Supponiamo di avere la seguente classe per i seguenti capitoli:

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

GKEntity

Un'entità è una raccolta di componenti e offre diverse funzioni per aggiungere, rimuovere e interagire con componenti di essa.

Mentre potremmo semplicemente usare GKEntity è normale sottoclassi per un tipo specifico di entità di gioco.

È importante che sia possibile aggiungere un componente di una classe solo una volta. Se aggiungi un secondo componente della stessa classe, sostituirà il primo componente esistente all'interno di `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
```

Potresti chiedere perché. La ragione di ciò sono i metodi chiamati `component(for: T.Type)` che restituiscono il componente di un tipo specifico dell'entità.

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

Oltre ai metodi-componenti, ha un metodo di `update` che viene utilizzato per delegare il delta time o l'ora corrente della logica di gioco ai suoi componenti.

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

GKComponent

Un componente rappresenta qualcosa di un'entità, ad esempio il componente visivo o il componente logico.

Se viene chiamato un metodo di aggiornamento di un'entità, lo delegherà a tutti i suoi componenti. L'override di questo metodo viene utilizzata per manipolare un'entità.

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

Oltre a questo è possibile sovrascrivere il metodo `didAddToEntity` e `willRemoveFromEntity` per informare altri componenti sulla sua rimozione o aggiunta.

Per manipolare un altro componente all'interno di un componente è possibile ottenere GKEntity a cui è aggiunto il componente.

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

Mentre questo è possibile, **non** è un modello comune, poiché fili i due componenti insieme.

GKComponentSystem

Mentre abbiamo appena parlato dell'utilizzo del meccanismo delegato di aggiornamento di `GKEntity` per aggiornare `GKComponents` c'è un modo diverso per aggiornare `GKComponents` che si chiama `GKComponentSystem` .

Viene utilizzato nel caso sia necessario che tutti i componenti di un tipo specifico debbano essere aggiornati in un colpo solo.

Un `GKComponentSystem` viene creato per un tipo specifico di componente.

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

Per aggiungere un componente puoi usare il metodo add:

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

Ma un modo più comune è passare l'entità creata con i suoi componenti a `GKComponentSystem` e troverà un componente corrispondente all'interno dell'entità.

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

Per aggiornare tutti i componenti di un tipo specifico chiama l'aggiornamento:

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

Nel caso in cui si desideri utilizzare `GKComponentSystem` anziché un meccanismo di aggiornamento basato sull'entità, è necessario disporre di un `GKComponentSystem` per ogni componente e chiamare l'aggiornamento su tutti i sistemi.

Leggi GameplayKit online: https://riptutorial.com/it/ios/topic/4966/gameplaykit

Capitolo 67: GCD (Grand Central Dispatch)

introduzione

Grand Central Dispatch (GCD) è la risposta di Apple al multithreading. È un framework leggero per l'esecuzione di attività in modo sincrono o asincrono nelle code e gestisce i thread della CPU per te dietro le quinte.

Argomento correlato: concorrenza

Examples

Crea una coda di spedizione

Puoi creare la tua coda utilizzando `dispatch_queue_create`

Objective-C

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

veloce

```
// 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
```

Ottenere la coda principale

La coda principale è la coda di invio in cui si verificano tutti gli aggiornamenti dell'interfaccia utente e viene inserito il codice che comporta modifiche all'interfaccia utente.

È necessario raggiungere la coda principale per aggiornare l'interfaccia utente al termine di un processo asincrono come `NSURLSession`

Esistono due tipi di chiamate di coda principali `synchronous` e `asynchronous` . Quando invochi qualcosa in modo `synchronously` , significa che il thread che ha avviato quell'operazione attenderà che l'attività finisca prima di continuare. `Asynchronous` significa che non aspetterà.

Codice Obiettivo-C

Chiamata di coda principale `Synchronous`

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

Chiamata di coda principale `Asynchronous`

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

SWIFT 3

Chiamata di coda principale `Asynchronous`

```
DispatchQueue.main.async {

}
```

Chiamata di coda principale `Synchronous`

```
DispatchQueue.main.sync {

}
```

Gruppo di spedizione

> DispatchGroup consente la sincronizzazione aggregata del lavoro. Puoi usarli per inviare più articoli di lavoro diversi e tracciare quando tutti sono completi, anche se potrebbero essere eseguiti su code diverse. Questo comportamento può essere utile quando non è possibile eseguire progressi finché tutte le attività specificate non sono state completate.

Uno scenario in cui questo può essere utile è se hai più chiamate al webservice che devono essere completate prima di continuare. Ad esempio, è necessario scaricare più set di dati che devono essere elaborati da una funzione. È necessario attendere il completamento di tutti i servizi Web prima di chiamare la funzione per elaborare tutti i dati ricevuti.

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!")
```

```
}
```

In alternativa, se non si desidera attendere il completamento dei gruppi, ma si desidera eseguire una funzione al termine di tutte le attività, utilizzare la funzione di `notify` al posto di `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
}
```

Esempio di output:

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

Per maggiori informazioni, consultare Apple Docs o l' argomento correlato

Dispatch Semaphore

> DispatchSemaphore fornisce un'implementazione efficiente di un semaforo di conteggio tradizionale, che può essere utilizzato per controllare l'accesso a una risorsa attraverso più contesti di esecuzione.

Uno scenario per quando utilizzare un semaforo potrebbe essere se stai facendo un po 'di lettura / scrittura di file, se più attività stanno cercando di leggere e scrivere dal file allo stesso tempo, potrebbe aumentare le tue prestazioni per far sì che ogni attività attenda il suo turno così come non sovraccaricare il controller I / O.

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
        }
    }
}
```

Esempio di output: (notare i timestamp)

```
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
```

Per maggiori informazioni, fare riferimento a Apple Docs

Code di invio seriale vs simultanee

Swift 3

Coda seriale

```
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)")
        }
    }
}
```

Esempio di output:

```
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
```

Coda concomitante

```
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)")
        }
    }
}
```

Esempio di output:

```
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
```

Discussione

Come possiamo vedere dagli esempi sopra, una coda seriale completerà ogni attività nell'ordine in cui sono inviati alla coda. Ogni attività attenderà il completamento dell'attività precedente prima dell'esecuzione. Per quanto riguarda la coda concorrente, ogni attività non attende gli altri nella coda ed esegue il prima possibile; il vantaggio è che tutte le attività in coda verranno eseguite contemporaneamente su thread separati, rendendo la coda simultanea meno tempo di una coda seriale.

Se l'ordine di esecuzione delle attività non è importante, utilizzare sempre una coda concorrente per la migliore efficienza.

Leggi GCD (Grand Central Dispatch) online: https://riptutorial.com/it/ios/topic/4626/gcd--grand-central-dispatch-

Capitolo 68: Gestione degli schemi URL

Sintassi

1. // metodo **canOpenURL** verifica se esiste un'app in grado di gestire lo schema URL indicato.
2. // Swift

   UIApplication.sharedApplication (). CanOpenURL (_ aUrl: NSURL)
3. // Obiettivo-C

   [[UIApplication sharedApplication] canOpenURL: (NSURL *) aUrl];
4. // metodo **openURL** tenta di aprire una risorsa localizzata dall'URL. SÌ / vero se è stato aperto altrimenti NO / falso.
5. // Swift

   UIApplication.sharedApplication (). OpenURL (_ aUrl: NSURL)
6. // Obiettivo-C

   [[UIApplication sharedApplication] openURL: (NSURL *) aUrl];

Parametri

Parametro	Senso
aUrl	un'istanza NSURL che memorizza una stringa di schema incorporata o personalizzata

Osservazioni

In iOS9 e versioni successive la tua app deve elencare qualsiasi schema URL che vorrà interrogare. Questo viene fatto aggiungendo `LSApplicationQueriesSchemes` a Info.plist

---

iOS ha il supporto integrato per gli schemi `tel` , `http` / `https` , `sms` , `mailto` , `facetime` . Supporta anche gli URL basati su http per le app di `Youtube` , `Maps` e `iTunes` .

Esempi di schemi di URL incorporati:

tel : `tel://123456890` o `tel:123456890`

http : `http://www.google.com`

facetime : `facetime://azimov@demo.com`

mailto : `mailto://azimov@demo.com`

sms : `sms://123456890` o `sms:123456890`

Youtube : `https://www.youtube.com/watch?v=-eCaif2QKfA`

Mappe :

- Utilizzando l'indirizzo: `http://maps.apple.com/?address=1,Infinite+Loop,Cupertino,California`
- Utilizzando le coordinate: `http://maps.apple.com/?ll=46.683155557,6.683155557`

iTunes : `https://itunes.apple.com/us/artist/randy-newman/id200900`

Nota : non tutti i caratteri speciali sono supportati nello schema `tel` (ad esempio `*` o `#`). Ciò avviene a causa di problemi di sicurezza per impedire agli utenti di reindirizzare le chiamate non autorizzate, pertanto in questo caso l'app `Phone` non verrà aperta.

Examples

Utilizzo dello schema URL incorporato per aprire l'app Mail

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");
}
```

Schemi URL Apple

Si tratta di schemi URL supportati da app native su iOS, OS X e watchOS 2 e versioni successive.

Link di apertura in 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)
}
```

Avvio di una conversazione telefonica

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>
```

Avvio di una conversazione 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>
```

Apertura Messaggi App per comporre un sms al destinatario:

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>
```

Aprire l'app Mail per comporre un'email al destinatario:

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>
```

È inoltre possibile includere un campo oggetto, un messaggio e più destinatari nei campi A, Cc e Ccn. (In iOS, l'attributo from viene ignorato.) L'esempio seguente mostra un URL mailto che include diversi attributi:

```
mailto:foo@example.com?cc=bar@example.com&subject=Greetings%20from%20Cupertino!&body=Wish%20you%20were
```

Nota: la finestra di dialogo di composizione della posta elettronica può essere presentata anche all'interno dell'app utilizzando `MFMailComposeViewController` .

Leggi Gestione degli schemi URL online: https://riptutorial.com/it/ios/topic/3646/gestione-degli-

schemi-url

Capitolo 69: Gestire la tastiera

Examples

Scorrimento di un UIScrollView / UITableView durante la visualizzazione della tastiera

Ci sono pochi approcci disponibili qui:

1. Puoi iscriverti alle notifiche degli eventi di apparizione della tastiera e modificare manualmente l'offset:

```
//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. O utilizzare soluzioni già pronte come TPKeyboardAvoidingTableView o TPKeyboardAvoidingScrollView https://github.com/michaeltyson/TPKeyboardAvoiding

Ignora una tastiera con tocco sulla vista

Se vuoi nascondere una tastiera toccandola al di fuori di essa, è possibile usare questo trucco hacky (funziona solo con 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];
}
```

per Swift ci sarà un po 'più di codice:

```
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)
}
```

Un altro esempio di 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
    }
}
```

Crea una tastiera in-app personalizzata

Questa è una tastiera in-app di base. Lo stesso metodo potrebbe essere utilizzato per creare praticamente qualsiasi layout di tastiera. Ecco le principali cose che devono essere fatte:

- Crea il layout della tastiera in un file .xib, il cui proprietario è una classe Swift o Objective-C che è una sottoclasse `UIView` .
- Dì a `UITextField` di usare la tastiera personalizzata.
- Utilizzare un delegato per comunicare tra la tastiera e il controller della vista principale.

Creare il file di layout della tastiera .xib

- In Xcode, vai su **File> Nuovo> File ...> iOS> Interfaccia utente> Visualizza** per creare il file .xib.
- Ho chiamato il mio Keyboard.xib
- Aggiungi i pulsanti di cui hai bisogno.
- Utilizza i vincoli di layout automatico in modo che, indipendentemente dalle dimensioni della tastiera, i pulsanti vengano ridimensionati di conseguenza.
- Impostare il proprietario del file (non la vista root) come classe della `Keyboard` . Questa è una fonte comune di errore. Creerai questo corso nel passaggio successivo. Vedi la nota alla fine.

Creare il file della tastiera della sottoclasse .swift UIView

- In Xcode vai su **File> Nuovo> File ...> iOS> Sorgente> Cocoa Touch Class** per creare la **classe** Swift o Objective-C. Scegli `UIView` come superclasse per la classe appena creata
- Ho chiamato il mio `Keyboard.swift` (classe `Keyboard` in Objective-C)
- Aggiungi il seguente codice per 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
      }

  }
```

- Aggiungere il seguente codice per Objective-C:

Keyboard.h File

```
#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
```

Keyboard.m File

```
#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
```

- Controlla le azioni di trascinamento dai pulsanti per richiamare i pulsanti nel file .xib e il metodo `@IBAction` nel proprietario Swift o Objective-C per agganciarli tutti.
- Si noti che il protocollo e il codice del delegato. Vedi questa risposta per una semplice spiegazione su come funzionano i delegati.

Configura il View Controller

- Aggiungi un `UITextField` allo storyboard principale e collegalo al controller di visualizzazione con un `IBOutlet` . Chiamalo `textField` .
- Usa il seguente codice per il View Controller in 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)
      }
  }
```

- Utilizzare il seguente codice per Objective-C:

.h File

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

```
@interface ViewController : UIViewController

@end
```

.m File

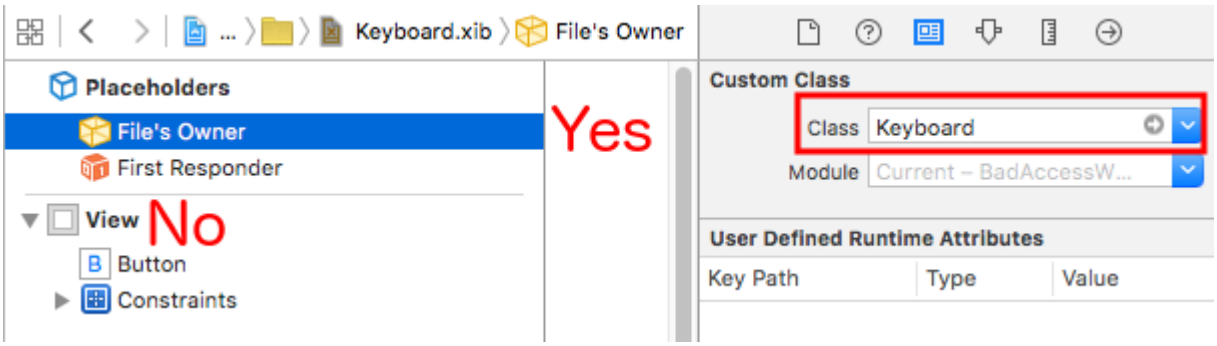
```
#import "ViewController.h"
#import "Keyboard.h"

@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
```

- Si noti che il controller della vista adotta il protocollo `KeyboardDelegate` che abbiamo definito sopra.

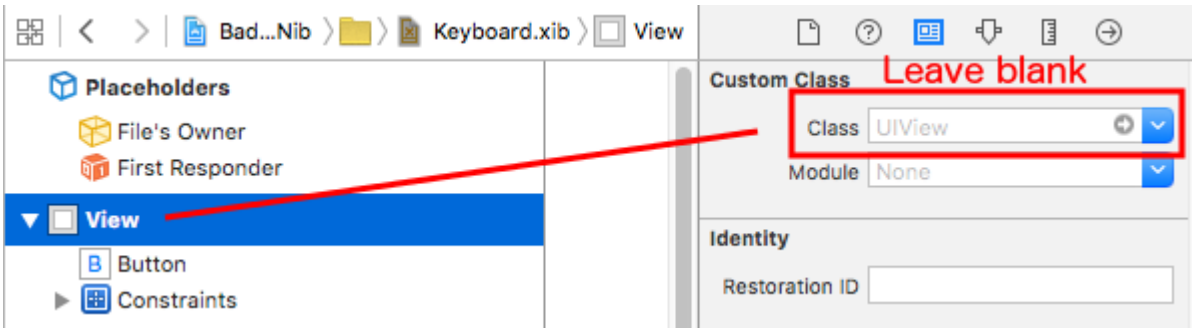
Errore comune

Se si verifica un errore `EXC_BAD_ACCESS` , è probabilmente perché si imposta la classe personalizzata della vista come `Keyboard` anziché eseguire questa operazione per il proprietario del file del pennino.

Seleziona `Keyboard.nib` e quindi scegli Proprietario file.



Assicurarsi che la classe personalizzata per la vista principale sia vuota.



Gli appunti

Questo esempio proviene originariamente da questa risposta di Overflow dello stack .

Gestione della tastiera mediante un delegato + un delegato

Quando ho iniziato a gestire la tastiera, usavo le notifiche separate in ogni ViewController.

Metodo di notifica (utilizzando 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)
    }
}
```

Il mio problema era che mi sono ritrovato a scrivere questo codice ancora e ancora per ogni singolo ViewController. Dopo aver sperimentato un po 'ho scoperto che usare un pattern Singleton + Delegate mi ha permesso di riutilizzare un po' di codice e organizzare tutta la gestione della tastiera in un unico posto!

Singleton + Delegate Method:

```
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)
    }
}
```

Ora, quando voglio gestire la tastiera da un ViewController, tutto ciò che devo fare è impostare il delegato su quel ViewController e implementare qualsiasi metodo delegato.

```
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)
    }
}
```

Anche questo metodo è molto personalizzabile! Supponiamo di voler aggiungere funzionalità per `UIKeyboardWillHideNotification` . Questo è facile come aggiungere un metodo al nostro `KeyboardManagerDelegate` .

`KeyboardManagerDelegate` con `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)
    }
}
```

Supponiamo di voler implementare `func keyboardWillHide(notificationUserInfo: [NSObject: AnyObject])` in un ViewController. Possiamo anche rendere questo metodo opzionale.

```
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])
}
```

* Nota questo modello aiuta a evitare l'uso eccessivo di `@objc` . Vedi http://www.jessesquires.com/avoiding-objc-in-swift/ per maggiori dettagli!

In sintesi, ho trovato che usare un delegato Singleton + per gestire la tastiera è sia più efficiente che più facile da usare rispetto all'utilizzo di Notifiche

Spostando la vista verso l'alto o verso il basso quando è presente la tastiera

Nota: funziona solo con la tastiera integrata fornita da iOS

SWIFT:

Affinché la vista di un **UIViewController** possa aumentare l'origine del frame quando viene presentato e diminuirlo quando è nascosto, aggiungere le seguenti funzioni alla classe:

```
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
        }
    }
}
```

E nel metodo `viewDidLoad()` della tua classe, aggiungi i seguenti osservatori:

```
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)
```

E questo funzionerà per qualsiasi dimensione dello schermo, usando la proprietà height della tastiera.

Objective-C:

Per fare la stessa cosa in Objective-C, questo codice può essere usato:

```
- (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;
    }];
}
```

Leggi Gestire la tastiera online: https://riptutorial.com/it/ios/topic/436/gestire-la-tastiera

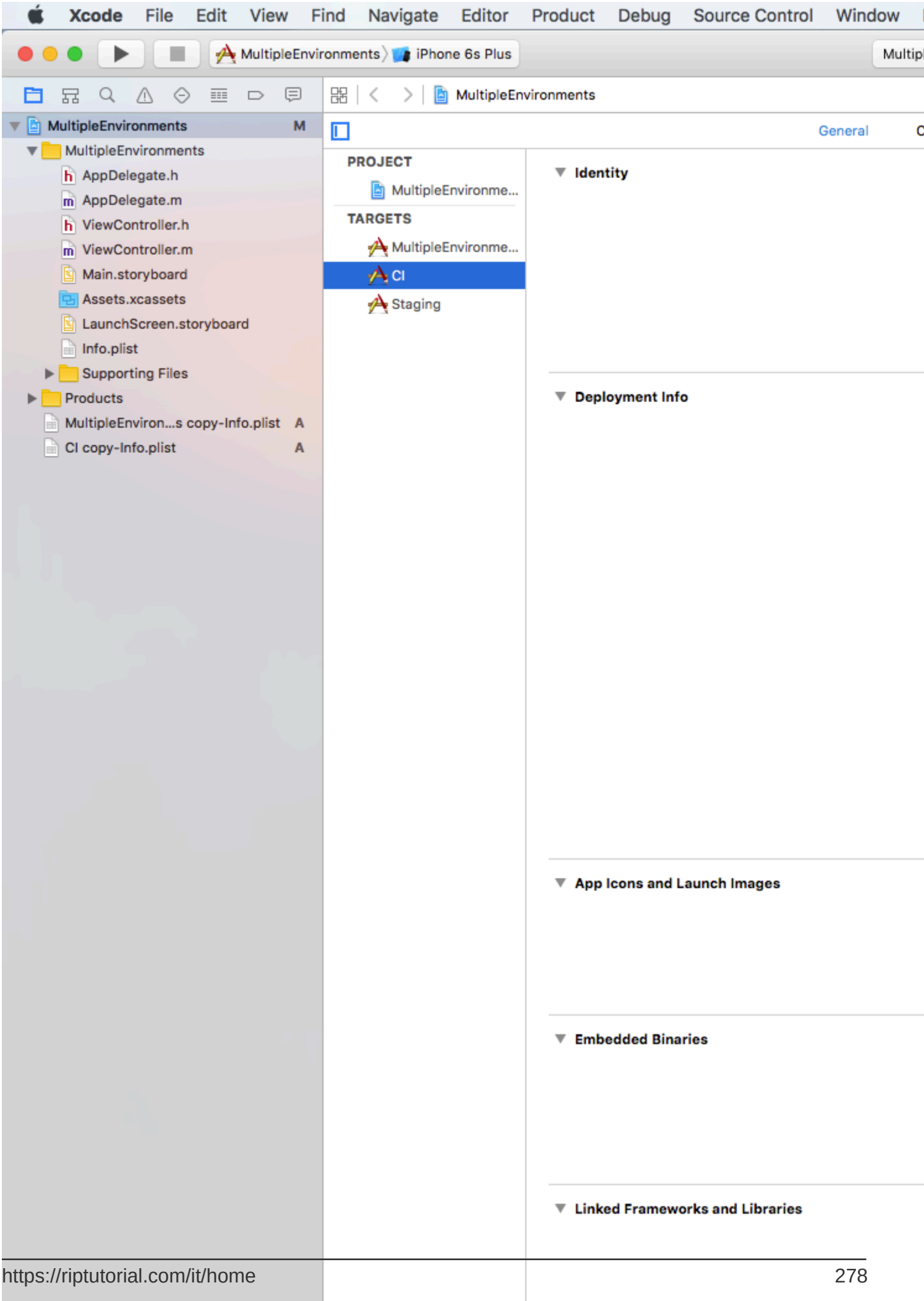
Capitolo 70: Gestire più ambienti utilizzando la macro

Examples

Gestire più ambienti utilizzando più target e macro

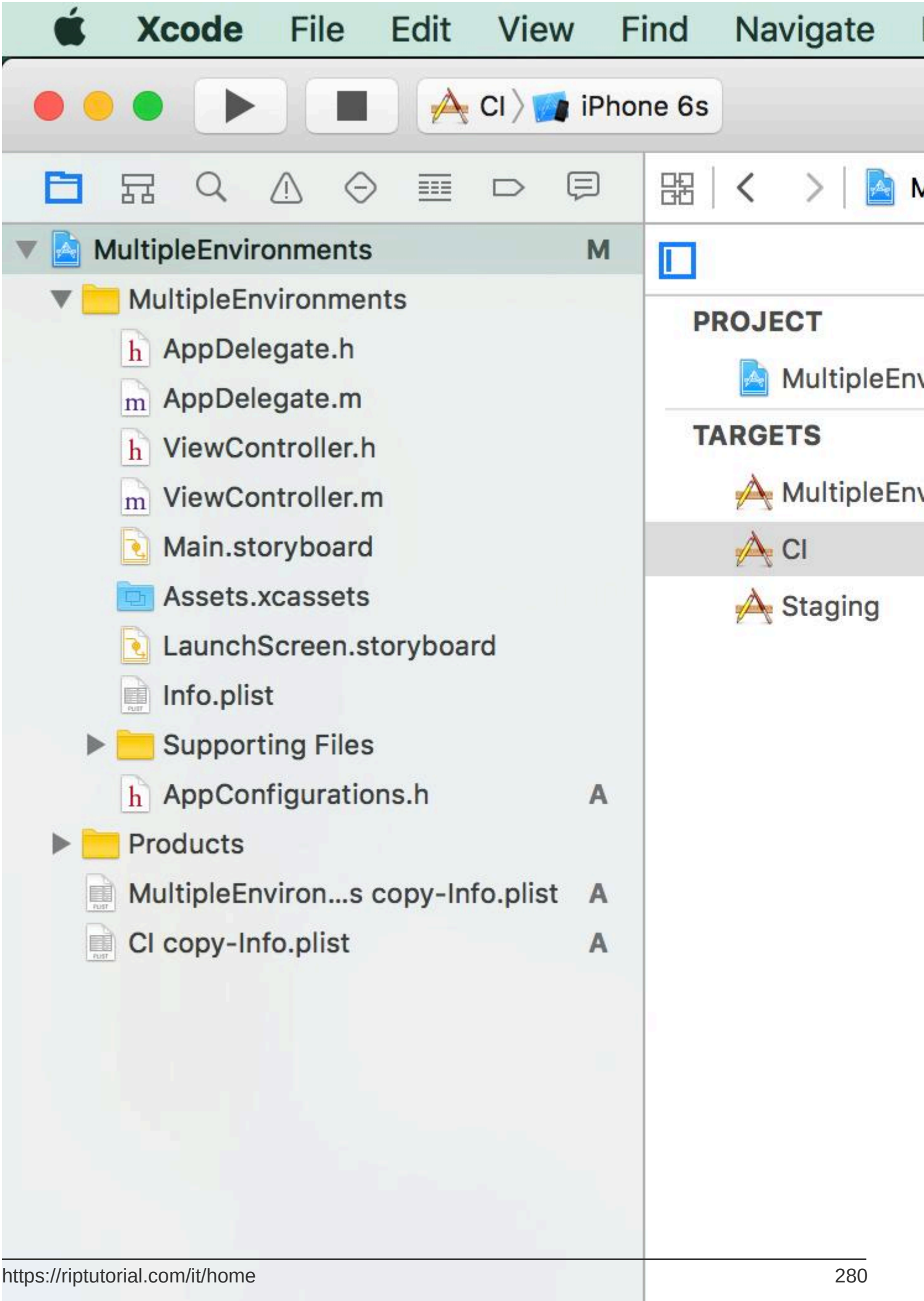
Ad esempio, abbiamo due ambienti: CI - Staging e vogliamo aggiungere alcune personalizzazioni per ogni ambiente. Qui proverò a personalizzare l'URL del server, il nome dell'app.

Innanzitutto, creiamo due target per 2 ambienti duplicando l'obiettivo principale:

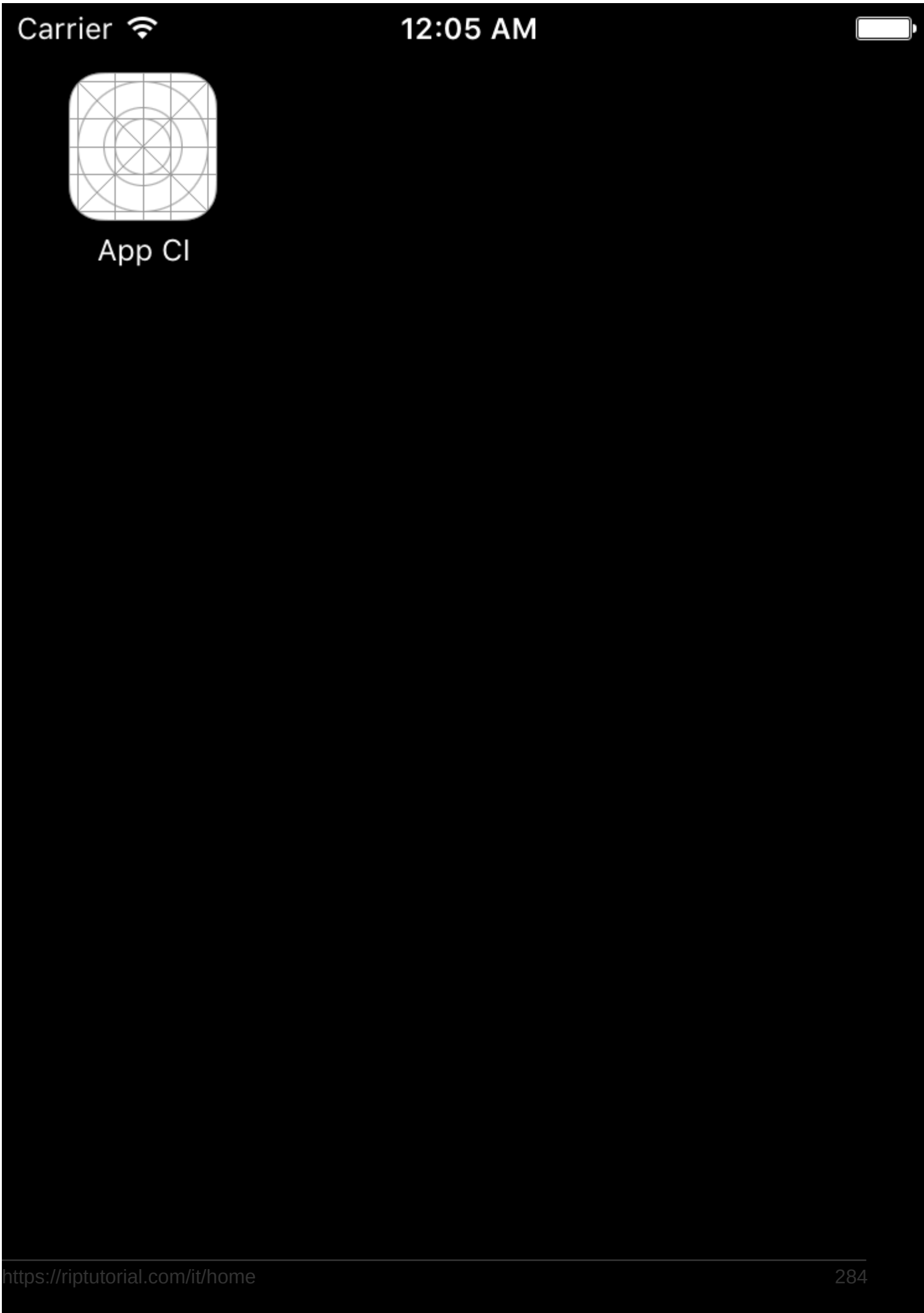

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window
MultipleEnvironments iPhone 6s Plus
MultipleEnvironments
MultipleEnvironments M
MultipleEnvironments
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Supporting Files
Products
MultipleEnviron...s copy-Info.plist A
CI copy-Info.plist A
MultipleEnvironments
General
PROJECT
MultipleEnvironme...
TARGETS
MultipleEnvironme...
CI
Staging
Identity
Deployment Info
App Icons and Launch Images
Embedded Binaries
Linked Frameworks and Libraries

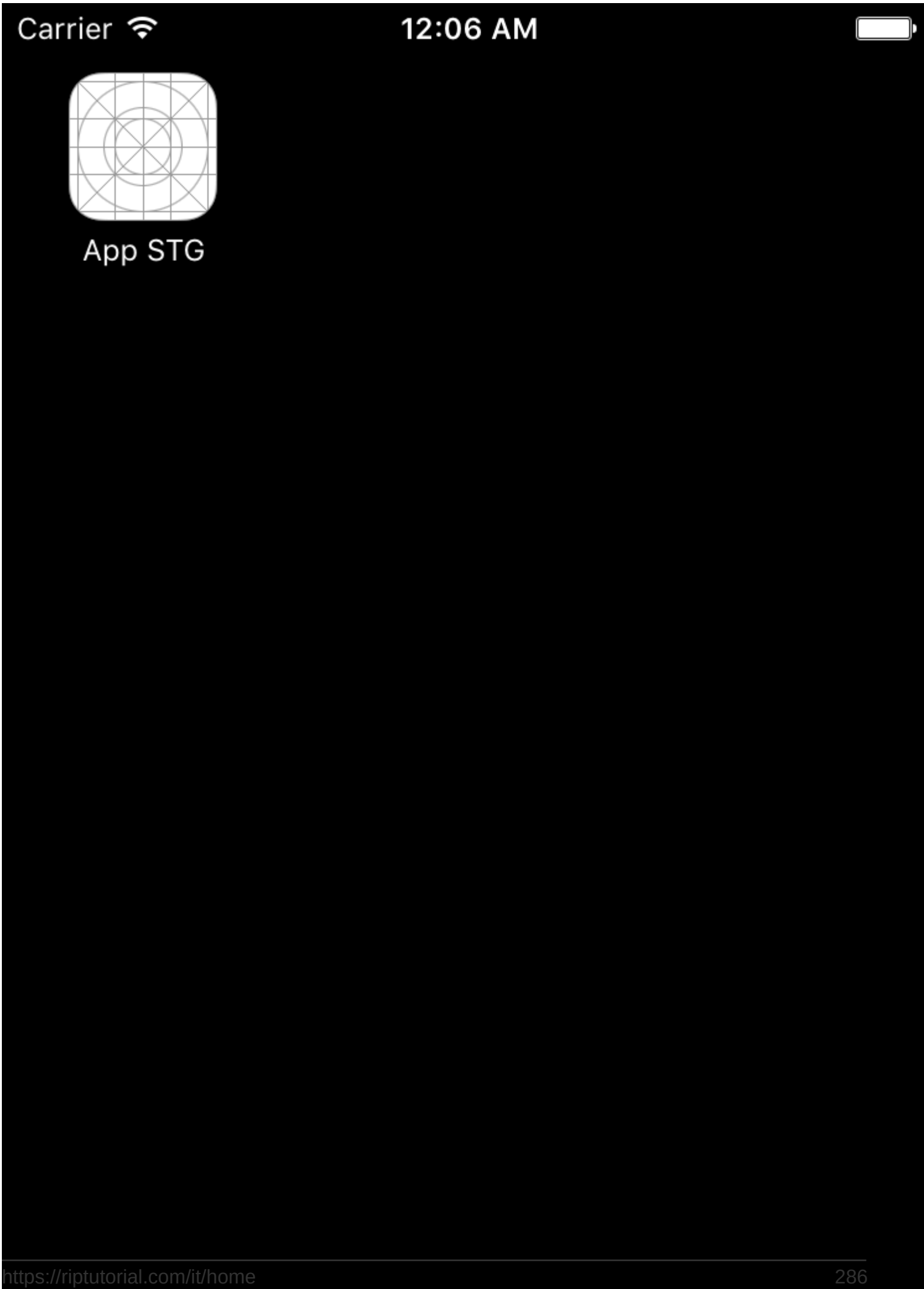
- Se eseguiamo / archiviamo utilizzando la destinazione CI, SERVER_URL è http://ci.api.example.com/ http://ci.api.example.com/
- Se eseguiamo / archiviamo utilizzando la destinazione STAGING, SERVER_URL è http://stg.api.example.com/

Se vuoi personalizzare di più, ad esempio: modifica il nome dell'app per ciascun target:


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


Carrier
12:03 AM
MultipleEn...


Carrier
12:05 AM
App CI


Carrier
12:06 AM
App STG

https://riptutorial.com/it/ios/topic/6849/gestire-piu-ambienti-utilizzando-la-macro

Capitolo 71: Grafico (Coreplot)

Examples

Realizzare grafici con CorePlot

Core Plot fornisce un podspec, quindi puoi usare cocoapods come gestore della libreria che dovrebbe rendere l'installazione e l'aggiornamento molto più semplici

Installa i cocoapod sul tuo sistema

Nella directory del progetto aggiungi un file di testo al tuo progetto chiamato Podfile digitando `pod init` nella directory del tuo progetto

Nel Podfile aggiungi il pod della linea 'CorePlot', '~> 1.6'

Nel terminale, cd nella directory del progetto ed esegui l'installazione di pod

Cocoapods genererà un file xcworkspace, che dovresti usare per avviare il tuo progetto (il file .xcodeproj non includerà le librerie di pod)

Apri il .xcworkspace generato da CocoaPods

Nel file ViewController.h

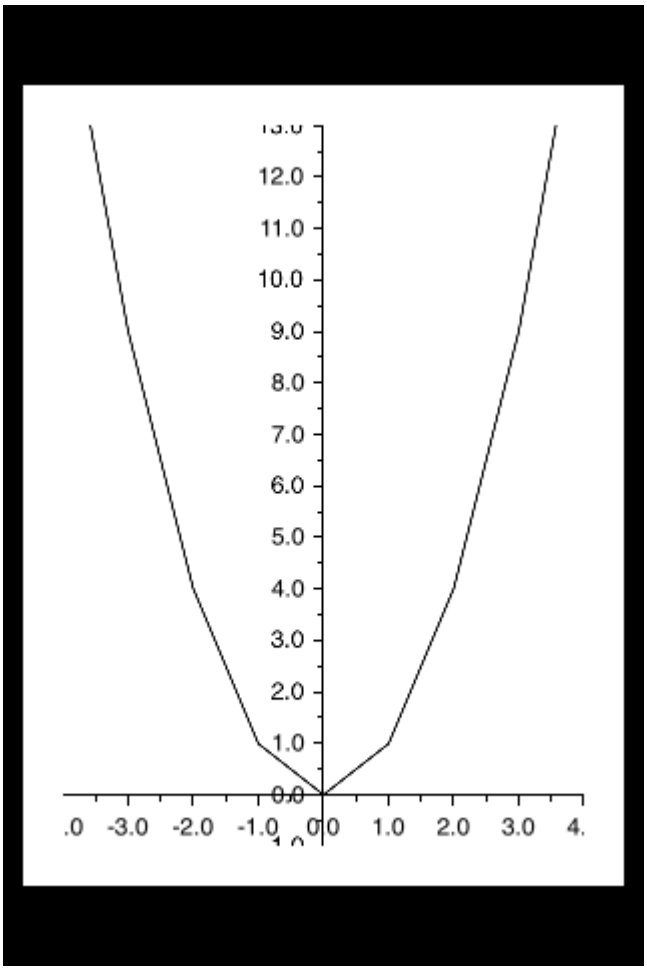
```
#import <CorePlot/ios/CorePlot.h>
//#import "CorePlot-CocoaTouch.h" or the above import statement
@interface ViewController : UIViewController<CPTPlotDataSource>
```

Nel file 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];
    }
}
```

L'output generato è come indicato di seguito:



Leggi Grafico (Coreplot) online: https://riptutorial.com/it/ios/topic/7302/grafico--coreplot-

Capitolo 72: Healthkit

Examples

HealthKit

Objective-C

Prima vai su `Target->Capabilities` e abilita `HealthKit` . Questo imposterà la voce info.plist.

Crea un nuovo `CocoaClass` di tipo `NSObject` Il nome file che ho dato è `GSHealthKitManager` e il file di intestazione è come mostrato di seguito

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
```

Chiamata da ViewController

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

    [[GSHealthKitManager sharedManager] requestAuthorization];
    NSDate *birthDate = [[GSHealthKitManager sharedManager] readBirthDate];
```

```
        NSLog(@"birthdate %@", birthDate);
        NSLog(@"gender 2131321 %@", [[GSHealthKitManager sharedManager] readGender]);


}
```

Uscita Log

```
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
```

Leggi Healthkit online: https://riptutorial.com/it/ios/topic/7412/healthkit

Capitolo 73: I / O di file di testo di base

Examples

Leggi e scrivi dalla cartella Documenti

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)")
```

Leggi I / O di file di testo di base online: https://riptutorial.com/it/ios/topic/8892/i---o-di-file-di-testo-di-base

Capitolo 74: IBeacon

Parametri

parametri	Dettagli
manager	Riferimento CLLocationManager
regione	CLRegion potrebbe essere una regione circolare (area geofence o beacon)
fari	Array di CLBeacon contiene tutti i beacon a distanza

Osservazioni

I beacon sono oggetti IOT. Ci stiamo concentrando su quelli che sono conformi al protocollo iBeacon uno standard Apple. Ogni faro è un dispositivo a senso unico che trasmette 3 oggetti

1. UUID
2. Maggiore
3. Minore

Possiamo scansionare iBeacons impostando il nostro oggetto CLLocation manager per cercare beacon per un particolare UUID. Verranno scansionati tutti i beacon con l'UUID specificato.

CLLocation Manager offre anche chiamate su entrata e uscita dalla regione beacon.

Examples

iBeacon Basic Operation

1. Setup monitoraggio beacon

```
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. Il gestore località entra e esce dalla regione

```
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. Indicatore di raggio del local manager

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

Scansione di beacon specifici

```
 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
```

Ibeacons che vanno

Innanzitutto, devi richiedere l'autorizzazione dei servizi di localizzazione

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

Quindi puoi ottenere tutte le informazioni di `didRangeBeacons` all'interno di `didRangeBeacons`

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

Leggi IBeacon online: https://riptutorial.com/it/ios/topic/1958/ibeacon

Capitolo 75: IBOutlets

Osservazioni

IBOutlet non è né una parola riservata né una variabile o classe, è zucchero sintattico per Interface Builder. Dopo che il codice sorgente Objective-C è stato pre-elaborato, non viene risolto.

In Swift è risolto come zero.

È dichiarato in `<UIKit/UINibDeclarations.h>` come

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

Examples

Utilizzo di un IBOutlet in un elemento dell'interfaccia utente

In generale, gli IBOutlet vengono utilizzati per connettere un oggetto interfaccia utente a un altro oggetto, in questo caso un UIViewController. La connessione serve a consentire che l'oggetto sia interessato al codice o agli eventi a livello di codice. Ciò può essere fatto semplicemente utilizzando l'assistente da uno storyboard e facendo clic tenendo premuto il tasto di controllo dall'elemento alla sezione delle proprietà .h del controller della vista, ma può anche essere eseguito a livello di codice e manualmente collegando il codice IBOutlet alla scheda "connections" dell'oggetto la barra delle utilità a destra. Ecco un esempio oggettivo di un UIViewController con un'etichetta:

```
//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
```

E veloce:

```
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"
    }
}
```

La connessione tra l'oggetto storyboard e l'oggetto programmato può essere verificata come connessa se il punto a sinistra della dichiarazione della presa in .h è pieno. Un cerchio vuoto implicava una connessione incompleta.

Leggi IBOutlets online: https://riptutorial.com/it/ios/topic/4713/iboutlets

Capitolo 76: Idiomi di inizializzazione

Examples

Impostare su tuple per evitare la ripetizione del codice

Evita la ripetizione del codice nei costruttori impostando una tupla di variabili con un solo liner:

```
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)
    }
}
```

Inizializza con costanti posizionali

```
let mySwitch: UISwitch = {
    view.addSubview($0)
    $0.addTarget(self, action: "action", forControlEvents: .TouchUpInside)
    return $0
}(UISwitch())
```

Inizializza attributi in didSet

```
@IBOutlet weak var title: UILabel! {
  didSet {
    label.textColor = UIColor.redColor()
    label.font = UIFont.systemFontOfSize(20)
    label.backgroundColor = UIColor.blueColor()
  }
}
```

È anche possibile impostare un valore e inizializzarlo:

```
private var loginButton = UIButton() {
    didSet(oldValue) {
        loginButton.addTarget(self, action: #selector(LoginController.didClickLogin),
forControlEvents: .TouchUpInside)
    }
}
```

Uscite di gruppo in un NSObject personalizzato

Spostare ogni presa su un NSObject. Quindi trascina un oggetto dalla libreria alla scena controller dello storyboard e aggancia gli elementi lì.

```
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!
}
```

Inizializza con allora

Questo è simile nella sintassi all'esempio che inizializza utilizzando costanti posizionali, ma richiede l'estensione `Then` da https://github.com/devxoul/Then (in allegato).

```
let label = UILabel().then {
    $0.textAlignment = .Center
    $0.textColor = UIColor.blackColor(
    $0.text = "Hello, World!"
}
```

L'estensione `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 {}
```

Metodo di fabbrica con blocco

```
internal func Init<Type>(value : Type, block: @noescape (object: Type) -> Void) -> Type
{
    block(object: value)
    return value
}
```

Uso:

```
Init(UILabel(frame: CGRect.zero)) {
    $0.backgroundColor = UIColor.blackColor()
}
```

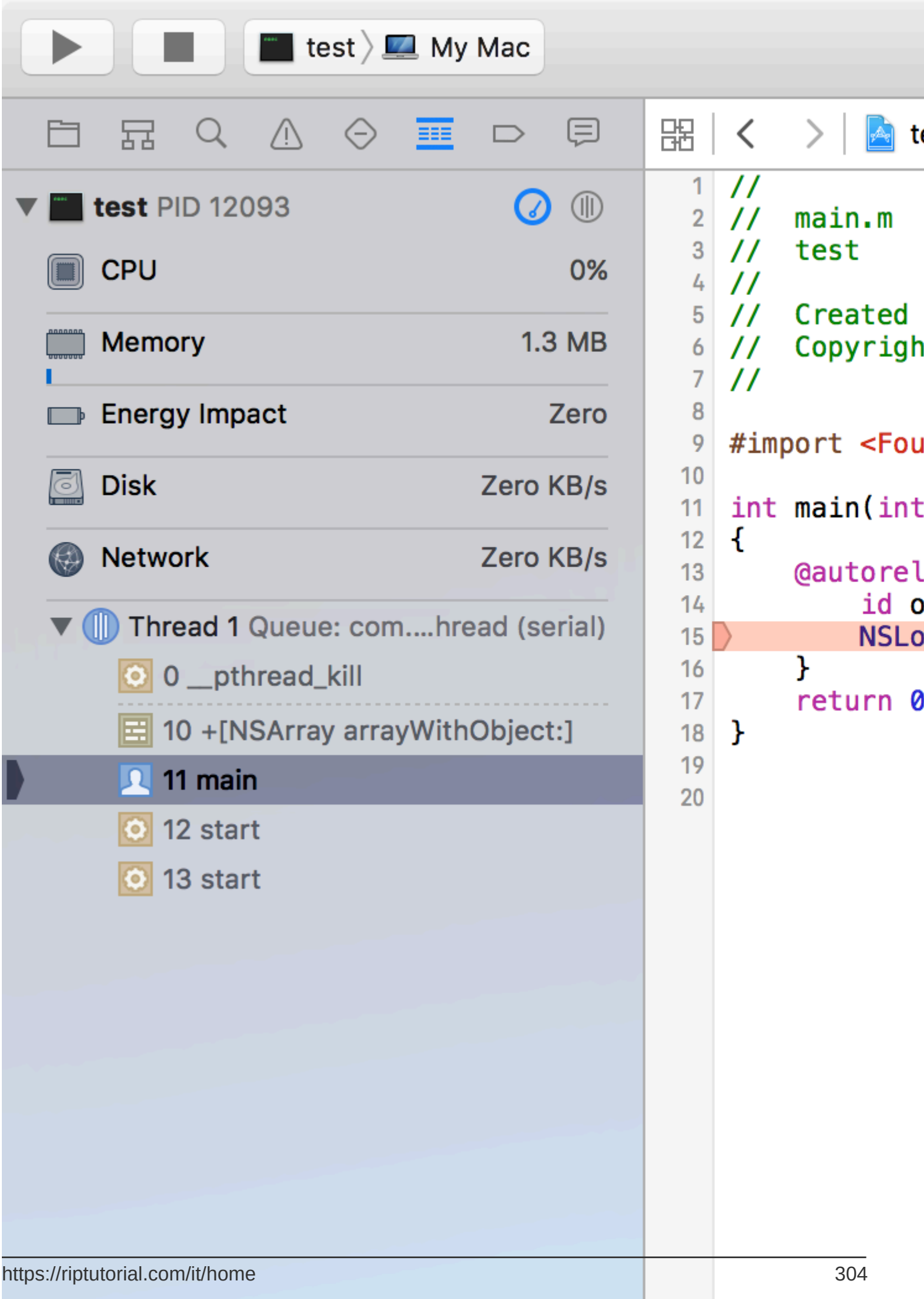
Leggi Idiomi di inizializzazione online: https://riptutorial.com/it/ios/topic/3513/idiomi-di-inizializzazione

Capitolo 77: Il debug si blocca

Examples

Trovare informazioni su un incidente

Quando l'app si arresta in modo anomalo, Xcode entra nel debugger e mostra ulteriori informazioni sullo schianto:


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
1 //
2 // main.m
3 // test
4 //
5 // Created
6 // Copyrigh
7 //
8
9 #import <Fou
10
11 int main(int
12 {
13 @autorel
14 id o
15 NSLo
16 }
17 return 0
18 }
19
20

si otterrà una rappresentazione testuale della traccia dello stack che è possibile copiare e incollare:

```
(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
```

Debug di SIGABRT e EXC_BAD_INSTRUCTION arresti anomali

Un `SIGABRT` o EXC_BAD_INSTRUCTION di solito significa che l'app si è bloccata intenzionalmente perché alcuni controlli non sono riusciti. Questi dovrebbero registrare un messaggio nella console del debugger con più informazioni; controllare lì per ulteriori informazioni.

Molti `SIGABRT` sono causati da eccezioni Objective-C non catturate. Ci sono *molti* motivi per cui possono essere lanciate eccezioni e registreranno *sempre* molte informazioni utili alla console.

- `NSInvalidArgumentException` , che significa che l'app ha passato un argomento non valido a un metodo
- `NSRangeException` , che significa che l'app ha provato ad accedere a un indice fuori limite di un oggetto come un `NSArray` o una `NSString`
- `NSInternalInconsistencyException` indica che un oggetto ha scoperto che si trovava in uno stato imprevisto.
- `NSUnknownKeyException` solito significa che hai una cattiva connessione in un XIB. Prova alcune delle risposte a questa domanda .

Debug di EXC_BAD_ACCESS

`EXC_BAD_ACCESS` significa che il processo ha tentato di accedere alla memoria in modo non valido, come il dereferenziamento di un puntatore `NULL` o la scrittura nella memoria di sola lettura. Questo è il tipo più difficile di arresto anomalo del debug, perché di solito non ha un messaggio di errore e alcuni arresti anomali possono essere *molto* difficili da riprodurre e / o verificarsi in codice completamente estraneo al problema. Questo errore è molto raro in Swift, ma se si verifica, spesso è possibile ottenere arresti più facili da debug riducendo le ottimizzazioni del compilatore.

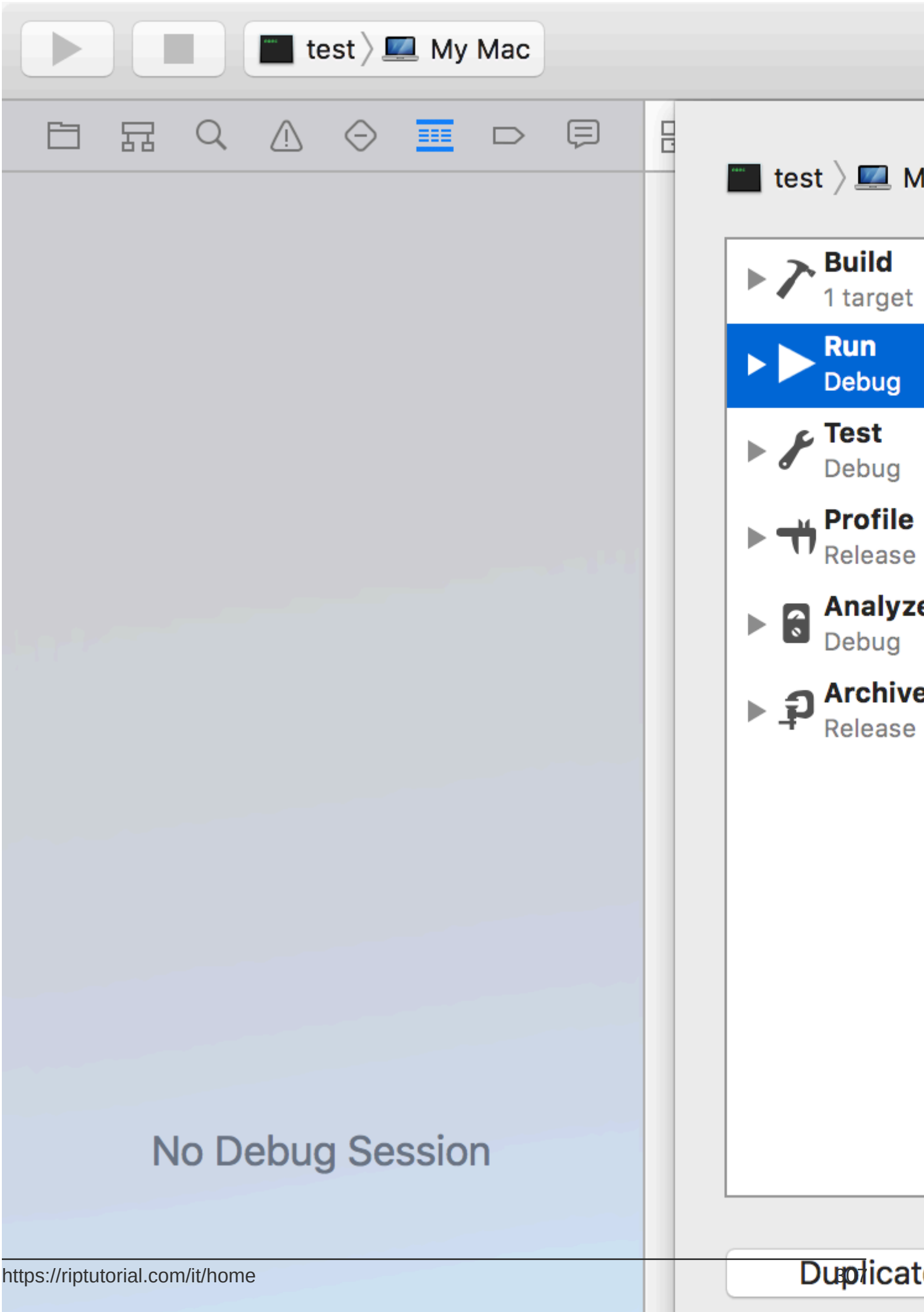
La maggior parte `EXC_BAD_ACCESS` errori `EXC_BAD_ACCESS` sono causati dal tentativo di dereferenziare un puntatore `NULL` . Se questo è il caso, l'indirizzo elencato nella freccia rossa sarà solitamente un

numero esadecimale che è inferiore a un normale indirizzo di memoria, spesso `0x0` . Impostare i punti di interruzione nel debugger o aggiungere istruzioni `printf` / `NSLog` occasionali per scoprire perché il puntatore è `NULL` .

Un `EXC_BAD_ACCESS` che si verifica in modo meno affidabile o non ha alcun senso potrebbe essere il risultato di un problema di gestione della memoria. I problemi comuni che possono causare questo sono:

- Usando la memoria che è stata deallocata
- Cercando di scrivere oltre la fine di un array C o di un altro tipo di buffer
- Utilizzando un puntatore che non è stato inizializzato

Nella sezione Diagnostics dell'Editor schema, Xcode include alcuni strumenti utili per aiutare a risolvere i problemi di memoria:


test
My Mac
test
Build
1 target
Run
Debug
Test
Debug
Profile
Release
Analyze
Debug
Archive
Release
No Debug Session

Capitolo 78: Imposta sfondo vista

Examples

Imposta Visualizza sfondo

Obiettivo C:

```
view.backgroundColor = [UIColor redColor];
```

Swift:

```
view.backgroundColor! = UIColor.redColor()
```

Swift 3

```
view.backgroundColor = UIColor.redColor
```

Riempi l'immagine di sfondo di un 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];
```

Imposta Visualizza sfondo con l'immagine

```
self.view.backgroundColor = [UIColor colorWithPatternImage:[UIImage
imageNamed:@"Background.png"]];
```

Creazione di una vista di sfondo sfumata

Per creare uno sfondo con una sfumatura puoi usare la classe 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)
}
```

Questo può essere chiamato su viewDidLoad () in questo modo:

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

Le variabili di posizioni e limiti CAGradientLayer possono assumere più valori per creare un livello sfumatura con quanti colori desideri. Dalla documentazione:

> Per impostazione predefinita, i colori vengono distribuiti uniformemente sul livello, ma è possibile specificare opzionalmente posizioni per il controllo sulle posizioni dei colori attraverso il gradiente.

Leggi Imposta sfondo vista online: https://riptutorial.com/it/ios/topic/6854/imposta-sfondo-vista

Capitolo 79: Installazione di Carthage iOS

Examples

Installazione di Cartagine Mac

Installazione di Carthage

Scarica l'ultima versione di Cartagine dal link di download indicato

Giù nella sezione Download scarica il file **Carthage.pkg** .

Una volta completato il download, installalo facendo doppio clic sul file pkg di download.

Per verificare il corretto download, eseguire il seguente comando nella versione di Carthage del terminale Questo dovrebbe fornire la versione installata come `0.18-19-g743fa0f`

Leggi Installazione di Carthage iOS online: https://riptutorial.com/it/ios/topic/7404/installazione-di-carthage-ios

Capitolo 80: Integrazione con SqlCipher

introduzione

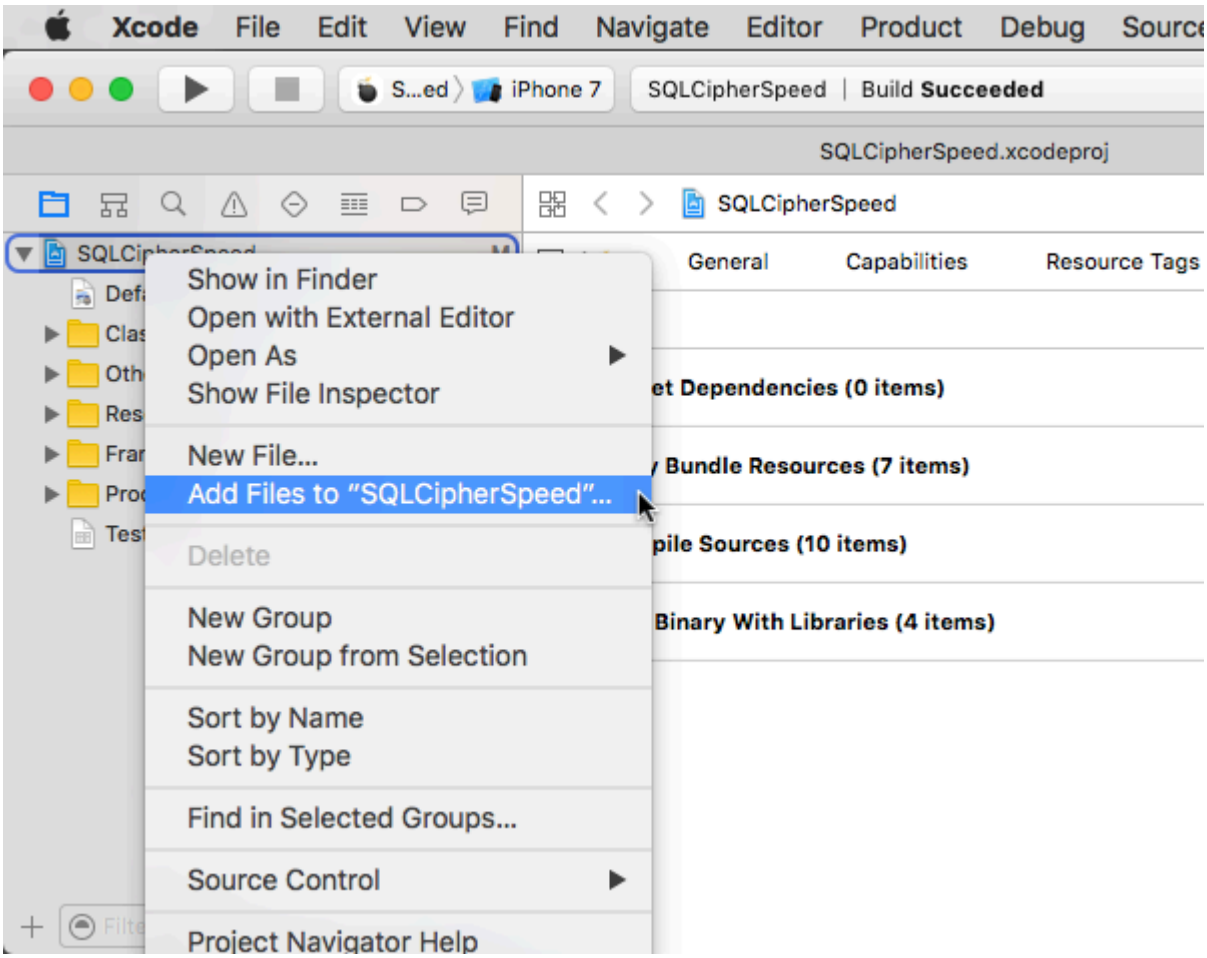
SQLite è già una popolare API per la memorizzazione persistente dei dati nelle app iOS, quindi il lato positivo per lo sviluppo è ovvio. Come programmatore lavori con API stabili e ben documentate che hanno molti buoni wrapper disponibili in Objective-C, come FMDB e Encrypted Core Data. Tutti i problemi di sicurezza sono nettamente disaccoppiati dal codice dell'applicazione e gestiti dal framework sottostante.

Osservazioni

1. Apri il terminale, passa alla directory principale del progetto e controlla il codice del progetto SQLCipher usando Git:

```
$ git clone https://github.com/sqlcipher/sqlcipher.git
```

2. Fai clic con il tasto destro del mouse sul progetto e seleziona "Aggiungi file a" La mia app "" (l'etichetta varia in base al nome della tua app). Poiché abbiamo clonato SQLCipher direttamente nella stessa cartella della tua app per iOS, dovresti vedere una cartella sqlcipher nella cartella del tuo progetto root. Apri questa cartella e seleziona **sqlcipher.xcodeproj**

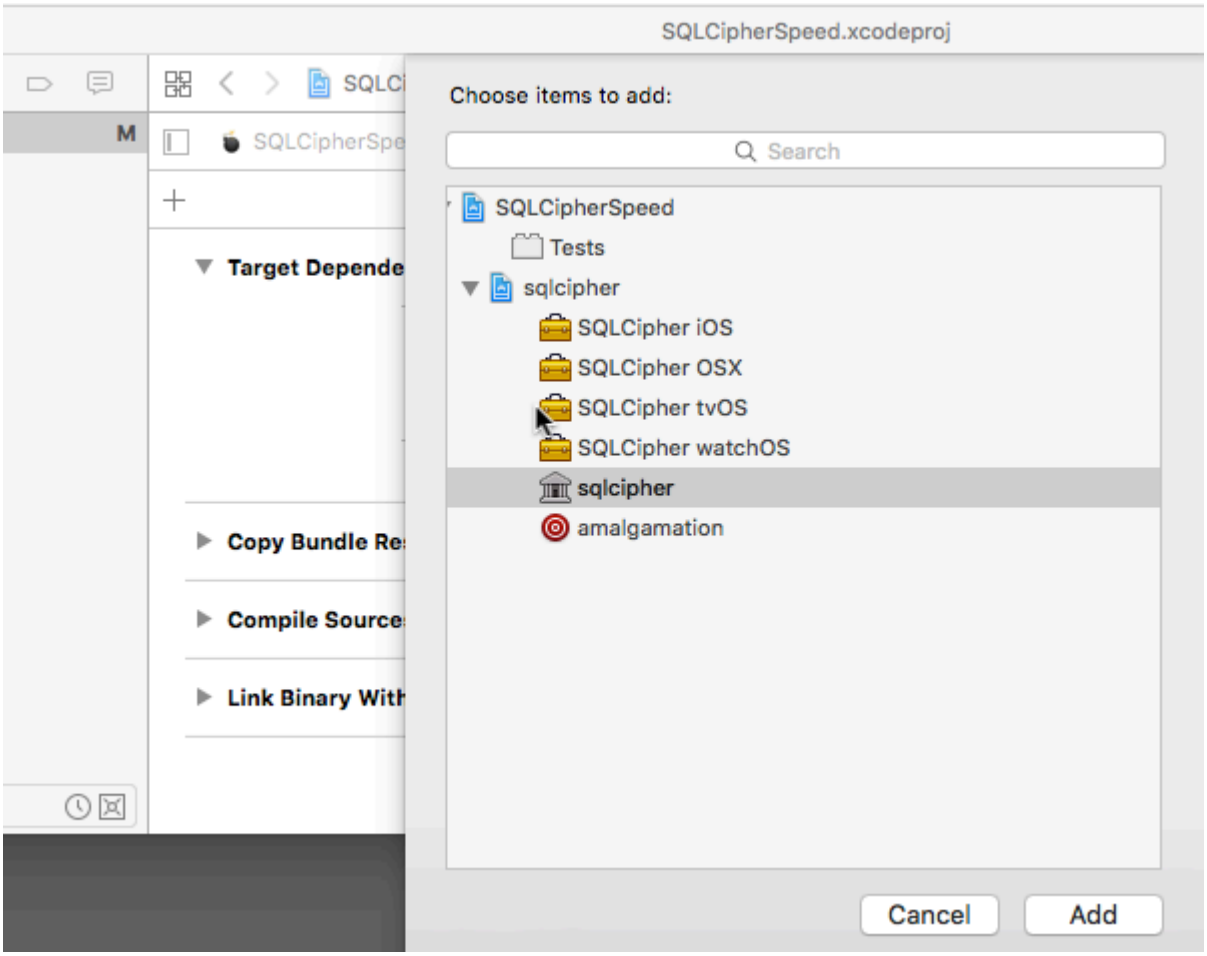


3. Seleziona il riquadro Impostazioni di costruzione. Nel campo di ricerca, digita "Percorsi di ricerca intestazione". Fare doppio clic sul campo sotto la colonna di destinazione e aggiungere il seguente percorso: **$ (PROJECT_DIR) / sqlcipher / src**

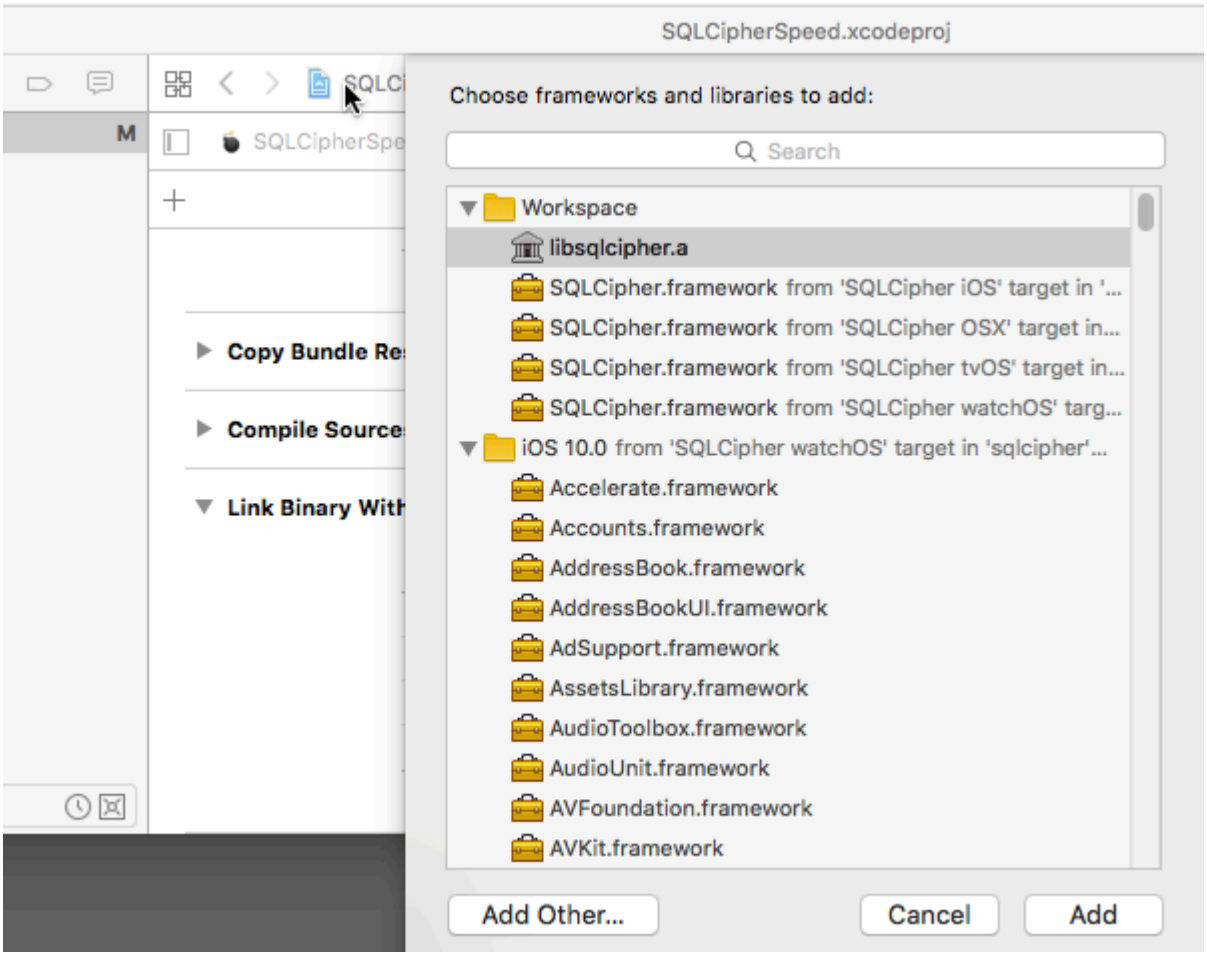
4. Inizia a digitare "Altri Linker Flags" nel campo di ricerca fino a quando appare l'impostazione, fai doppio clic per modificarlo e aggiungi il seguente valore: **$ (BUILT_PRODUCTS_DIR) /libsqlcipher.a**

5. Inizia a digitare "Altri C Flags" nel campo di ricerca fino a quando appare l'impostazione, fai doppio clic per modificarlo e nel pop-up aggiungi il seguente valore: **-DSQLITE_HAS_CODEC**

6. Espandere Dipendenze di destinazione e fare clic sul pulsante + alla fine dell'elenco. Nel browser che si apre, seleziona la destinazione della libreria statica **sqlcipher** :



7. Espandi Collega binario con le librerie, fai clic sul pulsante + alla fine dell'elenco e seleziona la libreria **libsqlcipher.a** .



8. Infine, anche sotto Link With Libraries, aggiungi **Security.framework** .

Examples

Integrazione del codice:

Integrazione per aprire il database usando la password.

```
-(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;
}
```

Leggi Integrazione con SqlCipher online: https://riptutorial.com/it/ios/topic/9969/integrazione-con-sqlcipher

Capitolo 81: Interoperabilità Swift e Objective-C

Examples

Utilizzo delle classi Objective-C in Swift

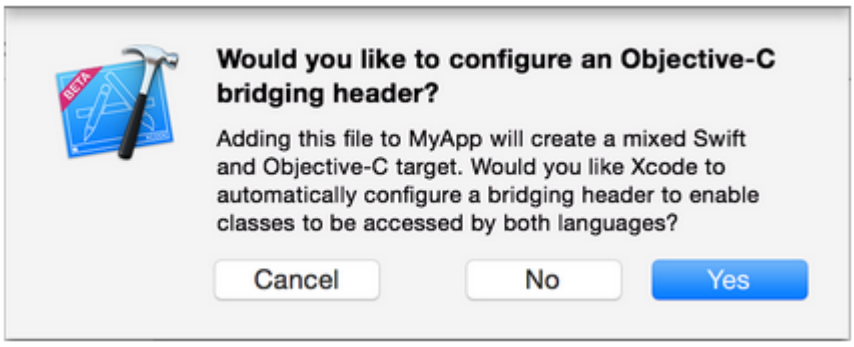
> Se si dispone di una classe esistente che si desidera utilizzare, eseguire il **passaggio 2** e quindi saltare al **passaggio 5** . (In alcuni casi, ho dovuto aggiungere un `#import <Foundation/Foundation.h` esplicito `#import <Foundation/Foundation.h` a un file ObjC precedente)

Passaggio 1: aggiungere l'implementazione Objective-C - .m

Aggiungi un file `.m` alla tua classe e `CustomObject.m`

Passaggio 2: aggiungi intestazione di ponte

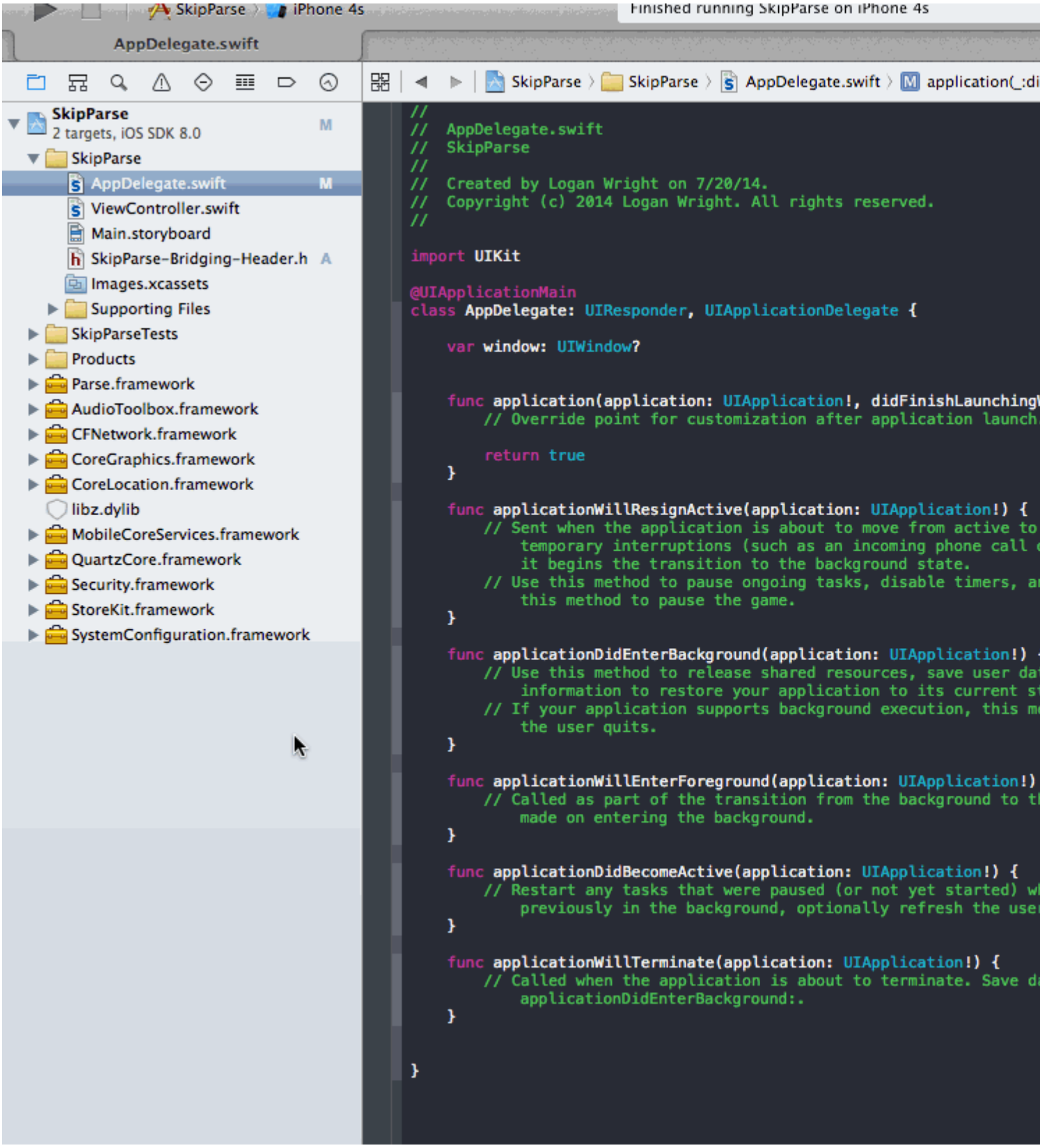
Quando aggiungi il tuo file `.m` , verrai colpito da un prompt simile al seguente:



Clicca **SÌ** !

Se non hai visualizzato il prompt o eliminato accidentalmente l'intestazione di bridging, aggiungi un nuovo file `.h` al progetto e `<#YourProjectName#>-Bridging-Header.h` nome `<#YourProjectName#>-Bridging-Header.h`

In alcune situazioni, in particolare quando si lavora con i framework ObjC, non si aggiunge esplicitamente una classe Objective-C e Xcode non trova il linker. In questo caso, crea il tuo file `.h` chiamato come sopra menzionato, quindi assicurati di collegare il suo percorso nelle impostazioni del progetto del tuo target in questo modo:



Nota

È consigliabile collegare il progetto usando la macro `$(SRCROOT)` modo che se si sposta il progetto o si lavori con altri utilizzando un repository remoto, funzionerà comunque. `$(SRCROOT)` può essere pensato come la directory che contiene il tuo file .xcodeproj. Potrebbe assomigliare a questo:

```
$(SRCROOT)/Folder/Folder/<#YourProjectName#>-Bridging-Header.h
```

Passaggio 3: aggiungere l'intestazione Objective-C - .h

Aggiungi un altro file `.h` e `CustomObject.h`

Step 4: Costruisci la tua classe Objective-C

In `CustomObject.h`

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

@interface CustomObject : NSObject

@property (strong, nonatomic) id someProperty;

- (void) someMethod;

@end
```

In `CustomObject.m`

```
#import "CustomObject.h"

@implementation CustomObject

- (void) someMethod {
    NSLog(@"SomeMethod Ran");
}

@end
```

Passaggio 5: aggiungi classe a Bridging-Header

In `YourProject-Bridging-Header.h` :

```
#import "CustomObject.h"
```

Passaggio 6: utilizza il tuo oggetto

In `SomeSwiftFile.swift` :

```
var instanceOfCustomObject: CustomObject = CustomObject()
instanceOfCustomObject.someProperty = "Hello World"
println(instanceOfCustomObject.someProperty)
instanceOfCustomObject.someMethod()
```

Non c'è bisogno di importare esplicitamente, questo è ciò che l'header bridging è per.

Utilizzo delle classi Swift in Objective-C

Passaggio 1: crea una nuova classe Swift

Aggiungi un file `.swift` al tuo progetto e `MySwiftObject.swift`

In `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
    }

}
```

Passaggio 2: importa i file Swift alla classe ObjC

In `SomeRandomClass.m` :

```
#import "<#YourProjectName#>-Swift.h"
```

Il file: `<#YourProjectName#>-Swift.h` dovrebbe già essere creato automaticamente nel tuo progetto, anche se non puoi vederlo.

Step 3: Usa la tua classe

```
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);
```

Nota:

1. CodeCompletion non si comportava in modo accurato come mi piacerebbe. Sul mio sistema, l'esecuzione di una build rapida con "cmd + r" sembrava aiutare Swift a trovare alcuni codici Objc e viceversa.

2. Se aggiungi `.swift` file `.swift` a un progetto precedente e ottieni l'errore: `dyld: Library not loaded: @rpath/libswift_stdlib_core.dylib` , prova a riavviare completamente Xcode.

3. Mentre inizialmente era possibile utilizzare le classi pure di Swift in Objective-C usando il prefisso `@objc` , dopo Swift 2.0, questo non è più possibile. Vedi la cronologia delle modifiche per la spiegazione originale. Se questa funzionalità viene riattivata nelle future versioni di Swift, la risposta verrà aggiornata di conseguenza.

Leggi Interoperabilità Swift e Objective-C online:
https://riptutorial.com/it/ios/topic/1497/interoperabilita-swift-e-objective-c

Capitolo 82: iOS TTS

introduzione

Scopri come produrre sintesi vocale dal testo su un dispositivo iOS

Examples

Sintesi vocale

Obiettivo C

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

veloce

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

Puoi anche cambiare la voce in questo modo:

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

E poi speek

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

Non dimenticare di importare AVFoundation

Metodi utili

Puoi interrompere o mettere in pausa tutti i discorsi utilizzando questi due metodi:

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

L'AVSpeechBoundary indica se il parlato dovrebbe fermarsi o fermarsi immediatamente (`AVSpeechBoundaryImmediate`) o dovrebbe fermarsi o fermarsi dopo che la parola è stata pronunciata

(`AVSpeechBoundaryWord`).

Leggi iOS TTS online: https://riptutorial.com/it/ios/topic/8909/ios-tts

Capitolo 83: iOS: implementazione di XMPP con framework Robbie Hanson

Examples

iOS XMPP Robbie Hanson Esempio con Openfire

SRXMPPDemo

Scarica l'esempio e tutte le classi qui - https://github.com/SahebRoy92/SRXMPPDemo

Una demo su XMPP in Objective C, con varie funzioni semplici e complesse implementate in esso. Tutte le funzionalità di XMPP sono fatte da funzioni **"in banda"** xmpp. Poche caratteristiche che questo progetto contiene sono

SRXMPP - Una classe Singleton wrapper che ha quasi tutte le funzionalità necessarie per l'applicazione di chat one-to-one.

- chat individuale
- Implementazione dei dati di base della chat (messaggio di testo) avendo così il salvataggio di messaggi precedenti, messaggi offline.
- implementazione di vCard (informazioni sul profilo dell'utente, proprio e altrui) da XML e Core Data forniti dal framework di Robbie Hanson.
- disponibilità dello stato degli amici (online / offline / digitando)

Passi da seguire

Vuoi utilizzare questo progetto come riferimento, quindi puoi fare quanto segue--

1. Installato Openfire in un server live - Noleggia un server, installa openfire.

2. Vuoi provarlo senza problemi nel tuo computer : devi avviare, installare e installare 3 cose

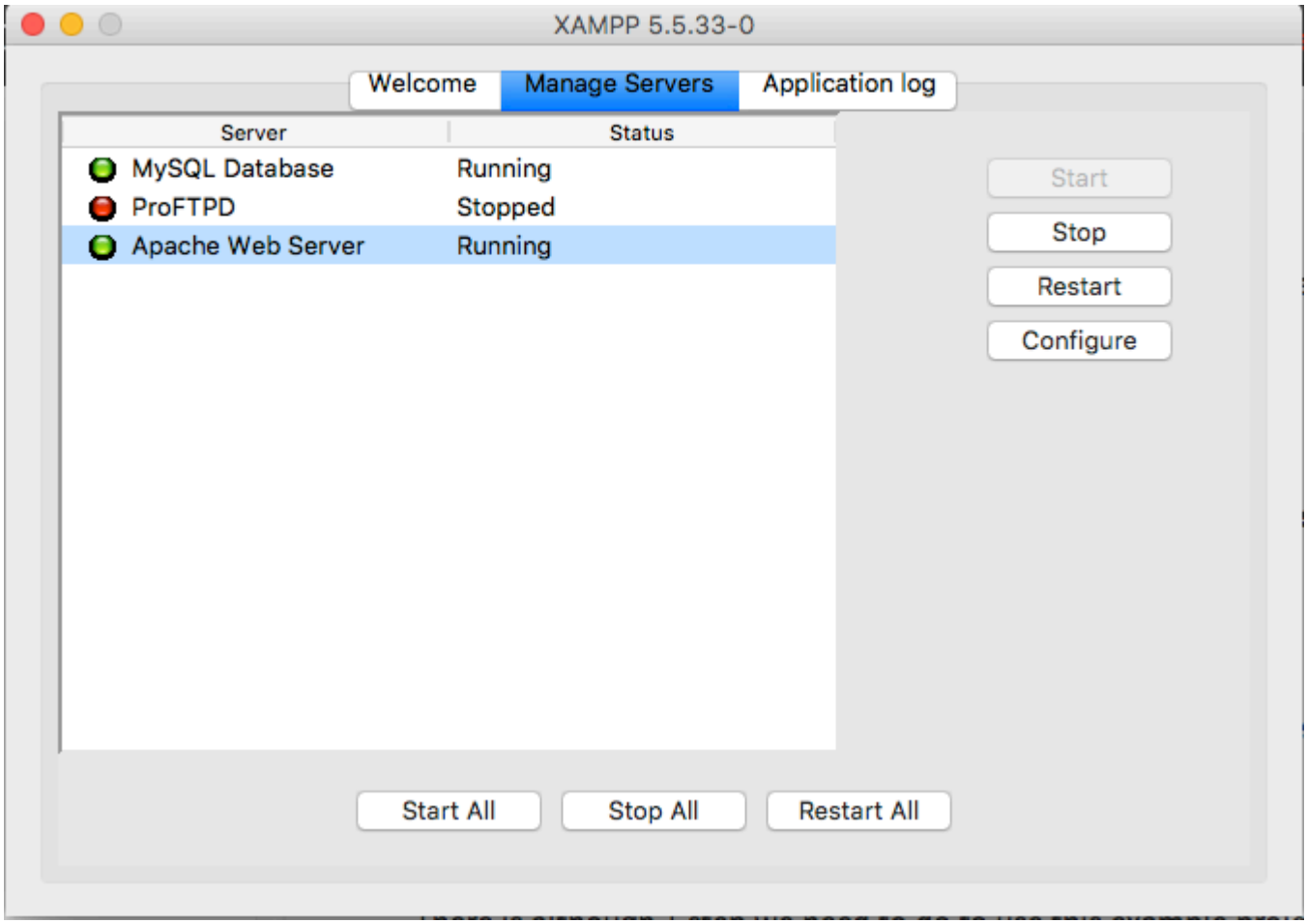
un. Java -

- Scarica e installa Java per Mac.

b. XAMPP -

- Installare XAMPP è relativamente facile.

- Dopo l'installazione è sufficiente avviare XAMPP e avviare **Database (SQL)** e **Apache Server** .

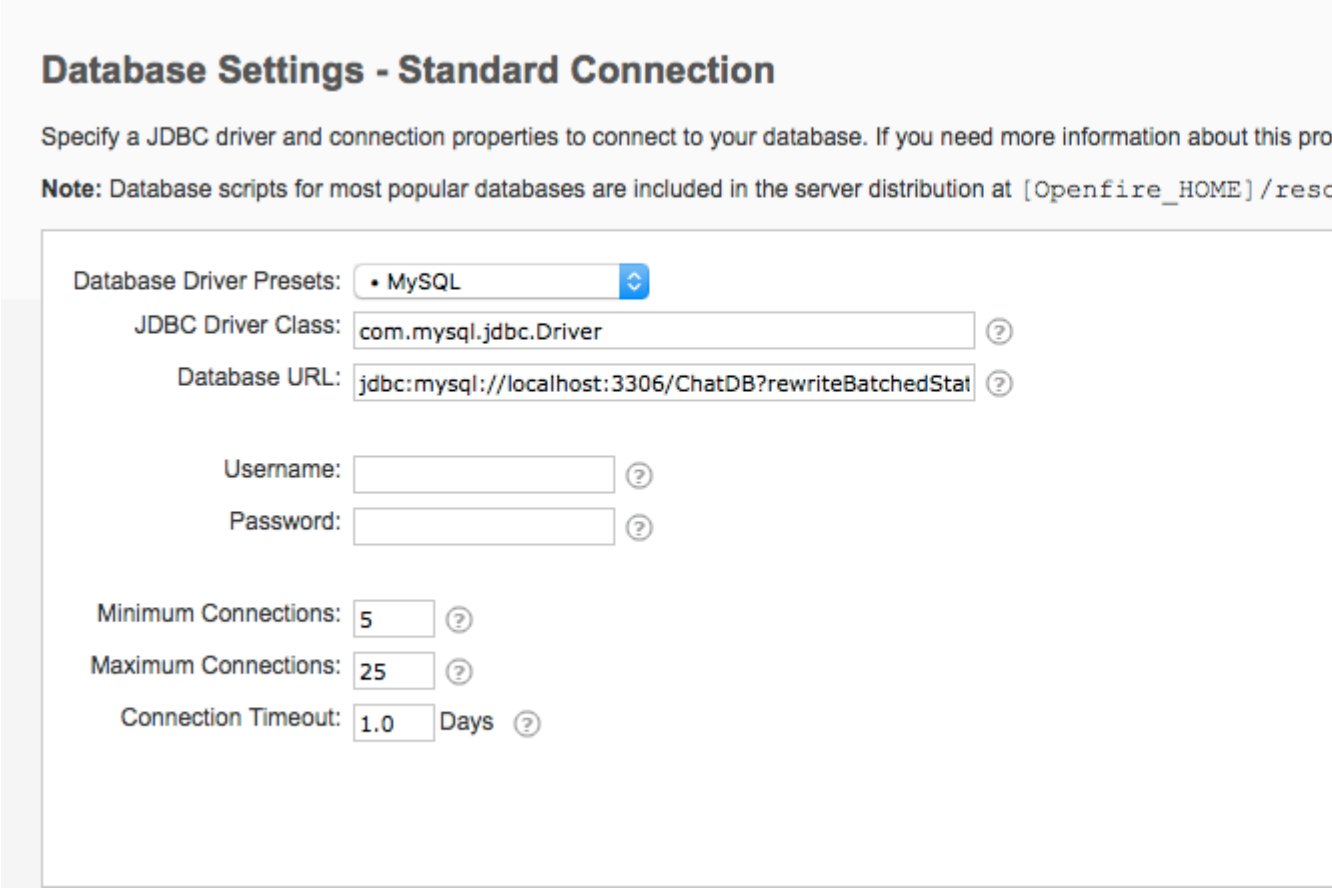


- Quindi apri il browser e incolla questo URL *[http: // localhost / phpmyadmin /]*
- . Crea un nuovo DB dal pannello laterale sinistro.
- **Assegna un nome al DB ma ricorda questo nome, supponiamo di chiamarlo ChatDB**

c. Openfire -

- Installa Openfire ed esegui l'applicazione e "Start Openfire"

- Apri il browser e incolla questo URL - [http: // localhost: 9090 / setup / index.jsp] (http: // localhost: 9090 / setup / index.jsp)
- Fai la configurazione normale
    - Seleziona la lingua>
    - Impostazioni del server, lasciare così com'è, basta continuare>
    - Impostazioni database, lascia come "Connessione database standard come selezionata>
    - Impostazioni database - Connessione standard ". Ora ricorda che il nome del DB che hai impostato era **ChatDB** .
    - Seleziona Preset driver del database come * *" MySQL"* . Lascia la classe di driver JDBC così com'è. Ora nell'URL del database è possibile vedere le parentesi che menzionano il nome host e il nome del database. Basta cambiare il nome host in **"localhost"** e il nome del database in **"ChatDB"** , o qualsiasi altro nome di DB impostato in precedenza, mentre si imposta XAMPP. Lascia il nome utente e la password come vuoti. Compila i dettagli come l'immagine qui



    .
    - Prossima installazione completa dando un nome utente e password e riconfermandolo. Questo è ciò che hai fatto Impostazione di Openfire.

Ora la parte arriva quando devi modificare un piccolo dettaglio nel codice.

Importante Dobbiamo andare alla classe - **SRXMPP.m** , localizzare NSString extern **SRXMPP_Hostname** (nella parte superiore) e sovrascriverne il valore

- IP del server su cui è installato OpenFire, **OR**

- se lo hai installato localmente, sovrascrivi il valore su - **"localhost"** .

Ecco, sei pronto per utilizzare questo progetto di esempio e iniziare a programmare e renderlo un tuo progetto migliore.

Questo pacchetto introduttivo ti aiuterà a comprendere meglio la struttura XMPP e ad acquisire familiarità con i protocolli XMPP.

Puoi trovare altri protocolli XMPP qui in questo sito - https://xmpp.org/rfcs/rfc3920.html

Lo sviluppo è ancora lasciato e parti in cui spero di includerle in seguito

1. Chat di gruppo
2. Supporto invio immagini

In breve, questo progetto di esempio insieme al singleton ha quasi tutte le funzionalità necessarie per un'applicazione di chat One-to-One.

Leggi iOS: implementazione di XMPP con framework Robbie Hanson online: https://riptutorial.com/it/ios/topic/1475/ios--implementazione-di-xmpp-con-framework-robbie-hanson

Capitolo 84: Istantanea di UIView

Examples

Ottenere l'istantanea

```
- (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;
}
```

veloce

```
var screenshot: UIImage
{
UIGraphicsBeginImageContext(self.bounds.size);
let context = UIGraphicsGetCurrentContext();
self.layer.render(in: context)
let screenShot = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
return screenShot
}
```

Istantanea con sottoview con altri markup e testo

- Supporta Portrait e Landscape entrambi i tipi di immagine
- Disegno e altre sottoview possono essere unite nel mio caso sto aggiungendo un'etichetta per disegnare

```
{
    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);
}
```

Leggi Istantanea di UIView online: https://riptutorial.com/it/ios/topic/4622/istantanea-di-uiview

Capitolo 85: Key Value Coding: Key Value Observation

Osservazioni

KVC : - Codifica valore-chiave

Normalmente le variabili di istanza sono accessibili tramite proprietà o accessor ma KVC offre un altro modo per accedere alle variabili in forma di stringhe. In questo modo la classe agisce come un dizionario e il nome della tua proprietà ad esempio "età" diventa la chiave e il valore che la proprietà detiene diventa valore per quella chiave.

```
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 : - Key-Value Observer

Il meccanismo attraverso il quale gli oggetti vengono notificati quando c'è un cambiamento in una qualsiasi delle proprietà è chiamato KVO. Ex.:keyboard notification

> Ad esempio, l'oggetto persona è interessato ad ottenere una notifica quando la proprietà accountBalance viene modificata nell'oggetto BankAccount. Per raggiungere questo obiettivo, Person Object deve registrarsi come osservatore della proprietà del saldo del conto BankAccount inviando un addObserver: forKeyPath: options: context: message.

Examples

Uso del contesto per l'osservazione KVO

```
-(void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object
change:(NSDictionary<NSString *,id> *)change context:(void *)context
```

Il contesto è importante se spedisci la tua classe affinché altri possano usarla. Con il tuo programma di ricerca, l'osservatore di classe verifica che il tuo osservatore sia chiamato.

Il problema di non passare un osservatore è, se qualcuno sottoclasse la tua classe e registra un osservatore per lo stesso oggetto, stessa chiave e lui non passa un contesto, allora l'osservatore di super classe può essere chiamato più volte.

Una variabile che è unica e interna per il tuo uso è un buon contesto.

Per maggiori informazioni.

importanza e buon contesto

Osservazione di una proprietà di una sottoclasse NSObject

La maggior parte delle funzionalità KVO e KVC è già implementata per impostazione predefinita su tutte le sottoclassi `NSObject` .

Per iniziare a osservare una proprietà denominata `firstName` di un oggetto denominato `personObject` fai questo nella classe observing:

```
[personObject addObserver:self
               forKeyPath:@"firstName"
                  options:NSKeyValueObservingOptionNew
                  context:nil];
```

L'oggetto a cui `self` riferisce il codice sopra riportato riceverà quindi un `observeValueForKeyPath:ofObject:change:context:` message ogni volta che il percorso chiave osservato cambia.

```
- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary<NSString *,id> *)change
                       context:(void *)context
{
    NSLog(@"new value of %@ is: %@", keyPath, change[NSKeyValueChangeNewKey]);
}
```

"Percorso chiave" è un termine KVC. `NSObject` sottoclassi `NSObject` implementano la funzionalità KVC per impostazione predefinita.

Una variabile di istanza denominata `_firstName` sarà accessibile tramite il percorso chiave `@"firstName"` .

Procedimento getter denominata `firstName` verrà chiamato quando si accede al `@"firstName"` percorso della chiave, indipendentemente dall'esistenza di un `_firstName` variabile istanza o `setFirstName` metodo setter.

Leggi Key Value Coding: Key Value Observation online:
https://riptutorial.com/it/ios/topic/3493/key-value-coding--key-value-observation

Capitolo 86: Layout automatico

introduzione

Layout automatico calcola dinamicamente la dimensione e la posizione di tutte le viste nella gerarchia della vista, in base ai vincoli posti su tali viste. fonte

Sintassi

- NSLayoutConstraint (elemento: Qualsiasi, attributo: NSLayoutAttribute, relatedBy: NSLayoutRelation, toItem: Qualsiasi ?, attributo: NSLayoutAttribute, moltiplicatore: CGFloat, costante: CGFloat) // Crea una contraint a livello di codice

Examples

Impostazione dei vincoli a livello di codice

Esempio di codice Boilerplate

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

    let myView = UIView()
    myView.backgroundColor = UIColor.blueColor()
    myView.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(myView)

    // Add constraints code here
    // ...
}
```

Negli esempi sotto lo Stile di ancoraggio è il metodo preferito su `NSLayoutConstraint` Style, tuttavia è disponibile solo da iOS 9, quindi se stai supportando iOS 8, dovresti comunque utilizzare `NSLayoutConstraint` Style.

pinning

Stile di ancoraggio

```
let margins = view.layoutMarginsGuide
myView.leadingAnchor.constraintEqualToAnchor(margins.leadingAnchor, constant: 20).active =
true
```

- Oltre a `leadingAnchor` , ci sono anche `trailingAnchor` , `topAnchor` e `bottomAnchor` .

NSLayoutConstraint Style

```
NSLayoutConstraint(item: myView, attribute: NSLayoutAttribute.Leading, relatedBy:
NSLayoutRelation.Equal, toItem: view, attribute: NSLayoutAttribute.LeadingMargin, multiplier:
1.0, constant: 20.0).active = true
```

- Oltre a `.Leading` c'è anche. `.Trailing` , `.Top` e `.Bottom` .
- Oltre a `.LeadingMargin` c'è anche. `.TrailingMargin` , `.TopMargin` e `.BottomMargin` .

Stile di linguaggio in formato visivo

```
NSLayoutConstraint.constraintsWithVisualFormat("H:|-20-[myViewKey]", options: [], metrics:
nil, views: ["myViewKey": myView])
```

Larghezza e altezza

Stile di ancoraggio

```
myView.widthAnchor.constraintEqualToAnchor(nil, constant: 200).active = true
myView.heightAnchor.constraintEqualToAnchor(nil, constant: 100).active = true
```

NSLayoutConstraint Style

```
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
```

Stile di linguaggio in formato visivo

```
NSLayoutConstraint.constraintsWithVisualFormat("H:[myViewKey(200)]", options: [], metrics:
nil, views: ["myViewKey": myView])
NSLayoutConstraint.constraintsWithVisualFormat("V:[myViewKey(100)]", options: [], metrics:
nil, views: ["myViewKey": myView])
```

Centro nel contenitore

Stile di ancoraggio

```
myView.centerXAnchor.constraintEqualToAnchor(view.centerXAnchor).active = true
myView.centerYAnchor.constraintEqualToAnchor(view.centerYAnchor).active = true
```

NSLayoutConstraint Style

```
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
```

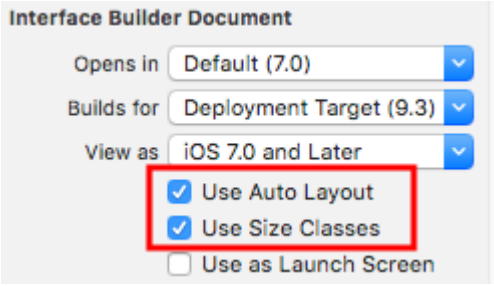
Stile di linguaggio in formato visivo

```
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])
```

Come usare Auto Layout

Il layout automatico viene utilizzato per disporre le viste in modo che risultino perfette su qualsiasi dispositivo e orientamento. I vincoli sono le regole che dicono come dovrebbe essere stabilito tutto. Includono bordi di pinning, centratura e dimensioni di impostazione, tra le altre cose.

Il layout automatico è abilitato di default, ma puoi ricontrollarlo. Se fai clic su *Main.storyboard* nel Project Navigator e poi mostra la *finestra* di ispezione File. Assicurati che le Classi Auto Layout e Dimensioni siano selezionate:

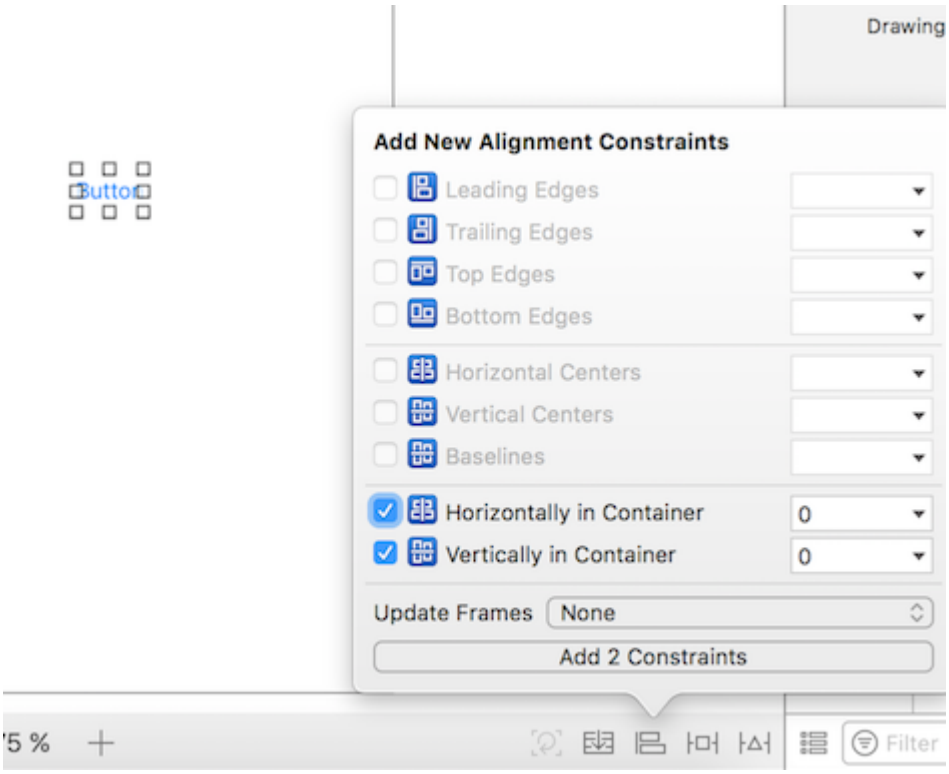


I vincoli di layout automatici possono essere impostati in Interface Builder o nel codice. In Interface Builder trovi gli strumenti di Auto Layout in basso a destra. Facendo clic su di essi verranno visualizzate diverse opzioni per l'impostazione dei vincoli su una vista.

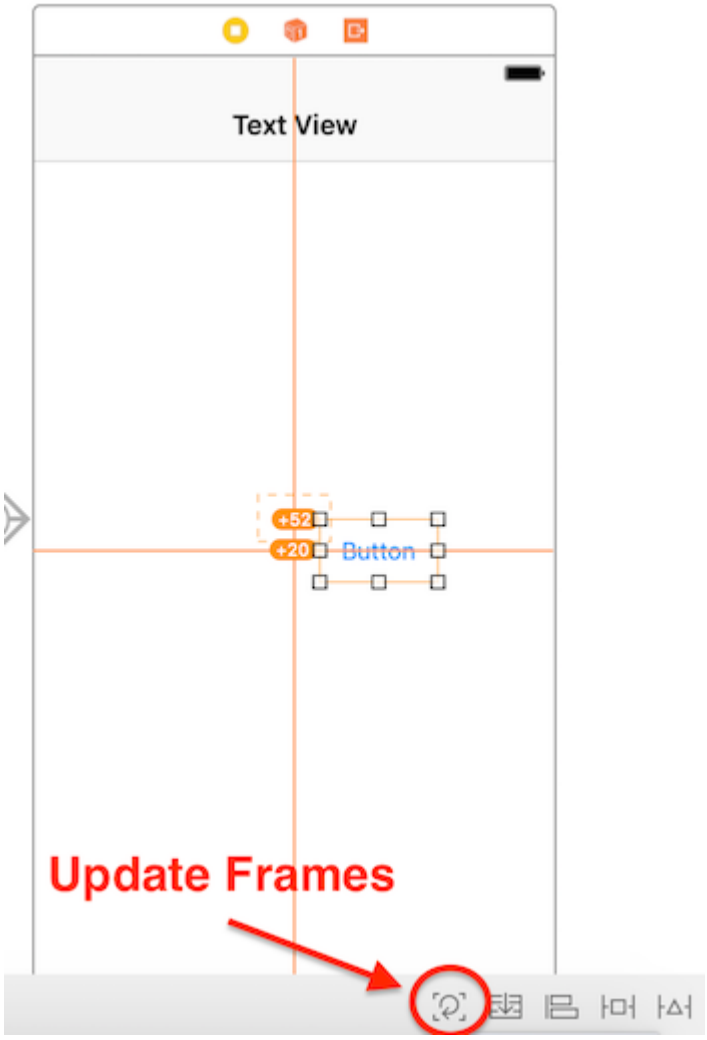
Se si desidera avere vincoli diversi per le diverse dimensioni o orientamenti dei dispositivi, è possibile impostarli in wAny hAny opzioni di classe di dimensioni che si trovano nella parte centrale inferiore.

Vincoli al centro

Seleziona il tuo pulsante (o qualsiasi altra vista che vuoi centrare) sullo **storyboard** . Quindi fare clic sul pulsante di allineamento in basso a destra. Selezionare `Horizontally in Container` e `Vertically in Container` . Fai clic su "Aggiungi 2 vincoli".



Se non fosse già perfettamente centrato, potrebbe essere necessario fare ancora una cosa. Fai clic sul pulsante "Aggiorna frame" che si trova a due a sinistra del pulsante "Incorpora nello stack"

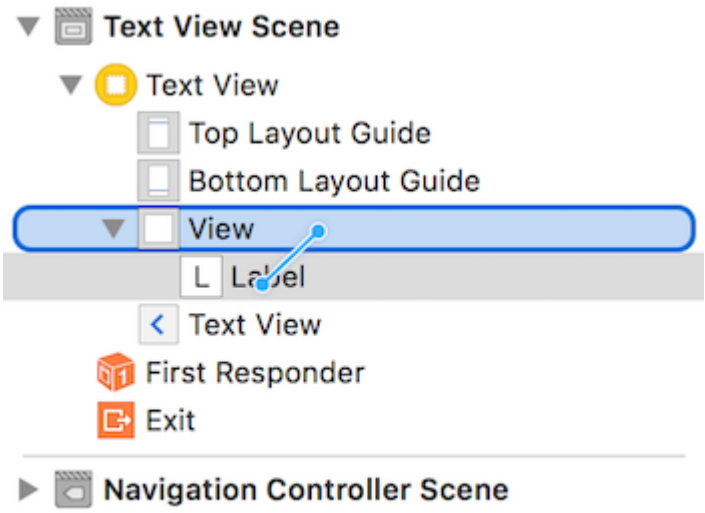


nella barra inferiore.

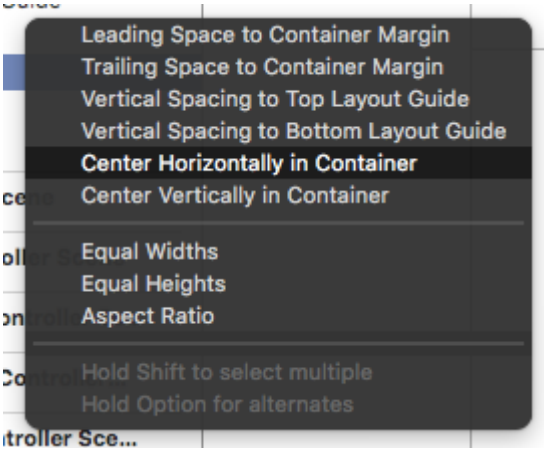
Puoi anche "aggiornare i fotogrammi se necessario" premendo insieme ⌘ + ⌥ + = (Comando + Opzione e uguale) dopo aver selezionato la vista, questo potrebbe far risparmiare un po 'di tempo.

Ora quando esegui la tua app dovrebbe essere centrata, indipendentemente dalla dimensione del dispositivo che stai utilizzando.

Un altro modo per centrare le viste usando Interface Builder è control-click-dragging. `UILabel` voler centrare un `UILabel` in una vista. Apri lo `Document Outline` del `Document Outline` nello storyboard facendo clic sul pulsante della barra laterale in basso a sinistra. Clicca e trascina dall'etichetta alla vista tenendo premuto `ctrl` (controllo), e dovrebbe apparire una linea blu:



Al rilascio, apparirà un menu di opzioni di vincolo:



Seleziona "Centra orizzontalmente nel contenitore" e "Centra verticalmente nel contenitore". Aggiorna i fotogrammi se necessario, e voilà! Un'etichetta centrata.

In alternativa, è possibile aggiungere i vincoli a livello di codice. Crea i vincoli e aggiungili agli elementi e alle viste dell'interfaccia utente desiderati come descritto nel seguente esempio, dove creiamo un pulsante e lo allineiamo al centro, in orizzontale e in verticale, alla sua superview:

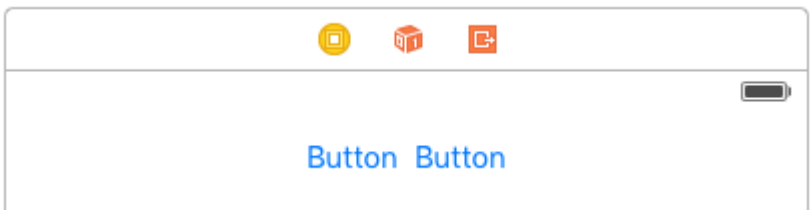
Objective-C

```
- (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
}
```

veloce

```
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])
}
```

Viste spaziali in modo uniforme



È comune desiderare che due viste siano affiancate, centrate nella loro superview. La risposta comune fornita su Stack Overflow consiste nell'incorporare queste due viste in un `UIView` e centrare l' `UIView` . Questo non è necessario o raccomandato. Dai documenti UILayoutGuide :

> Vi sono una serie di costi associati all'aggiunta di viste fittizie alla gerarchia della vista. Innanzitutto, c'è il costo di creare e mantenere la vista stessa. In secondo luogo, la vista fittizia è un membro completo della gerarchia della vista, il che significa che aggiunge un sovraccarico a tutte le attività eseguite dalla gerarchia. La cosa peggiore è che la vista fittizia invisibile può intercettare i messaggi destinati ad altre viste, causando problemi che sono molto difficili da trovare.

Puoi usare `UILayoutGuide` per fare questo, invece di aggiungere i pulsanti a un `UIView` non `UIView` . Una `UILayoutGuide` è essenzialmente uno spazio rettangolare che può interagire con Auto Layout. `UILayoutGuide` una `UILayoutGuide` sui lati sinistro e destro dei pulsanti e imposta la loro larghezza per essere uguale. Questo centrerà i pulsanti. Ecco come farlo nel codice:

Stile di linguaggio in formato visivo

```
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
```

Stile di ancoraggio

```
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
```

Sarà necessario aggiungere vincoli verticali anche a questo, ma questo centrerà i pulsanti nella

vista senza aggiungere alcuna vista "fittizia"! Ciò salverà il sistema dal sprecare tempo della CPU nel visualizzare quelle viste "fittizie". Questo esempio utilizza i pulsanti, ma puoi scambiare i pulsanti per qualsiasi vista su cui vuoi applicare i vincoli.

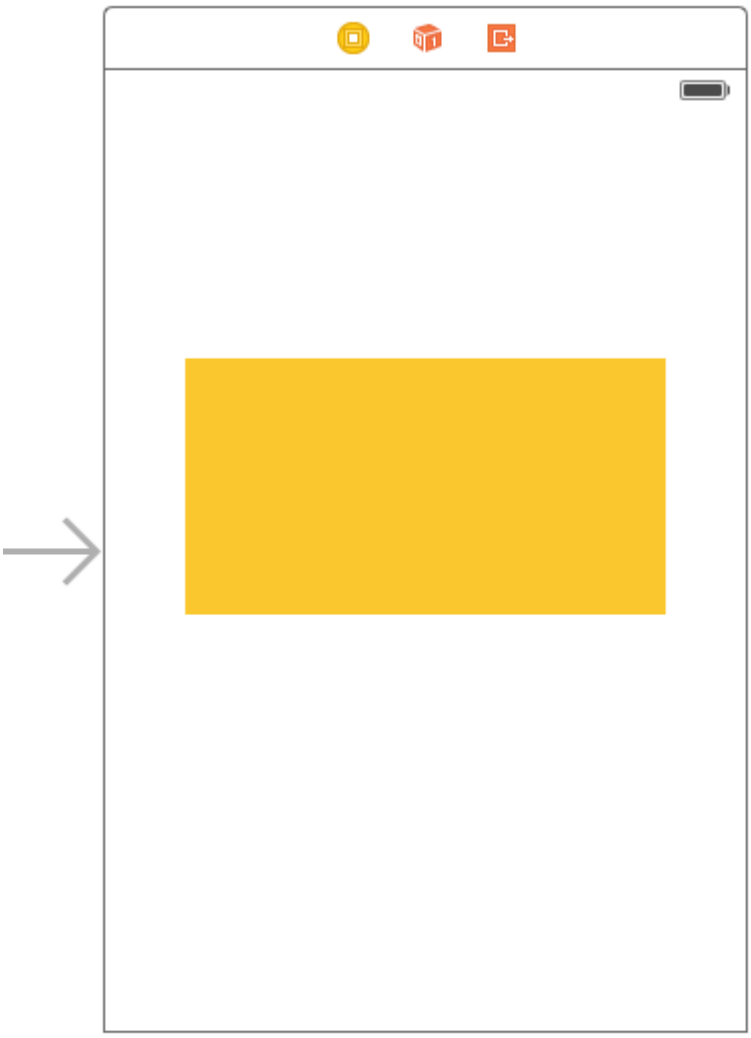
Se stai supportando iOS 8 o versioni precedenti, il modo più semplice per creare questo layout è aggiungere viste fittizie nascoste. Con iOS 9 è possibile sostituire le viste fittizie con le guide di layout.

Nota: Interface Builder non supporta ancora le guide di layout (Xcode 7.2.1). Quindi se vuoi usarli devi creare i tuoi vincoli nel codice. Fonte .

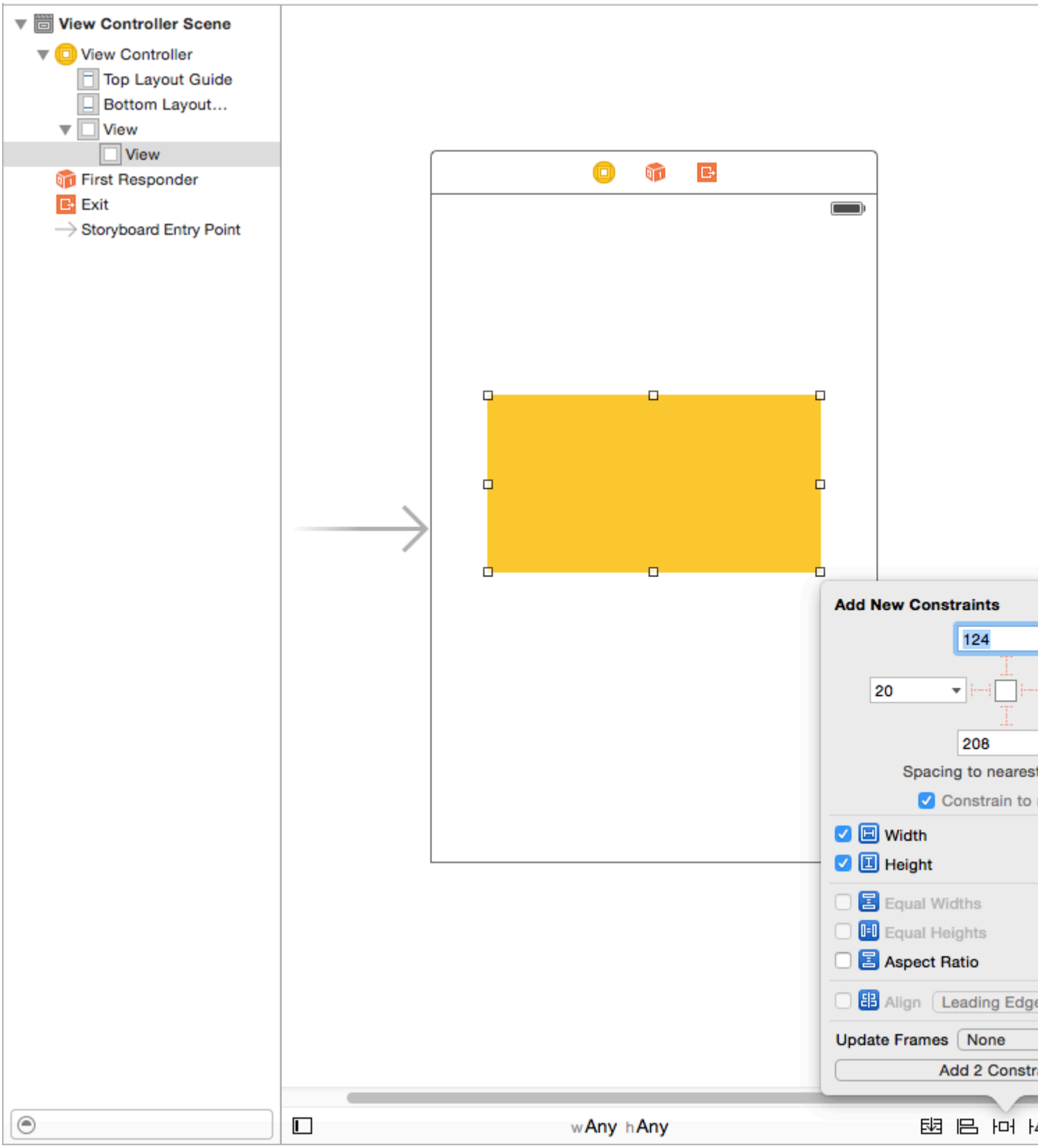
UILabel dimensione intrinseca

Dobbiamo creare una vista che avrà un prefisso dell'immagine per un testo. il testo potrebbe essere di lunghezza variabile. Dobbiamo ottenere un risultato in cui Image + text è sempre al centro di una vista genitore.

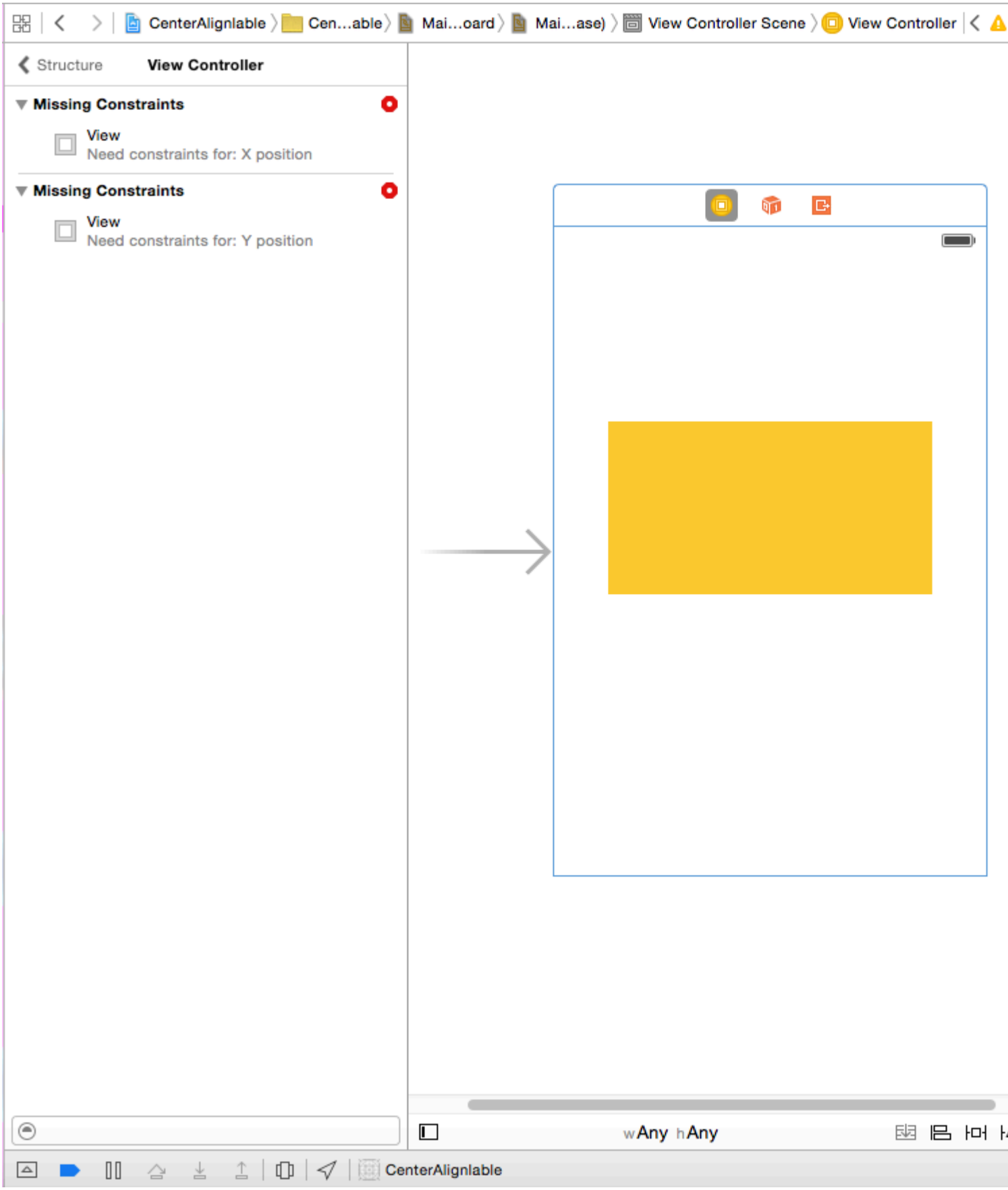
Passo 1: Prima crea un progetto a vista singola e assegnagli un nome a tua scelta e apri la visualizzazione del pacco story board. Scorri una vista con delle dimensioni ragionevoli e imposta il suo colore di sfondo su giallo. Ho ridimensionato il mio viewcontroller a 3.5 ". la vista dovrebbe sembrare qualcosa del genere



Passo 2: Ora aggiungeremo dei vincoli alla vista gialla. Per cominciare aggiungeremo i vincoli di larghezza e altezza (Aspetta un attimo non abbiamo detto che la vista avrà larghezza dinamica? Ok ci torneremo più tardi) Aggiungi la i seguenti vincoli come nell'immagine qui sotto non infastidiscono con il valore della larghezza, qualsiasi valore andrà bene per la larghezza, basta tenerlo abbastanza grande in modo che possiamo aggiungere correttamente gli autolayout.



Dopo aver aggiunto questi due vincoli, vedrai che XCode ti sta dando degli errori come nell'immagine sottostante, li vediamo e li capiamo.

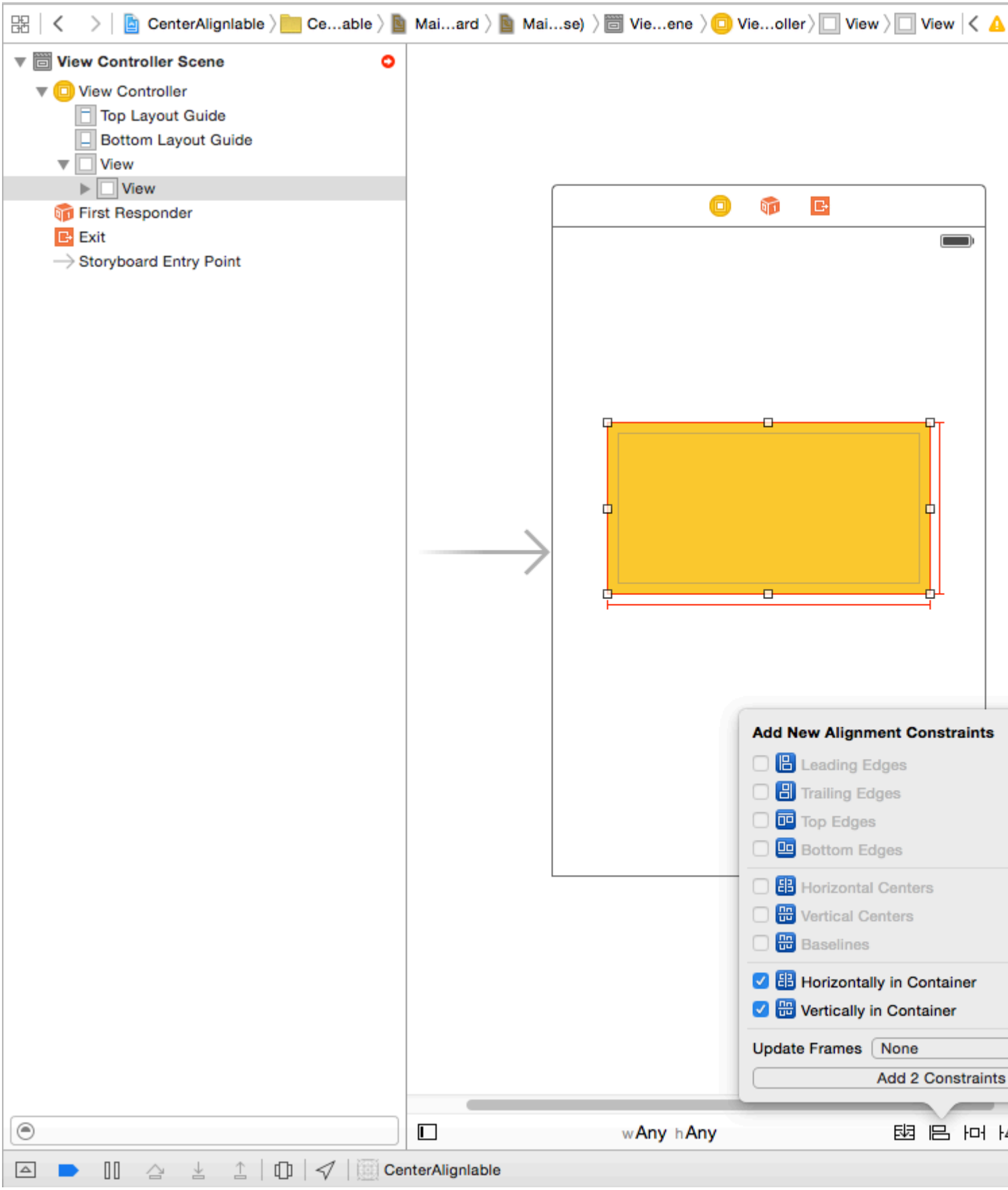


Abbiamo due errori (rosso significa errore) Come discusso sopra, rivisitiamo la parte dell'ambiguità

Vincoli mancanti: Necessità di vincoli per: posizione X: - Come discusso sopra abbiamo dato alla vista una larghezza e un'altezza in modo che i suoi "BOUNDS" siano definiti ma non abbiamo

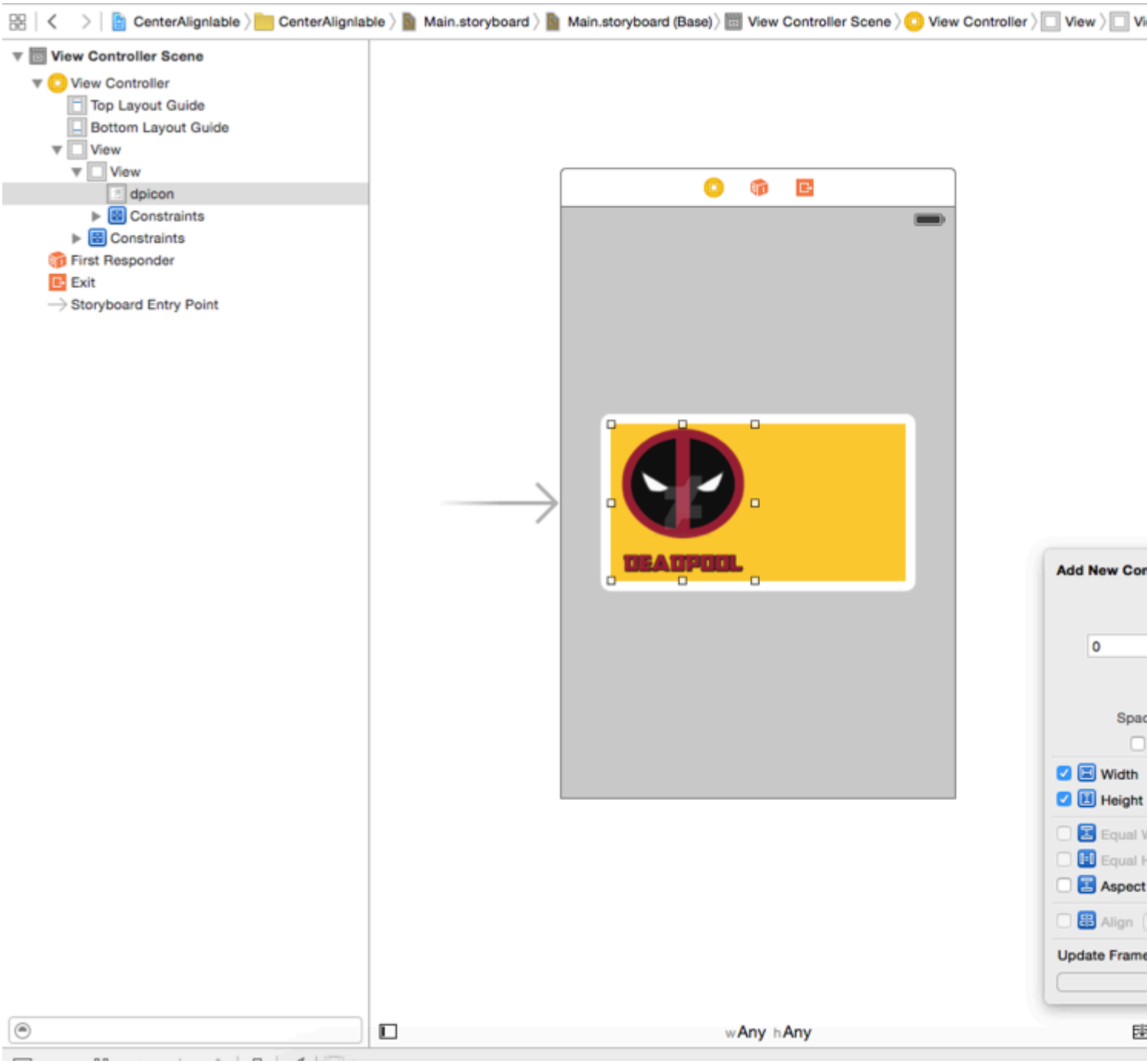
dato la sua origine quindi il suo "FRAME" non è definito. L'Autolayout non è in grado di determinare quale sarà la posizione X della nostra vista gialla

Vincoli mancanti: Necessità di vincoli per: Posizione Y: - Come discusso sopra abbiamo dato alla vista una larghezza e un'altezza in modo che i suoi "BOUNDS" siano definiti ma non abbiamo dato la sua origine in modo che il suo "FRAME" non sia definito. L'autolayout non è in grado di determinare quale sarà la posizione Y della nostra vista gialla Per risolvere questo problema dobbiamo dare un po 'di spazio all'autolayout per resettare X e Y. Poiché non possiamo impostare i fotogrammi, lo faremo in modalità autolayout. Aggiungere i seguenti vincoli come da immagine fornita qui di seguito lo spiegherò più tardi



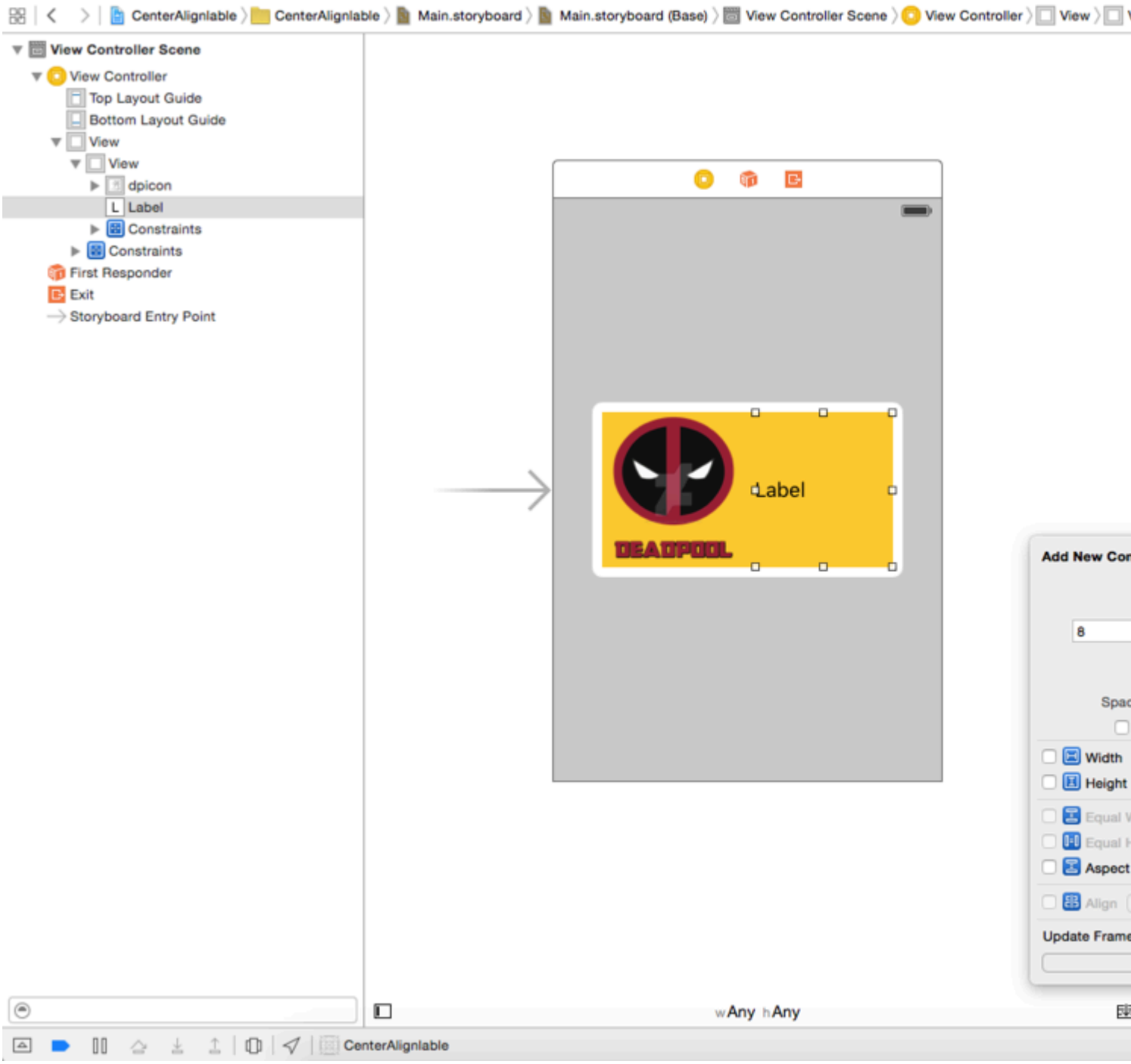
Quello che abbiamo fatto è, abbiamo aggiunto un "Centro Verticale" e un "Centro Orizzontale" questi vincoli che indicano l'autolayout che la nostra vista gialla sarà sempre al centro Orizzontalmente: quindi X in determinato è con vincolo verticale e Y è determinato. potrebbe essere necessario regolare la cornice).

Passo 3: Ormai la nostra vista di base gialla è pronta. Aggiungeremo l'immagine del prefisso come sottoview della nostra vista gialla con i seguenti vincoli. Puoi scegliere qualsiasi immagine a tua scelta.



Dato che abbiamo fissato la dimensione per la nostra immagine di prefisso avremo un'altezza di larghezza fissa per questa vista. Aggiungere i vincoli e procedere al passaggio successivo.

Step4: aggiungi un UILabel come sottoview della nostra vista gialla e aggiungi i seguenti vincoli

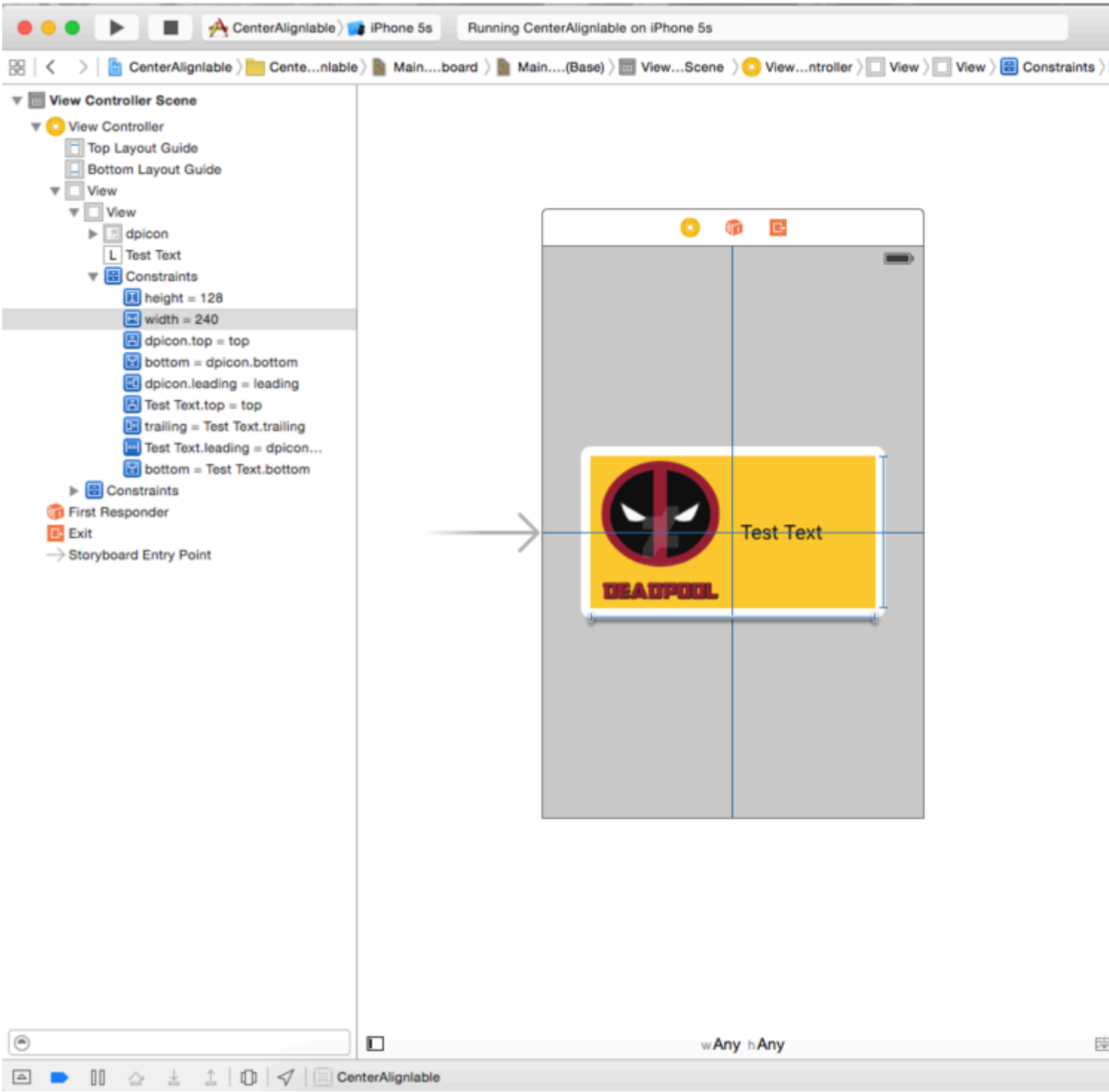


Come puoi vedere ho dato solo vincoli relativi al nostro UILabel.I suoi 8 punti dall'immagine del prefisso e 0,0,0 in alto e in basso dalla vista gialla. Visto che vogliamo che la larghezza sia dinamica non daremo limiti di larghezza o altezza .

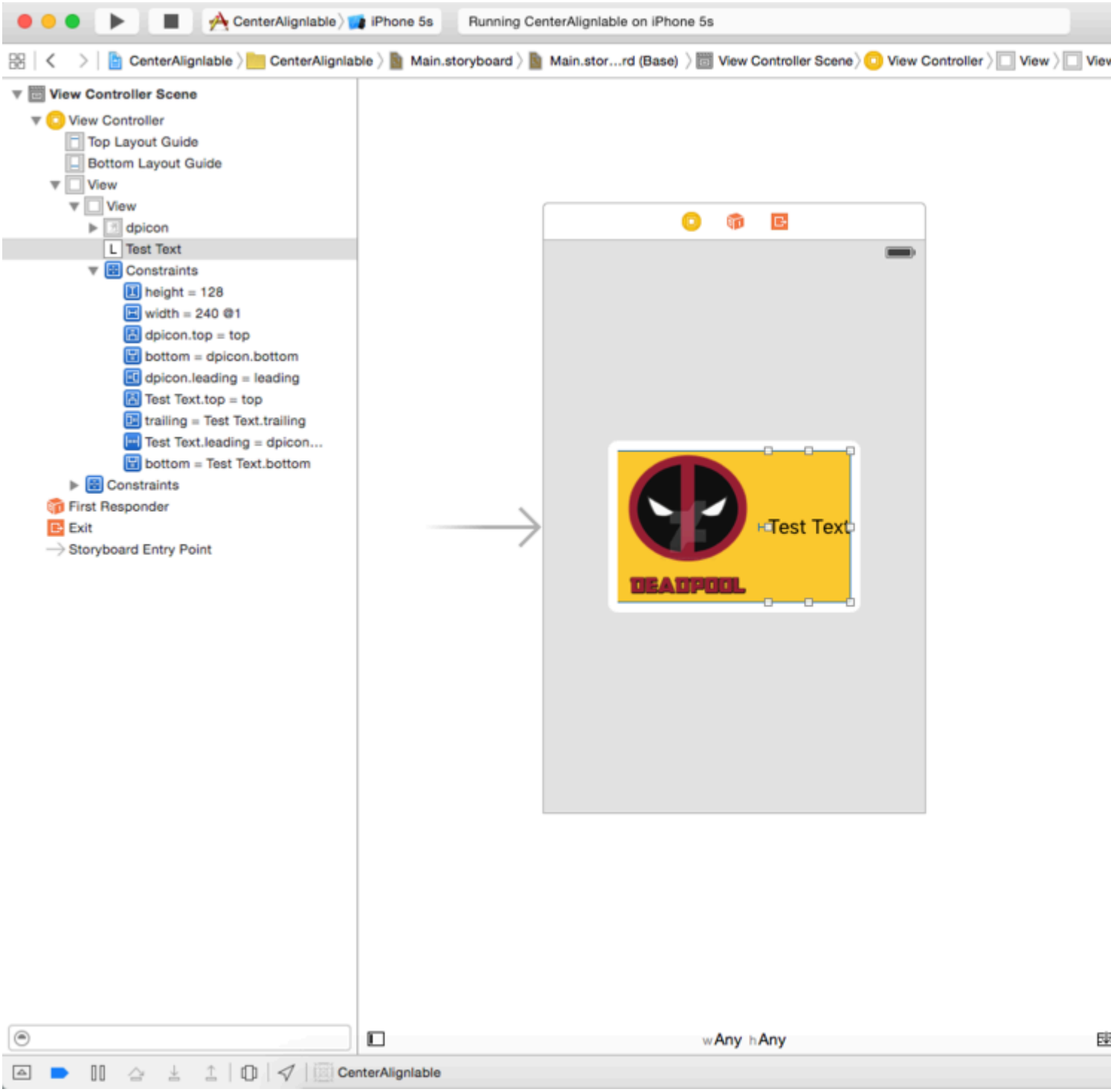
D: Perché ora non riceviamo errori, non abbiamo dato alcuna larghezza e altezza? Risposta: - Riceviamo un errore o un avvertimento solo quando il layout automatico non è in grado di risolvere qualsiasi cosa che è necessario per rendere una visualizzazione sullo schermo. La sua larghezza o origine. La nostra etichetta è relativa alla vista gialla e all'immagine del prefisso e ai relativi frame è ben definito l'autolayout è in grado di calcolare il frame della nostra etichetta.

