eBook Gratuit

APPRENEZ iOS

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#ios

Table des matières

À propos

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

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

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

Chapitre 1: Démarrer avec iOS

Remarques

Remarques

1- Vous n'avez pas besoin d'un compte de développeur Apple pour développer des applications iOS. La documentation et les outils peuvent être téléchargés gratuitement avec votre identifiant Apple. Vous pouvez également signer et installer des applications sur *vos appareils personnels en* utilisant le même identifiant Apple. Si vous souhaitez distribuer ou vendre des applications sur l' App Store , vous devez inscrire le programme Apple Developer à partir de 99 USD (il s'agit du prix au moment de la rédaction et peut changer). Cela ajoutera également des incidents de support au niveau du code et des tests bêta pour vos applications via TestFlight.

2- La création d'un identifiant Apple sans carte de crédit nécessite un processus court . Si cela ne vous dérange pas d'associer un moyen de paiement à l'inscription, rendez-vous sur https://appleid.apple.com/

- Commencer à développer des applications iOS (Swift)
- Aide Xcode (y compris mise en route)
- Téléchargements (y compris Xcode si vous ne souhaitez pas passer par l'AppStore)

Balises liées au dépassement de pile

- xcode IDE (Integrated Development Environment) d'Apple pour développer des applications iOS et macOS
- swift-language L'une des principales langues que vous pouvez utiliser pour développer iOS.
- objective-c-language L'une des principales langues que vous pouvez utiliser pour développer iOS.
- cacao Une API Apple pour le développement sous iOS et macOS.
- sprite-kit Pour les graphiques animés 2D.
- données de base Pour stocker et récupérer des données relationnelles.

Versions

Version	Date de sortie
iPhone OS 2	2008-07-11
iPhone OS 3	2009-06-17
iOS 4	2010-06-08

Version	Date de sortie
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

Création d'une application à vue unique par défaut

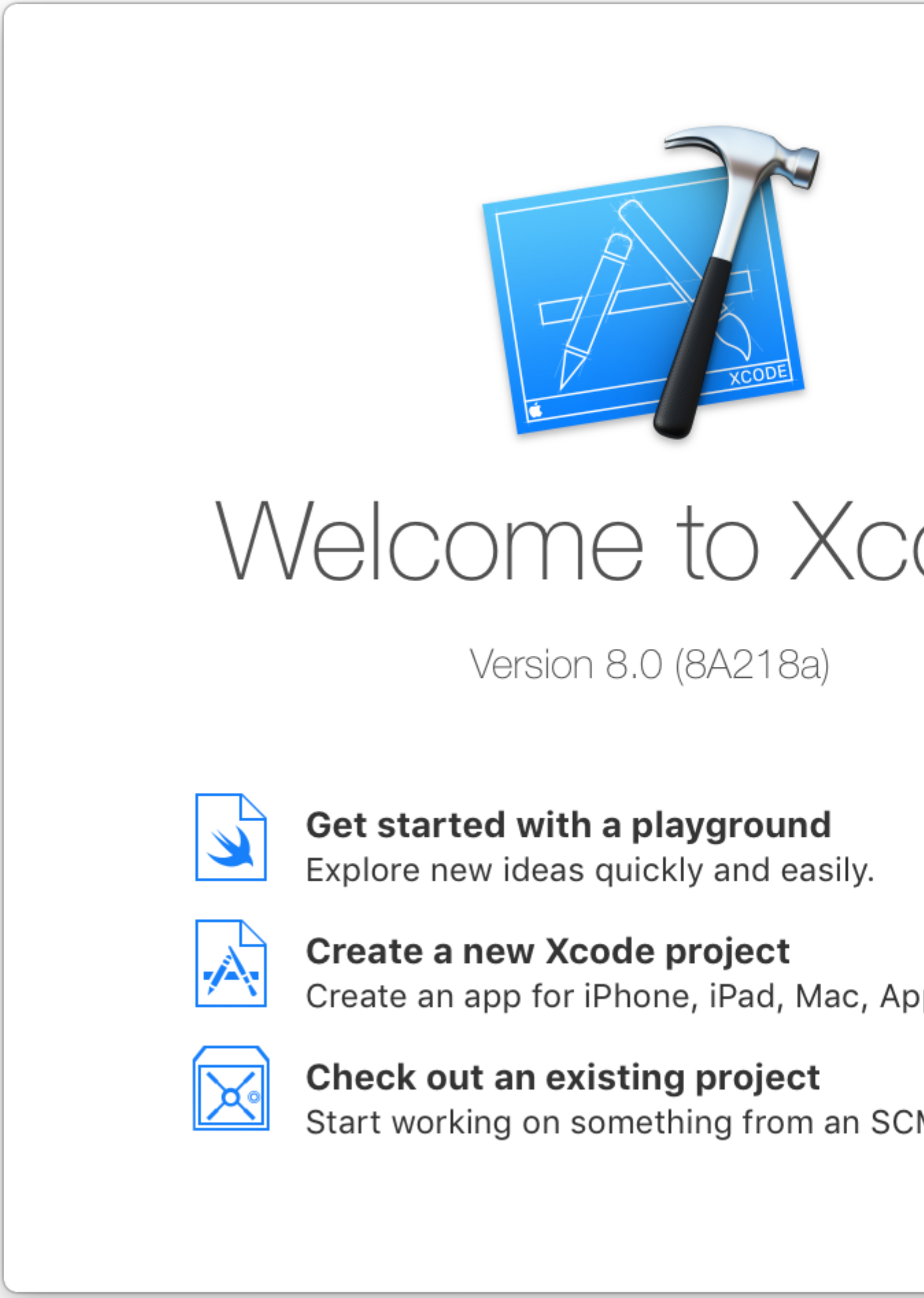
Pour développer une application pour iOS, vous devez commencer par une application appelée Xcode. Il existe d'autres outils alternatifs que vous pouvez utiliser, mais Xcode est l'outil officiel d'Apple. Notez, cependant, qu'il ne fonctionne que sur MacOS. La dernière version officielle est Xcode 8.3.3 avec Xcode 9 (actuellement en version bêta) qui devrait sortir plus tard cette année.

1. Démarrez votre Mac et installez Xcode depuis l'App Store s'il n'est pas déjà installé.

   (Si vous préférez ne pas utiliser l'App Store ou si vous rencontrez des problèmes, vous

pouvez également télécharger Xcode depuis le site Web Apple Developer , mais assurez-vous de sélectionner la version la plus récente et **non** la version bêta.)

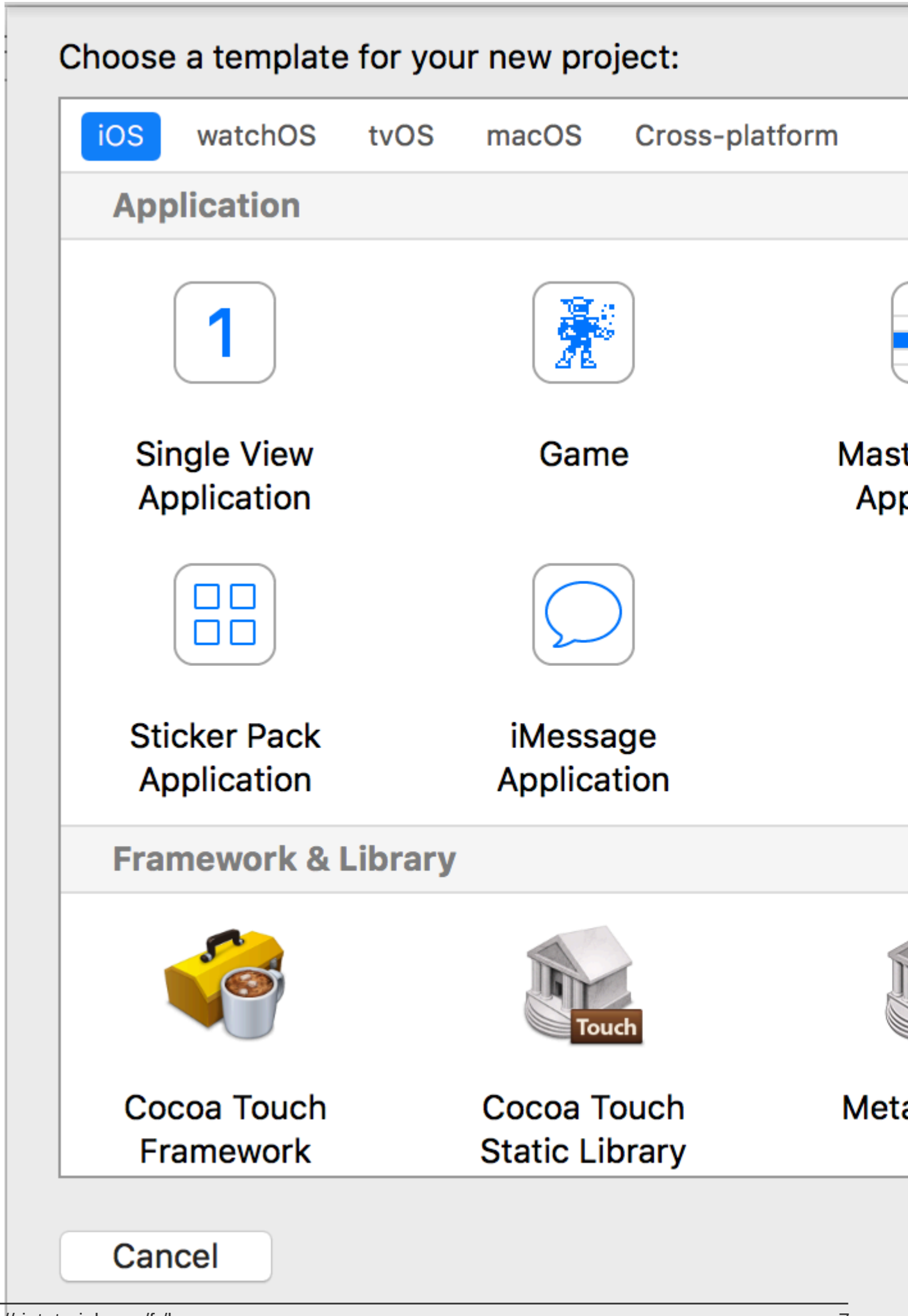
2. Ouvrez Xcode. La fenêtre suivante s'ouvrira:


XCODE
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 fenêtre vous présente les options suivantes:

- **Démarrer avec un terrain de jeu:** il a été introduit avec le langage Swift et Xcode 6. C'est un espace interactif qui peut être utilisé pour écrire de petits morceaux de code afin de vérifier les changements au moment de l'exécution. C'est un excellent moyen pour les apprenants de Swift de découvrir les nouvelles fonctionnalités de Swift.
  il a été introduit avec le langage Swift et Xcode 6. C'est un espace interactif qui peut être utilisé pour écrire de petits morceaux de code afin de vérifier les changements au moment de l'exécution. C'est un excellent moyen pour les apprenants de Swift de découvrir les nouvelles fonctionnalités de Swift.
- **Créer un nouveau projet Xcode:** *Choisissez cette option* , qui crée un nouveau projet avec la configuration par défaut.
- **Extraire un projet existant:** Ceci permet d'extraire un projet d'un emplacement de référentiel, par exemple, d'extraire un projet de SVN.

3. Sélectionnez la deuxième option **Créer un nouveau projet Xcode** et Xcode vous demandera de faire une configuration initiale du projet:


Choose a template for your new project:
iOS
watchOS
tvOS
macOS
Cross-platform
Application
1
Single View Application
Game
Sticker Pack Application
iMessage Application
Framework & Library
Touch
Cocoa Touch Framework
Cocoa Touch Static Library
Cancel

Cet assistant est utilisé pour sélectionner votre modèle de projet. Il y a 5 options:

- **iOS:** Utilisé pour créer des applications iOS, des bibliothèques et des frameworks
- **watchOS:** Utilisé pour créer des applications watchOS, des bibliothèques et des frameworks
  Utilisé pour créer des applications watchOS, des bibliothèques et des frameworks
- **tvOS:** utilisé pour créer des applications, bibliothèques et frameworks tvOS
- **macOS:** Utilisé pour créer des applications macOS, des bibliothèques, des frameworks, des packages, des AppleScripts, etc.
- **Cross-platform:** utilisé pour créer des applications, des modèles et des contenus d'achat intégrés à plusieurs plates-formes

Vous pouvez voir qu'il existe de nombreux modèles différents pour votre application. Ces modèles sont utiles pour stimuler votre développement; ils sont pré-construits avec des configurations de base comme des interfaces utilisateur et des fichiers de classes.

Ici, nous utiliserons la première option, **iOS** .

1. Application Master-Detail:

   Ce modèle contient une interface maître / détail combinée: le maître contient des objets liés à l'interface de détail. La sélection d'objets dans le maître changera l'interface de détails. Vous pouvez voir ce type d'interface utilisateur dans les applications Paramètres, Notes et Contacts de l'iPad.

2. Application basée sur la page:

   Ce modèle est utilisé pour créer l'application basée sur une page. Les pages sont des vues différentes détenues par un conteneur.

3. Application à vue unique:

   Ceci est un modèle de développement d'application normal. C'est bon pour les débutants d'apprendre le flux des applications.

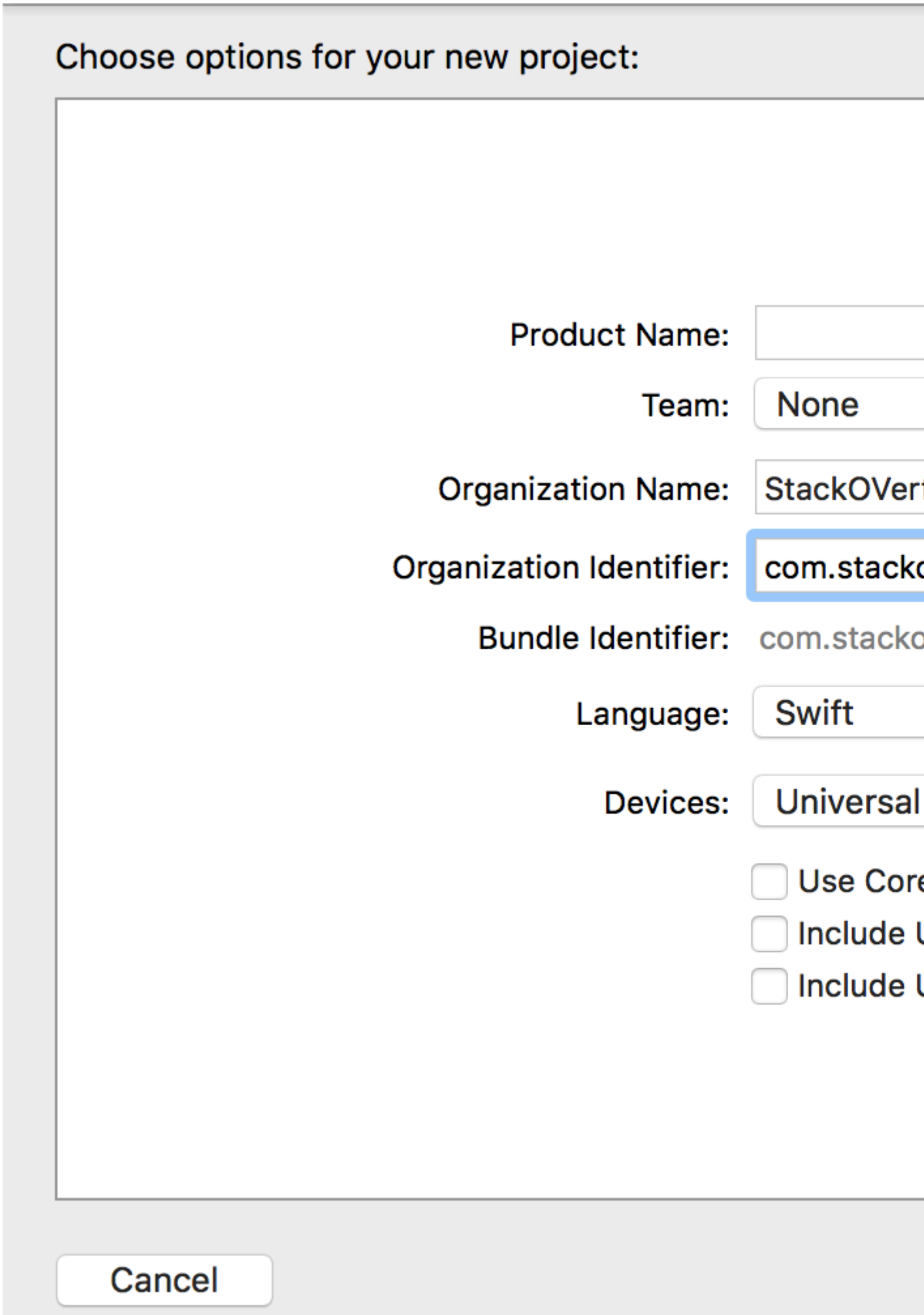
4. Application à onglets:

   Ce modèle crée des onglets dans la partie inférieure d'une application. Chaque onglet a une interface utilisateur différente et un flux de navigation différent. Vous pouvez voir ce modèle utilisé dans des applications telles que Clock, iTunes Store, iBooks et App Store.

5. Jeu:

   Ceci est un point de départ pour le développement de jeux. Vous pouvez aller plus loin avec les technologies de jeu telles que SceneKit, SpriteKit, OpenGL ES et Metal.

4. Dans cet exemple, nous allons commencer avec l' **application Single View**


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

L'assistant vous aide à définir les propriétés du projet:

- **Nom du produit:** Nom du projet / de l'application
- **Nom** de l'organisation **:** Nom de l'organisation dans laquelle vous êtes impliqué
- **Identifiant de l'organisation: identifiant** unique de l'organisation utilisé dans l'identifiant de l'ensemble. Il est recommandé de suivre la notation inversée du service de nom de domaine.
  unique de l'organisation utilisé dans l'identifiant de l'ensemble. Il est recommandé de suivre la notation inversée du service de nom de domaine.
- **Identifiant du lot:** *Ce champ est très important.* Il est basé sur le nom de votre projet et l'identifiant de l'organisation, choisissez judicieusement. L'identifiant de l'ensemble sera utilisé ultérieurement pour installer l'application sur un périphérique et télécharger l'application sur iTunes Connect (l'endroit où nous téléchargeons les applications à publier sur l'App Store). C'est une clé unique pour identifier votre application.
- **Langue:** Le langage de programmation que vous souhaitez utiliser. Ici, vous pouvez changer Objective-C en Swift s'il n'est pas sélectionné.
- **Périphériques: Périphériques** pris en charge pour votre application pouvant être modifiés ultérieurement. Il montre iPhone, iPad et Universal. Les applications universelles prennent en charge les périphériques iPhone et iPad. Il est recommandé de sélectionner cette option lorsqu'il n'est pas nécessaire d'exécuter l'application sur un seul type de périphérique.
- **Utiliser les données de base:** Si vous souhaitez utiliser le modèle de données de base dans votre projet, marquez-le comme sélectionné et il créera un fichier pour le fichier `.xcdatamodel` . Vous pouvez également ajouter ce fichier ultérieurement si vous ne le savez pas à l'avance.
- **Inclure les tests unitaires:** cette option configure la cible de test unitaire et crée des classes pour les tests unitaires.
- **Inclure le test de l'interface utilisateur:** cela configure la cible de test de l'interface utilisateur et crée des classes pour le test de l'interface utilisateur

Cliquez sur **Suivant** et il vous demandera un emplacement où vous souhaitez créer un répertoire de projet.
Cliquez sur **Créer** et vous verrez l'interface utilisateur Xcode avec une configuration de projet déjà définie. Vous pouvez voir quelques classes et fichiers Storyboard.

Ceci est un modèle de base pour une application à vue unique.

En haut à gauche de la fenêtre, vérifiez qu'un simulateur est sélectionné (par exemple "iPhone 6", comme illustré ici), puis appuyez sur le bouton RUN triangulaire.

5. Une nouvelle application s'ouvrira — Simulateur (cela peut prendre un certain temps la première fois que vous l'exécutez et vous devrez peut-être essayer deux fois si vous voyez une erreur la première fois). Cette application nous fournit une simulation de périphérique pour les applications créées. Cela ressemble presque à un vrai appareil! Il contient des

applications comme un appareil réel. Vous pouvez simuler des orientations, un emplacement, un geste de secousse, des avertissements de mémoire, une barre d'état en cours d'appel, un toucher du doigt, un verrouillage, un redémarrage, un retour à la maison, etc.

Vous verrez une application blanche car nous n’avons pas encore apporté de modifications au modèle.

Alors commencez votre propre. C'est une longue course et il y a beaucoup de nouvelles opportunités qui vous attendent!

Si vous ne savez pas où aller, essayez le tutoriel « Jump Right In » d'Apple. Vous avez déjà effectué les premières étapes et êtes donc en avance.

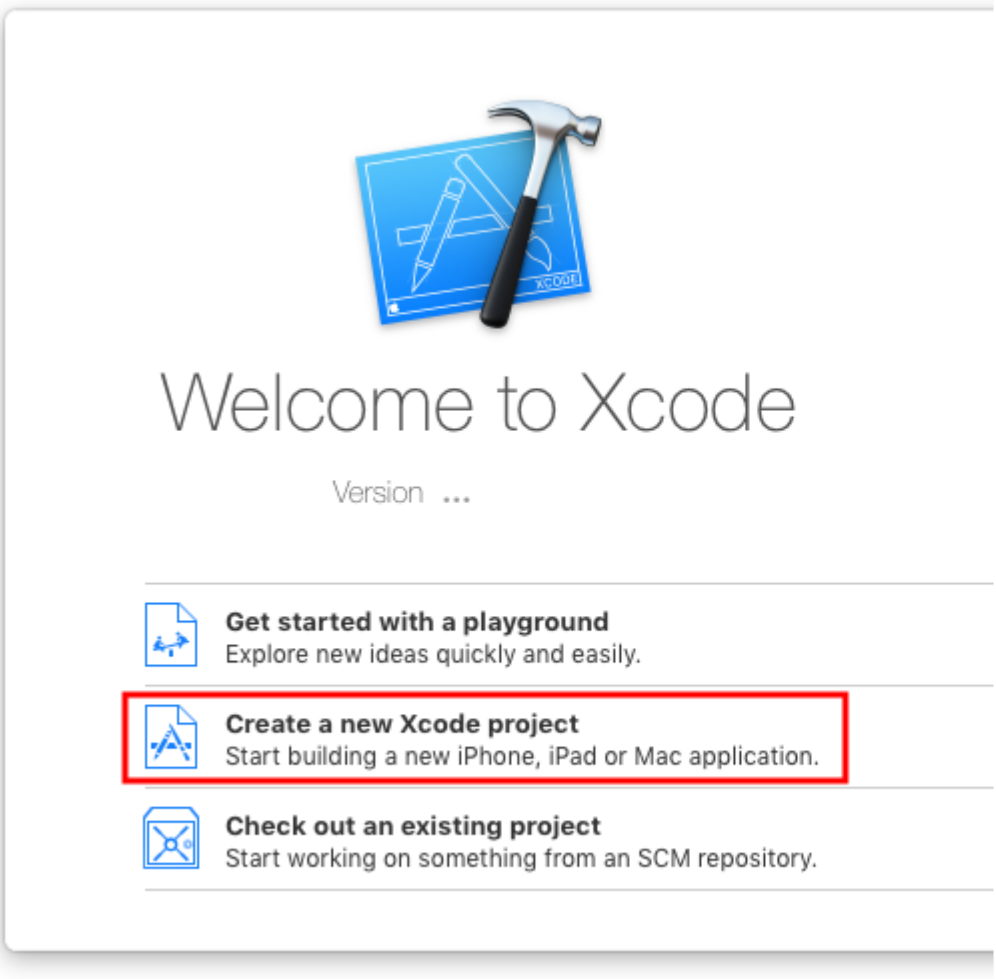
Bonjour le monde

Après avoir configuré Xcode, il n'est pas difficile de faire fonctionner votre premier iOS.
Dans l'exemple suivant, nous allons:

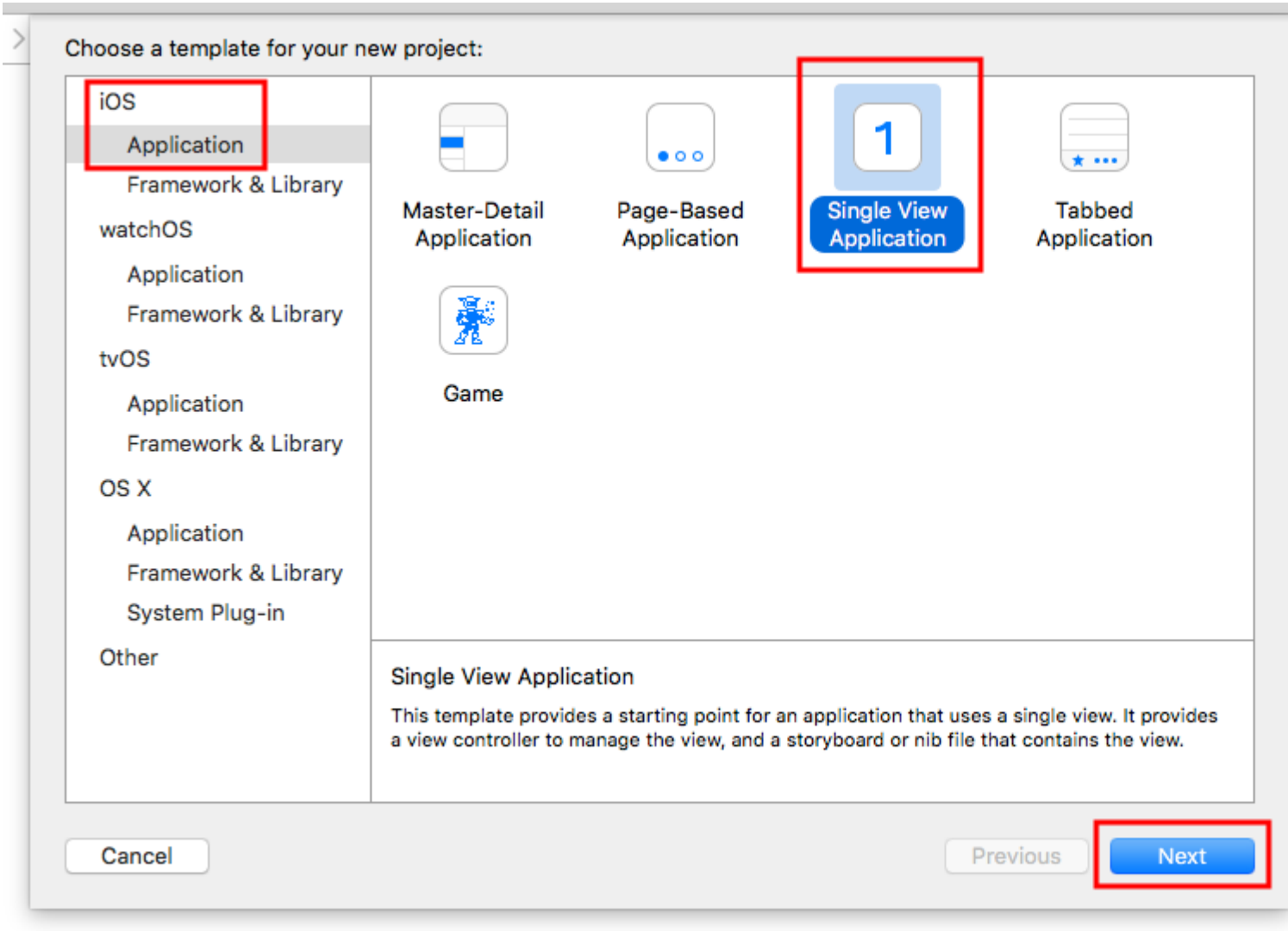
- Lancer un nouveau projet
- Ajouter une étiquette
- Impression du message sur la console.
- Exécuter dans le simulateur

Lancer un nouveau projet

Lorsque l'écran de bienvenue Xcode apparaît, choisissez **Créer un nouveau projet Xcode** .
Sinon, vous pouvez faire **Fichier> Nouveau> Projet ... à** partir du menu Xcode si vous l’avez déjà ouvert.

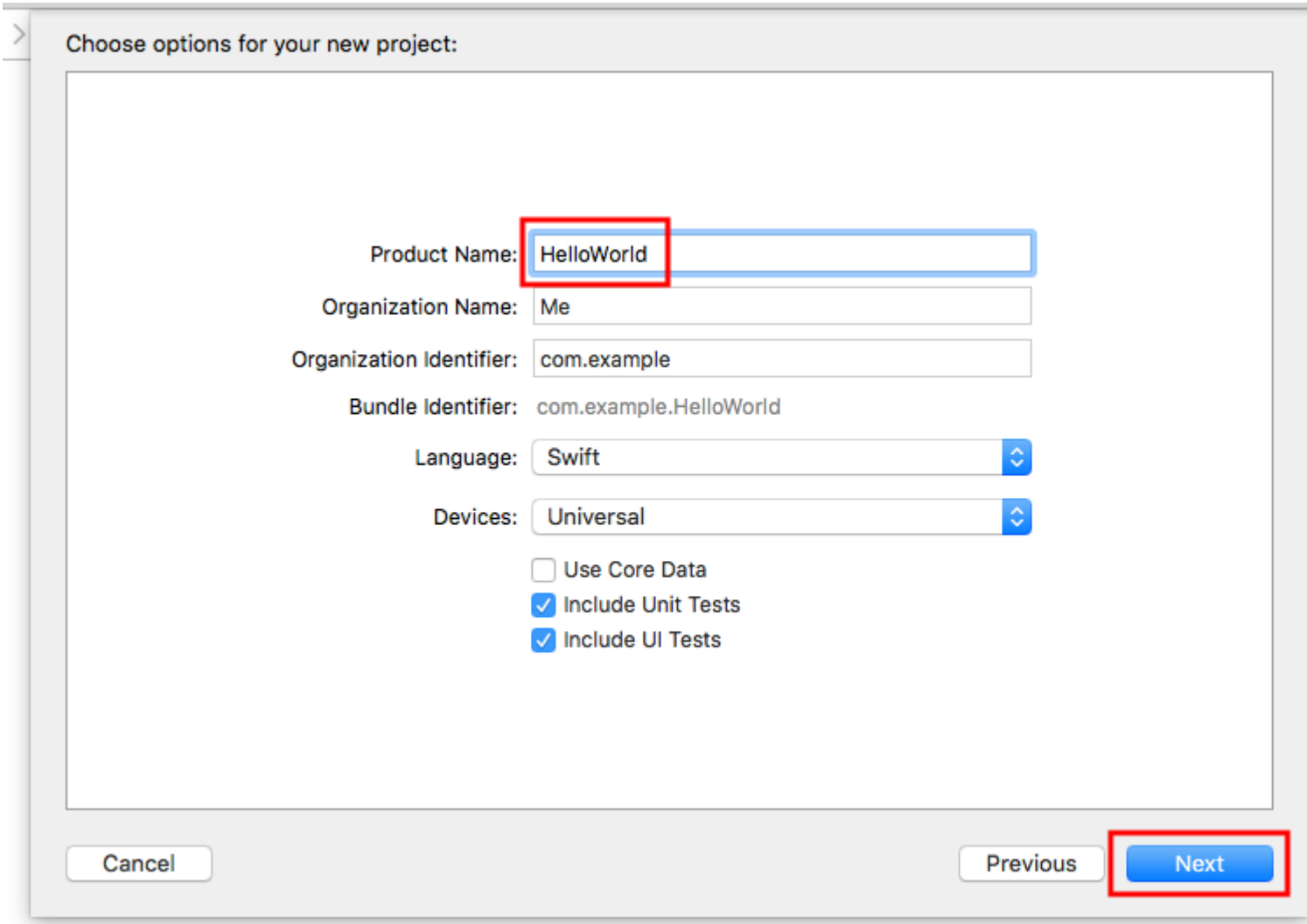


Choisissez une **application à vue unique** et cliquez sur **Suivant** .

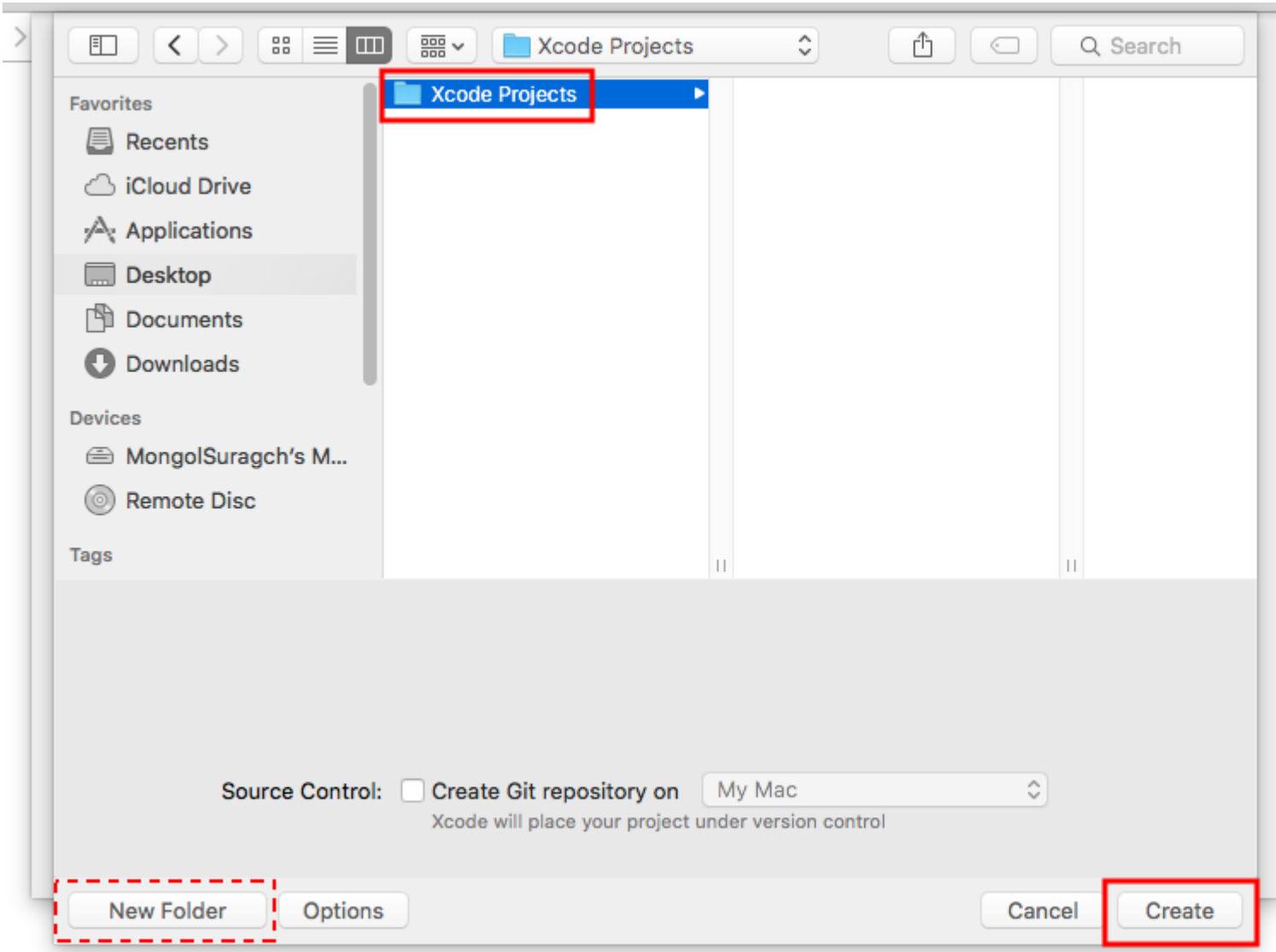


Écrivez "HelloWorld" pour le **nom** du **produit** (ou ce que vous voulez vraiment) et sous **Language** , assurez-vous que **Swift** est sélectionné.

- **Universal** signifie que votre application s'exécutera sur iPhone et iPad.
- **Utiliser les données de base** fait référence au stockage de données persistant, qui n'est pas nécessaire dans notre application Hello World.
- Nous ne ferons pas de **tests unitaires** ou de **tests d' interface** dans cet exemple, mais cela ne fait pas de mal de prendre l'habitude de les ajouter.



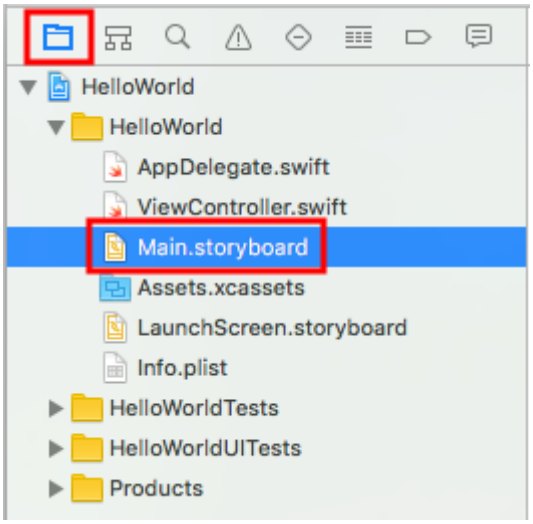
Choisissez un dossier existant ou créez-en un nouveau pour enregistrer vos projets Xcode. Ce sera la valeur par défaut dans le futur. Nous en avons créé un appelé "Projets Xcode". Puis cliquez sur **Créer** . Vous pouvez sélectionner le contrôle de source si vous le souhaitez (utilisé lors de la synchronisation avec des sites comme GitHub), mais nous n'en aurons pas besoin dans cet exemple.



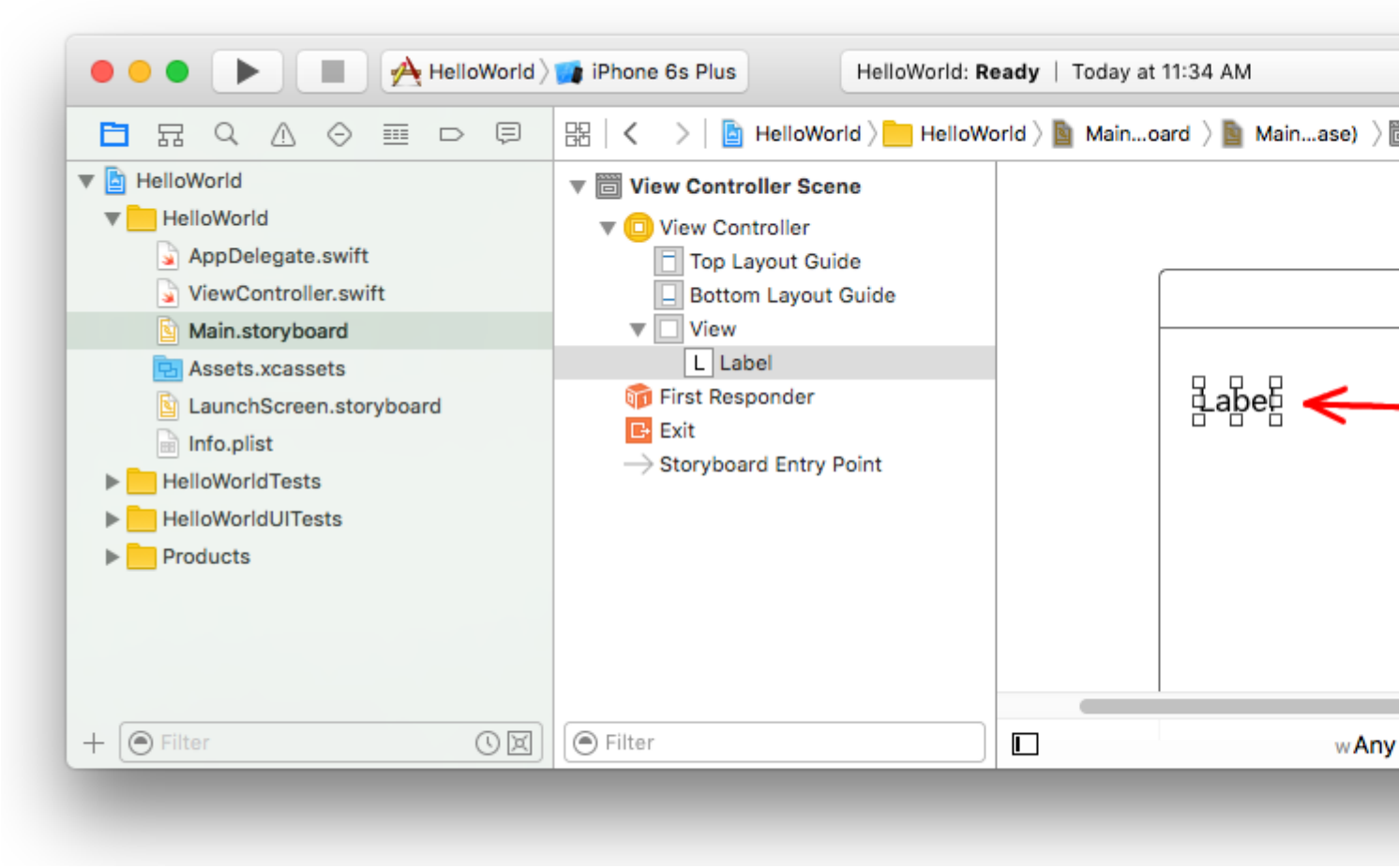
Ajouter une étiquette

C'est la structure de fichier d'un projet Xcode.

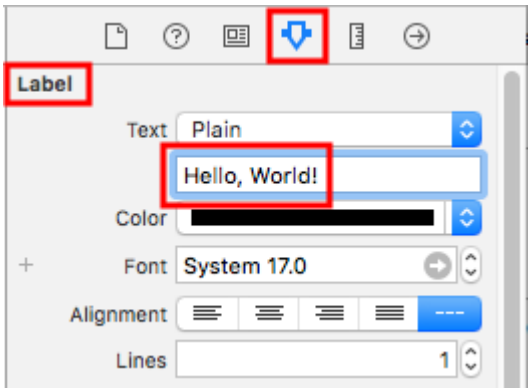
Sélectionnez *Main.storyboard* dans le navigateur de projet.



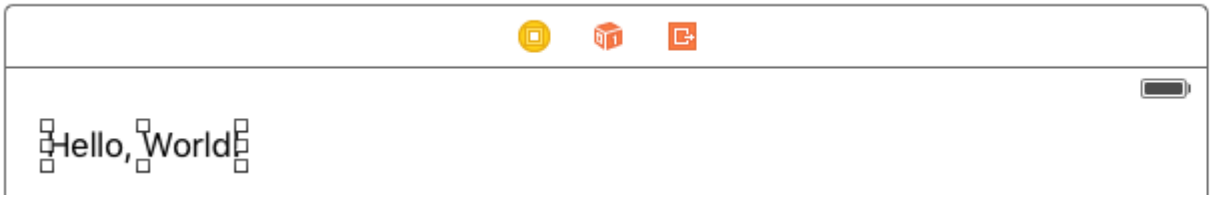
Tapez "label" dans le champ de recherche de la bibliothèque d'objets en bas à droite de Xcode. `UILabel` ensuite glisser l' `UILabel` sur le storyboard View Controller. Placez-le généralement dans la région du coin supérieur gauche.



Assurez-vous que l'étiquette est sélectionnée sur le storyboard, puis dans l' **inspecteur d'attributs** , remplacez le texte par "Hello, World!" Vous devrez ensuite redimensionner et repositionner l'étiquette sur le storyboard, car la longueur du texte est maintenant plus longue.

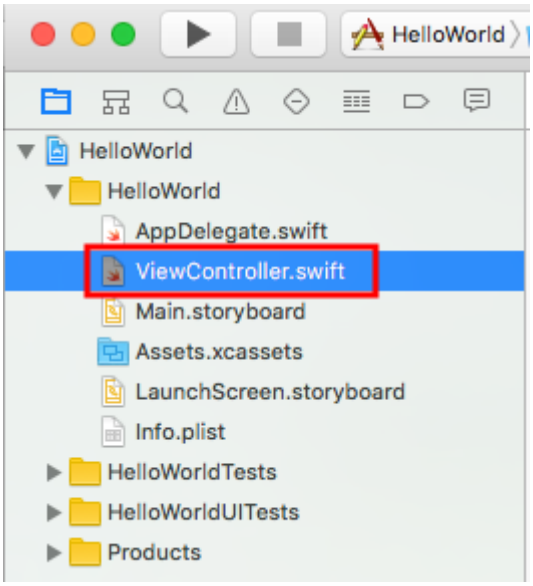


Vous pouvez également double-cliquer sur l'étiquette du storyboard pour l'éditer afin qu'elle soit "Hello, World!". En tout cas, le storyboard devrait ressembler à ceci:



Ajout de code

Sélectionnez *ViewController.swift* dans le navigateur de projet.



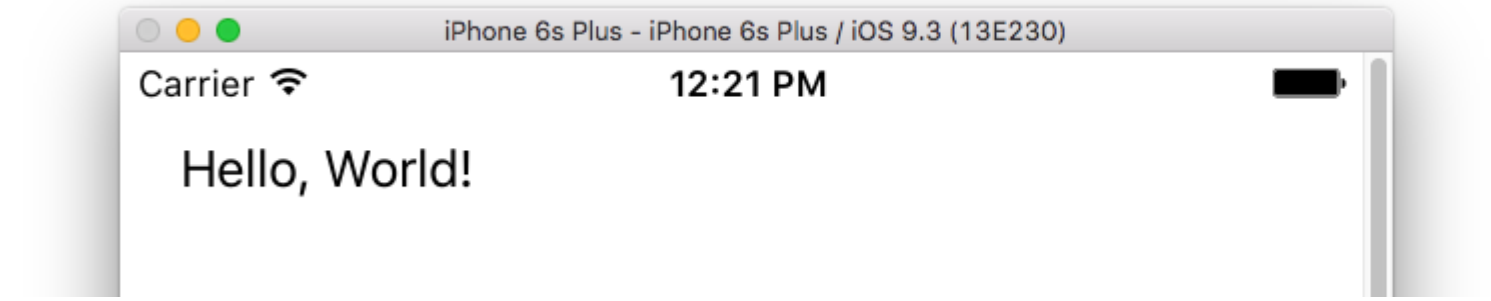
Ajoutez `print("Successfully created my first iOS application.") viewDidLoad() print("Successfully created my first iOS application.")` à la méthode `viewDidLoad()` . Ça devrait ressembler a quelque chose comme ca.

```
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.
    }
}
```

Lancer l'application dans le simulateur

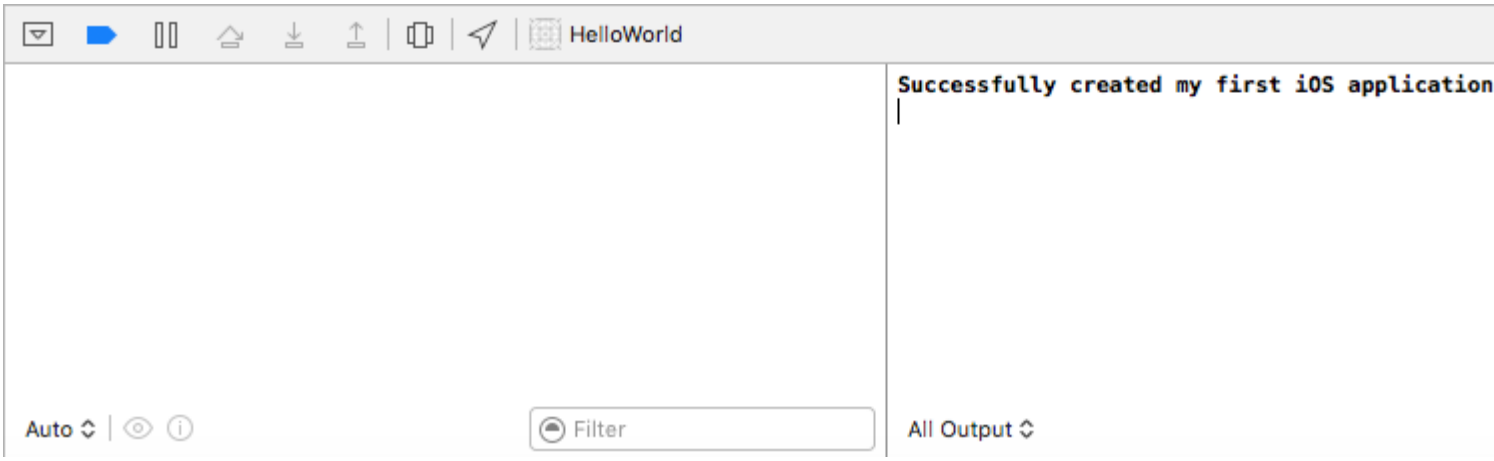
Appuyez sur le bouton Exécuter pour créer et exécuter l'application. Dans cet exemple, le périphérique de simulation actuel (appelé "schéma") est utilisé par défaut sur l'iPhone 6s Plus. Les versions les plus récentes de Xcode seront par défaut des schémas plus récents. Vous pouvez également choisir d'autres schémas en cliquant sur le nom. Nous allons rester avec le défaut.

Le simulateur mettra du temps à démarrer dès la première exécution. Une fois en cours d'exécution, cela devrait ressembler à ceci:



Dans le menu du simulateur, vous pouvez choisir **Fenêtre> Echelle** pour la réduire, ou appuyer sur cmd + 1/2/3/4/5 pour une échelle de 100% / 75% / 50% / 33% / 25% respectivement.

La zone de débogage Xcode (en bas) doit également avoir imprimé "Création réussie de ma première application iOS". à la console. "Création réussie de ma première application iOS" message est la chaîne que vous avez imprimée par programmation dans la partie **Ajouter du code** .

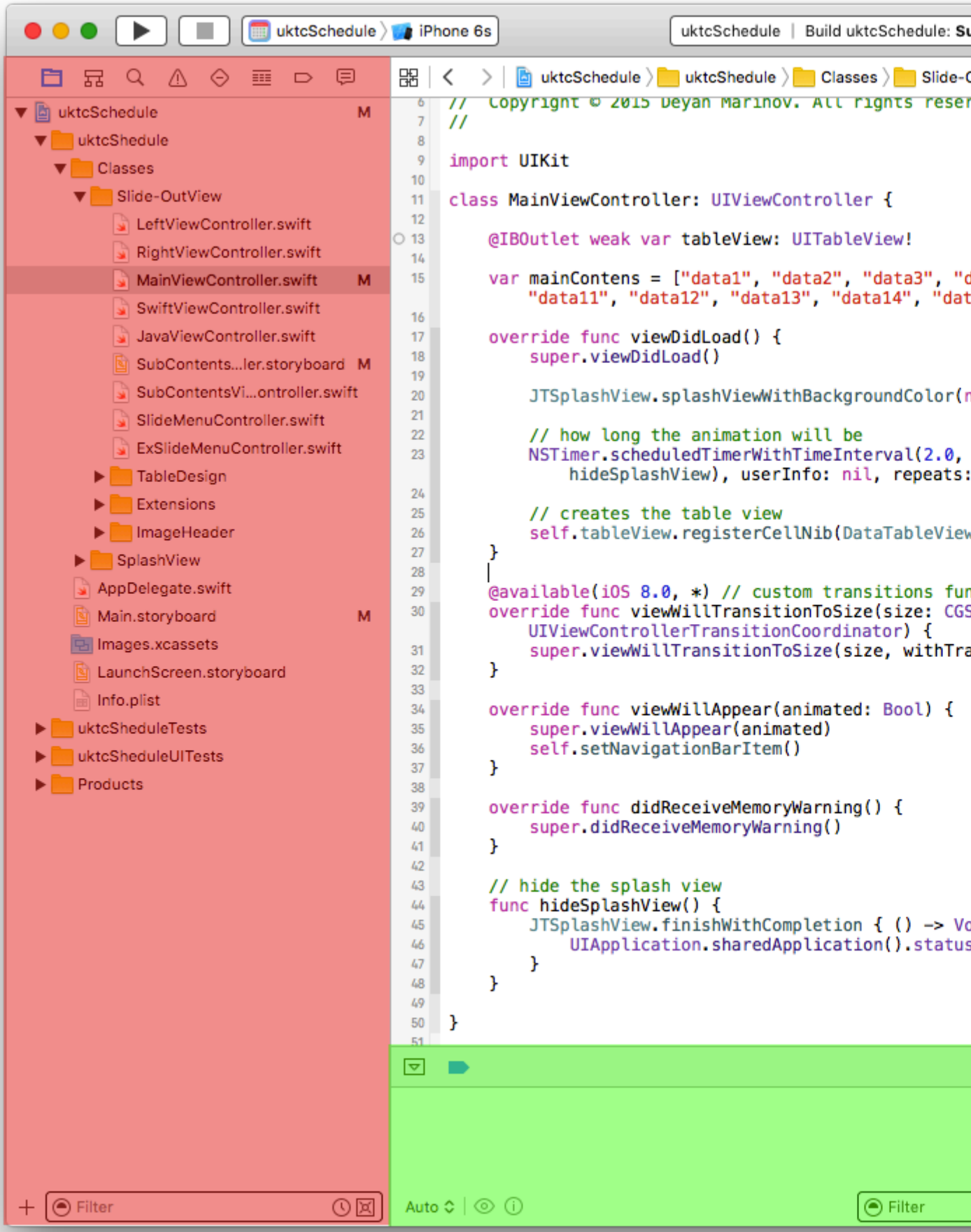


Continuer

Vous devriez en savoir plus sur les contraintes de mise en page automatique. Celles-ci vous aident à positionner vos contrôles sur le storyboard de manière à leur donner une bonne apparence, quelle que soit la taille et l'orientation de l'appareil.

Interface Xcode

Dans Xcode, vous disposez de trois zones de travail distinctes: les navigateurs (en rouge), la zone de débogage (en vert) et les utilitaires (en bleu).


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 fenêtre de l'espace de travail comprend toujours la zone de l'éditeur. Lorsque vous sélectionnez un fichier dans votre projet, son contenu apparaît dans la zone de l'éditeur, où Xcode ouvre le fichier dans un éditeur approprié. Par exemple, dans l'image ci-dessus, la zone de l'éditeur MainViewController.swift, un fichier de code rapide sélectionné dans la zone Navigateur située à gauche de la fenêtre de l'espace de travail.

Zone de navigation

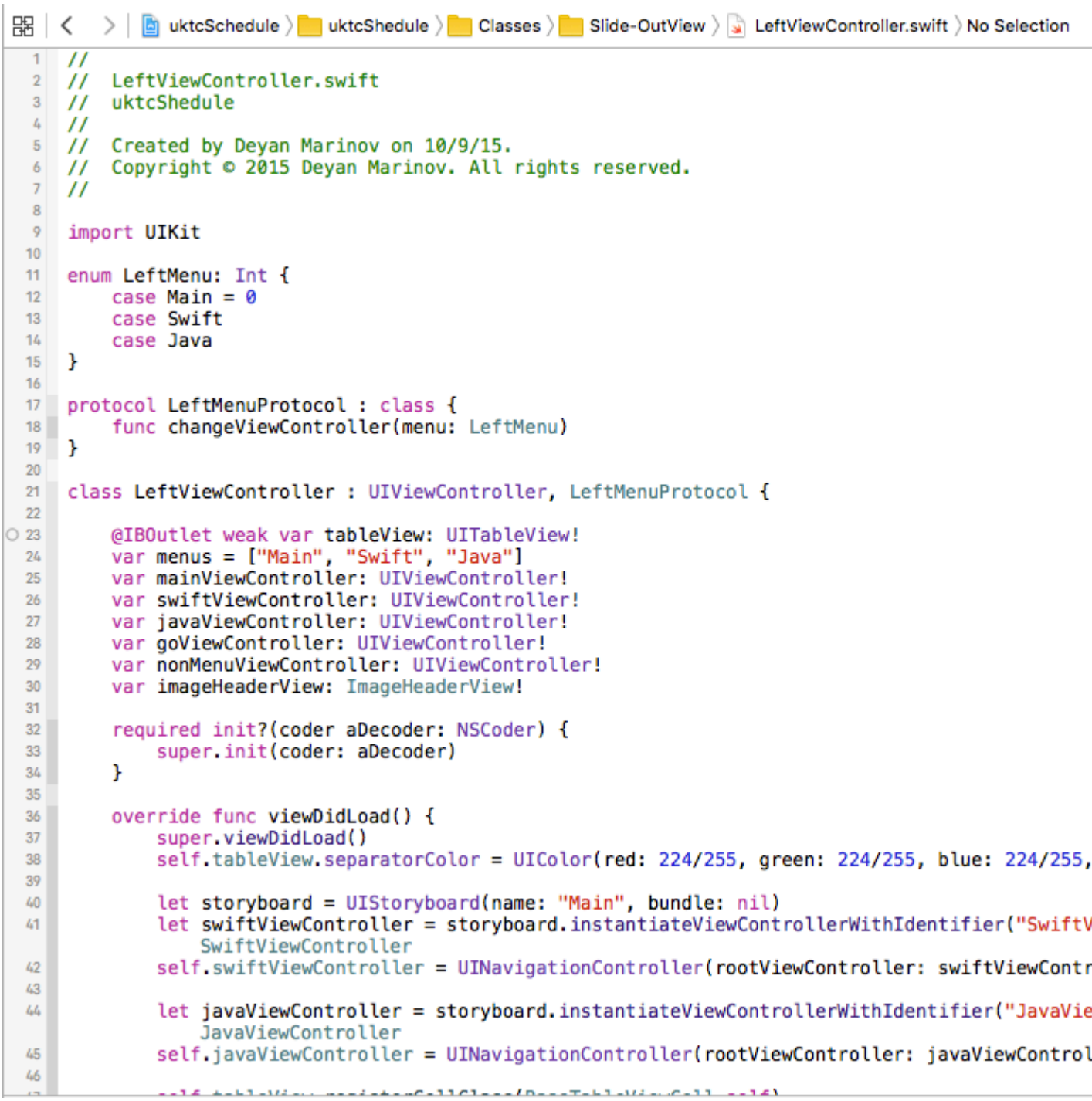
La fenêtre de navigation contient les huit options suivantes:

- **Navigateur de projet.** Ajoutez, supprimez, groupez et gérez autrement les fichiers de votre projet ou choisissez un fichier pour afficher ou modifier son contenu dans la zone de l'éditeur.
- **Navigateur de symboles.** Parcourez les symboles de votre projet sous forme de liste ou de hiérarchie. Les boutons situés à gauche de la barre de filtres vous permettent de limiter les symboles affichés à une combinaison de classes et de protocoles uniquement, uniquement des symboles dans votre projet ou uniquement des conteneurs.
- **Rechercher un navigateur** Utilisez les options de recherche et les filtres pour trouver rapidement une chaîne dans votre projet.
- **Navigateur de problème.** Affichez des problèmes tels que les diagnostics, les avertissements et les erreurs détectés lors de l'ouverture, de l'analyse et de la création de votre projet.
- **Test du navigateur.** Créer, gérer, exécuter et examiner des tests unitaires.
- **Navigateur de débogage.** Examinez les threads en cours d'exécution et les informations associées sur la pile à un moment ou à un moment précis pendant l'exécution du programme.
- **Navigateur de points d'arrêt.** Ajustez les points d'arrêt en spécifiant des caractéristiques telles que les conditions de déclenchement.
- **Navigateur de rapports.** Affichez l'historique de vos tâches de construction, d'exécution, de débogage, d'intégration continue et de contrôle de code source.

Les éditeurs

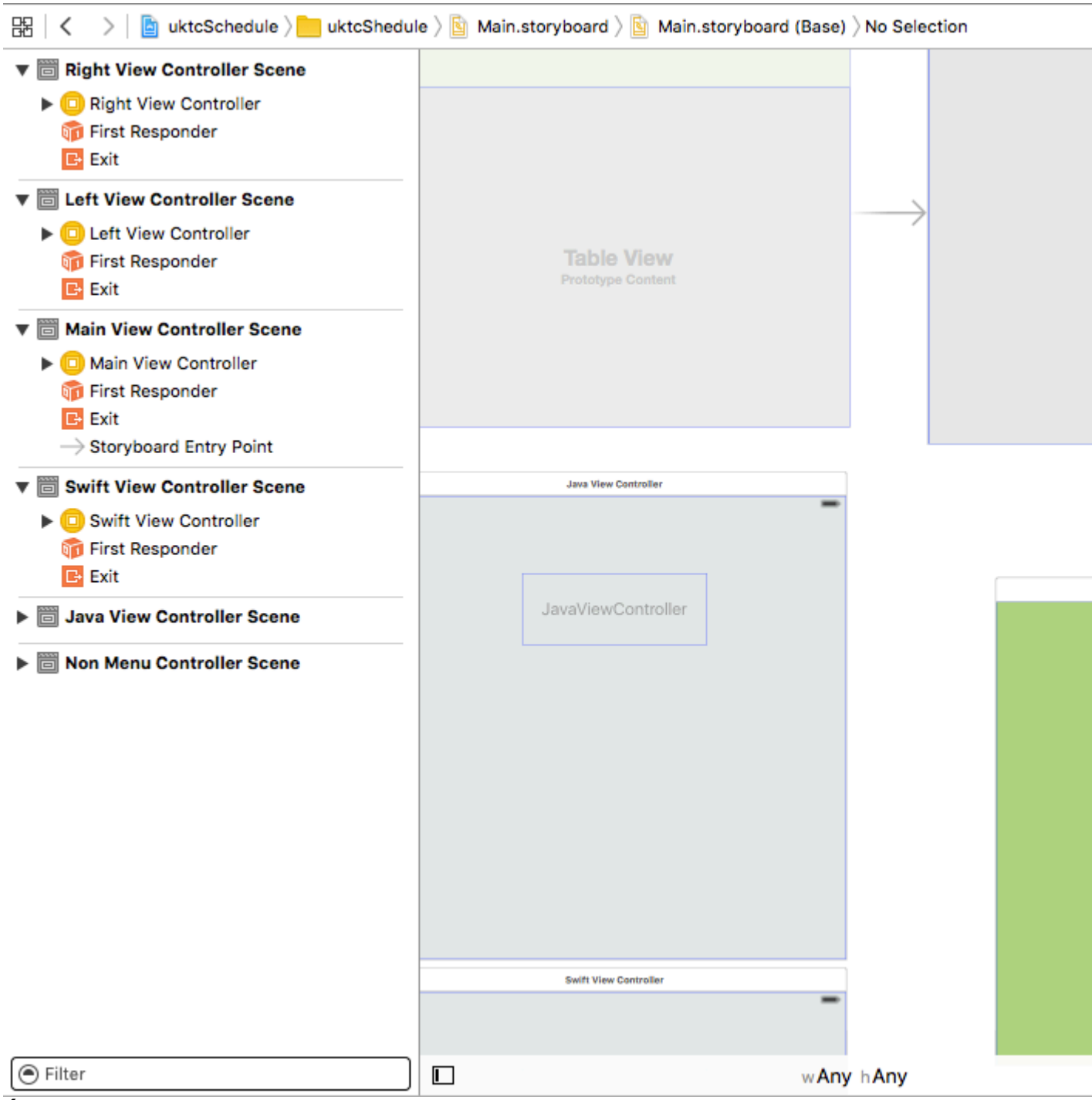
La plupart du travail de développement dans Xcode se produit dans la zone de l'éditeur, la zone principale qui est toujours visible dans la fenêtre de l'espace de travail. Les éditeurs que vous utilisez le plus souvent sont:

- **Éditeur source** Écrivez et modifiez le code source.

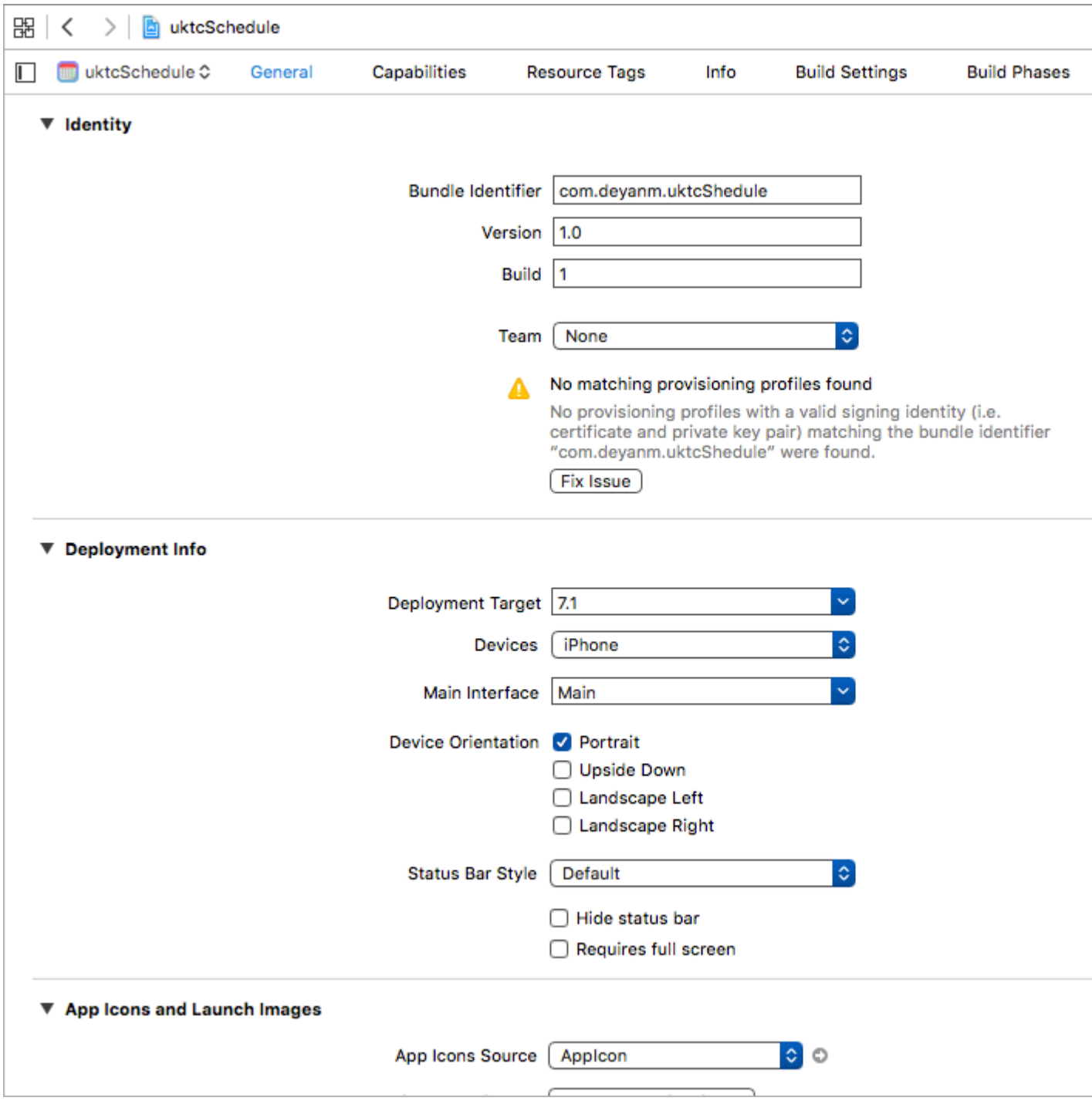



```swift
//
//  LeftViewController.swift
//  uktcShedule
//
//  Created by Deyan Marinov on 10/9/15.
//  Copyright © 2015 Deyan Marinov. All rights reserved.
//

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.** Créez et éditez graphiquement des fichiers d'interface utilisateur.



- **Éditeur de projet** Affichez et modifiez la manière dont vos applications doivent être créées, par exemple en spécifiant des options de construction, des architectures cibles et des droits d'accès aux applications.



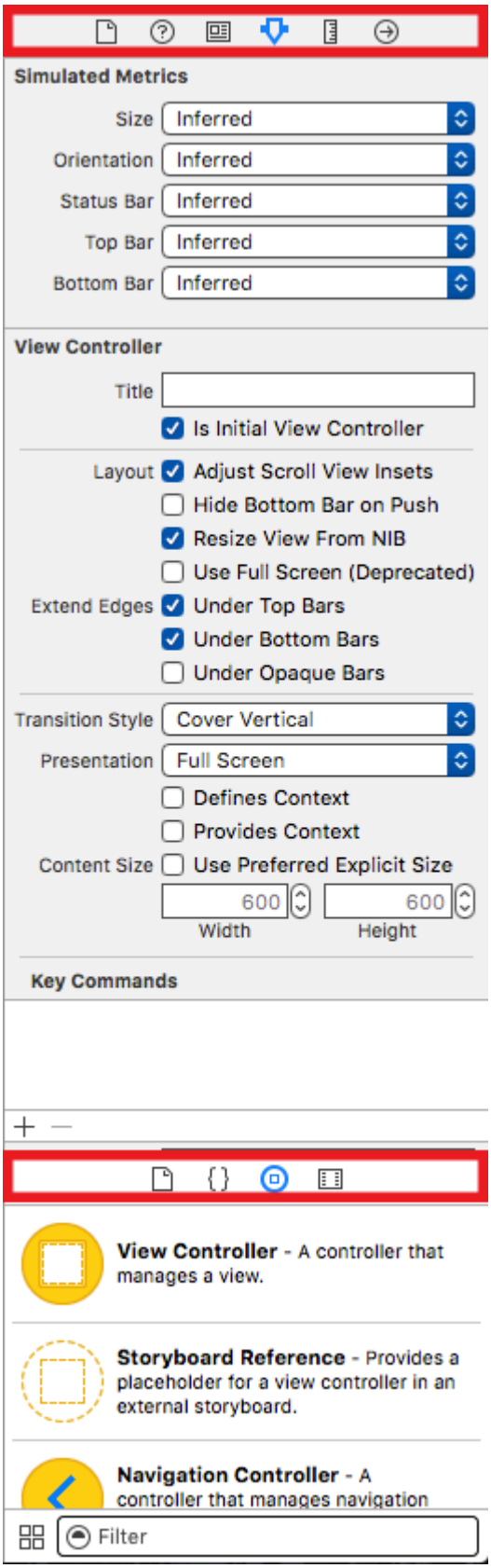
Configurez la zone de l'éditeur pour une tâche donnée à l'aide des boutons de configuration de l'éditeur situés dans la partie droite de la barre d'outils: 

- **Éditeur standard** Remplit la zone de l'éditeur avec le contenu du fichier sélectionné.
- **Assistant rédacteur** Présente un volet d'éditeur distinct avec un contenu lié logiquement au contenu dans le volet de l'éditeur standard. Vous pouvez également modifier le contenu.
- **Editeur de version.** Affiche les différences entre le fichier sélectionné dans un volet et une autre version du même fichier dans un second volet. Cet éditeur ne fonctionne que lorsque votre projet est sous contrôle de source.

Ressources et éléments dans le domaine des utilitaires

La zone des utilitaires située à l'extrême droite de la fenêtre de l'espace de travail vous permet d'accéder rapidement à ces ressources: Inspecteurs, pour afficher et modifier les caractéristiques du fichier ouvert dans un éditeur Bibliothèques de ressources prêtes à l'emploi à utiliser dans votre projet

Le panneau supérieur de la zone utilitaire affiche les inspecteurs. Le volet inférieur vous donne accès aux bibliothèques.



Le premier panneau (surligné en rouge) est la **barre d'inspecteur** . Utilisez-le pour choisir l'inspecteur le mieux adapté à votre tâche en cours. Deux inspecteurs sont toujours visibles dans la barre des inspecteurs (des inspecteurs supplémentaires sont disponibles dans certains éditeurs):

- **Inspecteur de fichiers.** Afficher et gérer les métadonnées du fichier sélectionné. En règle

générale, vous allez localiser les storyboards et autres fichiers multimédias et modifier les paramètres des fichiers d'interface utilisateur.
- **Aide rapide.** Afficher des détails sur un symbole, un élément d'interface ou un paramètre de génération dans le fichier. Par exemple, l'Aide rapide affiche une description concise d'une méthode, où et comment la méthode est déclarée, son étendue, les paramètres qu'elle prend, ainsi que la disponibilité de sa plate-forme et de son architecture.

Utilisez la **barre de bibliothèque** (la deuxième en surbrillance rouge) pour accéder aux bibliothèques de ressources prêtes à l'emploi pour votre projet:

- **Modèles de fichiers.** Modèles pour les types courants de fichiers et les constructions de code.
- **Extraits de code.** De courts morceaux de code source à utiliser dans votre logiciel, tels que des déclarations de classe, des flux de contrôle, des déclarations de bloc et des modèles pour les technologies Apple les plus utilisées.
- **Objets.** Articles pour l'interface utilisateur de votre application.
- **Médias.** Fichiers contenant des graphiques, des icônes, des fichiers audio, etc.

  Pour utiliser une bibliothèque, faites-la glisser directement dans la zone appropriée. Par exemple, pour utiliser un extrait de code, faites-le glisser de la bibliothèque vers l'éditeur source. Pour créer un fichier source à partir d'un modèle de fichier, faites glisser son modèle vers le navigateur de projet.

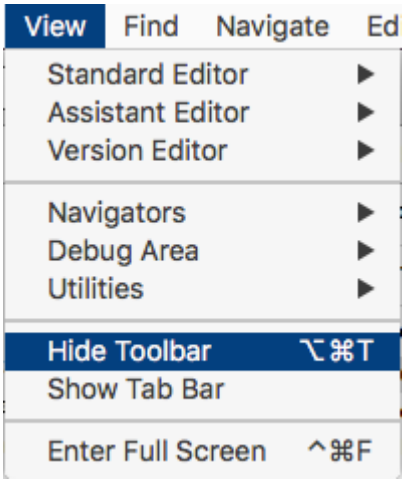
Pour restreindre les éléments affichés dans une bibliothèque sélectionnée, saisissez le texte approprié dans le champ de texte de la **barre de filtres** (le volet inférieur). Par exemple, tapez «bouton» dans le champ de texte pour afficher tous les boutons de la bibliothèque Objets.

Gérer les tâches avec la barre d'outils de l'espace de travail

La barre d’outils située en haut de la fenêtre de l’espace de travail permet d’accéder rapidement aux commandes fréquemment utilisées. Le **bouton Exécuter** génère et exécute vos produits. Le **bouton Stop** met fin à votre code en cours d'exécution. Le **menu Scheme** vous permet de configurer les produits que vous souhaitez créer et exécuter. Le **visualiseur d'activité** affiche la progression des tâches en cours d'exécution en affichant des messages d'état, la progression de la création et d'autres informations sur votre projet.

Les **boutons de configuration de l'éditeur** (le premier groupe de trois boutons) vous permettent de configurer la zone de l'éditeur et les **boutons de configuration de** l' **espace de travail** (le deuxième groupe de trois boutons) masquent ou affichent les zones facultatives du navigateur, du débogage et des utilitaires.

Le **menu Affichage** comprend des commandes permettant de masquer ou d'afficher la barre d'outils.



Créez votre premier programme dans Swift 3

Je présente ici comment créer le premier programme de base en langue Swift 3. Tout d'abord, vous devez avoir des connaissances de base en langage de programmation ou ne pas être prêt à l'apprendre dès le début.

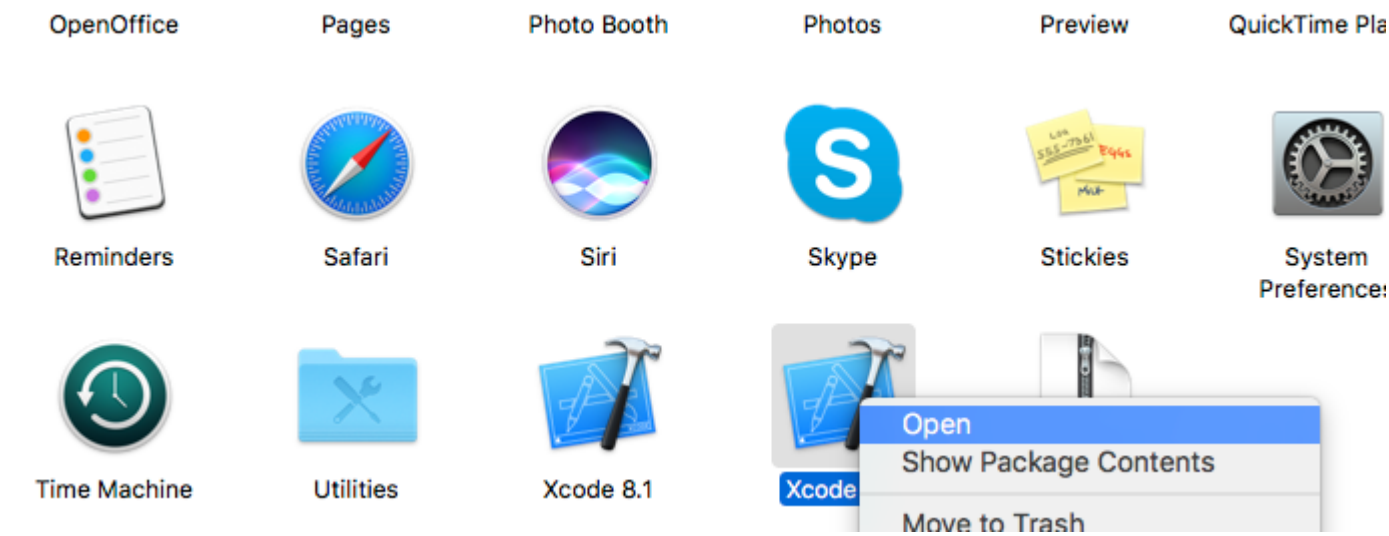
Conditions requises pour les développements:

1. MAC OS - Version 10.11.6 ou ultérieure pour le nouveau Xcode 8.2
2. Xcode - Version 8.2 Document Apple pour l'introduction Xcode.

Xcode 8.2 dispose de nouvelles fonctionnalités de langage Swift 3 avec les nouveaux APi compatibles iOS 10.

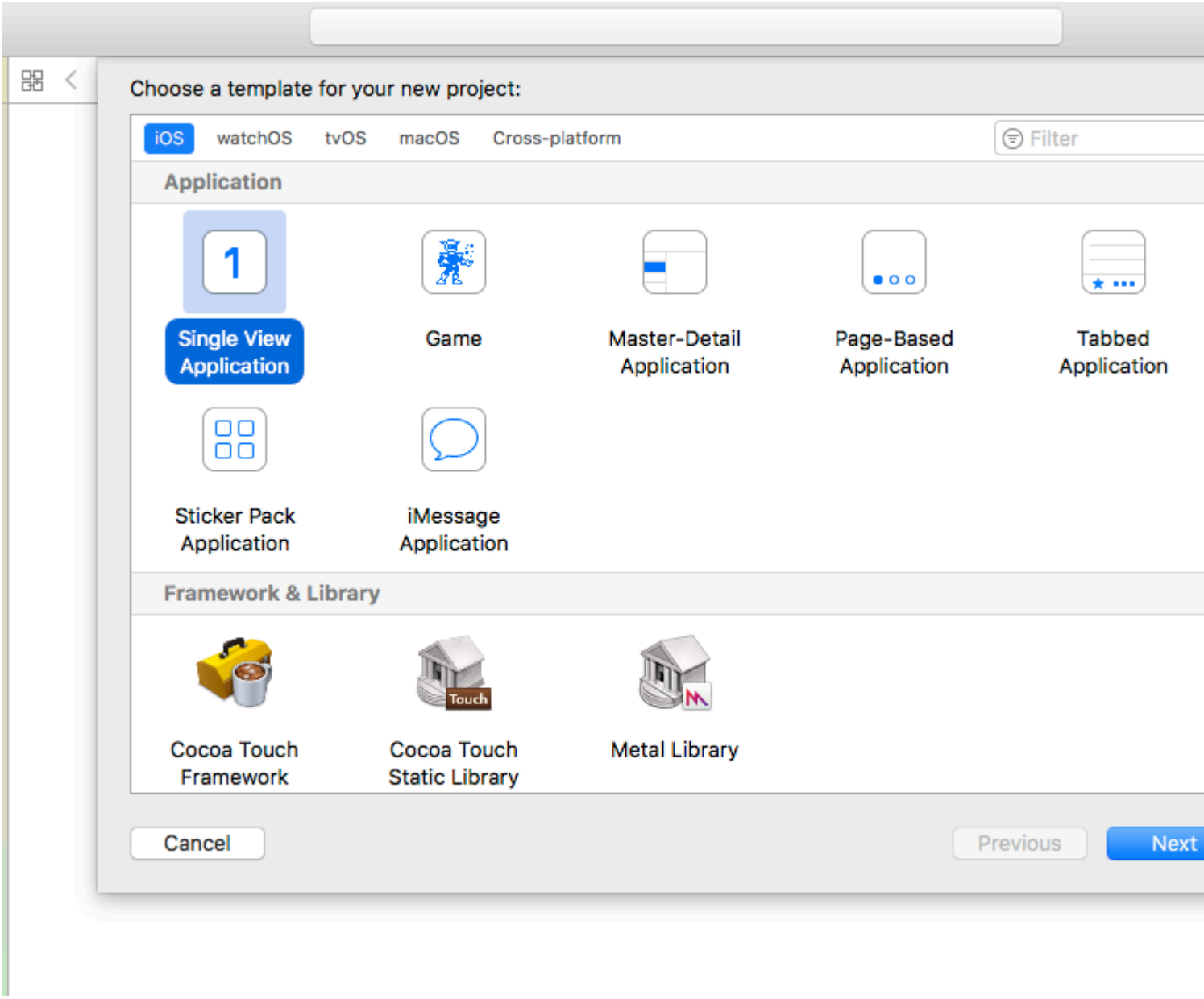
Créez votre premier programme

Allez d'abord dans Application et ouvrez votre Xcode 8.2.



Après cela, vous verrez l'écran

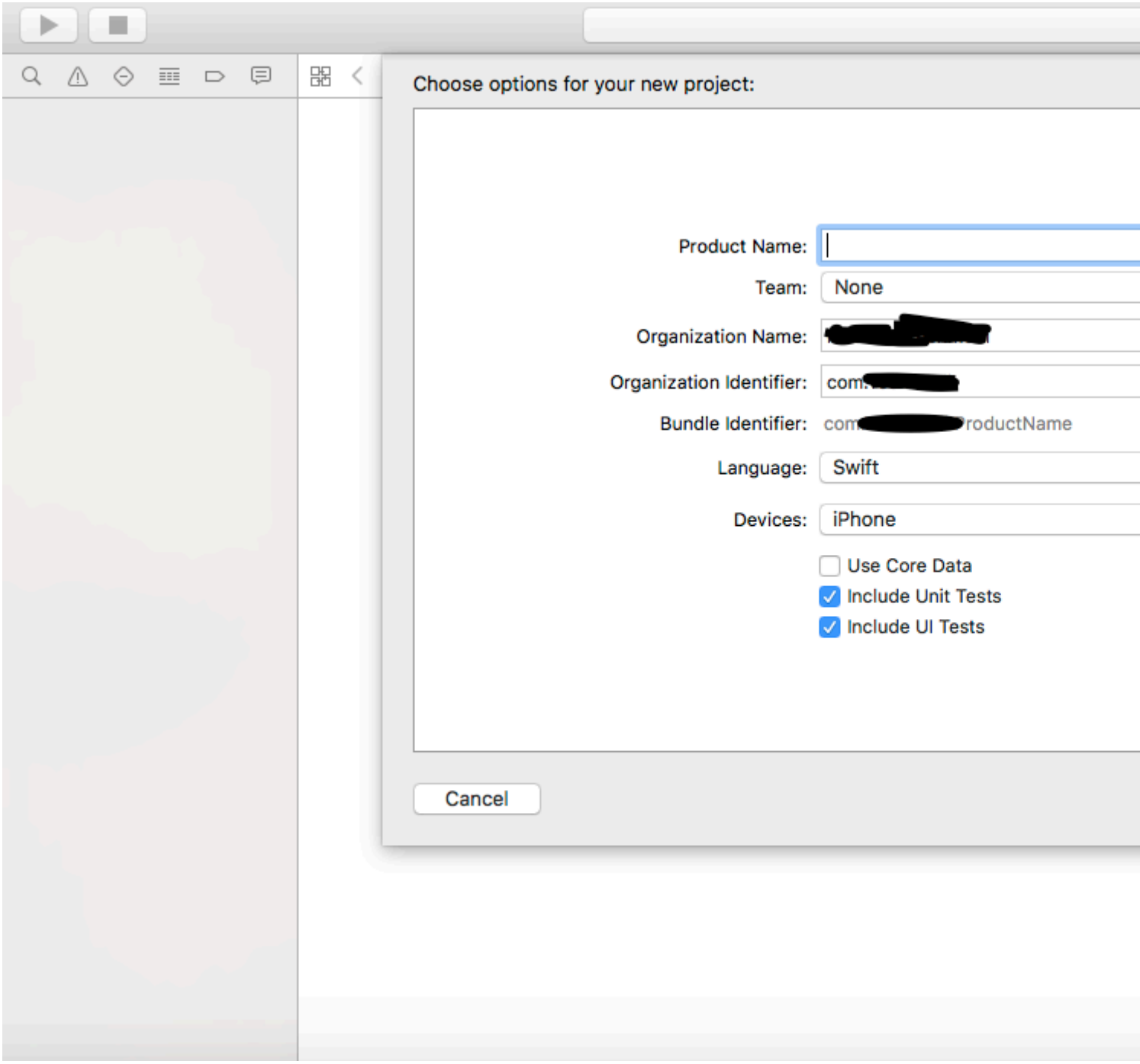
Ensuite, choisissez Créer un nouveau projet et ensuite vous verrez l'écran suivant



C'est aussi une partie très importante dans Xcode pour sélectionner notre type de projet. Nous devons choisir notre projet en fonction des types d'OS. Il y a cinq types d'options disponibles sur le dessus:

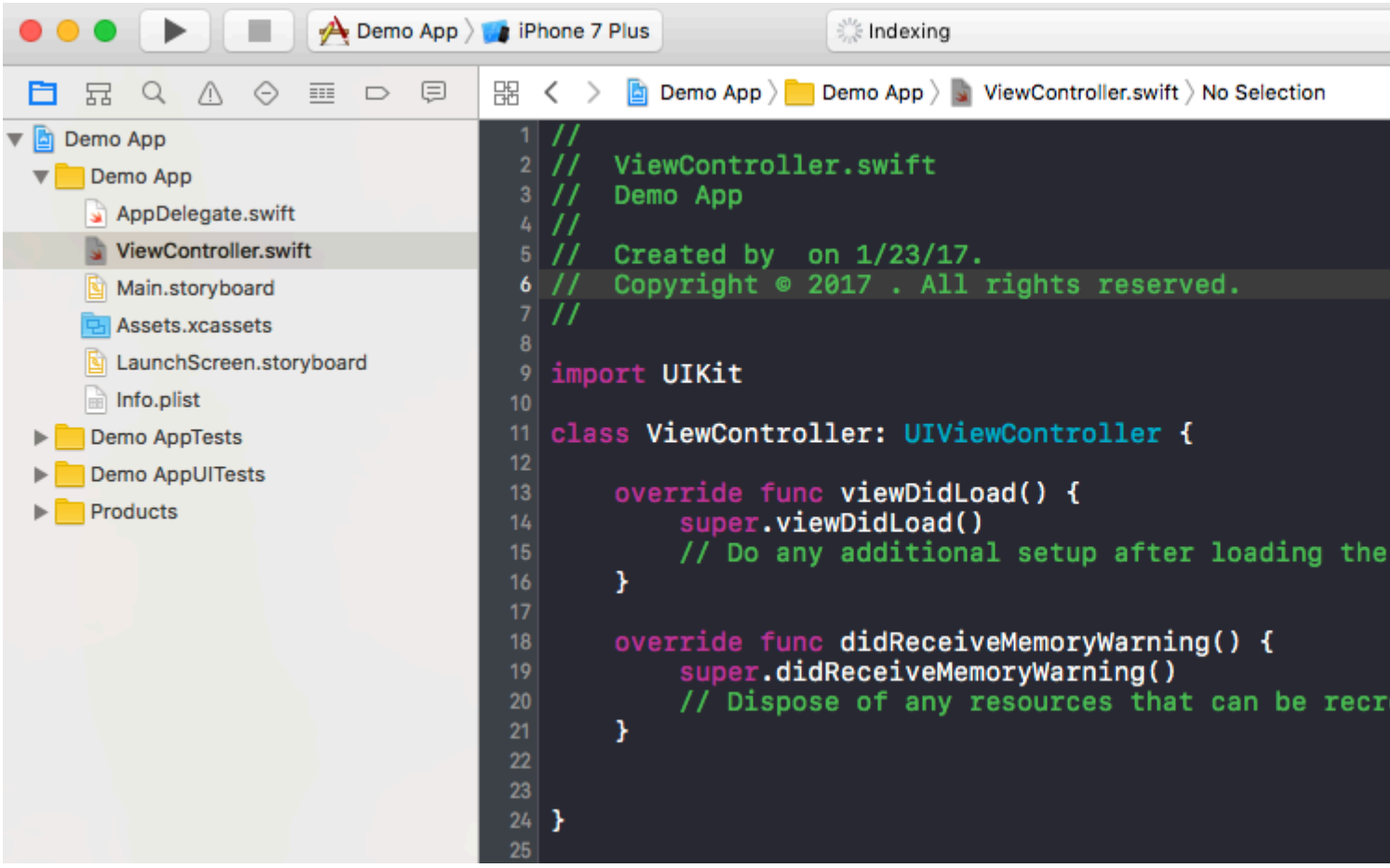
1. iOS
2. watchOS
3. macOS
4. Multiplateforme

Maintenant, nous choisissons la plate-forme iOS pour le développement et créons un projet très basique avec l'option d'application à vue unique:



Ensuite, nous devons indiquer le nom du produit, cela représentera le nom de votre ensemble et le nom de l'application.

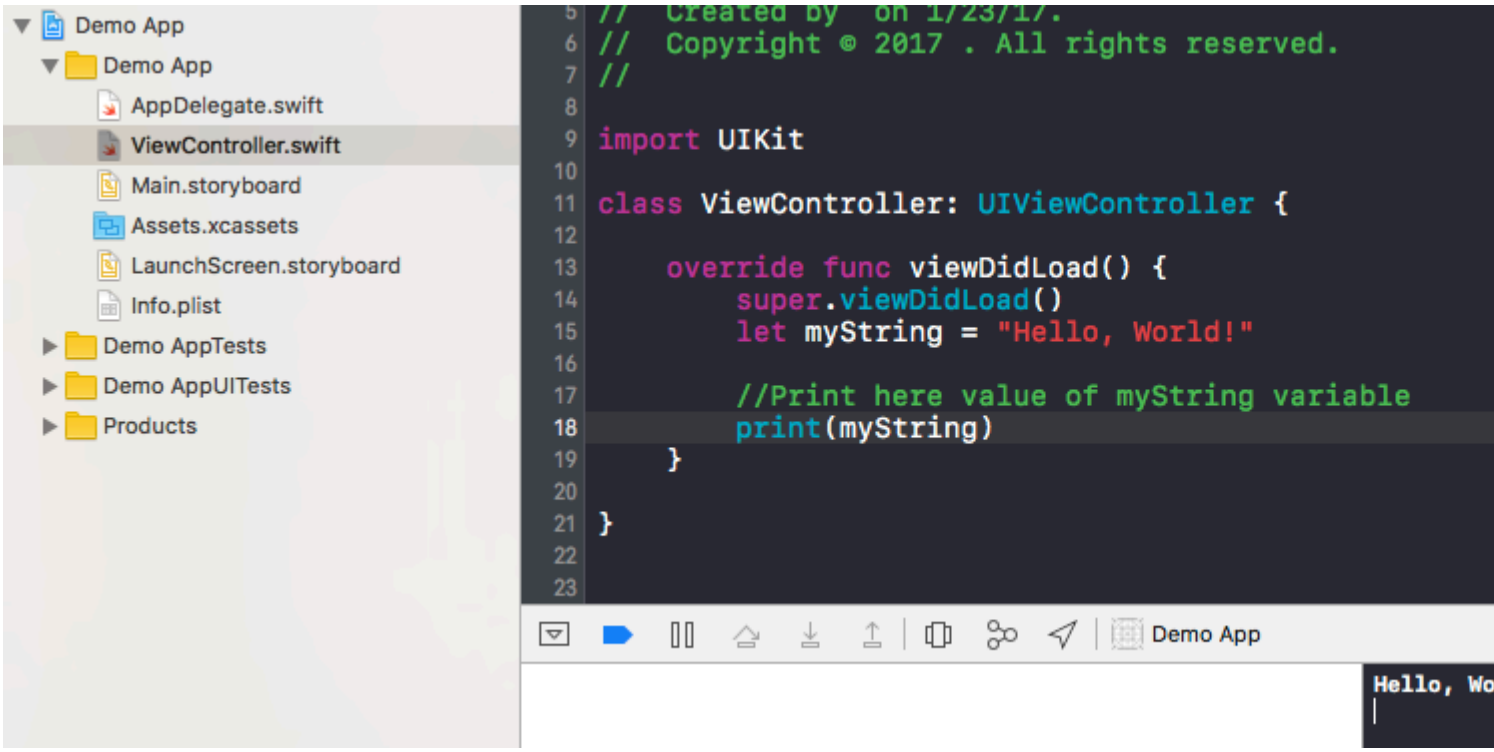
> Nom de l'application que vous pouvez modifier ultérieurement selon vos besoins.
> Ensuite, il faut cliquer sur "Créer" et votre écran ressemblera alors à celui-ci:



Dans cette classe, vous pouvez voir que le nom du fichier est ViewController.swift et à l'intérieur de la classe, le nom est également ViewController qui est l'héritage de la super classe UIViewController et nous créons notre première variable **nommée myString** Ajoutez ce qui suit sous 'super.viewDidLoad ()'

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

Nous allons imprimer le contenu de cette variable. Tout d'abord, sélectionnez votre type de simulateur en haut à gauche de l'écran, puis cliquez sur le bouton "Exécuter".



Après cela, votre sortie sera affichée sur le terminal qui est en bas à droite. Félicitations, ceci est votre premier programme Hello World dans Xcode.

Lire Démarrer avec iOS en ligne: https://riptutorial.com/fr/ios/topic/191/demarrer-avec-ios

Chapitre 2: 3D Touch

Examples

3D Touch avec Swift

Le toucher 3D a été introduit avec l'iPhone 6s Plus. Deux nouveaux comportements ont été ajoutés à cette nouvelle couche d'interface: Peek et Pop.

Peek and Pop en quelques mots

Peek - Appuyez fort

Pop - Presse vraiment difficile

Vérification du support 3D

Vous devez vérifier si le périphérique dispose d'un support tactile 3D. Vous pouvez le faire en vérifiant la valeur de la propriété *forceTouchCapability* d'un objet *UITraitCollection* . UITraitCollection décrit l'environnement d'interface iOS pour votre application.

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

Mise en œuvre du délégué

Vous devez implémenter les deux méthodes de *UIViewControllerPreviewingDelegate* dans votre classe. L'une des méthodes est pour le *coup* d' *oeil* et l'autre pour *le* comportement *pop* .

La méthode à implémenter pour le *peek* est *previewingContext* .