Passo 5: Ora, se ricordiamo, realizzeremo che abbiamo dato una vista fissa a una vista gialla, ma vogliamo che sia dinamica dipendente dal testo della nostra etichetta. Quindi modificheremo il nostro vincolo di larghezza della vista gialla. Larghezza della vista gialla è necessario per risolvere

l'ambiguità ma vogliamo che venga annullato in fase di esecuzione in base al contenuto di UILabel. Quindi selezioneremo la nostra vista gialla e andremo a Dimensione ispettore e ridurrà la priorità del vincolo di larghezza a 1 in modo che sia sovrascritta. Segui l'immagine qui sotto.

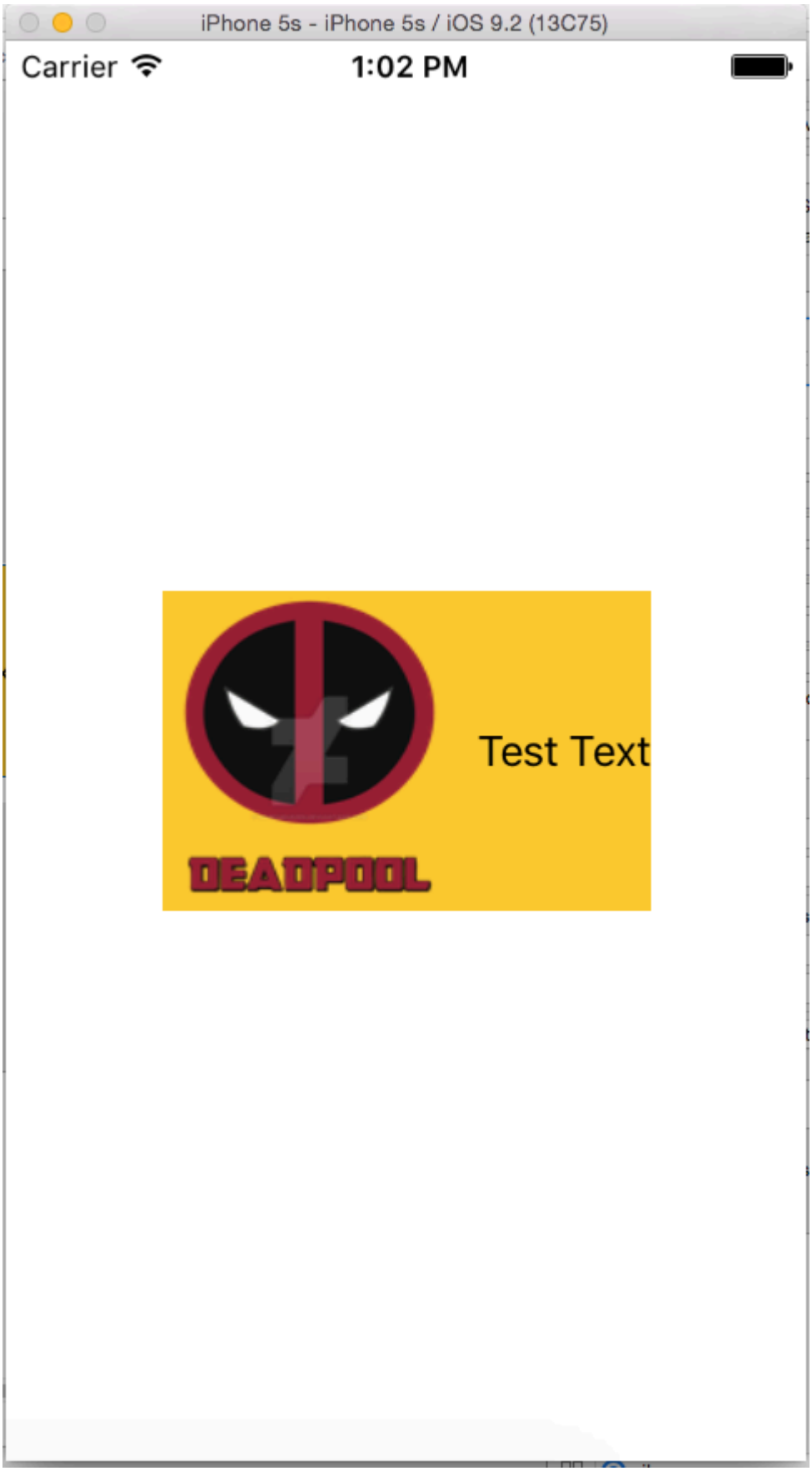


Passaggio 6: Vogliamo che UILabel si espanda in base al testo e spinga la nostra vista gialla. Così abbiamo ridotto la priorità della larghezza della vista gialla. Ora aumenteremo la priorità della resistenza alla compressione del testo della nostra UILabel. Vogliamo che la nostra vista riduca come bene, quindi aumenteremo la priorità di abbracciare i contenuti di UILabel. Segui l'immagine qui sotto



Come potete vedere abbiamo aumentato la priorità di abbraccio del contenuto a 500 e la priorità di resistenza alla compressione a 751, che supererà con successo la priorità 1 del vincolo di larghezza.

Ora costruisci e corri vedrai qualcosa come segue.



Come animare con Auto Layout

Senza il layout automatico, l'animazione si ottiene modificando il frame di una vista nel tempo. Con Auto Layout, i vincoli dettano il frame della vista, quindi devi animare i vincoli. Questa indiretta

rende l'animazione più difficile da visualizzare.

Ecco i modi per animare con Auto Layout:

1. **Modificare la costante del vincolo** dopo la creazione utilizzando chiamate periodiche (`CADisplayLink` , `dispatch_source_t` , `dispatch_after` , `NSTimer`). Quindi chiama `layoutIfNeeded` per aggiornare il vincolo. Esempio:

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. **Modificare i vincoli** e chiamare `[view layoutIfNeeded]` all'interno di un blocco di animazione. Questo interpola tra le due posizioni ignorando i vincoli durante l'animazione.

```
[UIView animateWithDuration:0.5 animations:^{
    [view layoutIfNeeded];
}]
```

3. **Cambia la priorità dei vincoli** . Questa operazione richiede meno CPU rispetto all'aggiunta e alla rimozione dei vincoli.

4. **Rimuovi tutti i vincoli e utilizza le maschere di autosizing** . Per il successivo, è necessario impostare `view.translatesAutoresizingMaskIntoConstraints = YES` .

5. **Utilizzare i vincoli che non interferiscono con l'animazione prevista** .

6. **Usa una vista del contenitore** . Posiziona la superview usando i vincoli. Quindi aggiungi una sottoview con i vincoli che non combattono l'animazione, ad esempio: un centro relativo alla superview. Questo scarica parte dei vincoli alla superview, in modo che non combattano l'animazione nella sottoview.

7. **Animare i livelli invece le viste** . Le trasformazioni di livello non attivano il layout automatico.

```
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. **Sostituisci layoutSubviews** . Chiama `[super layoutSubviews]` e perfeziona i vincoli.

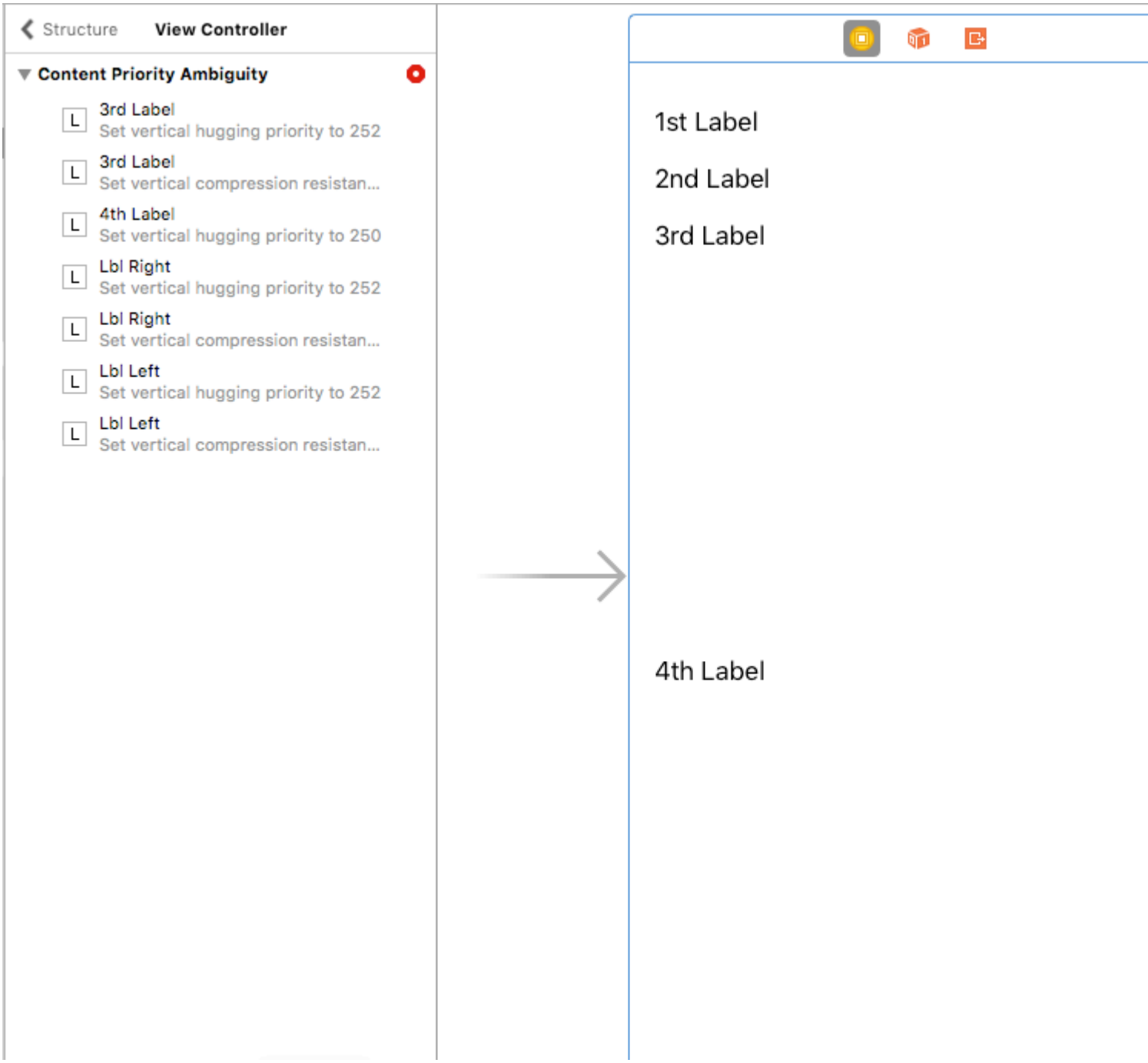
9. **Cambia il frame in viewDidLayoutSubviews** . Layout automatico viene applicato in `layoutSubviews` , quindi una volta fatto, cambiarlo in `viewDidLayoutSubviews` .

10. **Disattivare il layout automatico** e impostare le visualizzazioni manualmente. Puoi fare questo sovrascrivendo `layoutSubviews` / `layout` senza chiamare l'implementazione della super classe.

Suggerimento: se il genitore della vista animata non viene interpolato (ovvero, l'animazione salta dallo stato iniziale a quello finale), chiama `layoutIfNeeded()` nella vista più profonda che è il genitore della vista animata (in altre parole , che non è influenzato dall'animazione). Non so esattamente perché funzioni.

Risolvi conflitto priorità UILabel

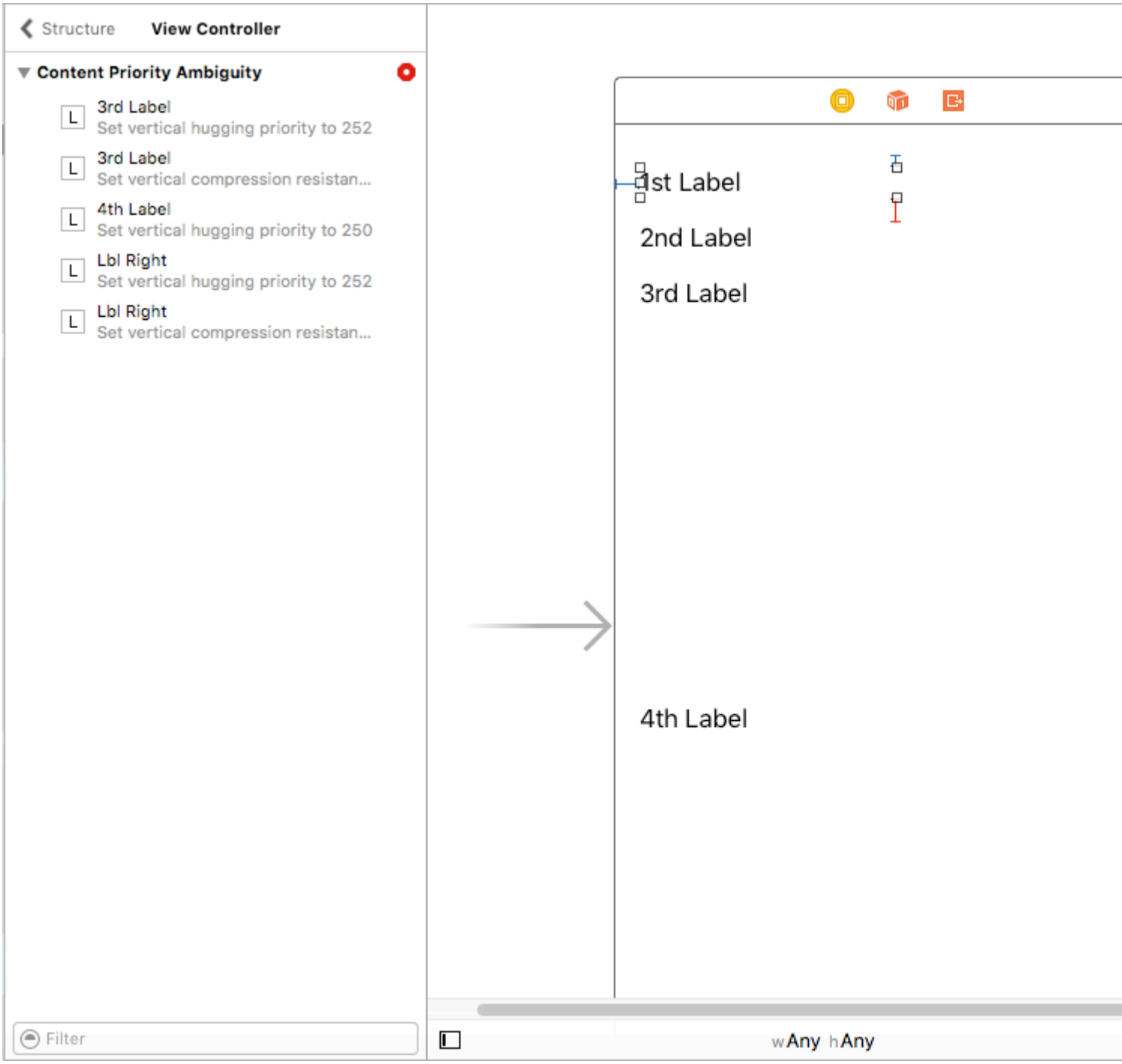
Problema : quando usi molte etichette in una vista, potresti ricevere un **avvertimento** :

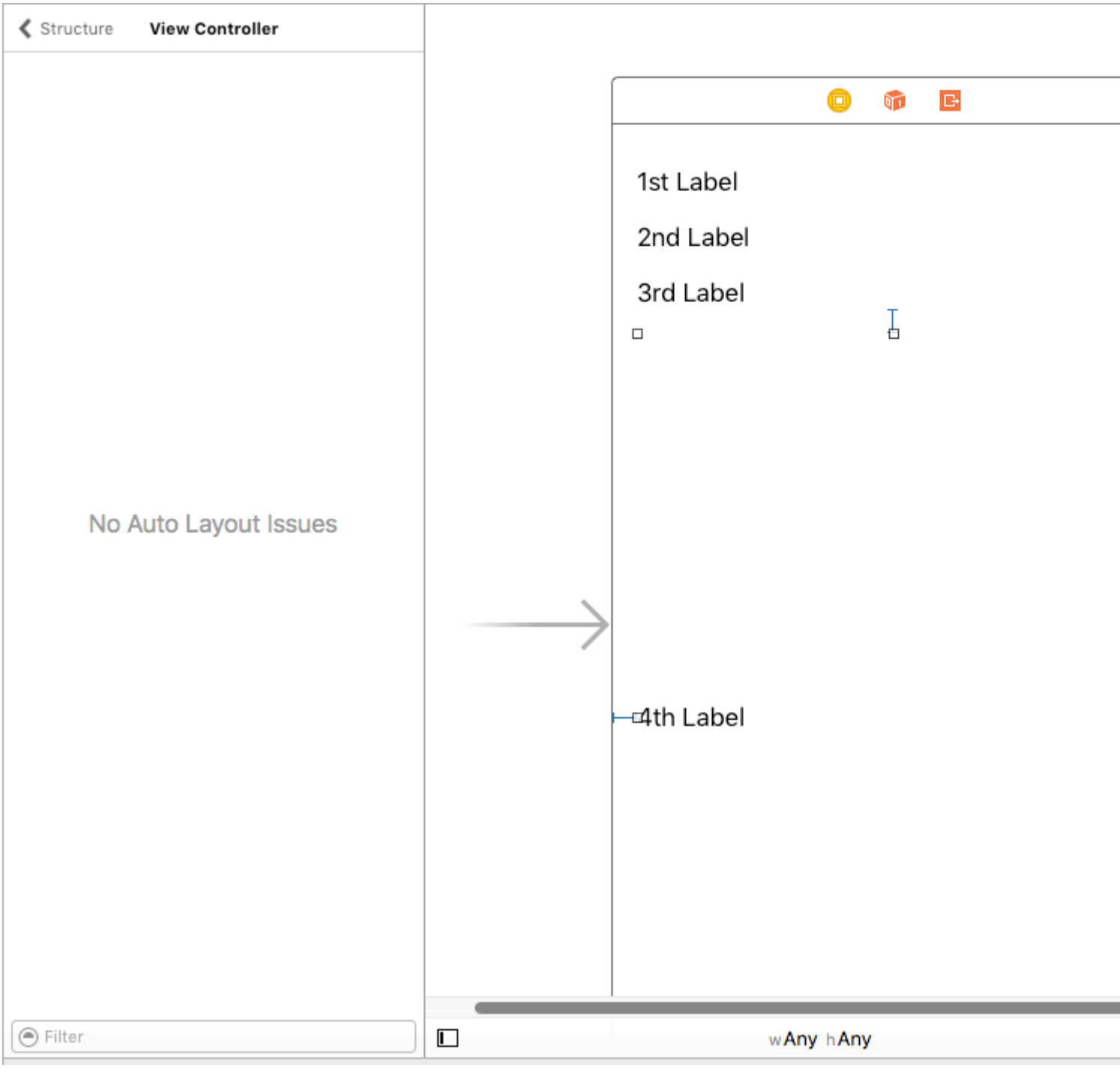


Come possiamo risolvere questo **avvertimento** ?

Soluzione : calcoliamo e definiamo le priorità in ordine. Le priorità devono essere diverse dalle etichette. Significa che è importante avrà una priorità più alta. Ad esempio, nel mio caso, ho impostato le priorità verticali per le mie etichette in questo modo:

Ho impostato la massima priorità per la 1a etichetta e la più bassa per la 4a etichetta.


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



In un ViewController, penso che sia difficile vedere l'effetto di quelle priorità. Tuttavia, è molto chiaro con UITableViewCell + stima l'altezza della cella.

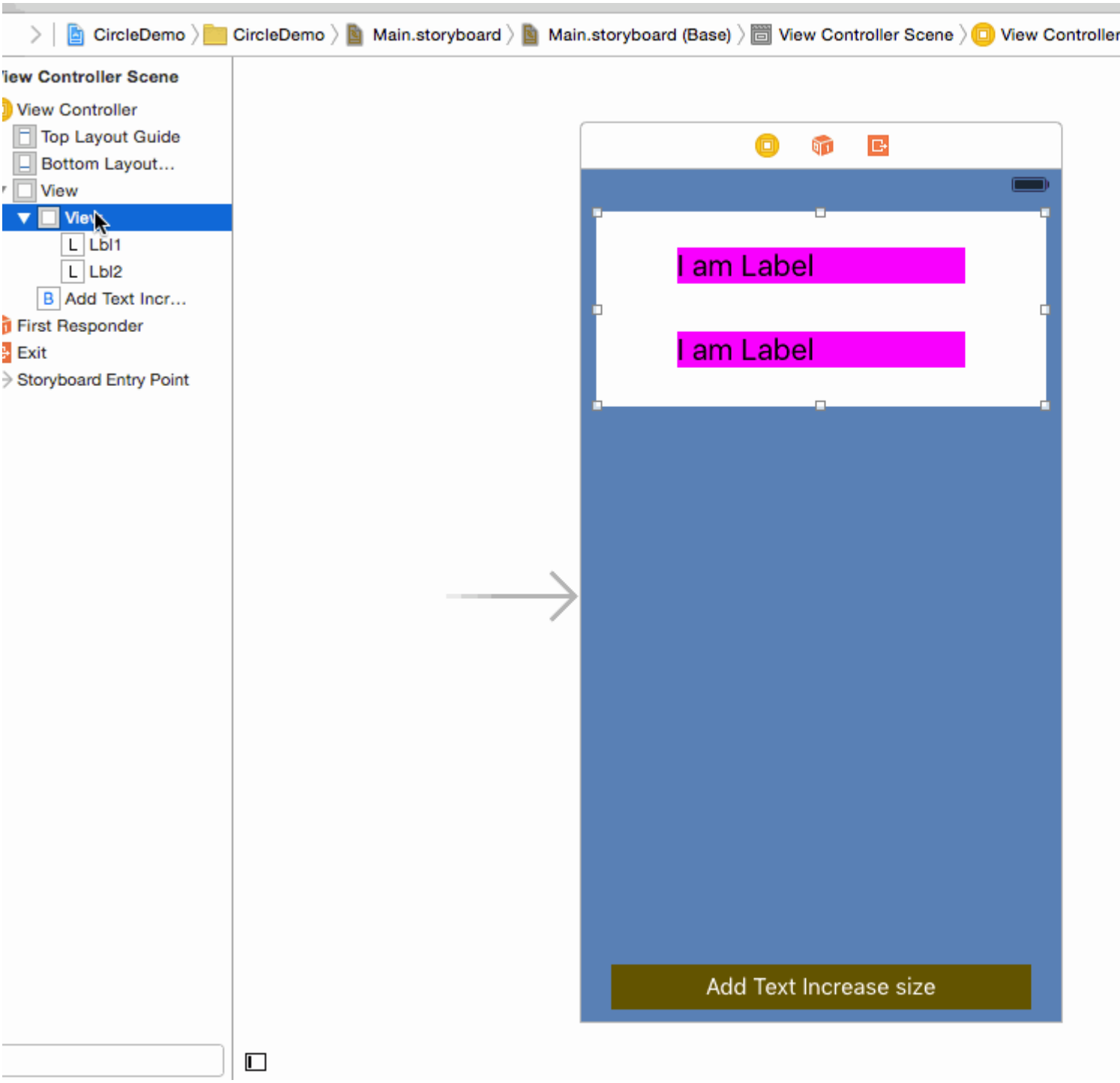
Spero che questo aiuto.

UILabel e dimensione Parentview in base al testo in UILabel

Guida passo passo: -

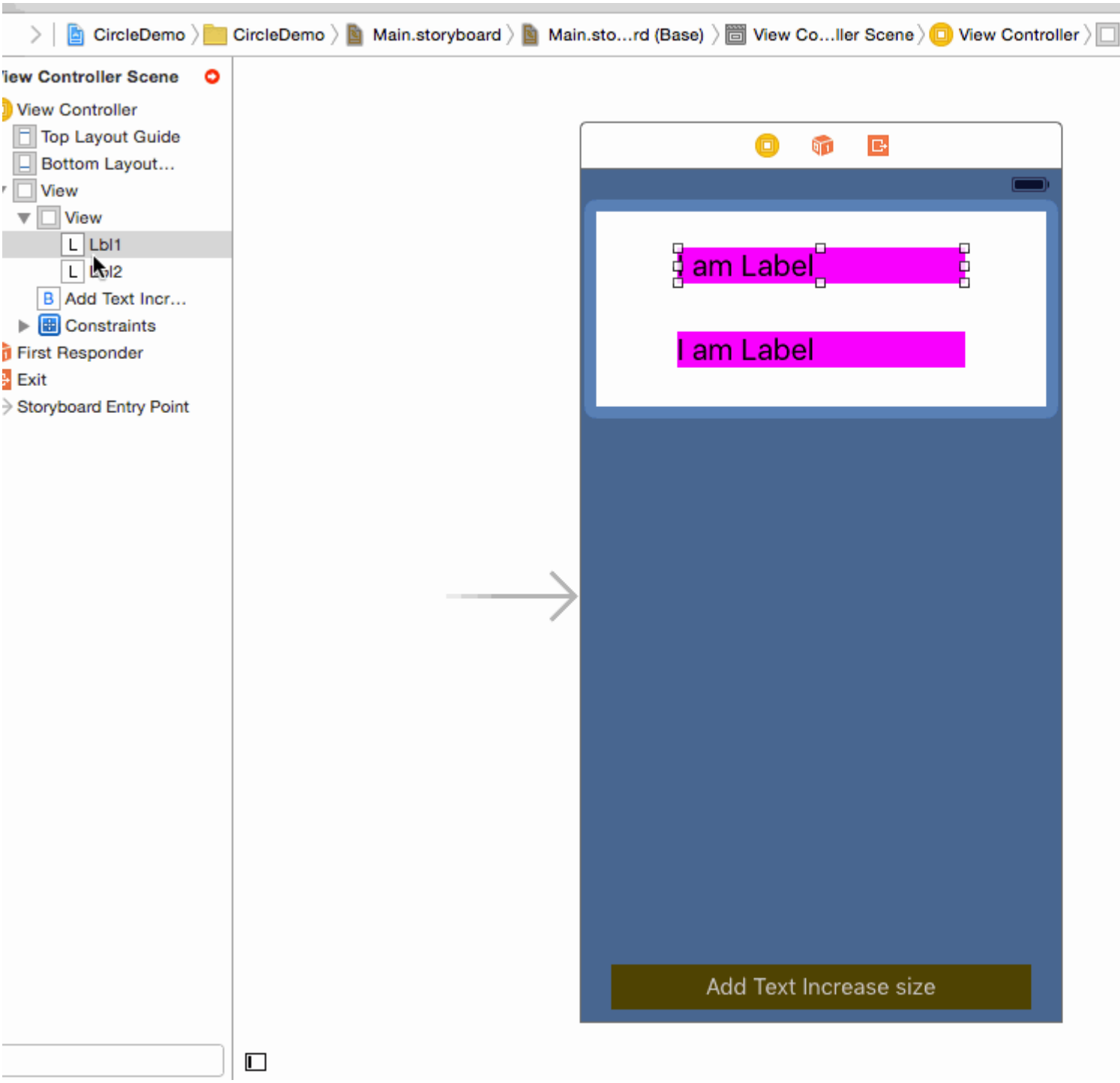
Passo 1: - Imposta il vincolo su UIView

1. Leading. 2) In alto. 3) Trailing. (Dalla schermata principale)



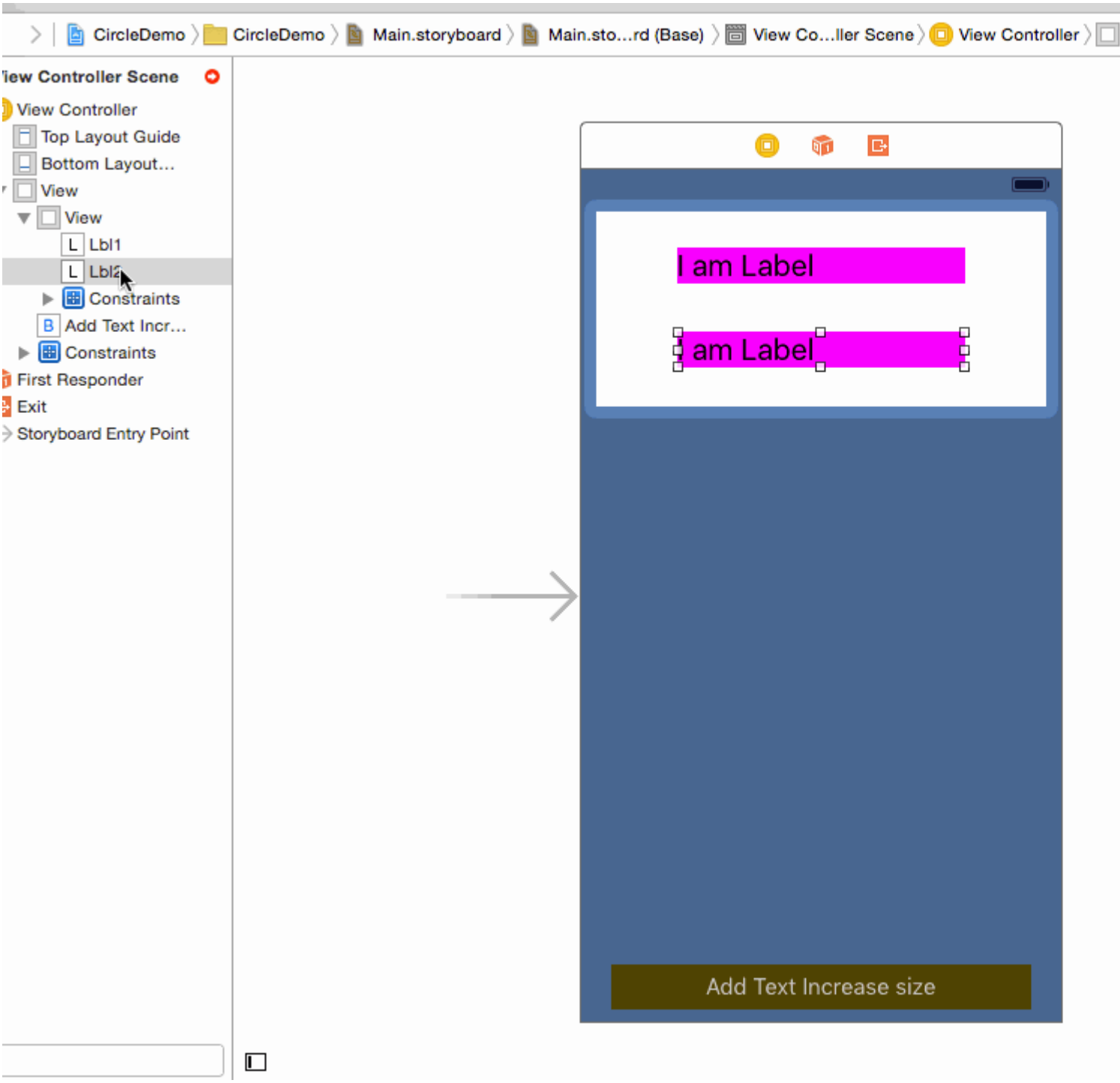
Step 2: - Imposta vincolo su Label 1

1. Leading 2) Top 3) Trailing (Dalla sua superview)

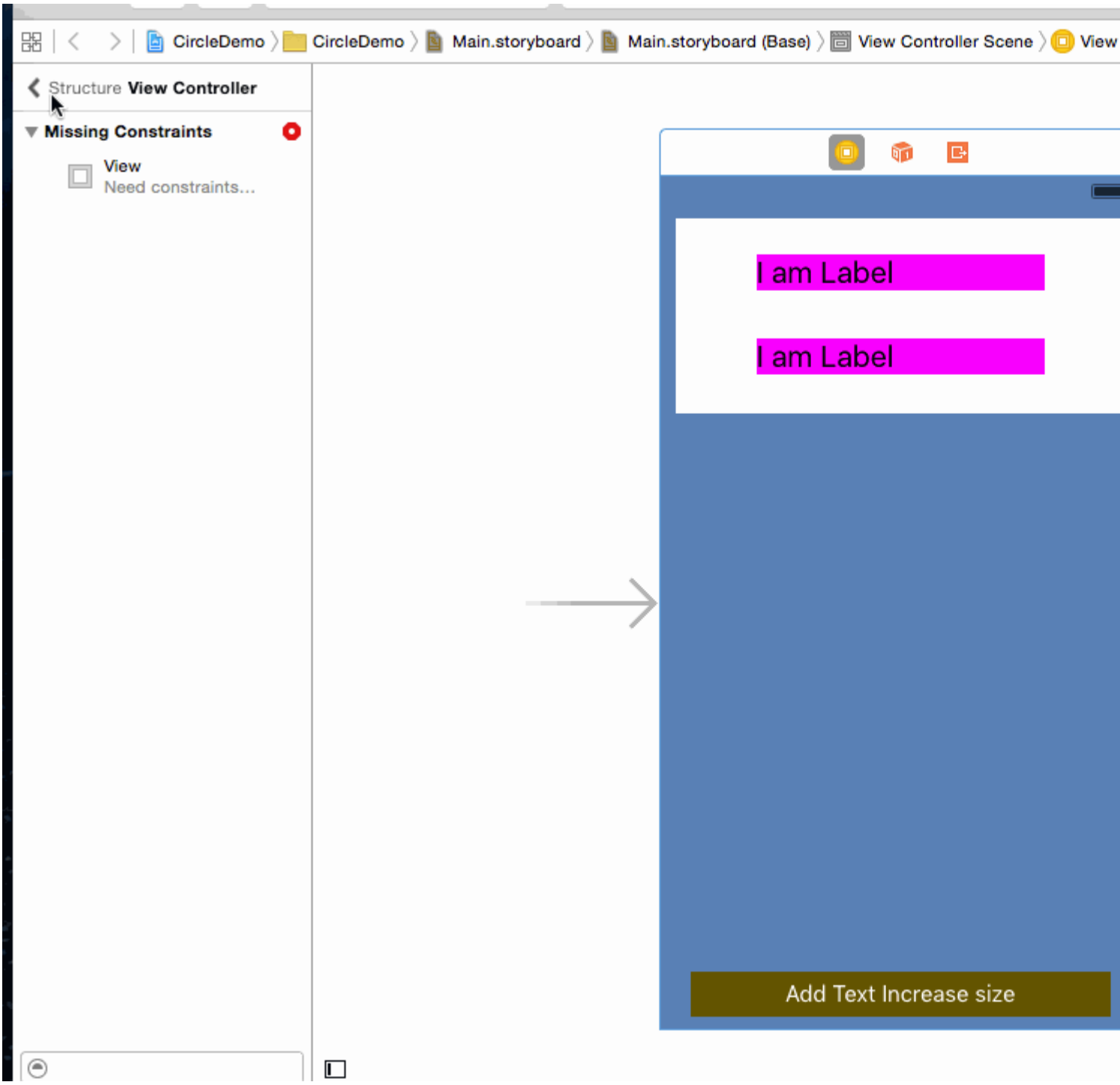


Passaggio 3: - Imposta il vincolo sull'etichetta 2

1. Leading 2) Top 3) Trailing (Dalla sua superview)

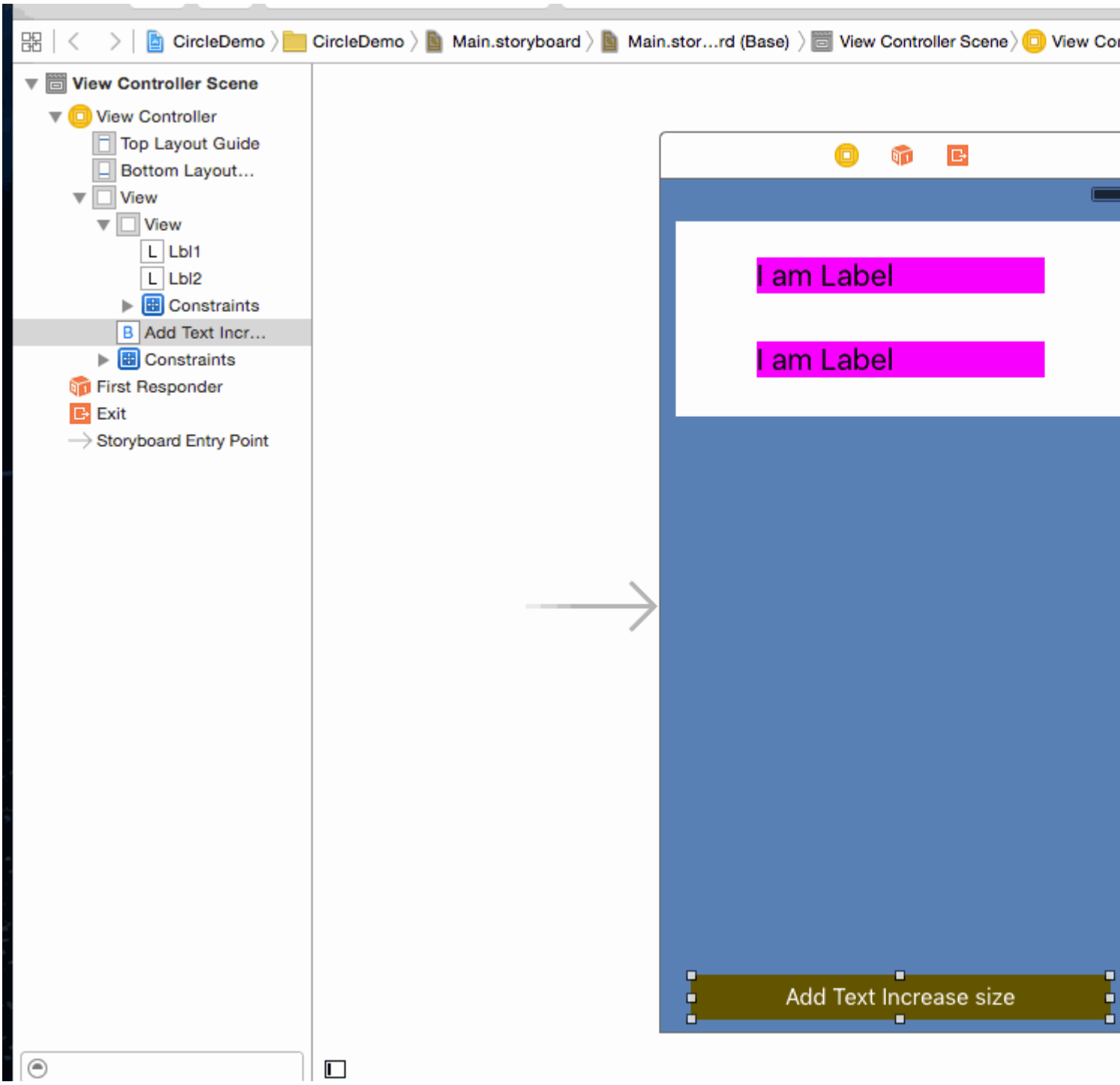


Passo 4: - Più difficile dare un fondo a UILabel da UIView.



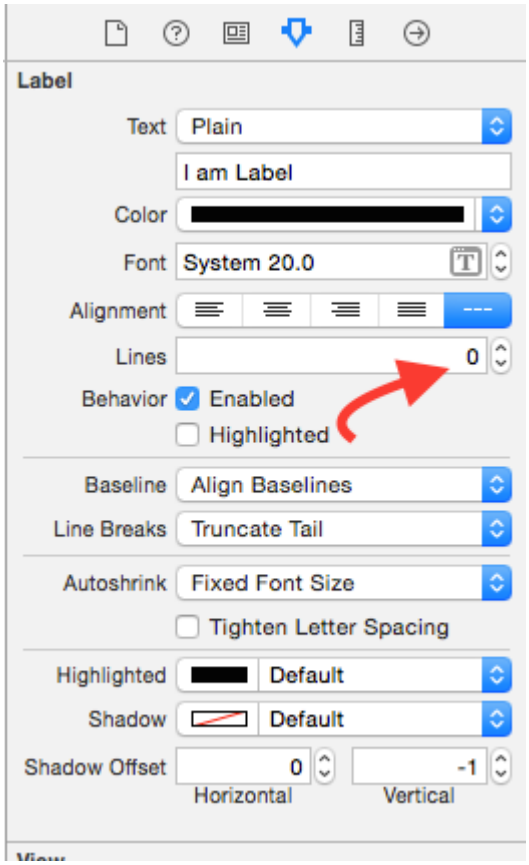
Passaggio 5: - (Facoltativo) Impostare il vincolo su UIButton

1. Leading 2) Bottom 3) Trailing 4) Fixed Height (From mainview)


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

Produzione :-

Nota: - Assicurarsi di aver impostato Numero di righe = 0 nella proprietà Label.



Spero che questa informazione sia sufficiente per comprendere l'Autoresize UIView in base all'altezza di UILabel e Autoresize UILabel in base al testo.

Nozioni di base sul linguaggio Visual Format: vincoli nel codice!

HVFL è un linguaggio progettato per limitare gli elementi dell'interfaccia utente in modo semplice e rapido. In generale, VFL ha un vantaggio rispetto alla tradizionale personalizzazione dell'interfaccia utente in Interface Builder perché è molto più leggibile, accessibile e compatto.

Ecco un esempio di VFL, in cui tre UIViews sono vincolate da sinistra a destra, riempiendo `superView.width` , con `aGradeView`

```
"H:|[bgView][aGradeView(40)][bGradeView(40)]|"
```

Ci sono due assi in cui possiamo limitare gli oggetti UI a, orizzontalmente e verticalmente.

Ogni riga di VFL inizia sempre con `H:` o `V:` Se nessuno dei due è presente, l'opzione predefinita è `H:`

Andando avanti, abbiamo una pipeline. `|` Questo simbolo, o il tubo, si riferisce alla superview. Se osservi più da vicino lo snippet del codice VFL riportato sopra, noterai due di queste pipeline.

Questo significa le due estremità orizzontali del superview, il limite esterno e i confini esterni.

Poi vedrai alcune parentesi quadre, all'interno del primo set di parentesi quadre, abbiamo `bgView` . Quando abbiamo parentesi quadre, si riferisce a un elemento dell'interfaccia utente, ora ti potresti chiedere come stabiliamo un collegamento tra il nome e l'effettivo elemento dell'interfaccia utente,

forse uno sbocco?

Lo copro alla fine del post.

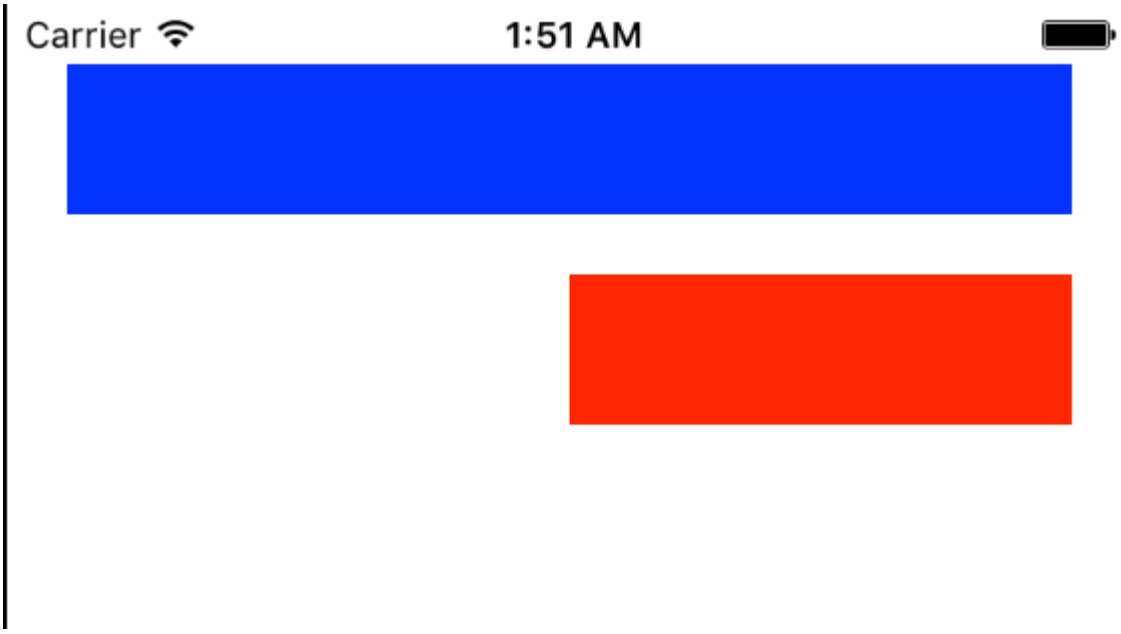
Se dai un'occhiata alla seconda coppia di parentesi quadre `[aGradeView(50)]` , abbiamo alcune parentesi incapsulate all'interno, quando è presente, definisce la larghezza / altezza in base agli assi, che in questo caso è 50 pixel in larghezza.

Le prime parentesi quadre `[bgView]` non hanno una larghezza esplicitamente definita, il che significa che si estenderà il più possibile.

Va bene, è tutto per le basi, più sulle cose avanzate in un altro esempio.

---

per esempio:




```
    // 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];
```

Utilizzo misto di layout automatico con layout non automatico

A volte potresti voler eseguire alcune azioni aggiuntive ai calcoli del **layout automatico** eseguiti da `UIKit` stessa.

Esempio: quando hai un `UIView` con `maskLayer` , potresti dover aggiornare `maskLayer` non appena **Layout automatico** cambia il `frame` `UIView`

```
// 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;
    ...
}
```

o se si desidera intraprendere qualche azione aggiuntiva su **Auto Layout** in `ViewController`

```
- (void)viewDidLayoutSubviews {
    [super viewDidLayoutSubviews];
    // now you can assume all your subviews are positioned/resized correctly
    self.customView.frame = self.containerView.frame;
}
```

Layout proporzionale

Vincolo creato come

```
NSLayoutConstraint(item: myView, attribute: NSLayoutAttribute.Leading, relatedBy:
NSLayoutRelation.Equal, toItem: view, attribute: NSLayoutAttribute.LeadingMargin, multiplier:
1.0, constant: 20.0)
```

o, dal punto di vista della matematica:

```
view.attribute * multiplier + constant          (1)
```

È possibile utilizzare il moltiplicatore per creare un layout proporzionale per un fattore di dimensioni diverse.

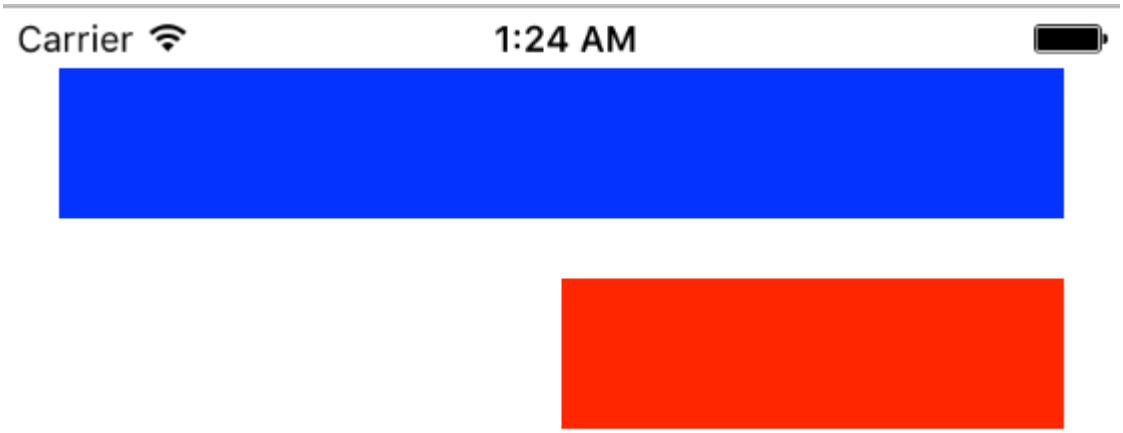
Esempio:

Turquoise View (V1) è un quadrato con larghezza proporzionale della larghezza superview con

rapporto 1: 1.1

Gary square (V2) è una sottoview di V1. Spazio inferiore impostato da costante = 60, Spazio finale impostato da moltiplicatore = 1.125 e costante = 0

Lo spazio finale è impostato in modo proporzionale, lo spazio inferiore impostato come costante.



- 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];
```

Leggi Layout automatico online: https://riptutorial.com/it/ios/topic/792/layout-automatico

Capitolo 87: Le notifiche push

Sintassi

- UIUserNotificationSettings.types: UIUserNotificationType // Una maschera di bit dei tipi di notifica che l'app può utilizzare
- UIUserNotificationSettings.categories: Set // I gruppi di azioni registrati dell'app

Parametri

Parametro	Descrizione
userInfo	Un dizionario che contiene informazioni di notifica remota, potenzialmente comprensivo di un numero di badge per l'icona dell'app, suono di avviso, messaggio di avviso, un identificatore di notifica e dati personalizzati.

Examples

Registrazione del dispositivo per le notifiche push

Per registrare il tuo dispositivo per le notifiche push, aggiungi il seguente codice al tuo file AppDelegate nel metodo `didFinishLaunchingWithOptions` :

veloce

```
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
         {}
    //----------------------------------------------------
```

Il codice sopra tenterà di comunicare con il server APN per ottenere il token del dispositivo (prerequisiti sono gli APN abilitati nel profilo di provisioning iOS).

Una volta stabilita una connessione affidabile con il server APN, il server fornisce un token dispositivo.

Dopo aver aggiunto il codice sopra, aggiungi questi metodi alla classe `AppDelegate` :

veloce

```
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);
}
```

I metodi sopra indicati sono chiamati in base allo scenario di successo o fallimento della registrazione.

Lo scenario di successo chiama:

veloce

```
func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    print("DEVICE TOKEN = \(deviceToken)")
}
```

In 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
}
}
```

Chiamate di scenario non riuscite:

veloce

```
func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    print(error)
}
```

Objective-C

```
- (void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error
```

> **Nota**
>
> Se nessuno dei metodi sopra indicati viene chiamato, il tuo dispositivo non è in grado di creare una connessione affidabile con il server APN, il che potrebbe essere dovuto a problemi di accesso a Internet.

Verifica se la tua app è già registrata per la notifica push

veloce

```
let isPushEnabled = UIApplication.sharedApplication().isRegisteredForRemoteNotifications()
```

Registrazione per la notifica push (non interattiva)

Si raccomanda di aggiungere la logica di registrazione per la notifica push in `AppDelegate.swift` quando le funzioni di callback (successo, fallimento) saranno chiamate loro. Per registrarti, fai quanto segue:

```
let application = UIApplication.sharedApplication()
let settings = UIUserNotificationSettings(forTypes: [.Alert, .Badge, .Sound], categories: nil)
application.registerUserNotificationSettings(settings)
```

Quindi la funzione di callback ha `didRegisterUserNotificationSettings` e, in tal caso, si attiva semplicemente il registro in questo modo:

```
func application(application: UIApplication, didRegisterUserNotificationSettings
notificationSettings: UIUserNotificationSettings) {
    application.registerForRemoteNotifications()
}
```

In questo caso verrà visualizzato un avviso di sistema che richiede la ricezione delle notifiche push. Una delle seguenti funzioni di callback sarà chiamata:

```
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)")

}
```

In casi molto rari, non vengono chiamate funzioni di callback di successo o di errore. Ciò accade quando si verificano problemi di connessione a Internet o l'APNS Sandbox non funziona. Il sistema esegue una chiamata API a APNS per eseguire alcune verifiche, in caso contrario non verranno richiamate le due funzioni di callback. Visita lo stato del sistema Apple per assicurarti che sia corretto.

Gestione della notifica push

Una volta che l'utente fa clic su una notifica push, verrà richiamata la seguente funzione di callback. Puoi analizzare il JSON per ottenere informazioni specifiche inviate dal back-end che ti aiuteranno nel deep linking:

veloce

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject
```

```
: AnyObject]) {
    print("Received notification: \(userInfo)")
}
```

Obiettivo 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);
}
```

Registrazione dell'app ID da utilizzare con le notifiche push

Cose di cui hai bisogno

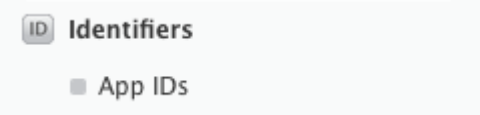
- Appartenenza al programma per sviluppatori Apple a pagamento
- Un ID app e un identificativo validi per la tua app (come com.example.MyApp) che non viene utilizzato prima da nessuna parte
- Accesso a developer.apple.com e al Centro membri
- Un dispositivo iOS da testare (poiché le notifiche push non funzionano su Simulator)

Abilitazione dell'accesso APN per l'ID app in Apple Developer Center

1- Accedi al Member Center developer.apple.com (il link Account nella home page)

2- Vai a "Certificati"

3- Seleziona "App ID" dal pannello di sinistra



4- Clicca su "+" in alto a destra

5- Aggiungi l'ID app con l'opzione Notifiche push selezionata

6- Fare clic su ID app creato e selezionare Modifica

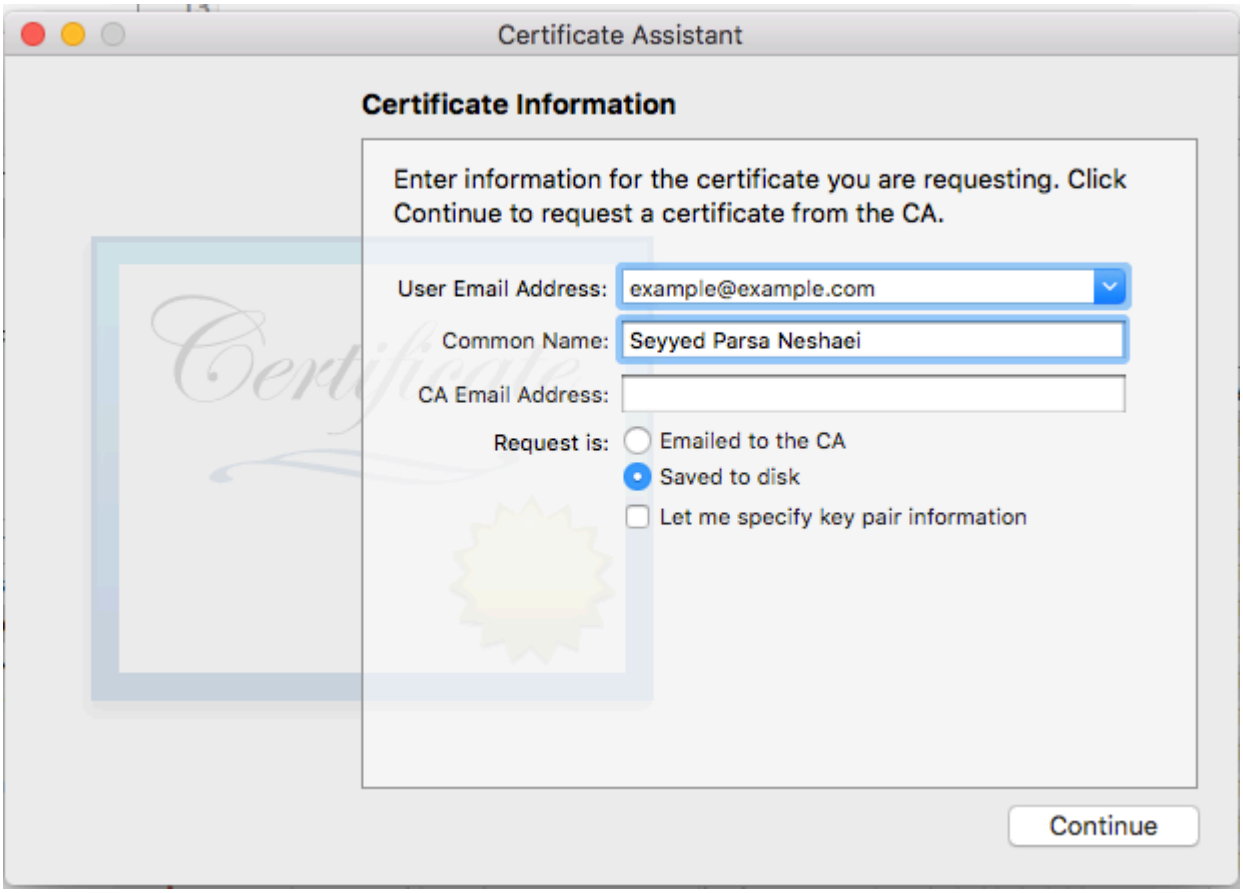
7- Fare clic su Configura nel pannello Notifiche push

8- Apri l'app Keychain Access sul tuo Mac

9- Dal menu Accesso portachiavi, fare clic su Assistente certificato -> Richiedi un certificato da un'autorità di certificazione

10- Inserisci la tua posta nel primo campo di testo

11- Inserisci il tuo nome nel secondo campo di testo



12- Lascia vuoto l'indirizzo email CA.

13- Seleziona salvato su disco anziché inviato via email alla CA.

14- Fare clic su Continua e caricare il file generato

15- Scarica il file generato da Apple e aprilo mentre Accesso Portachiavi è aperto

Abilitazione dell'accesso APN in Xcode

1- Seleziona il tuo progetto

2- scheda Apri capacità

3- Trova notifiche push e accendilo

4-Trova le modalità di background, accendilo e controlla le notifiche remote

Annullamento della registrazione da notifiche push

Per annullare la registrazione delle notifiche remote in modo programmatico, è possibile utilizzare

Objective-C

```
[[UIApplication sharedApplication] unregisterForRemoteNotifications];
```

veloce

```
UIApplication.sharedApplication().unregisterForRemoteNotifications()
```

questo è simile all'impostazione del telefono e alla disattivazione manuale delle notifiche per l'applicazione.

NOTA: potrebbero esserci rari casi in cui è necessario (ad esempio: quando la tua app non supporta più le notifiche push)

Se si desidera consentire all'utente di disabilitare temporaneamente le notifiche. È necessario implementare un metodo per rimuovere il token del dispositivo nel database sul proprio server. altrimenti, se disattivi la Notifica solo localmente sul tuo dispositivo, il tuo server continuerà a inviare messaggi.

Impostazione del numero di badge dell'icona dell'applicazione

Usa la seguente parte di codice per impostare il numero del badge dall'interno dell'applicazione (supponiamo `someNumber` sia stato precedentemente dichiarato un numero):

Objective-C

```
[UIApplication sharedApplication].applicationIconBadgeNumber = someNumber;
```

veloce

```
UIApplication.shared.applicationIconBadgeNumber = someNumber
```

Per *rimuovere* completamente il badge, basta impostare `someNumber = 0` .

Test delle notifiche push

È sempre buona norma provare come funzionano le notifiche push anche prima di avere il lato server pronto per loro, solo per assicurarsi che tutto sia impostato correttamente dalla tua parte. È abbastanza facile inviarti una notifica push utilizzando il seguente script PHP.

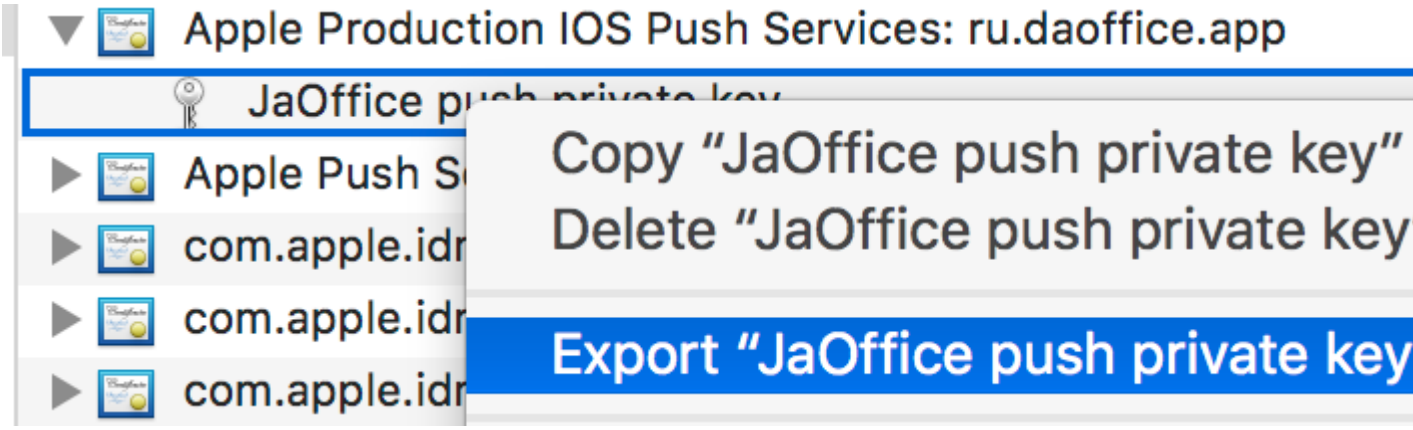
1. Salva lo script come file (ad esempio send_push.php) nella stessa cartella del tuo certificato (sviluppo o produzione)
2. Modificalo per inserire il token del dispositivo, password dal certificato
3. Scegli il percorso corretto per aprire una connessione, dev_path o prod_path (questo è dove 'Apri una connessione al server APNS' si verifica nello script)
4. cd alla cartella in Terminal ed esegui il comando 'php send_push'
5. Ricevi la notifica sul tuo dispositivo

```
<?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);
```

Generazione di un certificato .pem dal proprio file .cer, da passare allo sviluppatore del server

1. Salva aps.cer in una cartella
2. Apri "Accesso portachiavi" ed esporta la chiave che si trova sotto quel certificato in un file .p12 (chiamalo key.p12). Per farlo clicca con il tasto destro e seleziona Esporta. Salvalo nella stessa cartella del passaggio 1. All'esportazione ti verrà richiesta una password. Crea qualcosa e memorizzalo.



3. cd in quella cartella in Terminale ed esegui i seguenti comandi:
4. Convertire .cer in un certificato .pem

```
openssl x509 -in aps.cer -inform der -out aps.pem
```

5. Converti la tua chiave in formato .pem. Per aprire la chiave, inserisci la password che hai esportato dal portachiavi, nel passaggio 2. Quindi, inserisci un'altra password che proteggerà il file esportato. Ti verrà richiesto di inserirlo due volte per conferma.

```
openssl pkcs12 -nocerts -out key.pem -in key.p12
```

6. Unisci i file in un file finale

```
cat key.pem aps.pem > final_cert.pem
```

7. Il final_cert.pem è il risultato finale. Passalo agli sviluppatori di server con la password del passaggio 5, in modo che possano utilizzare il certificato protetto.

Leggi Le notifiche push online: https://riptutorial.com/it/ios/topic/3492/le-notifiche-push

Capitolo 88: Linee guida per scegliere i migliori modelli di architettura iOS

introduzione

Demistificare MVC, MVP, MVVM e VIPER o altri modelli di progettazione per scegliere l'approccio migliore per creare un'app

Examples

Modello 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
```

Modelli 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
```

Modello 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
```

Modello 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
```

Leggi Linee guida per scegliere i migliori modelli di architettura iOS online: https://riptutorial.com/it/ios/topic/10029/linee-guida-per-scegliere-i-migliori-modelli-di-architettura-ios

Capitolo 89: Link universali

Osservazioni

1. Quando supporti i collegamenti universali, gli utenti di iOS 9 possono toccare un collegamento al tuo sito Web e essere reindirizzati senza problemi all'app installata senza passare attraverso Safari. Se la tua app non è installata, toccando un link al tuo sito web si apre il tuo sito web in Safari.
2. In generale, qualsiasi link supportato cliccato in Safari o in istanze di `UIWebView` / `WKWebView` dovrebbe aprire l'app.
3. Per iOS 9.2 e meno, funziona solo su un dispositivo. iOS 9.3 supporta anche il simulatore.
4. iOS ricorda la scelta dell'utente all'apertura di Universal Links. Se toccano il breadcrumb in alto a destra per aprire il collegamento in Safari, tutti gli altri clic li porteranno su Safari e non sull'app. Possono tornare all'apertura dell'app per impostazione predefinita, scegliendo Apri nel banner dell'applicazione sul sito Web.

Examples

Setup Server

Devi avere un server in esecuzione online. Per associare in modo sicuro la tua app iOS con un server, Apple richiede di rendere disponibile un file di configurazione, chiamato `apple-app-site-association` . Questo è un file `JSON` che descrive il dominio e le rotte supportate.

Il file `apple-app-site-association` deve essere accessibile tramite `HTTPS` , senza reindirizzamenti, all'indirizzo *https: // {domain} / apple-app-site-association* .

Il file ha il seguente aspetto:

```
{
"applinks": {
    "apps": [ ],
    "details": [
        {
            "appID": "{app_prefix}.{app_identifier}",
            "paths": [ "/path/to/content", "/path/to/other/*", "NOT /path/to/exclude" ]
        },
        {
            "appID": "TeamID.BundleID2",
            "paths": [ "*" ]
        }
    ]
}
}
```

NOTA : *non aggiungere* `.json` *al nome file di* `apple-app-site-association` *.*

Le chiavi sono le seguenti:
`apps`

: dovrebbe avere un array vuoto come valore e deve essere presente. Questo è il modo in cui Apple lo vuole.
`details` : una serie di dizionari, uno per ogni app iOS supportata dal sito web. Ogni dizionario contiene informazioni sull'app, sul team e sugli ID bundle.

Ci sono 3 modi per definire i percorsi:
`Static` : l'intero percorso supportato è hardcoded per identificare un collegamento specifico, ad esempio / static / terms
`Wildcards` : A * può essere utilizzato per abbinare percorsi dinamici, ad esempio / libri / * può corrispondere al percorso della pagina di qualsiasi autore. ? all'interno di componenti specifici del percorso, ad es. libri / 1? può essere utilizzato per abbinare qualsiasi libro il cui ID inizia con 1.
`Exclusions` : la preimpostazione di un percorso con NOT esclude tale percorso dall'abbinamento.

L'ordine in cui i percorsi sono menzionati nell'array è importante. Gli indici precedenti hanno una priorità più alta. Una volta che un percorso corrisponde, la valutazione si interrompe e altri percorsi vengono ignorati. Ogni percorso è sensibile al maiuscolo / minuscolo.

Codice del sito

Il codice del sito Web può essere trovato nella sezione gh-pages su https://github.com/vineetchoudhary/iOS-Universal-Links/tree/gh-pages

Supporto di più domini

Ogni dominio supportato nell'app deve rendere disponibile il proprio file di associazione sito Apple-app. Se il contenuto servito da ciascun dominio è diverso, anche il contenuto del file cambierà per supportare i rispettivi percorsi. In caso contrario, è possibile utilizzare lo stesso file, ma deve essere accessibile in tutti i domini supportati.

Firma del file App-Site-Association

Nota : *è possibile saltare questa parte se il server utilizza `HTTPS` per servire il contenuto e passare alla guida alla configurazione dell'applicazione.*

Se la tua app è indirizzata a iOS 9 e il tuo server utilizza `HTTPS` per pubblicare i contenuti, non è necessario firmare il file. In caso contrario (ad esempio quando si supporta Handoff su iOS 8), è necessario firmarlo utilizzando un certificato `SSL` da un'autorità di certificazione riconosciuta.

Nota : questo non è il certificato fornito da Apple per inviare la tua app all'App Store. Dovrebbe essere fornito da una terza parte e si consiglia di utilizzare lo stesso certificato che si utilizza per il proprio server `HTTPS` (sebbene non sia richiesto).

Per firmare il file, devi prima creare e salvare una semplice versione .txt. Quindi, nel terminale, eseguire il seguente comando:

```
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
```

Questo produrrà il file firmato nella directory corrente. `example.com.key` , `example.com.pem` e `intermediate.pem` sono i file resi disponibili dalla tua autorità di certificazione.

Nota : se il file non è firmato, dovrebbe avere un `Content-Type` of `application/json` . Altrimenti, dovrebbe essere `application/pkcs7-mime` .

Convalida il tuo server con lo strumento di convalida della ricerca di app Apple
Metti alla prova la tua pagina web per le API di ricerca di iOS 9. Inserisci un URL e Applebot eseguirà la scansione della tua pagina Web e mostrerà come ottimizzare i risultati migliori
https://search.developer.apple.com/appsearch-validation-tool/

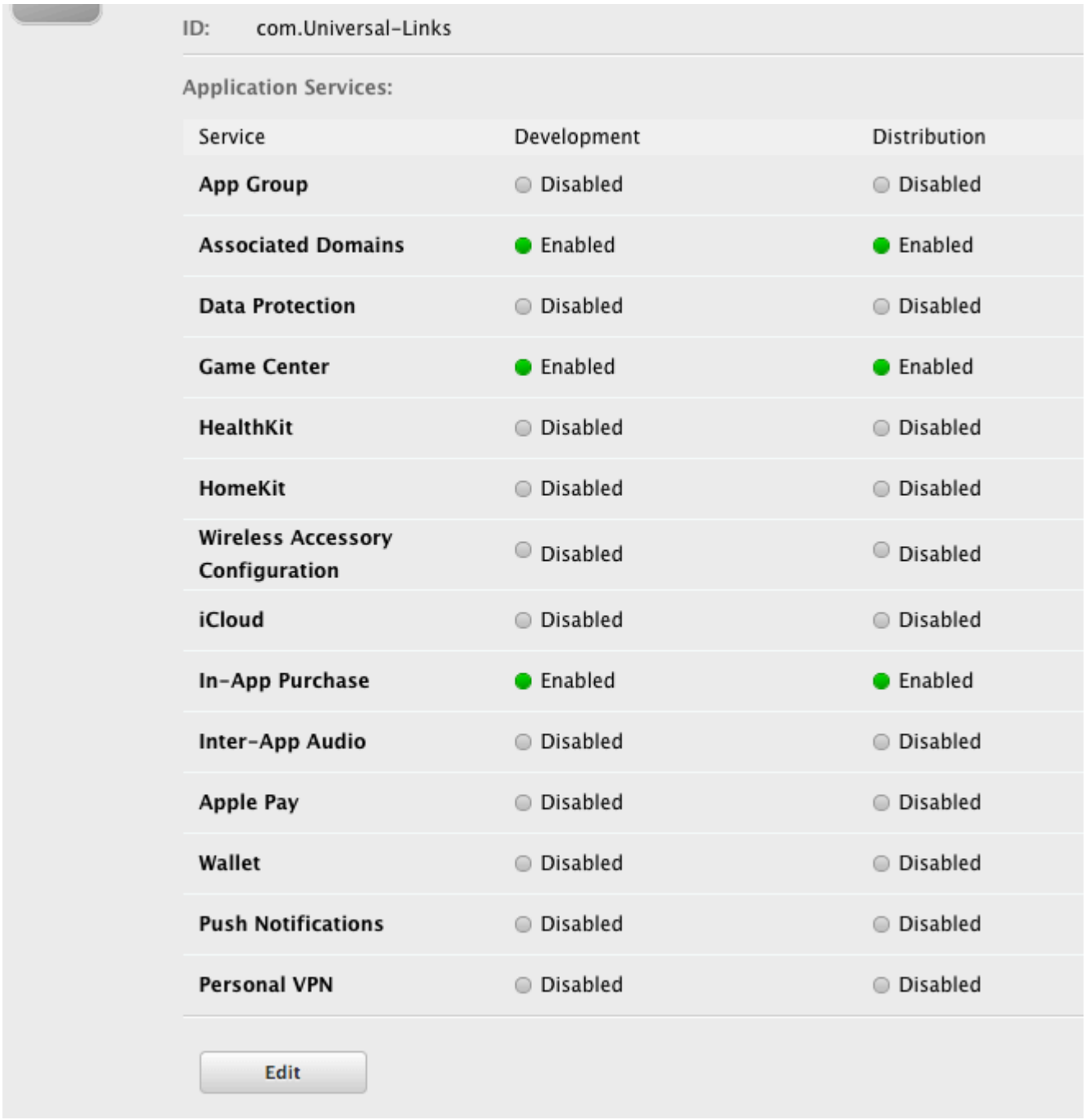
Imposta l'applicazione iOS (Abilitazione di collegamenti universali)

L'installazione sul lato dell'app richiede due cose:

1. Configurare l'autorizzazione dell'app e abilitare i collegamenti universali attivando la funzionalità Domains associati nel progetto.
2. Gestire i collegamenti in entrata nel tuo `AppDelegate` .

1. Configurazione dell'accreditamento dell'app e attivazione di collegamenti universali.

Il primo passaggio nella configurazione delle titolarità della tua app è abilitarlo per il tuo ID app. Fai questo nel Centro per gli sviluppatori degli sviluppatori Apple. Fare clic su Certificati, Identificatori e Profili e quindi Identificatori. Seleziona il tuo ID app (crealo prima se necessario), fai clic su Modifica e attiva la titolarità dei domini associati.



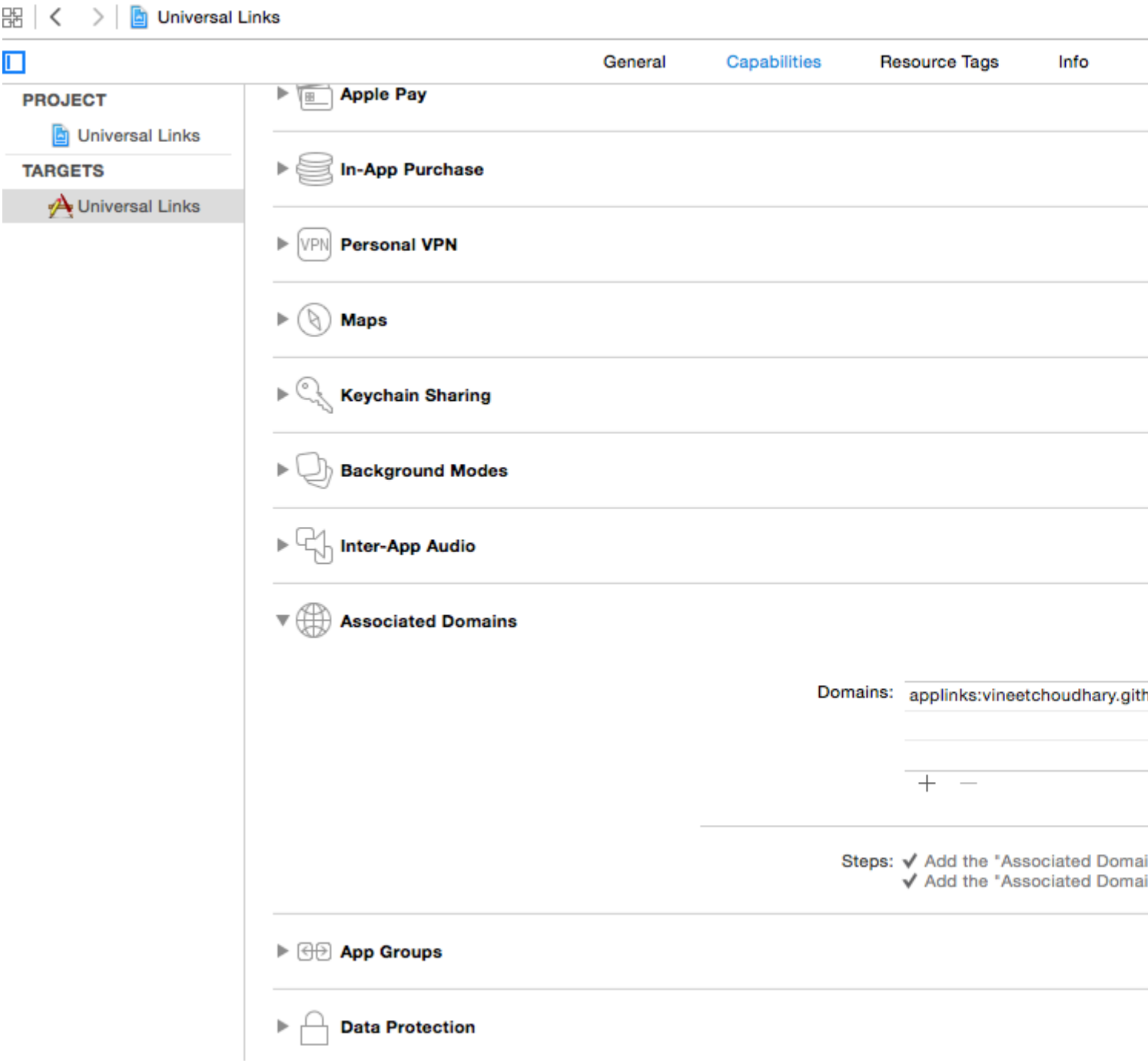
Successivamente, ottieni il prefisso e il suffisso ID app facendo clic sul rispettivo ID app.

Il prefisso e il suffisso ID app devono corrispondere a quello nel file associazione sito-apple-app.

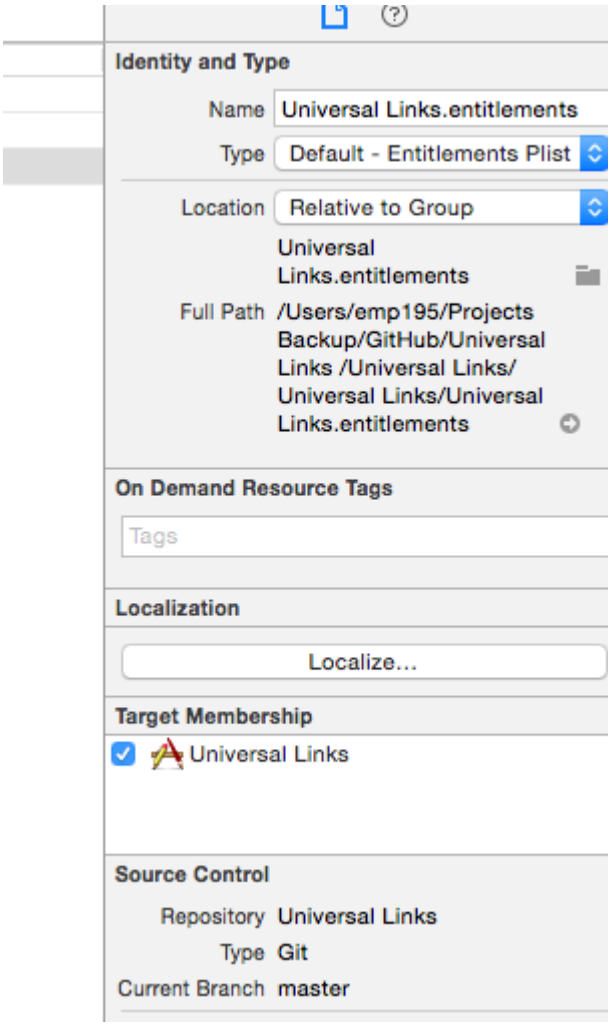
Successivamente in `Xcode` , seleziona la destinazione della tua app, fai clic su Capacità e attiva i domini associati. Aggiungi una voce per ogni dominio supportato dalla tua app, con prefisso dei **link delle app:**

Ad esempio, **collegamenti app: YourCustomDomainName.com**

Che assomiglia a questo per l'app di esempio:



Nota : assicurarsi di aver selezionato la stessa squadra e di aver inserito lo stesso ID bundle dell'ID app registrato nel Centro membri. Assicurati inoltre che il file delle autorizzazioni sia incluso in Xcode selezionando il file e in File Inspector, assicurati che il tuo target sia selezionato.



---

2. Gestire i collegamenti in entrata nel tuo AppDelegate

Tutti i reindirizzamenti da Safari all'app per i collegamenti universali passano attraverso il metodo seguente nella classe AppDelegate dell'applicazione. Analizzi questo URL per determinare l'azione corretta nell'app.

```
[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;
}
```

Swift:

```
func application(application: UIApplication, continueUserActivity userActivity:
NSUserActivity, restorationHandler: ([AnyObject]?) -> Void) -> Bool {
      if userActivity.activityType == NSUserActivityTypeBrowsingWeb {
          let url = userActivity.webpageURL!
          //handle url
      }
      return true
  }
```

Codice applicazione iOS

Il codice app può essere trovato qui .

Leggi Link universali online: https://riptutorial.com/it/ios/topic/2362/link-universali

Capitolo 90: Localizzazione

introduzione

La localizzazione è una funzione fornita da iOS che traduce la tua app in più lingue. Per la **localizzazione** , l' **internazionalizzazione** è necessaria. **L'internazionalizzazione** è un processo per rendere l'app iOS in grado di adattare culture, lingue e regioni diverse.

Examples

Localizzazione in iOS

Crea un singolo file `Localizable.strings` per ogni lingua. Il lato destro sarebbe diverso per ogni lingua. Pensala come una coppia chiave-valore:

```
"str" = "str-language";
```

Accesso a str in 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");
```

Accesso str in Swift:

```
let str = NSLocalizedString("string", comment: "language");
```

Leggi Localizzazione online: https://riptutorial.com/it/ios/topic/1579/localizzazione

Capitolo 91: Messaggistica FCM in Swift

Osservazioni

FCM: https://firebase.google.com/docs/cloud-messaging/ios/client

Examples

Inizializza FCM in Swift

segui il passaggio seguente per aggiungere FCM nel tuo progetto rapido

1- Se non hai ancora un progetto Xcode, creane uno ora. Crea un Podfile se non ne hai uno:

    $ cd directory del tuo progetto
    $ pod init

2- Aggiungi i pod che desideri installare. Puoi includere un Pod nel tuo Podfile in questo modo:

    pod 'Firebase / Core'
    pod 'Firebase / Messaging'

3- Installare i pod e aprire il file .xcworkspace per visualizzare il progetto in Xcode.

    $ pod install
    $ apri your-project.xcworkspace

4- Scarica un file GoogleService-Info.plist da plist e includilo nella tua app.

5- Carica il certificato APN su Firebase. APN Cert

6- aggiungi "Importa Firebase" nel tuo file AppDelegate del progetto

7- aggiungi questo "FIRApp.configure ()" nella tua "applicazione: didFinishLaunchingWithOptions"

8- registrati per la notifica remota

```
  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- per ottenere l'uso del token di registro

```
let token = FIRInstanceID.instanceID().token()!
```

10- e se si desidera monitorare il cambio di token, utilizzare sotto il codice nel file 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- per ricevere il messaggio da fcm aggiungere sotto il codice in appDelegate

```
func connectToFcm() {
  FIRMessaging.messaging().connectWithCompletion { (error) in
    if (error != nil) {
      print("Unable to connect with FCM. \(error)")
    } else {
      print("Connected to FCM.")
    }
  }
}
```

12- e per disconnessione

```
func applicationDidEnterBackground(application: UIApplication) {
  FIRMessaging.messaging().disconnect()
  print("Disconnected from FCM.")
}
```

nella tua appDelegate.

l'inizializzazione completa e il client è pronto a ricevere un messaggio dal pannello di fcm o inviato da un token da un server di terze parti

Leggi Messaggistica FCM in Swift online: https://riptutorial.com/it/ios/topic/7326/messaggistica-fcm-in-swift

Capitolo 92: Metodi personalizzati di selezione di UITableViewCells

introduzione

Modi avanzati per gestire le selezioni di UITableViewCell. Esempi in cui semplice `didSelect...` form `UITableViewDelegate` non è abbastanza per ottenere qualcosa.

Examples

Distinzione tra selezione singola e doppia sulla riga.

Un esempio di implementazione che offre la possibilità di rilevare se l'utente fa un singolo o doppio tocco su 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)
    }
}
```

Leggi Metodi personalizzati di selezione di UITableViewCells online:
https://riptutorial.com/it/ios/topic/9961/metodi-personalizzati-di-selezione-di-uitableviewcells

Capitolo 93: Metodi personalizzati di selezione di UITableViewCells

Examples

Distinzione tra selezione singola e doppia sulla riga.

Un esempio di implementazione di UITableView che consente di rilevare se la cella è stata sfruttata a tempo singolo o doppio.

```
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)
    }
}
```

Leggi Metodi personalizzati di selezione di UITableViewCells online:
https://riptutorial.com/it/ios/topic/9962/metodi-personalizzati-di-selezione-di-uitableviewcells

Capitolo 94: MKDistanceFormatter

Examples

Stringa dalla distanza

Dato un `CLLocationDistance` (semplicemente un `Double` rappresenta i metri), `CLLocationDistance` una stringa leggibile dall'utente:

```
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"
```

Per impostazione predefinita, rispetta le impostazioni locali dell'utente.

Unità di distanza

`import Mapkit` Impostare le `units` su uno di `.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
```

Stile unitario

Imposta `unitStyle` su uno di `.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"
```

Leggi MKDistanceFormatter online: https://riptutorial.com/it/ios/topic/6677/mkdistanceformatter

Capitolo 95: MKMapView

Examples

Aggiungi MKMapView

veloce

```
let mapView = MKMapView(frame: CGRect(x: 0, y: 0, width: 320, height: 500))
```

Si consiglia di memorizzare i MapView come una proprietà della contenente `ViewController` dal momento che si potrebbe desiderare di accedervi nelle implementazioni più complesse.

Obiettivo C

```
self.map = [[MKMapView alloc]initWithFrame:CGRectMake(0, 0, self.view.frame.size.width,
self.view.frame.size.height)];
[self.view addSubview:self.map];
```

Cambia il tipo di mappa

Esistono 5 tipi diversi (`MKMapType`), `MKMapView` può essere visualizzato.

iPhone OS 3

.standard

Visualizza una mappa stradale che mostra la posizione di tutte le strade e alcuni nomi di strade.

Swift 2

```
mapView.mapType = .Standard
```

Swift 3

```
mapView.mapType = .standard
```

Objective-C

```
_mapView.mapType = MKMapTypeStandard;
```

iPhone OS 3

.satellitare

Visualizza le immagini satellitari dell'area.

Swift 2

```
mapView.mapType = .Satellite
```

Swift 3

```
mapView.mapType = .satellite
```

Objective-C

```
_mapView.mapType = MKMapTypeSatellite;
```

iOS 9

.satelliteFlyover

Visualizza un'immagine satellitare dell'area con dati cavalcavia ove disponibili.

Swift 2

```
mapView.mapType = .SatelliteFlyover
```

Swift 3

```
mapView.mapType = .satelliteFlyover
```

Objective-C

```
_mapView.mapType = MKMapTypeSatelliteFlyover;
```

iPhone OS 3

.ibrido

Visualizza un'immagine satellitare dell'area con le informazioni sul nome della strada e della strada sovrapposte in alto.

Swift 2

```
mapView.mapType = .Hybrid
```

Swift 3

```
mapView.mapType = .hybrid
```

Objective-C

```
_mapView.mapType = MKMapTypeHybrid;
```

iOS 9

.hybridFlyover

Visualizza un'immagine satellitare ibrida con dati cavalcavia ove disponibili.

Swift 2

```
mapView.mapType = .HybridFlyover
```

Swift 3

```
mapView.mapType = .hybridFlyover
```

Objective-C

```
_mapView.mapType = MKMapTypeHybridFlyover;
```

Imposta zoom / regione per mappa

Per impostare un livello di zoom, diciamo che vogliamo ingrandire la posizione dell'utente con la posizione dell'utente come centro e 2 km di area come raggio. Quindi, usiamo il seguente codice

```
MKUserLocation *userLocation = _mapView.userLocation;
MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance
(userLocation.location.coordinate, 2000, 2000);
[_mapView setRegion:region animated:NO];
```

Implementazione della ricerca locale usando MKLocalSearch

MKLocalSearch consente agli utenti di cercare la posizione utilizzando stringhe di linguaggio naturale come "palestra". Una volta completata la ricerca, la classe restituisce un elenco di posizioni all'interno di una regione specificata che corrisponde alla stringa di ricerca.

I risultati della ricerca sono in forma di MKMapItem all'interno dell'oggetto MKLocalSearchResponse.

proviamo con l'esempio

```
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);
        }
}];
```

Tile-Overlay OpenStreetMap

In alcuni casi, potresti non voler utilizzare le mappe predefinite, fornisce Apple.

Puoi aggiungere una sovrapposizione al tuo `mapView` che contiene tessere personalizzate, ad esempio da *OpenStreetMap* .

Supponiamo che `self.mapView` sia il tuo `MKMapView` che hai già aggiunto al `ViewController` .

All'inizio, `ViewController` deve essere conforme al protocollo `MKMapViewDelegate` .

```
class MyViewController: UIViewController, MKMapViewDelegate
```

Quindi devi impostare `ViewController` come delegato di `mapView`

```
mapView.delegate = self
```

Successivamente, si configura l'overlay per la mappa. Avrai bisogno di un modello di URL per questo. L'URL dovrebbe essere simile a questo su tutti i server tile e anche se si memorizzassero i dati della mappa offline: `http://tile.openstreetmap.org/{z}/{x}/{y}.png`

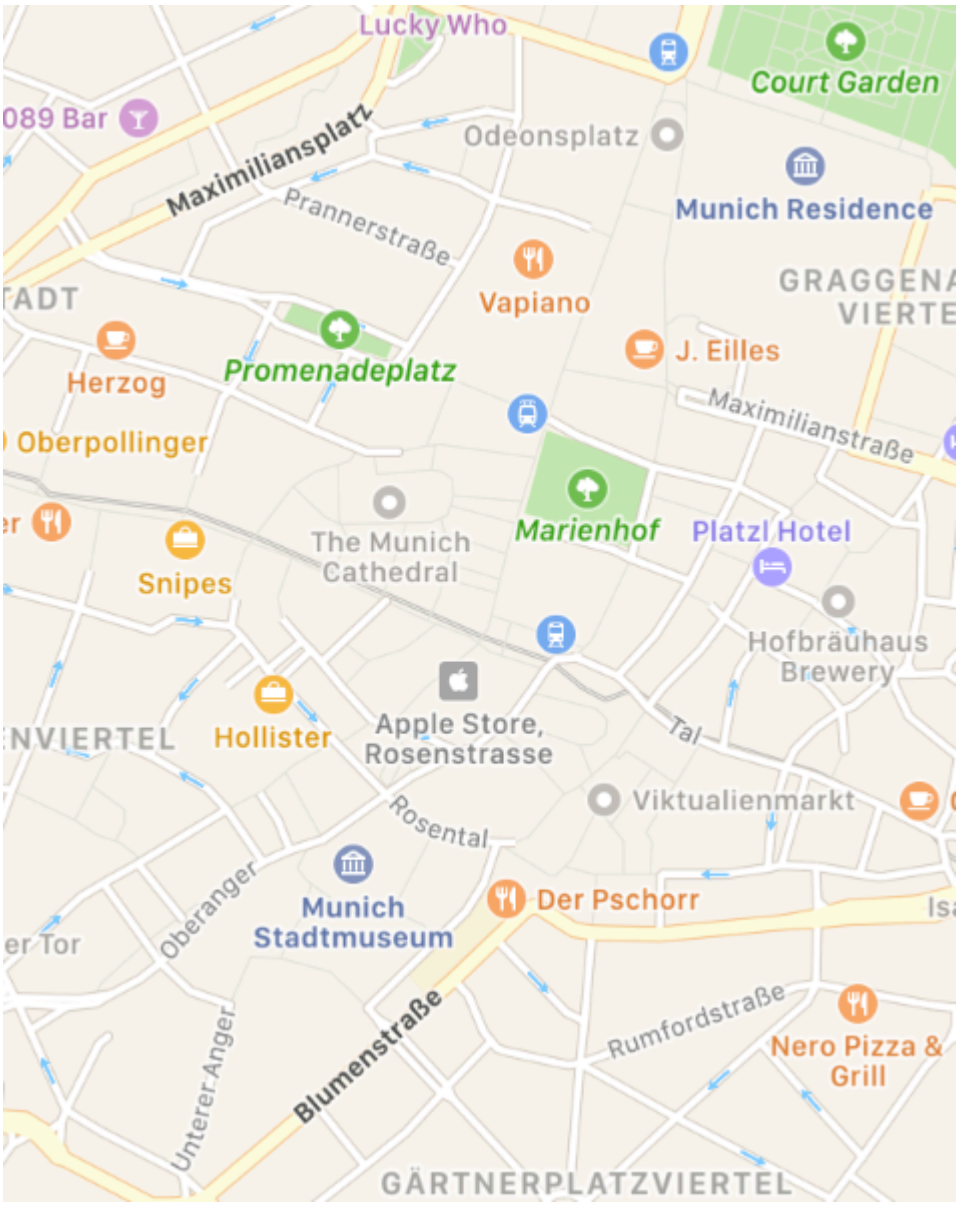
```
let urlTeplate = "http://tile.openstreetmap.org/{z}/{x}/{y}.png"
let overlay = MKTileOverlay(urlTemplate: urlTeplate)
overlay.canReplaceMapContent = true
```

Dopo aver configurato l'overlay, devi aggiungerlo a `mapView` .

```
mapView.add(overlay, level: .aboveLabels)
```

Per utilizzare mappe personalizzate, si consiglia di utilizzare `.aboveLabels` per il `level` . Altrimenti, le etichette predefinite saranno visibili sulla tua mappa personalizzata. Se vuoi vedere le etichette predefinite, puoi scegliere qui `.aboveRoads` .

Se eseguissi il tuo progetto ora, riconoscerai che la tua mappa mostrerà comunque la mappa predefinita:



Questo perché non abbiamo ancora detto `mapView` , come renderizzare l'overlay. Questo è il motivo per cui è stato necessario impostare il delegato in precedenza. Ora puoi aggiungere `func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer` al tuo controller di visualizzazione:

```
func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {
    if overlay is MKTileOverlay {
        let renderer = MKTileOverlayRenderer(overlay: overlay)
        return renderer
    } else {
        return MKTileOverlayRenderer()
    }
}
```

Ciò restituirà il corretto `MKOverlayRenderer` al tuo `mapView` . Se esegui il tuo progetto ora, dovresti vedere una mappa come questa:

Se vuoi visualizzare un'altra mappa, devi solo cambiare il modello di URL. C'è un elenco di server di tile nel Wiki OSM.

Mostra esempio UserLocation e UserTracking

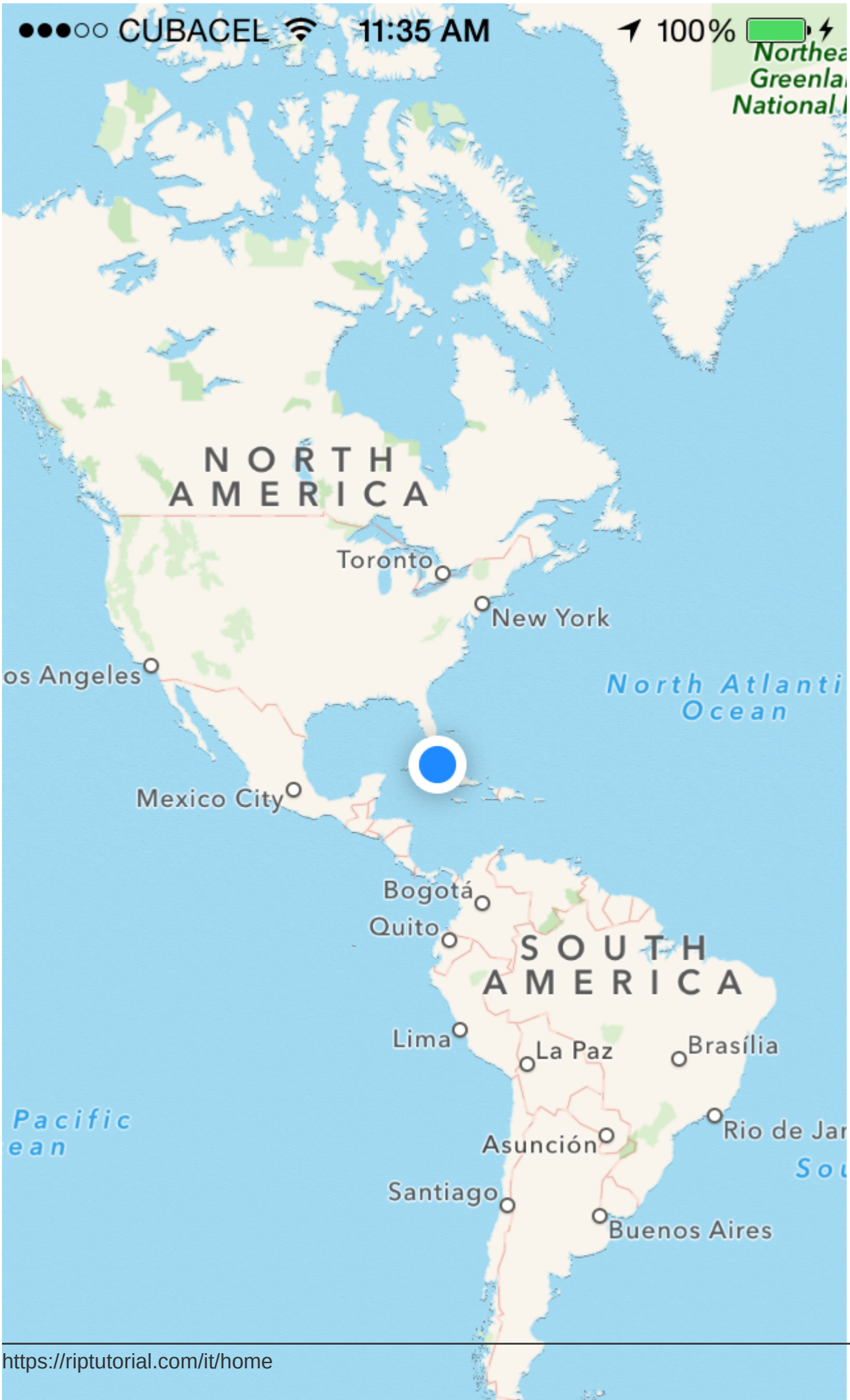
Questo mostrerà la posizione dell'utente sulla mappa

Objective-C

```
[self.map setShowsUserLocation:YES];
```

veloce

```
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

visualizzi una coordinata a livello di zoom invece di impostare una regione da mostrare. Questa funzionalità non è implementata per impostazione predefinita, quindi è necessario estendere `MKMapView` con metodi che eseguono il calcolo complesso da una *coordinata* e *un livello* di *zoom* a una `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)
        }
    }
}
```

(La versione originale di Swift 2 di Mylene Bayan è disponibile su GitHub)

Dopo aver implementato questa `extension` , puoi impostare le coordinate centrali come segue:

```
let centerCoordinate = CLLocationCoordinate2DMake(48.136315, 11.5752901) //latitude, longitude
mapView?.setCenter(centerCoordinate, zoomLevel: 15, animated: true)
```

`zoomLevel` è un valore `Double` , solitamente compreso tra `0` e `21` (che è un livello di zoom molto elevato), ma sono consentiti valori fino a `28` .

Lavorando con annotazione

Ottieni tutte le annotazioni

```
//following method returns all annotations object added on map
NSArray *allAnnotations = mapView.annotations;
```

Ottieni la vista di annotazione

```
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"];
    }
}
```

Rimuovi tutte le annotazioni

```
[mapView removeAnnotations:mapView.annotations]
```

Rimuovi annotazione singola

```
//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];

}
```

Regola il rect visibile della vista mappa per visualizzare tutte le annotazioni

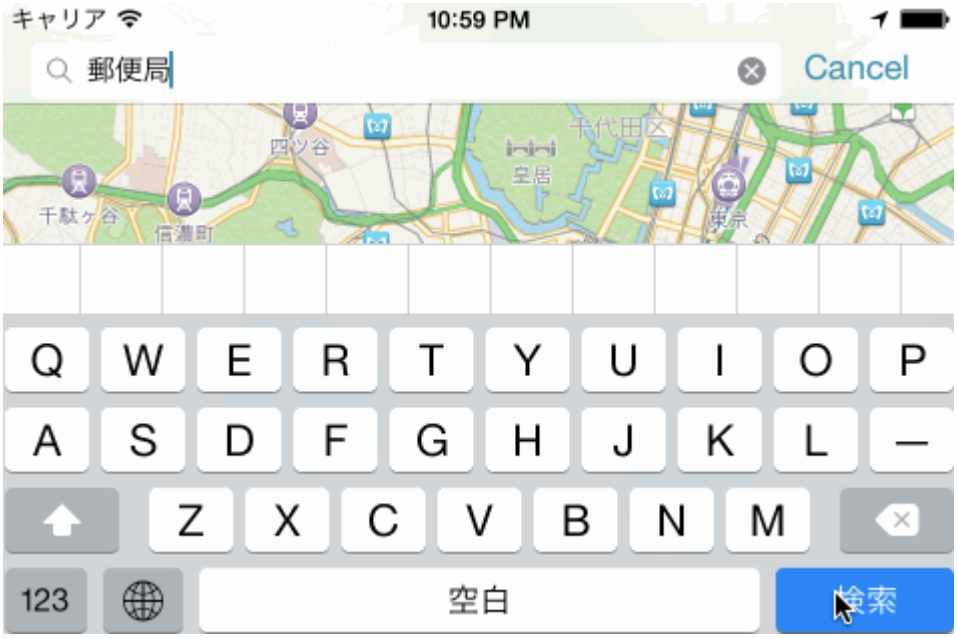
Swift:

```
mapView.showAnnotations(mapView.annotations, animated: true)
```

Objective-C:

```
[mapView showAnnotations:mapView.annotations animated:YES];
```

demo:



Leggi MKMapView online: https://riptutorial.com/it/ios/topic/915/mkmapview

Capitolo 96: Modalità di background

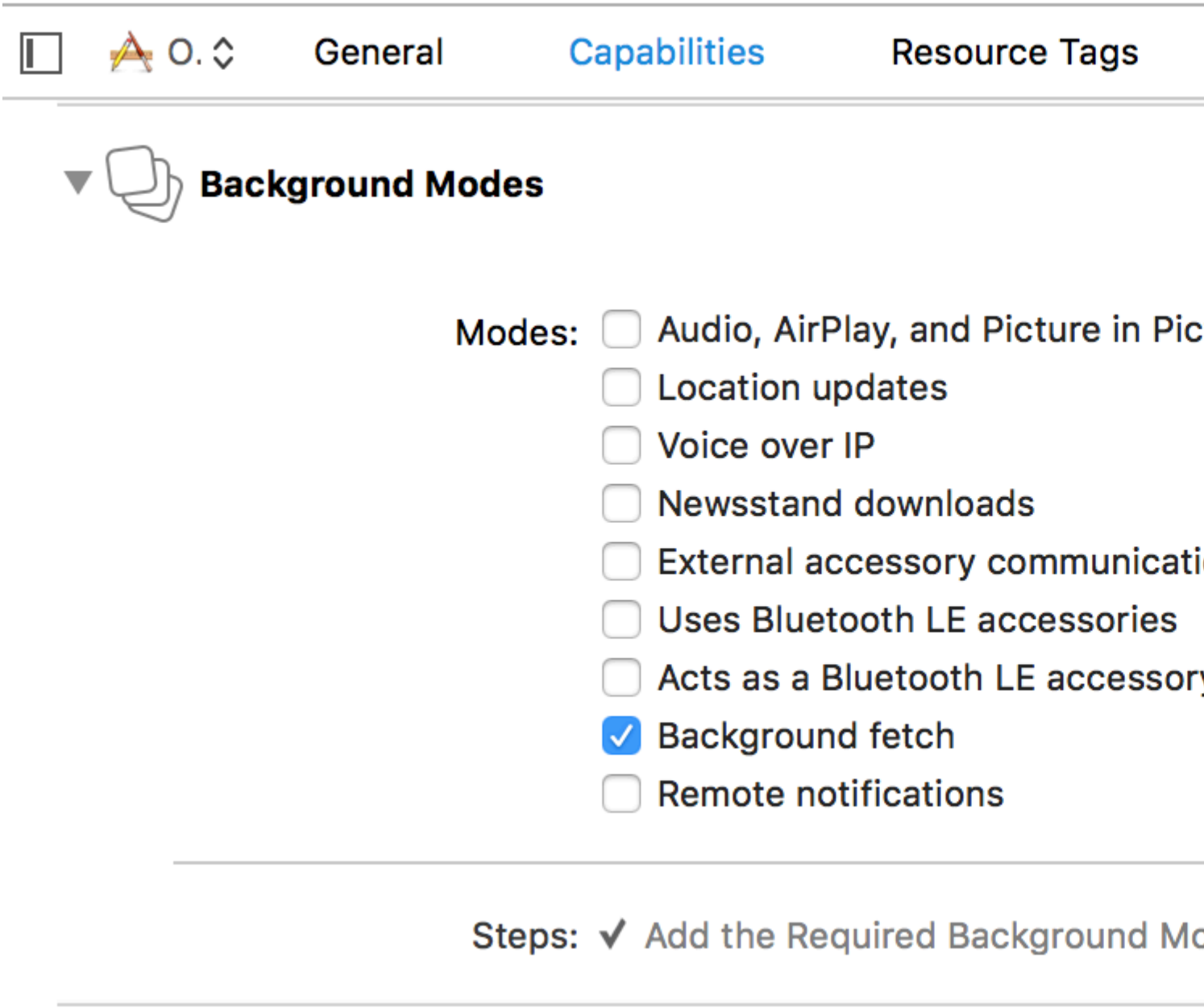
introduzione

Essere reattivi è un'esigenza per ogni app. Gli utenti desiderano avere app pronte per il loro contenuto quando vengono aperte, quindi gli sviluppatori devono utilizzare le modalità di background per rendere le proprie app più intuitive.

Examples

Attivare la funzionalità Modalità di background

1. Vai su Xcode e apri il tuo progetto.
2. Nel target dell'app, accedi alla scheda Funzionalità.
3. Attiva le modalità di background.



Fetch di sfondo

Il recupero in background è una nuova modalità che consente alla tua app di essere sempre aggiornata con le ultime informazioni riducendo al minimo l'impatto sulla batteria. È possibile scaricare i feed entro intervalli di tempo fissi con questa funzionalità.

Per iniziare:

1- Controlla la schermata delle funzionalità di recupero in background in Xcode.

2- Nel metodo `application(_:didFinishLaunchingWithOptions:)` in `AppDelegate` , aggiungere:

veloce

```
UIApplication.shared.setMinimumBackgroundFetchInterval(UIApplicationBackgroundFetchIntervalMinimum)
```

Objective-C

```
[[UIApplication shared]
setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMinimum]
```

> Invece di `UIApplicationBackgroundFetchIntervalMinimum` , è possibile utilizzare qualsiasi valore di `CGFloat` per impostare intervalli di recupero.

3- È necessario implementare l' `application(_:performFetchWithCompletionHandler:)` . Aggiungilo al tuo `AppDelegate` :

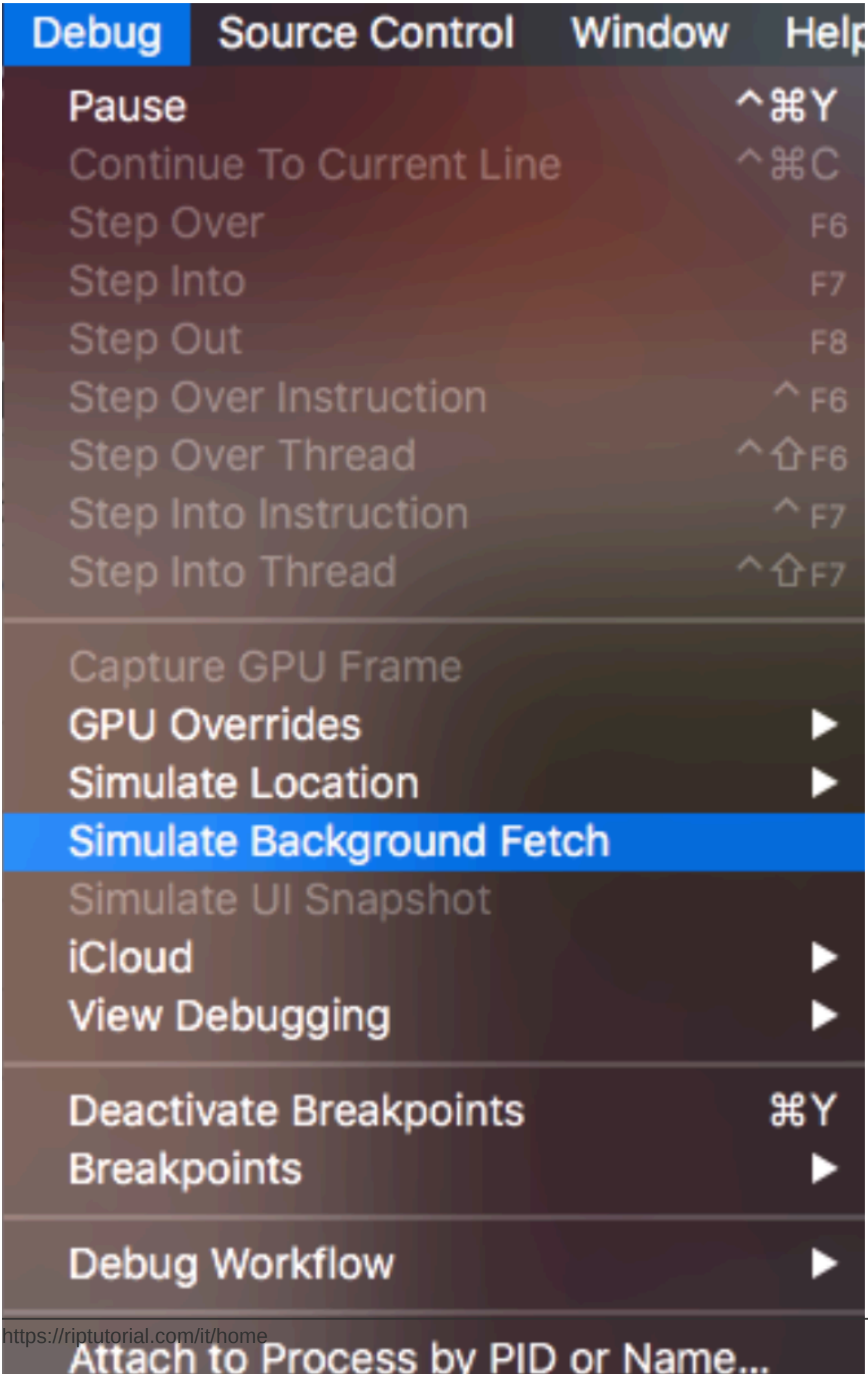
veloce

```
func application(_ application: UIApplication, performFetchWithCompletionHandler
completionHandler: @escaping (UIBackgroundFetchResult) -> Void) {
    // your code here
}
```

Test del recupero dello sfondo

1- Avvia l'app su un dispositivo reale e collegalo al debugger Xcode.

2- Dal menu Debug, seleziona **Simula recupero sfondi** :


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...

https://riptutorial.com/it/ios/topic/9178/modalita-di-background

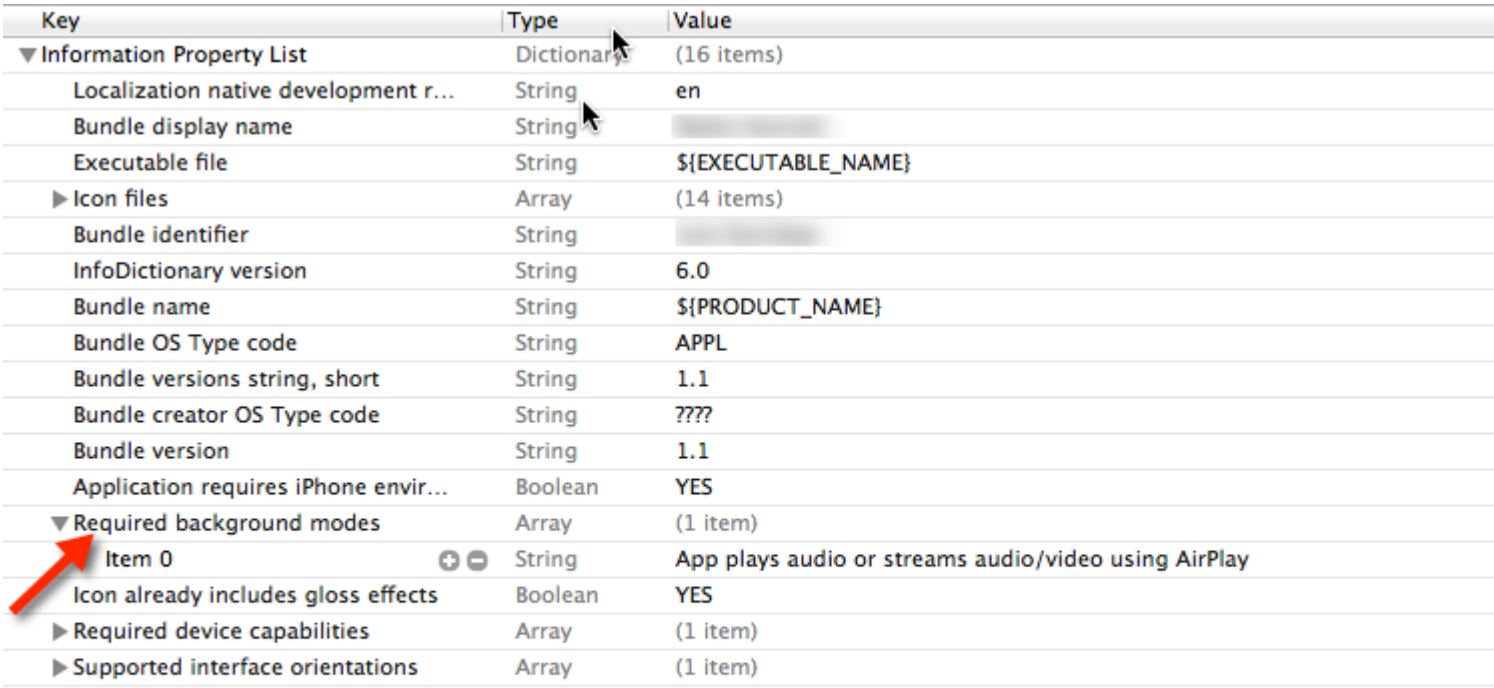
Capitolo 97: Modalità e eventi di sfondo

Examples

Riproduci l'audio in background

Aggiungi una chiave chiamata **Modalità sfondo richieste** nel file lista di proprietà (.plist) ..

come immagine seguente ..



E aggiungi il seguente codice in

AppDelegate.h

```
#import <AVFoundation/AVFoundation.h>
#import <AudioToolbox/AudioToolbox.h>
```

AppDelegate.m

in applicazione ha fattoFinishLaunchingWithOptions

```
[[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);
```

Se si desidera apportare modifiche in base agli eventi, è necessario aggiungere il seguente codice in 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;
        }
    }
}
```

Sulla base della notifica dobbiamo lavorarci sopra ..

Leggi Modalità e eventi di sfondo online: https://riptutorial.com/it/ios/topic/3515/modalita-e-eventi-di-sfondo

Capitolo 98: ModelPresentationStyles

introduzione

Gli stili di presentazione modali vengono utilizzati durante la transizione da un controller di visualizzazione a un altro. Ci sono 2 modi per raggiungere questa personalizzazione. Uno è attraverso il codice e un altro tramite Interface Builder (usando segues). Questo effetto si ottiene impostando la variabile `modalPresentationStyle` su un'istanza di enum `UIModalPresentationStyle` . `modalPresentationStyle` proprietà `modalPresentationStyle` è una variabile di classe di `UIViewController` e viene utilizzata per specificare come viene visualizzato `ViewController` sullo schermo.

Osservazioni

Ricorda sempre la seguente menzione di Apple.

> In un ambiente orizzontale compatto, i controller di visualizzazione modali vengono sempre visualizzati a schermo intero. In un ambiente orizzontale normale, ci sono diverse opzioni di presentazione.

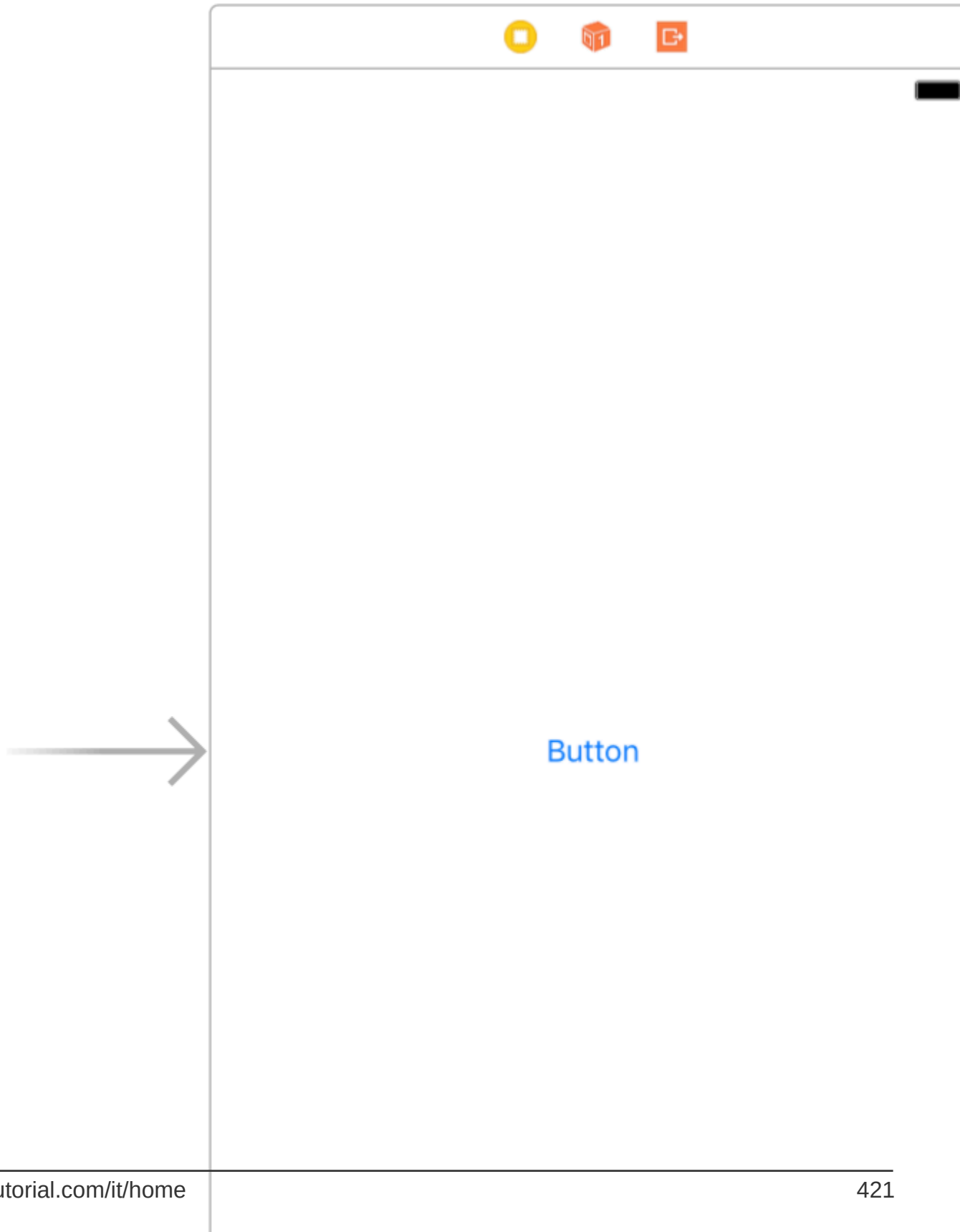
Examples

Esplorazione di ModalPresentationStyle utilizzando Interface Builder

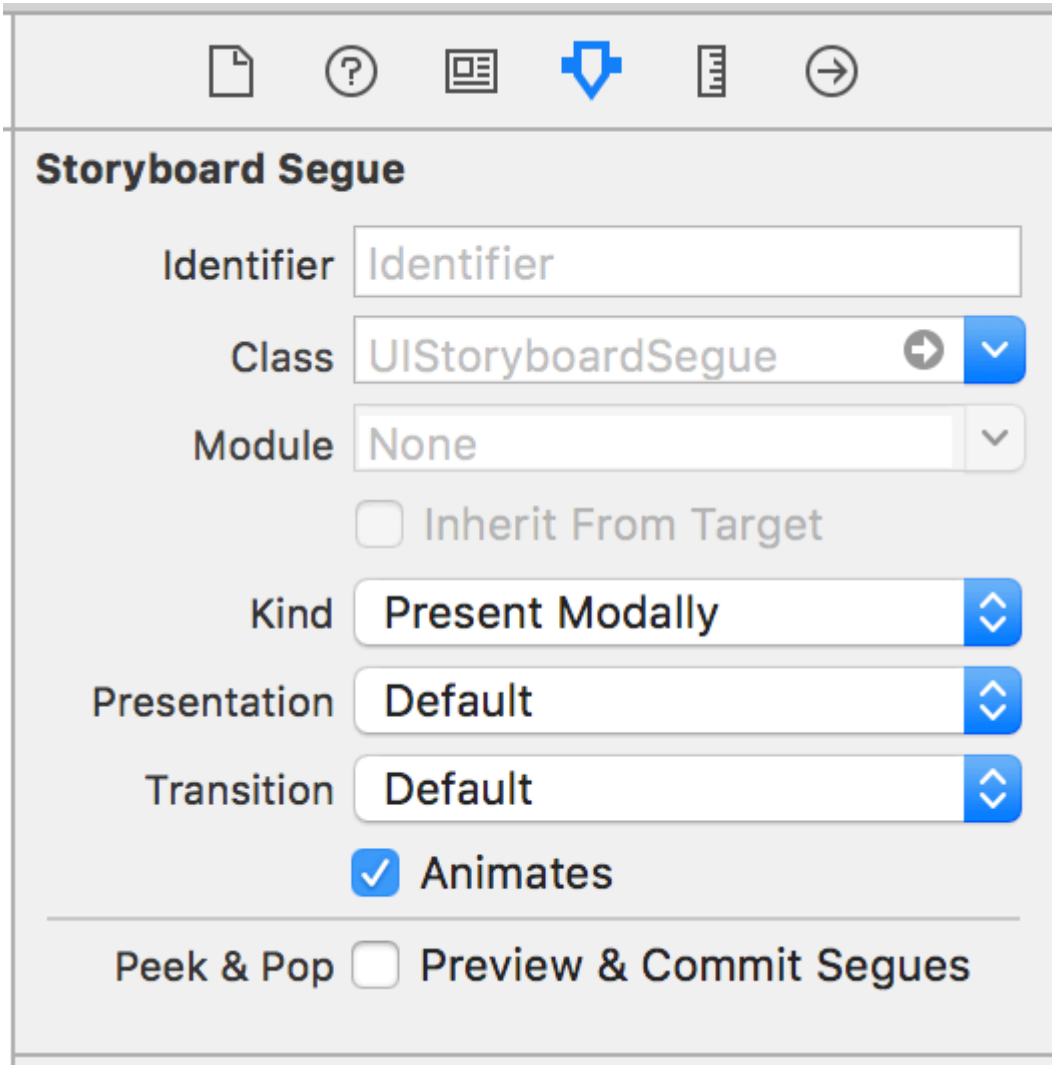
Questa sarà un'app molto semplice che illustrerà diversi `ModalpresentationStyle` in iOS. Secondo la documentazione trovata qui , ci sono 9 valori diversi per `UIModalPresentationStyle` che sono i seguenti,

1. `fullScreen`
2. `pageSheet`
3. `formSheet`
4. `currentContext`
5. `custom`
6. `overFullScreen`
7. `overCurrentContext`
8. `popover`
9. `none`

Per configurare un progetto, basta creare un normale progetto iOS e aggiungere 2 `ViewControllers` . Mettere un `UIButton` in te iniziale `ViewController` e collegarlo al 2 `ViewController` tramite un `Target -> Action` meccanismo. Per distinguere entrambi `ViewControllers` , impostare la proprietà di sfondo `UIView` in `ViewController` qualche altro colore. Se tutto va bene, il tuo Interface Builder dovrebbe sembrare qualcosa,


Button

(per i dettagli sul perché iPad, consulta la sezione Note). Una volta che hai finito di configurare il tuo progetto, seleziona il seguito e vai alla finestra di `attributes inspector` degli `attributes inspector` . Dovresti essere in grado di vedere qualcosa di simile,



Imposta la proprietà kind su `Present Modally` .

Ora, non vedremo tutti gli effetti in questo esempio in quanto alcuni di essi richiedono un po 'di codice.

Iniziamo con lo `fullscreen` . Questo effetto è selezionato per impostazione predefinita quando si seleziona `Present Modally` in `Kind` . Quando costruisci ed esegui, il 2 ° `ViewController` occuperà l'intero schermo del tuo iPad.


iPad Air – iOS 10.3 (14E269)
Carrier
6:11 PM

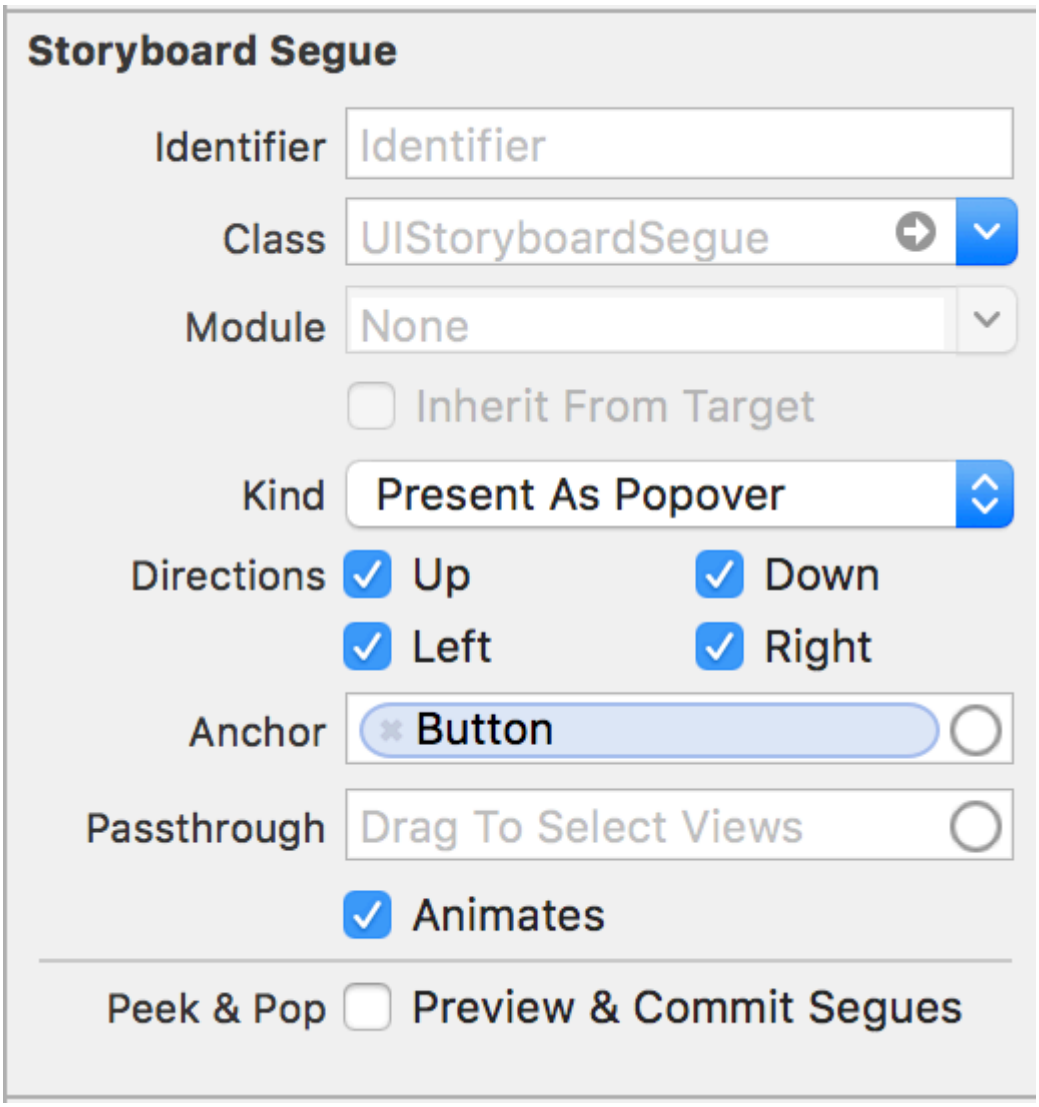
. In questa opzione, quando il dispositivo è in modalità `ViewController` , il 2 ° `ViewController` è simile allo schermo intero, ma in modalità orizzontale, 2 ° `ViewController` è molto stretto la larghezza del dispositivo. Inoltre, qualsiasi contenuto non coperto da 2nd `ViewController` sarà oscurato.


iPad Air – iO
Carrier
6

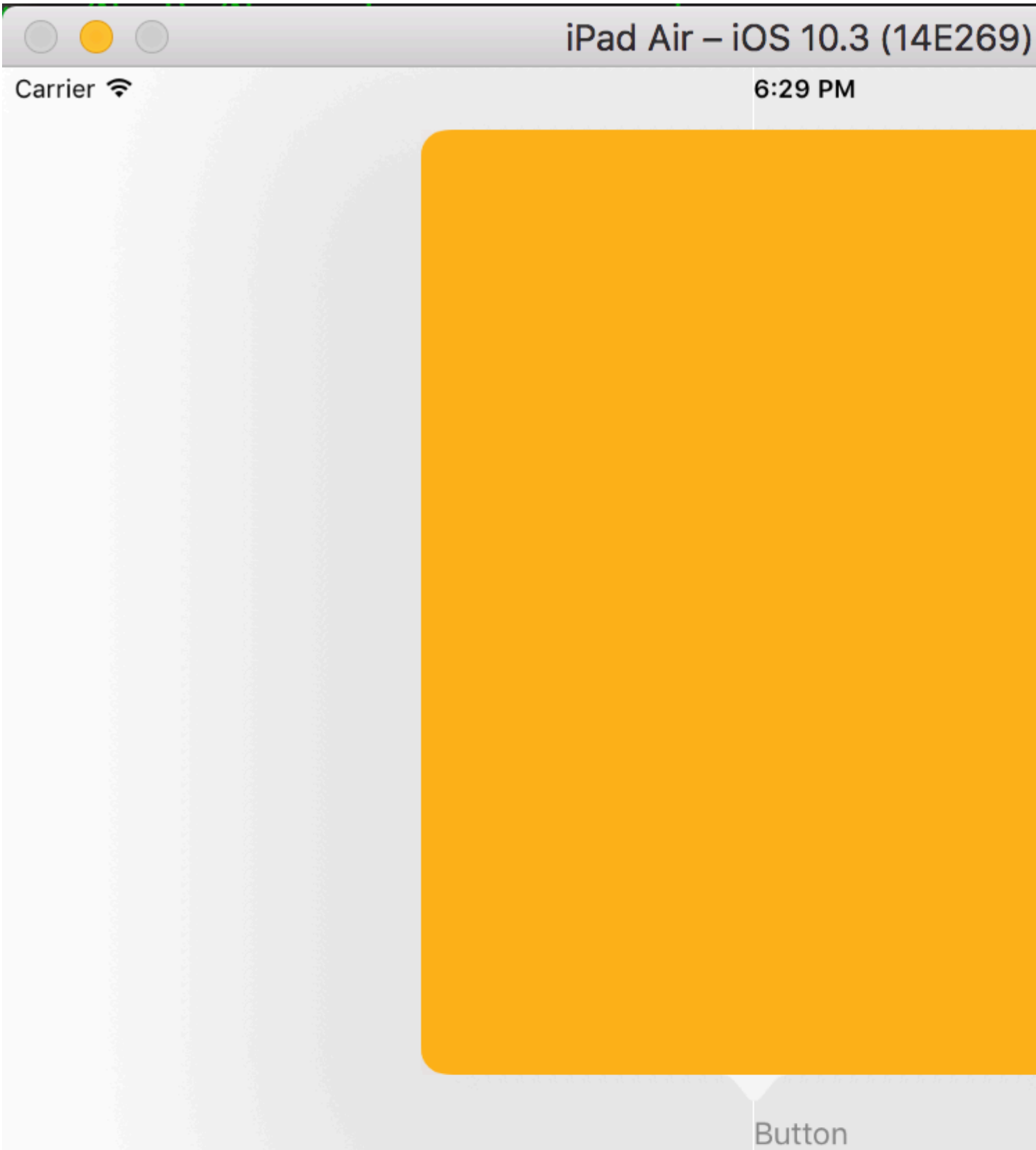
è collocato al centro del dispositivo e la dimensione è inferiore a quella del dispositivo. Anche quando il dispositivo è in modalità orizzontale e la tastiera è visibile, la posizione della vista viene regolata verso l'alto per mostrare `ViewController` .


iPad Air – iOS 10.3 (14E269)
Carrier
6:23 PM

`Present as Popover` nella scheda `Kind` . Il 2 ° `ViewController` è presentato come un piccolo popover (la dimensione può essere impostata). Il contenuto dello sfondo è oscurato. Ogni tocco al di fuori del popover eliminerebbe il popover. L' `Attributes Inspector` dovrebbe assomigliare a questo,



`Anchor` è l'elemento dell'interfaccia utente a cui desideri puntare la freccia del popover. `Directions` sono le direzioni che consentono al punto di `Anchor` popover di indicare.


iPad Air – iOS 10.3 (14E269)
Carrier
6:29 PM
Button

https://riptutorial.com/it/ios/topic/10122/modelpresentationstyles

Capitolo 99: MPMediaPickerDelegate

Osservazioni

Si prega di consultare la documentazione Apple per ulteriori informazioni sulla privacy.

Assicurati che l'app Music sia disponibile sul tuo iPhone. Non funzionerà nel simulatore.

Examples

Carica musica con MPMediaPickerControllerDelegate e riproducilo con AVAudioPlayer

Segui i passaggi:

- Aggiungi "NSAppleMusicUsageDescription" al tuo Info.plist per l'autorità sulla privacy.
- Assicurati che la tua musica sia disponibile sul tuo iPhone. Non funzionerà nel simulatore.

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)
    }

}
```

Leggi MPMediaPickerDelegate online:
https://riptutorial.com/it/ios/topic/7299/mpmediapickerdelegate

Capitolo 100: MPVolumeView

introduzione

La classe MPVolumeView è la vista volume per presentare all'utente un controllo a scorrimento per l'impostazione del volume di uscita audio del sistema e un pulsante per la scelta del percorso di uscita audio.

Osservazioni

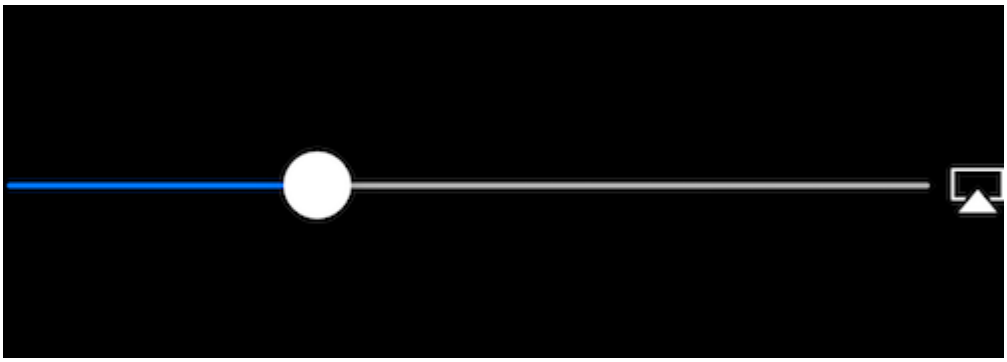
MPVolumeView si presenta solo quando si costruisce e funziona su un dispositivo iOS reale e non funziona in un simulatore.

Examples

Aggiunta di un 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
```

!!! Una nota molto importante è che MPVolumeView funziona solo su un dispositivo reale e non su un simulatore.



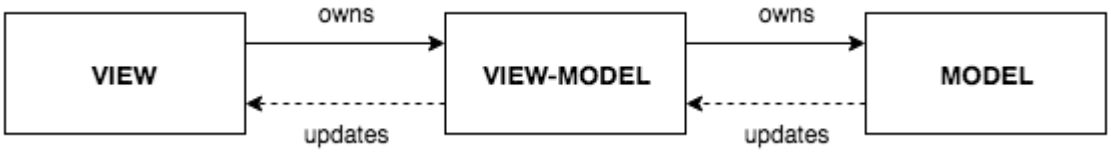
Leggi MPVolumeView online: https://riptutorial.com/it/ios/topic/9038/mpvolumeview

Capitolo 101: MVVM

Examples

MVVM senza programmazione reattiva

Inizierò con una spiegazione molto breve di cosa è e perché utilizzare il modello di progettazione Model-View-ViewModel (MVVM) nelle tue app iOS. Quando è comparso iOS per la prima volta, Apple ha suggerito di utilizzare MVC (Model-View-Controller) come modello di progettazione. Lo hanno mostrato in tutti i loro esempi e tutti i primi sviluppatori sono stati felici di utilizzarlo perché ha separato bene le preoccupazioni tra la business logic e l'interfaccia utente. Man mano che le applicazioni diventavano più grandi e complesse, un nuovo problema appariva appropriatamente chiamato Massive View Controller (MVC). Poiché tutta la logica aziendale è stata aggiunta nel ViewController, con il tempo sono diventati di solito troppo grandi e complessi. Per evitare il problema di MVC, è stato introdotto un nuovo modello di progettazione per il mondo di iOS: modello Model-View-ViewModel (MVVM).



Lo schema sopra mostra come appare MVVM. Hai un ViewController + View standard (nello storyboard, XIB o Code), che funge da Vista MVVM (nel testo successivo - View farà riferimento a MVVM's View). Una vista ha un riferimento a un ViewModel, dove è la nostra logica di business. È importante notare che ViewModel non sa nulla della vista e non ha mai un riferimento alla vista. ViewModel ha un riferimento a un modello.
Questo è sufficiente con una parte teorica di MVVM. Maggiori informazioni possono essere lette qui .

Uno dei **problemi principali di MVVM** è come aggiornare View tramite ViewModel quando ViewModel non ha riferimenti e non sa nulla della Vista.

La parte principale di questo esempio è mostrare come utilizzare MVVM (più precisamente, come associare ViewModel e View) senza alcuna programmazione reattiva (ReactiveCocoa, ReactiveSwift o RxSwif). Come nota: se si desidera utilizzare la programmazione reattiva, ancora meglio visto che i collegamenti MVVM sono fatti davvero facilmente. Ma questo esempio è su come usare MVVM senza programmazione Reattiva.

Creiamo un semplice esempio per dimostrare come utilizzare MVVM.

Il nostro `MVVMExampleViewController` è un semplice ViewController con un'etichetta e un pulsante. Quando viene premuto il pulsante, il testo dell'etichetta deve essere impostato su "Ciao". Dal momento che decidere cosa fare nell'interazione utente dell'utente fa parte della logica aziendale, ViewModel dovrà decidere cosa fare quando l'utente preme il pulsante. MVVM's View non dovrebbe fare alcuna logica di business.

```
class MVVMExampleViewController: UIViewController {

    @IBOutlet weak var helloLabel: UILabel!

    var viewModel: MVVMExampleViewModel?

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

    @IBAction func sayHelloButtonPressed(_ sender: UIButton) {
        viewModel?.userTriggeredSayHelloButton()
    }
}
```

`MVVMExampleViewModel` è un semplice ViewModel.

```
class MVVMExampleViewModel {

    func userTriggeredSayHelloButton() {
        // How to update View's label when there is no reference to the View??
    }
}
```

Potresti chiederti come impostare il riferimento ViewModel nella vista. Solitamente lo faccio quando ViewController viene inizializzato o prima che venga mostrato. Per questo semplice esempio, farei qualcosa di simile in `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
```

La vera domanda ora è: come aggiornare View from ViewModel senza dare un riferimento alla View to the ViewModel? (Ricorda, non useremo nessuna delle librerie iOS di Programmazione Reattiva)

Potresti pensare di usare KVO, ma questo complicherebbe le cose troppo. Alcune persone intelligenti hanno pensato al problema e hanno inventato la biblioteca di James Bond . La libreria potrebbe sembrare complicata e un po 'più difficile da capire all'inizio, quindi ne prenderò solo una piccola parte e renderò il nostro MVVM pienamente funzionante.

Introduciamo la classe `Dynamic` , che è il nucleo del nostro modello MVVM semplice ma pienamente funzionale.

```
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` classe `Dynamic` utilizza Generics and Closures per associare il nostro ViewModel alla nostra vista. Non entrerò nei dettagli di questa classe, possiamo farlo nei commenti (per rendere questo esempio più breve). `MVVMExampleViewController` ora il nostro `MVVMExampleViewController` e `MVVMExampleViewModel` per utilizzare tali classi.

Il nostro `MVVMExampleViewController` aggiornato

```
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()
    }
}
```

Aggiornato `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"
    }
}
```

È così. `ViewModel` è ora in grado di aggiornare `View` senza che abbia un riferimento alla `View` .