```
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

}
```

La méthode à implémenter pour la *pop* est *previewingContext* . :)

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

    let balanceViewController = viewControllerToCommit as! <YourViewController>

    // Do the stuff

    navigationController?.pushViewController(balanceViewController, animated: true)

}
```

Comme vous pouvez le voir, ce sont des méthodes surchargées. Vous pouvez utiliser le toucher 3D de quelque manière que ce soit en mettant en œuvre ces méthodes.

Objectif 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 Exemple

Objectif 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];
}
```

Lire 3D Touch en ligne: https://riptutorial.com/fr/ios/topic/6705/3d-touch

Chapitre 3: Accessibilité

Introduction

L'accessibilité d'iOS permet aux utilisateurs malentendants et malvoyants d'accéder à iOS et à votre application en prenant en charge diverses fonctionnalités telles que VoiceOver, Contrôle vocal, Blanc sur noir, Mono Audio, Parole au texte, etc. Fournir l'accessibilité dans l'application iOS signifie rendre l'application utilisable par tous.

Examples

Rendre une vue accessible

Marquez votre sous-classe `UIView` comme un élément accessible afin qu'il soit visible pour VoiceOver.

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

Assurez-vous que la vue indique une étiquette, une valeur et un indice significatifs. Apple fournit plus de détails sur la manière de choisir de bonnes descriptions dans le Guide de programmation de l' accessibilité .

Cadre d'accessibilité

Le cadre d'accessibilité est utilisé par VoiceOver pour effectuer des tests de réussite, dessiner le curseur VoiceOver et calculer où dans l'élément ciblé pour simuler un appui lorsque l'utilisateur double-touche l'écran. Notez que le cadre est en coordonnées d'écran!

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

Si vos éléments ou la disposition de votre écran changent souvent, envisagez de ne pas négliger - `accessibilityFrame` pour toujours fournir un rectificatif à jour. Le calcul de l'image relative à l'écran des sous-vues de la vue de défilement peut être source d'erreurs et fastidieux. iOS 10 introduit une nouvelle API pour faciliter les choses: `accessibilityFrameInContainerSpace` .

Changement d'écran

VoiceOver fonctionne très bien la plupart du temps, en lisant à voix haute des écrans pleins de contenu et en suivant intuitivement l'utilisateur. Hélas, aucune solution générale n'est parfaite. Parfois, vous seul, le développeur de l'application, savez où se concentrer VoiceOver pour une expérience utilisateur optimale. Heureusement, VoiceOver écoute les notifications d'accessibilité du système pour obtenir des indices sur la place du focus. Pour déplacer le curseur VoiceOver manuellement, publiez une notification sur l'écran d'accessibilité:

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

Lorsque cette notification est publiée, une courte série de tonalités informe les utilisateurs du changement. Le second paramètre peut être l'élément suivant à focaliser ou une chaîne annonçant la modification. Ne publiez une notification de changement d'écran que si l'expérience VoiceOver est mauvaise sans cette solution et qu'aucune autre solution n'existe. Déplacer le curseur VoiceOver revient à ouvrir l'écran d'un utilisateur voyant. Cela peut être gênant et désorientant de se retrouver dans cette situation.

Changement de disposition

Dans de nombreux cas, le contenu d'un seul écran sera mis à jour avec un contenu nouveau ou différent. Par exemple, imaginez un formulaire qui révèle des options supplémentaires en fonction de la réponse de l'utilisateur à une question précédente. Dans ce cas, une notification de «modification de la mise en page» vous permet d'annoncer la modification ou de vous concentrer sur un nouvel élément. Cette notification accepte les mêmes paramètres que la notification de changement d'écran.

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

Annonce

Les annonces sont utiles pour alerter les utilisateurs des événements qui ne nécessitent aucune interaction, tels que «verrouillé» ou «chargement terminé». Utilisez une annonce plus spécifique pour informer les utilisateurs des modifications d'écran ou d'autres modifications mineures de la mise en page.

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

Éléments de commande

VoiceOver navigue de haut en bas à droite, quelle que soit la hiérarchie de vue. C'est généralement la manière dont le contenu est arrangé dans les langues de gauche à droite, car les individus voyants ont tendance à scanner l'écran dans un «motif en forme de F». Les utilisateurs de VoiceOver s'attendent à naviguer de la même manière que les utilisateurs classiques. La prévisibilité et la cohérence sont très importantes pour l'accessibilité. Veuillez vous abstenir de faire des personnalisations qui «améliorent» le comportement par défaut (par exemple, ordonnez d'abord la barre d'onglets dans l'ordre de glissement). Cela dit, si vous avez reçu des commentaires selon lesquels l'ordre des éléments de votre application est surprenant, vous pouvez améliorer l'expérience de plusieurs façons.

Si VoiceOver doit lire les sous-vues d'une vue les unes après les autres mais que ce n'est pas le cas, vous devrez peut-être indiquer à VoiceOver que les éléments contenus dans une seule vue sont liés. Vous pouvez le faire en définissant `shouldGroupAccessibiltyChildren` :

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

Pour prendre en charge des structures de navigation complexes couvrant plusieurs conteneurs ou inclure des interfaces rendues sans UIKit, envisagez d'implémenter le protocole de conteneur dans la vue parente.

Conteneur d'accessibilité

VoiceOver peut naviguer dans de nombreuses applications sur iOS car la plupart des classes `UIKit` implémentent le `UIAccessibilityProtocol` . Les fonctionnalités qui ne représentent pas des éléments à l'écran à l'aide de `UIView` , y compris les applications qui utilisent Core Graphics ou Metal pour effectuer le dessin, doivent décrire ces éléments pour l'accessibilité. A partir d'iOS 8.0, cela peut être fait en assignant une propriété sur `UIView` contenant des éléments inaccessibles:

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

Chaque objet du tableau peut être une instance de `UIAccessibilityElement` ou toute autre classe adhérant à `UIAccessibilityProtocol` . Les éléments enfants doivent être renvoyés dans l'ordre dans lequel l'utilisateur doit les parcourir. En tant qu'auteur de l'application, vous pouvez utiliser des conteneurs d'accessibilité pour remplacer la commande par défaut en haut à gauche de la navigation par balayage VoiceOver. Étant donné `UIView` implémente `UIAccessibilityProtocol` , vous pouvez combiner des instances de `UIAccessibilityElement` et `UIView` dans le même tableau d'éléments d'accessibilité enfants. Notez que si vous affectez des éléments manuellement, vous n'avez pas besoin d'implémenter de méthodes de protocole d'accessibilité dynamique, mais vous devrez peut-être émettre une notification de changement d'écran pour les éléments à détecter par VoiceOver.

Vue modale

Les vues modales capturent complètement l'attention de l'utilisateur jusqu'à ce qu'une tâche soit terminée. iOS clarifie cela pour les utilisateurs en atténuant et en désactivant tous les autres contenus lorsqu'une vue modale, telle qu'une alerte ou une fenêtre pop-up, est visible. Une application qui implémente une interface modale personnalisée doit indiquer à VoiceOver que cette vue mérite toute l'attention de l'utilisateur en définissant `accessibilityViewIsModal` . Notez que cette propriété ne doit être définie que sur la vue contenant le contenu modal, pas sur les éléments contenus dans une vue modale.

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

Marquer une vue comme modale encourage VoiceOver à ignorer les vues fraternelles. Si, après avoir défini cette propriété, vous trouvez que VoiceOver navigue toujours sur d'autres éléments de votre application, essayez de masquer les vues de problème jusqu'à ce que le modal soit supprimé.

Éléments cachés

La plupart des classes UIKit, y compris UIView, adhèrent à `UIAccessibilityProtocol` et renvoient des valeurs correctes par défaut. Il est facile de considérer qu'un `UIView` défini sur caché est également absent de la hiérarchie d'accessibilité et ne sera pas parcouru par VoiceOver. Bien que ce comportement par défaut soit généralement suffisant, il peut arriver qu'une vue soit présente dans la hiérarchie de la vue, mais pas visible ou navigable. Par exemple, une collection de boutons peut être recouverte par une autre vue, ce qui les rend invisibles pour un utilisateur voyant. VoiceOver essaiera néanmoins de les naviguer car ils ne sont techniquement pas cachés par `UIKit` et sont donc toujours présents dans la hiérarchie d'accessibilité. Dans ce cas, vous devez indiquer à VoiceOver que la vue parent n'est pas accessible. Vous pouvez le faire en masquant explicitement la vue depuis UIKit en définissant la valeur masquée lorsque la vue devient hors écran:

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

Vous pouvez également laisser la vue parent visible et simplement masquer ses enfants dans la hiérarchie d'accessibilité:

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

Les vues temporaires sont un autre endroit où vous souhaitez masquer des éléments de la hiérarchie d'accessibilité tout en les laissant visibles aux utilisateurs. Par exemple, la vue qui apparaît lorsque vous appuyez sur le bouton de volume est visible pour les utilisateurs voyants mais ne nécessite pas d'attention comme le fait une alerte normale. Vous ne voudriez pas que VoiceOver interrompe l'utilisateur et déplace le curseur loin de ce qu'il faisait pour annoncer le nouveau volume, d'autant plus que l'ajustement du volume fournit déjà un retour auditif via le son de clic. Dans de tels cas, vous souhaiterez masquer la vue en utilisant `accessibilityElementsHidden` .

Lire Accessibilité en ligne: https://riptutorial.com/fr/ios/topic/773/accessibilite

Chapitre 4: Achat in-app

Examples

Single IAP dans Swift 2

Après avoir créé un IAP dans iTunesConnect:

Dans le contrôleur de vue que vous voulez acheter

```
import StoreKit
```

et ajouter les délégués concernés

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

déclarer une variable avec l'identifiant du produit depuis 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...")

   }

}
```

câbler un bouton à une fonction pour acheter 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");
    }

}
```

Et voici quelques méthodes d'aide

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

 }
```

les méthodes déléguées à déclarer

```
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;
            }
        }
    }

}
```

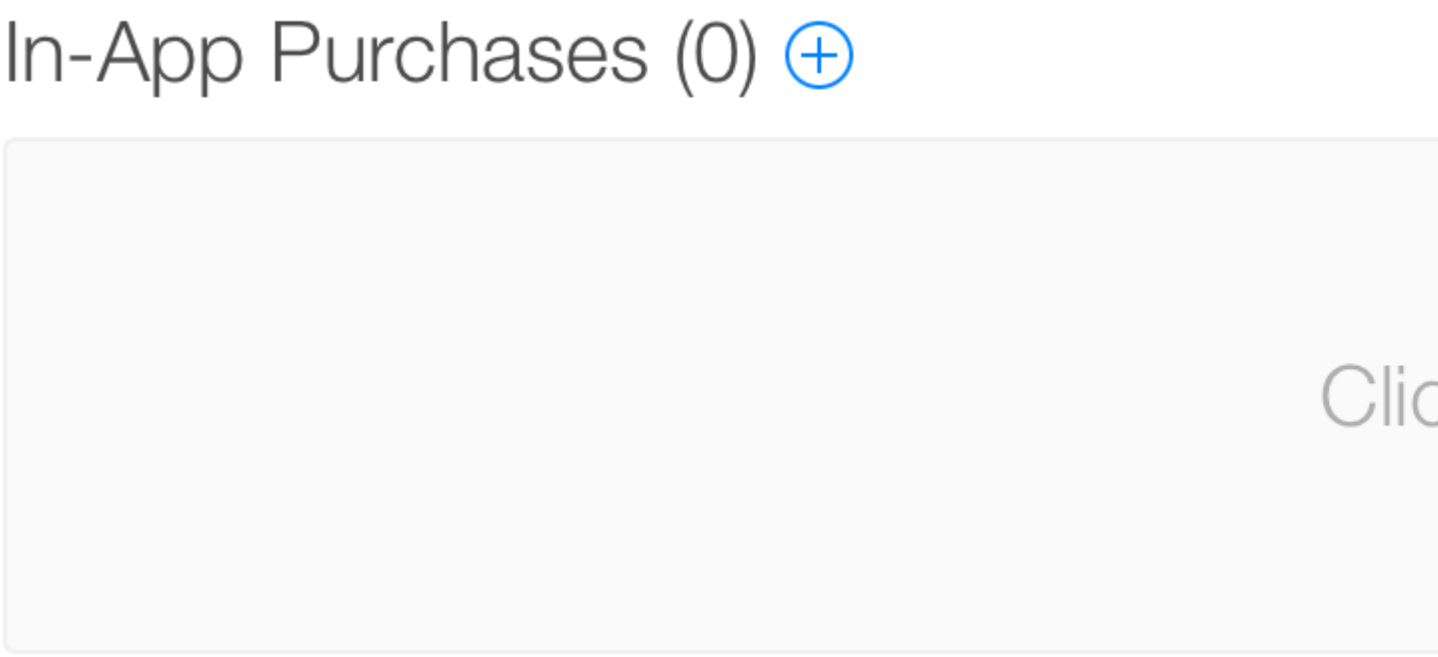
Et puis le code pour restaurer un achat non consommable dans l'application

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

Configurer dans iTunesConnect

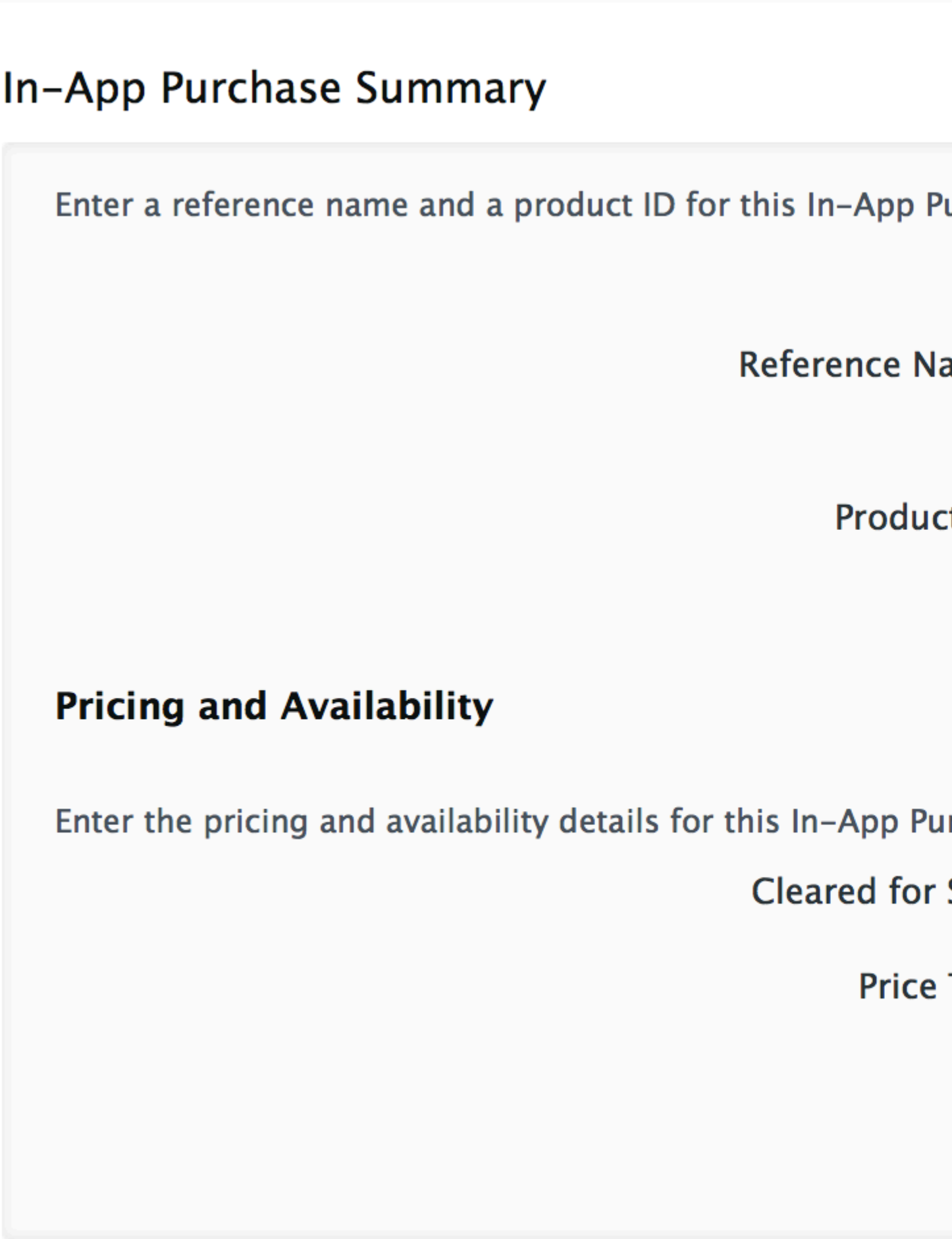
Dans iTunesConnect , sélectionnez l'application à laquelle vous souhaitez ajouter un IAP.

Cliquez sur les fonctionnalités et vous verrez ceci:



Cliquez sur le plus. Vous devrez ensuite sélectionner le type d'IAP que vous souhaitez créer.

Ensuite, vous devrez remplir toutes les informations pour votre IAP.



En cas de problème, vous pouvez consulter le Guide de configuration IAP .

Chapitre 5: AFNetworking

Examples

Envoi du bloc d'achèvement sur un thread personnalisé

Chaque fois que AFNetworking est utilisé, l'appel est envoyé sur un thread personnalisé fourni par AFNetworking. Lorsque l'appel retourne au bloc d'achèvement, il est exécuté sur le thread principal.

Cet exemple définit un thread personnalisé envoyé au bloc d'achèvement:

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];
```

Lire AFNetworking en ligne: https://riptutorial.com/fr/ios/topic/3002/afnetworking

Chapitre 6: AirDrop

Examples

AirDrop

Objectif c

Airdrop peut être utilisé à partir de `UIActivityViewController` . La classe `UIActivityViewController` est un contrôleur d'affichage standard qui fournit plusieurs services standard, tels que la copie d'éléments dans le presse-papiers, le partage de contenu sur des sites de médias sociaux, l'envoi d'éléments via Messages, AirDrop et certaines applications tierces.

Dans ce cas, nous `UIActivityViewController` une image via `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];
}
```

Rapide

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

        }
```

Lire AirDrop en ligne: https://riptutorial.com/fr/ios/topic/7360/airdrop

Chapitre 7: AJOUT D'UN EN-TÊTE DE PONT SWIFT

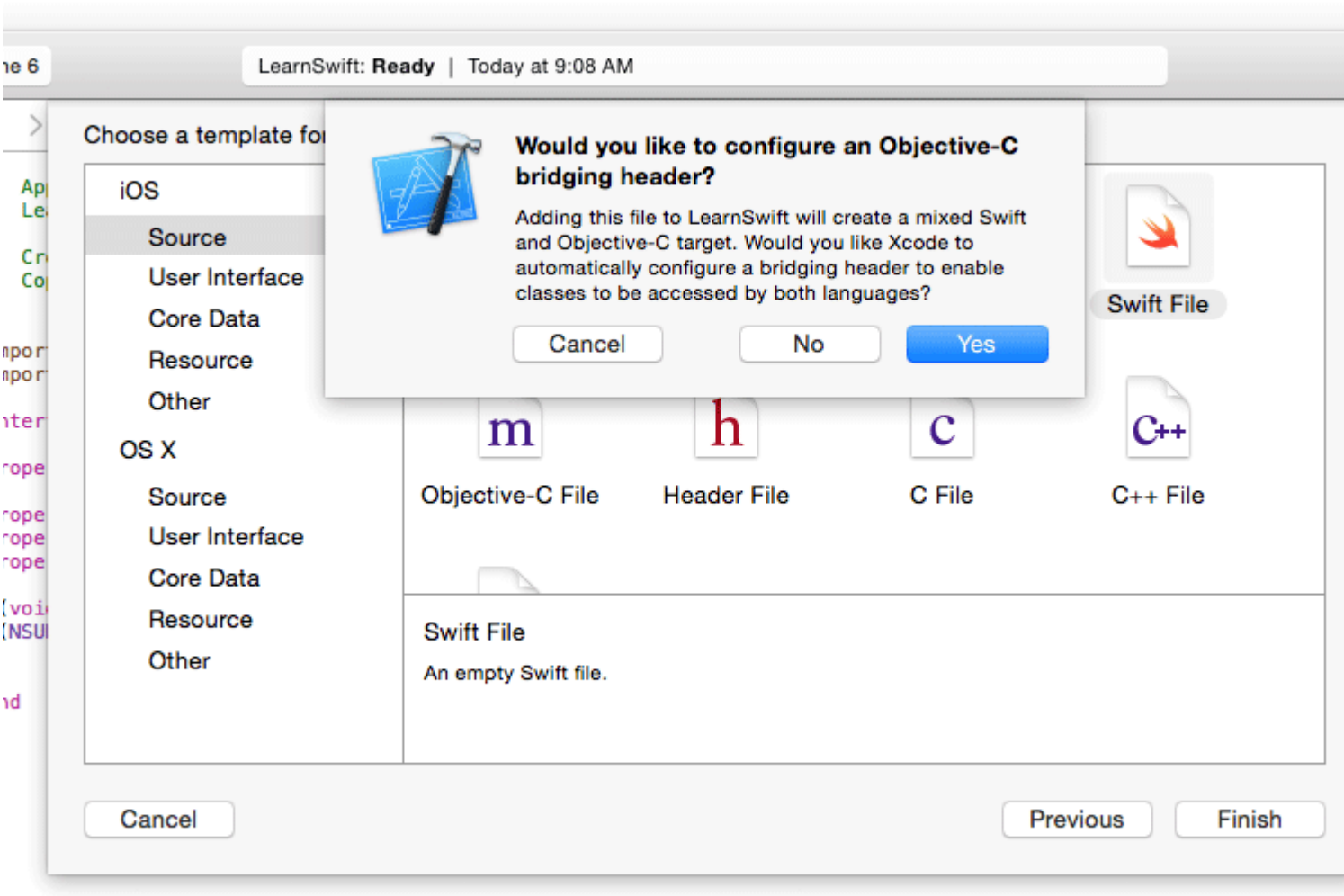
Examples

Comment créer un en-tête de pontage Swift manuellement

- Ajoutez un nouveau fichier à Xcode (Fichier> Nouveau> Fichier), puis sélectionnez «Source» et cliquez sur «Fichier d'en-tête».
- Nommez votre fichier «YourProjectName-Bridging-Header.h». Exemple: dans mon application Station, le fichier s'appelle “Station-Bridging-Header”.
- Créez le fichier.
- Accédez aux paramètres de construction de votre projet et recherchez la section «Compilateur rapide - Génération de code». Vous trouverez peut-être plus rapide de taper «Compilateur rapide» dans la zone de recherche pour limiter les résultats. Note: Si vous n'avez pas de section «Compilateur rapide - Génération de code», cela signifie que vous n'avez probablement pas encore ajouté de classes Swift à votre projet. Ajoutez un fichier Swift, puis réessayez.
- À côté de "Objective-C Bridging Header", vous devrez ajouter le nom / chemin de votre fichier d'en-tête. Si votre fichier réside dans le dossier racine de votre projet, insérez simplement le nom du fichier d'en-tête. Exemples: “ProjectName / ProjectName-Bridging-Header.h” ou simplement “ProjectName-Bridging-Header.h”.
- Ouvrez votre en-tête de pont nouvellement créé et importez vos classes Objective-C à l'aide des instructions #import. Toute classe répertoriée dans ce fichier pourra être accessible à partir de vos classes rapides.

Xcode crée automatiquement

Ajoutez un nouveau fichier Swift à votre projet Xcode. Nommez-le comme bon vous semble et vous devriez recevoir un message vous demandant si vous souhaitez créer un en-tête de pontage. Remarque: Si vous ne recevez pas d'invite pour ajouter un en-tête de pontage, vous avez probablement refusé ce message une fois auparavant et vous devrez ajouter l'en-tête manuellement (voir ci-dessous).



Lire AJOUT D'UN EN-TÊTE DE PONT SWIFT en ligne:
https://riptutorial.com/fr/ios/topic/10851/ajout-d-un-en-tete-de-pont-swift

Chapitre 8: Alamofire

Syntaxe

- réponse()
- responseData ()
- responseString (encodage: NSStringEncoding)
- responseJSON (options: NSJSONReadingOptions)
- responsePropertyList (options: NSPropertyListReadOptions)

Paramètres

Paramètre	Détails
Méthode	`.OPTIONS, .GET, .HEAD, .POST, .PUT, .PATCH, .DELETE, .TRACE, .CONNECT`
URLString	`URLStringConvertible`
paramètres	`[String: AnyObject]?`
codage	`ParameterEncoding`
les en-têtes	`[String: String]?`

Examples

Faire une demande

```
import Alamofire

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

Validation automatique

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

Traitement des réponses

```
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)")
         }
     }
```

Validation manuelle

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

Gestionnaire de réponse

```
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)
      }
```

Gestionnaires de réponse en chaîne

```
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)")
     }
```

Lire Alamofire en ligne: https://riptutorial.com/fr/ios/topic/1823/alamofire

Chapitre 9: API de reconnaissance vocale iOS 10

Examples

Discours au texte: reconnaître la parole d'un paquet contenant un enregistrement 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()
}
```

Lire API de reconnaissance vocale iOS 10 en ligne: https://riptutorial.com/fr/ios/topic/5986/api-de-reconnaissance-vocale-ios-10

Chapitre 10: API Google Adresses iOS

Examples

Obtenir des lieux proches de l'emplacement actuel

Conditions préalables

1. Installer des modules dans votre projet
2. Installer le SDK GooglePlaces
3. Activer les services de localisation

Nous devons d'abord obtenir l'emplacement de l'utilisateur en obtenant sa longitude et sa latitude actuelles.

1. Importer GooglePlaces et GooglePlacePicker

```
import GooglePlaces
import GooglePlacePicker
```

2. Ajoutez le protocole `CLLOcationManagerDelegate`

```
class ViewController: UIViewController, CLLocationManagerDelegate {

}
```

3. créer votre CLLocationManager ()

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

4. Demande d'autorisation

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

5. Créer un bouton pour appeler la méthode GooglePlacePicker

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

Lire API Google Adresses iOS en ligne: https://riptutorial.com/fr/ios/topic/6908/api-google-adresses-ios

Chapitre 11: App Transport Security (ATS)

Paramètres

Paramètre	Détails
`NSAppTransportSecurity`	Configurer ATS
`NSAllowsArbitraryLoads`	Réglez sur `YES` pour désactiver l'ATS partout. Dans iOS 10 et versions ultérieures, et macOS 10.12 et versions ultérieures, la valeur de cette clé est ignorée si l'une des clés suivantes est présente dans le fichier Info.plist de votre application: NSAllowsArbitraryLoadsInMedia, NSAllowsArbitraryLoadsInWebContent, NSAllowsLocalNetworking
`NSAllowsArbitraryLoadsInMedia`	Définissez la valeur sur `YES` pour désactiver l'ATS pour les médias chargés à l'aide des API du framework AV Foundation. ***(iOS 10+, MacOS 10.12+)***
`NSAllowsArbitraryLoadsInWebContent`	Définissez `YES` pour désactiver l'ATS dans les vues Web de votre application (`WKWebView` , `UIWebView` , `WebView`) sans affecter vos connexions NSURLSession. ***(iOS 10+, macOS 10.12+)***
`NSAllowsLocalNetworking`	Définissez sur `YES` pour désactiver les connexions aux domaines non qualifiés et aux domaines .local. ***(iOS 10+, macOS 10.12+)***
`NSExceptionDomains`	Configurer des exceptions pour des domaines spécifiques
`NSIncludesSubdomains`	Définissez la valeur sur `YES` pour appliquer les exceptions à tous les sous-domaines du domaine sélectionné.
`NSRequiresCertificateTransparency`	Définissez la valeur sur `YES` pour exiger que les horodatages de transparence de certificat (CT) valides et signés, provenant de journaux CT connus, soient présentés pour les certificats de serveur (X.509) sur un domaine. ***(iOS 10+, macOS 10.12+)***

Paramètre	Détails
`NSExceptionAllowsInsecureHTTPLoads`	Définissez sur `YES` pour autoriser HTTP sur le domaine sélectionné.
`NSExceptionRequiresForwardSecrecy`	La valeur par défaut est `YES` . Réglez sur `NO` pour désactiver le secret de transfert et acceptez plus de chiffrements.
`NSExceptionMinimumTLSVersion`	La valeur par défaut est `TLSv1.2` ; Les valeurs possibles sont: `TLSv1.0` , `TLSv1.1` , `TLSv1.2`
`NSThirdPartyExceptionAllowsInsecureHTTPLoads`	Similaire à `NSExceptionAllowsInsecureHTTPLoads` , mais pour les domaines sur lesquels vous n'avez aucun contrôle
`NSThirdPartyExceptionRequiresForwardSecrecy`	Similaire à `NSExceptionRequiresForwardSecrecy` , mais pour les domaines sur lesquels vous n'avez aucun contrôle
`NSThirdPartyExceptionMinimumTLSVersion`	Similaire à `NSExceptionMinimumTLSVersion` , mais pour les domaines sur lesquels vous n'avez aucun contrôle

Remarques

La sécurité du transport des applications est une fonction de sécurité dans iOS et macOS. Il empêche les applications d'établir des connexions non sécurisées. Par défaut, les applications ne peuvent utiliser que des connexions HTTPS sécurisées.

Si une application doit se connecter à un serveur via HTTP, des exceptions doivent être définies dans `Info.plist` . (voir les exemples pour plus d'informations à ce sujet)

Remarque: en 2017, Apple appliquera l'ATS. Cela signifie que vous ne pouvez plus télécharger des applications qui ont des exceptions ATS définies dans `Info.plist` . Si vous pouvez fournir de bons arguments, expliquez pourquoi vous devez utiliser HTTP, vous pouvez contacter Apple et ils peuvent vous permettre de définir des exceptions. (Source: WWDC 2016 - Session 706)

Vous trouverez plus d'informations sur la configuration de la sécurité du transport App dans la documentation de CocoaKeys .

Examples

Charger tout le contenu HTTP

Apple a introduit ATS avec iOS 9 comme nouvelle fonction de sécurité pour améliorer la confidentialité et la sécurité entre les applications et les services Web. ATS par défaut échoue

toutes les requêtes non HTTPS. Bien que cela puisse être vraiment intéressant pour les environnements de production, cela peut être gênant lors des tests.

ATS est configuré dans le fichier `Info.plist` la cible avec le dictionnaire `NSAppTransportSecurity` (`App Transport Security Settings` dans l'éditeur Xcode Info.plist). Pour autoriser tout le contenu HTTP, ajoutez le booléen `Allow Arbitrary Loads` (`NSAllowsArbitraryLoads`) et réglez-le sur `YES` . Ceci n'est pas recommandé pour les applications de production et si un contenu HTTP est requis, il est recommandé de l'activer de manière sélective.

Charger de manière sélective le contenu HTTP

Semblable à l'activation de tout le contenu HTTP, toute la configuration se produit sous les `App Transport Security Settings` . Ajoutez le dictionnaire des `Exception Domains` (`NSExceptionDomains`) aux paramètres ATS de niveau supérieur.

Pour chaque domaine, ajoutez un élément de dictionnaire aux domaines d'exception, où la clé correspond au domaine en question. Définissez `NSExceptionAllowsInsecureHTTPLoads` sur `YES` pour désactiver la configuration HTTPS requise pour ce domaine.

Les points d'extrémité nécessitent SSL

Introduit dans iOS 9, tous les points de terminaison doivent respecter la spécification HTTPS. Tout ordinateur d'extrémité n'utilisant pas SSL échouera avec un avertissement dans le journal de la console. Pour votre application, il apparaîtra que la connexion Internet a échoué.

Pour configurer des exceptions: Placez les éléments suivants dans votre fichier Info.plist:

1. Autoriser un domaine particulier (testdomain.com) **uniquement** :

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

La clé qui autorise un tel comportement est `NSExceptionAllowsInsecureHTTPLoads` . Dans ce cas, l'application autorisera la connexion HTTP au domaine mentionné (testdomain.com) uniquement et bloquera toutes les autres connexions HTTP.

La clé `NSIncludesSubdomains` spécifie que tous les **sous** - **domaines** du domaine mentionné (testdomain.com) doivent également être autorisés.

2. Autoriser n'importe quel domaine:

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

Dans ce cas, l'application autorisera la connexion HTTP à **n'importe quel** domaine. À compter du 1er janvier 2017, l'utilisation de cet indicateur entraînera une révision approfondie de l'App Store et les développeurs de l'application devront expliquer pourquoi ils doivent utiliser cette exception en premier lieu. Les explications possibles comprennent:

- Une application qui charge un contenu multimédia crypté ne contenant aucune information personnalisée.
- Connexions aux périphériques ne pouvant pas être mis à niveau pour utiliser des connexions sécurisées.
- Connexion à un serveur géré par une autre entité et ne prenant pas en charge les connexions sécurisées.

Lire App Transport Security (ATS) en ligne: https://riptutorial.com/fr/ios/topic/5435/app-transport-security--ats-

Chapitre 12: AppDelegate

Introduction

AppDelegate est un protocole qui définit les méthodes appelées par l'objet singleton UIApplication en réponse à des événements importants survenant pendant la durée de vie d'une application.

Normalement utilisé pour effectuer des tâches au démarrage de l'application (environnement d'application de configuration, analitycs (ex .: Mixpanel / GoogleAnalytics / Crashlitics), pile de base de données, etc.) les tâches.

Examples

Tous les états d'application via les méthodes AppDelegate

Pour vous mettre à jour ou faire quelque chose avant que l'application ne soit mise en ligne, vous pouvez utiliser la méthode ci-dessous.

AppDidFinishLaunching

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

Alors que l'application entre en premier plan:

```
- (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.
}
```

Lors du lancement de l'application et de l'arrière-plan du premier plan, cliquez sur la méthode ci-dessous:

```
- (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.
}
```

Alors que l'application entre en arrière-plan:

```
- (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.
}
```

Pendant que l'application démissionne active

```
- (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.
}
```

Alors que l'application se termine:

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

Rôles AppDelegate:

- AppDelegate contient le `startup code` votre application.
- Il répond aux `key changes` de l' `state` de votre application. Plus précisément, il répond aux interruptions temporaires et aux modifications de l'état d'exécution de votre application, par exemple lorsque votre application passe du premier plan à l'arrière-plan.
- Il `responds to notifications` provenant de l'extérieur de l'application, telles que les notifications à distance (également appelées notifications push), les avertissements de mémoire insuffisante, les notifications de fin de téléchargement, etc.
- Il `determines` si `state preservation` et la `restoration state preservation` doivent avoir lieu et aide au processus de préservation et de restauration, le cas échéant.
- Il `responds to events` qui ciblent l'application elle-même et ne sont pas spécifiques aux vues de votre application ou aux contrôleurs de vue. Vous pouvez l'utiliser pour stocker les objets de données centraux de votre application ou tout contenu ne possédant pas de contrôleur de vue propriétaire.

Ouverture d'une ressource spécifiée par URL

Demande au délégué d'ouvrir une ressource spécifiée par une URL et fournit un dictionnaire des options de lancement.

Exemple d'utilisation:

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

```
            annotation: options[.annotation]
        )
    }
```

Gestion des notifications locales et distantes

Exemple d'utilisation:

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

Enregistrement:

```
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)
}
```

Traitement des notifications à distance:

```
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
    )
}
```

Gestion des notifications locales:

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

Action de manipulation (obsolète):

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

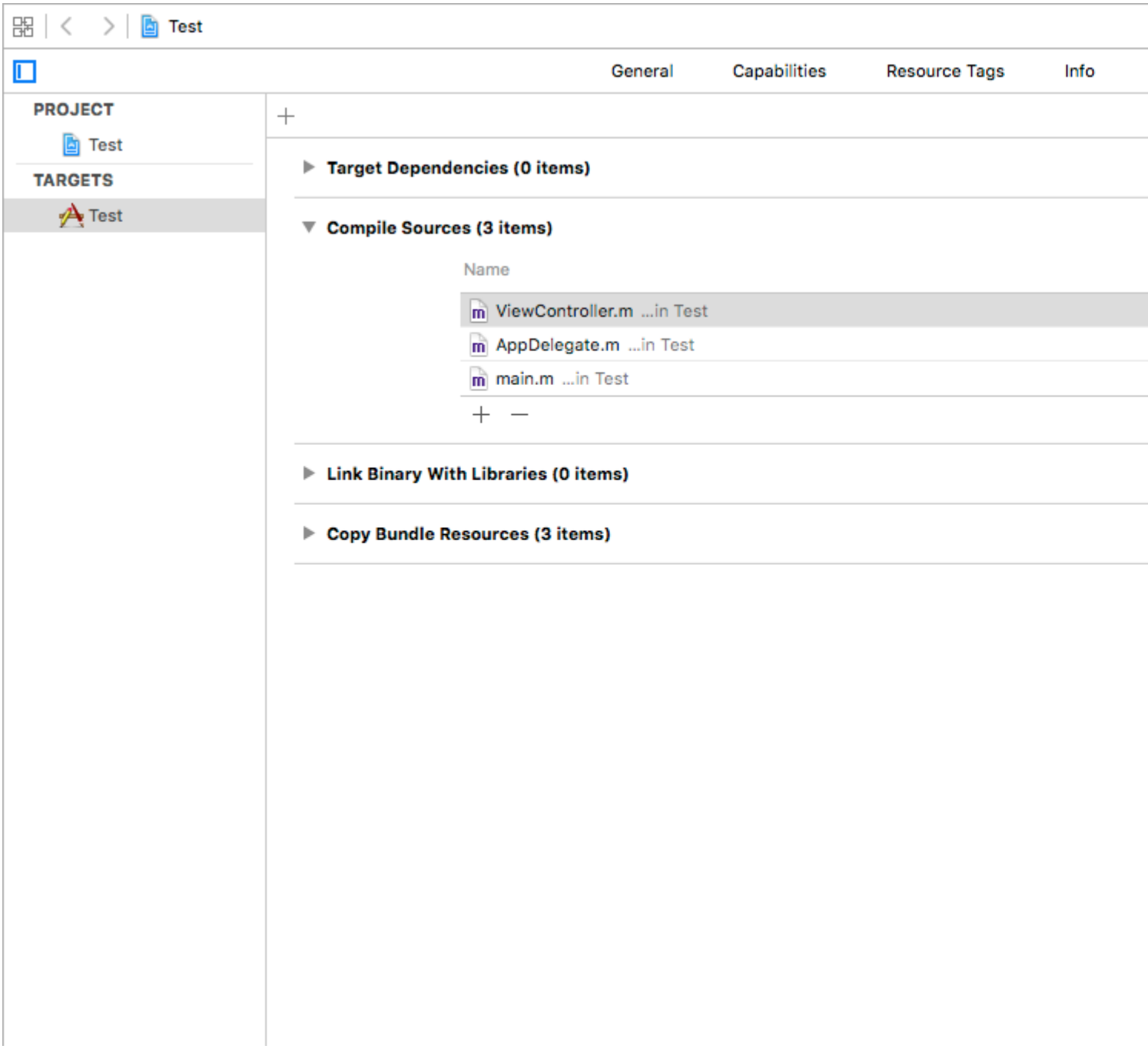
Lire AppDelegate en ligne: https://riptutorial.com/fr/ios/topic/8740/appdelegate

Chapitre 13: ARC (comptage automatique des références)

Examples

Activer / désactiver ARC sur un fichier

L'ARC peut être désactivé pour des fichiers individuels en ajoutant l' `-fno-objc-arc` compilation `-fno-objc-arc` pour chaque fichier. Inversement, il peut être ajouté dans *Cibles* ▸ `Phases de construction` ▸ `Compiler les sources`



Lire ARC (comptage automatique des références) en ligne:
https://riptutorial.com/fr/ios/topic/4150/arc--comptage-automatique-des-references-

Chapitre 14: Architecture MVP

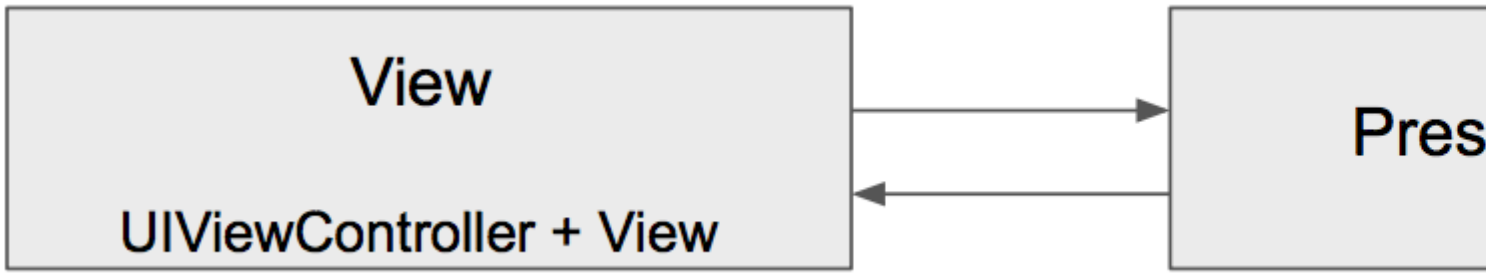
Introduction

MVP est un modèle architectural, une dérivation du modèle – View – Controller. Il est représenté par trois composants distincts: Modèle, Vue et Présentateur. Il a été conçu pour faciliter les tests unitaires automatisés et améliorer la séparation des problèmes dans la logique de présentation.

Dans des exemples, vous trouverez un projet simple conçu en fonction du modèle MVP.

Remarques

Composants:



- **Le modèle** est une interface responsable des données du domaine (à afficher ou à utiliser dans l'interface graphique)
- **View** est responsable de la couche de présentation (GUI)
- **Le présentateur** est le "intermédiaire" entre le modèle et la vue. Il réagit aux actions de l'utilisateur effectuées sur la vue, récupère les données du modèle et les formate pour les afficher dans la vue.

Devoir du composant:

Modèle	Vue	Présentateur
Communique avec le calque DB	Rend les données	Effectue des requêtes sur le modèle
Élever des événements appropriés	Reçoit des événements	Données de format du modèle
	Logique de validation très basique	Envoie des données formatées à la vue
		Logique de validation complexe

Différences entre MVC et MVP :

- View in MVC est étroitement lié au Controller, la partie View du MVP comprend à la fois UIViews et UIViewController
- MVP View est aussi stupide que possible et ne contient pratiquement aucune logique (comme dans MVVM), MVC View possède une certaine logique métier et peut interroger le modèle
- MVP View gère les gestes des utilisateurs et délègue les interactions au présentateur, dans MVC, le contrôleur gère les gestes et les commandes du modèle.
- Le modèle MVP supporte fortement les tests unitaires, MVC a un support limité
- MVC Controller a beaucoup de dépendances UIKit, MVP Presenter n'en a pas

Avantages:

- MVP fait de UIViewController une partie du composant View, il est stupide, passif et moins massif;]
- La plupart de la logique métier est encapsulée en raison des vues stupides, ce qui donne une excellente testabilité. Des objets simulés peuvent être introduits pour tester la partie domaine.
- Les entités séparées sont plus faciles à garder en tête, les responsabilités sont clairement divisées.

Les inconvénients

- Vous allez écrire plus de code.
- Barrière pour les développeurs inexpérimentés ou pour ceux qui ne travaillent pas encore avec le modèle.

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;
    }
}
```

Lire Architecture MVP en ligne: https://riptutorial.com/fr/ios/topic/9467/architecture-mvp

Chapitre 15: attributTexte dans UILabel

Introduction

Le texte stylé actuel affiché par l'étiquette.

Vous pouvez ajouter du texte `HTML` dans `UILabel` utilisant la propriété attributeText ou un texte `UILabel` unique personnalisé avec des propriétés différentes

Examples

Texte HTML dans 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;
```

Définir une propriété différente pour le texte dans un seul UILabel

La première étape que vous devez effectuer consiste à créer un objet `NSMutableAttributedString` . La raison pour laquelle nous créons un `NSMutableAttributedString` au lieu de `NSAttributedString` est que cela nous permet d'y ajouter une chaîne.

```
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];
```

Sortie:

Lire attributTexte dans UILabel en ligne: https://riptutorial.com/fr/ios/topic/10927/attributtexte-dans-uilabel

Chapitre 16: AVPlayer et AVPlayerViewController

Remarques

importer AVKit, importer AVFoundation.

Examples

Lecture de médias avec AVPlayerViewController

Objectif 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];
```

Rapide

```
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()
}
```

Lecture multimédia en utilisant AVPlayer et AVPlayerLayer

Objectif 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];
```

Rapide

```
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()
```

Exemple AVPlayer

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

Lire AVPlayer et AVPlayerViewController en ligne: https://riptutorial.com/fr/ios/topic/5092/avplayer-et-avplayerviewcontroller

Chapitre 17: AVSpeechSynthesizer

Syntaxe

- AVSpeechSynthesizer () // Crée un synthétiseur vocal
- speaker.speakUtterance (speech) // Convertit le texte en discours

Paramètres

Paramètre	Détails
orateur	Objet AVSpeechSynthesizer
discours	Objet AVSpeechUtterance

Examples

Créer un texte de base à la parole

Utilisez la méthode `speakUtterance:` de `AVSpeechSynthesizer` pour convertir du texte en parole. Vous devez passer un objet `AVSpeechUtterance` à cette méthode, qui contient le texte que vous souhaitez `AVSpeechUtterance` .

Objectif c

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

Rapide

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

Lire AVSpeechSynthesizer en ligne: https://riptutorial.com/fr/ios/topic/1526/avspeechsynthesizer

Chapitre 18: AWS SDK

Examples

Télécharger une image ou une vidéo sur S3 en utilisant AWS SDK

Avant de commencer avec l'exemple, je vous recommande de créer un Singleton avec un membre de classe délégué afin de pouvoir télécharger un fichier en arrière-plan et laisser l'utilisateur continuer à utiliser votre application pendant le téléchargement des fichiers, même lorsque l'application est téléchargée. est le fond

Commençons d'abord par créer une énumération représentant la configuration 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"
}
```

Maintenant, nous devrions définir les informations d'identification lors du lancement de votre application pour la première fois, nous devrions donc les `AppDelegate` dans la méthode `AppDelegate` à la méthode `didFinishLaunchingWithOptions` (faites attention à définir votre région sur le `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)
}
```

Comme nous sommes déjà dans AppDelegate, nous devrions implémenter le rappel en arrière-plan géré par le kit AWS SDK:

```
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)
}
```

Maintenant, lorsque l'utilisateur déplacera l'application en arrière-plan, votre téléchargement se poursuivra.

Afin de télécharger le fichier à l'aide du kit AWS SDK, nous devrons écrire le fichier sur le périphérique et donner au SDK le chemin réel. Par souci d'exemple, imaginons que nous ayons un UIImage (qui pourrait aussi être une vidéo) et nous l'écrirons dans un dossier temporaire:

```
// 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 et FileName seront utilisés pour le téléchargement réel ultérieurement.

Il y aura 2 fermetures à définir par le SDK AWS,

1. `AWSS3TransferUtilityUploadCompletionHandlerBlock` - Une fermeture qui avertit lorsque le téléchargement est terminé (ou non)
2. `AWSS3TransferUtilityUploadProgressBlock` - Une fermeture qui notifie chaque octet envoyé

Si vous prévoyez d'avoir un singleton, vous devez définir ces types en tant que membres de la classe. L'implémentation devrait ressembler à ceci:

```
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)
    }
```

REMARQUE: Si vous utilisez un singleton, vous souhaiterez peut-être définir un délégué qui vous informera de la progression ou de la fin du fichier. Si vous n'utilisez pas un Singleton, vous pouvez créer une méthode statique qui aurait les types appropriés:

```
    static func uploadImageToS3(fileURL : NSURL,
                                fileName : String,
```

```
                progressFunctionUpdater : Float -> Void,
                            resultBlock : (NSError?) -> Void)
{
 //    Actual implementation .....
 //    ...
 //    ...
}
```

1. `progressFunctionUpdater` - rapportera à une fonction avec progression.
2. `resultBlock` - Si vous retournez nil puis que le téléchargement a été effectué avec succès, vous envoyez l'objet d'erreur

Mesdames et Messieurs, le téléchargement actuel:

```
        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
        }
```

Happy S3 uploading :)

Lire AWS SDK en ligne: https://riptutorial.com/fr/ios/topic/4734/aws-sdk

Chapitre 19: Barre de navigation

Examples

Personnaliser l'apparence de la barre de navigation par défaut.

```
// 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]];
```

Exemple 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
```

Lire Barre de navigation en ligne: https://riptutorial.com/fr/ios/topic/7066/barre-de-navigation

Chapitre 20: Bloc

Syntaxe

- En tant que variable:

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

- En tant que propriété:

  @property (nonatomique, copie) returnType (^ blockName) (parameterTypes);

- En tant que paramètre de méthode:

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

- En typedef:

  typedef returnType (^ TypeName) (parameterTypes);

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

Examples

Animations UIView

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

Le carat «^» définit un bloc. Par exemple, `^{ … }` est un bloc. Plus précisément, il s'agit d'un bloc qui renvoie «void» et n'accepte aucun argument. Cela équivaut à une méthode telle que: "- (void) quelquechose", mais il n'y a pas de nom inhérent associé au bloc de code.

Définir un bloc pouvant accepter des arguments fonctionne de manière très similaire. Pour fournir un argument à un bloc, vous définissez le bloc comme *suit* : ^ *(BOOL someArg, NSString someStr) {…}* *. Lorsque vous utilisez des appels API prenant en charge des blocs, vous allez écrire des blocs qui ressemblent à cela, en particulier pour les blocs d'animation ou les blocs NSURLConnection, comme indiqué dans l'exemple ci-dessus.

Bloc d'achèvement personnalisé pour les méthodes personnalisées

1- Définir votre propre bloc personnalisé

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

2- Créez une méthode personnalisée qui prend en compte votre bloc d'achèvement personnalisé.

```
-(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- Comment utiliser block dans votre méthode

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

Modifier la variable capturée

Block capturera les variables apparaissant dans la même portée lexicale. Normalement, ces variables sont capturées en tant que "const" valeur:

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

Pour modifier la variable, vous devez utiliser le modificateur de type de stockage __block.

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

Lire Bloc en ligne: https://riptutorial.com/fr/ios/topic/6888/bloc

Chapitre 21: Blocs de chaîne dans une file d'attente (avec MKBlockQueue)

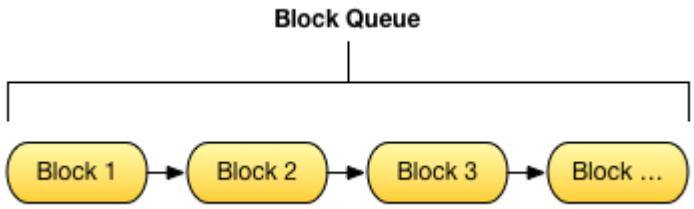
Introduction

MKBlockQueue vous permet de créer une chaîne de blocs et de les exécuter l'un après l'autre dans une file d'attente. Par rapport à NSOperation, avec MKBlockQueue, vous décidez vous-même quand un bloc est terminé et quand vous voulez que la file se poursuive. Vous pouvez également transmettre des données d'un bloc à l'autre.

https://github.com/MKGitHub/MKBlockQueue

Examples

Exemple de code




```
// 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)
```

Lire Blocs de chaîne dans une file d'attente (avec MKBlockQueue) en ligne: https://riptutorial.com/fr/ios/topic/9122/blocs-de-chaine-dans-une-file-d-attente--avec-mkblockqueue-

Chapitre 22: CAAnimation

Remarques

CAAnimation est une classe d'animation abstraite. Il fournit la prise en charge de base des protocoles **CAMediaTiming** et **CAAction** . Pour animer les couches Core Animation ou les objets Scene Kit, créez des instances des sous-classes concrètes **CABasicAnimation** , **CAKeyframeAnimation** , **CAAnimationGroup** ou **CATransition** .

Examples

Animer une vue d'une position à une autre.

Objectif c

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

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

Rapide

```
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 vue passera de 0 à 320 horizontalement. Si vous souhaitez déplacer la vue dans Verticalement, remplacez le raccourci clavier comme ceci:

```
"position.y"
```

Animer la vue - Lancer

OBJECTIF 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"];
```

RAPIDE

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

Revolve View

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

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

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

Shake View

Objectif 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")
```

Animation Push View

Objectif 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"];
```

Rapide

```
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")
```

Lire CAAnimation en ligne: https://riptutorial.com/fr/ios/topic/981/caanimation

Chapitre 23: Cache les images en ligne

Examples

AlamofireImage

Mise en cache des images en ligne avec `AlamofireImage` . Il travaille sur `Alamofire` à Swift. Installez `AlamofireImage` utilisant des `cocoapods`

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

Installer:

1. Importer `AlamofireImage` et `Alamofire`
2. Configurer le cache image: `let imageCache = AutoPurgingImageCache( memoryCapacity: 111_111_111, preferredMemoryUsageAfterPurge: 90_000_000)`
3. Faire une demande et ajouter l'image au 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. Récupérer des images du cache:

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

Pour plus d'informations, suivez ce lien

Lire Cache les images en ligne en ligne: https://riptutorial.com/fr/ios/topic/9450/cache-les-images-en-ligne

Chapitre 24: Cadre de contacts

Remarques

Liens utiles

- Documentation Apple
- Dépassement de la pile
- Vidéo de la session WWDC15

Examples

Autoriser l'accès aux contacts

Importer le framework

Rapide

```
import Contacts
```

Objectif c

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

Vérification de l'accessibilité

Rapide

```
switch CNContactStore.authorizationStatusForEntityType(CNEntityType.Contacts){
case .Authorized: //access contacts
case .Denied, .NotDetermined: //request permission
default: break
}
```

Objectif c

```
switch ([CNContactStore authorizationStatusForEntityType:CNEntityType.Contacts]){
```

```
case CNAuthorizationStatus.Authorized:
    //access contacts
    break;
case CNAuthorizationStatus.Denied:
    //request permission
    break;
case CNAuthorizationStatus.NotDetermined:
    //request permission
    break;
}
```

Demande de permission

Rapide

```
var contactStore = CKContactStore()
contactStore.requestAccessForEntityType(CKEntityType.Contacts, completionHandler: { (ok, _) ->
Void in
    if access{
        //access contacts
    }
}
```

Accéder aux contacts

Appliquer un filtre

Pour accéder aux contacts, nous devons appliquer un filtre de type `NSPredicate` à notre variable contactStore que nous avons définie dans l'exemple Authorizing Contact Access. Par exemple, ici, nous voulons trier les contacts avec un nom correspondant au nôtre:

Rapide

```
let predicate = CNContact.predicateForContactsMatchingName("Some Name")
```

Objectif c

```
NSPredicate *predicate = [CNContact predicateForContactsMatchingName:@"Some Name"];
```

Spécifier les clés à récupérer

Ici, nous voulons récupérer le prénom, le nom et l'image de profil du contact:

Rapide

```
let keys = [CNContactGivenNameKey, CNContactFamilyNameKey, CNContactImageDataKey]
```

Récupérer des contacts

Rapide

```
do {
    let contacts = try contactStore.unifiedContactsMatchingPredicate(predicate, keysToFetch:
keys)
} catch let error as NSError {
    //...
}
```

Accéder aux coordonnées

Rapide

```
print(contacts[0].givenName)
print(contacts[1].familyName)
let image = contacts[2].imageData
```

Ajouter un contact

Rapide

```
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)
```

Lire Cadre de contacts en ligne: https://riptutorial.com/fr/ios/topic/5872/cadre-de-contacts

Chapitre 25: CAGradientLayer

Syntaxe

- CAGradientLayer () // Retourne un objet CALayer initialisé.
- CAGradientLayer (layer: layer) // Remplace pour copier ou initialiser des champs personnalisés du calque spécifié.

Paramètres

Paramètre	Détails
Couleur	Un tableau d'objets `CGColorRef` définissant la couleur de chaque arrêt de dégradé. Animable
Emplacements	Un tableau facultatif d'objets `NSNumber` définissant l'emplacement de chaque arrêt de dégradé. Animable
endPoint	Le point final du dégradé lorsqu'il est dessiné dans l'espace de coordonnées du calque. Animable
point de départ	Le point de départ du dégradé lorsqu'il est dessiné dans l'espace de coordonnées du calque. Animable
type	Style de dégradé dessiné par le calque. La valeur par défaut est `kCAGradientLayerAxial` .

Remarques

- Utilisez `startPoint` et `endPoint` pour modifier l'orientation de `CAGradientLayer` .
- Utilisez les `locations` pour affecter la répartition / les positions des couleurs.

Examples

Créer 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)
```

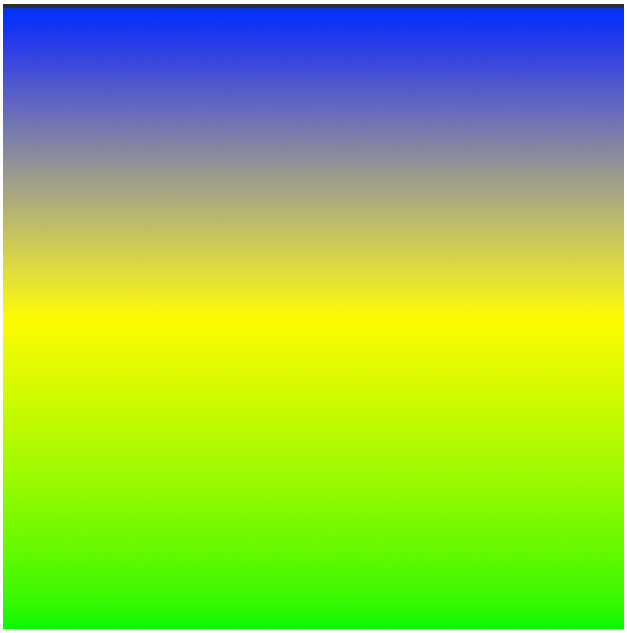
Résultat :

Créer un CGGradientLayer avec plusieurs couleurs.

```
// 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)
```

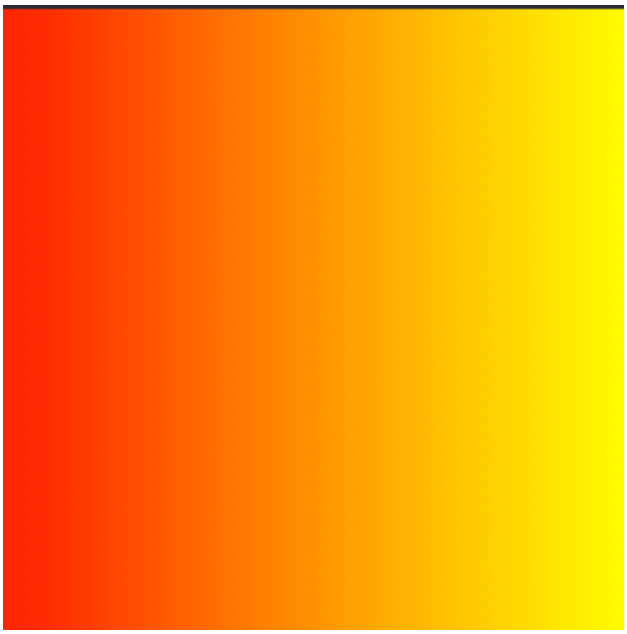
Résultat :



Créer un CAGradientLayer horizontal.

```
// 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)
```

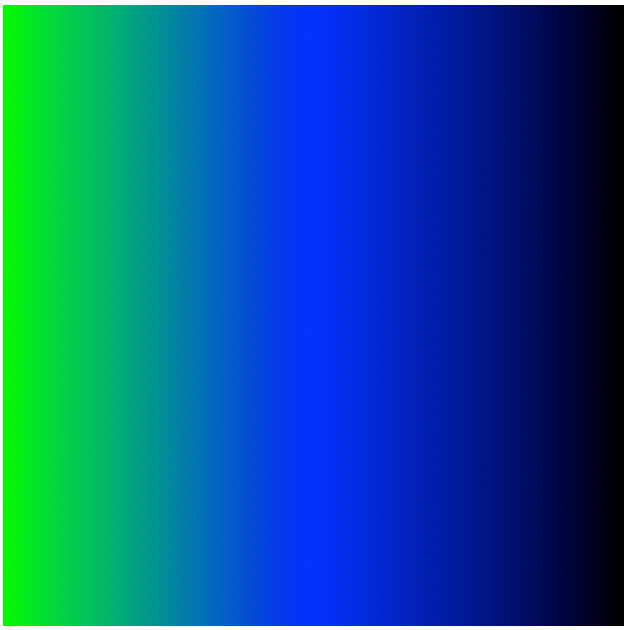
Résultat :



Créer un CAGradientLayer horizontal avec plusieurs couleurs.

```
// 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)
```

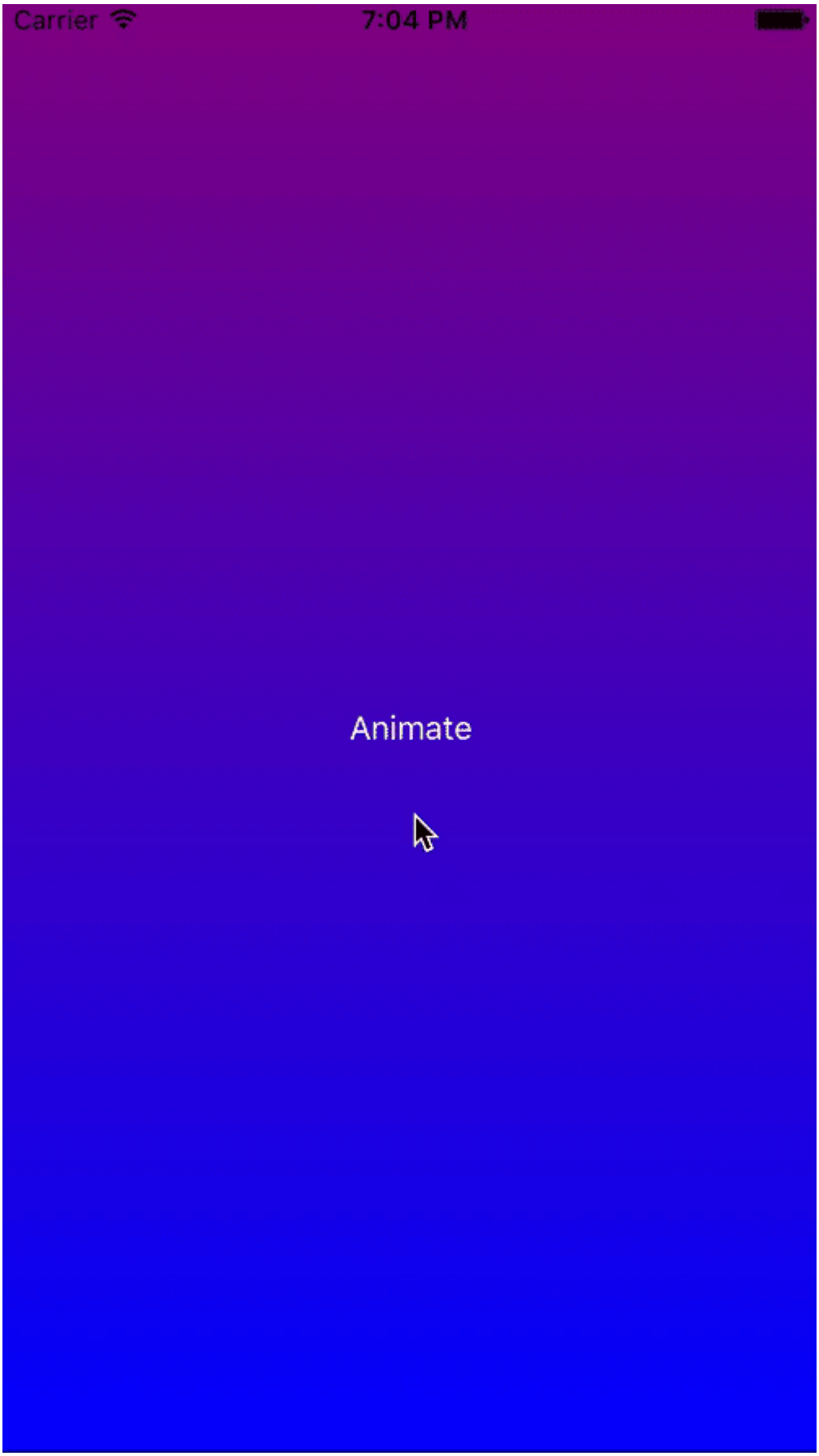
Résultat :



Animation d'un changement de couleur dans 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")
```

Résultat :



Lire CAGradientLayer en ligne: https://riptutorial.com/fr/ios/topic/1190/cagradientlayer

Chapitre 26: CALayer

Examples

Créer un calayer

Vous pouvez créer un calayer et définir son cadre comme suit:

Rapide:

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

Objectif c:

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

Vous pouvez ensuite l'ajouter en tant que sous-couche à un CALayer existant:

Rapide:

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

Objectif c:

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

Remarque:

Pour ce faire, vous devez inclure le framework QuartzCore.

Rapide:

```
 @import QuartzCore
```

Objectif c

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

Création de particules avec CAEmitterLayer

La classe **CAEmitterLayer** fournit un système émetteur de particules pour Core Animation. Les particules sont définies par des instances de **CAEmitterCell** .

Les particules sont dessinées au-dessus de la couleur d'arrière-plan et de la bordure du calque.

```
        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)
```

Emitter View avec une image personnalisée

Par exemple, nous allons créer une vue contenant une couche d'émetteur et animer des particules.

```
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
    }
}
```

Vous devez ajouter une image "confetti" ou définir rect avec **confetti.contentsRect**

Comment ajouter un UIImage à un CALayer

Vous pouvez ajouter une image au `layer` une vue en utilisant simplement sa propriété de `contents` :

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

- Notez que l' `UIImage` doit être converti en un `CGImage` .

Si vous souhaitez ajouter l'image dans son propre calque, vous pouvez le faire comme ceci:

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

Modifier l'apparence

Le code ci-dessus produit une vue comme celle-ci. Le bleu clair est l' `UIView` et l'étoile bleu foncé est l' `UIImage` .

Comme vous pouvez le voir, cependant, il semble pixélisé. Ceci est dû au fait que `UIImage` est plus petit que `UIView` , il est donc mis à l'échelle pour remplir la vue, ce qui est la valeur par défaut que vous ne spécifiez pas.

Les exemples ci-dessous présentent des variantes sur la propriété de `contentsGravity` du calque. Le code ressemble à ceci:

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

Dans iOS, vous souhaiterez peut-être définir la propriété `geometryFlipped` sur `true` si vous faites quelque chose avec la gravité supérieure ou inférieure, sinon ce sera le contraire de ce que vous attendez. (Seule la gravité est retournée verticalement, pas le rendu du contenu. Si vous rencontrez des problèmes avec le contenu retourné, consultez cette réponse Stack Overflow .)

Il y a deux exemples `UIView` ci-dessous pour chaque `contentsGravity` Réglage de la `UIImage` , une vue est plus grande que l' `UIImage` et l'autre est plus petite. De cette façon, vous pouvez voir les effets de la mise à l'échelle et de la gravité.

`kCAGravityResize`

Ceci est la valeur par défaut.

`kCAGravityResizeAspect`

kCAGravityResizeAspectFill

kCAGravityCenter

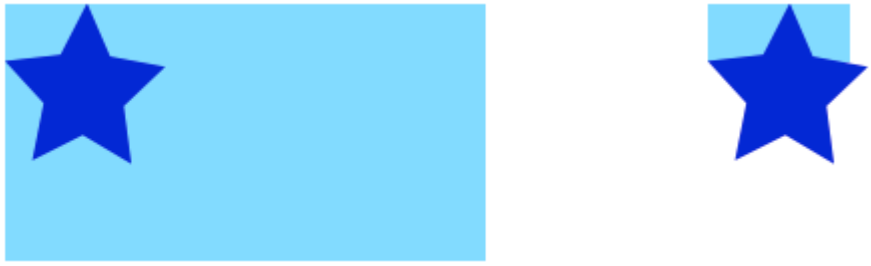
kCAGravityTop

kCAGravityBottom

kCAGravityLeft

kCAGravityRight

kCAGravityTopLeft



kCAGravityTopRight

kCAGravityBottomLeft

kCAGravityBottomRight

en relation

- Propriété du mode contenu d'une vue
- Dessiner un `UIImage` dans `drawRect` avec `CGContextDrawImage`
- Tutoriel CALayer: Mise en route

Remarques

- Cet exemple provient de cette réponse Stack Overflow .

Ajout de transformations à un calayer (traduction, rotation, mise à l'échelle)

Les bases

Il y a un certain nombre de transformations différentes que vous pouvez faire sur une couche, mais les bases sont

- traduire (déplacer)
- échelle
- tourner

Pour effectuer des transformations sur un `CALayer` , définissez la propriété `transform` la couche sur un type `CATransform3D` . Par exemple, pour traduire un calque, vous feriez quelque chose comme ceci:

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

Le mot `Make` est utilisé dans le nom pour créer la transformation initiale: CATransform3D **Make** Translation. Les transformations ultérieures qui sont appliquées omettent le `Make` . Voir, par exemple, cette rotation suivie d'une traduction:

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

Maintenant que nous avons la base de la manière de transformer, examinons quelques exemples de la façon de procéder. D'abord, cependant, je montrerai comment j'ai configuré le projet au cas où vous voudriez y jouer.

Installer

Pour les exemples suivants, j'ai configuré une application à `UIView` unique et ajouté un `UIView` avec un arrière-plan bleu clair au storyboard. J'ai branché la vue au contrôleur de vue avec le code suivant:

```
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 ...


    }

}
```

Il existe de nombreux types de `CALayer` , mais j'ai choisi d'utiliser `CATextLayer` pour que les transformations soient plus claires visuellement.

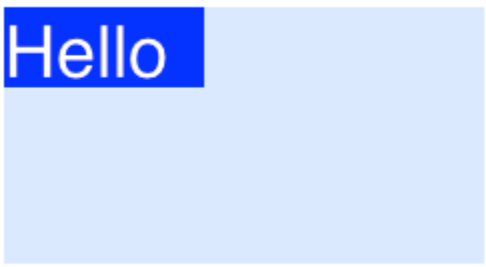
Traduire

La transformation de traduction déplace le calque. La syntaxe de base est

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

où `tx` est le changement des coordonnées x, `ty` est le changement de y et `tz` est le changement de z.

Exemple



Dans iOS, l'origine du système de coordonnées est en haut à gauche, donc si nous voulions déplacer la couche de 90 points vers la droite et de 50 points vers le bas, nous procéderions

comme suit:

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

Remarques

- N'oubliez pas que vous pouvez le coller dans la méthode `transformExample()` du code du projet ci-dessus.
- Puisque nous ne faisons que traiter de deux dimensions, `tz` est défini sur `0` .
- La ligne rouge dans l'image ci-dessus va du centre de l'emplacement d'origine au centre du nouvel emplacement. En effet, les transformations sont effectuées par rapport au point d'ancrage et le point d'ancrage par défaut au centre du calque.

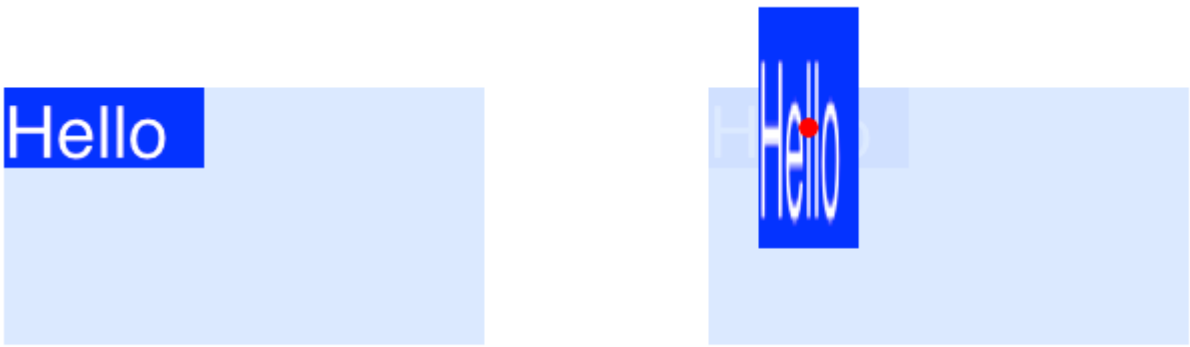
Échelle

La transformation d'échelle étend ou écrase le calque. La syntaxe de base est

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

où `sx` , `sy` et `sz` sont les nombres par lesquels mettre à l'échelle (multiplier) respectivement les coordonnées x, y et z.

Exemple



Si nous voulions réduire de moitié la largeur et tripler la hauteur, nous ferions ce qui suit

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

Remarques

- Comme nous ne travaillons que dans deux dimensions, nous multiplions simplement les coordonnées z par 1,0 pour ne pas les affecter.
- Le point rouge dans l'image ci-dessus représente le point d'ancrage. Remarquez comment la mise à l'échelle est effectuée par rapport au point d'ancrage. C'est-à-dire que tout est soit tendu vers ou loin du point d'ancrage.

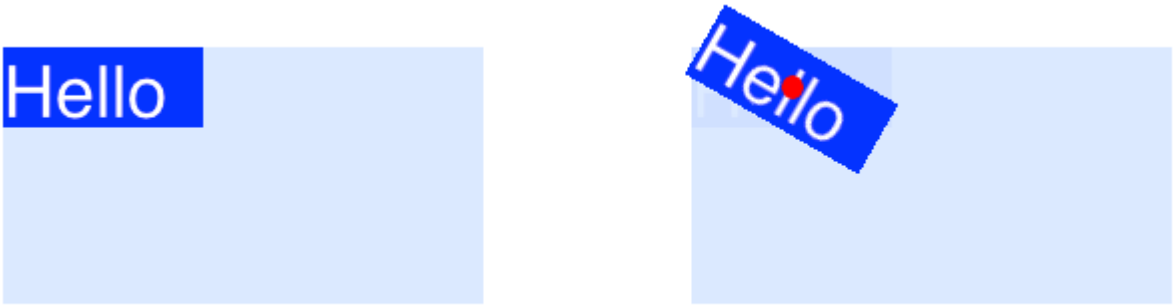
Tourner

La transformation de rotation fait pivoter le calque autour du point d'ancrage (le centre du calque par défaut). La syntaxe de base est

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

où `angle` est l'angle en radians avec lequel le calque doit être pivoté et `x` , `y` et `z` sont les axes autour desquels faire pivoter. Définir un axe sur 0 annule une rotation autour de cet axe particulier.

Exemple



Si nous voulions tourner une couche de 30 degrés dans le sens des aiguilles d'une montre, nous ferions ce qui suit:

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

Remarques

- Puisque nous travaillons en deux dimensions, nous voulons seulement que le plan xy soit pivoté autour de l'axe z. Nous définissons donc `x` et `y` à `0.0` et définissons `z` à `1.0` .
- Cela a fait pivoter la couche dans le sens des aiguilles d'une montre. Nous aurions pu tourner dans le sens inverse des aiguilles d'une montre en définissant `z` sur `-1.0` .
- Le point rouge indique où se trouve le point d'ancrage. La rotation se fait autour du point d'ancrage.

Transformations multiples

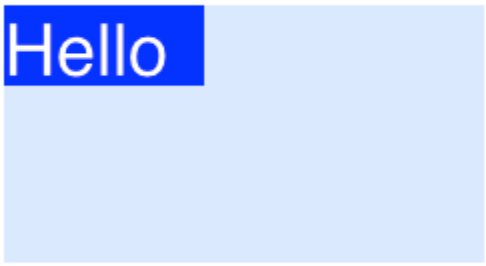
Afin de combiner plusieurs transformations, nous pourrions utiliser la concatination comme ceci

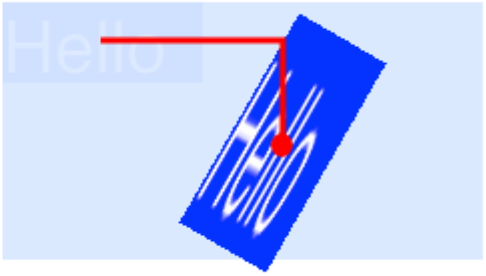
```
CATransform3DConcat(a: CATransform3D, b: CATransform3D)
```

Cependant, nous ne ferons que l'un après l'autre. La première transformation utilisera le `Make` dans

son nom. Les transformations suivantes n'utiliseront pas `Make` , mais elles prendront la transformation précédente en tant que paramètre.

Exemple





Cette fois, nous combinons les trois transformations précédentes.

```
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
```

Remarques

- L'ordre dans lequel les transformations se font dans les matières.
- Tout a été fait par rapport au point d'ancrage (point rouge).

Une remarque sur le point d'ancrage et la position

Nous avons fait toutes nos transformations ci-dessus sans changer le point d'ancrage. Parfois, il est nécessaire de le changer, par exemple si vous souhaitez effectuer une rotation autour d'un autre point que le centre. Cependant, cela peut être un peu difficile.

Le point d'ancrage et la position sont tous deux au même endroit. Le point d'ancrage est exprimé comme une unité du système de coordonnées de la couche (la valeur par défaut est `0.5, 0.5`) et la position est exprimée dans le système de coordonnées de la super-couche. Ils peuvent être mis comme ça

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

Si vous définissez uniquement le point d'ancrage sans modifier la position, le cadre change pour que la position soit au bon endroit. Ou plus précisément, la trame est recalculée en fonction du nouveau point d'ancrage et de l'ancienne position. Cela donne généralement des résultats inattendus. Les deux articles suivants ont une excellente discussion à ce sujet.

- A propos du point d'ancrage
- Traduire tourner traduit?

Voir également

- Bordure, coins arrondis et ombre sur un `CALayer`
- Utiliser une bordure avec un chemin de Bézier pour un calque

Cet exemple provient à l'origine de cet exemple de débordement de pile .

Désactiver les animations

`CALayer` animations de propriétés `CALayer` sont activées par défaut. Lorsque cela est indésirable, ils peuvent être désactivés comme suit.

Rapide

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

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

CATransaction.commit()
```

Objectif c

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

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

[CATransaction commit];
```

Coins arrondis

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

Ombres

Vous pouvez utiliser 5 propriétés sur chaque couche pour configurer vos ombres:

- `shadowOffset` - cette propriété déplace votre ombre gauche / droite ou haut / bas

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

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

- `shadowColor` - définit la couleur de votre ombre

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

- `shadowOpacity` - c'est l'opacité de l'ombre, de `0` à `1`

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

- `shadowRadius` - c'est le rayon de flou (équivalent de la propriété de flou dans Sketch ou Photoshop)

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

- `shadowPath` - il s'agit d'une propriété importante pour les performances, lorsque iOS non défini fonde l'ombre sur le canal alpha de la vue, ce qui peut nécessiter de nombreuses performances avec un format PNG complexe avec alpha. Cette propriété vous permet de forcer une forme pour votre ombre et d'être plus performant grâce à cela.

Objectif 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
```

Lire CALayer en ligne: https://riptutorial.com/fr/ios/topic/1462/calayer

Chapitre 27: CAShapeLayer

Syntaxe

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

Remarques

La classe CAShapeLayer dessine une spline Bézier cubique dans son espace de coordonnées. La forme est composée entre le contenu du calque et sa première sous-couche.

Examples

Fonctionnement de base de CAShapeLayer

UIBezierPath utilisant pour créer un chemin circulaire 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
```

Supprimer ShapeLayer

Gardez une référence à cette couche de forme. Par exemple, vous pouvez avoir une propriété currentShapeLayer: Maintenant que vous avez une référence, vous pouvez facilement supprimer la couche:

Type 1:

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

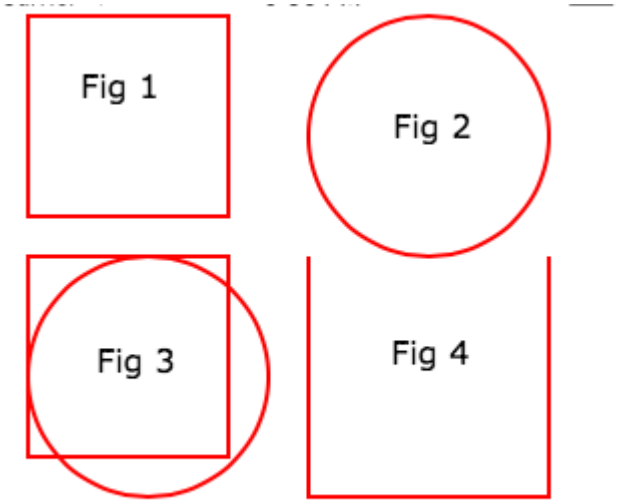
Type 2:

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

Autre opération

```
 //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)];
```

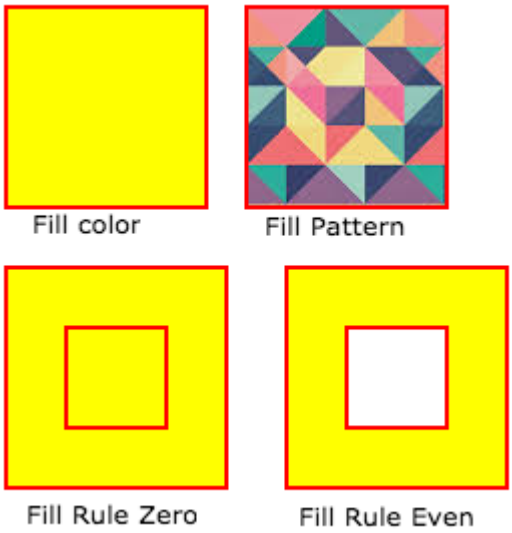
```
[linePath addLineToPoint:CGPointMake(120 , 0)];
shapeLayer.path = linePath.CGPath;
[[self.view layer] addSublayer:shapeLayer];
```



Remplir les concepts // Couleur de remplissage

```
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];
```



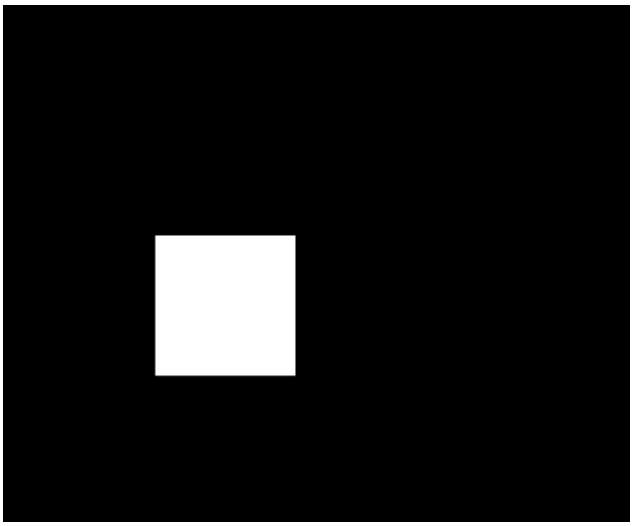
Répertorié les propriétés de style d'accès

```
   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.
```

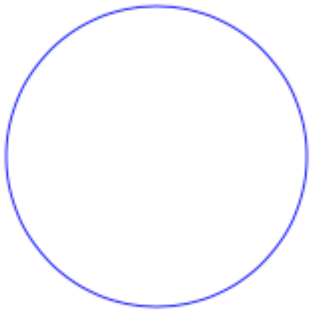
Dessiner un rectangle

```
 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;
```



Dessiner un cercle

```
 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];
```



Animation 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"];
```

Lire CAShapeLayer en ligne: https://riptutorial.com/fr/ios/topic/3575/cashapelayer

Chapitre 28: Catégories

Remarques

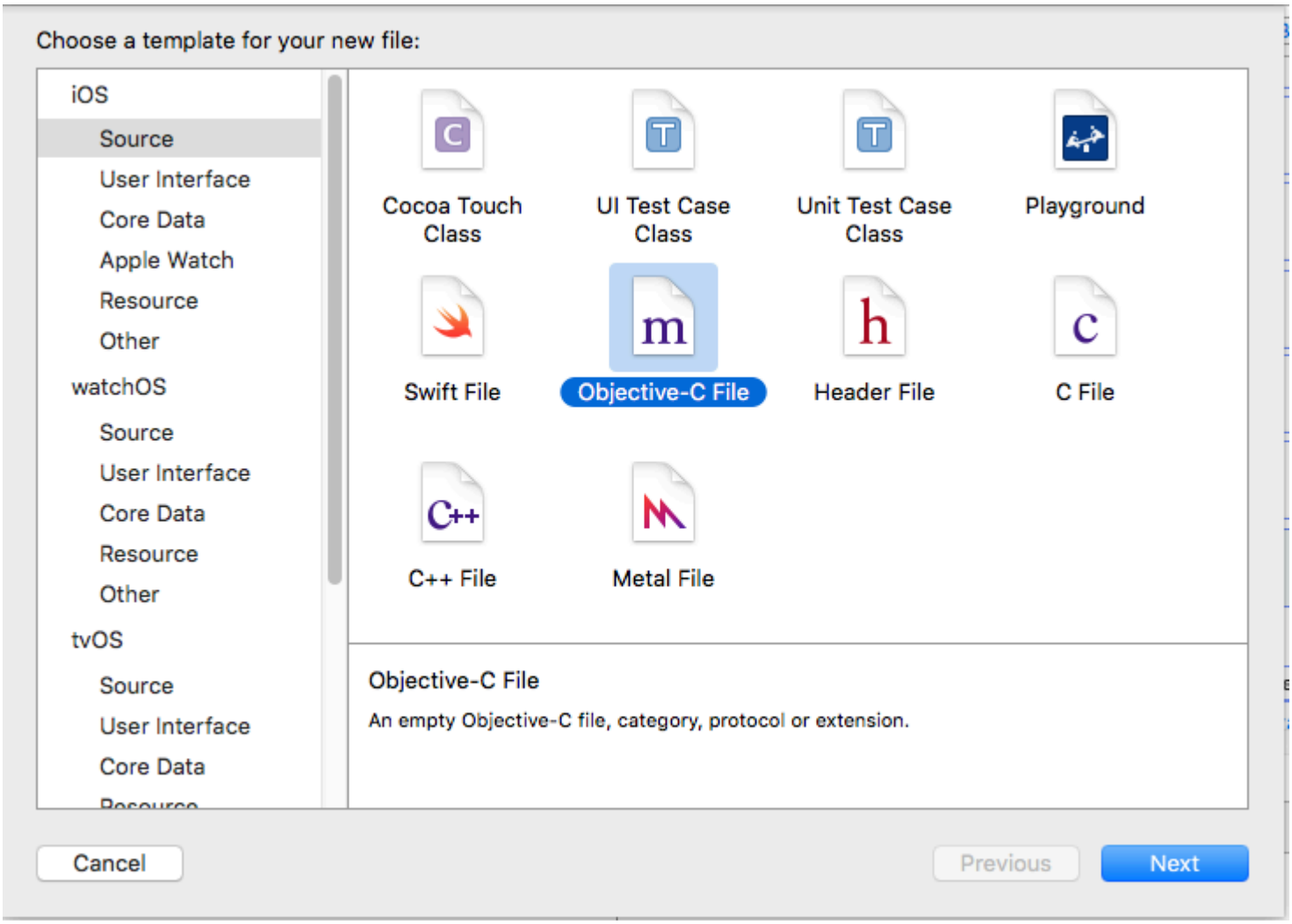
Les catégories peuvent être utilisées pour remplacer les méthodes d'une classe. Même si la méthode est en réalité privée. La méthode surchargée n'est pas accessible depuis la catégorie ou ailleurs. Il est donc important de s'assurer que lors de l'ajout de méthodes à une classe existante, ces méthodes n'existent pas déjà.

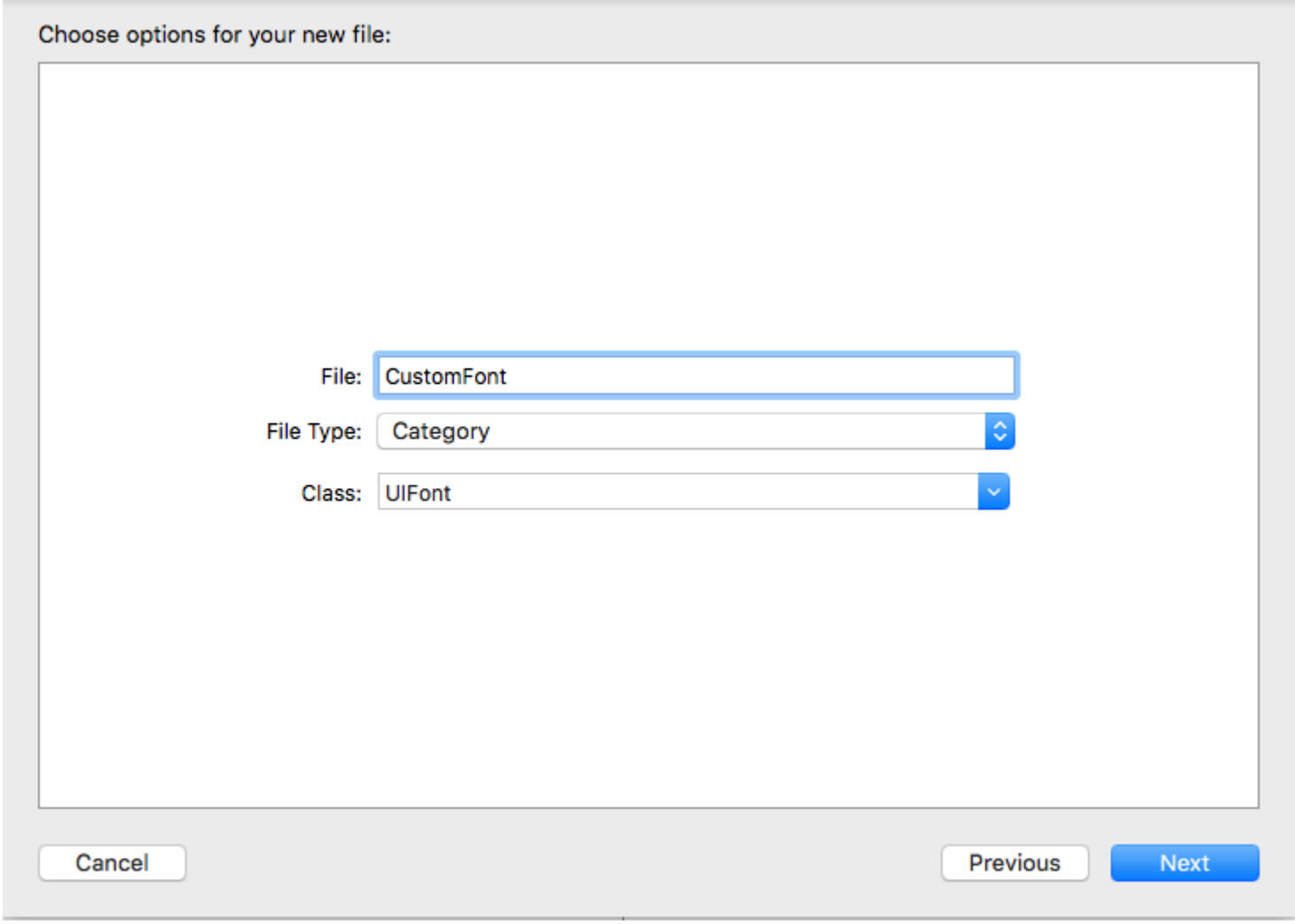
Examples

Créer une catégorie

Les catégories permettent d'ajouter des fonctionnalités supplémentaires à un objet sans sous-classer ni modifier l'objet réel.

Par exemple, nous voulons définir des polices personnalisées. Permet de créer une catégorie qui ajoute des fonctionnalités à la classe `UIFont` . Ouvrez votre projet Xcode, cliquez sur Fichier -> Nouveau -> Fichier et choisissez le fichier Objective-C, cliquez sur Suivant, entrez le nom de votre catégorie, puis «CustomFont», choisissez Type et Class comme UIFont, puis cliquez sur Suivant.
"


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



Déclarez la méthode de la catégorie: -

Cliquez sur "UIFont + CustomFonts.h" pour afficher le fichier d'en-tête de la nouvelle catégorie. Ajoutez le code suivant à l'interface pour déclarer la méthode.

```
@interface UIFont (CustomFonts)

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

@end
```

Maintenant, implémentez la méthode de la catégorie: -

Cliquez sur "UIFont + CustomFonts.m" pour afficher le fichier d'implémentation de la catégorie. Ajoutez le code suivant pour créer une méthode définissant la police ProductSansRegular.

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

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

}
```

Importez votre catégorie

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

Maintenant, définissez la police d'étiquette

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

Lire Catégories en ligne: https://riptutorial.com/fr/ios/topic/3633/categories

Chapitre 29: Changer la couleur de la barre d'état

Examples

Pour les barres d'état non-UINavigationBar

1. Dans info.plist, définissez l' `View controller-based status bar appearance` sur `YES`
2. Dans la vue, les contrôleurs non contenus dans `UINavigationController` implémentent cette méthode.

En Objective-C:

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

En Swift:

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

Pour les barres d'état UINavigationBar

Sous-classe UINavigationController, puis remplacez ces méthodes:

En Objective-C:

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

En Swift:

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

Vous pouvez également définir `barStyle` sur l'instance `UINavigationBar` :

Objectif c:

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

```
white status bar
```

Rapide

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

`UIBarStyle` options `UIBarStyle` sont les options `default` , `black` , `blackOpaque` , `blackTranslucent` . Le dernier 3 devrait tous vous donner une barre d'état avec du texte blanc, juste les deux derniers spécifient l'opacité de la barre.

Remarque: vous pouvez toujours modifier l'apparence de votre barre de navigation comme vous le souhaitez.

Si vous ne pouvez pas modifier le code de ViewController

Si vous utilisez une bibliothèque qui contient (par exemple) AwesomeViewController avec une couleur de barre d'état incorrecte, vous pouvez essayer ceci:

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

Pour le confinement de ViewController

Si vous utilisez `UIViewControllerContainment` il existe quelques autres méthodes qui méritent d'être examinées.

Lorsque vous souhaitez qu'un viewController enfant contrôle la présentation de la barre d'état (par exemple, si l'enfant est positionné en haut de l'écran)

à 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
    }
}
```

Modification du style de la barre d'état pour toute l'application

RAPIDE:

Étape 1:

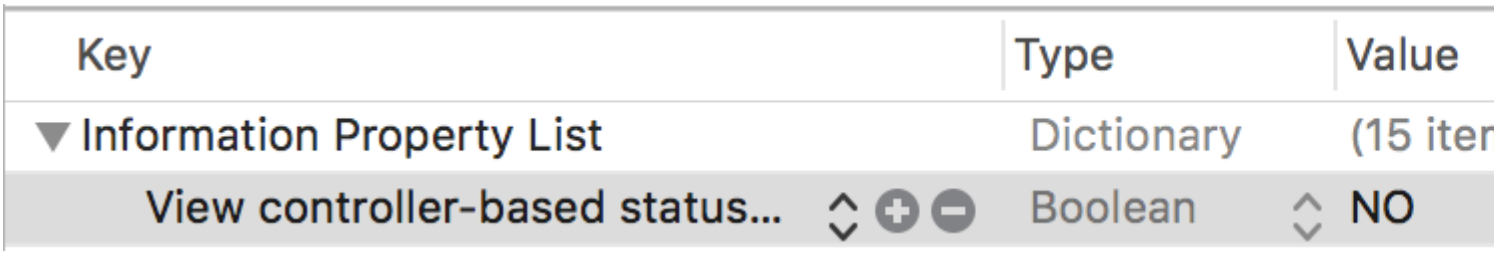
Dans votre **Info.plist,** ajoutez l'attribut suivant:

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

et définir sa valeur à

```
NO
```

comme décrit dans l'image ci-dessous:



Étape 2:

Dans votre fichier **AppDelegate.swift** , dans la méthode `didFinishLaunchingWithOptions` , ajoutez ce code:

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

ou

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

- L'option **.lightContent** définira la couleur du **statusBar** sur blanc pour toute l'application.
- L'option **.default** définira la couleur du **statusBar** sur la couleur noire d'origine, pour toute

l'application.

OBJECTIF C:

Suivez la première étape depuis la section **SWIFT** . Ajoutez ensuite ce code au fichier **AppDelegate.m** :

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

ou

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

Lire Changer la couleur de la barre d'état en ligne: https://riptutorial.com/fr/ios/topic/378/changer-la-couleur-de-la-barre-d-etat

Chapitre 30: Charger des images asynchrones

Examples

Manière la plus simple

La manière la plus simple de créer ceci est d'utiliser Alamofire et son UIImageViewExtension . Ce dont nous avons besoin, c'est d'une tableview avec une cellule qui contient une imageView et permet de l'appeler `imageView` .

Dans la fonction cellForRowAt: de la table, nous téléchargerions l'image et la définirions de la manière suivante:

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

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

L'URL doit pointer sur l'image que vous souhaitez télécharger et l'image placeHolder doit être une image stockée. Nous appelons ensuite la méthode `af_setImage` sur l' `imageView` qui télécharge l'image à l'URL donnée et, pendant le téléchargement, l'image de l'espace réservé sera affichée. Dès que l'image est téléchargée, l'image demandée est affichée

Vérifiez que la cellule est toujours visible après le téléchargement

Parfois, le téléchargement prend plus de temps que l'affichage de la cellule. Dans ce cas, il peut arriver que l'image téléchargée soit affichée dans la mauvaise cellule. Pour résoudre ce problème, nous ne pouvons pas utiliser l' extension UIImageView .

Nous utiliserons toujours Alamofire, mais nous utiliserons le gestionnaire d'achèvement pour afficher l'image.

Dans ce scénario, nous avons toujours besoin d'une tableView avec une cellule contenant une imageView. Dans la méthode cellForRowAt: nous téléchargerions l'image avec le code suivant:

```
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
    }
}
```

Dans cet exemple, nous définissons d'abord l'image sur l'image de l'espace réservé. Ensuite, nous téléchargeons l'image avec la méthode de `request` d' Alamofire . Nous passons l'URL comme premier argument et comme nous voulons juste obtenir l'image, nous utiliserons la méthode HTTP `.get` . Comme nous téléchargeons une image, nous voulons que la réponse soit une image. Nous utilisons donc la méthode `.responseImage` .

Une fois l'image téléchargée, la fermeture est appelée et tout d'abord, nous nous assurons que l'image téléchargée existe réellement. Ensuite, nous nous assurons que la cellule est toujours visible en vérifiant que le cellForRow (at: indexPath) ne retourne pas nil. Si cela ne se produit rien, si ce n'est pas le cas, nous attribuons l'image récemment téléchargée.

Cette dernière instruction if garantit que la cellule est toujours visible si l'utilisateur a déjà fait défiler la cellule pour que updateCell soit nulle et que l'instruction if renvoie nil. Cela nous aide à éviter d'afficher la mauvaise image dans une cellule.

Lire Charger des images asynchrones en ligne: https://riptutorial.com/fr/ios/topic/10793/charger-des-images-asynchrones

Chapitre 31: Classements GameCenter

Examples

Classements GameCenter

Conditions préalables:

1. Compte des développeurs Apple
2. Configuration des classements GameCenter avec iTunesConnect

Configuration des classements GameCenter:

1. Connectez-vous à *iTunesConnect*
2. Accédez à *Mes applications* . Créez une application pour votre projet, puis accédez à *Fonctionnalités* .
3. Cliquez sur *Game Center*
4. Cliquez sur le signe plus à côté de Leaderboards.
5. Choisissez le *classement unique* pour les types de classement.
6. Créez un *nom de référence de classement* pour votre référence.
7. Créez un *ID de classement* auquel votre application peut faire référence lors de la génération de rapports.
8. Définir le format de partition sur *Integer*
9. La soumission de score sera le *meilleur score*
10. Cliquez sur *Ajouter une langue* et remplissez les entrées.

Copiez votre `LeaderboardID` que vous avez créé et laissez-vous aller à Xcode.

Travailler avec Xcode

Il y a 4 fonctions avec lesquelles nous allons travailler.

1. Importer le framework et configurer les protocoles
2. Vérifier si l'utilisateur est connecté à GameCenter
3. Signaler les scores à GameCenter
4. Affichage des classements
5. Importation de l' `import GameKit` Protocoles `GKGameCenterControllerDelegate`
6. Maintenant, nous voulons vérifier si l'utilisateur est connecté à 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. Maintenant, l'utilisateur utilise l'application et soudain, l'utilisateur a un nouveau score élevé, nous rapportons le score élevé en appelant la fonction ci-dessous.

La fonction ci-dessous affiche 2 paramètres.

`Identifier` qui est défini comme une chaîne et utilisé pour entrer votre classement qui a été créé dans iTunesConnect.

`score` qui est défini comme un Int qui sera le score des utilisateurs à soumettre à 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. Maintenant, si l'utilisateur veut voir les classements, appelez la fonction ci-dessous

```
//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()

    }
```

Lire Classements GameCenter en ligne: https://riptutorial.com/fr/ios/topic/6720/classements-gamecenter

Chapitre 32: Classes de taille et adaptivité

Remarques

Lorsque vous créez des applications adaptatives, gardez à l'esprit les limites des classes de taille: ce sont des *généralisations* , et non des guides spécifiques pour les tailles de pixels ou les périphériques exacts. Ne tentez jamais de déterminer sur quel périphérique votre application s'exécute ou si elle est en mode écran partagé, en fonction des classes de taille.

Au lieu de cela, prenez des décisions de mise en page de haut niveau sur la classe de taille et utilisez la disposition automatique pour modifier des cadres de vue précis. (Voir aussi la méthode `viewWillTransition(to:with:)` pour une notification plus précise de la taille de la vue d'un contrôleur après une transition.)

Examples

Collections de traits

Dans une application iOS, votre interface utilisateur peut prendre différentes formes et tailles. Celles-ci sont définies à l'aide de **classes de taille** disponibles dans une vue ou dans la **collection de traits** du contrôleur.

Apple définit deux classes de taille: **régulière** et **compacte** . Chacune de ces classes de taille est disponible sur les deux axes de l'appareil (**horizontal** et **vertical**). Votre application peut exister dans ces quatre états tout au long de sa vie. En résumé, les développeurs décrivent souvent une combinaison de classes de taille en disant ou en écrivant les deux classes de taille, l'axe horizontal en premier: "Compact / Regular" décrit une interface horizontalement compacte mais verticale.

Dans votre application, utilisez les méthodes du protocole UITraitEnvironment pour vérifier votre classe de taille actuelle et répondre aux modifications:

```
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.")
    }
}
```

UIView et UIViewController sont tous deux conformes à UITraitEnvironment. Vous pouvez donc examiner votre collection de traits en cours et gérer les modifications des sous-classes.

Mise à jour de la mise en page automatique avec les modifications apportées à la collection de caractéristiques

Faire en sorte que l'application soit **adaptative** - c'est-à-dire répondre aux changements de classe de taille en modifiant votre disposition - implique souvent une grande aide du système Auto Layout. L'une des principales façons dont les applications deviennent adaptatives consiste à mettre à jour les contraintes de mise en forme automatique actives lorsque la classe de taille d'une vue change.

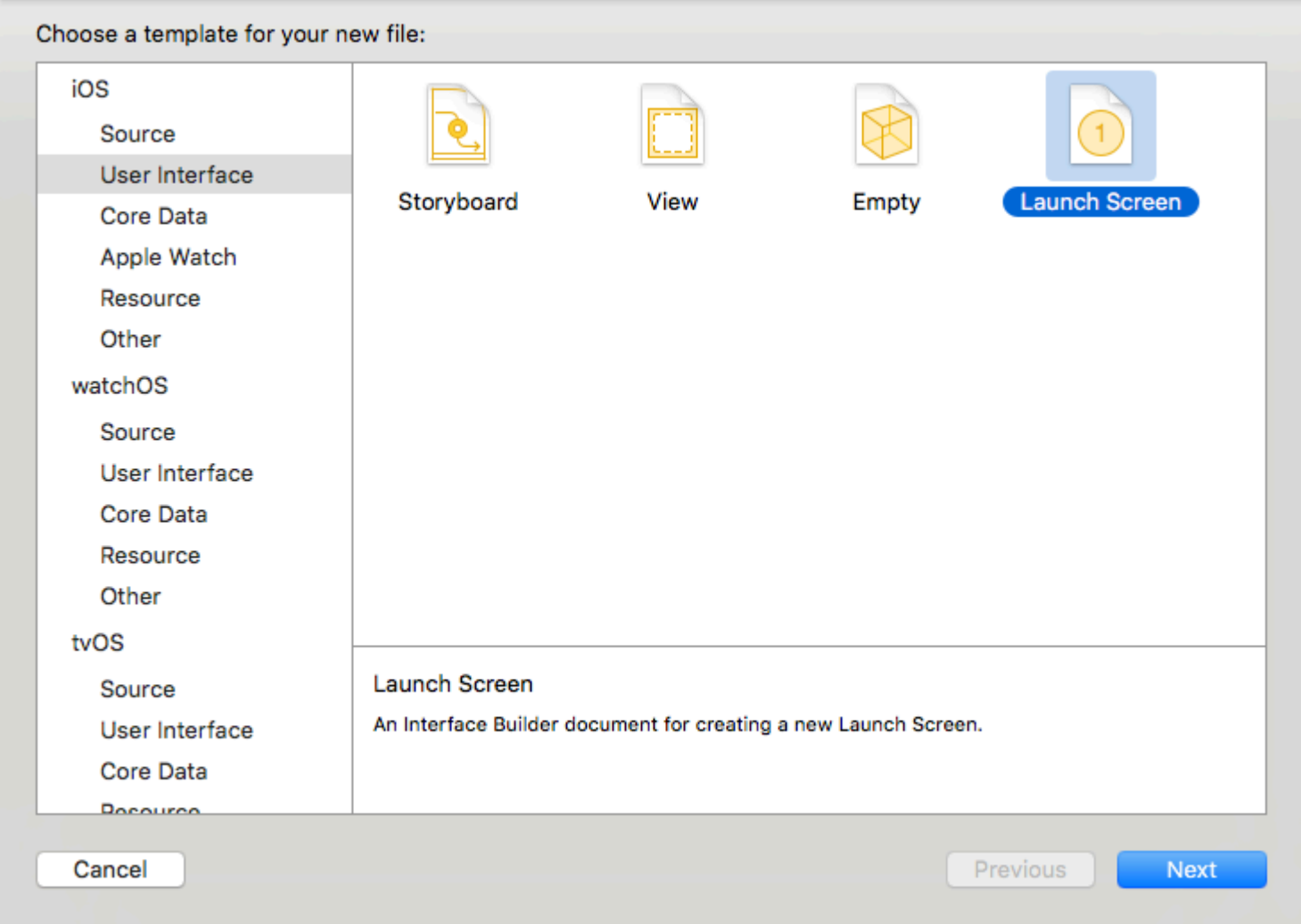
Par exemple, considérez une application qui utilise un UIStackView pour organiser deux UILabels. Nous pourrions souhaiter que ces étiquettes s'empilent les unes sur les autres dans des environnements horizontalement compacts, mais s'assoient côte à côte lorsque nous avons un peu plus de place dans des environnements horizontaux réguliers.

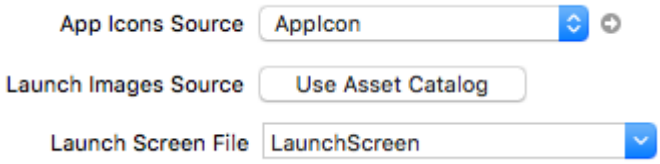
```
class ViewController: UIViewController {
    var stackView: UIStackView!

    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
        }
    }
}
```

Prise en charge du multitâche iOS sur iPad

Un élément clé de l'adaptabilité d'une application iOS moderne est le multitâche sur iPad. Par défaut, les applications créées dans Xcode 7 et versions ultérieures seront configurées pour prendre en charge le multitâche: elles contiendront un fichier LaunchScreen.storyboard utilisant la mise en page automatique.

La manière la plus simple pour les applications existantes d'opter pour le multitâche est de créer un tel storyboard, puis de le définir comme écran de lancement du projet:





Une fois que votre application prend en charge le multitâche iPad, auditez les vues existantes et affichez les contrôleurs pour vous assurer qu'ils utilisent la mise en page automatique et peuvent prendre en charge diverses combinaisons de classes de taille.

Lire Classes de taille et adaptivité en ligne: https://riptutorial.com/fr/ios/topic/4628/classes-de-taille-et-adaptivite

Chapitre 33: Classes de taille et adaptivité

Remarques

Pour plus de détails (classes de taille et adaptabilité via storyboard) de l'utilisation de la mise en page automatique pour l'adaptabilité dans iOS, nous pouvons suivre le lien vers le site développeur Apple .

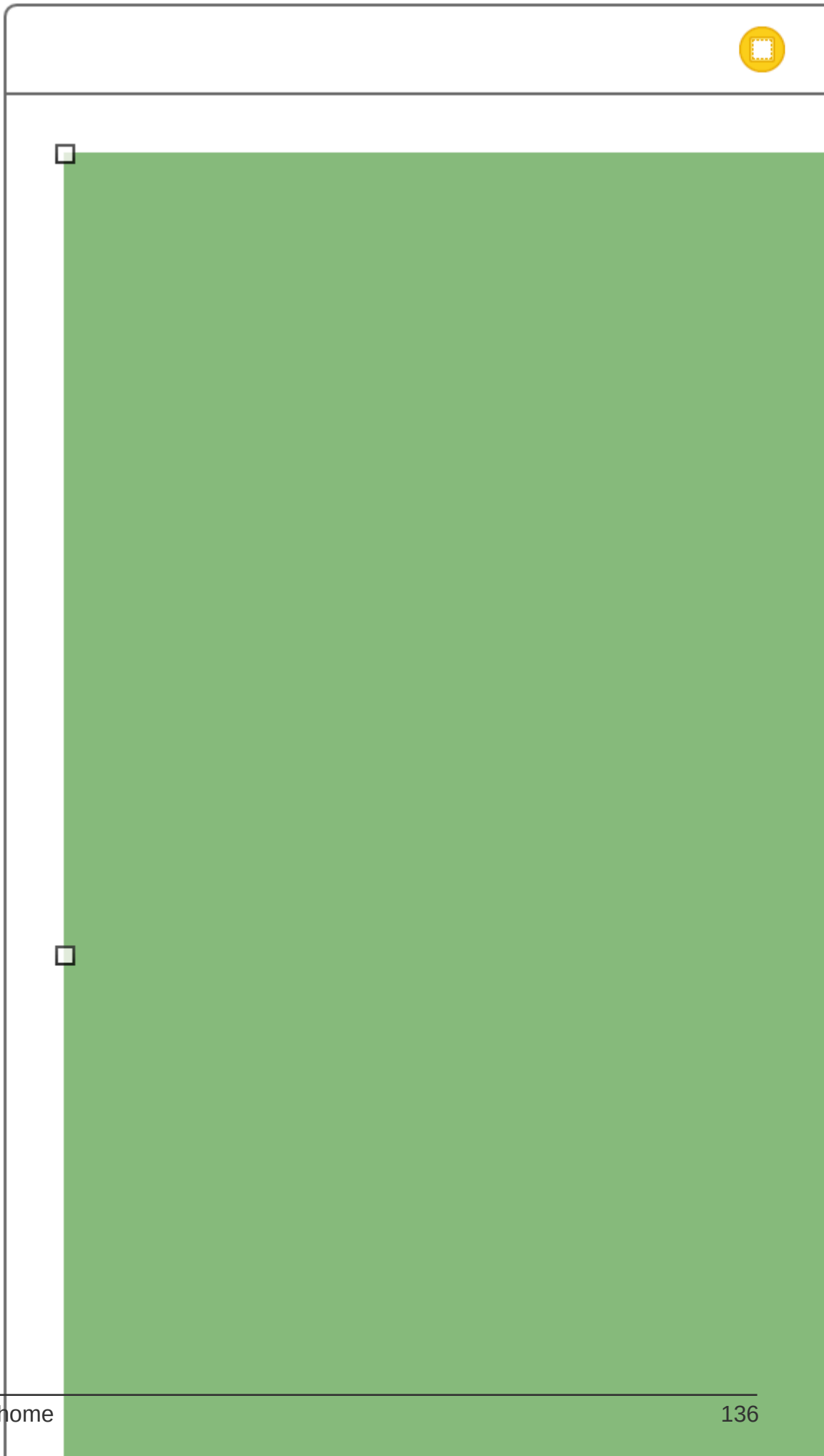
Nous pouvons également ajouter des contraintes à l' aide **Programatically Format** du **langage Visual** comme décrit ici sur le site des développeurs Apple .

Examples

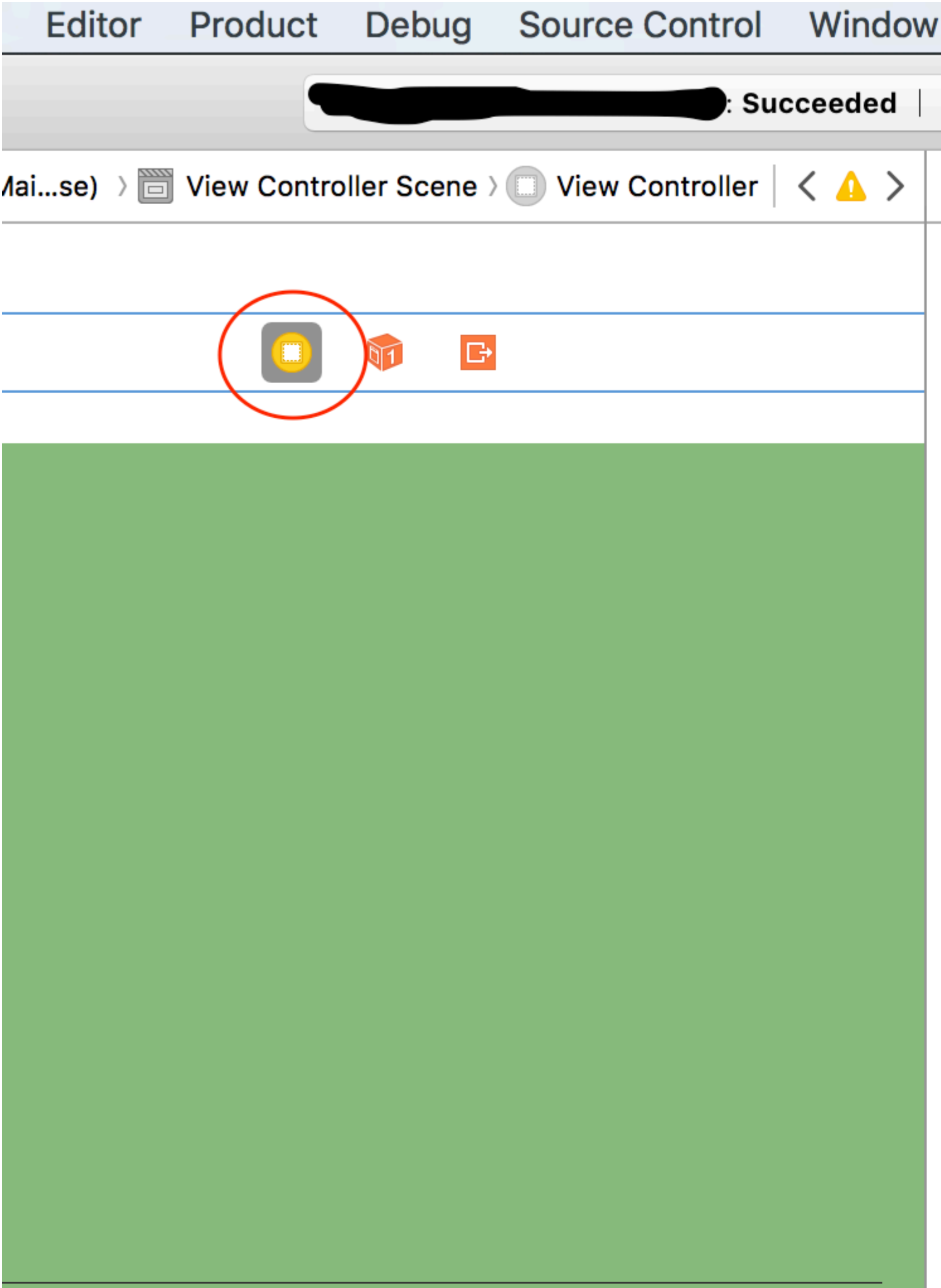
Classes de taille et adaptabilité via Storyboard

Nous pouvons ajouter une adaptabilité à toute sous-classe de `UIView` que nous ajoutons au contrôleur de vue dans le fichier nib.
Prenons un exemple d'ajout d'adaptivité en utilisant des classes de taille à une vue.

1. Ajouter une vue sur le contrôleur de vue en tant que:



Assistant Editor en tant que;


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

https://riptutorial.com/fr/ios/topic/6424/classes-de-taille-et-adaptivite

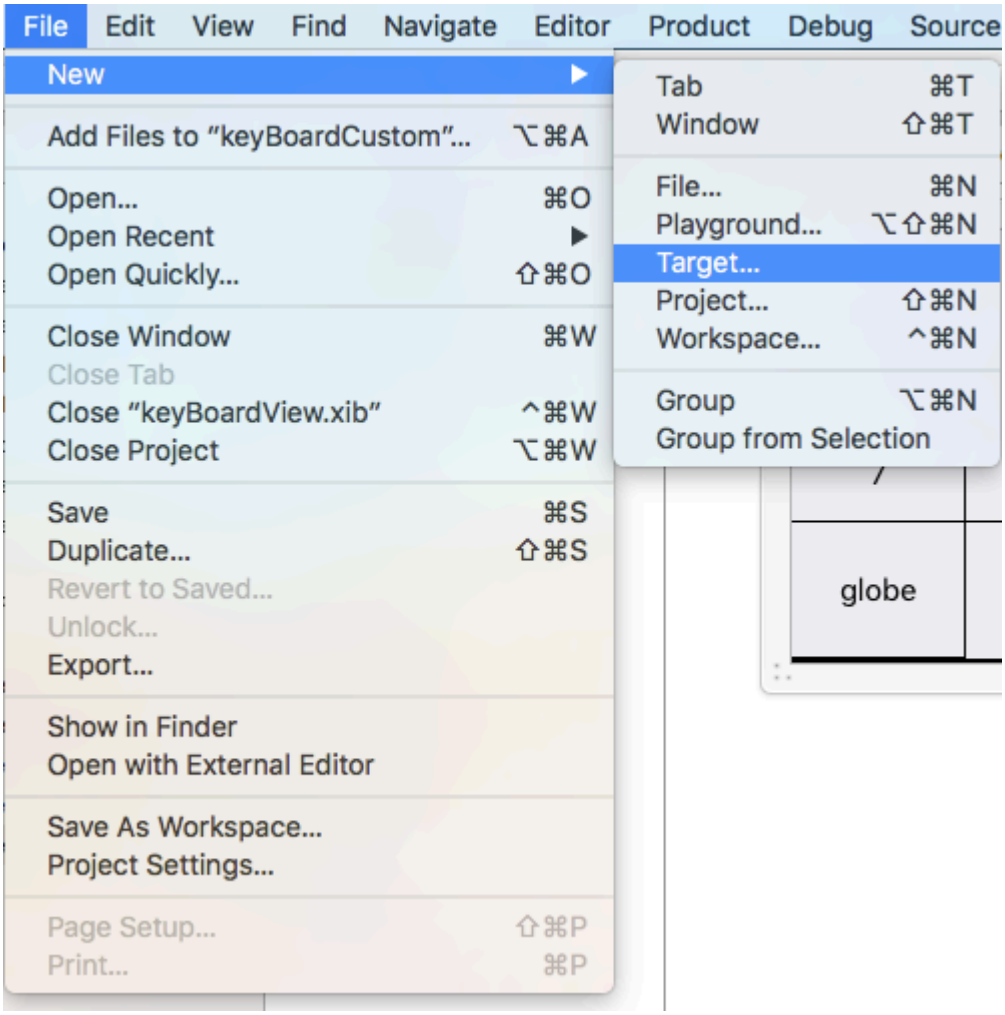
Chapitre 34: Clavier personnalisé

Examples

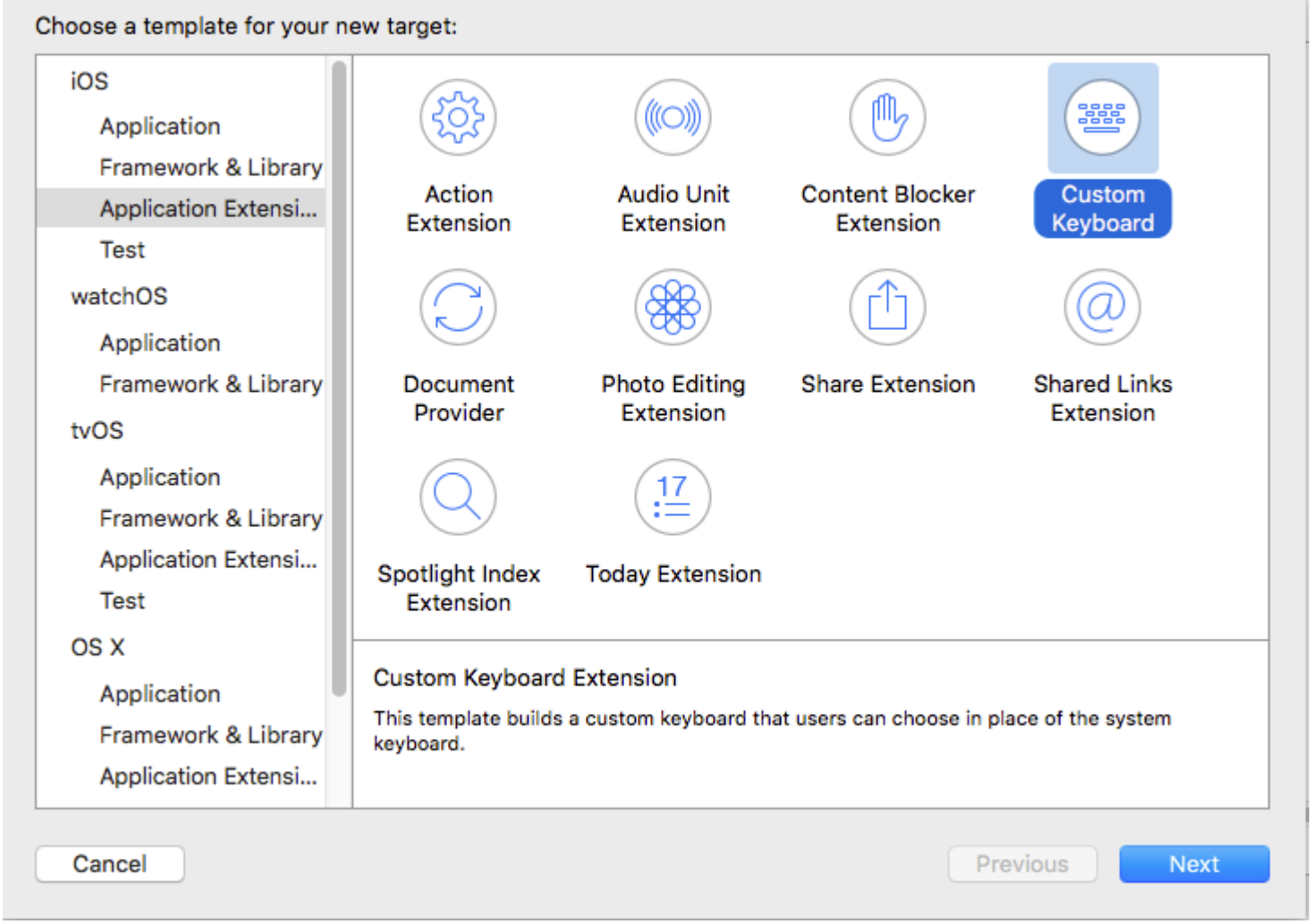
Exemple de clé personnalisée

Objective-C et Xib

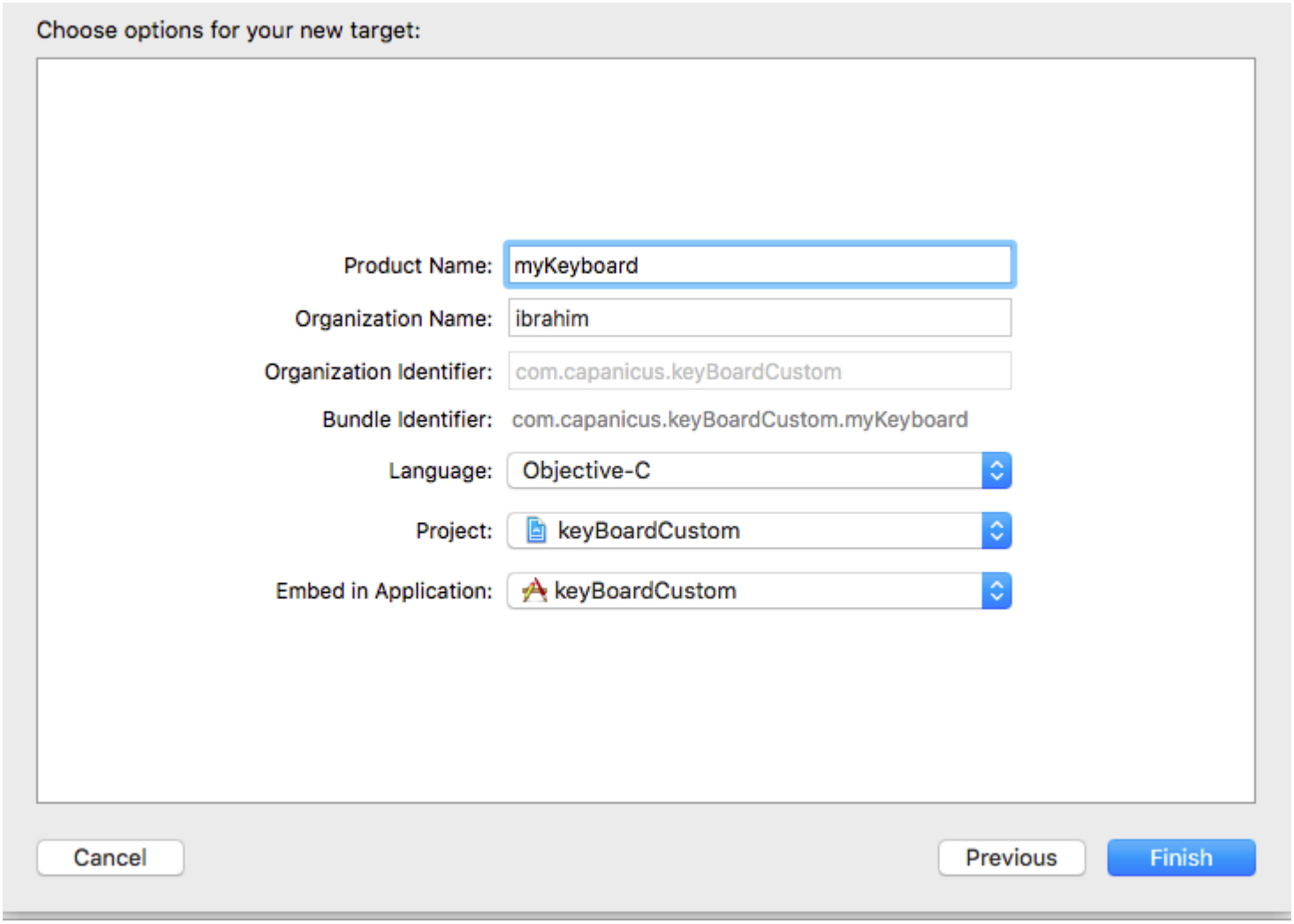
Ajouter une cible à un projet XCode existant



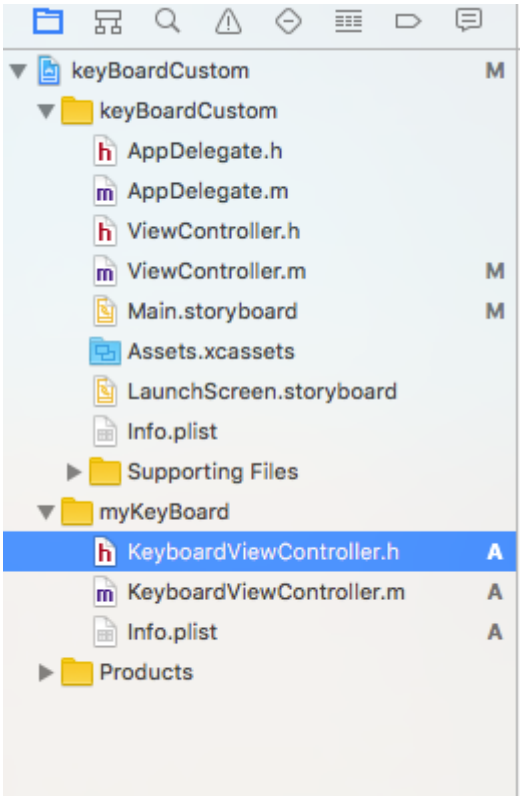
Dans la cible Ajouter, sélectionnez Custom KeyBoard



Ajoutez la cible comme ceci:

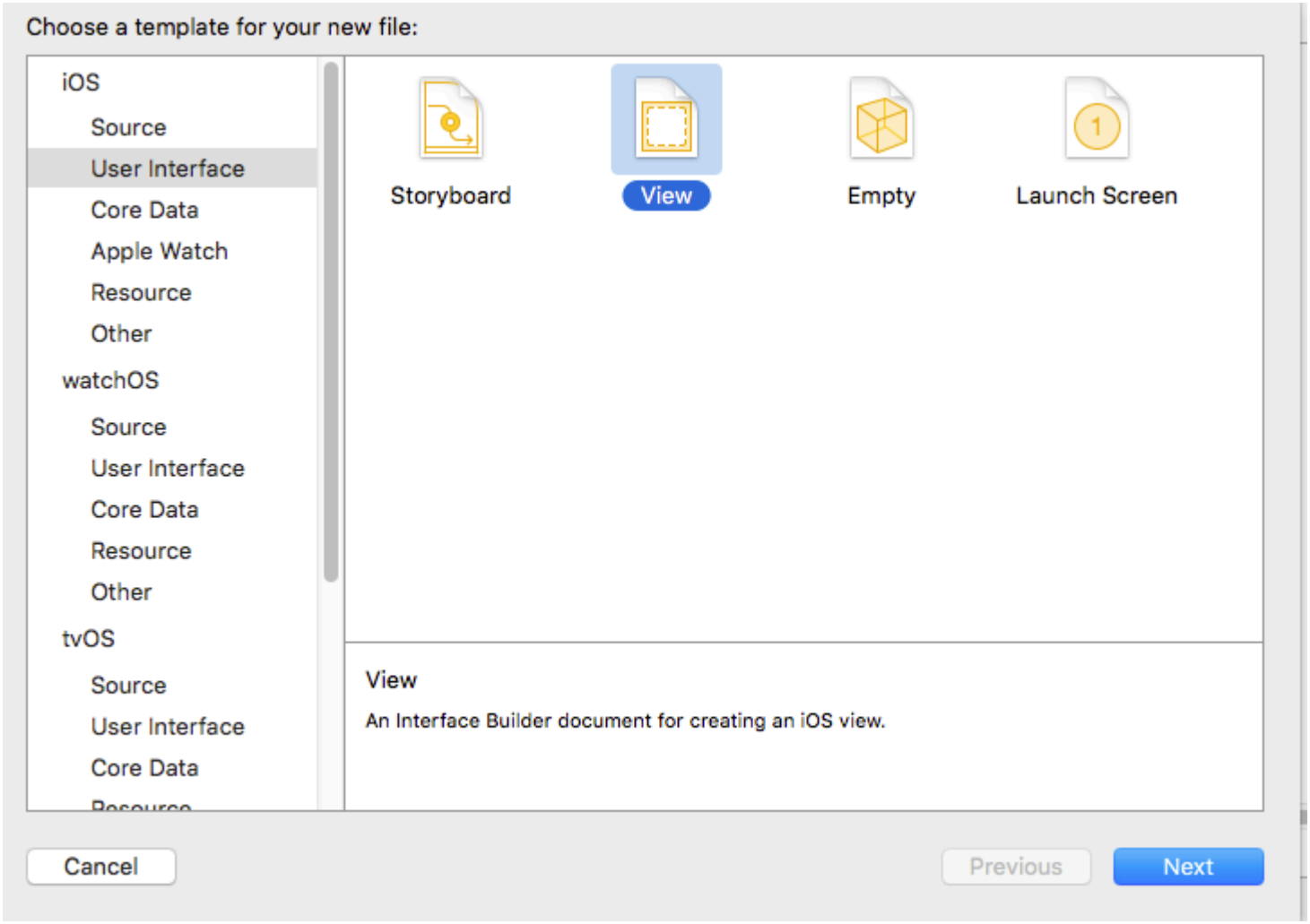


Votre répertoire de fichiers de projet devrait ressembler à ceci

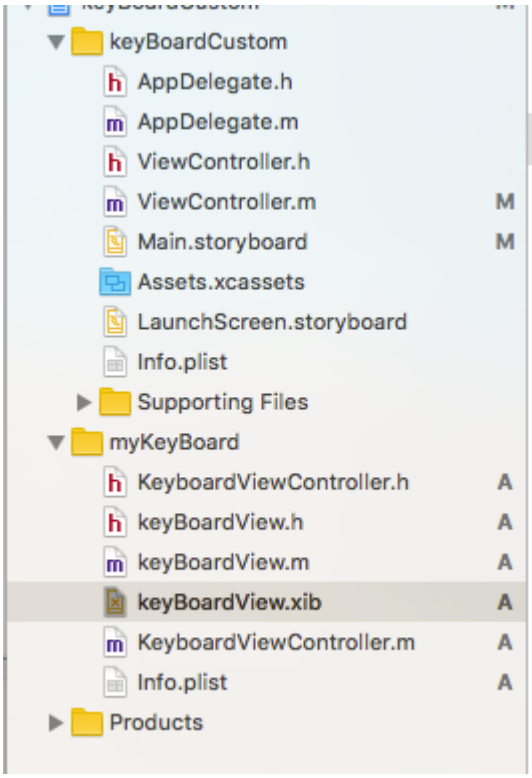


Ici, myKeyBoard est le nom de la cible ajoutée

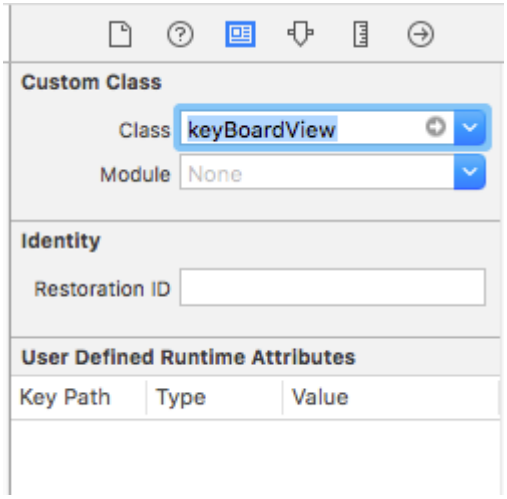
Ajouter un nouveau fichier Cocoatouch de type UIView et ajouter un fichier d'interface



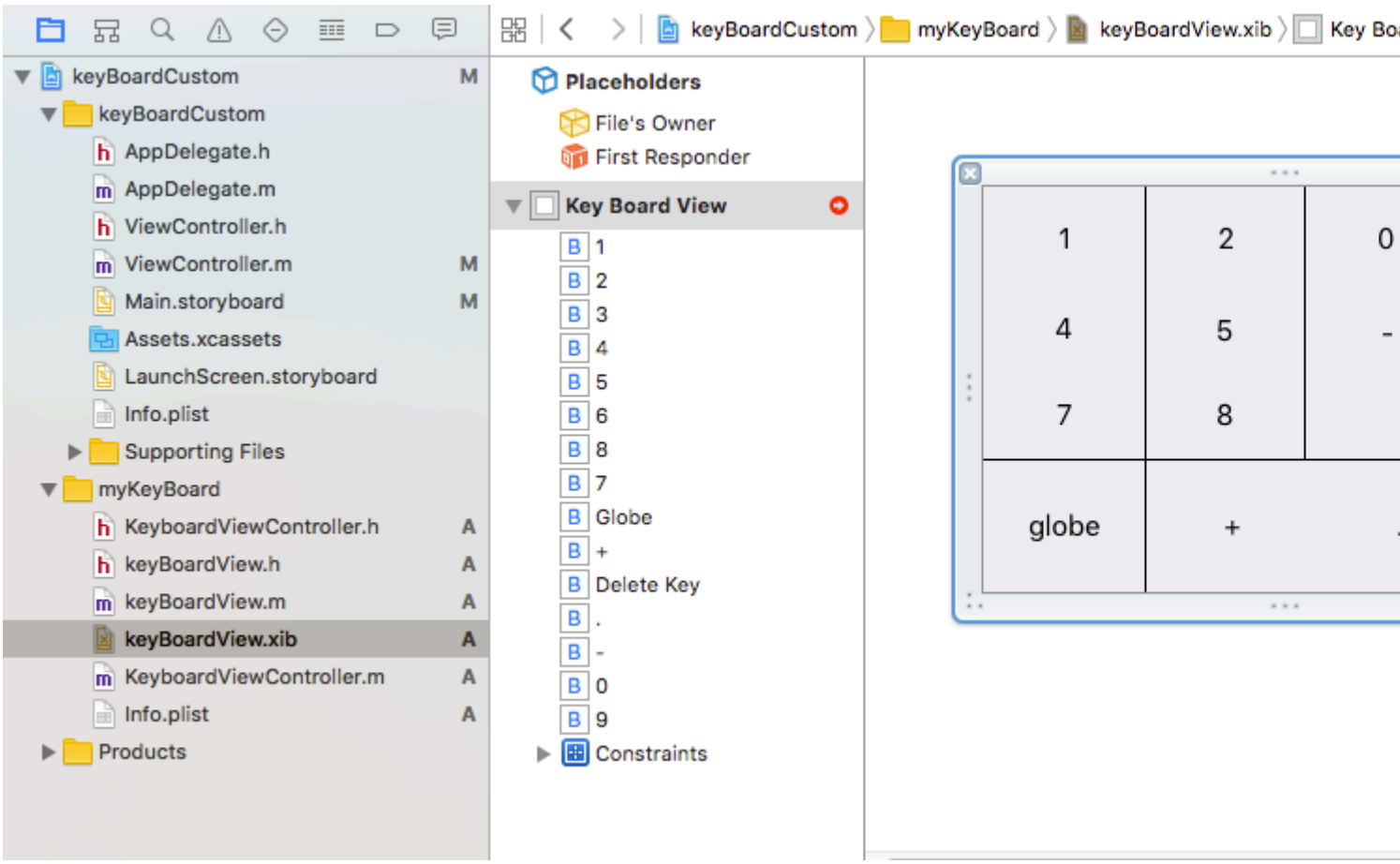
Enfin, votre répertoire de projet devrait ressembler à ceci



faire de `keyBoardView.xib` une sous-classe de `keyBoardView`



Faire une interface dans le fichier `keyBoardView.xib`



Établissez des connexions à partir du fichier `keyBoardView.xib` vers `keyBoardView.h`

`keyBoardView.h` devrait ressembler à

```
#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
```

Dans le fichier `keyBoardViewController.h` , importez `#import "keyBoardView.h"`

Déclarez une propriété pour le clavier `@property (strong, nonatomic)keyBoardView *keyboard;`

Commenter le

```
@property (nonatomic, strong) UIButton *nextKeyboardButton and all the code associated with it
```

La fonction viewDidLoad () du fichier KeyboardViewController.m devrait ressembler à ceci

```
- (void)viewDidLoad {
```

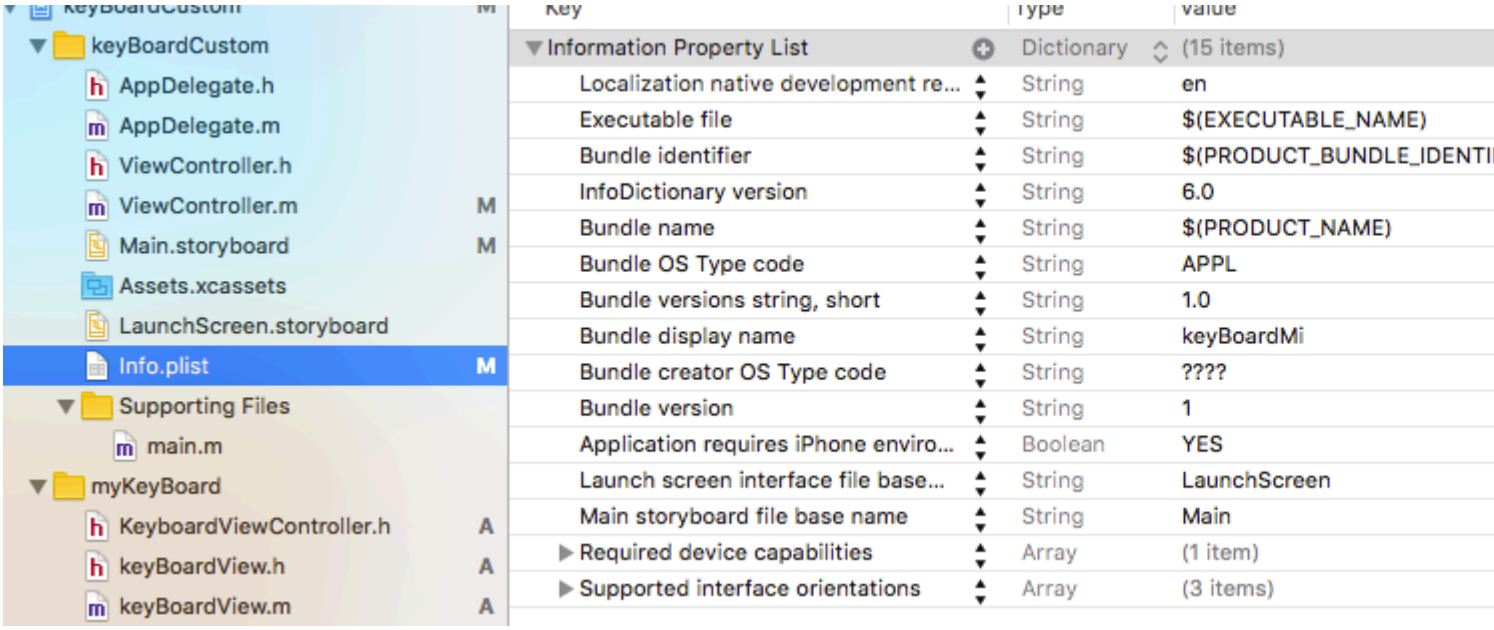
```
    [super viewDidLoad];
    self.keyboard=[[[NSBundle mainBundle]loadNibNamed:@"keyBoardView" owner:nil
options:nil]objectAtIndex:0];
      self.inputView=self.keyboard;
    [self addGestureToKeyboard];

    // 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;
}
```

Les fonctions `addGestureToKeyboard` , `pressDeleteKey` , `keyPressed` sont définies ci-dessous

```
-(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]];
}
```

Exécutez l'application principale et allez dans Paramètres-> Général-> Clavier-> Ajouter un nouveau clavier-> et ajoutez le clavier à partir de la section de clavier tierce (Le nom de clavier affiché serait keyBoardCustom)

Le nom du clavier peut être modifié en ajoutant une clé appelée `Bundle display name` et, dans la valeur de la chaîne de valeur, entrez le nom souhaité pour le clavier du projet principal.



Vous pouvez également regarder cette vidéo Youtube

Lire Clavier personnalisé en ligne: https://riptutorial.com/fr/ios/topic/7358/clavier-personnalise

Chapitre 35: CLLocation

Examples

Filtre de distance en utilisant

Exemple :

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

Par exemple, dans l'exemple de code ci-dessus, les changements d'emplacement inférieurs à 5 mètres ne seront pas envoyés au rappel, mais seront ignorés.

Obtenir l'emplacement de l'utilisateur à l'aide de CLLocationManager

1 - Inclure la structure CoreLocation.fr dans votre projet; Ceci est accompli en cliquant sur:

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

Cliquez sur le bouton (+), recherchez CoreLocation.framework et cliquez sur Ajouter.

2- Modifiez le fichier info.plist pour demander l'autorisation d'utiliser l'emplacement de l'utilisateur en l'ouvrant en tant que code source. Ajoutez l'une des clés suivantes: paire de valeurs sous la balise pour demander l'utilisation de l'emplacement de l'utilisateur lorsque l'application est utilisée:

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

3- importer CoreLocation dans le ViewController qui l'utilisera.

```
import CoreLocation
```

4- Assurez-vous que votre ViewController est conforme au protocole CLLocationManagerDelagate

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

Après ces étapes, nous pouvons créer un objet CLLocationManager en tant que variable d'instance et l'utiliser dans ViewController.