Questo è un esempio molto semplice, ma penso che tu abbia un'idea di quanto possa essere potente. Non entrerò nei dettagli sui vantaggi di MVVM, ma una volta passati da MVC a MVVM, non tornerai indietro. Provalo e vedi di persona.

Leggi MVVM online: https://riptutorial.com/it/ios/topic/8775/mvvm

Capitolo 102: MyLayout

introduzione

MyLayout è un framework oggettivo-c semplice e facile per il layout della vista iOS. MyLayout fornisce alcune semplici funzioni per creare una varietà di interfacce complesse. Integra le funzioni, tra cui: Autolayout e SizeClass di iOS, cinque classi di layout di Android, float e flex-box e bootstrap di HTML / CSS. puoi visitare da:

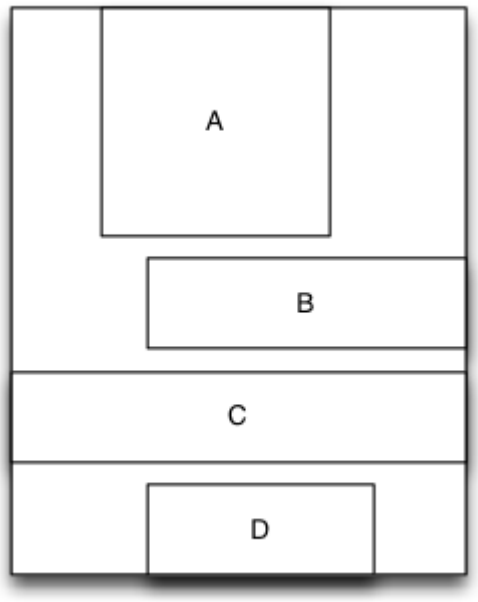
Obiettivo C: https://github.com/youngsoft/MyLinearLayout Swift: https://github.com/youngsoft/TangramKit

Examples

Una semplice demo per usare MyLayout

1. C'è una vista del contenitore S che ha una larghezza di 100 e l'altezza è a capo di tutta l'altezza delle sottoview. ci sono quattro sottoview A, B, C, D disposti dall'alto verso il basso.
2. Il margine sinistro di Subview A è del 20% in larghezza di S, il margine destro è del 30% in larghezza di S, l'altezza è uguale alla larghezza di A.
3. Il margine sinistro di Subview B è 40, la larghezza è riempita per la larghezza residua di S, l'altezza è 40. La sottospecifica della larghezza di C è riempita in S, l'altezza è 40.
4. Il margine destro di Subview D è 20, la larghezza è 50% larghezza di S, l'altezza è 40

come sotto figura:




```
    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];
```

Leggi MyLayout online: https://riptutorial.com/it/ios/topic/9692/mylayout

Capitolo 103: Notifiche Rich

introduzione

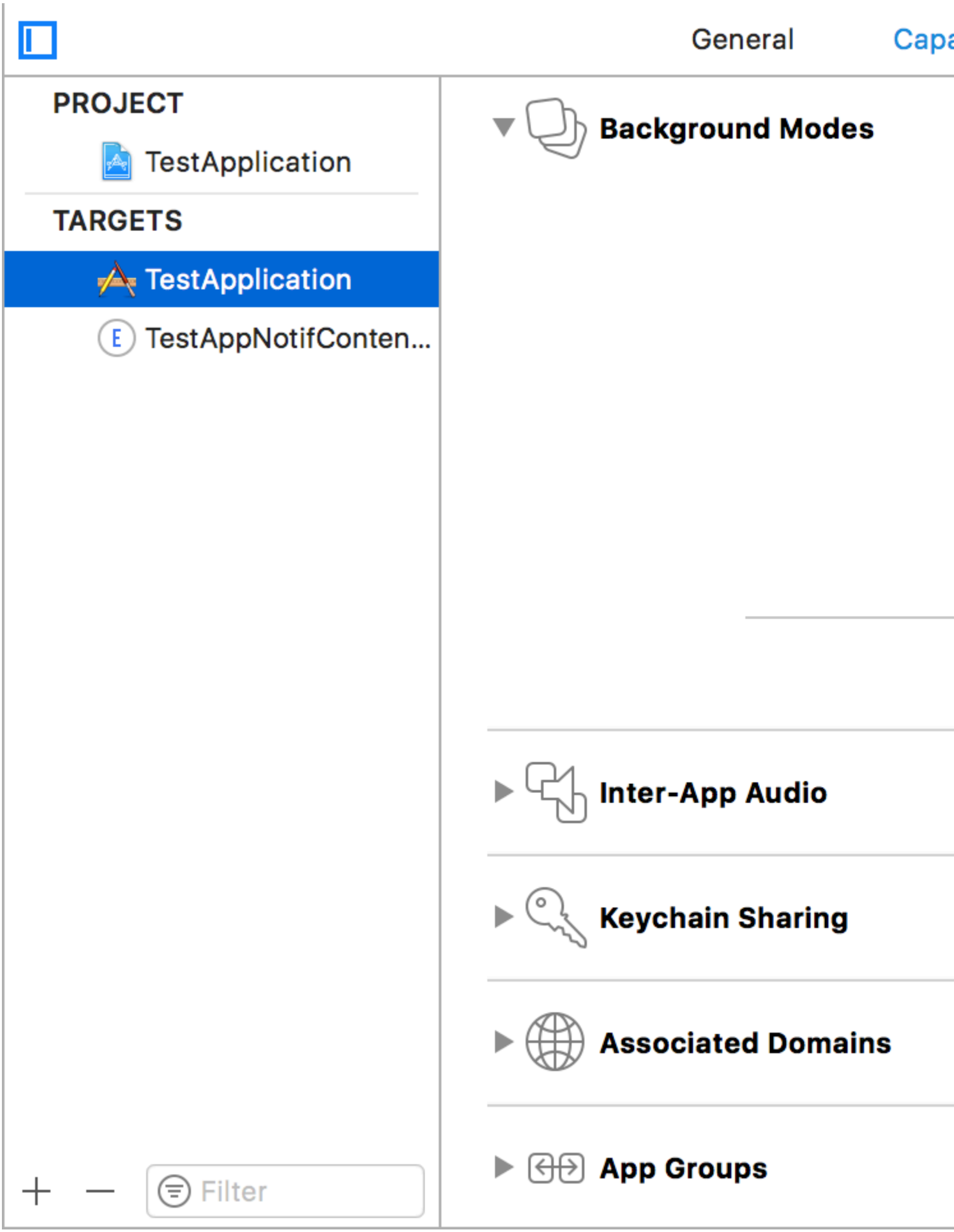
Le notifiche avanzate ti consentono di personalizzare l'aspetto delle notifiche locali e remote quando vengono visualizzate sul dispositivo dell'utente. La notifica di notifica include principalmente UNNotificationServiceExtension e UNNotificationContentExtension, ovvero la visualizzazione della notifica normale in modo esteso

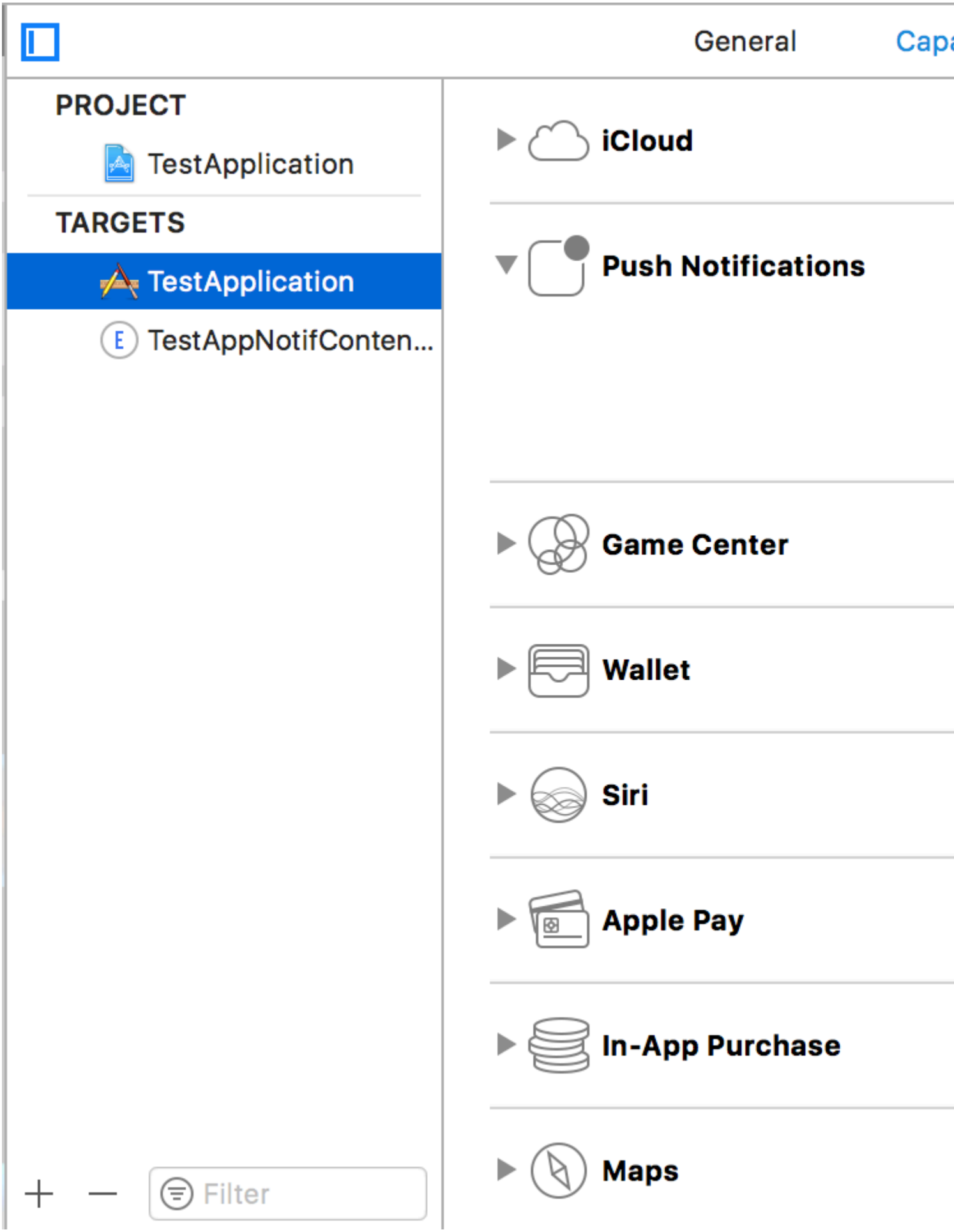
Examples

Creazione di un semplice UNNotificationContentExtension

Passo 1

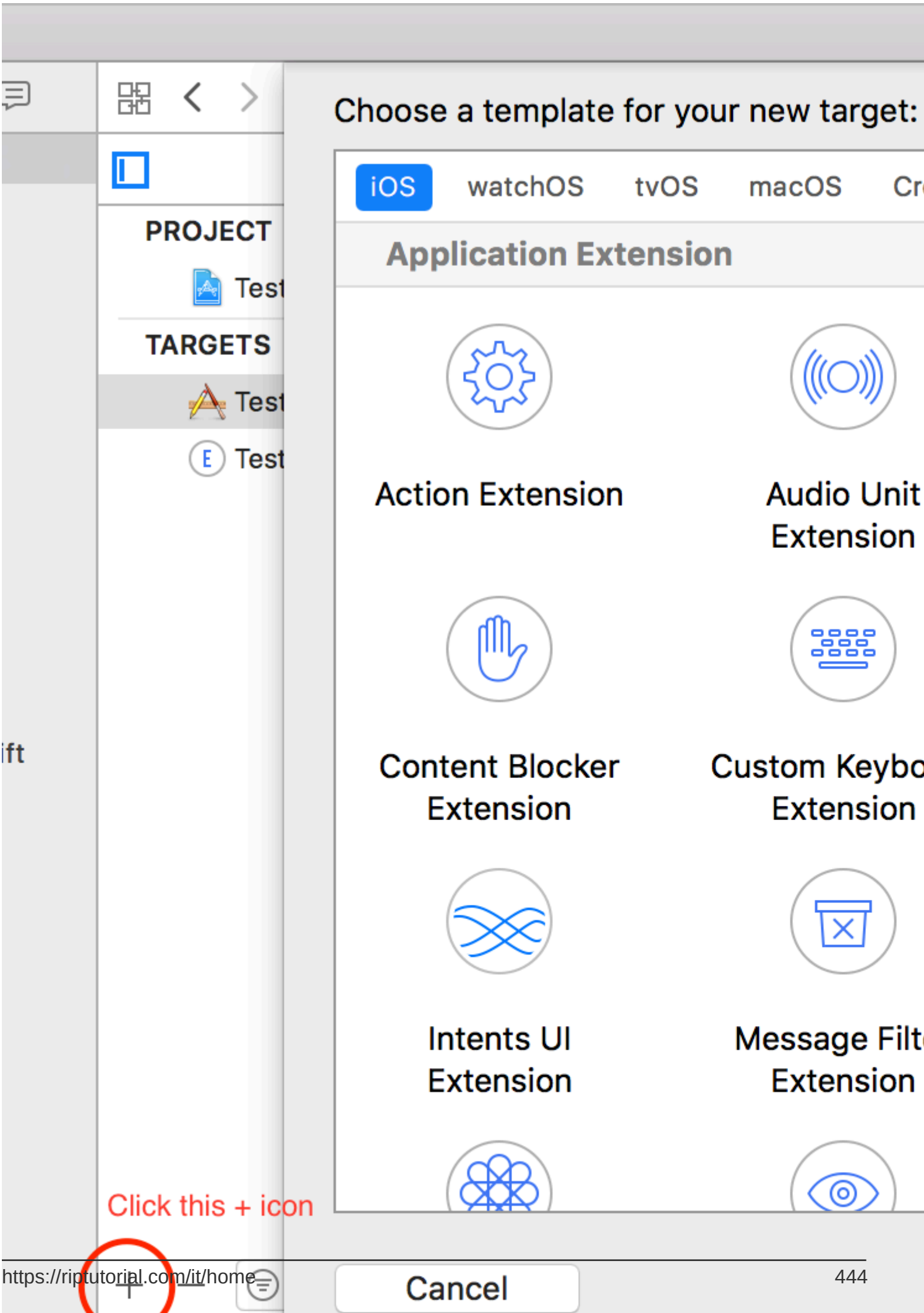
Rendere l'ambiente adatto alla notifica. Assicurati di aver abilitato le **Modalità di background** e **Push Notification**


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

Passaggio 2: creazione di UNNotificationContentExtension

Fai clic sull'icona **+** in basso che crea un modello di destinazione e seleziona Estensione contenuto di notifica -> successivo -> crea un nome per l'estensione di contenuto -> termina


Choose a template for your new target:
iOS
watchOS
tvOS
macOS
Application Extension
Action Extension
Audio Unit Extension
Content Blocker Extension
Custom Keyboard Extension
Intents UI Extension
Message Filter Extension
PROJECT
Test
TARGETS
Test
Test
Click this + icon
Cancel

- UNNotificationDefaultContentHidden: questa booleana determina se il corpo predefinito della notifica deve essere nascosto o meno
- UNNotificationCategory: la categoria viene creata in `UNUserNotificationCenter` nell'applicazione. Qui può essere una stringa o una serie di stringhe, quindi ogni categoria può dare diversi tipi di dati dai quali possiamo creare diverse UI. Il payload che inviamo deve contenere il nome della categoria per poter visualizzare questa particolare estensione nell'applicazione. Qui può essere una stringa o una serie di stringhe, quindi ogni categoria può dare diversi tipi di dati dai quali possiamo creare diverse UI. Il payload che inviamo deve contenere il nome della categoria per poter visualizzare questa particolare estensione
- UNNotificationExtensionInitialContentSizeRatio: la dimensione del contenuto iniziale, ovvero quando si visualizza ContentExtension per la prima volta la dimensione iniziale rispetto alla larghezza del dispositivo. qui 1 indica che l'altezza sarà uguale alla larghezza

Passaggio 4: Creazione di `UNNotificationAction` e `UNNotificationCategory` nella nostra applicazione

Nell'app AppDelegate.swift della tua app è `didFinishLaunchingWithOptions` aggiunta la funzioneFinishLaunchingWithOptions

```
    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()
```

Abbiamo creato due `UNNotificationAction` con identificatori `ID1` e `ID2` e abbiamo aggiunto tali azioni a `UNNotificationCategory` con identificativo `CATID1` (lo `CATID1` categoria nel file info.plist di ContentExtension è lo stesso, quello che abbiamo creato qui dovrebbe essere utilizzato nel payload e nel file plist). `UNUserNotificationCenter` la categoria su `UNUserNotificationCenter` della nostra applicazione e nella riga successiva ci stiamo registrando per la notifica che chiama la funzione `didRegisterForRemoteNotificationsWithDeviceToken` dove otteniamo il token del dispositivo

Nota: non dimenticare di `import UserNotifications` in AppDelegate.swift e aggiungere `UNUserNotificationCenterDelegate`

Passaggio 5: carico utile di esempio per NotificationContent

```
'aps': {
   'badge': 0,
   'alert': {
       'title': "Rich Notification",
       'body': "Body of RICH NOTIFICATION",
       },
   'sound' : "default",
   'category': "CATID1",
```

```
    'mutable-content':"1",
    },
'attachment': "2"
```

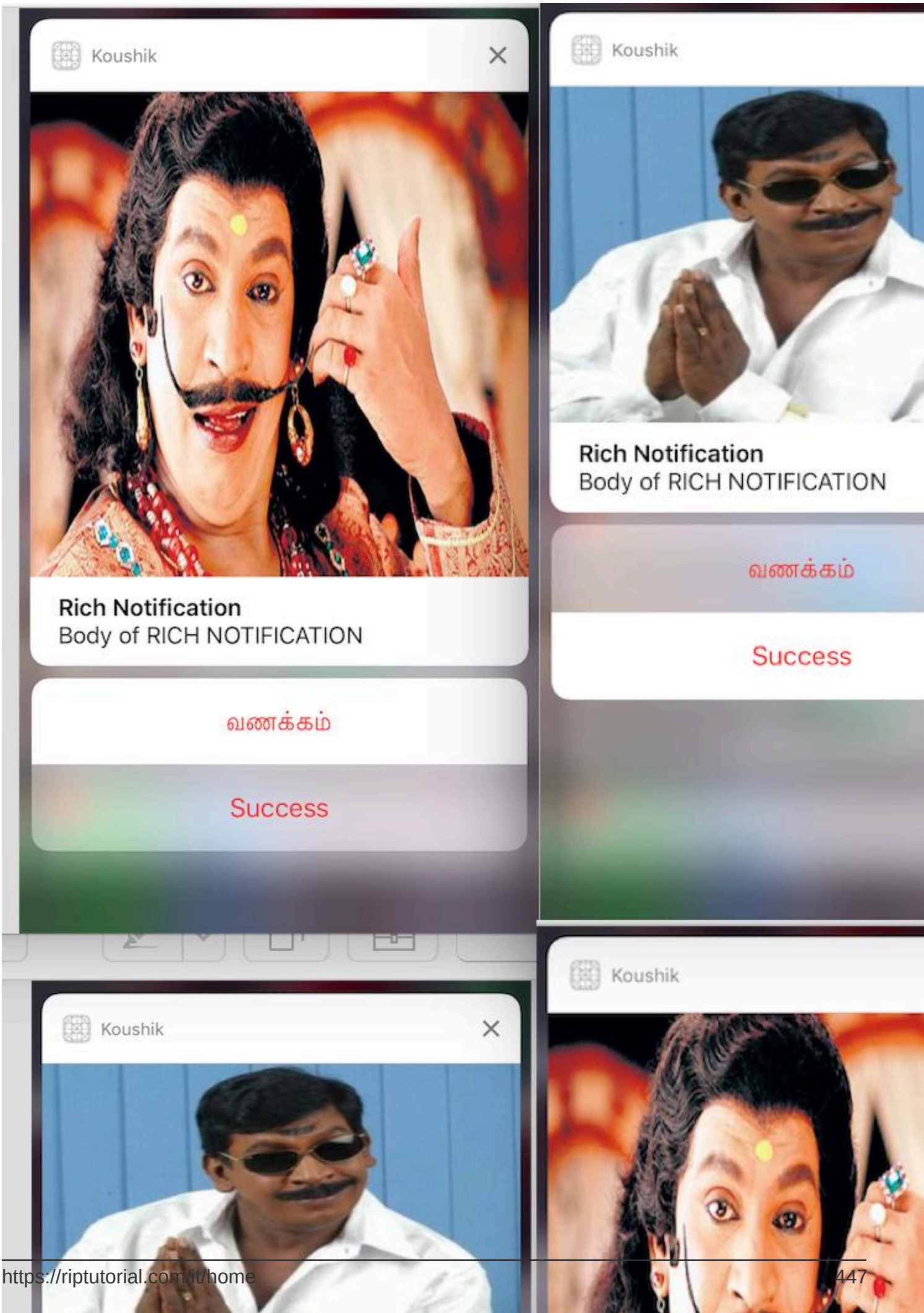
Passaggio 6: configurazione di ContentExtension

Le azioni corrispondenti per la categoria vengono visualizzate automaticamente mentre viene eseguita l'azione di notifica. Consente di vedere il codice come viene eseguito

```
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")
    }

    }
}
```

Passaggio 7: risultato

Dopo aver ricevuto e premuto a lungo / facendo clic su Visualizza notifica, la notifica appare come questa


Koushik
Rich Notification
Body of RICH NOTIFICATION
வணக்கம்
Success
Koushik
Rich Notification
Body of RICH NOTIFICATION
வணக்கம்
Success
Koushik
Koushik

`UNNotificationExtensionOverrideDefaultTitle` come YES. Nel passaggio 3 abbiamo dato `UNNotificationExtensionDefaultContentHidden` come NO se è SÌ, quindi la notifica avrà l'aspetto delle immagini 3 e 4.

Leggi Notifiche Rich online: https://riptutorial.com/it/ios/topic/10769/notifiche-rich

Capitolo 104: NSArray

introduzione

Ecco alcune utili funzioni / metodi di utilità che possono essere utilizzati come estensione Array per consentire allo sviluppatore di eseguire determinate operazioni critiche su array con l'aiuto del codice a riga singola.

Osservazioni

Una volta che il documento corrente viene approvato, aggiungerà anche molti miglioramenti per gli altri programmi di array. Questo è il mio primo documento e ho bisogno della vostra assistenza e approvazione nei miei sforzi.

Examples

Converti matrice in stringa json

Chiama questa funzione con parametro parameter come array con tipo 'any'. Ti restituirà la stringa json. La stringa JSON viene utilizzata per inviare l'array nella chiamata al servizio web come parametro di input della richiesta in 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
    }
```

Leggi NSArray online: https://riptutorial.com/it/ios/topic/9248/nsarray

Capitolo 105: NSAttributedString

Osservazioni

Imposta il colore del carattere usando NSAttributoStringa

Examples

Creazione di una stringa con crenatura personalizzata (spaziatura tra lettere)

`NSAttributedString` (e il suo fratello mutabile `NSMutableAttributedString`) consente di creare stringhe complesse nel loro aspetto all'utente.

Un'applicazione comune è quella di usarlo per visualizzare una stringa e aggiungere crenatura / interlinea personalizzata.

Ciò si otterrebbe come segue (dove label è un `UILabel`), dando una diversa crenatura per la parola "kerning"

veloce

```
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];
```

Crea una stringa con testo barrato

Objective-C

```
NSMutableAttributedString *attributeString = [[NSMutableAttributedString alloc]
initWithString:@"Your String here"];
[attributeString addAttribute:NSStrikethroughStyleAttributeName
                    value:@2
                    range:NSMakeRange(0, [attributeString length])];
```

veloce

```
let attributeString: NSMutableAttributedString =  NSMutableAttributedString(string: "Your
String here")
attributeString.addAttribute(NSStrikethroughStyleAttributeName, value: 2, range:
```

```
NSMakeRange(0, attributeString.length))
```

Quindi puoi aggiungerlo alla tua UILabel:

```
yourLabel.attributedText = attributeString;
```

Aggiunta di archi attribuiti e testo in grassetto in 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())]))
```

Il risultato è simile a:

Il valore è: **Qualcosa che l'utente ha inserito**

Cambia il colore di una parola o una stringa

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;
```

veloce

```
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
```

Nota :

Il principale è utilizzare `NSMutableAttributedString` e il selettore `addAttribute:value:range` con l'attributo `NSForegroundColorAttributeName` per modificare un colore di un intervallo di stringhe:

```
NSMutableAttributedString *attrsString =  [[NSMutableAttributedString alloc]
initWithAttributedString:label.attributedText];
[attrsString addAttribute:NSForegroundColorAttributeName value:color range:range];
```

È possibile utilizzare un altro modo per ottenere l'intervallo, ad esempio: NSRegularExpression.

Rimozione di tutti gli attributi

Objective-C

```
NSMutableAttributedString *mutAttString = @"string goes here";
NSRange range = NSMakeRange(0, mutAttString.length);
[mutAttString setAttributes:@{} range:originalRange];
```

Come per la documentazione Apple usiamo, `setAttributes` e non `addAttribute` .

veloce

```
mutAttString.setAttributes([:], range: NSRange(0..<string.length))
```

Leggi NSAttributedString online: https://riptutorial.com/it/ios/topic/979/nsattributedstring

Capitolo 106: NSBundle

Examples

Ottenere il pacchetto principale

1. Ottenere un riferimento al pacchetto principale usando Cocoa.

   Per ottenere il fascio principale di applicazione Cocoa, chiamare il metodo della classe ***mainBundle*** della classe **NSBundle.**

   ```
   NSBundle *mainBundle;
   // Get the main bundle for the app;
   mainBundle = [NSBundle mainBundle];
   ```

2. Ottenere un riferimento al pacchetto principale utilizzando Core Foundation.

   Utilizzare la funzione ***CFBundleGetMainBundle*** per recuperare il bundle principale per l'applicazione basata su C.

   ```
   CFBundleRef mainBundle;
   // Get the main bundle for the app
   mainBundle = CFBundleGetMainBundle();
   ```

Ottenere Bundle by Path

1. Individuazione di un fascio di cacao tramite il suo percorso

   Per ottenere il pacchetto in un percorso specifico utilizzando Cocoa, chiamare il ***bundleWithPath:*** metodo di classe di **NSBundle**

   ```
   NSBundle *myBundle;
   // obtain a reference to a loadable bundle
   myBundle = [NSBundle bundleWithPath:@"/Library/MyBundle.bundle";
   ```

2. Individuazione di un pacchetto di Cocoa Foundation utilizzando il relativo percorso

   Per ottenere il pacchetto in un percorso specifico utilizzando Core Foundation, chiamare la funzione ***CFBundleCreate*** e utilizzare il tipo **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);
```

Leggi NSBundle online: https://riptutorial.com/it/ios/topic/5862/nsbundle

Capitolo 107: NSData

Osservazioni

Risorse utili

Documentazione Apple (NSData)

NSData.dataWithContentsOfFile ()

NSData.bytes

Examples

Creazione di oggetti NSData

Utilizzando un file

veloce

```
let data = NSData(contentsOfFile: filePath) //assuming filePath is a valid path
```

Objective-C

```
NSData *data = [NSData dataWithContentsOfFile:filePath]; //assuming filePath is a valid path
```

Utilizzando un oggetto String

veloce

```
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
```

Conversione di NSData in altri tipi

Accordare

veloce

```
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];
```

Array

veloce

```
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
```

Alla matrice di byte

veloce

```
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
```

Conversione da NSData a stringa HEX

NSData

può essere rappresentato come una stringa esadecimale, simile a ciò che emette nel suo metodo di `description` .

veloce

```
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
```

Leggi NSData online: https://riptutorial.com/it/ios/topic/5084/nsdata

Capitolo 108: NSDate

Sintassi

- NSDate () // NSDate oggetto init alla data e ora correnti
- NSDate (). TimeIntervalSince1970 // Data e ora correnti in numero di secondi da 00:00:00 UTC del 1 gennaio 1970.
- NSDate (). Compare (other: NSDate) // Restituisce un confronto tra la data corrente e un'altra data restituisce un `NSComparisonResult`

Osservazioni

Esistono diversi tipi di formato di data che è possibile impostare: qui è l'elenco completo di essi.

Formato	Significato / Descrizione	Esempio 1	Esempio 2
y	Un anno con almeno 1 cifra.	175 AD → "175"	2016 ANNUNCIO → "2016"
aa	Un anno con esattamente 2 cifre.	5 ANNUNCIO → "05"	ANNUNCIO 2016 → "16"
yyy	Un anno con almeno 3 cifre.	5 ANNUNCIO → "005"	2016 ANNUNCIO → "2016"
aaaa	Un anno con almeno 4 cifre.	5 AD → "0005"	2016 ANNUNCIO → "2016"
M	Un mese con almeno 1 cifra.	Luglio → "7"	"Novembre" → "11"
MM	Un mese con almeno 2 cifre.	Luglio → "07"	"Novembre" → "11"
MMM	Abbreviazione di tre lettere al mese.	Luglio → "Jul"	"Novembre" → "Nov"
MMMM	Nome completo del mese.	Luglio → "Luglio"	"Novembre" → "Novembre"
MMMMM	Abbreviazione di un mese in lettere (Jan, June, July all avranno 'J').	Luglio → "J"	"Novembre" → "N"
d	Giorno con almeno una cifra.	8 → "8"	29 → "29"

Formato	Significato / Descrizione	Esempio 1	Esempio 2
dd	Giorno con almeno due cifre.	8 → "08"	29 → "29"
"E", "EE" o "EEE"	Abbreviazione di 3 lettere al giorno del nome del giorno.	Lunedì → "Mon"	Giovedi → "Gio"
EEEE	Nome del giorno completo.	Lunedì → "lunedì"	Giovedi → "Giovedi"
EEEEE	Abbreviazione di 1 lettera del nome del giorno. (Gio e Mar sarà 'T')	Lunedì → "M"	Giovedì → "T"
EEEEEE	Abbreviazione di 2 lettere al giorno del nome del giorno.	Lunedì → "Mo"	Giovedi → "Th"
un	Periodo del giorno (AM / PM).	22:00 → "PM"	2 AM → "AM"
h	Un'ora basata su 1-12 con almeno 1 cifra.	10 PM → "10"	2 AM → "2"
hh	Un'ora basata su 1-12 con almeno 2 cifre.	10 PM → "10"	2 AM → "02"
H	Un'ora basata su 0-23 con almeno 1 cifra.	10 PM → "14"	2 AM → "2"
HH	Un'ora basata su 0-23 con almeno 2 cifre.	10 PM → "14"	2 AM → "02"
m	Un minuto con almeno 1 cifra.	7 → "7"	29 → "29"
mm	Un minuto con almeno 2 cifre.	7 → "07"	29 → "29"
S	Un secondo con almeno 1 cifra.	7 → "7"	29 → "29"
ss	Un secondo con almeno 2 cifre.	7 → "07"	29 → "29"

Ce ne sono molti altri, per ottenere tempi diversi in base alla zona (z), per ottenere tempo con dettagli al millisecondo (S), ecc.

Examples

Ottieni la data corrente

Ottenere la data corrente è molto semplice. Ottieni l'oggetto NSDate della data corrente in una sola riga come segue:

veloce

```
var date = NSDate()
```

Swift 3

```
var date = Date()
```

Objective-C

```
NSDate *date = [NSDate date];
```

Ottieni oggetto NSDate N secondi dalla data corrente

Il numero di secondi dalla data e ora correnti per la nuova data. Utilizzare un valore negativo per specificare una data prima della data corrente.

Per fare questo abbiamo un metodo denominato `dateWithTimerIntervalSinceNow(seconds: NSTimeInterval) -> NSDate` (Swift) o `+ (NSDate*)dateWithTimeIntervalSinceNow:(NSTimeInterval)seconds` (Objective-C).

Ora, ad esempio, se hai bisogno di una data una settimana dalla data corrente e una settimana alla data attuale, allora possiamo farlo come.

veloce

```
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);
```

Data di confronto

Esistono 4 metodi per confrontare le date:

veloce

- `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`

Diciamo che abbiamo 2 date:

veloce

```
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
```

Quindi, per confrontarli, proviamo questo codice:

veloce

```
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
}
```

Se desideri confrontare date e gestire secondi, settimane, mesi e anni:

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")
```

Oppure puoi usare questo per ogni componente:

```
// 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
```

Ottieni l'ora di Unix Epoch

Per ottenere Unix Epoch Time , utilizza la costante `timeIntervalSince1970` :

veloce

```
let date = NSDate() // current date
let unixtime = date.timeIntervalSince1970
```

Objective-C

```
NSDate *date = [NSDate date]; // current date
int unixtime = [date timeIntervalSince1970];
```

NSDateFormatter

La conversione di un oggetto `NSDate` in stringa è di soli 3 passaggi.

1. Creare un oggetto `NSDateFormatter`

veloce

```
let dateFormatter = NSDateFormatter()
```

Swift 3

```
let dateFormatter = DateFormatter()
```

Objective-C

```
NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
```

2. Impostare il formato della data in cui si desidera la stringa

veloce

```
dateFormatter.dateFormat = "yyyy-MM-dd 'at' HH:mm"
```

Objective-C

```
dateFormatter.dateFormat = @"yyyy-MM-dd 'at' HH:mm";
```

3. Ottieni la stringa formattata

veloce

```
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];
```

Questo darà qualcosa di simile a questo: `2001-01-02 at 13:00`

> **Nota**
>
> La creazione di un'istanza `NSDateFormatter` è un'operazione costosa, quindi è consigliabile crearla una volta e riutilizzarla quando possibile.

Estensione utile per convertire la data in stringa.

```
extension Date {
        func toString() -> String {
            let dateFormatter = DateFormatter()
            dateFormatter.dateFormat = "MMMM dd yyyy"
            return dateFormatter.string(from: self)
        }
}
```

Link utili per una data-formazione rapida che diventa rapidamente-leggibile-data-nsdateformatter .

Per la costruzione di formati di data, vedere i modelli di formato della data .

Converti NSDate composto da ora e minuto (solo) a un NSDate completo

Esistono molti casi in cui uno ha creato un NSDate da un formato di un'ora e minuti, ovvero: 08:12 che restituisce da un server come una stringa e si avvia un'istanza NSDate **solo** con questi **valori.**

Il lato negativo di questa situazione è che il tuo NSDate è quasi completamente "nudo" e quello che devi fare è creare: giorno, mese, anno, secondo e fuso orario in modo che questo oggetto "giochi" con altri tipi NSDate.

Per fare un esempio, diciamo che hourAndMinute è il tipo NSDate composto dal formato ora e minuto:

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];
```

Ora il tuo oggetto è l'esatto contrario di essere "nudo".

Offset ora UTC da NSDate con TimeZone

Qui questo calcolerà l'offset dell'ora `UTC` dai dati correnti nel fuso orario desiderato.

```
+(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;
}
```

Ottieni il tipo di ciclo temporale (12 ore o 24 ore)

Verifica se la data corrente contiene il simbolo per AM o 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);
```

Richiesta del tipo di ciclo temporale da `NSDateFormatter`

Objective-C

```
NSString *formatStringForHours = [NSDateFormatter dateFormatFromTemplate:@"j" options:0
locale:[NSLocale currentLocale]];
NSRange containsA = [formatStringForHours rangeOfString:@"a"];
```

```
BOOL is24h = containsA.location == NSNotFound;
```

Questo utilizza una stringa di modello data speciale chiamata "j" che secondo la specifica ICU ...

> [...] richiede il formato ora preferito per le impostazioni internazionali (h, H, K o k), come determinato dall'attributo preferito dell'elemento hours nei dati supplementari. [...] Si noti che l'uso di 'j' in uno scheletro passato a un'API è l'unico modo per richiedere a uno scheletro il tipo di ciclo temporale preferito di una locale (12 ore o 24 ore).

Quest'ultima frase è importante. È "l'unico modo per fare in modo che uno scheletro richieda il tipo di ciclo temporale preferito da una locale". Poiché `NSDateFormatter` e `NSCalendar` sono costruiti sulla libreria ICU, lo stesso vale qui.

Riferimento

La seconda opzione è stata derivata da questa risposta .

Ottieni NSDate dal formato data JSON "/ Data (1268123281843) /"

Prima delle date di Json.NET 4.5 sono stati scritti utilizzando il formato Microsoft: "/ Date (1198908717056) /". Se il server invia la data in questo formato, è possibile utilizzare il codice seguente per serializzarlo su 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;
}
```

Ottieni storico da NSDate (es: 5s fa, 2 mesi fa, 3 ore fa)

Può essere utilizzato in varie applicazioni di chat, feed RSS e app social in cui è necessario disporre degli ultimi feed con timestamp:

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;
}
```

Leggi NSDate online: https://riptutorial.com/it/ios/topic/1502/nsdate

Capitolo 109: NSHTTPCookieStorage

Examples

Archivia e leggi i cookie da 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")
    }
}

}
```

Leggi NSHTTPCookieStorage online: https://riptutorial.com/it/ios/topic/7312/nshttpcookiestorage

Capitolo 110: NSInvocation

Examples

NSInvocation Objective-C

Fai riferimento a questo post originale di e.James

Secondo il riferimento alla classe NSInvocation di Apple :

> Un `NSInvocation` è un messaggio Objective-C reso statico, ovvero un'azione trasformata in oggetto.

E, in un *piccolo* dettaglio:

Il concetto di messaggi è centrale nella filosofia dell'obiettivo-c. Ogni volta che chiami un metodo o accedi a una variabile di qualche oggetto, stai inviando un messaggio. `NSInvocation` è utile quando si desidera inviare un messaggio a un oggetto in un momento diverso o inviare lo stesso messaggio più volte. `NSInvocation` consente di *descrivere* il messaggio che si sta per inviare e quindi di *invocarlo* (in realtà lo si invia all'oggetto di destinazione) in un secondo momento.

Ad esempio, supponiamo di voler aggiungere una stringa a un array. Normalmente si invierà il messaggio `addObject:` come segue:

```
[myArray addObject:myString];
```

`NSInvocation` ora che tu voglia utilizzare `NSInvocation` per inviare questo messaggio in un altro momento:

Innanzitutto, si prepara un oggetto `NSInvocation` da utilizzare con il selettore `addObject:` `NSMutableArray` :

```
NSMethodSignature * mySignature = [NSMutableArray
    instanceMethodSignatureForSelector:@selector(addObject:)];
NSInvocation * myInvocation = [NSInvocation
    invocationWithMethodSignature:mySignature];
```

Successivamente, devi specificare a quale oggetto inviare il messaggio:

```
[myInvocation setTarget:myArray];
```

Specifica il messaggio che desideri inviare a quell'oggetto:

```
[myInvocation setSelector:@selector(addObject:)];
```

E compila qualsiasi argomento per quel metodo:

```
[myInvocation setArgument:&myString atIndex:2];
```

Si noti che gli argomenti dell'oggetto devono essere passati dal puntatore. Grazie a Ryan McCuaig per averlo indicato e per favore vedi la documentazione di Apple per maggiori dettagli.

A questo punto, `myInvocation` è un oggetto completo, che descrive un messaggio che può essere inviato. Per inviare effettivamente il messaggio, chiameresti:

```
[myInvocation invoke];
```

Questo passaggio finale causerà l'invio del messaggio, eseguendo essenzialmente `[myArray addObject:myString];` .

Pensa ad esso come mandare una email. Si apre una nuova email (oggetto `NSInvocation`), si `NSInvocation` l'indirizzo della persona (oggetto) a cui si desidera inviarlo, si digita un messaggio per il destinatario (specificare un `selector` e gli argomenti), quindi fare clic su "invia" (`invoke` chiamata).

Vedere Utilizzo di NSInvocation per ulteriori informazioni.

`NSUndoManager` utilizza oggetti `NSInvocation` modo che possa *invertire i* comandi. In sostanza, quello che stai facendo è creare un oggetto `NSInvocation` per dire: "Ehi, se vuoi annullare ciò che ho appena fatto, invia questo messaggio a quell'oggetto, con questi argomenti". `NSInvocation` oggetto `NSInvocation` a `NSUndoManager` e aggiunge quell'oggetto a una serie di azioni annullabili. Se l'utente chiama "Annulla", `NSUndoManager` cerca semplicemente l'azione più recente dell'array e richiama l'oggetto `NSInvocation` memorizzato per eseguire l'azione necessaria.

Vedi Registrazione delle operazioni di annullamento per maggiori dettagli.

Leggi NSInvocation online: https://riptutorial.com/it/ios/topic/8276/nsinvocation

Capitolo 111: NSNotificationCenter

introduzione

Le notifiche iOS sono un modo semplice e potente per inviare i dati in modo flessibile. Cioè, il mittente di una notifica non deve preoccuparsi di chi (se qualcuno) riceve la notifica, lo pubblica solo là fuori per il resto dell'app e potrebbe essere raccolto da un sacco di cose o nulla a seconda di lo stato della tua app.

Fonte : - HACKING con Swift

Parametri

Parametro	Dettagli
nome	Il nome della notifica per cui registrare l'osservatore; ovvero, solo le notifiche con questo nome vengono utilizzate per aggiungere il blocco alla coda delle operazioni. Se si passa a zero, il centro notifiche non utilizza il nome di una notifica per decidere se aggiungere il blocco alla coda operazioni.
obj	L'oggetto di cui l'osservatore desidera ricevere le notifiche; cioè, solo le notifiche inviate da questo mittente vengono consegnate all'osservatore. Se si passa a zero, il centro di notifica non utilizza il mittente di una notifica per decidere se consegnarlo all'osservatore.
coda	La coda di operazione a quale blocco deve essere aggiunto. Se si passa nil, il blocco viene eseguito in modo sincrono sul thread di registrazione.
bloccare	Il blocco da eseguire quando viene ricevuta la notifica. Il blocco viene copiato dal centro di notifica e (la copia) trattenuto fino alla rimozione della registrazione dell'osservatore.

Osservazioni

Un oggetto NSNotificationCenter (o semplicemente un centro di notifica) fornisce un meccanismo per la trasmissione di informazioni all'interno di un programma. Un oggetto NSNotificationCenter è essenzialmente una tabella di invio delle notifiche.

Per maggiori informazioni, consulta la documentazione Apple qui

NSNotification & NSNotificationCenter in Swift

Examples

Convenzione di denominazione

Le notifiche sono identificate da oggetti NSString globali i cui nomi sono composti in questo modo:

`Name of associated class` + `Did | Will` + `UniquePartOfName` + `Notification`

Per esempio:

- 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: Vale anche la pena notare che il numero di volte che un osservatore è stato aggiunto deve essere esattamente il numero di volte che l'osservatore viene rimosso. Un errore da principiante consiste nell'aggiungere l'osservatore nella `viewWillAppear:` di un UIViewController, ma rimuovendo l'osservatore in `viewDidUnload:` causerà un numero non uniforme di push e quindi la fuoriuscita dell'osservatore e del selettore di notifica che viene richiamato in modo superfluo.

Rimozione degli osservatori

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];
```

Pubblicazione di una notifica

veloce

```
NSNotificationCenter.defaultCenter().postNotificationName("TestNotification", object: self)
```

Objective-C

```
[[NSNotificationCenter defaultCenter] postNotificationName:@"TestNotification" object:nil];
```

Pubblicazione di una notifica con dati

veloce

```
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];
```

Osservando una notifica

veloce

```
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"];
}
```

Aggiunta / rimozione di un osservatore con un blocco

Invece di aggiungere un osservatore con un selettore, è possibile utilizzare un blocco:

```
id testObserver = [[NSNotificationCenter defaultCenter] addObserverForName:@"TestNotification"
                                                                    object:nil
                                                                     queue:nil
                                                                usingBlock:^(NSNotification*
notification) {
    NSDictionary *userInfo = notification.userInfo;
    MyObject *myObject = [userInfo objectForKey:@"someKey"];
}];
```

L'osservatore può quindi essere rimosso con:

```
[[NSNotificationCenter defaultCenter] removeObserver:testObserver
```

```
                                                  name:@"TestNotification"
                                                object:nil];
```

Aggiungi e rimuovi l'osservatore per nome

```
// 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)
```

Leggi NSNotificationCenter online: https://riptutorial.com/it/ios/topic/1601/nsnotificationcenter

Capitolo 112: NSPredicate

Sintassi

- Sottostazioni di stringa del formato predicato
  - Specifiers di stringa del formato C:% d,% s,% f, ecc
  - Sostituzione oggetto:% @
  - Sostituzione Keypath:% K
- Operatori di confronto predicati
  - =, ==: l'espressione della mano sinistra è uguale all'espressione della mano destra
  - > =, =>: L'espressione della mano sinistra è maggiore o uguale all'espressione della mano destra
  - <=, = <: L'espressione della mano sinistra è minore o uguale all'espressione della mano destra
  - >: L'espressione della mano sinistra è maggiore dell'espressione della mano destra
  - <: L'espressione della mano sinistra è inferiore all'espressione della mano destra
  - ! =, <>: L'espressione della mano sinistra non è uguale all'espressione della mano destra
  - TRA: L'espressione della mano sinistra è compresa tra o uguale a uno dei valori nell'espressione a destra, che specifica i limiti inferiore e superiore, ad esempio: BETWEEN {0, 5}
- Operatori composti predicati
  - AND, &&: AND logico
  - OR, ||: OR logico
  - NON,!: NOT logico
- Operatori di confronto delle stringhe predicati
  - INIZIA CON: L'espressione della mano sinistra inizia con l'espressione della mano destra
  - FINE: l'espressione della mano sinistra termina con l'espressione della mano destra
  - CONTIENE: l'espressione della mano sinistra contiene l'espressione della mano destra
  - LIKE: l'espressione della mano sinistra è uguale all'espressione della mano destra, con la sostituzione con caratteri jolly
    - *: Corrisponde a zero o più caratteri
    - ?: Corrisponde a un personaggio

Examples

Creazione di un NSPredicate usando predicateWithBlock

Objective-C

```
NSPredicate *predicate = [NSPredicate predicateWithBlock:^BOOL(id item,
                                                               NSDictionary *bindings) {
    return [item isKindOfClass:[UILabel class]];
```

```
}];
```

---

veloce

```
let predicate = NSPredicate { (item, bindings) -> Bool in
    return item.isKindOfClass(UILabel.self)
}
```

In questo esempio, il predicato corrisponderà agli elementi che appartengono alla classe `UILabel` .

Creazione di un NSPredicate usando predicateWithFormat

Objective-C

```
NSPredicate *predicate = [NSPredicate predicateWithFormat: @"self[SIZE] = %d", 5)];
```

---

veloce

```
let predicate = NSPredicate(format: "self[SIZE] >= %d", 5)
```

In questo esempio, il predicato corrisponderà a elementi che sono matrici con una lunghezza di almeno 5.

Creazione di un NSPredicate con variabili sostitutive

Un `NSPredicate` può utilizzare variabili di sostituzione per consentire il vincolo dei valori al volo.

Objective-C

```
NSPredicate *template = [NSPredicate predicateWithFormat: @"self BEGINSWITH $letter"];
NSDictionary *variables = @{ @"letter": @"r" };
NSPredicate *beginsWithR = [template predicateWithSubstitutionVariables: variables];
```

---

veloce

```
let template = NSPredicate(format: "self BEGINSWITH $letter")
let variables = ["letter": "r"]
let beginsWithR = template.predicateWithSubstitutionVariables(variables)
```

Il predicato template non viene modificato da `predicateWithSubstitutionVariables` . Invece, viene creata una copia e quella copia riceve le variabili di sostituzione.

Utilizzo di NSPredicate per filtrare una matrice

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"]
```

---

veloce

```
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"]
```

Convalida del modulo tramite 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 con condizioni `AND`,` OR` e `NOT`

Il predicato condizionale sarà più pulito e più sicuro utilizzando la classe `NSCompoundPredicate` che fornisce operatori booleani di base per i predicati specificati.

Objective-C

E - Condizione

```
  NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
  NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
  NSPredicate *combinedPredicate = [NSCompoundPredicate andPredicateWithSubpredicates:
@[predicate,anotherPredicate]];
```

O - Condizione

```
 NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
 NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
 NSPredicate *combinedPredicate = [NSCompoundPredicate orPredicateWithSubpredicates:
@[predicate,anotherPredicate]];
```

NOT - Condizione

```
 NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
 NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
 NSPredicate *combinedPredicate = [NSCompoundPredicate notPredicateWithSubpredicate:
@[predicate,anotherPredicate]];
```

Leggi NSPredicate online: https://riptutorial.com/it/ios/topic/5796/nspredicate

Capitolo 113: NSTimer

Parametri

Parametro	Dettagli
`interval`	Il tempo, in secondi, di aspettare prima di sparare il timer; o, nel ripetere i timer, il tempo tra un licenziamento e l'altro.
`target`	L'oggetto su cui chiamare il `selector`
`selector`	In Swift, un oggetto `Selector` che specifica il metodo da chiamare sulla `target`
`repeats`	Se `false` , attiva il timer una sola volta. Se è `true` , spara il timer ogni secondo d' `interval` .

Osservazioni

Un `NSTimer` ti consente di inviare un messaggio a un target dopo che è trascorso un determinato periodo di tempo.

Examples

Creazione di un timer

Questo creerà un timer per chiamare il metodo `doSomething` su `self` in 5 secondi.

veloce

```
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];
```

Impostando le ripetizioni su `false/NO` indica che vogliamo che il timer si attivi solo una volta. Se lo impostiamo su `true/YES` , verrebbe generato ogni cinque secondi fino a quando non viene invalidato manualmente.

Spegnere manualmente un timer

veloce

```
timer.fire()
```

Objective-C

```
[timer fire];
```

La chiamata al metodo di `fire` fa sì che un NSTimer esegua l'attività che normalmente avrebbe eseguito su una pianificazione.

In un **timer non ripetuto** , ciò invalida automaticamente il timer. Cioè, chiamare il `fire` prima che l'intervallo di tempo sia scaduto comporterà solo una chiamata.

In un **timer ripetuto** , questo semplicemente invocherà l'azione senza interrompere il solito programma.

Invalidare un timer

veloce

```
timer.invalidate()
```

Objective-C

```
[timer invalidate];
```

Questo fermerà il timer dal licenziamento. **Deve essere chiamato dal thread in cui è stato creato il timer,** vedere le note di Apple :

> È necessario inviare questo messaggio dal thread su cui è stato installato il timer. Se si invia questo messaggio da un altro thread, la sorgente di input associata al timer non può essere rimossa dal suo ciclo di esecuzione, il che potrebbe impedire l'uscita corretta del thread.

Note: Una volta che il timer è stato invalidato, è impossibile attivare lo stesso timer invalidato. Invece, è necessario inizializzare nuovamente il timer invalidato e attivare il metodo di attivazione.

Opzioni di frequenza del timer

Evento timer ripetuto

veloce

```
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()
        }
    }
```

Deve essere invalidato manualmente se lo si desidera.

veloce

Evento timer ritardato non ripetuto

```
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)
```

Il timer verrà attivato una volta, 3 secondi dopo l'esecuzione. Sarà invalidato automaticamente, una volta sparato.

Passaggio di dati tramite Timer

Se si desidera passare alcuni dati con il trigger del timer, è possibile farlo con il parametro `userInfo` .

Ecco l'approccio semplice che fornisce una breve idea di come è possibile passare i dati al metodo attivato dal timer.

[*Swift 3*]

```
Timer.scheduledTimer(timeInterval: 1.0, target: self, selector:#selector(iGotCall(sender:)),
userInfo: ["Name": "i am iOS guy"], repeats:true)
```

[*Obiettivo - C*]

```
NSTimer* timer = [NSTimer scheduledTimerWithTimeInterval:1.0
                                                  target:self
                                                selector:@selector(iGotCall:)
                                                userInfo:@"i am iOS guy" repeats:YES];
```

La precedente riga di codice che passa `["Name": "i am iOS guy"]` in `userInfo` . Così ora, quando `iGotCall` riceve la chiamata, puoi ottenere il valore passato sotto lo snippet di codice.

[*Swift 3*]

```
func iGotCall(sender: Timer) {
        print((sender.userInfo)!)
    }
```

[*Obiettivo - C*]

```
- (void)iGotCall:(NSTimer*)theTimer {
    NSLog (@"%@", (NSString*)[theTimer userInfo]);
}
```

Leggi NSTimer online: https://riptutorial.com/it/ios/topic/2624/nstimer

Capitolo 114: NSURL

Examples

Come ottenere l'ultimo componente stringa dalla stringa NSURL.

```
NSURL *url = [NSURL URLWithString:@"http://www.example.com/images/apple-tree.jpg"];
NSString *fileName = [url lastPathComponent];
// fileName = "apple-tree.jpg"
```

Come ottenere l'ultimo componente di stringa dall'URL (NSURL) in 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
```

Leggi NSURL online: https://riptutorial.com/it/ios/topic/4610/nsurl

Capitolo 115: NSURLConnection

Examples

Metodi delegati

// conformare il protocollo NSURLConnectionDelegate.

```
@interface ViewController : UIViewController<NSURLConnectionDelegate>
{
    NSMutableData *_responseData;
}
```

// Implementazione dei metodi del protocollo 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
}
```

Richiesta sincrona

```
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
}
```

Richiesta asincrona

```
// 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];
```

Leggi NSURLConnection online: https://riptutorial.com/it/ios/topic/6004/nsurlconnection

Capitolo 116: NSURLSession

Osservazioni

La classe **NSURLSession** e le classi correlate forniscono un'API per il download del contenuto. Questa API fornisce un ricco set di metodi delegati per supportare l'autenticazione e offre all'app la possibilità di eseguire download in background quando l'app non è in esecuzione o, in iOS, mentre l'app è sospesa.

A un livello elevato, **NSURLSession** si basa sul concetto di sessioni e attività. Un'attività rappresenta una singola richiesta per un singolo URL (o un singolo caricamento su un singolo URL). Una sessione è un gruppo di richieste correlate.

Il sistema operativo fornisce una singola sessione preesistente, la sessione condivisa, che funziona fondamentalmente come NSURLConnection. Inoltre, puoi creare le tue sessioni nella tua app secondo necessità.

App diverse utilizzano le sessioni in modi diversi. Molte app creano una singola sessione all'avvio e continuano a riutilizzarla. Altre app traggono vantaggio dalla possibilità di annullare un gruppo di attività correlate (ad esempio un browser Web che annulla tutte le richieste in sospeso quando si chiude una scheda) e quindi creare una sessione per contenere ciascun gruppo di richieste correlate.

Il primo passaggio quando si utilizza NSURLSession è creare un oggetto di configurazione della sessione. L'oggetto (solitamente) riutilizzabile contiene varie impostazioni di sessione che puoi modificare per le tue esigenze particolari, come la massima concorrenza, intestazioni extra da inviare con ogni richiesta, se consentire l'invio di richieste tramite la radio cellulare (solo iOS), timeout, archiviazione credenziali, versione TLS minima e persino impostazioni proxy.

Esistono tre tipi di configurazioni di sessione, a seconda di come si desidera che si comporti la sessione risultante:

- **Le configurazioni predefinite** creano sessioni che funzionano in modo simile a NSURLConnection.
- **Le configurazioni in background** creano sessioni in cui le richieste avvengono fuori processo, consentendo ai download di continuare anche quando l'app non è più in esecuzione.
- **Le configurazioni effimere** creano sessioni che non memorizzano nulla sul disco, non memorizzano i cookie su disco, ecc. E sono quindi adatti per il backup di finestre del browser in incognito.

Quando si crea una configurazione in background, è necessario fornire un identificatore di sessione che consente di riassociare la sessione in background in un secondo momento (se l'app viene chiusa o sospesa o terminata dal sistema operativo). Non devi avere più di un'istanza di una sessione con lo stesso identificativo attivo nella tua app, quindi di regola queste configurazioni non sono riutilizzabili. Tutte le altre configurazioni di sessione possono essere riutilizzate per creare

tutte le sessioni che vuoi. Pertanto, se è necessario creare più sessioni con impostazioni simili, è possibile creare una volta la configurazione e riutilizzarla ogni volta che si crea una nuova sessione.

Dopo aver creato una sessione, è possibile creare attività in quella sessione. Esistono tre tipi di attività:

- **Le attività** dati restituiscono dati come oggetto **NSData** . Questi sono adatti per l'uso generale, ma non sono supportati nelle sessioni in background.
- **Le attività di download** restituiscono i dati come file su disco. Questi sono adatti per richieste più grandi o per l'uso in sessioni in background.
- **Le attività di** caricamento caricano i dati da un oggetto NSData o da un file su disco. Fornisci un oggetto dati o un file che fornisce il corpo POST. I dati / file del corpo forniti dall'utente sostituiscono tutti i dati / file del corpo forniti nell'oggetto NSURLRequest (se applicabile).

Ciascuno di questi tipi consente di ottenere i dati di risposta in un paio di modi diversi, utilizzando callback basati su blocchi o fornendo un delegato sulla sessione e implementando metodi delegati.

Inoltre NSURLSession consente di fornire metodi delegati per la gestione dell'autenticazione, l'esecuzione della gestione dei certificati TLS personalizzata (sia per i certificati client che per la convalida del server), la modifica del comportamento di memorizzazione nella cache e così via.

Examples

Richiesta GET semplice

```
    // 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()
```

Objective-C Crea un'attività di sessione e dati

```
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];
```

Impostazione della configurazione in background

Per creare una sessione in background

```
 // 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]
```

Inoltre, in iOS, devi impostare il supporto per la gestione del riavvio dell'app in background. Quando l'applicazione `application:handleEventsForBackgroundURLSession:completionHandler:` metodo (Objective-C) o `application(_:handleEventsForBackgroundURLSession:completionHandler:)` viene richiamato il metodo (Swift), significa che l'app è stata riavviata in background per gestire l'attività su una sessione.

In questo metodo, dovresti creare una nuova sessione con l'identificatore fornito e configurarla con un delegato per gestire gli eventi proprio come faresti normalmente in primo piano. Inoltre, è necessario memorizzare il gestore di completamento fornito in un dizionario, utilizzando la sessione come chiave.

Quando il `URLSessionDidFinishEventsForBackgroundURLSession:` del delegato `URLSessionDidFinishEventsForBackgroundURLSession:` (Obj-C) / `URLSessionDidFinishEventsForBackgroundURLSession` (Swift) viene chiamato per dirti che non ci sono altri eventi da gestire, la tua app deve cercare il gestore di completamento per quella sessione, rimuovere la sessione dal dizionario e chiama il gestore di completamento, dicendo al sistema operativo che non hai più alcuna elaborazione in sospeso relativa alla sessione. (Se si sta ancora facendo qualcosa per qualche motivo quando si riceve la chiamata del delegato, attendere fino al termine.) Non appena si chiama quel metodo, la sessione in background viene immediatamente invalidata.

Se la tua applicazione riceve `application:application:didFinishLaunchingWithOptions:` call (probabilmente indicante che l'utente ha messo in primo piano la tua app mentre stavi elaborando gli eventi in background), è sicuro creare una sessione in background con lo stesso identificatore, perché la vecchia sessione con quella l'identificatore non esiste più.

Se sei curioso dei dettagli, ad alto livello, quando crei una sessione in background, stai facendo due cose:

- Creare una sessione in un demone esterno (nsurlsessiond) per gestire i download
- Creazione di una sessione all'interno della tua app che parla con quel demone esterno tramite NSXPC

Normalmente, è pericoloso creare due sessioni con lo stesso ID di sessione in un singolo avvio dell'app, poiché entrambi stanno cercando di parlare con la stessa sessione nel daemon in background. Questo è il motivo per cui la documentazione ufficiale dice di non creare mai più sessioni con lo stesso identificativo. Tuttavia, se la prima sessione era una sessione temporanea creata come parte di una chiamata `handleEventsForBackgroundURLSession` , l'associazione tra la sessione in-app ora invalidata e la sessione nel daemon in background non esiste più.

Invio di una richiesta POST con argomenti utilizzando NSURLSession in Objective-C

Esistono due metodi comuni per codificare un corpo di richiesta POST: codifica URL (application / x-www-form-urlencoded) e dati di modulo (multipart / form-data). Gran parte del codice è simile, ma il modo in cui si costruiscono i dati del corpo è diverso.

Invio di una richiesta utilizzando la codifica URL

Sia che tu abbia un server per la tua piccola applicazione o che lavori in un team con un ingegnere back-end completo, ti consigliamo di parlare con quel server a un certo punto con la tua applicazione iOS.

Nel seguente codice scriveremo una stringa di argomenti che lo script del server di destinazione userà per fare qualcosa che cambia a seconda del caso. Ad esempio, potremmo voler inviare la stringa:

> name = Brendon & password = ABCDE

Al server quando un utente accede alla propria applicazione, in modo che il server possa memorizzare queste informazioni in un database.

Iniziamo. Dovrai creare una richiesta POST NSURLSession con il seguente codice.

```
// 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];
```

Il codice sopra appena creato e generato la richiesta POST al server. Ricordare che l'URL dello script e la stringa di dati POST cambiano in base alla situazione. Se stai leggendo questo, saprai con cosa riempire queste variabili.

Dovrai anche aggiungere un piccolo metodo che codifica l'URL:

```
- (NSString *)URLEncode:(NSString *)originalString encoding:(NSStringEncoding)encoding
{
    return (__bridge_transfer NSString *)CFURLCreateStringByAddingPercentEscapes(
        kCFAllocatorDefault,
        (__bridge CFStringRef)originalString,
        NULL,
        CFSTR(":/?#[]@!$&'()*+,;="),
        CFStringConvertNSStringEncodingToEncoding(encoding));
}
```

Quindi, quando il server ha finito di elaborare questi dati, invierà un ritorno alla tua app iOS. Quindi dobbiamo elaborare questo ritorno, ma come?

Utilizziamo la programmazione basata sugli eventi e usiamo i metodi dei delegati di NSURLSession. Ciò significa che quando il server invia una risposta, questi metodi inizieranno ad attivarsi. I seguenti 5 metodi sono quelli che verranno attivati nell'intera richiesta INTERA, ogni volta che ne viene fatta una:

```
- (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;
```

Di seguito vedrai i metodi sopra riportati utilizzati nel contesto. Ognuno dei loro scopi è piuttosto auto-esplicativo grazie ad Apple, ma ho comunque commentato i loro usi:

```
// 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);
        }
    }
}
```

Quindi è così! Questo è tutto il codice necessario per inviare, ricevere e analizzare una richiesta di API in iOS 9! Va bene ... era un po 'un bel po' di codice. Ma se implementato correttamente come sopra, sarà sicuro! Assicurati di gestire sempre gli errori dove suggerito sopra.

Invio di una richiesta utilizzando la codifica del modulo

La codifica dell'URL è un modo ampiamente compatibile per codificare dati arbitrari. Tuttavia, è relativamente inefficiente per il caricamento di dati binari (come le foto) poiché ogni byte non ASCII diventa un codice a tre caratteri. Inoltre, non supporta gli allegati di file, quindi è necessario passare nomi di file e dati di file come campi separati.

Supponiamo di voler caricare una fotografia in modo efficiente e in effetti simile a un file sul lato server. Un modo per farlo è usare invece la codifica del modulo. Per fare ciò, modificare il codice che crea NSURLSession come segue:

```
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];
```

Questo crea e attiva la richiesta NSURLSession esattamente come prima e, di conseguenza, i metodi delegati si comporteranno esattamente nello stesso modo. Assicurarsi che lo script l'immagine viene inviata a (che si trova presso l'URL nella variabile `url`) è in attesa di un'immagine e in grado di analizzare in modo corretto.

Leggi NSURLSession online: https://riptutorial.com/it/ios/topic/2009/nsurlsession

Capitolo 117: NSUserActivity

introduzione

Un oggetto NSUserActivity può essere utilizzato per coordinare eventi significativi in un'app con il sistema. È la base per Handoff tra diversi dispositivi che eseguono iOS e macOS. Inoltre, può anche essere utilizzato per migliorare l'indicizzazione pubblica e aumentare o creare risultati di ricerca Spotlight per un'app. A partire da iOS 10, può anche essere usato per coordinare le interazioni tra la tua app e Siri utilizzando SiriKit.

Osservazioni

Tipi di attività

I tipi di attività supportati devono essere definiti nel file `Info.plist` dell'app sotto la chiave `NSUserActivityTypes` . Le attività sono legate all'ID del tuo sviluppatore, il che significa che il coordinamento delle attività è limitato tra le app che hanno lo stesso ID di squadra (ad es. "Safari" non può accettare un'attività di Handoff da "Chrome" o viceversa).

Diventare / rassegnare l'attività corrente

Contrassegnare un'attività come corrente usando `becomeCurrent` rende disponibile per Handoff o Spotlight Indexing. Solo una attività può essere corrente alla volta. È possibile contrassegnare un'attività come inattiva senza invalidare chiamando `resignCurrent` .

Se `invalidate` un'attività, la stessa istanza potrebbe non essere resa di nuovo corrente.

Non contrassegnare un'attività come corrente quando la si fornisce a SiriKit .

Ricerca indicizzazione

Le attività **non** devono essere utilizzate come meccanismo di indicizzazione generico all'interno della tua app. Invece, dovrebbero essere utilizzati solo in risposta alle azioni avviate dall'utente. Per indicizzare tutti i contenuti della tua app, usa CoreSpotlight.

Examples

Creazione di NSUserActivity

Per creare un oggetto `NSUserActivity` , la tua app deve dichiarare i tipi di attività che supporta nel suo file `Info.plist` . Le attività supportate sono definite dalla tua applicazione e dovrebbero essere

uniche. Un'attività viene definita utilizzando uno schema di denominazione degli stili di dominio inverso (ad esempio "com.companyName.productName.activityName"). Ecco come può apparire una voce nel tuo Info.plist:

Chiave	Valore
NSUserActivityTypes	[Array]
- item0	com.companyName.productName.activityName01
- item1	com.companyName.productName.activityName02

Una volta definiti tutti i tipi di attività supportati, è possibile iniziare ad accedere e utilizzarli nel codice dell'applicazione.

Per creare un oggetto `NSUserActivity` devi eseguire quanto segue

```
// 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];
```

Dopo questo, l'attività di cui sopra dovrebbe essere disponibile per Handoff (anche se è necessario più lavoro per gestire correttamente il "Handoff").

Leggi NSUserActivity online: https://riptutorial.com/it/ios/topic/10716/nsuseractivity

Capitolo 118: NSUserDefaults

Sintassi

- `UserDefaults.standard.set(dic, forKey: "LoginSession") //Save value inside userdefaults`
    - `UserDefaults.standard.object(forKey: "LoginSession") as? [String:AnyObject] ?? [:] //Get value from UserDefaults`

Osservazioni

NSUserDefault che vengono utilizzati per archiviare tutti i tipi di DataType, e puoi ottenere il loro valore ovunque nella classe dell'app. **NSUserDefault**

Examples

Impostazione dei valori

Per impostare un valore in `NSUserDefaults` , è possibile utilizzare le seguenti funzioni:

Swift <3

```
setBool(_:forKey:)
setFloat(_:forKey:)
setInteger(_:forKey:)
setObject(_:forKey:)
setDouble(_:forKey:)
setURL(_:forKey:)
```

Swift 3

A Swift 3 i nomi di funzione viene modificata per `set` posto della `set` folloed dal tipo.

```
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;
```

L'utilizzo di esempio potrebbe essere:

Swift <3

```
NSUserDefaults.standardUserDefaults.setObject("Netherlands", forKey: "HomeCountry")
```

Swift 3

```
UserDefaults.standard.set("Netherlands", forKey: "HomeCountry")
```

Objective-C

```
[[NSUserDefaults standardUserDefaults] setObject:@"Netherlands" forKey:@"HomeCountry"];
```

Oggetti personalizzati

Per salvare gli oggetti personalizzati in `NSUserDefaults` devi rendere CustomClass confermato al protocollo di` NSCoding`. È necessario implementare i seguenti metodi:

veloce

```
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"];
}
```

Ottenere valori predefiniti

Per ottenere un valore in NSUserDefaults è possibile utilizzare le seguenti funzioni:

veloce

```
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;
```

L'utilizzo di esempio potrebbe essere:

veloce

```
let homeCountry = NSUserDefaults.standardUserDefaults().stringForKey("HomeCountry")
```

Objective-C

```
NSString *homeCountry = [[NSUserDefaults standardUserDefaults] stringForKey:@"HomeCountry"];
```

Salvataggio dei valori

`NSUserDefaults` sono scritti periodicamente sul disco dal sistema, ma ci sono momenti in cui si desidera che le modifiche vengano salvate immediatamente, ad esempio quando l'app passa allo stato di background. Questo viene fatto chiamando la `synchronize` .

veloce

```
NSUserDefaults.standardUserDefaults().synchronize()
```

Objective-C

```
[[NSUserDefaults standardUserDefaults] synchronize];
```

Utilizzare i gestori per salvare e leggere i dati

Sebbene sia possibile utilizzare i metodi `NSUserDefaults` ovunque, a volte può essere preferibile definire un gestore che salva e legge da `NSUserDefaults` e quindi utilizzare tale gestore per leggere o scrivere i dati.

Supponiamo di voler salvare il punteggio di un utente in `NSUserDefaults` . Possiamo creare una classe come quella qui sotto che ha due metodi: `setHighScore` e `highScore` . Ovunque tu voglia accedere ai punteggi più alti, crea un'istanza di questa classe.

veloce

```
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
```

I vantaggi sono:

1. L'implementazione del processo di lettura e scrittura è solo in un punto e puoi cambiarla (ad esempio passare da `NSUserDefaults` a Core Data) ogni volta che vuoi e non preoccuparti di cambiare tutte le posizioni con cui lavori con il punteggio più alto.
2. Basta chiamare un solo metodo quando si desidera accedere al punteggio o scriverlo.
3. Semplicemente esegui il debug quando vedi un bug o qualcosa di simile.

> **Nota**
>
> Se sei preoccupato per la sincronizzazione, è meglio usare una classe singleton che gestisca la sincronizzazione.

Cancellazione di NSUserDefaults

veloce

```
let bundleIdentifier = NSBundle.mainBundle().bundleIdentifier()

NSUserDefaults.standardUserDefaults().removePersistentDomainForName(bundleIdentifier)
```

Objective-C

```
NSString *bundleIdentifier = [[NSBundle mainBundle] bundleIdentifier];

[[NSUserDefaults standardUserDefaults] removePersistentDomainForName: bundleIdentifier];
```

UserDefaults utilizza in Swift 3

Tutte le applicazioni necessarie per archiviare la sessione utente oi dettagli relativi all'utente all'interno dell'applicazione in UserDefaults. Così abbiamo fatto tutta la logica all'interno di una classe per la gestione di UserDefaults in modo migliore.

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
    }

}
```

Uso della classe UserDefaults

```
//Saving user Details
Session.setUserSessionDetails(json ?? [:])

//Retriving user Details
let userId = Session.getUserValues(value: .userId) ?? ""
```

Leggi NSUserDefaults online: https://riptutorial.com/it/ios/topic/3150/nsuserdefaults

Capitolo 119: Objective-C Oggetti associati

introduzione

Introdotto per la prima volta in iOS 3.1 come parte del runtime Objective-C, gli oggetti associati forniscono un modo per aggiungere variabili di istanza a un oggetto di classe esistente (senza sottoclasse.

Ciò significa che potrai allegare qualsiasi oggetto a qualsiasi altro oggetto senza sottoclassi.

Sintassi

- void objc_setAssociatedObject (id id, void * key, id valore, objc_AssociationPolicy policy)
- id objc_getAssociatedObject (oggetto id, void * key)
- void objc_removeAssociatedObjects (oggetto id)

Parametri

Param	Dettagli
oggetto	L'oggetto esistente che si desidera modificare
chiave	Questo può essere praticamente qualsiasi puntatore che ha un indirizzo di memoria costante, ma una buona pratica è usare qui una proprietà calcolata (getter)
valore	L'oggetto che vuoi aggiungere
politica	La politica di memoria per questo nuovo `value` vale a dire dovrebbe essere mantenuta / assegnata, copiata ecc. Proprio come qualsiasi altra proprietà dichiarata

Osservazioni

Maggiori dettagli qui:

NSHipster

@kostiakoval

kingscocoa

Examples

Esempio di oggetto associato di base

Supponiamo di dover aggiungere un oggetto NSString a `SomeClass` (non possiamo `SomeClass` sottoclassi).

In questo esempio non creiamo solo un oggetto associato, ma lo racchiudiamo anche in una proprietà calcolata in una categoria per maggiore pulizia

```
#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));
}
```

Ora sarebbe facile come usare la proprietà

```
SomeClass *instance = [SomeClass alloc] init];
instance.associated = @"this property is an associated object under the hood";
```

Leggi Objective-C Oggetti associati online: https://riptutorial.com/it/ios/topic/9102/objective-c-oggetti-associati

Capitolo 120: OpenGL

introduzione

OpenGL ES è una libreria grafica che iOS usa per il rendering 3D.

Examples

Progetto di esempio

Un esempio di progetto (Git repo) che può essere usato come punto di partenza per fare un po 'di rendering 3D. Il codice per l'impostazione di OpenGL e degli shader è piuttosto lungo e noioso, quindi non si adatta bene a questo formato di esempio. In seguito pezzi di esso possono essere mostrati in esempi separati che descrivono cosa succede esattamente con ogni pezzo di codice, ma per ora qui è il progetto Xcode.

Leggi OpenGL online: https://riptutorial.com/it/ios/topic/9324/opengl

Capitolo 121: Operazioni a livello di app

Examples

Ottieni il massimo da UIViewController

Un approccio comune per ottenere il massimo da `UIViewController` è ottenere il RootViewController della `UIWindow` attiva. Ho scritto un'estensione per questo:

```
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!
}
```

Intercept System Events

Utilizzando NotificationCenter di iOS, che può essere molto potente, puoi intercettare determinati eventi a livello di app:

```
NotificationCenter.default.addObserver(
        self,
        selector: #selector(ViewController.do(_:)),
        name: NSNotification.Name.UIApplicationDidBecomeActive,
        object: nil)
```

È possibile registrarsi per un sacco di altri eventi, basta dare un'occhiata a https://developer.apple.com/reference/foundation/nsnotification.name .

Leggi Operazioni a livello di app online: https://riptutorial.com/it/ios/topic/7188/operazioni-a-livello-di-app

Capitolo 122: Panoramiche personalizzate dai file XIB

Osservazioni

Da Apple: creazione di una visualizzazione personalizzata che genera in Interface Builder

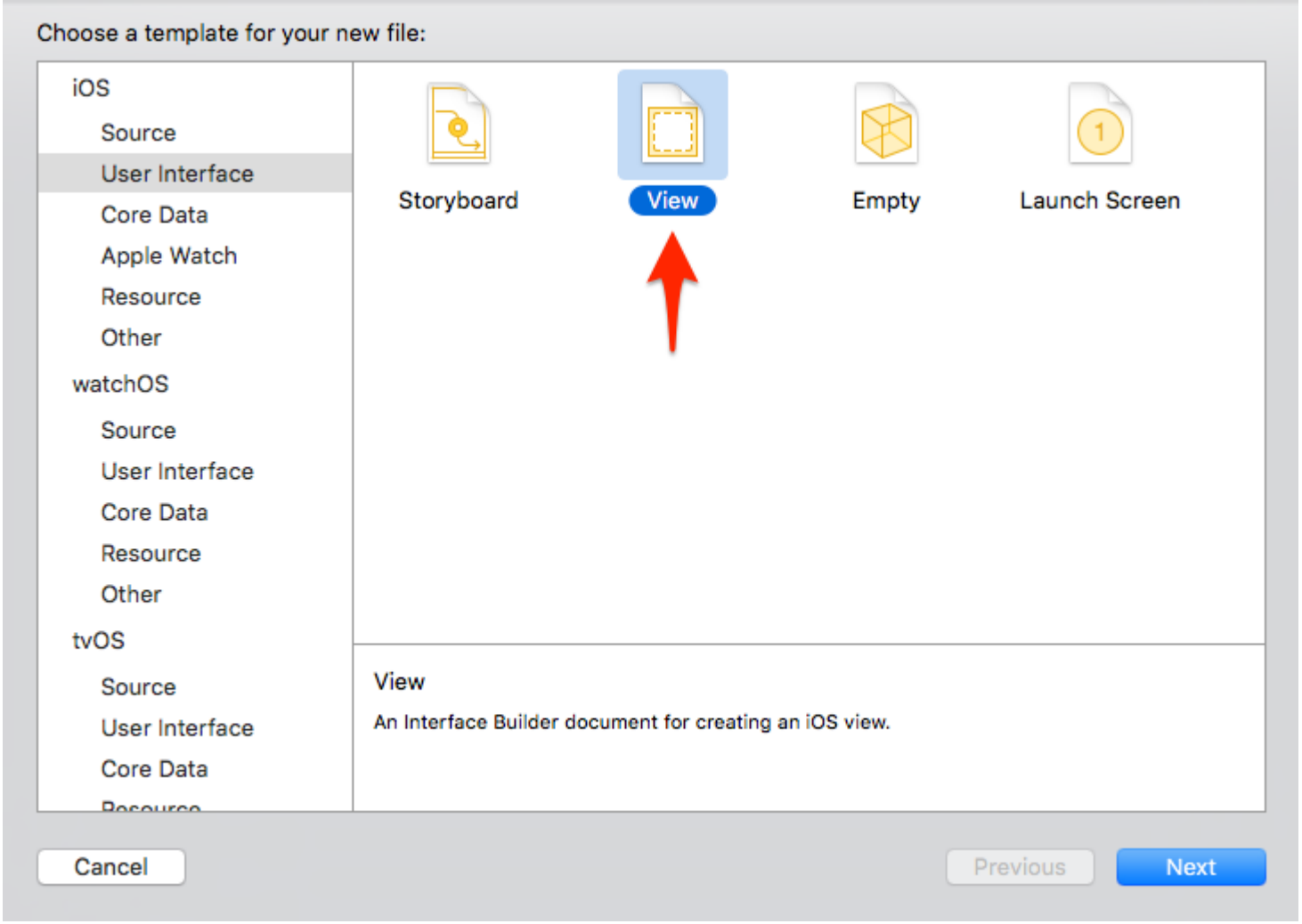
- Nota: tenere presente che se si utilizzano font personalizzati "personalizzati" nei propri elementi XIB (come UILabel, UITextField ecc.), Il tempo di caricamento iniziale dell'XIB sarà più lungo a seconda del font scelto e della versione del sistema.

Examples

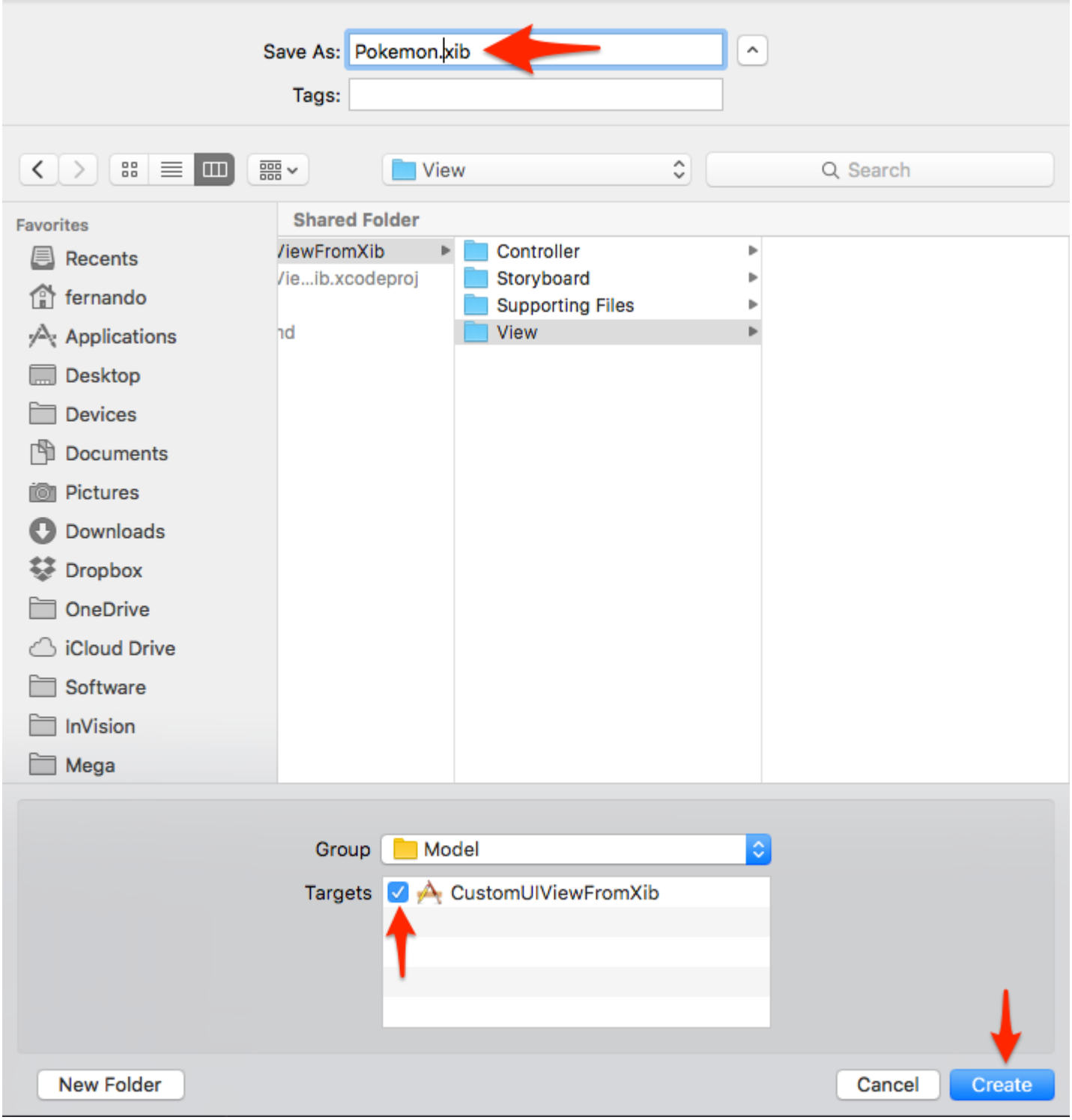
Elementi di cablaggio

Crea un file XIB

Barra dei menu Xcode> File> Nuovo> File.
Seleziona iOS, Interfaccia utente e quindi "Visualizza":



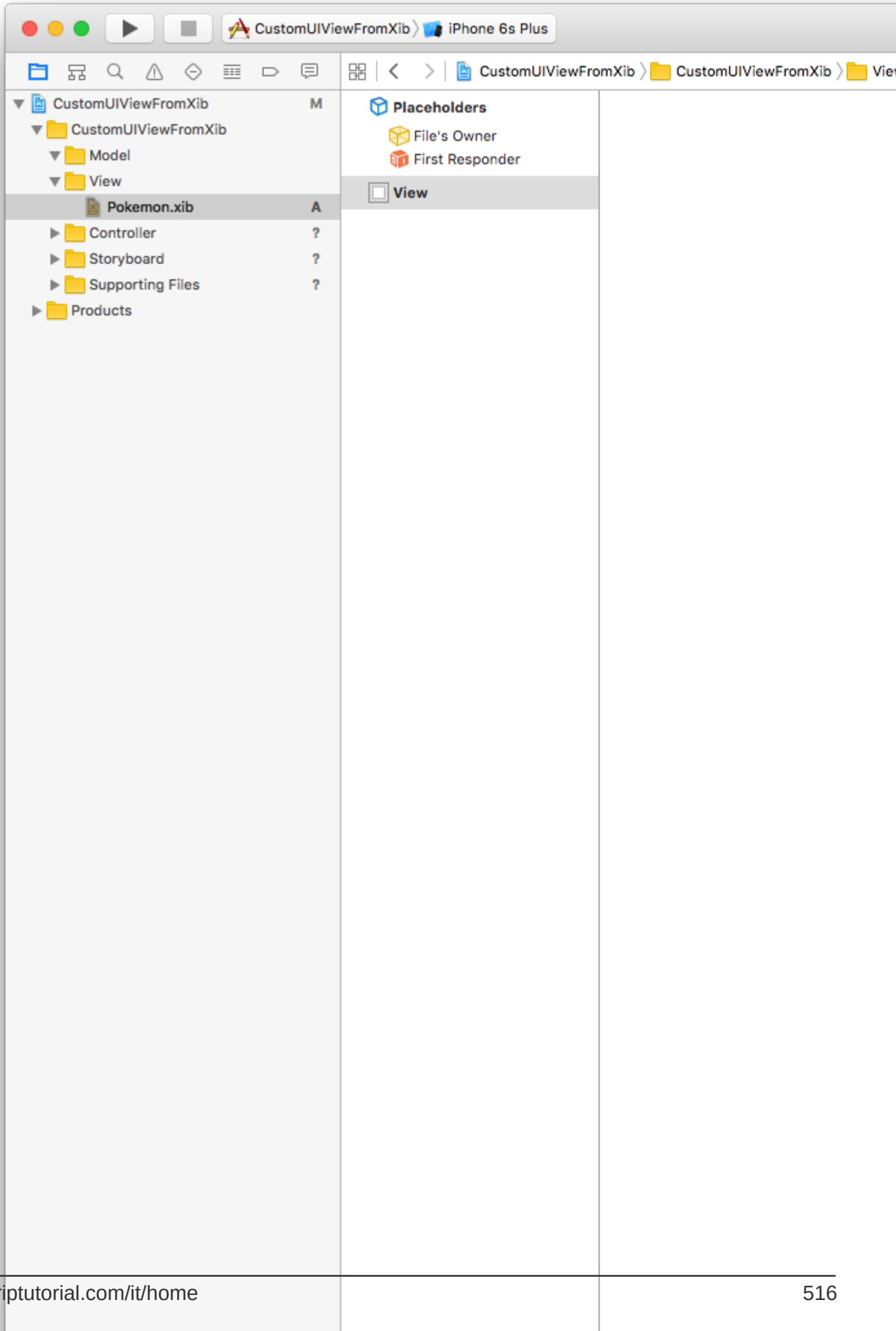
Dai un nome al tuo XIB (sì, stiamo facendo un esempio di Pokemon).
Ricordati di controllare il tuo obiettivo e premi "Crea".



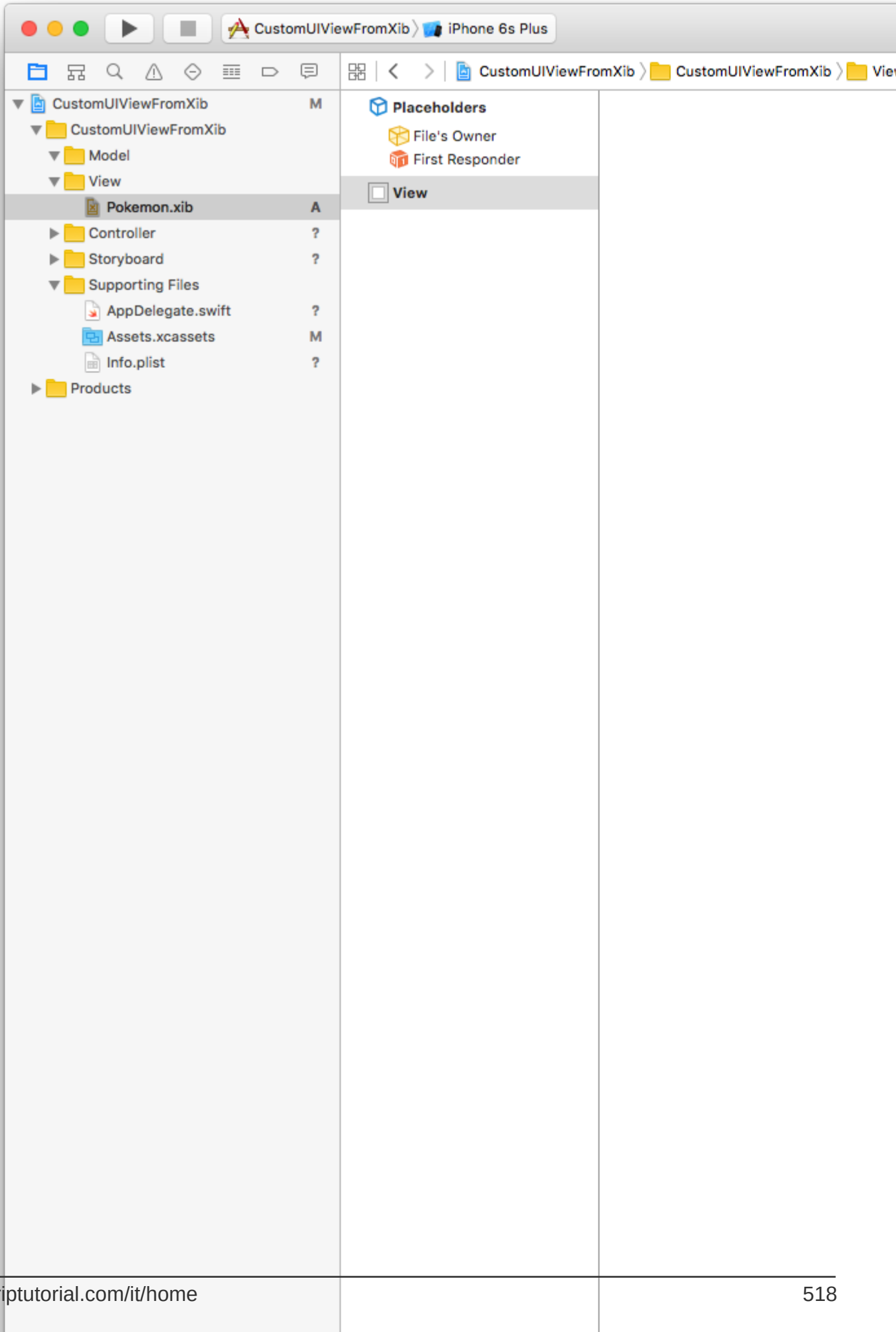
Progetta il tuo punto di vista

Per semplificare le cose, imposta:

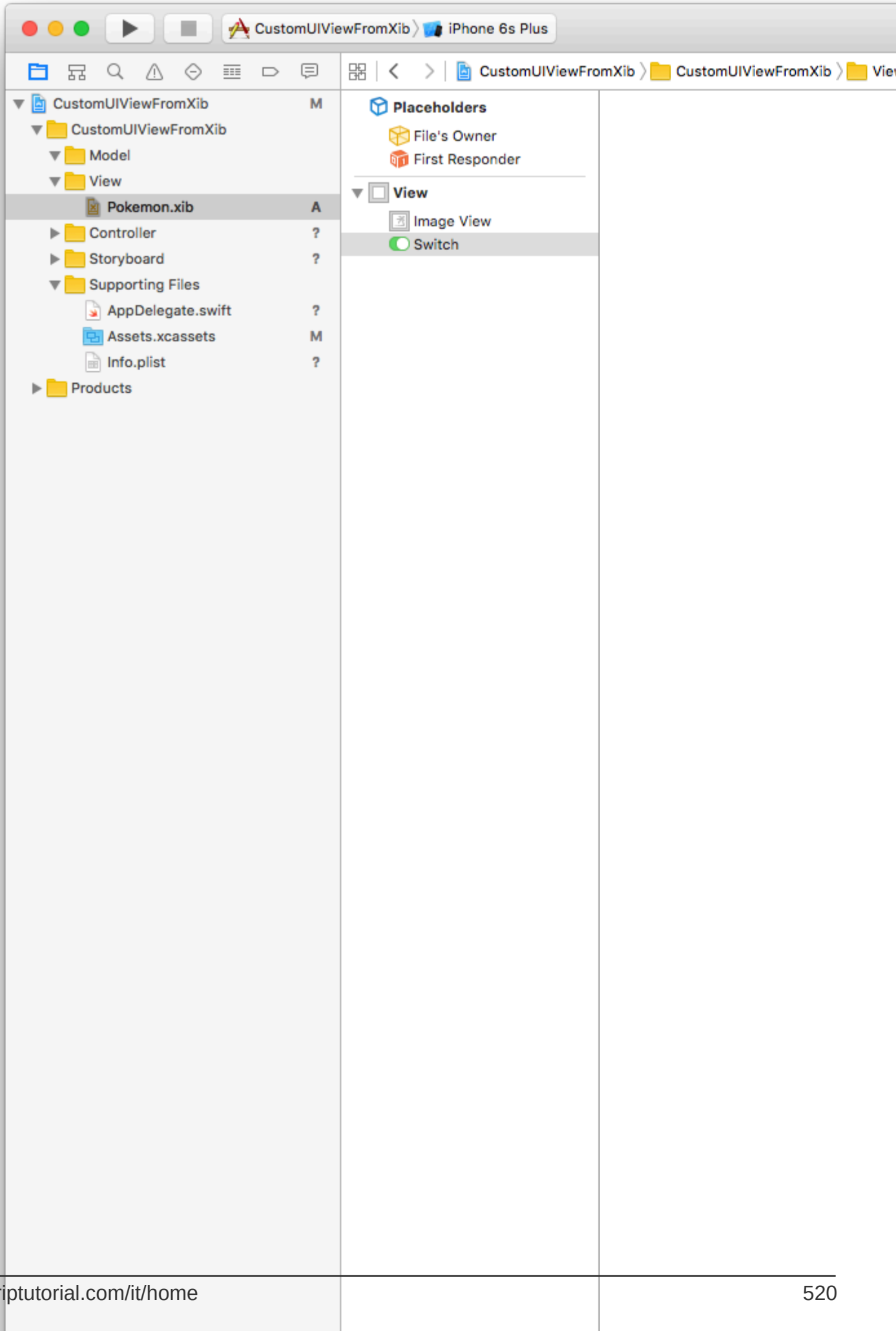
- Dimensione: Freeform
- Barra di stato: nessuna
- Barra superiore: nessuna
- Barra inferiore: nessuna


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

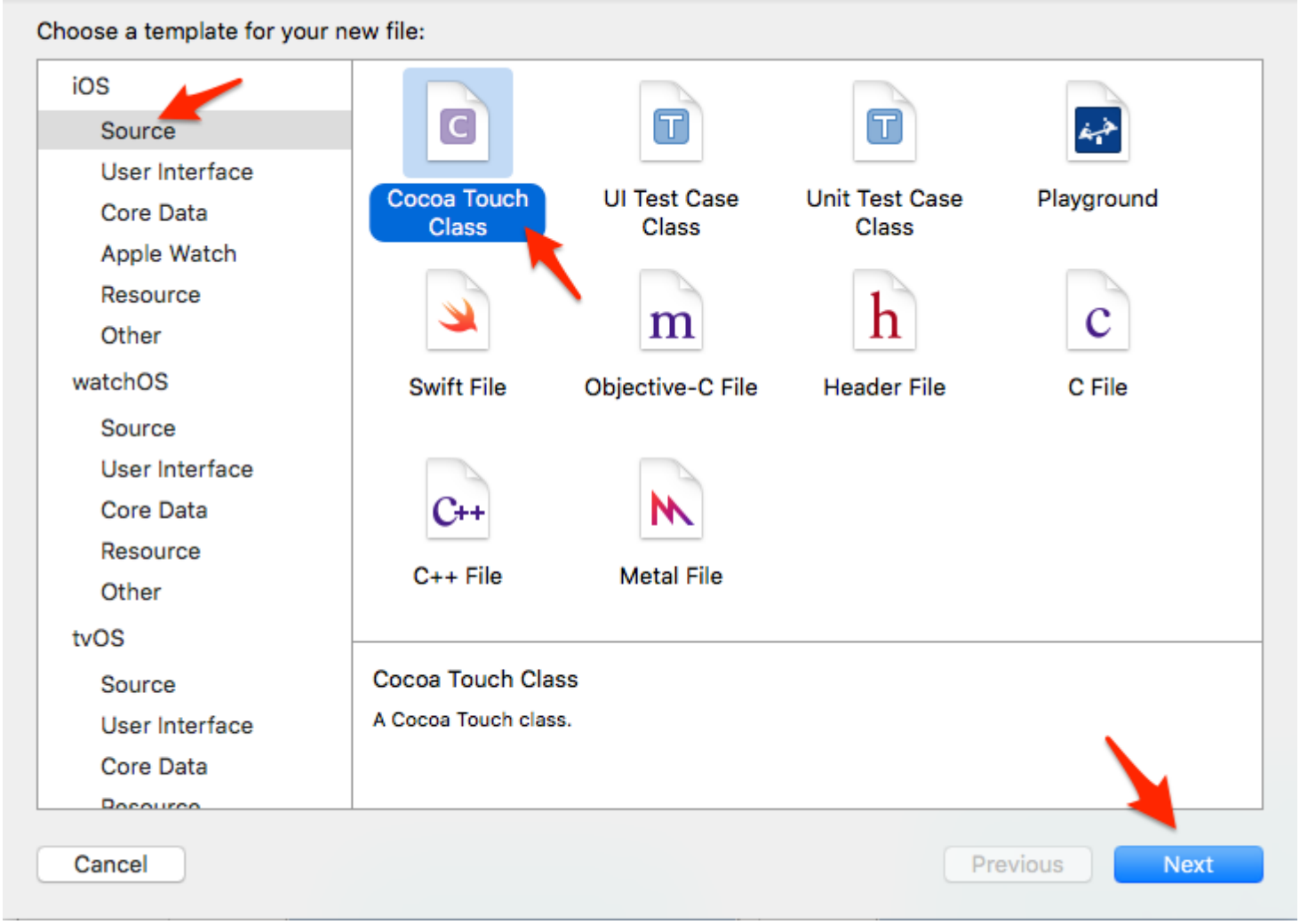
Per questo esempio useremo la larghezza 321 e l'altezza 256.


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

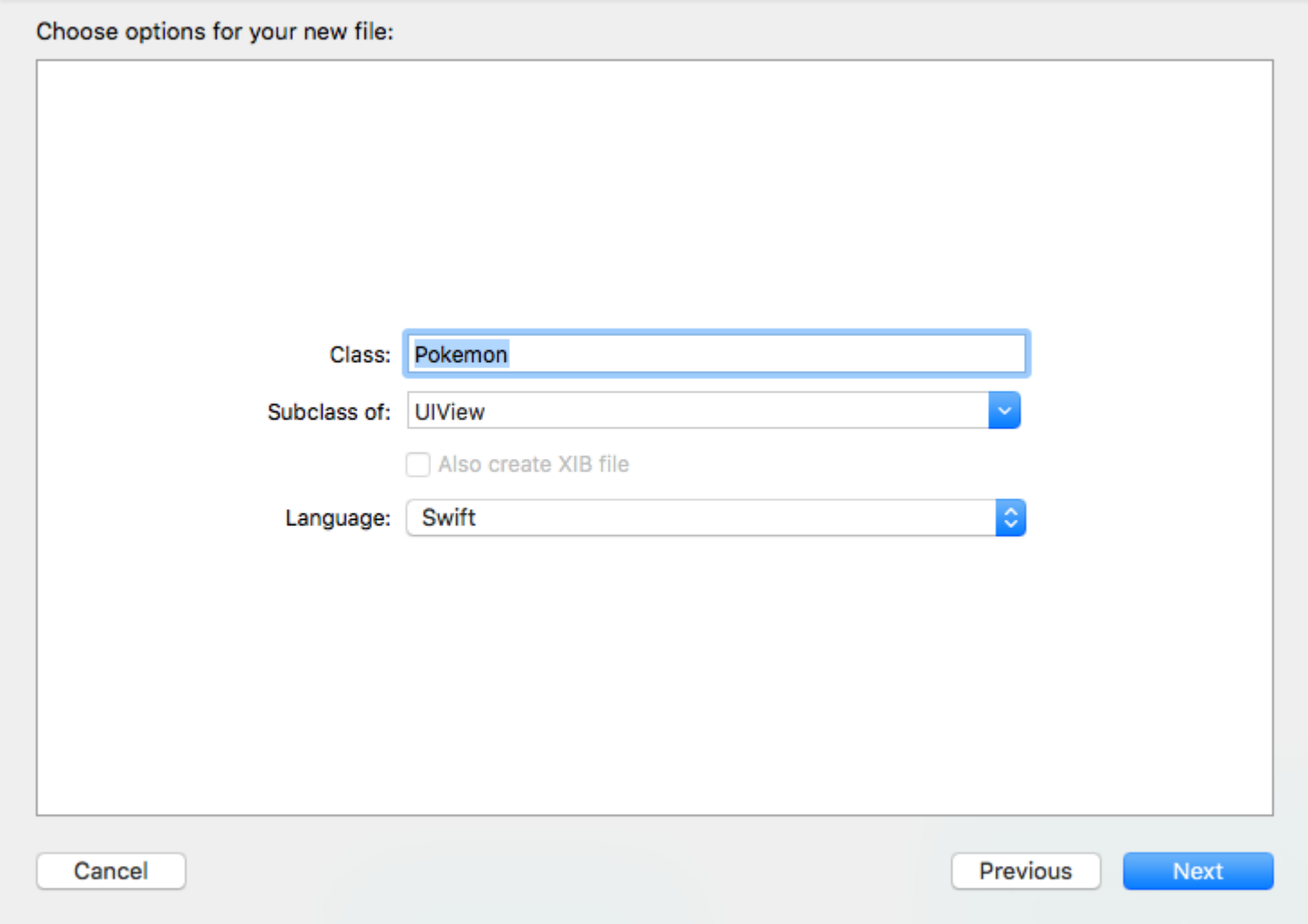
Qui aggiungeremo una **vista immagine** (256x256) e uno **switch** .


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

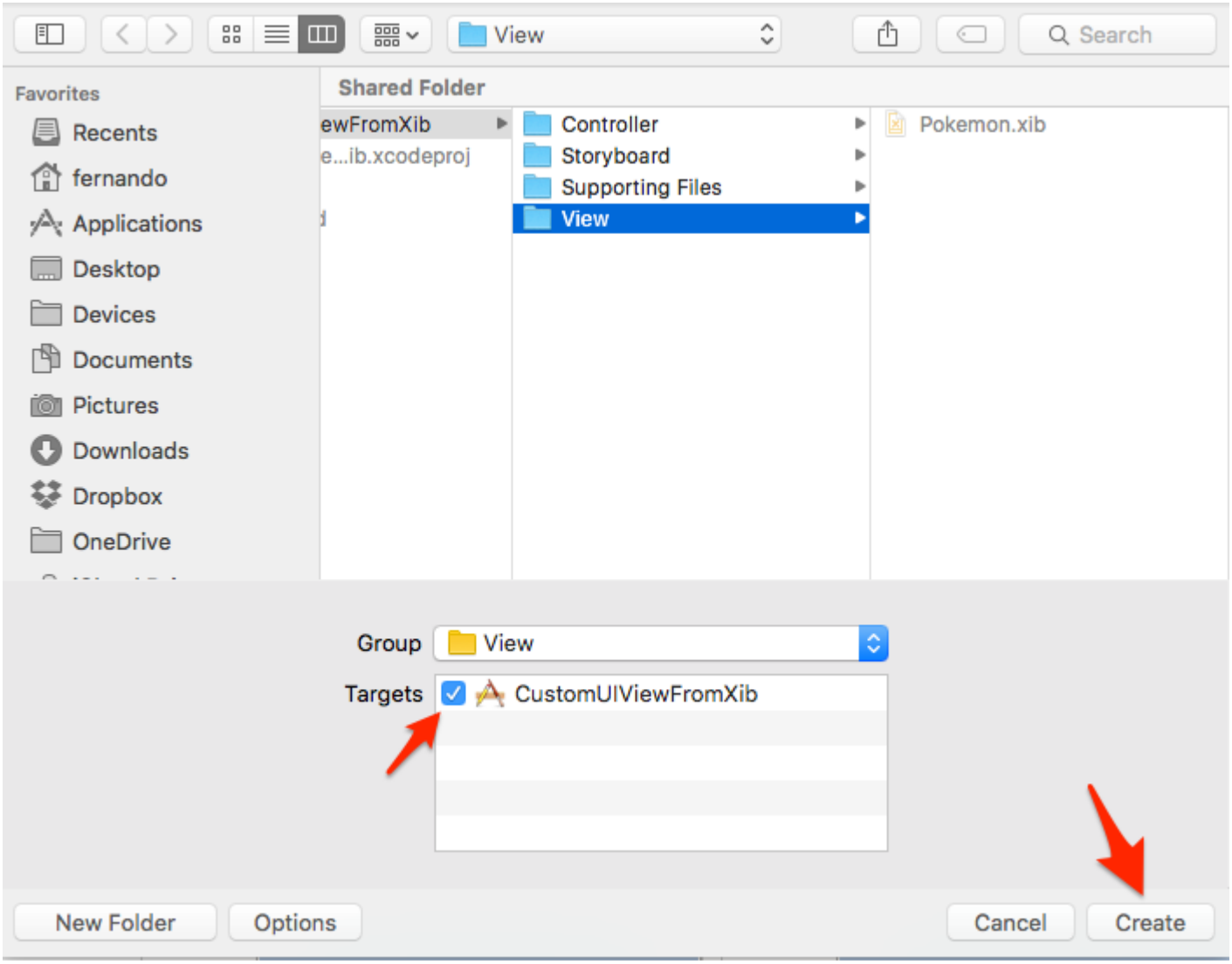
Barra dei menu Xcode> File> Nuovo> File.
Seleziona iOS / Source / Cocoa Touch Class. Premi "Avanti".



Assegna un nome alla classe, che deve essere lo stesso nome del file XIB (Pokemon).
Seleziona UIView come tipo di sottoclasse, quindi premi "Avanti".

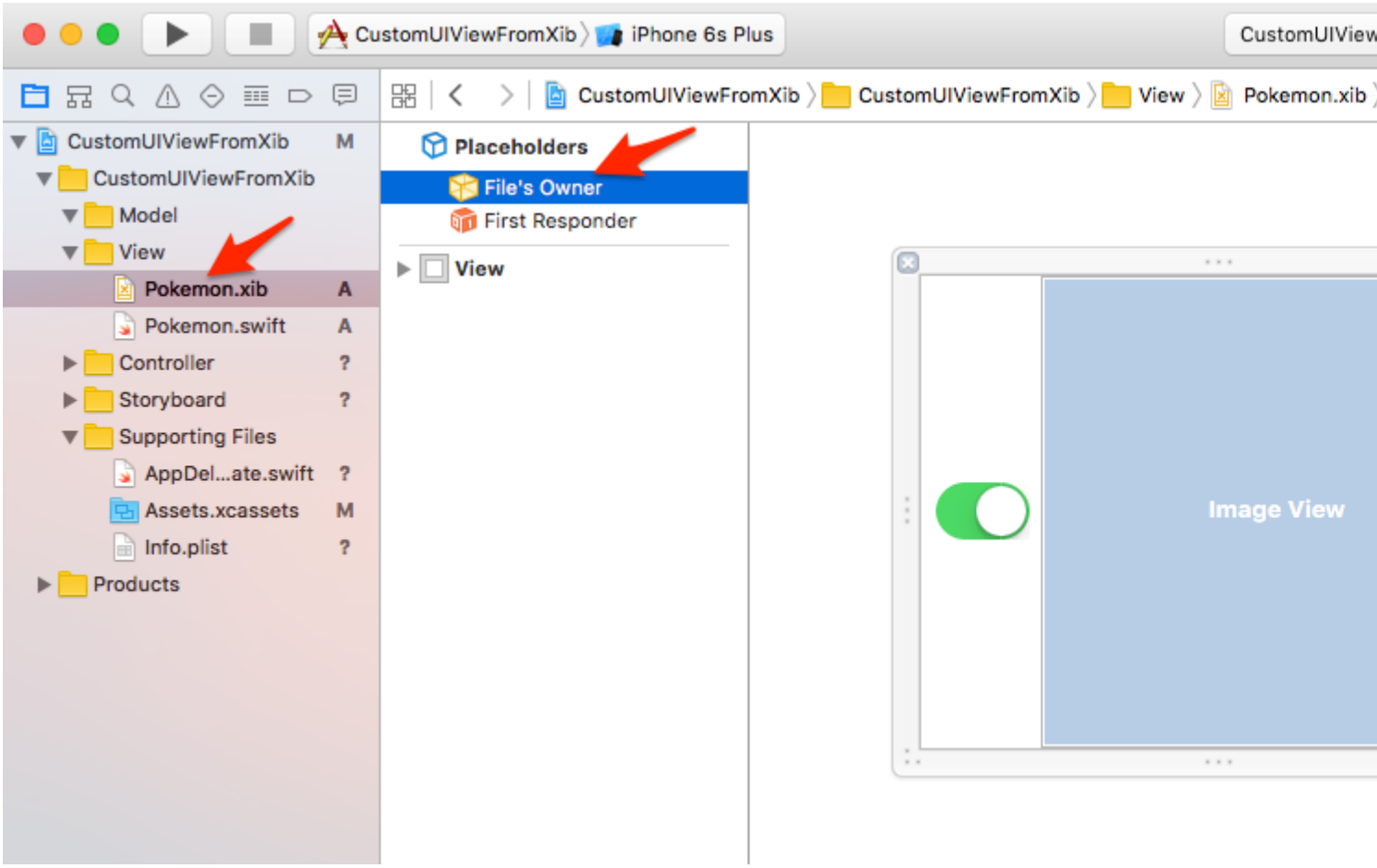


Nella finestra successiva, seleziona il tuo obiettivo e premi "Crea".



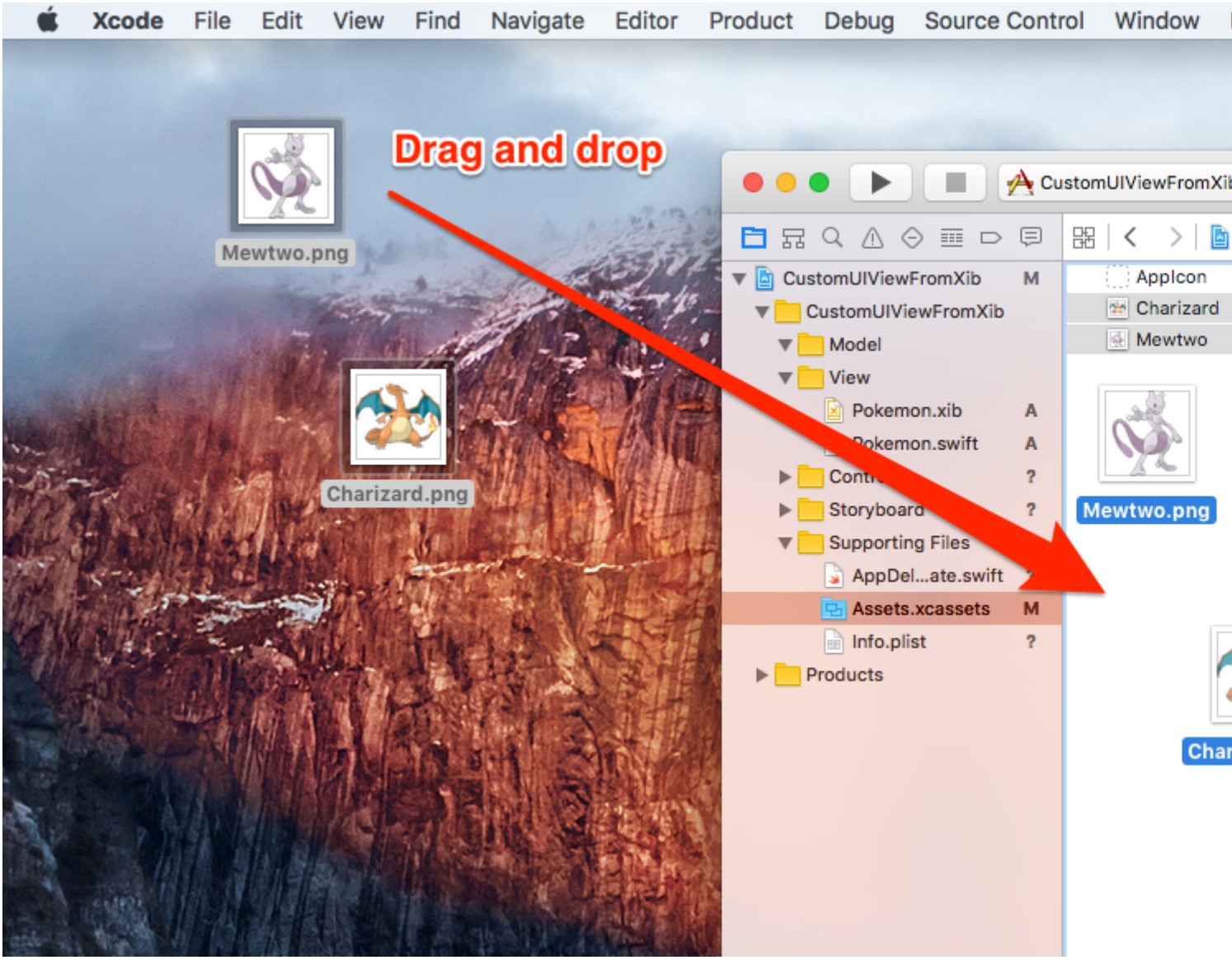
Collega Pokemon.xib a Pokemon.swift tramite l'attributo "Proprietario del file"

Fare clic sul file Pokemon.xib in Xcode.
Clicca sulla presa "File's Owner".
In "Identity inspector" (in alto a destra), imposta la classe sul nostro file Pokemon.swift creato di recente.



Pokemon !!!

Sì! Trascina e rilascia alcuni Pokemon nel tuo progetto per completare la nostra "infrastruttura". Qui stiamo aggiungendo due file PGN, 256x256, trasparente.



> Fammi vedere il codice già.

Tutto bene tutto bene.
È ora di aggiungere del codice alla nostra classe Pokemon.swift.

In realtà è piuttosto semplice:

1. Implementare gli inizializzatori richiesti
2. Carica il file XIB
3. Configurare la vista che visualizzerà il file XIB
4. Mostra la vista sopra

Aggiungi il seguente codice alla classe Pokemon.swift:

```
import UIKit

class Pokemon: UIView {

    // MARK: - Initializers

    override init(frame: CGRect) {
```

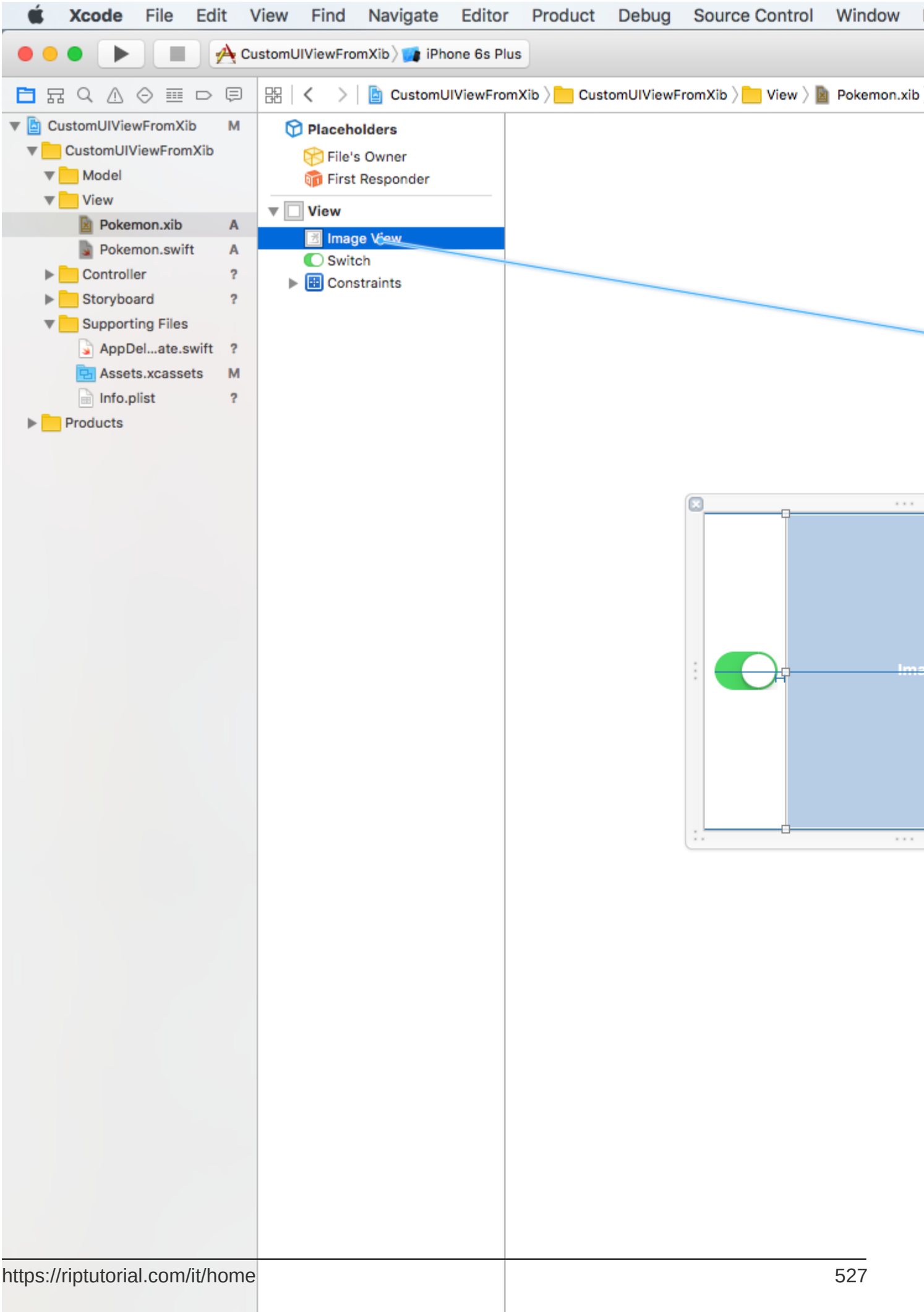
```
        super.init(frame: frame)
        setupView()
    }

    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 e @IBInspectable

Aggiungendo `@IBDesignable` alla tua classe, rendi possibile il rendering live in Interface Builder. Aggiungendo `@IBInspectable` alle proprietà della classe, è possibile visualizzare le visualizzazioni personalizzate modificando in Interface Builder non appena si modificano tali proprietà.

Facciamo la `Image View` della nostra vista personalizzata "Ispezionabile".

Innanzitutto, collega la `Image View` dal file Pokemon.xib alla classe Pokemon.swift.

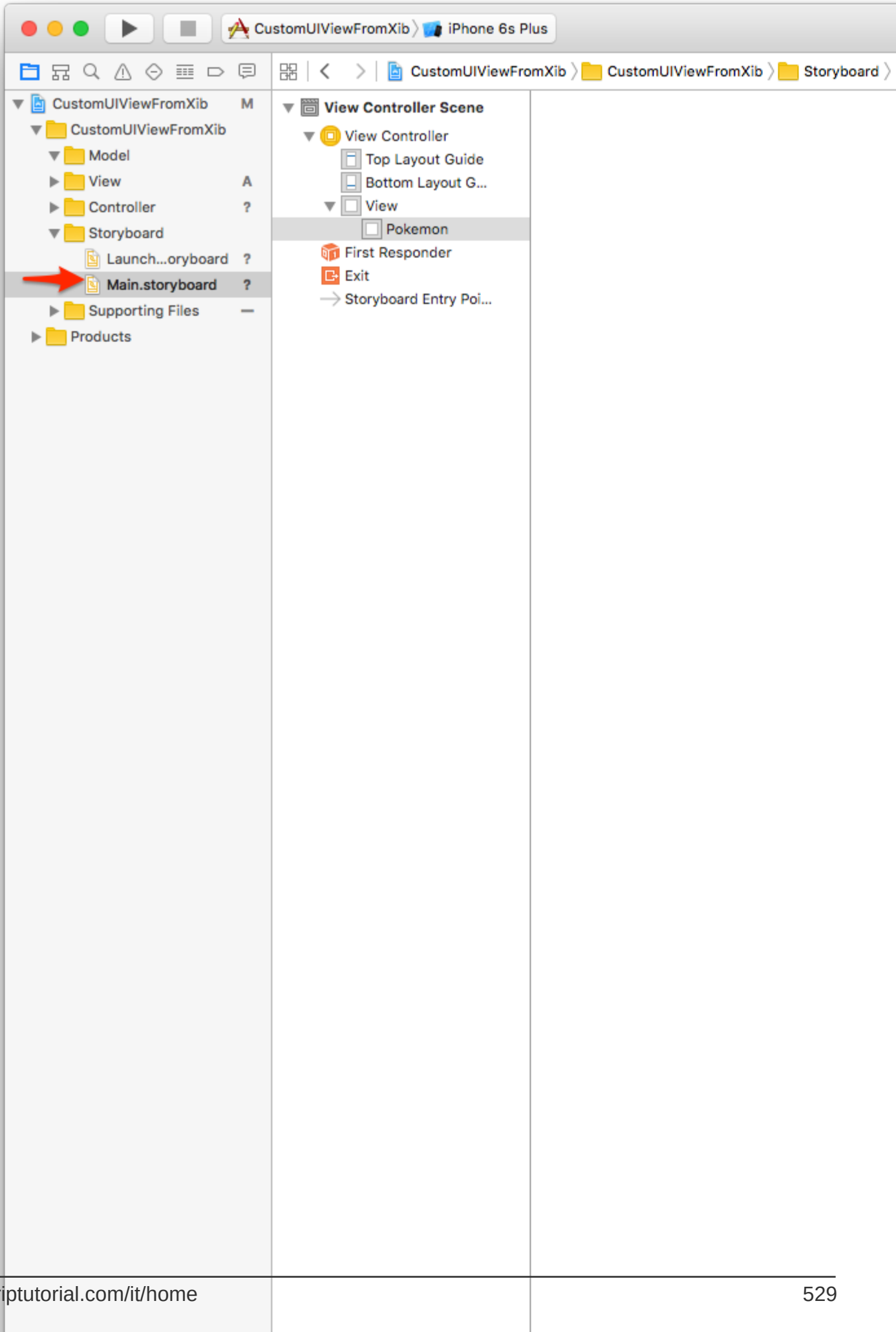

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window
CustomUIViewFromXib iPhone 6s Plus
CustomUIViewFromXib CustomUIViewFromXib View Pokemon.xib
CustomUIViewFromXib M
CustomUIViewFromXib
Model
View
Pokemon.xib A
Pokemon.swift A
Controller ?
Storyboard ?
Supporting Files
AppDel...ate.swift ?
Assets.xcassets M
Info.plist ?
Products
Placeholders
File's Owner
First Responder
View
Image View
Switch
Constraints

prima del nome della classe):

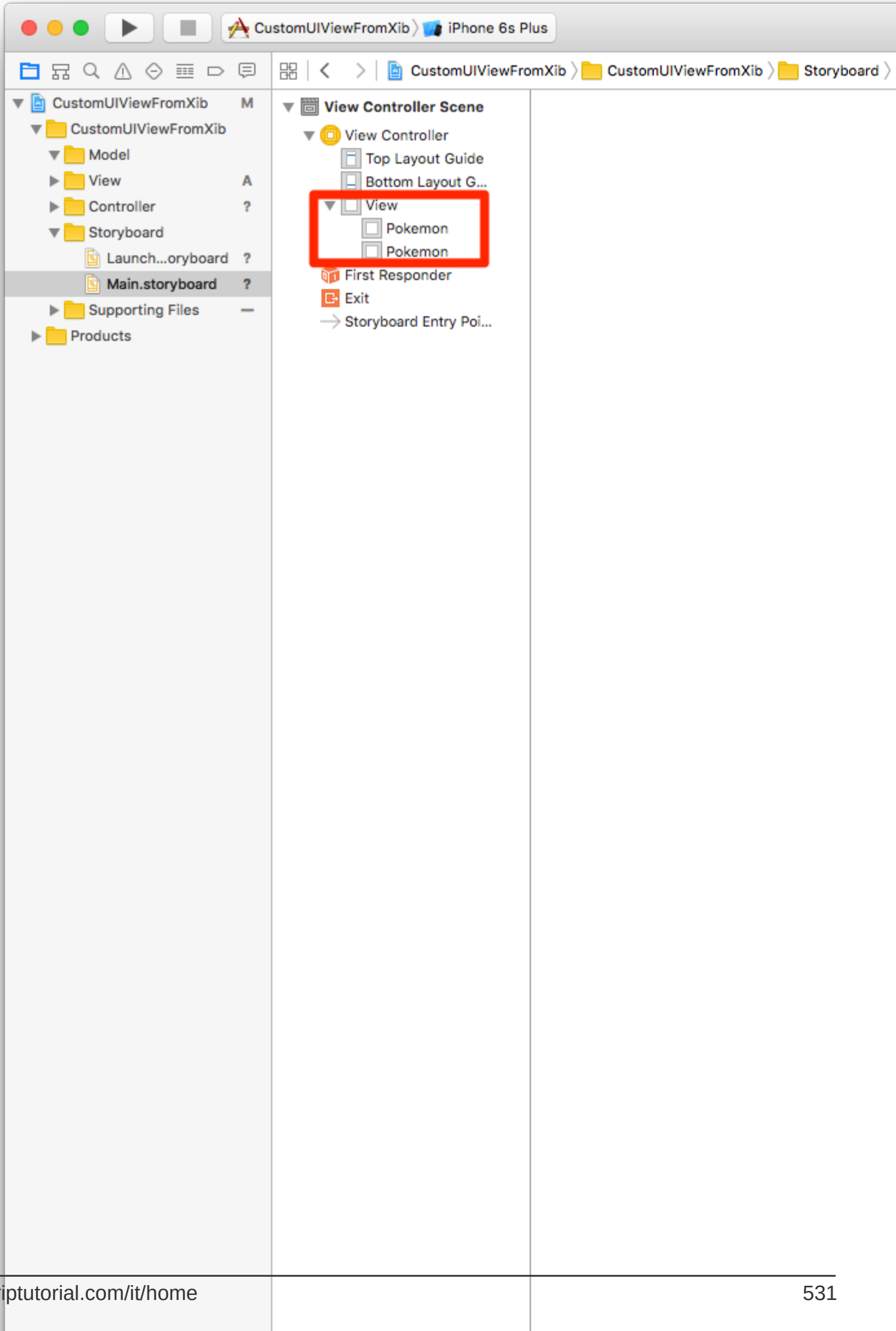
```
@IBDesignable class Pokemon: UIView {

    // MARK: - Properties

    @IBOutlet weak var imageView: UIImageView!

    @IBInspectable var image: UIImage? {
        get {
            return imageView.image
        }
        set(image) {
            imageView.image = image
        }
    }

    // MARK: - Initializers
    ...
```

Utilizzando le tue viste personalizzate

Vai al tuo file di storyboard principale, trascina un UIView in esso.
Ridimensiona la visualizzazione, ad esempio 200x200. Centralizzare.
Vai a Identity inspector (in alto a destra) e imposta la classe su Pokemon.


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...

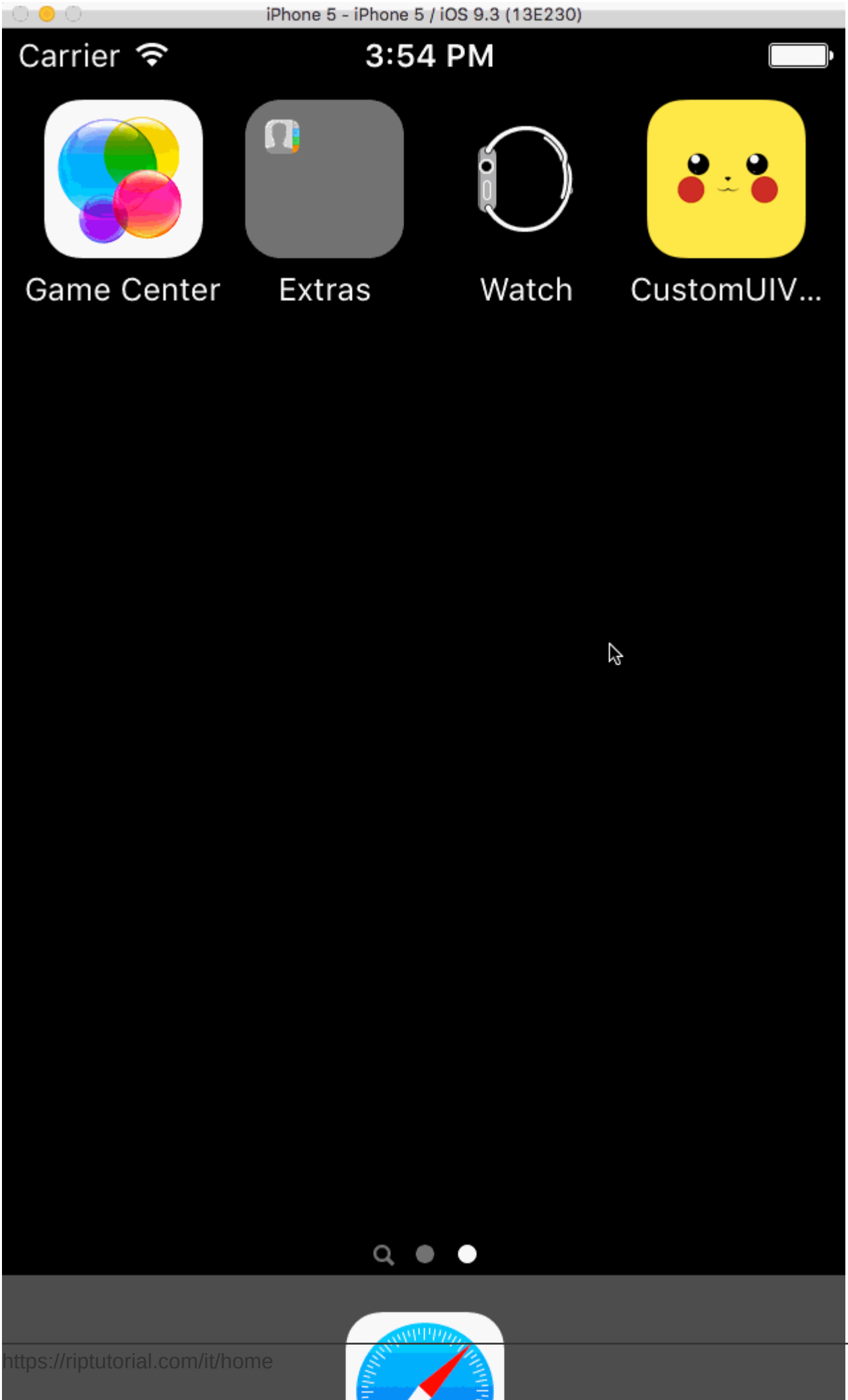
Dagli una dimensione diversa, diciamo 150x150.
Scegli un'altra immagine di Pokemon, osserva:


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...

Chiamare l'azione qualcosa come `switchTapped` .
Aggiungi il seguente codice ad esso:

```
// MARK: - Actions

@IBAction func switchTapped(sender: UISwitch) {
    imageView.alpha = sender.on ? 1.0 : 0.2
}

// MARK: - Initializers
...
```

Risultato finale:


iPhone 5 - iPhone 5 / iOS 9.3 (13E230)
Carrier
3:54 PM
Game Center
Extras
Watch
CustomUIV...

Ora puoi creare complesse visualizzazioni personalizzate e riutilizzarle ovunque desideri. Ciò consentirà di aumentare la produttività isolando il codice in elementi dell'interfaccia utente autonomi.

Il progetto finale può essere clonato in Github.
(**Aggiornato a Swift 3.1**)

Come rendere UIView riutilizzabile personalizzato usando XIB

L'esempio seguente mostra i passaggi coinvolti nell'inizializzazione di una vista da XIB.

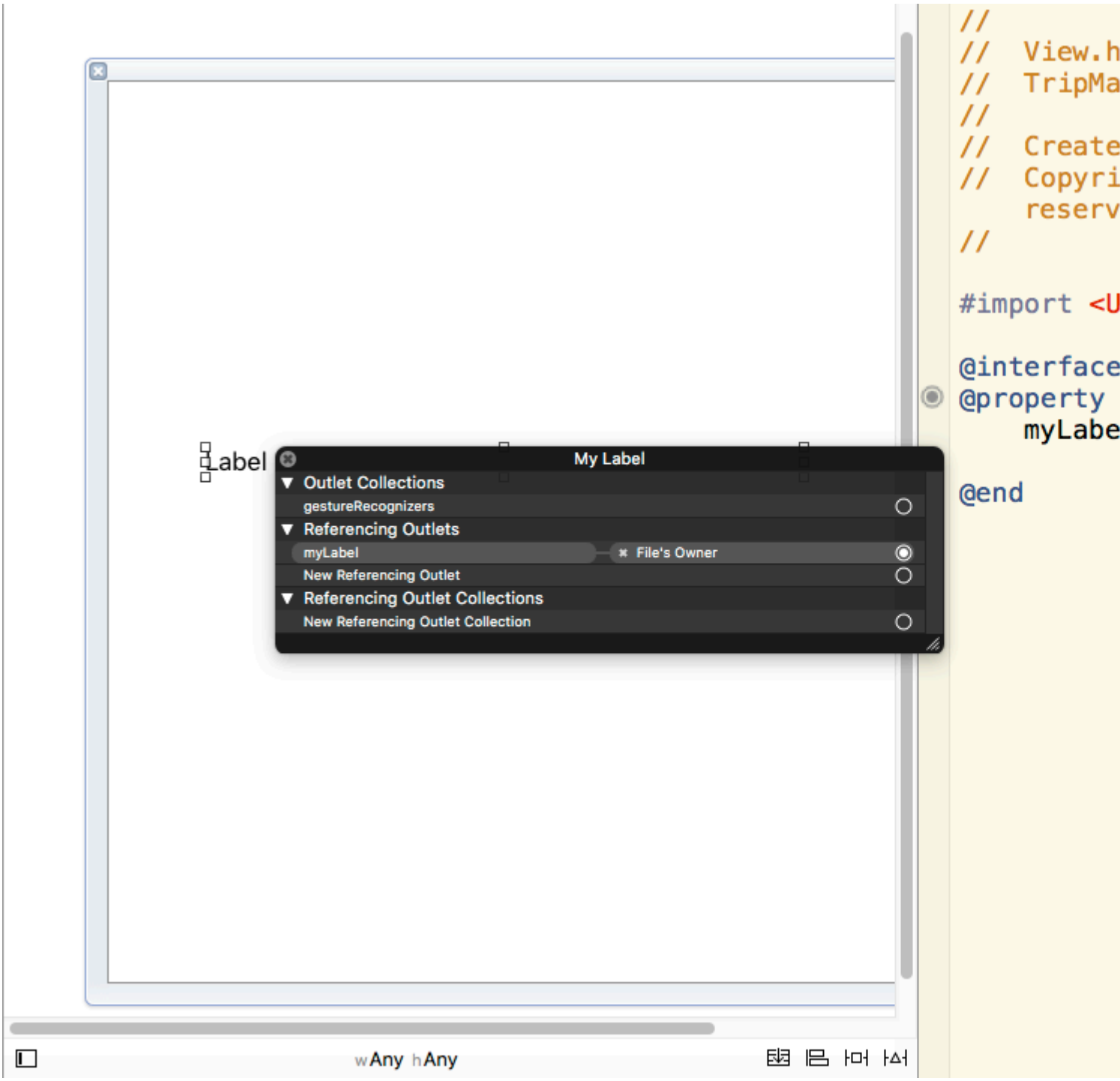
Non si tratta di un'operazione complessa, ma è necessario seguire i passaggi precisi per farlo correttamente la prima volta, evitando le eccezioni.

Come funziona loadNibNamed Works

I passaggi principali sono:

1. Crea XIB
2. Crea classe .h e .m
3. Definisci punti vendita in .h
4. Collegare le prese tra .h e XIB

Vedi screenshot allegato:



5. Richiamare loadNibNamed all'interno della funzione initWithCoder del file .m. Questo è necessario per assicurare che sia possibile posizionare direttamente l'oggetto UIView nello storyboard / file UIView padre XIB e definirlo come vista personalizzata. Nessun altro codice di inizializzazione è necessario una volta caricato lo storyboard / genitore XIB. La tua vista personalizzata può essere aggiunta ad altre viste proprio come gli altri oggetti di visualizzazione Objective C incorporati forniti in XCode.

Leggi Panoramiche personalizzate dai file XIB online:
https://riptutorial.com/it/ios/topic/1362/panoramiche-personalizzate-dai-file-xib

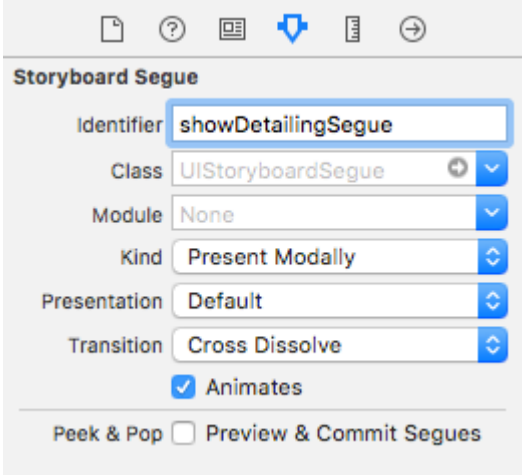
Capitolo 123: Passaggio dei dati tra i controller di visualizzazione

Examples

Utilizzo di Segues (passaggio dei dati in avanti)

Per passare i dati dal controller della vista corrente al nuovo controller della vista successivo (non un precedente controller della vista) usando segues, prima creare un seguito con un identificatore nello storyboard pertinente. Sostituisci il metodo `prepareForSegue` del tuo attuale controller di `prepareForSegue` . All'interno del metodo controlla il seguito appena creato dal suo identificatore. Trasmetti il controller di visualizzazione di destinazione e passa i dati ad esso impostando le proprietà sul controller di visualizzazione downcast.

Impostazione di un identificatore per un seguito:



Le sequenze possono essere eseguite in modo programmatico o utilizzando l'evento dell'azione del pulsante impostato nello storyboard da ctrl + trascinamento al controller della vista di destinazione. È possibile chiamare un passaggio in modo programmatico, quando necessario, utilizzando l'identificatore dei passaggi nel controllore della vista:

Objective-C

```
- (void)showDetail {
    [self performSegueWithIdentifier:@"showDetailingSegue" sender:self];
}
```

veloce

```
func showDetail() {
    self.performSegue(withIdentifier: "showDetailingSegue", sender: self)
}
```

È possibile configurare il payload dei passaggi nella versione sovrascritta del metodo `prepareForSegue` . È possibile impostare le proprietà richieste prima che il controller della vista di destinazione sia caricato.

Objective-C

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if([segue.identifier isEqualToString:@"showDetailingSegue"]){
        DetailViewController *controller = (DetailViewController
*)segue.destinationViewController;
        controller.isDetailingEnabled = YES;
    }
}
```

veloce

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "showDetailingSegue" {
        let controller = segue.destinationViewController as! DetailViewController
        controller.isDetailingEnabled = true
    }
}
```

`DetailViewController` è il nome del secondo controller di visualizzazione e `isDetailingEnabled` è una variabile pubblica in tale controller di visualizzazione.

Per espandere questo modello, puoi trattare un metodo pubblico su `DetailViewController` come pseudo inizializzatore, per aiutare a inizializzare le variabili richieste. Questo documenterà automaticamente le variabili che devono essere impostate su `DetailViewController` senza dover leggere il suo codice sorgente. È anche un posto comodo per mettere le impostazioni predefinite.

Objective-C

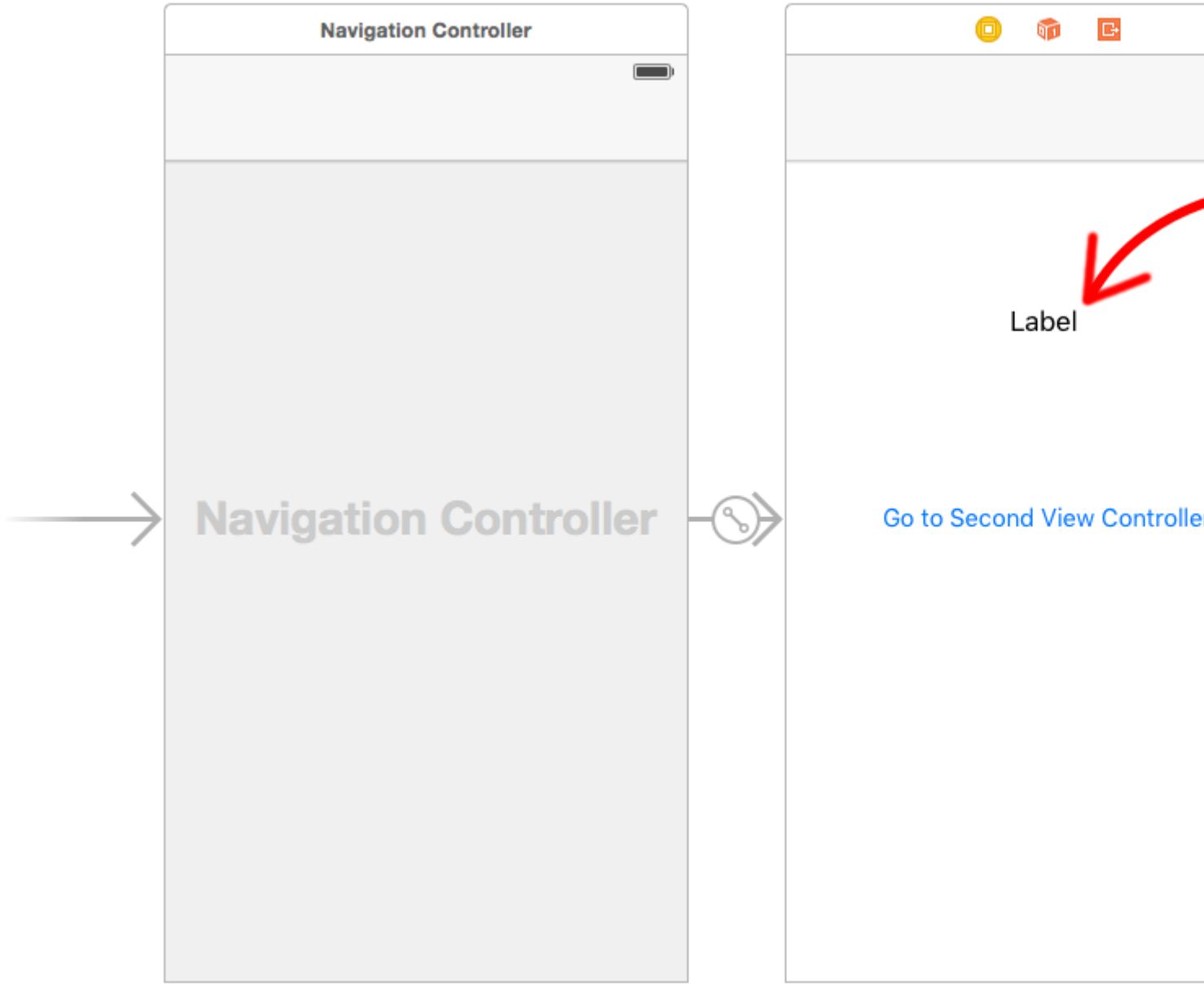
```
- (void)initVC:(BOOL *)isDetailingEnabled {
    self.isDetailingEnabled = isDetailingEnabled
}
```

veloce

```
func initVC(isDetailingEnabled: Bool) {
    self.isDetailingEnabled = isDetailingEnabled
}
```

Uso del pattern Delegate (passaggio di dati indietro)

Per trasferire i dati dal controller di visualizzazione corrente al controller di visualizzazione precedente, è possibile utilizzare il modello delegato.



In questo esempio si presuppone che sia stato eseguito un seguito in Interface Builder e che si sia impostato l'identificatore di `showSecondViewController` per `showSecondViewController` . Le uscite e le azioni devono anche essere collegate ai nomi nel seguente codice.

First View Controller

Il codice per il controller First View è

veloce

```
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
```

Nota l'uso del nostro protocollo personalizzato `DataEnteredDelegate` .

Secondo View Controller and Protocol

Il codice per il secondo controller di vista è

veloce

```
// 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
```

Si noti che il `protocol` è al di fuori della classe View Controller.

Passare i dati all'indietro usando lo svolgimento ai seguiti

A differenza di seguito che consente di passare i dati "in avanti" dal controller di visualizzazione corrente al controller di visualizzazione di destinazione:

(VC1) **->** *(VC2)*

Usando "unwind" puoi fare il contrario, passare i dati dalla destinazione o dal controller della vista corrente al suo controller di visualizzazione presentando:

(VC1) **<-** *(VC2)*

NOTA : Prestare attenzione al fatto che l'utilizzo di unwind consente di passare prima i dati e successivamente il controller di visualizzazione corrente (VC2) verrà deallocato.

Ecco come farlo:

Innanzitutto, è necessario aggiungere la seguente dichiarazione al controller della vista di presentazione (VC1) che è il controller della vista che vogliamo trasmettere i dati a:

```
@IBAction func unwindToPresentingViewController(segue:UIStoryboardSegue)
```

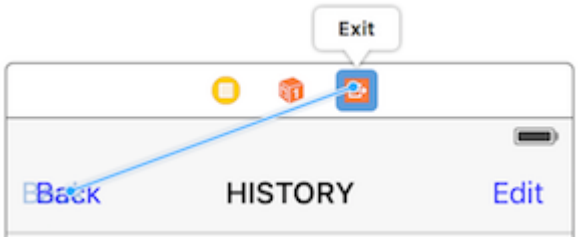
L'importante è usare il prefisso `unwind` , questo "informa" Xcode che questo è un metodo di svolgimento che ti dà la possibilità di usarlo anche nello storyboard.

Successivamente dovrai implementare il metodo, sembra quasi lo stesso di un seguito reale:

```
@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
        }
```

Ora hai 2 opzioni per invocare le chiamate a piacimento:

1. È possibile "hard code" richiamare: `self.performSegueWithIdentifier("YourCustomIdentifier", sender: self)` che eseguirà lo `performSegueWithIdentifier` per te ogni volta che `performSegueWithIdentifier` .
2. Puoi collegare il metodo di svolgimento utilizzando lo `storyboard` all'oggetto "Esci": ctrl + trascina il pulsante che vuoi richiamare il metodo di svolgimento, sull'oggetto "Esci":



Rilascia e avrai la possibilità di scegliere il tuo metodo di svolgimento personalizzato:

Trasmissione dei dati tramite le chiusure (trasmissione dei dati indietro)

Invece di usare il **pattern delegato** , che ha diviso l'implementazione in varie parti della classe `UIViewController` , è possibile utilizzare anche le `closures` per passare i dati indietro e avanti. Supponendo che tu stia utilizzando `UIStoryboardSegue` , nel metodo `prepareForSegue` puoi facilmente configurare il nuovo controller in un solo passaggio

```
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)
        }
    }
}
```

Questo è un esempio di utilizzo ed è meglio usare su Swift, la sintassi del blocco Objective-C non è così facile da rendere il codice più leggibile

Usando la chiusura (blocco) di richiamata che restituisce i dati

questo argomento è un problema classico nello sviluppo di iOS e la sua soluzione è diversa come già mostrato in altri esempi. In questo esempio mostrerò un altro uso quotidiano comune: passare i dati usando la `closure` adattando l'esempio di `delegate pattern` su questa pagina alla `closure` callback!

una cosa che questo metodo è superiore a `delegate pattern` è invece di dividere il codice di settaggio in due posti diversi (guarda l'esempio di delegato in questa pagina, `prepareForSegue` , `userDidEnterInformation`) piuttosto raccogliendoli insieme (solo in `prepareForSegue` , lo mostrerò)

Inizia da Second View Controller

dobbiamo capire come usare il callback, quindi possiamo scriverlo, questo è il motivo per cui partiamo dal secondo controller di vista poiché è dove usiamo il callback: quando abbiamo ottenuto il nuovo input di testo, chiamiamo il nostro callback, **usando** come **parametro il parametro callback** per passare i dati al primo ViewController, notare che ho detto usando il parametro callback, questo è molto importante, i novizi (come lo ero io) lo ignorano sempre e non sanno da dove iniziare a scrivere correttamente la callback

quindi in questo caso, sappiamo che il nostro callback accetta solo un parametro: il testo e il suo tipo è `String` , dichiariamolo e rendiamolo proprietà dal momento che abbiamo bisogno di compilare dal nostro primo controller di visualizzazione

Devo solo commentare tutta la parte del `delegate` e tenerla per il confronto

```
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)
    }
}
```

Termina il primo controller di visualizzazione

tutto quello che devi fare è passare la chiusura del callback, e abbiamo finito, la chiusura farà il lavoro futuro per noi dal momento che l'abbiamo già configurata in seconda visione

guarda come riduce il nostro codice rispetto al `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
    //}
}
```

e nell'ultimo, forse qualcuno di voi sarà confuso dal fatto che stiamo passando solo i dati (chiusura in questo caso) solo in un modo, dal primo controller di visualizzazione al secondo, non direttamente dal controller di seconda visione, come possiamo considerarlo come uno strumento di comunicazione? forse dovresti davvero eseguirlo e provarlo tu stesso, tutto quello che dirò è il **parametro** , il **parametro** di **callback closure** che restituisce i dati indietro!

Assegnando una proprietà (Passa dati in avanti)

È possibile passare i dati direttamente assegnando la proprietà del controller di visualizzazione successivo prima di inviarlo o inviarlo.

```
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)")
        }
    }
}
```

Leggi Passaggio dei dati tra i controller di visualizzazione online:
https://riptutorial.com/it/ios/topic/434/passaggio-dei-dati-tra-i-controller-di-visualizzazione

Capitolo 124: Passaggio dei dati tra i controller di visualizzazione (con MessageBox-Concept)

introduzione

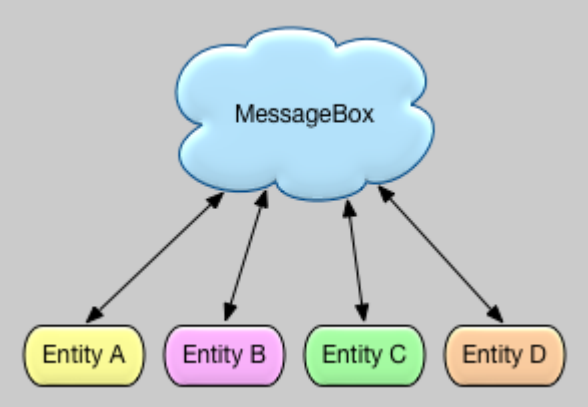
MessageBox è un concetto semplice per il disaccoppiamento delle entità.

Ad esempio, l'entità A può inserire un messaggio che l'entità B può leggere quando è opportuno.

Un controller di visualizzazione vorrebbe parlare con un altro controller di visualizzazione, ma non si desidera creare una relazione forte o debole.

Examples

Semplice esempio di utilizzo




```
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)
```

Leggi Passaggio dei dati tra i controller di visualizzazione (con MessageBox-Concept) online: https://riptutorial.com/it/ios/topic/9118/passaggio-dei-dati-tra-i-controller-di-visualizzazione--con-messagebox-concept-

Capitolo 125: plist iOS

introduzione

Plist viene utilizzato per la memorizzazione dei dati nell'app iOS. Plist salva i dati sotto forma di array e dizionari. In plist possiamo salvare i dati come: 1. Dati statici da utilizzare nell'app. 2. Dati che arriveranno dal server.

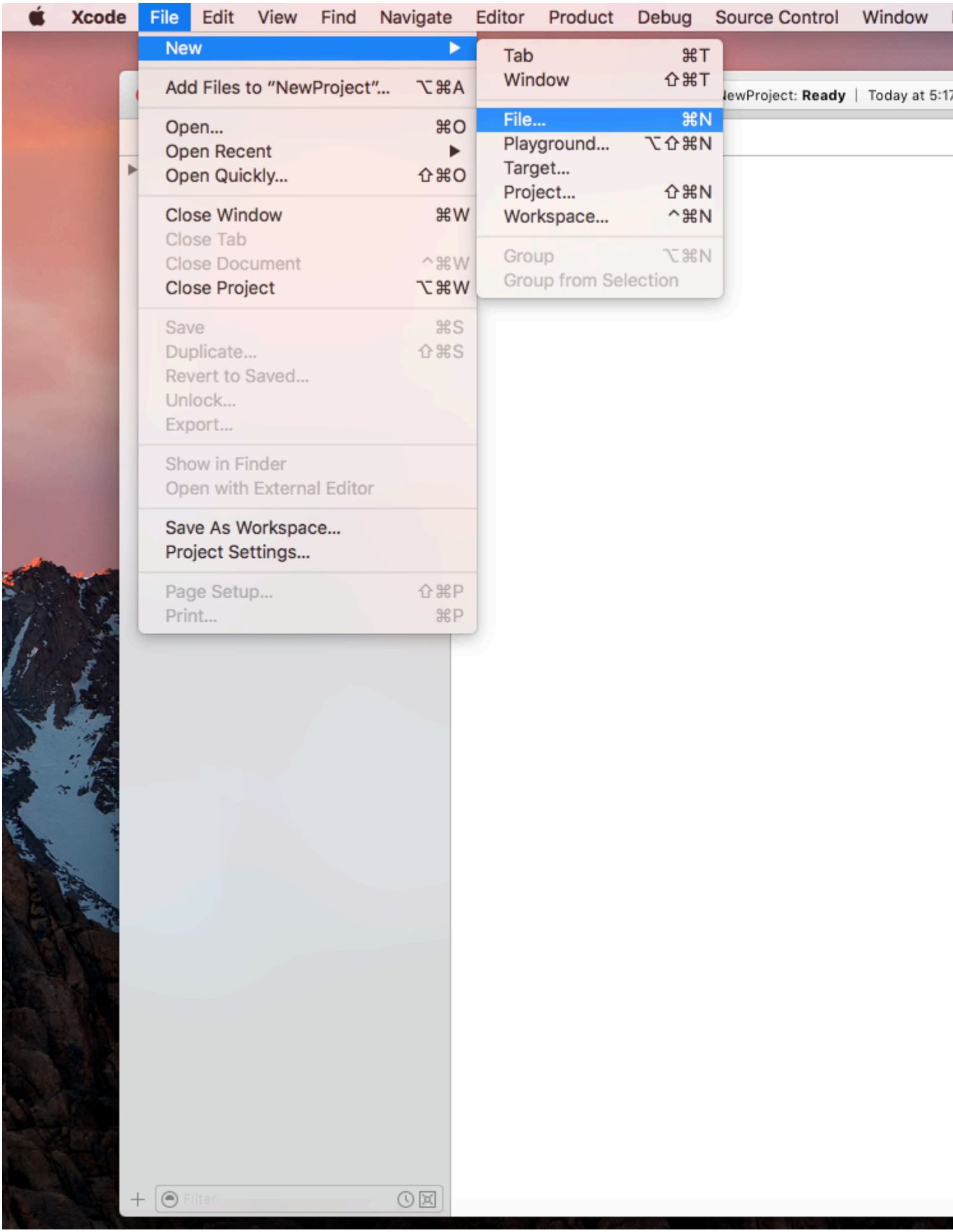
Examples

Esempio:

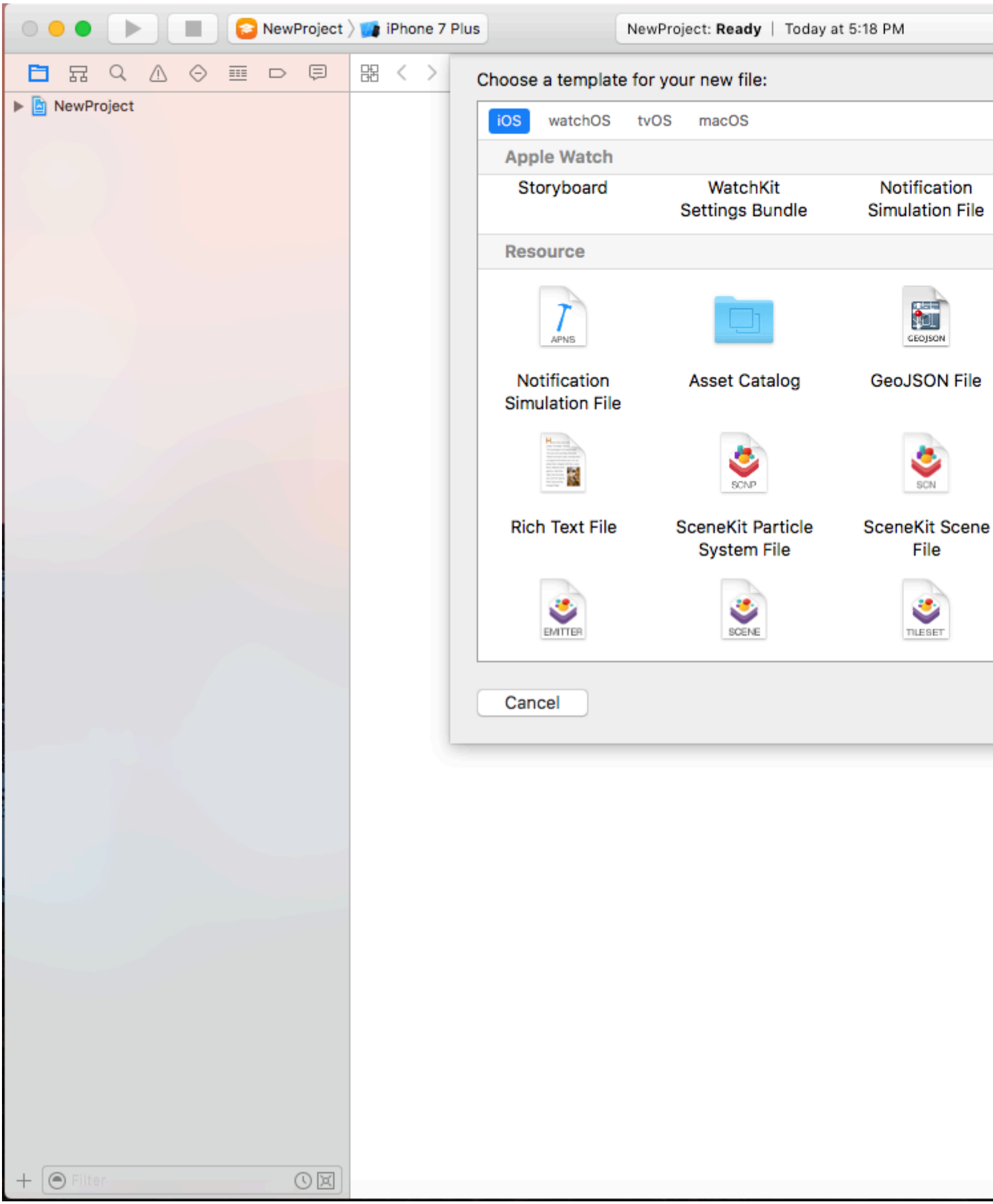
1. Dati statici da utilizzare nell'app.

Per salvare i dati statici in plist segui questi metodi:

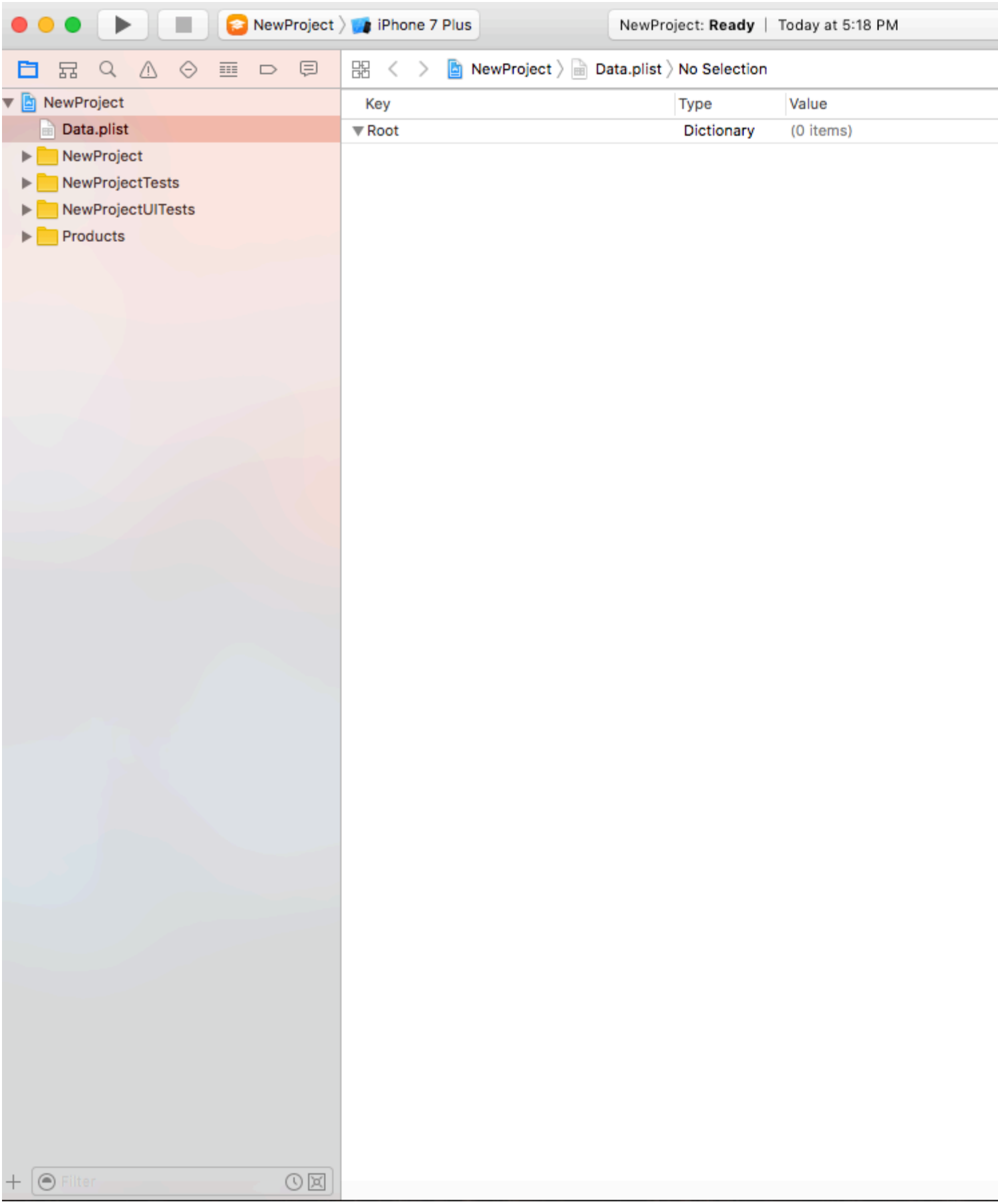
a) Aggiungi un nuovo file


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

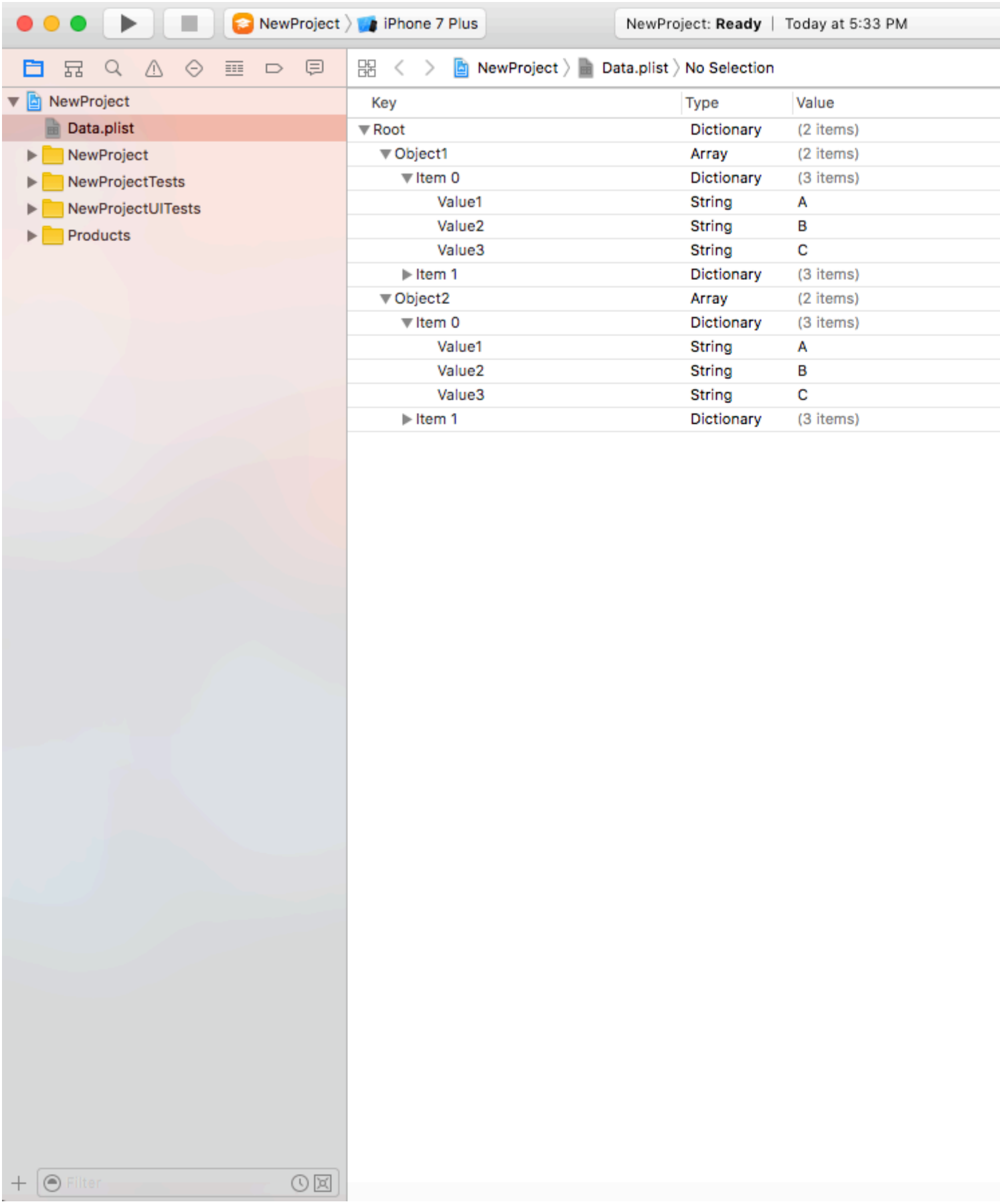
b) Fare clic su Elenco proprietà in Risorse



c) Nome della lista di proprietà e un file verrà creato come (data.plist qui)



d) È possibile creare un plist di array e dizionari come:



// Leggi plist da bundle e ottieni Root Dictionary

```
NSDictionary *dictRoot = [NSDictionary dictionaryWithContentsOfFile:[[NSBundle mainBundle]
pathForResource:@"Data" ofType:@"plist"]];
```

// Il tuo dizionario contiene una serie di dizionari // Ora estrai una matrice da essa.

```
NSArray *arrayList = [NSArray arrayWithArray:[dictRoot objectForKey:@"Object1"]];

for(int i=0; i< [arrayList count]; i++)
{
    NSMutableDictionary *details=[arrayList objectAtIndex:i];
}
```

Salva e modifica / cancella i dati da Plist

Hai già creato un plist. Questo plist rimarrà lo stesso in app. Se si desidera modificare i dati in questo plist, aggiungere nuovi dati in plist o rimuovere i dati da Plist, non è possibile apportare modifiche in questo file.

A tale scopo dovrai archiviare i tuoi plist in Document Directory. È possibile modificare il plist salvato nella directory del documento.

Salva plist nella directory del documento come:

```
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
{

}
```

Riscrivi i dati da Plist come:

```
    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"];
```

È possibile modificare rimuovere, aggiungere nuovi dati secondo le proprie esigenze e salvare nuovamente il plist in Document Directory.

Leggi plist iOS online: https://riptutorial.com/it/ios/topic/8141/plist-ios

Capitolo 126: Portachiavi

Sintassi

- kSecClassGenericPassword // Una chiave di valore che rappresenta una password non Internet
- kSecClassInternetPassword // Una chiave di valore che rappresenta una password internet
- kSecClassCertificate // Una chiave di valore che rappresenta un certificato
- kSecClassCertificate // Una chiave di valore che rappresenta una chiave
- kSecClassIdentity // Una chiave di valore che rappresenta un'identità, che è un certificato più una chiave

Osservazioni

iOS memorizza informazioni private come password, chiavi di crittografia, certificati e identità in un'area di archiviazione protetta chiamata Keychain. Questa area di memoria è gestita completamente da un co-processore chiamato Secure Enclave, che è incorporato all'interno del processore dell'applicazione. Poiché il portachiavi è in modalità sandbox su iOS, gli elementi portachiavi possono essere recuperati solo dall'applicazione che li ha messi lì in primo luogo.

In alcuni casi è necessario attivare Condivisione dei portachiavi nelle funzionalità Xcode per evitare errori.

Per interagire con il portachiavi, utilizziamo il framework ac chiamato Keychain Services. Per ulteriori informazioni, consulta la Guida alla programmazione dei servizi Keychain di Apple .

Poiché i servizi portachiavi sono al di sotto del livello `Foundation` , è limitato all'utilizzo di tipi `CoreFoundation` . Di conseguenza, la maggior parte degli oggetti sono internamente rappresentati come `CFDictionary` che `CFString` s come le loro chiavi e una varietà di tipi `CoreFoundation` come i loro valori.

Mentre i Keychain Services sono inclusi come parte del framework `Security` , l'importazione di `Foundation` è di solito una buona opzione poiché include alcune funzioni di supporto nel back-end.

Inoltre, se non vuoi gestire direttamente i Keychain Services, Apple fornisce il progetto di esempio Generic Keychain Swift che fornisce i tipi Swift che utilizzano i servizi Keychain dietro le quinte.

Examples

Aggiunta di una password al portachiavi

Ogni elemento portachiavi è spesso rappresentato come un `CFDictionary` . Tuttavia, è possibile utilizzare semplicemente `NSDictionary` in Objective-C e sfruttare il bridging, oppure in Swift è possibile utilizzare `Dictionary` e `CFDictionary` espressamente a `CFDictionary` .

È possibile costruire una password con il seguente dizionario:

veloce

```
var dict = [String : AnyObject]()
```

Innanzitutto, è necessaria una coppia chiave / valore che permetta al Portachiavi di sapere che questa è una password. Si noti che poiché la nostra chiave dict è una `String` è necessario `CFString` qualsiasi `CFString` a una `String` esplicito in Swift 3. CFString non può essere utilizzato come chiave per un dizionario Swift perché non è selezionabile.

veloce

```
dict[kSecClass as String] = kSecClassGenericPassword
```

Successivamente, la nostra password potrebbe avere una serie di attributi per descriverla e aiutarci a trovarla in seguito. Ecco una lista di attributi per le password generiche .

veloce

```
// 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
```

Infine, abbiamo bisogno dei nostri dati privati reali. Assicurati di non tenerlo in memoria per troppo tempo. Questo deve essere `CFData` .

veloce

```
dict[kSecValueData as String] = "my_password!!".data(using: .utf8) as! CFData
```

Infine, la funzione di aggiunta dei servizi Keychain vuole sapere come deve restituire l'elemento portachiavi di nuova costruzione. Dal momento che non dovresti tenere i dati molto a lungo nella memoria, ecco come puoi restituire gli attributi:

veloce

```
dict[kSecReturnAttributes as String] = kCFBooleanTrue
```

Ora abbiamo costruito il nostro oggetto. Aggiungiamolo:

veloce

```
var result: AnyObject?
let status = withUnsafeMutablePointer(to: &result) {
    SecItemAdd(dict as CFDictionary, UnsafeMutablePointer($0))
}
let newAttributes = result as! Dictionary<String, AnyObject>
```

Questo pone i nuovi attributi dettati all'interno del `result` . `SecItemAdd` prende il dizionario che abbiamo costruito e un puntatore a dove vorremmo il nostro risultato. La funzione restituisce quindi un `OSStatus` indica il successo o un codice di errore. I codici dei risultati sono descritti qui .

Trovare una password nel portachiavi

Per costruire una query, dobbiamo rappresentarla come un `CFDictionary` . Puoi anche usare `NSDictionary` in Objective-C o `Dictionary` in Swift e `CFDictionary` a `CFDictionary` .

Abbiamo bisogno di una chiave di classe:

veloce

```
var dict = [String : AnyObject]()
dict[kSecClass as String] = kSecClassGenericPassword
```

Successivamente, possiamo specificare gli attributi per restringere la nostra ricerca:

veloce

```
// 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
```

Possiamo anche specificare i tasti di modifica di ricerca speciali qui descritti.

Infine, dobbiamo dire come vorremmo che i nostri dati fossero restituiti. Di seguito, ti chiederemo di restituire solo la password privata come oggetto `CFData` :

veloce

```
dict[kSecReturnData as String] = kCFBooleanTrue
```

Ora, cerchiamo:

veloce

```
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)!
```

Qui, `SecItemCopyMatching` un dizionario di query e un puntatore a cui desideri ottenere il risultato. Restituisce un `OSStatus` con un codice di risultato. Ecco le possibilità.

Aggiornamento di una password nel portachiavi

Come al solito, abbiamo prima bisogno di un `CFDictionary` per rappresentare l'elemento che vogliamo aggiornare. Questo deve contenere tutti i vecchi valori per l'elemento, inclusi i vecchi dati privati. Quindi richiede un `CFDictionary` di qualsiasi attributo o dei dati stessi che si desidera modificare.

Quindi, per prima cosa, costruiamo una chiave di classe e una lista di attributi. Questi attributi possono restringere la nostra ricerca ma devi includere tutti gli attributi e i vecchi valori se li cambierai.

veloce

```
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
```

Ora dobbiamo aggiungere i vecchi dati:

veloce

```
dict[kSecValueData as String] = "my_password!!".data(using: .utf8) as! CFData
```

Ora creiamo gli stessi attributi ma una password diversa:

veloce

```
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
```

Ora, passiamo semplicemente ai servizi portachiavi:

veloce

```
let status = SecItemUpdate(dict as CFDictionary, newDict as CFDictionary)
```

`SecItemUpdate` restituisce un codice di stato. I risultati sono descritti qui .

Rimozione di una password dal portachiavi

Abbiamo bisogno solo di una cosa per eliminare un oggetto dal Portachiavi: un `CFDictionary` con attributi che descrivono gli elementi da eliminare. Tutti gli elementi che corrispondono al dizionario di query verranno eliminati in modo permanente, quindi se si intende eliminare un singolo elemento assicurarsi di essere specifici con la query. Come sempre, possiamo usare un `NSDictionary` in Objective-C o in Swift possiamo usare un `Dictionary` e quindi `CFDictionary` a `CFDictionary` .

Un dizionario di query, in questo contesto include esclusivamente una chiave di classe per descrivere l'elemento e gli attributi per descrivere le informazioni sull'elemento. L'inclusione di restrizioni di ricerca come `kSecMatchCaseInsensitive` non è consentita.

veloce

```
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
```

E ora possiamo semplicemente rimuoverlo:

veloce

```
let status = SecItemDelete(dict as CFDictionary)
```

`SecItemDelete` restituisce un `OSStatus` . I codici dei risultati sono descritti qui .

Portachiavi Aggiungi, Aggiorna, Rimuovi e Trova operazioni utilizzando un solo file.

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
```

Controllo accesso portachiavi (TouchID con fallback password)

Il portachiavi consente di salvare gli articoli con l'attributo speciale SecAccessControl che consente di ottenere l'elemento dal Portachiavi solo dopo che l'utente sarà autenticato con Touch ID (o passcode se tale fallback è consentito). L'app viene notificata solo se l'autenticazione ha avuto successo o meno, l'intera UI è gestita da iOS.

In primo luogo, l'oggetto SecAccessControl deve essere creato:

veloce

```
let error: Unmanaged<CFError>?
```

```
guard let accessControl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly, .userPresence, &error) else {
    fatalError("Something went wrong")
}
```

Successivamente, aggiungilo al dizionario con la chiave kSecAttrAccessControl (che si esclude a vicenda con la chiave kSecAttrAccessible che hai utilizzato in altri esempi):

veloce

```
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
```

E salvalo come hai fatto prima:

veloce

```
let lastResultCode = SecItemAdd(query as CFDictionary, nil)
```

Per accedere ai dati memorizzati, basta interrogare Keychain per una chiave. I servizi portachiavi presenteranno la finestra di autenticazione all'utente e restituiranno i dati o nil a seconda che sia stata fornita un'appropriata impronta digitale o che sia stato fornito il codice di accesso.

Facoltativamente, è possibile specificare una stringa di richiesta:

veloce

```
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))
}
```

Fai attenzione che lo `status` sarà `err` se l'utente rifiuta, ha cancellato o ha fallito l'autorizzazione.

veloce

```
if status == noErr {
    let password = String(data: queryResult as! Data, encoding: .utf8)!
    print("Password: \(password)")
} else {
    print("Authorization not passed")
}
```

Leggi Portachiavi online: https://riptutorial.com/it/ios/topic/6839/portachiavi

Capitolo 127: Processo di invio di app

introduzione

Questo tutorial copre tutti i passaggi necessari per caricare un'app iOS sull'App Store.

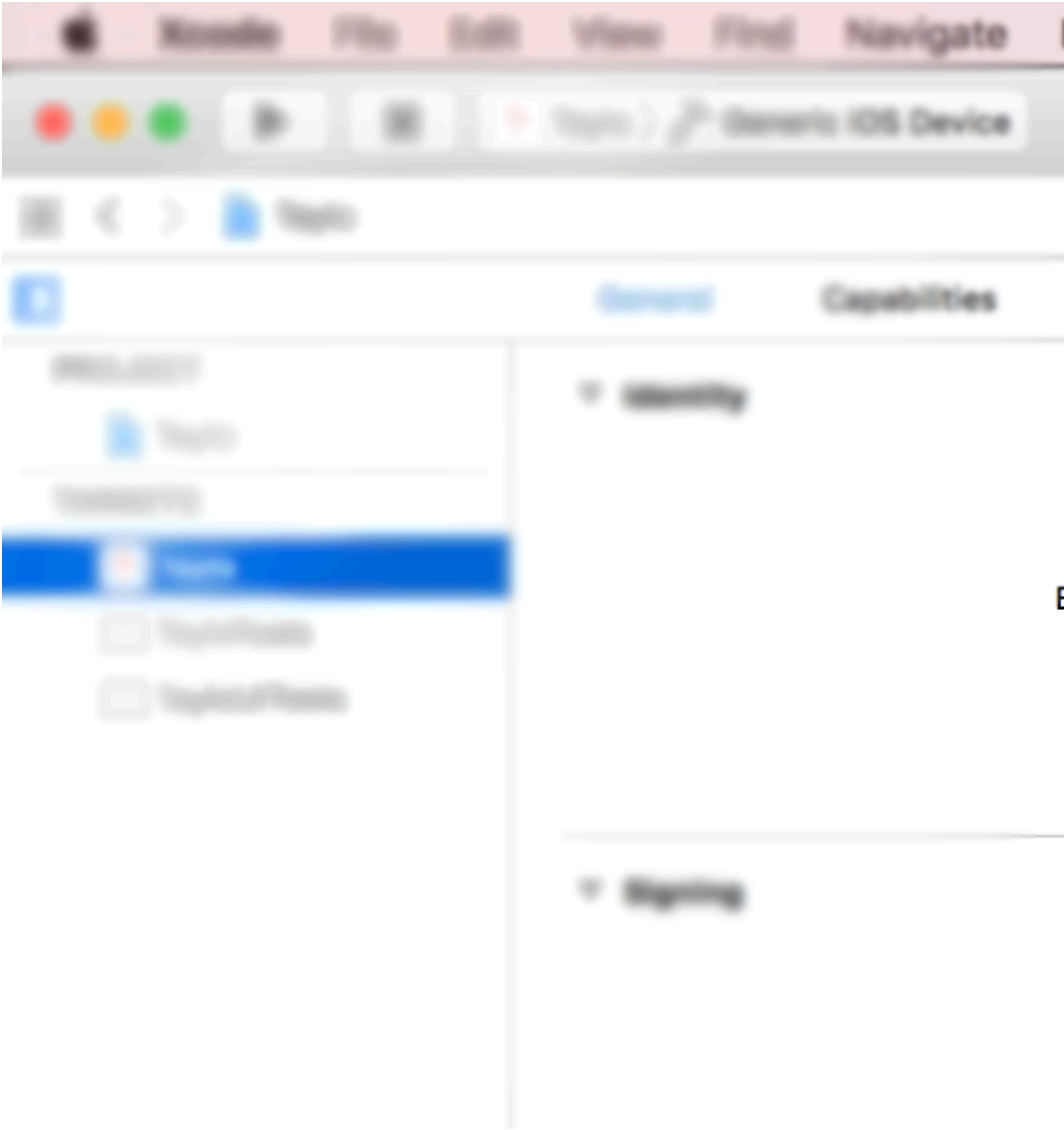
Examples

Impostare i profili di provisioning

Nelle versioni precedenti, l'impostazione dei profili di provisioning è stata eseguita manualmente. Generare il profilo di distribuzione, scaricarlo e quindi distribuire l'app. Questo doveva essere fatto per ogni macchina di sviluppo che richiedeva molto tempo. Tuttavia, nella maggior parte dei casi al giorno d'oggi, Xcode 8 farà la maggior parte di questo lavoro per te. Assicurati di accedere con l'account che verrà utilizzato per la distribuzione dell'app e quindi seleziona semplicemente "Gestisci automaticamente la firma del codice" in Obiettivi -> Generale.

Archivia il codice

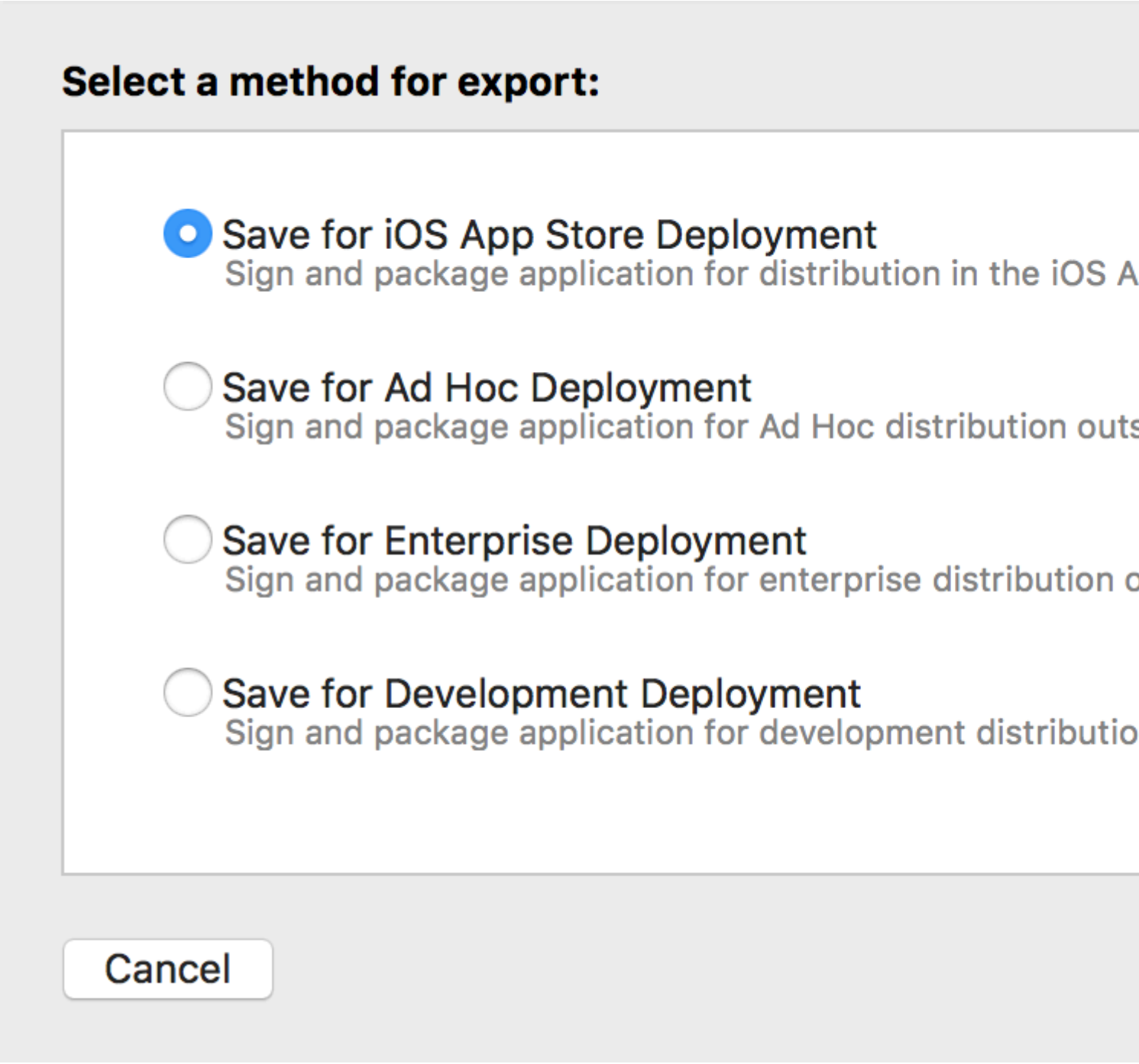
Una volta impostati i profili di provisioning, il prossimo passo nel processo di invio dell'app è archiviare il codice. Dal menu a discesa di dispositivi e simulatori seleziona l'opzione "Dispositivo iOS generico". Quindi, nel menu "Prodotto" selezionare l'opzione "Archivio".



Nel caso in cui l'invio sia un aggiornamento dell'app esistente nello store, assicurarsi che il numero di build sia superiore a quello corrente e che il numero di versione sia diverso. Ad esempio, l'applicazione corrente ha il numero di build 30 e l'etichetta di versione 1.0. Il prossimo aggiornamento dovrebbe avere almeno il numero di build 31 e l'etichetta di versione 1.0.1. Nella maggior parte dei casi, è necessario aggiungere un terzo decimale alla propria versione in caso di correzioni urgenti di bug o piccole patch, il secondo decimale è principalmente riservato agli aggiornamenti delle funzionalità mentre il primo decimale viene incrementato in caso di un importante aggiornamento dell'app.

Esporta file IPA

Una volta terminato, puoi trovare il tuo archivio nell'organizer Xcode. È qui che tutte le versioni precedenti e le build di archivio vengono salvate e organizzate nel caso in cui non le elimini. Noterai immediatamente un grande pulsante blu che dice "Carica su App Store ..." tuttavia in 9/10 casi questo non funzionerà a causa di vari motivi (principalmente i bachi Xcode). Soluzione alternativa è di esportare il tuo archivio e caricarlo utilizzando un altro strumento Xcode chiamato Application Loader. Tuttavia, poiché il caricatore applicazioni carica i file IPA sull'App Store, l'archivio deve essere esportato nel formato corretto. Questo è un compito banale che potrebbe richiedere circa mezz'ora. Fai clic sul pulsante "Esporta" nel pannello laterale destro.

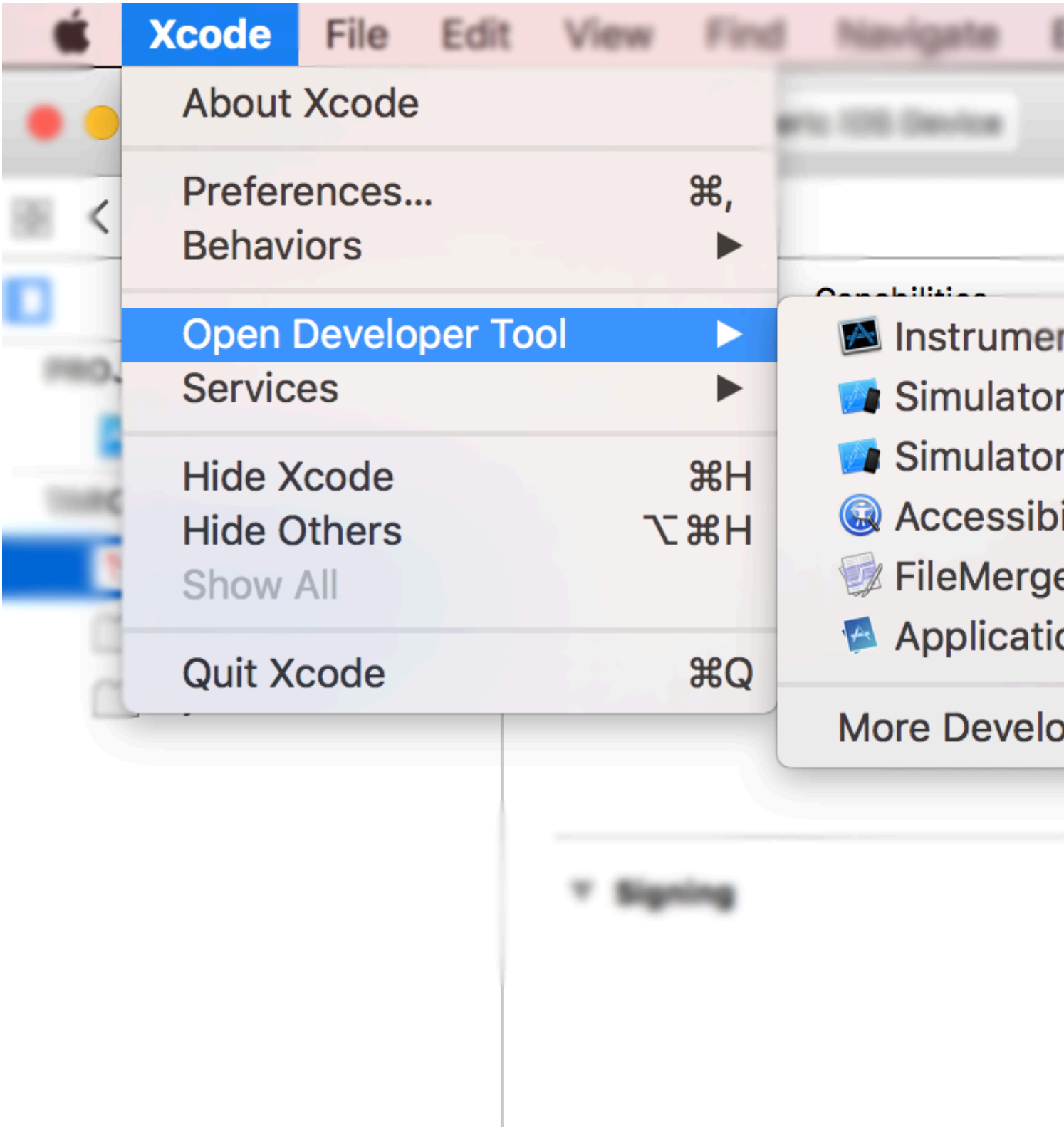


Se stai caricando un'app in App Store, seleziona la prima opzione e fai clic su Avanti. Accedi e convalida il tuo codice ancora una volta e prendi una tazza di caffè. Una volta terminato il

processo di esportazione, ti verrà chiesto dove salvare il file IPA generato. Di solito, il desktop è la scelta più conveniente.

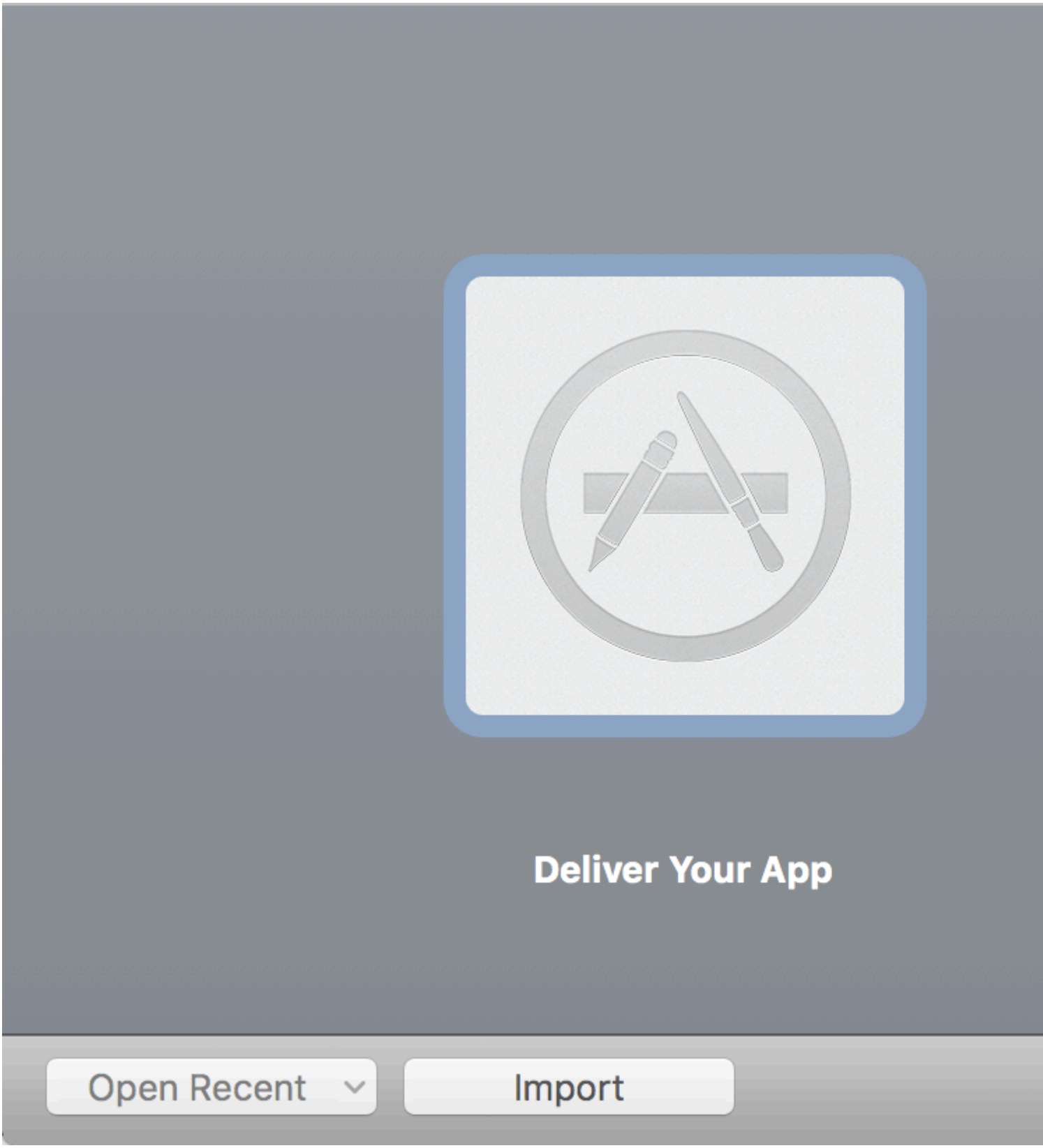
Carica il file IPA usando Application Loader

Una volta generato il file IPA, apri Xcode, vai agli strumenti di sviluppo e apri Application Loader.



Se hai più account nel tuo Xcode, ti verrà chiesto di scegliere. Scegli naturalmente quello che hai utilizzato per la firma del codice nel primo passaggio. Scegli "Consegna la tua app" e carica il

codice. Una volta completato il caricamento, potrebbe essere necessaria fino a un'ora per essere visualizzato nell'elenco di build su iTunes Connect.



Leggi Processo di invio di app online: https://riptutorial.com/it/ios/topic/8765/processo-di-invio-di-app

Capitolo 128: Profilo con strumenti

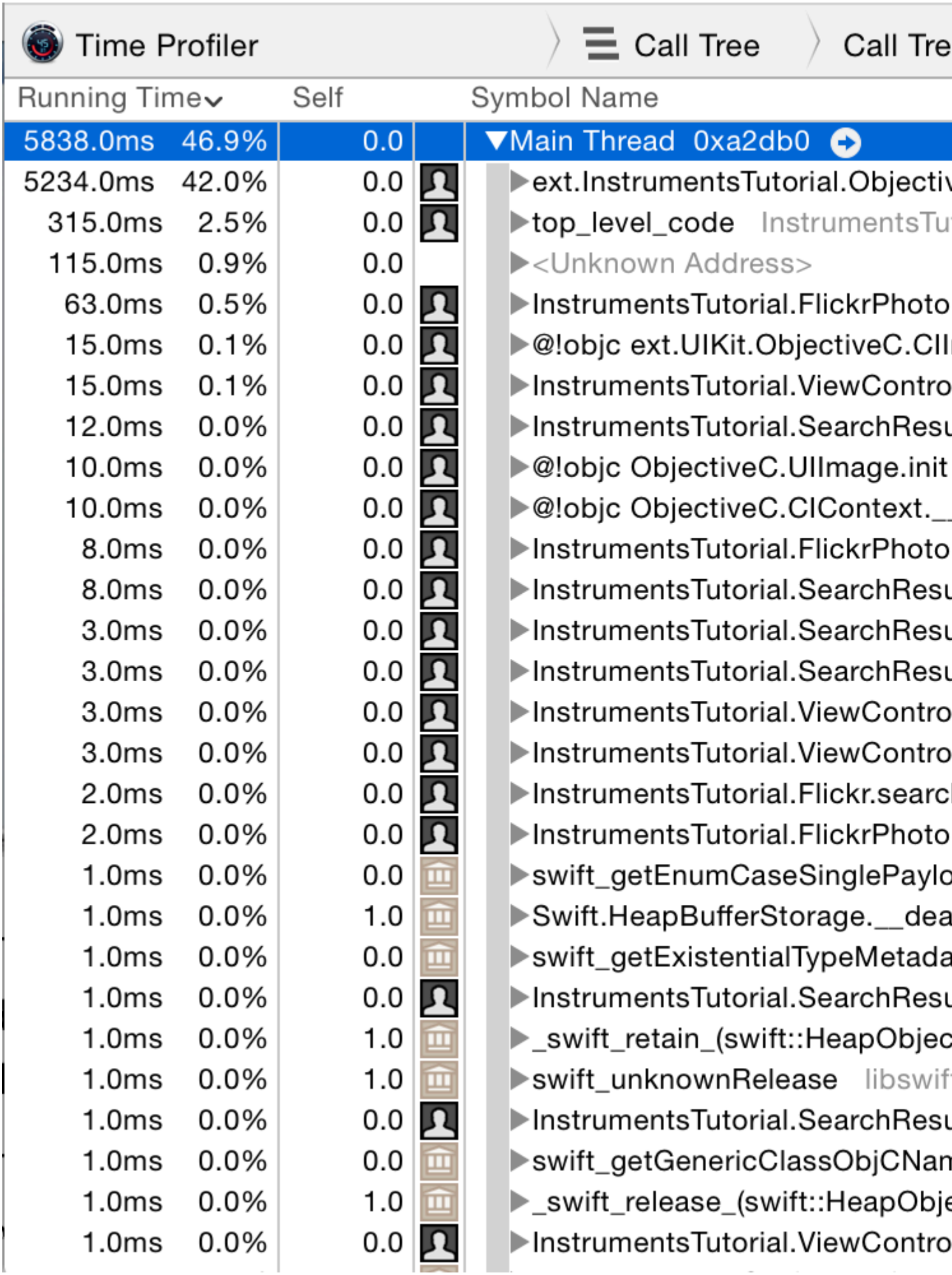
introduzione

Xcode include un'applicazione di ottimizzazione delle prestazioni denominata Instruments che è possibile utilizzare per profilare l'applicazione utilizzando tutti i tipi di metriche differenti. Hanno strumenti per controllare l'utilizzo della CPU, l'utilizzo della memoria, le perdite, l'attività di file / rete e l'utilizzo di energia, solo per citarne alcuni. È davvero facile iniziare a profilare la tua app da Xcode, ma a volte non è così facile capire cosa vedi quando si tratta di una profilazione, il che dissuade alcuni sviluppatori dall'essere in grado di utilizzare questo strumento al massimo potenziale.

Examples

Time Profiler

Il primo strumento che vedrai è il `Time Profiler` . A intervalli misurati, Instruments interromperà l'esecuzione del programma e prenderà una traccia di stack su ogni thread in esecuzione. Pensa a ciò premendo il pulsante di pausa nel debugger di Xcode. Ecco un'anteprima del Time Profiler: -



Questa schermata visualizza l'`Call Tree`. L'`Call Tree`

mostra la quantità di tempo trascorso nell'esecuzione in vari metodi all'interno di un'app. Ogni riga è un metodo diverso seguito dal percorso di esecuzione del programma. Il tempo trascorso in ciascun metodo può essere determinato dal numero di volte in cui il profiler viene fermato in ogni metodo. Ad esempio, se vengono eseguiti **100** campioni ad **intervalli di 1 millisecondo** e si trova un particolare metodo in cima allo stack in 10 campioni, è possibile dedurre che è stato impiegato circa il **10%** del tempo di esecuzione totale - **10 millisecondi** in quel metodo. È un'approssimazione abbastanza approssimativa, ma funziona!

Dalla barra dei menu `Xcode's` , selezionare `Product\Profile` , o `press ⌘I` . Questo costruirà l'app e avvierà gli strumenti. Sarai accolto con una finestra di selezione simile a questa:

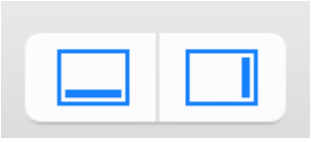
Questi sono tutti modelli diversi forniti con gli strumenti.

Seleziona lo strumento `Time Profiler` e fai clic su Scegli. Questo aprirà un nuovo documento di strumenti. Fai clic sul **pulsante** rosso di **registrazione** in alto a sinistra per avviare la registrazione e avviare l'app. È possibile che venga chiesta la password per autorizzare gli **strumenti** ad analizzare altri processi: non temere, è sicuro fornire qui! Nella **finestra** degli **strumenti** , puoi

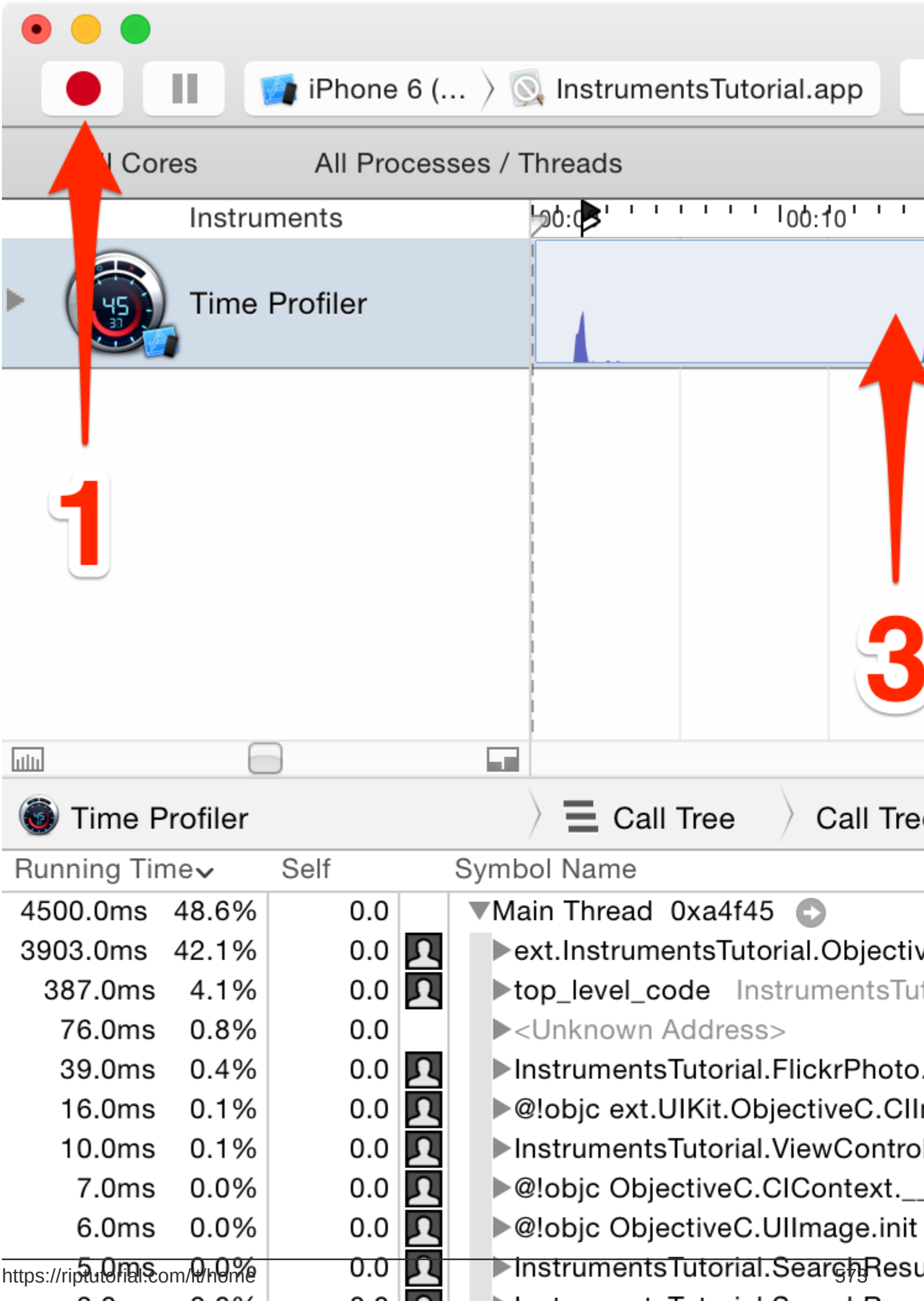
vedere il tempo che conta e una piccola freccia muoversi da sinistra a destra sopra il **grafico** al centro dello schermo. Questo indica che l'app è in esecuzione.

Ora inizia a utilizzare l'app. Cerca alcune immagini e visualizza in dettaglio uno o più risultati di ricerca. Probabilmente hai notato che entrare in un risultato di ricerca è noiosamente lento e scorrere un elenco di risultati di ricerca è anche incredibilmente fastidioso - è un'app terribilmente goffo!

Bene, sei fortunato, perché stai per iniziare a risolverlo! Tuttavia, per prima cosa inizierai a fare un giro veloce su ciò che stai guardando in **Strumenti** . Innanzitutto, assicurati che il selettore di visualizzazione sul lato destro della barra degli strumenti abbia entrambe le opzioni selezionate, in questo modo:



Ciò garantirà che tutti i pannelli siano aperti. Ora studia lo screenshot qui sotto e la spiegazione di ogni sezione sottostante:


iPhone 6 (...
InstrumentsTutorial.app
All Processes / Threads
Instrumentes
00:10
Time Profiler
1
3
Time Profiler
Call Tree
Call Tre
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
5.0ms 0.0% 0.0 InstrumentsTutorial.SearchResu
3.0ms 0.0% 0.0 InstrumentsTutorial.SearchResu

. Il pulsante rosso "registra" arresta e avvia l'applicazione che viene attualmente profilata quando viene cliccata (si passa da un'icona di registrazione a un'icona di arresto). Il pulsante di pausa fa esattamente ciò che ti aspetti e sospende l'esecuzione corrente dell'app.

2. Questo è il timer di esecuzione. Il timer conta per quanto tempo è stata eseguita l'applicazione che è stata profilata e quante volte è stata eseguita. Se si interrompe e quindi si riavvia l'app utilizzando i controlli di registrazione, si avvierà una nuova corsa e il display mostrerà Esegui 2 di 2.

3. Questo è chiamato una traccia. Nel caso del template di Time Profiler che hai selezionato, c'è solo uno strumento quindi c'è solo una traccia. Imparerai di più sulle specifiche del grafico mostrato qui più avanti nel tutorial.

4. Questo è il pannello dei dettagli. Mostra le principali informazioni sul particolare strumento che stai utilizzando. In questo caso, mostra i metodi più "caldi", cioè quelli che hanno consumato più tempo della CPU. Se fai clic sulla barra nella parte superiore che indica Albero delle chiamate (quella a sinistra) e seleziona Elenco di campioni, ti verrà presentata una diversa visualizzazione dei dati. Questa vista mostra ogni singolo campione. Fai clic su alcuni campioni e vedrai la traccia dello stack catturata apparire nella finestra di ispezione Dettagli estesi.

5. Questo è il pannello degli ispettori. Esistono tre ispettori: Impostazioni di registrazione, Impostazioni di visualizzazione e Dettagli estesi. A breve imparerai di più su alcune di queste opzioni.

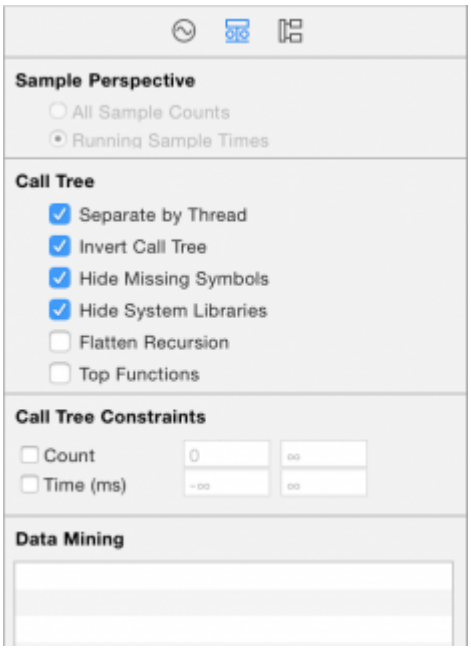
Drilling Deep

Esegui una ricerca di immagini e approfondisci i risultati. Personalmente mi piace cercare "cane", ma scegliere quello che desideri - potresti essere uno di quei gatti!

Ora, scorrere verso l'alto e verso il basso nell'elenco alcune volte in modo da avere una buona quantità di dati nel `Time Profiler` . Dovresti notare i numeri al centro dello schermo che cambiano e il **grafico che si** riempie; questo ti dice che **i cicli della CPU** sono in uso.

Davvero non ti aspetteresti che un'interfaccia utente sia così goffa come questa `table view` non è pronta per essere spedita fino a quando non scorre come un burro! Per aiutare a individuare il problema, è necessario impostare alcune opzioni.

Sul lato destro, seleziona l' **ispettore Impostazioni schermo** `(or press ⌥+2)` . **Nell'ispettore** , sotto la sezione `Call Tree` , selezionare Separa per **thread** , Inverti `Call Tree` , Nascondi simboli mancanti e Nascondi librerie di sistema. Sembrerà così:



Ecco cosa sta facendo ciascuna opzione per i dati visualizzati nella tabella a sinistra:

Separato per thread: ogni thread deve essere considerato separatamente. Ciò consente di capire quali thread sono responsabili della maggior quantità di utilizzo della **CPU** .

Inverti albero delle chiamate: con questa opzione, la `stack trace` dello `stack trace` viene considerata dall'alto verso il basso. Questo di solito è quello che vuoi, perché vuoi vedere i metodi più profondi in cui la **CPU** sta spendendo il suo tempo.

Nascondi simboli mancanti: se non è possibile trovare il file `dSYM` per la tua app o un `system framework` , invece di vedere i nomi dei metodi (simboli) nella tabella, vedrai solo i valori esadecimali corrispondenti agli indirizzi all'interno del binario. Se questa opzione è selezionata, vengono visualizzati solo i simboli completamente risolti ei valori **esadecimali** non risolti sono nascosti. Questo aiuta a declassare i dati presentati.

Nascondi librerie di sistema: quando questa opzione è selezionata, vengono visualizzati solo i simboli della tua app. È spesso utile selezionare questa opzione, poiché di solito ti interessa solo dove la **CPU** trascorre del tempo nel tuo codice - non puoi fare molto su quanta **CPU** le `system libraries` stanno usando!

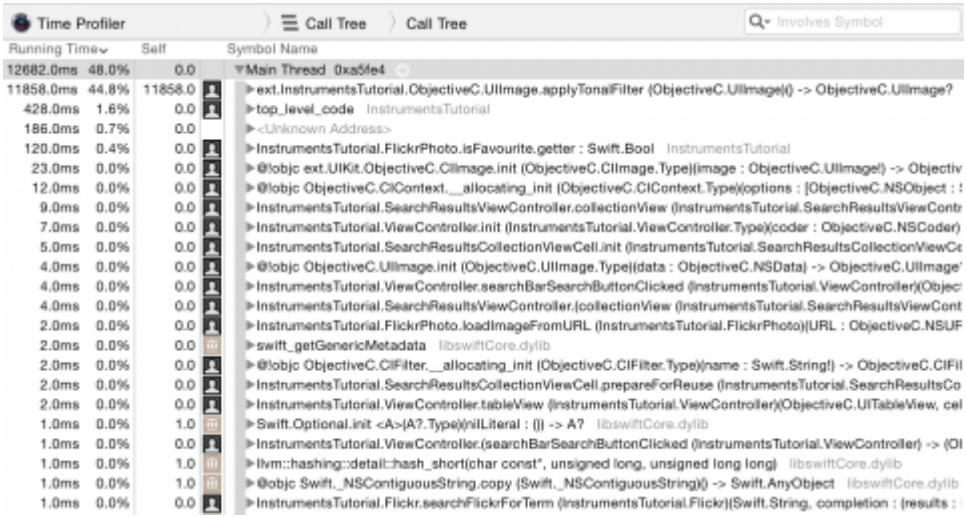
Flatten ricorsione: questa opzione tratta **le** funzioni **ricorsive** (quelle che si chiamano) come una voce in ogni `stack trace` , piuttosto che multiple.

Funzioni principali: Abilitando questa opzione, gli `Instruments` considerano il tempo totale trascorso in una funzione come la somma del tempo direttamente all'interno di tale funzione, nonché il tempo trascorso nelle funzioni chiamate da tale funzione.

Quindi se la funzione A chiama B, allora il tempo di A viene indicato come il tempo trascorso in A PLUS il tempo trascorso in B. Questo può essere davvero utile, in quanto ti consente di scegliere la cifra più grande ogni volta che scendi nello stack di chiamate, azzerando in sui metodi più dispendiosi in termini di tempo.

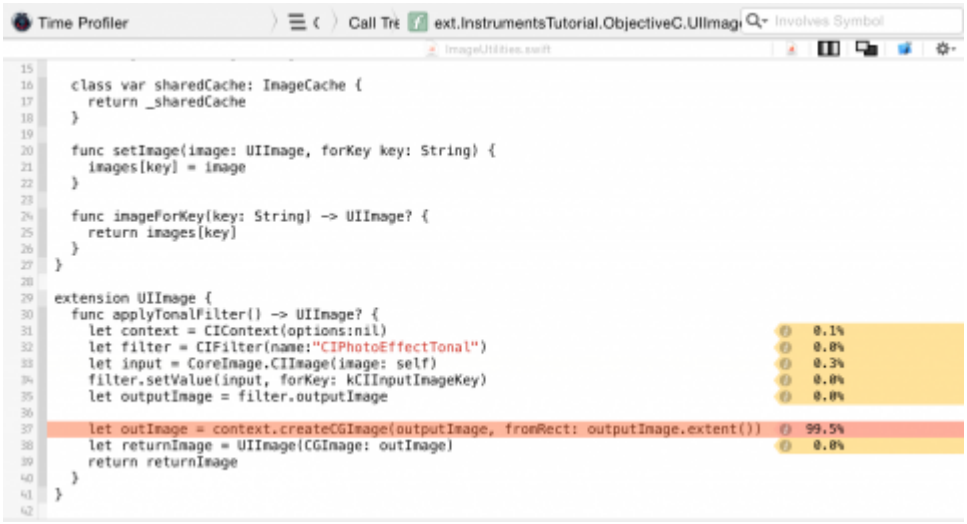
Se stai eseguendo un'app `Objective-C` , c'è anche un'opzione di Show **Obj-C Only** : se questa è selezionata, vengono visualizzati solo i metodi `Objective-C` , piuttosto che le funzioni `C` o `C++` . Non ce ne sono nel tuo programma, ma se stai guardando un'app `OpenGL` , potrebbe avere un `C++` , per esempio.

Sebbene alcuni valori potrebbero essere leggermente diversi, l'ordine delle voci dovrebbe essere simile alla tabella seguente una volta abilitate le opzioni sopra riportate:



Beh, questo sicuramente non sembra troppo bello. La maggior parte del tempo viene impiegata nel metodo che applica il filtro "tonale" alle foto in miniatura. Ciò non dovrebbe rappresentare uno shock per te, dato che il caricamento e lo scorrimento della tabella erano le parti più crudeli dell'interfaccia utente, e questo è il momento in cui le celle della tabella vengono costantemente aggiornate.

Per saperne di più su cosa sta succedendo all'interno di quel metodo, fai doppio clic sulla sua riga nella tabella. In questo modo verrà visualizzata la seguente visualizzazione:

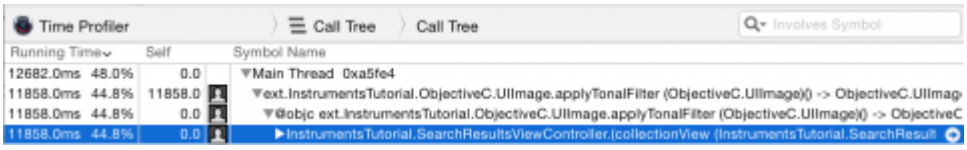


Beh, questo è interessante, no? `applyTonalFilter()` è un metodo aggiunto a `UIImage` in un'estensione e quasi il **100** % del tempo trascorso in esso viene impiegato per creare l'output di CGImage dopo aver applicato il filtro di immagine.

Non c'è molto che si possa fare per accelerare questo processo: la creazione dell'immagine è un

processo piuttosto intenso e richiede tutto il tempo necessario. Proviamo a fare un passo indietro e vediamo da dove viene chiamato `applyTonalFilter()` . **Fai clic su** `Call Tree` nella traccia di breadcrumb nella parte superiore della visualizzazione del codice per tornare alla schermata precedente:

Ora fai clic sulla piccola freccia a sinistra della riga applyTonalFilter nella parte superiore della tabella. Questo spiegherà l'albero delle chiamate per mostrare il chiamante di applyTonalFilter. Potrebbe essere necessario aprire anche la riga successiva; quando si esegue il profiling di Swift, a volte ci saranno delle righe duplicate nell'albero delle chiamate, precedute da @objc. Ti interessa la prima riga preceduta dal nome di destinazione della tua app (InstrumentsTutorial):

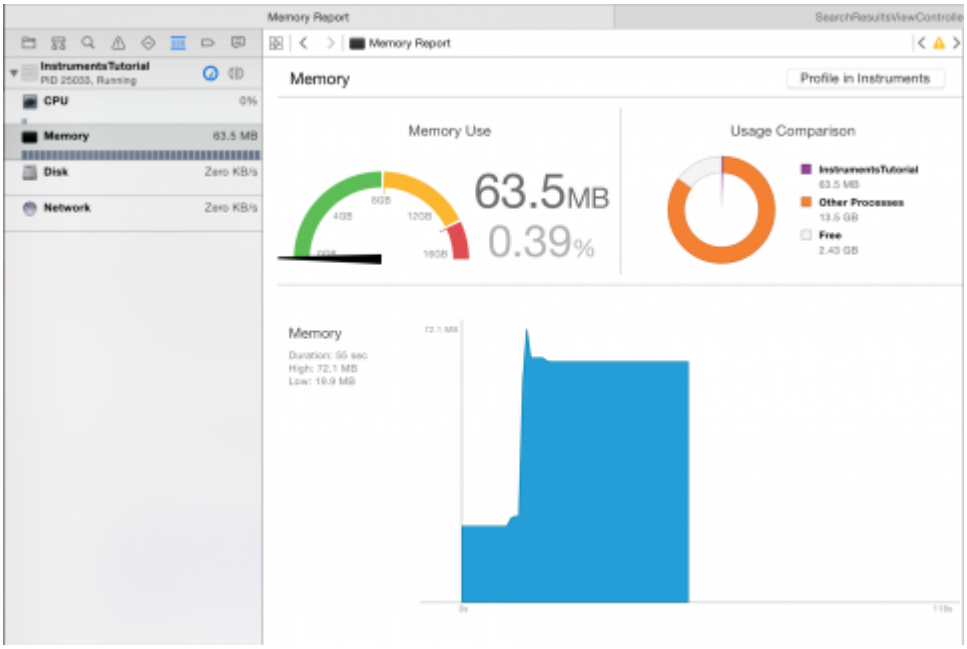


In questo caso, questa riga fa riferimento alla `cellForItemAtIndexPath` raccolta dei risultati `cellForItemAtIndexPath` . Fare doppio clic sulla riga per vedere il codice associato dal progetto.

Ora puoi vedere qual è il problema. Il metodo per applicare il filtro tonale richiede molto tempo per essere eseguito e viene chiamato direttamente da cellForItemAtIndexPath, che bloccherà il `main thread` (e quindi l'intera interfaccia utente) ogni volta che viene richiesta un'immagine filtrata.

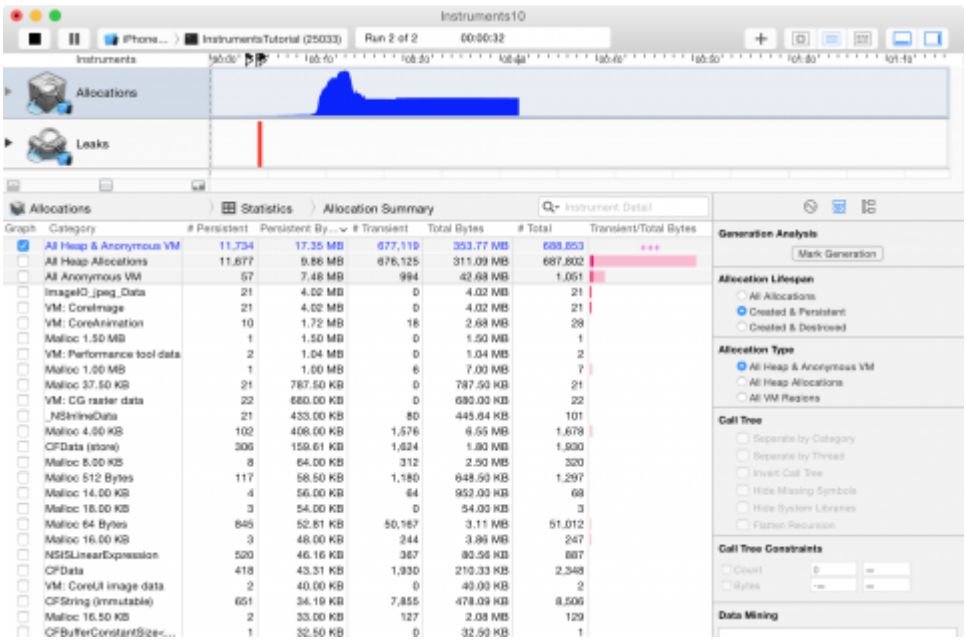
Accantonamenti

Vi sono informazioni dettagliate su tutti gli **oggetti** che vengono creati e sulla memoria che li supporta; mostra anche `retain counts` di ciascun oggetto. Per iniziare da capo con un nuovo `instruments profile` , esci dall'app Strumenti. Questa volta, crea ed esegui l'app e apri Debug Navigator nell'area Navigatori. Quindi fare clic su **Memoria** per visualizzare i grafici di utilizzo della memoria nella finestra principale:



Questi grafici sono utili per avere una rapida idea di come sta andando la tua app. Ma avrai

bisogno di un po 'più di energia. Fare clic sul pulsante `Profile in Instruments` e quindi su Trasferisci per portare questa sessione in **Strumenti** . Lo **strumento Allocations** si avvierà automaticamente.



Questa volta noterai due tracce. Uno è chiamato allocazioni e uno è chiamato perdite. La traccia delle allocazioni verrà discussa in dettaglio più avanti; la traccia di Leaks è generalmente più utile in Objective-C e non verrà trattata in questo tutorial. Quindi quale bug hai intenzione di rintracciare dopo? C'è qualcosa di nascosto nel progetto che probabilmente non sai è lì. Probabilmente hai sentito delle perdite di memoria. Ma quello che potresti non sapere è che in realtà ci sono due tipi di perdite:

Le vere perdite di memoria sono quelle in cui un oggetto non viene più referenziato da nulla ma ancora assegnato - ciò significa che la memoria non può mai essere riutilizzata. Anche con Swift e `ARC` aiutano a gestire la memoria, il tipo più comune di perdita di memoria è un `retain cycle or strong reference cycle` un `retain cycle or strong reference cycle` . Questo è quando due oggetti contengono forti riferimenti l'uno all'altro, così che ogni oggetto mantiene l'altro da essere deallocato. Ciò significa che la loro memoria non viene mai rilasciata!

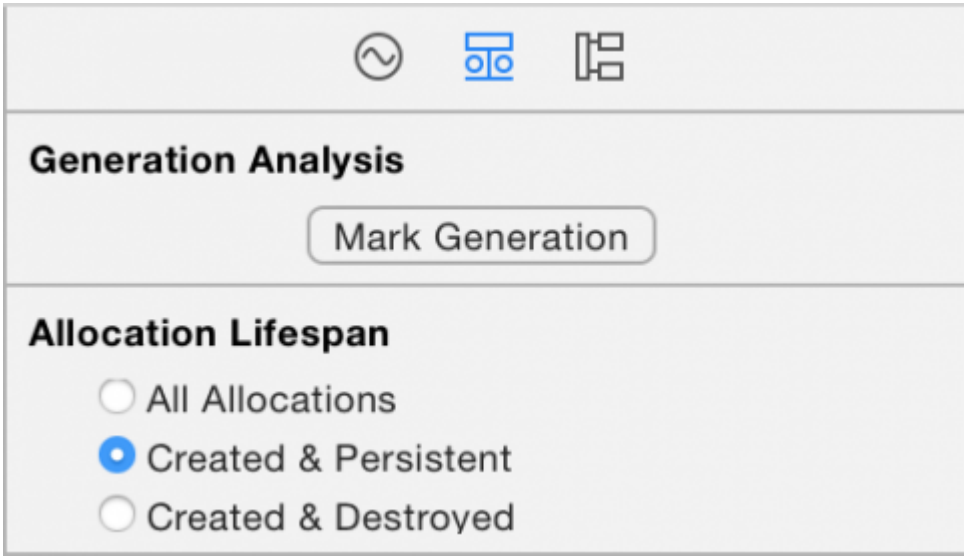
La crescita illimitata della memoria è dove la memoria continua ad essere allocata e non è mai data la possibilità di essere **deallocati** . Se continua così per sempre, allora ad un certo punto la `system's memory` sarà piena e avrai un grosso problema di memoria nelle tue mani. In iOS questo significa che l'app verrà uccisa dal sistema.

Con lo **strumento** Allocations in esecuzione sull'app, effettua cinque diverse ricerche nell'app ma non approfondisci ancora i risultati. Assicurati che le ricerche abbiano dei risultati! Ora lascia che l'app si stabilizzi un attimo aspettando qualche secondo.

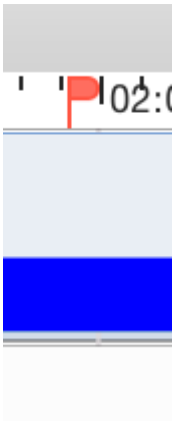
Avresti dovuto notare che il **grafico** nella traccia delle allocazioni è in aumento. Questo ti sta dicendo che la memoria è stata assegnata. È questa funzione che ti guiderà a trovare una `unbounded memory growth` .

Quello che stai per eseguire è `generation analysis` . Per fare ciò, premi il pulsante chiamato Mark

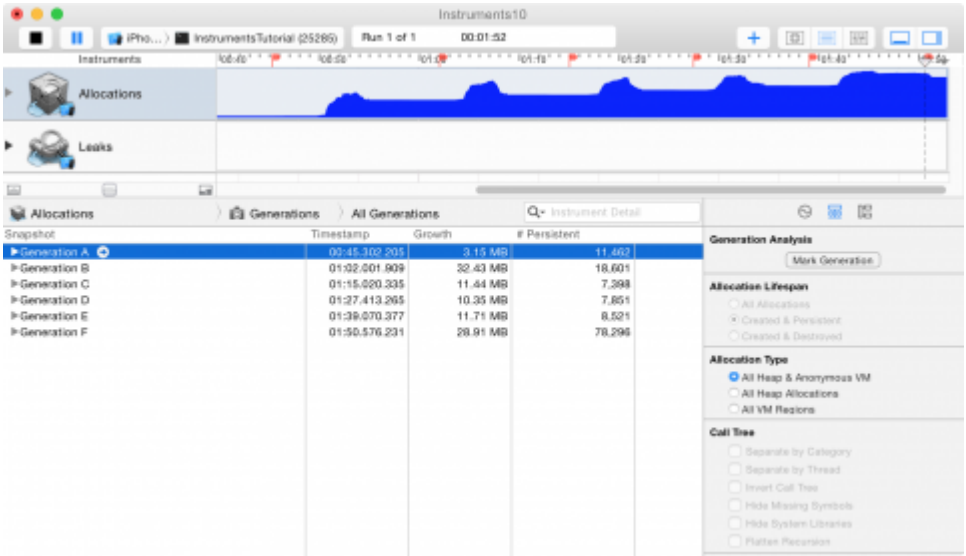
Generation. Troverai il pulsante nella parte superiore della finestra Impostazioni schermo:



Premerlo e vedrai apparire una bandiera rossa nella traccia, in questo modo:



Lo scopo `generation analysis` di `generation analysis` è eseguire un'azione più volte e vedere se la memoria sta crescendo in `unbounded fashion` . **Esegui** una ricerca, attendi qualche secondo per caricare le immagini, quindi torna alla pagina principale. Quindi segna di nuovo la generazione. Fallo ripetutamente per ricerche diverse. Dopo una **trivellazione** in alcune ricerche, gli **strumenti** saranno simili a questo:



A questo punto, dovresti diventare sospettoso. Si noti come il grafico blu sta salendo con ogni ricerca in cui si **fora** . Beh, non è certamente buono. Ma aspetta, per quanto riguarda gli `memory warnings?` sulla `memory warnings?` Sai di quelli, giusto? `Memory warnings` sono il modo di iOS di dire a un'app che le cose si stanno facendo più strette nel reparto di memoria e che è necessario cancellare un po 'di memoria.

È possibile che questa crescita non sia solo dovuta alla tua app; potrebbe essere qualcosa nelle profondità di `UIKit` che sta trattenendo la memoria. Dai al framework di sistema e alla tua app la possibilità di svuotare la **memoria** prima di puntare il dito su uno dei due.

Simula un `memory warning` selezionando `Instrument\Simulate Memory Warning` nella barra dei menu di Instruments o `Hardware\Simulate Memory Warning` dalla barra dei menu `simulator's` . Noterai che l'utilizzo della memoria diminuisce leggermente, o forse non del tutto. Certamente non si torna a dove dovrebbe essere. Quindi c'è ancora una **crescita di memoria illimitata** da qualche parte.

La ragione per contrassegnare una generazione dopo ogni iterazione di perforazione in una ricerca è che è possibile vedere quale **memoria** è stata allocata tra ogni generazione. Dai un'occhiata nel pannello dei dettagli e vedrai un paio di generazioni.

Leggi Profilo con strumenti online: https://riptutorial.com/it/ios/topic/9629/profilo-con-strumenti

Capitolo 129: Quadro dei contatti

Osservazioni

link utili

- Documentazione Apple
- Domande e risposte relative allo straripamento dello stack
- WWDC15 Session Video

Examples

Autorizzazione dell'accesso ai contatti

Importare il framework

veloce

```
import Contacts
```

Objective-C

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

Controllo accessibilità

veloce

```
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;
}
```

Richiesta di autorizzazione

veloce

```
var contactStore = CKContactStore()
contactStore.requestAccessForEntityType(CKEntityType.Contacts, completionHandler: { (ok, _) ->
Void in
    if access{
        //access contacts
    }
}
```

Accesso ai contatti

Applicazione di un filtro

Per accedere ai contatti, dovremmo applicare un filtro di tipo `NSPredicate` alla nostra variabile contactStore che abbiamo definito nell'esempio Autorizzazione Contact Access. Ad esempio, qui vogliamo ordinare i contatti con il nome corrispondente con il nostro:

veloce

```
let predicate = CNContact.predicateForContactsMatchingName("Some Name")
```

Objective-C

```
NSPredicate *predicate = [CNContact predicateForContactsMatchingName:@"Some Name"];
```

Specifica delle chiavi da recuperare

Qui, vogliamo recuperare il nome, il cognome e l'immagine del profilo del contatto:

veloce

```
let keys = [CNContactGivenNameKey, CNContactFamilyNameKey, CNContactImageDataKey]
```

Recupero di contatti

veloce

```
do {
    let contacts = try contactStore.unifiedContactsMatchingPredicate(predicate, keysToFetch:
keys)
} catch let error as NSError {
    //...
}
```

Accesso ai dettagli di contatto

veloce

```
print(contacts[0].givenName)
print(contacts[1].familyName)
let image = contacts[2].imageData
```

Aggiungere un contatto

veloce

```
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)
```

Leggi Quadro dei contatti online: https://riptutorial.com/it/ios/topic/5872/quadro-dei-contatti

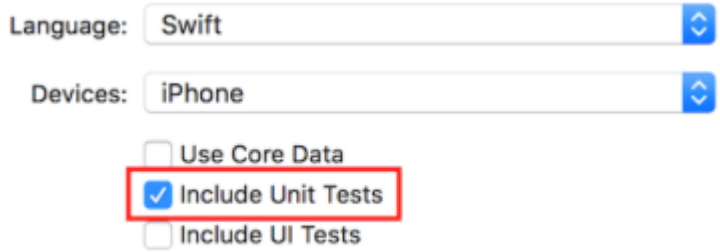
Capitolo 130: Quadro XCTest - Test unitario

Examples

Aggiunta di file di test a Xcode Project

Quando si crea il progetto

È necessario selezionare "Includi test unitari" nella finestra di dialogo di creazione del progetto.



Dopo aver creato il progetto

Se hai perso il controllo di quell'elemento durante la creazione del tuo progetto, puoi sempre aggiungere i file di test in un secondo momento. Fare così:

1- Vai alle impostazioni del progetto in Xcode

2- Vai a "Obiettivi"

3- Fai clic su "Aggiungi target"

4- In "Altro", selezionare "Pacchetto test cacao Touch Unit Test"

Alla fine, dovresti avere un file chiamato `[Your app name]Tests.swift` . In Objective-C, dovresti avere due file chiamati `[Your app name]Tests.h` e `[Your app name]Tests.m` .

`[Your app name]Tests.swift or .m` file `[Your app name]Tests.swift or .m` includerà per impostazione predefinita:

- `XCTest` moduli `XCTest`
- A `[Your app name]Tests` classe che estende `XCTestCase`
- `setUp` , `tearDown` , `testExample` , `testPerformanceExample`

veloce

```
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
```

Aggiunta di Storyboard e Visualizza controller come istanze per testare il file

Per iniziare con il test delle unità, che verrà eseguito nel file di test e testeremo il View Controller e Storyboard, dovremmo introdurre questi due file nel file di test.

Definizione del controller di visualizzazione

veloce

```
var viewController : ViewController!
```

Presentazione dello Storyboard e inizializzazione del View Controller

Aggiungi questo codice al metodo `setUp()` :

veloce

```
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];
```

In questo modo, potresti scrivere metodi di prova e sapranno dove cercare gli errori. In questo caso, ci sono View Controller e Storyboard.

Aggiunta di metodi di prova

Secondo Apple:

Metodi di prova

> Un metodo di prova è un metodo di istanza di una classe di test che inizia con il test prefisso, non accetta parametri e restituisce void, ad esempio, (void) testColorIsRed (). Un metodo di prova esercita il codice nel progetto e, se tale codice non produce il risultato previsto, segnala i guasti utilizzando un set di API di asserzione. Ad esempio, il valore di ritorno di una funzione potrebbe essere confrontato con un valore atteso o il test potrebbe affermare che l'uso improprio di un metodo in una delle classi genera un'eccezione.

Quindi aggiungiamo un metodo di test usando "test" come prefisso del metodo, come:

veloce

```
func testSomething() {

}
```

Objective-C

```
- (void)testSomething {

}
```

Per testare realmente i risultati, usiamo il metodo `XCTAssert()` , che prende un'espressione booleana, e se è vero, segna il test come riuscito, altrimenti lo contrassegnerà come non riuscito.

Diciamo che abbiamo un metodo nella classe View Controller chiamato `sum()` che calcola la somma di due numeri. Per testarlo, usiamo questo metodo:

veloce

```
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);
}
```

Nota

Per impostazione predefinita, non è possibile accedere all'etichetta, alla casella di testo o ad altri elementi dell'interfaccia utente della classe View Controller dalla classe di test se vengono creati per la prima volta nel file Storyboard. Questo perché sono inizializzati nel metodo `loadView()` della classe View Controller e questo non verrà chiamato durante il test. Il modo migliore per chiamare `loadView()` e tutti gli altri metodi richiesti è accedere alla proprietà `view` della nostra proprietà `viewController` . È necessario aggiungere questa riga prima di testare gli elementi dell'interfaccia utente:

```
XCTAssertNotNil(viewController.view)
```

Inizia i test

Test di un metodo specifico

Per testare un metodo specifico, fare clic sul quadrato accanto alla definizione del metodo.

Testare tutti i metodi

Per testare tutti i metodi, fare clic sul quadrato accanto alla definizione della classe.

Vedi il risultato del test

Se c'è un controllo verde accanto alla definizione, il test è riuscito.

Se c'è una croce rossa accanto alla definizione, il test è fallito.

Esecuzione di tutti i test

```
Product -> Test OR Cmd + U
```

Verranno eseguiti tutti i test da tutti gli obiettivi di test!

Importa un modulo che può essere testato

Classi, strutture, enumerazioni e tutti i loro metodi sono `internal` per impostazione predefinita. Ciò significa che è possibile accedervi solo dallo stesso modulo. I test case sono in un target diverso e questo significa che si trovano in un modulo diverso. Per poter accedere al metodo che vuoi testare, devi importare il modulo da testare usando la parola chiave `@testable` .

Diciamo che abbiamo un modulo principale chiamato `ToDo` e vogliamo scrivere test per questo. Vorremmo importare quel modulo in questo modo:

```
@testable import ToDo
```

Tutti i metodi di test nel file con questa istruzione import ora possono accedere a tutte `internal` classi `internal` , le strutture, le enumerazioni e tutti i loro metodi `internal` del modulo `ToDo` .

Non si dovrebbero mai aggiungere i file con gli elementi che si desidera testare al target di test perché ciò può portare a errori di debug difficili.

Attivare il caricamento e l'aspetto della vista

Visualizza il caricamento

In un test per un controller di visualizzazione a volte si desidera attivare l'esecuzione di `loadView()` o `viewDidLoad()` . Questo può essere fatto accedendo alla vista. Supponiamo che tu abbia l'istanza del controller di visualizzazione nel tuo test chiamato `sut` (system under test), quindi il codice sarebbe simile a questo:

```
XCTAssertNotNil(sut.view)
```

Visualizza aspetto

Puoi anche attivare i metodi `viewWillAppear(_:)` e `viewDidAppear(_:)` aggiungendo il seguente codice:

```
sut.beginAppearanceTransition(true, animated: true)
sut.endAppearanceTransition()
```

Scrivere una lezione di prova

```
import XCTest
@testable import PersonApp

class PersonTests: XCTestCase {
    func test_completeName() {
        let person = Person(firstName: "Josh", lastName: "Brown")
```

```
        XCTAssertEqual(person.completeName(), "Josh Brown")
    }
}
```

Ora discutiamo cosa sta succedendo qui. La riga `import XCTest` ci consentirà di estendere `XCTestCase` e utilizzare `XCTAssertEqual` (tra le altre affermazioni). L'estensione di `XCTestCase` e il prefisso del nostro nome test con il `test` garantiranno che Xcode esegua automaticamente questo test durante l'esecuzione dei test nel progetto (**U** o **Prodotto** > **Test**). La linea `@testable import PersonApp` importerà il nostro target `PersonApp` modo che possiamo testare e utilizzare classi da esso, come la `Person` nel nostro esempio sopra. Infine, il nostro `XCTAssertEqual` garantirà che `person.completeName()` sia uguale alla stringa `"Josh Brown"` .

Leggi Quadro XCTest - Test unitario online: https://riptutorial.com/it/ios/topic/5075/quadro-xctest---test-unitario

Capitolo 131: Regno

Osservazioni

Aggiunta di un nuovo RLMObject a un dominio esistente: schema e migrazioni

L'aggiunta di nuove classi di modelli in un reame non richiede una migrazione o un bump della versione dello schema; solo apportando modifiche a un Reame esistente.

Examples

RLMObject Base Model Class con chiave primaria - Objective-C

Un esempio di una classe del modello di base RLMObject che utilizza una chiave primaria e alcune proprietà predefinite generiche. Le sottoclassi possono quindi impostare metadati specifici per le loro esigenze.

```
@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
```

Leggi Regno online: https://riptutorial.com/it/ios/topic/4084/regno

Capitolo 132: Rendi gli angoli di UIView selettivi arrotondati

Examples

Obiettivo codice C per rendere arrotondato l'angolo selezionato di un UiView

Prima **import** `#import <QuartzCore/QuartzCore.h>` nella classe ViewController. Ecco come ho impostato il mio punto di vista nel codice

```
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];
```

Ecco la funzione che fa il sollevamento pesante e arrotonda i bordi selezionati che sono il bordo inferiore destro e il bordo superiore destro nel nostro caso

```
- (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;
}
```

Leggi Rendi gli angoli di UIView selettivi arrotondati online:
https://riptutorial.com/it/ios/topic/7224/rendi-gli-angoli-di-uiview-selettivi-arrotondati

Capitolo 133: Ridimensionamento di UIImage

Parametri

CGInterpolationQuality	Livelli di qualità di interpolazione per il rendering di un'immagine.
La qualità dell'interpolazione è un parametro dello stato della grafica	typedef enum CGInterpolationQuality CGInterpolationQuality;

Examples

Ridimensiona qualsiasi immagine per dimensione e qualità

```
- (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;
}
```

Leggi Ridimensionamento di UIImage online:
https://riptutorial.com/it/ios/topic/6422/ridimensionamento-di-uiimage

Capitolo 134: Riferimento CGContext

Osservazioni

Il tipo opaco di **CGContextRef** rappresenta una destinazione di disegno 2D di quarzo. Un contesto grafico contiene parametri di disegno e tutte le informazioni specifiche del dispositivo necessarie per rendere il disegno su una pagina alla destinazione, indipendentemente dal fatto che la destinazione sia una finestra in un'applicazione, un'immagine bitmap, un documento PDF o una stampante.

Examples

Disegnare la linea

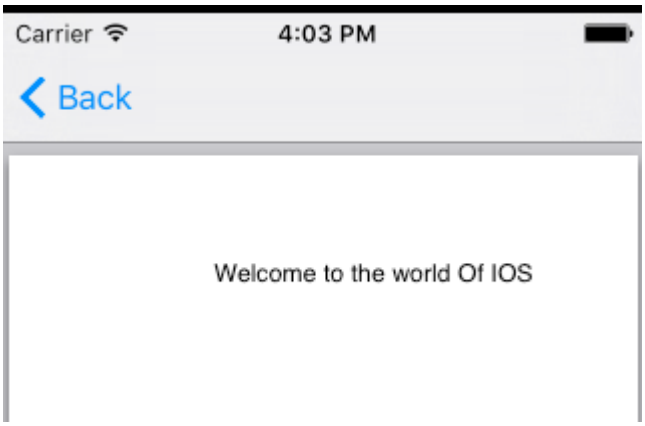
```
CGContextRef context = UIGraphicsGetCurrentContext();

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

Disegna testo

Draw To richiede che il **framework Core Text** sia aggiunto nella Build Phase

```
[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);
```



Leggi Riferimento CGContext online: https://riptutorial.com/it/ios/topic/2664/riferimento-cgcontext

Capitolo 135: Rilevamento volti mediante CoreImage / OpenCV

Examples

Rilevamento di volti e feature

Objective-C

Importa quanto segue per ViewController

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

Chiama la funzione

```
[self faceDetector];
```

Definizione della funzione:

```
-(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)];


}
```

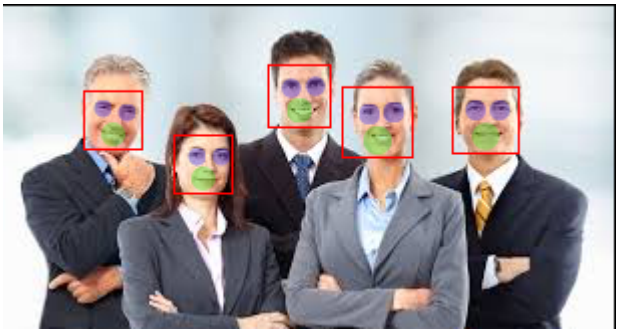
Mark Face Function

```
//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];
        }
    }

    // }
}
```

The Simulator ScreenShot per la funzione



Leggi Rilevamento volti mediante CoreImage / OpenCV online:
https://riptutorial.com/it/ios/topic/7298/rilevamento-volti-mediante-coreimage---opencv

Capitolo 136: Runtime in Objective-C

Examples

Utilizzando oggetti associati

Gli oggetti associati sono utili quando si desidera aggiungere funzionalità a classi esistenti che richiedono lo stato di attesa.

Ad esempio, aggiungendo un indicatore di attività a ogni UIView:

Implementazione 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
```

Puoi anche accedere al runtime Objective-C tramite Swift:

codice SWIFT

```
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()
    }
}
```

Leggi Runtime in Objective-C online: https://riptutorial.com/it/ios/topic/10120/runtime-in-objective-c

Capitolo 137: Scanner di codici QR

introduzione

I codici QR (Quick Response) sono codici a barre bidimensionali ampiamente utilizzati su etichette ottiche leggibili a macchina. iOS fornisce un modo per leggere i codici QR utilizzando il framework `AVFoundation` da iOS 7 in poi. Questo framework fornisce una serie di API per configurare / aprire la telecamera e leggere i codici QR dal feed della telecamera.

Examples

Scansione UIViewController per QR e visualizzazione dell'ingresso video

```
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` - verrà richiamato su una scansione di successo `initCaptureSession` - inizializza la scansione per QR e l'input della telecamera `dismissCaptureSession` - nasconde l'input della videocamera e interrompe la scansione

Scansione del codice QR con framework AVFoudation

Prima di iOS 7 quando si desidera eseguire la scansione di un codice QR, potrebbe essere necessario affidarsi a framework o librerie di terze parti come zBar o zXing . Ma Apple ha introdotto `AVCaptureMetaDataOutput` da iOS 7 per la lettura dei codici a barre.

Per leggere il codice QR utilizzando `AVFoundation` è necessario configurare / creare `AVCaptureSession` e utilizzare `captureOutput:didOutputMetadataObjects:fromConnection:` delegate method.

Passo 1

Importa il framework `AVFoundation` e conferma al protocollo `AVCaptureMetadataOutputObjectsDelegate`

```
import AVFoundation
class ViewController: UIViewController, AVCaptureMetadataOutputObjectsDelegate
```

Passo 2

La lettura del codice QR è totalmente basata sulla cattura video. Quindi per acquisire video continui creare una `AVCaptureSession` e impostare l'input e l'output del dispositivo. Aggiungi il codice seguente nel metodo `viewDidLoad` controller di visualizzazione

```
// 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
}
```

Passaggio 3

Implementare il metodo delegato `AVCaptureMetadataOutputObjectsDelegate` per leggere il codice QR

```
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
            }
        }
}
```

qui l'oggetto metadati può darti anche i limiti del codice QR letto sul feed della telecamera. Per ottenere i limiti basta passare l'oggetto metadati al metodo `transformedMetadataObject` di `videoPreviewLayer` come sotto.

```
let barCodeObject = videoPreviewLayer?.transformedMetadataObject(for: metadataObj)
            qrCodeFrameView?.frame = barCodeObject!.bounds
```

Leggi Scanner di codici QR online: https://riptutorial.com/it/ios/topic/7963/scanner-di-codici-qr

Capitolo 138: SDK AWS

Examples

Carica un'immagine o un video su S3 utilizzando AWS SDK

Prima di iniziare con l'esempio, consiglierei di creare un Singleton con un membro della classe delegato in modo da poter ottenere un caso di utilizzo del caricamento di un file in background e consentire all'utente di continuare a utilizzare l'app mentre i file vengono caricati anche quando l'app è lo sfondo.

Iniziamo, innanzitutto, dovremmo creare un enum che rappresenti la configurazione 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"
}
```

Ora, dovremmo impostare le credenziali quando la tua app `didFinishLaunchingWithOptions` per la prima volta, quindi dovremmo impostarle all'interno di `AppDelegate` nel metodo `didFinishLaunchingWithOptions` (fai attenzione che dovresti impostare la tua regione sul parametro `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)
}
```

Poiché siamo già all'interno di AppDelegate, dovremmo implementare la richiamata in background gestita dall'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)
}
```

Ora, quando l'utente sposta l'app sullo sfondo, il caricamento continuerà il caricamento effettivo.

Per caricare il file utilizzando l'SDK AWS dovremo scrivere il file sul dispositivo e fornire all'SDK il percorso effettivo. Per fare un esempio, immagina di avere un UIImage (potrebbe anche essere un video ...) e lo scriveremo in una cartella temporanea:

```
// 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 e fileName verranno utilizzati per il caricamento effettivo in seguito.

Ci sono 2 chiusure che dovremo definire fornite dall'SDK di AWS,

1. `AWSS3TransferUtilityUploadCompletionHandlerBlock` - Una chiusura che avvisa quando il caricamento è terminato (o meno)
2. `AWSS3TransferUtilityUploadProgressBlock` - Una chiusura che notifica ogni byte inviato

Se si prevede di avere un Singleton, è necessario definire tali tipi come membri della classe. L'implementazione dovrebbe assomigliare a questa:

```
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)
    }
```

NOTA: Se si utilizza un Singleton, è possibile che si desideri definire un delegato che lo segnalerà all'avanzamento o al termine del file. Se non si utilizza un Singleton, è possibile creare un metodo statico con i tipi rilevanti:

```
    static func uploadImageToS3(fileURL : NSURL,
                                fileName : String,
                progressFunctionUpdater : Float -> Void,
```

```
                            resultBlock : (NSError?) -> Void)
{
 //    Actual implementation .....
 //    ...
 //    ...
}
```

1. `progressFunctionUpdater` : riporterà a una funzione in corso.
2. `resultBlock` - Se si restituisce nil, il caricamento ha avuto esito positivo, si invia l'oggetto error

Signore e signori, il caricamento effettivo:

```
        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
        }
```

Caricamento S3 felice :)

Leggi SDK AWS online: https://riptutorial.com/it/ios/topic/4734/sdk-aws

Capitolo 139: segues

Examples

Una panoramica

Dalla documentazione di Apple:

> Un oggetto UIStoryboardSegue è responsabile **dell'esecuzione della transizione visiva tra due controller di vista** . Inoltre, gli oggetti segue vengono utilizzati per preparare la transizione da un controller di visualizzazione a un altro. **Gli oggetti Segue contengono informazioni sui controller della vista coinvolti in una transizione** . Quando viene attivato un seguito, ma prima che avvenga la transizione visiva, il runtime dello storyboard chiama il metodo readyForSegue: sender: method in modo che possa trasferire tutti i dati necessari al controller della vista che sta per essere visualizzato.

---

attributi

veloce

```
sourceViewController: UIViewController {get}
destinationViewController: UIViewController {get}
identifier: String? {get}
```

Riferimenti:

- Riferimento di classe UIViewController
- Riferimento di classe UIStoryBoardSegue

Preparare il tuo controller di visualizzazione prima di attivare un Segue

Preparare ForSegue :

```
func prepareForSegue(_ segue:UIStoryboardSegue, sender sender:AnyObject?)
```

> Notifica al controller della vista che sta per essere eseguito un seguito

parametri

segue : L'oggetto segue.

mittente : l'oggetto che ha inizializzato il seguito.

Esempio in Swift

Esegui un'attività se l'identificativo del seguito è "SomeSpecificIdentifier"

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
    if segue.identifier == "SomeSpecificIdentifier" {
        //- Do specific task
    }
}
```

Decidere se deve essere eseguita una Segue invocata.

ShouldPerformSegueWithIdentifier :

```
func shouldPerformSegueWithIdentifier(_ identifier:String, sender sender:AnyObject?) -> Bool
```

Determina se il follow con l'identificatore specificato deve essere eseguito.

parametri

Identificatore : stringa che identifica il seguito innescato

Mittente : l'oggetto che ha inizializzato il seguito.

Esempio in Swift

Esegui solo se l'identificatore è "SomeSpecificIdentifier"

```
override func shouldPerformSegueWithIdentifier(identifier:String, sender:AnyObject?) -> Bool {
    if identifier == "SomeSpecificIdentifier" {
        return true
    }
    return false
}
```

Usare Segues per navigare all'indietro nella pila di navigazione

---

Rilassati Segues

Unwind Segues ti offre un modo per "srotolare" lo stack di navigazione e specificare una destinazione a cui tornare. La firma di questa funzione è fondamentale per Interface Builder che lo riconosce. **Deve avere un valore di ritorno di IBAction e prendere un parametro di UIStoryboardSegue** . Il nome della funzione non ha importanza. In effetti, la funzione non ha nemmeno bisogno di fare nulla. È lì solo come un marker di cui UIViewController è la destinazione del Segmento di Unwind. [Source]

[1]

Firma richiesta di un seguito di svolgimento

Obiettivo C:

```
-(IBAction)prepareForUnwind:(UIStoryboardSegue *)segue {
}
```

Swift:

```
@IBAction func prepareForUnwind(segue: UIStoryboardSegue) {
}
```

Trigger Segue Programmatically

PerformSegueWithIdentifier:

```
 func performSegueWithIdentifier(_ identifier:String, sender sender:AnyObject?)
```

> Inizia il seguito con l'identificatore specificato dal file storyboard del controller della vista corrente

parametri

Identificatore : stringa che identifica il seguito innescato

Mittente : l'oggetto che avvierà il seguito.

Esempio in Swift

Esecuzione di un seguito con identificatore "SomeSpecificIdentifier" da una selezione di righe di visualizzazione tabella:

```
func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
    performSegueWithIdentifier("SomeSpecificIdentifier", sender: indexPath.item)
}
```

Leggi segues online: https://riptutorial.com/it/ios/topic/5575/segues

Capitolo 140: Servizi Safari

Examples

Implementare SFSafariViewControllerDelegate

È necessario implementare `SFSafariViewControllerDelegate` modo che la classe venga notificata quando l'utente preme il pulsante Fine su SafariViewController e si può anche ignorarlo.

Prima dichiari la tua classe per implementare il protocollo.

```
class MyClass: SFSafariViewControllerDelegate {

}
```

Implementare il metodo delegato da notificare in caso di licenziamento.

```
func safariViewControllerDidFinish(controller: SFSafariViewController) {
    // Dismiss the SafariViewController when done
    controller.dismissViewControllerAnimated(true, completion: nil)
}
```

Non dimenticare di impostare la classe come delegato di SafariViewController.

```
let safariVC = SFSafariViewController(URL: yourURL)
safariVC.delegate = self
```

Ulteriori metodi delegati che puoi implementare sono:

```
// 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] { }
```

Aggiungi elementi all'elenco di lettura di Safari

È possibile aggiungere elementi all'elenco di lettura di un utente in Safari chiamando il metodo `addItem` sul singleton `SSReadingList` .

```
let readingList = SSReadingList.default()
readingList?.addItem(with: yourURL, title: "optional title", previewText: "optional preview
text")
```

L'Elenco di lettura predefinito può essere `nil` se l'accesso all'Elenco di lettura non è consentito.

Inoltre è possibile verificare se l'Elenco di lettura supporta un URL chiamando `supportsURL` .

```
SSReadingList.default().supportsURL(URL(string: "https://example.com")!)
```

Ciò restituirà `true` o `false` indicando se l'URL indicato è supportato da Elenco di lettura di Safari. Ad esempio, per determinare se mostrare un pulsante per aggiungere un URL alla lista di lettura.

Apri un URL con SafariViewController

Non dimenticare di importare prima il framework necessario.

```
import SafariServices
//Objective-C
@import SafariServices;
```

`SafariViewController` un'istanza di un'istanza di `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];
```

Opzionalmente puoi anche dire a SafariViewController di entrare in modalità di lettura, se possibile, una volta completato il caricamento.

```
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];
```

Presenta il controller della vista.

```
present(safariVC, animated: true, completion: nil)
//Objective-C
[self presentViewController:sfvc animated:YES completion:nil];
```

Leggi Servizi Safari online: https://riptutorial.com/it/ios/topic/1371/servizi-safari

Capitolo 141: Sicurezza

introduzione

La sicurezza in iOS è legata alla sicurezza dei dati, alla sicurezza del trasporto, alla sicurezza del codice, ecc

Examples

Sicurezza del trasporto con SSL

Le app iOS devono essere scritte in modo da fornire sicurezza ai dati che vengono trasportati sulla rete.
SSL è il modo comune per farlo.

Ogni volta che l'app tenta di chiamare i servizi Web per estrarre o inviare dati ai server, dovrebbe **utilizzare SSL su HTTP, ovvero HTTPS** .
Per fare ciò, l' **app deve chiamare** `https://server.com/part` tali servizi Web e non `http://server.com/part` .
In questo caso, l'app deve considerare attendibile il server `server.com` utilizzando il certificato SSL.

Ecco l'esempio di convalida della fiducia del server

Implementa `URLSessionDelegate` come:

```
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)
}
```

Ecco il gestore di fiducia: (impossibile trovare il codice 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
```

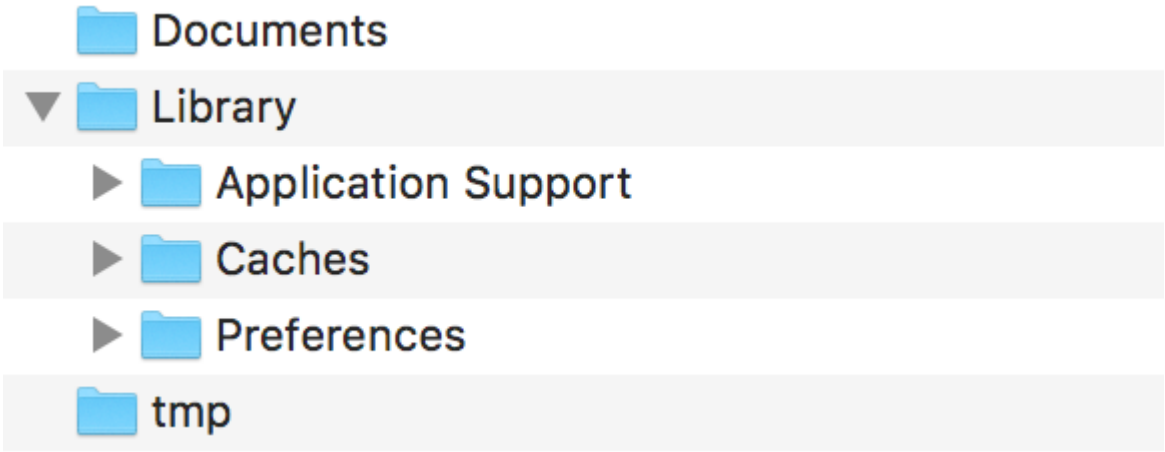
Server_Public_SSL_Cert.der è la chiave SSL pubblica dei server.

Utilizzando questo approccio la nostra app può assicurarsi che stia comunicando con il server previsto e nessuno stia intercettando la comunicazione app-server.

Protezione dei dati nei backup di iTunes

Se vogliamo che i nostri dati delle app siano protetti dai backup di iTunes, dobbiamo saltare i nostri dati delle app dal backup su iTunes.
Ogni volta che il dispositivo iOS esegue il backup utilizzando iTunes su macOS, tutti i dati memorizzati da tutte le app vengono copiati in quel backup e archiviati sul computer di backup.

Ma possiamo escludere i dati delle nostre app da questo backup utilizzando la chiave `URLResourceKey.isExcludedFromBackupKey` .
Ecco la struttura delle directory della nostra app:



Nota: i dati generalmente sensibili sono memorizzati nella directory 'Application Support'.

Ad esempio, se vogliamo escludere tutti i nostri dati memorizzati nella directory del **supporto applicativo** , possiamo utilizzare la suddetta chiave come segue:

```
    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
    }
```

Ci sono molti strumenti disponibili per vedere i backup di iTunes per tutti i dati di backup per confermare se l'approccio sopra funziona o meno.
iExplorer è buono per esplorare i backup di iTunes.

Leggi Sicurezza online: https://riptutorial.com/it/ios/topic/9999/sicurezza

Capitolo 142: Simulatore

introduzione

I simulatori iOS, watchOS e tvOS sono ottimi modi per testare le tue app senza utilizzare un dispositivo reale. Qui parleremo di lavorare con i simulatori.

Osservazioni

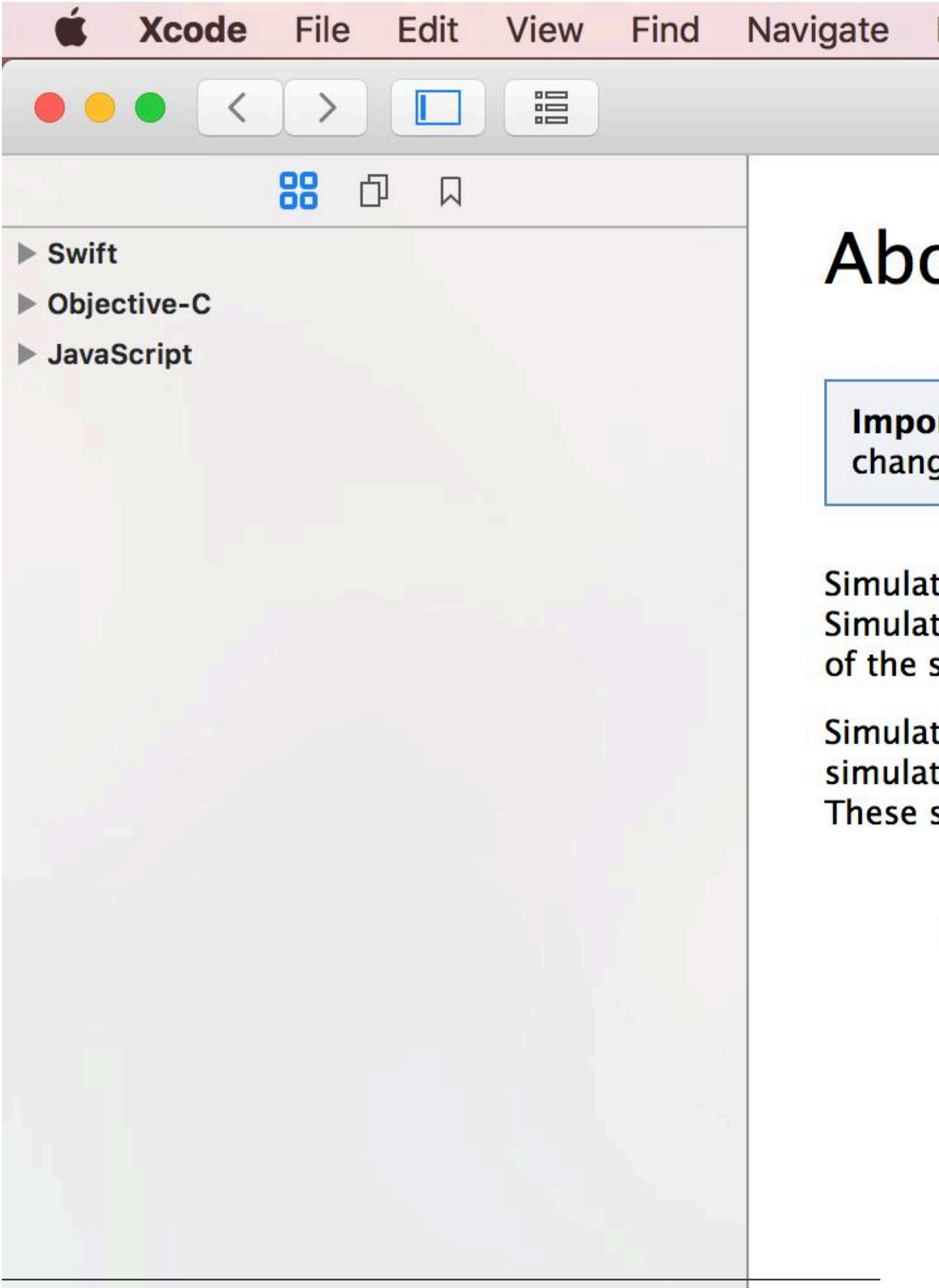
Diversi tipi di simulatori

- simulatore iOS
- watchOS Simulator
- simulatore TVOS
- Touch Bar Simulator

  Non esiste un simulatore per macOS, perché Xcode è eseguito su macOS e, quando necessario, eseguirà le app native.

Ottenere aiuto

Puoi sempre visitare la guida di Simulator in Aiuto -> Aiuto Simulatore:


Xcode File Edit View Find Navigate
Swift
Objective-C
JavaScript
Abo
Impo
chang
Simulat
Simulat
of the s
Simulat
simulat
These s

Capitolo 143: Simulazione della posizione utilizzando i file GPX iOS

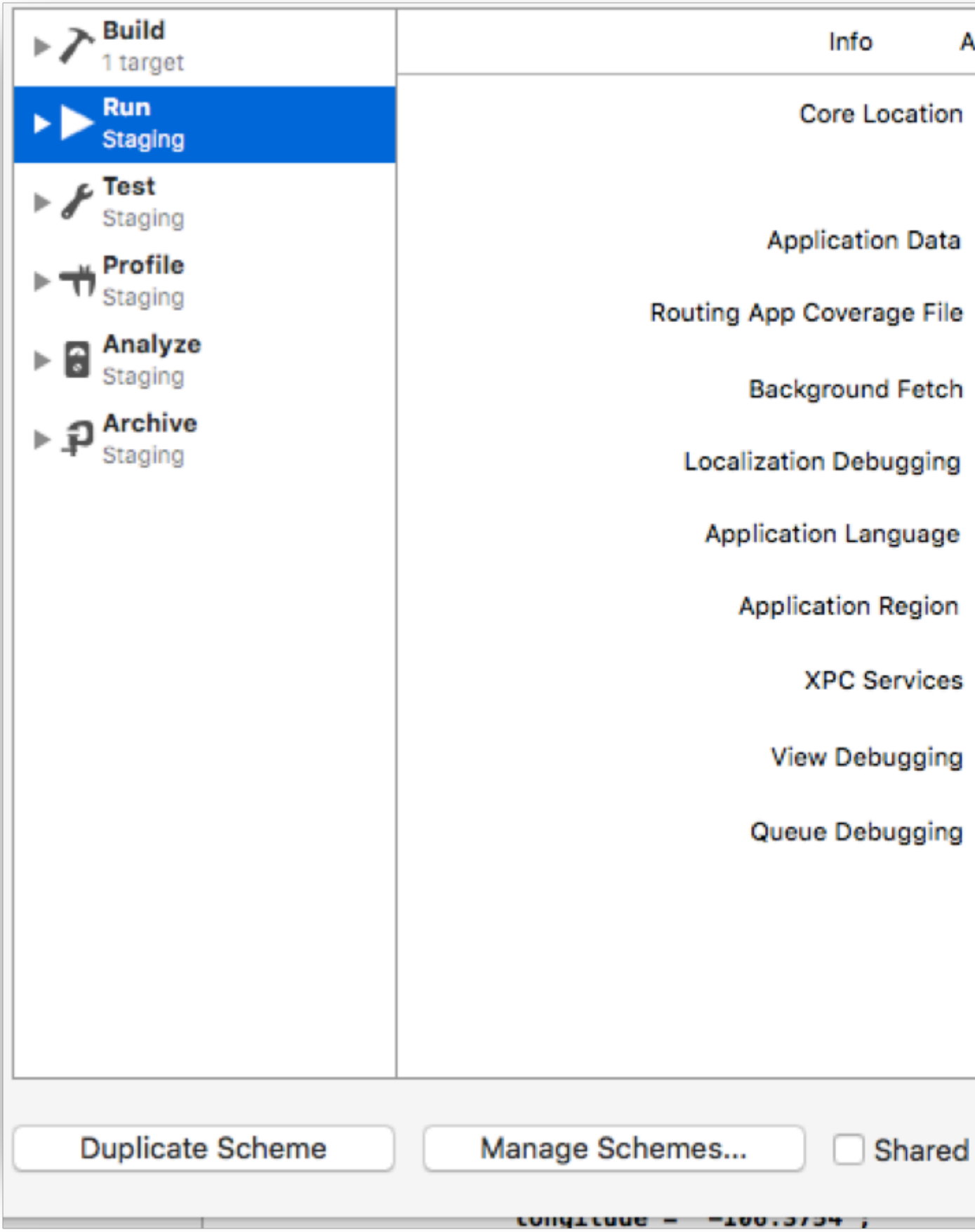
Examples

Il tuo file .gpx: 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>
```

Per impostare questa posizione:

1. Vai a Modifica schema.
2. Seleziona Esegui -> Opzioni.
3. Seleziona "Consenti simulazione posizione".
4. Selezionare il Nome file * .GPX dall'elenco a discesa "Posizione predefinita".



Leggi Simulazione della posizione utilizzando i file GPX iOS online:
https://riptutorial.com/it/ios/topic/9883/simulazione-della-posizione-utilizzando-i-file-gpx-ios

Capitolo 144: Sirikit

Osservazioni

Diversi tipi di richieste Siri

- Ride Booking (ad esempio, procurami un passaggio a New York tramite MyApp)
- Messaggistica (ad es. Invia un testo a John usando MyApp)
- Ricerca foto (ad esempio, cerca le foto spiaggia scattate la scorsa estate su MyApp)
- Pagamenti (es. Invia $ 20 a John per cena la scorsa notte usando MyApp)
- Chiamata VoIP (ad es. Chiama Mike sulla mia MyApp)
- Allenamenti (es. Avvia il mio allenamento giornaliero da MyApp)
- Clima e radio (appositamente progettati per CarPlay, ad es. Impostare il riscaldatore su 72 gradi)

Examples

Aggiunta di estensione Siri all'app

Per integrare le funzionalità di Siri nella tua app, dovresti aggiungere delle estensioni come faresti durante la creazione di un widget iOS 10 (la precedente estensione di visualizzazione di oggi) o una tastiera personalizzata.

Aggiungere capacità

1- Nelle impostazioni del progetto, seleziona il target dell'app iOS e vai alla scheda Capabilities

2- Abilitare la funzionalità Siri

Aggiungere l'estensione

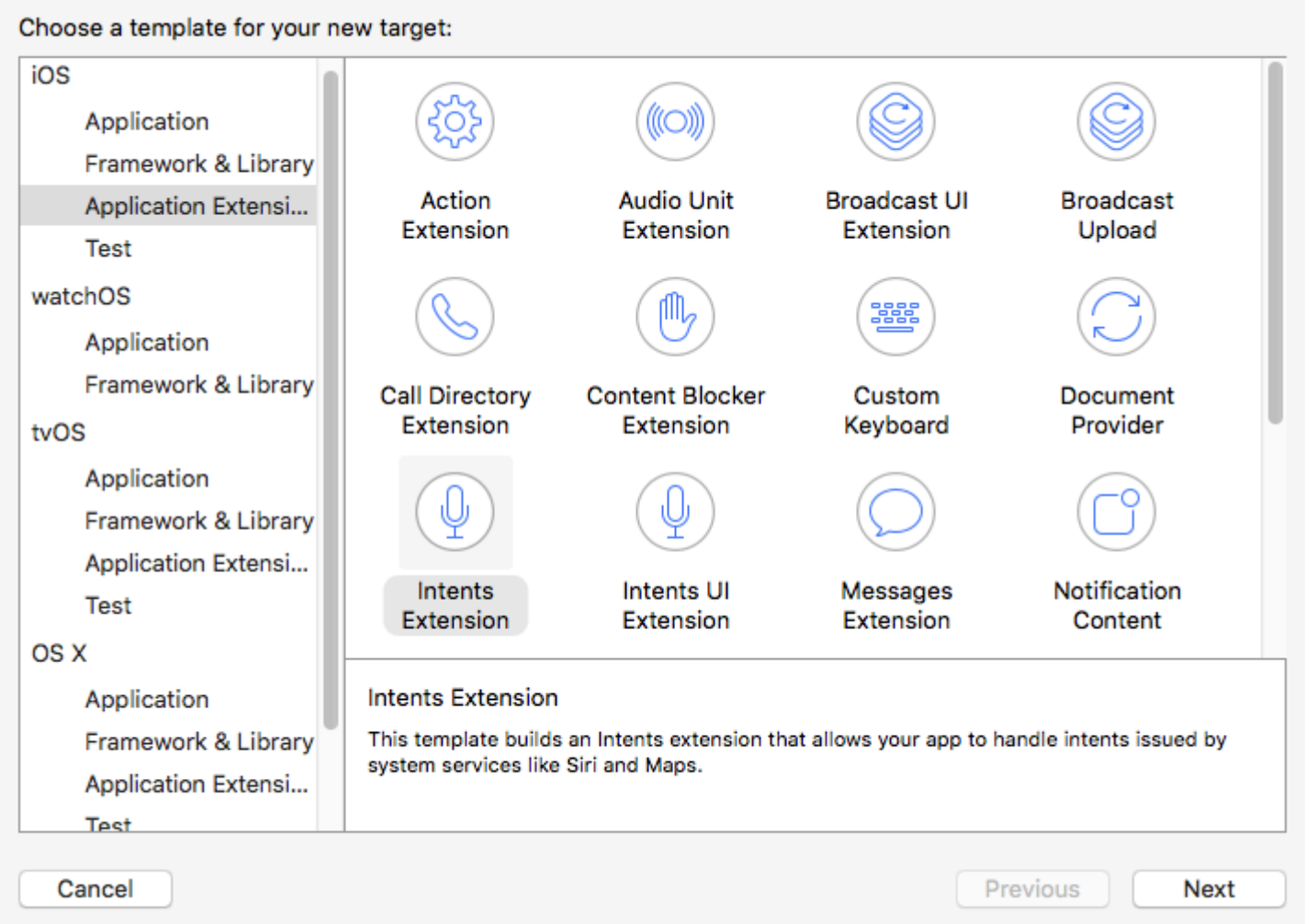
1- Vai a File -> Nuovo -> Target ...

2- Seleziona iOS -> Estensione applicazione dal riquadro di sinistra

3- Fare doppio clic su Intents Extension da destra

Secondo Apple:

Il modello Intents Extension crea un'estensione Intents che consente alla tua app di gestire gli intenti emessi dai servizi di sistema come Siri e Maps.



4- Scegli un nome e assicurati di selezionare "Includi l'estensione dell'interfaccia utente"

Effettuando questa procedura, vengono creati due nuovi target (Intents Extension e UI Extension) e, per impostazione predefinita, contengono il codice Workout Intent. Per diversi tipi di richieste Siri, vedere Note.

Nota

Ogni volta che vuoi eseguire il debug della tua estensione, basta selezionare lo schema Intent dagli schemi disponibili.

Nota

Non è possibile testare le app SiriKit nel simulatore. Invece, hai bisogno di un dispositivo reale.

Leggi Sirikit online: https://riptutorial.com/it/ios/topic/5869/sirikit

Capitolo 145: SLComposeViewController

Examples

SLComposeViewController per Twitter, Facebook, SinaWelbo e TencentWelbo

Objective-C

Per prima cosa aggiungi il `Social Framework` al progetto XCode.

Importa la classe `#import "Social/Social.h"` richiesto

Twitter con testo, immagine e link

```
//- - 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 con testo, immagine e link

```
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];
}
```

Sina Weibo

```
//- - 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];
}
```

Tencent Weibo

```
//- -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];
}
```

Leggi SLComposeViewController online:
https://riptutorial.com/it/ios/topic/7366/slcomposeviewcontroller

Capitolo 146: Sottolineatura del testo UILabel

Examples

Sottolineando un testo in un UILabel usando l'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];
```

Sottolineando un testo in UILabel usando 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
```

Leggi Sottolineatura del testo UILabel online: https://riptutorial.com/it/ios/topic/7219/sottolineatura-del-testo-uilabel

Capitolo 147: StoreKit

Examples

Ottieni informazioni sul prodotto localizzate dall'App Store

Ottieni informazioni sul prodotto localizzate da una serie di stringhe identificative del prodotto utilizzando `SKProductsRequest` :

```
import StoreKit

let productIdentifierSet = Set(["yellowSubmarine", "pennyLane"])
let productsRequest = SKProductsRequest(productIdentifiers: productIdentifierSet)
```

Per elaborare i prodotti dai `productsRequest` , è necessario assegnare un delegato alla richiesta che gestisce la risposta. Il delegato deve conformarsi al protocollo `SKProductsRequestDelegate` , il che significa che deve ereditare da `NSObject` (cioè qualsiasi oggetto `Foundation`) e implementare il metodo `productsRequest` :

```
class PaymentManager: NSObject, SKProductsRequestDelegate {

    var products: [SKProduct] = []

    func productsRequest(request: SKProductsRequest,
                         didReceiveResponse response: SKProductsResponse) {

        products = response.products

    }

}
```

Per avviare `productsRequest` , assegniamo `PaymentManager` come delegato della richiesta di prodotti e inviamo il metodo `start()` alla richiesta:

```
let paymentManager = PaymentManager()
productsRequest.delegate = paymentManager
productsRequest.start()
```

Se le richieste riusciranno, i prodotti saranno in `paymentManager.products` .

Leggi StoreKit online: https://riptutorial.com/it/ios/topic/6025/storekit

Capitolo 148: storyboard

introduzione

Normalmente, i controller di vista in uno storyboard vengono istanziati e creati automaticamente in risposta alle azioni definite all'interno dello storyboard stesso. Tuttavia, è possibile utilizzare un oggetto storyboard per istanziare il controller di visualizzazione iniziale in un file storyboard o creare un'istanza di altri controller di visualizzazione che si desidera presentare a livello di programmazione. Di seguito troverai esempi di entrambi i casi d'uso.

Examples

Inizializzare

```
//Swift
let storyboard = UIStoryboard(name: "Main", bundle: NSBundle.mainBundle())

//Objective-c
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:[NSBundle
mainBundle]];
```

Recupera ViewController iniziale

```
//Swift
let initialScreen = storyboard.instantiateInitialViewController()

//Objective-c
UIViewController *initailScreen = [storyboard instantiateInitialViewController];
```

Recupera ViewController

```
//Swift
let viewController = storyboard.instantiateViewControllerWithIdentifier("identifier")

//Objective-c
UIViewController *viewController = [storyboard
instantiateViewControllerWithIdentifier:@"identifier"];
```

Leggi storyboard online: https://riptutorial.com/it/ios/topic/3514/storyboard

Capitolo 149: Swift: modifica del rootViewController in AppDelegate per presentare il flusso principale o di accesso / onboarding

introduzione

È spesso utile presentare un'esperienza di prima esecuzione ai nuovi utenti della tua app. Questo potrebbe essere per un numero qualsiasi di motivi, come chiedere loro di accedere (se necessario per la tua situazione), spiegare come utilizzare l'App o semplicemente informarli di nuove funzionalità in un aggiornamento (come Notes, Photos e Music do in iOS11).

Osservazioni

In primo luogo, poiché hai a che fare con flussi multipli, è qui che gli storyboard possono essere usati efficacemente. Per impostazione predefinita, l'applicazione utilizza `Main.storyboard` per il flusso principale. Il tuo onboarding / flusso alternativo può essere contenuto in uno storyboard secondario, ad es. `Onboarding.storyboard`

Questo ha una serie di vantaggi:

- in un team di sviluppatori, il lavoro su ciascun flusso di utenti può essere separato
- controllo sorgente più chiaro (git)
- separazione degli interessi

Quando la tua app si avvia, puoi determinare quale flusso deve essere presentato. La logica per questo può essere contenuta nel tuo AppDelegate:

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    let isFirstRun = true // logic to determine goes here
    if isFirstRun {
        showOnboarding()
    }
    return true
}
```

Per mostrare il flusso di Onboarding, vale la pena considerare come vorresti gestire l'esperienza di licenziamento una volta che la persona che lo usa ha completato il viaggio e che è semanticamente corretto per ciò che stai cercando di creare.

approcci:

I due approcci principali sono:

1. Scambia il controller della vista radice della finestra principale dell'app
2. Presenta il flusso di bordo come un viaggio modale, sovrapponendo il flusso principale.

L'implementazione di questo dovrebbe essere contenuta in un'estensione di AppDelegate.

Examples

Opzione 1: scambia il controller di visualizzazione radice (buono)

Ci sono dei vantaggi nel cambiare il controller della vista radice, sebbene le opzioni di transizione siano limitate a quelle supportate da `UIViewAnimationOptions` , quindi a seconda di come desideri passare tra i flussi potrebbe significare che devi implementare una transizione personalizzata, che può essere complicata.

È possibile mostrare il flusso di `UIApplication.shared.keyWindow.rootViewController` semplicemente impostando `UIApplication.shared.keyWindow.rootViewController`

Il licenziamento viene gestito utilizzando `UIView.transition(with:)` e passando lo stile di transizione come `UIViewAnimationOptions` , in questo caso Cross Dissolve. (Anche Flip e ricci sono supportati).

Devi anche impostare la cornice della vista principale prima di ricollegarti ad essa, mentre la installi per la prima volta.

```
// 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)
        }
    }
}
```

Opzione 2: presentare il flusso alternativo in modo modale (migliore)

Nell'implementazione più semplice, il flusso di Onboarding può essere semplicemente presentato in un contesto modale, poiché semanticamente l'utente si trova su un singolo percorso.

> [Linee guida per l'interfaccia umana Apple - Modalità] [1]:
>
> Prendi in considerazione la possibilità di creare un contesto modale solo quando è fondamentale attirare l'attenzione di qualcuno, quando un'attività deve essere completata o abbandonata per continuare a utilizzare l'app o per salvare dati importanti.

La presentazione modale consente la semplice opzione di licenziamento alla fine del viaggio, con un po 'di cruft di controller di scambio.

Le transizioni personalizzate sono supportate anche nel modo standard, poiché utilizza l'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)
        }
    }
}
```

Leggi Swift: modifica del rootViewController in AppDelegate per presentare il flusso principale o di accesso / onboarding online: https://riptutorial.com/it/ios/topic/10880/swift--modifica-del-rootviewcontroller-in-appdelegate-per-presentare-il-flusso-principale-o-di-accesso---onboarding

Capitolo 150: SWRevealViewController

Osservazioni

L'uso della classe SWRevealViewController come navigazione principale potrebbe non sempre portare alla migliore esperienza utente. Se la barra laterale contiene solo 5 o meno voci (o il contenuto può essere compresso in 5 o meno voci), dovresti considerare l'utilizzo della barra delle schede predefinita.

La barra delle schede è intuitiva e consente all'utente di cambiare rapidamente tra viste / contesti. D'altra parte, la navigazione della barra laterale può eseguire più azioni rispetto alla commutazione della vista / contesto e occupa meno spazio quando viene compressa.

Per maggiori informazioni consulta le linee guida per l'interfaccia umana iOS di Apple.