```
var manager:CLLocationManager!
```

Nous n'utilisons pas 'let' ici car nous allons modifier le gestionnaire pour spécifier son délégué, la distance minimale avant l'événement de mise à jour et sa précision

```
//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()

  }
```

Maintenant, pour avoir accès aux mises à jour de l'emplacement, nous pouvons implémenter la fonction ci-dessous qui s'appelle les heures supplémentaires auxquelles le distanceFilter est atteint.

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

Le paramètre locations est un tableau d'objets CLLocation qui représente l'emplacement réel du périphérique. À partir de ces objets, vous pouvez accéder aux attributs suivants: `coordinate,altitude, floor, horizontalAccuracy, verticalAccuracy, timestamp, description, course, speed` et une `distance(from:)` fonction `distance(from:)` qui mesure la distance entre deux emplacements.

Remarque: lors de la demande d'autorisation pour la localisation, il existe deux types d'autorisation différents.

> L'autorisation "En cours d'utilisation" n'autorise l'application qu'à recevoir votre position lorsque l'application est utilisée ou au premier plan.
>
> L'autorisation «Toujours» donne à l'application des autorisations en arrière-plan, ce qui peut entraîner une diminution de l'autonomie de la batterie au cas où votre application

serait fermée.

Le fichier Plist doit être ajusté si nécessaire.

Lire CLLocation en ligne: https://riptutorial.com/fr/ios/topic/2002/cllocation

Chapitre 36: CloudKit

Remarques

Types pris en charge

- NSData
- NSDate (Date)
- NSNumber (Int / Double)
- NSString (String)
- NSArray (Array)
- CLLocation
- CKReference
- CKAsset

Plus de détails

Tableau de bord CloudKit

Examples

Enregistrer l'application pour l'utiliser avec CloudKit

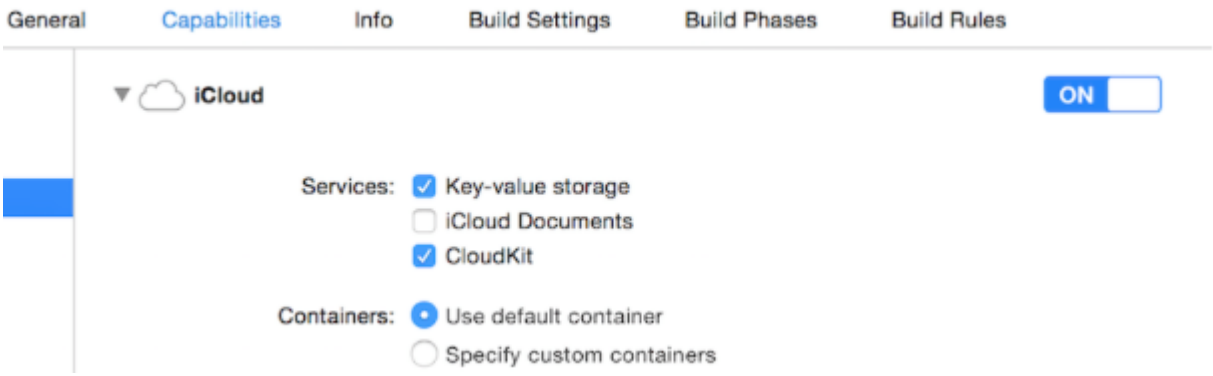
Ce dont vous avez besoin est d'obtenir un fichier de droits afin que l'application puisse accéder à votre iCloud et écrire des enregistrements à l'aide de CloudKit.

Suivez les étapes pour accorder l'accès à iCloud à partir de votre application:

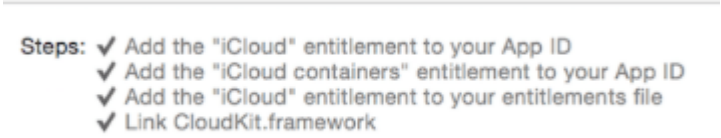
1- Sélectionnez le projet dans le navigateur de projet, puis ouvrez l'onglet Général.

2- Dans la section Identité, définissez l'identifiant Apple de votre développeur dans le menu déroulant Équipe. (S'il n'est pas disponible, ajoutez-le dans le menu Xcode -> Préférences -> Comptes.

3- Accédez à l'onglet Capabilities dans les propriétés du projet et activez iCloud. Ensuite, sélectionnez "Stockage Key-Value" et "CloudKit".



4- Assurez-vous que ces éléments sont cochés:



Si tous les éléments sont cochés, votre application est prête à utiliser CloudKit.

Utilisation du tableau de bord CloudKit

Tous les enregistrements créés à l'aide du code associé à CloudKit peuvent être prévisualisés, modifiés et même supprimés dans CloudKit Dashboard. Pour accéder à CloudKit Dashboard, cliquez ici .

Il y a plusieurs parties dans le tableau de bord:

- Types d'enregistrement (qui seront discutés plus tard)
- Rôles de sécurité (où vous pouvez définir des bases de données publiques ou privées)
- Types d'abonnement (que votre application peut enregistrer pour les notifications push Apple (APN)) afin de vous informer de la modification d'un enregistrement)

Types d'enregistrement

Ici, vous obtenez une liste de tous les types d'enregistrement existants dans l'application. Lorsque vous ouvrez pour la première fois CloudKit Dashboard pour une application, il existe un type d'enregistrement appelé Utilisateurs, que vous pouvez utiliser ou simplement supprimer et utiliser les vôtres.

Sur cette page, vous pouvez saisir manuellement vos données. Bien sûr, dans la plupart des cas, cela ne sert à rien, car iOS SDK peut le gérer mieux que le tableau de bord, mais les fonctionnalités sont également disponibles si vous préférez. La plus grande utilisation de cette page est la prévisualisation des types.

Enregistrement de données dans CloudKit

Pour enregistrer la date dans CloudKit, nous devons faire:

- Un `CKRecordID` (la clé de votre enregistrement unique)
- Un `CKRecord` (qui inclut des données)

Faire une clé d'enregistrement

Pour garantir que chaque nouvel identifiant d'enregistrement est unique, nous utilisons l'*horodatage* actuel, qui est unique. Nous obtenons l'horodatage en utilisant `NSDate` méthode de `timeIntervalSinceReferenceDate()` . Il est sous la forme de ###. ### (# sont des nombres), nous utiliserons la partie entière. Pour ce faire, nous divisons la chaîne:

Rapide

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

Faire le record

Pour faire l'enregistrement, nous devrions spécifier le type d'enregistrement (expliqué dans Utilisation de CloudKit Dashboard) en tant qu'Utilisateurs, l'ID que nous avons créé tout à l'heure et les données. Ici, nous allons ajouter un exemple de texte, une image et la date du jour à l'enregistrement:

Rapide

```
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")
```

Objectif 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"];
```

Remarque

Ici, nous n'avons pas ajouté l' `UIImage` directement à l'enregistrement, car comme mentionné dans Remarques, le format d'image n'est pas directement pris en charge dans CloudKit, nous avons donc converti `UIImage` en `CKAsset` .

Accéder au conteneur

Rapide

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

Enregistrement des enregistrements dans la base de données CloudKit

Rapide

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

Lire CloudKit en ligne: https://riptutorial.com/fr/ios/topic/4946/cloudkit

Chapitre 37: Codable

Introduction

Codable est ajouté avec Xcode 9, iOS 11 et Swift 4. Codable est utilisé pour rendre vos types de données encodables et décodables pour les rendre compatibles avec des représentations externes telles que JSON.

Utilisation codable pour prendre en charge à la fois l'encodage et le décodage, déclarer la conformité à Codable, qui combine les protocoles codables et décodables. Ce processus est appelé rendre vos types codables.

Examples

Utilisation de codable avec JSONEncoder et JSONDecoder dans Swift 4

Prenons un exemple avec Structure of Movie, ici nous avons défini la structure comme Codable. Donc, nous pouvons l'encoder et le décoder facilement.

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

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

Nous pouvons créer un objet à partir d'un film comme:

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

Le upMovie contient le nom «Up» et son film Genere est la comédie, l'aventure et l'animation qui contient 4 sur 5.

Encoder

JSONEncoder est un objet qui code les instances d'un type de données en tant qu'objets JSON. JSONEncoder prend en charge l'objet Codable.

```
// Encode data
let jsonEncoder = JSONEncoder()
do {
    let jsonData = try jsonEncoder.encode(upMovie)
    let jsonString = String(data: jsonData, encoding: .utf8)
    print("JSON String : " + jsonString!)
}
```

```
catch {
}
```

JSONEncoder nous donnera les données JSON utilisées pour récupérer la chaîne JSON.

La chaîne de sortie sera comme:

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

Décoder

JSONDecoder est un objet qui décode des instances d'un type de données à partir d'objets JSON. Nous pouvons récupérer l'objet à partir de la chaîne 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 {
}
```

En décodant la JSONDATA, nous recevrons l'objet Movie. Nous pouvons donc obtenir toutes les valeurs enregistrées dans cet objet.

Le résultat sera comme:

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

Lire Codable en ligne: https://riptutorial.com/fr/ios/topic/10639/codable

Chapitre 38: Concurrence

Introduction

Rubrique connexe: Dispatch Grand Central

Syntaxe

- dispatch_async - Exécute un bloc de code dans une file d'attente séparée et n'arrête pas la file d'attente en cours. Si la file d'attente se trouve sur un thread différent de celui sur lequel on a appelé dispatch_async, le code du bloc s'exécutera alors que le code est exécuté après l'exécution de dispatch_async
- dispatch_sync - exécute un bloc de code dans une file d' attente séparée, et *ne* se limite la file d' attente en cours. Si la file d'attente se trouve sur un thread différent de celui sur lequel on a appelé dispatch_async, le code du bloc s'exécutera et son exécution sur le thread où la méthode a été appelée ne reprendra qu'après la fin

Paramètres

queue	**La file d'attente dans laquelle le code dans le bloc de distribution sera exécuté. Une *file d'attente* est comme (mais pas exactement comme) un thread; le code dans différentes files d'attente peut s'exécuter en parallèle. Utilisez `dispatch_get_main_queue` pour obtenir la file d'attente du thread principal Pour créer une nouvelle file d'attente, qui à son tour crée un nouveau thread, utilisez `dispatch_queue_create("QUEUE_NAME", DISPATCH_QUEUE_CONCURRENT)` . Le premier paramètre est le nom de la file d'attente, qui est affiché dans le débogueur si vous faites une pause pendant que le bloc est toujours en cours d'exécution. Le second paramètre n'a pas d'importance à moins que vous ne souhaitiez utiliser la même file d'attente pour plusieurs appels à `dispatch_async` ou `dispatch_sync` . Il décrit ce qui se passe lorsqu'un autre bloc est placé dans la même file d'attente. `DISPATCH_QUEUE_CONCURRENT` provoquera l'exécution des deux blocs en même temps, tandis que `DISPATCH_QUEUE_SERIAL` fera attendre le second bloc pour terminer le deuxième bloc**
bloc	Le code dans ce bloc s'exécutera dans la file d' `queue` la file d' `queue` ; mettre le code que vous souhaitez exécuter dans la file d'attente séparée ici. Un conseil utile: si vous écrivez ceci dans Xcode et que l'argument de bloc a le contour bleu, double-cliquez sur l'argument et Xcode créera automatiquement un bloc vide (cela s'applique à tous les arguments de bloc de n'importe quelle fonction ou méthode)

Remarques

Chaque fois que vous faites quelque chose sur un thread séparé, ce qui se produit lors de

l'utilisation de files d'attente, il est important de maintenir la sécurité des threads. Certaines méthodes, en particulier celles de `UIView` s, peuvent ne pas fonctionner et / ou ne peuvent pas tomber en panne sur des threads autres que le thread principal. Veillez également à ne rien modifier (variables, propriétés, etc.) qui est également utilisé sur le thread principal, à moins que vous ne teniez compte de cette modification.

Examples

Exécution simultanée du code - Exécution du code lors de l'exécution d'un autre code

Supposons que vous souhaitiez effectuer une action (dans ce cas, vous connectez "Foo"), tout en faisant autre chose (en enregistrant "Bar"). Normalement, si vous n'utilisez pas la simultanéité, l'une de ces actions va être entièrement exécutée et l'autre exécution ne sera exécutée qu'après avoir été complètement terminée. Mais avec la concurrence, vous pouvez exécuter les deux actions en même temps:

```
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);
}
```

Cela va enregistrer "Foo" 100 fois, en faisant une pause de 100 ms à chaque fois qu'il se connecte, mais il fera tout cela sur un thread séparé. Pendant que `Foo` est en cours de connexion, "Bar" sera également enregistré dans des intervalles de 50 ms, en même temps. Vous devriez idéalement voir une sortie avec "Foo" et "Bars" mélangés ensemble

Exécution sur le thread principal

Lors de l'exécution asynchrone de tâches, il devient généralement nécessaire de s'assurer qu'un morceau de code est exécuté sur le thread principal. Par exemple, vous voudrez peut-être frapper une API REST de manière asynchrone, mais placez le résultat dans un UILabel à l'écran. Avant de mettre à jour le UILabel, vous devez vous assurer que votre code est exécuté sur le thread principal:

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

```
}
```

Chaque fois que vous mettez à jour des vues à l'écran, assurez-vous toujours de le faire sur le thread principal, sinon un comportement indéfini pourrait se produire.

Dispatch group - en attente d'autres threads terminés.

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

Mise à jour 1. Version 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")
}
```

Lire Concurrence en ligne: https://riptutorial.com/fr/ios/topic/1090/concurrence

Chapitre 39: Configuration iOS de Carthage

Examples

Installation Carthage Mac

Mise en Carthage

Téléchargez la dernière version de Carthage à partir du lien donné Lien de téléchargement

Dans la section Téléchargement, téléchargez le fichier **Carthage.pkg** .

Une fois le téléchargement terminé, installez-le en double cliquant sur le fichier pkg de téléchargement.

Pour vérifier le succès du téléchargement, exécutez la commande suivante dans votre version de Terminal carthage Cela devrait donner la version installée comme `0.18-19-g743fa0f`

Lire Configuration iOS de Carthage en ligne: https://riptutorial.com/fr/ios/topic/7404/configuration-ios-de-carthage

Chapitre 40: Configurer les balises avec CoreBluetooth

Introduction

Chaud pour lire et écrire des données sur un appareil Bluetooth basse consommation.

Remarques

Quelques points importants

- Aucune capacité n'est requise.
- iPhone stockez les octets au format Little Endian, vérifiez donc si les accessoires Bluetooth utilisent également Little Endian. Exemple:
  - Intel CPU utilise généralement peu d'endian.
  - L'architecture ARM était un peu endian avant la version 3 lorsqu'elle est devenue big-endian.
- Après une opération unique ou par lots, la connexion sera perdue, vous devez donc vous reconnecter avant de continuer.

Scan for SERVICE UUID

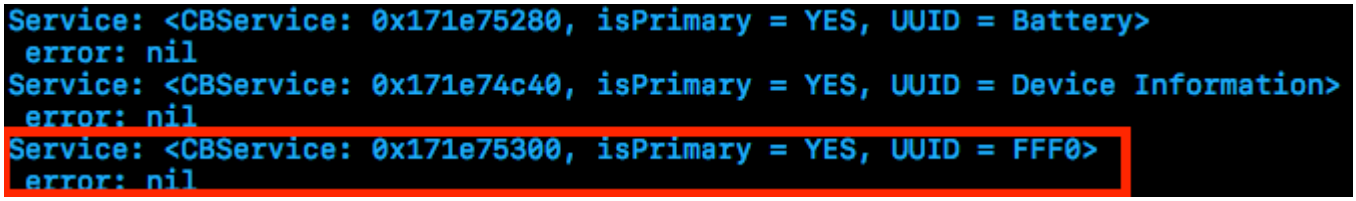
```
func SearchBLE(){
    cb_manager.scanForPeripherals(withServices:[service_uuid], options: nil)
    StopSearchBLE()
}
```

Comment découvrir UUID SERVICE sans documentation

```
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 signifie que tous les services seront retournés, ce qui n'est pas une bonne option. (LIRE les remarques 3)

- Si vous n'avez pas trouvé l'UUID de SERVICE, exécutez votre code et cherchez dans la console



- J'ai trouvé 3 services: Batterie, Informations sur l'appareil (Firmware) et FFF0
- Ce service uuid n'est pas un service standard, une liste de normes peut être trouvée ici
- FFF0 est l'UUID de SERVICE dans ce cas

Convertir des données en UInt16 et contraire

Ajoutez cette extension à votre 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

Affichage des noms de tous les Bluetooth Low Energy (BLE)

- Pour cet exemple, j'ai une salle contrôlée avec un seul périphérique BLE.
- Votre classe devrait étendre CBCentralManagerDelegate.
- Implémentez la méthode: centralManagerDidUpdateState (_ central: CBCentralManager).

- Utilisez la file d'attente globale pour ne pas geler l'écran lors de la recherche d'un périphérique.
- Instanciez CBCentralManager et attendez la réponse de 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)")
    }
}
```

Le rappel de centralManagerDidUpdateState indique que CoreBluetooth est prêt, vous pouvez donc rechercher BLE maintenant. Mettez à jour le code centralManagerDidUpdateState pour rechercher tout périphérique BLE lorsqu'il est prêt.

```
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 () recherche les périphériques BLE et arrête la recherche après 5 secondes
- cb_manager.scanForPeripherals (withServices: nil, options: nil) recherche chaque BLE à portée de main.
- StopSearchBLE () arrête la recherche après 5 secondes.
- Chaque BLE trouvée rappellera func centralManager (_ central: CBCentralManager, périphérique didDiscover: 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)
}
```

Connecter et lire la valeur majeure

- Je suis dans une pièce contrôlée avec une seule balise qui utilise le protocole IBEACON.
- BLEController doit étendre CBPeripheralDelegate
- Je vais utiliser le premier BLE pour se connecter après l'arrêt de la recherche.
- Modifier la méthode 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)
        }
    }
/...
}
```

- Dans la documentation de votre appareil BLE, vous devez rechercher la CARACTÉRISTIQUE SERVICE UUID et MAJOR UUID

```
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])!)
}
```

- Créez une variable 'service_uuid' et 'major_uuid' comme le code ci-dessus. '-0000-1000-8000-00805f9b34fb' fait partie de la norme. 'fff0' est mon UUID DE SERVICE, 'fff2' est ma caractéristique MAJEURE UUID et '0000' est nécessaire pour remplir le bloc uuid 1º de 4 octets.
- discoverCharacteristics ([major_uuid], pour: (périphérique.services?[0])!) aura une caractéristique majeure de mon serveur gatt de périphérique et aura pour valeur NIL pour le moment.
- (périphérique.services?[0])! - 0 beacuse renverra une seule valeur une fois que j'ai fait peripher.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)")
}
```

- La valeur de la caractéristique ne sera lisible qu'après un appel périphérique.readValue (pour: caractéristique)
- readValue se traduira par un périphérique func (périphérique: CBPeripheral, didUpdateValueFor caractéristique: caractéristique caractéristique, erreur: erreur?) avec une valeur dans le type de données.

Ecrire une valeur majeure

- Vous devez découvrir les services et les caractéristiques
- Vous n'avez pas besoin de lire la valeur de la caractéristique avant de l'écrire.
- continuera pour, pour cet exemple, après la valeur lue. Modifier le périphérique func (périphérique: CBPeripheral, didUpdateValueFor caractéristique: CBCcaracteristic, erreur: erreur?)
- Ajouter une variable new_major et 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 par deafult enverra et recevra des octets au format Little Endian, mais mon appareil MINEW avec son jeu de puces NRF51822 a une architecture ARM et a besoin d’octets au

format Big Endian, je dois donc l'échanger.
- La documentation de BLE Device indiquera quel type d'entrée et de sortie chaque caractéristique aura et si vous pouvez le lire comme ci-dessus (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)
    }
}
```

- Pour mettre à jour les informations d'un serveur gatt, vous devez les redémarrer par programme ou enregistrer des données, puis les désactiver et les activer manuellement.
- FFFF est caractéristique qui le fait dans cet appareil.
- 'minew123' est le mot de passe par défaut pour le redémarrage o enregistrer les informations dans ce cas.
- exécutez votre application et regardez-vous la console pour toute erreur, je l'espère pas, mais vous ne verrez pas encore la nouvelle valeur.

```
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)
```

}

- La dernière étape consiste à commenter la dernière ligne de la méthode didUpdateValueFor et à réexécuter l'application, maintenant vous aurez la nouvelle valeur.

Lire Configurer les balises avec CoreBluetooth en ligne:
https://riptutorial.com/fr/ios/topic/9488/configurer-les-balises-avec-corebluetooth

Chapitre 41: Content Hugging / Compression de contenu dans Autolayout

Remarques

Priorité de résistance à la compression de contenu

> Cette valeur détermine à quel point une vue doit être compressée ou réduite. Une valeur plus élevée signifie que la vue sera moins susceptible d'être compressée et plus susceptible de rester la même.

Priorité au contenu

> Cette valeur détermine à quel point une vue doit être étendue. Vous pouvez imaginer «étreindre» ici pour signifier «taille à adapter» - les limites de la vue seront «étreintes» ou seront proches de la taille du contenu intrinsèque. Une valeur plus élevée signifie que la vue risque moins de se développer et de rester la même.

Examples

Définition: taille du contenu intrinsèque

Avant la mise en page automatique, vous deviez toujours indiquer aux boutons et aux autres contrôles leur taille, en définissant leurs propriétés de cadre ou de limite ou en les redimensionnant dans Interface Builder. Mais il s'avère que la plupart des contrôles sont parfaitement capables de déterminer la quantité d'espace dont ils ont besoin, en fonction de leur contenu.

Une **étiquette** sait quelle est la largeur et la hauteur car elle connaît la longueur du texte qui a été défini, ainsi que la taille de la police pour ce texte. De même pour un **bouton** , qui peut combiner le texte avec une image d'arrière-plan et un peu de remplissage.

Il en va de même pour les contrôles segmentés, les barres de progression et la plupart des autres contrôles, même si certains n'ont qu'une hauteur prédéterminée mais une largeur inconnue.

C'est ce que l'on appelle la taille du contenu intrinsèque, et c'est un concept important dans la mise en page automatique. Auto Layout demande à vos contrôles quelle taille ils doivent avoir et affiche l'écran en fonction de ces informations.

Habituellement, vous voulez utiliser la `intrinsic content size` , mais dans certains cas, vous pouvez ne pas vouloir le faire. Vous pouvez empêcher cela en définissant une contrainte de largeur ou de hauteur explicite sur un contrôle.

Imaginez ce qui se passe lorsque vous définissez une image sur un UIImageView si cette image est beaucoup plus grande que l'écran. Vous souhaitez généralement donner aux images une

largeur et une hauteur fixes et mettre le contenu à l'échelle, à moins que vous ne souhaitiez redimensionner la vue aux dimensions de l'image.

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

Lire Content Hugging / Compression de contenu dans Autolayout en ligne: https://riptutorial.com/fr/ios/topic/6899/content-hugging---compression-de-contenu-dans-autolayout

Chapitre 42: Convertir le HTML en chaîne NSAttributed et vice versa

Examples

Objectif code C pour convertir une chaîne HTML en NSAttributedString et Vice Versa

Code de conversion HTML vers NSAttributedString: -

```
 //HTML String
 NSString *htmlString=[[NSString alloc]initWithFormat:@"<!DOCTYPE html><html><body><h1>My
First Heading</h1><p>My first paragraph.</p></body></html>"];
//Converting HTML string with UTF-8 encoding to NSAttributedString
 NSAttributedString *attributedString = [[NSAttributedString alloc]
                                         initWithData: [htmlString
dataUsingEncoding:NSUnicodeStringEncoding]
                                         options: @{ NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType }
                                         documentAttributes: nil
                                         error: nil ];
```

NSAttributedString to HTML Conversion: -

```
 //Dictionary to hold all the attributes of NSAttributed String
 NSDictionary *documentAttributes = @{NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType};
 //Saving the NSAttributedString with all its attributes as a NSData Entity
 NSData *htmlData = [attributedString dataFromRange:NSMakeRange(0, attributedString.length)
documentAttributes:documentAttributes error:NULL];
 //Convert the NSData into HTML String with UTF-8 Encoding
 NSString *htmlString = [[NSString alloc] initWithData:htmlData
encoding:NSUTF8StringEncoding];
```

Lire Convertir le HTML en chaîne NSAttributed et vice versa en ligne:
https://riptutorial.com/fr/ios/topic/7225/convertir-le-html-en-chaine-nsattributed-et-vice-versa

Chapitre 43: Convertir NSAttributedString en UIImage

Examples

NSAttributedString à la conversion UIImage

Objectif 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 fonction `imageFromAttributedString` est définie ci-dessous:

```
- (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;
}
```

Lire Convertir NSAttributedString en UIImage en ligne:
https://riptutorial.com/fr/ios/topic/7242/convertir-nsattributedstring-en-uiimage

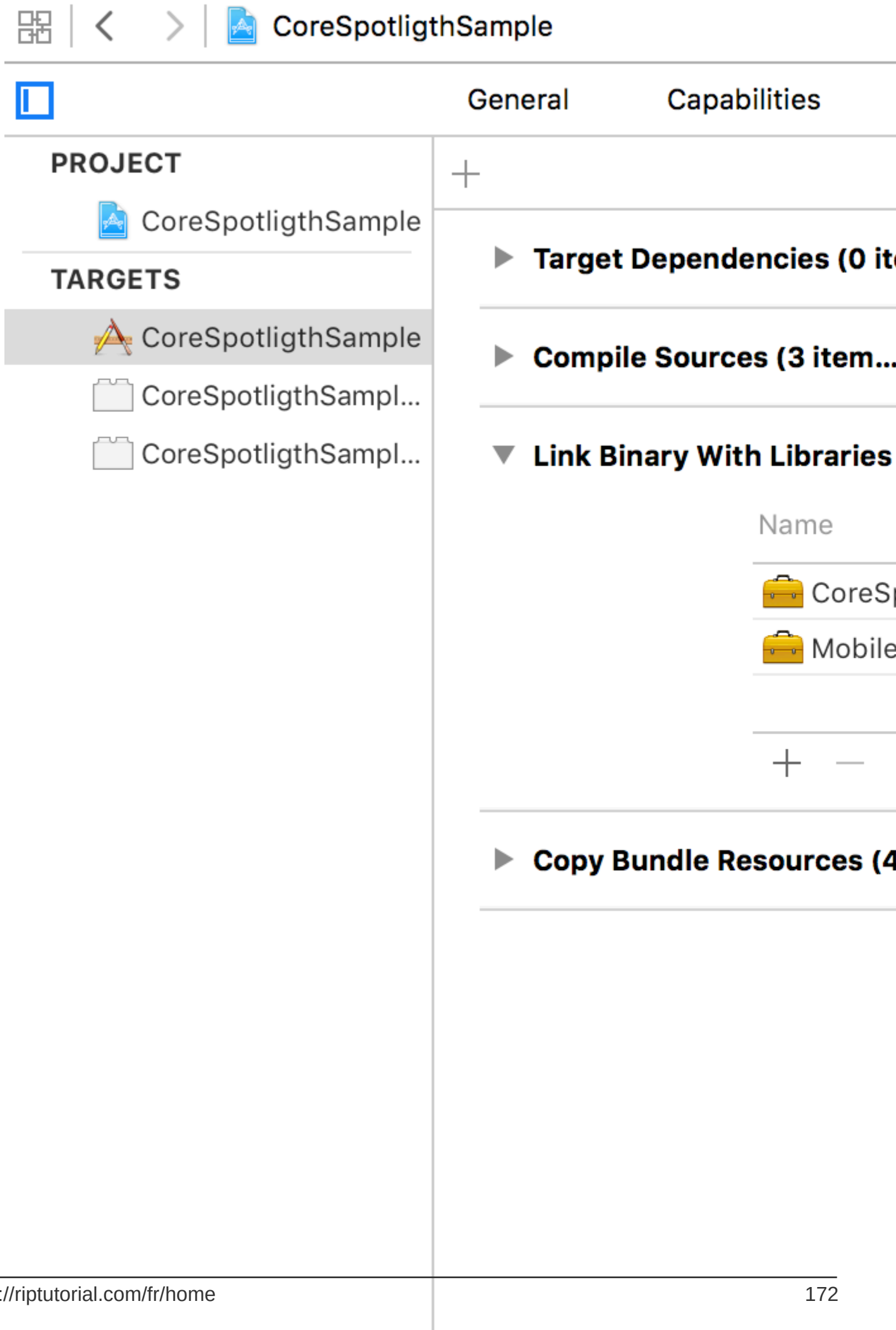
Chapitre 44: Core SpotLight dans iOS

Examples

Core-Spotlight

Objectif c

1. Créez un nouveau projet iOS et ajoutez le *framework CoreSpotlight* et *MobileCoreServices* à votre projet.


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. Créez le CSSearchableItem réel et associez l'uniqueIdentifier, le domainIdentifier et l'attributSet. Enfin, indexez le CSSearchableItem en utilisant [[CSSearchableIndex defaultSearchableIndex] ...] comme indiqué ci-dessous.

```
//
//  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! Testez l'index!

https://riptutorial.com/fr/ios/topic/7416/core-spotlight-dans-ios

Chapitre 45: Couper un UIImage en cercle

Examples

Couper une image en cercle - Objectif C

import `#include <math.h>`

Le code dans viewDidLoad ou loadView devrait ressembler à quelque chose comme ceci

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

Enfin, la fonction qui fait la lourde charge `circularScaleAndCropImage` est telle que définie ci-dessous

```
- (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;
}
```

Exemple SWIFT 3

```
 override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        let imageView = UIImageView(frame: CGRect(x: CGFloat(0), y: CGFloat(50), width:
CGFloat(320), height: CGFloat(320)))
        view.addSubview(imageView)
        let image = UIImage(named: "Dubai-Photos-Images-Travel-Tourist-Images-Pictures-
800x600.jpg")
        imageView.image = circularScaleAndCropImage(UIImage(named: "Dubai-Photos-Images-
Travel-Tourist-Images-Pictures-800x600.jpg")!, frame: CGRect(x: CGFloat(0), y: CGFloat(0),
width: CGFloat(100), height: CGFloat(100)))
    }
```

Enfin, la fonction qui fait la lourde charge circularScaleAndCropImage est telle que définie ci-dessous

```
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!
    }
```

Lire Couper un UIImage en cercle en ligne: https://riptutorial.com/fr/ios/topic/7222/couper-un-uiimage-en-cercle

Chapitre 46: Création PDF dans iOS

Examples

Créer un PDF

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

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

[self drawText];

UIGraphicsEndPDFContext();
```

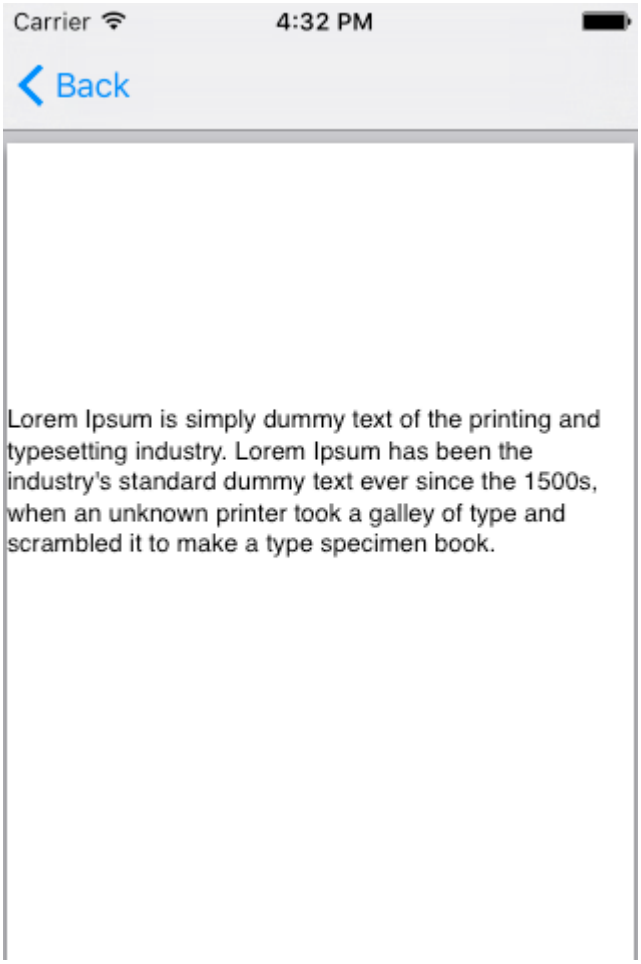
fileName est le fichier de document dans lequel vous allez ajouter ou attacher

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

    NSString *path = [arrayPaths objectAtIndex:0];

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

Où **drawText** est

```
   (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);
}
```



Afficher le 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];
```

Plusieurs pages PDF

```
 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();
```

Créer un PDF à partir de n'importe quel document Microsoft chargé dans UIWebview

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

Tout d'abord implémenter le protocole 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
```

Ensuite, appelez ci-dessous la méthode après le chargement du document dans `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");
    }
});}
```

Lire Création PDF dans iOS en ligne: https://riptutorial.com/fr/ios/topic/2416/creation-pdf-dans-ios

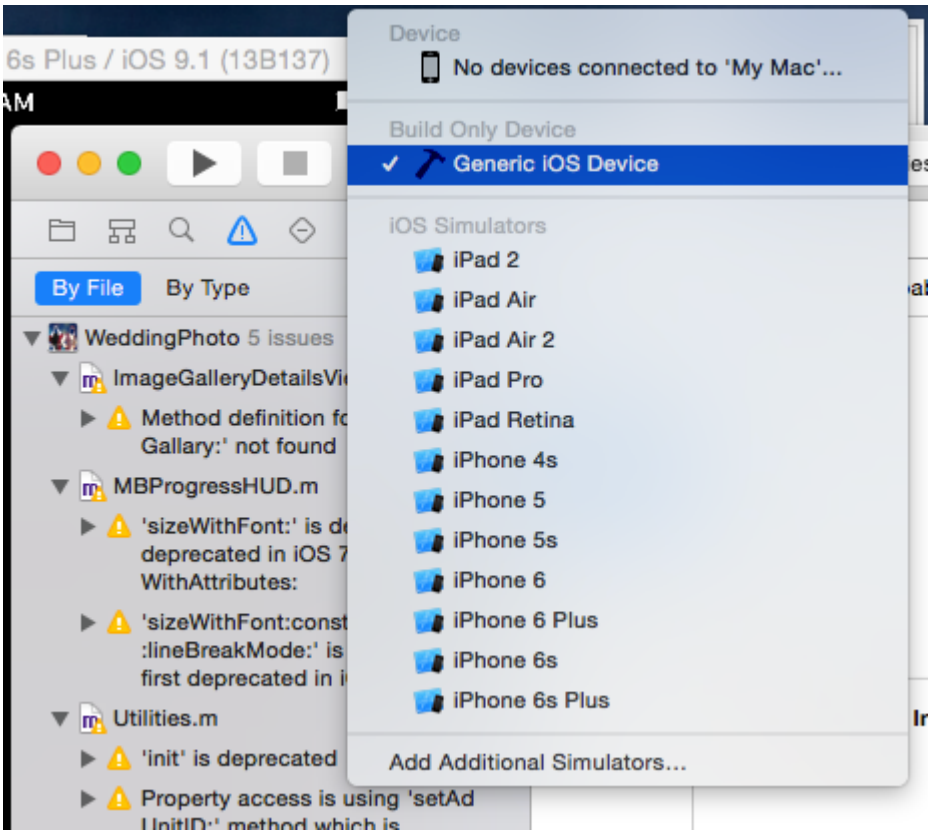
Chapitre 47: Créer un fichier .ipa à télécharger sur AppStore avec Applicationloader

Examples

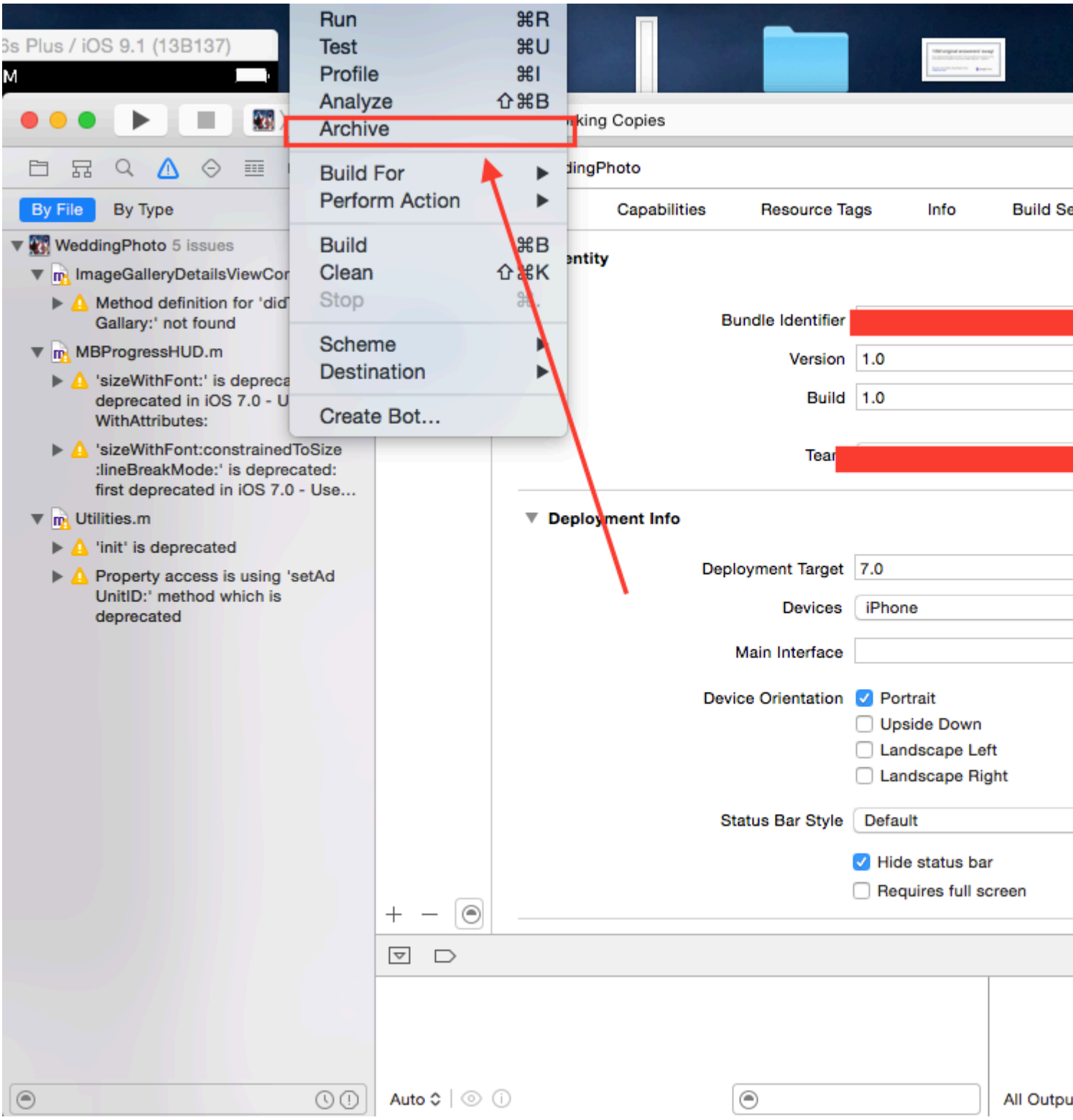
créer un fichier .ipa pour télécharger une application sur une application avec Application Loader

Si vous souhaitez télécharger le fichier .ipa vers itunesconnect **sans intégrer le compte développeur dans Xcode** et que vous souhaitez utiliser le **chargeur d'application** . alors vous pouvez **générer .ipa avec iTunes** .

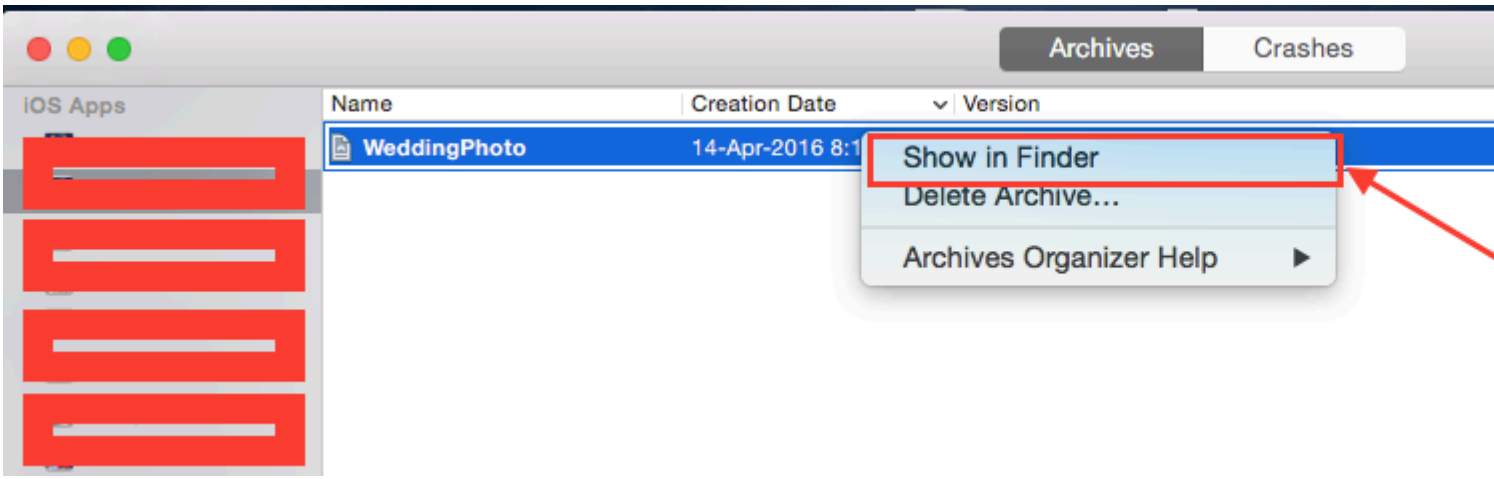
Étape 1: - Sélectionnez le dispositif en place du simulateur.



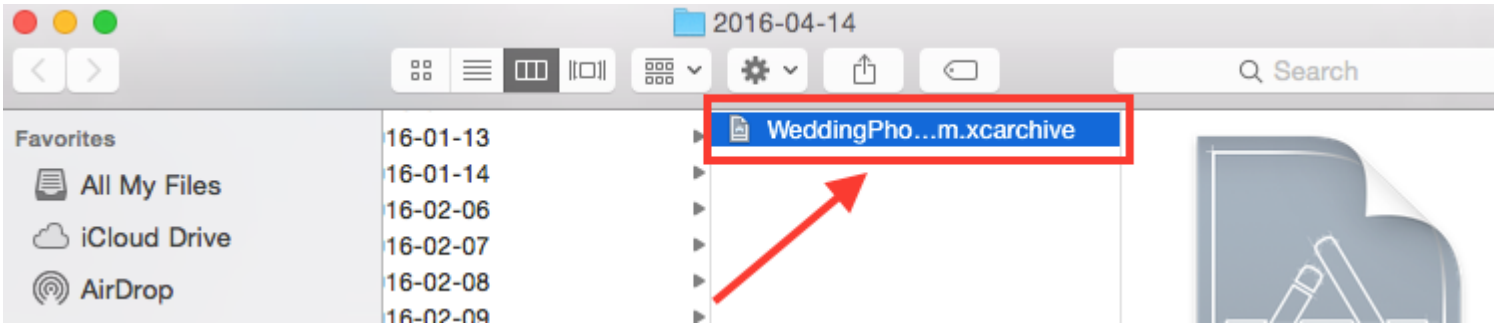
Étape 2: - Aller au produit -> sélectionner Archive



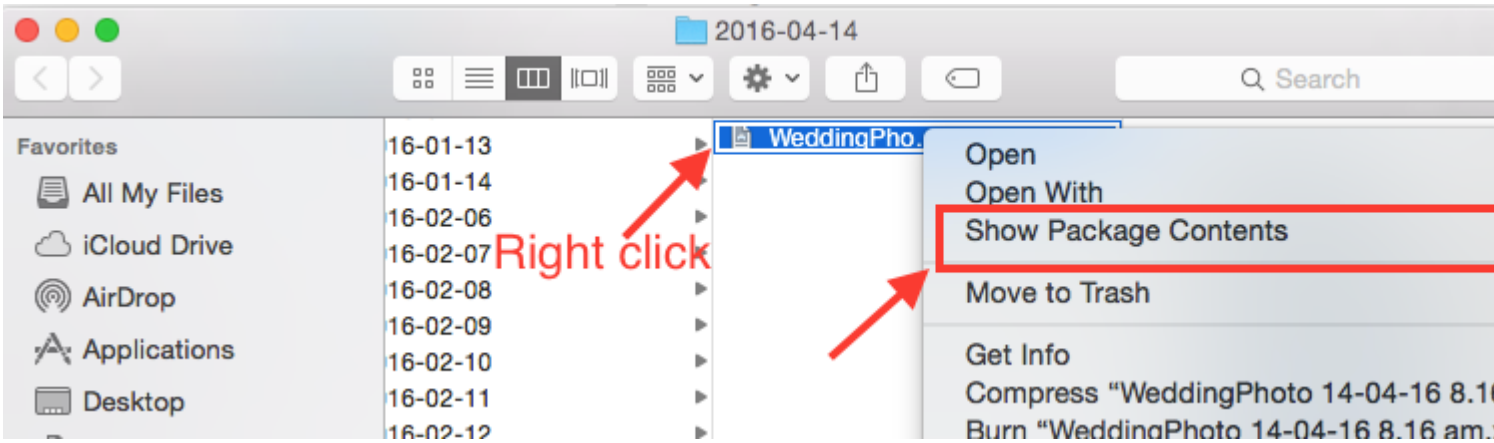
Étape 3: - Après le processus compliqué, cliquez avec le bouton droit de la souris sur votre archive -> et sélectionnez Afficher dans le Finder.



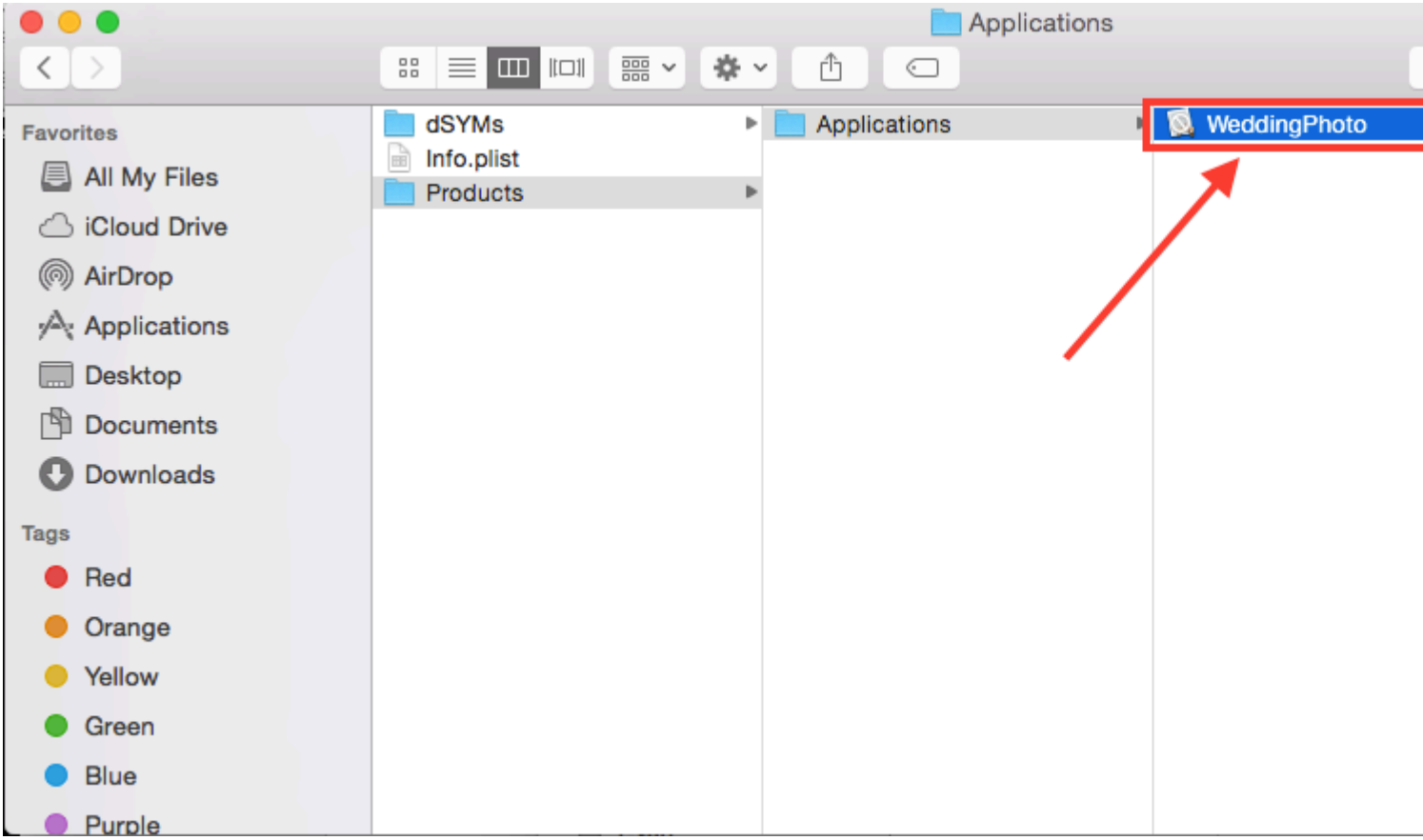
Etape 4: - lorsque vous cliquez sur show in finder, vous allez rediriger vers le dossier Archive, ressemble à ceci



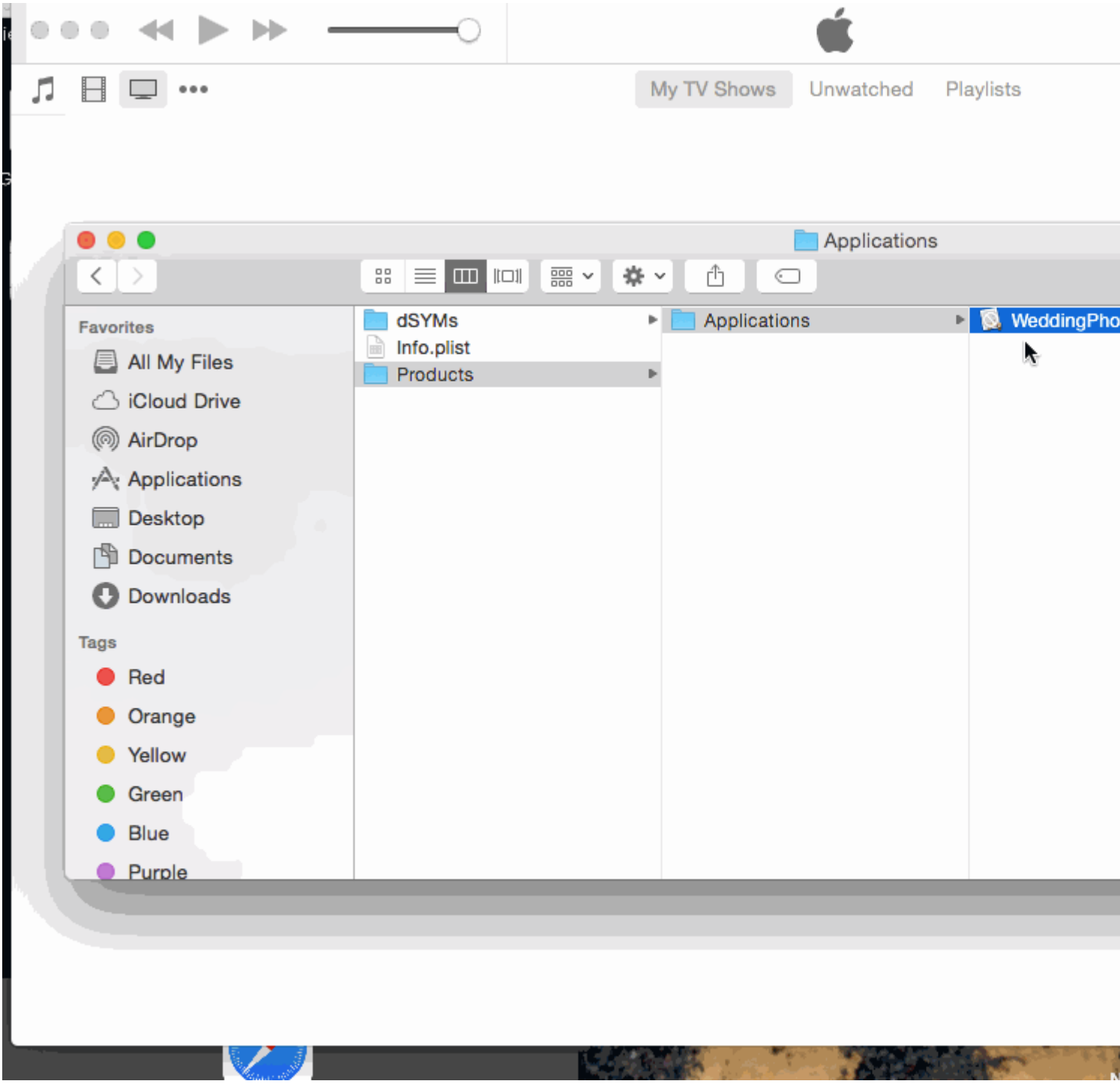
Étape 5: - Cliquez avec le bouton droit sur le fichier .xarchive -> sélectionnez l'option Afficher dans le Finder.



Étape 6: - Aller au dossier du produit -> Dossier d'application -> Vous trouverez votre nom de projet.app



Étape 7: - Maintenant, pour convertir .app en .ipa, faites simplement glisser et déposez-le dans iTunes. vérifier l'image ci-dessous,



Étape 8: - Maintenant, placez ce fichier .ipa en lieu sûr et utilisez-le lors du téléchargement avec le chargeur d'application.

Remarque: - si vous voulez savoir comment télécharger une application avec le chargeur d'application, cochez cette case,

Télécharger l'application avec l'application Loader

MODIFIER :-

ATTENTION: - Ne faites pas .ipa en changeant d'extension de .aap à .zip et de .zip à .ipa.

J'ai vu dans beaucoup de réponses que, ils ont suggéré compresser le fichier .app, puis changer

l'extension de .zip à .ipa. Cela ne fonctionne pas maintenant. Par cette méthode, vous obtiendrez une erreur comme,

> IPA n'est pas valide, il n'inclut pas de répertoire de charge utile.

Lire Créer un fichier .ipa à télécharger sur AppStore avec Applicationloader en ligne: https://riptutorial.com/fr/ios/topic/6119/creer-un-fichier--ipa-a-telecharger-sur-appstore-avec-applicationloader

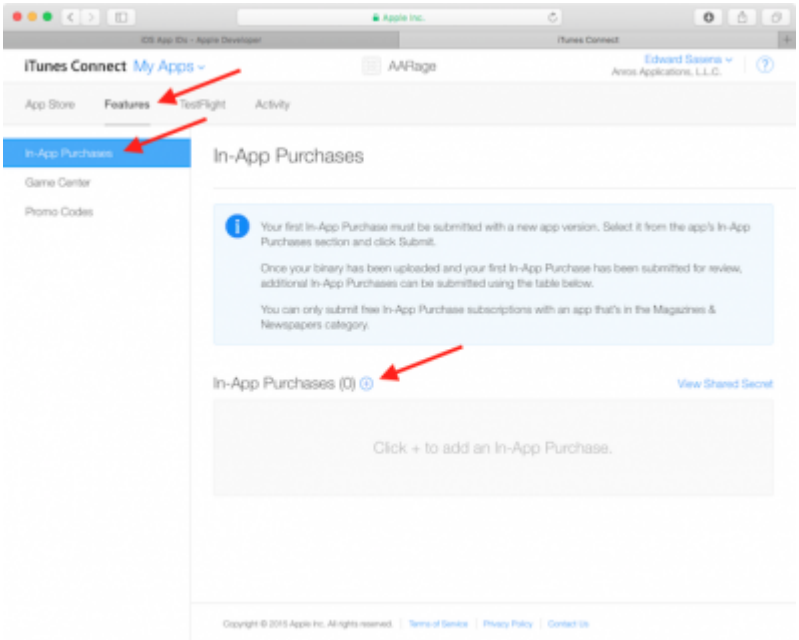
Chapitre 48: Créer un ID d'application

Examples

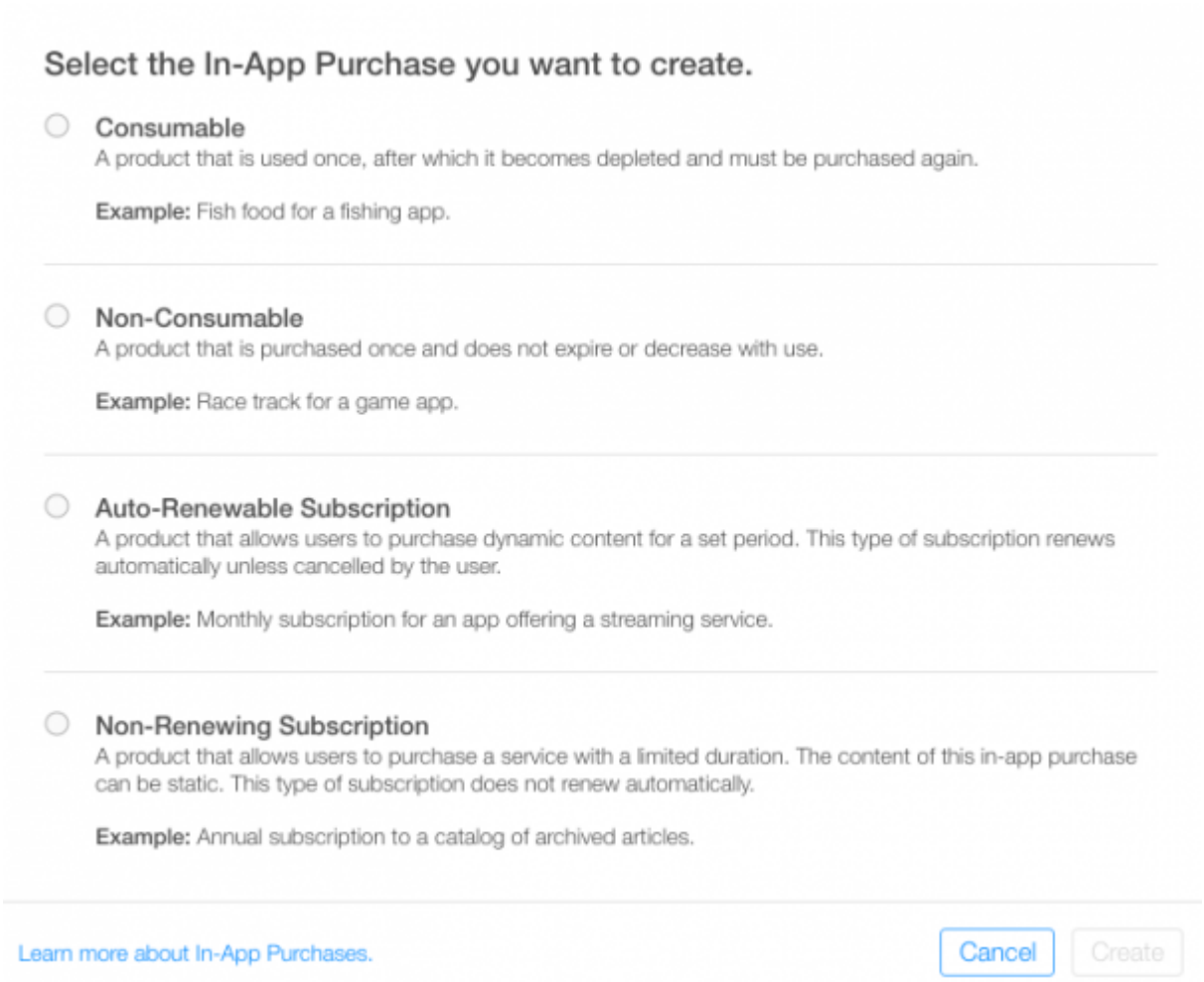
Création de produits d'achat intégrés

- Lorsque vous proposez un IAP dans une application, vous devez d'abord ajouter une entrée pour chaque achat individuel dans iTunes Connect. Si vous avez déjà répertorié une application à vendre dans le magasin, le processus est similaire et inclut des éléments tels que le choix d'un niveau de prix pour l'achat. Lorsque l'utilisateur effectue un achat, l'App Store gère le processus complexe de facturation du compte iTunes de l'utilisateur. Il existe de nombreux types d'IAP que vous pouvez ajouter:
    - **Consommable** : Ceux-ci peuvent être achetés plus d'une fois et peuvent être utilisés. Ce sont des choses comme des vies supplémentaires, de la monnaie en jeu, des bonus temporaires, etc.
    - **Non consommable** : Quelque chose que vous achetez une fois et que vous prévoyez avoir en permanence, comme des niveaux supplémentaires et un contenu déverrouillable.
    - **Abonnement non renouvelable** : contenu disponible pour une période déterminée.
    - **Renouvellement automatique** : Un abonnement répétitif tel qu'un abonnement mensuel à raywenderlich.com.

Vous ne pouvez proposer que des achats intégrés pour des articles numériques et non pour des biens ou des services physiques. Pour plus d'informations sur tout cela, consultez la documentation complète d'Apple sur la création de produits d'achat intégrés. Maintenant, tout en affichant l'entrée de votre application dans iTunes Connect, cliquez sur l'onglet Fonctions, puis sélectionnez Achats intégrés. Pour ajouter un nouveau produit IAP, cliquez sur le + à droite des achats intégrés.



Vous verrez la boîte de dialogue suivante apparaître:



Lorsqu'un utilisateur achète un comic rage dans votre application, vous voudrez qu'il y ait toujours accès. Sélectionnez donc Non-Consommable, puis cliquez sur Créer. Ensuite, remplissez les détails du PAI comme suit:

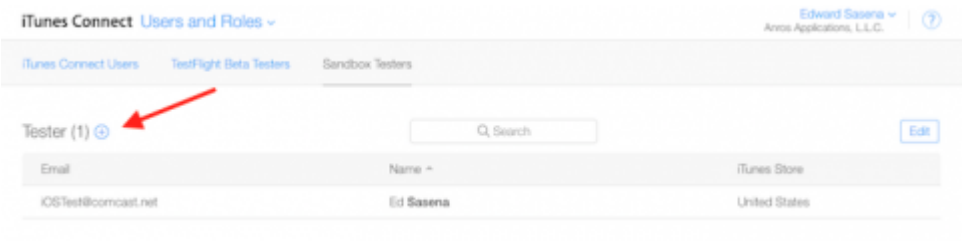
- **Nom de référence** : Un pseudonyme identifiant l'IAP dans iTunes Connect. Ce nom n'apparaît nulle part dans l'application. Le titre de la bande dessinée que vous allez débloquer avec cet achat est **«Girlfriend of Drummer»** , alors entrez ici.
- **ID du produit** : il s'agit d'une chaîne unique identifiant l'IAP. En règle générale, il est préférable de commencer par l'ID du lot, puis d'ajouter un nom unique spécifique à cet article. Pour ce tutoriel, assurez-vous d'ajouter «GirlfriendOfDrummerRage», car il sera utilisé ultérieurement dans l'application pour rechercher la bande dessinée à débloquer. Donc, par exemple: com.theNameYouPickedEarlier.Rage.GirlFriendOfDrummerRage.
- **Cleared for Sale** : Active ou désactive la vente de l'IAP. Vous voulez l'activer!
- **Niveau de prix** : Le coût du PAI. Choisissez le niveau 1.

Faites maintenant défiler jusqu'à la section Localizations et notez qu'il existe une entrée par défaut pour l'anglais (États-Unis). Entrez «Girlfriend of Drummer» pour le nom complet et la description. Cliquez sur Enregistrer. Génial! Vous avez créé votre premier produit IAP.

Une autre étape est nécessaire avant de pouvoir explorer un code. Lorsque vous testez des achats intégrés dans une version de développement d'une application, Apple fournit un environnement de test qui vous permet «d'acheter» vos produits IAP sans créer de transactions financières.

Création d'un utilisateur Sandbox

Dans iTunes Connect, cliquez sur iTunes Connect dans le coin supérieur gauche de la fenêtre pour revenir au menu principal. Sélectionnez Users and Roles, puis cliquez sur l'onglet Sandbox Testers. Cliquez sur + à côté du titre "Testeur".



Remplissez les informations et cliquez sur Enregistrer lorsque vous avez terminé. Vous pouvez créer un prénom et un nom pour votre utilisateur test, mais l'adresse e-mail choisie doit être une adresse e-mail réelle, car une vérification sera envoyée à l'adresse par Apple. Une fois que vous avez reçu cet email, assurez-vous de cliquer sur le lien pour vérifier votre adresse. L'adresse e-mail que vous entrez ne doit PAS non plus être associée à un compte d'identification Apple. Astuce: si vous avez un compte gmail, vous pouvez simplement utiliser un alias d'adresse au lieu d'avoir à créer un nouveau compte

Lire Créer un ID d'application en ligne: https://riptutorial.com/fr/ios/topic/10854/creer-un-id-d-application

Chapitre 49: Créer une infrastructure personnalisée dans iOS

Examples

Créer un cadre dans Swift

suivez ces étapes pour créer Custom Framework dans Swift-IOS:

1. Créez un nouveau projet. Dans Xcode
2. Choisissez iOS / Framework & Library / Cocoa Touch Framework pour créer un nouveau framework
3. cliquez sur suivant et définissez le productName
4. cliquez sur suivant et choisissez le répertoire pour y créer le projet
5. ajouter du code et des ressources au projet créé

Cadre créé avec succès

pour ajouter un cadre créé à un autre projet, vous devez d'abord créer un espace de travail Ajouter "projet cible" et "projet cadre" à l'espace de travail, puis:

1. aller à l'onglet général du projet cible
2. faites glisser le fichier "* .framework" dans le dossier de produit du projet d'infrastructure dans la section "Fichiers binaires incorporés"
3. à utiliser dans n'importe quel framework ViewController ou classe à importer dans chaque fichier

Lire Créer une infrastructure personnalisée dans iOS en ligne:
https://riptutorial.com/fr/ios/topic/7331/creer-une-infrastructure-personnalisee-dans-ios

Chapitre 50: Créer une vidéo à partir d'images

Introduction

Créer une vidéo à partir d'images à l'aide de AVFoundation

Examples

Créer une vidéo à partir de UIImages

Tout d'abord, vous devez créer `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` besoin d'au moins une entrée de rédacteur d'actifs.

```
 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
 }
```

Pour ajouter `CVPixelBufferRef` à `AVAssetWriterInput` nous devons créer `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];
```

Maintenant, nous pouvons commencer à écrire

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

Voici une fonction pour obtenir `CVPixelBufferRef` partir de `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;
}
```

Lire Créer une vidéo à partir d'images en ligne: https://riptutorial.com/fr/ios/topic/10607/creer-une-video-a-partir-d-images

Chapitre 51: CTCallCenter

Examples

Intercepter les appels depuis votre application, même en arrière-plan

De la documentation Apple:

> Utilisez la classe CTCallCenter pour obtenir une liste des appels cellulaires en cours et pour répondre aux modifications d'état des appels, par exemple d'un état de numérotation à un état connecté. Ces changements d'état sont appelés événements d'appel cellulaire.

CTCallCenter a pour but de permettre au développeur de mettre en pause l'état de son application lors d'un appel afin de donner à l'utilisateur la meilleure expérience possible.

Objectif c:

Tout d'abord, nous définirons un nouveau membre de classe dans la classe que nous voulons gérer les interceptions:

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

Dans notre classe init (constructeur), nous allouerons une nouvelle mémoire pour notre membre de la classe:

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

Ensuite, nous invoquerons notre nouvelle méthode qui gère réellement les interceptions:

```
- (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");
```

```
    }
 }];
}
```

Ça y est, si l'utilisateur utilisera votre application et recevra un appel téléphonique, vous pourrez intercepter cet appel et gérer votre application pour un état de sauvegarde.

Il est à noter qu'il existe 4 états d'appel que vous pouvez intercepter:

```
CTCallStateDialing
CTCallStateIncoming
CTCallStateConnected
CTCallStateDisconnected
```

Rapide:

Définissez votre classe dans la classe concernée et définissez-la:

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

Que se passera-t-il si votre application est en arrière-plan et que vous devez intercepter les appels lorsque l'application est en arrière-plan?

Par exemple, si vous développez une application d' **entreprise,** vous pouvez simplement ajouter 2 fonctionnalités (VOIP et fond de page) dans l'onglet Fonctionnalités:

Votre cible de projet -> Capacités -> Modes d'arrière-plan -> Marquer le transfert de voix sur IP et d'arrière-plan

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
    }
}
```

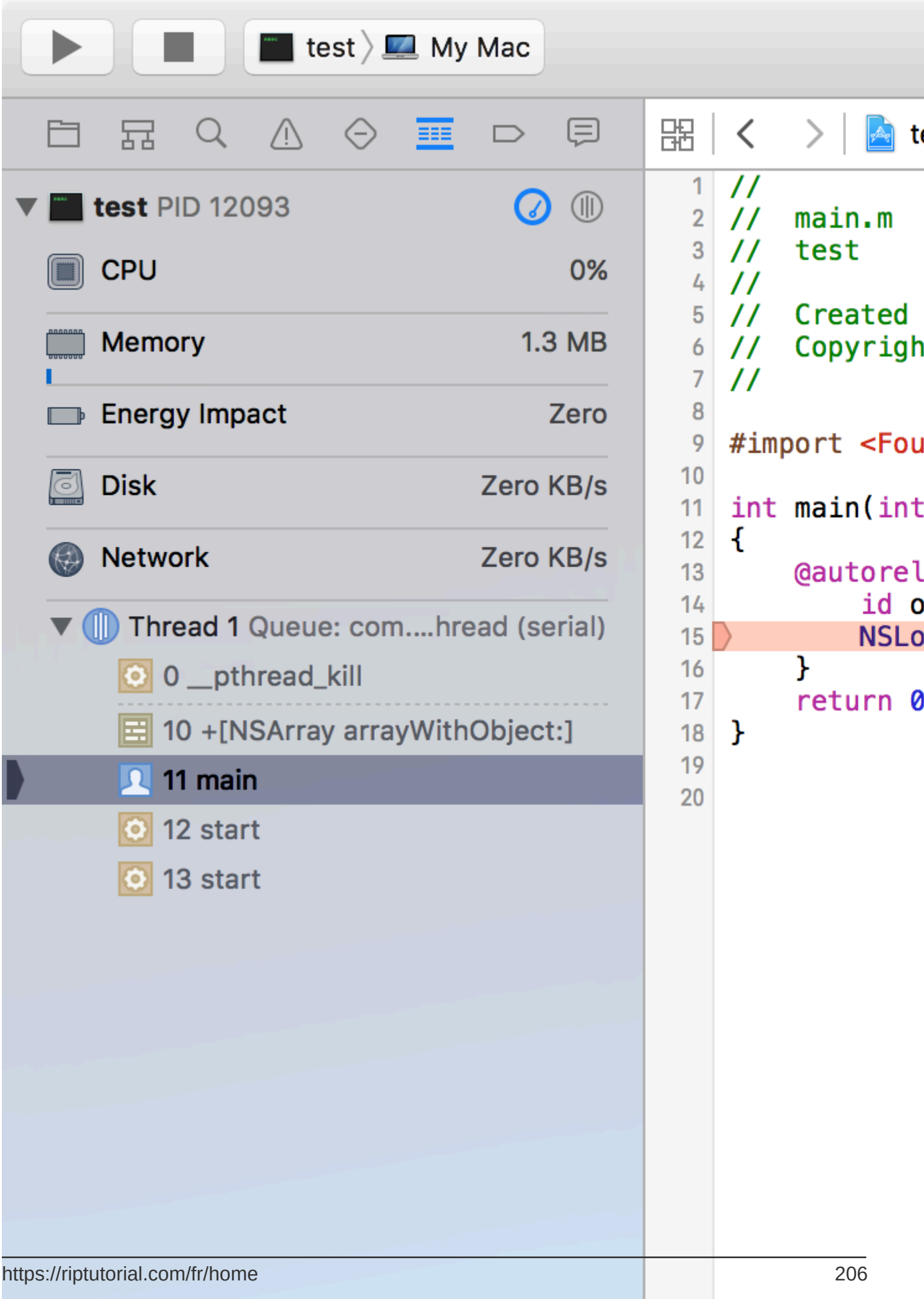
Lire CTCallCenter en ligne: https://riptutorial.com/fr/ios/topic/3007/ctcallcenter

Chapitre 52: Débogage des pannes

Examples

Recherche d'informations sur un crash

Lorsque votre application plante, Xcode entre dans le débogueur et affiche plus d'informations sur le crash:


test
My Mac
test PID 12093
CPU
0%
Memory
1.3 MB
Energy Impact
Zero
Disk
Zero KB/s
Network
Zero KB/s
Thread 1 Queue: com....hread (serial)
0 __pthread_kill
10 +[NSArray arrayWithObject:]
11 main
12 start
13 start
//
// main.m
// test
//
// Created
// Copyrigh
//
#import <Fou
int main(int
{
@autorel
id o
NSLo
}
return 0
}

`retour`, vous obtiendrez une représentation textuelle de la trace de la pile que vous pouvez copier et coller:

```
(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
```

Le débogage de SIGABRT et EXC_BAD_INSTRUCTION se bloque

Un `SIGABRT` ou un EXC_BAD_INSTRUCTION signifie généralement que l'application s'est planté intentionnellement car une vérification a échoué. Ceux-ci doivent enregistrer un message sur la console du débogueur avec plus d'informations; vérifiez ici pour plus d'informations.

De nombreux `SIGABRT` sont causés par des exceptions Objective-C non `SIGABRT` . Il y a *beaucoup* de raisons pour lesquelles des exceptions peuvent être levées, et elles enregistrent *toujours* beaucoup d'informations utiles sur la console.

- `NSInvalidArgumentException` , ce qui signifie que l'application a transmis un argument non valide à une méthode
- `NSRangeException` , ce qui signifie que l'application a tenté d'accéder à un index hors limites d'un objet tel qu'un `NSArray` ou un `NSString`
- `NSInternalInconsistencyException` signifie qu'un objet découvert était dans un état inattendu.
- `NSUnknownKeyException` signifie généralement que vous avez une mauvaise connexion dans un XIB. Essayez certaines des réponses à cette question .

Déboguer EXC_BAD_ACCESS

`EXC_BAD_ACCESS` signifie que le processus a tenté d'accéder à la mémoire de manière non valide, par exemple en déréférençant un pointeur `NULL` ou en écrivant dans une mémoire en lecture seule. C'est le type de crash le plus difficile à déboguer, car il ne contient généralement pas de message d'erreur, et certains plantages peuvent être *très* difficiles à reproduire et / ou se produire dans un code sans rapport avec le problème. Cette erreur est très rare dans Swift, mais si cela se produit, vous pouvez souvent obtenir des blocages plus faciles à déboguer en réduisant les optimisations du compilateur.

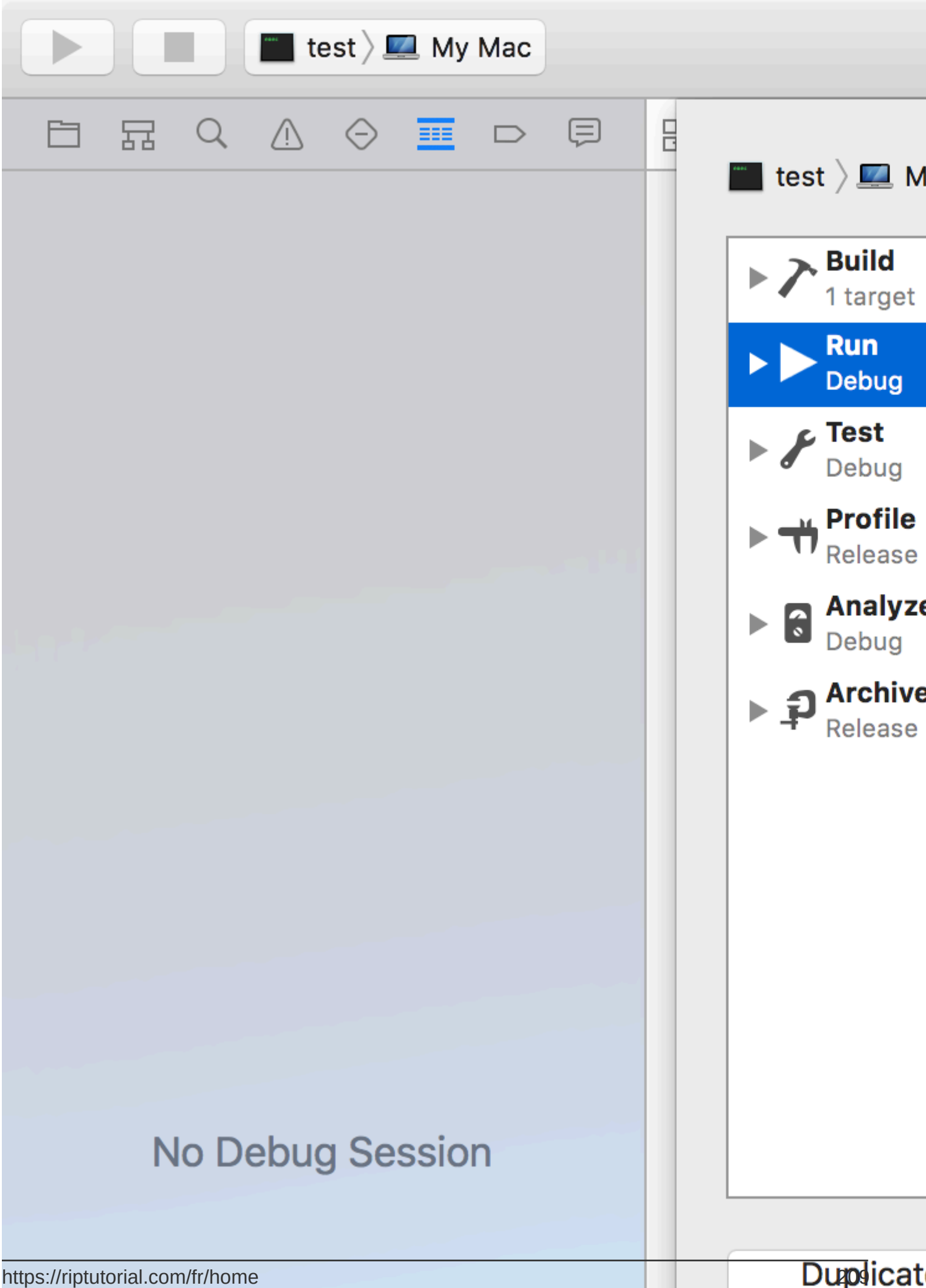
La plupart des erreurs `EXC_BAD_ACCESS` sont provoquées en essayant de déréférencer un pointeur

`NULL` . Si tel est le cas, l'adresse indiquée dans la flèche rouge sera généralement un nombre hexadécimal inférieur à une adresse mémoire normale, souvent `0x0` . Définissez des points d'arrêt dans le débogueur ou ajoutez des instructions `printf` / `NSLog` occasionnelles pour savoir pourquoi ce pointeur est `NULL` .

Un `EXC_BAD_ACCESS` qui se produit de manière moins fiable ou qui n'a aucun sens pourrait résulter d'un problème de gestion de la mémoire. Les problèmes courants qui peuvent causer ceci sont:

- Utiliser de la mémoire qui a été désallouée
- Essayer d'écrire après la fin d'un tableau C ou d'un autre type de tampon
- Utiliser un pointeur non initialisé

Dans la section Diagnostics de l'éditeur de schéma, Xcode inclut quelques outils utiles pour résoudre les problèmes de mémoire:


test
My Mac
No Debug Session
test
Build
1 target
Run
Debug
Test
Debug
Profile
Release
Analyze
Debug
Archive
Release

Chapitre 53: Deep Linking dans iOS

Remarques

Documentation Apple utile avec des exemples et des éclaircissements.

Examples

Ouvrir une application basée sur son schéma d'URL

Pour ouvrir une application avec un schéma d'URL défini pour la `todolist://` :

Objectif c

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

Rapide