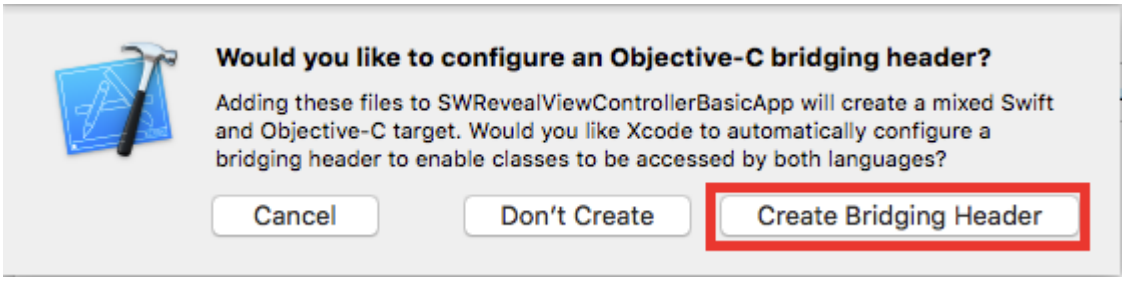
Examples

Impostazione di un'app di base con SWRevealViewController

Crea un'applicazione di base con modello di applicazione vista singola con swift come lingua

Aggiungi `SWRevealViewController.h` e `SWRevealViewController.m`

quindi fare clic sul pulsante Crea intestazione ponte

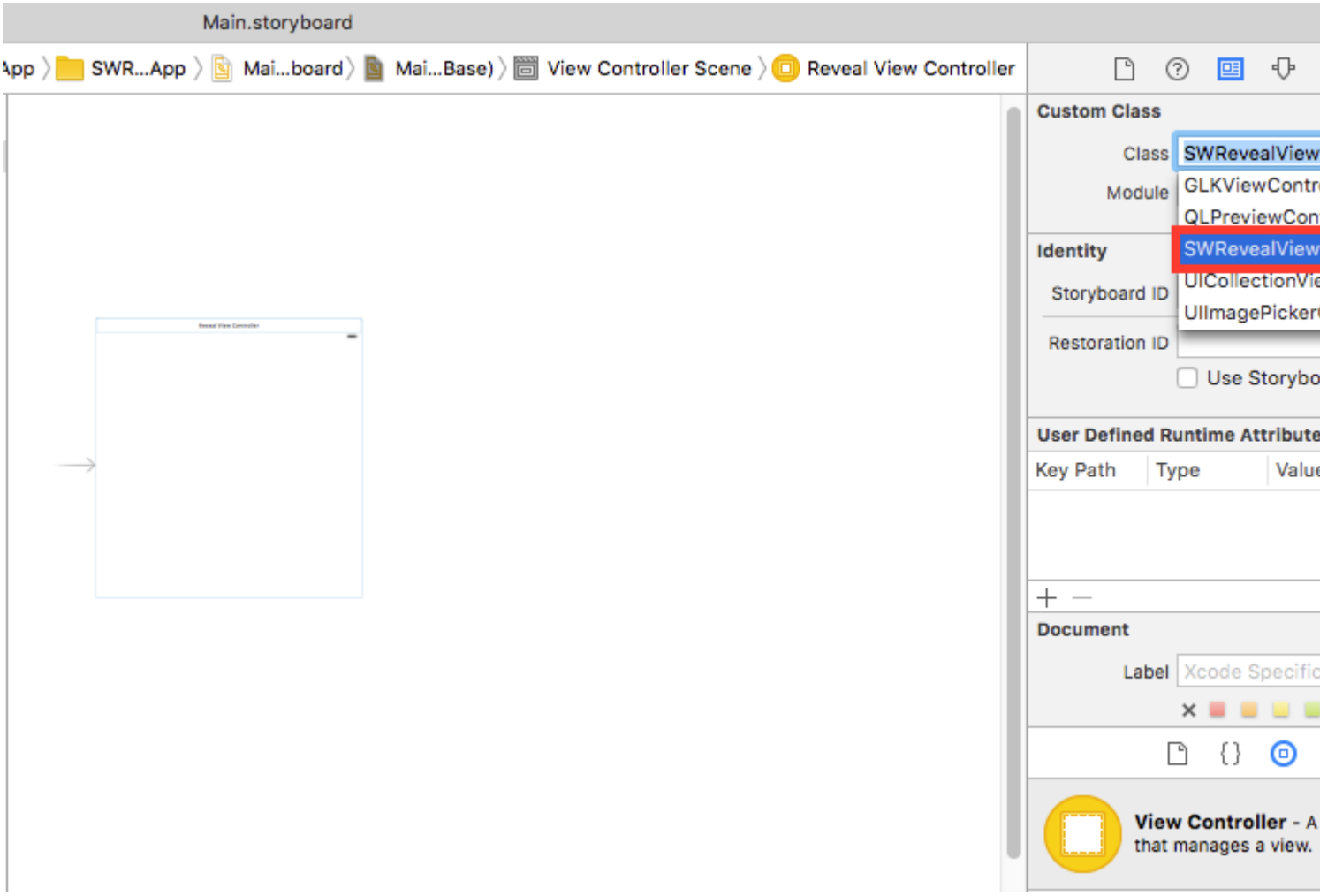


e aggiungi

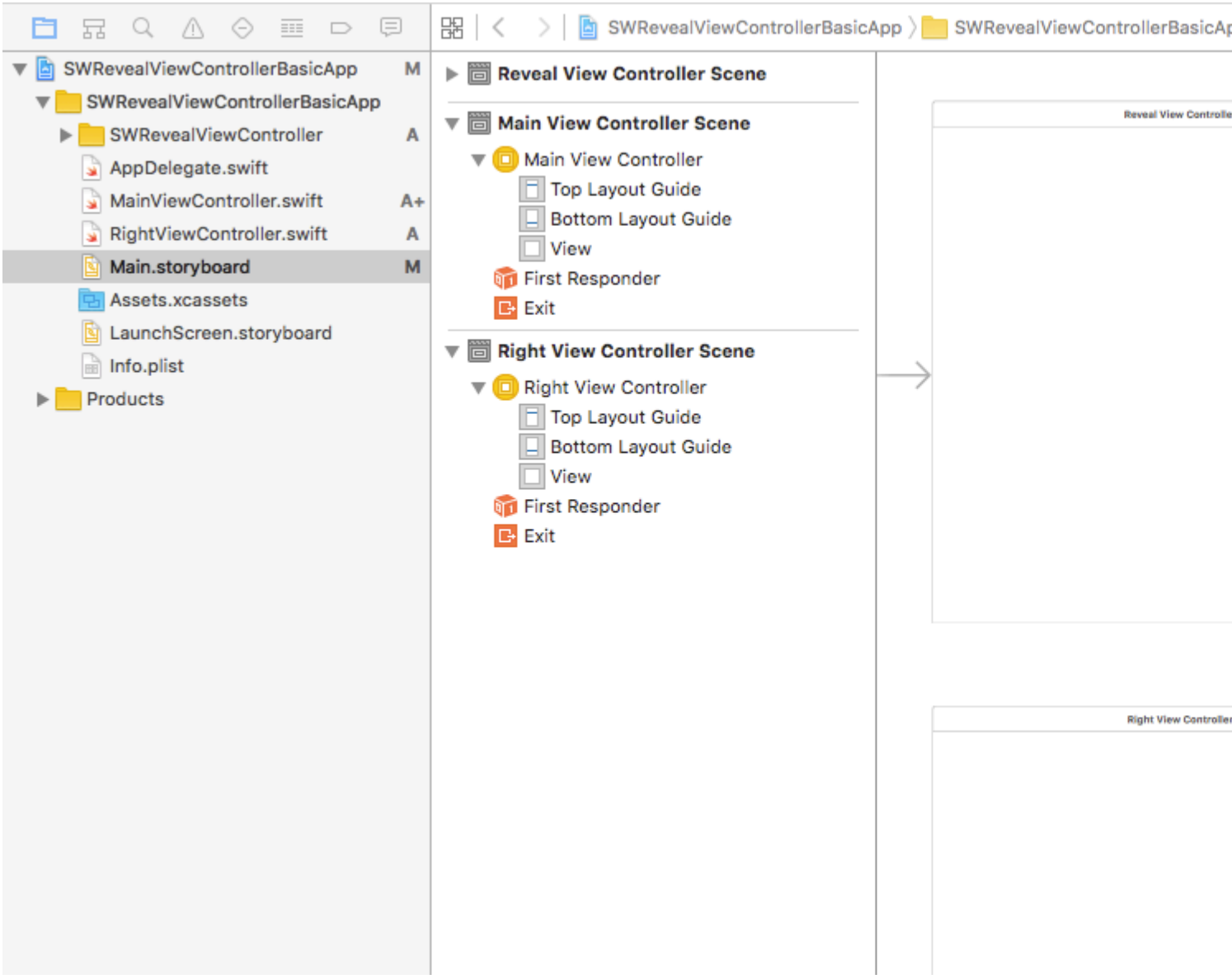
```
#import "SWRevealViewController.h"
```

sull'intestazione Bridging

Quindi selezionare viewController su storyboard e cambiare classe in `SWRevealViewController`

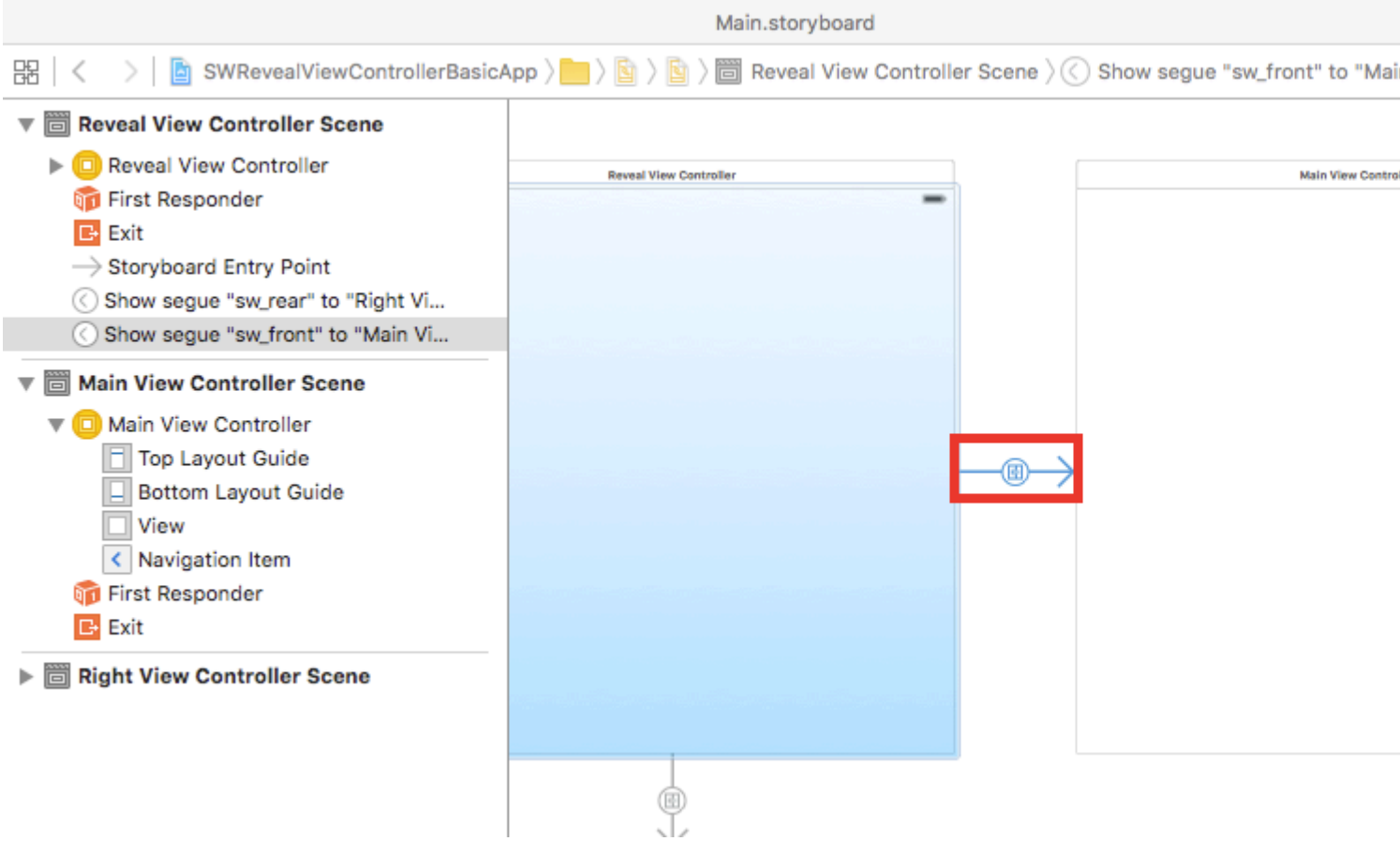


Quindi rinominare viewController sui file su MainViewController e aggiungere nuovo ViewController con il nome RightViewController



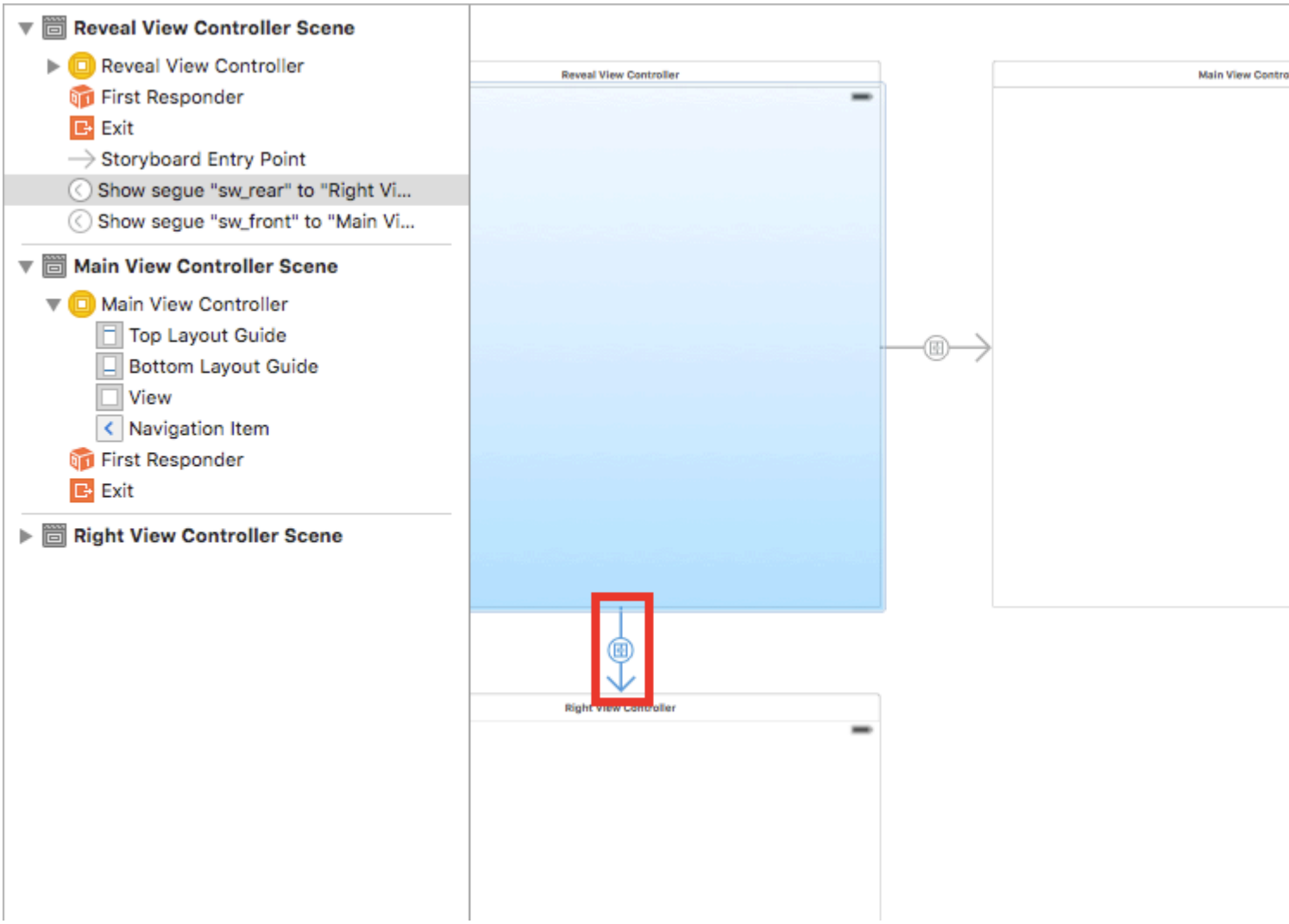
quindi aggiungiamo due seguiti da SWRevealViewController a MainViewController e da SWRevealViewController a RightViewController, quindi dobbiamo selezionare il primo (da SWRevealViewController a MainViewController) e modificare le proprietà

sul set di identificatori `sw_front` sul set di classi `SWRevealViewControllerSegueSetController`



dopo questo dobbiamo fare lo stesso con il seguito (da SWRevealViewController a RightViewController)

sul set di identificatori `sw_rear` sul set di classi `SWRevealViewControllerSegueSetController`



quindi su MainViewController aggiungi questa linea sul metodo `viewDidLoad`

```
self.view.addGestureRecognizer(self.revealViewController().panGestureRecognizer());
```

E questo è tutto, hai un'app di base con SWRevealViewController integrato, puoi scorrere verso destra per mostrare il `RightViewController` come menu laterale

Leggi SWRevealViewController online:
https://riptutorial.com/it/ios/topic/4614/swrevealviewcontroller

Capitolo 151: Taglia una creatura UII in un cerchio

Examples

Taglia un'immagine in un cerchio - Obiettivo C

import `#include <math.h>`

Il codice in viewDidLoad o loadView dovrebbe apparire qualcosa di simile a questo

```
- (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)];
}
```

Infine la funzione che fa il sollevamento pesante `circularScaleAndCropImage` è come definito di seguito

```
- (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;
}
```

SWIFT 3 Esempio

```
 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)))
    }
```

Infine la funzione che fa il sollevamento pesante circularScaleAndCropImage è come definito di seguito

```
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!
    }
```

Leggi Taglia una creatura UII in un cerchio online: https://riptutorial.com/it/ios/topic/7222/taglia-una-creatura-uii-in-un-cerchio

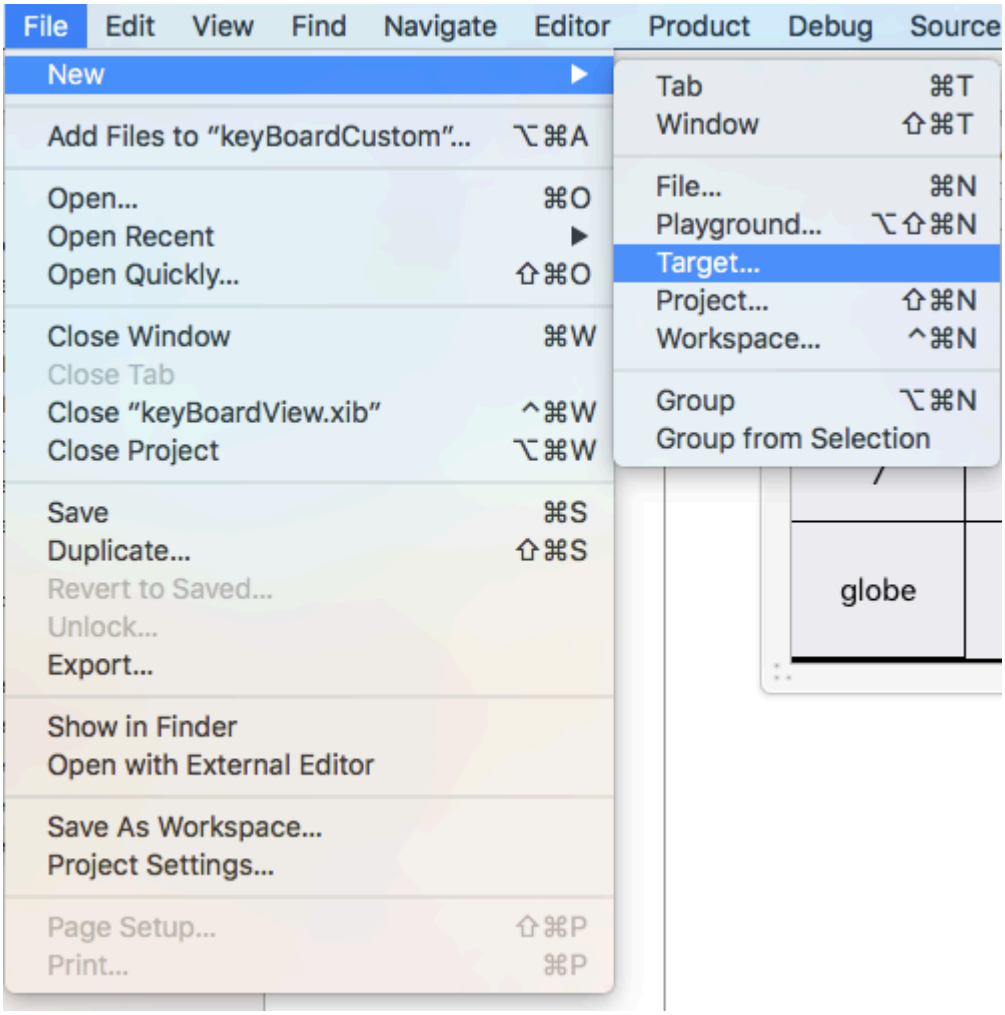
Capitolo 152: Tastiera personalizzata

Examples

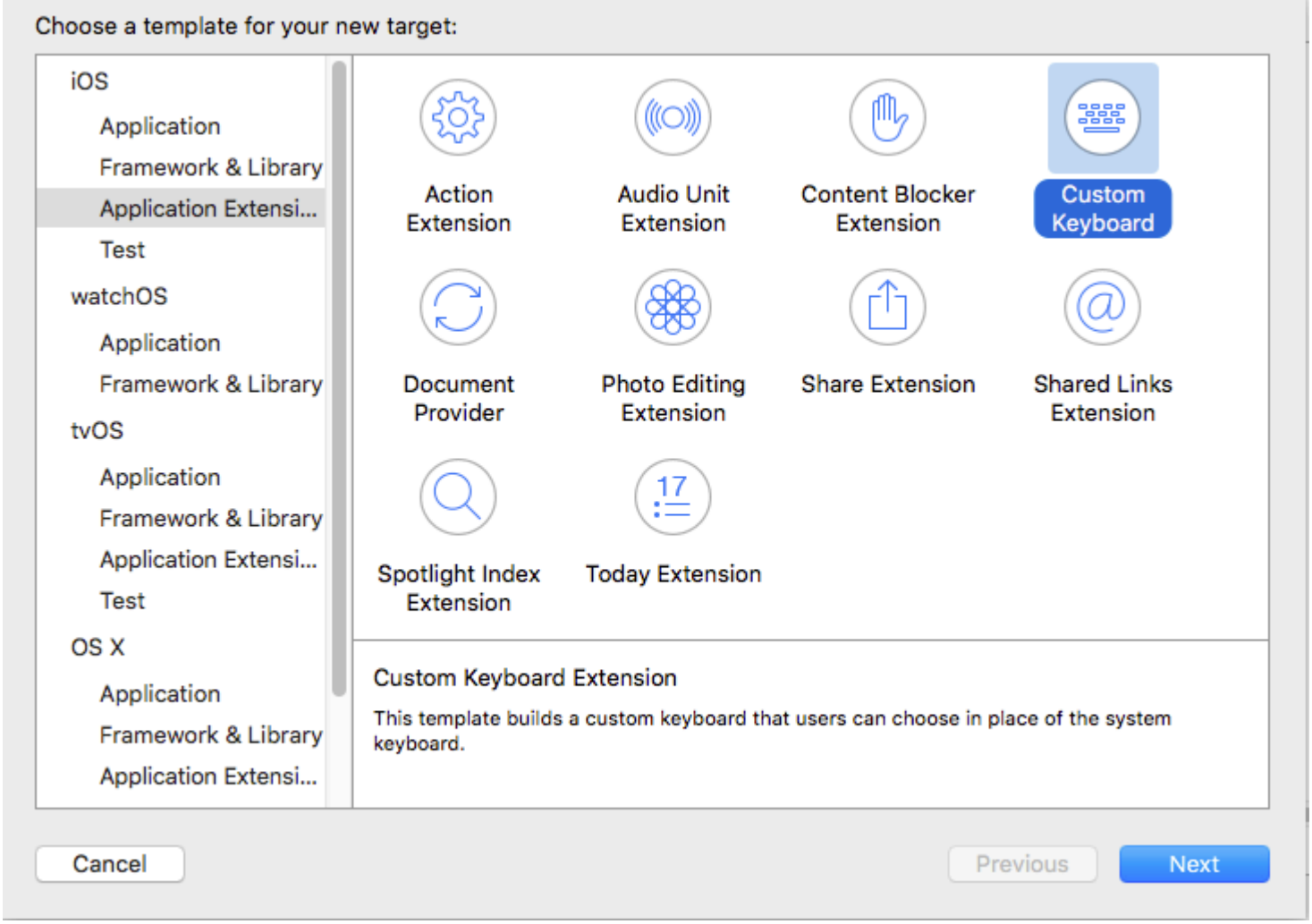
Esempio di KeyBoard personalizzato

Objective-C e Xib

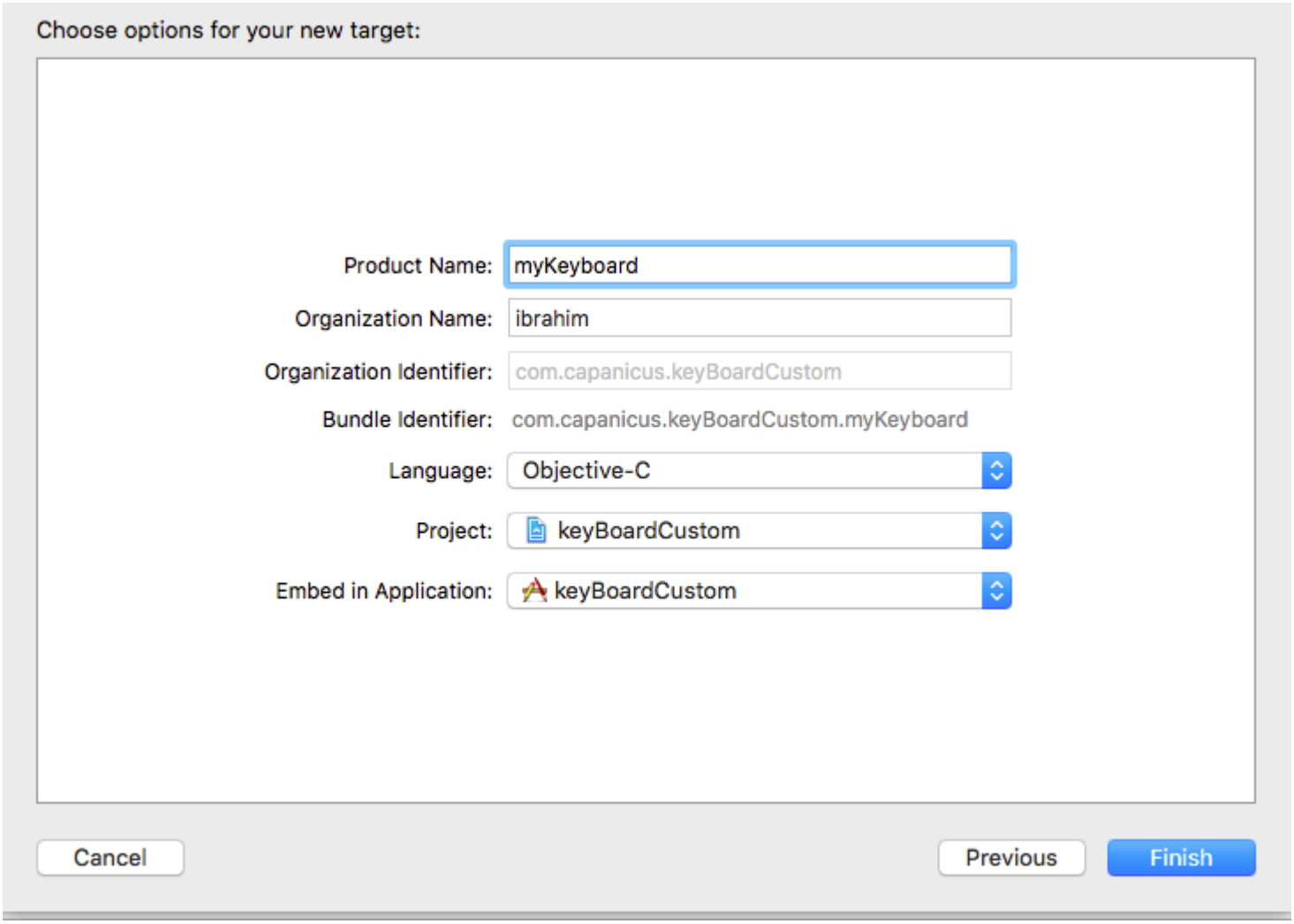
Aggiungi un obiettivo a un progetto XCode esistente



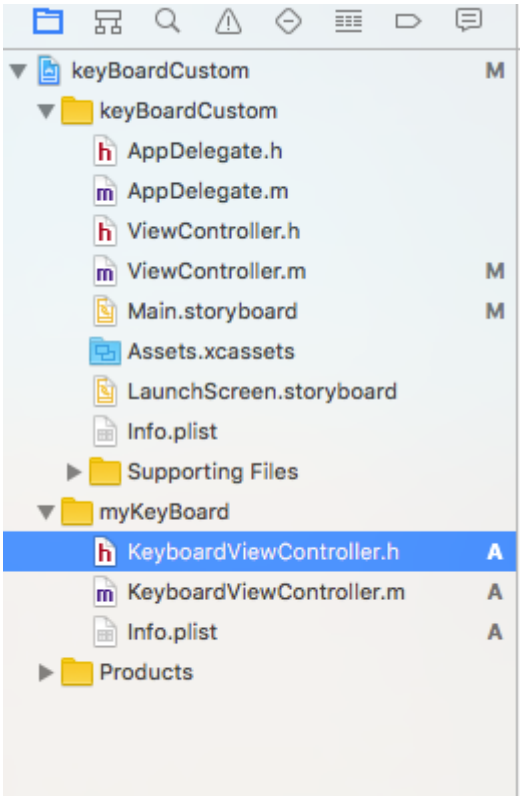
Nel Aggiungi destinazione selezionare Personalizza KeyBoard



Aggiungi il target in questo modo:

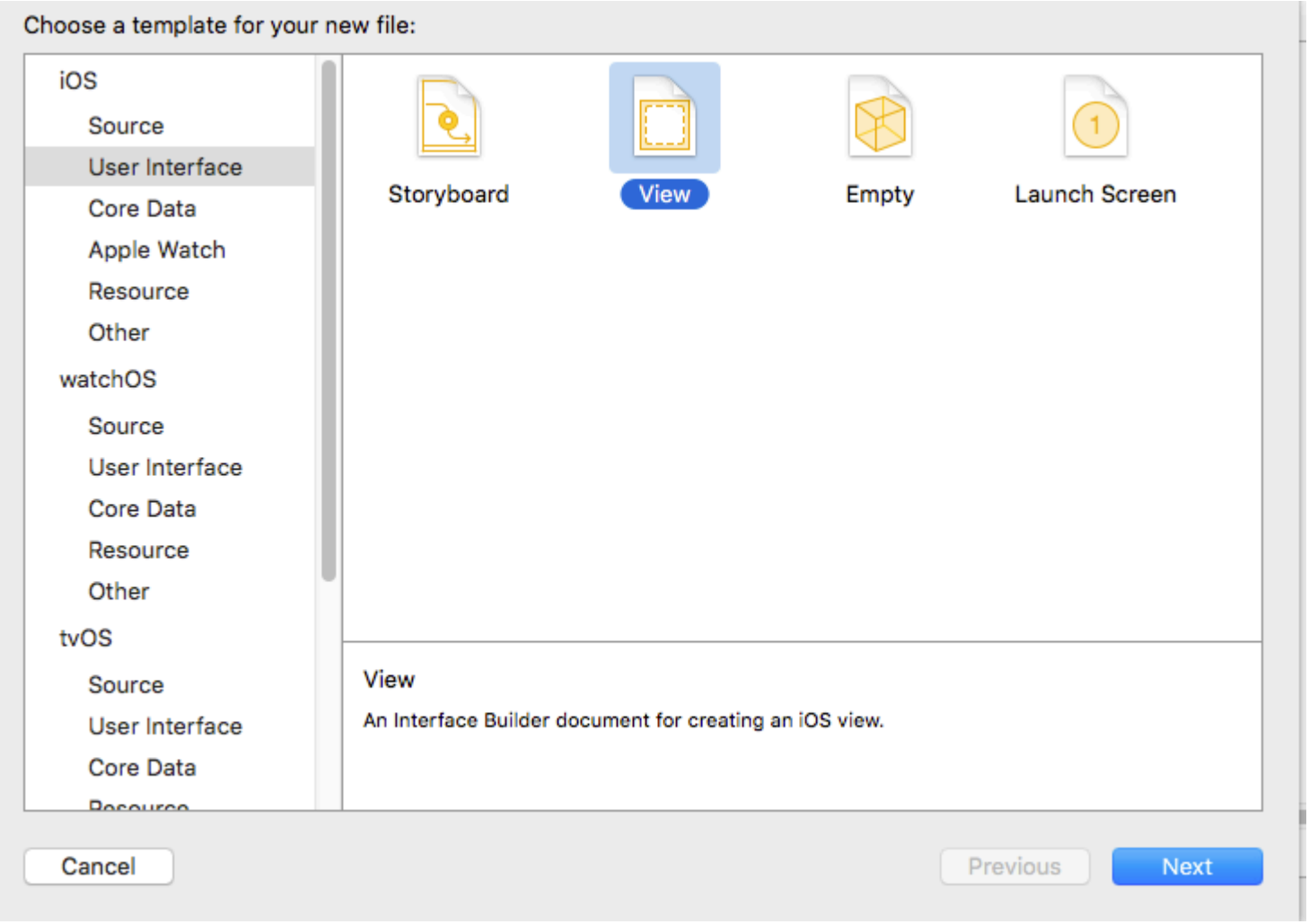


La directory del file di progetto dovrebbe essere simile a questa

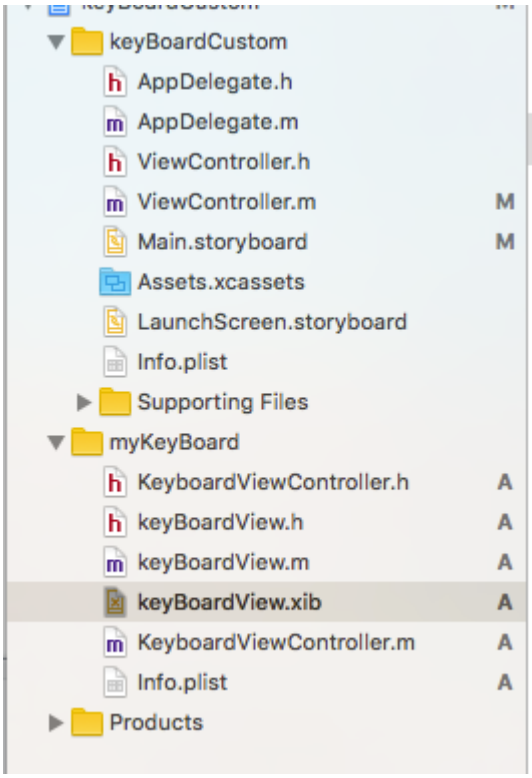


Qui myKeyBoard è il nome dell'obiettivo aggiunto

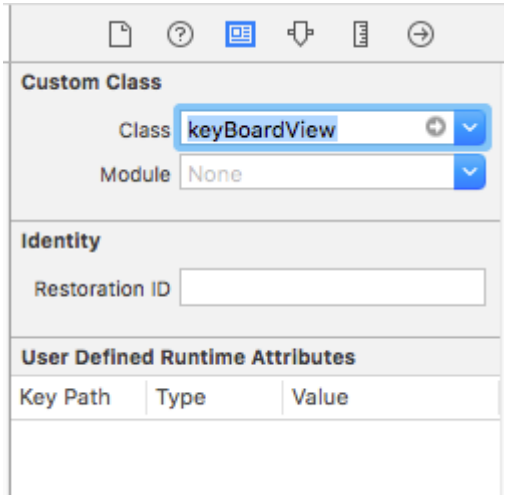
Aggiungi un nuovo file Cocoatouch di tipo di tipo UIView e aggiungi un file di interfaccia



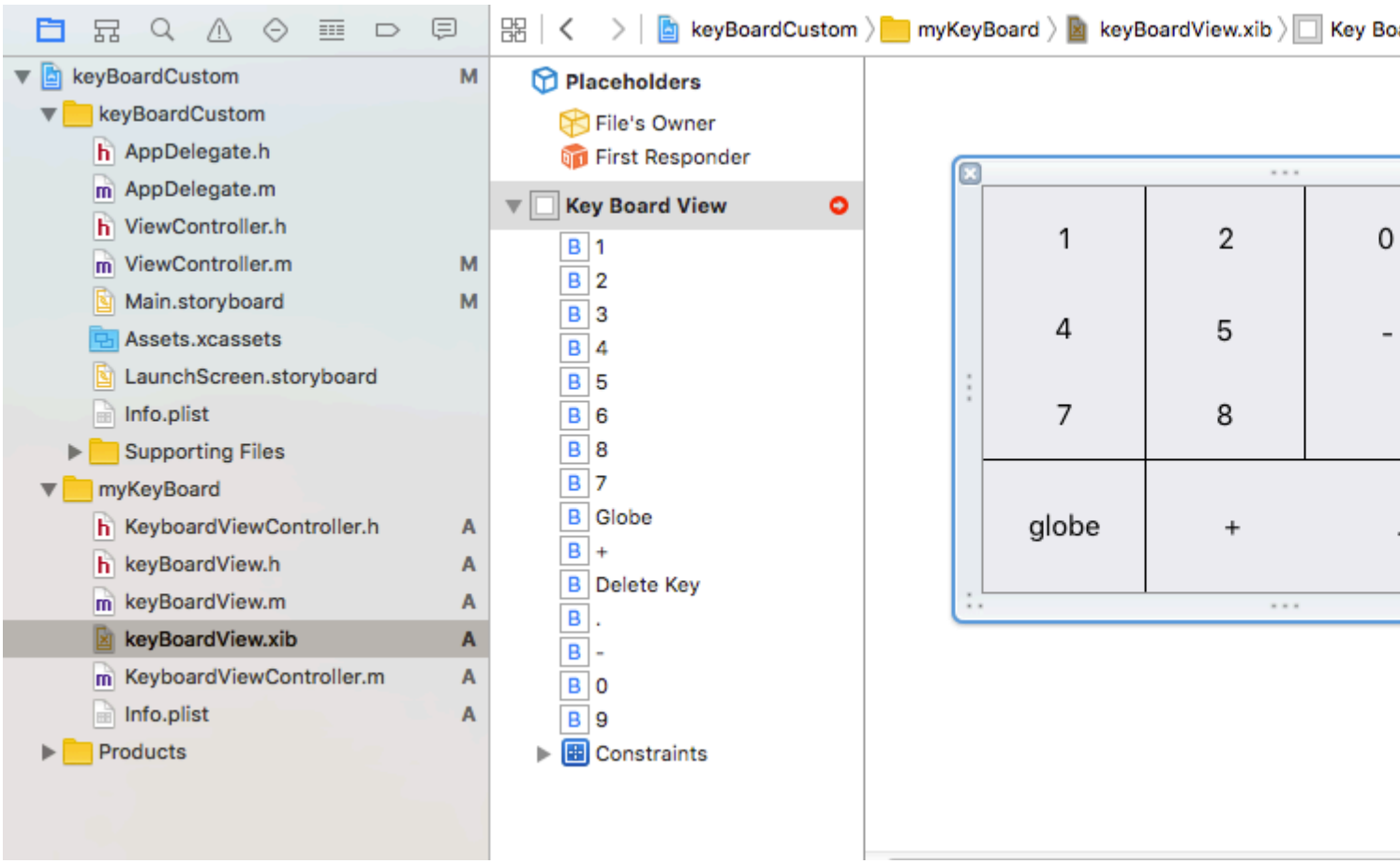
Finalmente la directory del tuo progetto dovrebbe assomigliare a questo



rendere `keyBoardView.xib` una sottoclasse di `keyBoardView`



Crea un'interfaccia nel file `keyBoardView.xib`



Effettua le connessioni da `keyBoardView.xib` al file `keyBoardView.h`

`keyBoardView.h` dovrebbe essere simile

```
#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
```

Nel file `keyBoardViewController.h` import `#import "keyBoardView.h"`

Dichiarare una proprietà per la tastiera `@property (strong, nonatomic)keyBoardView *keyboard;`

Commenta il

```
@property (nonatomic, strong) UIButton *nextKeyboardButton and all the code associated with it
```

La funzione viewDidLoad () di KeyboardViewController.m dovrebbe essere simile a questa

```
- (void)viewDidLoad {
```

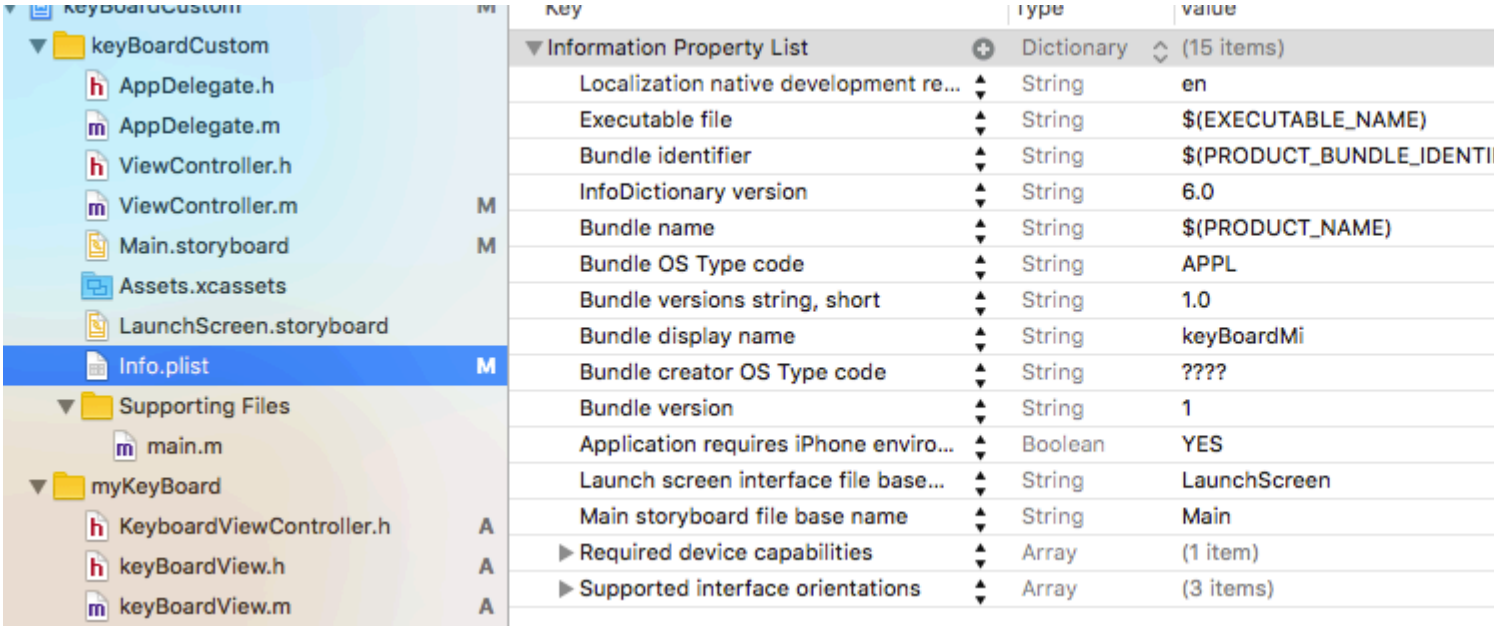
```
    [super viewDidLoad];
    self.keyboard=[[[NSBundle mainBundle]loadNibNamed:@"keyBoardView" owner:nil
options:nil]objectAtIndex:0];
      self.inputView=self.keyboard;
    [self addGestureToKeyboard];

    // 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;
}
```

Le funzioni `addGestureToKeyboard` , `pressDeleteKey` , `keyPressed` sono definiti di seguito

```
-(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]];
}
```

Esegui l'applicazione principale e vai su Impostazioni-> Generale-> Tastiera-> Aggiungi nuova tastiera-> e aggiungi la tastiera dalla sezione tastiera di terze parti (Il nome della tastiera visualizzato sarà KeyBoardCustom)

Il nome della tastiera può essere modificato aggiungendo una chiave denominata `Bundle display name` e nel Value String Value immettere il nome desiderato per la tastiera del progetto principale.



Puoi anche guardare questo video di Youtube

Leggi Tastiera personalizzata online: https://riptutorial.com/it/ios/topic/7358/tastiera-personalizzata

Capitolo 153: Test dell'interfaccia utente

Sintassi

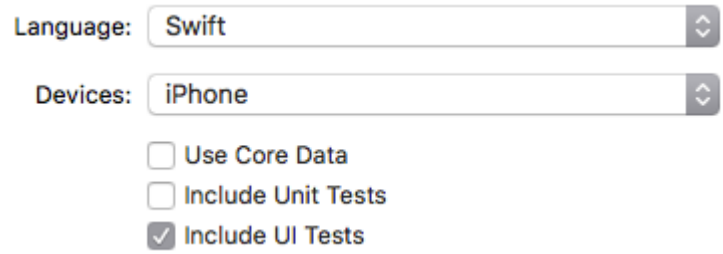
- XCUIApplication () // Proxy per un'applicazione. Le informazioni che identificano l'applicazione sono specificate nelle impostazioni del target Xcode come "Target Application".
- XCUIElement () // Un elemento dell'interfaccia utente in un'applicazione.

Examples

Aggiunta di file di test a Xcode Project

Quando si crea il progetto

Dovresti selezionare "Includi test UI" nella finestra di dialogo di creazione del progetto.

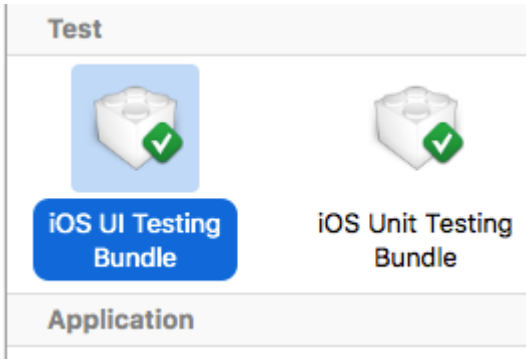


Dopo aver creato il progetto

Se ti sei perso il controllo `UI target` durante la creazione del progetto, puoi sempre aggiungere il target di prova in un secondo momento.

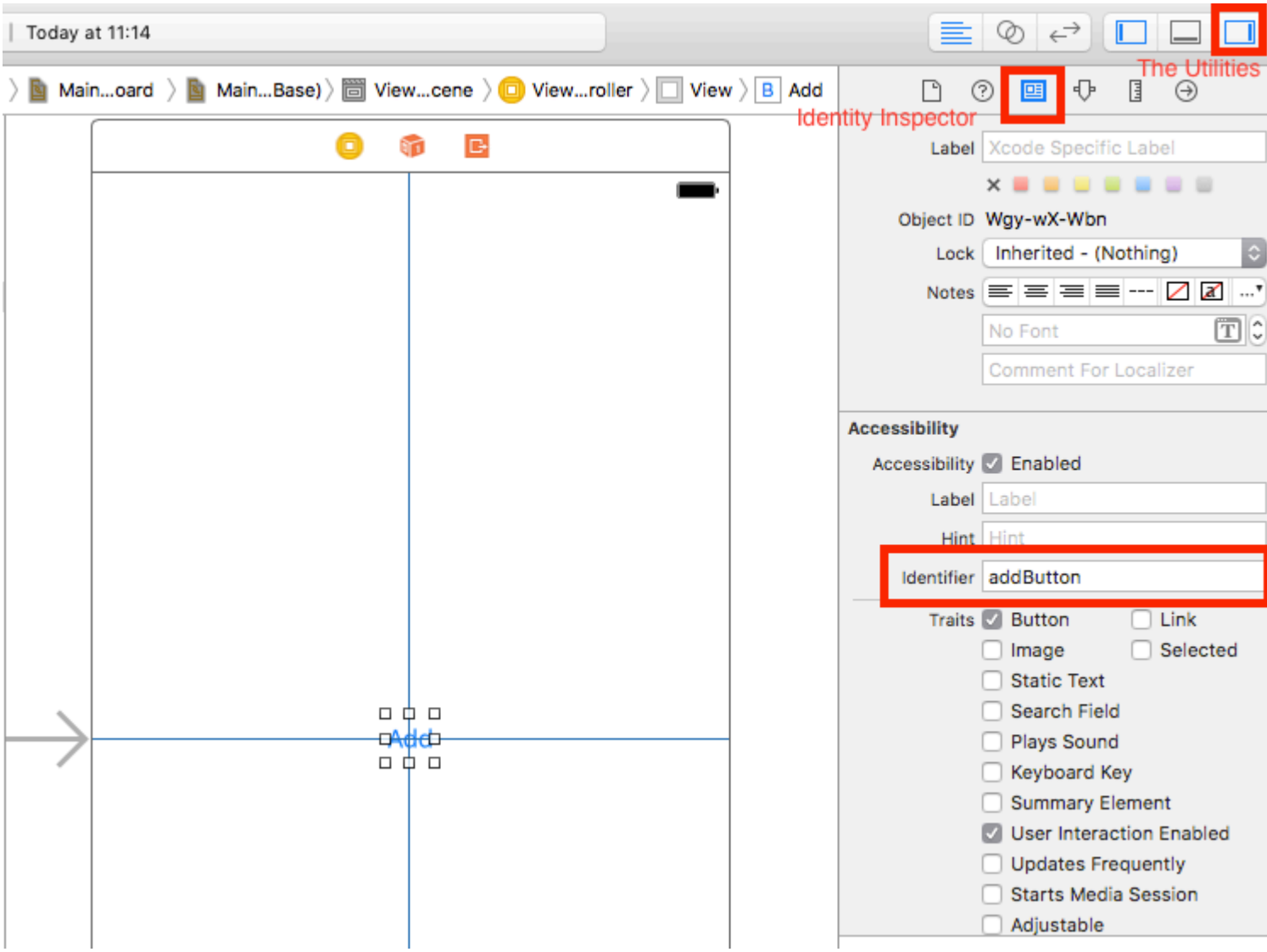
Setps:

- Mentre il progetto è aperto, vai su `File` -> `New` -> `Target`
- Trova `iOS UI Testing Bundle`



Quando l'accessibilità è abilitata nelle utilità

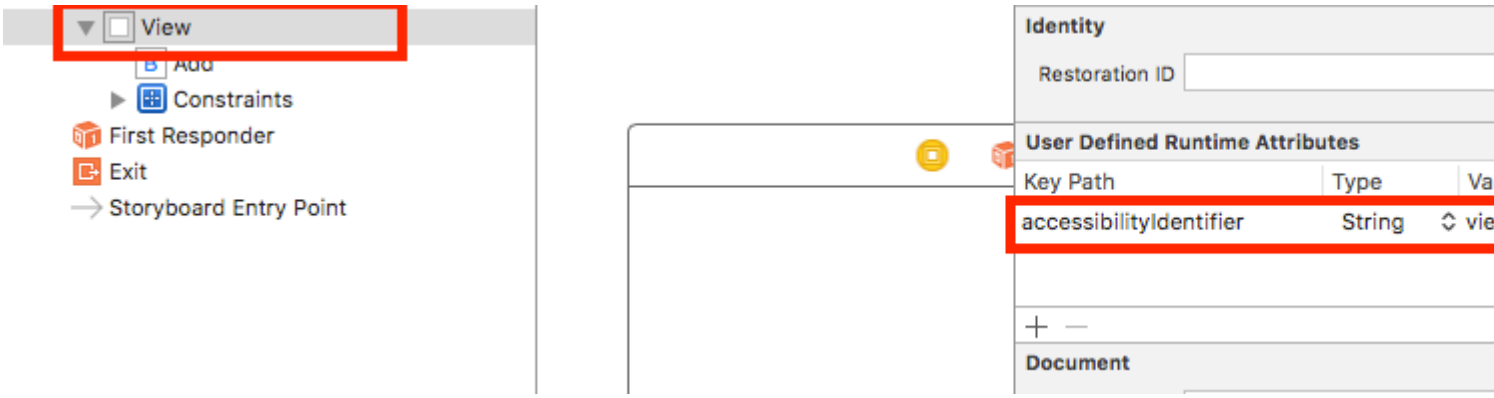
- Seleziona lo `storyboard` .
- Espandi `the Utilities`
- Seleziona `Identity Inspector`
- Seleziona il tuo elemento nello storyboard
- Aggiungi un nuovo identificatore di accessibilità (in esempio `addButton`)



Quando l'accessibilità è disabilitata nelle utilità

- Seleziona lo `storyboard` .
- Espandi `the Utilities`
- Seleziona `Identity Inspector`
- Seleziona il tuo elemento nello storyboard
- Aggiungi attributo in `User Defined Runtime Attributes`
- Per il tipo di `Key Path` : `accessibilityIdentifier`

- Per `Type` - `String
- Per `Value` : nuovo identificatore di accessibilità per il tuo elemento (nella `view` esempio)



Impostazione nel file 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()
    }
}
```

In `[ ]` aggiungi identificatore di accessibilità per elemento.

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"]
```

Elementi 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

- normale UITextField

```
let textField = app.textFields["textField"]
```

- password UITextField

```
let passwordTextField = app.secureTextFields["passwordTextField"]
```

UITextView

```
let textView = app.textViews["textView"]
```

UISwitch

```
let switch = app.switches["switch"]
```

avvisi

```
let alert = app.alerts["About yourself"] // Title of presented alert
```

Disattiva le animazioni durante il test dell'interfaccia utente

In un test puoi disabilitare le animazioni aggiungendo in `setUp` :

```
    app.launchEnvironment = ["animations": "0"]
```

Dove `app` è istanza di XCUIApplication.

Pranzo e terminare l'applicazione durante l'esecuzione

Applicazione del pranzo per i test

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

    let app = XCUIApplication()

    app.launch()
}
```

Termina l'applicazione

```
func testStacOverFlowApp() {

    app.terminate()
}
```

Ruota i dispositivi

Il dispositivo può essere ruotato cambiando l' `orientation` in `XCUIDevice.shared().orientation` :

```
XCUIDevice.shared().orientation = .landscapeLeft
XCUIDevice.shared().orientation = .portrait
```

Leggi Test dell'interfaccia utente online: https://riptutorial.com/it/ios/topic/7526/test-dell-interfaccia-utente

Capitolo 154: Tipo dinamico

Osservazioni

```
// Content size category constants
UIContentSizeCategoryExtraSmall
UIContentSizeCategorySmall
UIContentSizeCategoryMedium
UIContentSizeCategoryLarge
UIContentSizeCategoryExtraLarge
UIContentSizeCategoryExtraExtraLarge
UIContentSizeCategoryExtraExtraExtraLarge

// Accessibility sizes
UIContentSizeCategoryAccessibilityMedium
UIContentSizeCategoryAccessibilityLarge
UIContentSizeCategoryAccessibilityExtraLarge
UIContentSizeCategoryAccessibilityExtraExtraLarge
UIContentSizeCategoryAccessibilityExtraExtraExtraLarge
```

Examples

Ottieni la dimensione del contenuto attuale

veloce

```
UIApplication.sharedApplication().preferredContentSizeCategory
```

Objective-C

```
[UIApplication sharedApplication].preferredContentSizeCategory;
```

Restituisce una costante della categoria di dimensione del contenuto o una costante della categoria di dimensioni del contenuto di accessibilità.

Notifica di modifica delle dimensioni del testo

È possibile registrarsi per le notifiche di quando viene cambiata la dimensione del testo del dispositivo.

veloce

```
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];
```

L'oggetto `userInfo` notifica contiene la nuova dimensione in `UIContentSizeCategoryNewValueKey` .

Dimensione dei caratteri di tipo dinamico corrispondente in WKWebView

WKWebView ridimensiona i caratteri sui contenuti Web in modo che una pagina web di dimensioni standard si adatti al fattore di forma del dispositivo. Se si desidera che il testo Web in verticale e orizzontale sia di dimensioni simili alle dimensioni di lettura preferite dall'utente, è necessario impostarlo in modo esplicito.

veloce

```
// 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
}
```

Gestione delle modifiche alle dimensioni del testo preferite senza notifiche su iOS 10

`UILabel` , `UITextField` e `UITextView` hanno una nuova proprietà a partire da iOS 10 per ridimensionare automaticamente il carattere quando un utente modifica la dimensione di lettura preferita denominata `adjustsFontForContentSizeCategory` .

veloce

```
@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.
}
```

Leggi Tipo dinamico online: https://riptutorial.com/it/ios/topic/4466/tipo-dinamico

Capitolo 155: Tocco 3D

Examples

Tocco 3D con Swift

Il tocco 3D è stato introdotto con iPhone 6s Plus. Ci sono due comportamenti aggiunti con questo nuovo livello di interfaccia: Peek e Pop.

Peek e Pop in poche parole

Peek - Premi forte

Pop: stampa molto difficile

Verifica del supporto 3D

Dovresti controllare se il dispositivo ha un supporto touch 3D. Puoi farlo controllando il valore della proprietà *forceTouchCapability* di un oggetto *UITraitCollection* . UITraitCollection descrive l'ambiente di interfaccia iOS per la tua app.

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

Implementare il delegato

È necessario implementare i due metodi di *UIViewControllerPreviewingDelegate* nella classe. Uno dei metodi è per *sbirciare* e l'altro è per il comportamento *pop* .

Il metodo da implementare per la *sbirciata* è l' *anteprima diContext* .

```
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

}
```

Il metodo da implementare per il *pop* è l' *anteprima diContext* . :)

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

    let balanceViewController = viewControllerToCommit as! <YourViewController>

    // Do the stuff

    navigationController?.pushViewController(balanceViewController, animated: true)

}
```

Come puoi vedere sono metodi sovraccaricati. È possibile utilizzare 3D Touch in qualsiasi modo implementando questi metodi.

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 Touch Objective-C Esempio

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];
}
```

Leggi Tocco 3D online: https://riptutorial.com/it/ios/topic/6705/tocco-3d

Capitolo 156: Tutorial AirPrint in iOS

Examples

Stampa di AirPrint Banner Text

Objective-C

Aggiungere il delegato e un formattatore di testo al file `ViewController.h`

```
@interface ViewController : UIViewController <UIPrintInteractionControllerDelegate> {
    UISimpleTextPrintFormatter *_textFormatter;
}
```

Nel file `ViewController.m` definisci le seguenti costanti

```
#define DefaultFontSize 48
#define PaddingFactor 0.1f
```

La funzione che stampa il testo è la seguente: -

```
-(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
}
```

La funzione delegato che imposta la pagina di stampa: -

```
- (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;
}
```

Leggi Tutorial AirPrint in iOS online: https://riptutorial.com/it/ios/topic/7395/tutorial-airprint-in-ios

Capitolo 157: UIActivityViewController

Parametri

Nome del parametro	Descrizione
activityItems	Contiene una matrice di oggetti per eseguire l'attività. Questo array non deve essere nullo e deve contenere almeno un oggetto.
applicationActivities	Un array di oggetti UIActivity che rappresentano i servizi personalizzati supportati dall'applicazione. Questo parametro può essere nullo.

Examples

Inizializzazione del controller Vista attività

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];
```

veloce

```
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)
```

Leggi UIActivityViewController online:
https://riptutorial.com/it/ios/topic/2889/uiactivityviewcontroller

Capitolo 158: UIAlertController

Osservazioni

> Un oggetto `UIAlertController` visualizza un messaggio di avviso per l'utente. Questa classe sostituisce le classi `UIActionSheet` e `UIAlertView` per la visualizzazione degli avvisi. Dopo aver configurato il controller degli avvisi con le azioni e lo stile che desideri, presentalo utilizzando `presentViewController:animated:completion:` method.

Dalla documentazione di Apple

UIAlertController in Swift

Examples

AlertViews con UIAlertController

`UIAlertView` e `UIActionSheet` sono deprecati in `iOS 8` e `iOS 8` successive. Quindi Apple ha introdotto un nuovo controller per `AlertView` e `ActionSheet` chiamato `UIAlertController` , cambiando il `preferredStyle` , è possibile passare da `AlertView` `ActionSheet` . Non esiste un metodo di delega per questo perché tutti gli eventi dei pulsanti vengono gestiti nei relativi blocchi.

Semplice AlertView

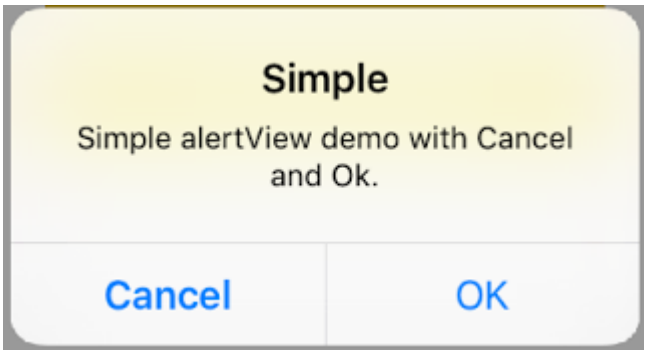
Swift:

```
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];
```



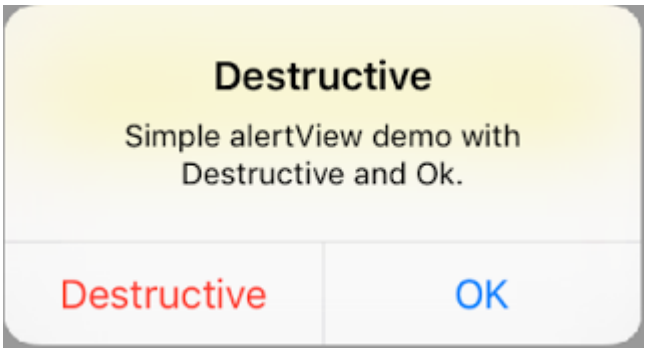
AlertView distruttivo

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];
```



Popup pop-up temporaneo

Buono per notifiche veloci che non richiedono interazione.

veloce

```
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)
     }
}
```

Aggiunta di campi di testo in UIAlertController come una finestra di dialogo

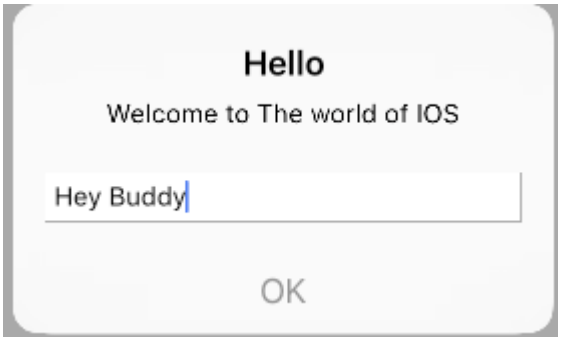
veloce

```
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];
```



Fogli d'azione con UIAlertController

Con `UIAlertController` , i fogli di azione come il `UIActionSheet` deprecato vengono creati con la stessa API utilizzata per AlertViews.

Semplice foglio d'azione con due pulsanti

veloce

```
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];
```

Crea i pulsanti "Annulla" e "OK"

veloce

```
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
    }];
```

E aggiungili al foglio d'azione:

veloce

```
alertController.addAction(cancelAction)
alertController.addAction(okAction)
```

Objective-C

```
[alertController addAction:cancelAction];
[alertController addAction:okAction];
```

Ora presenta `UIAlertController` :

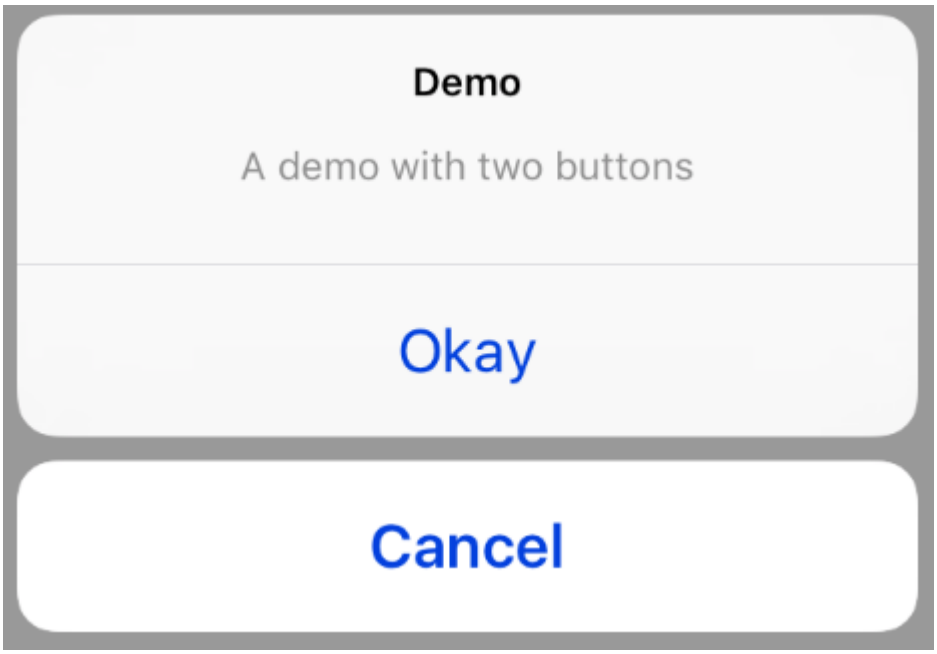
veloce

```
self.present(alertController, animated: true, completion: nil)
```

Objective-C

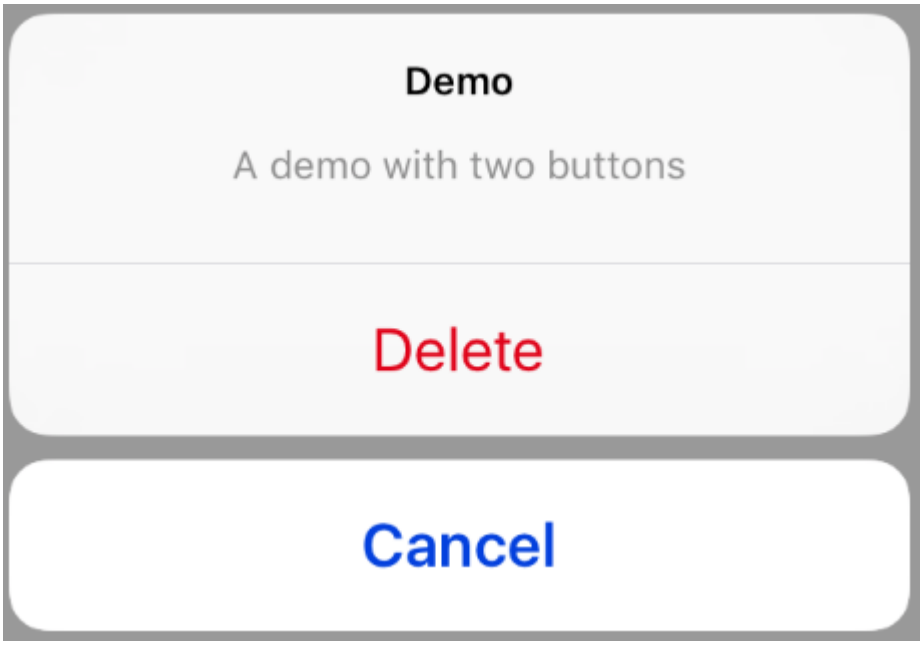
```
[self presentViewController:alertController animated: YES completion: nil];
```

Questo dovrebbe essere il risultato:



Foglio d'azione con pulsante distruttivo

Usando `UIAlertActionStyle .destructive` per `UIAlertAction` creerai un pulsante con un colore di tinta rosso.



Per questo esempio, `okAction` da sopra è stato sostituito da questa `UIAlertAction` :

veloce

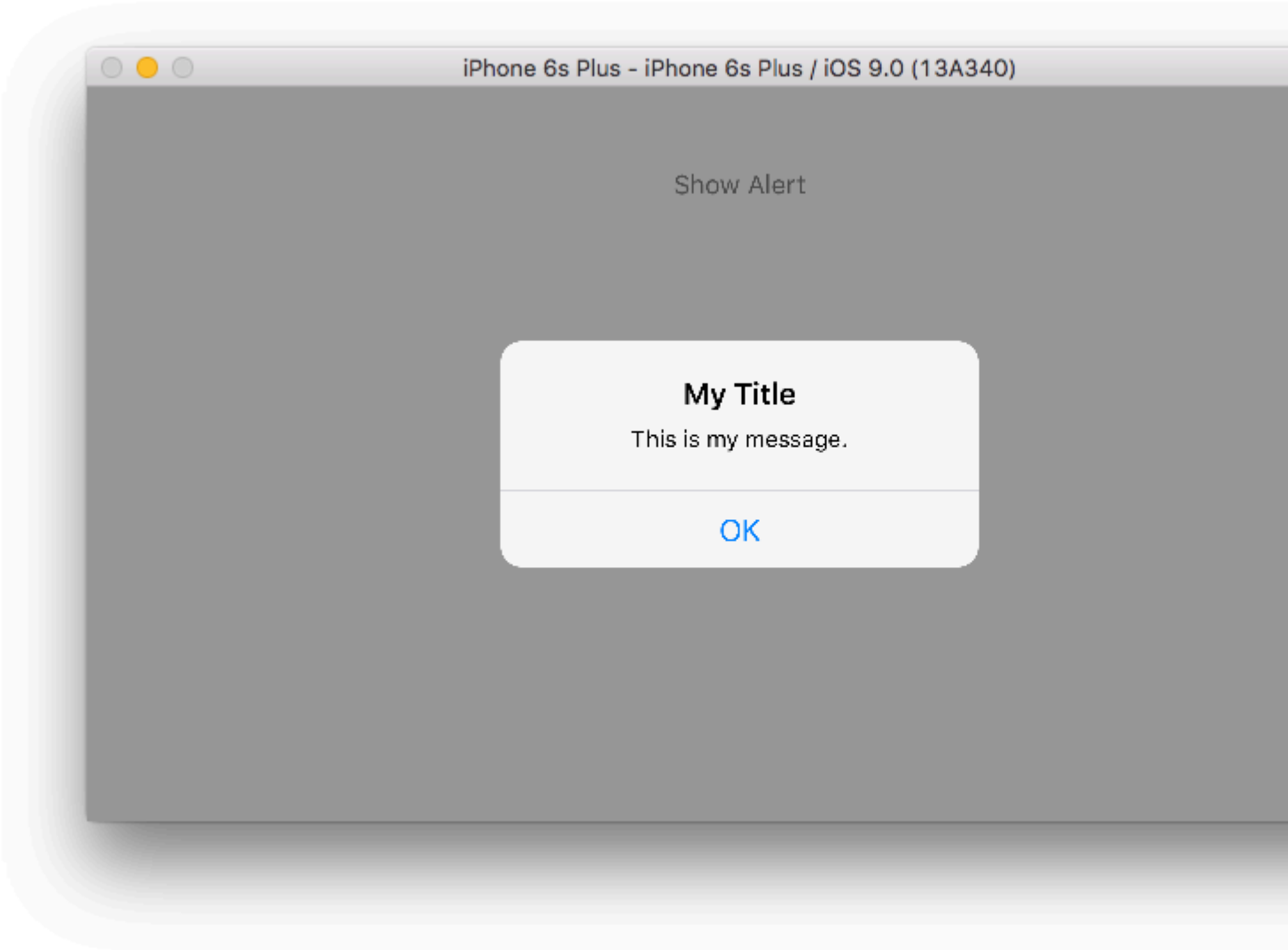
```
let destructiveAction = UIAlertAction(title: "Delete", style: .destructive) { (result :
UIAlertAction) -> Void in
    //action when pressed button
}
```

Objective-C

```
UIAlertAction * destructiveAction = [UIAlertAction actionWithTitle:@"Delete"
style:UIAlertActionStyleDestructive handler:^(UIAlertAction * action) {
            //action when pressed button
        }];
```

Visualizzazione e gestione degli avvisi

Un pulsante

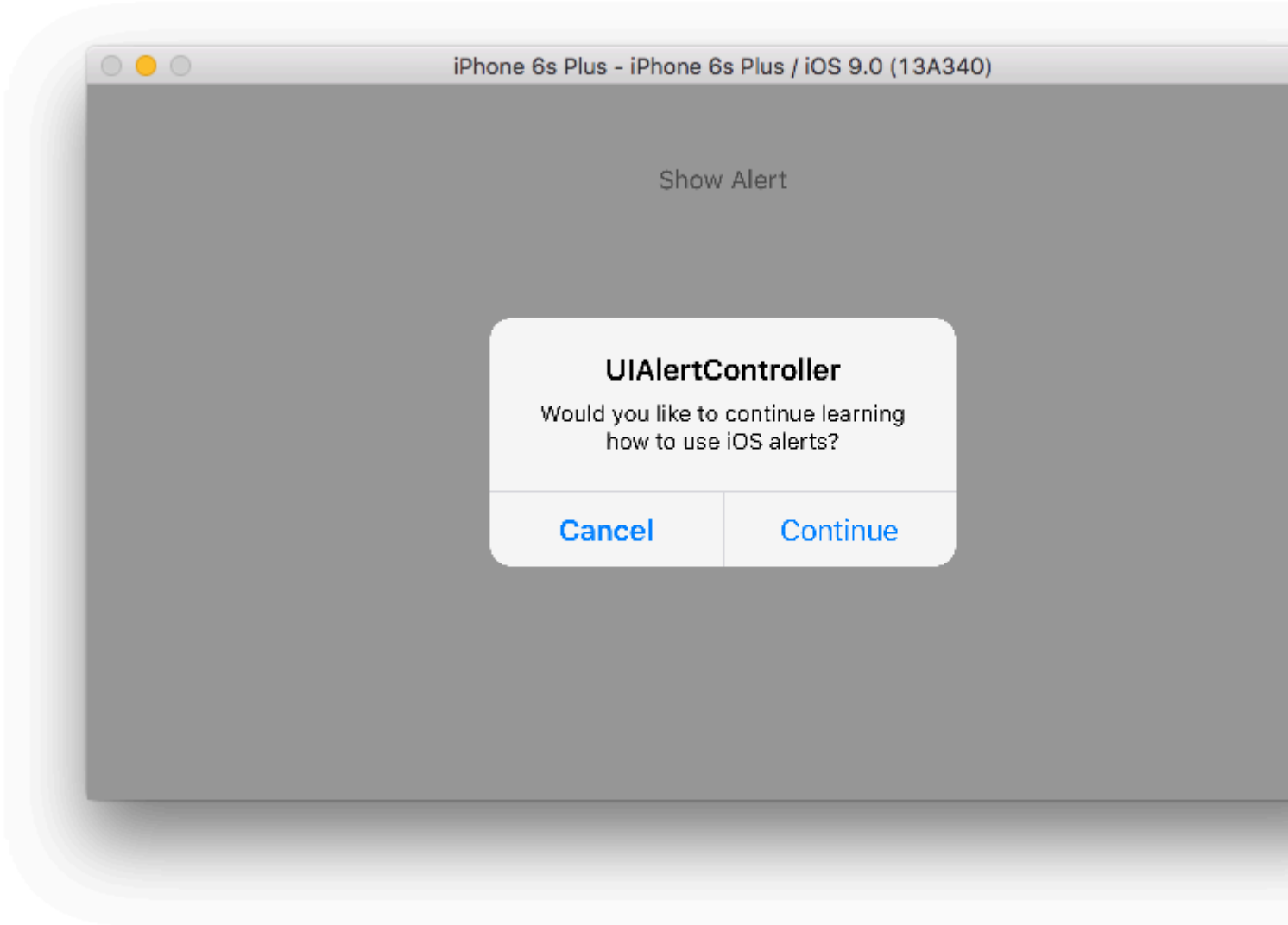


veloce

```
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)
    }
}
```

Due pulsanti

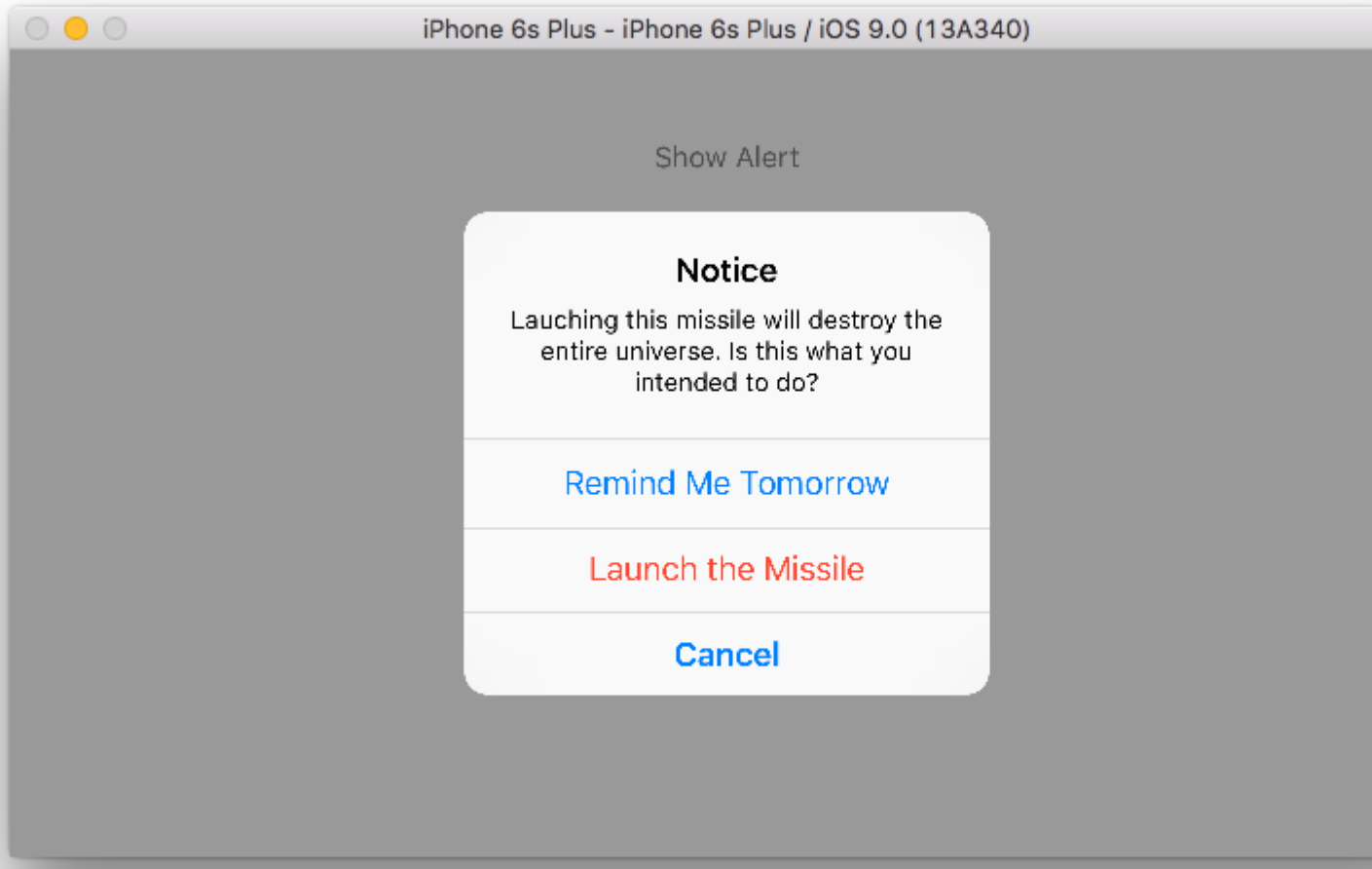


veloce

```
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)
    }
}
```

Tre pulsanti



veloce

```
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)
    }
}
```

Gestione dei pulsanti

Il `handler` era `nil` negli esempi precedenti. Puoi sostituire `nil` con una chiusura per fare qualcosa quando l'utente tocca un pulsante, come nell'esempio seguente:

veloce

```
alert.addAction(UIAlertAction(title: "Launch the Missile", style:
UIAlertActionStyle.Destructive, handler: { action in

    // do something like...
    self.launchMissile()

}))
```

Gli appunti

- Non è necessario che più pulsanti utilizzino diversi tipi `UIAlertActionStyle` . Potrebbero essere tutti. `.Default` .
- Per più di tre pulsanti, considera l'utilizzo di un foglio di azione. L'installazione è molto simile. Ecco un esempio.

Evidenziando un pulsante di azione

Il controller di avviso ha una proprietà che viene utilizzata per mettere enfasi su un'azione aggiunta nel controllore degli avvisi. Questa proprietà può essere utilizzata per evidenziare un'azione particolare per l'attenzione dell'utente. Per l'obiettivo C;

```
@property(nonatomic, strong) UIAlertAction *preferredAction
```

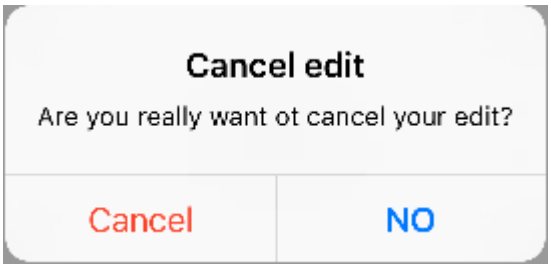
Un'azione **che è già stata aggiunta nel controller degli avvisi** può essere assegnata a questa proprietà. Alert Controller evidenzierà questa azione.

Questa proprietà può essere utilizzata solo con UIAlertControllerStyleAlert.

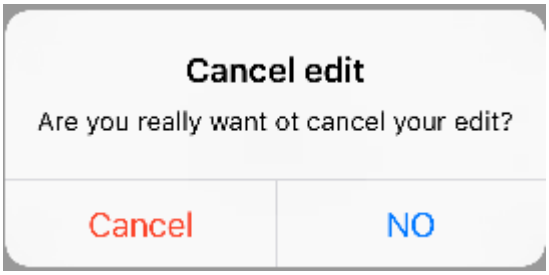
L'esempio seguente mostra come usarlo.

```
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 con **set di azioni preferito** . Il pulsante **NO** è evidenziato.



Alert Controller con l' **azione preferita non impostata** . Il pulsante **NO** non è evidenziato.



Leggi UIAlertController online: https://riptutorial.com/it/ios/topic/874/uialertcontroller

Capitolo 159: UIAppearance

Examples

Imposta l'aspetto di tutte le istanze della classe

Per personalizzare l'aspetto di tutte le istanze di una classe, accedere al proxy dell'aspetto della classe desiderata. Per esempio:

Imposta il colore della sfumatura di UIButton

Swift:

```
UIButton.appearance().tintColor = UIColor.greenColor()
```

Objective-C:

```
[UIButton appearance].tintColor = [UIColor greenColor];
```

Imposta il colore di sfondo di UIButton

Swift:

```
UIButton.appearance().backgroundColor = UIColor.blueColor()
```

Objective-C:

```
[UIButton appearance].backgroundColor = [UIColor blueColor];
```

Imposta il colore del testo UILabel

Swift:

```
UILabel.appearance().textColor = UIColor.redColor()
```

Objective-C:

```
[UILabel appearance].textColor = [UIColor redColor];
```

Imposta il colore di sfondo UILabel

Swift:

```
UILabel.appearance().backgroundColor = UIColor.greenColor()
```

Objective-C:

```
[UILabel appearance].backgroundColor = [UIColor greenColor];
```

Imposta il colore della tinta UINavigationBar

Swift:

```
UINavigationBar.appearance().tintColor = UIColor.cyanColor()
```

Objective-C:

```
[UINavigationBar appearance].tintColor = [UIColor cyanColor];
```

Imposta il colore di sfondo di UINavigationBar

Swift:

```
UINavigationBar.appearance().backgroundColor = UIColor.redColor()
```

Objective-C:

```
[UINavigationBar appearance].backgroundColor = [UIColor redColor];
```

Aspetto per classe quando contenuto in classe contenitore

Usa `appearanceWhenContainedInInstancesOfClasses:` per personalizzare l'aspetto per l'istanza di una classe quando è contenuta all'interno di un'istanza della classe contenitore. Ad esempio, la personalizzazione di `UILabel` e `backgroundColor` `textColor` all'interno della classe `ViewController` sarà simile a questa:

Imposta il colore del testo UILabel

Swift:

```
UILabel.appearanceWhenContainedInInstancesOfClasses([ViewController.self]).textColor =
UIColor.whiteColor()
```

Objective-C:

```
[UILabel appearanceWhenContainedInInstancesOfClasses:@[[ViewController class]]].textColor =
[UIColor whiteColor];
```

Imposta il colore di sfondo UILabel

Swift:

```
UILabel.appearanceWhenContainedInInstancesOfClasses([ViewController.self]).backgroundColor =
UIColor.blueColor()
```

Objective-C:

```
[UILabel appearanceWhenContainedInInstancesOfClasses:@[[ViewController
class]]].backgroundColor = [UIColor blueColor];
```

Leggi UIAppearance online: https://riptutorial.com/it/ios/topic/3422/uiappearance

Capitolo 160: UIBarButtonItem

Parametri

Parametro	Descrizione
titolo	Il titolo UIBarButtonItem
stile	Lo stile di UIBarButtonItem
bersaglio	L'oggetto per ricevere l'azione UIBarButtonItem
azione	Il selettore (metodo) da eseguire quando viene premuto UIBarButtonItem

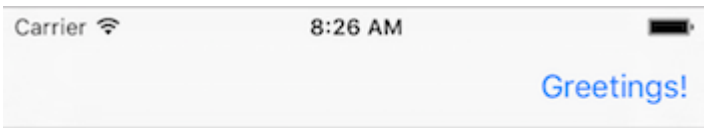
Osservazioni

Il riferimento a `self.navigationItem` presuppone che UIViewController sia incorporato in un UINavigationController.

Examples

Creazione di un 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;
```

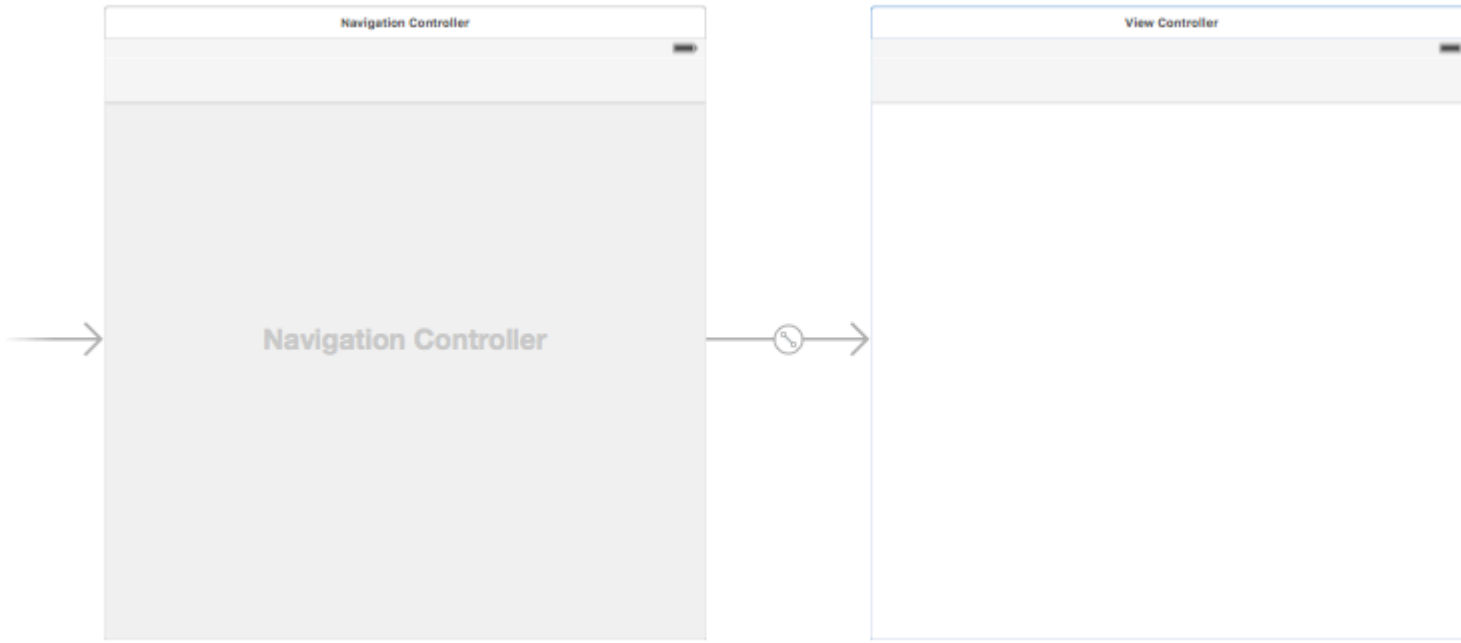


Creazione di un UIBarButtonItem in Interface Builder

L'esempio seguente mostra come aggiungere un pulsante della barra di navigazione (chiamato `UIBarButtonItem`) in Interface Builder.

Aggiungi un controller di navigazione allo storyboard

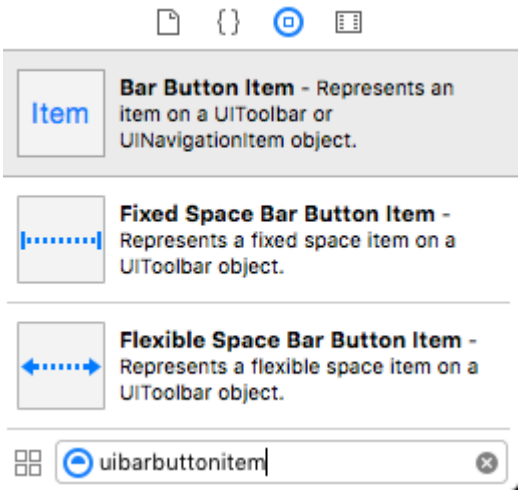
Seleziona il controller di visualizzazione e quindi nel menu Xcode scegli **Editor> Incorpora in> Controller di navigazione** .



In alternativa, è possibile aggiungere una `UINavigationBar` dalla libreria degli oggetti.

Aggiungi un elemento del pulsante Bar

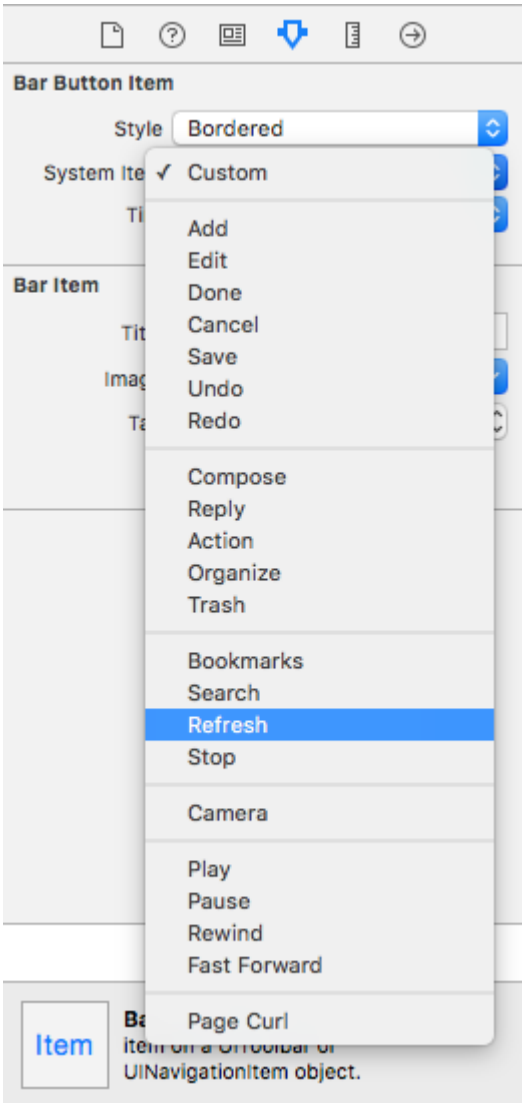
Trascina un `UIBarButtonItem` dalla libreria degli oggetti nella barra di navigazione in alto.



Dovrebbe sembrare come questo:

Imposta gli attributi

È possibile fare doppio clic su "Elemento" per cambiare il testo in qualcosa come "Aggiorna", ma c'è un'icona reale per l' *aggiornamento* che è possibile utilizzare. Basta selezionare l'ispettore degli attributi per `UIBarButtonItem` e per l' **elemento di sistema** scegliere **Aggiorna** .



Questo ti darà l'icona di aggiornamento predefinita.

Aggiungi un'azione IB

Controlla il trascinamento da `UIBarButtonItem` al View Controller per aggiungere un `@IBAction` .

```
class ViewController: UIViewController {

    @IBAction func refreshBarButtonItemTap(sender: UIBarButtonItem) {

        print("How refreshing!")
    }

}
```

Questo è tutto.

Gli appunti

- Questo esempio deriva originariamente da questa risposta di Overflow dello stack .

Bar Button Item Immagine originale senza colore Tint

A condizione che `barButtonItem` abbia una proprietà immagine non nullo (ad es. Impostata in Interface Builder).

Objective-C

```
   barButtonItem.image = [barButtonItem.image
imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
```

Leggi UIBarButtonItem online: https://riptutorial.com/it/ios/topic/1543/uibarbuttonitem

Capitolo 161: UIBezierPath

Examples

Come applicare il raggio dell'angolo ai rettangoli disegnati da UIBezierPath

Raggio d'angolo per tutti e 4 i bordi:

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) cornerRadius: 11];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Raggio dell'angolo per il margine superiore sinistro:

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopLeft cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Raggio dell'angolo per il margine superiore destro:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopRight cornerRadii:
```

```
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

raggio dell'angolo per il bordo inferiore sinistro:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomLeft cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

raggio dell'angolo per il bordo inferiore destro:

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomRight cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

raggio dell'angolo per i bordi inferiori:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomLeft |
UIRectCornerBottomRight cornerRadii: CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
```

```
[rectanglePath fill];
```

raggio dell'angolo per i bordi superiori:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopLeft | UIRectCornerTopRight
cornerRadii: CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Come creare forme semplici usando UIBezierPath

Per un cerchio semplice:

```
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()
```

Per un semplice rettangolo:

```
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()
```

Per una linea semplice:


```
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()
```

Per un semicerchio:

```
 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()
```

Per un triangolo semplice:

```
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

Affinché il percorso di Bezier venga ridimensionato in base al fotogramma della vista, eseguire l'override del drawRect della vista in cui si disegna il percorso di Bezier:

```
- (void)drawRect:(CGRect)frame
{
    UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect:
CGRectMake(CGRectGetMinX(frame), CGRectGetMinY(frame), CGRectGetWidth(frame),
CGRectGetHeight(frame))];
    [UIColor.grayColor setFill];
```

```
    [rectanglePath fill];
}
```

Come applicare le ombre a UIBezierPath

Considera un semplice rettangolo disegnato dal percorso di Bezier.

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect:
CGRectMake(x,y,width,height)];
 [UIColor.grayColor setFill];
 [rectanglePath fill];
```

Ombra di riempimento esterna di base:

```
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);
```

Ombra di riempimento interna di base:

```
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);
```

Progettare e disegnare un percorso di Bezier

Questo esempio mostra il processo dalla progettazione della forma che si desidera disegnare su una vista. Viene usato uno shap specifico ma i concetti che impari possono essere applicati a qualsiasi forma.

Come disegnare un tracciato di Bézier in una vista personalizzata

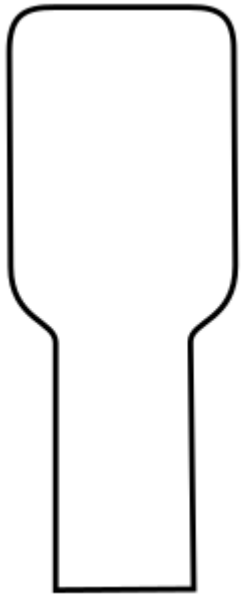
Questi sono i passaggi principali:

1. Progetta il contorno della forma che desideri.

2. Dividere il tracciato del contorno in segmenti di linee, archi e curve.
3. Costruisci quel percorso a livello di codice.
4. Disegna il percorso in `drawRect` o utilizzando un `CAShapeLayer` .

Disegna il contorno della forma

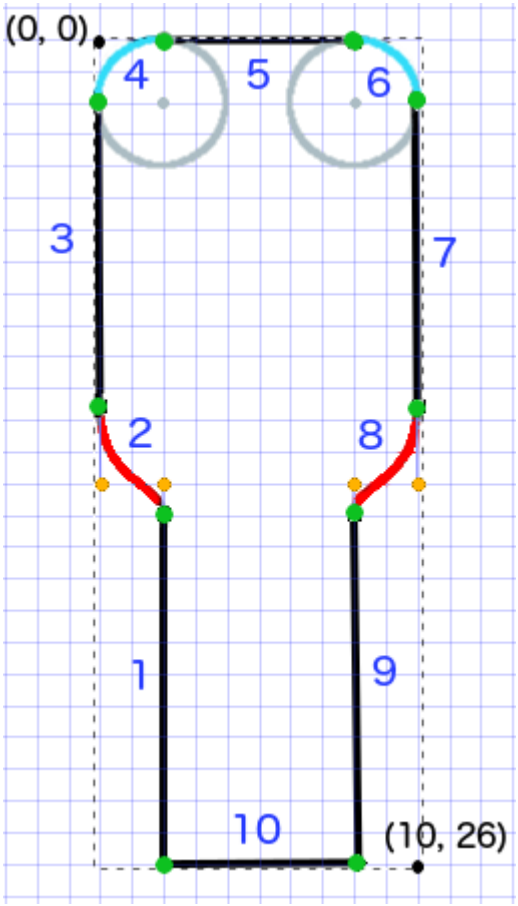
Potresti fare qualsiasi cosa, ma come esempio ho scelto la forma qui sotto. Potrebbe essere un tasto popup su una tastiera.



Dividere il percorso in segmenti

Osserva il design della tua forma e scomporlo in elementi di linee più semplici (per linee rette), archi (per cerchi e angoli arrotondati) e curve (per qualsiasi altra cosa).

Ecco come sarà il nostro esempio di design:



- Il nero sono segmenti di linea
- Blu chiaro sono segmenti di arco
- Il rosso sono curve
- I punti arancioni sono i punti di controllo per le curve
- I punti verdi sono i punti tra i segmenti del percorso
- Le linee tratteggiate mostrano il rettangolo di delimitazione
- I numeri blu scuro sono i segmenti nell'ordine in cui verranno aggiunti a livello di codice

Costruisci il percorso a livello di programmazione

Iniziamo arbitrariamente nell'angolo in basso a sinistra e lavoriamo in senso orario. Userò la griglia nell'immagine per ottenere i valori xey per i punti. Farò un hardcode di tutto qui, ma ovviamente non lo faresti in un vero progetto.

Il processo di base è:

1. Creare un nuovo `UIBezierPath`
2. Scegli un punto di partenza sul percorso con `moveToPoint`
3. Aggiungi segmenti al percorso

- linea: `addLineToPoint`
- arc: `addArcWithCenter`

- curva: `addCurveToPoint`

4. Chiudi il percorso con `closePath`

Ecco il codice per rendere il percorso nell'immagine sopra.

```
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
}
```

Nota: alcuni dei codici precedenti possono essere ridotti aggiungendo una linea e un arco in un singolo comando (poiché l'arco ha un punto di partenza implicito). Vedi qui per maggiori dettagli.

Disegna il percorso

Possiamo tracciare il percorso sia in layer che in `drawRect` .

Metodo 1: traccia il percorso in un livello

La nostra classe personalizzata sembra così. Aggiungiamo il nostro percorso Bezier a un nuovo `CAShapeLayer` quando la vista è inizializzata.

```
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
    }
}
```

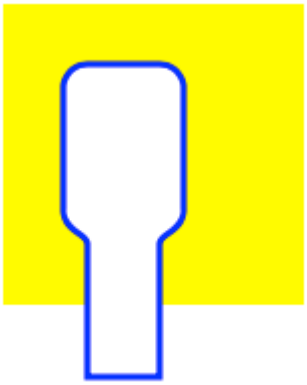
E creando la nostra vista nel View Controller come questo

```
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)

}
```

Noi abbiamo...

Hmm, è un po 'piccolo perché ho codificato tutti i numeri. Posso ridimensionare la dimensione del percorso, però, in questo modo:

```
let path = createBezierPath()
let scale = CGAffineTransformMakeScale(2, 2)
path.applyTransform(scale)
shapeLayer.path = path.CGPath
```



Metodo 2: tracciare il percorso in `drawRect`

L'uso di `drawRect` è più lento del disegno sul livello, quindi questo non è il metodo raccomandato se non ne hai bisogno.

Ecco il codice revisionato per la nostra visualizzazione personalizzata:

```
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
    }
}
```

che ci dà lo stesso risultato ...

Ulteriore studio

Articoli eccellenti per comprendere i percorsi di Bezier.

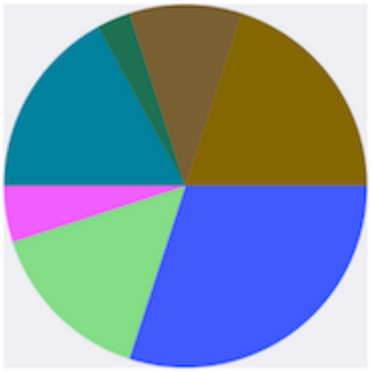
- Pensando come un percorso di Bézier (Tutto quello che ho letto da questo autore è buono e l'ispirazione per il mio esempio qui sopra è venuta da qui.)
- Coding Math: Episode 19 - Bezier Curves (divertenti e belle illustrazioni visive)
- Bezier Curves (come vengono utilizzati nelle applicazioni grafiche)
- Bezier Curves (buona descrizione di come vengono derivate le formule matematiche)

Gli appunti

- Questo esempio deriva originariamente da questa risposta di Overflow dello stack .
- Nei tuoi progetti attuali probabilmente non dovresti usare numeri codificati, ma piuttosto prendere le dimensioni dai limiti della tua vista.

vista a torta e vista a colonne con UIBezierPath

- vista a torta

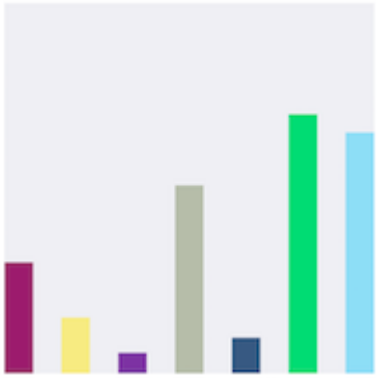


```
- (void)drawRect:(CGRect)rect {

    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
    }
  }
```

- vista colonna



```
- (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()
    }
  }
```

Leggi UIBezierPath online: https://riptutorial.com/it/ios/topic/3186/uibezierpath

Capitolo 162: UIButton

introduzione

UIButton : UIControl intercetta gli eventi di tocco e invia un messaggio di azione a un oggetto di destinazione quando viene toccato. È possibile impostare il titolo, l'immagine e altre proprietà di aspetto di un pulsante. Inoltre, è possibile specificare un aspetto diverso per ogni stato del pulsante.

Osservazioni

Tipi di pulsanti

Il tipo di un pulsante definisce il suo aspetto e comportamento di base. Dopo aver creato un pulsante, non è possibile cambiarne il tipo. I tipi di pulsante più comuni sono i tipi Personalizzato e Sistema, ma se possibile, utilizzare gli altri tipi

- 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.
```

Quando si crea un pulsante personalizzato, ovvero un pulsante con il tipo personalizzato, la cornice del pulsante viene inizialmente impostata su (0, 0, 0, 0). Prima di aggiungere il pulsante

all'interfaccia, è necessario aggiornare la cornice con un valore più appropriato.

Examples

Creazione di un UIButton

Gli UIButtons possono essere inizializzati in un frame:

veloce

```
let button = UIButton(frame: CGRect(x: x, y: y, width: width, height: height)
```

Obiettivo C

```
UIButton *button = [[UIButton alloc] initWithFrame:CGRectMake(x, y, width, height)];
```

Un tipo specifico di UIButton può essere creato in questo modo:

veloce

```
let button = UIButton(type: .Custom)
```

Obiettivo C

```
UIButton *button = [UIButton buttonWithType:UIButtonTypeCustom];
```

dove `type` è un `UIButtonType` :

```
enum UIButtonType : Int {
    case Custom
    case System
    case DetailDisclosure
    case InfoLight
    case InfoDark
    case ContactAdd
    static var RoundedRect: UIButtonType { get }
}
```

Imposta il titolo

veloce

```
button.setTitle(titleString, forState: controlState)
```

Obiettivo C

```
[button setTitle:(NSString *) forState:(UIControlState)];
```

Per impostare il titolo predefinito su "Hello, World!"

veloce

```
button.setTitle("Hello, World!", forState: .normal)
```

Obiettivo C

```
[button setTitle:@"Hello, World!" forControlState:UIControlStateNormal];
```

Imposta il colore del titolo

```
//Swift
button.setTitleColor(color, forControlState: controlState)

//Objective-C
[button setTitleColor:(nullable UIColor *) forState:(UIControlState)];
```

Per impostare il colore del titolo su blu

```
//Swift
button.setTitleColor(.blue, for: .normal)

//Objective-C
[button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal]
```

Allinea orizzontalmente i contenuti

veloce

```
//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
```

Obiettivo 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;
```

Ottenere l'etichetta del titolo

L'etichetta del titolo sottostante, se ne esiste una, può essere recuperata usando

veloce

```
var label: UILabel? = button.titleLabel
```

Obiettivo C

```
UILabel *label = button.titleLabel;
```

Questo può essere usato per impostare il carattere dell'etichetta del titolo, per esempio

veloce

```
button.titleLabel?.font = UIFont.boldSystemFontOfSize(12)
```

Obiettivo C

```
button.titleLabel.font = [UIFont boldSystemFontOfSize:12];
```

Disabilitare un UIButton

Un pulsante può essere disabilitato da

veloce

```
myButton.isEnabled = false
```

Objective-C:

```
myButton.enabled = NO;
```

Il pulsante diventerà grigio:

Se non si desidera che l'aspetto del pulsante cambi quando è disattivato, impostare `adjustsImageWhenDisabled` su `false` / `NO`

Aggiunta di un'azione a un UIButton tramite codice (a livello di codice)

Per aggiungere un metodo a un pulsante, innanzitutto creare un metodo di azione:

Objective-C

```
-(void)someButtonAction:(id)sender {
  // sender is the object that was tapped, in this case its the button.
    NSLog(@"Button is tapped");
}
```

veloce

```
func someButtonAction() {
    print("Button is tapped")
}
```

Ora per aggiungere questo metodo di azione al tuo pulsante, devi scrivere la seguente riga di codice:

Obiettivo C

```
[yourButtonInstance addTarget:self action:@selector(someButtonAction)
forControlEvents:UIControlEventTouchUpInside];
```

veloce

```
yourButtonInstance.addTarget(self, action: #selector(someButtonAction), forControlEvents:
.TouchUpInside)
```

Per il parametro ControlEvents, tutti i membri di `ENUM` UIControlEvents sono validi.

Impostazione del carattere

veloce

```
myButton.titleLabel?.font =  UIFont(name: "YourFontName", size: 20)
```

Obiettivo C

```
myButton.titleLabel.font = [UIFont fontWithName:@"YourFontName" size:20];
```

Collegamento di un metodo a un pulsante

Per aggiungere un metodo a un pulsante, innanzitutto creare un metodo di azione:

Objective-C

```
-(void) someButtonAction{
    NSLog(@"Button is tapped");

}
```

veloce

```
func someButtonAction() {
        print("Button is tapped")
    }
```

Ora per aggiungere questo metodo di azione al tuo pulsante, devi scrivere la seguente riga di codice:

Obiettivo C

```
[yourButtonInstance addTarget:self action:@selector(someButtonAction)
forControlEvents:UIControlEventTouchUpInside];
```

veloce

```
yourButtonInstance.addTarget(self, action: #selector(someButtonAction), forControlEvents:
.touchUpInside)
```

Per ControlEvents, tutti i membri di `ENUM` UIControlEvents sono validi.

Ottieni le dimensioni di UIButton in base al testo e al carattere

Per ottenere la dimensione esatta del testo di un UIButton in base al suo carattere, utilizzare la funzione `intrinsicContentSize` .

veloce

```
button.intrinsicContentSize.width
```

Objective-C

```
button.intrinsicContentSize.width;
```

Imposta immagine

veloce

```
button.setImage(UIImage(named:"test-image"), forState: .normal)
```

Obiettivo C

```
[self.button setImage:[UIImage imageNamed:@"test-image"] forState:UIControlStateNormal];
```

Più stati di controllo

È inoltre possibile impostare un'immagine per più `UIControlStates` , ad esempio per impostare la stessa immagine per lo stato `Selected` e `Highlighted` :

veloce

```
button.setImage(UIImage(named:"test-image"), forState:[.selected, .highlighted])
```

Obiettivo C

```
[self.button setImage:[UIImage imageNamed:@"test-image"]
forState:UIControlStateSelected|UIControlStateHighlighted];
```

Leggi UIButton online: https://riptutorial.com/it/ios/topic/516/uibutton

Capitolo 163: UICollectionView

Examples

Creare una vista insieme a livello di programmazione

veloce

```
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
    }
}
```

Crea un UICollectionView

Inizializza un `UICollectionView` con un frame `CGRect` :

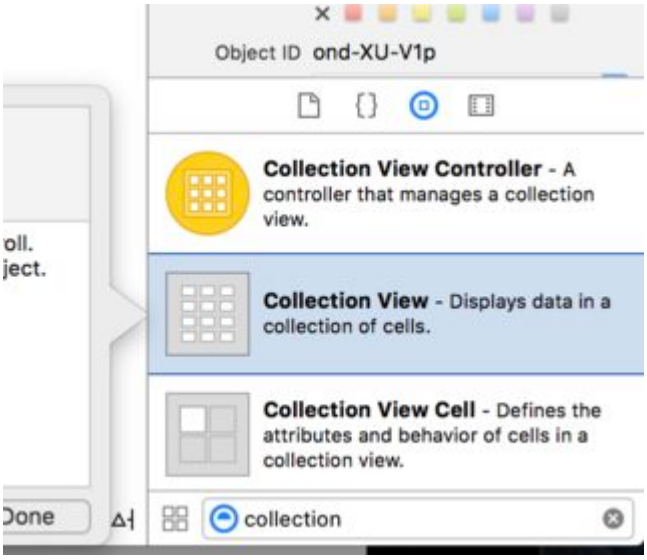
Swift:

```
let collection = UICollectionView(frame: CGRect(x: 0, y: 0, width: 200, height: 21))
```

Obiettivo C:

```
UICollectionView *collection = [[UICollectionView alloc] initWithFrame:CGRectMake(0, 0, 200,
21)];
```

È inoltre possibile creare un `UICollectionView` in Interface Builder



UICollectionView - Origine dati

Ogni vista `Datasource` deve avere un oggetto `Datasource` . L'oggetto `Datasource` è il contenuto che l'app visualizzerà all'interno di `UICollectionView` . Come minimo, tutti gli oggetti `Datasource` devono implementare `collectionView:numberOfItemsInSection:` e `collectionView:cellForItemAtIndexPath:` metodi.

Metodi richiesti

veloce

```
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
}
```

Obiettivo 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;
}
```

Esempio di base Swift di una vista insieme

Crea un nuovo progetto

Può essere solo un'applicazione vista singola.

Aggiungi il codice

Creare un nuovo file di classe Cocoa Touch (File> Nuovo> File ...> iOS> Cocoa Touch Class). `MyCollectionViewCell` . Questa classe terrà gli outlets per le viste che aggiungi alla tua cella nello storyboard.

```
import UIKit
class MyCollectionViewCell: UICollectionViewCell {

    @IBOutlet weak var myLabel: UILabel!
}
```

Collegheremo questa presa più tardi.

Apri ViewController.swift e assicurati di avere il seguente contenuto:

```
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)!")
    }
}
```

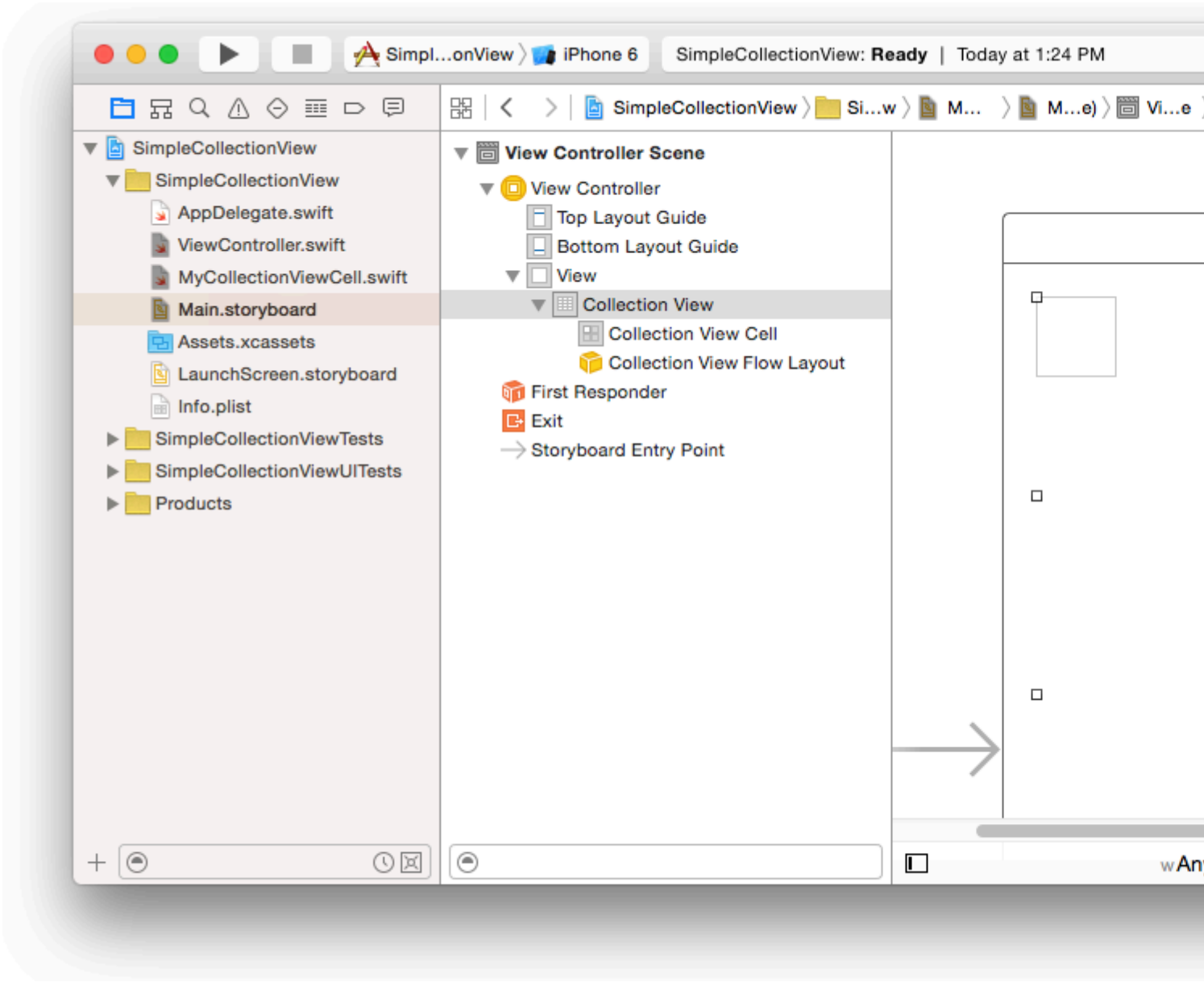
Gli appunti

- `UICollectionViewDataSource` e `UICollectionViewDelegate` sono i protocolli `UICollectionViewDelegate` dalla vista `UICollectionViewDelegate` . È anche possibile aggiungere il protocollo `UICollectionViewDelegateFlowLayout` per modificare la dimensione delle viste a livello di `UICollectionViewDelegateFlowLayout` , ma non è necessario.
- Stiamo semplicemente mettendo delle semplici stringhe nella nostra griglia, ma potreste sicuramente fare le immagini in seguito.

Imposta lo storyboard

Trascina una vista raccolta sul controller di visualizzazione nello storyboard. Puoi aggiungere

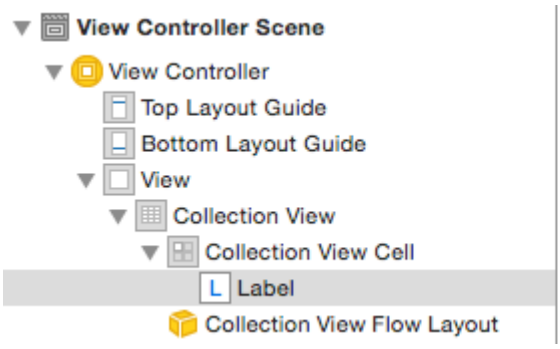
vincoli per farlo riempire la vista genitore, se lo desideri.



Assicurati che anche i tuoi valori di default nell'Inspector degli attributi

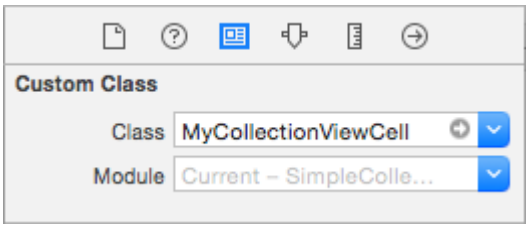
- Articoli: 1
- Layout: flusso

La piccola scatola in alto a sinistra della Vista raccolta è una cella Vista raccolta. Lo useremo come nostra cellula prototipo. Trascina un'etichetta nella cella e centrala. Puoi ridimensionare i bordi della cella e aggiungere vincoli per centrare l'etichetta, se lo desideri.



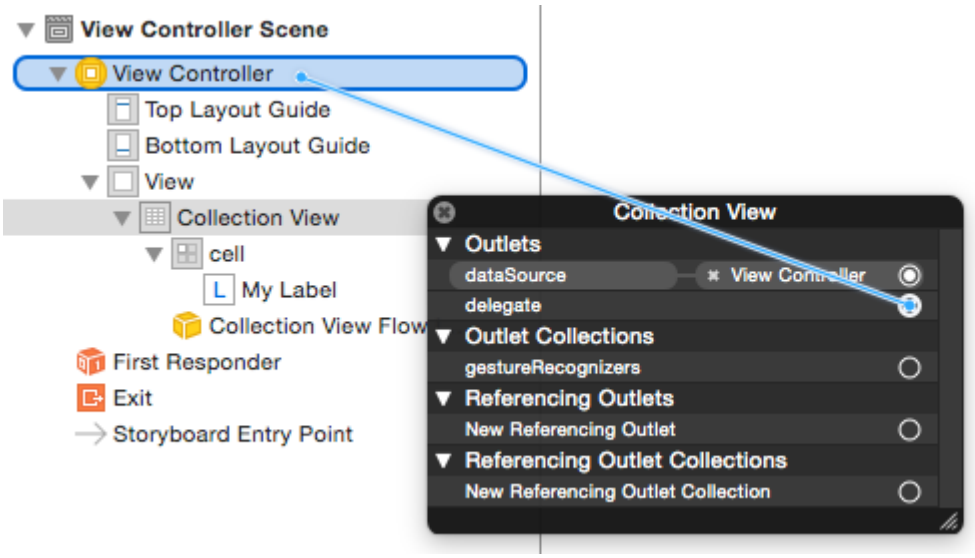
Scrivi "cella" (senza virgolette) nella casella Identificatore dell'Ispettore Attributi per la cella Vista raccolta. Si noti che questo è lo stesso valore di `let reuseIdentifier = "cell"` in ViewController.swift.

E in Identity Inspector per la cella, imposta il nome della classe su `MyCollectionViewCell` , la nostra classe personalizzata che abbiamo creato.



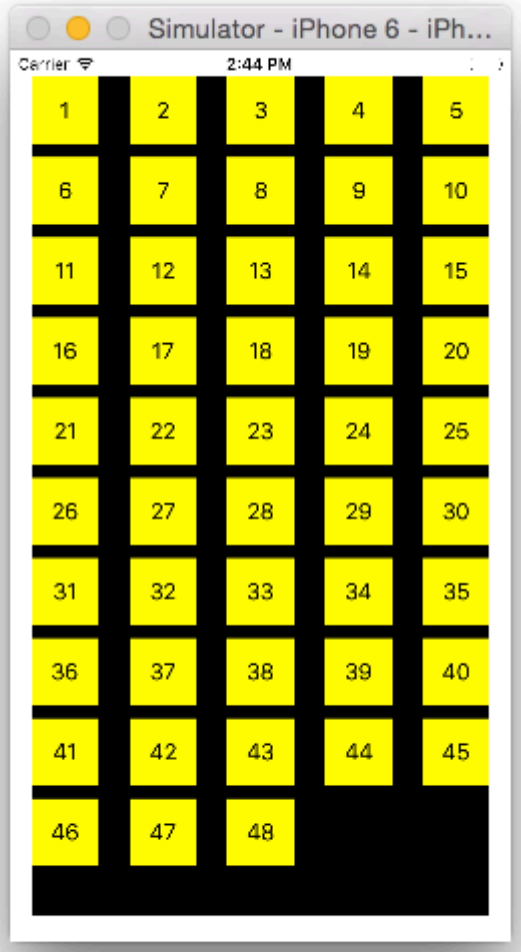
Agganciare le prese

- Aggancia l'etichetta nella cella di raccolta a `myLabel` nella classe `MyCollectionViewCell` . (Puoi trascinare il controllo .)
- Agganciare il `delegate` Vista View e `dataSource` al View Controller. (Fare clic con il pulsante destro del mouse su Vista raccolta nella struttura del documento, quindi fare clic e trascinare la freccia più verso l'alto fino a visualizzare il controller.)



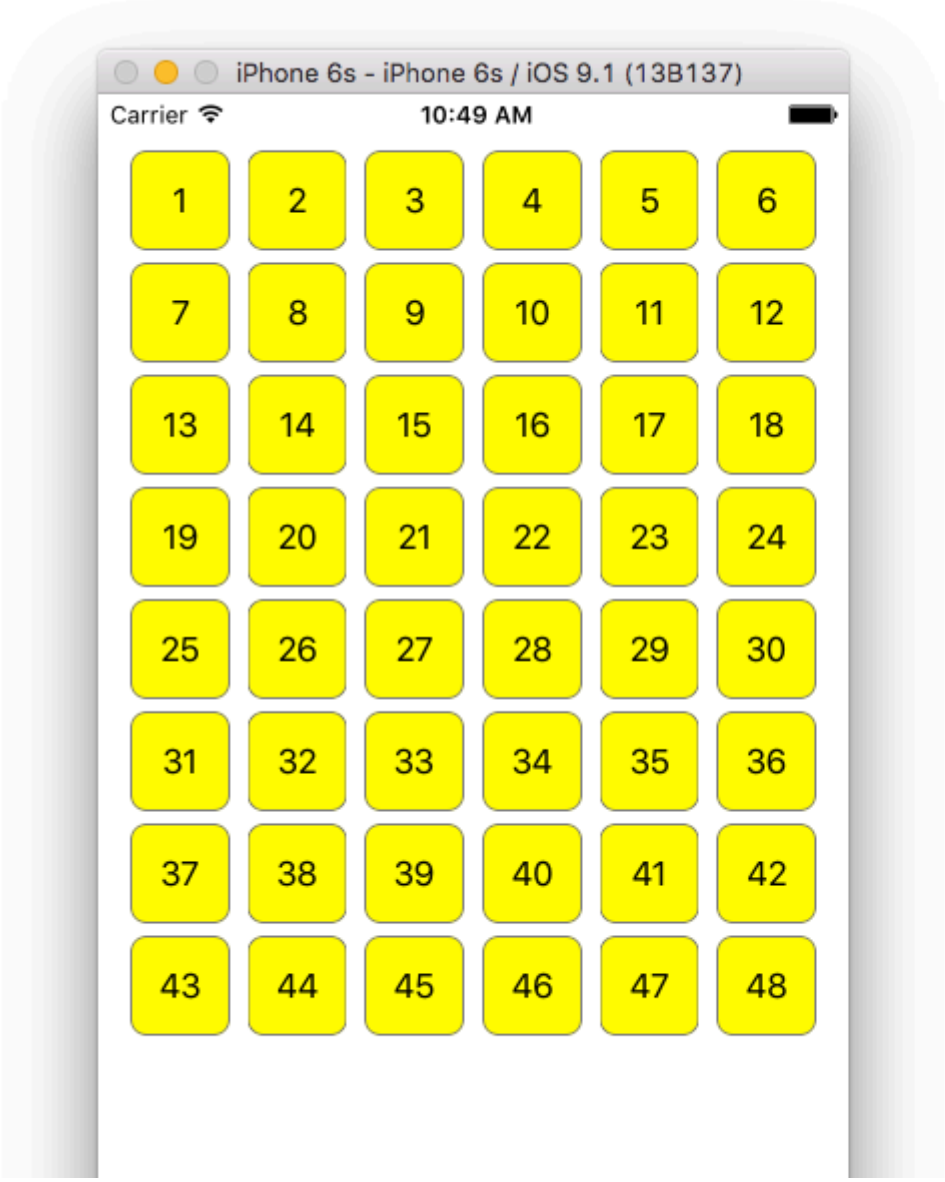
Finito

Ecco come appare dopo aver aggiunto i vincoli per centrare l'etichetta nella cella e bloccare la vista raccolta sui muri del genitore.



Fare miglioramenti

Se vuoi migliorare l'aspetto, guarda il post originale da cui proviene questo esempio .



Ulteriore studio

- Un semplice tutorial UICollectionView
- Esercitazione UICollectionView Parte 1: Introduzione
- Esercitazione UICollectionView Parte 2: Viste riutilizzabili e selezione delle celle

Esecuzione di aggiornamenti batch

È possibile animare modifiche complesse alla vista raccolta utilizzando il metodo `performBatchUpdates` . All'interno del blocco di aggiornamento, puoi specificare diverse modifiche per farli animare tutto in una volta.

```
collecitonView.performBatchUpdates({
    // Perform updates
}, nil)
```

All'interno del blocco di aggiornamento è possibile eseguire inserimenti, eliminazioni, spostamenti e ricariche. Ecco come determinare quale indexPath utilizzare:

genere	NSIndexPath
Inserimento	Indice nel nuovo array
cancellazione	Indice nel vecchio array
Mossa	da: vecchio array, a: nuovo array
Ricaricare	Array nuovo o vecchio (non dovrebbe essere importante)

Dovresti chiamare solo ricaricare su celle che non sono state spostate, ma il loro contenuto è cambiato. È importante notare che una mossa non aggiorna il contenuto di una cella, ma sposta solo la sua posizione.

Per verificare che l'aggiornamento batch venga eseguito correttamente, assicurarsi che il set di indexPath per l' `deletion` , `move-from` e `reload` sia univoco e che l'insieme di indexPath per l' `insertion` , `move-to` e `reload` sia univoco.

Ecco un esempio di un aggiornamento batch corretto:

```
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)
```

UICollectionViewDelegate l'installazione e la selezione degli oggetti

A volte, se un'azione deve essere `UICollectionViewDelegate` alla selezione di celle di una vista raccolta, è necessario implementare il protocollo `UICollectionViewDelegate` .

Diciamo che la vista `UIViewController MyViewController` è all'interno di un `UIViewController MyViewController` .

Objective-C

Nel tuo *MyViewController.h* dichiara che implementa il protocollo `UICollectionViewDelegate` , come di seguito

```
@interface MyViewController : UIViewController <UICollectionViewDelegate, .../* previous
existing delegate, as UICollectionDataSource *>
```

veloce

Nel tuo *MyViewController.swift* aggiungi quanto segue

```
class MyViewController : UICollectionViewDelegate {
}
```

Il metodo che deve essere implementato è

Objective-C

```
-(void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath
*)indexPath
{
}
```

veloce

```
func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath:
NSIndexPath)
{
}
```

Come solo un esempio possiamo impostare il colore di sfondo della cella selezionata in verde.

Objective-C

```
-(void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath
*)indexPath
{
    UICollectionViewCell* cell = [collectionView cellForItemAtIndexPath:indexPath];
    cell.backgroundColor = [UIColor greenColor];
}
```

veloce

```
class MyViewController : UICollectionViewDelegate {
    func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath:
NSIndexPath)
    {
        var cell : UICollectionViewCell = collectionView.cellForItemAtIndexPath(indexPath)!
        cell.backgroundColor = UIColor.greenColor()
    }
}
```

Gestisci la vista Raccolta multipla con DataSource e Flowlayout

Qui gestiamo una raccolta multipla con metodi delegati con eventi selezionati.

```
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
        }

    }
```

Leggi UICollectionView online: https://riptutorial.com/it/ios/topic/2399/uicollectionview

Capitolo 164: UIColor

Examples

Creare un UIColor

Esistono molti modi per creare un `UIColor` :

veloce

- Utilizzando uno dei colori predefiniti:

```
let redColor = UIColor.redColor()
let blueColor: UIColor = .blueColor()

// In Swift 3, the "Color()" suffix is removed:
let redColor = UIColor.red
let blueColor: UIColor = .blue
```

  Se il compilatore sa già che la variabile è un'istanza di `UIColor` , puoi saltare il tipo tutti insieme:

```
let view = UIView()
view.backgroundColor = .yellowColor()
```

- Utilizzando il valore di scala di grigi e l'alfa:

```
let grayscaleColor = UIColor(white: 0.5, alpha: 1.0)
```

- Usando tonalità, saturazione, luminosità e alfa:

```
let hsbColor = UIColor(
    hue: 0.4,
    saturation: 0.3,
    brightness: 0.7,
    alpha: 1.0
)
```

- Utilizzando i valori RGBA:

```
let rgbColor = UIColor(
    red: 30.0 / 255,
    green: 70.0 / 255,
    blue: 200.0 / 255,
    alpha: 1.0
)
```

- Utilizzando un'immagine modello:

```
let patternColor = UIColor(patternImage: UIImage(named: "myImage")!)
```

Objective-C

- Utilizzando uno dei colori predefiniti:

```
UIColor *redColor = [UIColor redColor];
```

- Utilizzando il valore di scala di grigi e l'alfa:

```
UIColor *grayscaleColor = [UIColor colorWithWhite: 0.5 alpha: 1.0];
```

- Usando tonalità, saturazione, luminosità e alfa:

```
UIColor *hsbColor = [UIColor
    colorWithHue: 0.4
    saturation: 0.3
    brightness: 0.7
    alpha: 1.0
];
```

- Utilizzando i valori RGBA:

```
UIColor *rgbColor = [UIColor
    colorWithRed: 30.0 / 255.0
    green: 70.0 / 255.0
    blue: 200.0 / 255.0
    alpha: 1.0
];
```

- Utilizzando un'immagine modello:

```
UIColor *pattenColor = [UIColor colorWithPatternImage:[UIImage
imageNamed:@"myImage.png"]];
```

Metodi non documentati

Esistono vari metodi non documentati su `UIColor` che espongono colori o funzionalità alternativi. Questi possono essere trovati nel file di intestazione privato `UIColor` . Documenterò l'uso di due metodi privati, `styleString()` e `_systemDestructiveTintColor()` .

`styleString`

Da iOS 2.0 esiste un metodo di istanza privato su `UIColor` chiamato `styleString` che restituisce una rappresentazione di stringa RGB o RGBA del colore, anche per i colori come `whiteColor` al di fuori dello spazio 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)
```

In Swift puoi usare un'intestazione di bridging per esporre l'interfaccia. Con pure Swift, è necessario creare un protocollo `@objc` con il metodo privato e `unsafeBitCast UIColor` con il protocollo:

```
@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()`

Esiste un metodo di classe non documentato su `UIColor` chiamato `_systemDestructiveTintColor` che restituirà il colore rosso utilizzato dai pulsanti di sistema distruttivo:

```
let red = UIColor.performSelector("_systemDestructiveTintColor").takeUnretainedValue()
```

Restituisce un oggetto non gestito, che deve essere chiamato `.takeUnretainedValue()` , poiché la proprietà del colore non è stata trasferita al nostro oggetto.

Come con qualsiasi API non documentata, dovresti fare attenzione quando provi ad utilizzare questo metodo:

```
if UIColor.respondsToSelector("_systemDestructiveTintColor") {
    if let red = UIColor.performSelector("_systemDestructiveTintColor").takeUnretainedValue()
as? UIColor {
        // use the color
    }
}
```

o usando un protocollo:

```
@objc protocol UIColorPrivateStatic {
    func _systemDestructiveTintColor() -> UIColor
}

let privateClass = UIColor.self as! UIColorPrivateStatic
privateClass._systemDestructiveTintColor() // UIDeviceRGBColorSpace 1 0.231373 0.188235 1
```

Colore con componente Alpha

È possibile impostare l'opacità su un determinato `UIColor` senza crearne uno nuovo utilizzando l' `init(red:_,green:_,blue:_,alpha:_)` .

veloce

```
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];
```

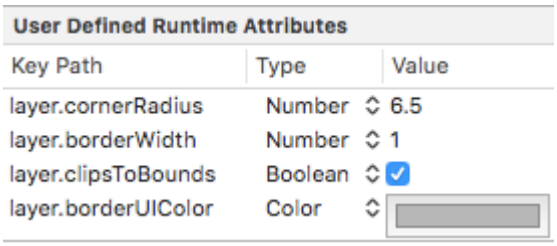
Definisci attributi definiti dall'utente applica il tipo di dati CGColor

Per impostazione predefinita, Interface Builder non accetta il tipo di dati `CGColor` , quindi per consentire l'aggiunta di un `CGColor` utilizzando gli attributi definiti dall'utente nel builder dell'interfaccia; si potrebbe voler usare un'estensione come questa:

Estensione rapida:

```
extension CALayer {
    func borderUIColor() -> UIColor? {
        return borderColor != nil ? UIColor(CGColor: borderColor!) : nil
    }

    func setBorderUIColor(color: UIColor) {
        borderColor = color.CGColor
    }
}
```

Il nuovo attributo definito dall'utente (borderUIColor) verrà riconosciuto e applicato senza problemi.



Creazione di un UIColor da numero o stringa esadecimale

Puoi creare un `UIColor` da un numero esadecimale o da una stringa, ad esempio 0xff00cc, "#FFFFFF"

veloce

Valore int

```
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)
    }
}
```

Esempio:

```
let color = UIColor(hex: 0xff00cc, alpha: 1.0)
```

Si noti che per `alpha` viene fornito il valore predefinito di `1.0` , quindi può essere utilizzato come segue:

```
let color = UIColor(hex: 0xff00cc)
```

Valore stringa

```
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)
    }
}
```

Esempio di utilizzo:

Esadecimale con alfa

```
let color = UIColor("#80FFFFFF")
```

Esadecimale senza alfa (`color` alpha uguale a 1.0)

```
let color = UIColor("#FFFFFF")
let color = UIColor("#FFF")
```

Objective-C

Valore int

```
@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
```

Esempio:

```
UIColor *color = [UIColor colorWithHex:0xff00cc alpha:1.0];
```

Valore stringa

```
- (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];
}
```

Esempio di utilizzo:

Esadecimale con alfa

```
UIColor* color = [self hex:@"#80FFFFFF"];
```

Esadecimale senza alfa (il `color` alpha sarà uguale a 1)

```
UIColor* color = [self hex:@"#FFFFFF"];
UIColor* color = [self hex:@"#FFF"];
```

Luminosità del colore regolata da UIColor

L'esempio di codice seguente ti darà una versione modificata di quel colore dove una percentuale più alta sarà più luminosa e una percentuale più bassa sarà più scura.

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;
}
```

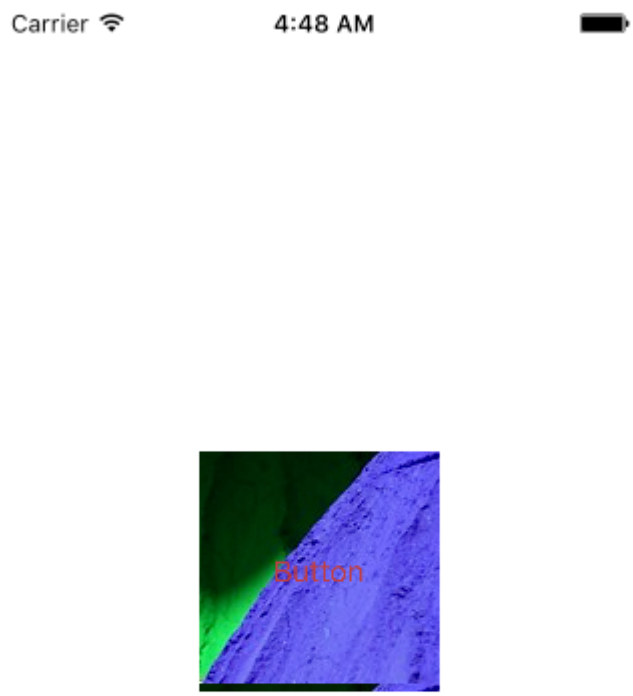
veloce

```
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 da un modello di immagine

È possibile creare un oggetto `UIColor` utilizzando un modello di immagine utilizzando il `UIColor(patternImage:_)` .

```
btn.backgroundColor = UIColor(patternImage: UIImage(named: "image")!)
```



Ombra più chiara e più scura di un determinato colore UIC

L'esempio di codice seguente mostra come ottenere una tonalità più chiara e più scura di un determinato colore, utile in applicazioni con temi dinamici

Per **colore più scuro**

```
+ (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;
}
```

Per **colori più chiari**

```
+ (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;
}
```

Vedi le differenze visive qui sotto, considerando il colore dato è `[UIColor orangeColor]`

Leggi UIColor online: https://riptutorial.com/it/ios/topic/956/uicolor

Capitolo 165: UIControl - Gestione degli eventi con i blocchi

Examples

introduzione

In genere, quando si utilizza `UIControl` o `UIButton` , si aggiunge un `selector` come azione di callback per quando un evento si verifica su un pulsante o un controllo, ad esempio l'utente che preme il pulsante o tocca il controllo.

Ad esempio, faremmo quanto segue:

```
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")
    }
}
```

Quando si tratta di `selector` , il compilatore deve solo sapere che esiste. Questo può essere fatto attraverso un `protocol` e non essere implementato.

Ad esempio, il seguente potrebbe danneggiare la tua applicazione:

```
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()
    }
}
```

Questo perché l'applicazione NON implementa la funzione `onButtonPress` .

Ora, se potessi fare tutto questo insieme all'inizializzazione del pulsante? Cosa accadrebbe se non fosse necessario specificare i callback e specificare invece i blocchi che possono essere aggiunti e rimossi in qualsiasi momento? Perché preoccuparsi di implementare i selettori?

Soluzione

```
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
    }
}
```

Quanto sopra è una semplice estensione su `UIControl` . Aggiunge una classe privata interna che ha una `func run(_ control: UIControl)` callback `func run(_ control: UIControl)` che viene utilizzata come azione degli eventi.

Successivamente utilizzeremo l' `object association` per aggiungere e rimuovere la destinazione perché non verrà mantenuta da `UIControl` .

La funzione del gestore di eventi restituisce un `Protocol` per nascondere i meccanismi interni della classe `Target` , ma anche per consentire all'utente di `enable` e `disable` il target in qualsiasi momento.

Esempio di utilizzo:

```
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()
    }
}
```

Leggi UIControl - Gestione degli eventi con i blocchi online:
https://riptutorial.com/it/ios/topic/3180/uicontrol---gestione-degli-eventi-con-i-blocchi

Capitolo 166: UIDatePicker

Osservazioni

`UIDatePicker` non eredita da `UIPickerView` , ma gestisce un oggetto di visualizzazione selettore personalizzato come una sottoview.

Examples

Crea un selettore di date

veloce

```
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)];
```

Impostazione della data minima-massima

È possibile impostare la data minima e massima che UIDatePicker può mostrare.

Data minima

```
[datePicker setMinimumDate:[NSDate date]];
```

Data massima

```
[datePicker setMaximumDate:[NSDate date]];
```

Modalità

UIDatePicker ha varie modalità di selezione.

```
enum UIDatePickerMode : Int {
    case Time
    case Date
    case DateAndTime
    case CountDownTimer
}
```

- `Time` : il selettore di date visualizza ore, minuti e (facoltativamente) una designazione AM / PM.
- `Date` : il selettore della data visualizza mesi, giorni del mese e anni.
- `DateAndTime` - Il selettore di date visualizza le date (come giorno unificato della settimana, mese e giorno dei valori del mese) più ore, minuti e (facoltativamente) una designazione AM / PM.
- `CountDownTimer` - Il selettore di date visualizza i valori di ora e minuti, ad esempio [1 | 53]. L'applicazione deve impostare un timer per attivare l'intervallo corretto e impostare il selettore di data mentre i secondi si spengono.

Impostazione proprietà datePickerMode

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.datePickerMode = .Date
```

Impostazione dell'intervallo minuto

È possibile modificare la proprietà `minuteInterval` per impostare l'intervallo visualizzato dalla ruota dei minuti. Il valore predefinito è 1, il valore massimo è 30.

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.minuteInterval = 15
```

Durata del conto alla rovescia

Il valore `NSTimeInterval` di questa proprietà indica i secondi da cui il selettore di date in modalità conto alla rovescia `NSTimeInterval` conto alla rovescia. Se la modalità del selettore di date non è `CountDownTimer` , questo valore viene ignorato. Il valore massimo è 86.399 secondi (23:59)

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.countDownDuration = 60 * 60
```

Leggi UIDatePicker online: https://riptutorial.com/it/ios/topic/5643/uidatepicker

Capitolo 167: UIDevice

Parametri

Proprietà	Descrizione
nome	Il nome che identifica il dispositivo.
systemName: String	Il nome del sistema operativo in esecuzione sul dispositivo rappresentato dal ricevitore.
modello: String	Il modello del dispositivo.
systemVersion: String	La versione corrente del sistema operativo ..

Osservazioni

La classe UIDevice fornisce un'istanza Singleton che rappresenta il dispositivo corrente. Da questa istanza è possibile ottenere informazioni sul dispositivo come nome assegnato, modello del dispositivo e nome e versione del sistema operativo.

Examples

Ottieni il nome del modello del dispositivo 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
}
```

Ottenere lo stato della batteria e il livello della batteria

```
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%, ...)
}
```

Identificazione del dispositivo e funzionamento

```
 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
```

Ottenere l'orientamento del dispositivo

```
UIDevice *deviceInfo = [UIDevice currentDevice];
int d = deviceInfo.orientation;
```

`deviceInfo.orientation` restituisce un valore UIDeviceOrientation che viene mostrato come di seguito:

```
UIDeviceOrientationUnknown 0
UIDeviceOrientationPortrait 1
UIDeviceOrientationPortraitUpsideDown 2
UIDeviceOrientationLandscapeLeft 3
UIDeviceOrientationLandscapeRight 4
UIDeviceOrientationFaceUp 5
UIDeviceOrientationFaceDown 6
```

Ascolto delle modifiche all'orientamento del dispositivo in un controller di visualizzazione:

```
- (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
}
```

Per disabilitare il controllo di eventuali cambiamenti di orientamento:

```
- (void)viewWillDisappear:(BOOL)animated {
    [super viewWillDisappear:animated];
    [[UIDevice currentDevice] endGeneratingDeviceOrientationNotifications];
}
```

Ottenere lo stato della batteria del dispositivo

```
//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");
    }
}
```

Utilizzo del sensore di prossimità

```
//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.");
    }
}
```

Leggi UIDevice online: https://riptutorial.com/it/ios/topic/4878/uidevice

Capitolo 168: UIFeedbackGenerator

introduzione

`UIFeedbackGenerator` e le sue sottoclassi offrono un'interfaccia pubblica al Taptic Engine® che si trova sui dispositivi iOS a partire da iPhone 7. Haptics, Taptics marchiati, fornisce un feedback tattile per gli eventi sullo schermo. Mentre molti controlli di sistema forniscono `UIFeedbackGenerator` , gli sviluppatori possono utilizzare sottoclassi `UIFeedbackGenerator` per aggiungere elementi aptici ai controlli personalizzati e ad altri eventi. `UIFeedbackGenerator` è una classe astratta che non dovrebbe essere utilizzata direttamente, piuttosto gli sviluppatori usano una delle sue sottoclassi.

Examples

Trigger Impact Haptic

L'esempio mostra come attivare un effetto tattile di impatto usando `UIImpactFeedbackGenerator` dopo aver premuto un pulsante.

veloce

```
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
```

Leggi UIFeedbackGenerator online: https://riptutorial.com/it/ios/topic/10048/uifeedbackgenerator

Capitolo 169: UIFont

introduzione

UIFont è una classe che viene utilizzata per ottenere e impostare le informazioni relative ai font. Si eredita da `NSObject` e si conforma a `Hashable` , `Equatable` , `CVarArg` e `NSCopying` .

Examples

Dichiarazione e inizializzazione di UIFont

È possibile dichiarare un `UIFont` come segue:

```
var font: UIFont!
```

`UIFont` ha più metodi `init()` :

- `UIFont.init(descriptor: UIFontDescriptor, size: CGFloat)`
- `UIFont.init(name: String, size: CGFloat)`

Pertanto, è possibile inizializzare un `UIFont` come questo:

```
let font = UIFont(name: "Helvetica Neue", size: 15)
```

Il carattere predefinito è `System` , dimensione `17` .

Cambiare il carattere di un'etichetta

Per cambiare il carattere del testo di un'etichetta, devi accedere alla sua proprietà `font` :

```
label.font = UIFont(name:"Helvetica Neue", size: 15)
```

Il codice sopra cambierà il carattere dell'etichetta su `Helvetica Neue` , taglia `15` . Attenzione che è necessario scrivere correttamente il nome del font, altrimenti genererà questo errore, perché il valore inizializzato sopra è un Optional, e quindi può essere nullo:

> Trovato in modo imprevisto nil durante lo srotolamento di un valore facoltativo

Leggi UIFont online: https://riptutorial.com/it/ios/topic/9792/uifont

Capitolo 170: UIGestureRecognizer

Examples

UITapGestureRecognizer

Inizializza `UITapGestureRecognizer` con un target, `self` in questo caso e `action` che è un metodo con un singolo parametro: un `UITapGestureRecognizer` .

Dopo l'inizializzazione, aggiungilo alla vista in cui dovrebbe riconoscere i tap in.

veloce

```
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 {

}
```

Esempio di eliminazione della tastiera tramite UITapGestureRecognizer:

Innanzitutto, si crea la funzione per il rifiuto della tastiera:

```
func dismissKeyboard() {
        view.endEditing(true)
    }
```

Quindi, aggiungi un riconoscitore di gesti tocco nel controller di visualizzazione, chiamando il metodo appena creato

```
let tap: UITapGestureRecognizer = UITapGestureRecognizer(target: self, action:
"dismissKeyboard")
        view.addGestureRecognizer(tap)
```

Esempio di ottenere la posizione del gesto UITapGestureRecognizer (Swift 3):

```
func handleTap(gestureRecognizer: UITapGestureRecognizer) {
print("tap working")
if gestureRecognizer.state == UIGestureRecognizerState.recognized
   {
     print(gestureRecognizer.location(in: gestureRecognizer.view))
   }
}
```

UIPanGestureRecognizer

I riconoscimenti dei gesti di Pan rilevano i gesti di trascinamento. L'esempio seguente aggiunge un'immagine a un controller di visualizzazione e consente all'utente di trascinarlo sullo schermo.

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];
}
```

veloce

```
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)
}
```

> **Nota:** sebbene `UIPanGestureRecognizer` sia utile per rilevare qualsiasi movimento di trascinamento, se si desidera rilevare un gesto di base come l'utente che trascina il dito verso sinistra / destra o su / giù, utilizzare `UISwipeGestureRecognizer` . `UIPanGestureRecognizer` è una scelta migliore se hai bisogno di accedere a metodi come `translationInView:` o `velocityInView:`

UITapGestureRecognizer (Double Tap)

Il doppio tocco, come un singolo tocco, utilizza anche `UITapGestureRecognizer` . Devi semplicemente impostare `numberOfTapsRequired` su `2` .

veloce

```
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"
}
```

Gli appunti

- Un esempio di progetto può essere trovato qui .
- È possibile riconoscere un tocco triplo impostando il `numberOfTapsRequired` di `numberOfTapsRequired` su `3` .

UILongPressGestureRecognizer

`UILongPressGestureRecognizer` ti consente di ascoltare una pressione prolungata su una vista. È possibile impostare la durata del ritardo prima che venga chiamato il metodo di azione.

veloce

```
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"
    }
}
```

Gli appunti

- Un progetto di esempio più completo può essere trovato qui .
- Modificare la `minimumPressDuration` per impostare la lunghezza della pressione prolungata.

UISwipeGestureRecognizer

I gesti di scorrimento consentono di ascoltare l'utente spostando rapidamente il dito sullo schermo in una determinata direzione.

veloce

```
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");
    }

}
```

Gli appunti

- Un esempio di progetto più completo può essere trovato qui .

UIPinchGestureRecognizer

I pizzichi sono un gesto a due dita in cui le dita si avvicinano o si allontanano l'una dall'altra. Questo gesto è generalmente utilizzato per ridimensionare una vista.

veloce

```
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
    }
}
```

Gli appunti

- Un esempio di progetto più completo può essere trovato qui .

UIRotationGestureRecognizer

Con `UIRotationGestureRecognizer` possibile ascoltare due dita che ruotano attorno a un centro. Questo è generalmente usato per ruotare una vista.

veloce

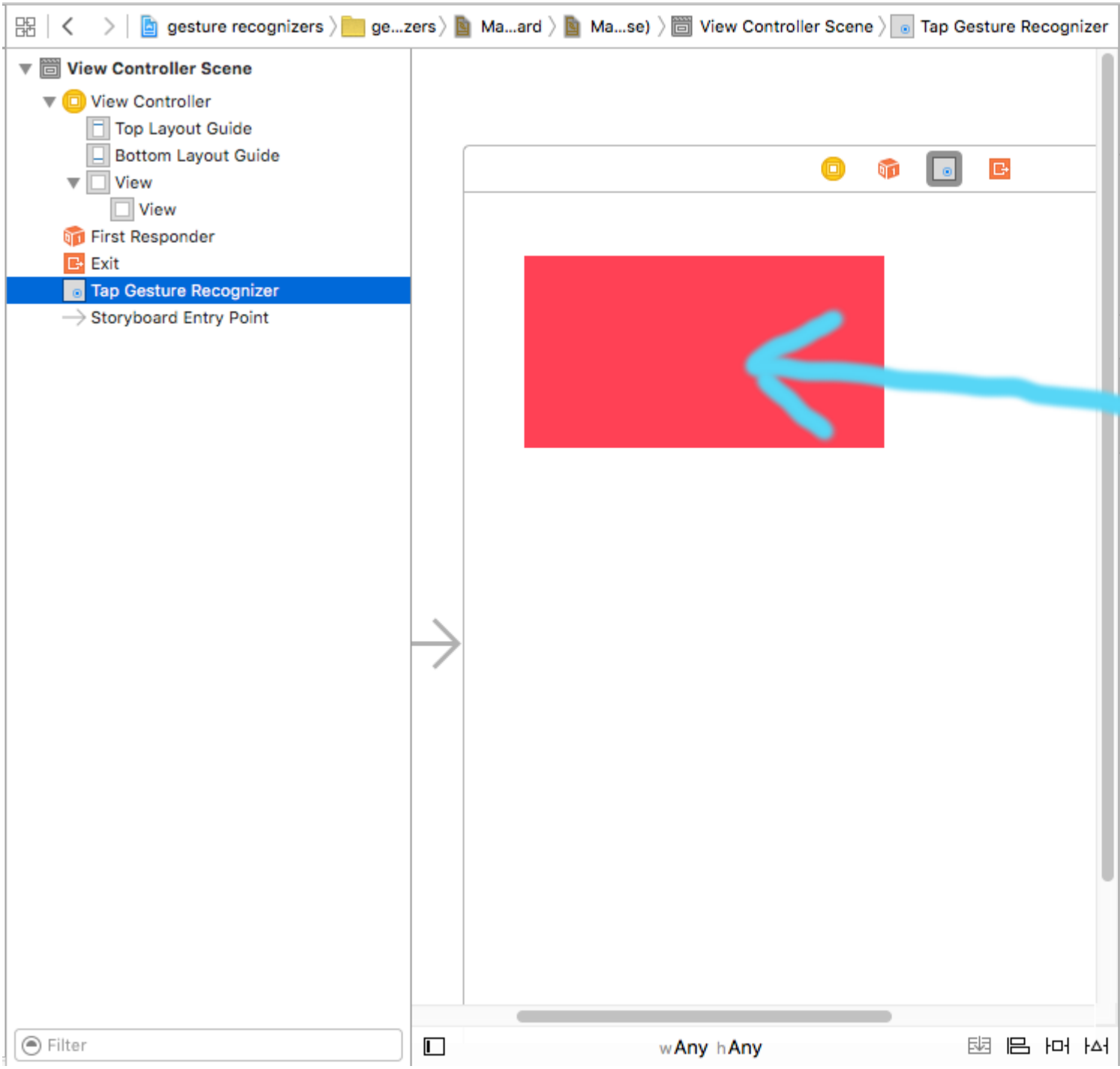
```
override func viewDidLoad() {
    super.viewDidLoad()

    // 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
    }
}
```

Gli appunti

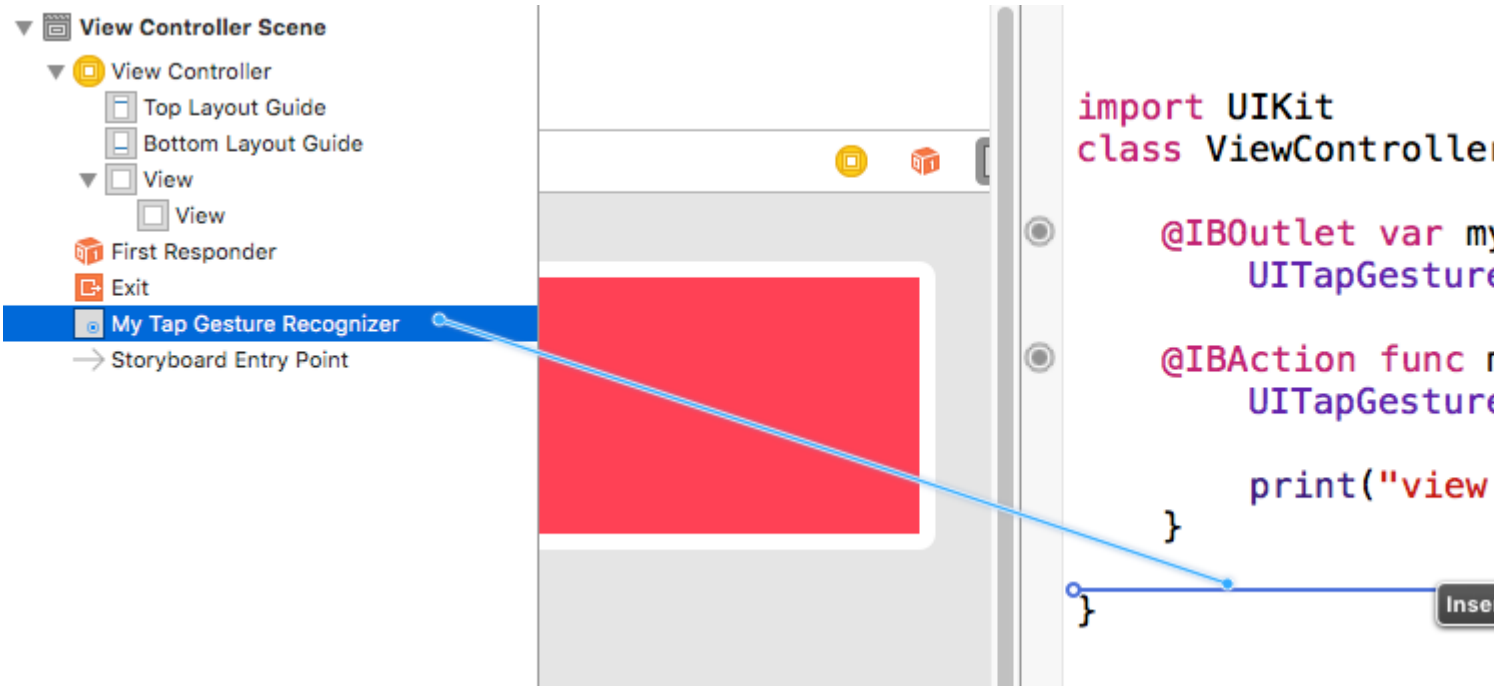
- Un esempio di progetto può essere trovato qui .

Aggiunta di un riconoscimento gestuale in Interface Builder

Trascina un riconoscitore di gesti dalla libreria degli oggetti sulla tua vista.



Controlla il trascinamento dal gesto nella struttura del documento al codice del controller di visualizzazione per creare un punto vendita e un'azione.



Gli appunti

- Questo esempio proviene da questo progetto di esempio più completo che dimostra i riconoscimenti gestuali.

Leggi UIGestureRecognizer online: https://riptutorial.com/it/ios/topic/1289/uigesturerecognizer

Capitolo 171: UIImage

Osservazioni

Argomento degli sviluppatori Apple per UIImage

Examples

Creazione di UIImage

Con l'immagine locale

veloce

```
let image = UIImage(named: "imageFromBundleOrAsset")
```

Objective-C

```
UIImage *image = [UIImage imageNamed:@"imageFromBundleOrAsset"];
```

> # Nota
>
> Il metodo `imageNamed` memorizza nella cache i contenuti dell'immagine. Il caricamento di molte immagini di grandi dimensioni in questo modo può causare avvisi di memoria insufficiente che possono portare alla chiusura dell'app. Questo `imageWithContentsOfFile` può essere risolto utilizzando il metodo `imageWithContentsOfFile` di `UIImage` , che non utilizza la memorizzazione nella cache.

Con NSData

veloce

```
let imageData = Data(base64Encoded: imageString, options:
Data.Base64DecodingOptions.ignoreUnknownCharacters)

let image = UIImage(data: imageData!)
```

Con UIColor

veloce

```
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();
```

Con il contenuto del file

Objective-C

Esempio:

```
UIImage *image = [UIImage imageWithContentsOfFile:[[NSBundle mainBundle]
pathForResource:[cellCountry objectForKey:@"Country_Flag"] ofType:nil]];
```

Utilizzo della matrice:

Esempio:

```
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;
    }
}
```

// Usando l'array di immagini per le animazioni qui:

```
self.myImageView.animationImages = imageArray;
```

Creazione e inizializzazione di oggetti immagine con contenuti di file

Creazione e restituzione di un oggetto immagine caricando i dati dell'immagine dal file nel percorso specificato.

Esempio:

```
 UIImage *image = [UIImage imageWithContentsOfFile:[[NSBundle mainBundle]
pathForResource:[cellCountry objectForKey:@"Country_Flag"] ofType:nil]];
```

Utilizzo della matrice:

Esempio

```
 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;
```

Immagine ridimensionabile con tappi

Nell'esempio di una bolla di messaggi illustrata di seguito: gli angoli dell'immagine dovrebbero rimanere invariati, specificati da `UIEdgeInsets` , ma i bordi e il centro dell'immagine dovrebbero espandersi per coprire le nuove dimensioni.

```
let insets = UIEdgeInsetsMake(12.0, 20.0, 22.0, 12.0)
let image = UIImage(named: "test")
image?.resizableImageWithCapInsets(insets, resizingMode: .Stretch)
```

Confronto delle immagini

> Il metodo `isEqual:` è l'unico metodo affidabile per determinare se due immagini contengono gli stessi dati di immagine. Gli oggetti immagine che crei possono essere diversi l'uno dall'altro, anche quando li si inizializza con gli stessi dati dell'immagine memorizzati nella cache. L'unico modo per determinare la loro uguaglianza è usare il metodo `isEqual:` che confronta i dati reali dell'immagine. Il listato 1 illustra i modi corretti e incorretti per confrontare le immagini.
>
> Fonte: documentazione Apple

veloce

```
// 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.
}
```

Crea UIImage con UIColor

veloce

```
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:

Aggiungi questo metodo come estensione di `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;
}
```

Immagine sfumata con colori

Creazione di sfumature `UIImage` con colori in `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!
    }
}
```

Uso:

```
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;
}
```

Livello sfondo sfumato per limiti

```
+ (CALayer *)gradientBGLayerForBounds:(CGRect)bounds colors:(NSArray *)colors
{
    CAGradientLayer * gradientBG = [CAGradientLayer layer];
    gradientBG.frame = bounds;
    gradientBG.colors = colors;
    return gradientBG;
}
```

Converti UIImage in / dalla codifica base64

Codifica

```
//convert the image to NSData first
let imageData:NSData = UIImagePNGRepresentation(image)!
// convert the NSData to base64 encoding
let strBase64:String =
imageData.base64EncodedStringWithOptions(.Encoding64CharacterLineLength)
```

decodifica

```
let dataDecoded:NSData = NSData(base64EncodedString: strBase64, options:
NSDataBase64DecodingOptions(rawValue: 0))!
let decodedimage:UIImage = UIImage(data: dataDecoded)!
```

Fai un'istantanea di un 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
```

Applica UIColor a UIImage

Utilizza la stessa UIImage con più app per tema base semplicemente applicando UIColor all'istanza UIImage come segue.

```
// *** 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
```

Ora diciamo che vuoi fare lo stesso con UIImageView quindi utilizzare il seguente codice

```
[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.
```

Cambia colore UIImage

Swift Aggiungi questa estensione a 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
        }
    }
}
```

Quindi, per cambiare il colore del tuo UIImage

```
my_image.maskWithColor(UIColor.blueColor())
```

Trovato a questo link

Leggi UIImage online: https://riptutorial.com/it/ios/topic/1409/uiimage

Capitolo 172: UIImagePickerController

introduzione

UIImagePickerController fornisce una soluzione quasi pronta all'uso per consentire all'utente di selezionare un'immagine dal proprio dispositivo o scattare una foto con la fotocamera e quindi presentare quell'immagine. Conformandosi a UIImagePickerControllerDelegate, puoi creare la logica che specifica nella tua app come presentare l'immagine e cosa fare con esso (usando didFinishPickingMediaWithInfo) e anche cosa fare se l'utente rifiuta di selezionare un'immagine o scattare una foto (usando imagePickerControllerDidCancel).

Examples

Uso generico di UIImagePickerController

Passaggio 1: creare il controller, impostare il delegato e conformarsi al protocollo

```
//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
```

nota: Realmente non implementeremo nulla definito in `UINavigationControllerDelegate` , ma

`UIImagePickerController` eredita da `UINavigationController` e modifica il comportamento di `UINavigationController` . Pertanto, dobbiamo ancora dire che il nostro controller di visualizzazione è conforme a `UINavigationControllerDelegate` .

Passaggio 2: ogni volta che è necessario mostrare `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];
```

Passaggio 3: implementare i metodi delegati:

```
//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];

}
```

Leggi UIImagePickerController online:
https://riptutorial.com/it/ios/topic/3023/uiimagepickercontroller

Capitolo 173: UIImageView

Examples

Crea un UIImageView

Per creare una `UIImageView` a livello di `UIImageView` , tutto ciò che devi fare è creare un'istanza di `UIImageView` :

```
//Swift
let imageView = UIImageView()

//Objective-C
UIImageView *imageView = [[UIImageView alloc] init];
```

È possibile impostare la dimensione e la posizione di `UIImageView` con un `CGRect` :

```
//Swift
imageView.frame = CGRect(x: 0, y: 0, width: 200, height: 200)

//Objective-C
imageView.frame = CGRectMake(0,0,200,200);
```

Oppure puoi impostare la dimensione durante l'inizializzazione:

```
//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;
```

> Nota: è necessario importare `UIKit` per utilizzare `UIImageView` .

Assegnazione di un'immagine a UIImageView

È possibile assegnare un'immagine a `UIImageView` durante l'inizializzazione o in seguito utilizzando la proprietà `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"];
```

Animazione di UIImageView

È possibile animare una `UIImageView` visualizzando rapidamente le immagini su di essa in una sequenza utilizzando le proprietà di animazione di `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
```

La proprietà `animationImages` è una `Array` di `UIImages` che viene eseguita dall'alto verso il basso quando viene attivata l'animazione.

La proprietà `animationDuration` è un `Double` dice quanti secondi per cui verrà eseguita l'animazione.

La proprietà `animationRepeatCount` è una `Int` che dice quante volte verrà eseguita l'animazione.

Per avviare e interrompere l'animazione, puoi chiamare i metodi appropriati per farlo:

```
imageView.startAnimating()
imageView.stopAnimating()
```

C'è un metodo `isAnimating()` che restituisce un valore `Boolean` indica se l'animazione è in esecuzione in un momento oppure no.

Si noti che questo non è un modo molto efficiente per creare animazioni: è piuttosto lento e richiede molte risorse. Prendi in considerazione l'utilizzo di livelli o sprite per risultati migliori

Rendere un'immagine in un cerchio o arrotondato

Questo esempio mostra come creare un `UIView` o `UIImageView` , arrotondato con un raggio di questo tipo:

Objective-C

```
someImageView.layer.cornerRadius = CGRectGetHeight(someImageView.frame) / 2;
someImageView.clipsToBounds = YES;
```

veloce

```
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)
```

Si suggerisce che se si utilizza layout automatico che si inserisce il `someImageView.layer.cornerRadius` codice `viewDidLayoutSubviews` . Ciò consentirà `cornerRadius` dell'immagine di aggiornarsi se l'immagine cambia dimensione.

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

```
    someImageView.layer.cornerRadius = someImageView.frame.size.width/2
    someImageView.layer.masksToBounds = true
}
```

UIImmagine mascherata con etichetta

Ciò rende l'immagine mascherata dalla forma delle lettere dell'etichetta:

Objective-C

```
self.maskImage.layer.mask = self.maskLabel.layer;
self.maskImage.layer.masksToBounds = YES;
```

Swift 3

```
maskImageView.mask = maskLabel
maskImageView.masksToBounds = true
```

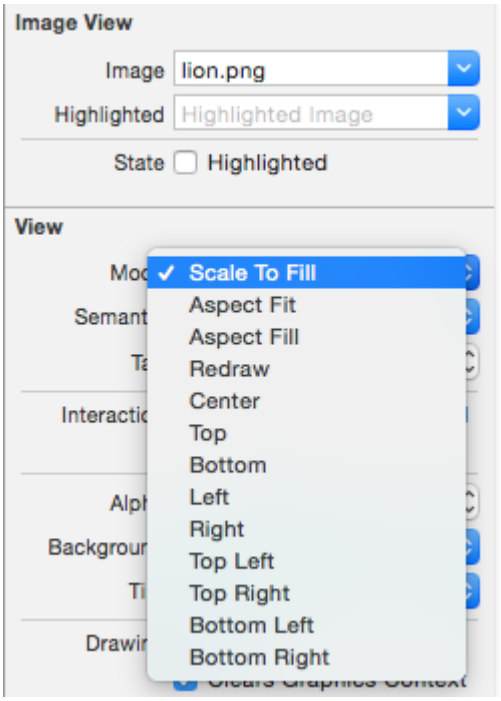
Ecco il risultato:

Cambia colore di un'immagine

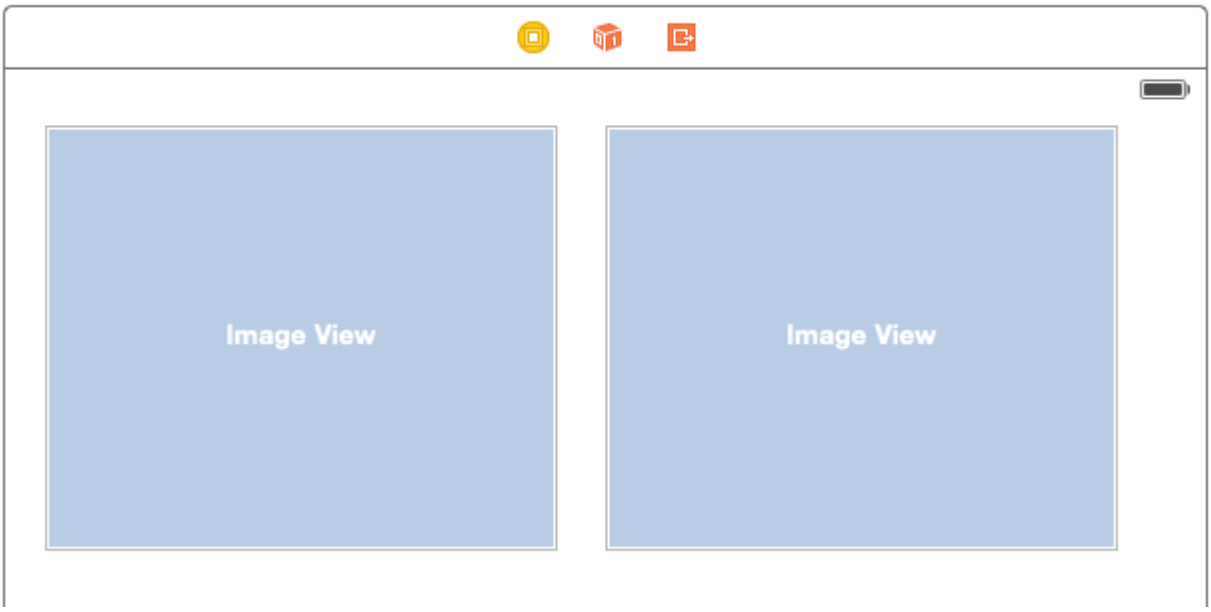
```
//Swift
imageView.tintColor = UIColor.redColor()
imageView.image = imageView.image?.imageWithRenderingMode(.AlwaysTemplate)

//Swift 3
imageView.tintColor = UIColor.red
imageView.image = imageView.image?.withRenderingMode(.alwaysTemplate)

//Objective-C
imageView.tintColor = [UIColor redColor];
imageView.image = [imageView.image imageWithRenderingMode:UIImageRenderingModeAlwaysTemplate]
```

In che modo la proprietà Mode influisce su un'immagine

La proprietà in modalità contenuto di una vista indica come deve essere definito il contenuto. In Interface Builder, le varie modalità possono essere selezionate nell'Inspector degli attributi.



Usiamo due immagini per vedere come funzionano le varie modalità.

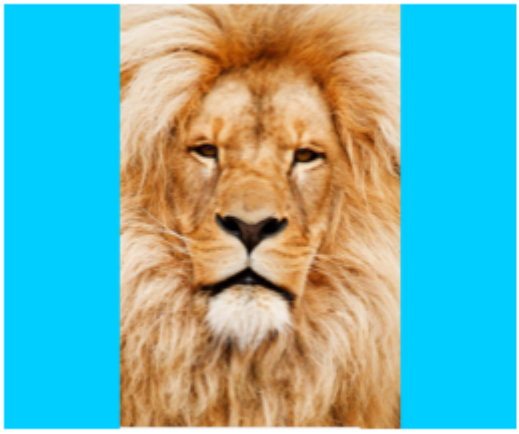
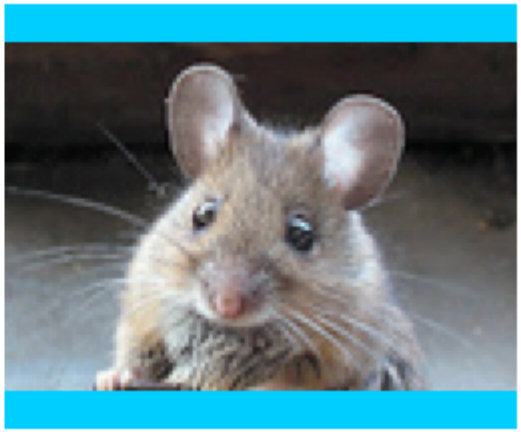


Ridimensiona per riempire

Le altezze e le larghezze dell'immagine sono allungate per adattarsi alle dimensioni di `UIImageView` .

Vestibilità

Il lato più lungo (altezza o larghezza) dell'immagine è allungato per adattarsi alla vista. Ciò rende l'immagine più grande possibile mentre mostra ancora l'intera immagine e non distorce l'altezza o la larghezza. (Ho impostato lo sfondo di `UIImageView` su blu in modo che la sua dimensione sia chiara.)

Aspetto Riempimento


Il lato più corto (altezza o larghezza) dell'immagine è allungato per adattarsi alla vista. Come "Aspect Fit", le proporzioni dell'immagine non sono distorte dalle proporzioni originali.

Ridisegna


Il ridisegno è solo per le visualizzazioni personalizzate che devono eseguire il ridimensionamento e il ridimensionamento. Non stiamo usando una vista personalizzata, quindi non dovremmo usare Redraw. Notare che qui `UIImageView` ci dà solo lo stesso risultato di Scale to Fill, ma sta facendo più lavoro dietro le quinte.

A proposito di Redraw, la documentazione di Apple dice:

> Le modalità di contenuto sono utili per riciclare i contenuti della vista, ma è anche possibile impostare la modalità di contenuto sul valore `UIViewContentModeRedraw` quando si desidera che le proprie viste personalizzate `UIViewContentModeRedraw` ridisegnate durante il ridimensionamento e il ridimensionamento delle operazioni. Impostando la modalità di contenuto della vista su questo valore, il sistema richiama il `drawRect:` della tua vista `drawRect:` metodo in risposta alle modifiche della geometria. In generale, dovresti evitare di usare questo valore quando possibile, e non dovresti certamente usarlo con le viste di sistema standard.

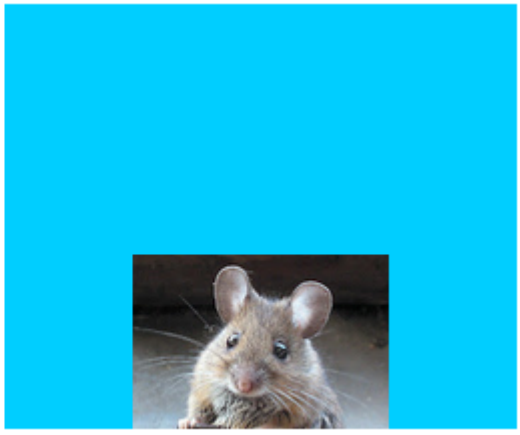
Centro

L'immagine è centrata nella vista, ma la lunghezza e la larghezza dell'immagine non sono allungate.

Superiore

Il bordo superiore dell'immagine è centrato orizzontalmente nella parte superiore della vista e la lunghezza e la larghezza dell'immagine non sono allungate.

Parte inferiore



Il bordo inferiore dell'immagine è centrato orizzontalmente nella parte inferiore della vista e la lunghezza e la larghezza dell'immagine non vengono allungate.

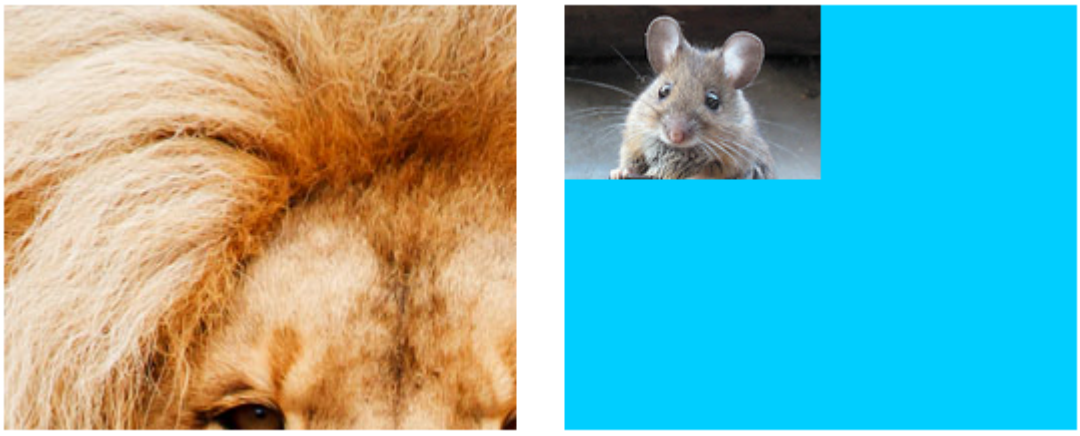
Sinistra

Il bordo sinistro dell'immagine è centrato verticalmente a sinistra della vista e la lunghezza e la larghezza dell'immagine non sono allungate.

Destra

Il bordo destro dell'immagine è centrato verticalmente a destra della vista e la lunghezza e la larghezza dell'immagine non sono allungate.

In alto a sinistra



L'angolo in alto a sinistra dell'immagine è posizionato nell'angolo in alto a sinistra della vista. La lunghezza e la larghezza dell'immagine non sono allungate.

In alto a destra

L'angolo in alto a destra dell'immagine è posizionato nell'angolo in alto a destra della vista. La lunghezza e la larghezza dell'immagine non sono allungate.

In basso a sinistra

L'angolo in basso a sinistra dell'immagine è posizionato nell'angolo in basso a sinistra della vista. La lunghezza e la larghezza dell'immagine non sono allungate.

In basso a destra

L'angolo in basso a destra dell'immagine è posizionato nell'angolo in basso a destra della vista. La lunghezza e la larghezza dell'immagine non sono allungate.

Gli appunti

- Questo esempio viene originariamente da qui .
- Se il contenuto (nel nostro caso l'immagine) ha le stesse dimensioni della vista (nel nostro caso `UIImageView`), la modifica della modalità di contenuto non farà alcuna differenza evidente.
- Consulta questa e questa domanda per una discussione sulle modalità di contenuto per le visualizzazioni diverse da `UIImageView` .
- In Swift, per impostare la modalità di contenuto a livello di programmazione, fai quanto segue:

```
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
```

Leggi UIImageView online: https://riptutorial.com/it/ios/topic/695/uiimageview

Capitolo 174: Uikit Dynamics

introduzione

UIKit Dynamics è un motore fisico completo del mondo reale integrato in UIKit. Ti permette di creare interfacce che sembrano reali aggiungendo comportamenti come gravità, attaccamenti, collisione e forze. Definisci i tratti fisici che vorresti che i tuoi elementi dell'interfaccia adottassero e il motore dinamico si occuperà del resto.

Osservazioni

Una cosa importante da tenere a mente quando si utilizza UIKit Dynamics è che le viste posizionate dall'animatore non possono essere facilmente posizionate da altri metodi di layout iOS comuni.

I nuovi arrivati in UIKit Dynamics spesso combattono con questo importante avvertimento. Posizionare i vincoli su una vista che è anche un elemento di un `UIDynamicBehavior` causerà probabilmente confusione poiché sia il motore di layout automatico che il motore di animatore dinamico combattono nella posizione appropriata. Allo stesso modo, il tentativo di impostare il frame direttamente su una vista controllata dall'animatore genererà in genere un'animazione distorta e un posizionamento imprevisto. L'aggiunta di una vista come elemento a un `UIDynamicBehavior` significa che l'animatore si assumerà la responsabilità di posizionare una vista e in quanto tale le modifiche delle posizioni della vista dovrebbero essere implementate tramite l'animatore.

È possibile impostare la cornice di una vista che viene aggiornata da un animatore dinamico, ma che dovrebbe essere immediatamente seguita dalla messaggistica dell'animatore per aggiornare il modello interno dell'animatore della gerarchia della vista. Ad esempio, se ho `UILabel` , `label` che è un elemento di un `UIGravityBehavior` posso spostarlo nella parte superiore dello schermo per vederlo cadere di nuovo dicendo:

veloce

```
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];
```

Dopo di che l'animatore applicherà il comportamento gravitazionale dalla nuova posizione

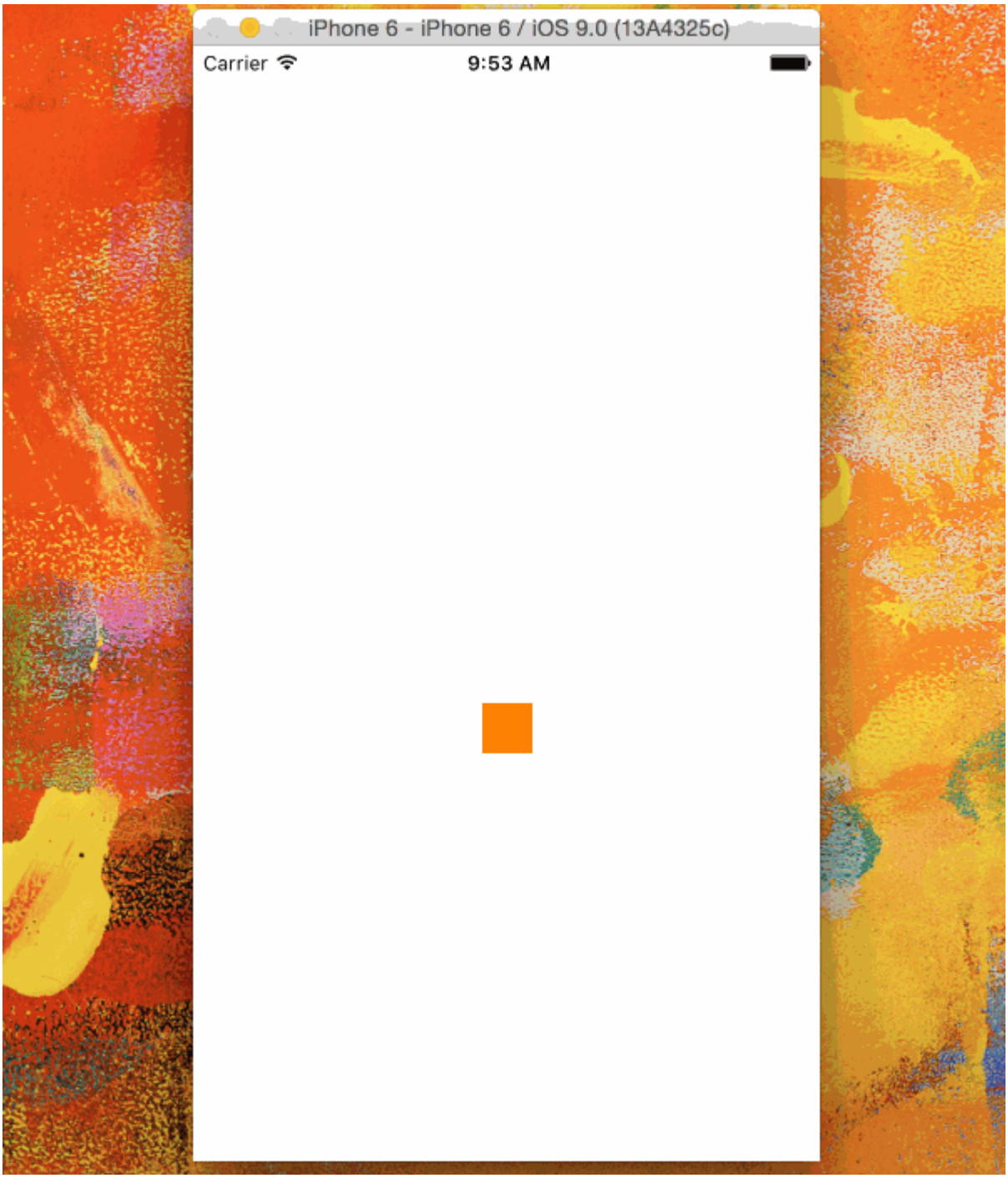
dell'etichetta.

Un'altra tecnica comune consiste nell'usare `UIDynamicBehaviors` per posizionare le viste. Ad esempio, se il posizionamento in vista sotto un evento di tocco è desiderato, creando un `UIAttachmentBehavior` e aggiorna il suo `anchorPoint` sia `touchesMoved` o `UIGestureRecognizer` azione s' è una strategia efficace.

Examples

The Falling Square

Consente di disegnare un quadrato al centro della nostra vista e farlo cadere verso il basso e fermarsi sul bordo inferiore della collisione con il bordo inferiore dello schermo.




```
@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)
}
```

Flick View basato sulla velocità del gesto

Questo esempio mostra come avere una vista per tracciare un gesto di pan e partire in modo basato sulla fisica.

veloce

```
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
```

Effetto "Sticky Corners" usando UIFieldBehaviors

Questo esempio mostra come ottenere un effetto simile a FaceTime in cui una vista è attratta dal punto in cui entra in una particolare regione, in questo caso due regioni in alto e in basso.

veloce

```
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

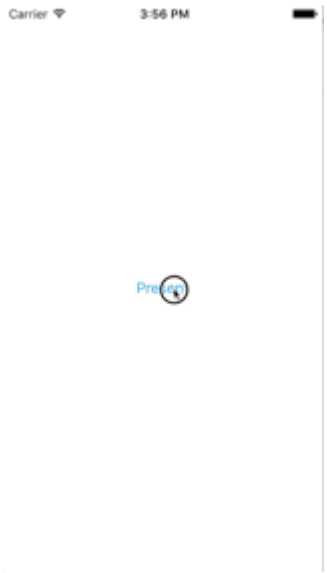
```
@interface ViewController ()

@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
```

Per ulteriori informazioni su `UIFieldBehaviors` è possibile vedere la sessione WWDC 2015 "Novità in Uikit Dynamics ed effetti visivi" e il relativo codice di esempio .

Transizione personalizzata guidata da UIDynamicBehavior



Questo esempio mostra come creare una transizione di presentazione personalizzata guidata da un `UIDynamicBehavior` composito. Possiamo iniziare creando un controller di visualizzazione che presenterà una modale.

veloce

```
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
```

`ModalViewController` il pulsante attuale, creiamo un `ModalViewController` e impostiamo lo stile di presentazione su `.custom` e impostiamo la sua `transitionDelegate` su se stesso. Questo ci permetterà di vendere un animatore che guiderà la sua transizione modale. Impostiamo anche la cornice della vista `modal` in modo che sia più piccola di quella a schermo intero.

Diamo ora un'occhiata a `ModalViewController` :

veloce

```
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
```

Qui creiamo il controller della vista che viene presentato. Anche perché `ModalViewController` ha il proprio `transitioningDelegate` , è anche responsabile della distribuzione di un oggetto che gestirà la sua animazione di transizione. Per noi ciò significa passare un'istanza della nostra sottoclasse `UIDynamicBehavior` composito.

Il nostro animatore avrà due diverse transizioni: una per la presentazione e l'altra per il licenziamento. Per la presentazione, la vista del controller della vista che presenta cadrà dall'alto. E per il licenziamento, la vista sembrerà oscillare da una corda e poi abbandonare. Poiché `DropOutAnimator` conforme a `UIViewControllerAnimatedTransitioning` maggior parte di questo lavoro verrà eseguita nella sua implementazione di `func animateTransition(using transitionContext: UIViewControllerContextTransitioning)` .

veloce

```
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;
        }
    }
}
```

Come comportamento composito, `DropOutAnimator` , può combinare un numero di comportamenti diversi per eseguire le sue animazioni di presentazione e di licenziamento. `DropOutAnimator` mostra anche come utilizzare il blocco di `action` di un comportamento per ispezionare le posizioni dei suoi oggetti e il tempo trascorso una tecnica che può essere utilizzata per rimuovere le viste che si spostano fuori dallo schermo o troncano le animazioni che devono ancora raggiungere la stasi.

Per ulteriori informazioni Sessione WWDC 2013 "Tecniche avanzate con UIKit Dynamics" e SOLPresentingFun

Transizione all'ombra con la fisica del mondo reale usando i parametri UIDynamic

Questo esempio mostra come eseguire una transizione di presentazione interattiva con la fisica "reale" simile alla schermata delle notifiche di iOS.

Per cominciare, abbiamo bisogno di un controller di visualizzazione che presenta la sfumatura sopra. Questo controller di visualizzazione fungerà anche da nostro `UIViewControllerTransitioningDelegate` per il nostro controller di visualizzazione presentato e venderà gli animatori per la nostra transizione. Quindi creeremo istanze dei nostri animatori interattivi (uno per la presentazione, uno per la rimozione). Creeremo anche un'istanza del controller della vista ombra, che, in questo esempio, è solo un controller di visualizzazione con un'etichetta. Poiché desideriamo lo stesso gesto di pan per guidare l'intera interazione, passiamo i riferimenti al controller della vista che presenta e all'ombra nei nostri animatori interattivi.

veloce

```
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<UIViewControllerAni

{
    return self.dismissingAnimator;
}

@end
```

Vogliamo davvero solo la voglia di presentare la nostra ombra attraverso una transizione interattiva, ma a causa di come funziona `UIViewControllerTransitioningDelegate` se non restituiamo un normale controller di animazione, il nostro controller interattivo non verrà mai utilizzato. Per questo `EmptyAnimator` creiamo una classe `EmptyAnimator` conforme a `UIViewControllerAnimatedTransitioning` .

veloce

```
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
```

Infine, è necessario creare effettivamente `ShadeAnimator` una sottoclasse di `UIDynamicBehavior` conforme a `UIViewControllerInteractiveTransitioning` .

veloce

```
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)];
```

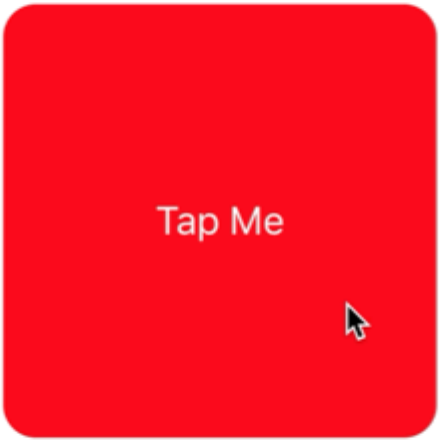
```
    }
    else if (sender.state == UIGestureRecognizerStateEnded || sender.state ==
UIGestureRecognizerStateCancelled)
    {
        UIView *view = self.isAppearing ? toVC.view : fromVC.view;
        BOOL isCancelled = self.isAppearing ? (velocity.y < 0.5 || view.center.y < 0.0) :
(velocity.y > 0.5 || 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
```

L'animatore fa scattare l'inizio della transizione quando inizia il gesto di panoramica. E sposta semplicemente la vista mentre il gesto cambia. Ma quando termina il gesto è quando `UIDynamicBehaviors` determina se la transizione deve essere completata o annullata. Per fare ciò utilizza un comportamento allegato e collisione. Per ulteriori informazioni, consultare la sessione WWDC del 2013 "Tecniche avanzate con UIKit Dynamics .

Mappa Posizione dinamica animazione Cambia in limiti

Questo esempio mostra come personalizzare il protocollo `UIDynamicItem` per mappare le modifiche di posizione di una vista dinamicamente animata alle modifiche dei limiti per creare un `UIButton` che si espande e si contrae in modo elastico.



Per iniziare abbiamo bisogno di creare un nuovo protocollo che implementi `UIDynamicItem` ma che abbia anche una proprietà `bounds` impostabile e gettabile.

veloce

```
protocol ResizableDynamicItem: UIDynamicItem
{
    var bounds: CGRect { set get }
}
extension UIView: ResizableDynamicItem {}
```

Objective-C

```
@protocol ResizableDynamicItem <UIDynamicItem>
@property (nonatomic, readwrite) CGRect bounds;
@end
```

Creeremo quindi un oggetto wrapper che avvolgerà un `UIDynamicItem` ma `UIDynamicItem` modifiche al centro alla larghezza e all'altezza dell'oggetto. Forniremo anche passthroughs per i `bounds` e la `transform` dell'elemento sottostante. Ciò farà sì che qualsiasi modifica apportata dall'animatore dinamico ai valori xey centrale dell'elemento sottostante venga applicata agli elementi larghezza e altezza.

veloce

```
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
```

Infine, creeremo un `UIViewController` con un pulsante. Quando viene premuto il pulsante creeremo `PositionToBoundsMapping` con il pulsante come elemento dinamico spostato. Creiamo un `UIAttachmentBehavior` alla sua posizione corrente, quindi aggiungiamo un `UIPushBehavior` istantaneo ad esso. Tuttavia, poiché abbiamo mappato le modifiche, il pulsante non si sposta ma cresce e si restringe.

veloce

```
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
```

Per ulteriori informazioni consultare Catalogo UIKit Dynamics

Leggi Uikit Dynamics online: https://riptutorial.com/it/ios/topic/9479/uikit-dynamics

Capitolo 175: Uikit Dynamics con UICollectionView

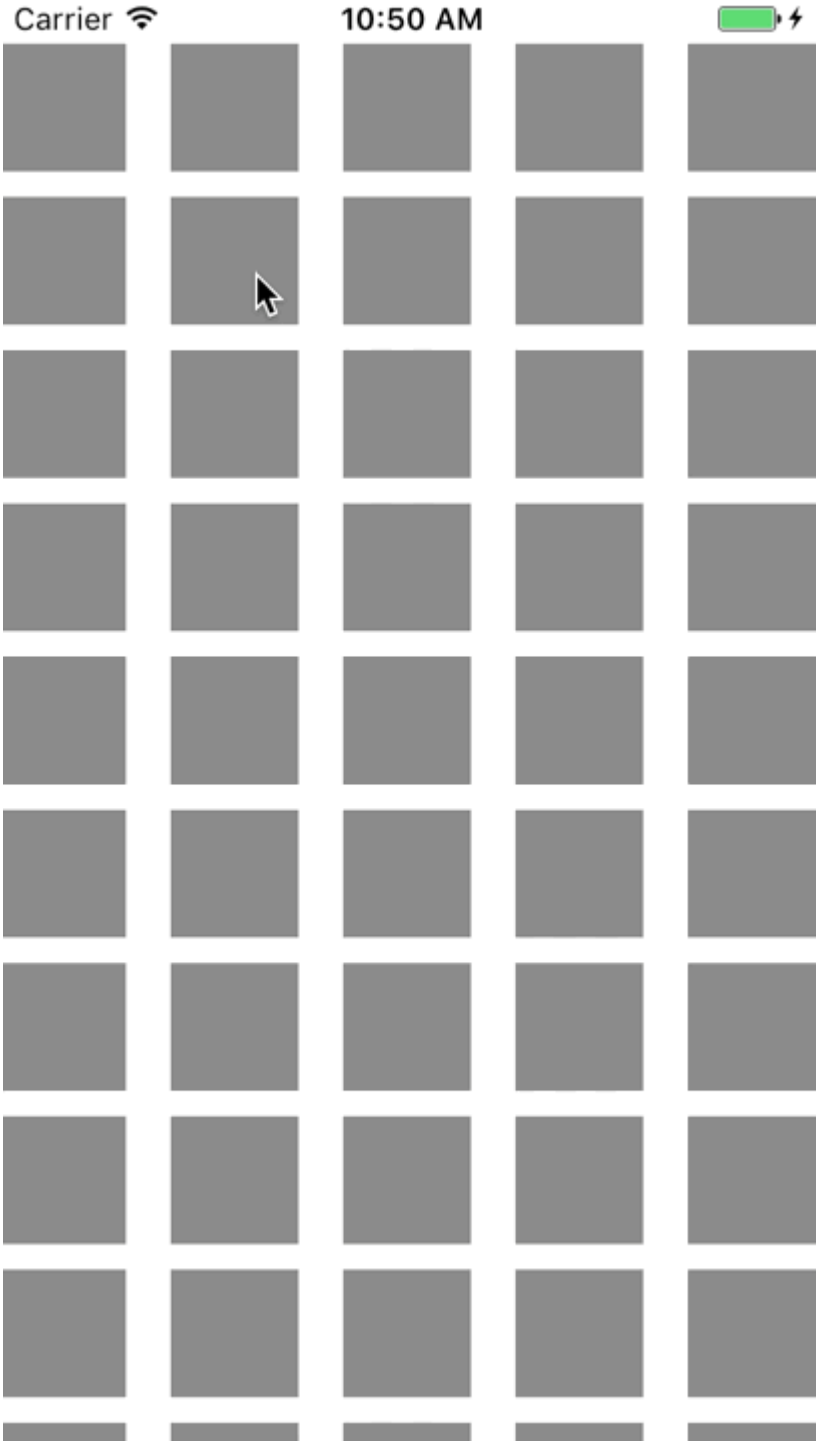
introduzione

UIKit Dynamics è un motore fisico integrato in UIKit. Uikit Dynamics offre un set di API che offre interoperabilità con `UICollectionView` e `UICollectionViewLayout`

Examples

Creazione di un comportamento di trascinamento personalizzato con UIDynamicAnimator

Questo esempio mostra come creare un comportamento di trascinamento personalizzato sottoclassando `UIDynamicBehavior` e sottoclasse `UICollectionViewFlowLayout` . Nell'esempio, abbiamo `UICollectionView` che consente la selezione di più elementi. Quindi con un gesto di pressione prolungata questi elementi possono essere trascinati in un'animazione elastica "elastica" guidata da un `UIDynamicAnimator` .



Il comportamento di trascinamento viene prodotto combinando un comportamento di basso livello che aggiunge un `UIAttachmentBehavior` agli angoli di un `UIDynamicItem` e un comportamento di alto livello che gestisce il comportamento di basso livello per un numero di `UIDynamicItems` .

Possiamo iniziare creando questo comportamento di basso livello, chiameremo `RectangleAttachmentBehavior`

veloce

```
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
```

Successivamente possiamo creare il comportamento di alto livello che combinerà un numero di `RectangleAttachmentBehavior` .

veloce

```
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
```

Ora con i nostri comportamenti in atto, il passo successivo è aggiungerli alla nostra vista raccolta quando. Poiché normalmente vogliamo un layout di griglia standard, possiamo sottoclasse `UICollectionViewFlowLayout` e cambiamo solo gli attributi durante il trascinamento. Lo facciamo principalmente attraverso l'override di `layoutAttributesForElementsInRect` e l'utilizzo degli `UIDynamicAnimator's` metodo di convenienza di `itemsInRect` .

veloce

```
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
```

Infine, creeremo un controller di visualizzazione che creerà il nostro `UICollectionView` e gestirà la nostra lunga pressione.

veloce

```
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
```

Per maggiori informazioni Sessione WWDC 2013 "Tecniche avanzate con dinamica UIKit"

Leggi Uikit Dynamics con UICollectionView online: https://riptutorial.com/it/ios/topic/10079/uikit-dynamics-con-uicollectionview

Capitolo 176: UILabel

introduzione

La classe UILabel implementa una visualizzazione di testo di sola lettura. È possibile utilizzare questa classe per disegnare una o più righe di testo statico, ad esempio quelle che è possibile utilizzare per identificare altre parti dell'interfaccia utente. La classe UILabel di base fornisce supporto per lo stile semplice e complesso del testo dell'etichetta. Puoi anche controllare gli aspetti dell'aspetto, ad esempio se l'etichetta utilizza un'ombra o disegna con un'evidenziazione. Se necessario, è possibile personalizzare ulteriormente l'aspetto del testo mediante sottoclassi.

Sintassi

- UILabel.numberOfLines: Int // ottiene o imposta il numero massimo di righe che l'etichetta può avere. 0 è illimitato
- UILabel.text: String? // ottiene o imposta il testo visualizzato dall'etichetta
- UILabel.textColor: UIColor! // ottiene o imposta il colore del testo sull'etichetta
- UILabel.tintColor: UIColor! // ottiene o imposta il colore della tinta dell'etichetta
- UILabel.attributedText: NSAttributedString? // ottiene o imposta il testo attribuito dell'etichetta
- UILabel.font: UIFont! // ottiene o imposta il carattere del testo sull'etichetta
- UILabel.textAlignment: NSTextAlignment // ottiene o imposta l'allineamento del testo

Osservazioni

UILabels sono viste che possono essere utilizzate per visualizzare una o più righe di testo. Contiene diversi modi di stilizzare il testo, come ombre, colori del testo e caratteri.

UILabels può anche visualizzare le stringhe attribuite, che è testo + markup in linea per applicare gli stili a porzioni di testo.

UILabel non è conforme al protocollo UIAppearance, quindi non è possibile utilizzare i metodi proxy di UIAppearance per personalizzare l'aspetto di UILabels. Vedi questa discussione per ulteriori informazioni.

Riferimento per gli sviluppatori Apple qui

Examples

Modifica del testo in un'etichetta esistente

La modifica del testo di una `UILabel` esistente può essere effettuata accedendo e modificando la proprietà `text` di `UILabel` . Questo può essere fatto direttamente usando letterali `String` o indirettamente usando variabili.

Impostazione del testo con valori letterali `String`

veloce

```
label.text = "the new text"
```

Objective-C

```
// Dot Notation
label.text = @"the new text";

// Message Pattern
[label setText:@"the new text"];
```

Impostazione del testo con una variabile

veloce

```
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];
```

Colore del testo

È possibile utilizzare la proprietà `textColor` dell'etichetta per applicare un colore di testo all'intero testo dell'etichetta.

veloce

```
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];
```

Applicazione del colore del testo a una parte del testo

Puoi anche variare il colore del testo (o altri attributi) di porzioni del testo usando `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;
```

veloce

```
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
```

Allineamento del testo

veloce

```
label.textAlignment = NSTextAlignment.left
//or the shorter
label.textAlignment = .left
```

Qualsiasi valore nella `NSTextAlignment` enum è valida: `.left` , `.center` , `.right` , `.justified` , `.natural`

Objective-C

```
label.textAlignment = NSTextAlignmentLeft;
```

Qualsiasi valore `NSTextAlignment` è valido: `NSTextAlignmentLeft` , `NSTextAlignmentCenter` , `NSTextAlignmentRight` , `NSTextAlignmentJustified` , `NSTextAlignmentNatural`

L'allineamento verticale in `UILabel` non è supportato `UILabel` : allinea verticalmente il testo in cima a

un UILabel

Crea un UILabel

Con una cornice

Quando conosci le dimensioni esatte che desideri impostare per l'etichetta, puoi inizializzare un `UILabel` con una cornice `CGRect` .

veloce

```
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];
```

Con layout automatico

È possibile aggiungere vincoli su un `UILabel` quando si desidera che iOS calcoli dinamicamente il frame in fase di runtime.

veloce

```
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];
```

Con 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}]]
```

La documentazione VFL può essere trovata qui

Dopo aver creato l'etichetta, assicurati di impostare le dimensioni tramite Auto Layout. Xcode mostrerà errori se è fatto in modo improprio.

Con Interface Builder

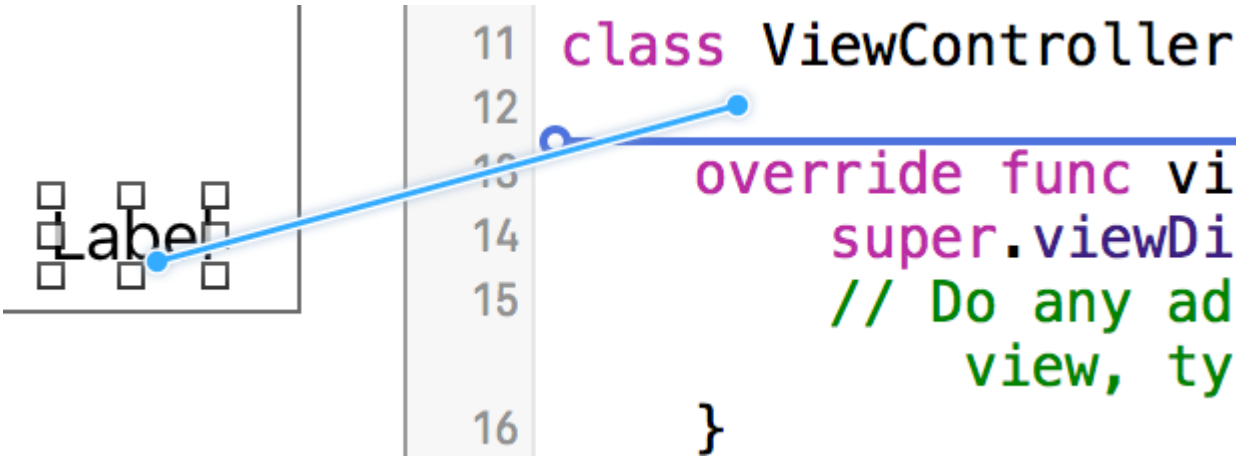
Inoltre, si utilizza Interface Builder per aggiungere una `UILabel` allo `Storyboard` o `.xib` file `.xib` trascinando `Label` dal pannello Libreria oggetti e rilasciandola in una vista nell'area di disegno:

Invece di specificare un frame (posizione e dimensione) per un `UILabel` a livello di `UILabel` , uno `Storyboard` o un file `.xib` consente di utilizzare Auto Layout per aggiungere vincoli al controllo.

Per accedere a questa etichetta creata da `storyboard` o `xib` creare un IBOutlet di questa etichetta.

Collegamento tra Interface Builder e View Controller

Dopo aver aggiunto un `UILabel` al vostro `Storyboard` o `.xib` il file è possibile collegare al vostro codice premendo `Control ^` e trascinando il mouse tra `UILabel` al `ViewController` , o si potrebbe trascinare al codice mentre clic destro per la sua avere lo stesso effetto



Nella finestra di dialogo delle proprietà, puoi impostare il nome di `UILabel` e impostarlo come `strong` o `weak` . Per ulteriori informazioni su `strong` e `weak` , vedere questo ,

L'altro modo è di creare lo sbocco programmaticamente come segue:

veloce

```
@IBOutlet weak var nameLabel : UILabel!
```

Objective-C

```
@property (nonatomic, weak) IBOutlet UILabel *nameLabel;
```

Imposta carattere

veloce

```
let label = UILabel()
```

Objective-C

```
UILabel *label = [[UILabel alloc] init];
or
UILabel *label = [UILabel new]; // convenience method for calling alloc-init
```

Cambia la dimensione del carattere predefinito

veloce

```
label.font = UIFont.systemFontOfSize(17)
```

Swift 3

```
label.font = UIFont.systemFont(ofSize: 17)
```

Objective-C

```
label.font = [UIFont systemFontOfSize:17];
```

Usa un peso specifico per il font

iOS 8.2

veloce

```
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

veloce

```
label.font = UIFont.boldSystemFontOfSize(17)
```

Swift3

```
label.font = UIFont.boldSystemFont(ofSize: 17)
```

Objective-C

```
label.font = [UIFont boldSystemFontOfSize:17];
```

Usa uno stile di testo di tipo dinamico.

La dimensione del carattere e del punto sarà basata sulla dimensione di lettura preferita dell'utente.

veloce

```
label.font = UIFont.preferredFontForTextStyle(UIFontTextStyleBody)
```

Swift 3

```
label.font = UIFont.preferredFont(forTextStyle: .body)
```

Objective-C

```
label.font = [UIFont preferredFontForTextStyle:UIFontTextStyleBody];
```

Utilizzare un font diverso completamente

veloce

```
label.font = UIFont(name: "Avenir", size: 15)
```

Objective-C

```
label.font = [UIFont fontWithName:@"Avenir" size:15];
```

Sostituisci la dimensione del carattere

Un modo per impostare la dimensione del font senza conoscere la famiglia di font è usare la proprietà **font** di `UILabel` .

veloce

```
label.font = label.font.fontWithSize(15)
```

Swift 3

```
label.font = label.font.withSize(15)
```

Objective-C

```
label.font = [label.font fontWithSize:15];
```

Usa carattere personalizzato Swift

Fare riferimento a questo link

Numero di linee

Quando si crea un'etichetta e si imposta il testo in modo che sia più di una singola riga che può essere visualizzata, verrà troncata e verrà visualizzata solo una riga di testo che termina con tre punti (...). Ciò è dovuto al fatto che una proprietà denominata `numberOfLines` è impostata su 1 e

pertanto verrà visualizzata una sola riga. È un errore comune nella gestione di `UILabel` , e molte persone pensano che sia un bug, oppure possono usare più di un'etichetta per mostrare più di una riga di testo, ma semplicemente modificando questa proprietà, possiamo dire a una `UILabel` di accetta fino al numero specificato di righe. Ad esempio, se questa proprietà è impostata su 5, l'etichetta può mostrare 1, 2, 3, 4 o 5 linee di dati.

Impostazione del valore a livello di codice

Per impostare questa proprietà, assegnagli semplicemente un nuovo numero intero:

veloce

```
label.numberOfLines = 2
```

Objective-C

```
label.numberOfLines = 2;
```

Nota

È possibile impostare questa proprietà su 0. Tuttavia, ciò non significa che non accetterà alcuna riga, ma significa che l'etichetta può avere tutte le linee necessarie (ovvero "Infinity"):

veloce

```
label.numberOfLines = 0
```

Objective-C

```
label.numberOfLines = 0;
```

Nota

Se l'etichetta ha un vincolo di altezza, il vincolo verrà rispettato. In questo caso, `label.numberOfLines = 0` potrebbe non funzionare come previsto.

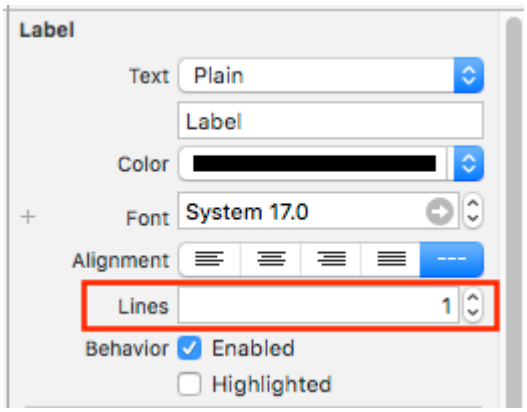
Nota

Per un testo multilinea più complesso, UITextView potrebbe essere più adatto. *

Impostazione del valore in Interface Builder

Invece di impostare `numberOfLines` di `numberOfLines` , è possibile utilizzare uno `Storyboard` o un file `.xib` e impostare la proprietà `numberOfLines` . In questo modo, otteniamo gli stessi risultati del codice precedente.

Come di seguito:



Dimensioni per adattarsi

Supponiamo che tu abbia un `UILabel` sullo `storyboard` e che tu abbia creato un `IBOutlet` per esso in `ViewController.swift` / `ViewController.m` e lo `labelOne` .

Per rendere le modifiche facilmente visibili, cambia il `backgroundColor` e `textColor` di labelOne nel metodo `viewDidLoad` :

La funzione `sizeToFit` viene utilizzata quando si desidera ridimensionare automaticamente un'etichetta in base al contenuto memorizzato al suo interno.

veloce

```
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];
```

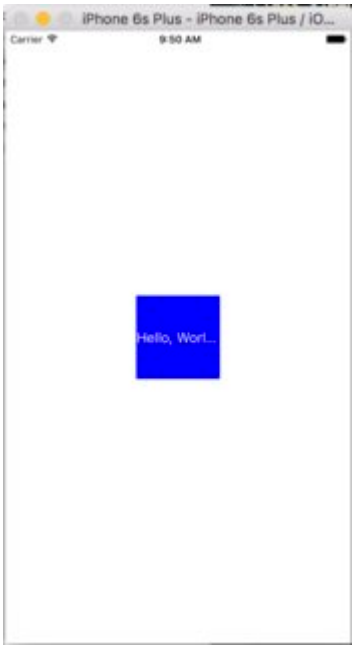
L'output per il codice sopra riportato è:

Come puoi vedere, non vi è alcun cambiamento in quanto il testo si adatta perfettamente a labelOne. sizeToFit cambia solo la cornice dell'etichetta.

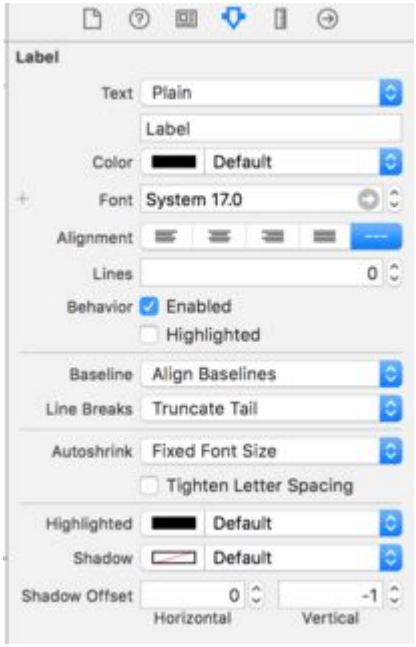
Cambiamo il testo in uno leggermente più lungo:

```
labelOne.text = "Hello, World! I’m glad to be alive!"
```

Ora, labelOne si presenta così:



Anche chiamare `sizeToFit` non cambia nulla. Questo perché, per impostazione predefinita, il numero di linee mostrato da UILabel è impostato su 1. Passiamo a zero sullo storyboard:



Questa volta, quando eseguiamo l'app, labelOne appare correttamente:

La proprietà `numberOfLines` può anche essere modificata nel file `ViewController` :

```
// Objective-C
labelOne.numberOfLines = 0;

// Swift
labelOne.numberOfLines = 0
```

Colore di sfondo

veloce

```
label.backgroundColor = UIColor.redColor()

label.backgroundColor = .redColor()
```

Swift 3

```
label.backgroundColor = UIColor.red
```

Objective-C

```
label.backgroundColor = [UIColor redColor];
```

Aggiungi ombre al testo

veloce

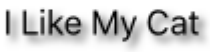
```
label1.layer.shadowOffset = CGSize(width: 3, height: 3)
label1.layer.shadowOpacity = 0.7
label1.layer.shadowRadius = 2
```

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;
```



Altezza variabile usando i vincoli

Puoi creare un `UILabel` con un'altezza dinamica utilizzando il layout automatico.

È necessario impostare `numberOfLines` su zero (0) e aggiungere un'altezza minima impostando un vincolo con una relazione di tipo `.GreaterThanOrEqual` sull'attributo `.Height`

iOS 6

veloce

```
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

veloce

```
label.numberOfLines = 0
label.translatesAutoresizingMaskIntoConstraints = false
label.heightAnchor.constraintGreaterThanOrEqualToConstant(20).active = true
```

LineBreakMode

Usando il codice

```
UILabel.lineBreakMode: NSLineBreakMode
```

veloce

```
label.lineBreakMode = .ByTruncatingTail
```

- .ByWordWrapping
- .ByCharWrapping
- .ByClipping
- .ByTruncatingHead
- .ByTruncatingTail
- .ByTruncatingMiddle

Swift 3

```
label.lineBreakMode = .byTruncatingTail
```

- .byWordWrapping
- .byCharWrapping
- .byClipping

- `.byTruncatingHead`
- `.byTruncatingTail`
- `.byTruncatingMiddle`

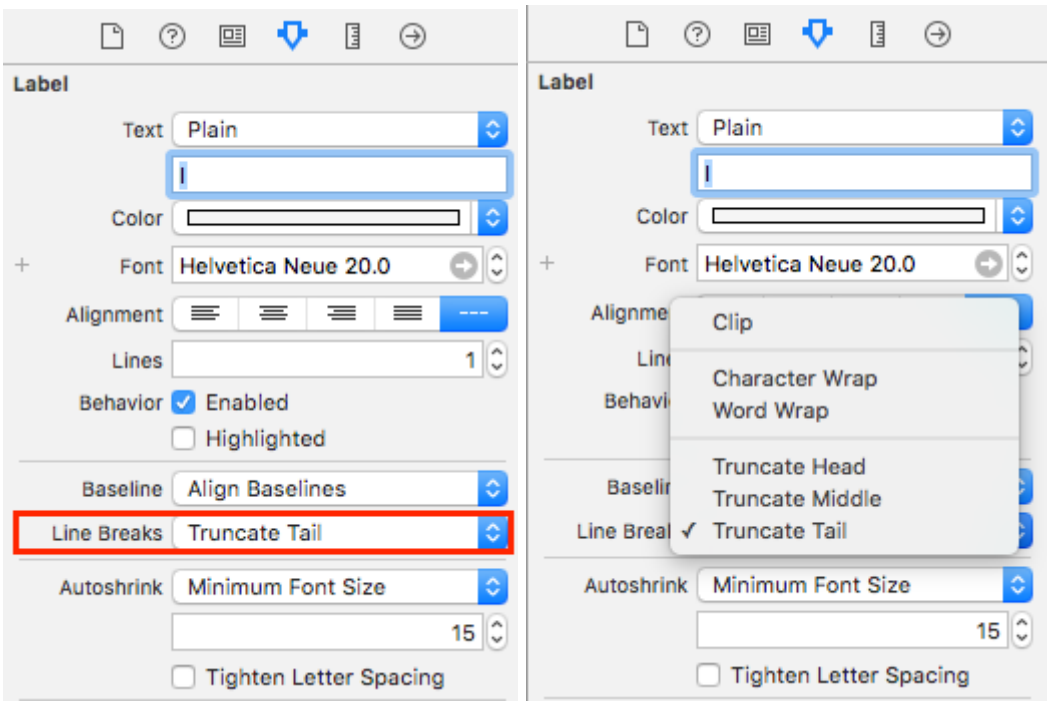
Objective-C

```
[label setLineBreakMode:NSLineBreakByTruncatingTail];
```

- `NSLineBreakByWordWrapping`
- `NSLineBreakByCharWrapping`
- `NSLineBreakByClipping`
- `NSLineBreakByTruncatingHead`
- `NSLineBreakByTruncatingTail`
- `NSLineBreakByTruncatingMiddle`

Usando lo storyboard

Questo può anche essere impostato nell'ispettore attributi di un UILabel:



costanti

- Word Wrapping - il wrapping avviene ai confini delle parole, a meno che la parola stessa non si adatti a una singola riga
- Char Wrapping: il wrapping avviene prima del primo carattere che non si adatta
- Ritaglio: le linee non sono semplicemente disegnate oltre il bordo del contenitore di testo
- Testa troncante - la linea viene visualizzata in modo che la fine si inserisca nel contenitore e il testo mancante all'inizio della linea sia indicato da un glifo con ellissi

- Truncating Tail - la linea viene visualizzata in modo che l'inizio si inserisca nel contenitore e il testo mancante alla fine della linea sia indicato da un glifo con ellissi
- Truncating Middle - la linea viene visualizzata in modo che l'inizio e la fine si inseriscano nel contenitore e il testo mancante nel mezzo sia indicato da un glifo con ellissi

Calcola i limiti di contenuto (per esempio altezze delle celle dinamiche)

Un caso d'uso comune per voler calcolare il fotogramma che un'etichetta occuperà è dimensionare opportunamente le celle della vista tabella. Il modo consigliato per farlo è utilizzare il metodo `NSString boundingRectWithSize:options:attributes:context:`

`options` prendono le `options` disegno per le stringhe:

- `NSStringDrawingUsesLineFragmentOrigin` deve essere utilizzato per le etichette con più righe
- `NSStringDrawingTruncatesLastVisibleLine` dovrebbe essere aggiunto utilizzando il carattere `|` operatore se ci sono un numero massimo di linee

`attributes` è un `NSDictionary` di attributi che `NSDictionary` sulle stringhe attribuite (lista completa: Apple Docs) ma i fattori che influenzano l'altezza includono:

- **NSFontAttributeName** : molto importante, la dimensione e la famiglia di caratteri è una parte critica delle dimensioni visualizzate dell'etichetta.
- **NSParagraphStyleAttributeName** : per personalizzare la modalità di visualizzazione del testo. Ciò include l'interlinea, l'allineamento del testo, lo stile di troncamento e alcune altre opzioni. Se non hai modificato esplicitamente nessuno di questi valori, non dovresti preoccuparti di questo, ma potrebbe essere importante se hai attivato alcuni valori su IB.

`context` dovrebbe essere `nil` poiché il caso d'uso principale di `NSStringDrawingContext` è quello di consentire il ridimensionamento del font per adattarlo a un rect specificato, il che non dovrebbe essere il caso se stiamo calcolando un'altezza dinamica.

Obiettivo 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
}
```

Viceversa, se si dispone di un numero massimo di righe impostato, è necessario innanzitutto calcolare l'altezza di una singola riga per assicurarsi che non si ottenga un valore più alto della dimensione consentita:

```
    // 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;
```

Etichetta cliccabile

> **NOTA:** nella maggior parte dei casi, è preferibile utilizzare un `UIButton` anziché creare un `UILabel` è possibile toccare. Utilizza questo esempio, se sei sicuro, che non vuoi utilizzare un `UIButton` per qualche motivo.

1. Crea etichetta
2. Abilita l'interazione dell'utente
3. Aggiungi `UITapGestureRecognizer`

La chiave per creare un `UILabel` cliccabile è consentire l'interazione dell'utente.

veloce

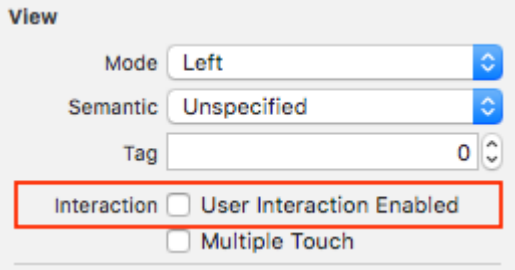
```
let label = UILabel()
label.userInteractionEnabled = true

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];
```

Impostazione "userInteractionEnabled" nell'ispettore degli attributi dello storyboard

Invece di usare il codice, puoi selezionare l'UILabel all'interno dello storyboard e controllare l'opzione:



Cornice dell'etichetta dinamica dalla lunghezza del testo sconosciuta

A volte dobbiamo ridimensionare un UILabel basato su contenuti dinamici in cui la lunghezza del testo è sconosciuta. In questo esempio, la larghezza di UILabel è fissata a 280 punti e l'altezza è infinita, diciamo 9999. Stima del fotogramma rispetto allo stile del testo e a 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;
```

veloce

```
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
```

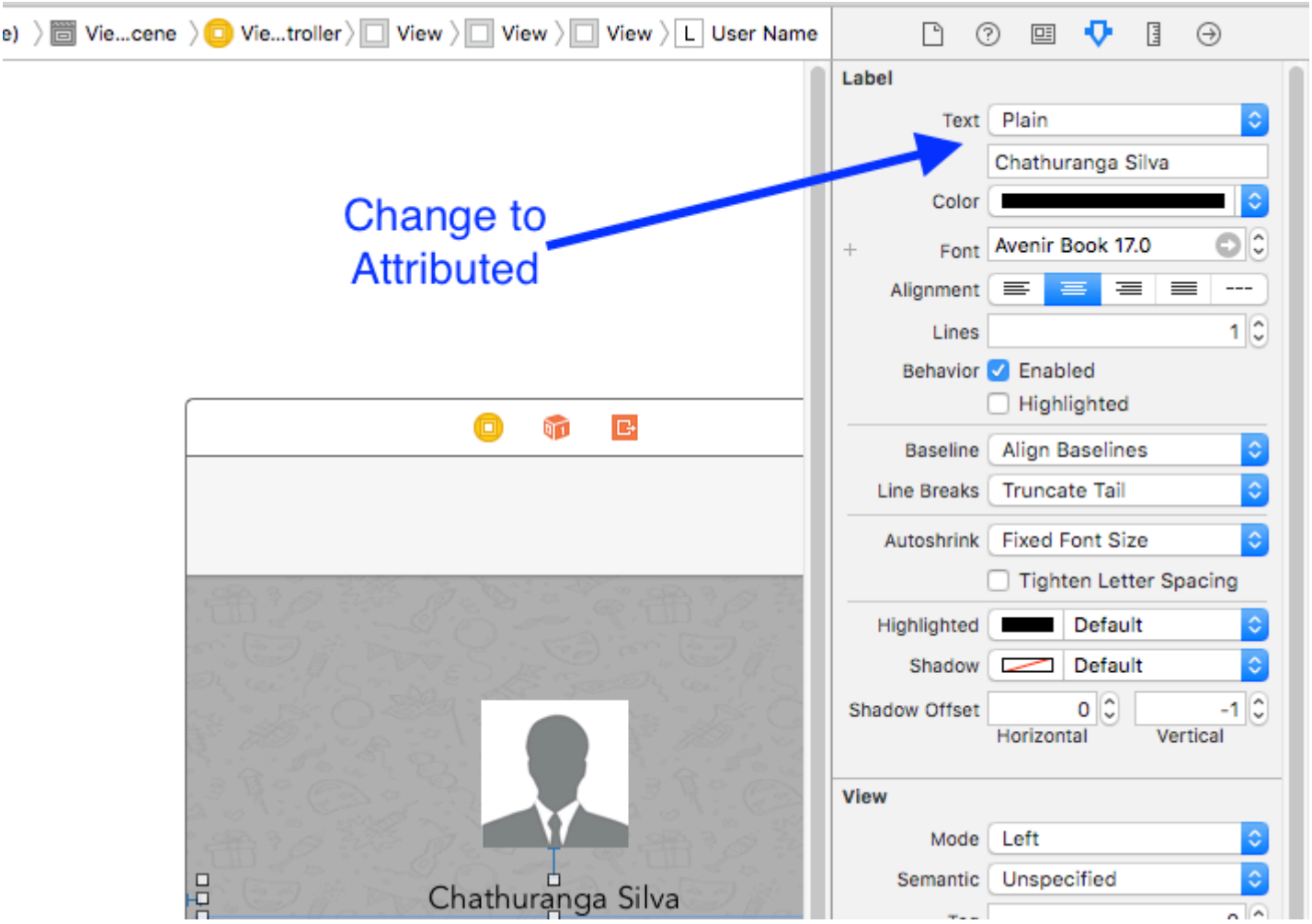
Etichetta attribuita testo

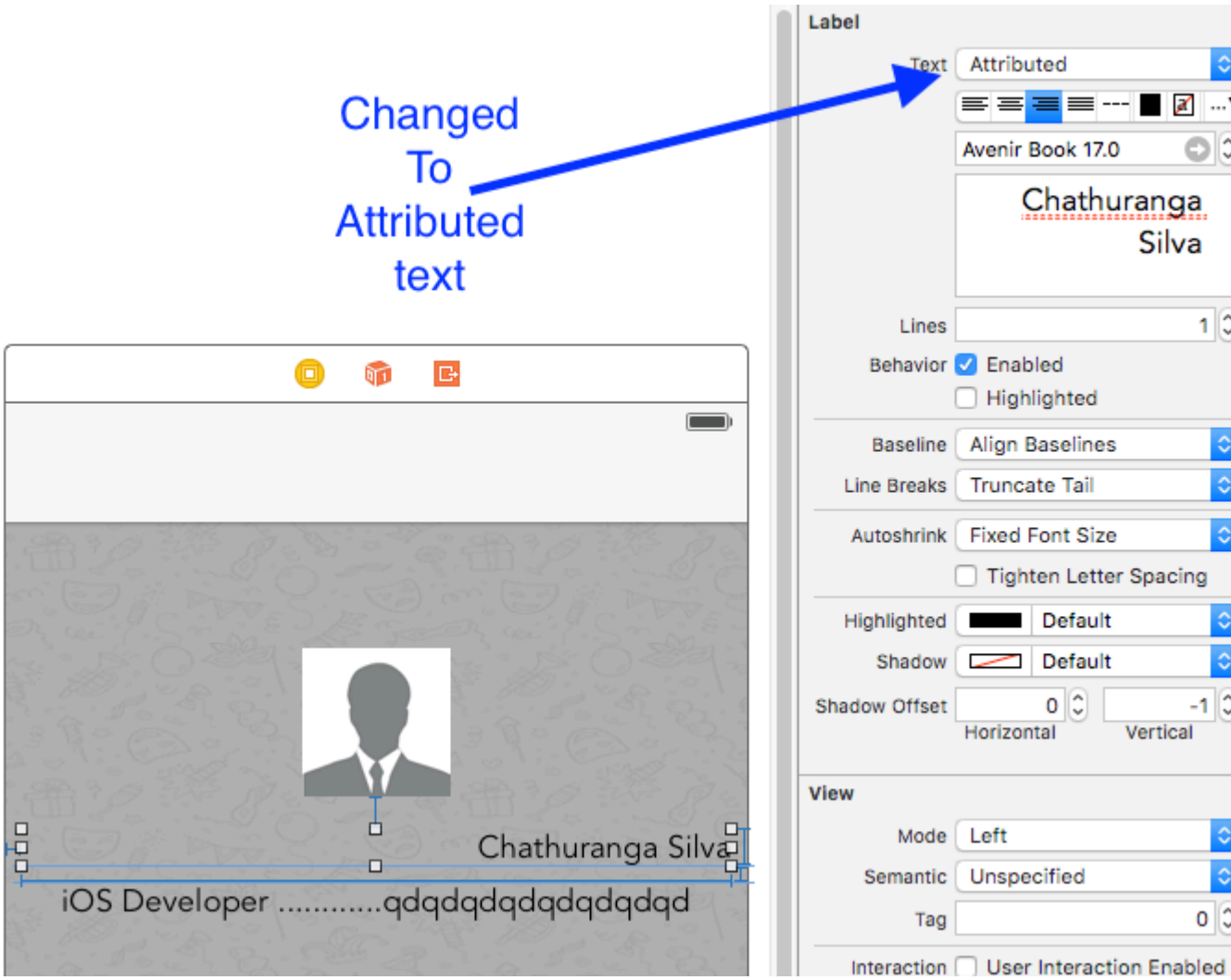
01. Sottolineatura del testo: - Linea singola / doppia, Strike Through: - Linea singola /

doppia

Passo 1

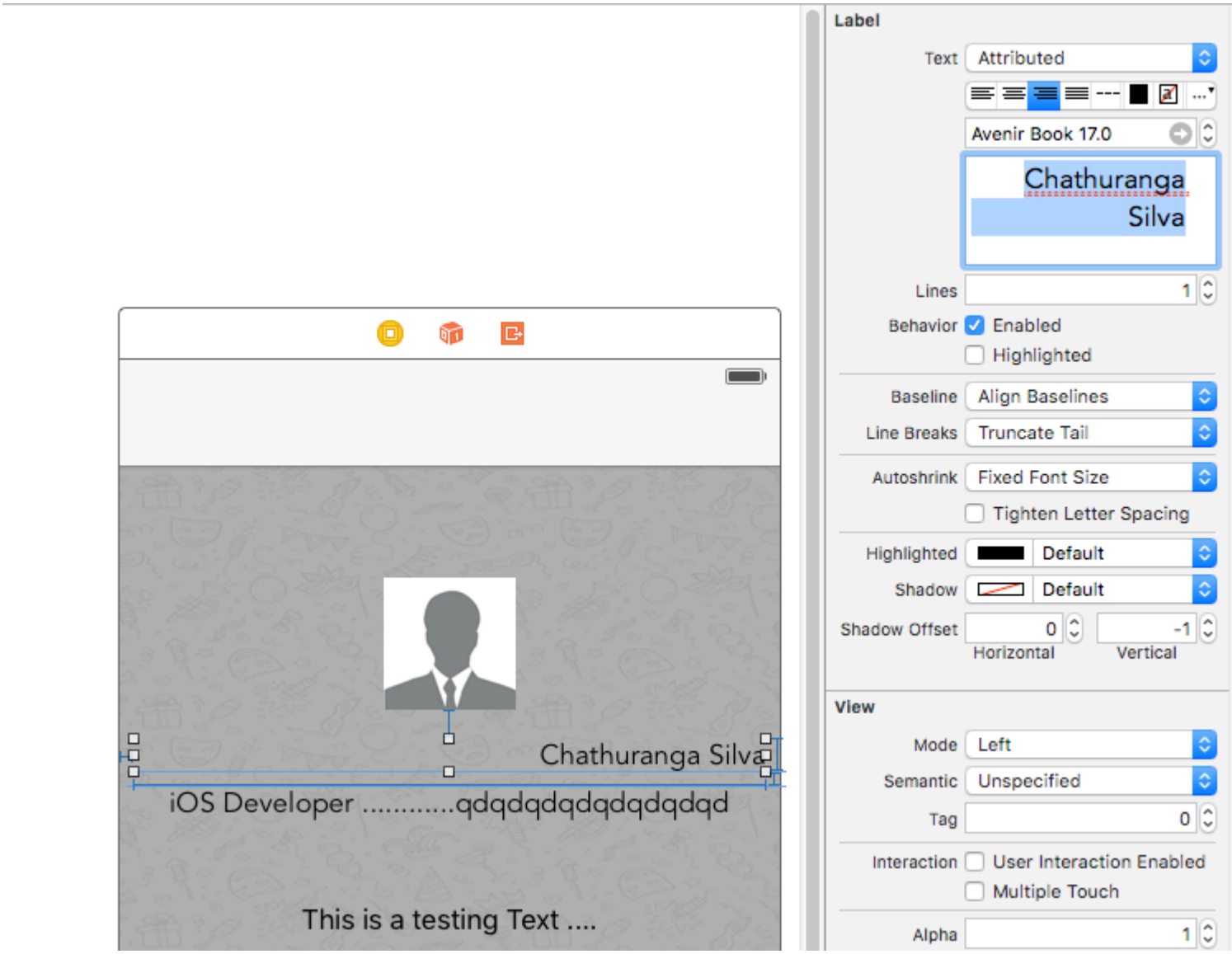
Seleziona l'etichetta e modifica il tipo di etichetta Pianura su Attribuito





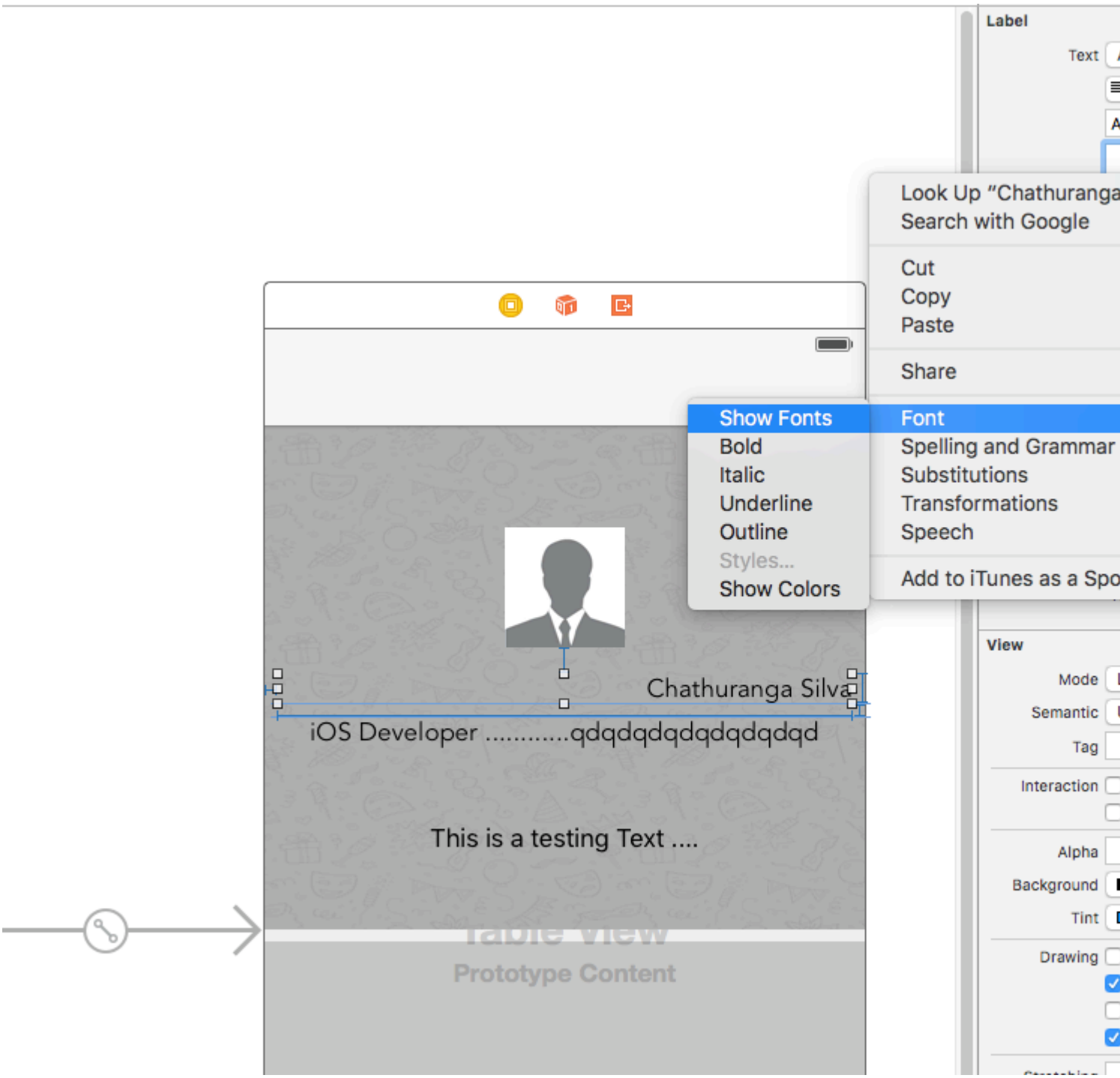
Passo 2

Fai clic sul testo dell'etichetta e fai clic con il pulsante destro del mouse



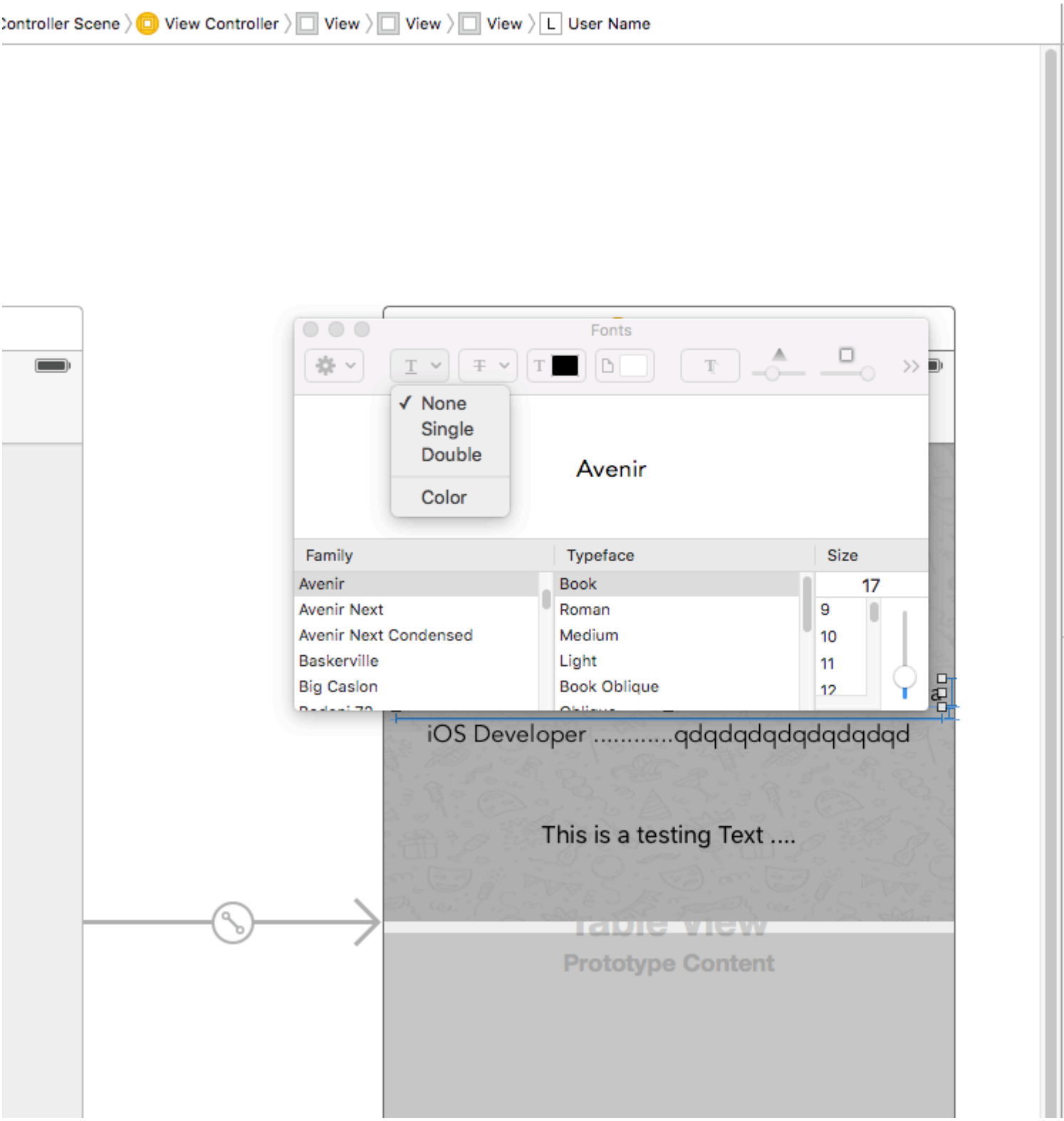
Passaggio 3

Quindi fare clic su Font -> Mostra caratteri

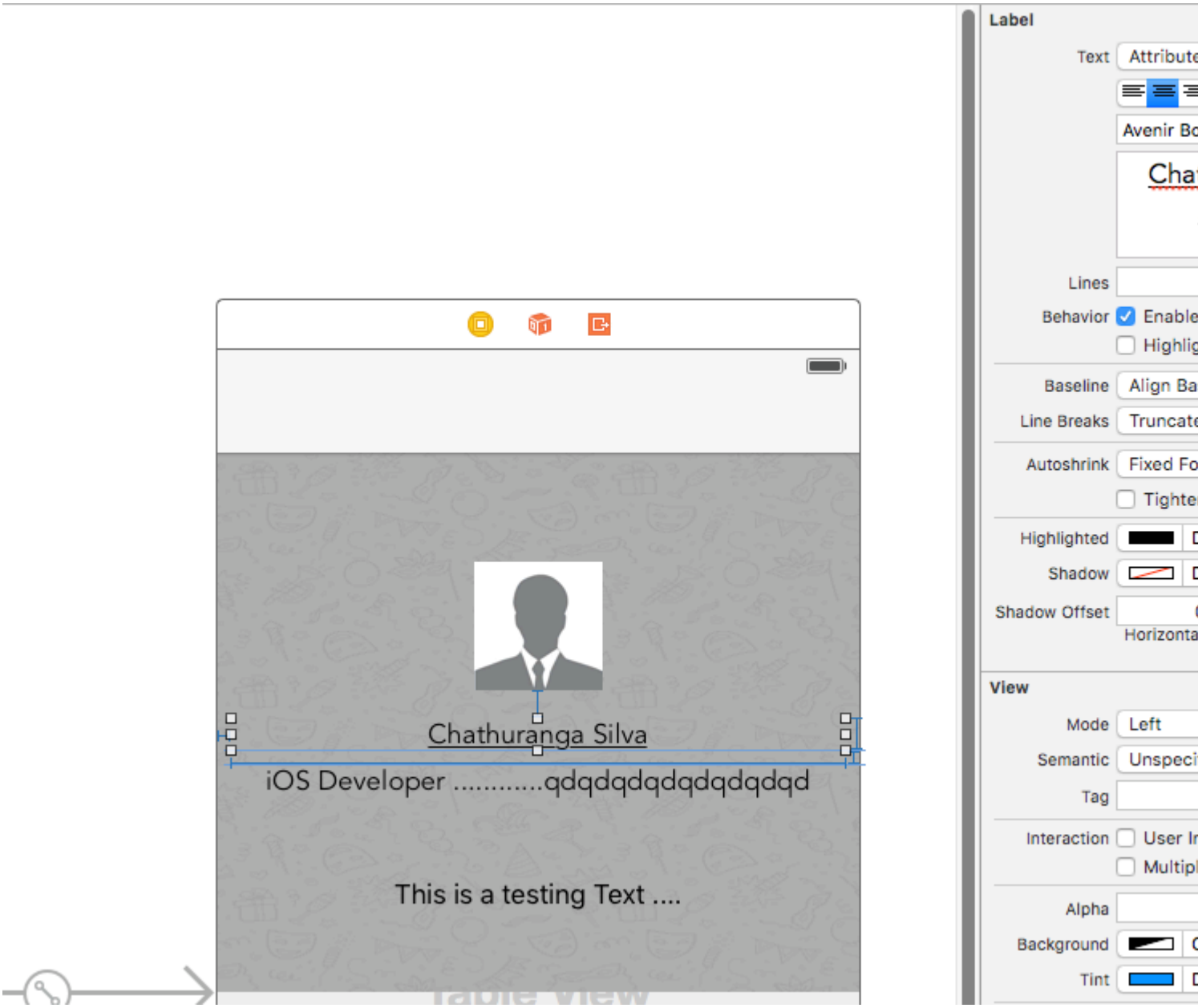


Passaggio 4

Quindi verrà visualizzata la visualizzazione dei caratteri e fare clic sul pulsante di sottolineatura per rendere il testo sottolineato o fare clic sul pulsante barrato per rendere il testo barrato. E selezionare una riga singola o una doppia linea.

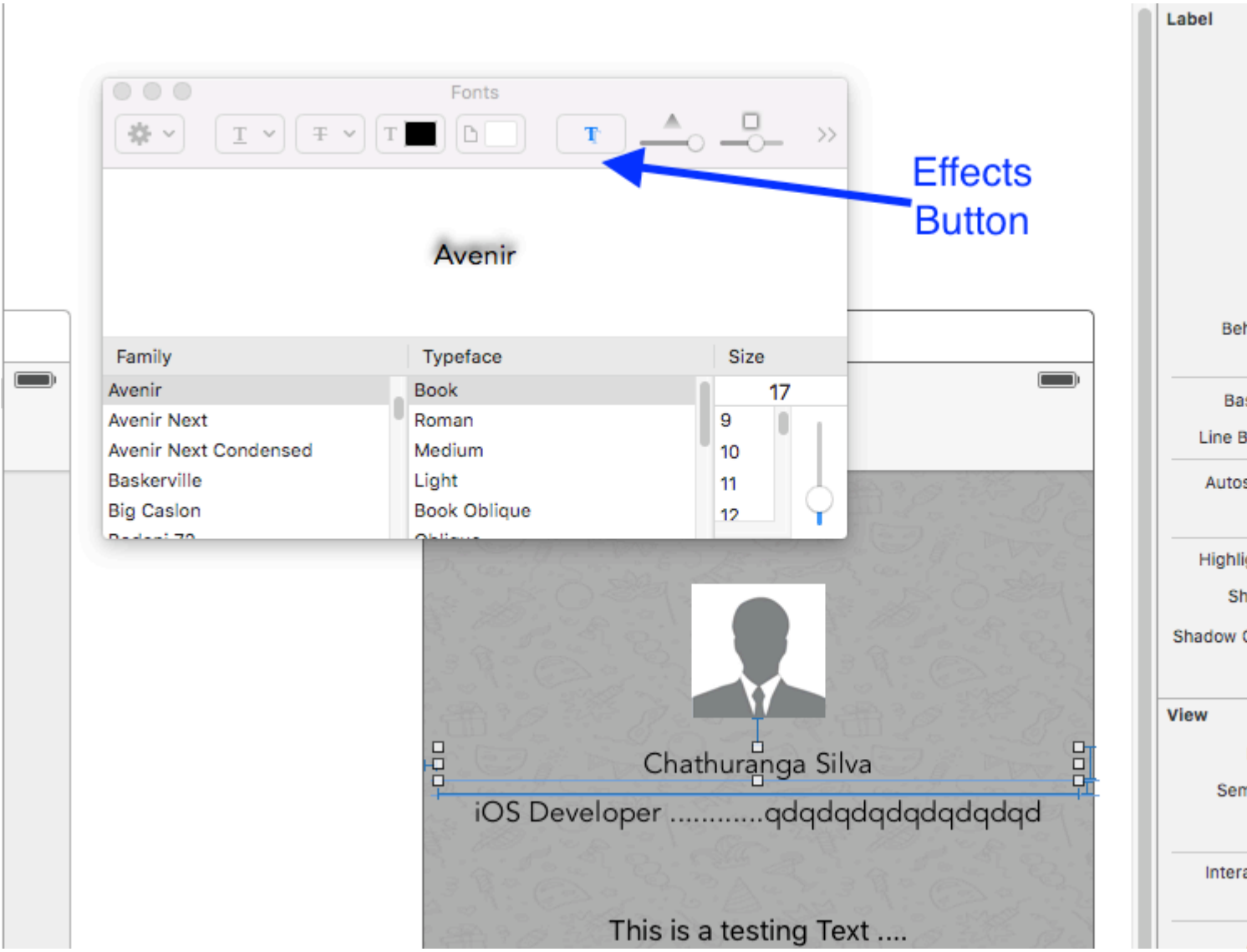


Infine, fai clic su invio e l'etichetta verrà mostrata come sottolineato o barrato in base alla selezione.

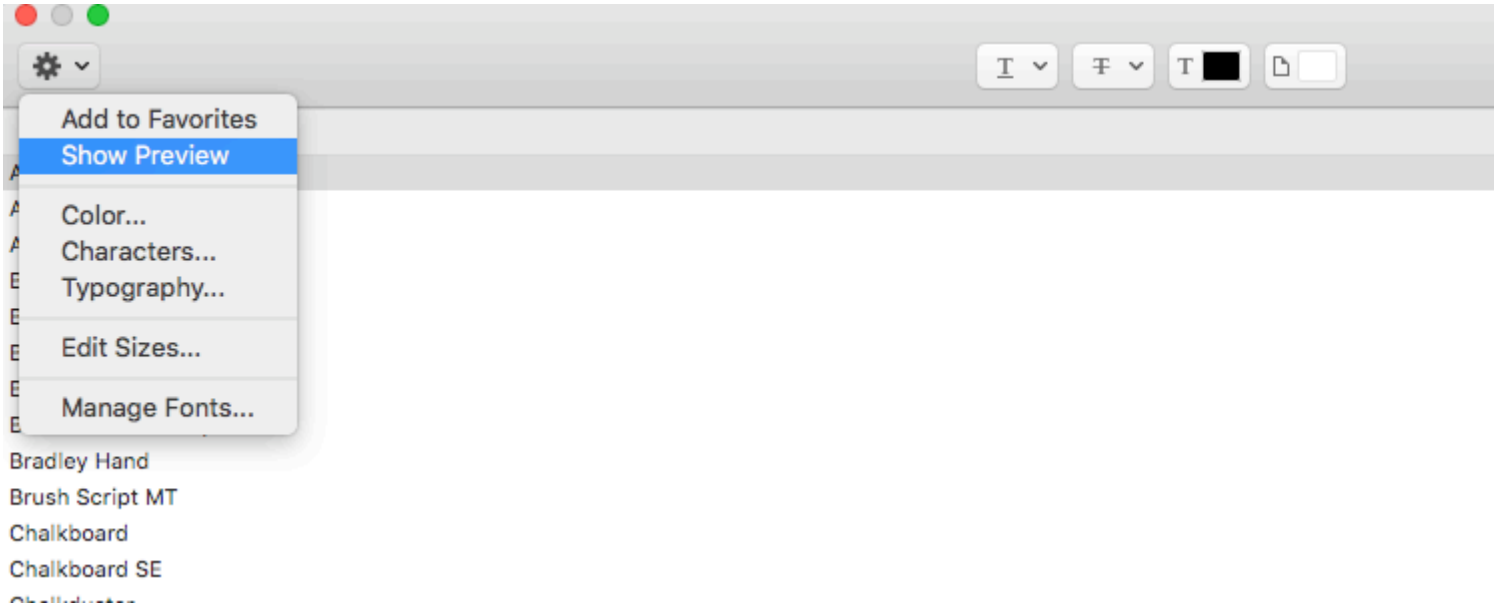


02. Aggiungi effetti shaddow / sfocatura dello sfondo del testo

Ottieni la vista Font come sopra descritto e fai clic sul pulsante degli effetti.



Se non vedi l'anteprima clicca l'immagine dello spettacolo nelle impostazioni



Infine cambia shaddow e offset in base alle tue preferenze.

Giustifica il testo

veloce

```
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)
```

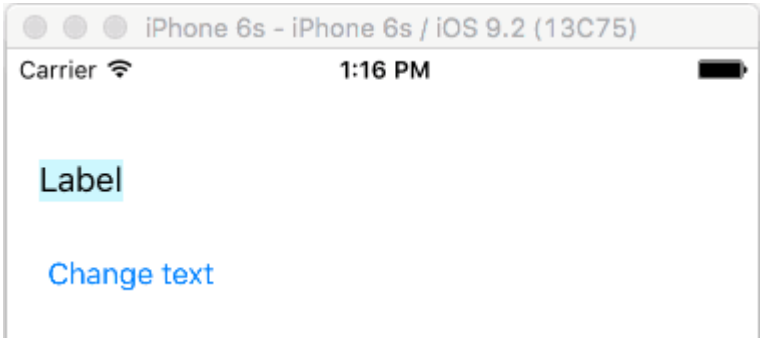
Objective-C

```
  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];
```

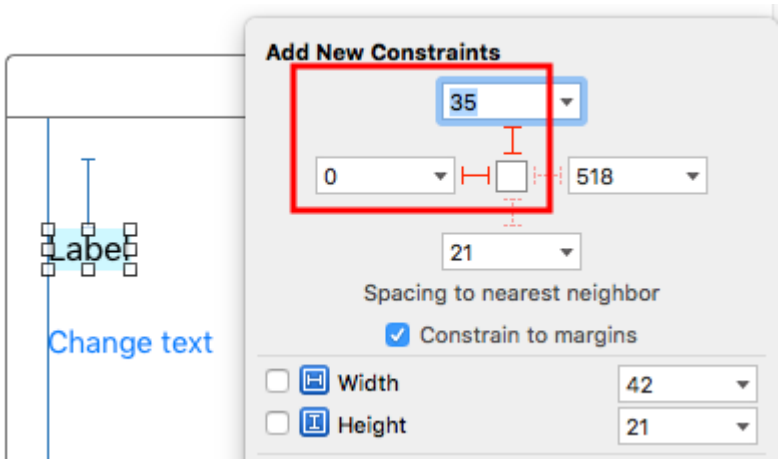
Etichetta di ridimensionamento automatico per adattarsi al testo

Questo esempio mostra come la larghezza di un'etichetta può ridimensionarsi automaticamente quando il contenuto del testo cambia.



Pin i bordi sinistro e superiore

Basta usare il layout automatico per aggiungere vincoli per bloccare i lati sinistro e superiore dell'etichetta.



Dopo questo ridimensionerà automaticamente.

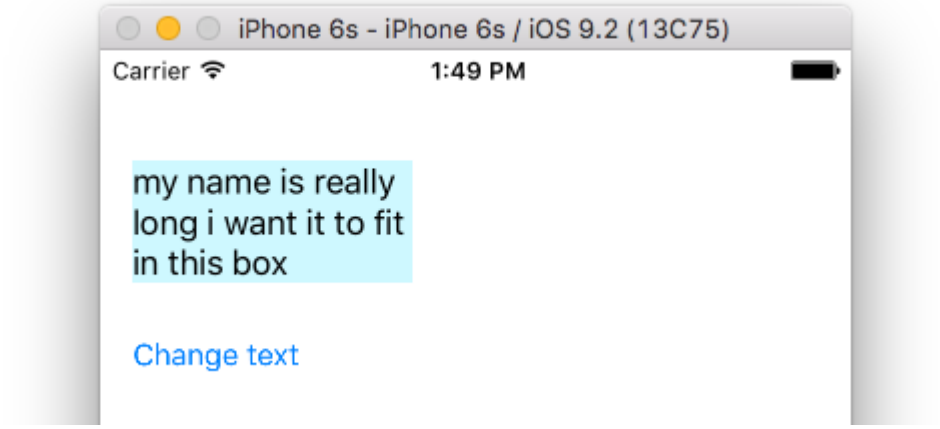
Gli appunti

- Questo esempio deriva da questa risposta di Overflow dello stack .
- Non aggiungere vincoli per la larghezza e l'altezza. Le etichette hanno una dimensione *intrinseca* basata sul loro contenuto testuale.
- Non è necessario impostare `sizeToFit` quando si utilizza il layout automatico. Il codice completo per il progetto di esempio è qui:

```
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"
    }
}
```

- Questo metodo può anche essere utilizzato per spaziare correttamente più etichette in orizzontale come in questo esempio .

- Se vuoi che la tua etichetta `myLabel.preferredMaxLayoutWidth = 150 // or whatever` imposta il numero di linee su 0 in IB e aggiungi `myLabel.preferredMaxLayoutWidth = 150 // or whatever` nel codice. (Il pulsante è anche appuntato sul fondo dell'etichetta in modo che si muova verso il basso quando l'altezza dell'etichetta aumenta.)



Ottieni le dimensioni di UILabel in base al testo e al carattere

`NSString` fornisce il metodo `boundingRectWithSize` che può essere utilizzato per prevedere il CGSize risultante di un `UILabel` base al testo e al font senza la necessità di creare un `UILabel`

Objective-C

```
[[text boundingRectWithSize:maxSize options:(NSStringDrawingTruncatesLastVisibleLine |
NSStringDrawingUsesLineFragmentOrigin) attributes:@{NSFontAttributeName: fontName}
context:nil] size];
```

veloce

```
let nsText = text as NSString?
nsText?.boundingRectWithSize(maxSize, options: [.TruncatesLastVisibleLine,
.UsesLineFragmentOrigin], attributes: [NSFontAttributeName: fontName], context: nil).size
```

veloce

Crea etichetta ed etichetta Uscita vincolo altezza. Aggiungi sotto il codice in cui dovrai assegnare il testo all'etichetta.

```
@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
```

Nota: "40" è lo spazio del lato sinistro e destro dello schermo.

Colore del testo evidenziato e evidenziato

Objective-C

```
UILabel *label = [[UILabel alloc] init];
label.highlighted = YES;
label.highlightedTextColor = [UIColor redColor];
```

veloce

```
let label = UILabel()
label.highlighted = true
label.highlightedTextColor = UIColor.redColor()
```

Swift 3

```
let label = UILabel()
label.isHighlighted = true
label.highlightedTextColor = UIColor.red
```

Leggi UILabel online: https://riptutorial.com/it/ios/topic/246/uilabel

Capitolo 177: UILocalNotification

introduzione

Le notifiche locali consentono alla tua app di notificare all'utente contenuti che non richiedono l'uso di un server.

A differenza delle notifiche remote che vengono attivate da un server, le notifiche locali sono programmate e attivate all'interno di un'app. Le notifiche in generale mirano ad aumentare l'interazione dell'utente con l'app, invitando o tentando di aprire e interagire con esso.

UILocalNotification è stato dichiarato obsoleto in iOS 10. Utilizzare invece il framework UserNotifications.

Osservazioni

Non confondere UILocalNotification con le notifiche push. UILocalNotification viene attivato dal dispositivo e, quando pianificato, viene copiato nel sistema.

link:

- Riferimento alla classe UILocalNotification
- UILocalNotification on Stack Overflow

Examples

Pianificazione di una notifica locale

Assicurati di vedere la registrazione delle notifiche locali affinché funzioni:

veloce

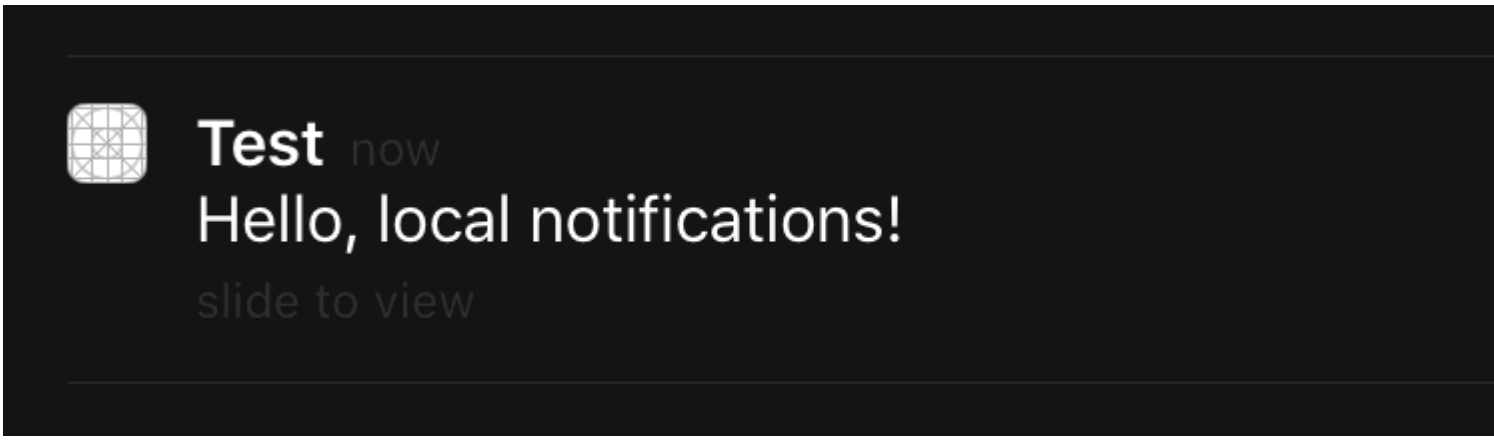
```
let notification = UILocalNotification()
notification.alertBody = "Hello, local notifications!"
notification.fireDate = NSDate().dateByAddingTimeInterval(10) // 10 seconds after now
UIApplication.sharedApplication().scheduleLocalNotification(notification)
```

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];
```

Per visualizzare la notifica in iOS Simulator, digitare ^⌘H (control-command-H) per tornare a casa e quindi digitare ⌘L (command-L) per bloccare il dispositivo. Attendere alcuni secondi e dovrebbe

comparire la notifica (questo aspetto varia in base al tipo di notifica discusso in "Registrazione per le notifiche locali"):



Scorri sulla notifica per tornare all'app (`viewDidLoad` presente che se hai chiamato questo nel `viewDidLoad` del controller della prima visualizzazione, `viewWillAppear` , `viewDidAppear` , ecc., La notifica verrà pianificata di nuovo).

Registrazione per le notifiche locali

iOS 8

Per poter presentare le notifiche locali all'utente, devi registrare la tua app con il dispositivo:

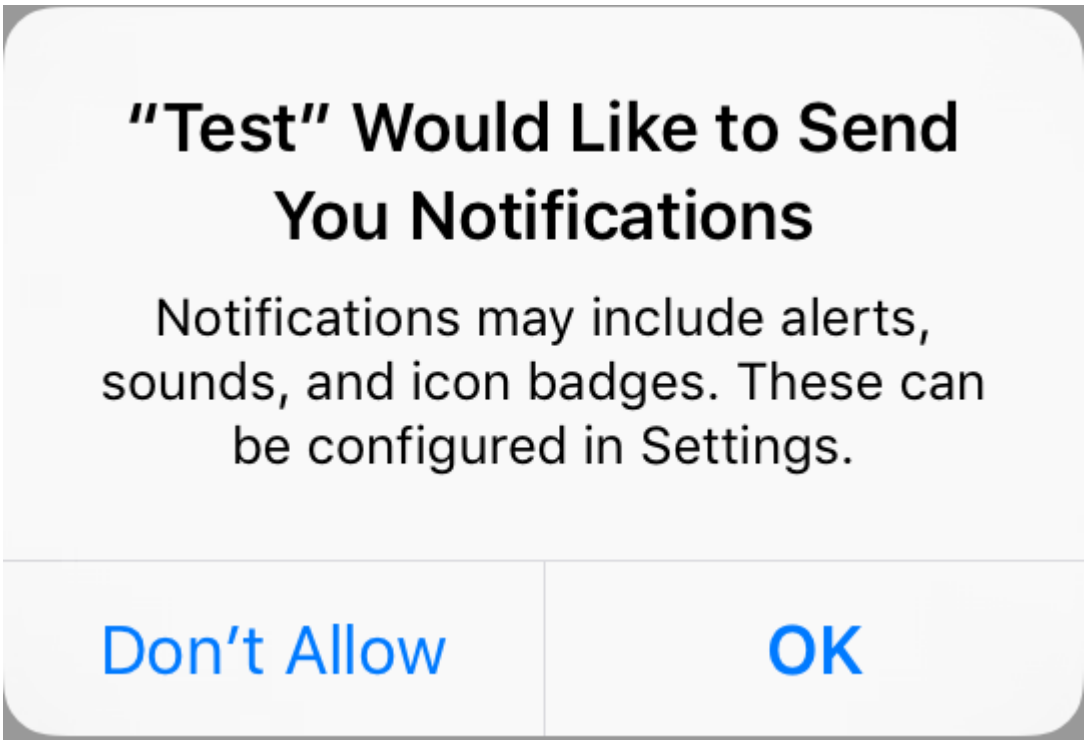
veloce

```
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];
```

Questo presenterà un avviso la prima volta che viene chiamato:



Indipendentemente da ciò che l'utente sceglie, l'avviso non verrà più visualizzato e le modifiche dovranno essere avviate dall'utente in Impostazioni.

Risposta alla notifica locale ricevuta

IMPORTANTE: questo metodo delegato viene chiamato solo in primo piano.

veloce

```
func application(application: UIApplication, didReceiveLocalNotification notification:
UILocalNotification) {

}
```

Objective-C

```
- (void)application:(UIApplication *)application
didReceiveLocalNotification:(UILocalNotification *)notification {

}
```

Questo metodo è generalmente sovrascritto in AppDelegate, che è conforme al protocollo UIApplicationDelegate.

Gestire le notifiche locali usando UUID

Spesso è necessario essere in grado di gestire le notifiche, potendo tenerle traccia e cancellarle.

Traccia una notifica

Puoi assegnare un UUID (identificatore univoco universale) a una notifica, in modo da poterlo rintracciare:

veloce

```
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];
```

Annulla una notifica

Per cancellare una notifica, prima otteniamo un elenco di tutte le notifiche e poi troviamo quello con un UUID corrispondente. Infine, lo cancelliamo.

veloce

```
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;
    }
}
```

Probabilmente vorrai memorizzare tutti questi UUID in Core Data o Realm.

Presentare immediatamente una notifica locale

Se si desidera visualizzare immediatamente la notifica locale, è necessario chiamare:

Swift 3

```
UIApplication.shared.presentLocalNotificationNow(notification)
```

Swift 2

```
UIApplication.sharedApplication().presentLocalNotificationNow(notification)
```

Objective-C

```
[[UIApplication sharedApplication] presentLocalNotificationNow:notification];
```

Un vantaggio di utilizzare questo è così non sarà necessario impostare le `fireDate` e `timeZone` proprietà del `UILocalNotification` oggetto.

Suono di notifica

Suoni personalizzati possono essere forniti per le notifiche generate dalla tua app. Quando il sistema visualizza un avviso per una notifica locale o badge un'icona di app, riproduce questo suono (purché l'utente non abbia disabilitato i suoni di notifica).

Il valore predefinito è zero, il che significa che non viene riprodotto alcun suono per la notifica.

Per fornire un suono personalizzato, aggiungi un file `.caf` , `.wav` o `.aiff` al pacchetto dell'app. I suoni che durano più di 30 secondi non sono supportati. Fornire un suono che non soddisfa tali requisiti farà riprodurre il suono predefinito (`UILocalNotificationDefaultSoundName`).

Objective-C

```
UILocalNotification *notification = [UILocalNotification new];
notification.soundName = @"nameOfSoundInBundle.wav"; // Use
UILocalNotificationDefaultSoundName for the default alert sound
```

veloce

```
let notification = UILocalNotification()
notification.soundName = "nameOfSoundInBundle.wav"
```

Registra e pianifica notifiche locali in Swift 3.0 (iOS 10)

Registrazione

in **AppDelegate**

```
import UserNotifications
```

nel metodo **didFinishLaunchingWithOptions** ,

```
UNUserNotificationCenter.current().requestAuthorization(options: [.alert,.sound,.badge]) {
(granted, error) in

// Here you can check Request is Granted or not.

}
```

Crea e pianifica notifiche.

```
    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)
```

Ovunque questa parte di codice viene attivata, se hai autorizzato l'autorizzazione di notifica, riceverai una notifica.

Per testarlo correttamente, assicurati che la tua applicazione sia in modalità Background.

Novità in UILocalNotification con iOS10

È possibile utilizzare `UILocalNotification` , anche le vecchie API funzionano correttamente con iOS10, ma è preferibile utilizzare le API nel framework delle notifiche utente. Ci sono anche alcune nuove funzionalità, che puoi utilizzare solo con il framework Notifiche utente iOS10.

Questo succede anche alla notifica remota, per ulteriori informazioni: qui .

Nuove caratteristiche:

1. Ora puoi presentare avviso, audio o aumentare il badge mentre l'app è in primo piano anche con iOS 10
2. Ora puoi gestire tutti gli eventi in un unico posto quando l'utente tocca (o fa scorrere) il pulsante di azione, anche se l'app è già stata uccisa.
3. Supporta il tocco 3D anziché il gesto scorrevole.
4. Ora puoi rimuovere la notifica locale specifica solo con un codice di riga.
5. Supporta le notifiche avanzate con l'interfaccia utente personalizzata.

È davvero facile per noi convertire `UILocalNotification` API di `UILocalNotification` API di framework di notifiche utente iOS10, sono davvero simili.

Scrivo una demo qui per mostrare come utilizzare le API nuove e vecchie contemporaneamente:

iOS10AdaptationTips .

Per esempio,

Con l'implementazione di Swift:

1. import UserNotifications

```
///    Notification become independent from UIKit
import UserNotifications
```

2. richiesta di autorizzazione per localNotification

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

3. pianificare localNotification

4. aggiornare il numero del badge dell'icona dell'applicazione

```
@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"])
}
```

Implementazione Objective-C:

1. import UserNotifications

```
// Notifications are independent from UIKit
#import <UserNotifications/UserNotifications.h>
```

2. richiesta di autorizzazione per 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. pianificare localNotification

4. aggiornare il numero di badge dell'icona dell'applicazione

```
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!");
    }
}];
```

Vai a qui per ulteriori informazioni: **iOS10AdaptationTips** .

#updated

> Terminare l'app a causa dell'eccezione non rilevata 'NSInternalInconsistencyException', motivo: 'l'intervallo di tempo deve essere di almeno 60 se si ripete'

```
let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 60, repeats: true)
```

Leggi UILocalNotification online: https://riptutorial.com/it/ios/topic/635/uilocalnotification

Capitolo 178: UINavigationController

Osservazioni

Dalla documentazione :

> La classe UINavigationController implementa un controller di visualizzazione specializzato che gestisce la navigazione del contenuto gerarchico. Questa interfaccia di navigazione consente di presentare i dati in modo efficiente e rende più semplice all'utente navigare all'interno di tali contenuti. Generalmente usi questa classe così com'è, ma puoi anche creare sottoclassi per personalizzare il comportamento della classe.

Examples

Popping in un controller di navigazione

Per il controller della vista precedente

Per tornare alla pagina precedente puoi farlo:

veloce

```
navigationController?.popViewControllerAnimated(true)
```

Objective-C

```
[self.navigationController popViewControllerAnimated:YES];
```

Per il controllo della vista principale

Per eseguire il pop alla radice dello stack di navigazione, puoi fare ciò:

veloce

```
navigationController?.popToRootViewControllerAnimated(true)
```

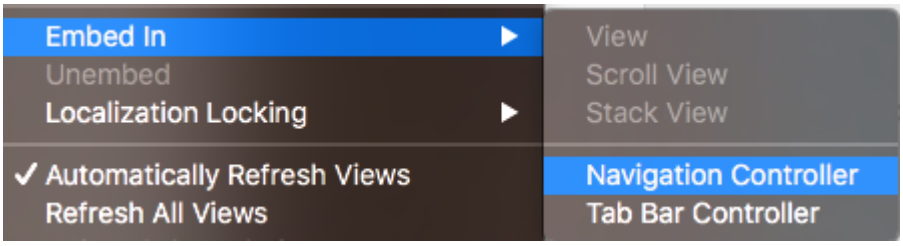
Obiettivo C

```
[self.navigationController popToRootViewControllerAnimated:YES];
```

Creare un NavigationController

Nello storyboard seleziona il ViewController che desideri incorporare in un controller di navigazione.

Quindi accedere a Editor> Incorpora in> Controller di navigazione



E ciò creerà il tuo controller di navigazione

Incorporare un controller di visualizzazione in un controller di navigazione a livello di codice

veloce

```
//Swift
let viewController = UIViewController()
let navigationController = UINavigationController(rootViewController: viewController)

//Objective-C
UIViewController *viewController = [[UIViewController alloc] init];
UINavigationController *navigationController = [[UINavigationController alloc]
initWithRootViewController:viewController];
```

Spingere un controller di visualizzazione sullo stack di navigazione

```
//Swift
let fooViewController = UIViewController()
navigationController?.pushViewController(fooViewController, animated: true)

//Objective-C
UIViewController *fooViewController = [[UIViewController alloc] init];
[navigationController pushViewController:fooViewController animated:YES];
```

Scopo

`UINavigationController` viene utilizzato per formare una gerarchia di strutture di visualizzazione ad albero, che è nota come `navigation stack` .

Dalla prospettiva degli sviluppatori:

Puoi connettere controller indipendentemente creato e ottenere tutti i vantaggi di un gestore di gerarchia gratuito e di un presentatore dell'interfaccia utente comune gratis. `UINavigationController` anima la transizione verso i nuovi controller e fornisce automaticamente le funzionalità di back-up. `UINavigationController` consente inoltre l'accesso a tutti gli altri controller nello `navigation stack` che possono aiutare ad accedere ad alcune funzionalità o dati.

Dal punto di vista dell'utente:

`UINavigationController` aiuta a ricordare dove si trova l'utente al momento (titolo della barra di navigazione) e come può tornare indietro (pulsante Indietro incorporato) a una delle schermate precedenti.

Leggi UINavigationController online: https://riptutorial.com/it/ios/topic/1079/uinavigationcontroller

Capitolo 179: UIPageViewController

introduzione

UIPageViewController offre agli utenti la possibilità di passare facilmente da una vista all'altra usando un gesto di scorrimento. Per creare un UIPageViewController, è necessario implementare i metodi UIPageViewControllerDataSource. Questi includono metodi per restituire sia UIPageViewController prima e dopo l'attuale UIPageViewController insieme ai metodi presentationCount e presentationIndex.

Sintassi

1. UIPageViewControllerTransitionStyle
2. UIPageViewControllerNavigationOrientation
3. UIPageViewControllerSpineLocation
4. UIPageViewControllerNavigationDirection

Osservazioni

Riferimento per gli sviluppatori Apple qui

Examples

Creare un UIPageViewController di paginazione orizzontale programmaticamente

1. Init array of view controller che sarà gestito da UIPageViewController. Aggiungere una classe controller di visualizzazione di base che abbia `identifier` proprietà che verrà utilizzato per identificare i controller di visualizzazione quando si lavora con i metodi dell'origine dati UIPageViewController. Lascia che i controller di visualizzazione ereditino da quella classe base.

```
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. Crea un'istanza UIPageViewController.

```
UIPageViewController *pageViewController = [[UIPageViewController alloc]
initWithTransitionStyle:UIPageViewControllerTransitionStyleScroll

navigationOrientation:UIPageViewControllerNavigationOrientationHorizontal
```

```
options:nil];
```

3. L'origine dati è la classe corrente che deve implementare il protocollo `UIPageViewControllerDataSource` .

```
pageViewController.dataSource = self;
```

4. `setViewControllers` aggiungerà solo il controller della prima vista, successivamente verrà aggiunto allo stack utilizzando i metodi di origine dei dati

```
if (viewControllers.count) {
    [pageViewController setViewControllers:@[[viewControllers objectAtIndex:0]]
                                 direction:UIPageViewControllerNavigationDirectionForward
                                  animated:NO
                                completion:nil];
}
```

5. Aggiungere UIPageViewController come un controller di vista del bambino in modo che riceverà dal suo genitore View Controller `appearance` e `rotation` eventi.

```
 [self addChildViewController:pageViewController];
 pageViewController.view.frame = self.view.frame;
 [self.view addSubview:pageViewController.view];
 [pageViewController didMoveToParentViewController:self];
```

6. Implementazione dei metodi 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;
}
```

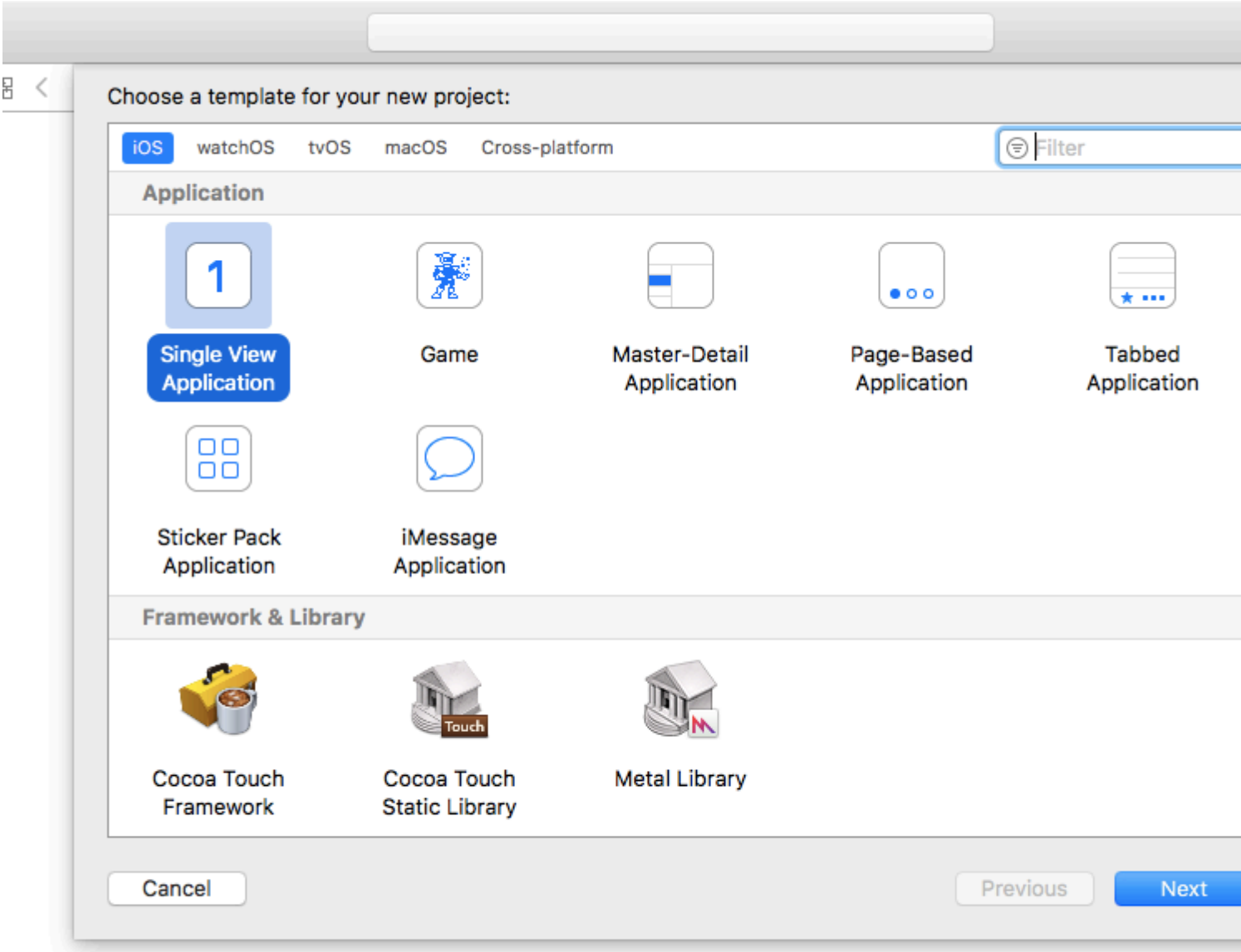
7. Metodo di utilità che restituisce un controller di visualizzazione utilizzando un indice, se

l'indice è fuori dai limiti restituisce nulla.