```
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>
```

> **Remarque:** Il est utile de vérifier si le lien peut être ouvert pour afficher un message approprié à l'utilisateur. Cela peut être fait en utilisant la méthode `canOpenURL:`

Ajout d'un schéma d'URL à votre propre application

Disons que vous travaillez sur une application appelée `MyTasks` et que vous souhaitez autoriser les URL entrantes à créer une nouvelle tâche avec un titre et un corps. L'URL que vous concevez pourrait ressembler à ceci:

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

(Bien sûr, les paramètres de `text` et de `body` sont utilisés pour remplir notre tâche que nous créons!)

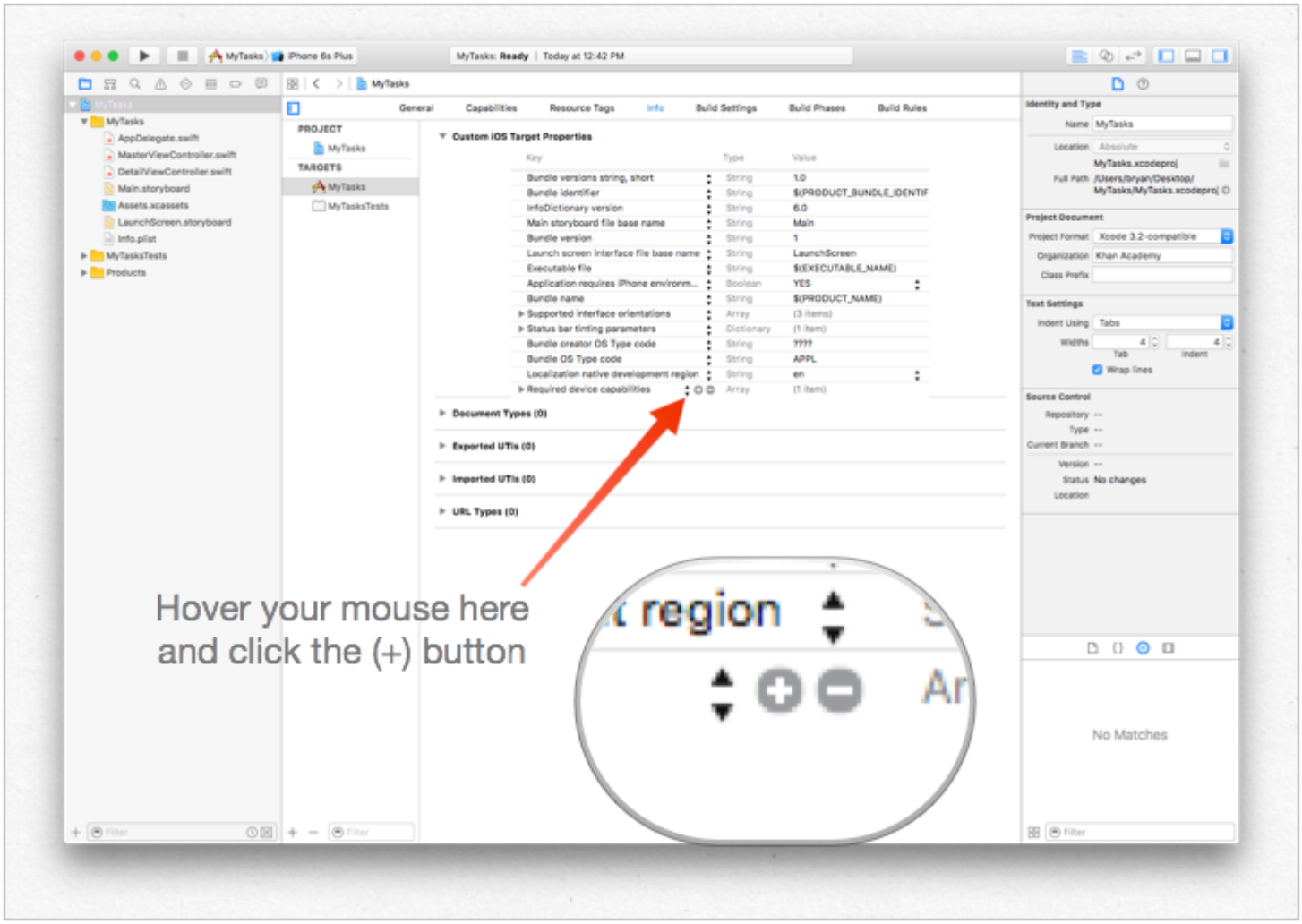
Voici les grandes étapes pour ajouter ce schéma d'URL à votre projet:

1. Enregistrez un schéma d'URL dans le fichier `Info.plist` votre application afin que le système sache quand acheminer une URL vers votre application.

2. Ajoutez une fonction à votre `UIApplicationDelegate` qui accepte et gère les URL entrantes.
3. Effectuez n'importe quelle tâche qui doit se produire lorsque cette URL est ouverte.

Première étape: Enregistrez un schéma d'URL dans Info.plist:

Tout d'abord, nous devons ajouter une entrée "Types d'URL" à notre fichier Info.plist. Cliquez sur le bouton (+) ici:



... puis entrez un identifiant unique pour votre application, ainsi que le schéma d'URL que vous souhaitez utiliser. Être spécifique! Vous ne voulez pas que le schéma d'URL entre en conflit avec l'implémentation d'une autre application. Mieux vaut être trop long ici que trop court!

Deuxième étape: gérer l'URL dans

UIApplicationDelegate

Nous devons implémenter `application:openURL:options:` sur notre `UIApplicationDelegate` . Nous allons inspecter l' `URL` entrante et voir si une action est possible!

Une mise en œuvre serait la suivante:

```
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
}
```

Troisième étape: Effectuez une tâche en fonction de l'URL.

Lorsqu'un utilisateur ouvre votre application via une URL, il s'attend probablement à ce que *quelque chose* se produise. Peut-être que la navigation vers un élément de contenu, peut-être la création d'un nouvel élément - dans cet exemple, nous allons créer une nouvelle tâche dans l'application!

Dans le code ci-dessus, nous pouvons voir un appel à `self.rootViewController.createTaskWithTitle(:body:)` - en supposant que votre `AppDelegate` ait un pointeur sur son contrôleur de vue racine qui implémente correctement la fonction, vous êtes prêt!

Configuration de Deeplink pour votre application

Configurer les liens profonds pour votre application est facile. Vous avez juste besoin d'une petite URL à l'aide de laquelle vous souhaitez ouvrir votre application.

Suivez les étapes pour configurer la liaison profonde pour votre application.

1. Permet de créer un projet et nommez-le DeepLinkPOC.
2. Maintenant, sélectionnez votre cible de projet.
3. Après avoir sélectionné la cible, sélectionnez l'onglet "info".
4. Faites défiler vers le bas jusqu'à ce que vous voyiez une option de **types d'URL**
5. Cliquez sur l'option '+'.

6. Vous verrez des **schémas d'URL** ajouter une chaîne à l'aide de laquelle vous voulez ouvrir votre application. Permet d'ajouter " **DeepLinking** " dans les schémas d'URL.

Donc, pour ouvrir votre application, vous pouvez la lancer en tapant **"DeepLinking: //"** dans votre safari. Votre chaîne de liens profonds a le format suivant.

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

où, Schéma: "DeepLinking" Host: "path" path: "Page1"

Note : Même si vous n'ajoutez pas d'hôte ni de chemin, il lancera l'application, donc pas de soucis.

7. Maintenant, ajoutez la méthode suivante à votre applicationdelegate.

Rapide:

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

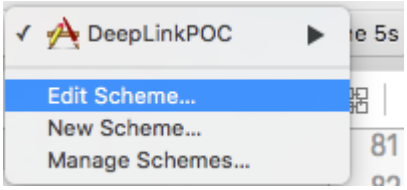
Objectif c:

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

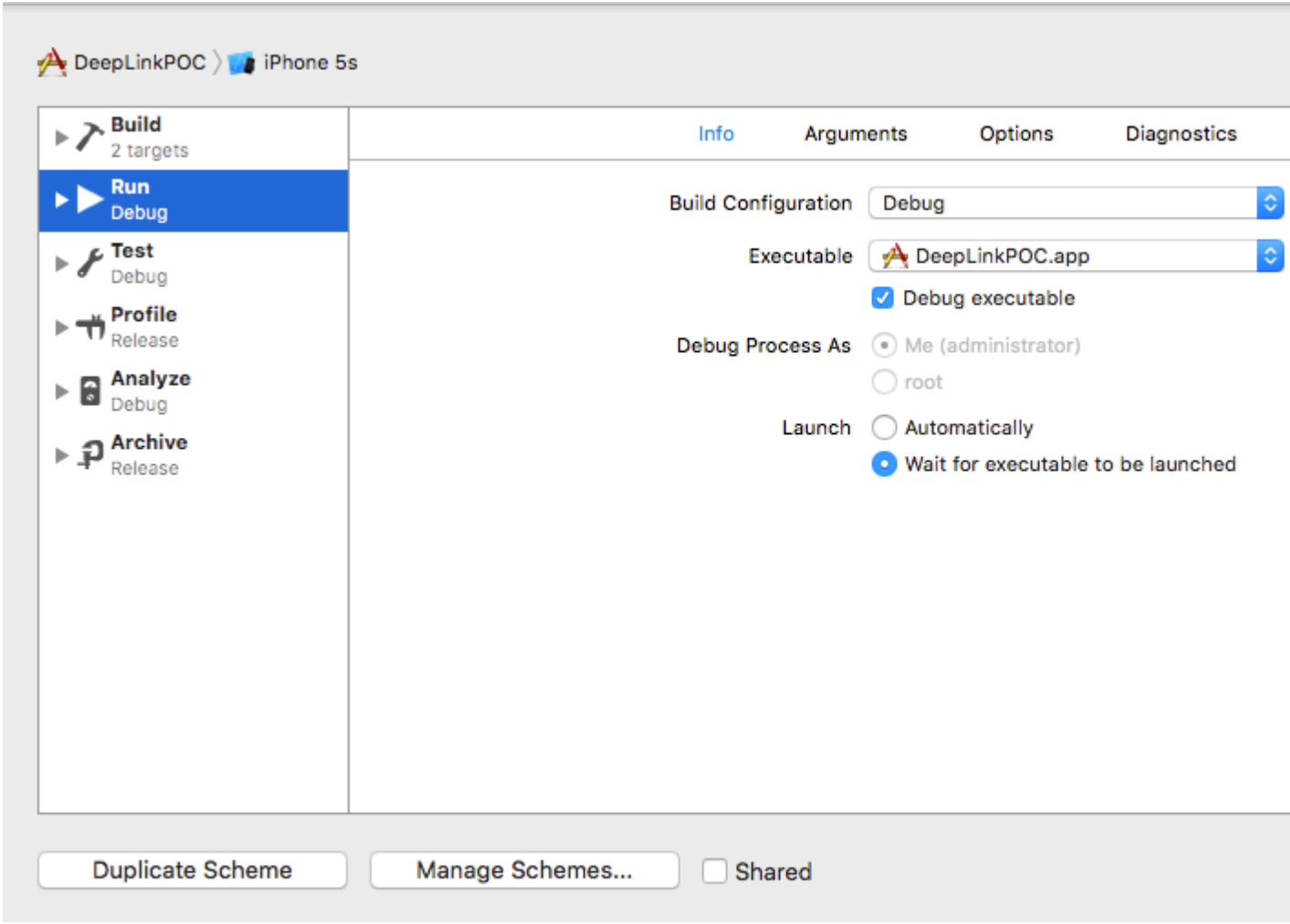
La méthode ci-dessus est appelée chaque fois que votre application est lancée en utilisant une chaîne de liens profonds que vous avez définie pour votre application.

8. Il est maintenant temps d'installer votre application, mais attendez avant de sauter directement pour exécuter le bouton. Faites un petit changement dans la méthode de lancement de l'application du schéma.

- Sélectionnez et modifiez votre schéma comme



- changer son type de lancement et fermer



9. Maintenant, cliquez sur le bouton Exécuter (si vous le souhaitez, vous pouvez ajouter un point d'arrêt à vos méthodes didFinishLaunchingWithOptions et openURL pour observer les valeurs)

10. Vous verrez un message "En attente de DeepLinkPOC (ou du nom de votre application) pour lancer".

11. Ouvrez safari et tapez " **DeepLinking: //** " dans la barre de recherche. Cela affichera une invite "ouvrez cette page dans DeepLinkPOC". Cliquez sur Ouvrir pour lancer votre application.

J'espère que vous saurez comment configurer les liens profonds pour votre application :)

Lire Deep Linking dans iOS en ligne: https://riptutorial.com/fr/ios/topic/5173/deep-linking-dans-ios

Chapitre 54: Définir l'arrière-plan de la vue

Examples

Définir le fond de vue

Objectif c:

```
view.backgroundColor = [UIColor redColor];
```

Rapide:

```
view.backgroundColor! = UIColor.redColor()
```

Swift 3

```
view.backgroundColor = UIColor.redColor
```

Remplir le fond Image d'un UIView

Objectif c

```
 UIGraphicsBeginImageContext(self.view.frame.size);
 [[UIImage imageNamed:@"image.png"] drawInRect:self.view.bounds];
 UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
 UIGraphicsEndImageContext();
 self.view.backgroundColor = [UIColor colorWithPatternImage:image];
```

Définir le backround avec l'image

```
self.view.backgroundColor = [UIColor colorWithPatternImage:[UIImage
imageNamed:@"Background.png"]];
```

Création d'une vue en arrière-plan dégradé

Pour créer un arrière-plan avec un dégradé, vous pouvez utiliser 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)
}
```

Cela peut être appelé sur viewDidLoad () comme ceci:

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

Les variables CAGradientLayer locations et limites peuvent prendre plusieurs valeurs pour créer un calque dégradé avec le nombre de couleurs souhaité. De la documentation:

> Par défaut, les couleurs sont réparties uniformément sur le calque, mais vous pouvez éventuellement spécifier des emplacements pour contrôler les positions de couleur dans le dégradé.

Lire Définir l'arrière-plan de la vue en ligne: https://riptutorial.com/fr/ios/topic/6854/definir-l-arriere-plan-de-la-vue

Chapitre 55: Délégués de multidiffusion

Introduction

Modèle permettant d'ajouter des fonctionnalités de multidiffusion aux contrôles iOS existants. L'ajout de la multidiffusion permet d'améliorer la clarté et la réutilisation du code.

Examples

Délégués de multidiffusion pour tous les contrôles

Transférer les messages d'un objet à un autre par les délégués, en diffusant ces messages à plusieurs observateurs.

Etape 1: - Créez la **classe** `NSObject` de `RRMulticastDelegate`

Étape 2: - Suivre le code implémenter dans le fichier `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
```

Étape 3: - Suivre le code implémenter dans le fichier `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
```

Étape 4: - Créez la **classe** de **catégorie** `NSObject` de `RRProperty`

Étape 5: - Suivre le code implémenter dans le fichier `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
```

Étape 6: - Suivre le code implémenter dans le fichier `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
```

Enfin, vous pouvez utiliser multicast dalegate pour tous les contrôles ...

Pour ex ...

Importez votre classe viewcontroller dans le fichier `NSObject+RRProperty.h` pour accéder à ses méthodes de **définition du délégué / source de données de multidiffusion** .

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

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

Lire Délégués de multidiffusion en ligne: https://riptutorial.com/fr/ios/topic/10081/delegues-de-multidiffusion

Chapitre 56: Demande d'évaluation / révision

Introduction

Désormais à partir d'iOS 10.3, il n'est plus nécessaire de naviguer entre l'application et Apple Store pour l'évaluation / la révision. Apple a introduit la classe SKStoreReviewController dans le framework storekit. Dans quel développeur il suffit d'appeler la méthode de classe requestReview () de la classe SKStoreReviewController et le système gère l'intégralité du processus pour vous.

En outre, vous pouvez continuer à inclure un lien persistant dans les paramètres ou les écrans de configuration de votre application, qui sont liés de manière approfondie à la page de votre produit App Store. Opérer automatiquement

Examples

Noter / Réviser Application iOS

Saisissez simplement un code de ligne à partir duquel vous souhaitez que l'utilisateur notifie / examine votre application.

SKStoreReviewController.requestReview ()

Lire Demande d'évaluation / révision en ligne: https://riptutorial.com/fr/ios/topic/9678/demande-d-evaluation---revision

Chapitre 57: Détection de visage avec CoreImage / OpenCV

Examples

Détection des visages et des entités

Objectif c

Importez les éléments suivants sur votre ViewController

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

Appeler la fonction

```
[self faceDetector];
```

Définition de la fonction:

```
-(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)];


}
```

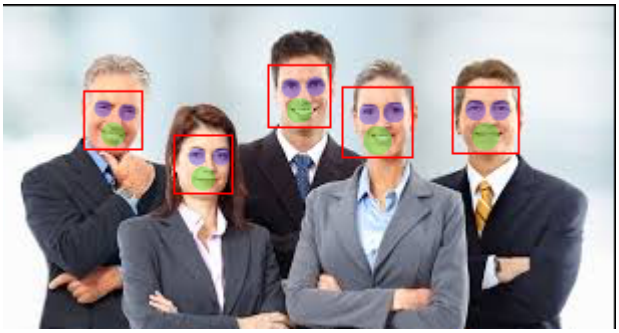
Fonction de visage de marque

```
//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];
        }
    }

    // }
}
```

L'écran de simulation pour la fonction



Lire Détection de visage avec CoreImage / OpenCV en ligne:
https://riptutorial.com/fr/ios/topic/7298/detection-de-visage-avec-coreimage---opencv

Chapitre 58: DispatchGroup

Introduction

Rubriques connexes:

Expédition Grand Central

Concurrence

Examples

introduction

Supposons que vous ayez plusieurs threads en cours d'exécution. Chaque thread effectue une tâche. Vous voulez être notifié soit sur le thread principal ou un autre thread, lorsque tous les threads de tâches sont terminés.

La solution la plus simple à un tel problème est un `DispatchGroup` .

Lorsque vous utilisez un `DispatchGroup` , pour chaque demande, vous `enter` le groupe et pour chaque demande terminée, vous `leave` le groupe.

Quand il n'y a pas de demandes plus dans le groupe, vous serez en `notify` (notification).

Usage:

```
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()
    }
}
```

Avec ce qui précède, vous n'avez pas à `wait` indéfiniment jusqu'à ce que toutes les tâches soient terminées. Vous pouvez afficher un chargeur AVANT que toutes les tâches aient démarré et fermer le chargeur APRÈS que toutes les tâches soient terminées. De cette façon, votre thread principal n'est pas bloqué et votre code reste propre.

Supposons maintenant que vous souhaitiez également `ordered` les tâches ou ajouter leurs réponses à un tableau de manière séquentielle. Vous pouvez faire ce qui suit:

```
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()
    }
}
```

Remarques

Chaque entrée doit avoir une sortie dans un `DispatchGroup` . Si vous oubliez de `leave` après être `entering` , vous vous mettez en place. Vous ne serez JAMAIS averti lorsque les tâches sont terminées.

Le montant de l' `enter` doit être égal au montant du `leave` .

Lire DispatchGroup en ligne: https://riptutorial.com/fr/ios/topic/4624/dispatchgroup

Chapitre 59: Domaine

Remarques

Ajout d'un nouvel objet RLMO à un domaine existant - Schéma et migrations

L'ajout de nouvelles classes de modèle à un domaine ne nécessite pas de migration ou de modification de version de schéma; apporter uniquement des modifications à un domaine existant.

Examples

Classe de modèle de base RLMObject avec clé primaire - Objective-C

Exemple de classe de modèle de base RLMObject utilisant une clé primaire et certaines propriétés par défaut génériques. Les sous-classes peuvent ensuite définir des métadonnées spécifiques à leurs besoins.

```
@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
```

Lire Domaine en ligne: https://riptutorial.com/fr/ios/topic/4084/domaine

Chapitre 60: Données de base

Introduction

Core Data est la couche modèle de votre application au sens le plus large possible. C'est le modèle dans le modèle Model-View-Controller qui imprègne le SDK iOS.

Core Data n'est pas la base de données de votre application, ni une API pour la persistance des données dans une base de données. Core Data est un framework qui gère un graphe d'objets. C'est aussi simple que ça. Les données de base peuvent conserver ce graphe en l'écrivant sur le disque, mais ce n'est pas l'objectif principal du framework.

Examples

Opérations sur les données de base

Pour obtenir le contexte:

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

Pour récupérer des données:

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

Pour récupérer des données avec le tri:

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

Pour ajouter des données:

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

Pour enregistrer le contexte:

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

Lire Données de base en ligne: https://riptutorial.com/fr/ios/topic/9489/donnees-de-base

Chapitre 61: Dynamique UIKit

Introduction

UIKit Dynamics est un moteur physique réel intégré à UIKit. Il vous permet de créer des interfaces qui se sentent réelles en ajoutant des comportements tels que la gravité, les pièces jointes, les collisions et les forces. Vous définissez les caractéristiques physiques que vous souhaitez que vos éléments d'interface adoptent, et le moteur dynamique prend en charge le reste.

Remarques

Lorsque vous utilisez UIKit Dynamics, vous devez garder à l'esprit que les vues positionnées par l'animateur ne peuvent pas être facilement positionnées par d'autres méthodes de mise en page iOS courantes.

Les nouveaux arrivants à UIKit Dynamics ont souvent du mal avec cette importante mise en garde. Placer des contraintes sur une vue qui est également un élément d'un `UIDynamicBehavior` entraînera probablement une confusion à la fois en tant que moteur de présentation automatique et lutte contre le moteur d'animation dynamique sur la position appropriée. De même, si vous tentez de définir directement le cadre d'une vue contrôlée par l'animateur, vous obtenez généralement une animation instable et un placement inattendu. L'ajout d'une vue en tant qu'élément à un `UIDynamicBehavior` signifie que l'animateur prendra la responsabilité de positionner une vue et que de tels changements de position de vue devront être implémentés via l'animateur.

Il est possible de définir une image en cours de mise à jour par un animateur dynamique, mais celle-ci doit être immédiatement suivie par l'envoi de messages à l'animateur afin de mettre à jour le modèle interne de l'animateur. Par exemple, si j'ai `UILabel` , `label` qui est un élément d'un `UIGravityBehavior` je peux le déplacer en haut de l'écran pour le regarder retomber en disant:

Rapide

```
label.frame = CGRect(x: 0.0, y: 0.0, width: label.intrinsicContentSize.width, height:
label.intrinsicContentSize.height)
dynamicAnimator.updateItem(usingCurrentState: label)
```

Objectif c

```
self.label.frame = CGRectMake(0.0, 0.0, self.label.intrinsicContentSize.width,
self.label.intrinsicContentSize.height);
[self.dynamicAnimator updateItemUsingCurrentState: self.label];
```

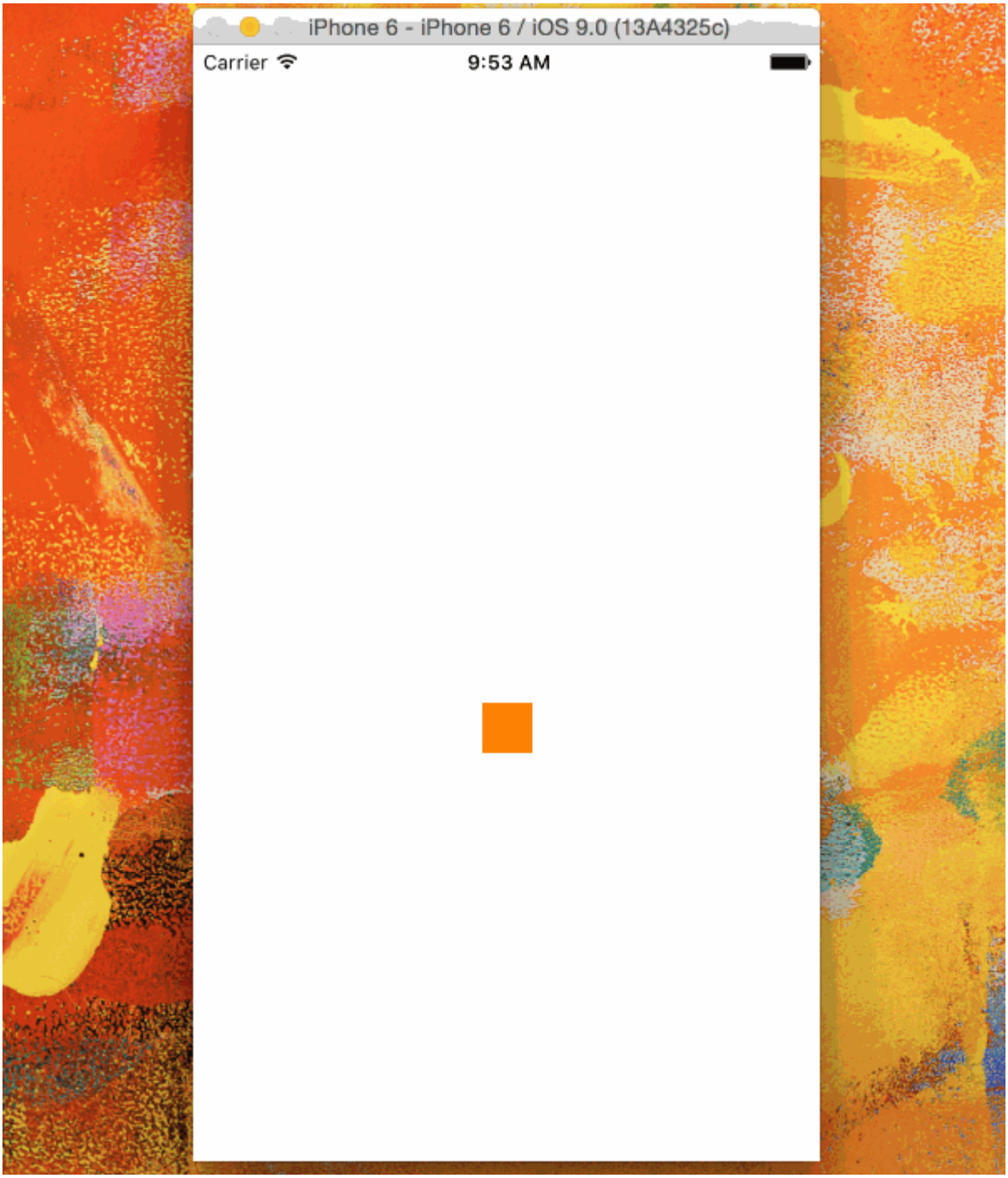
Après quoi l'animateur appliquera le comportement de gravité du nouvel emplacement de l'étiquette.

Une autre technique courante consiste à utiliser `UIDynamicBehaviors` pour positionner des vues. Par exemple, si vous souhaitez positionner une vue sous un événement tactile, la création d'un `UIAttachmentBehavior` et la mise à jour de son `anchorPoint` dans les `anchorPoint` de `touchesMoved` ou d' `UIGestureRecognizer` constituent une stratégie efficace.

Examples

La place tombante

Permet de tracer un carré au milieu de notre vue et de le faire tomber en bas et d’arrêter au bord inférieur de la limite inférieure de l’écran.



```
@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)
}
```

Vue panoramique basée sur la vitesse de geste

Cet exemple montre comment faire en sorte qu'une vue suive un geste panoramique et se déclenche de manière physique.

Rapide

```
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)
        }
    }
}
```

Objectif 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
```

Effet "Sticky Corners" en utilisant UIFieldBehaviors

Cet exemple montre comment obtenir un effet similaire à FaceTime si une vue est attirée vers le point une fois qu'elle pénètre dans une région particulière, en l'occurrence deux régions en haut et en bas.

Rapide

```
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)
        }
    }
}
```

Objectif 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)
    {
```

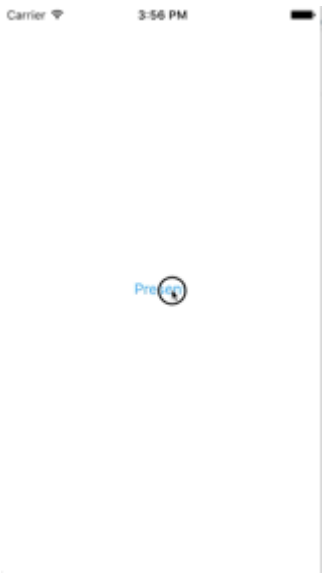
```
        [self.dynamicAnimator addBehavior:field];
    }

    [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
```

Pour plus d'informations sur `UIFieldBehaviors` consultez la session 2015 WWDC "Nouveautés de la dynamique UIKit et des effets visuels" et un exemple de code `UIFieldBehaviors` .

Transition personnalisée UIDynamicBehavior



Cet exemple montre comment créer une transition de présentation personnalisée pilotée par un `UIDynamicBehavior` composite. Nous pouvons commencer par créer un contrôleur de vue de présentation qui présentera un modal.

Rapide

```
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)
    }
}
```

Objectif 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
```

Lorsque vous appuyez sur le bouton actuel, nous créons un `ModalViewController` et définissons son style de présentation sur `.custom` et définissons son `transitionDelegate` sur lui-même. Cela nous permettra de proposer un animateur qui pilotera sa transition modale. Nous définissons également

le cadre de la vue `modal` afin qu'il soit plus petit que le plein écran.

Regardons maintenant `ModalViewController` :

Rapide

```
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)
    }
}
```

Objectif 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
```

Ici, nous créons le contrôleur de vue qui est présenté. De plus, parce que `ModalViewController` est son propre `transitioningDelegate` il est également chargé de vendre un objet qui gérera son animation de transition. Pour nous, cela signifie transmettre une instance de notre sous-classe composite `UIDynamicBehavior` .

Notre animateur aura deux transitions différentes: une pour la présentation et une pour le rejet. Pour la présentation, la vue du contrôleur de vue de présentation apparaîtra d'en haut. Et pour le rejeter, la vue semblera balancer d’une corde puis tomber. Comme `DropOutAnimator` est conforme à `UIViewControllerAnimatedTransitioning` majeure partie de ce travail sera effectuée dans son

implémentation de `func animateTransition(using transitionContext: UIViewControllerContextTransitioning)` .

Rapide

```
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
        }
    }
}
}
```

Objectif 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;
        }
    }
}
```

En tant que comportement composite, `DropOutAnimator` peut combiner différents comportements pour effectuer ses animations de présentation et de rejet. `DropOutAnimator` montre également comment utiliser le bloc d' `action` d'un comportement pour inspecter les emplacements de ses éléments, ainsi que le temps écoulé depuis une technique pouvant être utilisée pour supprimer les vues déplacées ou tronquer les animations qui n'ont pas encore atteint la stase.

Pour plus d'informations Session 2013 de la WWDC "Techniques avancées avec UIKit Dynamics" ainsi que SOLPresentingFun

Transformation de l'ombre avec la physique du monde réel à l'aide de UIDynamicBehaviors

Cet exemple montre comment effectuer une transition de présentation interactive avec une physique "réaliste" similaire à l'écran de notification d'iOS.

Pour commencer, nous avons besoin d'un contrôleur de vue de présentation sur lequel l'ombre apparaîtra. Ce contrôleur de vues fera également office de `UIViewControllerTransitioningDelegate` pour notre contrôleur de vue présenté et proposera aux animateurs notre transition. Nous allons donc créer des instances de nos animateurs interactifs (un pour la présentation et un pour le rejet). Nous allons également créer une instance du contrôleur de vue d'ombre, qui, dans cet exemple, n'est qu'un contrôleur de vue avec une étiquette. Parce que nous voulons que le même geste de panoramique dirige l'intégralité de l'interaction, nous transmettons des références au contrôleur de présentation et à la nuance à nos animateurs interactifs.

Rapide

```
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
    }
}
```

Objectif 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
```

Nous ne voulons vraiment que vouloir présenter notre nuance à travers une transition interactive, mais à cause de la façon dont `UIViewControllerTransitioningDelegate` fonctionne si nous ne retournons pas un contrôleur d'animation régulier, notre contrôleur interactif ne sera jamais utilisé. Pour cette raison, nous créons une classe `EmptyAnimator` conforme à `UIViewControllerAnimatedTransitioning` .

Rapide

```
class EmptyAnimator: NSObject
{

}

extension EmptyAnimator: UIViewControllerAnimatedTransitioning
{
```

```
    func animateTransition(using transitionContext: UIViewControllerContextTransitioning)
    {

    }

    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval
    {
        return 0.0
    }
}
```

Objectif c

```
@implementation EmptyAnimator

- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{

}

-
(NSTimeInterval)transitionDuration:(id<UIViewControllerContextTransitioning>)transitionContext
{
    return 0.0;
}

@end
```

Enfin, nous devons créer le `ShadeAnimator` qui est une sous-classe de `UIDynamicBehavior` conforme à `UIViewControllerInteractiveTransitioning` .

Rapide

```
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)
    }
}
```

Objectif 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)];
```

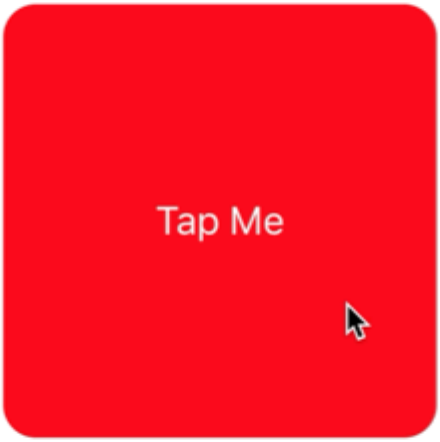
```
    }
    else if (sender.state == UIGestureRecognizerStateEnded || sender.state ==
UIGestureRecognizerStateCancelled)
    {
        UIView *view = self.isAppearing ? toVC.view : fromVC.view;
        BOOL isCancelled = self.isAppearing ? (velocity.y < 0.5 || view.center.y < 0.0) :
(velocity.y > 0.5 || 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'animateur déclenche le début de la transition lorsque le geste panoramique commence. Et déplace simplement la vue au fur et à mesure que le geste change. Mais lorsque le geste se termine, `UIDynamicBehaviors` détermine si la transition doit être terminée ou annulée. Pour ce faire, il utilise un comportement d'attachement et de collision. Pour plus d'informations, consultez la session 2013 de WWDC "Techniques avancées avec UIKit Dynamics .

Mapper la position de l'animation dynamique sur les limites

Cet exemple montre comment personnaliser le protocole `UIDynamicItem` pour mapper les modifications de position d'une vue animée dynamiquement aux modifications de limites pour créer un `UIButton` qui se développe et se contracte de manière élastique.



Pour commencer, nous devons créer un nouveau protocole qui implémente `UIDynamicItem` mais qui possède également une propriété de `bounds` configurables et personnalisables.

Rapide

```
protocol ResizableDynamicItem: UIDynamicItem
{
    var bounds: CGRect { set get }
}
extension UIView: ResizableDynamicItem {}
```

Objectif c

```
@protocol ResizableDynamicItem <UIDynamicItem>
@property (nonatomic, readwrite) CGRect bounds;
@end
```

Nous allons ensuite créer un objet wrapper qui `UIDynamicItem` un `UIDynamicItem` mais qui `UIDynamicItem` modifications du centre sur la largeur et la hauteur de l'élément. Nous allons également fournir des relais pour les `bounds` et la `transform` de l'élément sous-jacent. Cela entraînera toute modification apportée par l'animateur dynamique aux valeurs centrales x et y de l'élément sous-jacent sera appliquée à la largeur et à la hauteur des éléments.

Rapide

```
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
        }
    }
}
```

Objectif 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
```

Enfin, nous allons créer un `UIViewController` qui aura un bouton. Lorsque le bouton est pressé, nous allons créer `PositionToBoundsMapping` avec le bouton comme élément dynamique enveloppé. Nous créons un `UIAttachmentBehavior` à sa position actuelle puis y ajoutons un `UIPushBehavior` instantané. Cependant, parce que nous avons mappé change ses limites, le bouton ne bouge pas mais plutôt grandit et rétrécit.

Rapide

```
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
    }
}
```

Objectif 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
```

Pour plus d'informations, voir Catalogue dynamique UIKit

Lire Dynamique UIKit en ligne: https://riptutorial.com/fr/ios/topic/9479/dynamique-uikit

Chapitre 62: Emplacement central

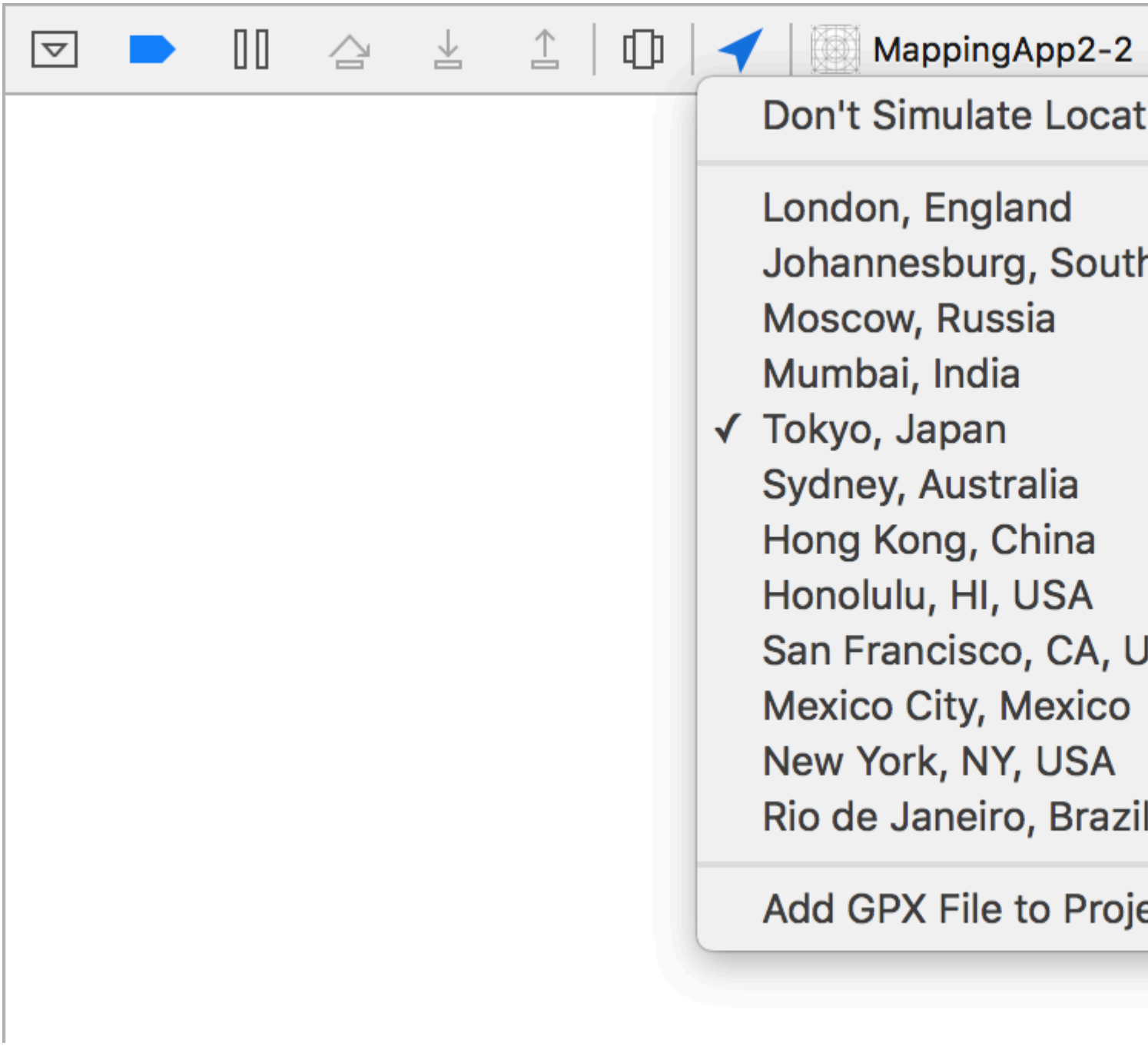
Syntaxe

1. Exactitude désirée
2. distanceFilter
3. requestLocation ()
4. startUpdatingLocation ()
5. allowDeferredLocationUpdates (jusqu'àTraveled: timeout :)
6. startMonitoringSignificantLocationChanges ()
7. allowDeferredLocationUpdates (jusqu'àTraveled: timeout :)
8. autoriséAlways
9. authorisedWhenInse
10. locationManager (_: didChangeAuthorization :)

Remarques

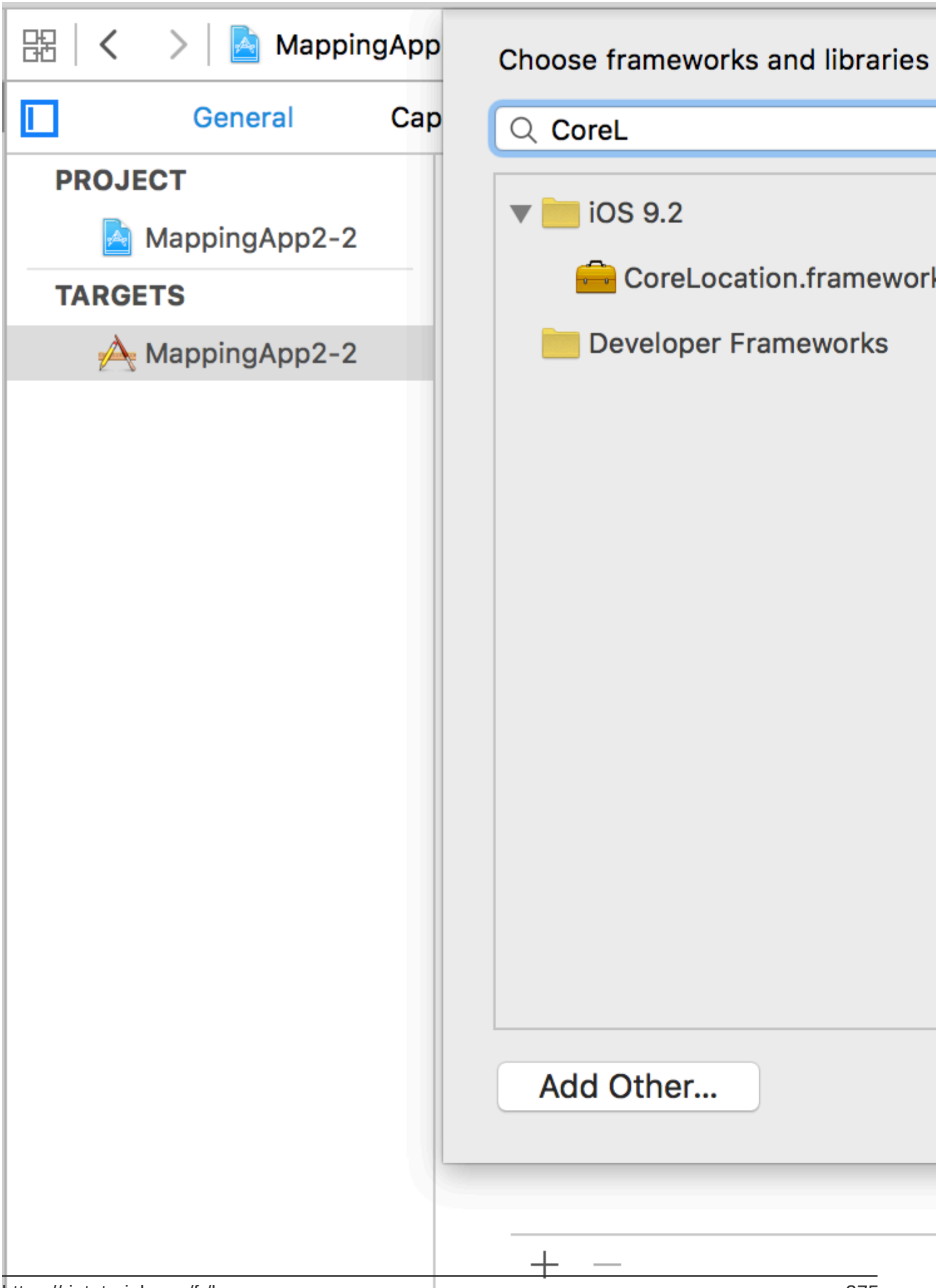
Simuler un emplacement au runtime

1. Exécutez l'application depuis Xcode.
2. Dans la barre de débogage, cliquez sur le bouton "Simuler l'emplacement".
3. Choisissez un emplacement dans le menu.

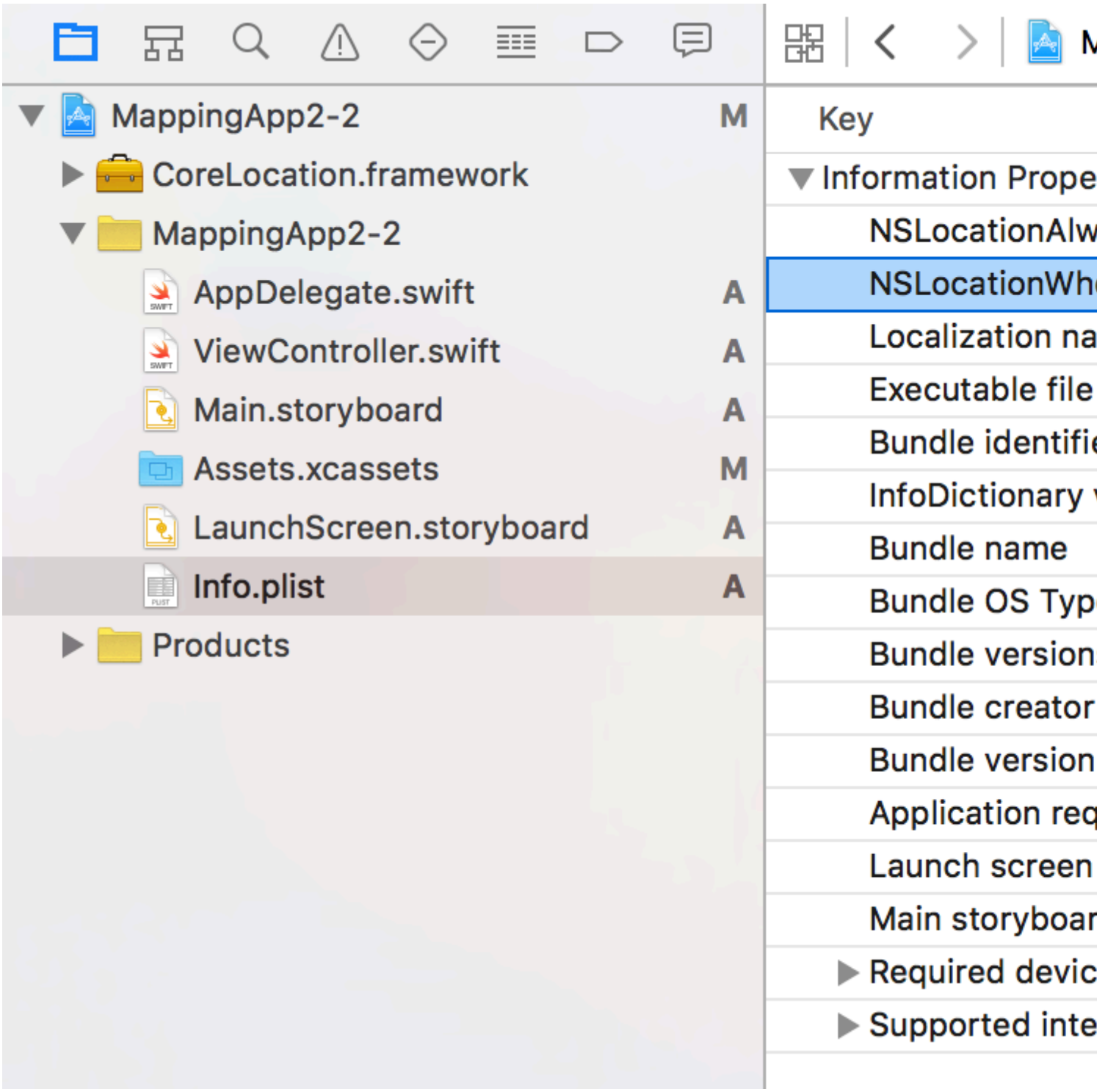


Examples

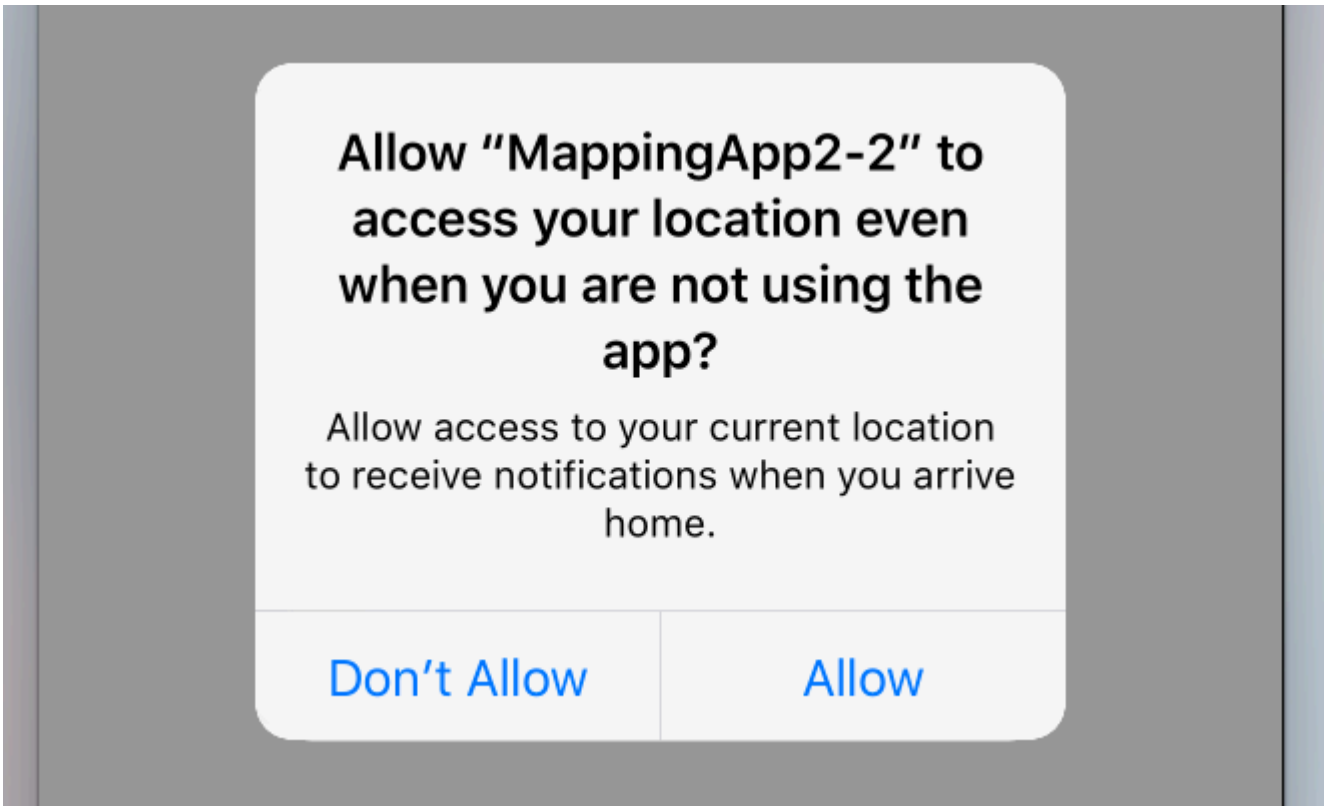
Link CoreLocation Framework


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

. La valeur sera utilisée dans l'étiquette de `message` du contrôleur d'alertes.



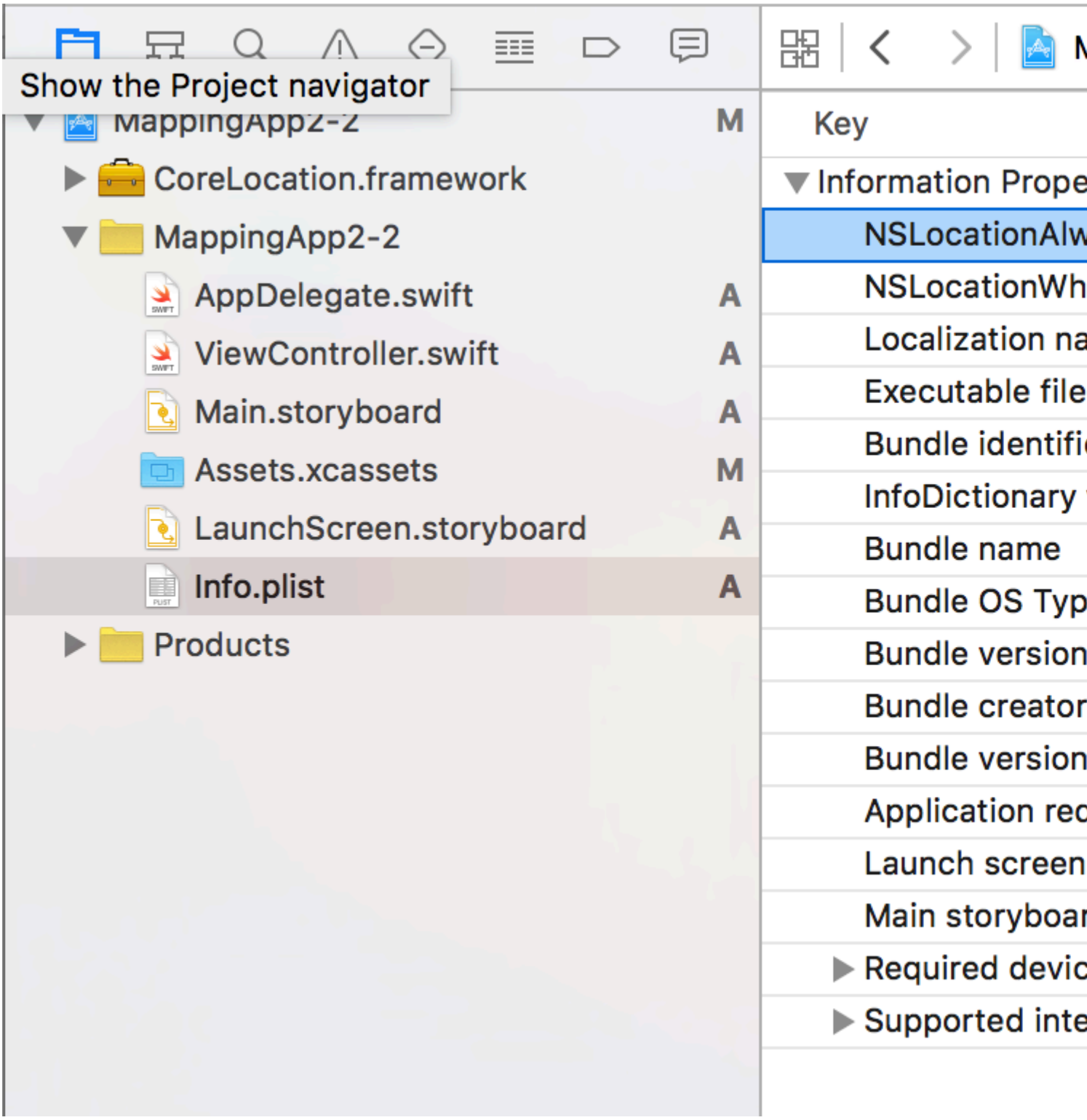
Obtenir toujours l'autorisation de service de localisation



Pour demander l'autorisation d'utiliser les services de localisation même lorsque l'application n'est pas active, utilisez plutôt l'appel suivant:

```
//Swift
locationManager.requestAlwaysAuthorization()

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

Ajoutez ensuite la clé **NSLocationAlwaysUsageDescription** à votre *Info.plist* . Là encore, la valeur sera utilisée dans l'étiquette de `message` du contrôleur d'alertes.



Ajouter un emplacement personnalisé à l'aide du fichier GPX

Pour vérifier les services de localisation, nous avons besoin d'un appareil réel, mais pour des raisons de test, nous pouvons également utiliser un simulateur et ajouter notre propre emplacement en suivant les étapes ci-dessous:

- ajouter un nouveau fichier GPX dans votre projet.
- dans le fichier GPX, ajoutez des waypoints comme

```
<?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>
```

- puis allez à product -> Scheme -> Edit Scheme et dans RUN définissez l'emplacement par défaut comme nom de fichier GPX.

Services de localisation en arrière-plan

Pour utiliser les services de localisation standard pendant que l'application est en arrière-plan, vous devez d'abord activer les `Background Modes` dans l'onglet Capacités des paramètres de la cible, puis sélectionner `Location updates` .

Ou ajoutez-le directement à Info.plist.

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

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

Ensuite, vous devez configurer le CLLocationManager

Objectif 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];
```

Rapide

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

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

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

self.locationManager.startUpdatingLocation()
```

Lire Emplacement central en ligne: https://riptutorial.com/fr/ios/topic/2937/emplacement-central

Chapitre 63: EventKit

Examples

Demande de permission

Votre application ne peut pas accéder à vos rappels et à votre calendrier sans autorisation. Au lieu de cela, il doit afficher une alerte à l'utilisateur, lui demandant d'accorder l'accès aux événements pour l'application.

Pour commencer, importez le framework `EventKit` :

Rapide

```
import EventKit
```

Objectif c

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

Faire un `EKEventStore`

Ensuite, nous `EKEventStore` un objet `EKEventStore` . C'est l'objet à partir duquel nous pouvons accéder aux données du calendrier et des rappels:

Rapide

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

Objectif c

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

Remarque

Faire un objet `EKEventStore` chaque fois que nous avons besoin d'accéder au calendrier n'est pas efficace. Essayez de le faire une fois et utilisez-le partout dans votre code.

Vérifier les disponibilités

La disponibilité a trois statuts différents: Autorisé, Refusé et Non déterminé. Non déterminé signifie que l'application doit accorder l'accès.

Pour vérifier la disponibilité, nous utilisons la méthode `authorizationStatusForEntityType()` de l'objet `EKEventStore` :

Rapide

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

Objectif c

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

Demande de permission

Placez le code suivant dans le cas `NotDetermined` :

Rapide

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

Accéder à différents types de calendriers

Accéder au tableau des calendriers

Pour accéder au tableau de `EKCalendar` , nous utilisons la méthode `calendarsForEntityType` :

Rapide

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

Itérer à travers les calendriers

Utilisez simplement un simple `for` boucle:

Rapide

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

Accéder au titre et à la couleur du calendrier

Rapide

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

Objectif c

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

Ajouter un événement

Création de l'objet événement

Rapide

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

Objectif c

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

Définition du calendrier, du titre et des dates associés

Rapide

```
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
```

Ajout d'événement au calendrier

Rapide

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

Objectif c

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

Lire EventKit en ligne: https://riptutorial.com/fr/ios/topic/5854/eventkit

Chapitre 64: Extension pour notification Push enrichie - iOS 10.

Introduction

iOS 10 nous a donné `UserNotifications.framework` , la nouvelle API pour les notifications locales / distantes. Il offre l'affichage des pièces jointes aux médias ou la réponse aux messages directement depuis la notification.

Le contenu de la notification est constitué de: titre, sous-titre, corps et pièce jointe. La pièce jointe peut contenir des images / gifs / vidéos jusqu'à 50 Mo.

Examples

Extension de contenu de notification

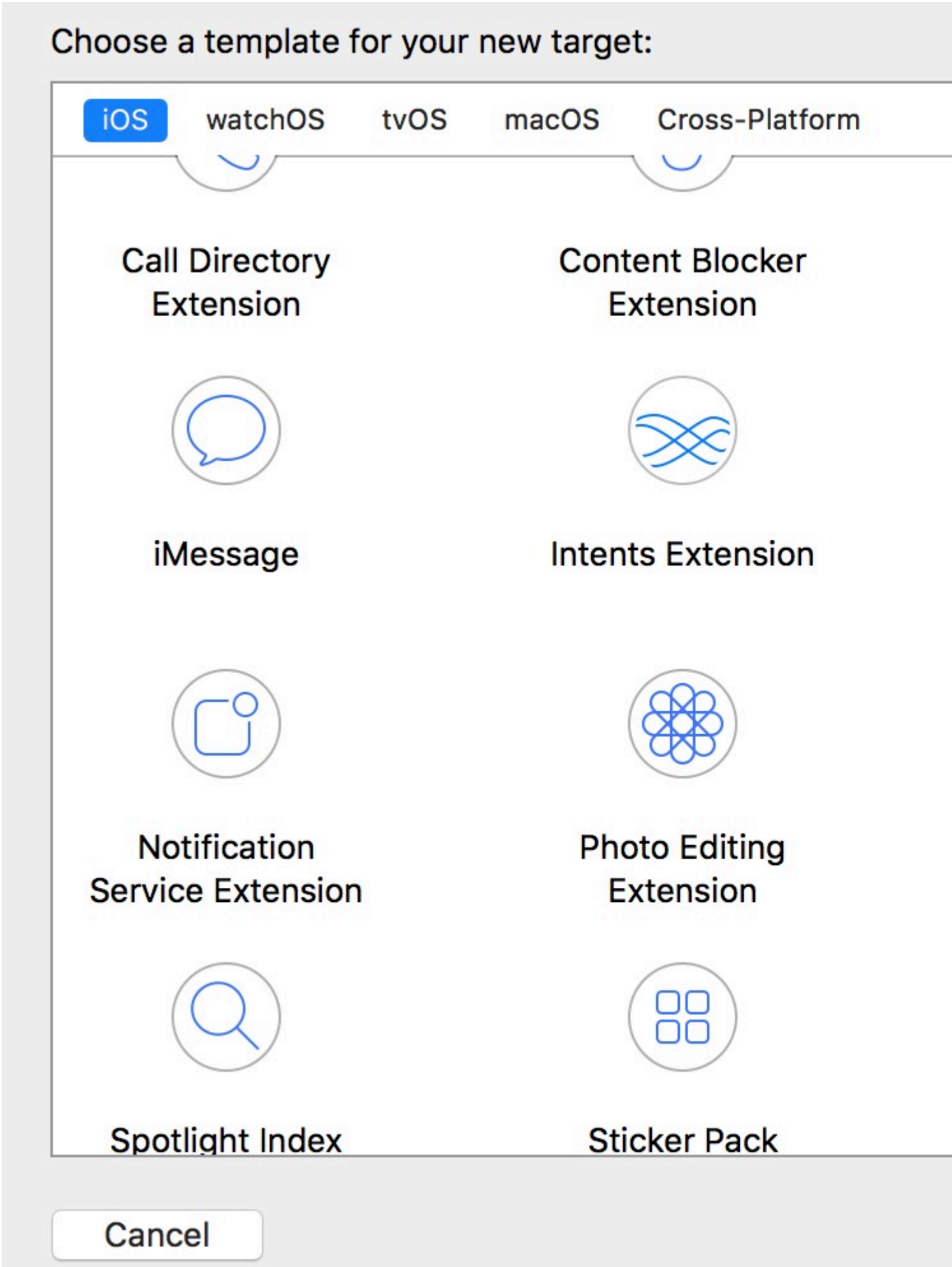
Pourquoi en avons-nous besoin?

L'extension de contenu nous aide à créer une interface utilisateur personnalisée lors de la diffusion des notifications.

Vous utilisez ce cadre pour définir une extension qui reçoit les données de notification et fournit la représentation visuelle correspondante. Votre extension peut également répondre aux actions personnalisées associées à ces notifications.

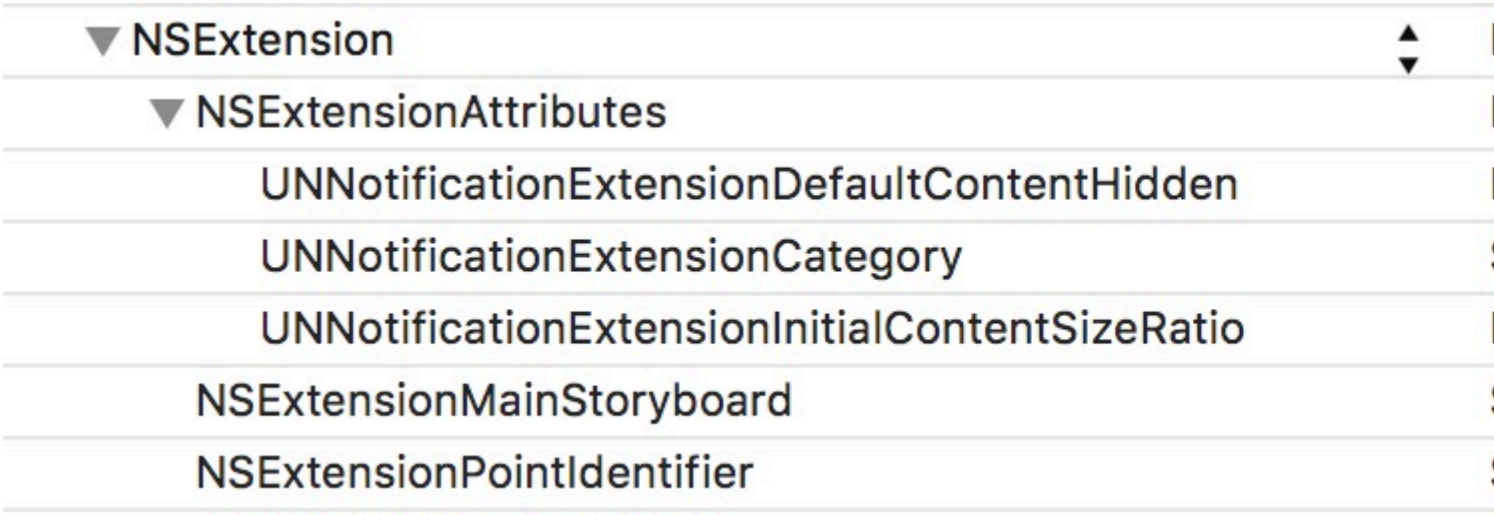
la mise en oeuvre

1. Dans la fenêtre xCode `Navigator` , accédez à la section `Targets` . Appuyez sur `Add New Target` .
2. Sélectionnez le modèle d' `Notification Content Extension` :



3. Dans votre fichier `info.plist` , définissez l'identificateur de la clé

`UNNotificationExtensionCategory` :



NSExtensionAttributes :

`UNNotificationExtensionCategory` (Obligatoire)

La valeur de cette clé est une chaîne ou un tableau de chaînes. Chaque chaîne contient l'identificateur d'une catégorie déclarée par l'application à l'aide de la classe UNNotification Category.

`UNNotificationExtensionInitialContentSizeRatio` (Obligatoire)

Nombre représentant la taille initiale de la vue de votre contrôleur de vue exprimée sous la forme d'un rapport entre sa hauteur et sa largeur.

`UNNotificationExtensionDefaultContentHidden` (Facultatif)

Lorsqu'il est défini sur YES, le système affiche uniquement votre contrôleur d'affichage personnalisé dans l'interface de notification. Lorsqu'il est défini sur NO, le système affiche le contenu de la notification par défaut en plus du contenu de votre contrôleur de vue.

`UNNotificationExtensionOverridesDefaultTitle` (Facultatif)

La valeur de cette clé est une valeur booléenne. Lorsqu'il est défini sur true, le système utilise la propriété title de votre contrôleur de vue comme titre de la notification. Lorsqu'il est défini sur false, le système définit le titre de la notification sur le nom de votre application. Si vous ne spécifiez pas cette clé, la valeur par défaut est définie sur false.

4. Créer une vue personnalisée dans le fichier `NotificationViewController.swift`
5. Ajoutez une nouvelle `category key` et définissez sa valeur en fonction de ce que nous avons tapé dans Info.plist (étape 3):

Pousser:

```
{
 aps: {
```

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

Local:

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

Consultez également la référence officielle de l'API:
https://developer.apple.com/reference/usernotificationsui/unnotificationcontentextension?utm_source=swi

Lire Extension pour notification Push enrichie - iOS 10. en ligne:
https://riptutorial.com/fr/ios/topic/9501/extension-pour-notification-push-enrichie---ios-10-

Chapitre 65: FacebookSDK

Examples

Intégration de FacebookSDK

Étape 1: Installez le SDK

Vous pouvez installer le SDK manuellement ou via `CocoaPods` . Cette dernière option est fortement recommandée.

Mettez ces lignes dans `Podfile` :

```
target 'MyApp' do
  use_frameworks!

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

Exécutez `pod install` dans le terminal et ouvrez ensuite `.xcworkspace` au lieu de `.xcodeproj` .

`FBSDKLoginKit` et `FBSDKShareKit` sont facultatifs. Vous pouvez ou non en avoir besoin.

Étape 2: créer une application sur Facebook

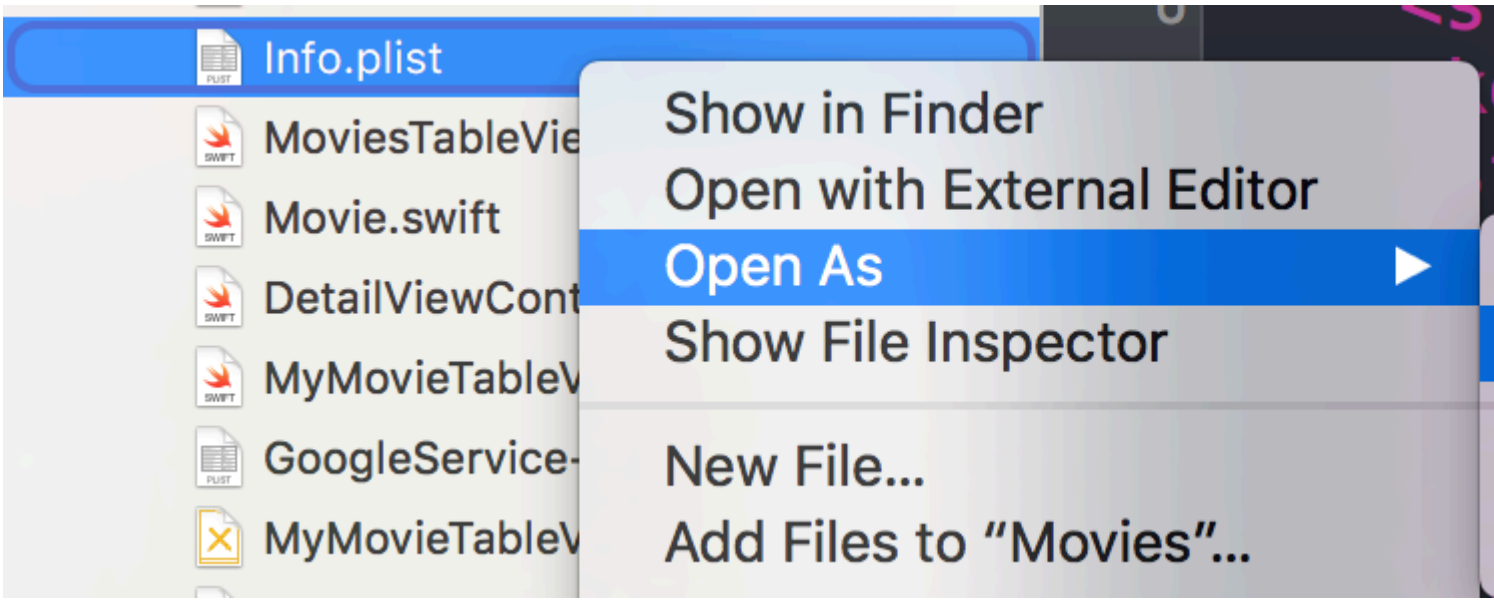
Accédez à Démarrages rapides - Facebook pour les développeurs pour créer une application.

Facebook vous demandera de télécharger le SDK après avoir créé l'application. Vous pouvez ignorer cette partie si vous avez déjà installé le SDK via CocoaPods.

Étape 3: Modifier `.plist`

une. Pour que votre application puisse "communiquer" avec Facebook, vous devez définir des paramètres dans votre fichier `.plist` . Facebook vous donnera l'extrait personnalisé sur la page Démarrages rapides.

b. Modifiez votre fichier `.plist` en tant que code source.



c. Collez votre extrait de code personnalisé dans le code source. **Faites attention!** L'extrait de code doit être exactement l'enfant de la `<dict>` . Votre code source devrait être quelque chose comme:

```
<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>
```

Si vous collez l'extrait de code au mauvais endroit, vous rencontrerez des problèmes.

Étape 4: Indiquez à Facebook l'identifiant de votre bundle sur la page Quick Starts.

=> Comment obtenir l'identifiant du bundle

Étape 5: Modifiez votre `AppDelegate.swift`

une.

```
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)
}
```

Créer votre propre bouton "Connexion avec Facebook" personnalisé

Parfois, nous voulons concevoir notre propre interface utilisateur pour le bouton "Sign In With Facebook" au lieu du bouton d'origine fourni avec FacebookSDK.

1. Dans votre storyboard, faites glisser votre UIButton et réglez-le comme vous le souhaitez.
2. Ctrl + faites glisser votre bouton sur votre contrôleur de vue en tant que IBAction.
3. **Dans** la méthode IBAction, vous simulerez un clic sur le bouton Facebook actuel comme suit:

Rapide:

```
    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)
```

Objectif 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];
```

Vous avez terminé.

Récupérer les données utilisateur facebook

Une fois que l'utilisateur s'est connecté à Facebook sur votre application, il est temps de récupérer les données demandées dans `FBButton.readPermissions` .

Rapide:

```
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
                        }
                    }
                }
            }
        }
    })
}
```

Lire FacebookSDK en ligne: https://riptutorial.com/fr/ios/topic/2972/facebooksdk

Chapitre 66: Fichier texte de base I / O

Examples

Lire et écrire à partir du dossier Documents

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)")
```

Lire Fichier texte de base I / O en ligne: https://riptutorial.com/fr/ios/topic/8892/fichier-texte-de-base-i---o

Chapitre 67: FileHandle

Introduction

Lire le fichier en morceaux à partir du répertoire du document

Examples

Lire le fichier du répertoire de documents en morceaux

J'obtiens le chemin du fichier depuis le répertoire du document et je lis ce fichier en morceaux de 1024 et enregistre (ajoute) dans l'objet `NSMutableData` ou vous pouvez directement écrire dans le 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)")
        }
    }
}
```

Après la lecture complète du fichier , vous obtenez des données de fichiers dans `ReadData` variables Ici outputFileHandle est un objet de `FileHandle`

```
var outputFileHandle:FileHandle?
```

Lire FileHandle en ligne: https://riptutorial.com/fr/ios/topic/10665/filehandle

Chapitre 68: Filtres CoreImage

Examples

Exemple de filtre d'image central

Objectif c

Il suffit de se connecter pour voir comment utiliser un filtre particulier

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

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

Dans le cas de CISepiaTone, le journal système est le suivant

```
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;
    };
}
```

En utilisant le journal du système ci-dessus, nous configurons le filtre comme suit:

```
    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);
```

Image générée par le code ci-dessus

Une autre manière de configurer un filtre

```
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];
```

Image générée à partir de ce code

Tous les filtres disponibles sont comme ci-dessous

```
/* 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*/
```

Lire Filtres CoreImage en ligne: https://riptutorial.com/fr/ios/topic/7278/filtres-coreimage

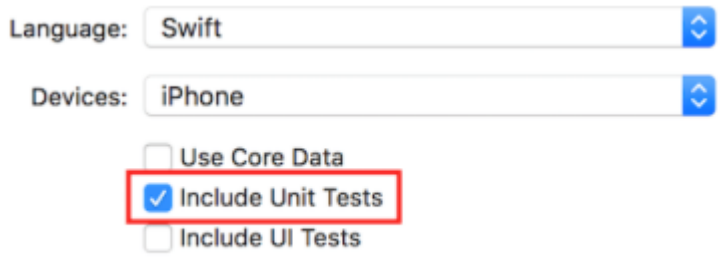
Chapitre 69: Framework XCTest - Tests unitaires

Examples

Ajout de fichiers de test à un projet Xcode

Lors de la création du projet

Vous devez cocher "Inclure les tests unitaires" dans la boîte de dialogue de création de projet.



Après avoir créé le projet

Si vous avez manqué de vérifier cet élément lors de la création de votre projet, vous pouvez toujours ajouter des fichiers de test ultérieurement. Faire cela:

1- Accédez à vos paramètres de projet dans Xcode

2- Aller à "Cibles"

3- Cliquez sur "Ajouter une cible"

4- Sous "Autre", sélectionnez "Paquet de test de test Cocoa Touch Unit"

À la fin, vous devriez avoir un fichier nommé `[Your app name]Tests.swift` . En Objective-C, vous devez avoir deux fichiers nommés `[Your app name]Tests.h` et `[Your app name]Tests.m` place.

`[Your app name]Tests.swift or .m` fichier `[Your app name]Tests.swift or .m` inclura par défaut:

- Une importation de module `XCTest`
- Une classe de `[Your app name]Tests` qui étend `XCTestCase`
- le `tearDown` `testExample` `testPerformanceExample` `setUp` d' `setUp` , `tearDown` , `testExample` , `testPerformanceExample` méthodes

Rapide

```
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.
    }
}

}
```

Objectif 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
```

Ajout de storyboard et de View Controller en tant qu'instances pour tester un fichier

Pour commencer avec les tests unitaires, qui seront effectués dans le fichier de tests et testeront View Controller et Storyboard, nous devrions introduire ces deux fichiers dans le fichier de test.

Définition du contrôleur de vue

Rapide

```
var viewController : ViewController!
```

Présentation du storyboard et initialisation du View Controller

Ajoutez ce code à la méthode `setUp()` :

Rapide

```
let storyboard = UIStoryboard(name: "Main", bundle: nil)
viewController = storyboard.instantiateInitialViewController() as! ViewController
```

Objectif c

```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:"Main" bundle:nil];
viewController = (ViewController *) [storyboard instantiateInitialViewController];
```

De cette façon, vous pouvez écrire des méthodes de test et ils sauront où rechercher les erreurs. Dans ce cas, il y a View Controller et le Storyboard.

Ajouter des méthodes de test

Selon Apple:

Méthodes d'essai

> Une méthode de test est une méthode d'instance d'une classe de test qui commence par le test de préfixe, ne prend aucun paramètre et renvoie un résultat nul, par exemple (void) testColorIsRed (). Une méthode de test exerce du code dans votre projet et, si ce code ne produit pas le résultat attendu, signale les échecs à l'aide d'un ensemble d'API d'assertion. Par exemple, la valeur de retour d'une fonction peut être comparée à une valeur attendue ou votre test peut affirmer qu'une utilisation incorrecte d'une méthode dans l'une de vos classes génère une exception.

Nous ajoutons donc une méthode de test utilisant "test" comme préfixe de la méthode, comme:

Rapide

```
func testSomething() {

}
```

Objectif c

```
- (void)testSomething {

}
```

Pour tester réellement les résultats, nous utilisons la méthode `XCTAssert()` , qui prend une expression booléenne et, si elle est vraie, marque le test comme réussi, sinon elle le marquera comme ayant échoué.

Disons que nous avons une méthode dans la classe View Controller appelée `sum()` qui calcule la somme de deux nombres. Pour le tester, nous utilisons cette méthode:

Rapide

```
func testSum(){
    let result = viewController.sum(4, and: 5)
    XCTAssertEqual(result, 9)
}
```

Objectif c

```
- (void)testSum {
    int result = [viewController sum:4 and:5];
    XCTAssertEqual(result, 9);
}
```

Remarque

Par défaut, vous ne pouvez pas accéder à l'étiquette, à la zone de texte ou à d'autres éléments d'interface utilisateur de la classe View Controller à partir de la classe de test s'ils ont été créés pour la première fois dans le fichier Storyboard. En effet, ils sont initialisés dans la méthode `loadView()` de la classe View Controller, et cela ne sera pas appelé lors du test. La meilleure façon d'appeler `loadView()` et toutes les autres méthodes requises est d'accéder à la propriété `view` de notre propriété `viewController` . Vous devez ajouter cette ligne avant de tester les éléments de l'interface utilisateur:

```
XCTAssertNotNil(viewController.view)
```

Lancer le test

Tester une méthode spécifique

Pour tester une méthode spécifique, cliquez sur le carré en regard de la définition de la méthode.

Tester toutes les méthodes

Pour tester toutes les méthodes, cliquez sur le carré situé en regard de la définition de classe.

Voir le résultat du test

S'il y a une coche verte à côté de la définition, le test a réussi.

S'il y a une croix rouge à côté de la définition, le test a échoué.

Lancer tous les tests

```
Product -> Test OR Cmd + U
```

Il exécutera tous les tests de toutes les cibles de test!

Importer un module pouvant être testé

Les classes, les structures, les énumérations et toutes leurs méthodes sont `internal` par défaut. Cela signifie qu'ils ne peuvent être accédés qu'à partir du même module. Les cas de test se trouvent dans une cible différente, ce qui signifie qu'ils se trouvent dans un module différent. Pour pouvoir accéder à la méthode que vous souhaitez tester, vous devez importer le module à tester à l'aide du mot clé `@testable` .

Disons que nous avons un module principal appelé `ToDo` et que nous voulons y écrire des tests. Nous importons ce module comme ceci:

```
@testable import ToDo
```

Toutes les méthodes de test du fichier avec cette instruction d'importation peuvent désormais accéder à toutes `internal` classes `internal` , structures, énumérations et toutes `internal` méthodes `internal` du module `ToDo` .

Vous ne devez jamais ajouter les fichiers avec les éléments que vous souhaitez tester à la cible de test car cela peut entraîner des erreurs de débogage difficiles.