```
- (UIViewController *)childViewControllerAtIndex:(NSInteger)index
{
    if (index <= ([viewControllers count] - 1)) {
        return [viewControllers objectAtIndex:index];
    } else {
        return nil;
    }
}
```

Un modo semplice per creare controller di visualizzazione di pagina orizzontali (pagine infinite)

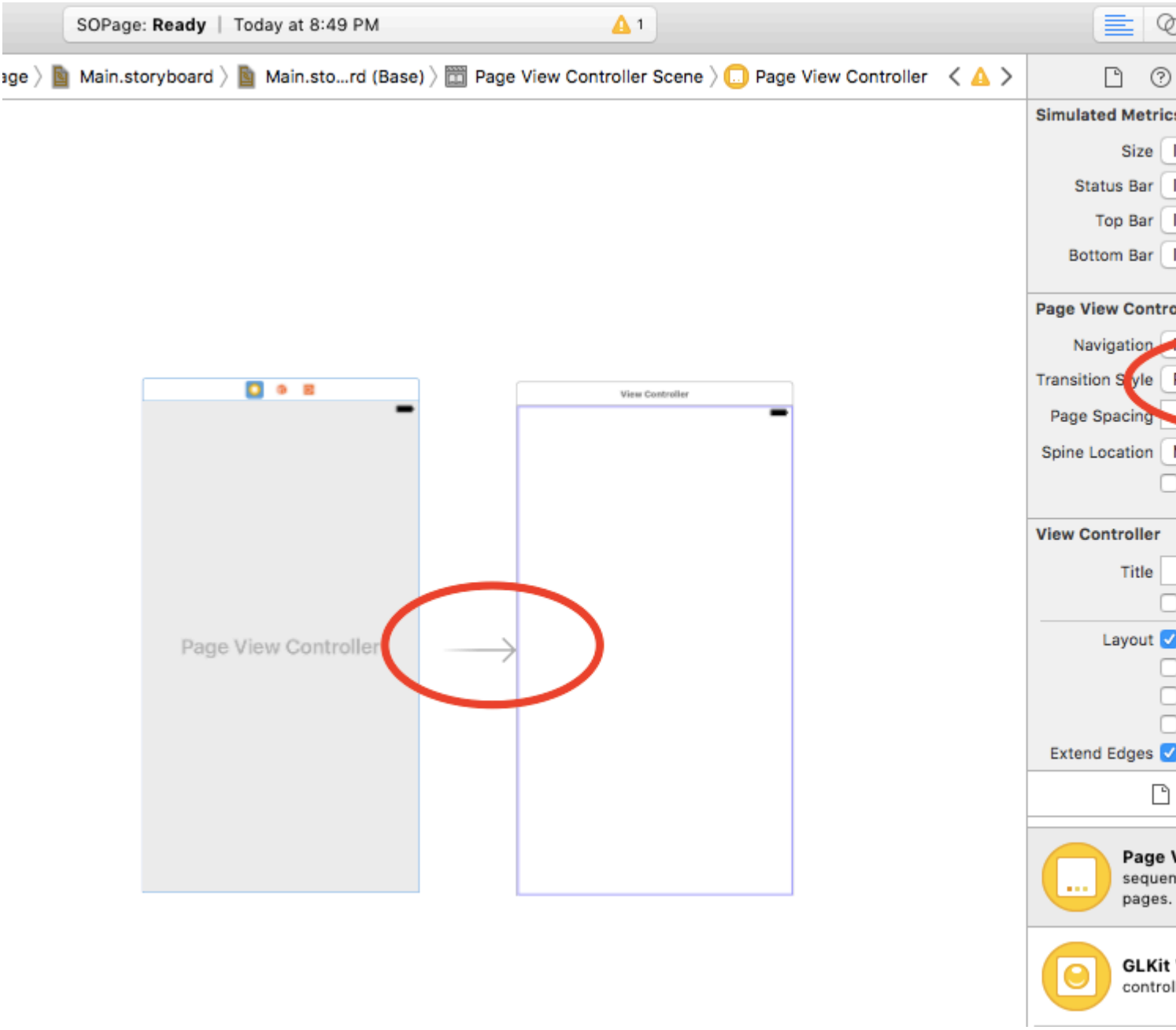
1. Creiamo un nuovo progetto, sto scegliendo l'applicazione Single View per una migliore dimostrazione



2. Trascina un controller di visualizzazione pagina nello storyboard, ci sono 2 cose che dovresti

cambiare dopo:

1. Imposta il controller della visualizzazione della pagina come controller della vista iniziale
2. Cambia lo stile di transizione per scorrere



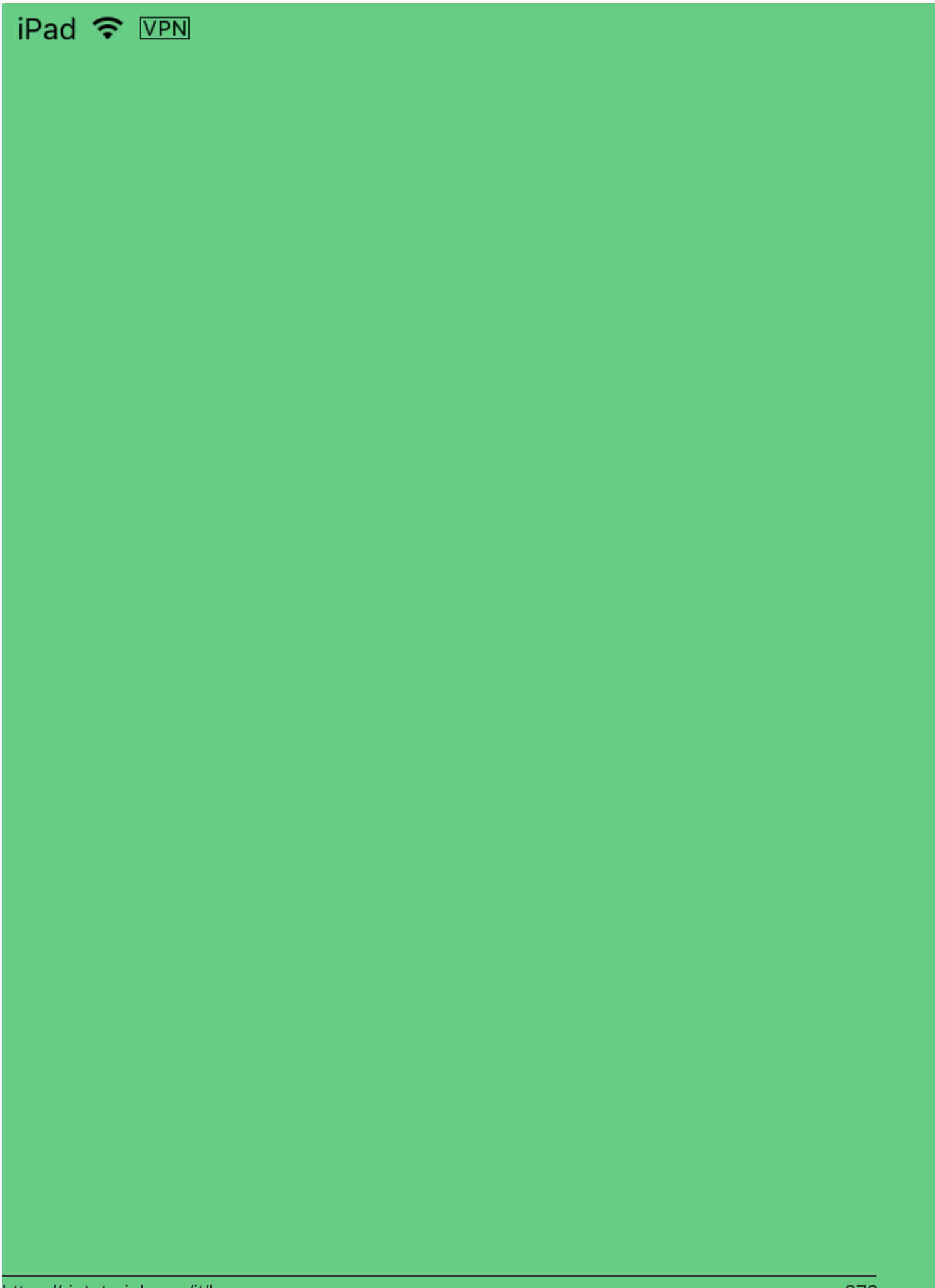
3. E devi creare una classe UIPageViewController, quindi impostarla come classe personalizzata del controller della visualizzazione della pagina nello storyboard

4. Incolla questo codice nella tua classe UIPageViewController, dovresti ottenere una colorata app a pagine infinite :)

```
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
    }
}
```

Ecco come si presenta il progetto finale, ottieni un controller di visualizzazione con colori diversi ad ogni scorrimento:


iPad
VPN

Capitolo 180: UIPheonix: framework UI facile, flessibile, dinamico e altamente scalabile

introduzione

Ispirato allo sviluppo del gioco UIPheonix è un framework UI + concetto super facile, flessibile, dinamico e altamente scalabile per la creazione di app riutilizzabili basate su componenti / controllo per macOS, iOS e tvOS. La stessa API si applica per lo sviluppo multipiattaforma! Pensa a come usare i blocchi Lego, puoi usare quelli simili e muoverli facilmente come torta.

https://github.com/MKGitHub/UIPheonix

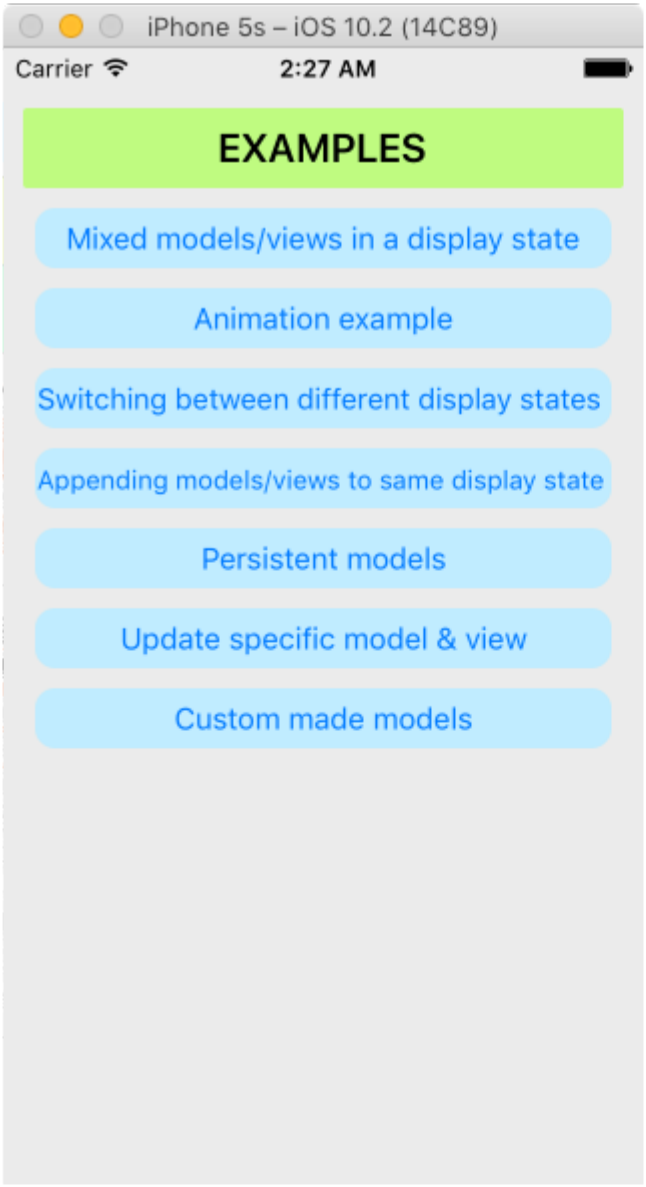
Osservazioni

- Dimentica i layout statici, i problemi di vincoli e le esplosioni di avviso nella console.
- Dimentica tutto il codice della colla, tutto il codice boilerplate e tutto il mucchio di codice inutile inutilmente sovradimensionato e troppo comune nelle tue app.
- Costruisci e apporta modifiche alla tua interfaccia utente rapidamente in un attimo.
- Rendi la tua interfaccia utente riutilizzabile.
- Concentrati sulla creazione della tua app, senza combattere i problemi di layout.
- Configurazione minima, impatto minimo sulla tua app, leggero, senza dipendenze, nessun dolore ma tanto guadagno!
- Costruisce sulla sommità delle viste delle collezioni e delle tabelle, così puoi facilmente mischiarle.
- Non sostituisce le tecnologie Apple con implementazioni personalizzate, quindi sarai sempre sicuro e aggiornato e potrai facilmente tornare in qualsiasi momento.
- App demo fornite per macOS, iOS e tvOS (Kung Fu!)

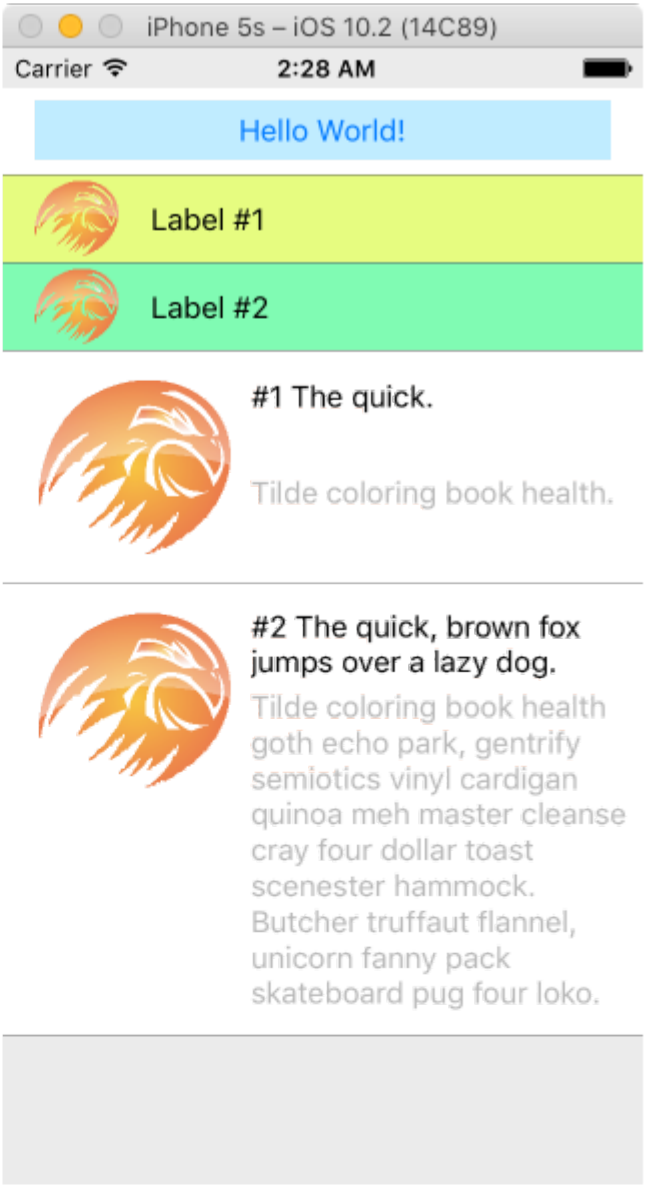
Examples

Esempi di componenti dell'interfaccia utente

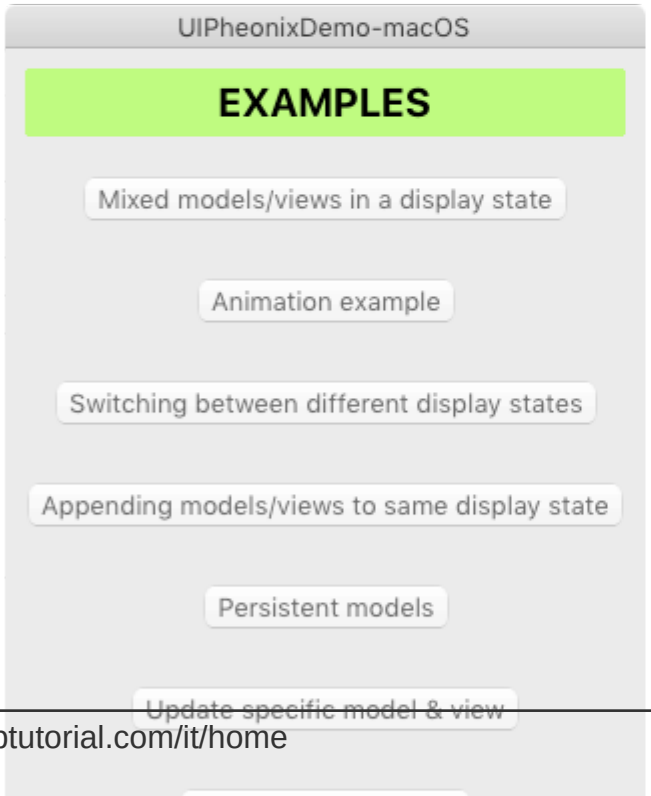
iOS - Collection View



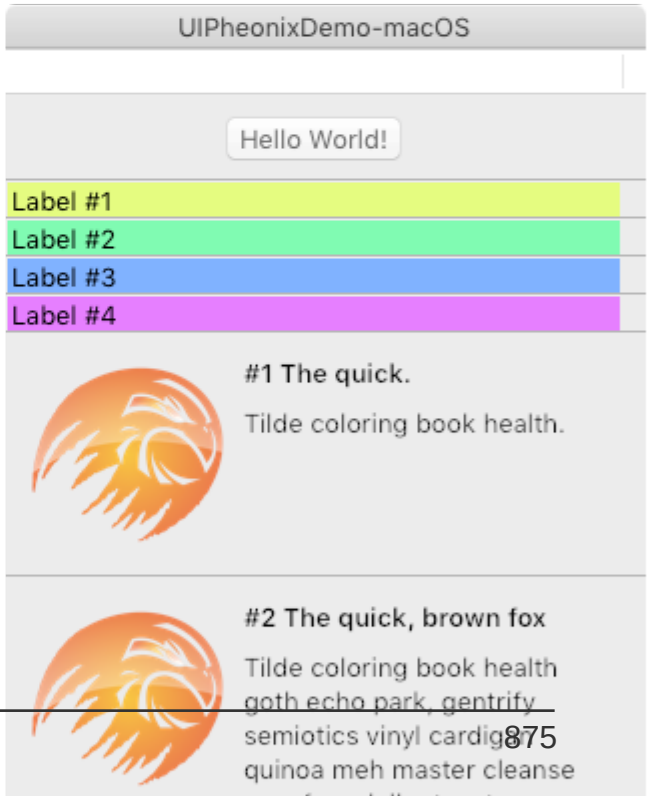
iOS - Table View



macOS - Collection View



macOS - Table View



https://riptutorial.com/it/ios/topic/9120/uipheonix--framework-ui-facile--flessibile--dinamico-e-altamente-scalabile

Capitolo 181: UIPickerView

Examples

Esempio di base

veloce

```
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];
}
```

Cambiare il selettore Visualizza sfondo Colore e colore del testo

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"];
```

veloce

```
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")
```

Leggi UIPickerView online: https://riptutorial.com/it/ios/topic/4242/uipickerview

Capitolo 182: UIRefreshControl TableView

introduzione

Un oggetto UIRefreshControl fornisce un controllo standard che può essere utilizzato per avviare l'aggiornamento dei contenuti di una vista tabella. Si collega un controllo di aggiornamento a una tabella tramite un oggetto controller della vista tabella associato. Il controller di visualizzazione tabella gestisce il lavoro di aggiunta del controllo all'aspetto visivo della tabella e gestisce la visualizzazione di tale controllo in risposta a gesti appropriati dell'utente.

Examples

Esempio-C Esempio

Prima dichiarare una proprietà come questa nel ViewController

```
@property (nonatomic) UIRefreshControl *refreshControl;
```

Più tardi in `viewDidLoad()` configura refreshControl come indicato di seguito:

```
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;
```

Ora la funzione `refreshTable` è definita come:

```
- (void)refreshTable {
    //TODO: refresh your data
    [self.refreshControl endRefreshing];
    [self.refreshControl beginRefreshing];
    [self.tableView reloadData];
    [self.refreshControl endRefreshing];
}
```

Configura refreshControl su 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];
    });
});
```

}

Leggi UIRefreshControl TableView online: https://riptutorial.com/it/ios/topic/8278/uirefreshcontrol-tableview

Capitolo 183: UIScrollView

Examples

Crea un UIScrollView

Creare un'istanza di `UIScrollView` con un `CGRect` come frame.

veloce

```
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)];
```

Scorri Visualizza dimensioni del contenuto

La proprietà `contentSize` deve essere impostata sulla dimensione del contenuto scorrevole. Questo specifica la dimensione dell'area scorrevole. Lo scorrimento è visibile quando l'area scorrevole, ad esempio `contentSize` è più grande della dimensione del frame di `UIScrollView` .

Con Autolayout:

Quando il contenuto della vista di scorrimento viene impostato utilizzando l'autolayout, deve essere dimensionato in modo esplicito sia verticalmente che orizzontalmente e avere tutti e 4 i bordi aggiunti alla vista di scorrimento contenente. In questo modo, `contentSize` viene calcolato automaticamente in base ai contenuti della vista di scorrimento e viene inoltre aggiornato quando viene modificato il layout del contenuto.

manualmente:

veloce

```
scrollview.contentSize = CGSize(width: 640, height: 800)
```

Objective-C

```
scrollview.contentSize = CGSizeMake(640, 800);
```

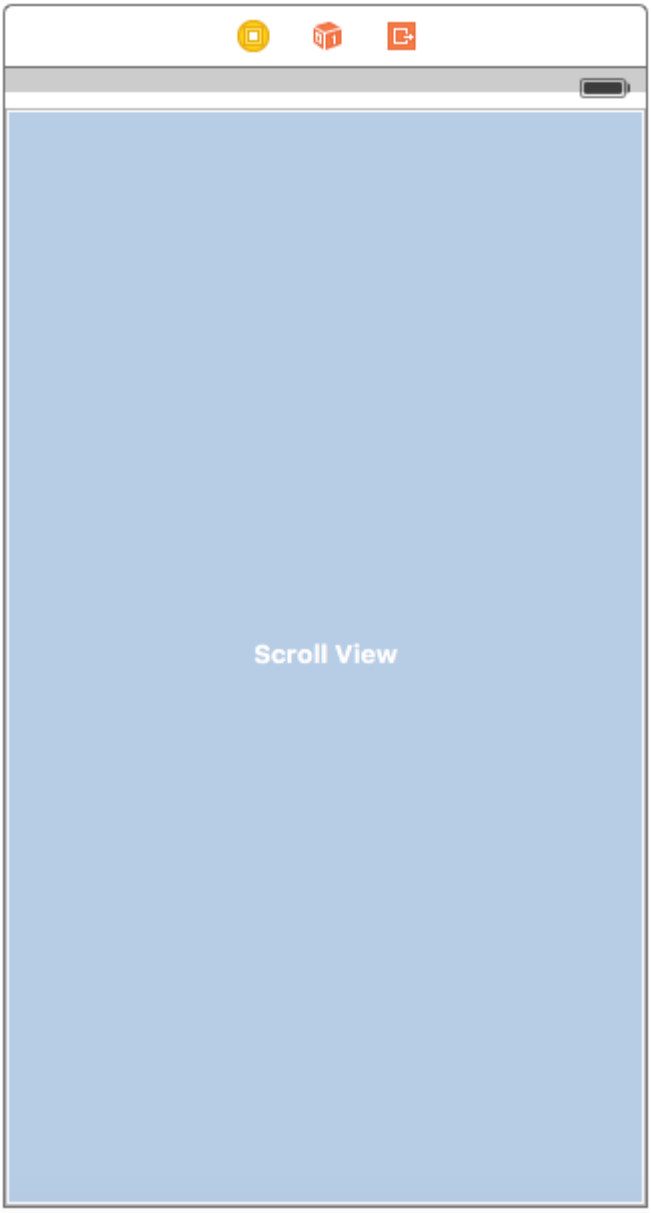
ScrollView con AutoLayout

Semplici passi per usare scrollview con autolayout.

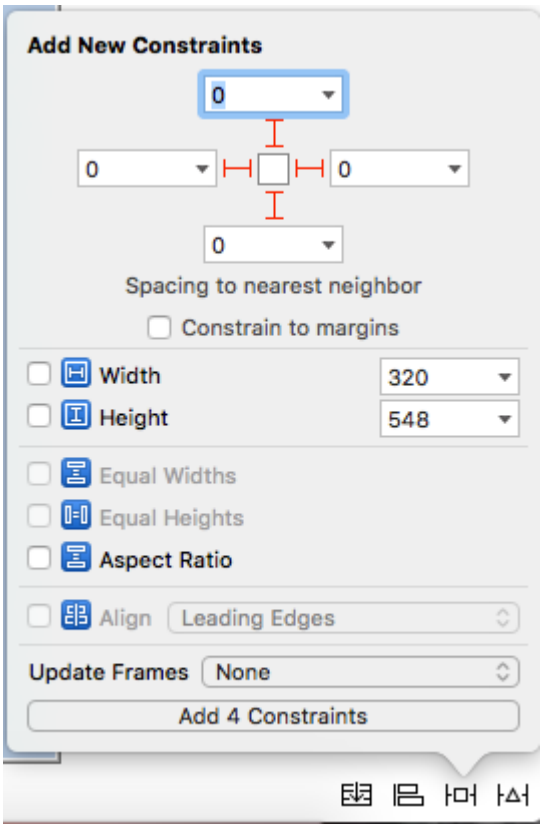
- Crea un nuovo progetto con un'applicazione a vista singola
- Seleziona il viewcontroller predefinito e cambia le sue dimensioni dello schermo in iPhone-

4inch dall'ispettore Attributi.

- Aggiungi una vista a scorrimento alla vista del tuo viewcontroller come segue e imposta il colore di sfondo su blu



- Aggiungi vincoli su di esso come mostrato nell'immagine qui sotto



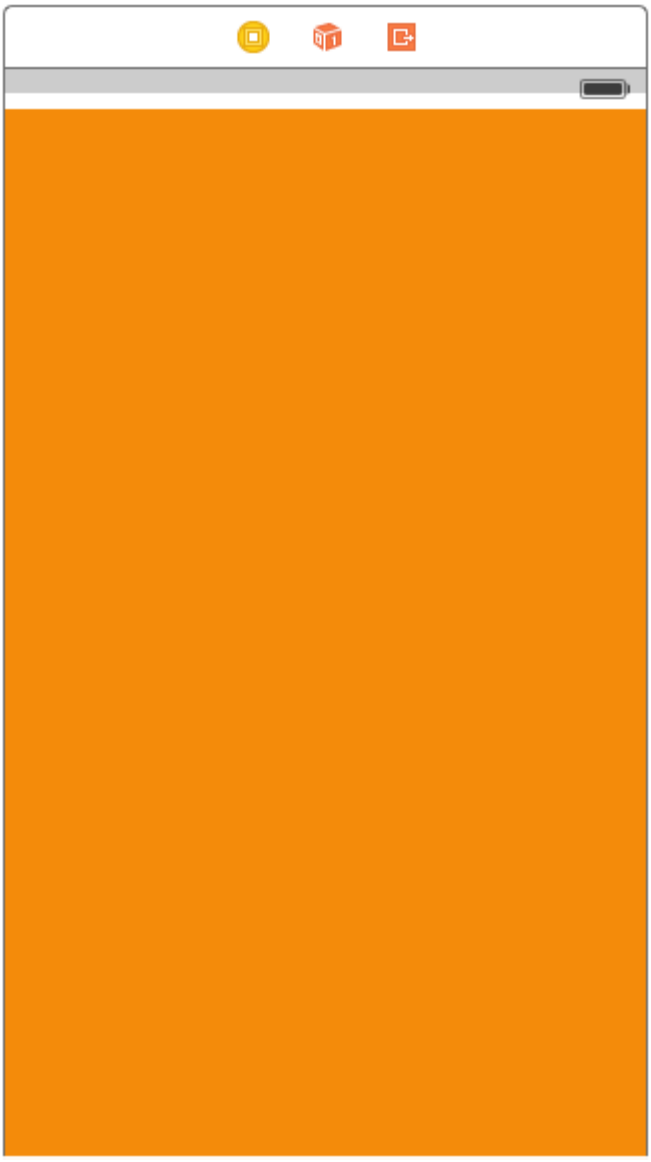
Ciò che questo farà è semplicemente attaccare ogni angolo di scrollview alla vista di viewcontroller

Scenario 1:

Ora diciamo che il nostro contenuto è enorme e vogliamo che scorra sia orizzontalmente che verticalmente.

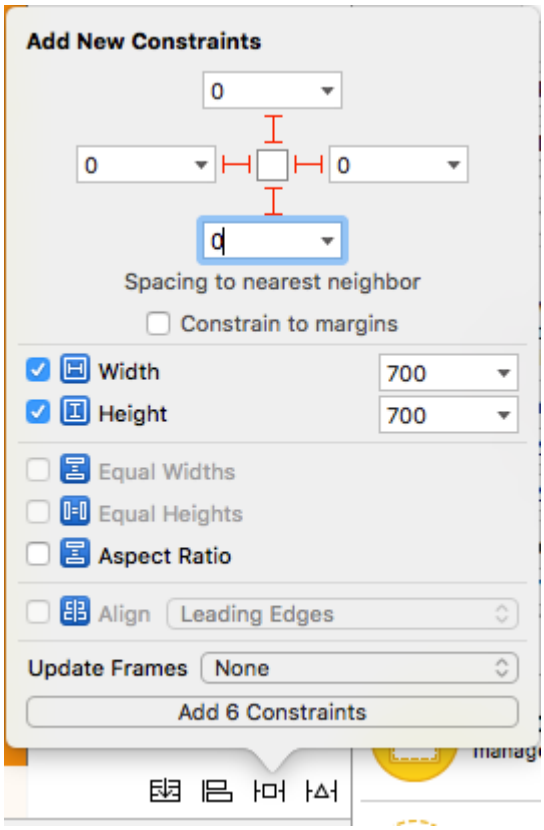
Per questo,

- Aggiungi un UIView alla scrollview del frame (0,0,700,700). Ti dà il colore di sfondo arancione per identificarlo in modo diverso.



Poi arriva la parte importante, ne abbiamo bisogno per scorrere orizzontalmente e verticalmente.

- Seleziona la vista arancione e aggiungi i seguenti vincoli



Lascia che ti spieghi cosa abbiamo fatto nel passaggio precedente.

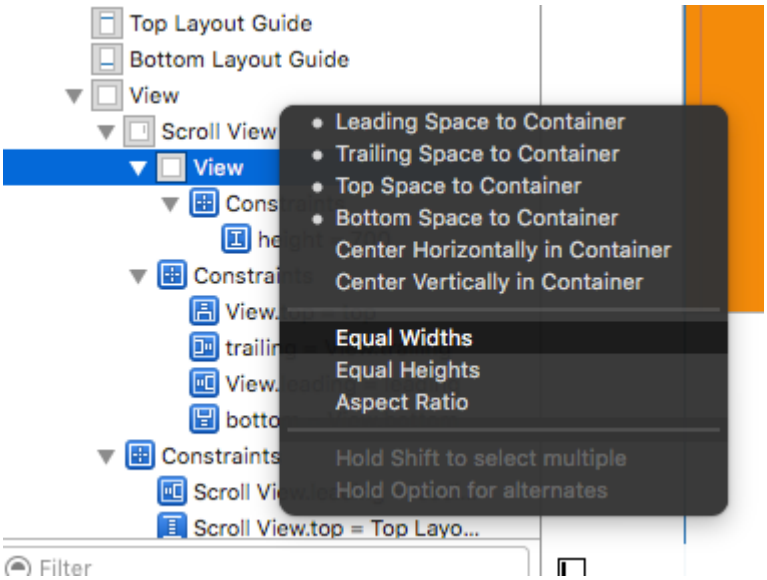
- Abbiamo fissato l'altezza e la larghezza a 700.
- Impostiamo lo spazio finale a scrollview = 0 che indica allo scrollview che il contenuto è scorrevole orizzontalmente.
- Impostiamo lo spazio in basso per scrollview = 0, che indica allo scrollview che il contenuto è scorrevole verticalmente.

Ora esegui il progetto e controlla.

Scenario 2: ***consideriamo uno scenario in cui sappiamo che la larghezza del contenuto sarà uguale alla larghezza della larghezza del rotolo, ma l'altezza è maggiore di scrollview.***

Seguire i passaggi per scorrere il contenuto verticalmente.

- Elimina il vincolo di larghezza nel caso precedente.
- Cambia la larghezza della vista arancione in modo che corrisponda alla larghezza della vista.
- Tenere premuto Ctrl dalla vista arancione per scorrere la vista e aggiungere **un** vincolo di **larghezza uguale** .



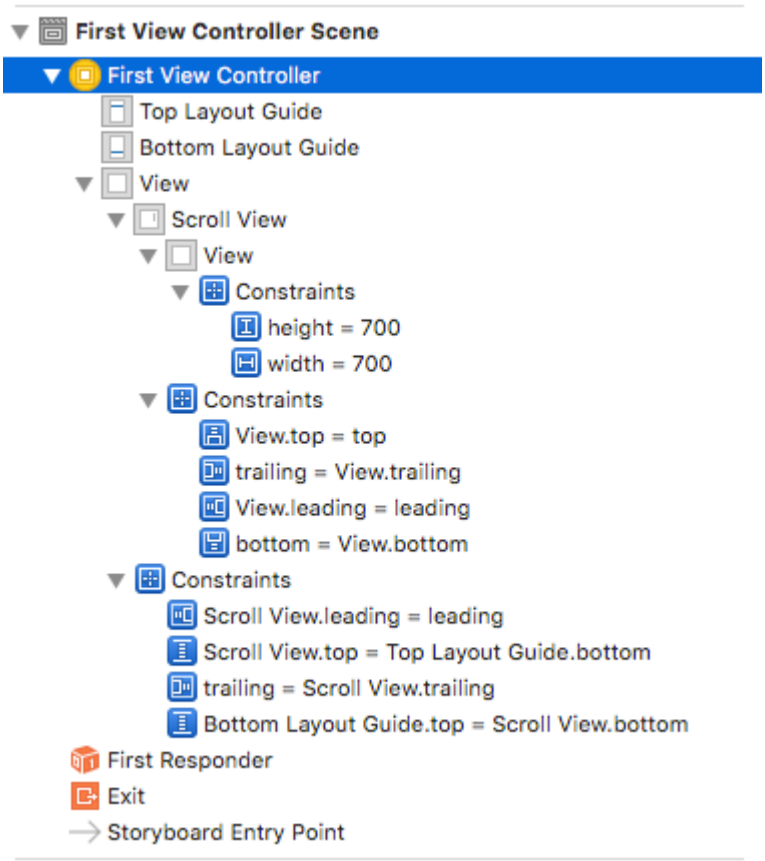
- E fatto !!! Basta eseguire e controllare se scorre verticalmente

Scenario 3:

Ora vogliamo scorrere solo orizzontalmente e non verticalmente.

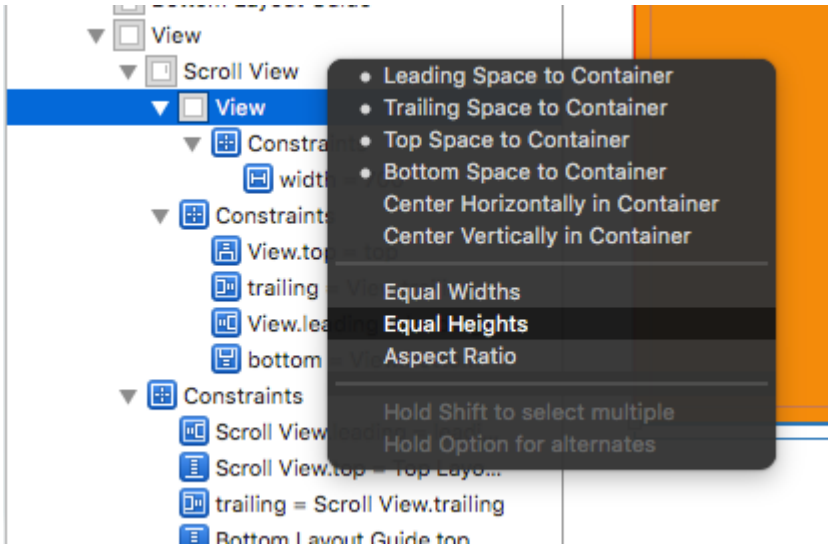
Seguire i passaggi per scorrere orizzontalmente il contenuto.

- Annullare tutte le modifiche per ottenere i vincoli come di seguito (cioè **ripristinare i vincoli originali che hanno raggiunto lo scorrimento verticale e orizzontale**)



- Verifica la cornice della vista arancione, che dovrebbe essere (0,0,700,700)

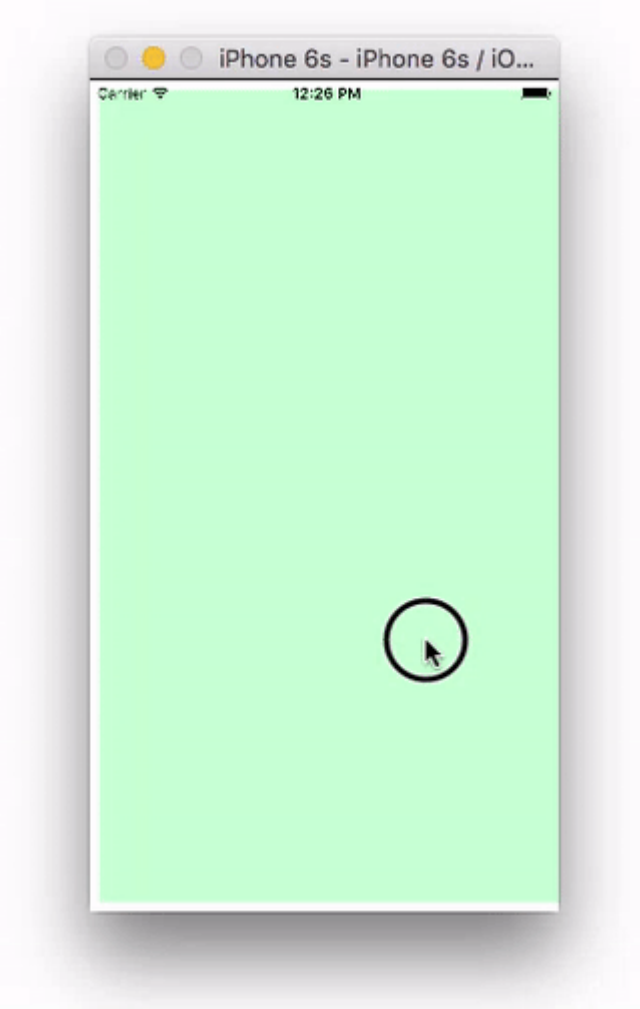
- Elimina il vincolo di altezza della vista arancione.
- Cambia l'altezza della vista arancione in modo che corrisponda all'altezza della vista.
- Ctrl-trascina dalla vista arancione per scorrere la vista e aggiungere **un** vincolo di **uguale altezza** .



- E fatto !!! Basta eseguire e controllare se scorre verticalmente

Scorrimento del contenuto con Auto Layout abilitato

Questo progetto è un esempio autonomo svolto completamente in Interface Builder. Dovresti essere in grado di analizzarlo in 10 minuti o meno. Quindi puoi applicare i concetti che hai imparato al tuo progetto.



Qui uso solo `UIView` s ma possono rappresentare qualsiasi visualizzazione (ad es. Pulsante, etichetta, ecc.). Ho anche scelto lo scorrimento orizzontale perché gli screenshot dello storyboard sono più compatti per questo formato. I principi sono gli stessi per lo scrolling verticale, però.

Concetti chiave

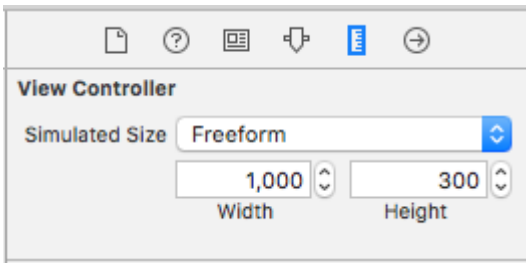
- `UIScrollView` dovrebbe usare solo una sottoview. Questo è un 'UIView' che funge da visualizzazione del contenuto per contenere tutto ciò che desideri scorrere.
- Crea la vista del contenuto e il *genitore* della vista di scorrimento ha un'altezza uguale per lo scorrimento orizzontale. (Larghezze uguali per lo scorrimento verticale)
- Assicurati che tutto il contenuto scorrevole abbia una larghezza impostata ed è bloccato su tutti i lati.

Inizia un nuovo progetto

Può essere solo un'applicazione a vista singola.

storyboard

In questo esempio, creeremo una vista di scorrimento orizzontale. Seleziona il View Controller e quindi scegli Freeform in Impostazioni dimensioni. Fai la larghezza `1,000` e l'altezza `300` . Questo ci dà la stanza sullo storyboard per aggiungere contenuti che scorreranno.

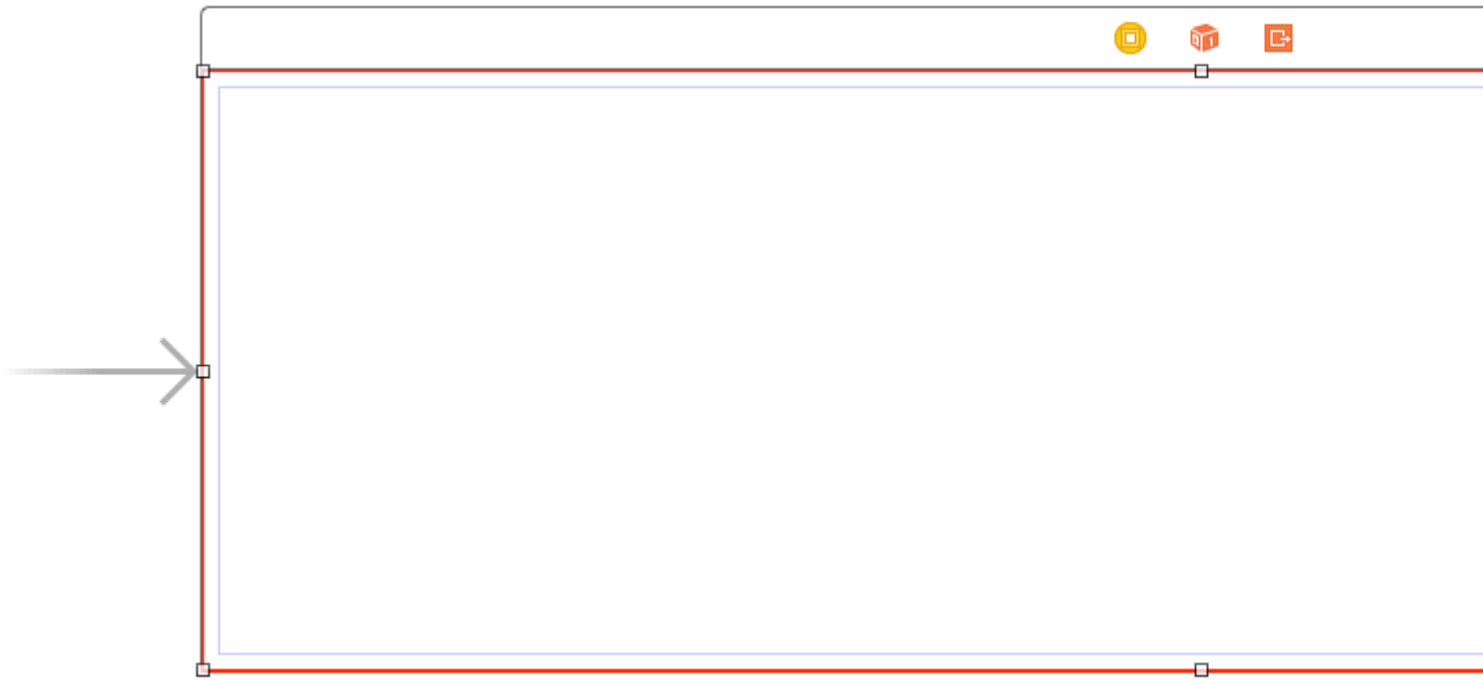


Aggiungi una vista di scorrimento

Aggiungi un `UIScrollView` e `UIScrollView` tutti e quattro i lati alla vista radice del controller della vista.

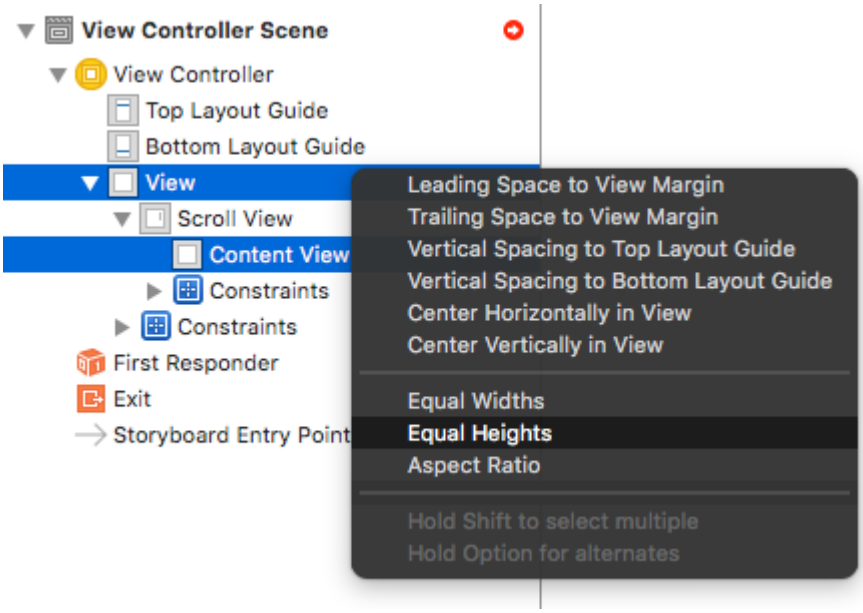
Aggiungi una vista del contenuto

Aggiungi un `UIView` come sottoview alla vista di scorrimento. *Questa è la chiave.* Non provare ad aggiungere molte visualizzazioni secondarie alla vista di scorrimento. Basta aggiungere un singolo `UIView` . Questa sarà la vista del contenuto per le altre viste che desideri scorrere. Pin la vista del contenuto per la visualizzazione di scorrimento su tutti e quattro i lati.



Altezze uguali

Ora nella struttura del documento, `Comando` , fare clic sulla vista del contenuto e sulla vista *principale* della vista di scorrimento per selezionarli entrambi. Quindi imposta le altezze per essere uguale (controllo </ kbd trascinare dalla vista del contenuto alla vista di scorrimento>). *Questa è anche la chiave.* Poiché stiamo scorrendo orizzontalmente, la vista del contenuto della vista di scorrimento non saprà quanto dovrebbe essere alta a meno che non la impostiamo in questo modo.

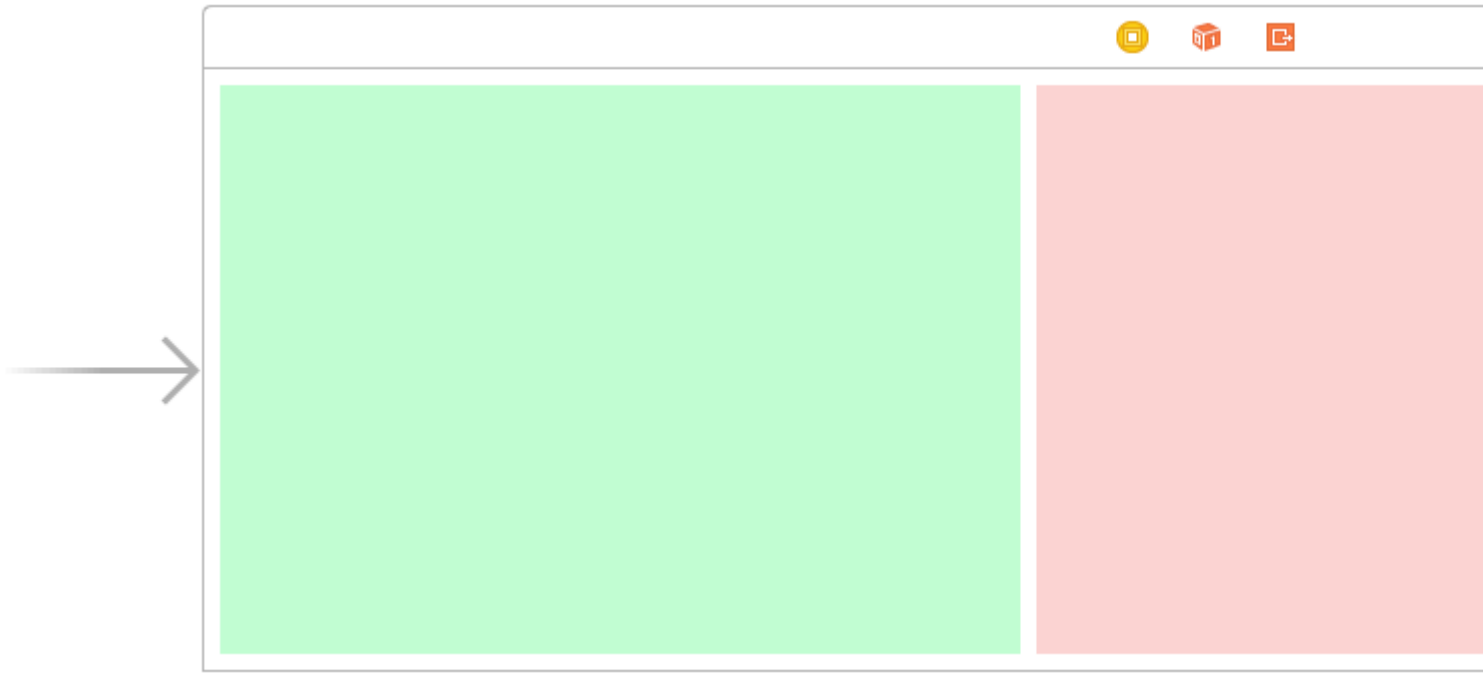


Nota:

- Se stessimo facendo scorrere il contenuto verticalmente, imposteremo la larghezza della vista del contenuto uguale alla larghezza del genitore della vista di scorrimento.

Aggiungi contenuto

Aggiungi tre `UIView` e dai loro tutti i vincoli. Ho usato margini di 8 punti per tutto.



vincoli:

- Vista verde: pin i bordi superiore, sinistro e inferiore. Fai la larghezza 400.
- Vista rossa: pin i bordi superiore, sinistro e inferiore. Fai la larghezza 300.
- Vista viola: spilla tutti e quattro i bordi. Rendi la larghezza qualunque sia lo spazio rimanente (268 in questo caso).

L'impostazione dei vincoli di larghezza è anche la chiave in modo che la vista di scorrimento sappia quanto sarà ampia la sua vista del contenuto.

Finito

È tutto. Puoi eseguire il tuo progetto ora. Dovrebbe comportarsi come l'immagine a scorrimento nella parte superiore di questa risposta.

Ulteriore studio

- iOS: Come eseguire il layout automatico su un ScrollView
- Come configurare un UIScrollView con Auto Layout in Interface Builder
- Esercitazione video di YouTube: UIScrollView - Come mantenere le tue visualizzazioni sullo schermo

Abilita / disabilita lo scorrimento

Proprietà `scrollEnabled` memorizza un valore `Boolean` che determina se lo scorrimento è abilitato o meno.
Se il valore di questa proprietà è true / YES, lo scorrimento è abilitato, altrimenti no. Il valore predefinito è `true`

veloce

```
scrollview.isScrollEnabled = true
```

Objective-C

```
scrollview.scrollEnabled = YES;
```

Zoom avanti / indietro UIImageView

Crea un'istanza UIScrollView

```
let scrollview = UIScrollView.init(frame: self.view.bounds)
```

E quindi imposta queste proprietà:

```
scrollView.minimumZoomScale = 0.1
scrollView.maximumZoomScale = 4.0
scrollView.zoomScale = 1.0
scrollview.delegate = self as? UIScrollViewDelegate
```

Per ingrandire e ridurre l'immagine dobbiamo specificare la quantità che l'utente può ingrandire e rimpicciolire. Lo facciamo impostando i valori delle proprietà `minimumZoomScale` e `maximumZoomScale` della vista di `minimumZoomScale` . Entrambi sono impostati su 1.0 per impostazione predefinita.

E `zoomScale` a 1.0 che specifica il fattore di zoom per lo zoom minimo e massimo.

Per supportare lo zoom, dobbiamo impostare un delegato per la tua vista di scorrimento. L'oggetto delegato deve essere conforme al protocollo `UIScrollViewDelegate` . Quella classe delegata deve implementare il metodo `viewForZoomingInScrollView()` e restituire la vista per lo zoom.

Modifica il tuo ViewController come mostrato

```
class ViewController: UIViewController, UIScrollViewDelegate
```

Quindi aggiungere la seguente funzione di delega alla classe.

```
func viewForZoomingInScrollView(scrollView: UIScrollView) -> UIView? {
    return imageView
}
```

Ora crea l'istanza UIImageView

Rendi questa variabile come variabile di classe

```
var imageView:UIImageView = UIImageView.init(image: UIImage.init(named: "someImage.jpg"))
```

E poi aggiungilo a scrollview

```
scrollView?.addSubview(imageView)
```

Riferimento

- Scorri la vista Guida alla programmazione per iOS
- UIScrollView Tutorial

Rilevare quando UIScrollView ha terminato lo scorrimento con i metodi dei delegati

scrollViewDidEndDecelerating: indica al delegato che la visualizzazione a scorrimento ha terminato di rallentare il movimento di scorrimento.

Obiettivo 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
}
```

Limita la direzione di scorrimento

Puoi limitare le indicazioni che l'utente può scorrere utilizzando il seguente codice:

```
func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if scrollView.contentOffset.x != 0 {
        scrollView.contentOffset.x = 0
    }
}
```

Ogni volta che l'utente scorre sull'asse x, l'offset del contenuto della scrollView è impostato su 0. Ovviamente è possibile cambiare le `x` s `y` e quindi bloccare la direzione in modo che sia solo orizzontale.

È inoltre necessario assicurarsi di inserire questo codice nel metodo delegato `scrollViewDidScroll(_ scrollView: UIScrollView)` . Altrimenti, non lo farai funzionare.

Inoltre, assicurarsi di aver importato `UIScrollViewDelegate` nella dichiarazione della classe, in questo modo:

```
class ViewController: UIViewController, UIScrollViewDelegate
```

... e imposta il delegate scrollView su self in un metodo come `viewDidLoad(_:)`

```
scrollView.delegate = self
```

Leggi UIScrollView online: https://riptutorial.com/it/ios/topic/1575/uiscrollview

Capitolo 184: UIScrollView con figlio StackView

Examples

Un StackView complesso all'interno dell'esempio di Scrollview

Segue un esempio di cosa può essere fatto con StackViews nidificati, dando all'utente l'impressione di un'esperienza di scorrimento continuo che utilizza elementi o allineamenti di un'interfaccia utente complessi.

Prevenire il layout ambiguo

Una frequente domanda su StackViews all'interno di Scrollviews deriva da ambigui avvisi / heigh sul builder dell'interfaccia. Come ha spiegato questa risposta , è necessario:

1. Aggiungi in UIScrollView a UIView (the contentScrollView);
2. In questo contenutoScrollView, imposta i margini superiore, inferiore, sinistro e destro su 0
3. Impostare anche allineare il centro orizzontalmente e verticalmente;

Scorrimento verso il contenuto all'interno di StackViews nidificati

Il grande trambusto sullo scorrimento è determinare l'offset necessario per presentare (ad esempio) un campo di **testo all'interno di uno StackView con all'interno del ScrollView** .

Se si tenta di ottenere **la posizione di `Textfield.frame.minY` può essere 0** , poiché il frame minY considera solo la distanza tra l'elemento e la parte superiore dello StackView. Quindi devi **considerare tutte le altre visualizzazioni / viste dello stack genitore.**

Una buona soluzione per questo è:

1 - Implementa l'estensione 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)
    }

}
```

Questo considererà tutte le viste genitore e sommerà l'offset necessario per la vista a scorrimento presenti la vista necessaria sullo schermo (per esempio un campo di testo che non può rimanere dietro la tastiera dell'utente)

Esempio di utilizzo:

```
func textViewDidBeginEditing(_ textView: UITextView) {
    self.contentScrollView.scrollToShowView(view: textView)
}
```

Leggi UIScrollView con figlio StackView online: https://riptutorial.com/it/ios/topic/9404/uiscrollview-con-figlio-stackview

Capitolo 185: UISearchController

Sintassi

- UISearchController (searchResultsController: UIViewController?) // Passa nil come parametro se il controller di aggiornamento ricerca visualizza anche il contenuto ricercabile.
- func updateSearchResults (per searchController: UISearchController) // Metodo obbligatorio da implementare quando si adotta il protocollo UISearchResultsUpdating

Parametri

Parametro	Dettagli
`UISearchController.searchBar`	La barra di ricerca per l'installazione nell'interfaccia. *(sola lettura)*
`UISearchController.searchResultsUpdater`	L'oggetto responsabile dell'aggiornamento del contenuto del controller dei risultati di ricerca.
`UISearchController.isActive`	Lo stato presentato dell'interfaccia di ricerca.
`UISearchController.obscuresBackgroundDuringPresentation`	Un booleano che indica se il contenuto sottostante è oscurato durante una ricerca.
`UISearchController.dimsBackgroundDuringPresentation`	Un booleano che indica se il contenuto sottostante è disattivato durante una ricerca.
`UISearchController.hidesNavigationBarDuringPresentation`	Un booleano che indica se la barra di navigazione deve essere nascosta durante la ricerca.
`UIViewController.definesPresentationContext`	Un valore booleano che indica se la vista di questo controller di visualizzazione è coperta quando il controller di visualizzazione o uno dei suoi discendenti presenta un controller di visualizzazione.
`UIViewController.navigationItem.titleView`	Una vista personalizzata visualizzata al centro della barra di navigazione quando il ricevitore è

Parametro	Dettagli
	l'elemento principale in cui può essere posizionata una barra di ricerca.
`UITableViewController.tableView.tableHeaderView`	Restituisce una vista accessoria che viene visualizzata sopra la tabella in cui è possibile posizionare una barra di ricerca.

Osservazioni

Riferimento del framework UIKit:

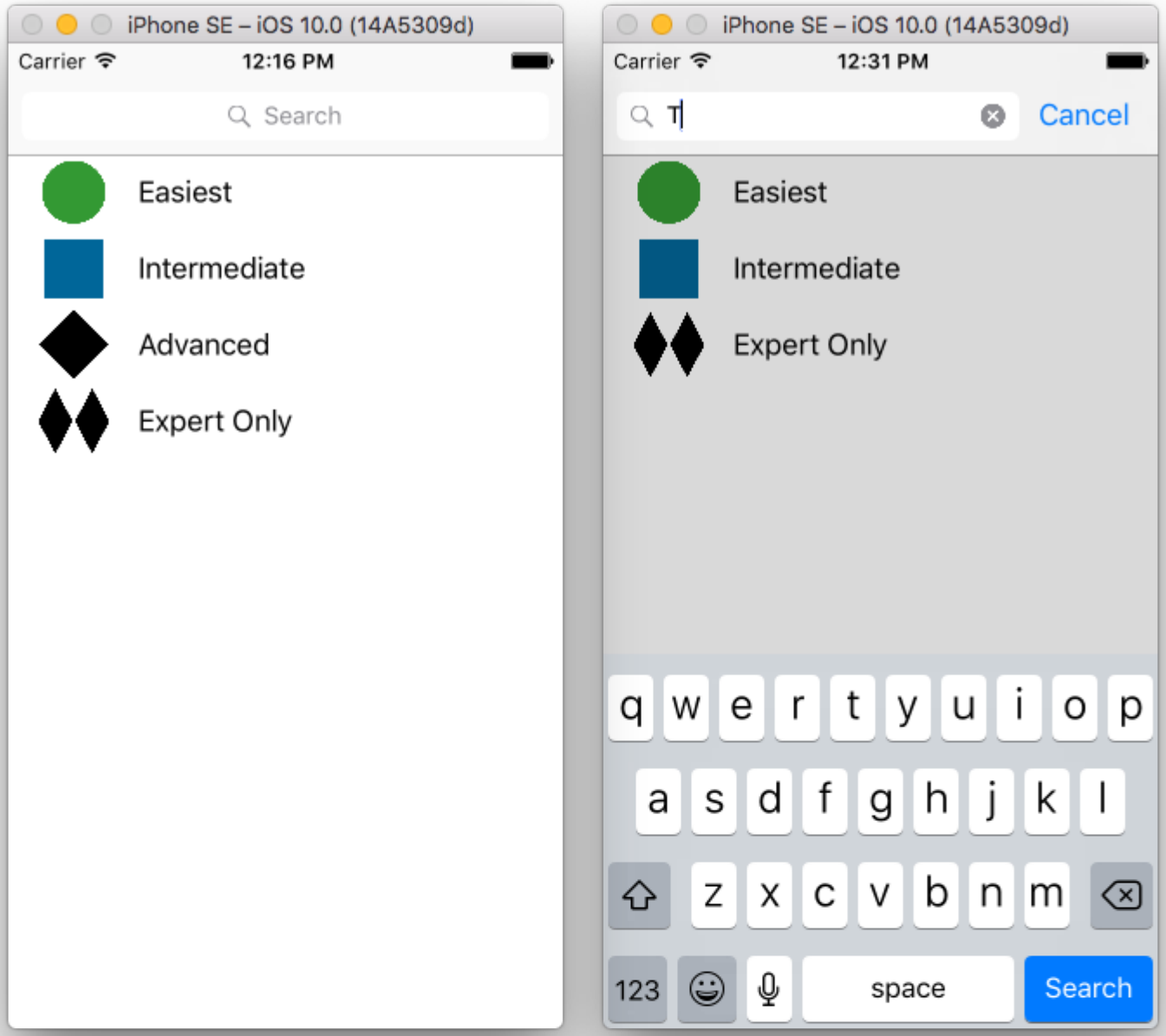
UISearchController

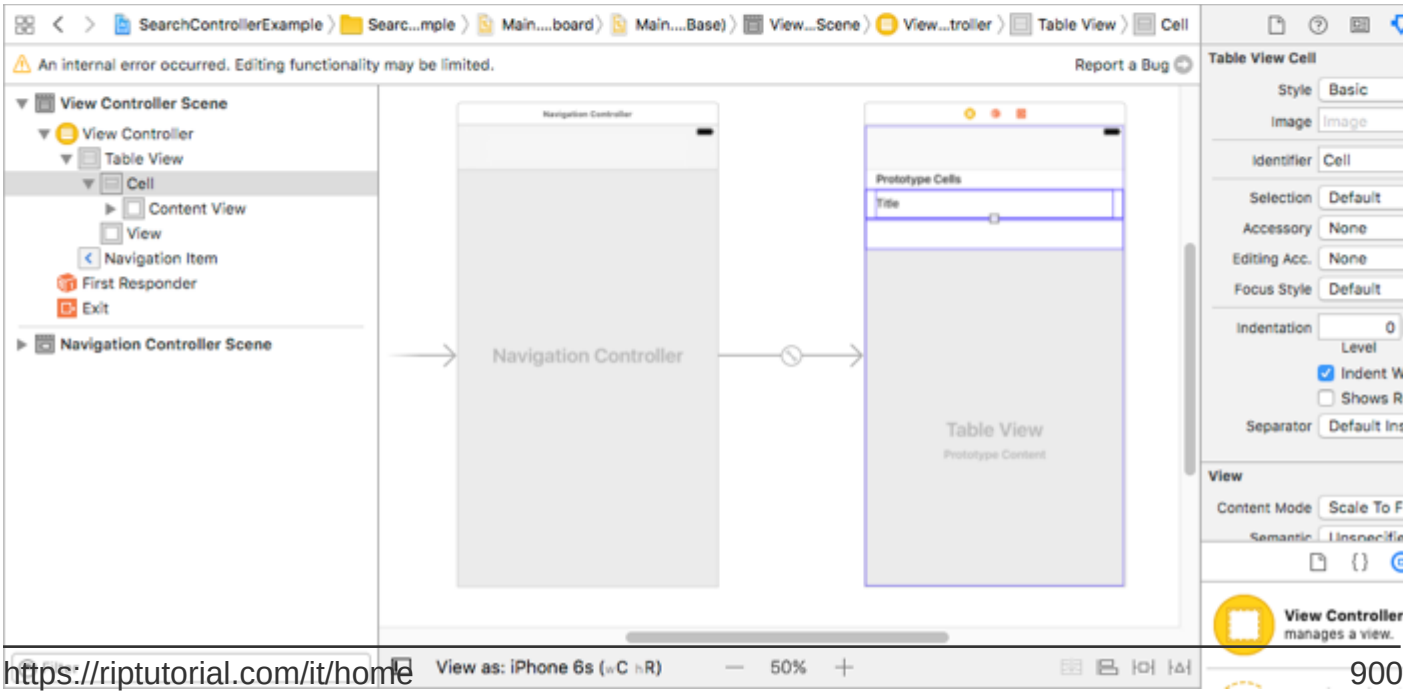
UISearchResultsUpdating

Examples

Barra di ricerca nel titolo della barra di navigazione

Questo esempio utilizza un controller di ricerca per filtrare i dati all'interno di un controller di visualizzazione tabella. La barra di ricerca è posizionata all'interno della barra di navigazione in cui è incorporata la vista tabella.


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
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%

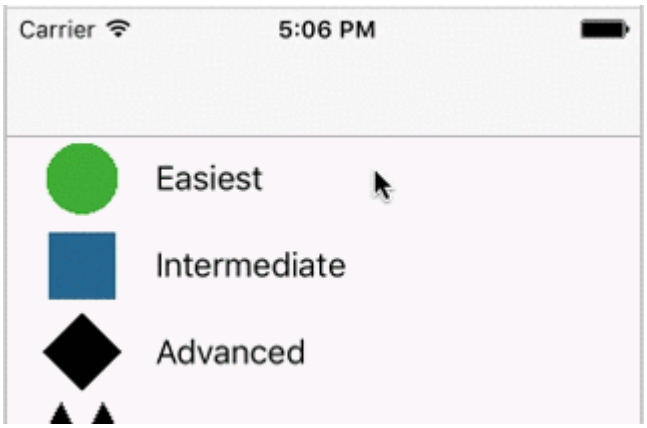
(che contiene la barra di navigazione). Quindi imposta la classe ViewController personalizzata in modo che erediti da `UITableViewController` e adotti il protocollo `UISearchResultsUpdating` .

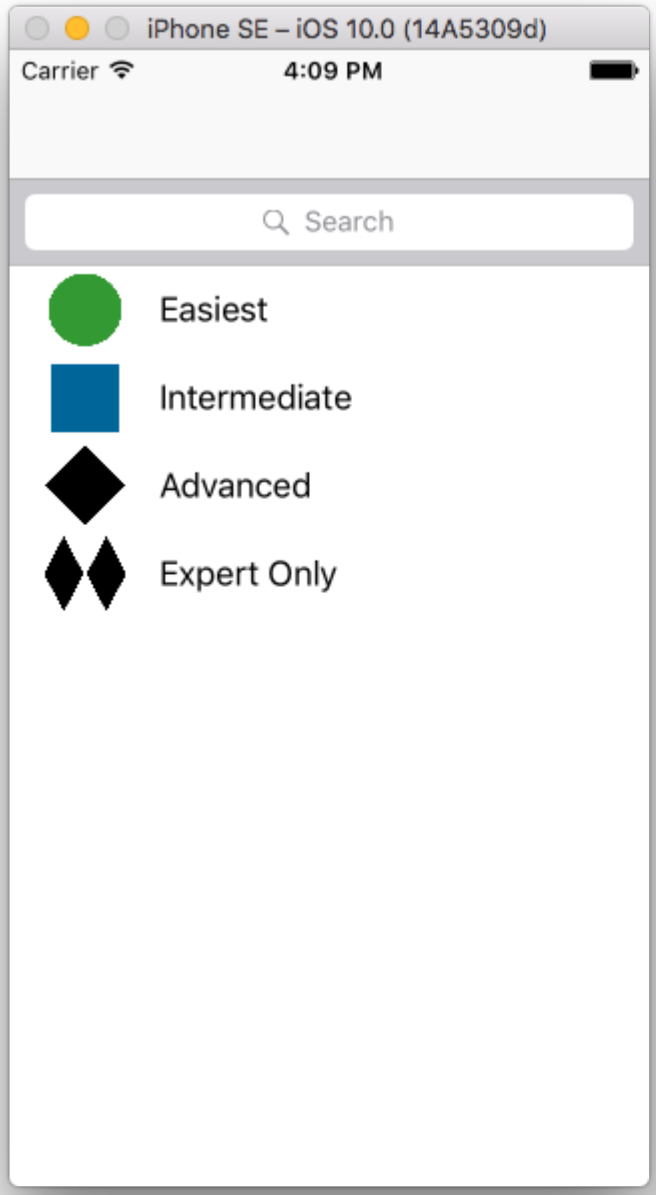
```
class ViewController: UITableViewController, UISearchResultsUpdating {

    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) ->
```

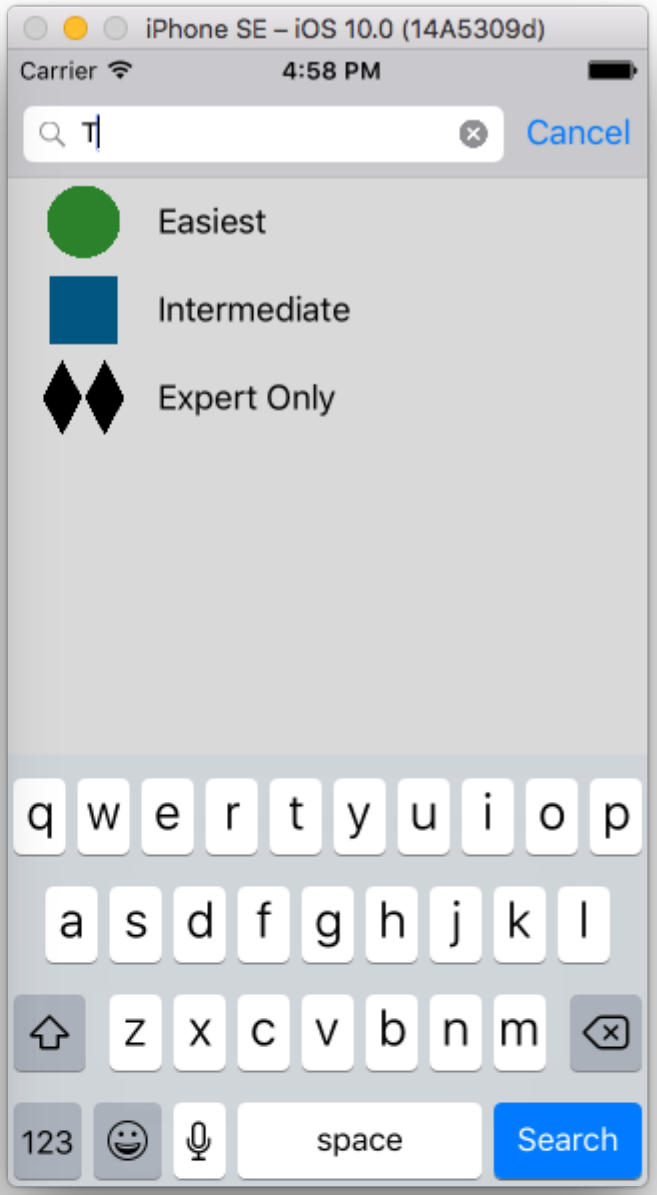
```
UITableViewCell {
        // If the search bar is active, use the searchResults data.
        let entry = searchController.isActive ?
                    searchResults[indexPath.row] : entries[indexPath.row]

        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
        cell.textLabel?.text = entry.title
        cell.imageView?.image = UIImage(named: entry.image)
        return cell
    }
}
```

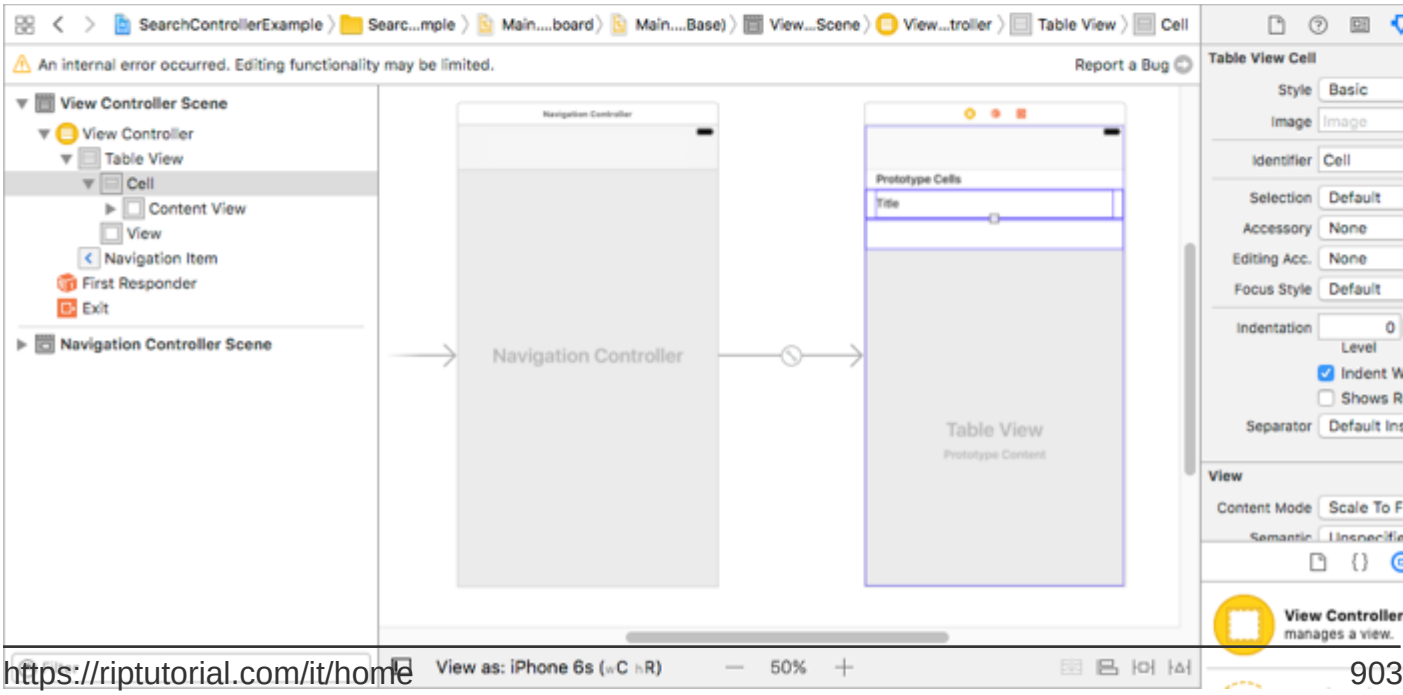
Barra di ricerca in intestazione Vista tabella

Questo esempio utilizza un controller di ricerca per filtrare le celle in un controller di visualizzazione tabella. La barra di ricerca viene posizionata all'interno della vista dell'intestazione della vista tabella. Il contenuto della vista tabella è sfalsato con la stessa altezza della barra di ricerca in modo che la barra di ricerca sia nascosta all'inizio. Scorrendo oltre il bordo superiore della vista tabella, viene rivelata la barra di ricerca. Quindi, quando la barra di ricerca diventa attiva, nasconde la barra di navigazione.




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
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
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%

`UISearchResultsUpdating` :

```
func updateSearchResultsForSearchController(searchController: UISearchController) {

}
```

UISerachController in 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.
}
```

Leggi UISearchController online: https://riptutorial.com/it/ios/topic/2813/uisearchcontroller

Capitolo 186: UISegmentedControl

introduzione

Un oggetto UISegmentedControl è un controllo orizzontale costituito da più segmenti, ciascun segmento funziona come un pulsante discreto. Un controllo segmentato offre un mezzo compatto per raggruppare un numero di controlli.

Examples

Creazione di UISegmentedControl tramite codice

1. Crea una nuova istanza di UISegmentedControl riempita con 3 elementi (segmenti):

```
let mySegmentedControl = UISegmentedControl (items: ["One", "Two", "Three"])
```

2. Cornice di installazione;

```
mySegmentedControl.frame = CGRect(x: 0.0, y: 0.0, width: 300, height: 50)
```

3. Effettua la selezione predefinita (non che i segmenti siano indicizzati da 0):

```
mySegmentedControl.selectedSegmentIndex = 0
```

4. Configura target:

```
mySegmentedControl.addTarget(self, action: #selector(segmentedValueChanged(_:)), for:
.valueChanged)
```

5 Valore maniglia modificato:

```
func segmentedValueChanged(_ sender:UISegmentedControl!) {
    print("Selected Segment Index is : \(sender.selectedSegmentIndex)")
}
```

6. Aggiungi UISegmentedControl alla gerarchia delle viste

```
yourView.addSubview(mySegmentedControl)
```

Leggi UISegmentedControl online: https://riptutorial.com/it/ios/topic/9963/uisegmentedcontrol

Capitolo 187: UISlider

Examples

UISlider

Objective-C

Dichiarare una proprietà del cursore in `ViewController.h` o nell'interfaccia di `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];
```

Gestire l'evento di modifica del cursore

```
- (IBAction)sliderAction:(id)sender {
    NSLog(@"Slider Value %f", sender.value);
}
```

Esempio 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)
```

Gestire l'evento di modifica del cursore

```
 func sliderAction(sender:UISlider!)
 {
    print("value--\(sender.value)")
```

```
}
```

Aggiunta di un'immagine thumb personalizzata

Per aggiungere un'immagine personalizzata per il pollice del cursore, chiama semplicemente il metodo setThumbImage con la tua immagine personalizzata:

Swift 3.1:

```
let slider = UISlider()
let thumbImage = UIImage
slider.setThumbImage(thumbImage, for: .normal)
```

Leggi UISlider online: https://riptutorial.com/it/ios/topic/7402/uislider

Capitolo 188: UISplitViewController

Osservazioni

`UISplitViewController` è una classe contenitore come `UITabViewController` , `UINavigationController` . Separa la vista principale in due controllori di vista masterViewController (PrimaryViewController) e detailViewController (SecondaryViewController). possiamo inviare un array con due controller di visualizzazione e Apple consiglia a `UISplitViewController` come rootviewcontroller per la vostra applicazione. Per interagire tra i viewcontrolers uso `NSNotificationCenter` .

Examples

Interazione Master e Detail View utilizzando i delegati nell'obiettivo C

`UISplitViewController` deve essere il rootViewController dell'applicazione.

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;
}
```

Basta creare un oggetto per `UISplitViewController` e impostare quel viewcontroller come rootviewcontroller per la propria applicazione.

SplitViewController.h

```
#import <UIKit/UIKit.h>
#import "MasterViewController.h"
#import "DetailViewController.h"
@interface ViewController : UISplitViewController
{
DetailViewController *detailVC;
MasterViewController *masterVC;
NSMutableArray *array;
}
@end
```

`MasterViewController` trova sempre sul lato sinistro del dispositivo, è possibile impostare la larghezza nei metodi delegati `UISplitViewCOntroller` e `DetailViewController` trova sul lato destro

dell'applicazione

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;
}
```

Created master and detail ViewControllers vengono aggiunti a un array impostato su `self.viewControllers` in `UISplitViewController` . `self.preferredDisplayMode` è la modalità impostata per la visualizzazione della documentazione Apple e `DetailViewController` Apple per DisplayMode . `self.presentsWithGesture` abilita il gesto di scorrimento per la visualizzazione di `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
```

Creare un `DetailViewDelegate` delegato con `sendSelectedNavController:(UIViewController *)viewController` metodo per inviare `UIViewController` al `DetailViewcontroller` . Quindi in `MasterViewController` il mainTableView è il tableview nel lato sinistro. ViewControllerArray contiene tutti i `UIViewControllers` che devono essere visualizzati in `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
```

Creare alcuni `UIViewControllers` e aggiungerlo a un array. La vista Tabella viene inizializzato allora `didSelectRowAtIndexPath` metodo mando un `UIViewController` al `DetailViewController` utilizzando `detailDelegate` con il corrispondente `UIViewController` nella matrice come parametro

DetailViewController.h

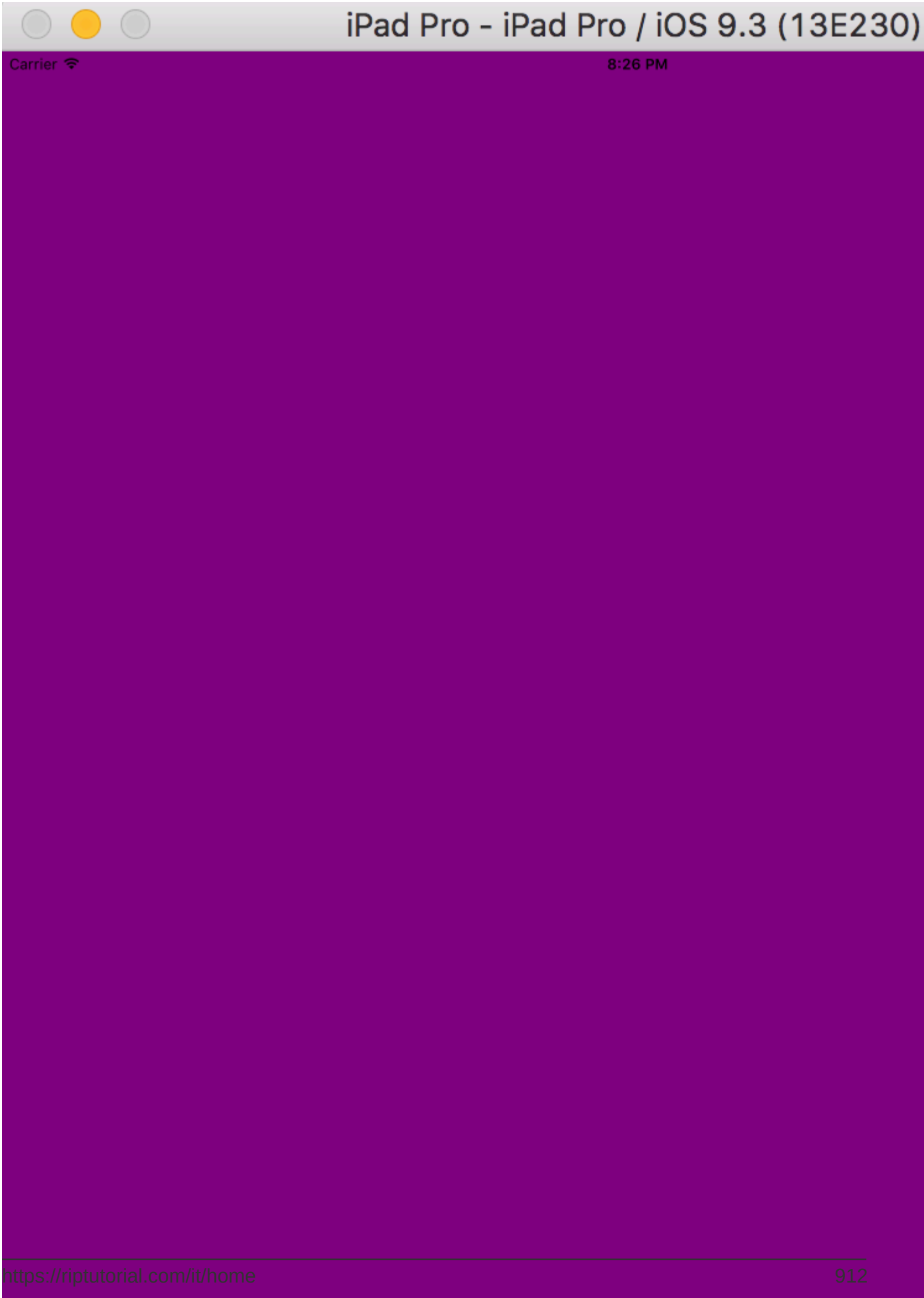
```
#import <UIKit/UIKit.h>

@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
```

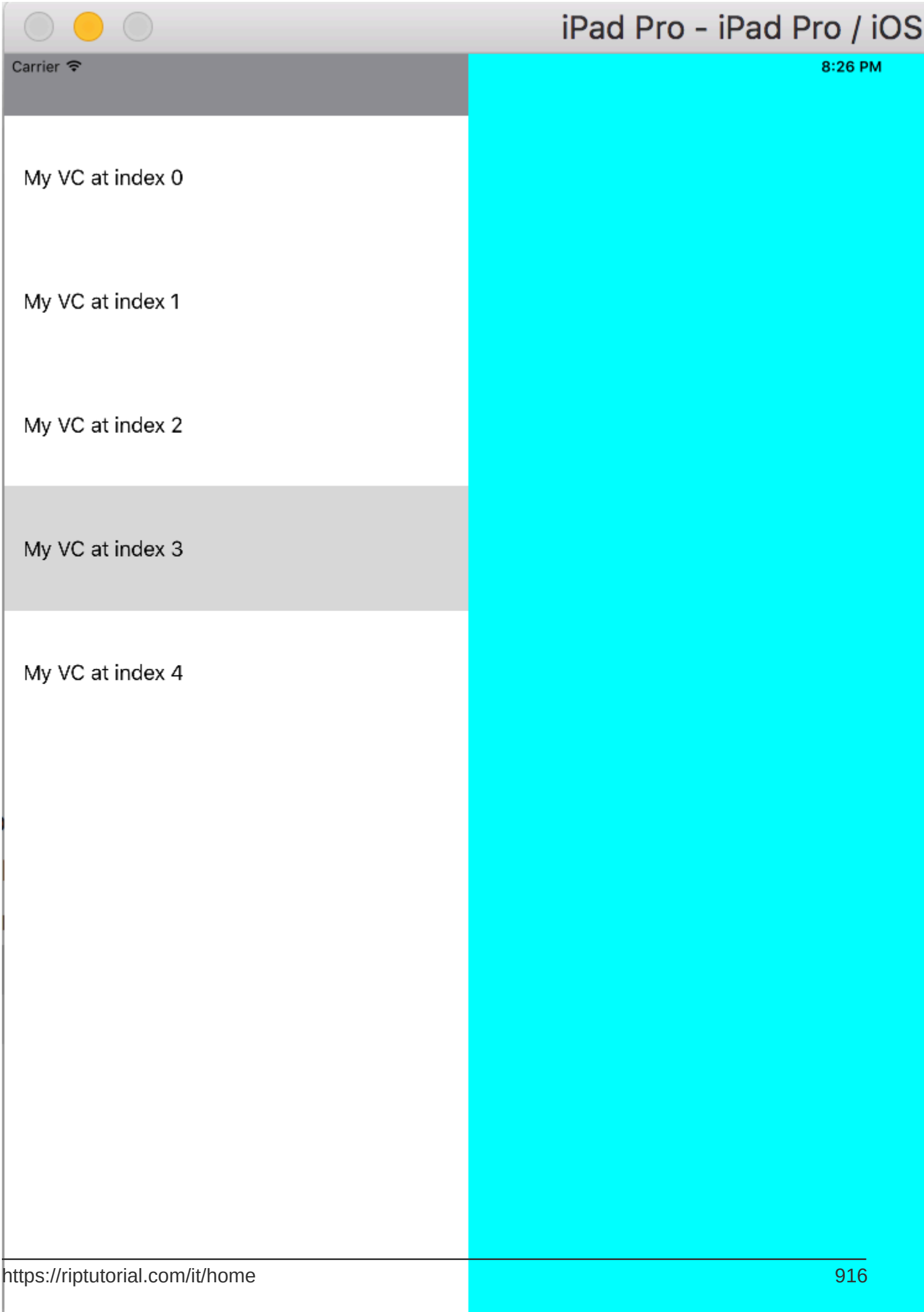
`sendSelectedNavController` è dichiarato qui con la rimozione di tutte le viste in `DetailViewController` e l'aggiunta di `UIViewController` passato da `MasterViewController`

Aggiunta di alcune schermate dell'applicazione


iPad Pro - iPad Pro / iOS 9.3 (13E230)
Carrier
8:26 PM

poiché abbiamo dato il `preferredDisplayMode` DisplayMode come automatico per lo scorrimento dello schermo otteniamo il `MasterViewController` come allegato nell'immagine sottostante, ma in modalità Landscape otteniamo sia il `MasterViewController` che il `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

Capitolo 189: UISplitViewController

Osservazioni

In iOS 8 e versioni successive, è possibile utilizzare la classe `UISplitViewController` su tutti i dispositivi iOS, nelle versioni precedenti di iOS, la classe è disponibile solo su iPad. `UISplitViewController` è una classe contenitore come `UITabViewController` , `UINavigationController` . Separa la vista principale in due `UIViewControllers` `masterViewController` (PrimaryViewController) e `detailViewController` (SecondaryViewController). possiamo inviare un `NSArray` con due `UIViewControllers` e Apple consiglia `UISplitViewController` come rootviewcontroller per la tua applicazione. Per interagire tra `UIViewControllers` utilizzo `NSNotificationCenter` .

Examples

Interagire tra vista principale e dettaglio utilizzando i delegati nell'obiettivo C

`UISplitViewController` bisogno del controller della vista radice della finestra dell'app

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;
}
```

Basta creare un oggetto per il tuo `UISplitVIewController` e impostarlo come rootViewController per la tua applicazione.

SplitViewController.h

```
#import <UIKit/UIKit.h>
#import "MasterViewController.h"
#import "DetailViewController.h"
@interface ViewController : UISplitViewController
{
DetailViewController *detailVC;
MasterViewController *masterVC;
NSMutableArray *array;
}
@end
```

`MasterViewController` è un `UIViewController` impostato sul lato sinistro del dispositivo, è possibile

impostare la larghezza in `UISplitViewController` utilizzando `maximumPrimaryColumnWidth` e `DetailViewController` trova sul lato destro

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;
}
```

Il master e il dettaglio di `UIViewController` sono aggiunti a un `NSArray` impostato su `self.viewControllers` . `self.preferredDisplayMode` è la modalità impostata per la visualizzazione di `MasterViewController` e `DetailViewController` . `self.presentsWithGesture` abilita il gesto di scorrimento per la visualizzazione di `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
```

Creare un `DetailViewDelegate` delegato con `sendSelectedNavController` metodo per inviare i `UIViewControllers` al `DetailViewController` . Quindi in `MasterViewController` viene creato un `UITableView` . `ViewControllerArray` contiene tutti i `UIViewControllers` che devono essere visualizzati in `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
```

Creato alcuni `UIViewController` e lo ha aggiunto a un `NSMutableArray` . `UITableView` viene inizializzato allora `didselectrowatindexpath` metodo mando un `UIViewController` al `DetailViewController` utilizzando `detailDelegate` delegato con il corrispondente `UIViewController` nella `NSMutableArray` come parametro

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
```

Il `sendSelectedNavController` è dichiarato qui con la rimozione di tutto `UIView` s' nel `DetailViewController` e aggiungendo il passato `UIViewController` dal `MasterViewController` .

Leggi UISplitViewController online: https://riptutorial.com/it/ios/topic/4844/uisplitviewcontroller

Capitolo 190: UIStackView

Examples

Creare una vista stack orizzontale a livello di codice

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.
```

veloce

```
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.
```

Creare una vista stack verticale a livello di codice

veloce

```
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.
```

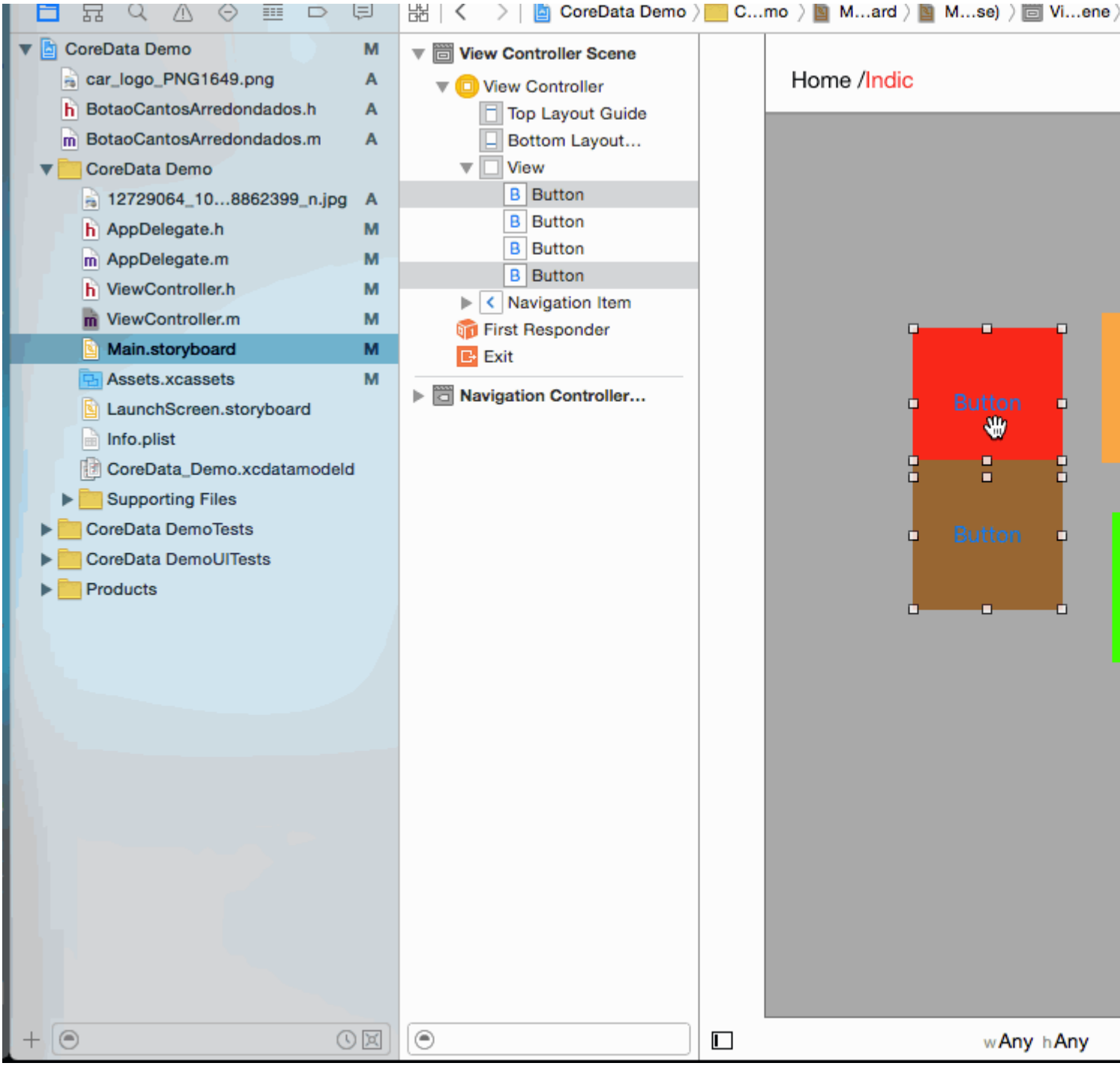
Objective-C

```
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.
```

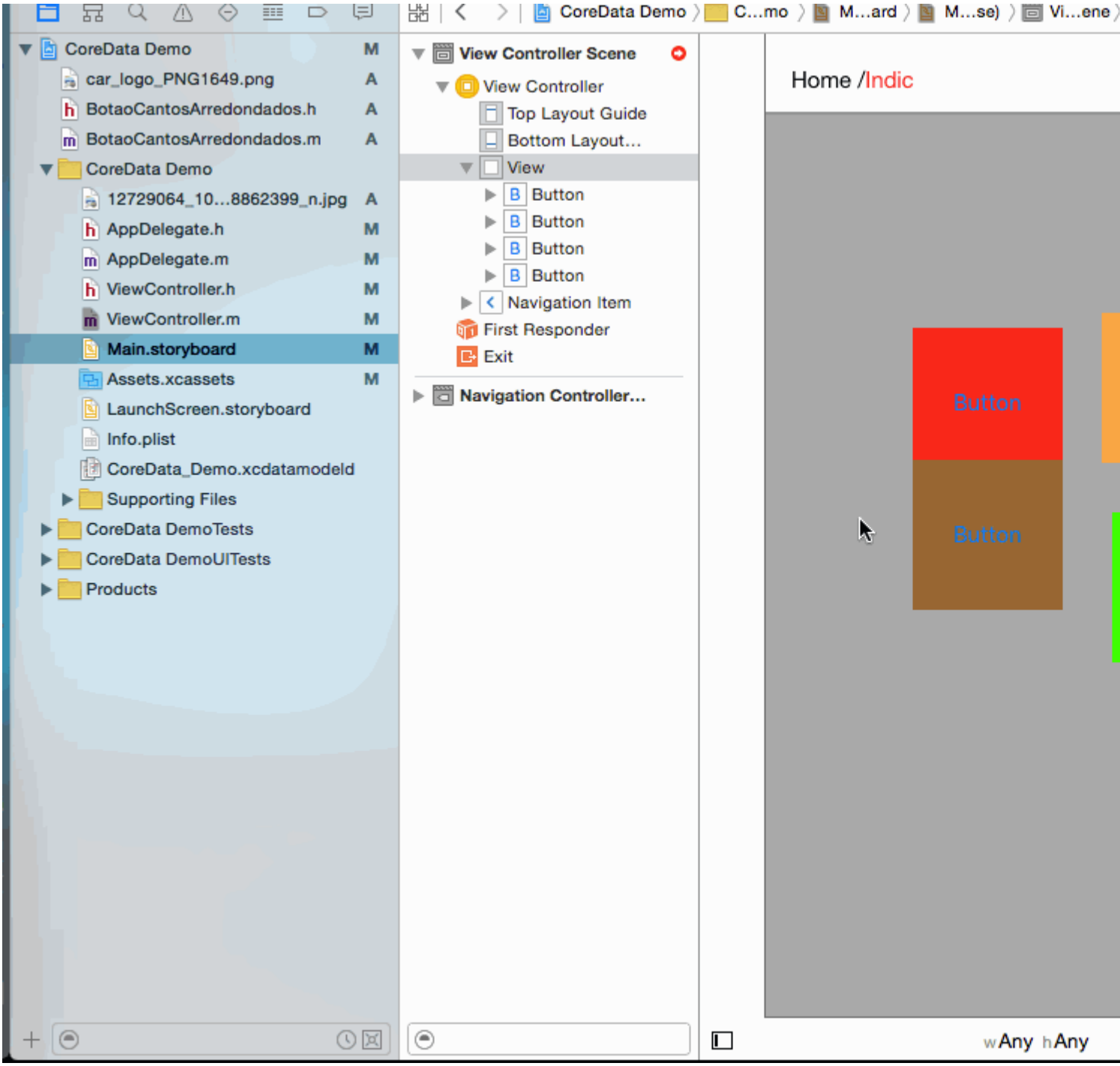
Pulsanti centrali con UIStackview

Step 1: - Prendi 4 pulsanti nella tua Storyboard. Button1, Button2, Button 3, Button4

Passaggio 2: - Assegna altezza e larghezza fisse a tutti i pulsanti.

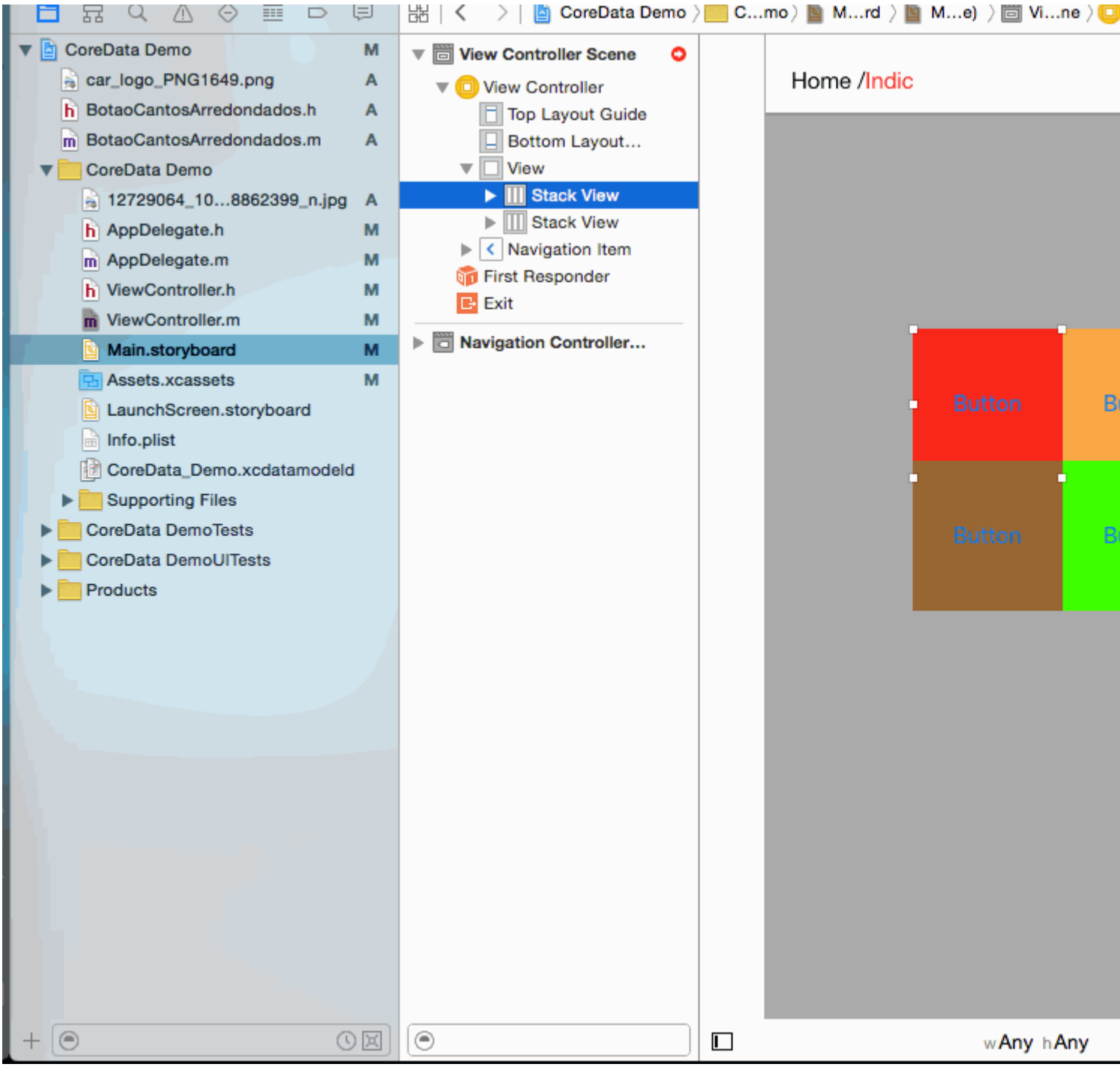


Step 3: - Tutte le coppie di 2 - 2 pulsanti in 2 stackview.

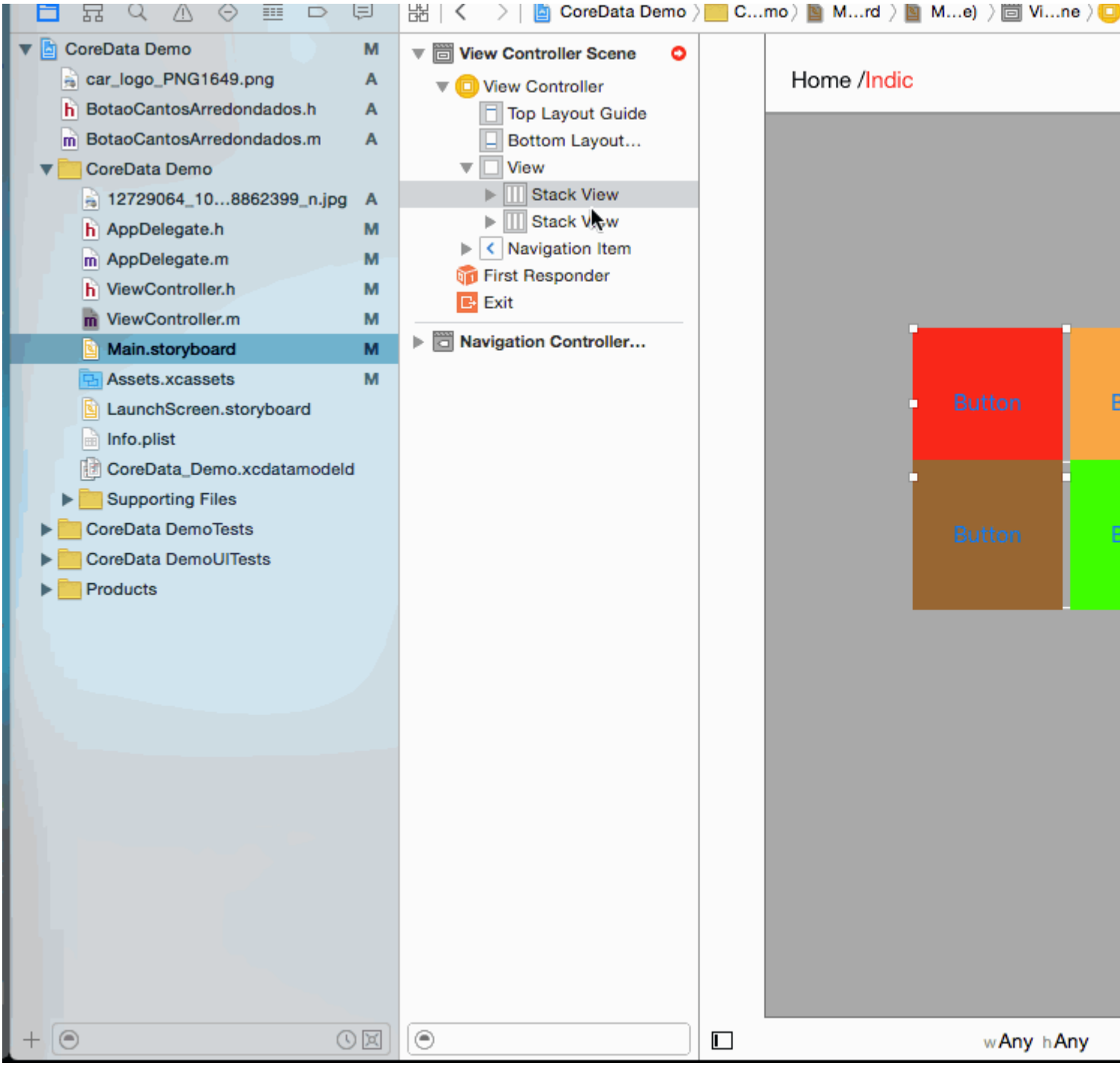


Passo 4: - Imposta la proprietà UIStackview per entrambi.

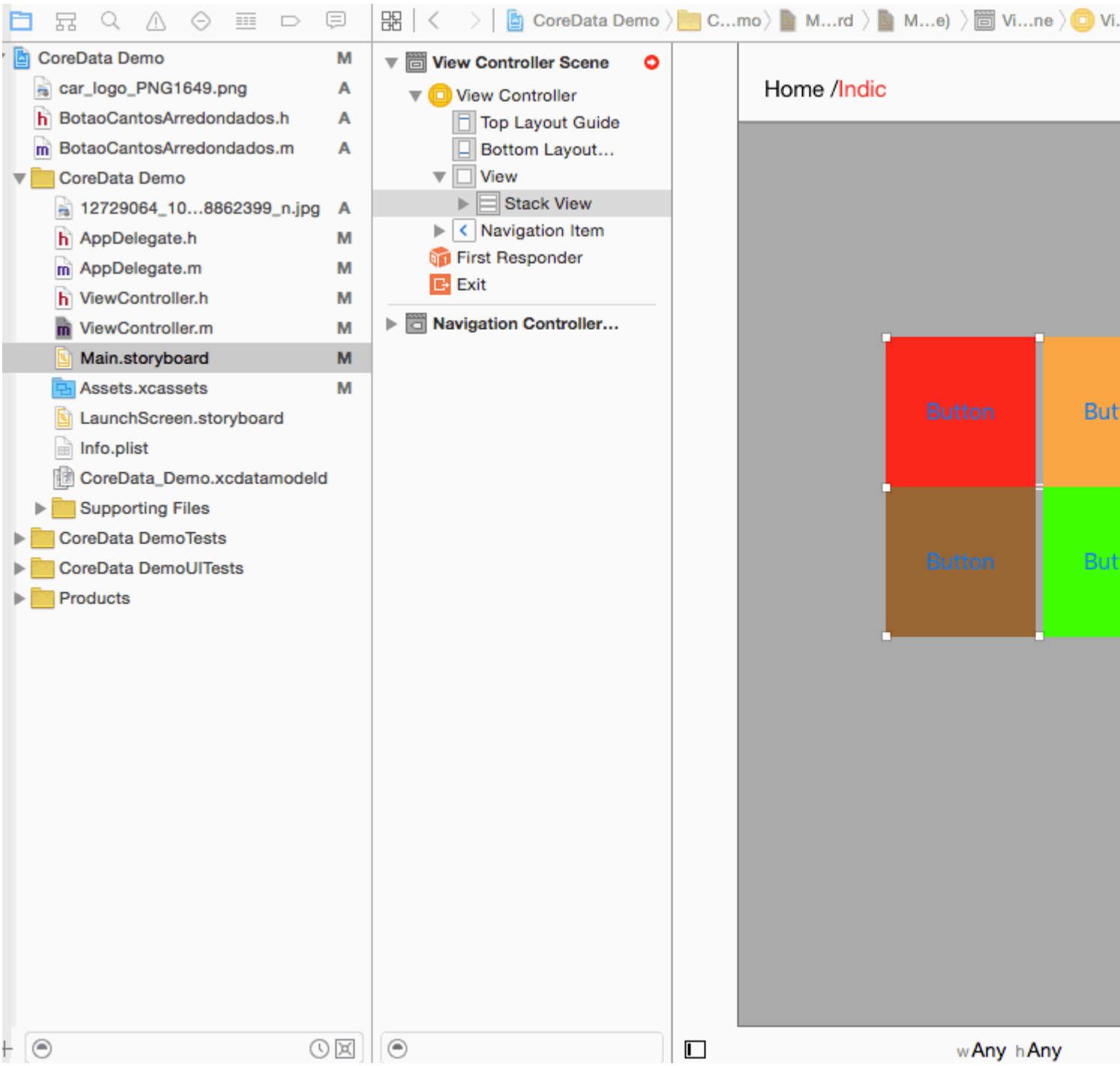
```
    Distribution -> Fill Equally
         Spacing -> 5 (as per your requirement)
```



Step 5: - Aggiungi entrambi Stackview in uno Stackview

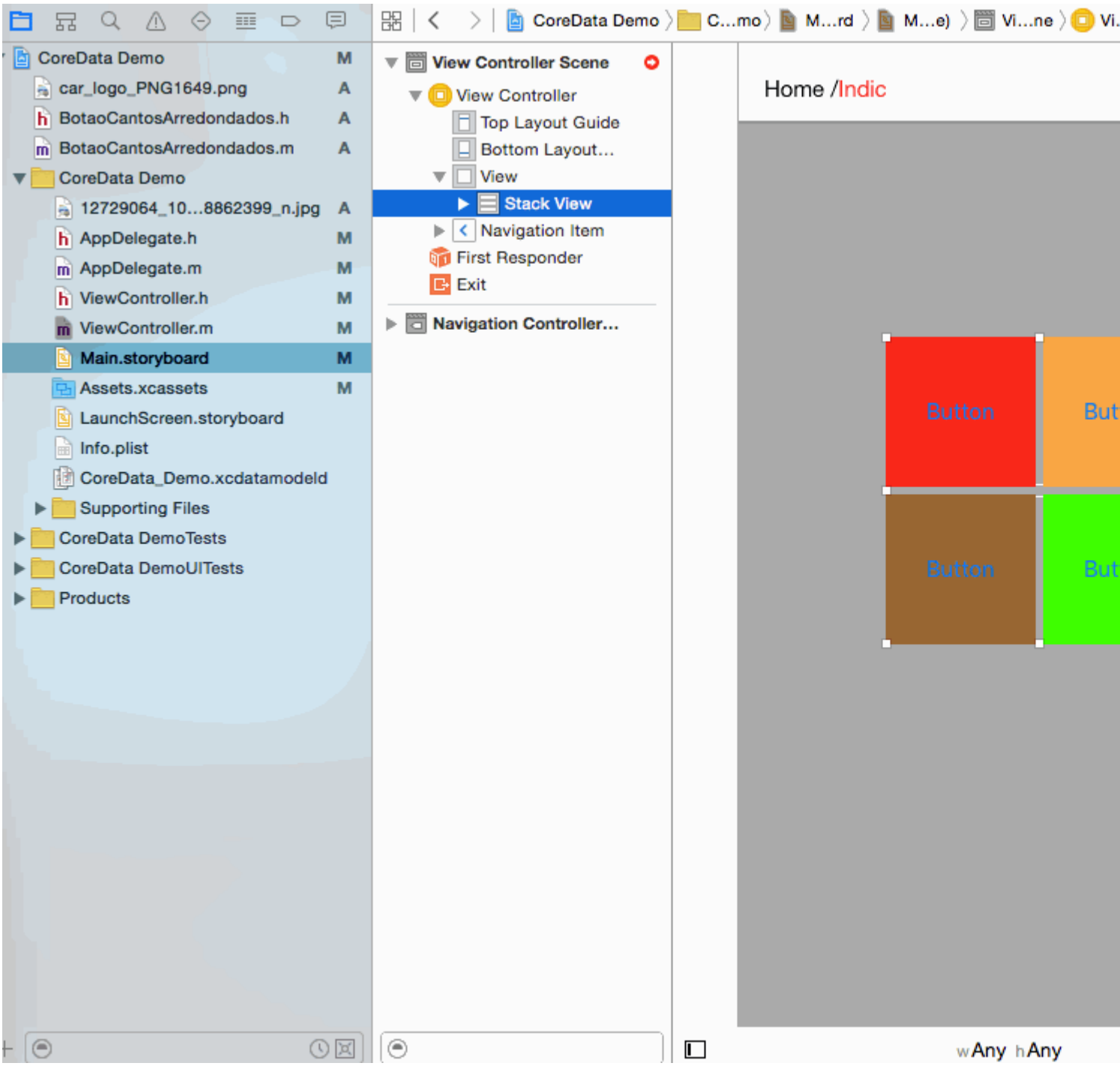


Step 6: - Set `Distribution = Fill equally Spacing =5` in stackview principale (set Secondo il tuo requisito)

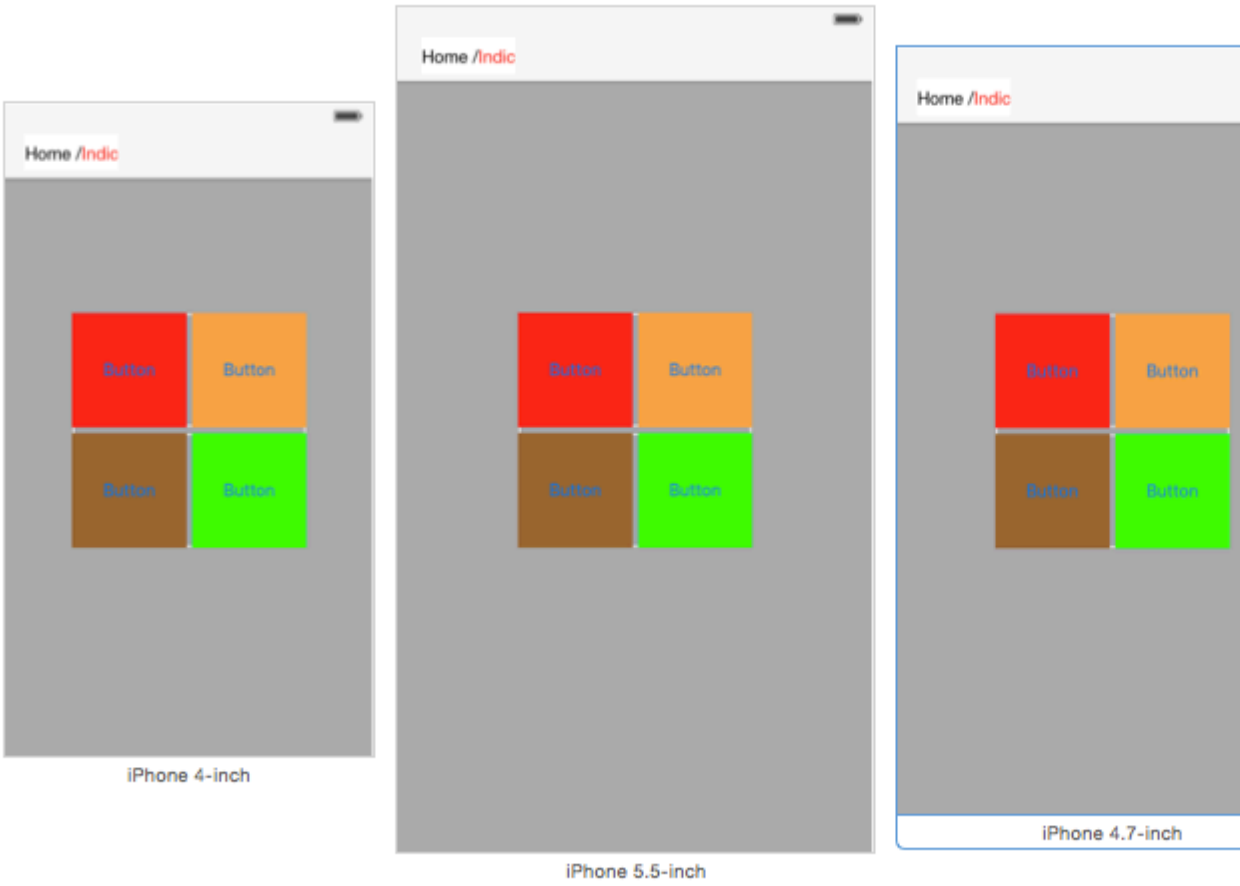


Step 7: - Ora imposta Vincolo su stackview principale

```
    center Horizontally in container

    center vertically in container

      and select Update Frame.
```



Passaggio 8: - È l'ora dell'output per tutti i dispositivi.



Leggi UIStackView online: https://riptutorial.com/it/ios/topic/1390/uistackview

Capitolo 191: UIStoryboard

introduzione

Un oggetto UIStoryboard incapsula il grafico del controller di visualizzazione memorizzato in un file di risorse dello storyboard di Interface Builder. Questo grafico del controller di visualizzazione rappresenta i controller di visualizzazione dell'intera o dell'intera interfaccia utente dell'applicazione.

Examples

Ottenere un'istanza di UIStoryboard a livello di codice

SWIFT:

Ottenere un'istanza di **UIStoryboard a livello** di **codice** può essere eseguita come segue:

```
    let storyboard = UIStoryboard(name: "Main", bundle: nil)
```

dove:

- **name** => il nome dello storyboard senza l'estensione
- **bundle** => il pacchetto contenente il file storyboard e le relative risorse. Se si specifica nil, questo metodo viene visualizzato nel bundle principale dell'applicazione corrente.

Ad esempio, puoi utilizzare l'istanza creata in precedenza per accedere a un determinato **UIViewController** istanziato all'interno di tale storyboard:

```
   let viewController = storyboard.instantiateViewController(withIdentifier: "yourIdentifier")
```

Objective-C:

Ottenere un'istanza di **UIStoryboard** in Objective-C può essere fatto come segue:

```
 UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"MainStoryboard" bundle:nil];
```

Esempio di accesso a **UIViewController** istanziato all'interno di tale storyboard:

```
MyViewController *myViewController = [storyboard
instantiateViewControllerWithIdentifier:@"MyViewControllerIdentifier"];
```

Apri un altro storyboard

```
let storyboard = UIStoryboard(name: "StoryboardName", bundle: nil)
let vc = storyboard.instantiateViewController(withIdentifier: "ViewControllerID") as
YourViewController
self.present(vc, animated: true, completion: nil)
```

Leggi UIStoryboard online: https://riptutorial.com/it/ios/topic/8795/uistoryboard

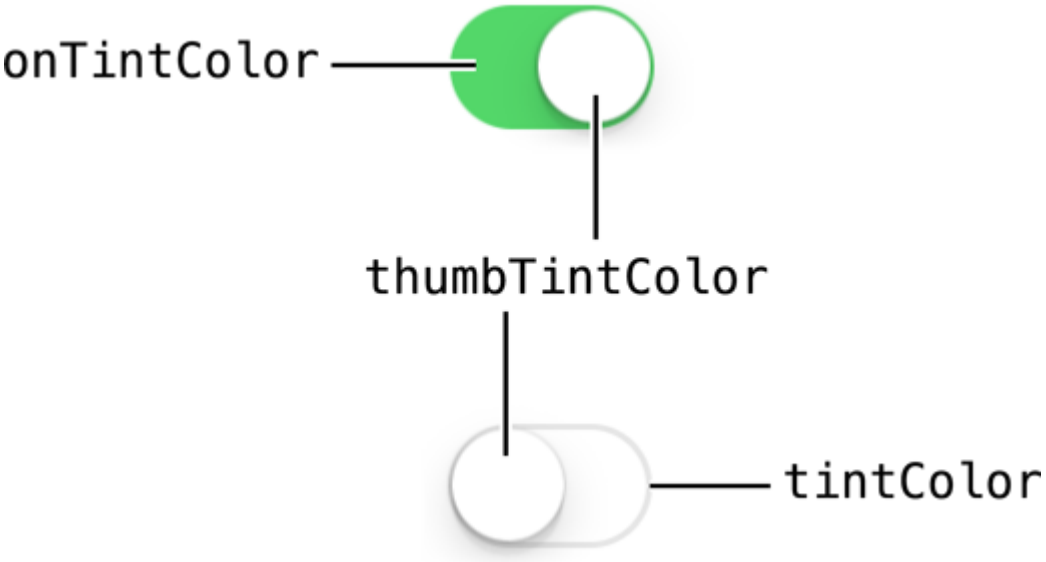
Capitolo 192: UISwitch

Sintassi

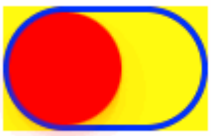
- (Instancetype) initWithFrame: (CGRect) telaio;
- (void) setOn: (BOOL) su animato: (BOOL) animato;
- (instancetype nullable) initWithCoder: (NSCoder *) aDecoder;

Osservazioni

1. Riferimento UISwitch: documentazione Apple



2. Un altro riferimento dato da: Enoch Huang



Examples

Imposta On / Off

Objective-C

```
[mySwitch setOn:YES];
//or
[mySwitch setOn:YES animated:YES];
```

veloce

```
mySwitch.setOn(false)
//or
mySwitch.setOn(false, animated: false)
```

Imposta il colore di sfondo

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];
```

veloce

```
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)
```

Imposta colore tinta

Objective-C

```
//for off-state
mySwitch.tintColor = [UIColor blueColor];
[mySwitch setTintColor: [UIColor blueColor]];

//for on-state
mySwitch.onTintColor = [UIColor cyanColor];
[mySwitch setOnTintColor: [UIColor cyanColor]];
```

veloce

```
//for off-state
mySwitch.tintColor = UIColor.blueColor()

//for on-state
mySwitch.onTintColor = UIColor.cyanColor()
```

Imposta immagine per stato On / Off

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"]];
```

veloce

```
//set off-image
mySwitch.offImage = UIImage(named: "off_image")

//set on-image
mySwitch.onImage = UIImage(named: "on_image")
```

Leggi UISwitch online: https://riptutorial.com/it/ios/topic/2182/uiswitch

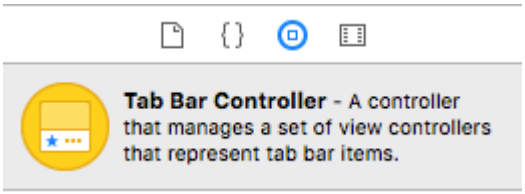
Capitolo 193: UITabBarController

Examples

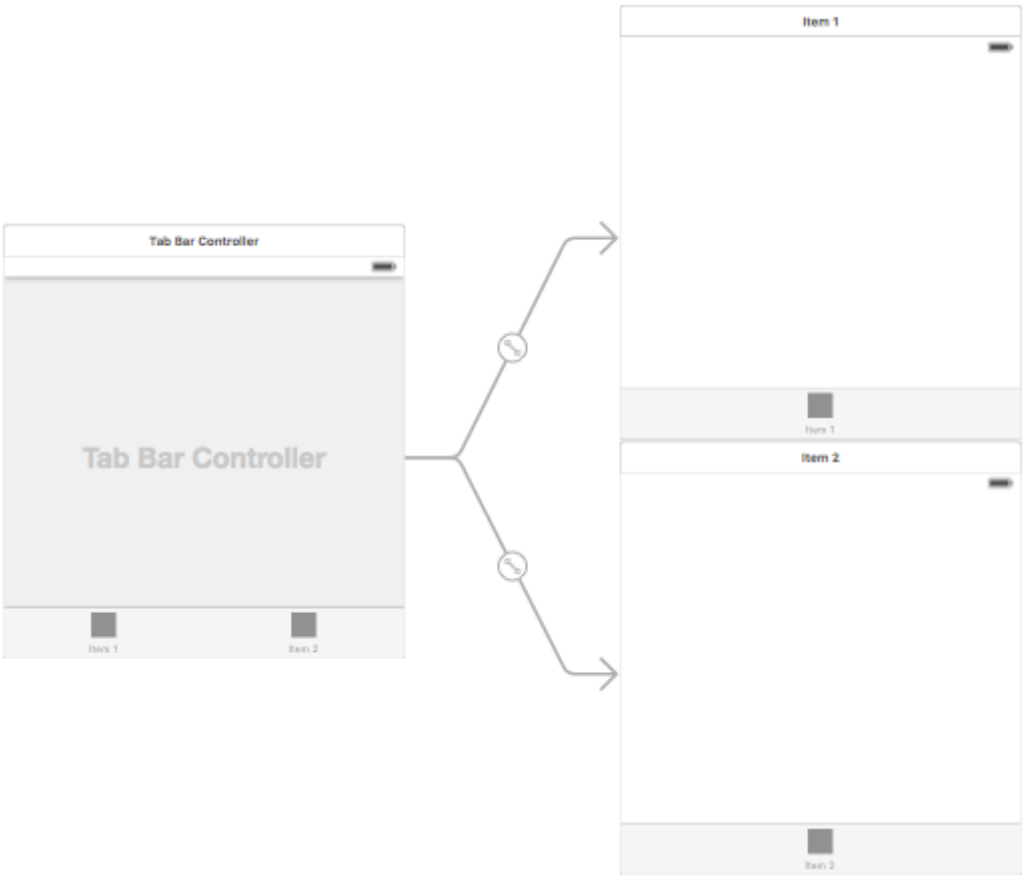
Crea un'istanza

Una "barra delle schede" si trova comunemente nella maggior parte delle app iOS e viene utilizzata per presentare viste distinte in ciascuna scheda.

Per creare un controller della barra delle schede utilizzando il generatore di interfacce, trascinare una barra della scheda Controller dalla libreria degli oggetti nell'area di disegno.



Per impostazione predefinita, un controller barra delle linguette viene fornito con due viste. Per aggiungere altre viste, controlla il trascinamento dal controller della barra delle schede alla nuova vista e seleziona "Visualizza controller" nel menu a discesa "segue".



Modifica del titolo e dell'icona della barra delle schede

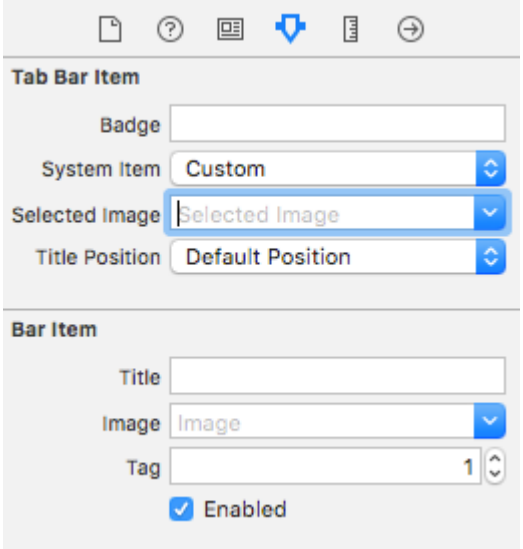
Usando la Story Board:

Seleziona la voce della barra delle schede dal controller della vista corrispondente e vai all'ispettore degli attributi

Se si desidera un'icona e un titolo incorporati, impostare "Voce sistema" sul valore corrispondente.

Per un'icona personalizzata, aggiungi le immagini necessarie alla cartella delle risorse e imposta l''elemento del sistema' da precedente a 'personalizzato'.

Ora imposta l'icona da mostrare quando la scheda è selezionata dall'elenco a discesa "immagine selezionata" e l'icona della scheda predefinita dal menu a discesa "immagine". Aggiungi il titolo corrispondente nel campo "titolo".



livello di programmazione:

Nel metodo `viewDidLoad()` del controller di visualizzazione, aggiungere il seguente codice:

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")
```

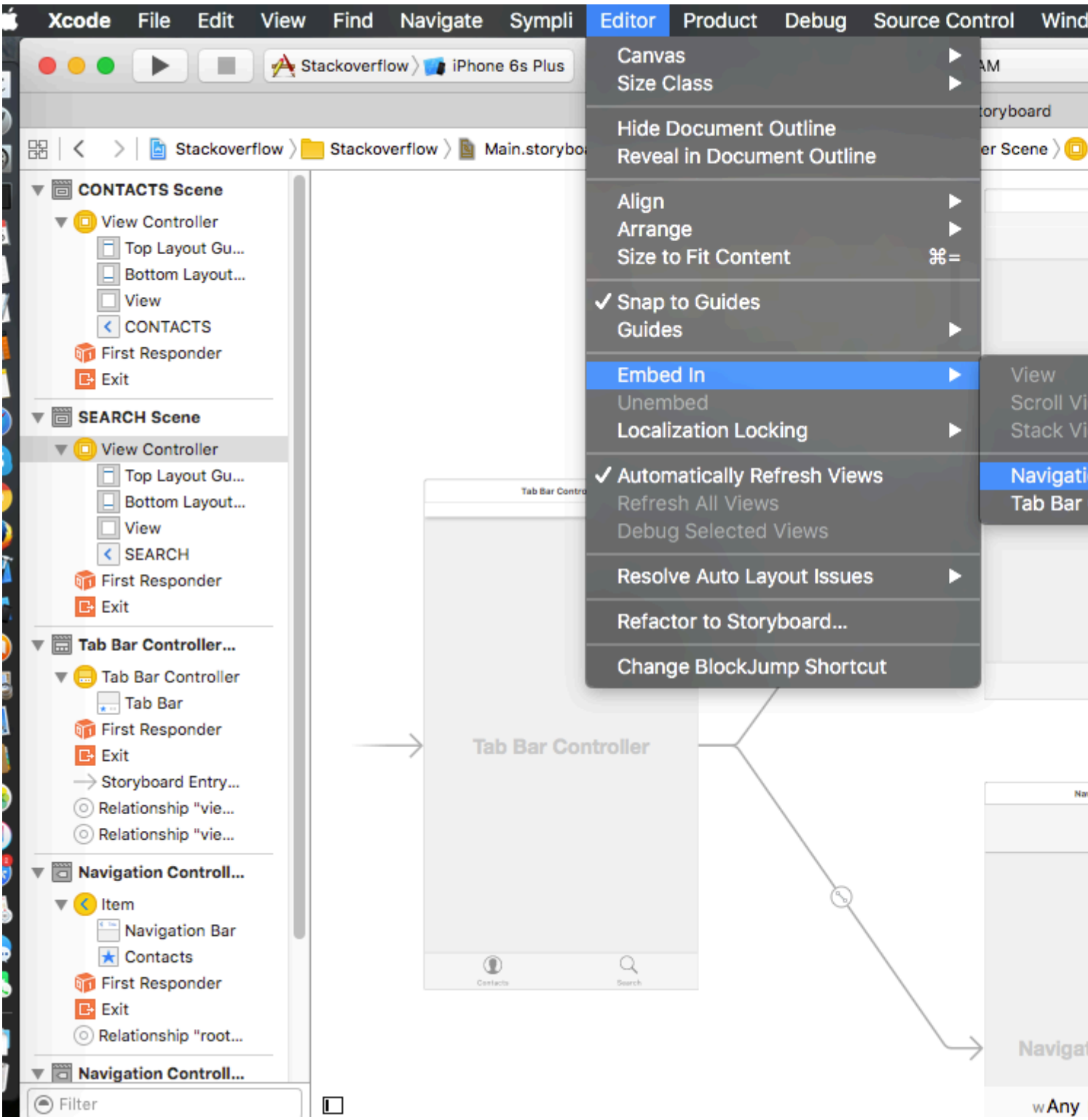
Controller di navigazione con TabBar

Il controller di navigazione può essere incorporato in ogni scheda usando lo storyboard stesso.

Può essere come nello screenshot aggiunto.

Per aggiungere un controller di navigazione a un controller di visualizzazione che si collega dal controller della barra delle linguette, ecco il flusso

- Seleziona il controller di visualizzazione per il quale dobbiamo aggiungere il controller di navigazione. Qui lascia che sia Search View Controller come display di selezione.
- Dal menu **Editor** di Xcode, selezionare **Incorpora in ->** Opzione **Controller di navigazione**



Personalizzazione del colore della barra delle tabulazioni

```
[[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];
```

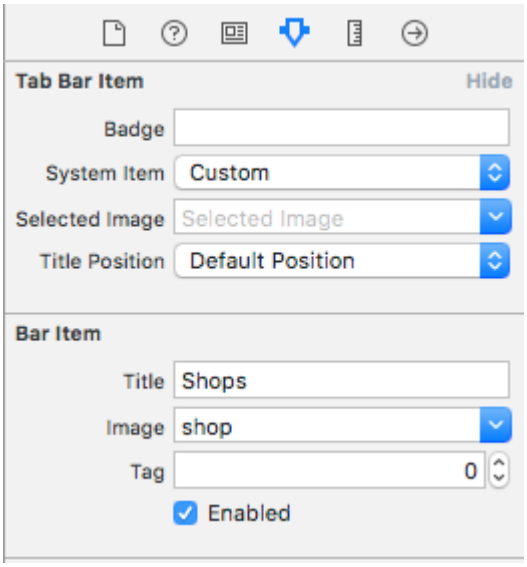
UITabBarController con selezione dei colori personalizzata

UITabBarController building in `Swift 3` Cambia il colore dell'immagine e il titolo in base alla selezione con la modifica del colore della scheda selezionata.

```
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
    }

}
```

Scegliere l'immagine per la barra delle schede e impostare qui il titolo della scheda



Seleziona un'altra scheda

Crea un controller Barra di Tab in modo programmatico senza Storyboard

```
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
    }
```

Leggi UITabBarController online: https://riptutorial.com/it/ios/topic/2763/uitabbarcontroller

Capitolo 194: UITableView

introduzione

Una vista semplice, ampiamente utilizzata, ma molto potente che può presentare i dati in un modulo elenco utilizzando le righe e una singola colonna. Gli utenti possono scorrere verticalmente attraverso gli elementi in una vista tabella e, facoltativamente, manipolare e selezionare il contenuto.

Sintassi

- - (CGFloat) tableView: (UITableView *) tableView heightForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (CGFloat) tableView: (UITableView *) tableView heightForHeaderInSection: (NSInteger) sezione;
- - (CGFloat) tableView: (UITableView *) tableView heightForFooterInSection: sezione (NSInteger);
- - (UIView *) tableView: (UITableView *) tableView viewForHeaderInSection: (NSInteger) sezione;
- - (UIView *) tableView: (UITableView *) tableView viewForFooterInSection: (NSInteger) sezione;
- - (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 *) proposedDestinationIndexPath;
- - (NSInteger) tableView: (UITableView *) tableView indentationLevelForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSInteger) tableView: (UITableView *) tableView numberOfRowsInSection: (NSInteger) section;
- - (UITableViewCell *) tableView: (UITableView *) tableView cellForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSInteger) numberOfSectionsInTableView: (UITableView *) tableView;
- - (NSString *) tableView: (UITableView *) tableView titleForHeaderInSection: (NSInteger) sezione; // stile di carattere fisso. usa la visualizzazione personalizzata (UILabel) se vuoi qualcosa di diverso
- - (NSString *) tableView: (UITableView *) tableView titleForFooterInSection: (NSInteger) section;
- - (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) index;
- - (void) tableView: (UITableView *) tableView commitEditingStyle: (UITableViewCellEditingStyle) editStyle forRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView moveRowAtIndexPath: (NSIndexPath *) sourceIndexPath toIndexPath: (NSIndexPath *) destinationIndexPath;

Osservazioni

`UITableView` è una sottoclasse di `UIScrollView` . Le classi che seguono il protocollo `UITableViewDelegate` seguono anche il protocollo `UIScrollViewDelegate` . `UITableView` può essere utile per mostrare elenchi lunghi o indeterminati attraverso le sue celle, mentre `UIScrollView` è migliore per quando la dimensione delle viste da mostrare è nota in anticipo.

Examples

Celle auto dimensionanti

In iOS 8 Apple ha introdotto la cella di dimensionamento automatico. Disegna esplicitamente UITableViewCells con Autolayout e UITableView si prende cura del resto per te. L'altezza della riga viene calcolata automaticamente, per impostazione predefinita il valore di `rowHeight` è UITableViewAutomaticDimension.

UITableView struttura `estimatedRowHeight` viene utilizzata quando auto-dimensionamento delle cellule è il calcolo.

> Quando si crea una cella di visualizzazione tabella con ridimensionamento automatico, è necessario impostare questa proprietà e utilizzare i vincoli per definire la dimensione della cella.
>
> *- Apple, la documentazione di UITableView*

```
self.tableView.estimatedRowHeight = 44.0
```

Notare che heightForRowAtIndexPath del delegato di `heightForRowAtIndexPath` *non* è *necessario* se si desidera avere un'altezza dinamica per tutte le celle. Basta impostare la proprietà sopra se necessario e prima di ricaricare o caricare la vista tabella. Tuttavia, puoi impostare l'altezza specifica di una cella mentre altre sono dinamiche tramite la seguente funzione:

veloce

```
override func tableView(tableView: UITableView, heightForRowAtIndexPath indexPath:
NSIndexPath) -> CGFloat {
    switch indexPath.section {
    case 1:
        return 60
    default:
        return UITableViewAutomaticDimension
    }
}
```

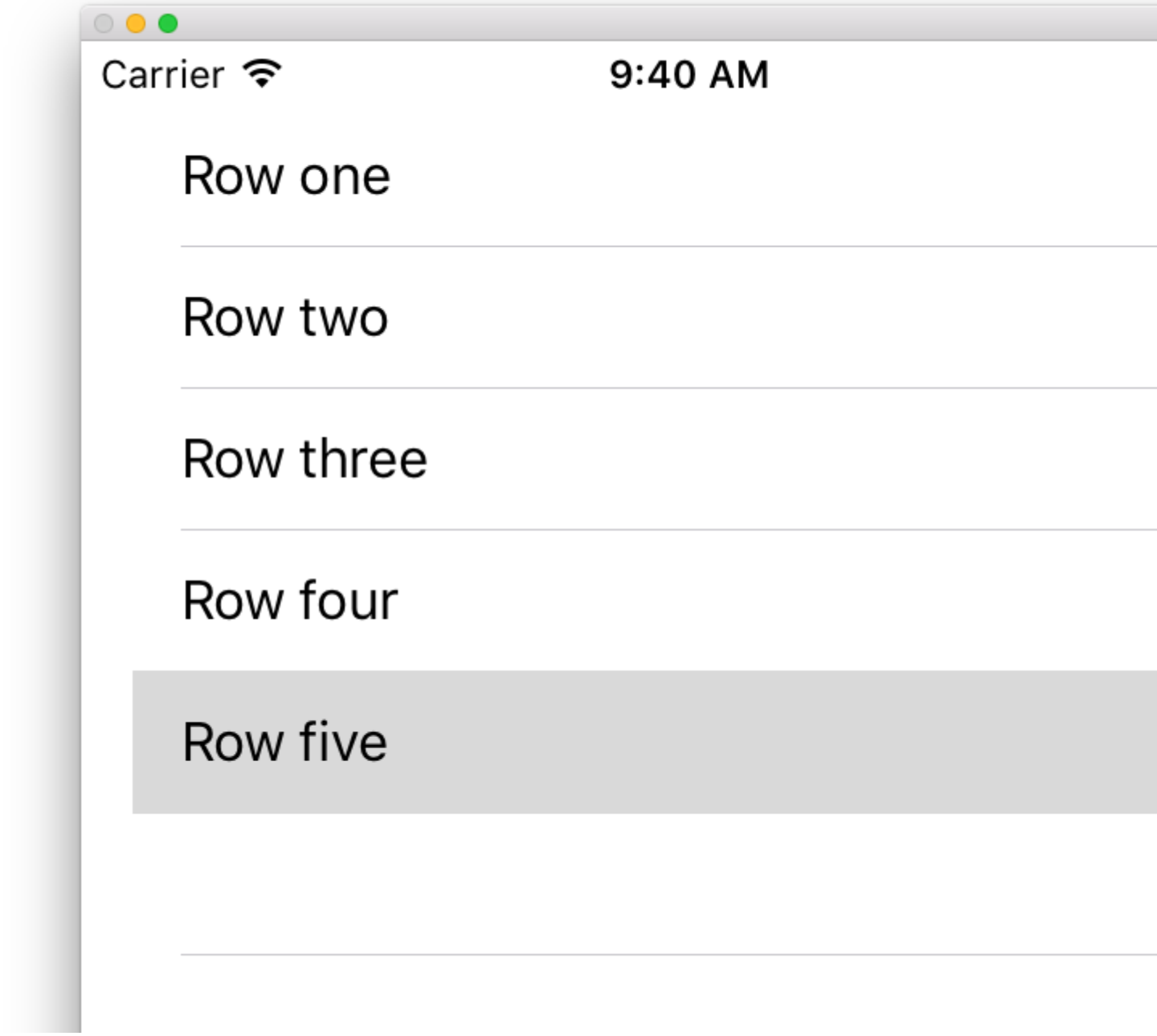
Objective-C

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
  switch (indexPath.section) {
```

```
      case 1:
        return 60;
      default:
        return UITableViewAutomaticDimension;
    }
}
```

Creare un UITableView

Una vista tabella è un elenco di righe che è possibile selezionare. Ogni riga viene popolata da un'origine dati. Questo esempio crea una vista tabella semplice in cui ogni riga è una singola riga di testo.



Aggiungi un UITableView allo storyboard

Sebbene ci siano molti modi per creare un `UITableView` , uno dei più semplici è quello di aggiungerne uno a uno Storyboard. Apri lo storyboard e trascina un `UITableView` sul tuo `UIViewController` . Assicurati di utilizzare il layout automatico per allineare correttamente il tavolo (spilli su tutti e quattro i lati).

Compilare la tabella con i dati

Per visualizzare il contenuto in modo dinamico (cioè caricarlo da un'origine dati come un array, un modello Core Data, un server in rete, ecc.) Nella visualizzazione tabella è necessario impostare l'origine dati.

Creare una semplice fonte di dati

Una fonte di dati potrebbe, come detto sopra, essere qualsiasi cosa con i dati. Dipende esclusivamente da te come formattare e che cosa è in esso. L'unico requisito è che devi essere in grado di leggerlo in un secondo momento in modo da poter popolare ogni riga della tabella con i dati quando necessario.

In questo esempio, imposteremo un array con alcune stringhe (testo) come origine dati:

veloce

```
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"];
```

Impostazione dell'origine dati nel View Controller

Assicurati che il tuo controller di visualizzazione sia conforme al protocollo `UITableViewDataSource` .

veloce

```
class ViewController: UIViewController, UITableViewDataSource {
```

Objective-C

```
@interface ViewController : UIViewController <UITableViewDataSource>
```

Non appena il controller di visualizzazione ha dichiarato che sarà **conforme** a `UITableViewDataSource` (questo è ciò che abbiamo appena fatto sopra), è *necessario* implementare almeno i seguenti metodi nella classe del controller di visualizzazione:

- `tableView:numberOfRowsInSection` , ti chiede quante righe dovrebbe avere la tua vista tabella.

```
// Swift

func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
     return self.myDataArray.count
}
```

- `tableView:cellForRowAtIndexPath` , richiede di creare e restituire una cella per ciascuna riga specificata in `tableView:numberOfRowsInSection` . Quindi, se hai detto che avevi bisogno di 10 righe, questo metodo verrà chiamato dieci volte per ogni riga e dovrai creare una cella per ciascuna di queste righe.

```
// 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
 }
```

ATTENZIONE : **NON** puoi restituire nil per nessuna cella in `cellForRowAtIndexPath:` Ciò causerà l'arresto anomalo dell'app e verrà visualizzato il seguente errore nella console:

```
Uncaught exception 'NSInternalInconsistencyException', reason: 'UITableView
dataSource must return a cell from tableView:cellForRowAtIndexPath:'
```

Collegamento dell'origine dati della vista tabella al controller di visualizzazione

È possibile farlo tramite il codice impostando della vostra tabella `dataSource` proprietà per `self` sul controller della vista. Oppure puoi selezionare la visualizzazione tabella nello storyboard, aprire l'ispettore Attributi, selezionare il pannello "Outlets" e trascinare da `dataSource` al controller di visualizzazione (**NOTA** : assicurati di connetterti a UIViewCONTROLLER, **non a** una UIView o a un altro oggetto *nel* tuo UIViewController).

Gestione delle selezioni di riga

Quando un utente tocca una riga nella tua vista tabella, generalmente, vuoi fare qualcosa - per rispondere. In molte app, quando si tocca una riga, vengono visualizzate ulteriori informazioni sull'elemento selezionato. Pensa all'app Messaggi: quando tocchi la riga che mostra uno dei tuoi contatti, la conversazione con quella persona viene quindi visualizzata sullo schermo.