Chargement de la vue et apparence

Afficher le chargement

Dans un test pour un contrôleur de vue, vous voulez parfois déclencher l'exécution de `loadView()` ou `viewDidLoad()` . Cela peut être fait en accédant à la vue. Disons que vous avez une instance de contrôleur de vue dans votre test appelée `sut` (système sous test), alors le code ressemblerait à ceci:

```
XCTAssertNotNil(sut.view)
```

Voir l'apparence

Vous pouvez également déclencher les méthodes `viewWillAppear(_:)` et `viewDidAppear(_:)` en ajoutant le code suivant:

```
sut.beginAppearanceTransition(true, animated: true)
sut.endAppearanceTransition()
```

Ecrire une classe de test

```
import XCTest
@testable import PersonApp

class PersonTests: XCTestCase {
```

```
    func test_completeName() {
        let person = Person(firstName: "Josh", lastName: "Brown")
        XCTAssertEqual(person.completeName(), "Josh Brown")
    }
}
```

Maintenant, discutons de ce qui se passe ici. La ligne d' `import XCTest` nous permettra d'étendre `XCTestCase` et d'utiliser `XCTAssertEqual` (parmi d'autres assertions). En étendant `XCTestCase` et en préfixant notre nom de `test` avec `test` , vous vous assurez que Xcode exécute automatiquement ce test lors de l'exécution des tests du projet (**U** ou **Product** > **Test**). La `@testable import PersonApp` importera notre cible `PersonApp` afin que nous puissions tester et utiliser des classes, comme la `Person` dans notre exemple ci-dessus. Et enfin, notre `XCTAssertEqual` fera en sorte que `person.completeName()` soit égal à la chaîne `"Josh Brown"` .

Lire Framework XCTest - Tests unitaires en ligne:
https://riptutorial.com/fr/ios/topic/5075/framework-xctest---tests-unitaires

Chapitre 70: GameplayKit

Examples

Générer des nombres aléatoires

Bien que `GameplayKit` (introduit avec iOS 9 SDK) concerne l'implémentation de la logique de jeu, il peut également être utilisé pour générer des nombres aléatoires, ce qui est très utile dans les applications et les jeux.

Outre le `GKRandomSource.sharedRandom` utilisé dans les chapitres suivants, il existe trois types supplémentaires de `GKRandomSource` .

- **GKARC4RandomSource** qui utilise l'algorithme ARC4
- **GKLinearCongruentialRandomSource** Ce qui est un moyen rapide mais pas si aléatoire `GKRandomSource`
- **GKMersenneTwisterRandomSource** qui implémente un algorithme MersenneTwister. C'est plus lent mais plus aléatoire.

Dans le chapitre suivant, nous utilisons uniquement la méthode `nextInt()` d'un `GKRandomSource` . En plus de cela, il y a le `nextBool() -> Bool` et le `nextUniform() -> Float`

Génération

Tout d'abord, importez `GameplayKit` :

Rapide

```
import GameplayKit
```

Objectif c

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

Ensuite, pour générer un nombre aléatoire, utilisez ce code:

Rapide

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

Objectif c

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

Remarque

La fonction nextInt (), lorsqu'elle est utilisée sans paramètres, renverra un nombre aléatoire compris entre -2 147 483 648 et 2 147 483 647, y compris eux-mêmes. Nous ne sommes donc pas certains qu'il s'agisse toujours d'un nombre positif ou non nul.

Générer un nombre de 0 à n

Pour ce faire, vous devez donner la méthode n à `nextIntWithUpperBound()` :

Rapide

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

Objectif c

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

Ce code nous donnera un numéro entre 0 et 10, y compris eux-mêmes.

Générer un nombre de m à n

Pour ce faire, créez un objet `GKRandomDistribution` avec un `GKRandomSource` et transmettez-le dans les limites. Un `GKRandomDistribution` peut être utilisé pour modifier le comportement de distribution comme `GKGaussianDistribution` ou `GKShuffledDistribution` .

Après cela, l'objet peut être utilisé comme tous les `GKRandomSource` ordinaires, car il implémente également le protocole `GKRandom` .

Rapide

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

Objective-C *obsolète*

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

Par exemple, pour générer un nombre aléatoire compris entre 3 et 10, vous utilisez ce code:

Rapide

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

Objective-C *obsolète*

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

GKEntity et GKComponent

Une entité représente un objet d'un jeu comme une figure de joueur ou une figure ennemie. Puisque cet objet ne fait pas grand chose sans les bras et les jambes, nous pouvons y ajouter les composants. Pour créer ce système, Apple a les classes `GKEntity` et `GKComponent` .

Supposons que nous avons la classe suivante pour les chapitres suivants:

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

GKEntity

Une entité est un ensemble de composants et offre plusieurs fonctions pour ajouter, supprimer et interagir avec ses composants.

Bien que nous puissions simplement utiliser GKEntity, il est courant de le sous-classer pour un type spécifique d'entité de jeu.

Il est important de n'ajouter qu'un composant d'une classe. Si vous ajoutez un second composant de la même classe, il remplacera le premier composant existant dans `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
```

Vous pouvez demander pourquoi. La raison en est les méthodes appelées `component(for: T.Type)` qui renvoie le composant d'un type spécifique de l'entité.

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

En plus des méthodes de composants, il dispose d'une méthode de `update` qui permet de déléguer

le delta time ou l'heure actuelle de la logique de jeu à ses composants.

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

GKComponent

Un composant représente quelque chose d'une entité, par exemple le composant visuel ou le composant logique.

Si une méthode de mise à jour d'une entité est appelée, elle la déléguera à tous ses composants. La substitution de cette méthode est utilisée pour manipuler une entité.

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

En plus de cela, il est possible de remplacer la méthode `didAddToEntity` et `willRemoveFromEntity` pour informer les autres composants de sa suppression ou de son ajout.

Pour manipuler un autre composant à l'intérieur d'un composant, il est possible d'obtenir le GKEntity auquel le composant est ajouté.

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

Bien que cela soit possible, ce n'est **pas** un modèle courant car il relie les deux composants.

GKComponentSystem

Alors que nous venons de parler de l'utilisation du mécanisme de `GKEntity` de mise à jour de `GKEntity` pour mettre à jour les `GKComponents` il existe une autre façon de mettre à jour `GKComponents` appelée `GKComponentSystem` .

Il est utilisé dans le cas où il est nécessaire que tous les composants d'un type spécifique doivent être mis à jour en une fois.

Un `GKComponentSystem` est créé pour un type de composant spécifique.

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

Pour ajouter un composant, vous pouvez utiliser la méthode add:

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

Mais une manière plus courante est de passer l'entité créée avec ses composants au `GKComponentSystem` et elle trouvera un composant correspondant à l'intérieur de l'entité.

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

Pour mettre à jour tous les composants d'un type spécifique, appelez la mise à jour:

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

Si vous souhaitez utiliser `GKComponentSystem` au lieu d'un mécanisme de mise à jour basé sur une entité, vous devez avoir un `GKComponentSystem` pour chaque composant et appeler la mise à jour sur tous les systèmes.

Lire GameplayKit en ligne: https://riptutorial.com/fr/ios/topic/4966/gameplaykit

Chapitre 71: GCD (Expédition Central Grand)

Introduction

Grand Central Dispatch (GCD) est la réponse d'Apple au multithreading. Il s'agit d'une infrastructure légère permettant d'effectuer des tâches de manière synchrone ou asynchrone dans les files d'attente et de gérer les threads de CPU pour vous dans les coulisses.

Rubrique connexe: Concurrence

Examples

Créer une file d'attente de distribution

Vous pouvez créer votre propre file d'attente en utilisant `dispatch_queue_create`

Objectif c

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

Rapide

```
// 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
```

Obtenir la file d'attente principale

La file d'attente principale est la file d'attente de distribution dans laquelle toutes les mises à jour de l'interface utilisateur ont lieu et le code impliquant les modifications de l'interface utilisateur est placé.

Vous devez accéder à la file d'attente principale afin de mettre à jour l'interface utilisateur à la fin d'un processus asynchrone tel que `NSURLSession`

Il existe deux types d'appels de file d'attente principale `synchronous` et `asynchronous` . Lorsque vous appelez quelque chose de manière `synchronously` , cela signifie que le thread qui a lancé cette opération attend que la tâche se termine avant de continuer. `Asynchronous` signifie qu'il n'attendra pas.

Code Objective-C

Appel de la file d'attente principale `Synchronous`

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

Appel de file d'attente principale `Asynchronous`

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

SWIFT 3

Appel de file d'attente principale `Asynchronous`

```
DispatchQueue.main.async {

}
```

Appel de la file d'attente principale `Synchronous`

```
DispatchQueue.main.sync {

}
```

Groupe d'expédition

> DispatchGroup permet la synchronisation globale du travail. Vous pouvez les utiliser pour soumettre plusieurs éléments de travail différents et effectuer un suivi lorsqu'ils sont tous terminés, même s'ils peuvent s'exécuter sur des files d'attente différentes. Ce comportement peut être utile lorsque la progression ne peut pas être effectuée tant que toutes les tâches spécifiées ne sont pas terminées.

Un scénario, si cela peut être utile, si vous avez plusieurs appels de services Web qui doivent tous être terminés avant de continuer. Par exemple, vous devez télécharger plusieurs ensembles de données devant être traités par une fonction. Vous devez attendre que tous les services Web soient terminés avant d'appeler la fonction pour traiter toutes les données reçues.

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!")
}
```

Sinon, si vous ne souhaitez pas attendre la fin des groupes, mais que vous souhaitez exécuter une fonction une fois toutes les tâches terminées, utilisez la fonction `notify` à la place du `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
}
```

Exemple de sortie:

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

Pour plus d'informations, reportez-vous à la documentation Apple ou à la rubrique associée.

Sémaphore d'expédition

> DispatchSemaphore fournit une implémentation efficace d'un sémaphore de comptage traditionnel, qui peut être utilisé pour contrôler l'accès à une ressource dans plusieurs contextes d'exécution.

Un scénario pour savoir quand utiliser un sémaphore peut être si vous faites de la lecture / écriture de fichiers, si plusieurs tâches essaient de lire et d'écrire en même temps, cela peut augmenter vos performances pour que chaque tâche attende son tour. pour ne pas surcharger le contrôleur d'E / S.

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
        }
    }
}
```

Exemple de sortie: (notez les horodatages)

```
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
```

Pour plus d'informations, consultez le document Apple Docs

Files d'attente de répartition série / simultanée

Swift 3

File d'attente série

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

Exemple de sortie:

```
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
```

File d'attente simultanée

```
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)")
        }
    }
```

```
}
```

Exemple de sortie:

```
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
```

Discussion

Comme nous pouvons le voir dans les exemples ci-dessus, une file d'attente série complète chaque tâche dans l'ordre dans lequel elles sont soumises à la file d'attente. Chaque tâche attend que la tâche précédente se termine avant de l'exécuter. En ce qui concerne la file d'attente concurrente, chaque tâche n'attend pas les autres en attente et s'exécute dès que possible. L'avantage est que toutes les tâches de la file d'attente s'exécuteront en même temps sur des threads distincts, ce qui fait qu'une file d'attente simultanée prend moins de temps qu'une file d'attente en série.

Si l'ordre d'exécution des tâches n'est pas important, utilisez toujours une file d'attente simultanée pour optimiser l'efficacité.

Lire GCD (Expédition Central Grand) en ligne: https://riptutorial.com/fr/ios/topic/4626/gcd--expedition-central-grand-

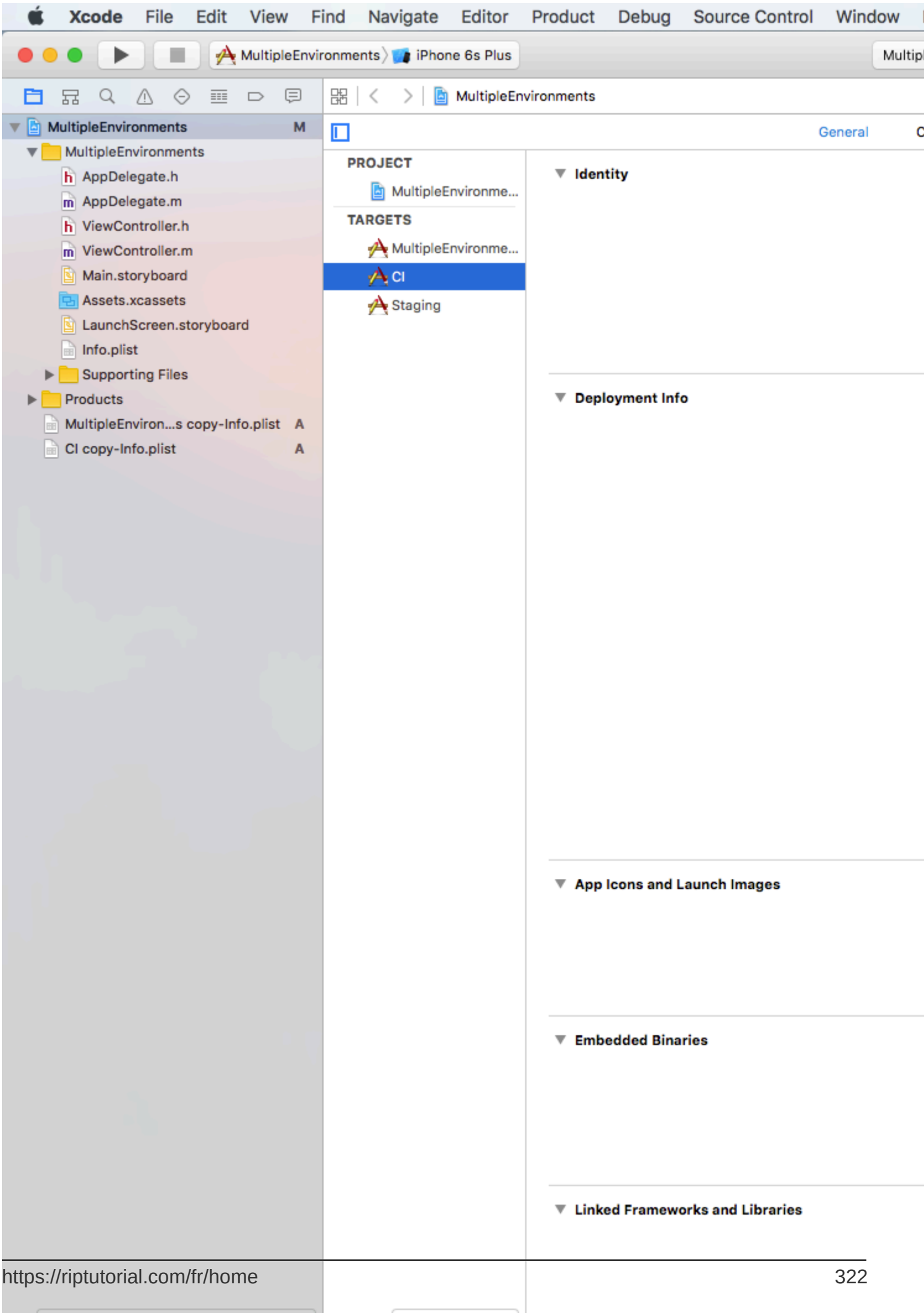
Chapitre 72: Gérer plusieurs environnements en utilisant la macro

Examples

Gérer plusieurs environnements à l'aide de plusieurs cibles et macro

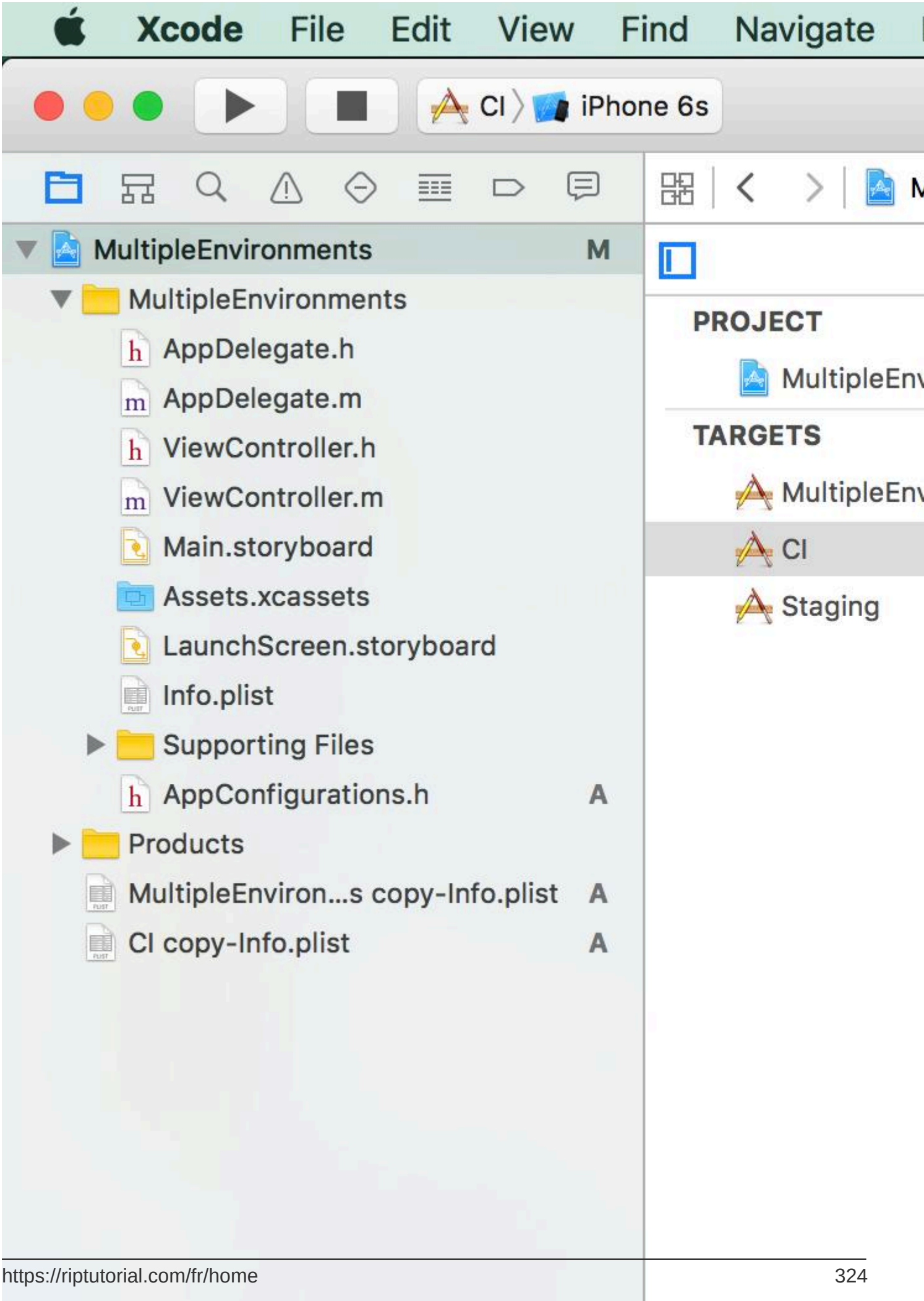
Par exemple, nous avons deux environnements: CI - Staging et vous souhaitez ajouter des personnalisations pour chaque environnement. Ici, je vais essayer de personnaliser l'URL du serveur, le nom de l'application.

Premièrement, nous créons deux cibles pour 2 environnements en dupliquant la cible principale:

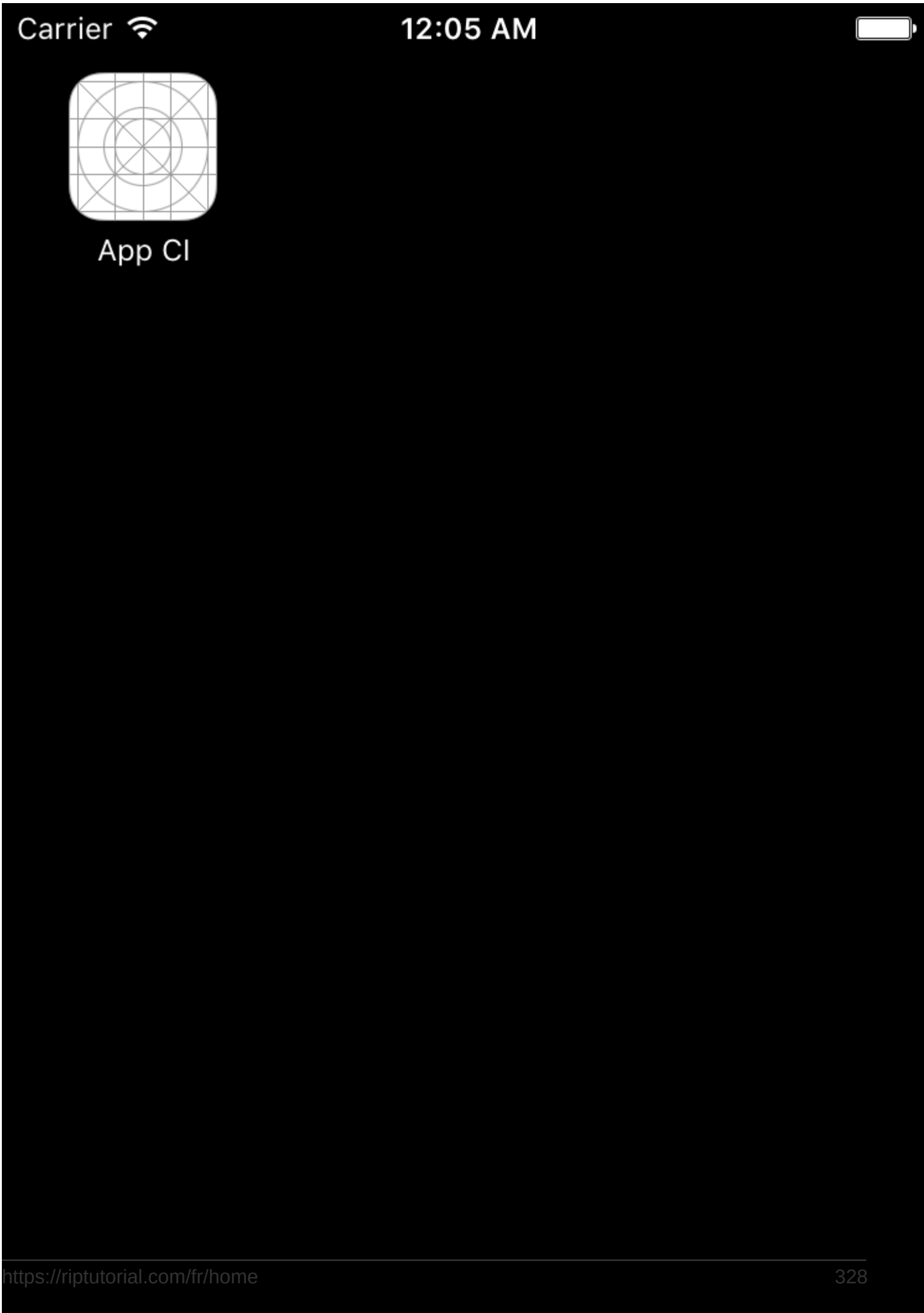

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window
MultipleEnvironments iPhone 6s Plus
MultipleEnvironments
MultipleEnvironments M
MultipleEnvironments
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Supporting Files
Products
MultipleEnviron...s copy-Info.plist A
CI copy-Info.plist A
General
PROJECT
MultipleEnvironme...
TARGETS
MultipleEnvironme...
CI
Staging
Identity
Deployment Info
App Icons and Launch Images
Embedded Binaries
Linked Frameworks and Libraries

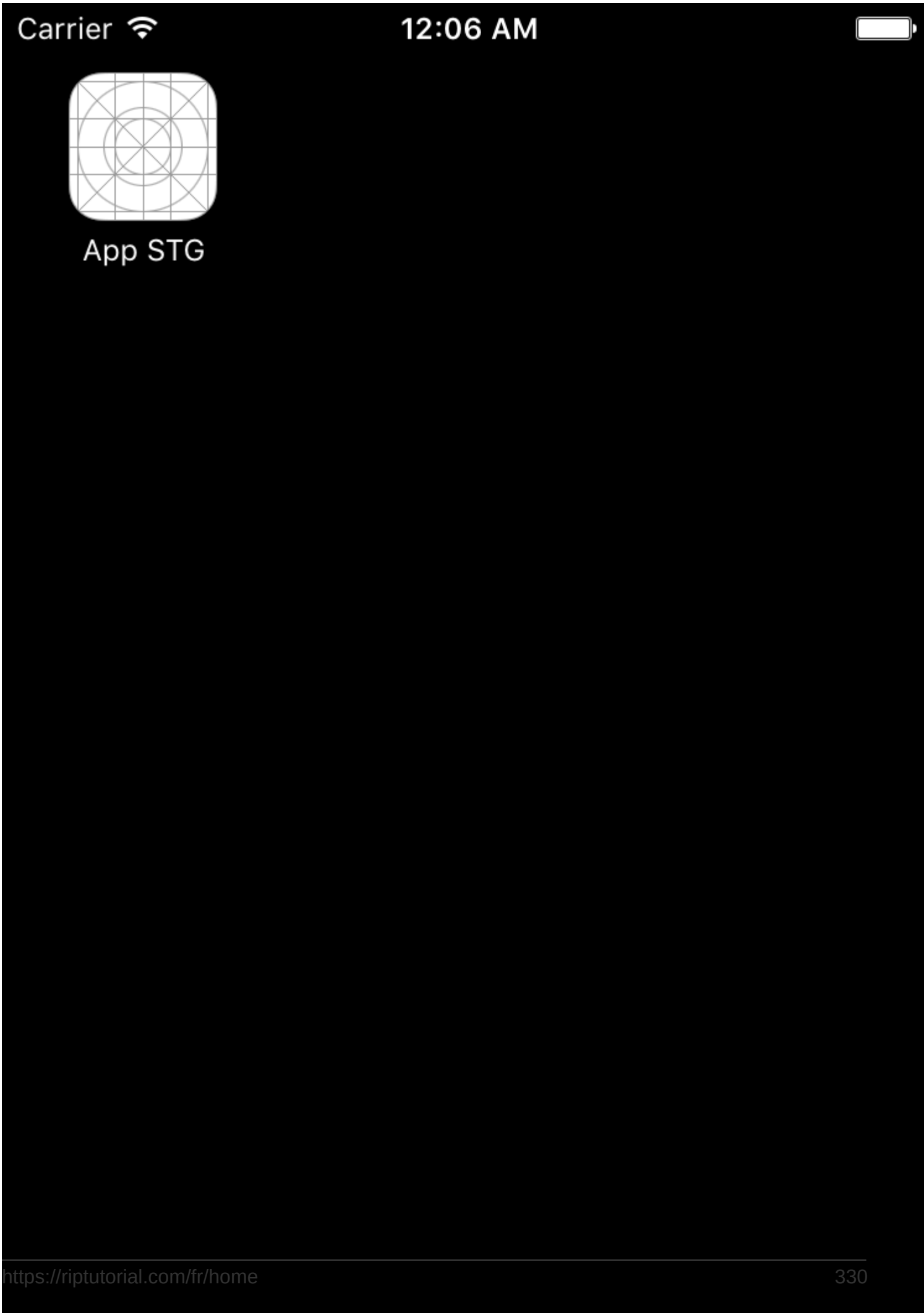
- Si nous exécutons / archivons à l'aide de la cible CI, le SERVER_URL est http://ci.api.example.com/ http://ci.api.example.com/
- Si nous exécutons / archive en utilisant la cible STAGING, le SERVER_URL est http://stg.api.example.com/

Si vous souhaitez personnaliser davantage, par exemple: Modifier le nom de l'application pour chaque cible:


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/fr/ios/topic/6849/gerer-plusieurs-environnements-en-utilisant-la-macro

Chapitre 73: Gestion des schémas d'URL

Syntaxe

1. // **La** méthode **canOpenURL** vérifie s'il existe une application capable de gérer le schéma d'URL indiqué.
2. // Rapide

   UIApplication.sharedApplication (). CanOpenURL (_ aUrl: NSURL)
3. // Objectif c

   [[UIApplication sharedApplication] canOpenURL: (NSURL *) aUrl];
4. // **La** méthode **openURL** tente d'ouvrir une ressource située par URL. OUI / vrai s'il était ouvert sinon NON / faux.
5. // Rapide

   UIApplication.sharedApplication (). OpenURL (_ aUrl: NSURL)
6. // Objectif c

   [[UIApplication sharedApplication] openURL: (NSURL *) aUrl];

Paramètres

Paramètre	Sens
aUrl	une instance NSURL qui stocke une chaîne de schéma intégrée ou personnalisée

Remarques

Dans iOS9 et au-dessus, votre application doit répertorier tous les modèles d'URL à interroger. Cela se fait en ajoutant `LSApplicationQueriesSchemes` à Info.plist

---

iOS a une prise en charge `facetime` schémas `tel` , `http` / `https` , `sms` , `mailto` , `facetime` . Il prend également en charge les URL basées sur http pour les applications `Youtube` , `Maps` et `iTunes` .

Exemples de schémas d'URL intégrés:

tel : `tel://123456890` ou `tel:123456890`

http : `http://www.google.com`

facetime : `facetime://azimov@demo.com`

mailto : `mailto://azimov@demo.com`

sms : `sms://123456890` ou `sms:123456890`

Youtube : `https://www.youtube.com/watch?v=-eCaif2QKfA`

Cartes :

- En utilisant l'adresse: `http://maps.apple.com/?address=1,Infinite+Loop,Cupertino,California`
- Utilisation des coordonnées: `http://maps.apple.com/?ll=46.683155557,6.683155557`

iTunes : `https://itunes.apple.com/us/artist/randy-newman/id200900`

Remarque : Tous les caractères spéciaux ne sont pas pris en charge dans le schéma `tel` (par exemple `*` ou `#`). Cela est dû à des problèmes de sécurité pour empêcher les utilisateurs de rediriger les appels sans autorisation. Dans ce cas, l'application `Phone` ne sera pas ouverte.

Examples

Utilisation du schéma d'URL intégré pour ouvrir l'application Mail

Rapide:

```
if let url = URL(string: "mailto://azimov@demo.com") {
    if UIApplication.shared.canOpenURL(url) {
        UIApplication.shared.openURL(url)
    } else {
        print("Cannot open URL")
    }
}
```

Objectif c:

```
NSURL *url = [NSURL URLWithString:@"mailto://azimov@demo.com"];
if ([[UIApplication sharedApplication] canOpenURL:url]) {
    [[UIApplication sharedApplication] openURL:url];
} else {
    NSLog(@"Cannot open URL");
}
```

Schémas d'URL Apple

Ce sont des schémas d'URL pris en charge par les applications natives sur iOS, OS X et watchOS 2 et versions ultérieures.

Lien d'ouverture dans Safari:

Objectif c

```
NSString *stringURL = @"http://stackoverflow.com/";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Rapide:

```
let stringURL = "http://stackoverflow.com/"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

Commencer une conversation téléphonique

Objectif c

```
NSString *stringURL = @"tel:1-408-555-5555";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Rapide:

```
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>
```

Démarrer une conversation FaceTime

Objectif c

```
NSString *stringURL = @"facetime:14085551234";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Rapide:

```
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>
```

Ouverture de l'application Messages pour composer un sms au destinataire:

Objectif c

```
NSString *stringURL = @"sms:1-408-555-1212";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Rapide:

```
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>
```

Ouvrir l'application Mail pour composer un email au destinataire:

Objectif c

```
NSString *stringURL = @"mailto:foo@example.com";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Rapide:

```
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>
```

Vous pouvez également inclure un champ d'objet, un message et plusieurs destinataires dans les champs À, Cc et Cci. (Dans iOS, l'attribut from est ignoré.) L'exemple suivant montre une URL mailto qui inclut plusieurs attributs différents:

```
mailto:foo@example.com?cc=bar@example.com&subject=Greetings%20from%20Cupertino!&body=Wish%20you%20were
```

Remarque: La boîte de dialogue Compose email peut également être présentée dans l'application

à l'aide de `MFMailComposeViewController` .

Lire Gestion des schémas d'URL en ligne: https://riptutorial.com/fr/ios/topic/3646/gestion-des-schemas-d-url

Chapitre 74: Gestion du clavier

Examples

Défilement d'un UIScrollView / UITableView lors de l'affichage du clavier

Il existe peu d'approches disponibles:

1. Vous pouvez vous abonner aux notifications d'événements d'apparence du clavier et modifier manuellement le décalage:

```
//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. Ou utilisez des solutions prêtes à l'emploi telles que TPKeyboardAvoidingTableView ou TPKeyboardAvoidingScrollView https://github.com/michaeltyson/TPKeyboardAvoiding

Rejeter un clavier avec robinet sur vue

Si vous voulez masquer un clavier en appuyant dessus, il est possible d'utiliser cette astuce (fonctionne uniquement avec 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];
}
```

pour Swift, il y aura un peu plus de code:

```
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 autre exemple 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
    }
}
```

Créer un clavier personnalisé dans l'application

Ceci est un clavier de base dans l'application. La même méthode pourrait être utilisée pour créer à peu près n'importe quelle disposition de clavier. Voici les principales choses à faire:

- Créez la disposition du clavier dans un fichier .xib dont le propriétaire est une classe Swift ou Objective-C qui est une sous-classe `UIView` .
- Dites au `UITextField` d'utiliser le clavier personnalisé.
- Utilisez un délégué pour communiquer entre le clavier et le contrôleur de la vue principale.

Créez le fichier de disposition du clavier .xib

- Dans Xcode, accédez à **Fichier> Nouveau> Fichier ...> iOS> Interface utilisateur> Afficher** pour créer le fichier .xib.
- J'ai appelé le mien Keyboard.xib
- Ajoutez les boutons dont vous avez besoin.
- Utilisez les contraintes de mise en page automatique afin que, quelle que soit la taille du clavier, les boutons seront redimensionnés en conséquence.
- Définissez le propriétaire du fichier (pas la vue racine) comme étant la classe du `Keyboard` . C'est une source d'erreur commune. Vous allez créer cette classe à l'étape suivante. Voir la note à la fin.

Créez le fichier de clavier de la sous-classe .swift UIView

- Dans Xcode, accédez à **Fichier> Nouveau> Fichier ...> iOS> Source> Classe Cocoa Touch** pour créer la **classe** Swift ou Objective-C. Choisissez `UIView` comme superclasse pour la classe nouvellement créée

- J'ai appelé le mien `Keyboard.swift` (classe de `Keyboard` dans Objective-C)

- Ajoutez le code suivant pour 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
        }

    }
```

- Ajoutez le code suivant pour Objective-C:

Fichier Keyboard.h

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

// The view controller will adopt this protocol (delegate)
// and thus must contain the keyWasTapped method
@protocol KeyboardDelegate<NSObject>
- (void)keyWasTapped:(NSString *)character;
@end

@interface Keyboard : UIView
@property (nonatomic, weak) id<KeyboardDelegate> delegate;
@end
```

Fichier de clavier

```
#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
```

- Contrôlez les actions de glisser depuis le bouton vers le bouton dans le fichier .xib vers la méthode `@IBAction` dans le propriétaire Swift ou Objective-C pour les `@IBAction` .
- Notez que le protocole et le code délégué. Voir cette réponse pour une explication simple sur le fonctionnement des délégués.

Configurez le View Controller

- Ajoutez un objet `UITextField` à votre storyboard principal et connectez-le à votre contrôleur de vue avec un `IBOutlet` . Appelez-le `textField` .

- Utilisez le code suivant pour View Controller dans 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)
      }
  }
```

- Utilisez le code suivant pour Objective-C:

.h Fichier

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

@interface ViewController : UIViewController
```

```
@end
```

Fichier .m

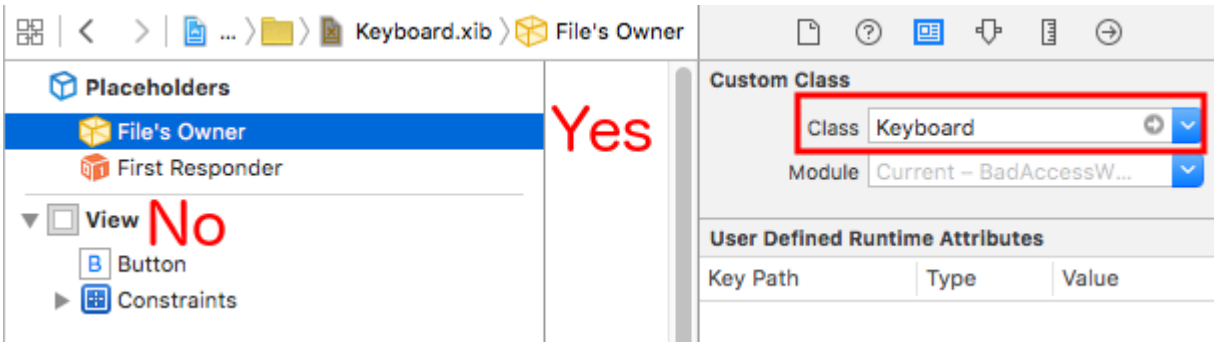
```
#import "ViewController.h"
#import "Keyboard.h"

@interface ViewController ()<KeyboardDelegate>

@property (nonatomic, weak) IBOutlet UITextField *textField;

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // initialize custom keyboard
    Keyboard *keyboardView = [[Keyboard alloc] initWithFrame:CGRectMake(0, 0, 0, 300)];
    keyboardView.delegate = self; // the view controller will be notified by the keyboard
whenever a key is tapped

    // replace system keyboard with custom keyboard
    self.textField.inputView = keyboardView;
}

- (void)keyWasTapped:(NSString *)character {
    [self.textField insertText:character];
}

@end
```

- Notez que le contrôleur de vue adopte le protocole `KeyboardDelegate` que nous avons défini ci-dessus.

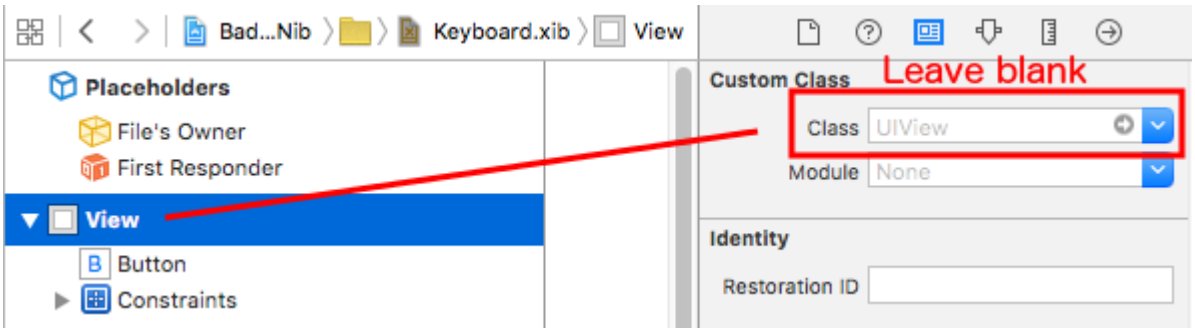
Erreur commune

Si vous obtenez une erreur `EXC_BAD_ACCESS` , c'est probablement parce que vous définissez la classe personnalisée de la vue en tant que `Keyboard` plutôt que de le faire pour le propriétaire du fichier nib.

Sélectionnez `Keyboard.nib` , puis choisissez File's Owner.



Assurez-vous que la classe personnalisée pour la vue racine est vide.



Remarques

Cet exemple provient de cette réponse Stack Overflow .

Gestion du clavier avec un Singleton + délégué

Lorsque j'ai commencé à gérer le clavier, j'utiliserais des notifications séparées dans chaque ViewController.

Méthode de notification (à l'aide de 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)
    }
}
```

Mon problème était que je me retrouvais à écrire ce code encore et encore pour chaque ViewController. Après avoir expérimenté un peu, j'ai découvert en utilisant un pattern Singleton + Delegate que je pouvais réutiliser un tas de code et organiser tout le Keyboard Management en un seul endroit!

Méthode Singleton + Délégué:

```
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)
    }
}
```

Maintenant, quand je veux gérer le clavier à partir d'un ViewController, tout ce que j'ai à faire est de définir le délégué sur ce ViewController et d'implémenter toutes les méthodes de délégué.

```
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)
    }
}
```

Cette méthode est également très personnalisable! Disons que nous voulons ajouter des fonctionnalités pour `UIKeyboardWillHideNotification` . C'est aussi simple que d'ajouter une méthode à notre `KeyboardManagerDelegate` .

`KeyboardManagerDelegate` avec `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)
    }
}
```

Disons que nous voulons seulement implémenter `func keyboardWillHide(notificationUserInfo: [NSObject: AnyObject])` dans un ViewController. Nous pouvons également rendre cette méthode facultative.

```
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])
}
```

* Notez que ce motif permet d'éviter l'utilisation excessive de `@objc` . Voir http://www.jessesquires.com/avoiding-objc-in-swift/ pour plus de détails!

En résumé, j'ai trouvé que l'utilisation d'un Singleton + Delegate pour gérer le clavier était à la fois plus efficace et plus facile à utiliser que l'utilisation des notifications.

Déplacement de la vue vers le haut ou le bas lorsque le clavier est présent

Remarque: Cela ne fonctionne que pour le clavier intégré fourni par iOS

RAPIDE:

Pour que la vue d'un **UIViewController** augmente l'origine du cadre lorsqu'il est présenté et le diminue lorsqu'il est masqué, ajoutez les fonctions suivantes à votre 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
        }
    }
}
```

Et dans la méthode `viewDidLoad()` de votre classe, ajoutez les observateurs suivants:

```
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)
```

Et cela fonctionnera pour n'importe quelle taille d'écran, en utilisant la propriété height du clavier.

OBJECTIF C:

Pour faire la même chose en Objective-C, ce code peut être utilisé:

```
- (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;
    }];
}
```

Lire Gestion du clavier en ligne: https://riptutorial.com/fr/ios/topic/436/gestion-du-clavier

Chapitre 75: Graphique (Coreplot)

Examples

Faire des graphiques avec CorePlot

Core Plot fournit un podspec, vous pouvez donc utiliser les cocoapod comme gestionnaire de bibliothèque, ce qui simplifiera l'installation et la mise à jour

Installez des cocoapodes sur votre système

Dans le répertoire du projet, ajoutez un fichier texte à votre projet appelé Podfile en tapant `pod init` dans votre répertoire de projet.

Dans le Podfile, ajoutez le pod de ligne 'CorePlot', '~> 1.6'

Dans le terminal, accédez au répertoire de votre projet et exécutez pod installation

Cocoapods générera un fichier xcworkspace, que vous devez utiliser pour lancer votre projet (le fichier .xcodeproj n'inclura pas les bibliothèques de pod)

Ouvrez le .xcworkspace généré par CocoaPods

Dans le fichier ViewController.h

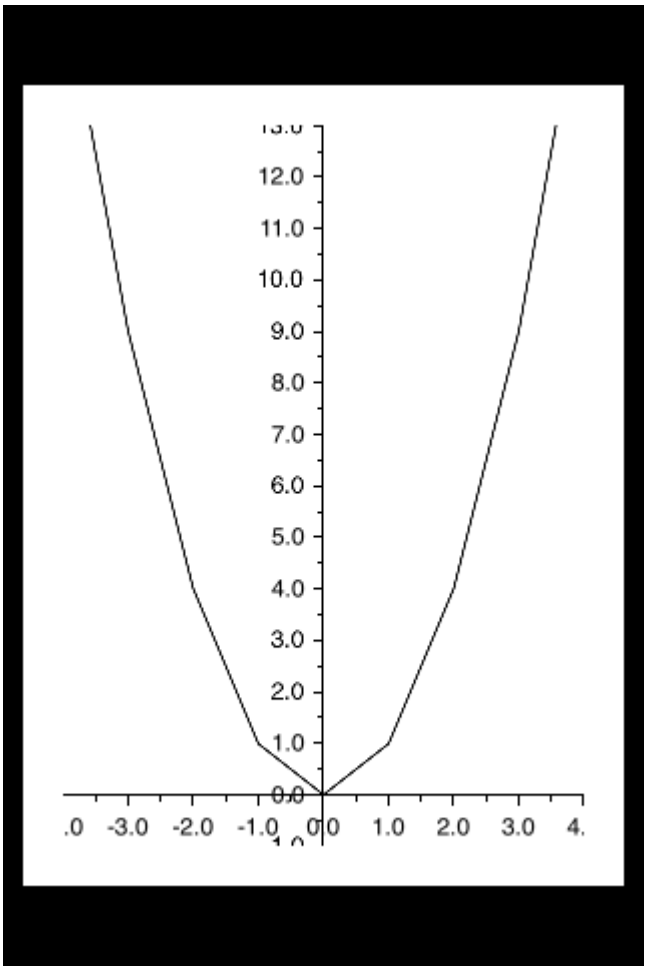
```
#import <CorePlot/ios/CorePlot.h>
//#import "CorePlot-CocoaTouch.h" or the above import statement
@interface ViewController : UIViewController<CPTPlotDataSource>
```

Dans le fichier 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];
    }
}
```

La sortie générée est donnée ci-dessous:



Lire Graphique (Coreplot) en ligne: https://riptutorial.com/fr/ios/topic/7302/graphique--coreplot-

Chapitre 76: Graphiques de base

Examples

Créer un contexte graphique de base

Contexte graphique de base

Un contexte Core Graphics est un canevas que nous pouvons dessiner et définir des propriétés telles que l'épaisseur du trait.

Faire un contexte

Pour créer un contexte, nous utilisons la fonction `UIGraphicsBeginImageContextWithOptions()`. Ensuite, lorsque nous en avons fini avec le dessin, nous appelons simplement `UIGraphicsEndImageContext()` pour terminer le contexte:

Rapide

```
let size = CGSize(width: 256, height: 256)

UIGraphicsBeginImageContextWithOptions(size, false, 0)

let context = UIGraphicsGetCurrentContext()

// drawing code here

UIGraphicsEndImageContext()
```

Objectif c

```
CGSize size = [CGSize width:256 height:256];

UIGraphicsBeginImageContextWithOptions(size, NO, 0);

CGContext *context = UIGraphicsGetCurrentContext();

// drawing code here

UIGraphicsEndImageContext();
```

Dans le code ci-dessus, nous avons passé 3 paramètres à la fonction `UIGraphicsBeginImageContextWithOptions()` :

1. Un objet `CGSize` qui stocke toute la taille du contexte (la toile)

2. Une valeur booléenne qui, si elle est vraie, sera opaque

3. Une valeur entière qui définit l'échelle (1 pour la non-rétine, 2 pour la rétine et 3 pour la rétine HD). S'il est défini sur 0, le système gère automatiquement l'échelle en fonction du périphérique cible.

Présentation du canevas dessiné à l'utilisateur

Rapide

```
let image = UIGraphicsGetImageFromCurrentImageContext()
imageView.image = image //assuming imageView is a valid UIImageView object
```

Objectif c

```
UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
imageView.image = image; //assuming imageView is a valid UIImageView object
```

Lire Graphiques de base en ligne: https://riptutorial.com/fr/ios/topic/5530/graphiques-de-base

Chapitre 77: Guide pour choisir les meilleurs modèles d'architecture iOS

Introduction

Démystifier MVC, MVP, MVVM et VIPER ou tout autre modèle de conception pour choisir la meilleure approche pour créer une application

Examples

Modèle 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
```

Modèles 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
```

Motif 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
```

Motif 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
```

Lire Guide pour choisir les meilleurs modèles d'architecture iOS en ligne:
https://riptutorial.com/fr/ios/topic/10029/guide-pour-choisir-les-meilleurs-modeles-d-architecture-ios

Chapitre 78: Healthkit

Examples

HealthKit

Objectif c

Allez d'abord dans `Target->Capabilities` et activez `HealthKit` . Cela configurerait l'entrée info.plist.

Créer un nouveau `CocoaClass` de type `NSObject` Le nom de fichier que j'ai donné est `GSHealthKitManager` et le fichier d'en-tête est comme indiqué ci-dessous

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
```

Appel de ViewController

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

    [[GSHealthKitManager sharedManager] requestAuthorization];
    NSDate *birthDate = [[GSHealthKitManager sharedManager] readBirthDate];
```

```
        NSLog(@"birthdate %@", birthDate);
        NSLog(@"gender 2131321 %@", [[GSHealthKitManager sharedManager] readGender]);


}
```

Sortie de journal

```
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
```

Lire Healthkit en ligne: https://riptutorial.com/fr/ios/topic/7412/healthkit

Chapitre 79: iBeacon

Paramètres

Paramètres	Détails
directeur	Référence CLLocationManager
Région	CLRegion pourrait être une région circulaire (géofence ou région de balise)
balises	Tableau de CLBeacon contient toutes les balises à distance

Remarques

Les balises sont des objets IOT. Nous nous concentrons sur ceux qui sont conformes au protocole iBeacon et à la norme Apple. Chaque balise est un dispositif à sens unique qui transmet 3 choses

1. UUID
2. Majeur
3. Mineur

Nous pouvons analyser les balises iBeacons en configurant notre objet gestionnaire CLLocation pour rechercher les balises correspondant à un UUID particulier. Toutes les balises avec l'UUID donné seront analysées.

CLLocation manager donne également un appel sur l'entrée et la sortie de la région de balise.

Examples

Fonctionnement de base d'iBeacon

1. Configuration des balises de surveillance

```
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. Le gestionnaire d'emplacement entre et sort de la région

```
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. Balise de distance de gestionnaire de localisation

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

Numérisation de balises spécifiques

```
 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
```

Rangement des iBeacons

Tout d'abord, vous devez demander l'autorisation des services de localisation

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

Vous pouvez ensuite obtenir toutes les informations sur les `didRangeBeacons` dans `didRangeBeacons`

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

Lire iBeacon en ligne: https://riptutorial.com/fr/ios/topic/1958/ibeacon

Chapitre 80: IBOutlets

Remarques

IBOutlet n'est ni un mot réservé, ni une variable ou une classe, est le sucre syntaxique pour Interface Builder. Une fois le code source d'Objective-C pré-traité, il est résolu à rien.

Dans Swift, il est résolu comme nul.

Il est déclaré dans `<UIKit/UINibDeclarations.h>` comme

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

Examples

Utilisation d'un IBOutlet dans un élément d'interface utilisateur

En général, IBOutlets permet de connecter un objet d'interface utilisateur à un autre objet, en l'occurrence un UIViewController. La connexion permet à l'objet d'être affecté mon code ou les événements par programmation. Cela peut se faire simplement en utilisant l'assistant depuis un storyboard et en cliquant sur l'élément dans la section de propriétés .h du contrôleur de vue, mais cela peut aussi se faire par programmation et connecter manuellement le code IBOutlet à l'onglet "connections" de l'objet. la barre d'utilité à droite. Voici un exemple objectif-c d'un UIViewController avec une sortie d'étiquette:

```
//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
```

Et vite:

```
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 connexion entre l'objet storyboard et l'objet programmé peut être vérifiée comme étant connectée si le point situé à gauche de la déclaration de la prise dans le .h est rempli. Un cercle vide impliquait une connexion incomplète.

Lire IBOutlets en ligne: https://riptutorial.com/fr/ios/topic/4713/iboutlets

Chapitre 81: Idiomes d'initialisation

Examples

Défini sur tuples pour éviter la répétition du code

Évitez la répétition de code dans les constructeurs en définissant un tuple de variables avec une ligne unique:

```
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)
    }
}
```

Initialiser avec des constantes de position

```
let mySwitch: UISwitch = {
    view.addSubview($0)
    $0.addTarget(self, action: "action", forControlEvents: .TouchUpInside)
    return $0
}(UISwitch())
```

Initialiser les attributs dans didSet

```
@IBOutlet weak var title: UILabel! {
  didSet {
    label.textColor = UIColor.redColor()
    label.font = UIFont.systemFontOfSize(20)
    label.backgroundColor = UIColor.blueColor()
  }
```

```
}
```

Il est également possible de définir une valeur et de l'initialiser:

```
private var loginButton = UIButton() {
    didSet(oldValue) {
        loginButton.addTarget(self, action: #selector(LoginController.didClickLogin),
forControlEvents: .TouchUpInside)
    }
}
```

Grouper les points de vente dans un NSObject personnalisé

Déplacez chaque point de vente vers un objet NSObject. Faites ensuite glisser un objet de la bibliothèque vers la scène du contrôleur du storyboard et connectez-y les éléments.

```
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!
}
```

Initialiser avec alors

Cette syntaxe est similaire à celle de l'exemple qui initialise à l'aide des constantes de position, mais nécessite l'extension `Then` de https://github.com/devxoul/Then (ci-dessous).

```
let label = UILabel().then {
    $0.textAlignment = .Center
    $0.textColor = UIColor.blackColor(
    $0.text = "Hello, World!"
}
```

L'extension `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 {}
```

Méthode d'usine avec bloc

```
internal func Init<Type>(value : Type, block: @noescape (object: Type) -> Void) -> Type
{
    block(object: value)
    return value
}
```

Usage:

```
Init(UILabel(frame: CGRect.zero)) {
    $0.backgroundColor = UIColor.blackColor()
}
```

Lire Idiomes d'initialisation en ligne: https://riptutorial.com/fr/ios/topic/3513/idiomes-d-initialisation

Chapitre 82: Instantané de UIView

Examples

Obtenir l'instantané

```
- (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;
}
```

Rapide

```
var screenshot: UIImage
{
UIGraphicsBeginImageContext(self.bounds.size);
let context = UIGraphicsGetCurrentContext();
self.layer.render(in: context)
let screenShot = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
return screenShot
}
```

Instantané avec sous-vue avec un autre balisage et du texte

- Prise en charge Portrait et Paysage à la fois type d'image
- Le dessin et les autres sous-vues peuvent être fusionnés dans mon cas, j'ajoute une étiquette pour dessiner

```
{
    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);
}
```

Lire Instantané de UIView en ligne: https://riptutorial.com/fr/ios/topic/4622/instantane-de-uiview

Chapitre 83: Intégration SqlCipher

Introduction

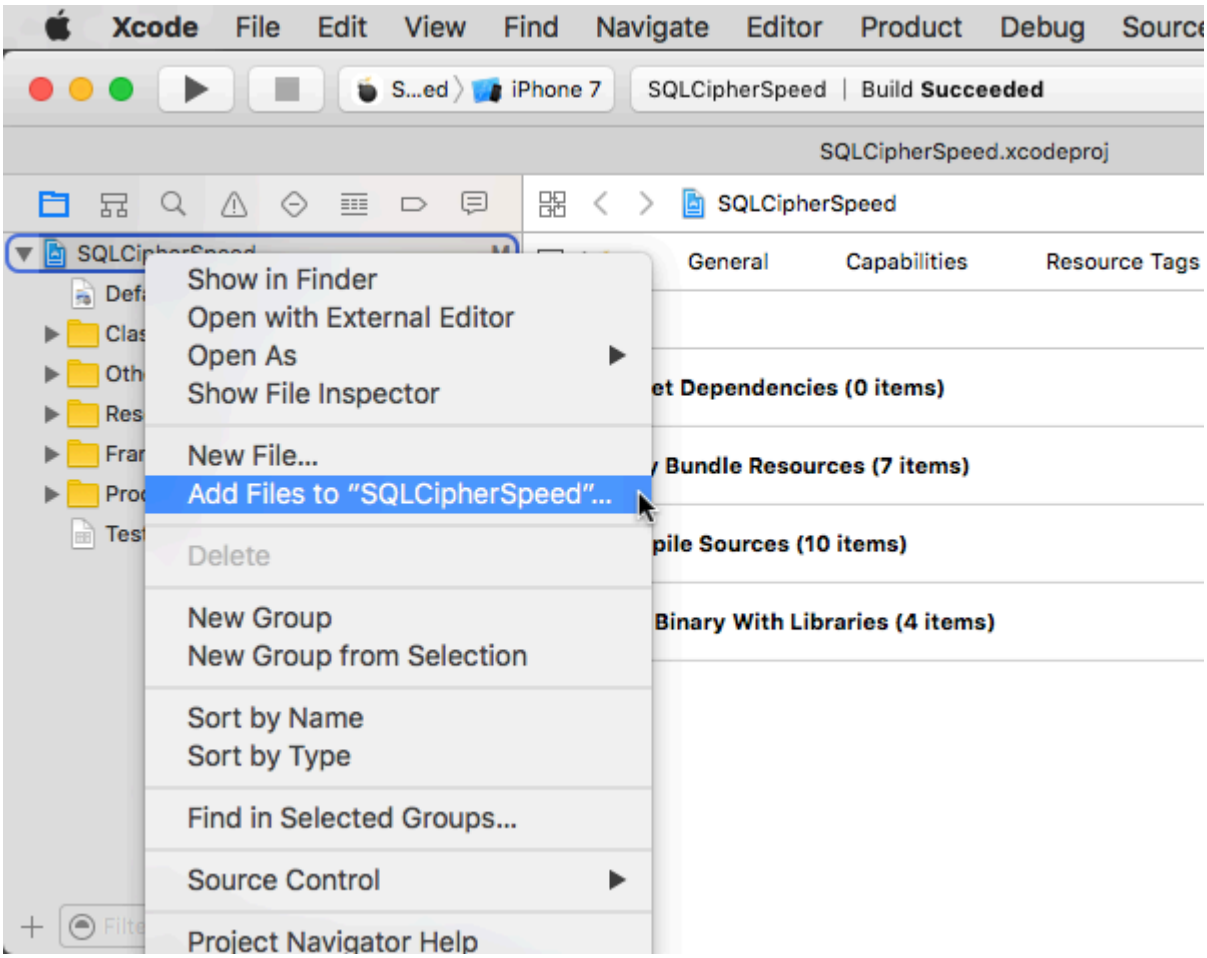
SQLite est déjà une API populaire pour le stockage de données persistant dans les applications iOS. Le développement est donc évident. En tant que programmeur, vous travaillez avec une API stable, bien documentée, et de nombreux bons wrappers sont disponibles dans Objective-C, tels que FMDB et les données de base cryptées. Tous les problèmes de sécurité sont découplés proprement du code d'application et gérés par la structure sous-jacente.

Remarques

1. Ouvrez le terminal, accédez au répertoire racine de votre projet et extrayez le code du projet SQLCipher à l'aide de Git:

```
$ git clone https://github.com/sqlcipher/sqlcipher.git
```

2. Faites un clic droit sur le projet et choisissez "Ajouter des fichiers à" Mon application "" (l'étiquette variera en fonction du nom de votre application). Puisque nous avons cloné SQLCipher directement dans le même dossier que votre application iOS, vous devriez voir un dossier sqlcipher dans votre dossier de projet racine. Ouvrez ce dossier et sélectionnez **sqlcipher.xcodeproj**

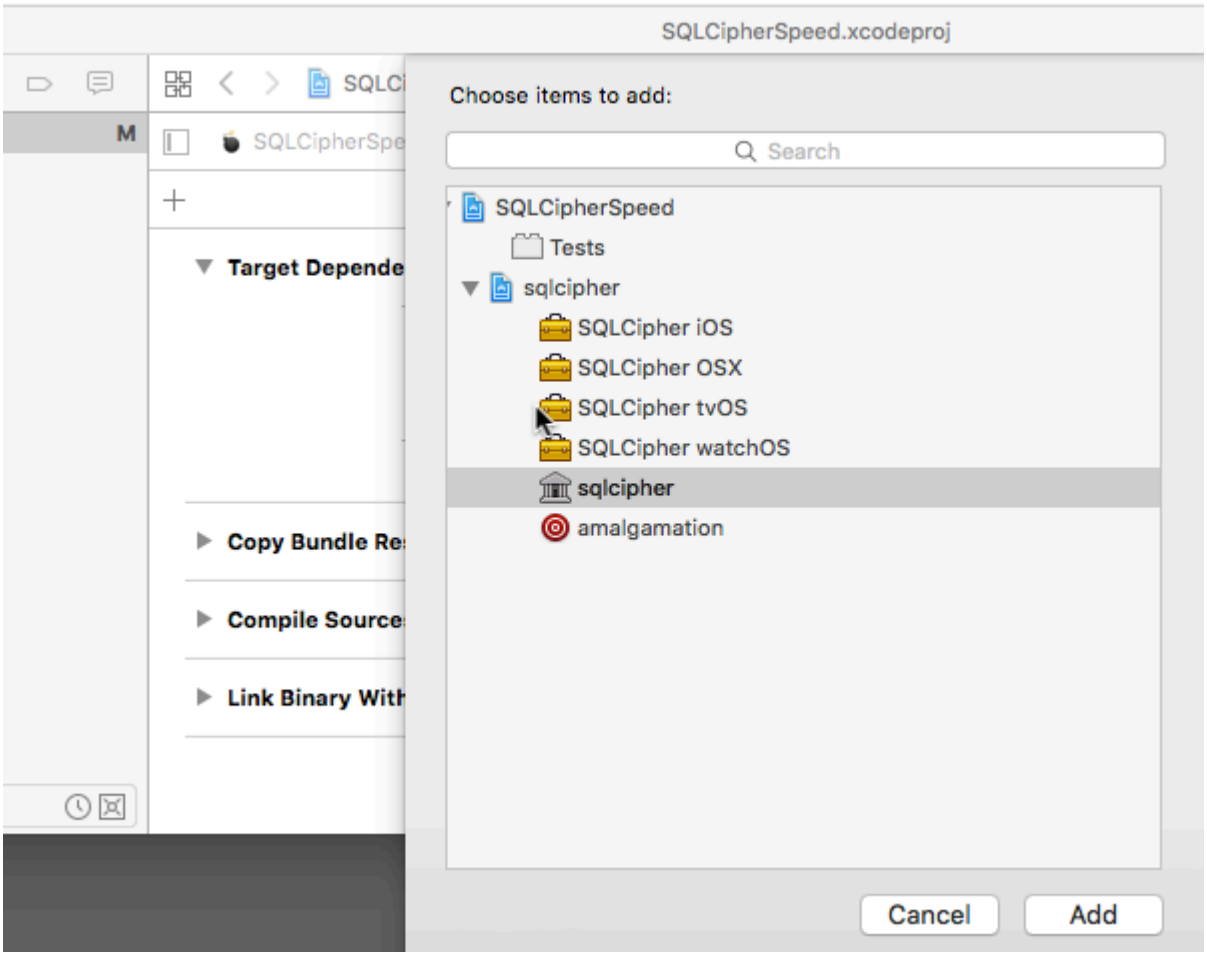


3. Sélectionnez le volet Paramètres de construction. Dans le champ de recherche, saisissez "Chemins de recherche d'en-tête". Double-cliquez sur le champ sous la colonne cible et ajoutez le chemin suivant: **$ (PROJECT_DIR) / sqlcipher / src**

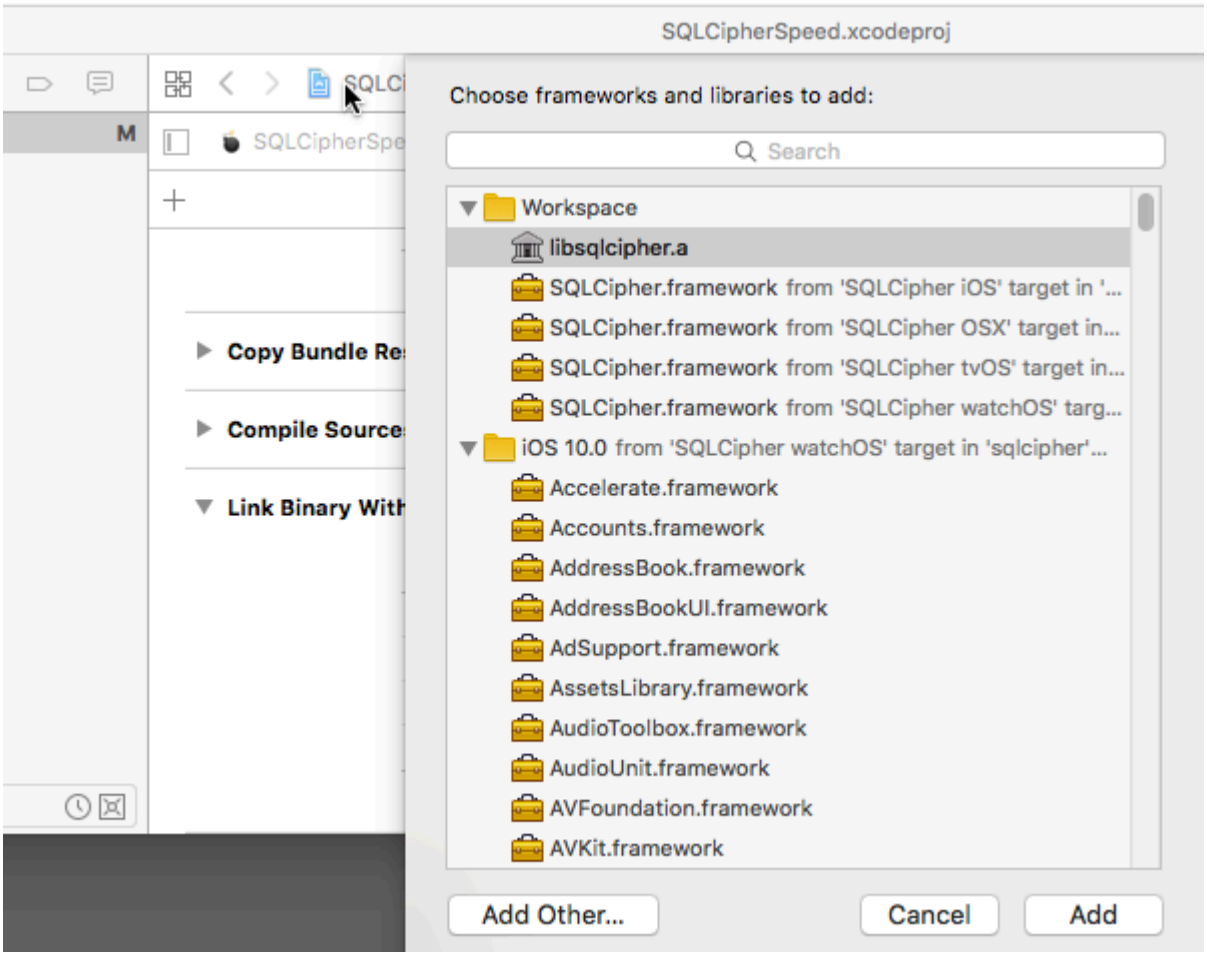
4. Commencez à taper "Autres **indicateurs de liaison** " dans le champ de recherche jusqu'à ce que le paramètre apparaisse, double-cliquez pour le modifier et ajoutez la valeur suivante: **$ (BUILT_PRODUCTS_DIR) /libsqlcipher.a**

5. Commencez à taper "Other C Flags" dans le champ de recherche jusqu'à ce que le paramètre apparaisse, double-cliquez pour le modifier et dans la **fenêtre contextuelle,** ajoutez la valeur suivante: **-DSQLITE_HAS_CODEC**

6. Développez les dépendances cibles et cliquez sur le bouton + à la fin de la liste. Dans le navigateur qui s'ouvre, sélectionnez la **cible de** la bibliothèque statique **sqlcipher** :



7. Développez Link Binary With Libraries, cliquez sur le bouton + à la fin de la liste et sélectionnez la bibliothèque **libsqlcipher.a** .



8. Enfin, également sous Link With Libraries, ajoutez **Security.framework** .

Examples

Intégration du code:

Intégration pour ouvrir la base de données à l'aide d'un mot de passe.

```
-(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;
}
```

Lire Intégration SqlCipher en ligne: https://riptutorial.com/fr/ios/topic/9969/integration-sqlcipher

Chapitre 84: Interopérabilité Swift et Objective-C

Examples

Utilisation des classes Objective-C dans Swift

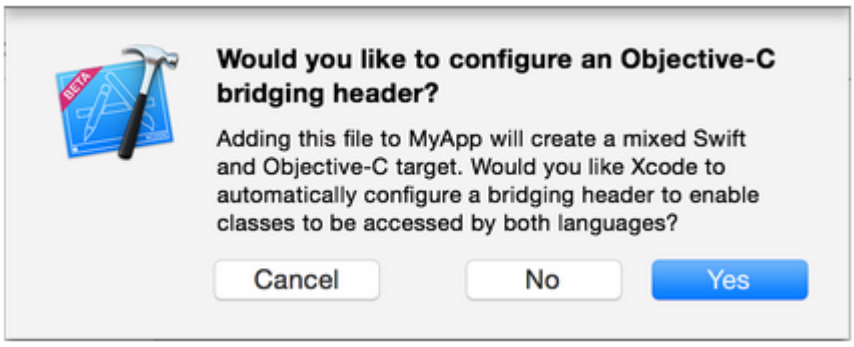
> Si vous souhaitez utiliser une classe existante, effectuez l' **étape 2** , puis passez à l' **étape 5** . (Dans certains cas, j'ai dû ajouter un `#import <Foundation/Foundation.h` explicite à un fichier ObjC plus ancien)

Étape 1: Ajouter une implémentation Objective-C - .m

Ajoutez un fichier `.m` à votre classe et nommez-le `CustomObject.m`

Étape 2: Ajouter un en-tête de pontage

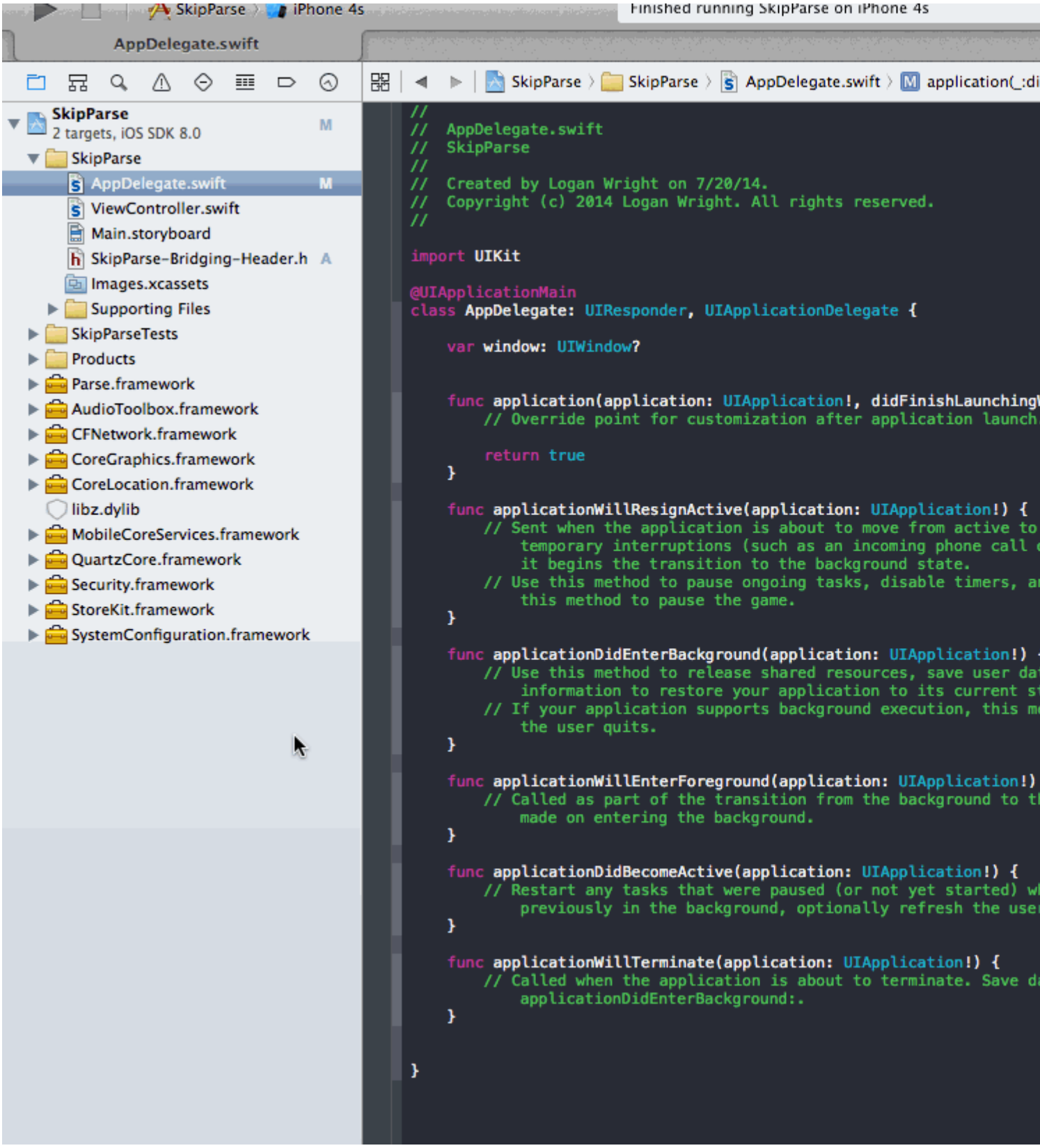
Lorsque vous ajoutez votre fichier `.m` , vous serez probablement frappé par une invite qui ressemble à ceci:



Cliquez sur **OUI** !

Si vous n'avez pas vu l'invite ou supprimé accidentellement votre en-tête de pontage, ajoutez un nouveau fichier `.h` à votre projet et nommez-le `<#YourProjectName#>-Bridging-Header.h`

Dans certaines situations, en particulier lorsque vous travaillez avec des frameworks ObjC, vous n'ajoutez pas explicitement une classe Objective-C et Xcode ne trouve pas l'éditeur de liens. Dans ce cas, créez votre fichier `.h` nommé comme mentionné ci-dessus, puis assurez-vous de lier son chemin dans les paramètres du projet de votre cible, comme suit:



Remarque

Il est `$(SRCROOT)` de lier votre projet à l'aide de la macro `$(SRCROOT)` pour que, si vous déplacez votre projet ou que vous le `$(SRCROOT)` avec d'autres utilisateurs à l'aide d'un dépôt à distance, cela fonctionne toujours. `$(SRCROOT)` peut être considéré comme le répertoire contenant votre fichier .xcodeproj. Cela pourrait ressembler à ceci:

```
$(SRCROOT)/Folder/Folder/<#YourProjectName#>-Bridging-Header.h
```

Étape 3: Ajouter un en-tête Objective-C - .h

Ajoutez un autre fichier `.h` et nommez-le `CustomObject.h`

Étape 4: Construisez votre classe Objective-C

Dans `CustomObject.h`

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

@interface CustomObject : NSObject

@property (strong, nonatomic) id someProperty;

- (void) someMethod;

@end
```

Dans `CustomObject.m`

```
#import "CustomObject.h"

@implementation CustomObject

- (void) someMethod {
    NSLog(@"SomeMethod Ran");
}

@end
```

Étape 5: Ajouter la classe à l'en-tête de pontage

Dans `YourProject-Bridging-Header.h` :

```
#import "CustomObject.h"
```

Étape 6: Utilisez votre objet

Dans `SomeSwiftFile.swift` :

```
var instanceOfCustomObject: CustomObject = CustomObject()
instanceOfCustomObject.someProperty = "Hello World"
println(instanceOfCustomObject.someProperty)
instanceOfCustomObject.someMethod()
```

Pas besoin d'importer explicitement, c'est ce que l'en-tête de pont est pour.

Utilisation des classes rapides dans Objective-C

Étape 1: Créer une nouvelle classe Swift

Ajoutez un fichier `.swift` à votre projet et nommez-le `MySwiftObject.swift`

Dans `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
    }

}
```

Étape 2: Importation de fichiers Swift dans la classe ObjC

Dans `SomeRandomClass.m` :

```
#import "<#YourProjectName#>-Swift.h"
```

Le fichier `<#YourProjectName#>-Swift.h` devrait déjà être créé automatiquement dans votre projet, même si vous ne pouvez pas le voir.