In or per fare ciò, è necessario conformarsi al protocollo `UITableViewDelegate` . Ciò è simile alla conformità al protocollo dell'origine dati. Questa volta tuttavia, lo aggiungerai accanto a `UITableViewDataSource` e lo separerai con una virgola. Quindi dovrebbe assomigliare a questo:

veloce

```
class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {
```

Objective-C

```
@interface ViewController : UIViewController <UITableViewDataSource, UITableViewDelegate>
```

Non sono richiesti metodi da implementare per il delegato della vista tabella. Tuttavia, per gestire le selezioni di riga è necessario utilizzare il seguente metodo:

- `tableView:didSelectRowAtIndexPath` , questo viene chiamato ogni volta che viene `tableView:didSelectRowAtIndexPath` una riga, che ti permette di fare qualcosa in risposta. Per il nostro esempio, stamperemo semplicemente una dichiarazione di conferma sul registro 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);
}
```

La soluzione finale

Vedi sotto per la configurazione completa con solo codice, nessuna spiegazione.

veloce

```
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
```

Delegato e origine dati

`UITableViewDelegate` viene utilizzato per controllare come viene visualizzata la tabella e `UITableViewDataSource` viene utilizzato per definire i dati di `UITableView` . Esistono due metodi obbligatori e molti facoltativi che possono essere utilizzati per personalizzare dimensioni, sezioni, intestazioni e celle in `UITableView` .

UITableViewDataSource

Metodi richiesti

`numberOfRowsInSection:` questo metodo definisce il numero di celle che verranno visualizzate in ciascuna sezione di Tableview.

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:` questo metodo è dove vengono create e configurate le celle di `UITableView` . Dovrebbe restituire un `UITableViewCell` o una sottoclasse personalizzata.

Nota: L'uso `dequeueReusableCellWithIdentifier:forIndexPath:` richiede che la classe o il pennino è stata registrata per tale identificatore utilizzando `UITableView` s' `registerClass:forCellReuseIdentifier:` o `registerNib:forCellReuseIdentifier:` metodi. Di solito, questo sarà fatto nel `UIViewController` s' `viewDidLoad` metodo.

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
}
```

Metodi opzionali

`titleForHeaderInSection:` definisce una stringa come titolo per ogni intestazione di sezione nella vista tabella. Questo metodo consente solo di modificare il titolo, è

possibile personalizzare ulteriormente definendo la vista per l'intestazione.

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:` definisce una stringa come titolo per ogni intestazione di sezione nella vista tabella.

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:` utilizzato per determinare se l'interfaccia utente di modifica debba essere visualizzata per la riga specificata. Dovrebbe restituire `YES` se la riga specificata può essere cancellata o aggiunta.

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` Dovrebbe eseguire il lavoro richiesto per gestire l'aggiunta o la rimozione della riga specificata. Ad esempio, rimuovere la cella da `UITableView` con l'animazione e rimuovere l'oggetto associato dal modello di dati della tabella.

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` Consente di aggiungere azioni o pulsanti aggiuntivi alla modalità di modifica di una riga all'interno di `UITableview` . Ad esempio, se si desideravano due pulsanti, un pulsante di modifica e di eliminazione quando l'utente esegue lo scorrimento per modificare la riga, si utilizzerà questo metodo.

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

Tutti i metodi in `UITableViewDelegate` sono facoltativi, ma un delegato che li implementa abiliterà funzionalità extra per `UITableView` .

`numberOfSectionsInTableView:` per impostazione predefinita restituisce 1, ma il supporto per più sezioni è abilitato restituendo un numero diverso di sezioni.

Objective-C

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return self.numSections;
}
```

Swift 3

```
func numberOfSectionsInTableView(_ tableView: UITableView) -> Int {
    return self.numSections
}
```

`viewForHeaderInSection` Consente la configurazione di una vista personalizzata come intestazione per la sezione.

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:` definisce l'altezza di ogni cella nella vista tabella.

Objective-C

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    return 44;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, heightForRowAtIndexPath indexPath: NSIndexPath) ->
CGFloat {
    return 44
}
```

`heightForHeaderInSection:` and `heightForFooterInSection` Definire l'altezza per l'intestazione e il piè di pagina di ogni sezione nella vista tabella

Objective-C

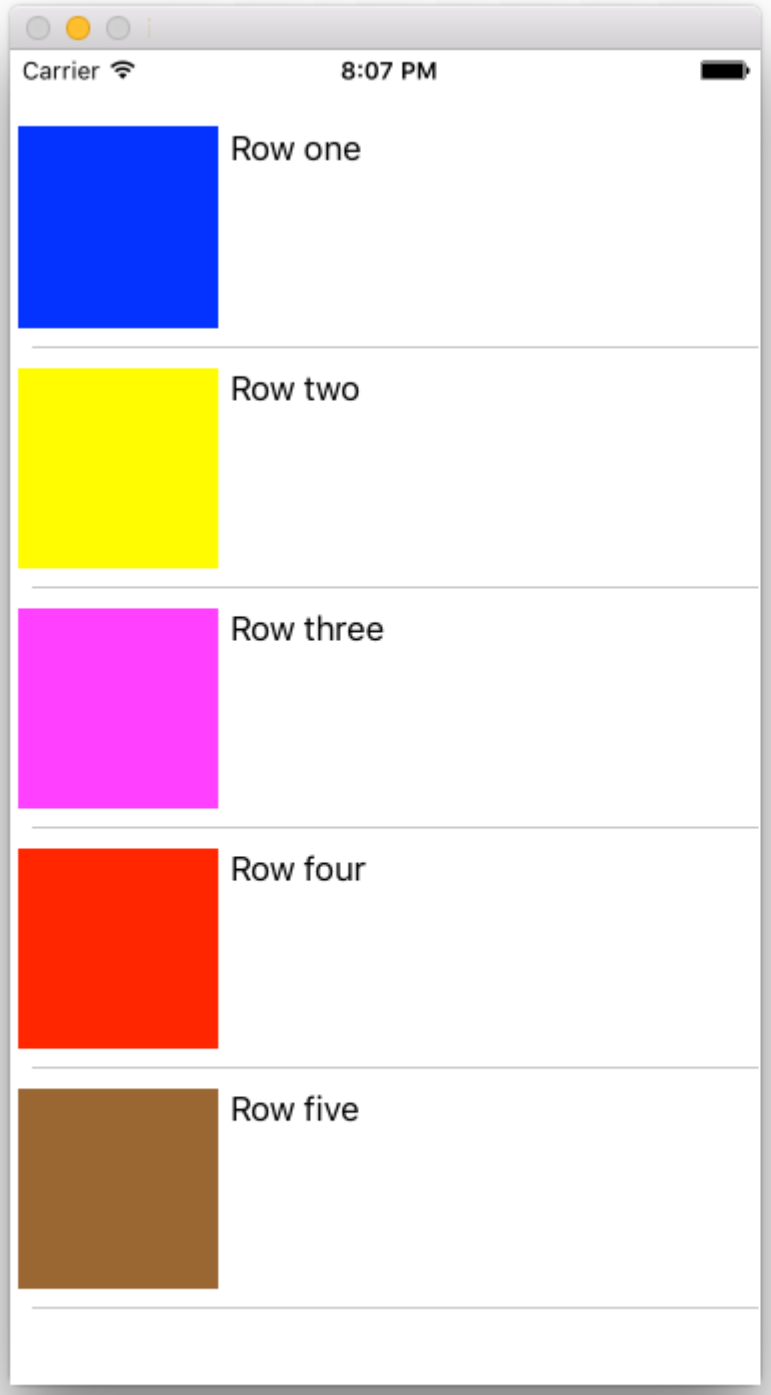
```
- (CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section {
    return 33;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
    return 33
}
```

Celle personalizzate

La personalizzazione di `UITableViewCell` può consentire interfacce molto potenti, dinamiche e reattive. Con un'ampia personalizzazione e in combinazione con altre tecniche puoi fare cose come: aggiornare proprietà specifiche o elementi dell'interfaccia mentre cambiano, animano o disegnano oggetti nella cella, caricano in modo efficiente i video mentre l'utente scorre, o persino visualizzano le immagini mentre scaricano da un Rete. Le possibilità qui sono quasi infinite. Di seguito è riportato un semplice esempio di come può apparire una cella personalizzata.



Questa sezione illustra le nozioni di base e, si spera, verrà ampliata per descrivere i processi più complessi come quelli descritti sopra.

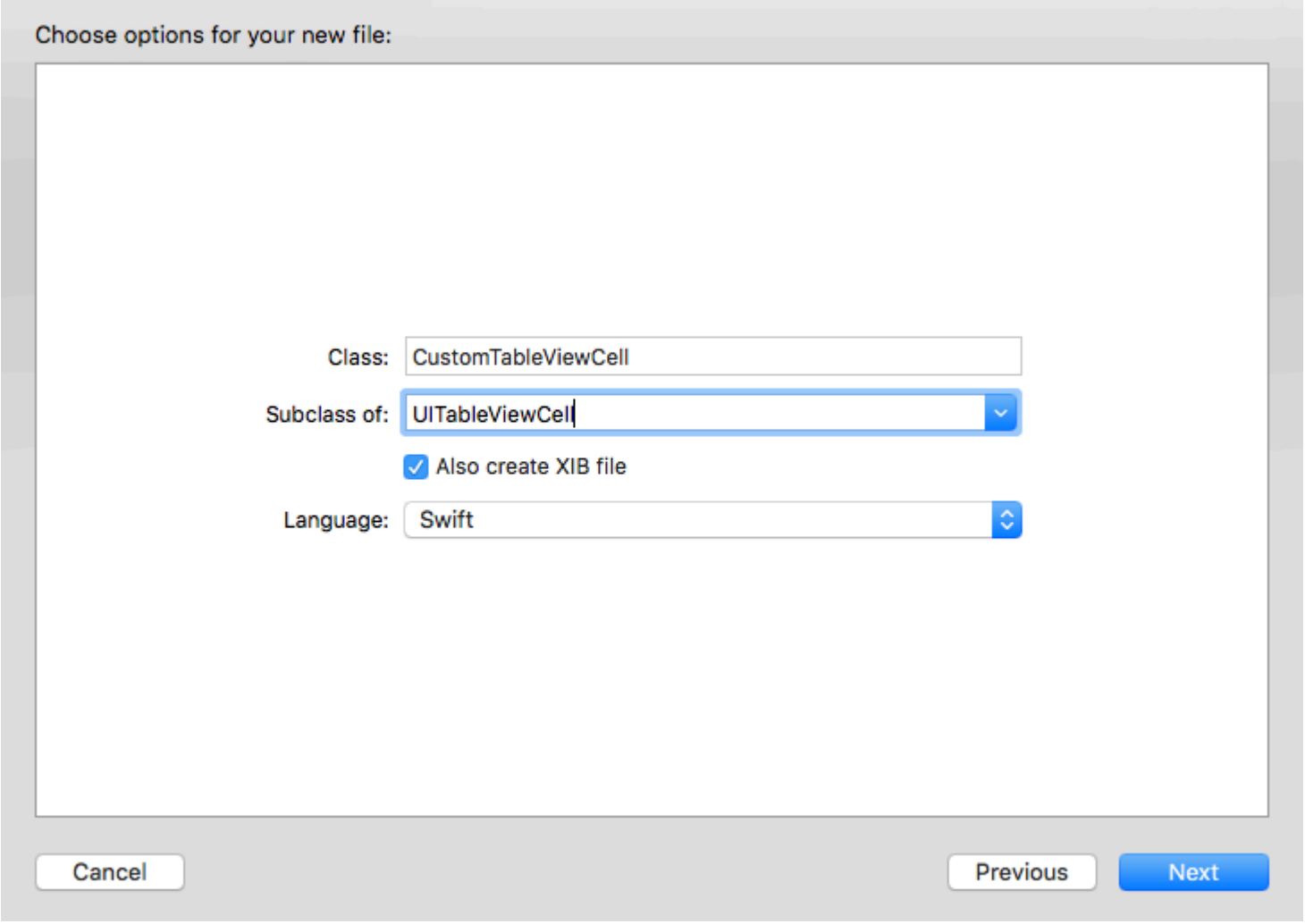
Creazione della tua cella personalizzata

Innanzitutto, crea una nuova sottoclasse di `UITableViewCell` (crea una nuova Cocoa Touch Class in Xcode e imposta `UITableViewCell` come superclasse). Di seguito è riportato l'aspetto del codice dopo la sottoclasse.

veloce

```
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)
    }
}
```

Facoltativamente, selezionare "Crea anche un file XIB" quando si crea il nuovo file da personalizzare utilizzando Interface Builder. Nel caso in cui lo fai, connetti `customLabel` come `@IBOutlet`



In un `UIViewController` contenente `tableView` , registra la nuova classe della cella personalizzata (vedi sotto). *Nota, questo è necessario solo se non si progetta la cella con uno Storyboard nell'interfaccia della vista tabella.*

veloce

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

    // Register Cell Class
    tableView.register(CustomTableViewCell.self, forCellReuseIdentifier:
CustomTableViewCell.identifier)
}
```

Se si sceglie di utilizzare un file XIB, `registerNib` invece:

veloce

```
// Register Nib
tableView.register(UINib(nibName: CustomTableViewCell.identifier, bundle: nil),
forCellReuseIdentifier: CustomTableViewCell.identifier)
```

Ora che il tuo `tableView` conosce la tua cella personalizzata, puoi `cellForRowAtIndexPath` in `cellForRowAtIndexPath` :

veloce

```
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
}
```

Espansione e compressione di UITableViewCells

Nella tua Storyboard, aggiungi un oggetto UITableView sul tuo UIViewController e lascia che copra l'intera vista. Imposta le connessioni `UITableviewDataSource` e `UITableviewDelegate` .

Objective-C

Nel tuo file `.h`

```
NSMutableArray *arrayForBool;
NSMutableArray *sectionTitleArray;
```

Nel tuo file `.m`

```
- (void)viewDidLoad  {
```

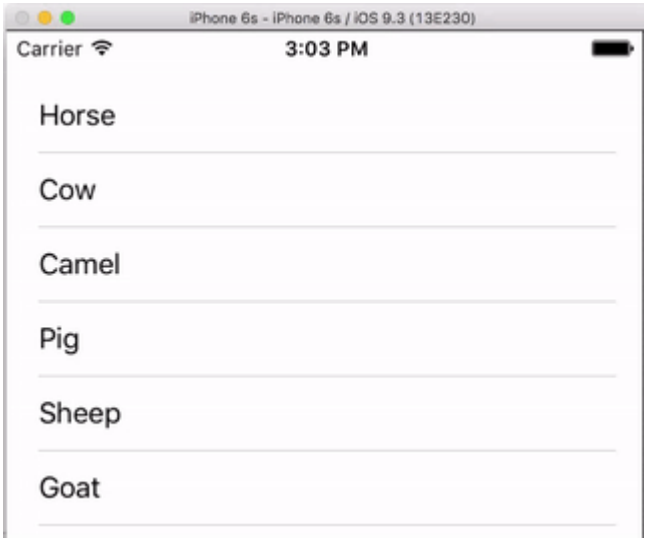
```
    [super viewDidLoad];

    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];

   }
 }
```

Scorri per eliminare le righe

Penso sempre che sia bello avere un esempio molto semplice e autonomo in modo che nulla venga assunto quando sto imparando un nuovo compito. Questa risposta è quella per l'eliminazione di righe `UITableView` . Il progetto si comporta in questo modo:



Questo progetto è basato sull'esempio UITableView per Swift .

Aggiungi il codice

Crea un nuovo progetto e sostituisci il codice ViewController.swift con quanto segue.

```
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.
        }
    }

}
```

Il metodo a chiave singola nel codice sopra che consente l'eliminazione delle righe è l'ultimo. Eccolo di nuovo per enfasi:

```
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)
    }
```

```
}
```

storyboard

Aggiungi `UITableView` al controller di visualizzazione nello storyboard. Utilizzare il layout automatico per fissare i quattro lati della vista tabella ai bordi del controller di visualizzazione. Controllo del trascinamento dalla visualizzazione tabella nello storyboard alla `@IBOutlet var tableView: UITableView!` linea nel codice.

Finito

È tutto. Dovresti essere in grado di eseguire l'app ora ed eliminare le righe scorrendo verso sinistra e toccando "Elimina".

Gli appunti

- Questo è disponibile solo da iOS 8. Vedi questa risposta per maggiori dettagli.
- Se è necessario modificare il numero di pulsanti visualizzati o il testo del pulsante, consultare questa risposta per ulteriori dettagli.

Ulteriori letture

- Come creare una cella di visualizzazione tabella a scorrimento con azioni - Senza andare con i dadi con le visualizzazioni di scorrimento
- Documentazione Apple

Linee di separazione

Modifica della larghezza delle linee di separazione

Puoi impostare che le linee di separazione della vista tabella si estendano su varie larghezze della tabella cambiando il `layoutMargins:` proprietà sulle celle. Questo può essere raggiunto in diversi modi.

Modifica delle linee di separazione per celle specifiche

Nella tabella visualizza il valore `cellForRowAtIndexPath:` metodo *o* `willDisplayCell:` metodo, imposta il `layoutMargins:` della cella `layoutMargins:` proprietà su `UIEdgeInsetsZero` (si estende a tutta

la larghezza della tabella) o su qualsiasi cosa tu desideri qui.

Objective-C

```
[cell setLayoutMargins:UIEdgeInsetsZero];

// May also use separatorInset
[cell setSeparatorInset:UIEdgeInsetsZero];
```

veloce

```
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
}
```

Rimuovi tutte le linee di separazione

Le sottili linee grigie tra ogni cella potrebbero non essere esattamente l'aspetto che stai cercando. È abbastanza semplice nasconderli dalla vista.

Nel tuo `UIViewController` di `UIViewController` comprende `viewDidLoad:` metodo aggiungi il seguente codice. È inoltre possibile impostare questa proprietà in qualsiasi momento prima di caricare o ricaricare la vista tabella (non necessariamente deve essere nel metodo `viewDidLoad:` .

Swift:

```
 tableView.separatorStyle = .None
```

Objective-C:

```
tableView.separatorStyle = UITableViewCellSeparatorStyleNone;
```

In alternativa, la proprietà può essere modificata nello Storyboard o XIB selezionando la tabella View e l'impostazione del `separator` (sotto l'ispettore attributi) su `None` .

Nascondi le linee di separazione in eccesso

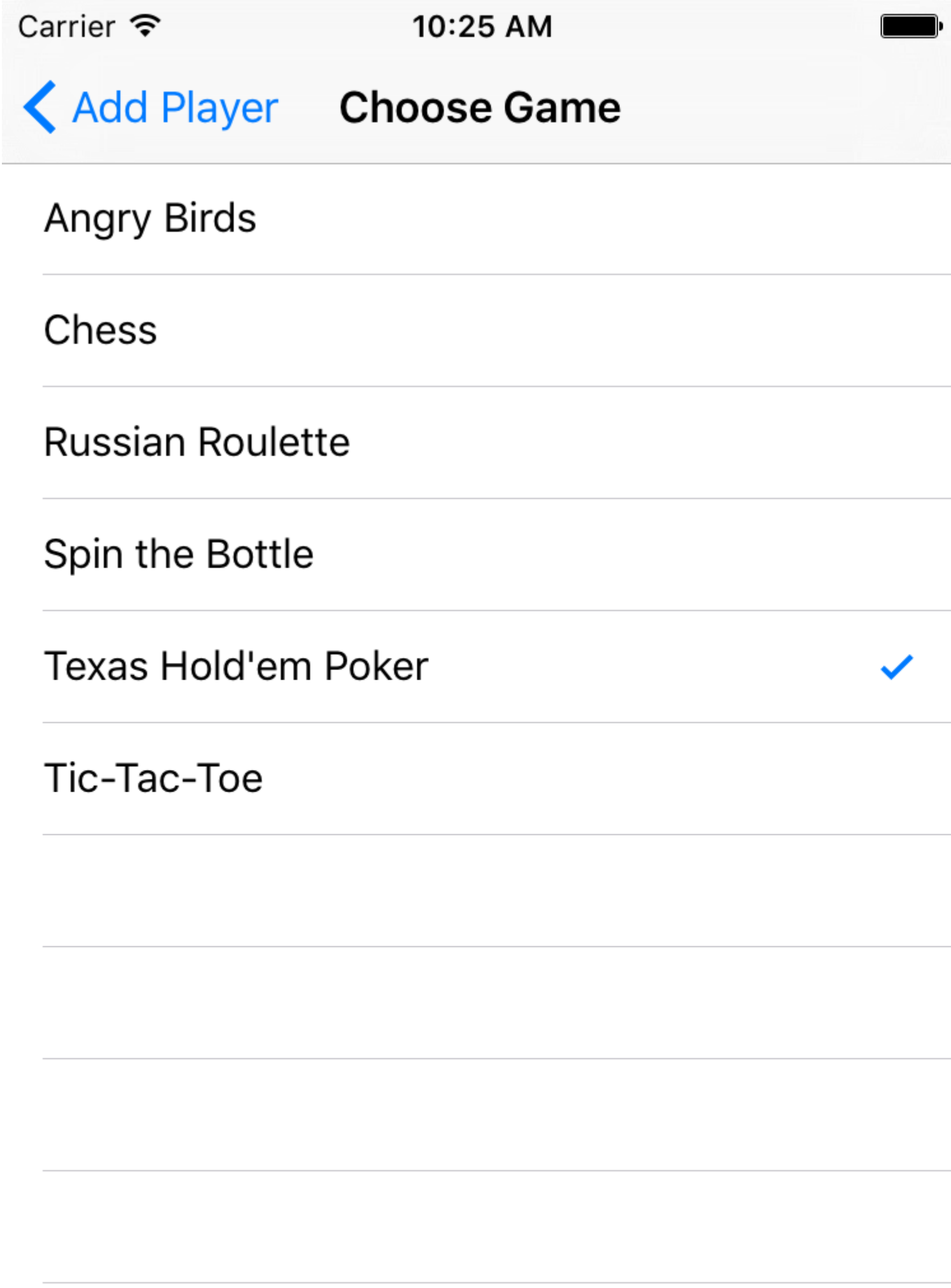
Puoi nascondere le linee del separatore di `UITableViewCell` per le celle vuote impostando una vista vuota del piè di pagina nella parte inferiore di UITableView:

veloce

```
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

Capitolo 195: UITableViewCell

introduzione

Carica il file xib personalizzato della cella utilizza la classe di categoria della cella, non è necessario registrare il file del pennino

Examples

File Xib di UITableViewCell

Creare una classe di categoria di celle `UITableView` .

UITableViewCell + RRCell.h file

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

@interface UITableViewCell (RRCell)

-(id)initWithOwner:(id)owner;

@end
```

UITableViewCell + RRCell.m file

```
#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
```

Importa classe categoria di celle per utilizzare questo metodo nel metodo `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;
}
```

Leggi UITableViewCell online: https://riptutorial.com/it/ios/topic/10101/uitableviewcell

Capitolo 196: UITableViewController

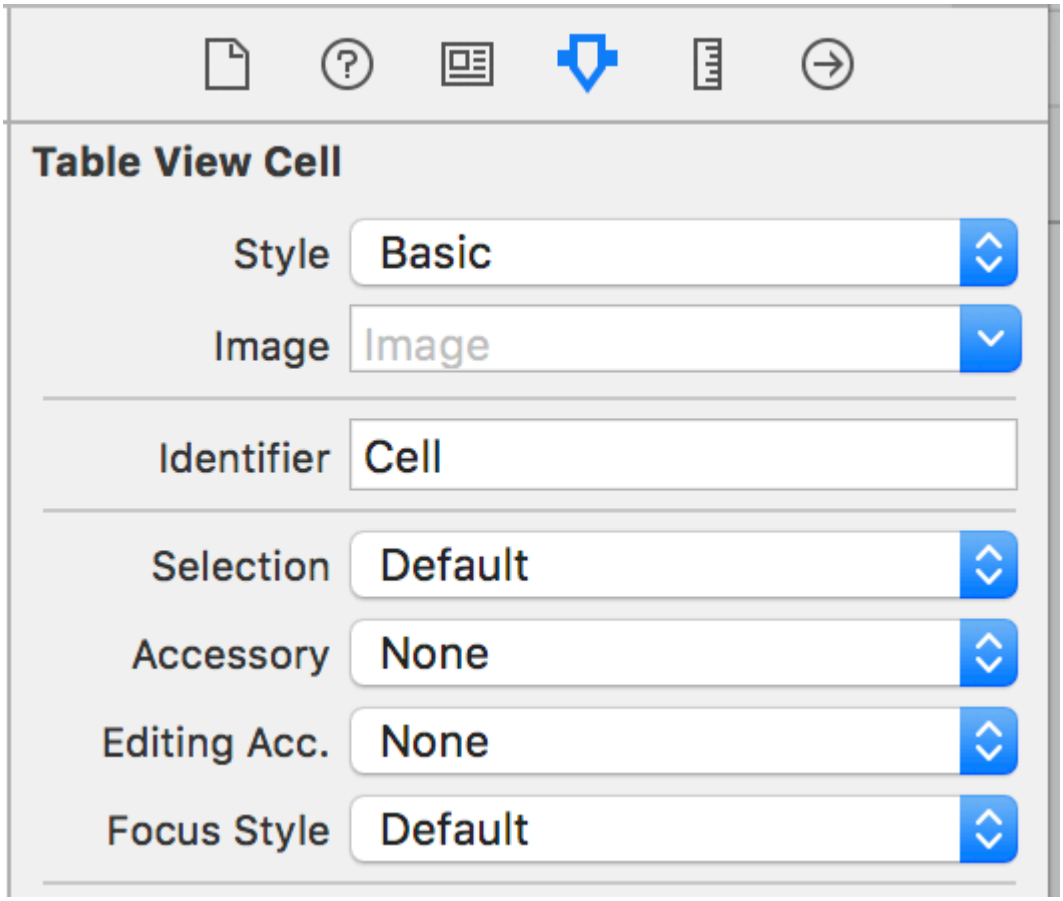
introduzione

Oggetto controller UITableViewController che gestisce una vista tabella. Per alcuni scenari specifici si consiglia di utilizzare UITableViewController, ad esempio se si hanno molte celle e alcuni hanno UITextfield.

Examples

TableView con proprietà dinamiche con 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 con cella personalizzata

Per la cella tableview personalizzata hai bisogno di una classe che sia sottoclasse da `UITableViewCell` , una classe di esempio che puoi vedere di seguito.

```
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
}

}
```

I tuoi delegati di tableview

```
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
}
```

Leggi UITableViewController online: https://riptutorial.com/it/ios/topic/10953/uitableviewcontroller

Capitolo 197: UITextField

introduzione

UITextField fa parte del framework UIKit e viene utilizzato per visualizzare un'area per raccogliere l'input di testo dall'utente utilizzando la tastiera su schermo

Sintassi

- UITextField.text: String // ottiene o imposta il testo visualizzato nel campo.
- UITextField.attributedText: NSAttributedString // ottiene o imposta il testo attribuito visualizzato nel campo.
- UITextField.textColor: UIColor // ottiene o imposta il colore del testo sul campo
- UITextField.font: UIFont // ottiene o imposta il carattere del testo sul campo
- UITextField.textAlignment: NSTextAlignment // default è NSLeftTextAlignment
- UITextField.borderStyle: UITextBorderStyle // default è UITextBorderStyleNone. Se impostato su UITextBorderStyleRoundedRect, le immagini di sfondo personalizzate vengono ignorate.
- UITextField.placeholder: String // default è nil. la stringa è disegnata al 70% in grigio
- UITextField.attributedPlaceholder: NSAttributedString // ottiene o imposta il segnaposto attribuito del campo
- UITextField.clearsOnBeginEditing: Bool // default è NO che sposta il cursore sulla posizione su cui si fa clic. se SÌ, tutto il testo è stato cancellato
- UITextField.adjustsFontSizeToFitWidth: Bool // default è NO. se SÌ, il testo si ridurrà a minFontSize lungo la linea di base
- UITextField.minimumFontSize: CGFloat // valore predefinito è 0.0. Minimo effettivo può essere bloccato a qualcosa di leggibile. usato se adjustsFontSizeToFitWidth è YES
- UITextField.delegate: UITextFieldDelegate? // il valore predefinito è nullo. riferimento debole
- UITextField.clearButtonMode: UITextFieldViewMode // imposta quando viene visualizzato il pulsante di cancellazione. l'impostazione predefinita è UITextFieldViewModeNever
- UITextField.leftView: UIView? // ad es. lente d'ingrandimento
- UITextField.leftViewMode: UITextFieldViewMode // imposta quando viene visualizzata la vista sinistra. l'impostazione predefinita è UITextFieldViewModeNever
- UITextField.rightView: UIView? // ad es. pulsante dei segnalibri
- UITextField.rightViewMode: UITextFieldViewMode // imposta quando viene visualizzata la vista destra. l'impostazione predefinita è UITextFieldViewModeNever
- UITextField.inputView: UIView? // Presentato quando l'oggetto diventa il primo soccorritore. Se impostato su nil, ritorna alla seguente catena di responder. Se impostato durante il primo intervento, non avrà effetto fino a quando non viene chiamato reloadInputViews.
- UITextField.inputAccessoryView: UIView?
- UITextField.isSecureTextEntry: Bool // es. Se il campo contiene input riservati come password o numero di carta

Examples

Inizializza campo di testo

veloce

```
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];
```

Interface Builder

È inoltre possibile aggiungere un `UITextField` a uno storyboard trascinandolo dalla libreria degli oggetti.

Visualizzazione degli accessori di input (barra degli strumenti)

Aggiungi una vista accessoria sopra la tastiera. Questo è comunemente usato per aggiungere i pulsanti next / previous, o pulsanti aggiuntivi come Done / Submit (specialmente per i tipi di tastiera numero / telefono / decimale che non hanno una chiave di ritorno incorporata).

veloce

```
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;
```

Autocapitalizzazione

veloce

```
textField.autocapitalizationType = .None
```

Objective-C

```
textField.autocapitalizationType = UITextAutocapitalizationTypeNone;
```

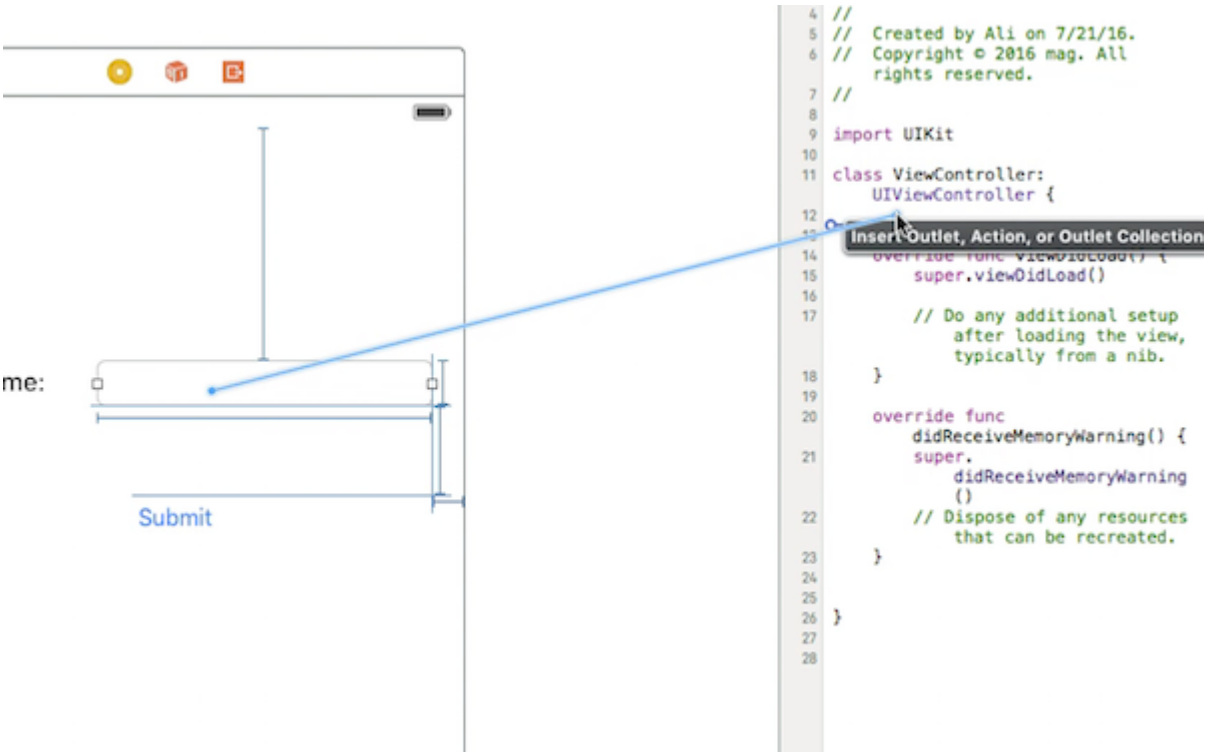
Tutte le opzioni:

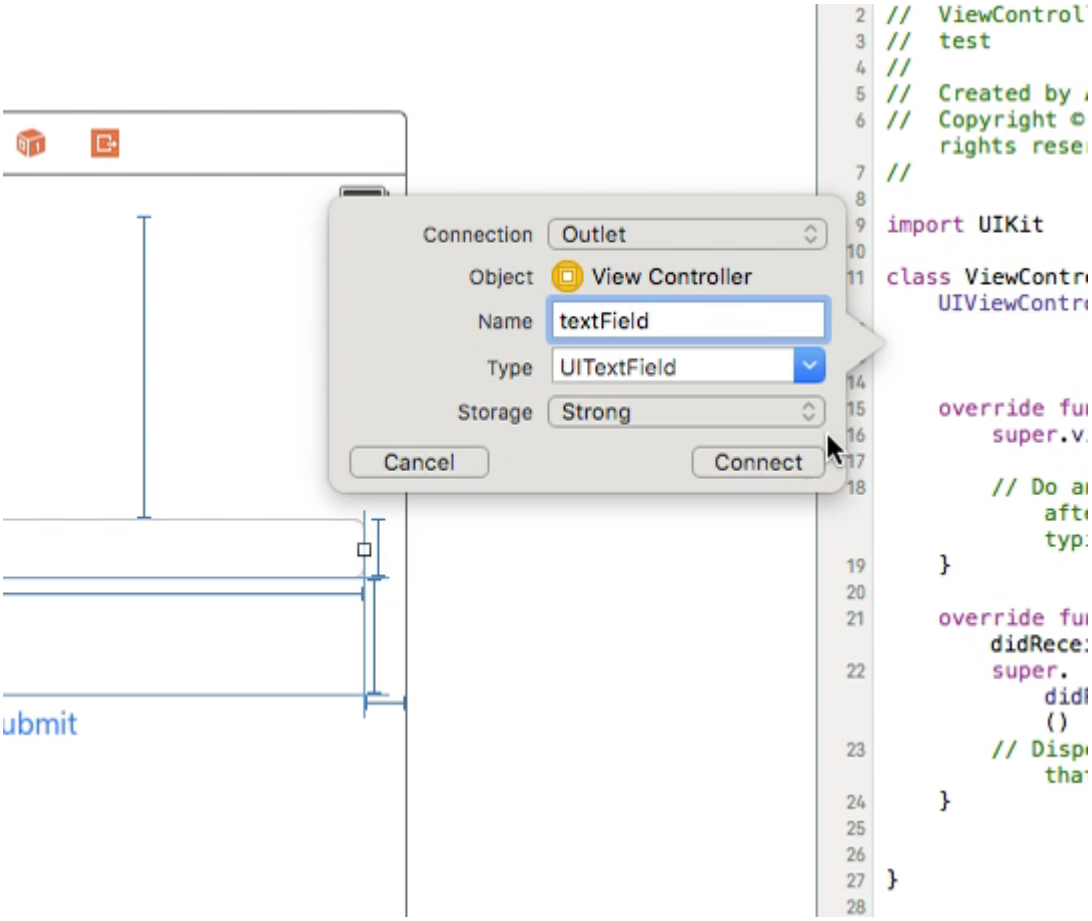
- `.None` \ `UITextAutocapitalizationTypeNone` : non autorizzare automaticamente nulla
- `.Words` \ `UITextAutocapitalizationTypeWords` : Autocapitalizza ogni parola
- `.Sentences` \ `UITextAutocapitalizationTypeSentences` : Autocapitalizza la prima parola in una frase
- `.AllCharacters` \ `UITextAutocapitalizationTypeAllCharacters` : Autocapitalize ogni lettera (cioè blocco maiuscole)

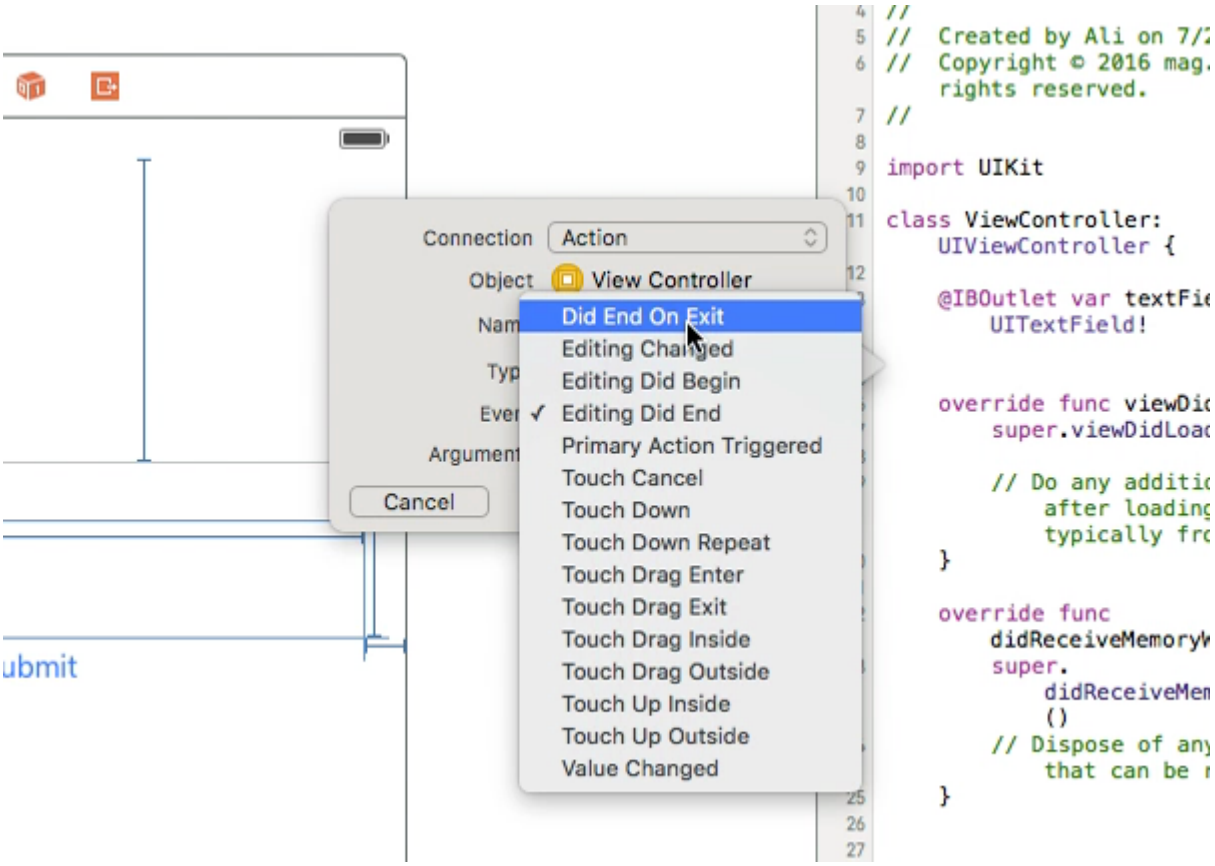
Ignora tastiera

veloce

Ctrl + Trascina da UItextfield in MainStoryboard alla classe ViewController e crea un UITextField Outlet


me:
Submit
Insert Outlet, Action, or Outlet Collection
// Created by Ali on 7/21/16.
// Copyright © 2016 mag. All rights reserved.
//
import UIKit
class ViewController: UIViewController {
super.viewDidLoad()
// Do any additional setup after loading the view, typically from a nib.
override func didReceiveMemoryWarning() {
super.didReceiveMemoryWarning()
// Dispose of any resources that can be recreated.


Connection Outlet
Object View Controller
Name textField
Type UITextField
Storage Strong
Cancel
Connect
ubmit
import UIKit



Successivamente selezionare nuovamente UItextField e Ctrl + trascinare nella classe ViewController ma questa volta selezionare Connessione **azione** e in memoria selezionare **Esegui su Esci,** quindi fare clic su Connetti.

Nell'azione appena creata scrivi il nome del tuo UItextField `.resignFirstResponder()`

```
    @IBAction func textFieldResign(sender: AnyObject) {
         yourTextFieldName.resignFirstResponder()
     }
```

Questo si prenderà cura di nascondere la tastiera quando si preme il tasto Invio sulla tastiera.

Un altro esempio di nascondere la tastiera quando viene premuto il tasto Invio:

aggiungiamo il protocollo `UITextFieldDelegate` accanto a `UIViewController`

nella funzione `self.yourTextFieldName.delegate = self` aggiungiamo
`self.yourTextFieldName.delegate = self`

E infine aggiungiamo questo

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
                yourTextFieldName.resignFirstResponder()
                return true
            }
```

Il codice finale è questo:

```
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];
```

Imposta allineamento

veloce

```
textField.textAlignment = .Center
```

Objective-C

```
[textField setTextAlignment: NSTextAlignmentCenter];
```

Nell'esempio, abbiamo impostato `NSTextAlignment` al centro. È anche possibile impostare a `.Left` , `.Right` , `.Justified` e `.Natural` .

`.Natural` è l'allineamento predefinito per la localizzazione corrente. Ciò significa che per le lingue da sinistra a destra (ad esempio l'inglese), l'allineamento è `.Left` ; per le lingue da destra a sinistra, è `.Right` .

KEYBOARDTYPE

Per cambiare l'aspetto della tastiera, i seguenti tipi possono essere impostati individualmente su ogni proprietà di `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
};
```

Spostamento dello scroll quando UITextView diventa il primo soccorritore

Osservare le notifiche `UIKeyboardWillShowNotification` e `UIKeyboardWillHideNotification` , aggiornare gli `scrollView` contenuto `scrollView` base all'altezza della tastiera, quindi scorrere fino al controllo focalizzato.

```
- (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];
}
```

Ottieni la messa a fuoco della tastiera e nascondi la tastiera

Metti a fuoco

veloce

```
textField.becomeFirstResponder()
```

Objective-C

```
[textField becomeFirstResponder];
```

Dimettersi

veloce

```
textField.resignFirstResponder()
```

Objective-C

```
[textField resignFirstResponder];
```

Sostituisci tastiera con UIPickerView

In alcuni casi, si desidera mostrare agli utenti un `UIPickerView` con contenuti predefiniti per un `UITextField` anziché una tastiera.

Crea un UIPickerView personalizzato

In un primo momento, è necessaria una classe wrapper personalizzata per `UIPickerView` conforme ai protocolli `UIPickerViewDataSource` e `UIPickerViewDelegate` .

```
class MyPickerView: UIPickerView, UIPickerViewDataSource, UIPickerViewDelegate
```

È necessario implementare i seguenti metodi per DataSource e 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 ""
    }
}
```

Per gestire i dati, `MyPickerView` richiede i `data` proprietà, `selectedValue` e `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 ""
        }
    }
}
```

Prepara ViewController

Il `ViewController` che contiene il tuo textField, deve avere una proprietà per il tuo `UIPickerView` personalizzato. (Supponendo che tu abbia già un'altra proprietà o `@IBOutlet` contiene il tuo campo di testo)

```
/**
 The picker view to present as keyboard
 */
var picker: MyPickerView?
```

Nel tuo `viewDidLoad()` , devi inizializzare il `picker` e configurarlo un po ':

```
picker = MyPickerView()
picker?.autoresizingMask = [.flexibleHeight, .flexibleWidth]
picker?.backgroundColor = UIColor.white()

picker?.data = ["One", "Two", "Three", "Four", "Five"] //The data shown in the picker
```

Ora puoi aggiungere `MyPicker` come `inputView` del tuo `UITextField` :

```
textField.inputView = picker
```

Eliminare la tastiera di selezione

Ora, hai sostituito la tastiera con un `UIPickerView` , ma non c'è possibilità di chiuderla. Questo può essere fatto con un custom `.inputAccessoryView` :

Aggiungere la proprietà `pickerAccessory` al `ViewController` .

```
/**
 A toolbar to add to the keyboard when the `picker` is presented.
```

```
 */
var pickerAccessory: UIToolbar?
```

In `viewDidLoad()` , devi creare una `UIToolbar` per `inputAccessoryView` :

```
pickerAccessory = UIToolbar()
pickerAccessory?.autoresizingMask = .flexibleHeight

//this customization is optional
pickerAccessory?.barStyle = .default
pickerAccessory?.barTintColor = UIColor.red()
pickerAccessory?.backgroundColor = UIColor.red()
pickerAccessory?.isTranslucent = false
```

È necessario impostare la cornice della barra degli strumenti. Per adattarsi al design di iOS, si consiglia di utilizzare un'altezza di `44.0` :

```
var frame = pickerAccessory?.frame
frame?.size.height = 44.0
pickerAccessory?.frame = frame!
```

Per una buona esperienza utente, dovresti aggiungere due pulsanti ("Fatto" e "Annulla"), ma funzionerà anche con uno solo che disattiva la tastiera.

```
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]
```

Ora puoi aggiungere la barra degli strumenti come `inputAccessoryView`

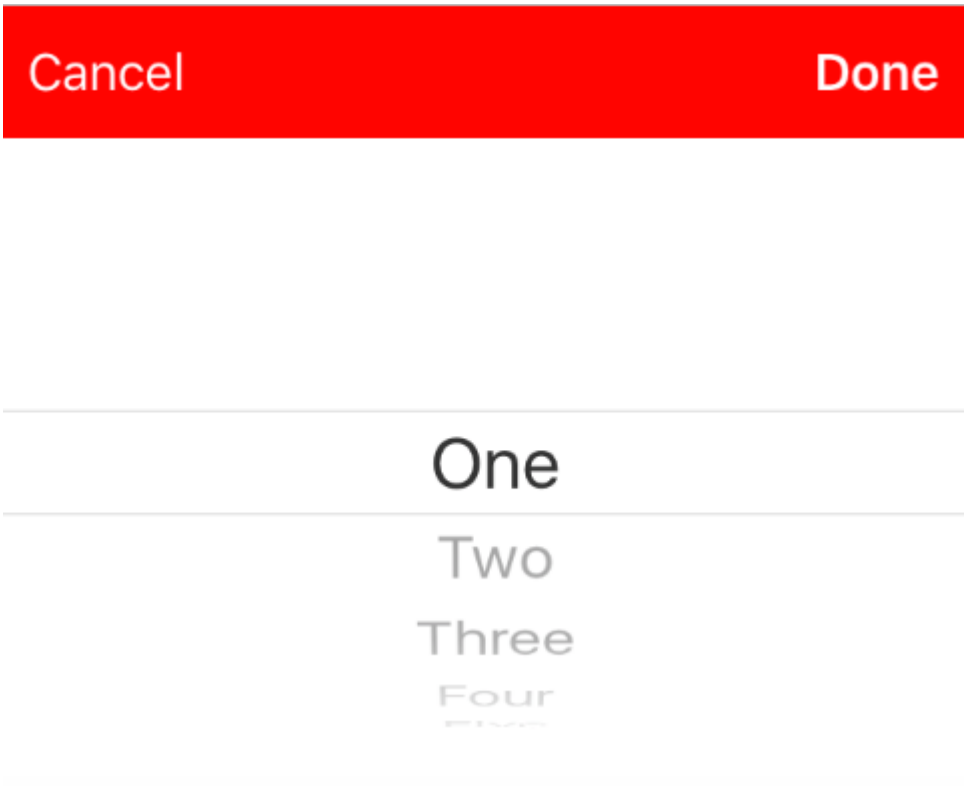
```
textField.inputAccessoryView = pickerAccessory
```

Prima di poter costruire il tuo progetto, devi implementare i metodi, i pulsanti chiamano:

```
/**
 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
}
```

Eseguire il progetto, toccare il `textField` e si dovrebbe vedere un selettore come questo al posto della tastiera:



Seleziona un valore a livello di codice (facoltativo)

Se non si desidera che la prima riga venga selezionata automaticamente, è possibile impostare la riga selezionata come in `UIPickerView` :

```
picker?.selectRow(3, inComponent: 0, animated: false) //Will select the row at index 3
```

Ignora la tastiera quando l'utente preme il pulsante di ritorno

Imposta il tuo controller di visualizzazione per gestire la modifica del testo per il campo di testo.

```
class MyViewController: UITextFieldDelegate {

    override viewDidLoad() {
        super.viewDidLoad()

        textField.delegate = self
    }

}
```

`textFieldShouldReturn` viene chiamato ogni volta che viene premuto il pulsante di ritorno sulla tastiera.

Swift:

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
    textField.resignFirstResponder()
    return true;
}
```

Objective-C:

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField {
   [textField resignFirstResponder];
   return true;
}
```

Ottenere e impostare la posizione del cursore

Informazioni utili

L'inizio del testo del campo di testo:

```
let startPosition: UITextPosition = textField.beginningOfDocument
```

La fine del testo del campo di testo:

```
let endPosition: UITextPosition = textField.endOfDocument
```

La gamma attualmente selezionata:

```
let selectedRange: UITextRange? = textField.selectedTextRange
```

Ottieni la posizione del cursore

```
if let selectedRange = textField.selectedTextRange {

    let cursorPosition = textField.offsetFromPosition(textField.beginningOfDocument,
toPosition: selectedRange.start)

    print("\(cursorPosition)")
}
```

Imposta la posizione del cursore

Per impostare la posizione, tutti questi metodi stanno effettivamente impostando un intervallo con gli stessi valori di inizio e fine.

All'inizio

```
let newPosition = textField.beginningOfDocument
textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

All'estremità

```
let newPosition = textField.endOfDocument
textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

Ad una posizione a sinistra della posizione corrente del cursore

```
// 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)
    }
}
```

Ad una posizione arbitraria

Inizia dall'inizio e sposta 5 caratteri a destra.

```
let arbitraryValue: Int = 5
if let newPosition = textField.positionFromPosition(textField.beginningOfDocument,
inDirection: UITextLayoutDirection.Right, offset: arbitraryValue) {

    textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
}
```

Relazionato

Seleziona tutto il testo

```
textField.selectedTextRange = textField.textRangeFromPosition(textField.beginningOfDocument,
toPosition: textField.endOfDocument)
```

Seleziona un intervallo di testo

```
// 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!)
}
```

Inserisci il testo nella posizione corrente del cursore

```
textField.insertText("Hello")
```

Gli appunti

- Questo esempio deriva originariamente da questa risposta di Overflow dello stack .
- Questa risposta utilizza un campo di testo, ma gli stessi concetti si applicano a `UITextView` .
- Usa `textField.becomeFirstResponder()` per focalizzare il campo di testo e far apparire la tastiera.
- Vedi questa risposta per come ottenere il testo a un certo intervallo.

Relazionato

- Come creare un intervallo in Swift (Si occupa in modo indiretto del problema del motivo per cui dobbiamo utilizzare `selectedTextRange` qui anziché solo `selectedRange`)

Nascondere il cursore lampeggiante

Per nascondere il cursore lampeggiante, è necessario sovrascrivere caretRectForPosition di un UITextField e restituire 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;
}
```

Cambia colore e carattere segnaposto

Possiamo cambiare lo stile del segnaposto impostando `attributedPlaceholder` (a `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
}
```

In questo esempio cambiamo solo il `color` e il `font` . È possibile modificare altre proprietà come lo stile sottolineato o barrato. Fare riferimento a `NSAttributedString` per le proprietà che possono essere modificate.

Crea un campo UIText

Inizializza `UITextField` con un CGRect come frame:

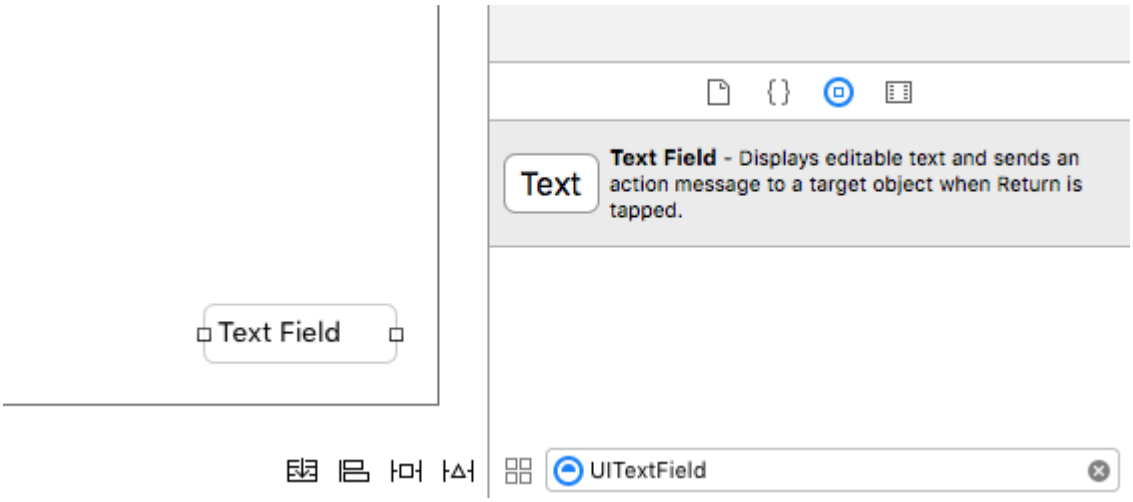
veloce

```
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)];
```

Puoi anche creare un `UITextField` in Interface Builder:



Leggi UITextField online: https://riptutorial.com/it/ios/topic/1630/uitextfield

Capitolo 198: UITextField delegato

Examples

UITextField - Limita il testo a determinati caratteri

Se si desidera eseguire una convalida dell'input dell'utente del proprio campo di testo, utilizzare il seguente frammento di codice:

```
// 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];
}
```

Inoltre, è possibile utilizzare i set di caratteri forniti da Apple per eseguire la convalida:

Dai un'occhiata a https://developer.apple.com/reference/foundation/nscharacterset

```
let allowedCharacters = CharacterSet.alphanumerics.inverted
```

```
let allowedCharacters = CharacterSet.capitalizedLetters.inverted
```

Trova tag successivo e gestisci tastiera

Il campo di testo chiama diversi metodi delegati (solo se i delegati sono impostati) Uno dei metodi delegati chiamati da textfield è * **- (BOOL) textFieldShouldReturn: (UITextField *) textField***

Questo metodo viene chiamato ogni volta che gli utenti toccano il pulsante di ritorno. Utilizzando questo metodo, possiamo implementare qualsiasi comportamento personalizzato.

Per esempio,

> Nell'esempio seguente, il prossimo risponditore verrà individuato sulla base del tag e gestirà la tastiera. Qui 20 è la costante, mentre i tag assegnati al campo di testo sono come questo 50,70,90 ecc.

Qui trovando un nuovo oggetto textfield come responder, renderà il campo di testo corrente come nuovo risponditore e aprirà la tastiera di conseguenza.

```
- (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;
           }
```

Azioni quando un utente ha iniziato / terminato l'interazione con un campo di testo

Per Swift 3.1:

Nel primo esempio si può vedere come si intercetta l'utente che interagisce con un campo di testo durante la scrittura. Allo stesso modo, ci sono metodi in UITextFieldDelegate che vengono chiamati quando un utente ha iniziato e terminato la sua interazione con un campo di testo.

Per poter accedere a questi metodi, è necessario conformarsi al protocollo UITextFieldDelegate e, per ogni campo di testo di cui si desidera ricevere una notifica, assegnare la classe padre come delegato:

```
class SomeClass: UITextFieldDelegate {
```

```
    @IBOutlet var textField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        textField.delegate = self
    }

}
```

Ora sarete in grado di implementare tutti i metodi UITextFieldDelegate.

Per ricevere una notifica quando un utente ha iniziato a modificare un campo di testo, è possibile implementare textFieldDidBeginEditing (metodo _ :) in questo modo:

```
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
    }
}
```

Allo stesso modo, se ti viene notificato se un utente ha interrotto l'interazione con un campo di testo, puoi utilizzare il metodo textFieldDidEndEditing (_ :) in questo modo:

```
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
    }
}
```

Se si desidera controllare se un campo di testo dovrebbe iniziare / terminare la modifica, i metodi textFieldShouldBeginEditing (_ :) e textFieldShouldEndEditing (_ :) possono essere utilizzati restituendo true / false in base alla logica richiesta.

Leggi UITextField delegato online: https://riptutorial.com/it/ios/topic/7185/uitextfield-delegato

Capitolo 199: UITextField personalizzato

introduzione

Usando UITextField personalizzato, possiamo manipolare il comportamento del campo di testo!

Examples

UITextField personalizzato per filtrare il testo di input

Ecco un esempio di `UITextField` personalizzato che prende solo testo numerico e scarta tutti gli altri.

NOTA: per iPhone è facile farlo utilizzando la tastiera del tipo numerico, ma per iPad non esiste una tastiera con solo numeri

```
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: "")
}
}
```

Quindi, ovunque vogliamo campi di testo che prenderebbero solo numeri come testo di input,

allora possiamo usare questo UITextField personalizzato

UITextField personalizzato per non consentire tutte le azioni come copia, incolla, ecc

Se vogliamo disabilitare tutte le azioni come Copia, Incolla, Sostituisci, Seleziona, ecc. Da `UITextField` , possiamo usare il seguente campo di testo personalizzato:

```
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
}
}
```

Usando la proprietà `enableLongPressActions` , possiamo abilitare tutte le azioni in qualsiasi momento successivo, se necessario.

Leggi UITextField personalizzato online: https://riptutorial.com/it/ios/topic/9997/uitextfield-personalizzato

Capitolo 200: UITextView

Examples

Cambia il testo

veloce

```
textView.text = "Hello, world!"
```

Objective-C:

```
textView.text = @"Hello, world!";
```

Imposta il testo attribuito

```
// 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
```

Cambia l'allineamento del testo

veloce

```
textView.textAlignment = .left
```

Objective-C

```
textView.textAlignment = NSTextAlignmentLeft;
```

UITextViewDelegate metodi

Risposta alle notifiche di modifica

- `textViewShouldBeginEditing(_:)`
- `textViewDidBeginEditing(_:)`
- `textViewShouldEndEditing(_:)`
- `textViewDidEndEditing(_:)`

Risposta alle modifiche del testo

- `textView(_:shouldChangeTextIn:replacementText:)`
- `textViewDidChange(_:)`

Risposta all'URL

- `textView(_: UITextView, shouldInteractWithURL: NSURL, inRange: NSRange) -> Bool`

Cambia carattere

veloce

```
//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];
```

Cambia il colore del testo

veloce

```
textView.textColor = UIColor.red
```

Objective-C

```
textView.textColor = [UIColor redColor];
```

UITextView con testo 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];
```

Rileva automaticamente collegamenti, indirizzi, date e altro

`UITextView` ha integrato il supporto per rilevare automaticamente una varietà di dati. I dati che possono essere rilevati automaticamente includono:

```
enum {
   UIDataDetectorTypePhoneNumber   = 1 << 0,
   UIDataDetectorTypeLink          = 1 << 1,
   UIDataDetectorTypeAddress       = 1 << 2,
   UIDataDetectorTypeCalendarEvent = 1 << 3,
   UIDataDetectorTypeNone          = 0,
   UIDataDetectorTypeAll           = NSUIntegerMax
};
```

Abilitazione del rilevamento automatico

```
// you may add as many as you like by using the `|` operator between options
textView.dataDetectorTypes = (UIDataDetectorTypeLink | UIDataDetectorTypePhoneNumber);
```

Se abilitato, il testo verrà visualizzato come collegamento ipertestuale su `UITextView`

Dati cliccabili

Per consentire il clic sul collegamento (che comporterà azioni diverse a seconda del tipo di dati) è necessario assicurarsi che `UITextView` sia selezionabile ma non modificabile e che l'interazione dell'utente sia abilitata

```
textView.editable = NO;
textView.selectable = YES;
textView.userInteractionEnabled = YES; // YES by default
```

Controlla se vuoto o nullo

veloce

```
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
}
```

Ottenere e impostare il Post del cursore

Informazioni utili

L'inizio del testo del campo di testo:

```
let startPosition: UITextPosition = textView.beginningOfDocument
```

La fine del testo del campo di testo:

```
let endPosition: UITextPosition = textView.endOfDocument
```

La gamma attualmente selezionata:

```
let selectedRange: UITextRange? = textView.selectedTextRange
```

Ottieni la posizione del cursore

```
if let selectedRange = textView.selectedTextRange {

    let cursorPosition = textView.offsetFromPosition(textView.beginningOfDocument, toPosition:
selectedRange.start)

    print("\(cursorPosition)")
}
```

Imposta la posizione del cursore

Per impostare la posizione, tutti questi metodi stanno effettivamente impostando un intervallo con gli stessi valori di inizio e fine.

All'inizio

```
let newPosition = textView.beginningOfDocument
textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

All'estremità

```
let newPosition = textView.endOfDocument
textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

Ad una posizione a sinistra della posizione corrente del cursore

```
// 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)
    }
}
```

Ad una posizione arbitraria

Inizia dall'inizio e sposta 5 caratteri a destra.

```
let arbitraryValue: Int = 5
if let newPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: arbitraryValue) {

    textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
}
```

Relazionato

Seleziona tutto il testo

```
textView.selectedTextRange = textView.textRangeFromPosition(textView.beginningOfDocument,
toPosition: textView.endOfDocument)
```

Seleziona un intervallo di testo

```
// 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!)
}
```

Inserisci il testo nella posizione corrente del cursore

```
textView.insertText("Hello")
```

Gli appunti

- Questo esempio deriva originariamente da un adattamento di questa risposta di Overflow dello stack .
- Questa risposta utilizza un campo di testo, ma gli stessi concetti si applicano a `UITextView` .
- Usa `textView.becomeFirstResponder()` per focalizzare il campo di testo e far apparire la tastiera.
- Vedi questa risposta per come ottenere il testo a un certo intervallo.

Relazionato

- Come creare un intervallo in Swift (Si occupa in modo indiretto del problema del motivo per cui dobbiamo utilizzare `selectedTextRange` qui anziché solo `selectedRange`)

Rimuovi gli imbottiture extra per adattarli a un testo misurato con precisione.

`UITextView` ha `UITextView` extra per impostazione predefinita. A volte è fastidioso soprattutto se si desidera misurare del testo senza istanze di visualizzazione e posizionarle in un'area precisa.

Fare questo per rimuovere tali paddings.

```
messageTextView.textContainerInset = UIEdgeInsetsZero
messageTextView.textContainer.lineFragmentPadding = 0
```

Ora puoi misurare le dimensioni del testo usando `NSAttributedString.boundingRectWithSize(...)` e ridimensionare un `UITextView` solo per adattarlo al testo.

```
let budget = getSomeCGSizeBudget()
let text = getSomeAttributedString()
let textSize = text.boundingRectWithSize(budget, options: [.UsesLineFragmentOrigin,
.UsesFontLeading], context: nil).size
messageTextView.frame.size = textSize // Just fits.
```

Leggi UITextView online: https://riptutorial.com/it/ios/topic/1043/uitextview

Capitolo 201: UIView

Sintassi

1. // Obiettivo-C
2. [UIView new] // Ottieni un oggetto vista allocato e inizializzato
3. [[UIView alloc] initWithFrame: (Pass CGRect)] // Ottieni la vista allocata e inizializzata con una cornice
4. [[UIView alloc] init] // Ottieni un oggetto vista allocato e inizializzato
5. // Swift
6. UIView () // Crea un'istanza UIView con frame CGRect.zero
7. UIView (frame: CGRect) // Crea un'istanza UIView specificando il frame
8. UIView.addSubview (UIView) // aggiungi un'altra istanza di UIView come sottoview
9. UIView.hidden // Ottieni o imposta la visibilità della vista
10. UIView.alpha // Ottieni o imposta l'opacità della vista
11. UIView.setNeedsLayout () // Forza la vista ad aggiornare il suo layout

Osservazioni

La classe **UIView** definisce un'area rettangolare sullo schermo e le interfacce per la gestione del contenuto in quell'area. In fase di runtime, un oggetto vista gestisce il rendering di qualsiasi contenuto nella sua area e gestisce anche eventuali interazioni con quel contenuto.

Examples

Crea un 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;
```

veloce

```
let myFrame = CGRect(x: 0, y: 0, width: 320, height: 35)
let view = UIView(frame: myFrame)
```

Rendi la vista arrotondata

Per rendere `UIView` arrotondato, specifica un `cornerRadius` per il `layer` della vista.

Ciò vale anche per qualsiasi classe che eredita da `UIView`, come `UIImageView`.

programmazione

codice SWIFT

```
someImageView.layoutIfNeeded()
someImageView.clipsToBounds = true
someImageView.layer.cornerRadius = 10
```

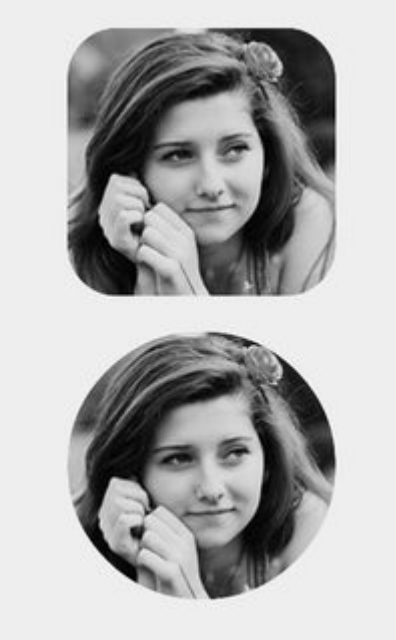
Codice Objective-C

```
[someImageView layoutIfNeeded];
someImageView.clipsToBounds = YES;
someImageView.layer.cornerRadius = 10;
```

Esempio

```
//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;
```

Ecco il risultato, che mostra l'effetto della vista arrotondata utilizzando il raggio dell'angolo specificato:



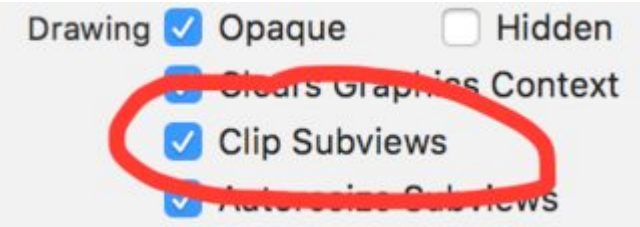
Nota

Per fare questo è necessario includere il framework QuartzCore.

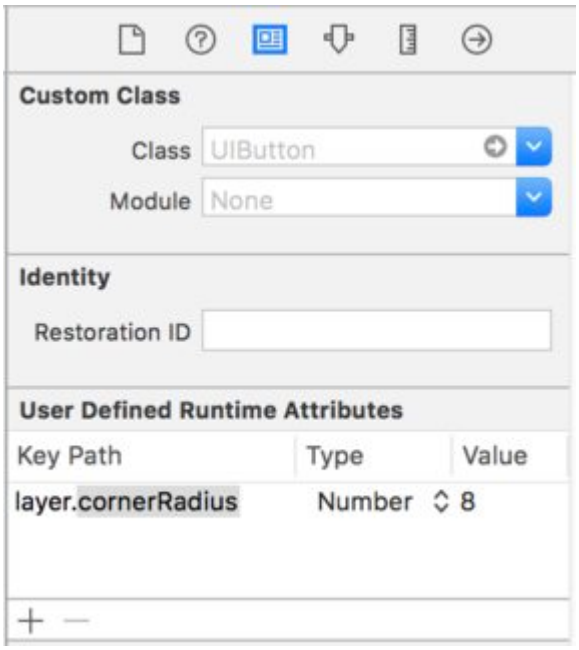
```
#import <QuartzCore/QuartzCore.h>
```

Configurazione Storyboard

Un effetto di visualizzazione arrotondato può anche essere ottenuto `non-programmatically` impostando le proprietà corrispondenti in **Storyboard** .



Poiché le proprietà del `layer` non sono esposte in Storyboard, è necessario modificare l'attributo `cornerRadius` tramite la sezione Attributi runtime definiti dall'utente.



Estensione rapida

È possibile utilizzare questa pratica estensione per applicare la vista arrotondata purché abbia la stessa larghezza e altezza.

```
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
    }
}
```

Per usarlo:

```
yourView.setAsCircle()
```

Prendendo un'istantanea

Puoi scattare un'istantanea da un `UIView` come questo:

veloce

```
let snapshot = view.snapshotView(afterScreenUpdates: true)
```

Objective-C

```
UIView *snapshot = [view snapshotViewAfterScreenUpdates: YES];
```

Utilizzando IBInspectable e IBDesignable

Uno (o due) dei più cool nuove funzionalità di versioni più recenti Xcode sono i `IBInspectable` proprietà e `IBDesignable` `UIView` s. Questi non hanno nulla a che fare con la funzionalità della tua applicazione, ma influiscono invece sull'esperienza degli sviluppatori in Xcode. L'obiettivo è essere in grado di ispezionare visivamente le visualizzazioni personalizzate nella tua applicazione iOS senza eseguirla. Supponiamo quindi di avere una vista personalizzata chiamata in modo creativo `CustomView` che eredita da `UIView` . In questa visualizzazione personalizzata, verrà visualizzata una stringa di testo con un colore designato. Puoi anche scegliere di non visualizzare alcun testo. Avremo bisogno di tre proprietà:

```
var textColor: UIColor = UIColor.blackColor()
var text: String?
var showText: Bool = true
```

Possiamo quindi eseguire l'override della funzione `drawRect` nella classe:

```
if showText {
    if let text = text {
        let s = NSString(string: text)
        s.drawInRect(rect,
            withAttributes: [
                NSForegroundColorAttributeName: textColor,
                NSFontAttributeName: UIFont(name: "Helvetica Neue", size: 18)!
            ])
    }
}
```

Supponendo che la proprietà `text` sia impostata, questa disegnerà una stringa nell'angolo in alto a sinistra della vista quando si esegue l'applicazione. Il problema è che non sapremo come apparire senza eseguire l'applicazione. Qui è dove `IBInspectable` e `IBDesignable` entrano. `IBInspectable` ci permette di impostare visivamente i valori delle proprietà della vista in Xcode, proprio come con i controlli integrati. `IBDesignable` ci mostrerà un'anteprima visiva nello storyboard. Ecco come dovrebbe apparire la classe:

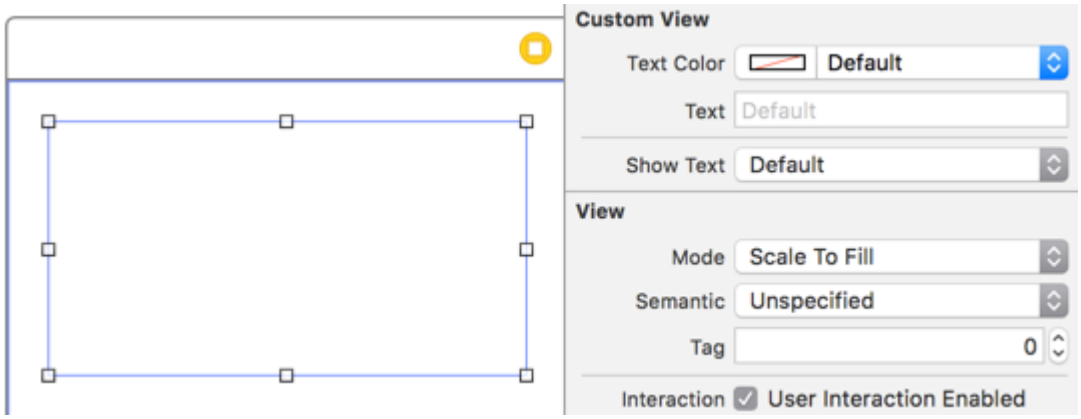
```
@IBDesignable
class CustomView: UIView {
    @IBInspectable var textColor: UIColor = UIColor.blackColor()
    @IBInspectable var text: String?
    @IBInspectable var showText: Bool = true

    override func drawRect(rect: CGRect) {
        // ...
    }
}
```

O nell'Obiettivo C:

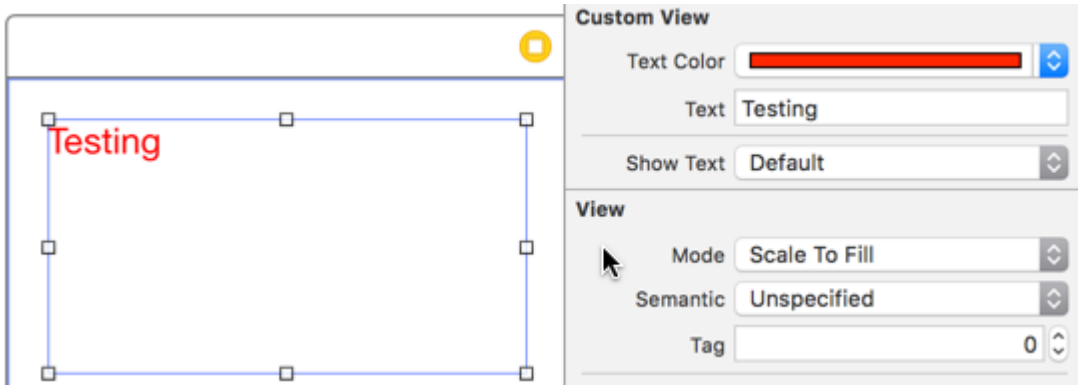
```
IB_DESIGNABLE
@interface CustomView: UIView

@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
```

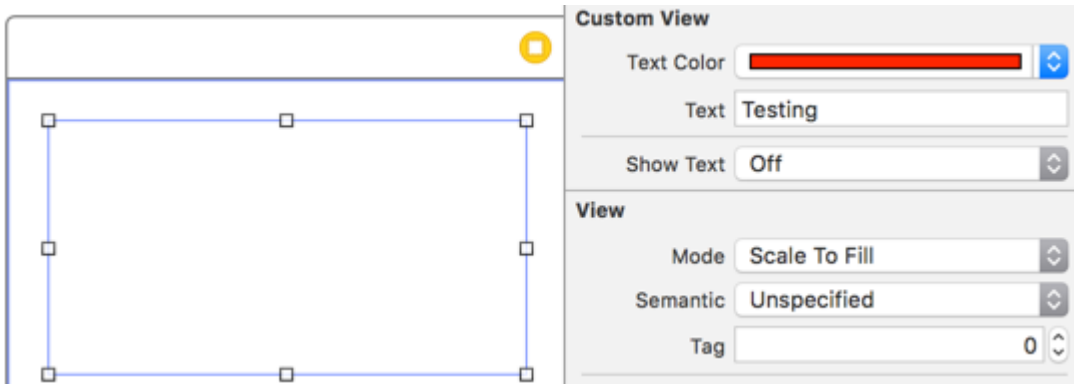
I prossimi screenshot mostrano cosa succede in Xcode. Il primo è quello che succede dopo aver aggiunto la classe rivista. Si noti che ci sono tre nuovi elementi dell'interfaccia utente per le tre proprietà. Il *Colore del testo* mostrerà un selettore di colori, *Testo* è solo una casella di input e *Mostra testo* ci darà le opzioni per `Off` e `On` che sono rispettivamente `false` e `true` .



Il prossimo è dopo aver cambiato il *colore* del *testo* in rosso usando il selettore di colori. Inoltre, è stato fornito del testo per renderlo visualizzabile dalla funzione `drawRect` . Si noti che anche la vista in Interface Builder è stata aggiornata.

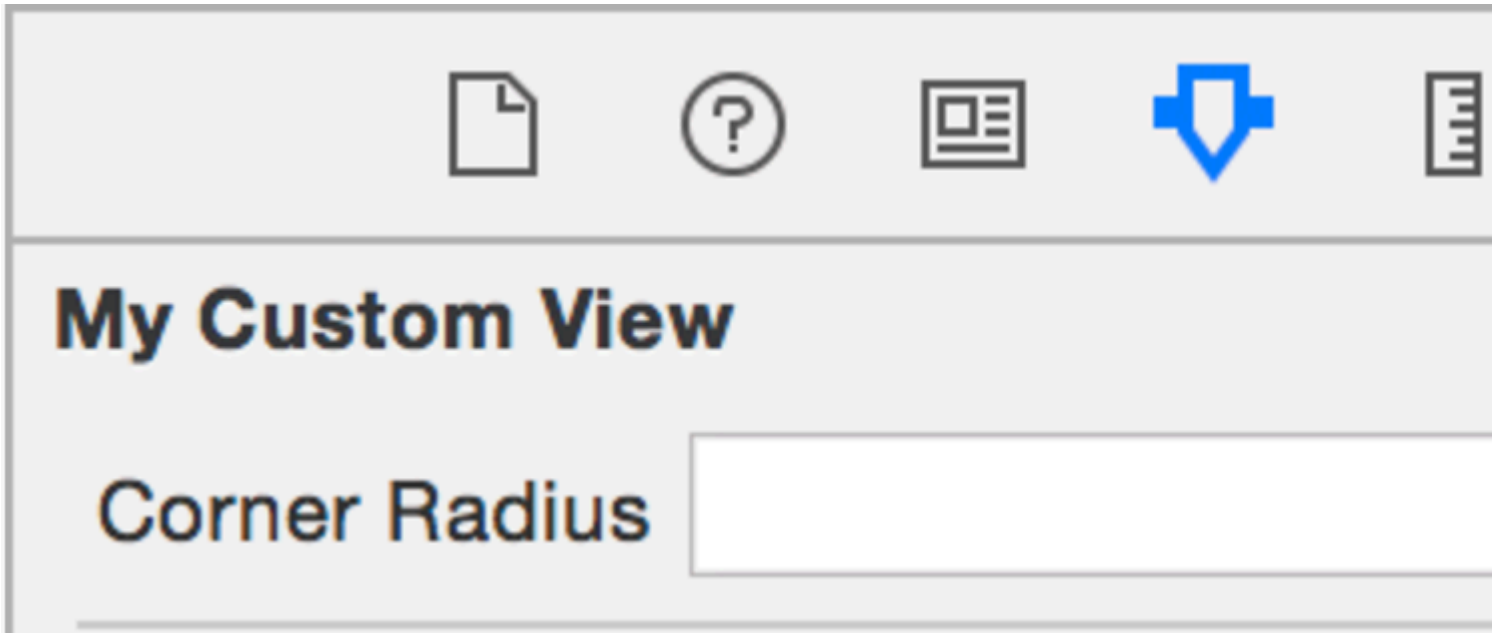


Infine, l'impostazione *Mostra testo* su `Off` nella finestra di ispezione delle proprietà fa scomparire la visualizzazione del testo in Interface Builder.



Tuttavia, veniamo tutti in una situazione in cui dobbiamo creare `UIView` arrotondato a più viste nello `Storyboard` Invece di dichiarare `IBDesignable` a tutte le visualizzazioni di `Storyboard` , è meglio creare `Extension` di `UIView` e ottenere un'interfaccia utente costruita per ogni tuo `UIView` attraverso il progetto per creare una vista arrotondata impostando il raggio dell'angolo. Un raggio di confine configurabile su qualsiasi UIView creato nello storyboard.

```
extension UIView {

    @IBInspectable var cornerRadius:CGFloat {
        set {
            layer.cornerRadius = newValue
            clipsToBounds = newValue > 0
        }
        get {
            return layer.cornerRadius
        }
    }

}
```



Animazione di un 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()
}
```

Estensione UIView per attributi di dimensioni e frame

Se vogliamo ottenere la x-coordinate di origine della vista, allora dobbiamo scrivere come:

```
view.frame.origin.x
```

Per larghezza, dobbiamo scrivere:

```
view.frame.size.width
```

Ma se aggiungiamo un'estensione semplice a un `UIView` , possiamo ottenere tutti gli attributi in modo molto semplice, come:

```
view.x
view.y
view.width
view.height
```

Aiuterà anche ad impostare questi attributi come:

```
view.x = 10
view.y = 10
view.width = 100
view.height = 200
```

E la semplice estensione sarebbe:

```
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)
        }
    }
}
```

Abbiamo bisogno di aggiungere questo file di classe in un progetto e sarà disponibile per l'uso in tutto il progetto!

Gestisci programmaticamente l'inserimento e la cancellazione di UIView in e da un altro UIView

Supponiamo di avere un `parentView` in cui si desidera inserire una nuova `subView` (ad esempio, quando si desidera inserire un `UIImageView` in una vista di `UIViewController`), di quanto si possa fare come di seguito.

Objective-C

```
[parentView addSubview:subView];
```

veloce

```
parentView.addSubview(subView)
```

È anche possibile aggiungere la subView sotto un'altra `subView2` , che è già una vista secondaria di parentView utilizzando il seguente codice:

Objective-C

```
[parentView insertSubview:subView belowSubview:subView2];
```

veloce

```
parentView.insertSubview(subView, belowSubview: subView2)
```

Se vuoi inserirlo sopra `subView2` puoi farlo in questo modo:

Objective-C

```
[parentView insertSubview:subView aboveSubview:subView2];
```

veloce

```
parentView.insertSubview(subView, aboveSubview: subView2)
```

Se da qualche parte nel tuo codice hai bisogno di portare una certa `subView` in primo piano, quindi sopra tutte le `parentView` degli altri parentView, puoi farlo in questo modo:

Objective-C

```
[parentView bringSubviewToFront:subView];
```

veloce

```
parentView.bringSubviewToFront(subView)
```

Infine, se si desidera rimuovere `subView` da `parentView` , è possibile fare come di seguito:

Objective-C

```
[subView removeFromSuperview];
```

veloce

```
subView.removeFromSuperview()
```

Crea UIView usando l'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];
```

funzioni

Funzione per aggiungere la vista ad altezza fissa usando i vincoli di autolayout

```
-(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];

}
```

La funzione aggiunge il vincolo di ridimensionamento completo per UIView creato.

```
-(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];
}
```

Utilizzo della dimensione del contenuto intrinseco

Quando si crea una sottoclasse UIView, la dimensione del contenuto intrinseco aiuta a evitare l'impostazione dei vincoli di altezza e larghezza hardcoded

uno sguardo di base su come una classe può utilizzare questo

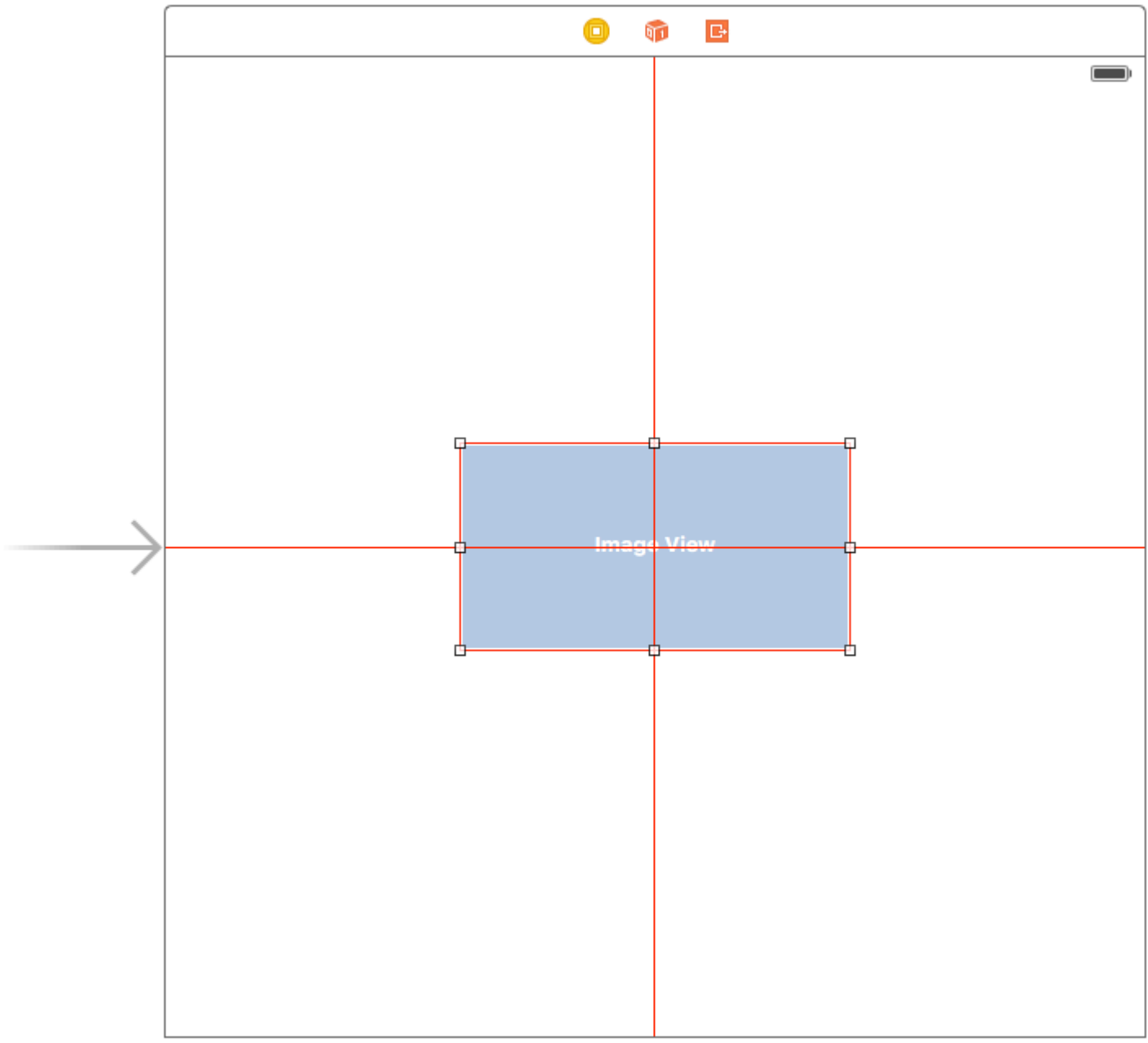
```
class ImageView: UIView {
    var image: UIImage {
        didSet {
            invalidateIntrinsicContentSize()
        }
    }
    // omitting initializers
    // convenience init(image: UIImage)

    override func intrinsicContentSize() -> CGSize {
        return CGSize(width: image.size.width, height: image.size.height)
    }
}
```

Se si desidera fornire solo una dimensione intrinsecamente, è possibile fornire il valore `UIViewNoIntrinsicMetric` per il valore che si desidera ignorare.

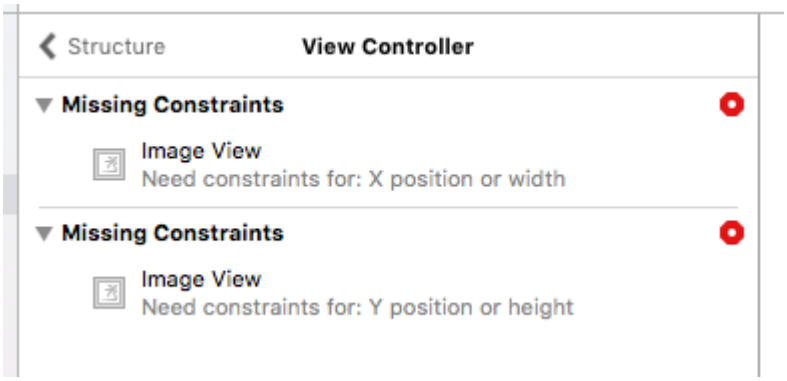
```
override func intrinsicContentSize() -> CGSize {
    return CGSize(width: UIViewNoIntrinsicMetric, height: image.size.width)
}
```

Vantaggi quando si utilizza con AutoLayout e Interface Builder

Uno potrebbe prendere questo ImageView (o UIImageView) e impostare l'allineamento orizzontale al centro superview X e l'allineamento verticale al centro superview Y.

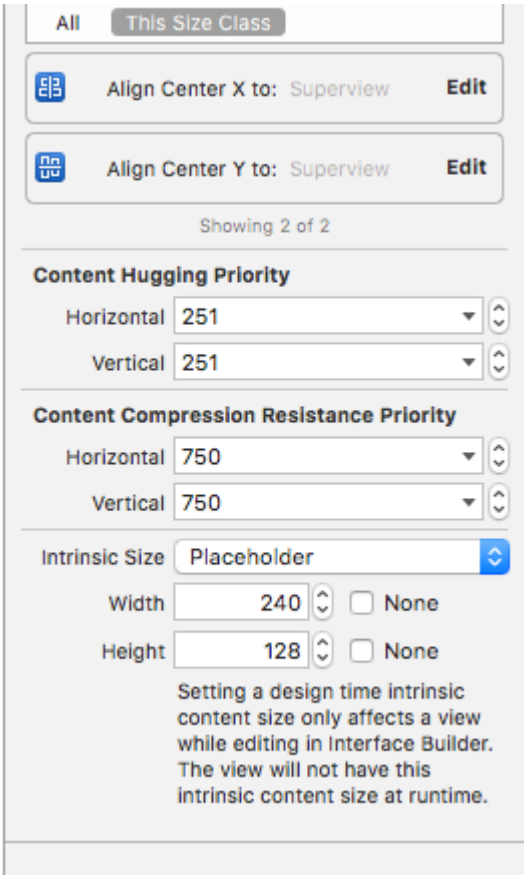


Il generatore di interfacce ti lamenterà a questo punto fornendo il seguente avviso:



È qui che entra in gioco la `Placeholder Intrinsic Size` .

Entrando nel pannello Dimensione ispettore e fino al menu a discesa Dimensione intrinseca, puoi cambiare questo valore da Predefinito a Segnaposto.



e ora l'interfaccia builder rimuoverà gli avvertimenti precedenti e puoi usare questa dimensione per avere viste di dimensioni dinamiche disposte nel builder dell'interfaccia.

Scuotere una vista

```
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")
    }
```

```
}
```

Questa funzione può essere utilizzata per attirare l'attenzione su una vista specifica scuotendola un po '.

Leggi UIView online: https://riptutorial.com/it/ios/topic/858/uiview

Capitolo 202: UIViewController

Examples

subclassing

Sottoclasse `UIControl` ci dà accesso ai seguenti metodi:

- `beginTrackingWithTouch` viene chiamato quando il dito tocca prima i limiti del controllo.
- `continueTrackingWithTouch` viene chiamato ripetutamente mentre il dito scorre sul controllo e anche al di fuori dei limiti del controllo.
- `endTrackingWithTouch` viene chiamato quando il dito si solleva dallo schermo.

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

Questo è il modo in cui il controller di visualizzazione è impostato per essere il delegato e rispondere agli eventi di tocco dal nostro controllo personalizzato.

```
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"
    }
}
```

Gli appunti

- I metodi alternativi per ottenere lo stesso risultato senza sottoclassi includono l'aggiunta di un obiettivo o l'utilizzo di un riconoscitore di gesti.
- Non è necessario utilizzare un delegato con questi metodi se vengono utilizzati solo all'interno del controllo personalizzato stesso. Potremmo aver appena aggiunto una dichiarazione di `print` per mostrare come vengono chiamati gli eventi. In tal caso, il codice verrà semplificato in

```
  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")
      }
  }
```

Crea un'istanza

veloce

```
let viewController = UIViewController()
```

Objective-C

```
UIViewController *viewController = [UIViewController new];
```

Imposta la vista a livello di codice

veloce

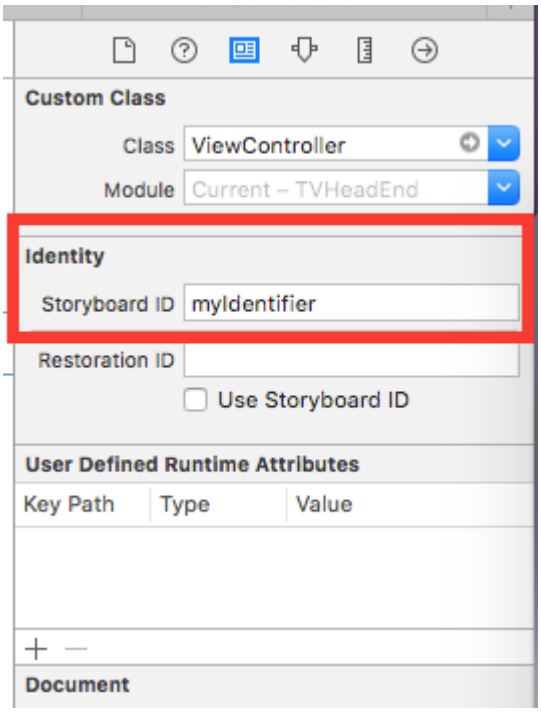
```
class FooViewController: UIViewController {

  override func loadView() {
    view = FooView()
  }

}
```

Istanziare da uno storyboard

```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
```

Con un identificatore :

Assegna alla scena un ID storyboard all'interno del controllo identità dello storyboard.

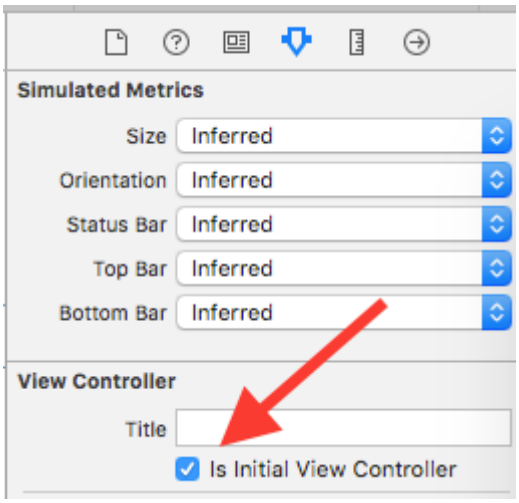


Istanziare nel codice:

```
UIViewController *controller = [storyboard
instantiateViewControllerWithIdentifier:@"myIdentifier"];
```

Istanziare un viewcontroller iniziale :

All'interno dello storyboard selezionare il controller della vista, quindi selezionare l'ispettore degli attributi, selezionare la casella "È il controller della vista iniziale".




```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
UIViewController *controller = [storyboard instantiateInitialViewController];
```

Accedi al controller di visualizzazione del contenitore

Quando il controller della vista viene presentato all'interno di un controller della barra delle linguette, è possibile accedere al controller della barra delle schede in questo modo:

veloce

```
let tabBarController = viewController.tabBarController
```

Objective-C

```
UITabBarController *tabBarController = self.tabBarController;
```

Quando il controller della vista fa parte di una pila di navigazione, puoi accedere al controller di navigazione in questo modo:

veloce

```
let navigationController = viewController.navigationController
```

Objective-C

```
UINavigationController *navigationController = self.navigationController;
```

Aggiunta / rimozione di un controller di visualizzazione figlio

Per aggiungere un controller di visualizzazione figlio:

```
- (void)displayContentController:(UIViewController *)vc {
   [self addChildViewController:vc];
   vc.view.frame = self.view.frame;
   [self.view addSubview:vc.view];
   [vc didMoveToParentViewController:self];
}
```

Per rimuovere un controller di visualizzazione figlio:

```
- (void)hideContentController:(UIViewController *)vc {
   [vc willMoveToParentViewController:nil];
   [vc.view removeFromSuperview];
   [vc removeFromParentViewController];
}
```

Leggi UIViewController online: https://riptutorial.com/it/ios/topic/1956/uiviewcontroller

Capitolo 203: UIWebView

Osservazioni

Funzioni del delegato di UIWebView: -

Objective-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

Creare un'istanza UIWebView

veloce

```
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;
```

Fare una richiesta di URL

Carica contenuto in WebView dal `url`

veloce

```
webview.loadRequest(NSURLRequest(URL: NSURL(string: "http://www.google.com")!))
```

Objective-C

```
[webview loadRequest:[NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://www.google.com"]]];
```

Interrompere il caricamento di contenuto Web

Metodo `stopLoading()` interrompe il processo di caricamento corrente della webview.

veloce

```
webview.stopLoading()
```

Objective-C

```
[webview stopLoading];
```

Ricarica il contenuto Web corrente

veloce

```
webview.reload()
```

Objective-C

```
[webview reload];
```

Determinazione della dimensione del contenuto

In molti casi, ad esempio quando si utilizzano viste Web in celle di visualizzazione tabella, è importante determinare la dimensione del contenuto della pagina HTML renderizzata. Dopo aver caricato la pagina, questo può essere calcolato nel metodo delegato `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);
}
```

Il codice impiega un ulteriore trucco per impostare brevemente l'altezza della visualizzazione web su 1 prima di misurare la dimensione del raccordo. Altrimenti riporterebbe semplicemente la dimensione attuale del frame. Dopo la misurazione, impostiamo immediatamente l'altezza all'altezza del contenuto effettivo.

fonte

Carica una stringa HTML

Le visualizzazioni Web sono utili per caricare stringhe HTML generate localmente.

```
NSString *html = @"<!DOCTYPE html><html><body>Hello World</body></html>";
[webView loadHTMLString:html baseURL:nil];
```

veloce

```
  let htmlString = "<h1>My First Heading</h1><p>My first paragraph.</p>"
  webView.loadHTMLString(htmlString, baseURL: nil)
```

È possibile specificare un URL di base locale. È utile per fare riferimento a immagini, fogli di stile o script dal pacchetto di app:

```
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]]];
```

In questo caso, `style.css` viene caricato localmente dalla directory delle risorse dell'app. Ovviamente è anche possibile specificare un URL remoto.

Carica JavaScript

Possiamo eseguire JavaScript personalizzato su un `UIWebView` usando il metodo `stringByEvaluatingJavaScriptFromString()` . Questo metodo restituisce il risultato dell'esecuzione dello script JavaScript passato nel parametro dello script, o zero se lo script non riesce.

veloce

Carica script da String

```
webview.stringByEvaluatingJavaScriptFromString("alert('This is JavaScript!');")
```

Carica lo script dal file locale

```
//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

Carica script da String

```
[webview stringByEvaluatingJavaScriptFromString:@"alert('This is JavaScript!');"];
```

Carica lo script dal file locale

```
//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];
```

Nota `stringByEvaluatingJavaScriptFromString:` metodo attende in modo sincrono il completamento della valutazione JavaScript. Se carichi contenuti web di cui non hai controllato il codice JavaScript, l'attivazione di questo metodo potrebbe bloccare la tua app. È consigliabile adottare la classe `WKWebView` e utilizzare invece il metodo `evaluateJavaScript:completionHandler:` . Ma `WKWebView` è disponibile da iOS 8.0 e versioni successive.

Carica file di documenti come .pdf, .txt, .doc ecc.

Invece di pagine Web, possiamo anche caricare i file di documento in iOS WebView come .pdf, .txt, .doc ecc. `loadData` metodo `loadData` viene utilizzato per caricare `NSData` in webview.

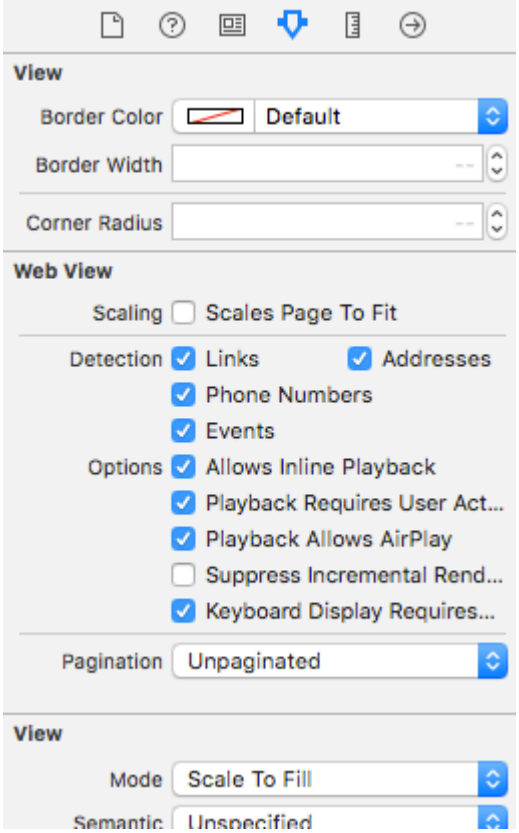
veloce

```
//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]];
```

Crea collegamenti che all'interno di UIWebview cliccabili



In vc.h

```
@interface vc : UIViewController<UIWebViewDelegate>
```

in 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;

}
```

Carica il file HTML locale in webView

Innanzitutto, aggiungi il file HTML al tuo progetto (se ti viene chiesto di scegliere le opzioni per aggiungere il file, seleziona *Copia gli elementi se necessario*)

La seguente riga di codice carica il contenuto del file HTML nel webView

```
webView.loadRequest(NSURLRequest(URL: NSURL(fileURLWithPath:
NSBundle.mainBundle().pathForResource("YOUR HTML FILE", ofType: "html")!})
```

- Se il tuo file HTML è chiamato index.html, sostituisci il **TUO FILE HTML** con l' **indice**
- È possibile utilizzare questo codice in *viewDidLoad ()* o *viewDidAppear ()* o qualsiasi altra funzione

Leggi UIWebView online: https://riptutorial.com/it/ios/topic/1452/uiwebview

Capitolo 204: Utilizzando Image Aseets

introduzione

Le risorse immagine sono utilizzate per gestire e organizzare diversi tipi di risorse immagine nella nostra app iOS utilizzando Xcode.

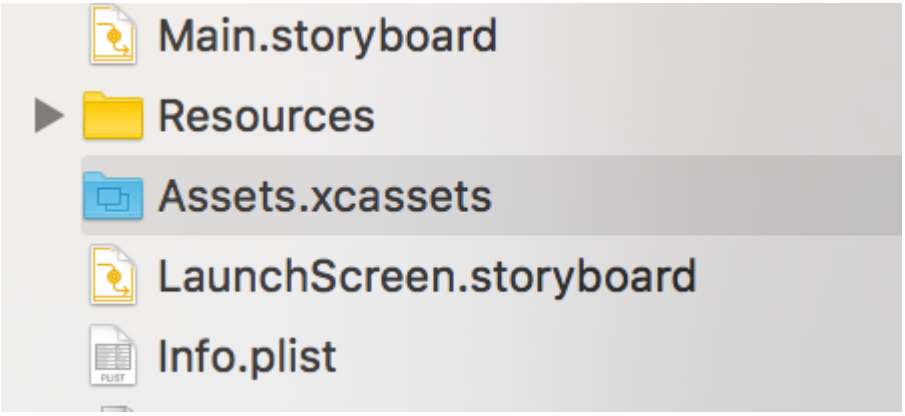
Queste risorse possono essere **icone di app, immagini di lancio, immagini utilizzate in tutta l'app, immagini a grandezza naturale, immagini di dimensioni casuali** ecc.

Examples

Icona dell'app che utilizza risorse immagine

Ogni volta che creiamo un nuovo progetto in Xcode per la nostra nuova app, ci fornisce diverse classi, target, test, file plist, ecc. In modo simile ci fornisce anche il file `Assets.xcassets` , che gestisce tutte le risorse dell'immagine nel nostro progetto.

Ecco come appare questo file nel navigatore di file:



Se lo clicchiamo, sarà simile a questo:

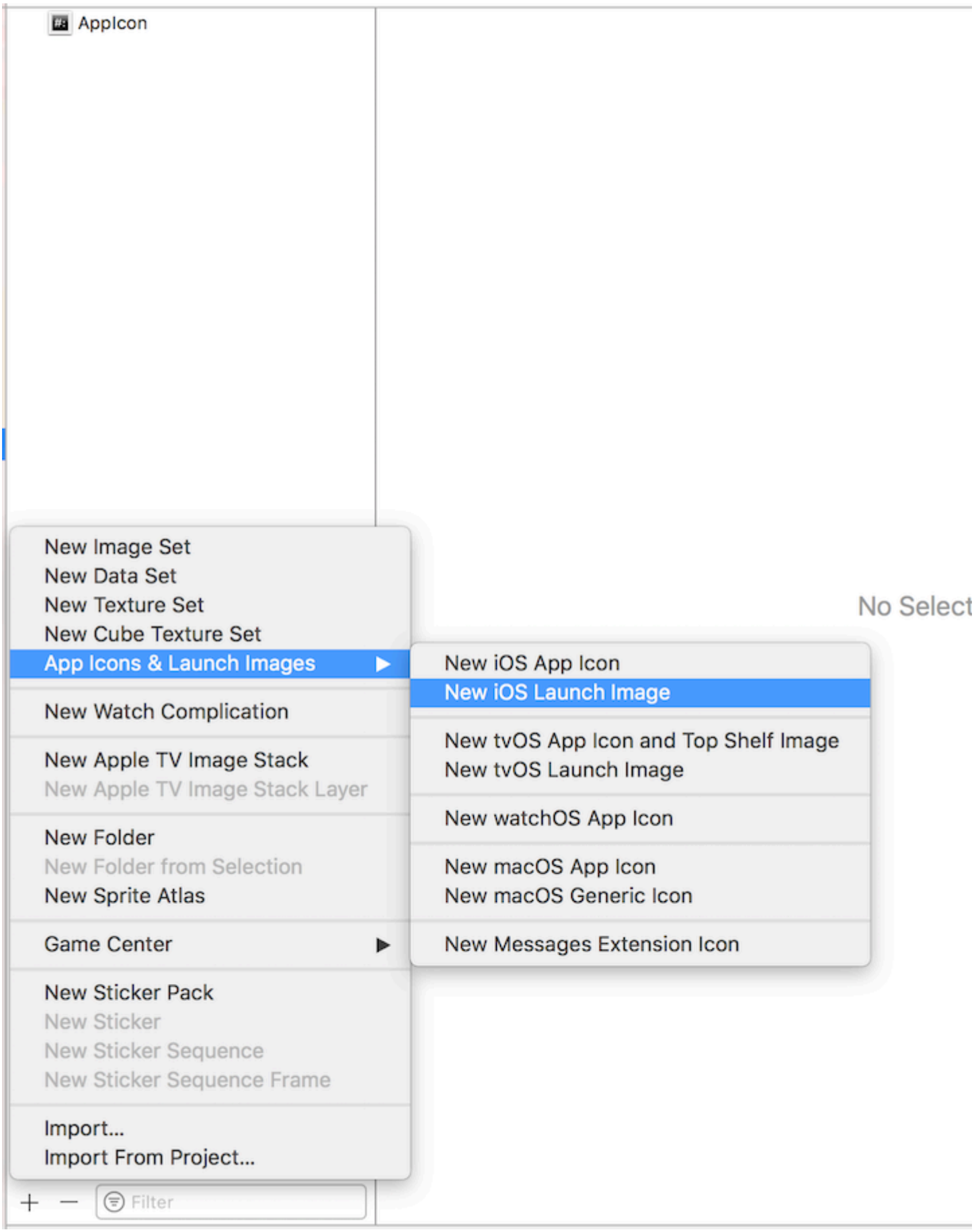
AppIcon

Dobbiamo solo **trascinare e rilasciare la** rispettiva immagine su ogni blocco quadrato vuoto. Ogni nero ci dirà quale dimensione dovrebbe essere quell'immagine, è scritta proprio sotto di essa. Dopo aver trascinato e rilasciato tutte le immagini in tutti i quadrati, sarà simile a questo:


AppIcon


AppIcon

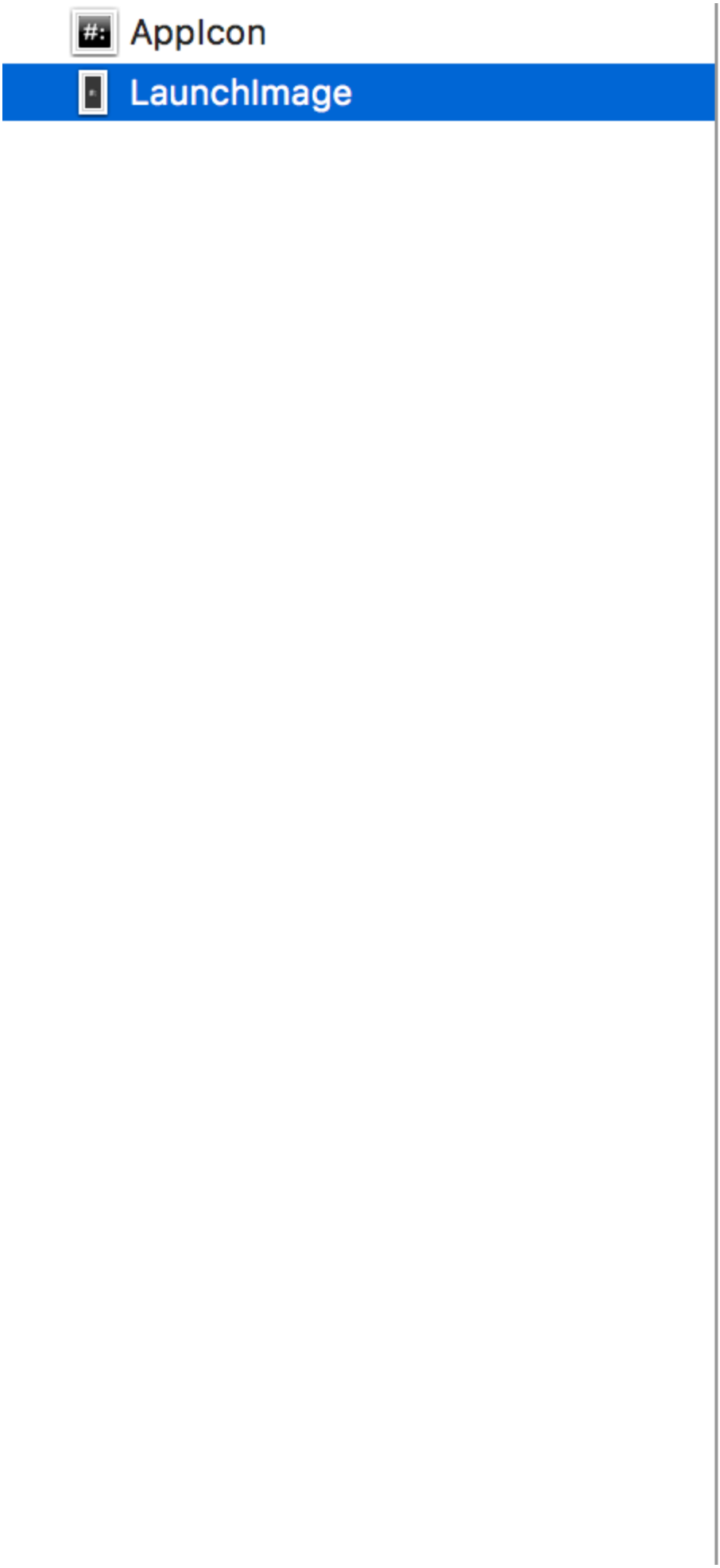
in basso come:



Dopodiché, in base al nostro requisito, possiamo modificare le caselle vuote per i dispositivi che

supportiamo utilizzando gli attributi inspector selezionando / deselezionando le caselle.

Ho riempito queste immagini per iPhone da 4 "a 5,5" e per tutti gli iPad come:


AppIcon
LaunchImage
LaunchImage

Ecco le dimensioni di tutte le immagini di avvio:

```
Retina HD 5.5" iPhone Portrait – iPhone (6, 6S, 7)Plus – 1242x2208px
Retina HD 4.7" iPhone Portrait – iPhone 6, 6S, 7 – 750x1334px
Retina HD 5.5" iPhone Landscape – iPhone (6, 6S, 7)Plus – 2208x1242px
2x iPhone Portrait – (3.5") iPhone 4S – 640x960px
Retina 4 iPhone Portrait – (4") iPhone 5, 5S, 5C, iPod Touch, SE – 640x1136px
2x iPad Portrait – All Retina iPads – 1536x2048px
2x iPad Landscape – All Retina iPads – 2048x1536px
```

Gli appunti:

1 **iPad senza retina:** ho lasciato in bianco `1x iPad Portrait and Landscape` perché gli iPad senza retina useranno le immagini di avvio `2x` mediante ridimensionamento

2 **12.9 "iPad Pro** : non esiste un quadratino per questo iPad perché questo iPad utilizzerà anche `2x iPad` immagini `2x iPad` ridimensionandole

3 **Retina HD 5.5 ": gli** iPad dovrebbero avere `1920x1080px` per `1080x1920px` e `1080x1920px` per orizzontale, ma Xcode darà waring e l'immagine di lancio non verrà mostrata su quei dispositivi

4 **SplitView:** poiché stiamo utilizzando `LaunchImage Asset` invece di `LaunchScreen XIB` , la nostra app non supporterà `SplitView` su iPad e iPhone 5.5 "

5 **Reinstalla:** se la nostra app è già installata sul dispositivo e tentiamo di eseguire queste risorse immagine di avvio appena aggiunte, a volte il dispositivo non mostrerà le immagini di avvio durante l'avvio dell'app. In questo caso basta eliminare l'app dal dispositivo, pulire + costruire il progetto ed eseguirlo, mostrerà le nuove immagini di avvio

Leggi Utilizzando Image Aseets online: https://riptutorial.com/it/ios/topic/10087/utilizzando-image-aseets

Capitolo 205: UUID (Universally Unique Identifier)

Osservazioni

Per salvare l'UUID possiamo usare SSKeychainUtility . L'esempio può essere trovato sulla pagina Github

Examples

Generazione UUID

UUID casuale

veloce

```
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;
}
```

Identificatore per il venditore

iOS 6

All'interno di una singola riga, possiamo ottenere un UUID come di seguito:

veloce

```
let UDIDString = UIDevice.currentDevice().identifierForVendor?.UUIDString
```

Objective-C

```
NSString *UDIDString = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
```

`identifierForVendor` è un identificatore univoco che rimane uguale per ogni app di un singolo fornitore su un singolo dispositivo, a meno che tutte le app del fornitore non vengano eliminate da questo dispositivo. Vedi la documentazione di Apple su quando questo `UUID` cambia.

IFA di Apple vs IFV (identificativo Apple per inserzionisti vs. identificatore per fornitori)

- È possibile utilizzare l'IFA per misurare i clic sugli annunci e l'IFV per misurare le installazioni di app.
- IFA ha meccanismi di privacy incorporati che lo rendono perfetto per la pubblicità. Al contrario, l'IFV è per gli sviluppatori di utilizzare internamente per misurare gli utenti che installano le loro app.

SE UNA

- La classe ASIdentifierManager fornisce
    - **advertisingIdentifier: UUID** : una stringa alfanumerica unica per ciascun dispositivo, utilizzata solo per pubblicare annunci pubblicitari.
    - **isAdvertisingTrackingEnabled** : un valore booleano che indica se l'utente ha un monitoraggio degli annunci limitato.

IFV

- La classe ASIdentifierManager fornisce
    - **identifierForVendor: UUID** : una stringa alfanumerica che identifica in modo univoco un dispositivo per il fornitore dell'app.

Trova il tuo dispositivo IFA e IFV qui .

Crea una stringa UUID per dispositivi iOS

Qui possiamo creare una `UUID String` con in una sola riga.

Rappresenta le stringhe UUID, che possono essere utilizzate per identificare in modo univoco tipi, interfacce e altri elementi.

Swift 3.0

```
print(UUID().uuidString)
```

È molto utile per identificare più dispositivi con ID univoco.

Leggi UUID (Universally Unique Identifier) online: https://riptutorial.com/it/ios/topic/3629/uuid--universally-unique-identifier-

Capitolo 206: Valutazione della domanda / richiesta di revisione

introduzione

Ora da iOS 10.3, non è necessario passare da un'applicazione all'app store Apple per la valutazione / revisione. Apple ha introdotto la classe SKStoreReviewController nel framework storekit. In quale sviluppatore è sufficiente chiamare il metodo di classe requestReview () della classe SKStoreReviewController e il sistema gestisce l'intero processo per te.

Inoltre, puoi continuare a includere un link permanente nelle impostazioni o nelle schermate di configurazione della tua app che collegano in profondità alla pagina del tuo prodotto App Store. Per ope automaticamente

Examples

Valuta / Rivedi l'applicazione iOS

Basta digitare sotto un codice di linea da cui si desidera che l'utente valuti / riveda la propria applicazione.

SKStoreReviewController.requestReview ()

Leggi Valutazione della domanda / richiesta di revisione online:
https://riptutorial.com/it/ios/topic/9678/valutazione-della-domanda---richiesta-di-revisione

Capitolo 207: Verifica della connettività di rete

Osservazioni

Il codice sorgente di `Reachability.h` e `Reachability.m` sono disponibili sul sito di documentazione per sviluppatori Apple.

Avvertenze

A differenza di altre piattaforme, Apple deve ancora fornire un set standard di API per determinare lo stato della rete di un dispositivo iOS e offrire solo questi esempi di codice collegati sopra. Il file di origine cambia nel tempo, ma una volta importati in un progetto di app, vengono raramente aggiornati dagli sviluppatori.

Per questo motivo, la maggior parte degli sviluppatori di app tende a utilizzare una delle tante librerie gestite da Github / Cocoapod per raggiungibilità.

Apple consiglia inoltre, per le richieste fatte su richiesta dell'utente, di tentare sempre *prima* una connessione, prima di utilizzare Reachability / SCNetworkReachability per diagnosticare l'errore o attendere il ritorno della connessione .

Examples

Creazione di un listener Reachability

La classe Reachability di Apple controlla periodicamente lo stato della rete e avvisa gli osservatori delle modifiche.

```
Reachability *internetReachability = [Reachability reachabilityForInternetConnection];
[internetReachability startNotifier];
```

Aggiungi osservatore alle modifiche di rete

`Reachability` utilizza i messaggi `NSNotification` per avvisare gli osservatori quando lo stato della rete è cambiato. La tua classe dovrà diventare un osservatore.

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(reachabilityChanged:) name:kReachabilityChangedNotification object:nil];
```

Altrove nella tua classe, implementa la firma del metodo

```
- (void) reachabilityChanged:(NSNotification *)note {
    //code which reacts to network changes
```

```
}
```

Avvisa quando la rete non è più disponibile

```
- (void)reachabilityChanged:(NSNotification *)note {
    Reachability* reachability = [note object];
    NetworkStatus netStatus = [reachability currentReachabilityStatus];

    if (netStatus == NotReachable) {
        NSLog(@"Network unavailable");
    }
}
```

Avvisa quando la connessione diventa WIFI o rete cellulare

```
- (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;
    }
}
```

Verifica se è connesso alla rete

veloce

```
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:

possiamo controllare la connettività di rete entro poche righe di codice come:

```
-(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;
    }
}
```

Leggi Verifica della connettività di rete online: https://riptutorial.com/it/ios/topic/704/verifica-della-connettivita-di-rete

Capitolo 208: WCSessionDelegate

introduzione

`WCSessionDelegate` funziona con watch OS2 + usando WatchConnectivity. var watchSession: WCSession? func startWatchSession () {if (WCSession.isSupported ()) {watchSession = WCSession.default () watchSession! .delegate = self watchSession! .activate ()}} Implementa il metodo richiesto: - `didReceiveApplicationContext`

Examples

Controller del kit di controllo (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")
    }
```

Leggi WCSessionDelegate online: https://riptutorial.com/it/ios/topic/8289/wcsessiondelegate

Capitolo 209: WKWebView

introduzione

WKWebView è il fulcro della moderna API WebKit introdotta in iOS 8 e OS X Yosemite. Sostituisce UIWebView in UIKit e WebView in AppKit, offrendo un'API coerente tra le due piattaforme.

Grazie allo scrolling reattivo a 60fps, ai gesti integrati, alla comunicazione semplificata tra app e pagina web e allo stesso motore JavaScript di Safari, WKWebView è uno degli annunci più significativi che usciranno dal WWDC 2014.

Examples

Creare un semplice browser web

```
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)
```

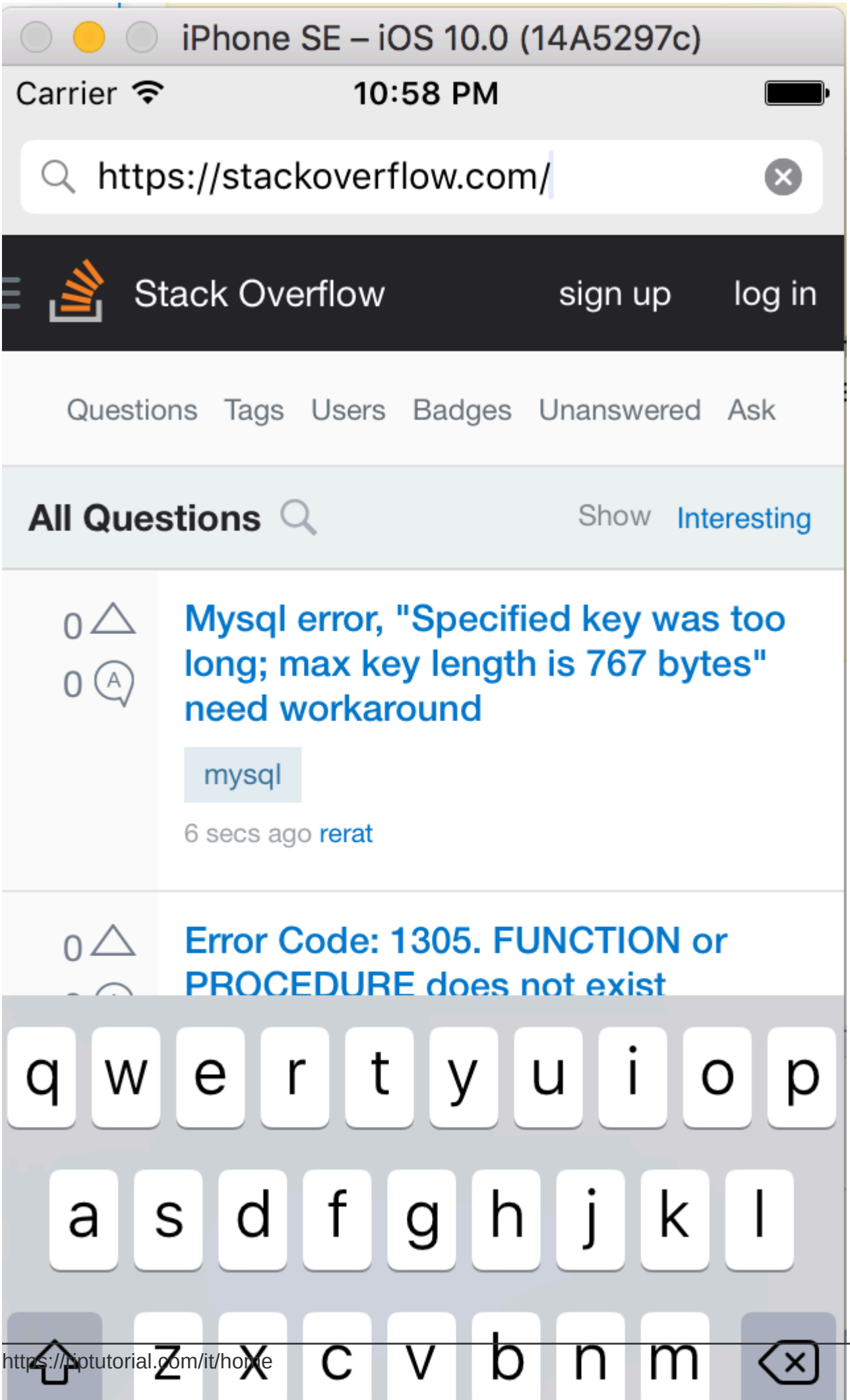
```
        //Setup which views will be constrained.

        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.
    }
}
```

Mostrare il pulsante GO personalizzato sulla tastiera:


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

più di una volta, `NSInvalidArgumentException` eccezione `NSInvalidArgumentException` .

Leggi WKWebView online: https://riptutorial.com/it/ios/topic/3602/wkwebview

Capitolo 210: Xcode Build & Archive From Command Line

Sintassi

- `xcodebuild [-project name.xcodeproj] -scheme schemename [[-destination destinationspecifier] ...] [-destination-timeout value] [-configuration configurationname] [-sdk [sdkfullpath | sdkname]] [action ...] [buildsetting=value ...] [-userdefault=value ...]`

Parametri

Opzione	Descrizione
-progetto	Costruisci il nome name.xcodeproj.
-schema	Obbligatorio se si costruisce uno spazio di lavoro.
-destinazione	Usa il dispositivo di destinazione
-configurazione	Usa la configurazione di build
-SDK	SDK specificato

Osservazioni

Esegui `xcodebuild` dalla directory che contiene il tuo progetto per creare un progetto Xcode. Per creare uno spazio di lavoro Xcode, è necessario passare entrambe le opzioni **-workspace** e **-scheme** per definire la build. I parametri dello schema controlleranno quali obiettivi sono costruiti e come sono costruiti, sebbene tu possa passare altre opzioni a xcodebuild per sovrascrivere alcuni parametri dello schema.

Examples

Costruisci e archivia

Costruire:

```
xcodebuild -exportArchive -exportFormat ipa \
    -archivePath "/Users/username/Desktop/MyiOSApp.xcarchive" \
    -exportPath "/Users/username/Desktop/MyiOSApp.ipa" \
    -exportProvisioningProfile "MyCompany Distribution Profile"
```

Archivio:

```
xcodebuild -project <ProjectName.xcodeproj>
    -scheme <ProjectName>
    -sdk iphonesimulator
    -configuration Debug
    -destination "platform=iOS Simulator,name=<Device>,OS=9.3"
    clean build
```

Leggi Xcode Build & Archive From Command Line online:
https://riptutorial.com/it/ios/topic/5027/xcode-build--amp--archive-from-command-line

Titoli di coda

S. No	Capitoli	Contributors
1	Inizia con 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	Accessibilità	Harshal Bhavsar, Justin, Ruby, Stephen Leppik, Zev Eisenberg
3	Acquisto in app	Cyril Ivar Garcia, Martin, rigdonmr, WMios
4	AFNetworking	4444, Mayank Patel, OhadM, Ruby
5	Aggiornamento dinamico di UIStackView	Harshal Bhavsar, Rahul, Stephen Leppik
6	AGGIUNTA DI UN'INTESTINA A BRIDGING SWIFT	yogesh wadhwa
7	Airdrop	FelixSFD, Md. Ibrahim Hassan
8	Alamofire	Alex Koshy, Josh Caswell, Sour LeangChhean, yogesh wadhwa
9	API 10 riconoscimento vocale	rohit90, Stephen Leppik
10	API di Google Places per iOS	Cyril Ivar Garcia, Vignan
11	App Transport Security (ATS)	breakingobstacles, D4ttatraya, esthepiking, FelixSFD, Mehul Chuahan, nathan
12	AppDelegate	CodeChanger, Oleh Zayats, Saumil Shah

13	ARC (conteggio di riferimento automatico)	4444, Irfan, John Militer, Ketan P, Tricertops
14	Architettura MVP	Oleh Zayats
15	attribuitoText in UILabel	vp2698
16	AutoLayout di UIScrollView	Aaron, Brandon, Shrikant K
17	AVPlayer e AVPlayerViewController	Bonnie, Chirag Desai, Gazi Alankus, Harshal Bhavsar, Konda Yadav, Stephen Leppik
18	AVSpeechSynthesizer	Ali Beadle, Bhumit Mehta, Harshal Bhavsar, Midhun MP, Stephen Leppik
19	Barra di navigazione	Md. Ibrahim Hassan, Mehul Chuahan
20	Bloccare	4444, animuson, Joshua, Mehul Chuahan, Ruby, Tamarous, user459460
21	Build Simulator	Durai Amuthan.H
22	CAAnimation	Bhavin Ramani, James P, Mr. Xcoder, Narendra Pandey, Rahul, Rob, Undo
23	Cache immagini online	Md. Ibrahim Hassan
24	CAGradientLayer	Bhavin Ramani, Harshal Bhavsar, Sam Fischer, Stephen Leppik, Undo
25	CALayer	Alistra, Dunja Lalic, HariKrishnan.P, Harshal Bhavsar, ignotusverum, iOS BadBoy, Kamil Harasimowicz, Luiz Henrique Guimaraes, Stephen Leppik, Suragch, Viktor Simkó, william205
26	Cambia colore barra di stato	Alex Rouse, danshevluk, Harshal Bhavsar, Mr. Xcoder, shim, Stephen Leppik, Steve Moser, william205, WMios
27	Caratteri personalizzati	Alexi, Dima Deplov, Harshal Bhavsar, Maddyゾゾ, njuri, Stephen Leppik, Tommie C.
28	Carica immagini asincrone	J.Paravicini
29	CAShapeLayer	Filip Radelic, HariKrishnan.P, Harshal Bhavsar, Narendra Pandey, Stephen Leppik
30	categorie	Faran Ghani, simple_code
31	Chain Blocks in una coda	StackUnderflow

	(con MKBlockQueue)	
32	Classi di dimensioni e adattabilità	Tim
33	Classifiche di GameCenter	4444, Cyril Ivar Garcia, Harshal Bhavsar, Stephen Leppik
34	CLLocation	amar, Duly Kinsky, FelixSFD, Siddharth Sunil, Sujania, That lazy iOS Guy , void, Zee
35	CloudKit	Seyyed Parsa Neshaei
36	codificabile	Ashish Kakkad
37	Compressione del contenuto / compressione dei contenuti in Autolayout	Mehul Chuahan
38	Concorrenza	Doc, Fonix, Juan Campa, Kevin DiTraglia, Tien
39	Configura i beacon con CoreBluetooth	Beto Caldas
40	Controllo della versione di iOS	Bhavin Ramani, byJeevan, James P, Joshua, njuri, Samuel Teferra, Sandy
41	Converti HTML in stringa NSAtribuita e viceversa	Md. Ibrahim Hassan
42	Converti NSAttributedString in UIImage	Md. Ibrahim Hassan
43	Core Graphics	Dunja Lalic, Josh Caswell, Seyyed Parsa Neshaei, Sunil Sharma, Unheilig
44	Core Location	Harshal Bhavsar, Mayuri R Talaviya, Mehul Chuahan, mtso, quant24, Stephen Leppik, sushant jagtap, william205
45	Core Motion	Md. Ibrahim Hassan, RamenChef
46	Core SpotLight in iOS	Md. Ibrahim Hassan
47	Corsia di sorpasso	J F, KrauseFx, SM18, tharkay
48	Crea il file .ipa da caricare su appstore con Applicationloader	Anuj Joshi
49	Crea un framework	Saeed-rz

	personalizzato in iOS	
50	Crea un video dalle immagini	Tiko
51	Creazione di PDF in iOS	Mansi Panchal, Narendra Pandey
52	Creazione di un ID app	yogesh wadhwa
53	CTCallCenter	MANI, Md. Ibrahim Hassan, OhadM
54	CydiaSubstrate tweak	gkpln3
55	Dati principali	Ankit chauhan, Md. Ibrahim Hassan
56	Deep linking in iOS	bryanjclark, Dunja Lalic, FelixSFD, sanman
57	Delegati multicast	Rahul
58	DispatchGroup	Brandon, Fonix
59	Estensione per Rich Push Notification - iOS 10.	Oleh Zayats
60	EventKit	Seyyed Parsa Neshaei
61	FacebookSDK	Brian, Harshal Bhavsar, Irfan, Mehul Chuahan, OhadM, Ravi Prakash Verma, Stephen Leppik
62	FileHandle	Nikhlesh Bagdiya
63	Filtri CoreImage	Md. Ibrahim Hassan
64	Firma del codice	HaemEternal
65	GameplayKit	BennX, Seyyed Parsa Neshaei
66	GCD (Grand Central Dispatch)	Andrea Antonioni, DS Dharma, Fonix, Md. Ibrahim Hassan, skyline75489
67	Gestione degli schemi URL	azimov, Brian, Dunja Lalic, Harshal Bhavsar, James P, Stephen Leppik
68	Gestire la tastiera	Alexander Tkachenko, Greg, Harshal Bhavsar, Mr. Xcoder, Richard Ash, Shog9, slxl, Stephen Leppik, Steve Moser, Suragch, V1P3R, william205, WMios
69	Gestire più ambienti utilizzando la macro	Tien
70	Grafico (Coreplot)	MarmiK, Md. Ibrahim Hassan

71	Healthkit	Md. Ibrahim Hassan
72	I / O di file di testo di base	Idan
73	IBeacon	amar, Arefly, Harshal Bhavsar, Stephen Leppik
74	IBOutlets	Fabio, SharkbaitWhohaha
75	Idiomi di inizializzazione	Jano
76	Il debug si blocca	NobodyNada
77	Imposta sfondo vista	Adriana Carelli, Andreas, Bhadresh Kathiriya, Harshal Bhavsar, Md. Ibrahim Hassan, user459460
78	Installazione di Carthage iOS	Md. Ibrahim Hassan
79	Integrazione con SqlCipher	Nirav
80	Interoperabilità Swift e Objective-C	Harshal Bhavsar, njuri, Stephen Leppik
81	iOS TTS	Ali Abbas, Stephen Leppik
82	iOS: implementazione di XMPP con framework Robbie Hanson	Saheb Roy
83	Istantanea di UIView	Bright Future, Darshit Shah, Md. Ibrahim Hassan, SpaceDog
84	Key Value Coding: Key Value Observation	D4ttatraya, Harshal Bhavsar, Mehul Chuahan, Mihriban Minaz, Mithrandir, Muhammad Zohaib Ehsan, Pärserk, sanman
85	Layout automatico	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
86	Le notifiche push	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
87	Linee guida per scegliere i	Phani Sai

	migliori modelli di architettura iOS	
88	Link universali	Harshal Bhavsar, Irfan, satheeshwaran, Stephen Leppik, Vineet Choudhary
89	Localizzazione	4444, animuson, Joshua, Ruby, WMios
90	Messaggistica FCM in Swift	Saeed-rz
91	Metodi personalizzati di selezione di UITableViewCells	Kamil Harasimowicz
92	MKDistanceFormatter	Harshal Bhavsar, Md. Ibrahim Hassan, Stephen Leppik, Undo
93	MKMapView	Arnon Rodrigues, Brian, FelixSFD, Harshal Bhavsar, Kosuke Ogawa, Mahesh, Mehul Thakkar, Ortwin Gentz, Reinier Melian, Stephen Leppik
94	Modalità di background	Seyyed Parsa Neshaei
95	Modalità e eventi di sfondo	Ashish Kakkad
96	ModelPresentationStyles	Dishant Kapadiya
97	MPMediaPickerDelegate	FelixSFD, George Lee
98	MPVolumeView	lostAtSeaJoshua
99	MVVM	JPetric
100	MyLayout	
101	Notifiche Rich	Koushik
102	NSArray	Krunal, user5553647
103	NSAttributedString	Bhavin Ramani, Harshal Bhavsar, Jinhuan Li, Kirit Modi, Luiz Henrique Guimaraes, Mansi Panchal, Stephen Leppik, Tim, Tim Ebenezer, Undo
104	NSBundle	wdywayne
105	NSData	Felipe Cypriano, maxkonovalov, Seyyed Parsa Neshaei
106	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
107	NSHTTPCookieStorage	balagurubaran
108	NSInvocation	Md. Ibrahim Hassan
109	NSNotificationCenter	Alex Kallam, Alex Koshy, Anand Nimje, Bence Pattogato, Bright Future, Ichthyocentaurs, Jacopo Penzo, James P, Kirit Modi, Tarun Seera
110	NSPredicate	Brendon Roberto, Joshua, Mehul Chuahan
111	NSTimer	AJ9, James P, Maddyツツ, Samuel Teferra, tfrank377, That lazy iOS Guy , Undo, william205
112	NSURL	Adnan Aftab, ApolloSoftware, tharkay
113	NSURLConnection	byJeevan
114	NSURLSession	bluey31, dasdom, dgatwood, Duly Kinsky, Harshal Bhavsar, Narendra Pandey, Otávio, R P, sage444, Stephen Leppik
115	NSUserActivity	Samuel Spencer
116	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
117	Objective-C Oggetti associati	Noam
118	OpenGL	Fonix
119	Operazioni a livello di app	midori
120	Panoramiche personalizzate dai file XIB	backslash-f, Code.Warrior, Harshal Bhavsar, idocode, Nirav Bhatt, Sharpkits Innovations, Stephen Leppik
121	Passaggio dei dati tra i controller di visualizzazione	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

122	Passaggio dei dati tra i controller di visualizzazione (con MessageBox-Concept)	StackUnderflow
123	plist iOS	SNarula
124	Portachiavi	abjurato, avojak, Matthew Seaman, Mehul Chuahan
125	Processo di invio di app	Nermin Sehic
126	Profilo con strumenti	Vinod Kumar
127	Quadro dei contatti	Md. Ibrahim Hassan, Seyyed Parsa Neshaei
128	Quadro XCTest - Test unitario	D4ttatraya, dasdom, Jan ATAC, Josh Brown, msohng, Raphael Silva, Seyyed Parsa Neshaei, Tarun Seera
129	Regno	subv3rsion
130	Rendi gli angoli di UIView selettivi arrotondati	Md. Ibrahim Hassan
131	Ridimensionamento di UIImage	Rahul
132	Riferimento CGContext	4444, Narendra Pandey
133	Rilevamento volti mediante CoreImage / OpenCV	Md. Ibrahim Hassan
134	Runtime in Objective-C	halil_g
135	Scanner di codici QR	Bluewings, Efraim Weiss
136	SDK AWS	OhadM
137	segues	Daniel Ormeño
138	Servizi Safari	Arnon Rodrigues, Harshal Bhavsar, Kilian Koeltzsch, Md. Ibrahim Hassan, Stephen Leppik
139	Sicurezza	D4ttatraya
140	Simulatore	Seyyed Parsa Neshaei
141	Simulazione della posizione utilizzando i file GPX iOS	Uma

142	Sirikit	Seyyed Parsa Neshaei
143	SLComposeViewController	Md. Ibrahim Hassan
144	Sottolineatura del testo UILabel	Md. Ibrahim Hassan
145	StoreKit	askielboe
146	storyboard	Harshal Bhavsar, Kirit Vaghela, Stephen Leppik, Tommie C.
147	Swift: modifica del rootViewController in AppDelegate per presentare il flusso principale o di accesso / onboarding	cleverbit
148	SWRevealViewController	Reinier Melian, tharkay
149	Taglia una creatura UII in un cerchio	Md. Ibrahim Hassan
150	Tastiera personalizzata	Md. Ibrahim Hassan
151	Test dell'interfaccia utente	P. Pawluś
152	Tipo dinamico	Alvin Abia, H. M. Madrone, Harshal Bhavsar, James P, Stephen Leppik
153	Tocco 3D	4444, Harshal Bhavsar, LinusGeffarth, Md. Ibrahim Hassan, Onur Tuna, Stephen Leppik, tobeiosdeveloper
154	Tutorial AirPrint in iOS	Md. Ibrahim Hassan
155	UIActivityViewController	Amandeep, Harshal Bhavsar, Stephen Leppik, Vivek Molkar
156	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
157	UIAppearance	azimov, Harshal Bhavsar, Stephen Leppik, Undo
158	UIBarButtonItem	Ahmed Khalaf, Dunja Lalic, hgwhittle, Suragch, william205

159	UIBezierPath	Bean, Igor Bidiniuc, Suragch, Teja Nandamuri
160	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
161	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
162	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
163	UIControl - Gestione degli eventi con i blocchi	Brandon
164	UIDatePicker	Pavel Gatilov
165	UIDevice	Bhavin Ramani, FelixSFD, Md. Ibrahim Hassan, Mehul Chuahan, Nef10, pableiros, Ramkumar chintala
166	UIFeedbackGenerator	beyowulf
167	UIFont	Mr. Xcoder
168	UIGestureRecognizer	Adam Preble, dannyzlo, Dunja Lalic, Harshal Bhavsar, John Leonardo, Josh Caswell, Md. Ibrahim Hassan, Ruby , Stephen Leppik, Sujania, Suragch, Undo
169	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
170	UIImagePickerController	Brian, stonybrooklyn, william205

171	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
172	Uikit Dynamics	beyowulf, Mark Stewart, Md. Ibrahim Hassan
173	Uikit Dynamics con UICollectionView	beyowulf
174	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
175	UILocalNotification	Bhumit Mehta, Brian, Byte1518, D4ttatraya, David, ElonChan, Harshal Bhavsar, hgwhittle, kamwysoc, KrishnaCA, rajesh sukumaran, Rex, Samuel Spencer, themathsrobot, tktsubota, william205, Wolverine, Xenon
176	UINavigationController	dasdom, Oleh Zayats, sage444, Suragch, william205, WMios
177	UIPageViewController	azimov, Bright Future, Harshal Bhavsar, Mayuri R Talaviya, Stephen Leppik, stonybrooklyn, Victor M

178	UIPheonix: framework UI facile, flessibile, dinamico e altamente scalabile	StackUnderflow
179	UIPickerView	FelixSFD, Hasintha Janka, MCMatan, Md. Ibrahim Hassan, Moritz, NinjaDeveloper
180	UIRefreshControl TableView	Md. Ibrahim Hassan, Mohammad Rana
181	UIScrollView	Bhavin Ramani, LinusGeffarth, maxkonovalov, Rex, sanman, Sujania, Sunil Sharma, Suragch, tharkay, torinpitchers
182	UIScrollView con figlio StackView	mourodrigo
183	UISearchController	Harshal Bhavsar, Mehul Chuahan, mtso, Stephen Leppik, Tarvo Mäesepp
184	UISegmentedControl	Kamil Harasimowicz
185	UISlider	Andreas, Md. Ibrahim Hassan
186	UISplitViewController	Cerbrus, Koushik
187	UIStackView	Anuj Joshi, danshevluk, Harshal Bhavsar, Kof, Lior Pollak , Sally, sasquatch, Stephen Leppik, william205
188	UIStoryboard	Adriana Carelli, Mr. Xcoder, Vignan
189	UISwitch	Bhavin Ramani, FelixSFD, Md. Ibrahim Hassan, Mr. Xcoder, RamenChef, Sujay
190	UITabBarController	Alexi, Anand Nimje, Cristina, Mehul Chuahan, Quantm, Srinija
191	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
192	UITableViewCell	Rahul
193	UITableViewController	Aju

194	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
195	UITextField delegato	Andreas, animuson, Md. Ibrahim Hassan, midori, Ruby
196	UITextField personalizzato	D4ttatraya
197	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
198	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
199	UIViewController	dasdom, Dunja Lalic, shim, Suragch, tassinari, william205
200	UIWebView	Allan Burleson, dchar4life80X, iOS BadBoy, J F, Julian135, KANGKANG, Kevin DiTraglia, maxkonovalov, Md. Ibrahim Hassan, Ortwin Gentz, Ramkumar chintala, Sunil Sharma
201	Utilizzando Image Aseets	D4ttatraya
202	UUID (Universally Unique Identifier)	Anand Nimje, FelixSFD, Harshal Bhavsar, James P, Mehul Chuahan, Rahul Vyas, Seyyed Parsa Neshaei, shim, Stephen Leppik, sushant jagtap
203	Valutazione della domanda / richiesta di revisione	Abhijit
204	Verifica della connettività di rete	ajmccall, breakingobstacles, Mick MacCallum, pableiros, sushant jagtap
205	WCSessionDelegate	pkc456
206	WKWebView	Brandon, byJeevan, Mahmoud Adam, Yevhen Dubinin

207	Xcode Build & Archive From Command Line	Kyle Decot, Shardul