Étape 3: Utilisez votre 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);
```

Remarque:

1. CodeCompletion ne se comportait pas aussi précisément que je le voudrais. Sur mon système, exécuter une compilation rapide avec "cmd + r" semblait aider Swift à trouver une partie du code Objc et vice versa.

2. Si vous ajoutez le fichier `.swift` à un projet plus ancien et obtenez une erreur: `dyld: Library not loaded: @rpath/libswift_stdlib_core.dylib` , essayez de redémarrer complètement Xcode.

3. Alors qu'il était à l'origine possible d'utiliser des classes Swift pures dans Objective-C en utilisant le préfixe `@objc` , après Swift 2.0, cela n'est plus possible. Voir modifier l'historique pour l'explication originale. Si cette fonctionnalité est réactivée dans les futures versions de Swift, la réponse sera mise à jour en conséquence.

Lire Interopérabilité Swift et Objective-C en ligne:
https://riptutorial.com/fr/ios/topic/1497/interoperabilite-swift-et-objective-c

Chapitre 85: iOS - Implémentation de XMPP avec le framework Robbie Hanson

Examples

Exemple avec XMPP iOS Robbie Hanson avec Openfire

SRXMPPDemo

Téléchargez l'exemple et toutes les classes ici - https://github.com/SahebRoy92/SRXMPPDemo

Une démo sur XMPP en Objective C, avec diverses fonctionnalités simples et complexes. Toutes les fonctionnalités de XMPP se font par les fonctions xmpp **"in band"** . Peu de fonctionnalités contenues dans ce projet sont -

SRXMPP - Une classe Singleton wrapper qui a presque toutes les fonctionnalités nécessaires pour une application de chat en tête à tête.

- un à un chat
- Implémentation des données de base du chat (message texte) permettant ainsi de sauvegarder les messages précédents, les messages hors ligne.
- Implémentation de vCard (informations de profil de l'utilisateur, propres et autres) à partir de XML et de données de base fournies par le propre framework de Robbie Hanson.
- disponibilité du statut d'amis (en ligne / hors ligne / saisie)

Étapes à suivre

Vous souhaitez utiliser ce projet comme référence, vous pouvez alors effectuer les opérations suivantes:

1. Openfire installé sur un serveur live - Louez un serveur, installez openfire.

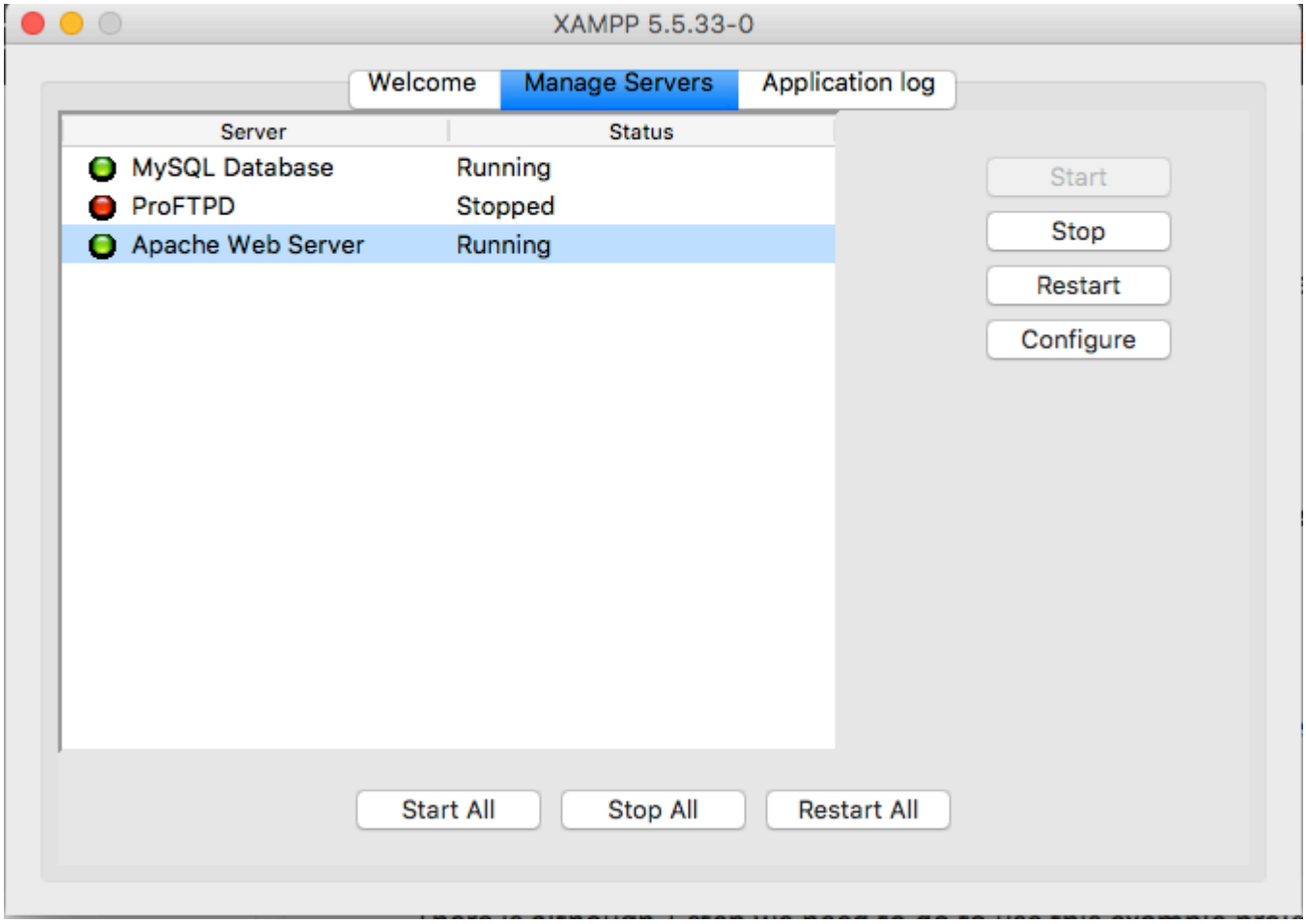
2. Voulez-vous l'essayer sans tracas dans votre propre ordinateur ? Vous devez télécharger, installer et configurer 3 choses pour commencer

une. Java -

- Téléchargez et installez Java pour Mac.

b. XAMPP -

- Installer XAMPP est relativement facile.
- Après l'installation, démarrez simplement XAMPP et lancez **Database (SQL)** et **Apache Server** .

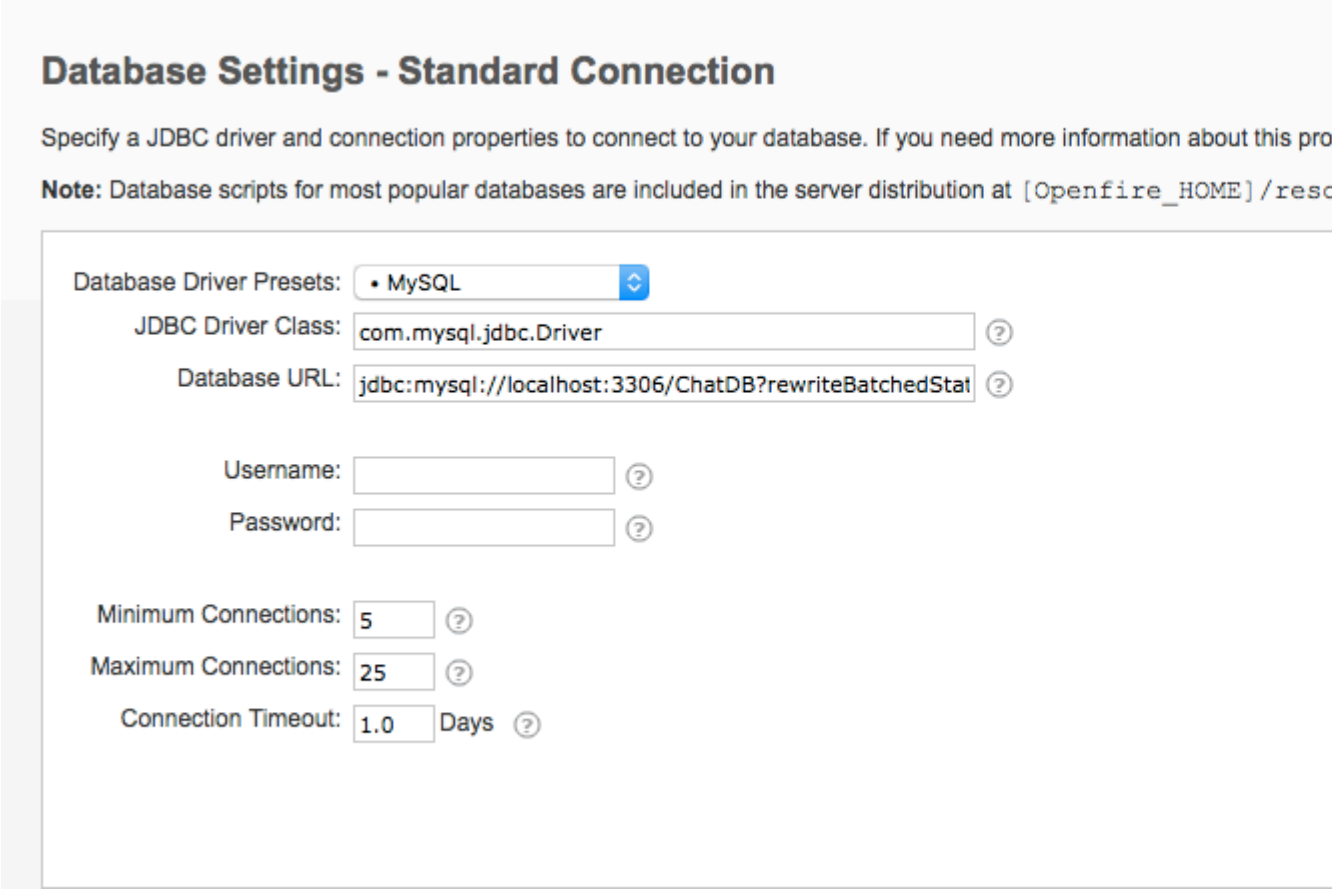


- Ensuite, ouvrez le navigateur et collez cette URL *[http: // localhost / phpmyadmin /]*
- . Créez une nouvelle base de données à partir du panneau de gauche.
- **Nommez n'importe quoi mais souvenez-vous de ce nom, supposons que nous l'appelions ChatDB**

c. Openfire -

- Installez Openfire et exécutez l'application et "Start Openfire"

- Ouvrez le navigateur et collez cette URL - [http: // localhost: 9090 / setup / index.jsp] (http: // localhost: 9090 / setup / index.jsp)
- Faire une configuration normale
    - Sélectionnez la langue>
    - Paramètres du serveur, laissez-le tel quel, continuez simplement>
    - Paramètres de base de données, laissez-le tel quel "Connexion à la base de données standard sélectionnée"
    - Paramètres de base de données - Connexion standard ". Rappelez-vous maintenant que le nom de la base de données que vous avez défini était **ChatDB** .
    - Sélectionnez les paramètres prédéfinis du pilote de base de données sous la forme * " *MySQL"* . Laissez la classe de pilote JDBC telle quelle. Maintenant, dans l'URL de la base de données, vous pouvez voir les crochets mentionnant le nom d'hôte et le nom de la base de données. **Modifiez simplement le** nom d'hôte en **"localhost"** et le nom de la base de données en **"ChatDB"** , ou tout autre nom de base de données que vous avez défini précédemment, lors de la configuration de XAMPP. Laissez le nom d'utilisateur et le mot de passe en blanc. Remplissez les détails comme l'image ici



.

    - Procédez ensuite à la configuration complète en donnant un nom d'utilisateur et un mot de passe et en le reconfirmant. C'est vous qui avez terminé Configurer Openfire.

Maintenant, la partie vient quand vous devez changer un tout petit détail dans le code.

Important Nous devons aller à la classe - **SRXMPP.m** , localiser le NSString extern **SRXMPP_Hostname** (en haut) et remplacer la valeur de celui-ci par le

- IP du serveur sur lequel OpenFire est installé, **OU**
- si vous l'avez installé localement, remplacez la valeur par - **"localhost"** .

Thats it, vous êtes prêt à utiliser cet exemple de projet et à commencer à coder et à en faire un meilleur projet de votre choix.

Ce pack de démarrage vous aidera à mieux comprendre la structure XMPP et à comprendre les protocoles XMPP.

Vous pouvez trouver d'autres protocoles XMPP ici sur ce site - https://xmpp.org/rfcs/rfc3920.html

Il reste encore du développement et des parties où j'espère les inclure plus tard

1. Discussion de groupe
2. Support d'envoi d'images

En bref, cet exemple de projet, avec le singleton, possède presque toutes les fonctionnalités nécessaires à une application de conversation en ligne.

Lire iOS - Implémentation de XMPP avec le framework Robbie Hanson en ligne: https://riptutorial.com/fr/ios/topic/1475/ios---implementation-de-xmpp-avec-le-framework-robbie-hanson

Chapitre 86: iOS TTS

Introduction

Trouver comment produire des paroles synthétisées à partir de texte sur un appareil iOS

Examples

Texte pour parler

Objectif c

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

Rapide

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

Vous pouvez également changer la voix comme ceci:

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

Et puis speek

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

N'oubliez pas d'importer AVFoundation

Méthodes utiles

Vous pouvez arrêter ou suspendre tout discours en utilisant ces deux méthodes:

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

AVSpeechBoundary indique si le discours doit être `AVSpeechBoundaryImmediate` ou arrêté immédiatement (`AVSpeechBoundaryImmediate`) ou s'il doit être `AVSpeechBoundaryWord` ou arrêté après

le mot en cours de `AVSpeechBoundaryWord` (`AVSpeechBoundaryWord`).

Lire iOS TTS en ligne: https://riptutorial.com/fr/ios/topic/8909/ios-tts

Chapitre 87: Liens universels

Remarques

1. Lorsque vous prenez en charge les liens universels, les utilisateurs d'iOS 9 peuvent accéder à un lien vers votre site Web et être redirigé de manière transparente vers votre application installée sans passer par Safari. Si votre application n'est pas installée, appuyez sur un lien vers votre site Web pour ouvrir votre site Web dans Safari.
2. En général, tout lien supporté cliqué dans Safari ou dans des instances de `UIWebView` / `WKWebView` doit ouvrir l'application.
3. Pour iOS 9.2 et moins, cela ne fonctionnera que sur un appareil. iOS 9.3 prend également en charge le simulateur.
4. iOS se souvient du choix de l'utilisateur lors de l'ouverture de liens universels. S'ils appuient sur le fil d'Ariane en haut à droite pour ouvrir le lien dans Safari, tous les clics ultérieurs les mèneront à Safari, et non à l'application. Ils peuvent revenir à l'ouverture de l'application par défaut en choisissant Ouvrir dans la bannière de l'application sur le site Web.

Examples

Serveur d'installation

Vous devez avoir un serveur en ligne. Pour associer en toute sécurité votre application iOS à un serveur, Apple vous demande de mettre à disposition un fichier de configuration, appelé `apple-app-site-association` . Ceci est un fichier `JSON` qui décrit le domaine et les itinéraires pris en charge.

Le fichier `apple-app-site-association` doit être accessible via `HTTPS` , sans aucune redirection, à l' *adresse https: // {domain} / apple-app-site-association* .

Le fichier ressemble à ceci:

```
{
"applinks": {
    "apps": [ ],
    "details": [
        {
            "appID": "{app_prefix}.{app_identifier}",
            "paths": [ "/path/to/content", "/path/to/other/*", "NOT /path/to/exclude" ]
        },
        {
            "appID": "TeamID.BundleID2",
            "paths": [ "*" ]
        }
    ]
}
}
```

REMARQUE - *N'ajoutez pas* `.json` *au nom de* `.json apple-app-site-association` *.*

Les clés sont les suivantes:
`apps` : Doit avoir un tableau vide comme valeur et il doit être présent. C'est comme ça que Apple le veut.
`details` : est un tableau de dictionnaires, un pour chaque application iOS prise en charge par le site Web. Chaque dictionnaire contient des informations sur l'application, les identifiants d'équipe et de bundle.

Il y a 3 façons de définir des chemins:
`Static` : l'intégralité du chemin pris en charge est codé en dur pour identifier un lien spécifique, par exemple / statique / termes
`Wildcards` : Un * peut être utilisé pour correspondre à des chemins dynamiques, par exemple / books / * can correspond au chemin d'accès à la page d'un auteur. ? à l'intérieur des composants de chemin spécifiques, par exemple des livres / 1? peut être utilisé pour faire correspondre tous les livres dont l'identifiant commence par 1.
`Exclusions` : la préparation d'un chemin avec NOT exclut la correspondance de ce chemin.

L'ordre dans lequel les chemins sont mentionnés dans le tableau est important. Les indices antérieurs ont une priorité plus élevée. Une fois qu'un chemin correspond, l'évaluation s'arrête et d'autres chemins sont ignorés. Chaque chemin est sensible à la casse.

Code de site Web

Le code du site Web peut être trouvé dans la branche gh-pages sur https://github.com/vineetchoudhary/iOS-Universal-Links/tree/gh-pages

Prise en charge de plusieurs domaines

Chaque domaine pris en charge par l'application doit mettre à disposition son propre fichier apple-app-site-association. Si le contenu servi par chaque domaine est différent, le contenu du fichier changera également pour prendre en charge les chemins respectifs. Sinon, le même fichier peut être utilisé, mais il doit être accessible sur tous les domaines pris en charge.

Signature du fichier app-site-association

Remarque : *vous pouvez ignorer cette partie si votre serveur utilise le `HTTPS` pour diffuser du contenu et accéder au guide Configuration de l'application.*

Si votre application cible iOS 9 et que votre serveur utilise `HTTPS` pour diffuser du contenu, vous n'avez pas besoin de signer le fichier. Si ce n'est pas le cas (par exemple, lors de la prise en charge de Handoff sur iOS 8), il doit être signé à l'aide d'un certificat `SSL` provenant d'une autorité de certification reconnue.

Remarque : il ne s'agit pas du certificat fourni par Apple pour soumettre votre application à l'App Store. Il doit être fourni par un tiers et il est recommandé d'utiliser le même certificat que celui utilisé pour votre serveur `HTTPS` (même si ce n'est pas obligatoire).

Pour signer le fichier, créez et enregistrez d'abord une version simple .txt. Ensuite, dans le

terminal, exécutez la commande suivante:

```
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
```

Cela affichera le fichier signé dans le répertoire en cours. Les `example.com.key` , `example.com.pem` et `intermediate.pem` sont les fichiers qui vous seraient mis à disposition par votre autorité de certification.

Note : Si le fichier n'est pas signé, il doit avoir un `Content-Type` d' `application/json` . Sinon, cela devrait être `application/pkcs7-mime` .

Validez votre serveur avec l'outil de validation de recherche Apple App
Testez votre page Web pour les API de recherche iOS 9. Entrez une URL et Applebot explorera votre page Web et montrera comment vous pouvez optimiser les meilleurs résultats
https://search.developer.apple.com/appsearch-validation-tool/

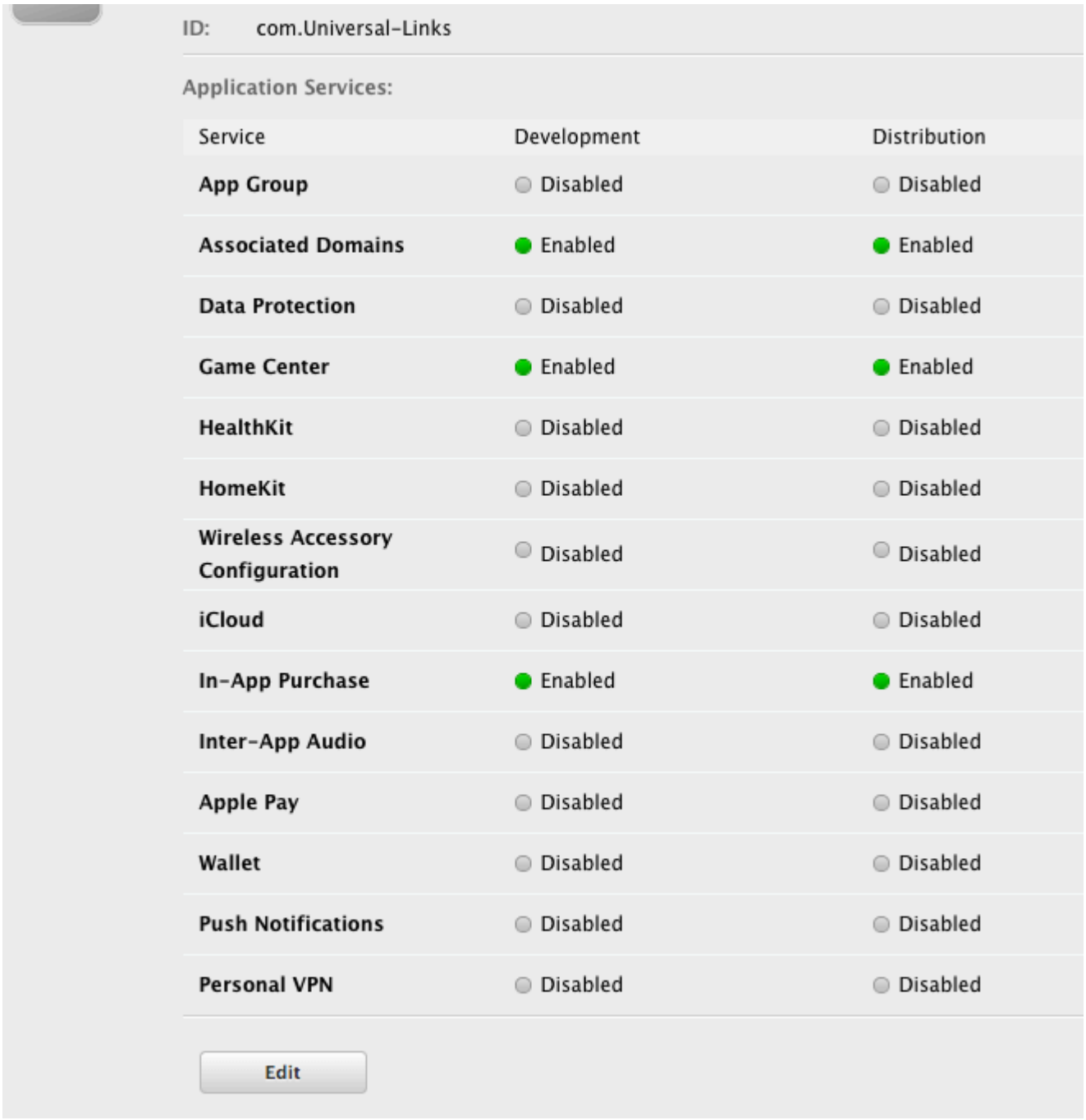
Configuration de l'application iOS (Activation des liens universels)

La configuration côté application nécessite deux choses:

1. Configuration du droit d'accès à l'application et activation des liens universels en activant la fonctionnalité Domaines associés dans le projet.
2. Gestion des liens entrants dans votre `AppDelegate` .

1. Configurer le droit d'accès à l'application et activer les liens universels.

La première étape de la configuration des droits d'accès de votre application consiste à l'activer pour votre ID d'application. Faites-le dans le Centre des membres Apple Developer. Cliquez sur Certificats, identifiants et profils, puis identificateurs. Sélectionnez votre ID d'application (créez-le d'abord si nécessaire), cliquez sur Modifier et activez le droit aux domaines associés.



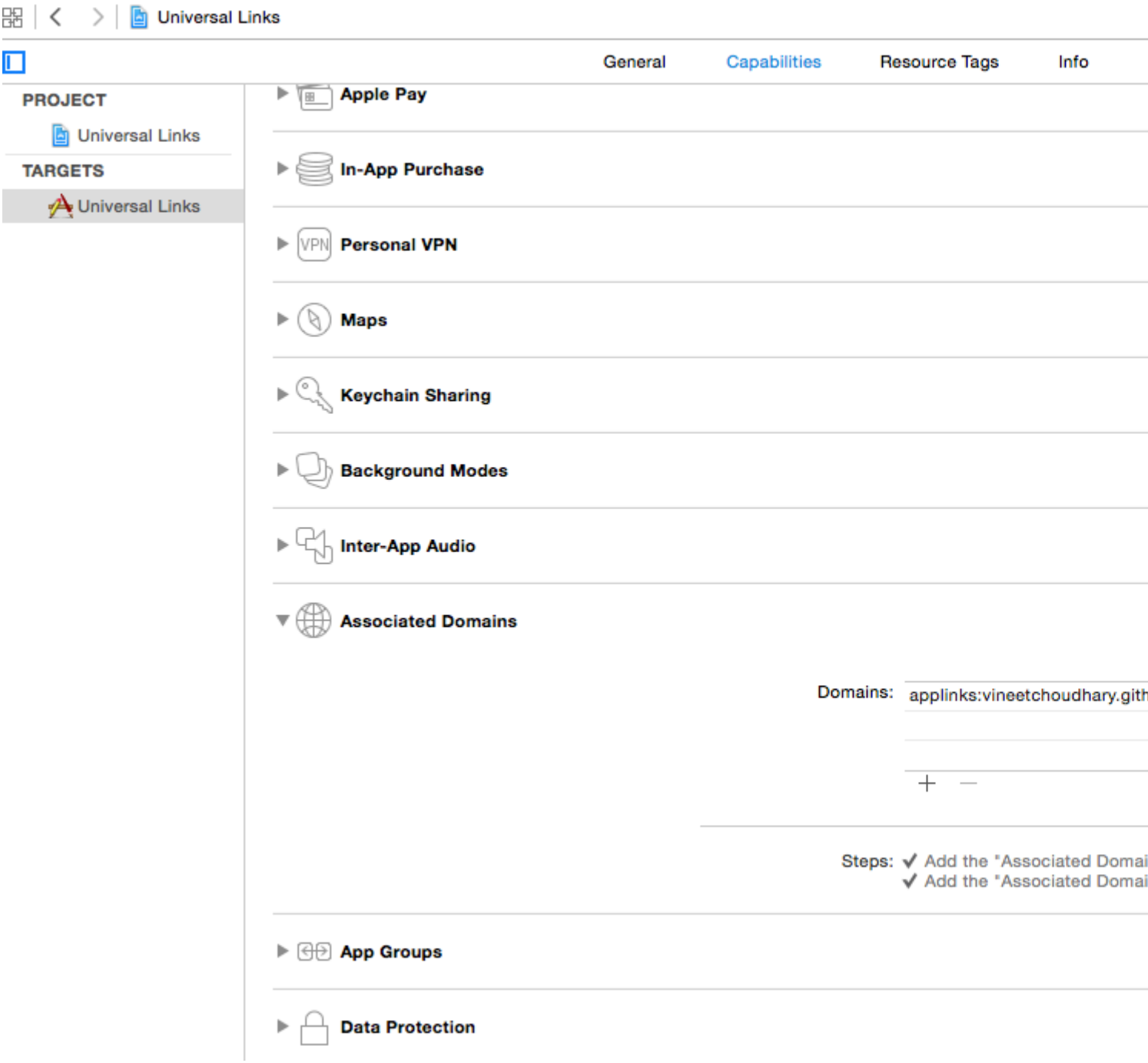
Ensuite, obtenez le préfixe et le suffixe de l'ID d'application en cliquant sur l'ID de l'application correspondante.

Le préfixe et le suffixe de l'ID d'application doivent correspondre à celui du fichier apple-app-site-association.

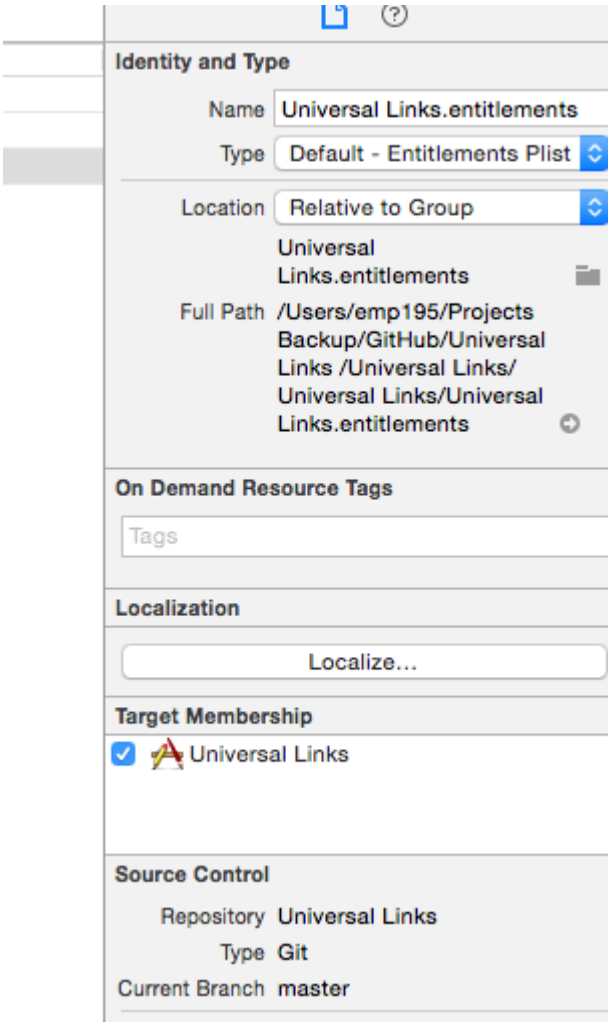
Ensuite, dans `Xcode` , sélectionnez la cible de votre application, cliquez sur Capabilities et activez l'option. Ajoutez une entrée pour chaque domaine pris en charge par votre application, préfixée par des **liens d'application:**

Par exemple, **applinks: YourCustomDomainName.com**

Ce qui ressemble à ceci pour l'exemple d'application:



Remarque : Assurez-vous d'avoir sélectionné la même équipe et saisi le même ID de lot que l'ID d'application enregistrée sur le Centre de membres. Assurez-vous également que le fichier de droits est inclus par Xcode en sélectionnant le fichier et, dans l'inspecteur de fichiers, vérifiez que votre cible est cochée.



---

2. Gestion des liens entrants dans votre AppDelegate

Toutes les redirections de Safari vers l'application pour les liens universels passent par la méthode ci-dessous dans la classe AppDelegate de l'application. Vous analysez cette URL pour déterminer la bonne action dans l'application.

```
[UIApplicationDelegate application: continueUserActivity: restorationHandler:]
```

Objectif 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;
}
```

Rapide :

```
func application(application: UIApplication, continueUserActivity userActivity:
NSUserActivity, restorationHandler: ([AnyObject]?) -> Void) -> Bool {
      if userActivity.activityType == NSUserActivityTypeBrowsingWeb {
          let url = userActivity.webpageURL!
          //handle url
      }
      return true
  }
```

Code d'application iOS

Le code de l'application peut être trouvé ici .

Lire Liens universels en ligne: https://riptutorial.com/fr/ios/topic/2362/liens-universels

Chapitre 88: Localisation

Introduction

La localisation est une fonctionnalité fournie par iOS qui traduit votre application en plusieurs **langues.Pour la localisation** , l' **internationalisation** est nécessaire. **L'internationalisation** est un processus qui permet à l'application iOS d'adapter différentes cultures, langues et régions.

Examples

Localisation dans iOS

Créez un fichier `Localizable.strings` individuel pour chaque langue. Le côté droit serait différent pour chaque langue. Considérez-le comme une paire clé-valeur:

```
"str" = "str-language";
```

Accédez à str dans 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");
```

Accès str à Swift:

```
let str = NSLocalizedString("string", comment: "language");
```

Lire Localisation en ligne: https://riptutorial.com/fr/ios/topic/1579/localisation

Chapitre 89: Messagerie FCM dans Swift

Remarques

FCM: https://firebase.google.com/docs/cloud-messaging/ios/client

Examples

Initialiser FCM dans Swift

suivez l'étape ci-dessous pour ajouter FCM dans votre projet rapide

1- Si vous n'avez pas encore de projet Xcode, créez-en un maintenant. Créez un Podfile si vous n'en avez pas:

```
$ cd le répertoire de votre projet
$ pod init
```

2- Ajoutez les modules que vous souhaitez installer. Vous pouvez inclure un pod dans votre Podfile comme ceci:

```
pod 'Firebase / Core'
pod 'Firebase / Messaging'
```

3- Installez les modules et ouvrez le fichier .xcworkspace pour voir le projet dans Xcode.

```
$ pod installation
$ ouvrez votre-project.xcworkspace
```

4- Téléchargez un fichier GoogleService-Info.plist de plist et incluez-le dans votre application.

5- Téléchargez le certificat APNs sur Firebase. APN Cert

6- ajouter "import Firebase" dans votre fichier appDelegate du projet

7- ajoutez ceci "FIRApp.configure ()" dans votre "application: didFinishLaunchingWithOptions"

8- enregistrer pour notification à distance

```
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- pour obtenir l'utilisation du jeton de registre

```
let token = FIRInstanceID.instanceID().token()!
```

10- et si vous voulez surveiller le changement de jeton, utilisez le code ci-dessous dans le fichier 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- recevoir un message de fcm ajouter le code ci-dessous dans appDelegate

```
func connectToFcm() {
  FIRMessaging.messaging().connectWithCompletion { (error) in
    if (error != nil) {
      print("Unable to connect with FCM. \(error)")
    } else {
      print("Connected to FCM.")
    }
  }
}
```

12- et pour une utilisation de déconnexion

```
func applicationDidEnterBackground(application: UIApplication) {
  FIRMessaging.messaging().disconnect()
  print("Disconnected from FCM.")
}
```

dans votre appDelegate.

l'initialisation terminée et le client prêt à recevoir un message du panneau fcm ou à envoyer par jeton depuis un serveur tiers

Lire Messagerie FCM dans Swift en ligne: https://riptutorial.com/fr/ios/topic/7326/messagerie-fcm-dans-swift

Chapitre 90: Méthodes personnalisées de sélection de UITableViewCells

Introduction

Moyens avancés pour gérer les sélections de UITableViewCell. Exemples où simple `didSelect...` forme `UITableViewDelegate` ne suffit pas pour réaliser quelque chose.

Examples

Distinction entre la sélection simple et double en ligne.

Un exemple d'implémentation qui donne la possibilité de détecter si l'utilisateur tape un ou deux fois sur 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)
    }
}
```

Lire Méthodes personnalisées de sélection de UITableViewCells en ligne:
https://riptutorial.com/fr/ios/topic/9961/methodes-personnalisees-de-selection-de-uitableviewcells

Chapitre 91: Méthodes personnalisées de sélection de UITableViewCells

Examples

Distinction entre la sélection simple et double en ligne.

Un exemple d'implémentation UITableView qui permet de détecter si la cellule a été exploitée une ou deux fois.

```
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)
    }
}
```

Lire Méthodes personnalisées de sélection de UITableViewCells en ligne: https://riptutorial.com/fr/ios/topic/9962/methodes-personnalisees-de-selection-de-uitableviewcells

Chapitre 92: Mise à jour dynamique d'un UIStackView

Examples

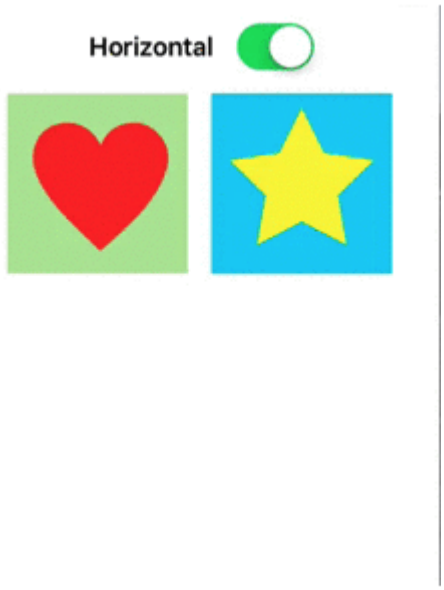
Connectez le UISwitch à une action que nous pouvons animer en basculant entre une disposition horizontale ou verticale des vues de l'image

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

La fonction updateConstraintForAxis définit simplement l'axe de la vue de pile contenant les deux vues d'image:

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

Le gif animé ci-dessous vous donne une idée de la façon dont cela apparaît:



Lire Mise à jour dynamique d'un UIStackView en ligne:

https://riptutorial.com/fr/ios/topic/5884/mise-a-jour-dynamique-d-un-uistackview

Chapitre 93: Mise en forme automatique UIScrollView

Examples

ScrollableController

Lorsque vous utilisez Autolayout avec un `UIScrollView` , il ne redimensionne PAS correctement en fonction de la taille de son contenu ou de ses sous-vues.

Pour qu'un `UIScrollView` défile automatiquement lorsque son contenu devient trop volumineux pour tenir dans la zone visible, nous devons ajouter un `ContentView` et des contraintes permettant à `UIScrollView` de déterminer la taille de son contenu ET sa largeur et sa hauteur dans son parent. vue.

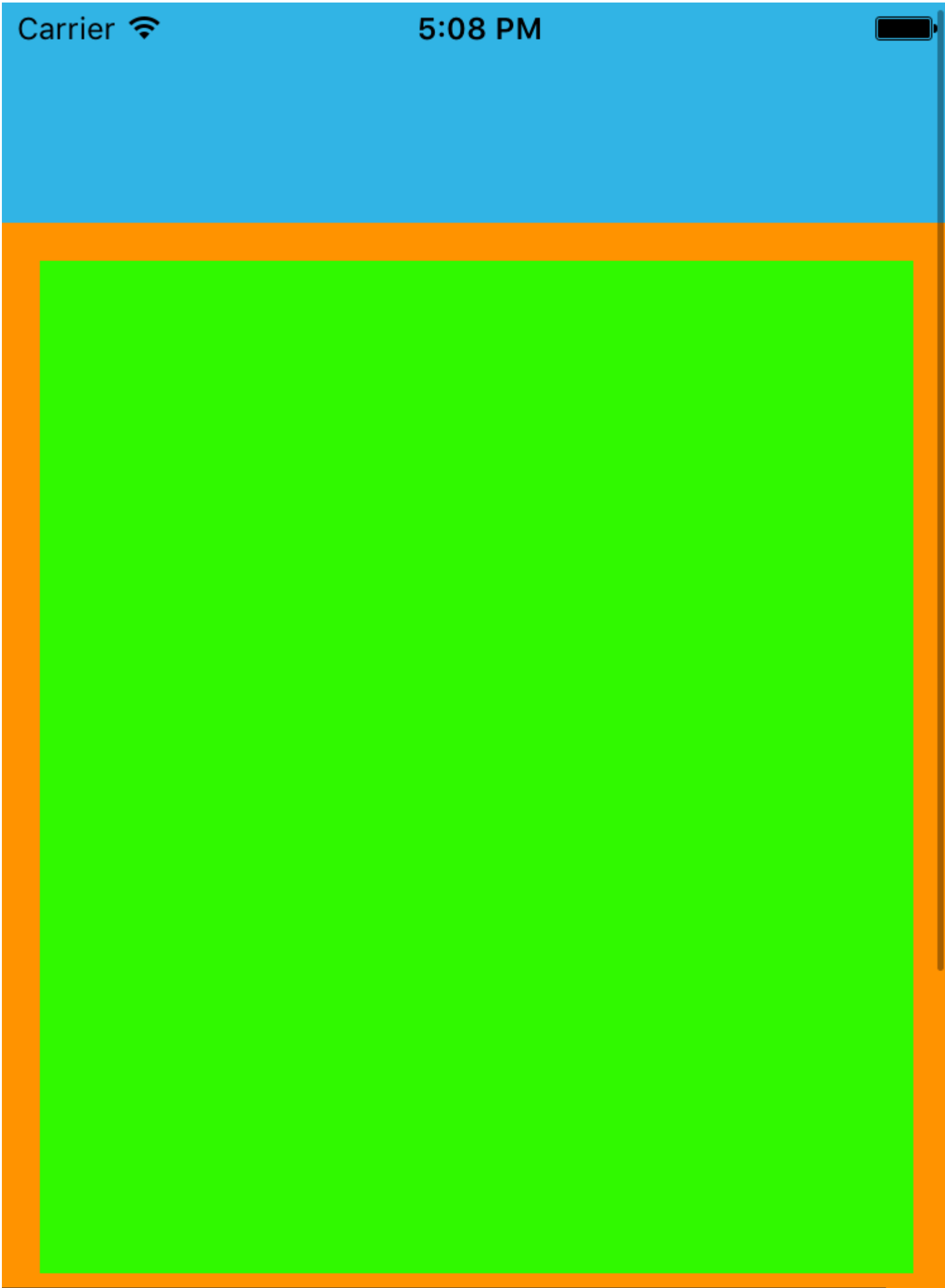
```
import Foundation
import UIKit

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
    }
}
```

Maintenant, nous pouvons voir que le greenView (400 hauteurs) + le whiteView (500 hauteurs) est plus grand que notre écran. Cela fera croître le contentSize de ScrollView pour qu'il s'adapte aux deux vues, ce qui lui permettra de défiler verticalement.

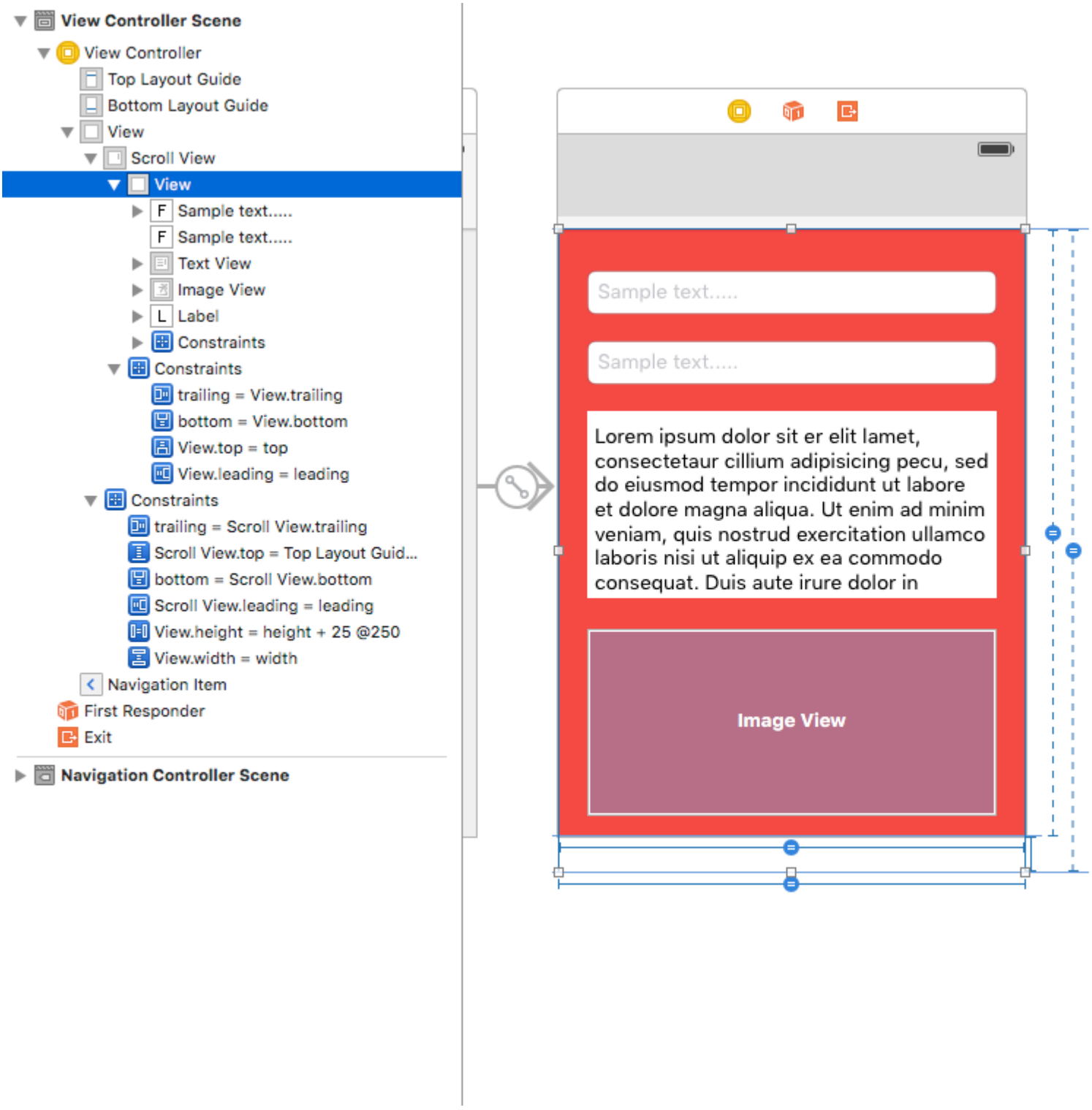
Nous avons désactivé le défilement horizontal en utilisant la contrainte `EqualWidth` sur `contentView` et `self.view`

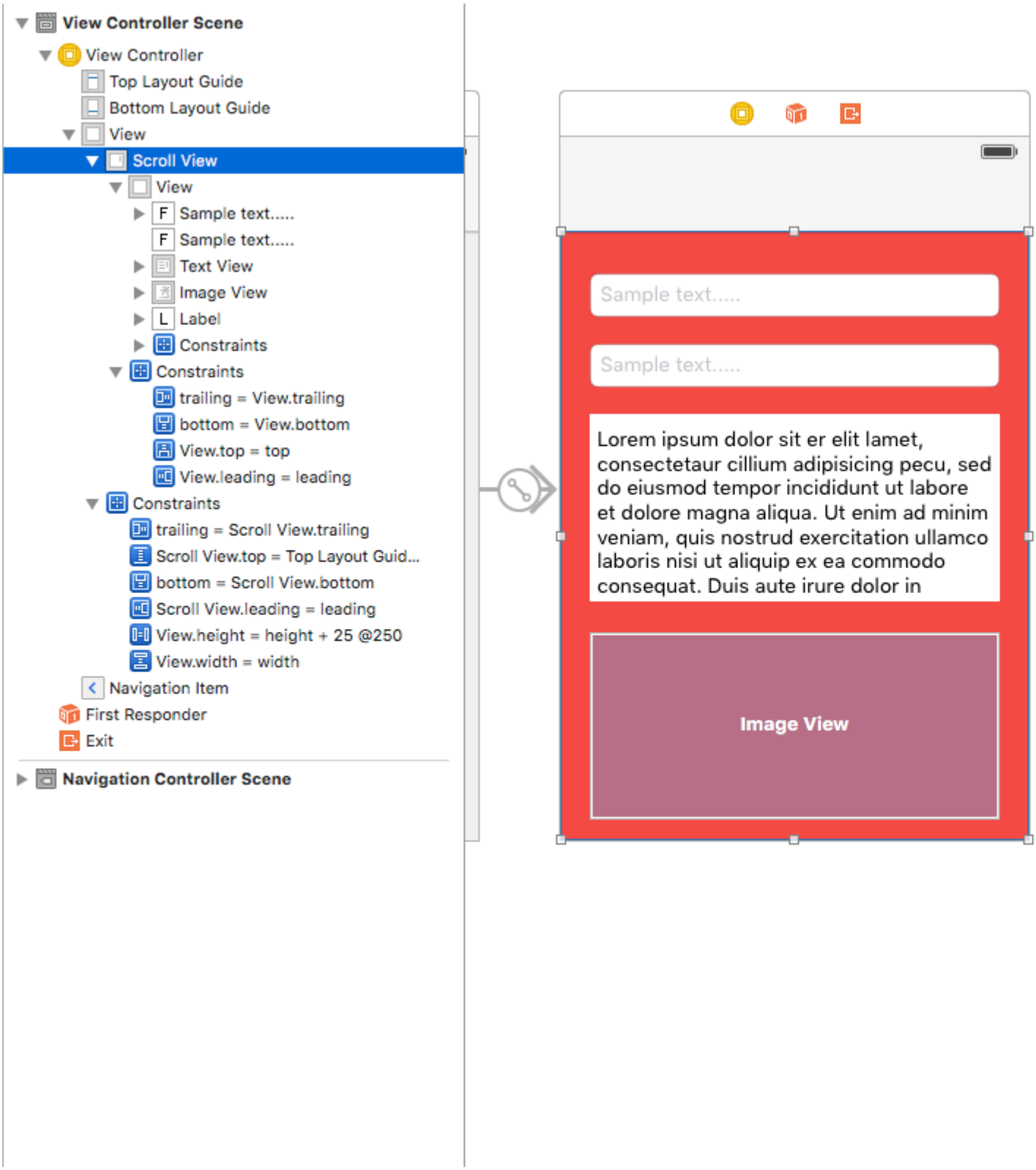

Carrier
5:08 PM

2. Ajoutez la même hauteur, la même largeur à la vue principale (c.-à-d. Qui contient la défilement). Pour une hauteur égale, définissez la priorité sur faible. (C'est l'étape importante pour définir la taille du contenu).

3. La hauteur de cette vue de contenu sera fonction du nombre de vues ajoutées à la vue. Si vous avez ajouté la dernière vue à une étiquette et que sa position Y est 420 et que la hauteur est 20, votre vue du contenu sera 440.

Étape 3: ajoutez des contraintes à toutes les vues que vous avez ajoutées dans la vue du contenu, conformément à vos exigences.





Lire Mise en forme automatique UIScrollView en ligne:
https://riptutorial.com/fr/ios/topic/4671/mise-en-forme-automatique-uiscrollview

Chapitre 94: Mise en page automatique

Introduction

La mise en forme automatique calcule dynamiquement la taille et la position de toutes les vues de votre hiérarchie de vue, en fonction des contraintes imposées à ces vues. La source

Syntaxe

- NSLayoutConstraint (item: Any, attribut: NSLayoutAttribute, relatedBy: NSLayoutRelation, toItem: Any?) Attribut: NSLayoutAttribute, multiplicateur: CGFloat, constante: CGFloat) // Créer une contraint par programme

Examples

Définition de contraintes par programmation

Exemple de code de chaudière

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

    let myView = UIView()
    myView.backgroundColor = UIColor.blueColor()
    myView.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(myView)

    // Add constraints code here
    // ...
}
```

Dans les exemples ci-dessous, le style d'ancrage est la méthode préférée sur le style `NSLayoutConstraint` , mais il n'est disponible qu'à partir d'iOS 9, donc si vous supportez iOS 8, vous devez toujours utiliser le style `NSLayoutConstraint` .

Pinning

Style d'ancre

```
let margins = view.layoutMarginsGuide
myView.leadingAnchor.constraintEqualToAnchor(margins.leadingAnchor, constant: 20).active =
true
```

- En plus de `leadingAnchor` , il existe également `trailingAnchor` , `topAnchor` et `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
```

- En plus de `.Leading` il est également `.Trailing` , `.Top` et `.Bottom` .
- En plus de `.LeadingMargin` il existe également `.TrailingMargin` , `.TopMargin` et `.BottomMargin` .

Format de langage de format visuel

```
NSLayoutConstraint.constraintsWithVisualFormat("H:|-20-[myViewKey]", options: [], metrics:
nil, views: ["myViewKey": myView])
```

Largeur et hauteur

Style d'ancre

```
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
```

Format de langage de format visuel

```
NSLayoutConstraint.constraintsWithVisualFormat("H:[myViewKey(200)]", options: [], metrics:
nil, views: ["myViewKey": myView])
NSLayoutConstraint.constraintsWithVisualFormat("V:[myViewKey(100)]", options: [], metrics:
nil, views: ["myViewKey": myView])
```

Centre en conteneur

Style d'ancre

```
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
```

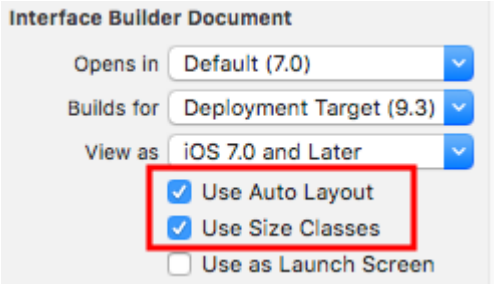
Format de langage de format visuel

```
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])
```

Comment utiliser la mise en page automatique

La mise en page automatique est utilisée pour organiser les vues de manière à leur donner une apparence agréable sur tous les périphériques et toutes les orientations. Les contraintes sont les règles qui indiquent comment tout doit être défini. Ils comprennent, entre autres, l'épinglage des bords, le centrage et la taille des paramètres.

La mise en page automatique est activée par défaut, mais vous pouvez vérifier cela deux fois. Si vous cliquez sur *Main.storyboard* dans le navigateur de projet, puis affichez l'inspecteur de fichiers. Assurez-vous que la mise en page automatique et les classes de taille sont cochées:



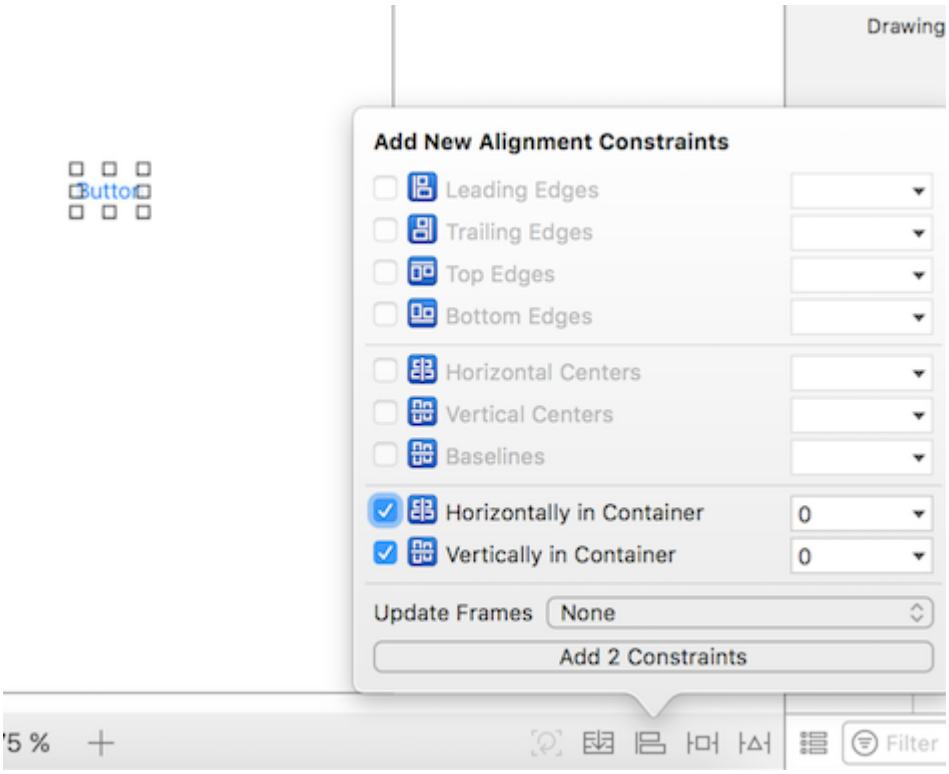
Les contraintes de présentation automatique peuvent être définies dans Interface Builder ou dans le code. Dans Interface Builder, vous trouvez les outils de mise en page automatique en bas à droite. En cliquant dessus, vous découvrirez différentes options pour définir les contraintes sur une vue.

Si vous souhaitez appliquer différentes contraintes pour différentes tailles ou orientations d'appareils, vous pouvez les définir dans les options de classe de taille disponibles dans la partie inférieure.

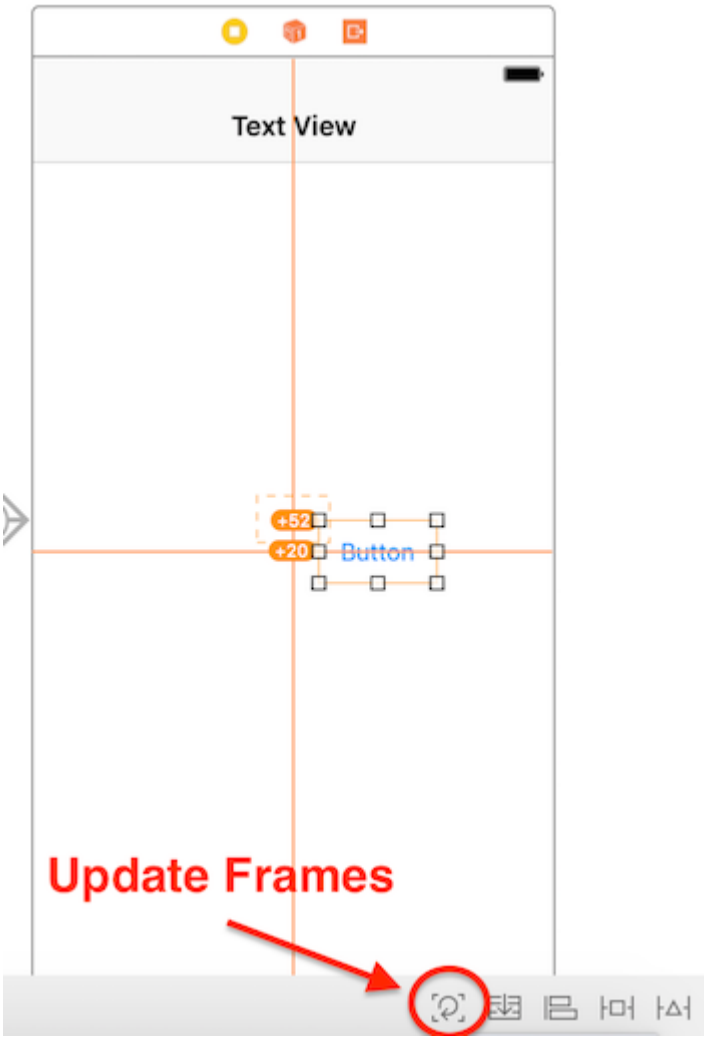
Contraintes Centrales

Sélectionnez votre bouton (ou la vue que vous souhaitez centrer) sur le **storyboard** . Cliquez ensuite sur le bouton d'alignement en bas à droite. Sélectionnez `Horizontally in Container` et

`Vertically in Container` . Cliquez sur "Ajouter 2 contraintes".



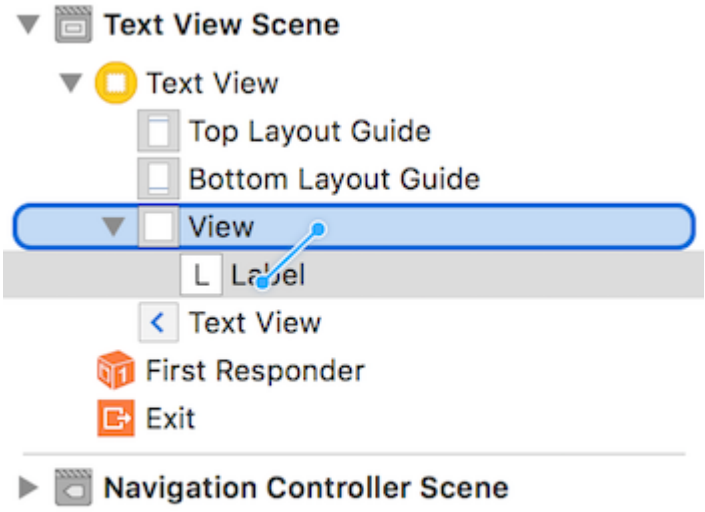
Si cela n'était pas déjà parfaitement centré, vous devrez peut-être faire encore une chose. Cliquez sur le bouton "Mettre à jour les cadres" qui se trouve à gauche du bouton "Embed In Stack" sur la barre inférieure.



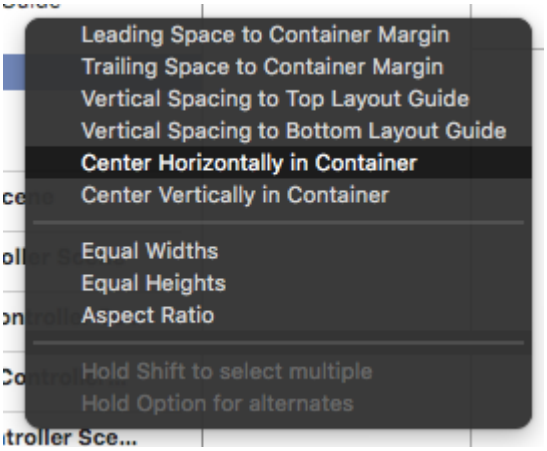
Vous pouvez également "mettre à jour les images si nécessaire" en appuyant `simultanément` sur ⌘ + ⌥ + = (Commande + Option et égale) après avoir sélectionné la vue, cela pourrait vous faire gagner du temps.

Maintenant, lorsque vous exécutez votre application, elle doit être centrée, quelle que soit la taille de votre appareil.

Un autre moyen de centrer les vues à l'aide d'Interface Builder consiste à effectuer un glisser-déposer par contrôle. Supposons que vous souhaitiez centrer un `UILabel` dans une vue. Ouvrez la `Document Outline` du `Document Outline` dans votre storyboard en cliquant sur le bouton de la barre latérale en bas à gauche. Cliquez et faites glisser depuis l'étiquette vers la vue tout en maintenant la `touche ctrl` (contrôle) `enfoncée` , et une ligne bleue doit apparaître:



A la sortie, un menu d'options de contrainte apparaîtra:



Sélectionnez "Centrer horizontalement dans le conteneur" et "Centrer verticalement dans le conteneur". Mettez à jour les cadres si nécessaire, et le tour est joué! Une étiquette centrée

Vous pouvez également ajouter les contraintes par programmation. Créez les contraintes et ajoutez-les aux éléments d'interface utilisateur et aux vues souhaités, comme le décrit l'exemple suivant, où nous créons un bouton et l'alignons au centre, horizontalement et verticalement sur sa vue d'ensemble:

Objectif c

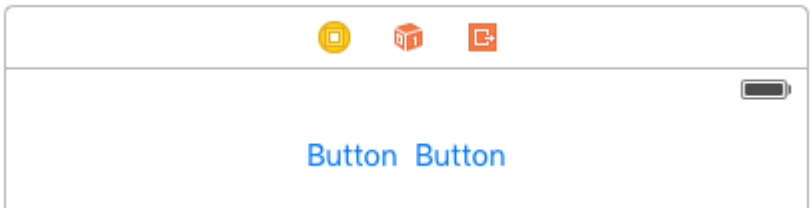
```
- (void)viewDidLoad
{
    [super viewDidLoad];
    UIButton *yourButton = [[UIButton alloc] initWithFrame:CGRectMake(0, 0, 100, 18)];
    [yourButton setTitle:@"Button" forState:UIControlStateNormal];


    [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
}
```

Rapide

```
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])
}
```

Vues de l'espace uniformément



Il est courant de vouloir que deux vues soient côte à côte, centrées dans leur superview. La réponse commune donnée à Stack Overflow est d'incorporer ces deux vues dans un `UIView` et de centrer le `UIView` . Ce n'est pas nécessaire ou recommandé. De la documentation UILayoutGuide :

> Il existe un certain nombre de coûts associés à l'ajout de vues factices à votre hiérarchie de vues. Tout d'abord, il y a le coût de la création et du maintien de la vue elle-même. Deuxièmement, la vue factice est un membre à part entière de la hiérarchie des vues, ce qui signifie qu'elle ajoute une surcharge à chaque tâche exécutée par la hiérarchie. Pire encore, la vue factice invisible peut intercepter les messages destinés à d'autres vues, ce qui pose des problèmes très difficiles à trouver.

Vous pouvez utiliser `UILayoutGuide` pour cela, au lieu d'ajouter les boutons dans un `UIView` inutile.

Un `UILayoutGuide` est essentiellement un espace rectangulaire pouvant interagir avec Auto Layout. Vous placez un `UILayoutGuide` sur les côtés gauche et droit des boutons et définissez leur largeur pour qu'ils soient égaux. Cela permettra de centrer les boutons. Voici comment le faire en code:

Format de langage de format visuel

```
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
```

Style d'ancre

```
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
```

Vous aurez également besoin d'ajouter des contraintes verticales à cela, mais cela centrera les boutons dans la vue sans ajouter de vues "factices"! Cela évitera au système de perdre du temps CPU pour afficher ces vues "factices". Cet exemple utilise des boutons, mais vous pouvez échanger des boutons pour n'importe quelle vue sur laquelle vous souhaitez imposer des contraintes.

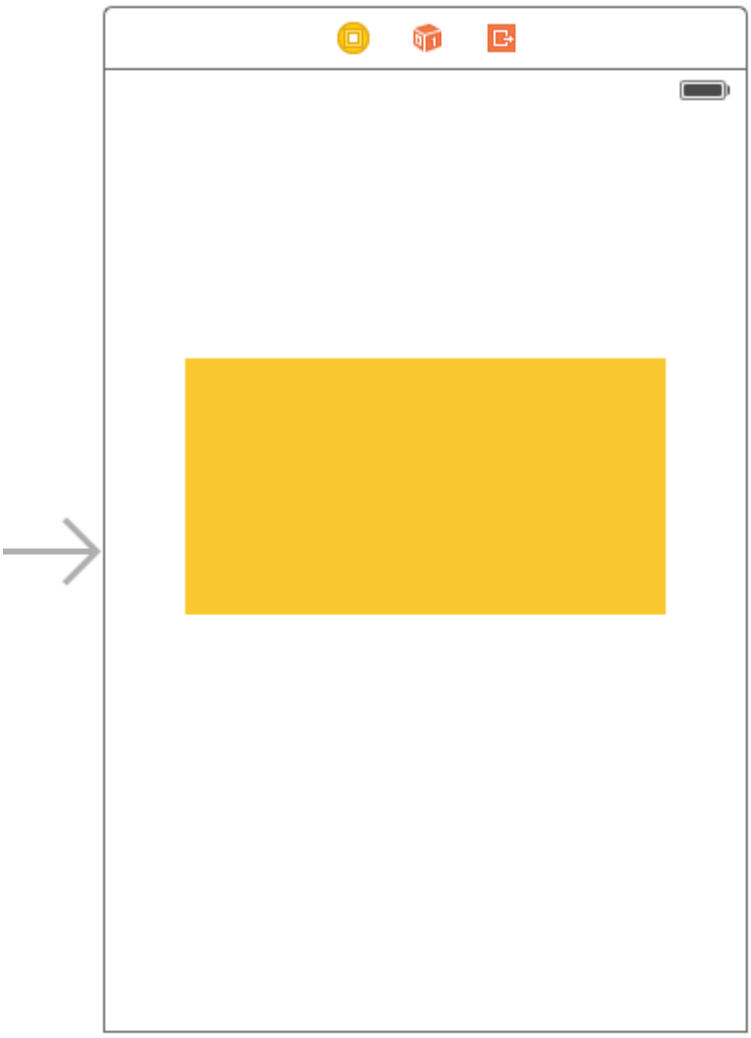
Si vous prenez en charge iOS 8 ou une version antérieure, la méthode la plus simple consiste à ajouter des vues factices masquées. Avec iOS 9, vous pouvez remplacer les vues factices par des guides de mise en page.

Remarque: Interface Builder ne prend pas encore en charge les guides de mise en page (Xcode 7.2.1). Donc, si vous voulez les utiliser, vous devez créer vos contraintes dans le code. Source

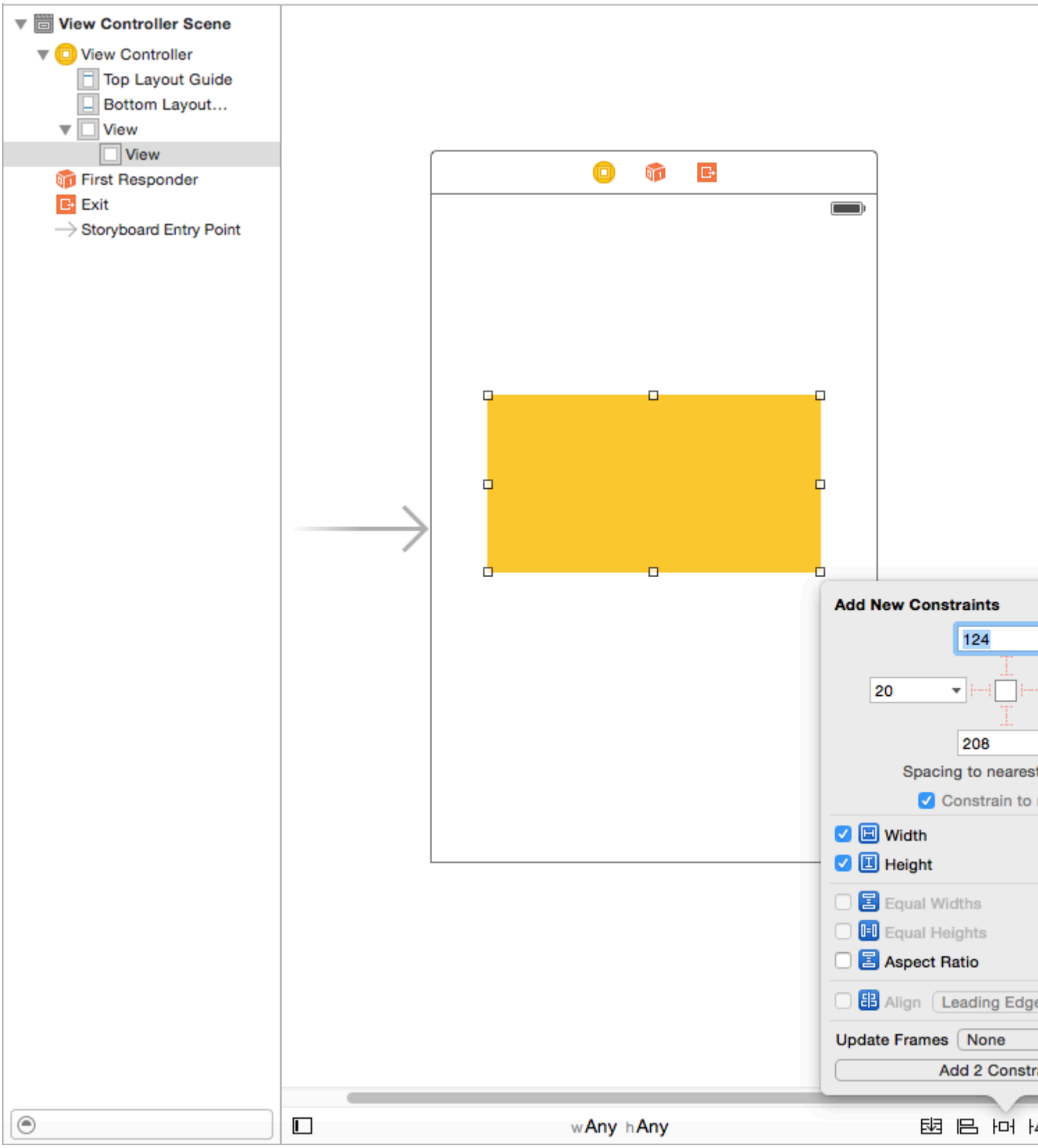
Taille intrinsèque UILabel

Nous devons créer une vue qui aura un préfixe d'image pour un texte. le texte peut être de longueur variable. Nous devons obtenir un résultat où Image + texte est toujours au centre d'une vue parent.

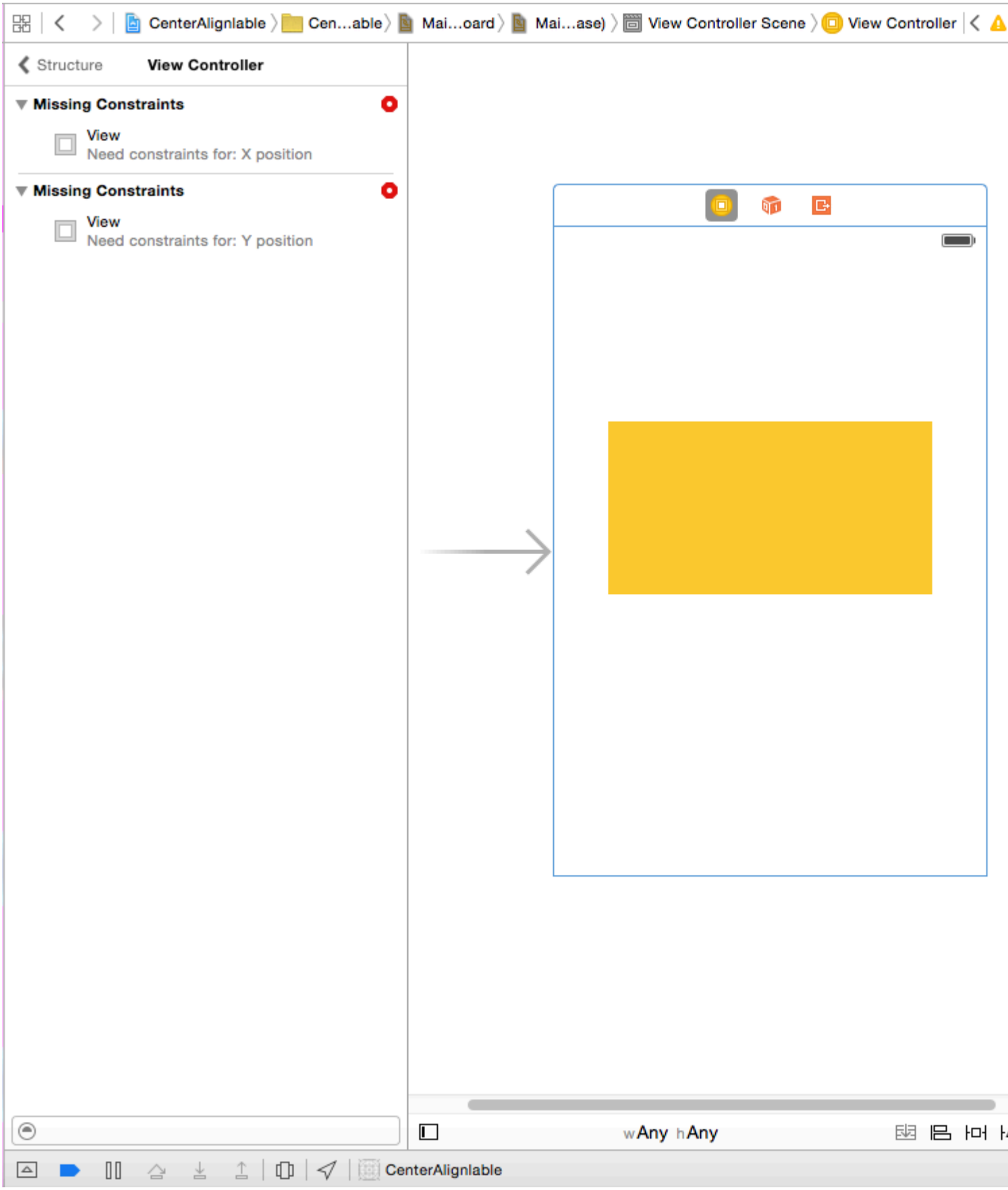
Etape 1: Créez d'abord un projet de vue unique et nommez-le comme vous le souhaitez et ouvrez la première vue du storyboard. Faites glisser une vue de taille raisonnable et définissez sa couleur d'arrière-plan sur jaune. vue devrait ressembler à quelque chose comme ça



Etape 2: Maintenant, nous allons ajouter des contraintes à la vue jaune. Pour commencer, nous allons ajouter des contraintes de largeur et de hauteur (Attendez une minute, n’avons-nous pas dit que cette vue aura une largeur dynamique? Ok, nous y reviendrons plus tard) les contraintes suivantes, comme dans l'image ci-dessous, ne se soucient pas de la valeur de largeur. Toute valeur est juste suffisante pour la largeur. Gardez-la juste assez grande pour que nous puissions ajouter des mises en page automatiques correctement.



Après avoir ajouté ces deux contraintes, vous verrez que XCode vous donne des erreurs, car l'image ci-dessous permet de les voir et de les comprendre.

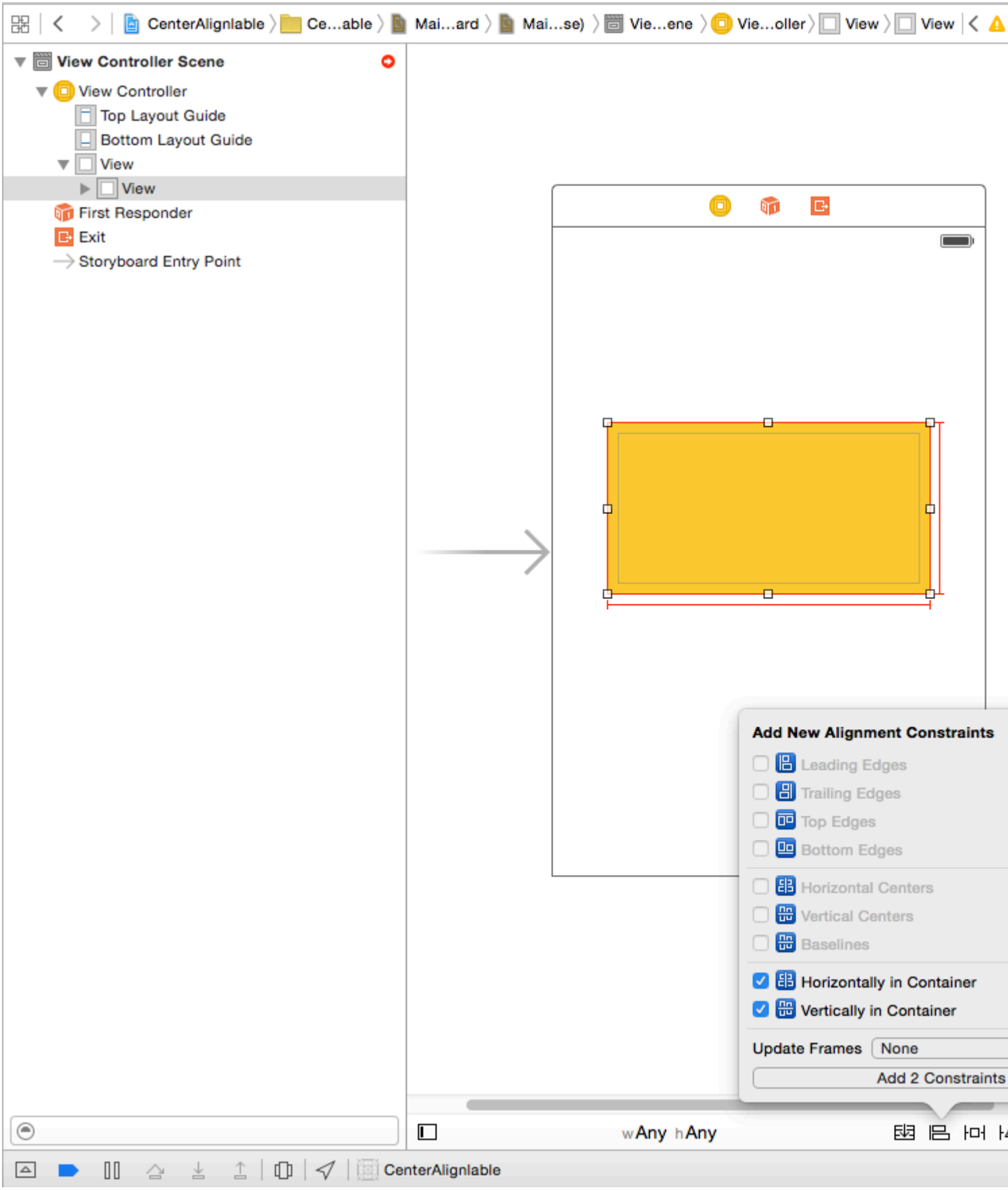


Nous avons deux erreurs (rouge signifie erreur) Comme indiqué ci-dessus, revisitons la partie ambiguïté

Contraintes manquantes: Contraintes nécessaires pour: Position X: - Comme indiqué ci-dessus, nous avons donné à la vue une largeur et une hauteur pour que ses «BOUNDS» soient définis,

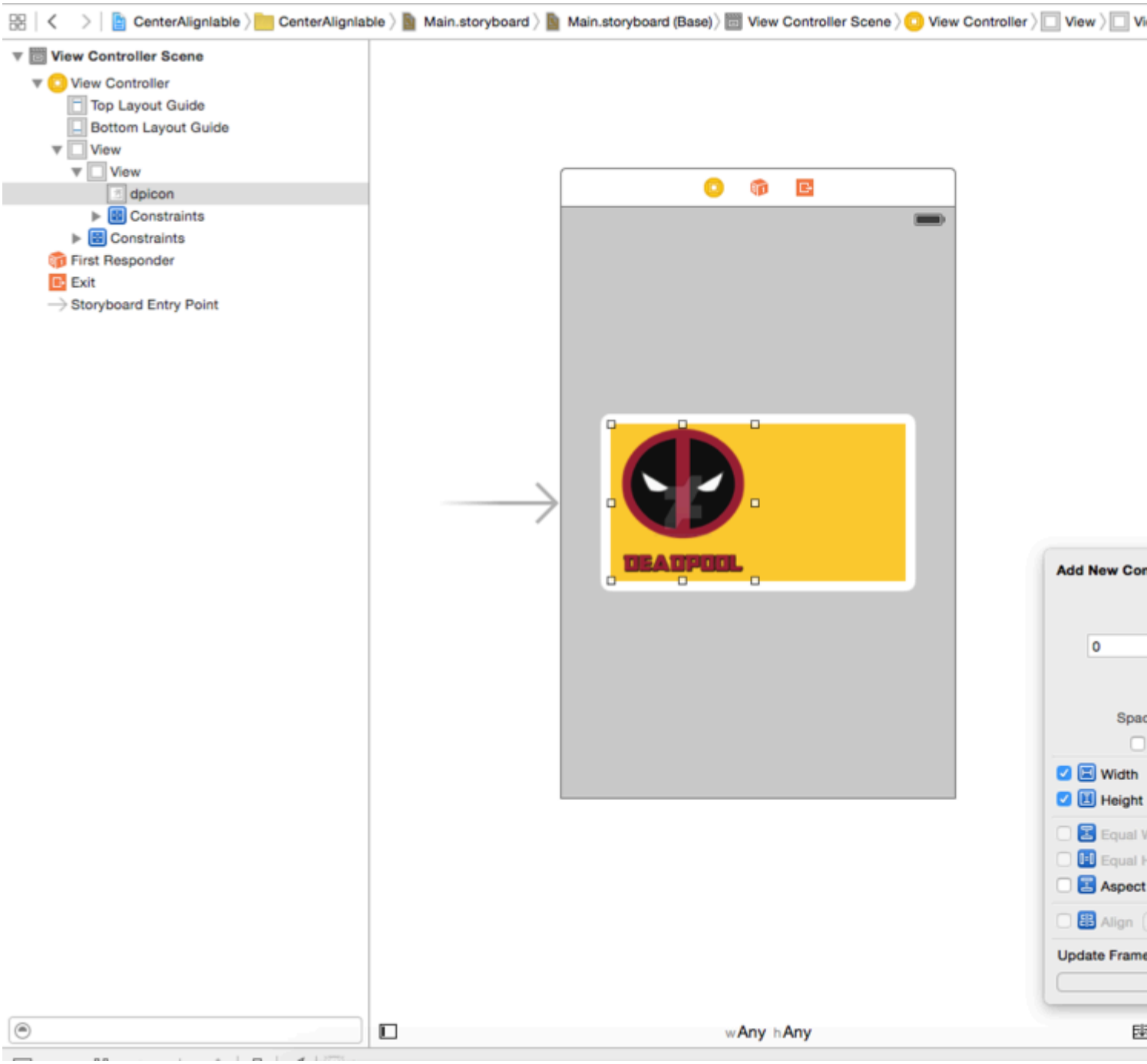
mais nous n'avons pas donné son origine pour que son «FRAME» ne soit pas défini. Autolayout n'est pas en mesure de déterminer quelle sera la position X de notre vue jaune

Contraintes manquantes: Contraintes nécessaires pour: Position Y: - Comme nous l'avons vu plus haut, nous avons donné à la vue une largeur et une hauteur afin de définir ses «BOUNDS», mais nous n'avons pas défini son origine. Autolayout n'est pas en mesure de déterminer quelle sera la position Y de notre vue jaune. Pour résoudre ce problème, nous devons attribuer à autolayout des fonctions de résolution X et Y. Comme nous ne pouvons pas définir de cadres, nous les ferons automatiquement. image ci-dessous, je l'expliquerai plus tard



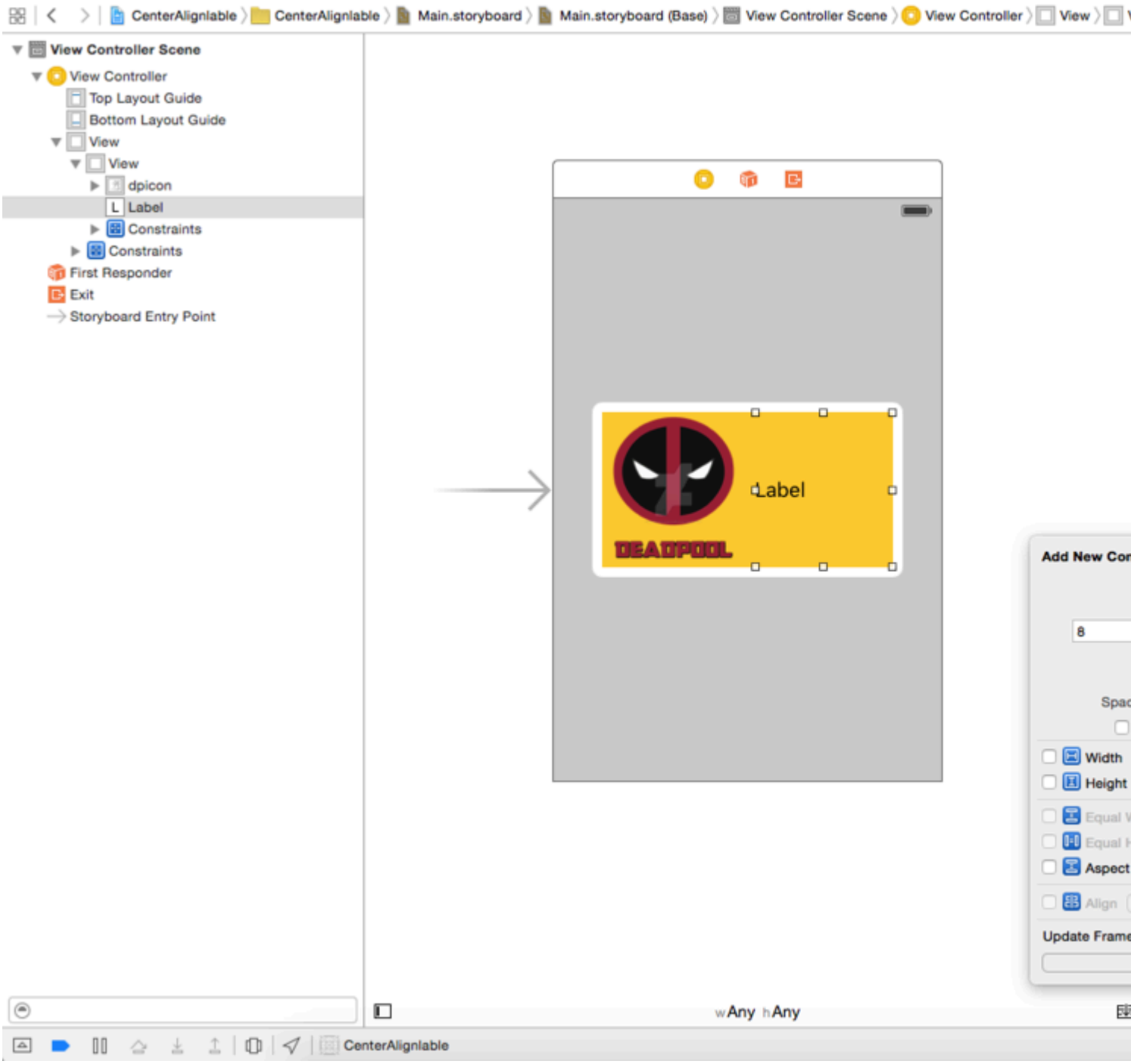
Ce que nous avons fait, c'est que nous avons ajouté un «centre vertical» et un «centre horizontal». Ces contraintes indiquent à autolayout que notre vue jaune sera toujours au centre. Horizontalement: X est déterminé avec la contrainte verticale et Y déterminé. pourrait avoir à ajuster le cadre).

Étape 3: Notre vue jaune de base est maintenant prête. Nous allons ajouter l'image de préfixe comme sous-vue de notre vue jaune avec les contraintes suivantes. Vous pouvez choisir n'importe quelle image de votre choix.



Comme nous avons une dimension fixe pour notre image de préfixe, nous aurons une hauteur de largeur fixe pour cette image. Ajoutez les contraintes et passez à l'étape suivante.

Etape 4: Ajoutez un UILabel comme sous-vue de notre vue jaune et ajoutez les contraintes suivantes

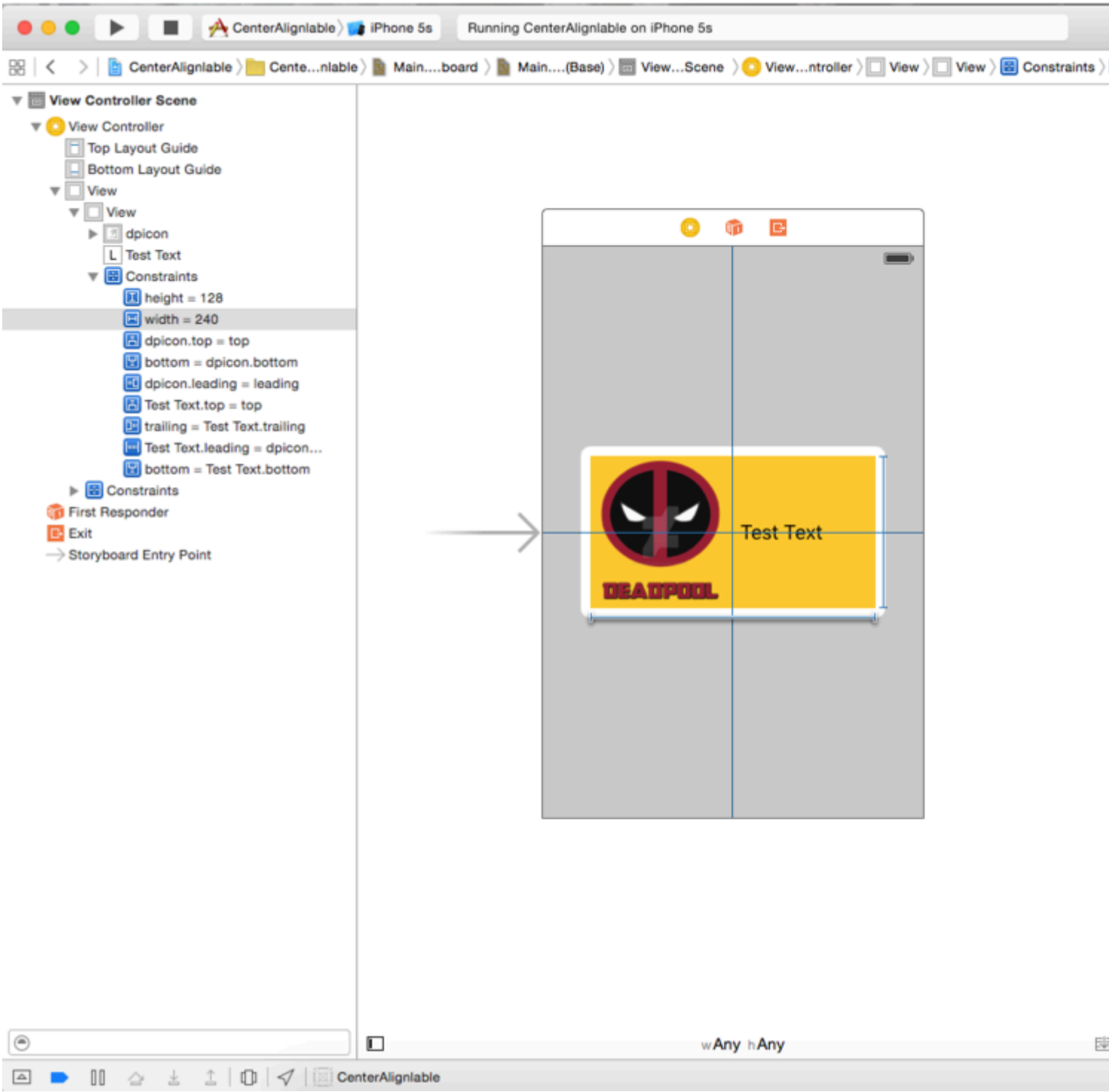


Comme vous pouvez le voir, j'ai donné seulement des contraintes relatives à notre UILabel.Son 8 points de l'image de préfixe et 0,0,0 en haut et en bas de la vue jaune.Comme nous voulons que la largeur soit dynamique, nous ne donnerons pas de contraintes de largeur ou de hauteur .

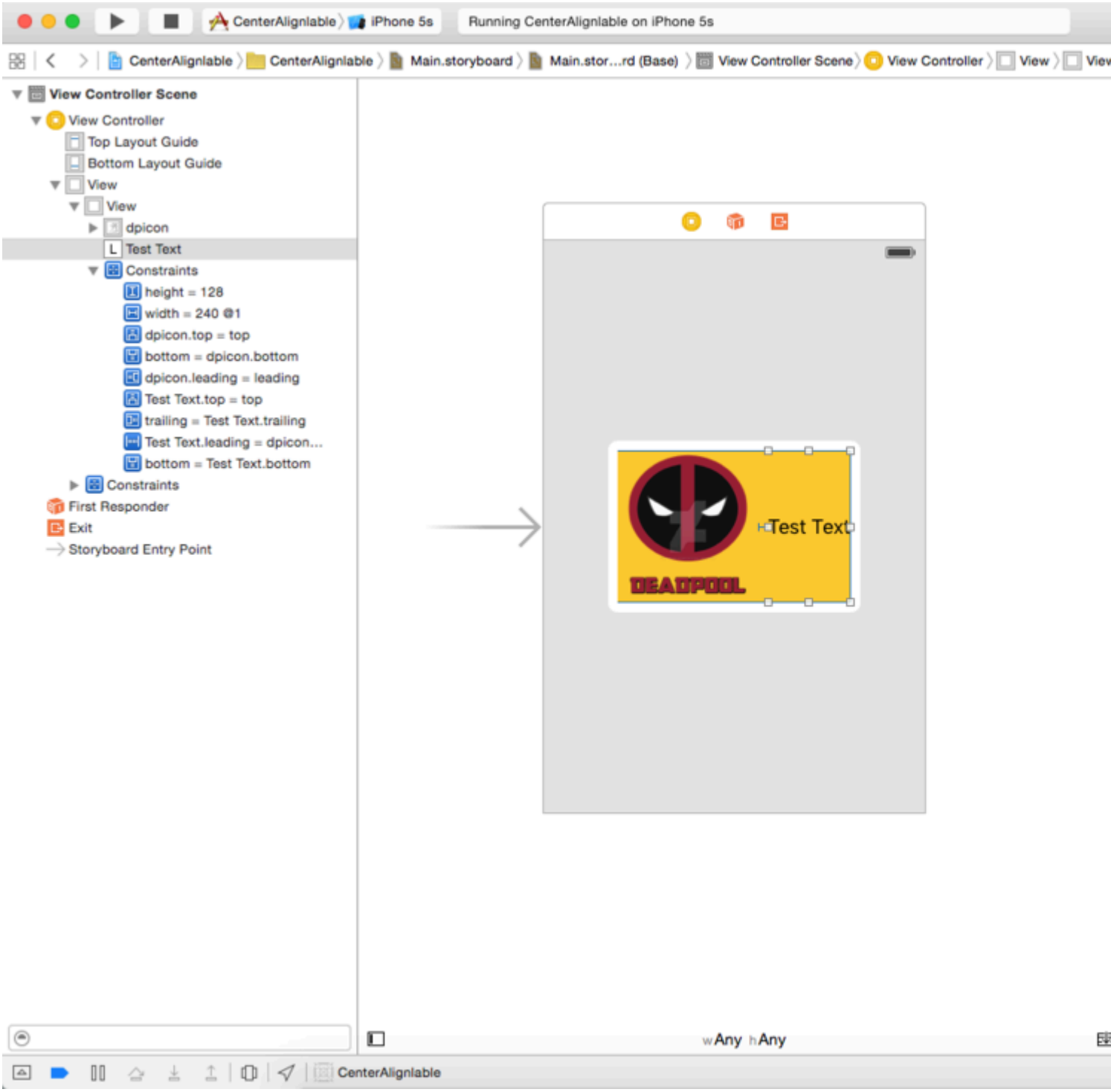
Q: Pourquoi nous ne recevons aucune erreur maintenant, nous n'avons pas donné de largeur et de hauteur? Réponse: - Nous obtenons une erreur ou un avertissement uniquement lorsque la mise en page automatique n'est pas en mesure de résoudre quelque chose qui est nécessaire pour rendre une vue à l'écran.Avec sa hauteur, sa largeur ou son origine. est bien défini autolayout est capable de calculer le cadre de notre étiquette.

Etape 5: Maintenant, si nous nous rappelons, nous nous rendrons compte que nous avons donné une vue fixe à la vue jaune mais nous voulons qu'elle soit dynamique en fonction du texte de notre étiquette. Nous modifierons donc notre contrainte de largeur de la vue jaune. est nécessaire pour

résoudre l'ambiguïté, mais nous voulons qu'il soit annulé à l'exécution en fonction du contenu de UILabel. Nous allons donc sélectionner notre vue jaune et aller dans l'inspecteur Taille et réduire la priorité de la contrainte de largeur à 1 pour qu'elle soit annulée. Suivez l'image ci-dessous.

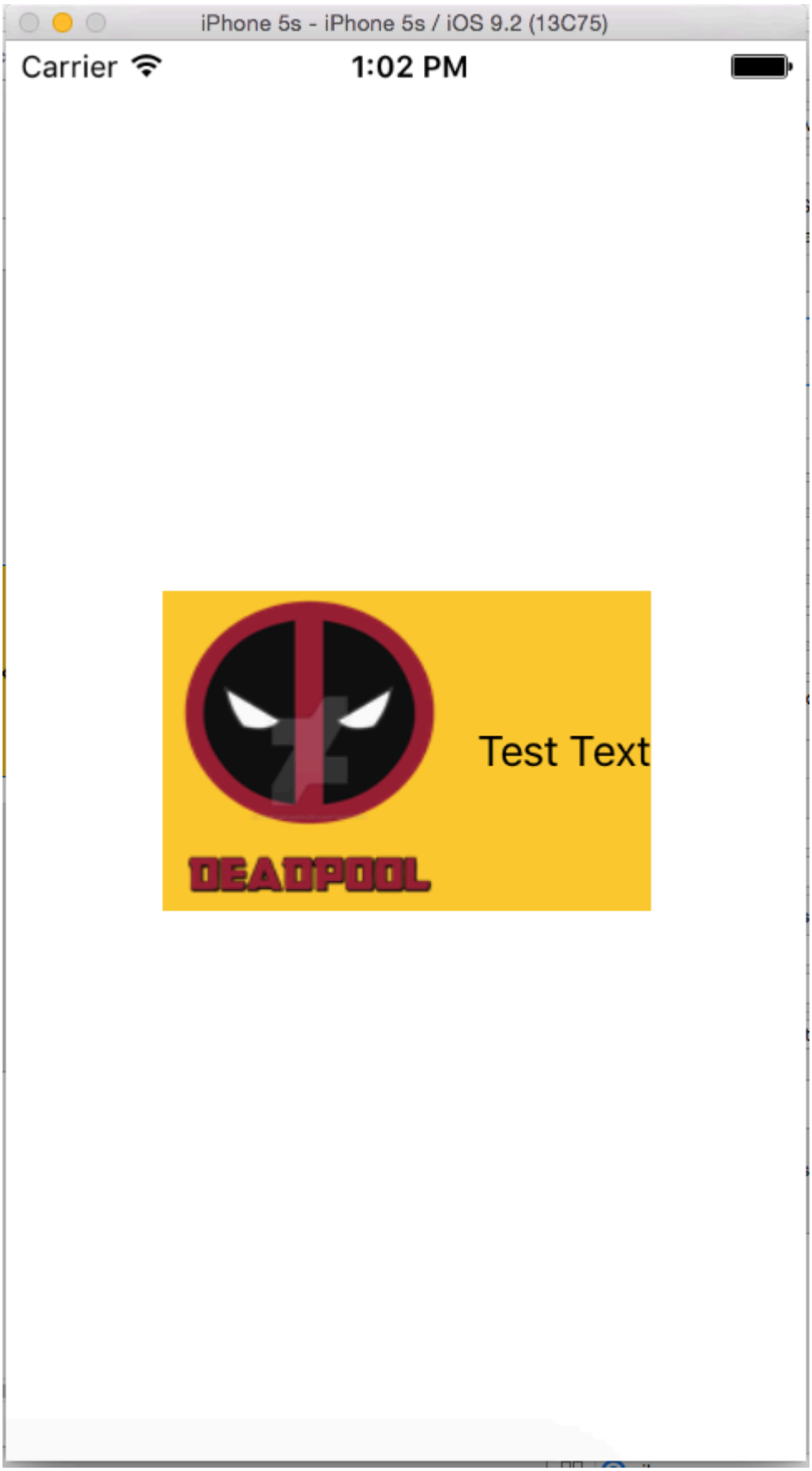


Etape 6: Nous voulons que UILabel se développe en fonction du texte et repousse notre vue en jaune.Ainsi, nous avons réduit la priorité de la largeur de la vue en jaune.Maintenant, nous allons augmenter la priorité de la résistance de compression de texte de notre UILabel. ainsi nous augmenterons la priorité de la couverture de contenu d'UILabel. Suivez l'image ci-dessous



Comme vous pouvez le constater, nous avons augmenté la priorité de mise en cache du contenu à 500 et la priorité de la résistance de compression à 751, ce qui permettra de surpasser la priorité 1 de la contrainte de largeur.

Maintenant, construisez et exécutez, vous verrez quelque chose comme suit.



Comment animer avec la mise en page automatique

Sans mise en forme automatique, l'animation est effectuée en modifiant l'image d'une vue au fil du temps. Avec la disposition automatique, les contraintes dictent le cadre de la vue, vous devez

donc animer les contraintes à la place. Cette indirection rend l'animation plus difficile à visualiser.

Voici comment animer avec la mise en page automatique:

1. **Modifiez la constante de la contrainte** après la création à l'aide d'appels périodiques (`CADisplayLink` , `dispatch_source_t` , `dispatch_after` , `NSTimer`). Appelez ensuite `layoutIfNeeded` pour mettre à jour la contrainte. Exemple:

Objectif c:

```
self.someConstraint.constant = 10.0;
[UIView animateWithDuration:0.25 animations:^{
    [self.view layoutIfNeeded];
}];
```

Rapide:

```
self.someConstraint.constant = 10.0
UIView.animate(withDuration: 0.25, animations: self.view.layoutIfNeeded)
```

2. **Modifiez les contraintes** et appelez `[view layoutIfNeeded]` dans un bloc d'animation. Cela interpole entre les deux positions en ignorant les contraintes pendant l'animation.

```
[UIView animateWithDuration:0.5 animations:^{
    [view layoutIfNeeded];
}]
```

3. **Changer la priorité des contraintes** . Cela consomme moins de ressources processeur que l'ajout et la suppression de contraintes.

4. **Supprimez toutes les contraintes et utilisez des masques autosizing** . Pour la suite, vous devez définir `view.translatesAutoresizingMaskIntoConstraints = YES` .

5. **Utilisez des contraintes qui n'interfèrent pas avec l'animation voulue** .

6. **Utilisez une vue de conteneur** . Positionnez la superview à l'aide de contraintes. Ajoutez ensuite une sous-vue avec des contraintes qui ne combattent pas l'animation, par exemple un centre par rapport à la superview. Cela décharge une partie des contraintes à la superview, afin qu'elles ne combattent pas l'animation dans la sous-vue.

7. **Animez les calques à la place des vues** . Les transformations de couche ne déclenchent pas la mise en forme automatique.

```
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. **Remplacez layoutSubviews** . Appelez `[super layoutSubviews]` et affinez les contraintes.

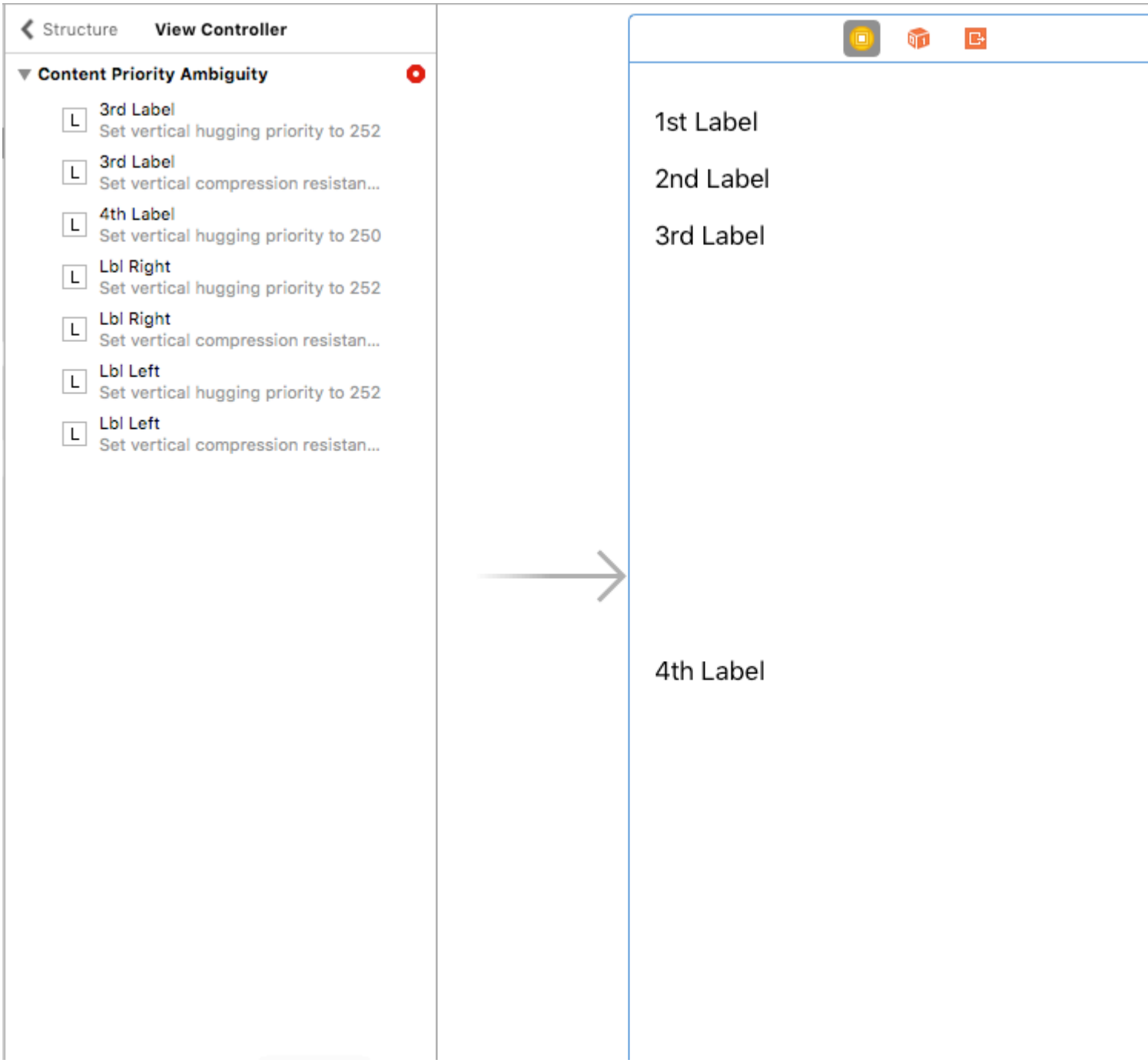
9. **Modifier le cadre dans viewDidLayoutSubviews** . La mise en forme automatique est appliquée dans `layoutSubviews` . Une fois cette opération terminée, modifiez-la dans `viewDidLayoutSubviews` .

10. **Désactivez l'option Mise en forme automatique** et définissez les vues manuellement. Vous pouvez faire cela en `layoutSubviews` / `layout` sans appeler l'implémentation de la super classe.

Astuce: si le parent de la vue animée n'est pas interpolé (c'est-à-dire que l'animation saute du début à la fin), appelez `layoutIfNeeded()` dans la vue la plus profonde qui est le parent de la vue animée (en d'autres termes: , qui n'est pas affecté par l'animation). Je ne sais pas exactement pourquoi cela fonctionne.

Résoudre les conflits de priorité de l'étiquette UILabel

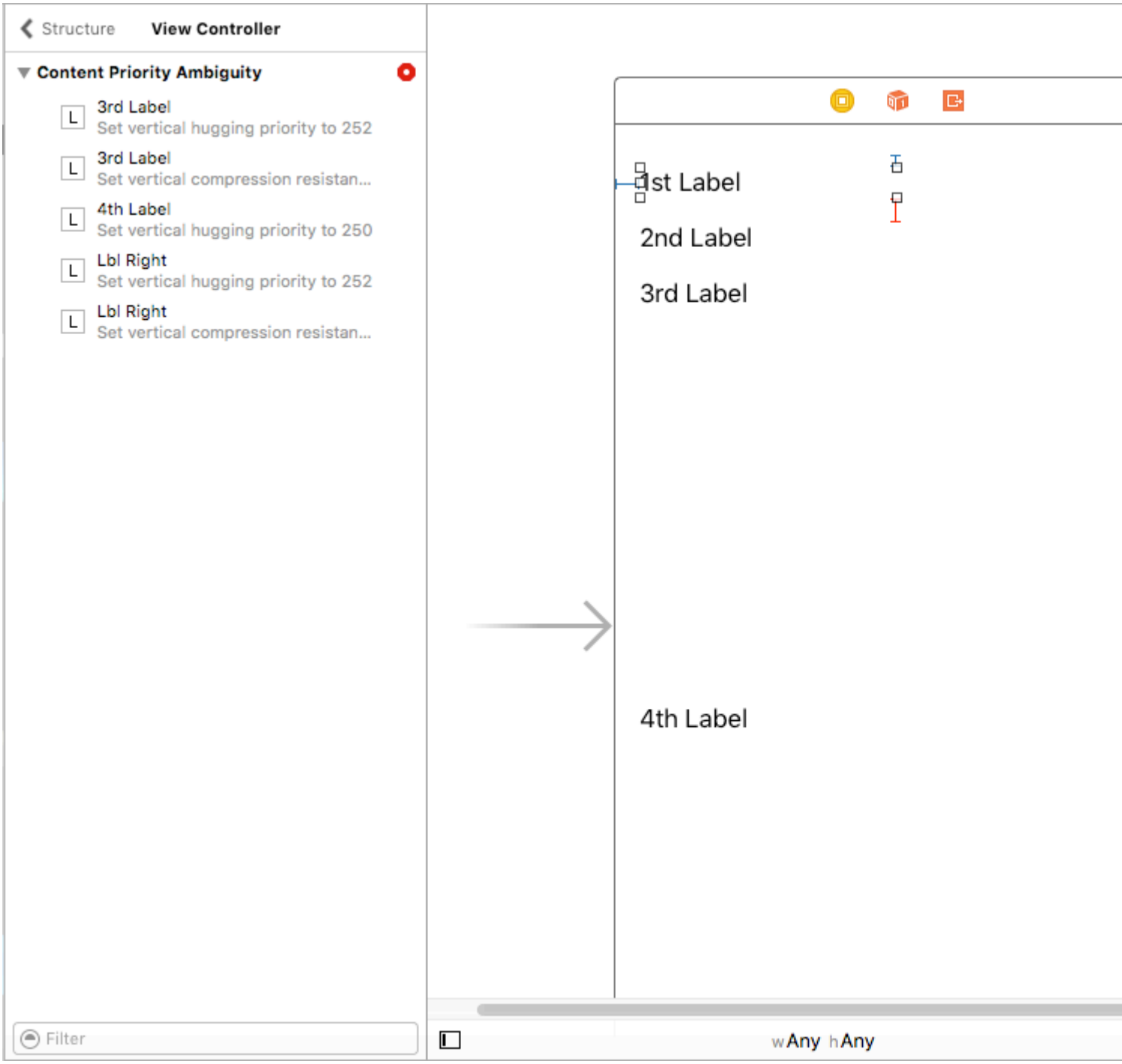
Problème : lorsque vous utilisez plusieurs étiquettes dans une vue, vous obtenez peut-être un **avertissement** :

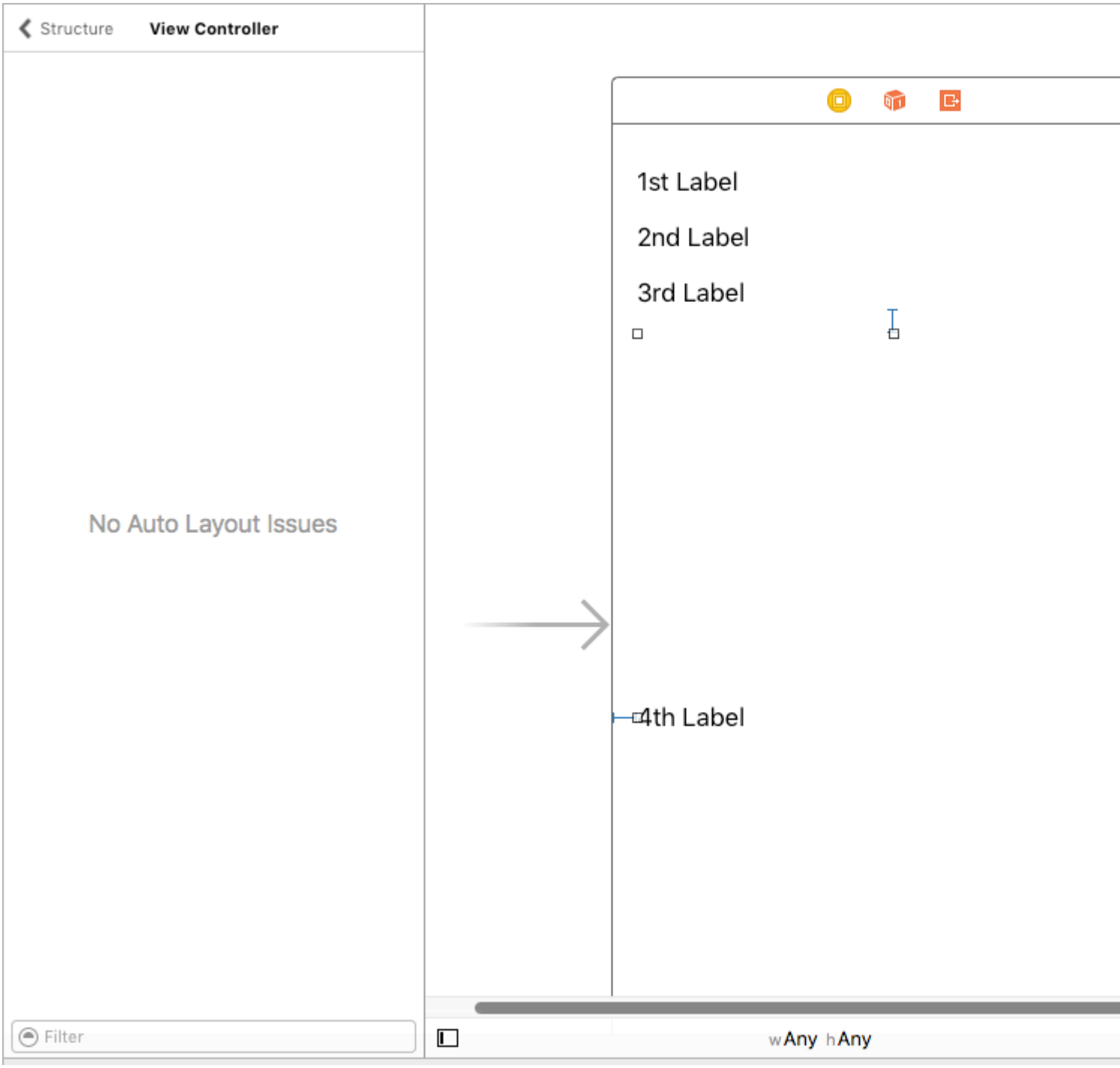


Comment pouvons-nous corriger cet **avertissement** ?

Solution : Nous calculons et définissons les priorités dans l'ordre. Les priorités doivent être différentes des étiquettes. Cela signifie que ce qui est important obtiendra une priorité plus élevée. Par exemple, dans mon cas, j'ai défini les priorités verticales de mes étiquettes comme suit:

J'ai défini la plus haute priorité pour la 1ère étiquette et la plus basse pour la 4ème.


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



Dans un ViewController, je pense que vous avez du mal à voir l'effet de ces priorités. Cependant, il est très clair avec UITableViewCell + estimer la hauteur de la cellule.

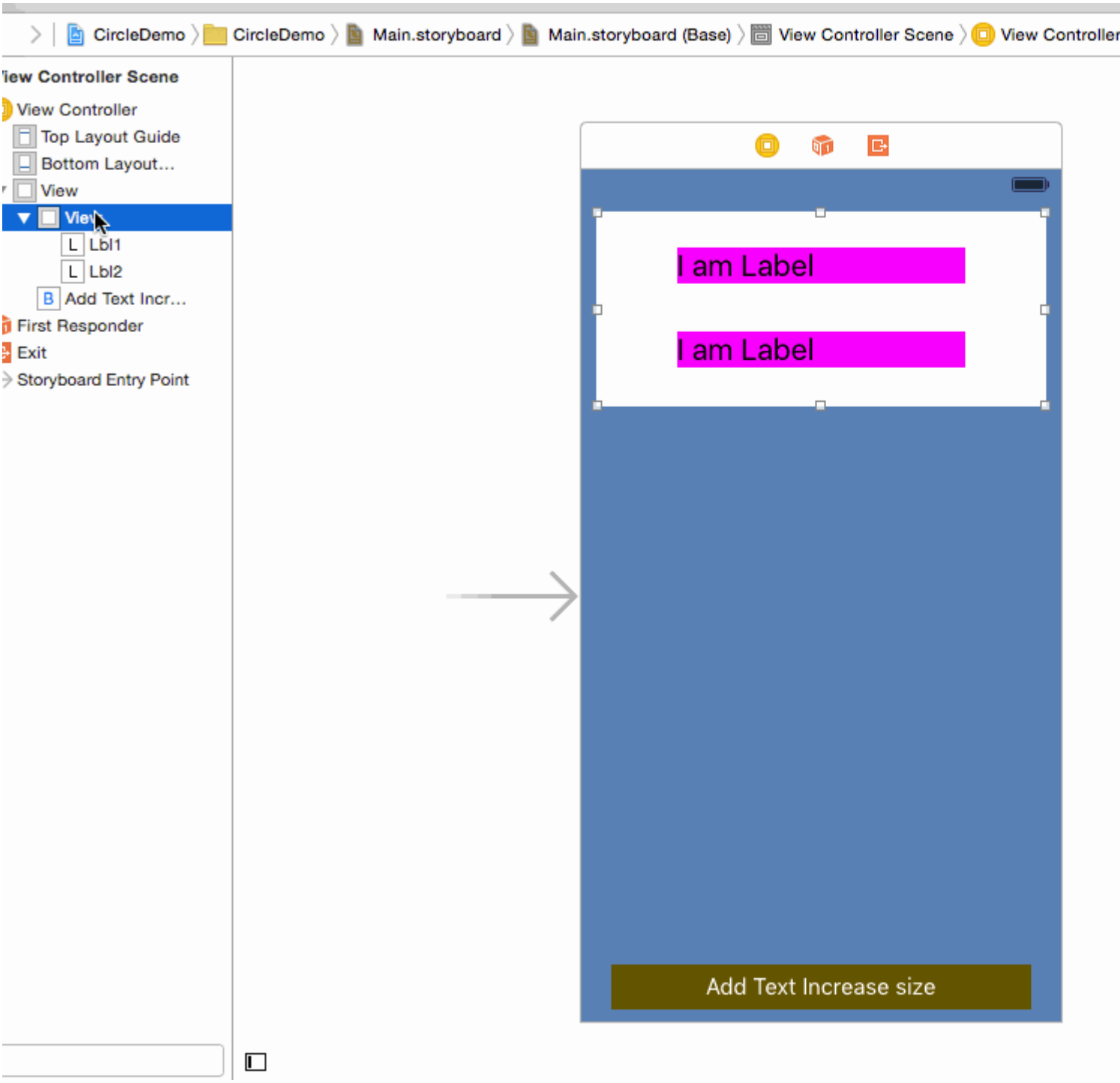
J'espère que cette aide

Taille UILabel & Parentview selon le texte dans UILabel

Guide étape par étape: -

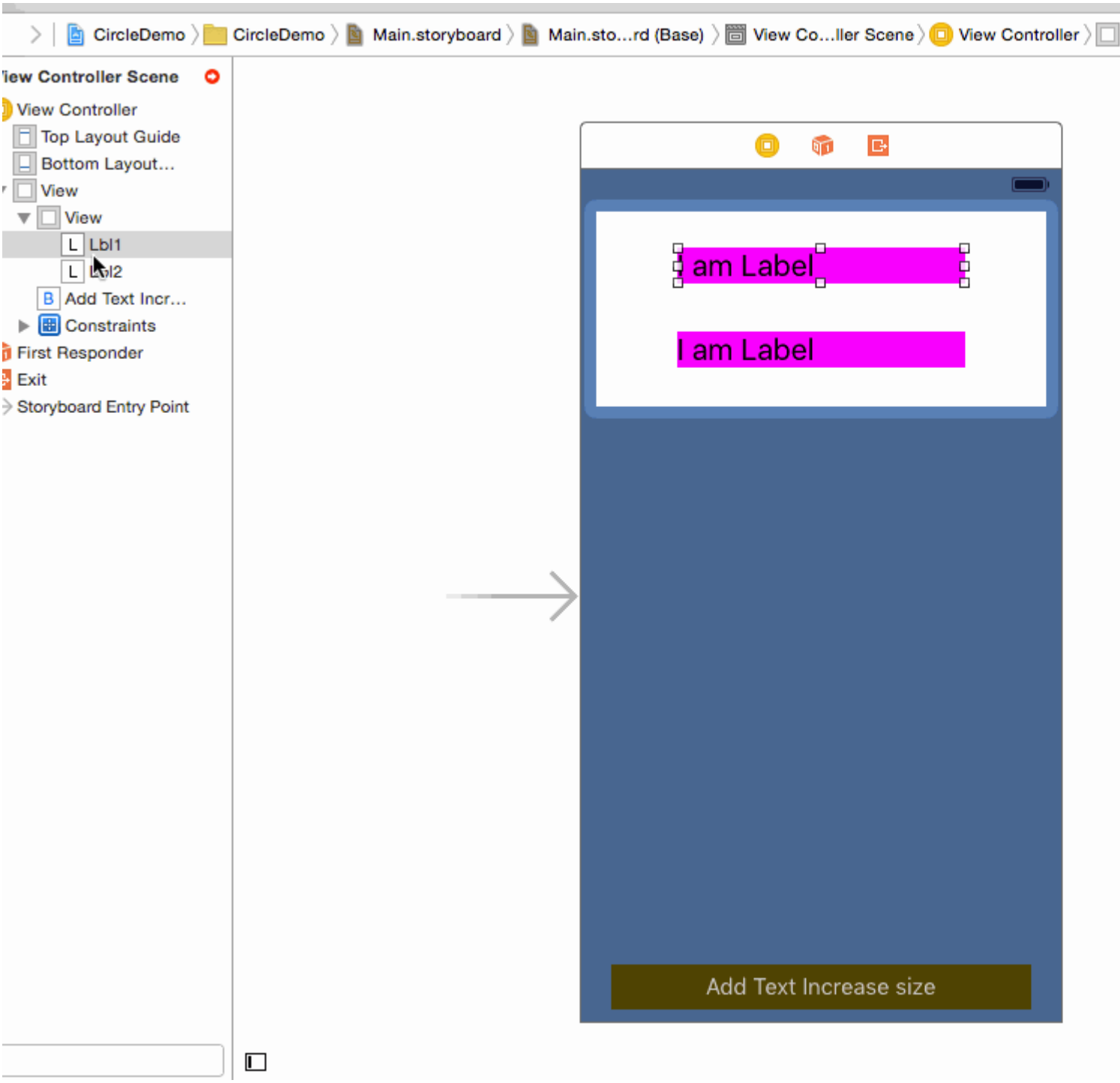
Etape 1: - Définir la contrainte sur UIView

1. De premier plan. 2) en haut. 3) Trailing. (De mainview)



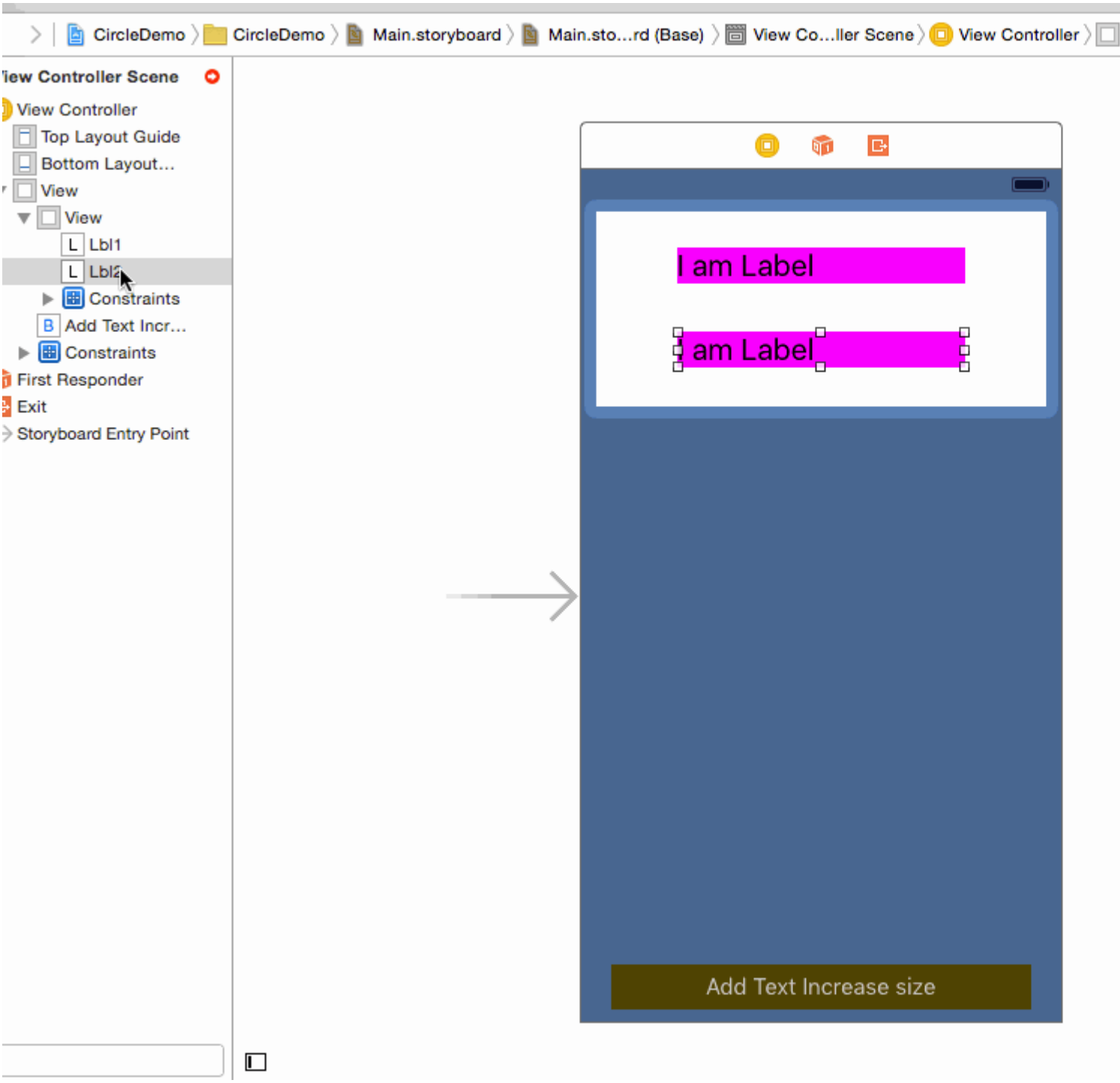
Étape 2: - Définir la contrainte sur l'étiquette 1

1. Leading 2) Top 3) Trailing (De sa superview)

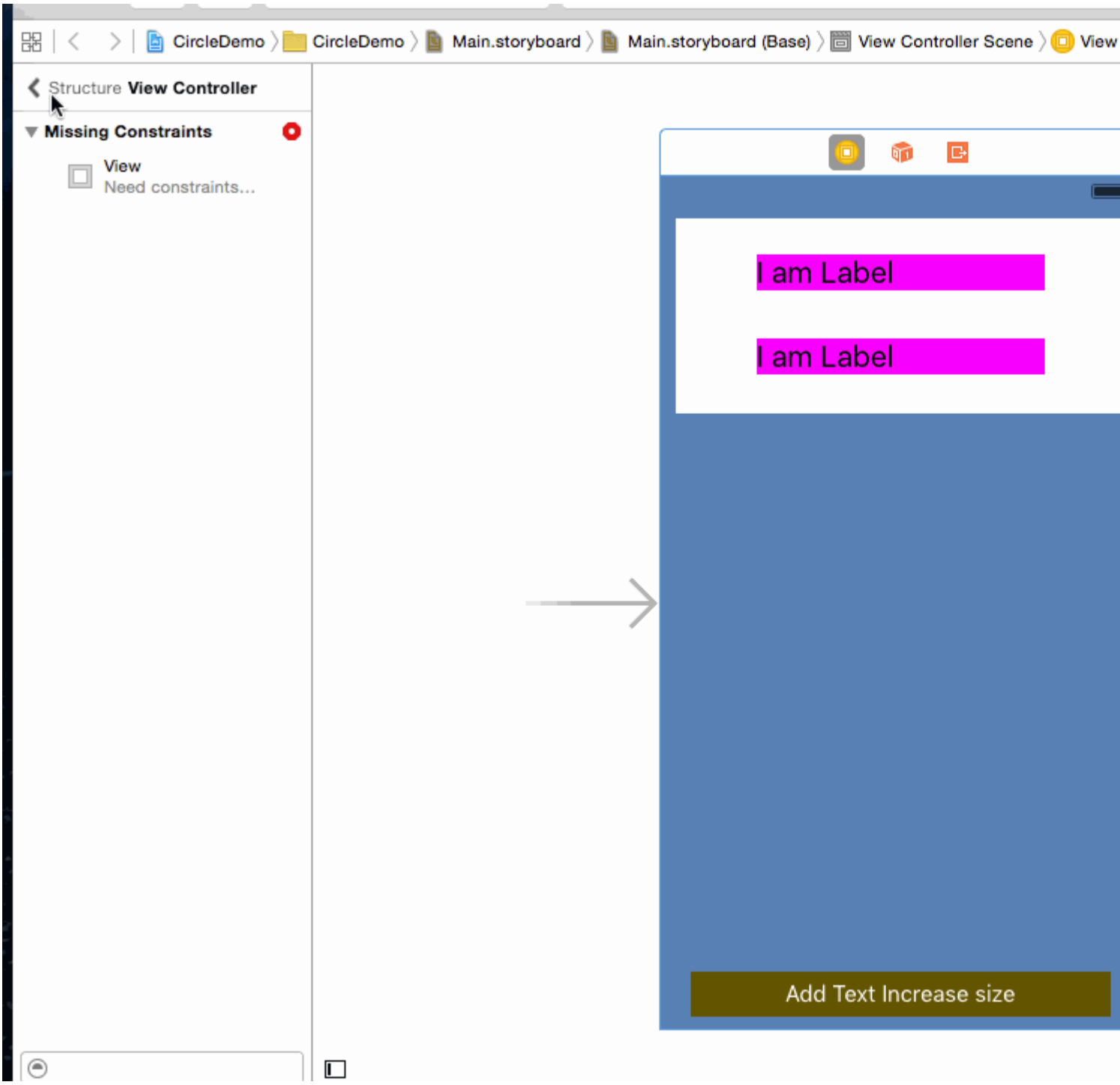


Étape 3: - Définir la contrainte sur Label 2

1. Leading 2) Top 3) Trailing (De sa superview)

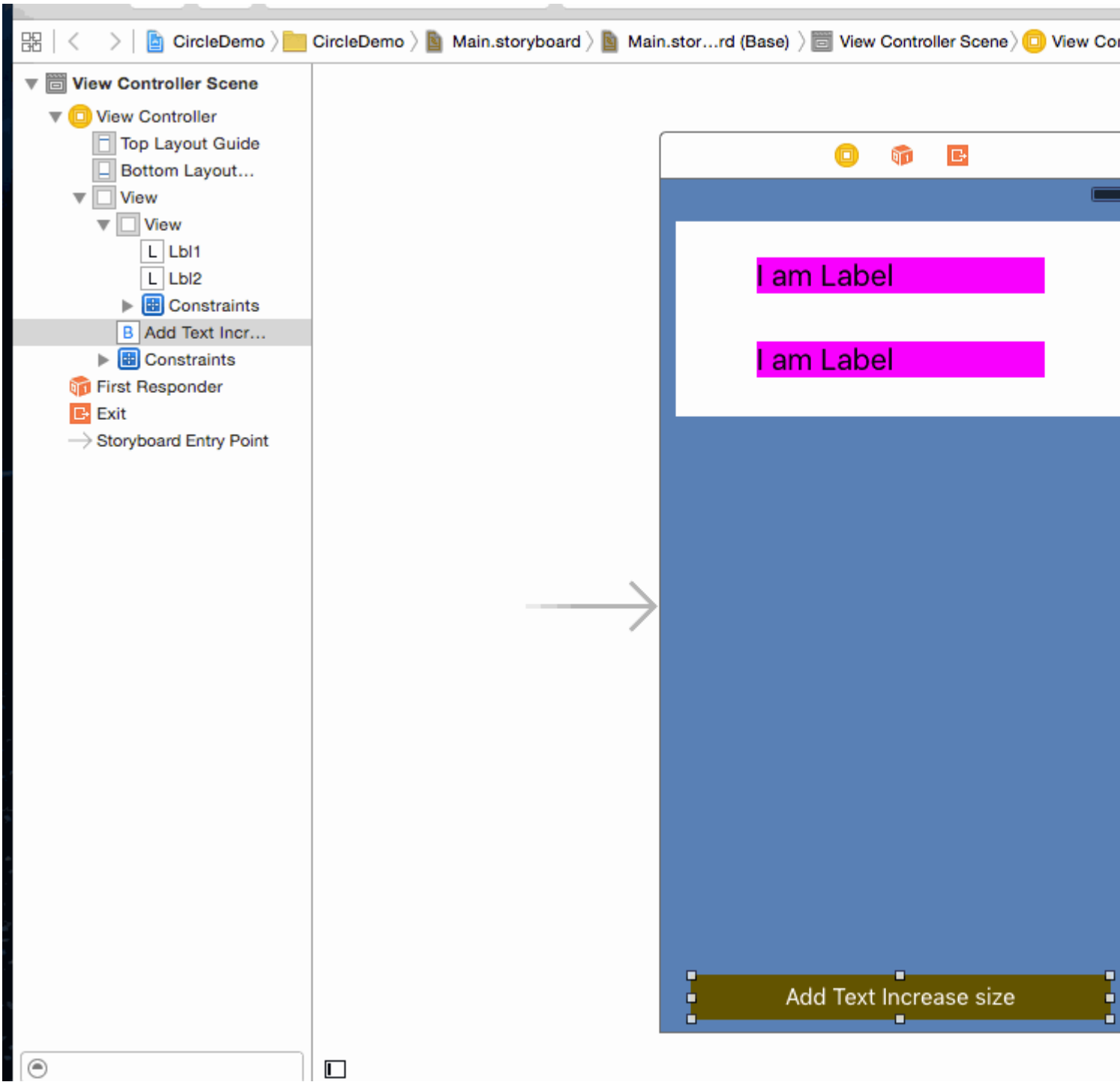


Etape 4: - Le plus difficile consiste à donner un bas à UILabel depuis UIView.



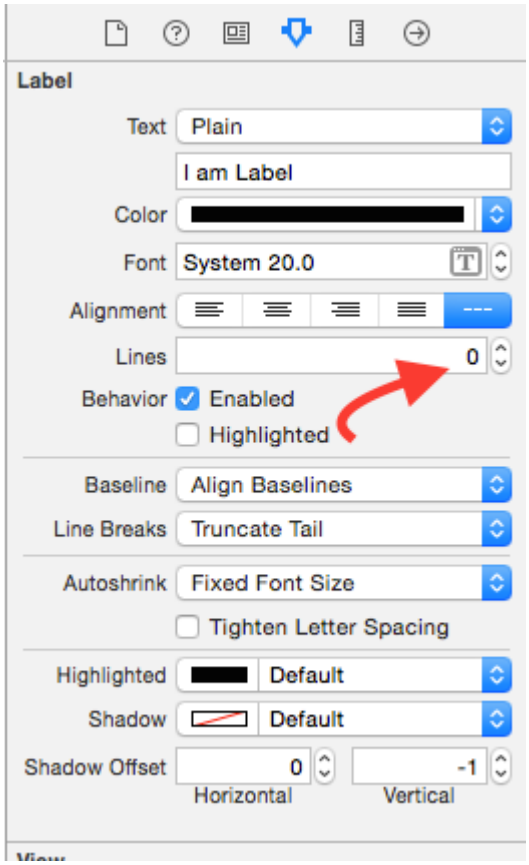
Étape 5: - (Facultatif) Définissez la contrainte sur UIButton

1. En tête 2) En bas 3) Trailing 4) Hauteur fixe (de la vue principale)


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

Sortie: -

Remarque: - Assurez-vous d'avoir défini le nombre de lignes = 0 dans la propriété Label.



J'espère que cette information assez pour comprendre Autoresize UIView selon la hauteur de UILabel et Autoresize UILabel Selon le texte.

Visual Basic Basics: contraintes dans le code!

HVFL est un langage conçu pour contraindre les éléments de l'interface utilisateur de manière simple et rapide. En général, VFL a un avantage sur la personnalisation de l'interface utilisateur traditionnelle dans Interface Builder, car il est beaucoup plus lisible, accessible et compact.

Voici un exemple de VFL, dans lequel trois UIViews sont contraintes de gauche à droite, remplissant `superView.width` , avec `aGradeView`

```
"H:|[bgView][aGradeView(40)][bGradeView(40)]|"
```

Il y a deux axes dans lesquels on peut contraindre les objets d'interface utilisateur, horizontalement et verticalement.

Chaque ligne de VFL commence toujours par `H:` ou `V:` Si aucun n'est présent, l'option par défaut est `H:`

En passant, nous avons un pipeline. `|` Ce symbole, ou le tuyau, fait référence à la superview. Si vous regardez de plus près l'extrait de code VFL ci-dessus, vous remarquerez deux de ces pipelines.

Cela signifie les deux extrémités horizontales de la superview, les limites extérieures et extérieures.

Ensuite, vous verrez des crochets, dans le premier ensemble de crochets, nous avons `bgView` . Quand nous avons des crochets, cela fait référence à un élément d'interface utilisateur, maintenant vous pourriez vous demander comment établir un lien entre le nom et l'élément d'interface utilisateur réel, un exutoire peut-être?

Je couvrirai cela à la fin du post.

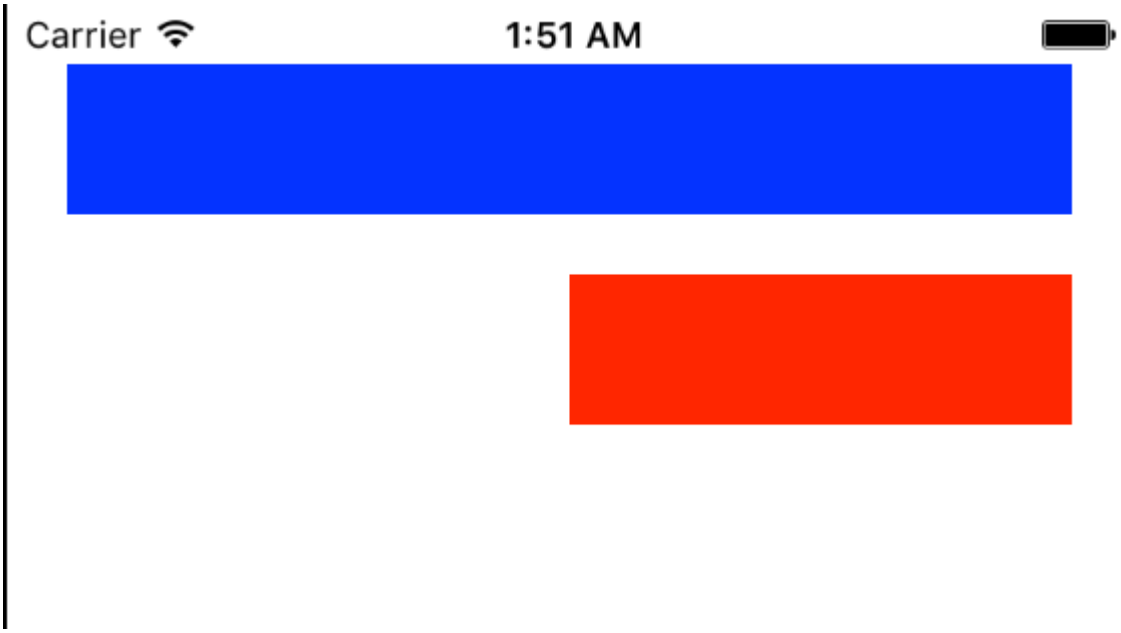
Si vous regardez la deuxième paire de crochets `[aGradeView(50)]` , nous avons des parenthèses encapsulées à l'intérieur, lorsque cela est présent, il définit la largeur / hauteur en fonction des axes, qui est dans ce cas de 50 pixels en largeur.

Les premiers crochets `[bgView]` n'ont pas de largeur explicitement définie, ce qui signifie qu'ils vont s'étendre le plus possible.

Bon, c'est tout pour les bases, plus sur les choses avancées dans un autre exemple.

---

par exemple:




```
    // 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];
```

Utilisation mixte de la mise en page automatique avec une mise en page non automatique

Parfois, vous pouvez vouloir effectuer des actions supplémentaires pour les calculs de **mise en page automatique** effectués par `UIKit` lui-même.

Exemple: lorsque vous avez un `UIView` avec un `maskLayer` , vous devrez peut-être mettre à jour `maskLayer` dès que **Auto Layout** modifie le `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;
    ...
}
```

ou si vous souhaitez prendre des mesures supplémentaires pour la **mise** en `ViewController` **automatique** dans `ViewController`

```
- (void)viewDidLayoutSubviews {
    [super viewDidLayoutSubviews];
    // now you can assume all your subviews are positioned/resized correctly
    self.customView.frame = self.containerView.frame;
}
```

Disposition proportionnelle

Contrainte créée comme

```
NSLayoutConstraint(item: myView, attribute: NSLayoutAttribute.Leading, relatedBy:
NSLayoutRelation.Equal, toItem: view, attribute: NSLayoutAttribute.LeadingMargin, multiplier:
1.0, constant: 20.0)
```

ou, du point de vue mathématique:

```
view.attribute * multiplier + constant          (1)
```

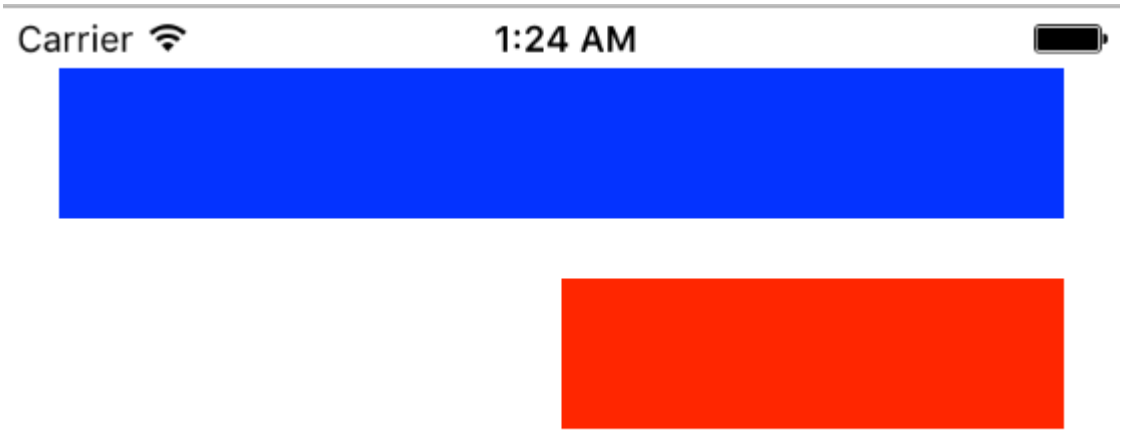
Vous pouvez utiliser le multiplicateur pour créer une disposition proportionnelle pour différents facteurs de taille.

Exemple:

Turquoise View (V1) est un carré dont la largeur est proportionnelle à la largeur de la vue avec un ratio de 1: 1,1

Gary square (V2) est une sous-vue de V1. Espace inférieur défini par la constante = 60, espace de fin défini par le multiplicateur = 1,125 et constante = 0

L'espace de fin est défini proportionnellement, l'espace inférieur défini comme constante.



- Objectif 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];
```

Lire Mise en page automatique en ligne: https://riptutorial.com/fr/ios/topic/792/mise-en-page-automatique

Chapitre 95: MKDistanceFormatter

Examples

Ficelle de distance

Étant donné une `CLLocationDistance` (simplement un `Double` représentant des mètres), `CLLocationDistance` une chaîne lisible par l'utilisateur:

```
let distance = CLLocationDistance(42)
let formatter = MKDistanceFormatter()
let answer = formatter.stringFromDistance(distance)
// answer = "150 feet"
```

Objectif c

```
CLLocationDistance distance=42;
MKDistanceFormatter *formatter=[[MKDistanceFormatter alloc]init];
NSString *answer=[formatter stringFromDistance:distance];
// answer = "150 feet"
```

Par défaut, cela respecte les paramètres régionaux de l'utilisateur.

Unités de distance

`import Mapkit units` Set de `import Mapkit` vers l'une des `.Default, .Metric, .Imperial, .ImperialWithYards` :

```
formatter.units = .Metric
var answer = formatter.stringFromDistance(distance)
// "40 m"

formatter.units = .ImperialWithYards
answer = formatter.stringFromDistance(distance)
// "50 yards"
```

Objectif 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
```

Style unitaire

Définissez `unitStyle` sur l'un des `.Default, .Abbreviated, .Full` :

```
formatter.unitStyle = .Full
var answer = formatter.stringFromDistance(distance)
// "150 feet"

formatter.unitStyle = .Abbreviated
answer = formatter.stringFromDistance(distance)
// "150 ft"
```

Objectif c

```
formatter.unitStyle=MKDistanceFormatterUnitStyleFull;
NSString *answer=[formatter stringFromDistance:distance];
// "150 feet"

 formatter.unitStyle=MKDistanceFormatterUnitStyleAbbreviated;
 NSString *answer=[formatter stringFromDistance:distance];
// "150 ft"
```

Lire MKDistanceFormatter en ligne: https://riptutorial.com/fr/ios/topic/6677/mkdistanceformatter

Chapitre 96: MKMapView

Examples

Ajouter MKMapView

Rapide

```
let mapView = MKMapView(frame: CGRect(x: 0, y: 0, width: 320, height: 500))
```

Il est recommandé de stocker mapView en tant que propriété du `ViewController` car vous souhaiterez peut-être y accéder dans des implémentations plus complexes.

Objectif c

```
self.map = [[MKMapView alloc]initWithFrame:CGRectMake(0, 0, self.view.frame.size.width,
self.view.frame.size.height)];
[self.view addSubview:self.map];
```

Changer le type de carte

Il existe 5 types différents (`MKMapType`), `MKMapView` peut afficher.

iPhone OS 3

.la norme

Affiche un plan de ville indiquant la position de toutes les routes et certains noms de route.

Swift 2

```
mapView.mapType = .Standard
```

Swift 3

```
mapView.mapType = .standard
```

Objectif c

```
_mapView.mapType = MKMapTypeStandard;
```

iPhone OS 3

.Satellite

Affiche des images satellites de la zone.

Swift 2

```
mapView.mapType = .Satellite
```

Swift 3

```
mapView.mapType = .satellite
```

Objectif c

```
_mapView.mapType = MKMapTypeSatellite;
```

iOS 9

.satelliteFlyover

Affiche une image satellite de la zone avec les données de survol disponibles.

Swift 2

```
mapView.mapType = .SatelliteFlyover
```

Swift 3

```
mapView.mapType = .satelliteFlyover
```

Objectif c

```
_mapView.mapType = MKMapTypeSatelliteFlyover;
```

iPhone OS 3

.hybride

Affiche une image satellite de la zone avec des informations sur les routes et les noms de routes superposées.

Swift 2

```
mapView.mapType = .Hybrid
```

Swift 3

```
mapView.mapType = .hybrid
```

Objectif c

```
_mapView.mapType = MKMapTypeHybrid;
```

iOS 9

.hybridFlyover

Affiche une image satellite hybride avec des données de survol disponibles.

Swift 2

```
mapView.mapType = .HybridFlyover
```

Swift 3

```
mapView.mapType = .hybridFlyover
```

Objectif c

```
_mapView.mapType = MKMapTypeHybridFlyover;
```

Définir le zoom / région pour la carte

Pour définir un certain niveau de zoom, disons que nous voulons agrandir l'emplacement de l'utilisateur avec l'emplacement de l'utilisateur en tant que centre et 2 km de zone en tant que rayon. Ensuite, nous utilisons le code suivant

```
MKUserLocation *userLocation = _mapView.userLocation;
MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance
(userLocation.location.coordinate, 2000, 2000);
[_mapView setRegion:region animated:NO];
```

Implémentation de la recherche locale à l'aide de MKLocalSearch

MKLocalSearch permet aux utilisateurs de rechercher un emplacement en utilisant des chaînes en langage naturel comme "gym". Une fois la recherche terminée, la classe renvoie une liste d'emplacements dans une région spécifiée correspondant à la chaîne de recherche.

Les résultats de la recherche sont sous la forme de MKMapItem dans l'objet MKLocalSearchResponse.

permet d'essayer par exemple

```
MKLocalSearchRequest *request =
    [[MKLocalSearchRequest alloc] init];//initialising search request
request.naturalLanguageQuery = @”Gym”; // adding query
request.region = _mapView.region; //setting region
MKLocalSearch *search =
     [[MKLocalSearch alloc]initWithRequest:request];//initiate search

[search startWithCompletionHandler:^(MKLocalSearchResponse
     *response, NSError *error)
{
   if (response.mapItems.count == 0)
       NSLog(@"No Matches");
   else
       for (MKMapItem *item in response.mapItems)
       {
           NSLog(@"name = %@", item.name);
           NSLog(@"Phone = %@", item.phoneNumber);
        }
}];
```

OpenStreetMap Tile-Overlay

Dans certains cas, vous ne souhaitez peut-être pas utiliser les cartes par défaut, indique Apple.

Vous pouvez ajouter une superposition à votre `mapView` qui contient des tuiles personnalisées, par exemple à partir d' *OpenStreetMap* .

Supposons que `self.mapView` est votre `MKMapView` que vous avez déjà ajouté à votre `ViewController` .

Au début, votre `ViewController` doit être conforme au protocole `MKMapViewDelegate` .

```
class MyViewController: UIViewController, MKMapViewDelegate
```

Ensuite, vous devez définir le `ViewController` tant que délégué de `mapView`

```
mapView.delegate = self
```

Ensuite, vous configurez la superposition pour la carte. Vous aurez besoin d'un modèle d'URL pour cela. L'URL devrait être similaire à ceci sur tous les serveurs de tuiles et même si vous stockeriez les données de carte hors ligne: `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
```

Après avoir configuré la superposition, vous devez l'ajouter à votre `mapView` .

```
mapView.add(overlay, level: .aboveLabels)
```

Pour utiliser des cartes personnalisées, il est recommandé d'utiliser `.aboveLabels` pour `level` . Sinon, les étiquettes par défaut seraient visibles sur votre carte personnalisée. Si vous voulez voir les étiquettes par défaut, vous pouvez choisir `.aboveRoads` ici.

Si vous exécutez votre projet maintenant, vous reconnaîtrez que votre carte affichera toujours la carte par défaut:

C'est parce que nous n'avons pas encore dit à `mapView` comment rendre la superposition. C'est la raison pour laquelle vous avez dû définir le délégué auparavant. Maintenant, vous pouvez ajouter `func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer` à votre contrôleur de vue:

```
func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {
    if overlay is MKTileOverlay {
        let renderer = MKTileOverlayRenderer(overlay: overlay)
        return renderer
    } else {
        return MKTileOverlayRenderer()
    }
}
```

Cela retournera le bon `MKOverlayRenderer` à votre `mapView` . Si vous exécutez votre projet maintenant, vous devriez voir une carte comme celle-ci:

Si vous souhaitez afficher une autre carte, il vous suffit de modifier le modèle d'URL. Il y a une liste de serveurs de tuiles dans le wiki OSM.

Exemple d'utilisation de UserLocation et UserTracking

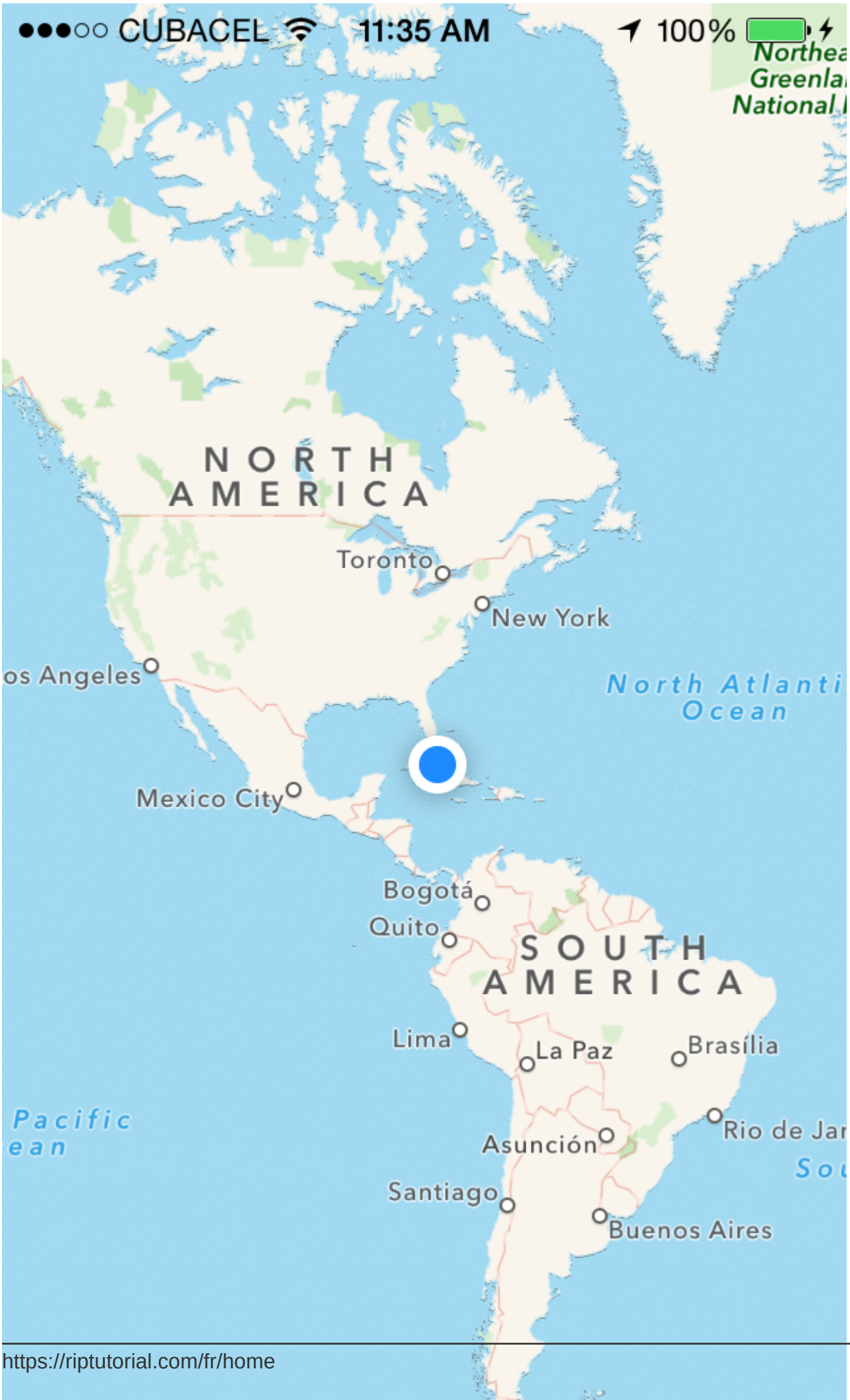
Cela montrera l'emplacement de l'utilisateur sur la carte

Objectif c

```
[self.map setShowsUserLocation:YES];
```

Rapide

```
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

affiche une coordonnée au niveau du zoom au lieu de définir une région à afficher. Cette fonctionnalité n'est pas implémentée par défaut. Vous devez donc étendre `MKMapView` avec une méthode qui effectue le calcul complexe d'un niveau de *coordonnées* et d'un *niveau* de *zoom* à une `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 version originale de Swift 2 de Mylene Bayan est disponible sur GitHub)

Après avoir implémenté cette `extension` , vous pouvez définir la coordonnée centrale comme suit:

```
let centerCoordinate = CLLocationCoordinate2DMake(48.136315, 11.5752901) //latitude, longitude
mapView?.setCenter(centerCoordinate, zoomLevel: 15, animated: true)
```

`zoomLevel` est une valeur `Double` , généralement comprise entre `0` et `21` (ce qui correspond à un niveau de zoom très élevé), mais des valeurs allant jusqu'à `28` sont autorisées.

Travailler avec l'annotation

Obtenir toutes les annotations

```
//following method returns all annotations object added on map
NSArray *allAnnotations = mapView.annotations;
```

Obtenir une vue d'annotation

```
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"];
    }
}
```

Supprimer toutes les annotations

```
[mapView removeAnnotations:mapView.annotations]
```

Supprimer une seule annotation

```
//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];

}
```

Ajustez le rectangle visible de la carte pour afficher toutes les annotations

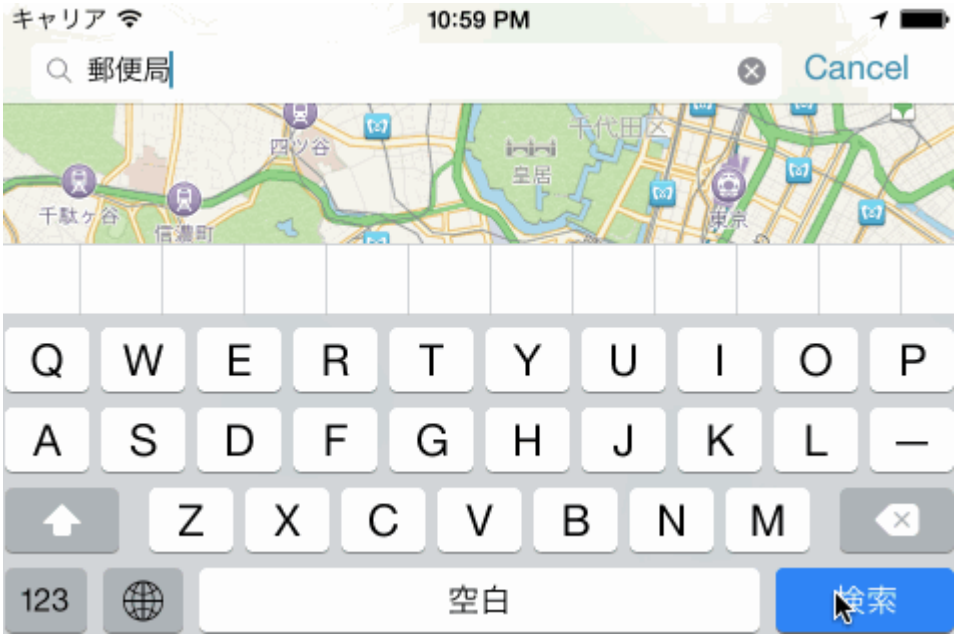
Rapide:

```
mapView.showAnnotations(mapView.annotations, animated: true)
```

Objectif c:

```
[mapView showAnnotations:mapView.annotations animated:YES];
```

Démo:



Lire MKMapView en ligne: https://riptutorial.com/fr/ios/topic/915/mkmapview

Chapitre 97: ModelPresentationStyles

Introduction

Les styles de présentation modale sont utilisés lorsque vous passez d'un contrôleur de vue à un autre. Il y a 2 façons de réaliser cette personnalisation. L'un se fait par code et l'autre par Interface Builder (à l'aide de segues). Cet effet est obtenu en définissant la variable `modalPresentationStyle` sur une instance de `UIModalPresentationStyle` enum. `modalPresentationStyle` propriété `modalPresentationStyle` est une variable de classe de `UIViewController` et permet de spécifier comment un `ViewController` est présenté à l'écran.

Remarques

Rappelez-vous toujours la mention suivante d'Apple.

> Dans un environnement compact horizontalement, les contrôleurs de vues modales sont toujours présentés en plein écran. Dans un environnement horizontal régulier, il existe plusieurs options de présentation différentes.

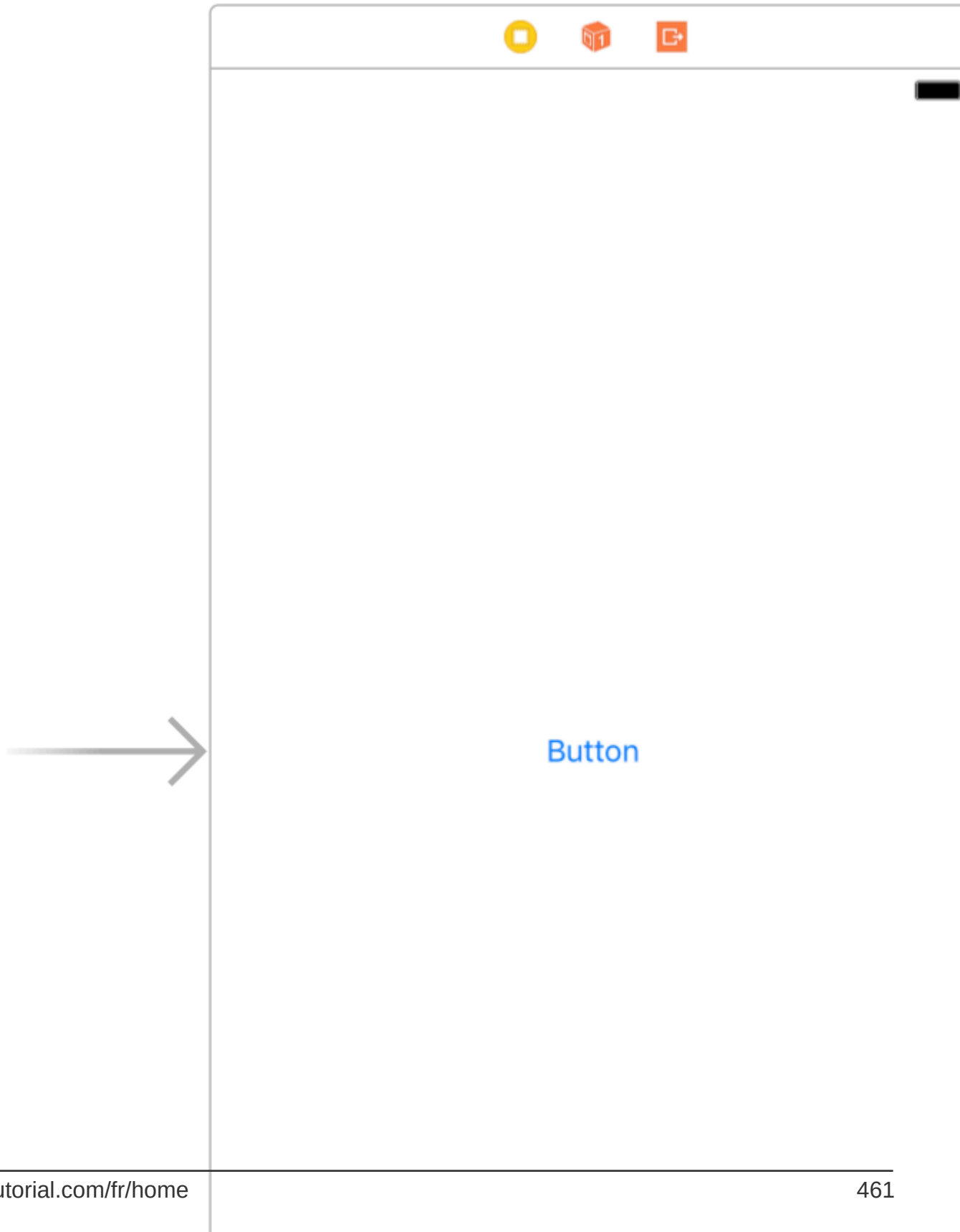
Examples

Exploration de ModalPresentationStyle à l'aide d'Interface Builder

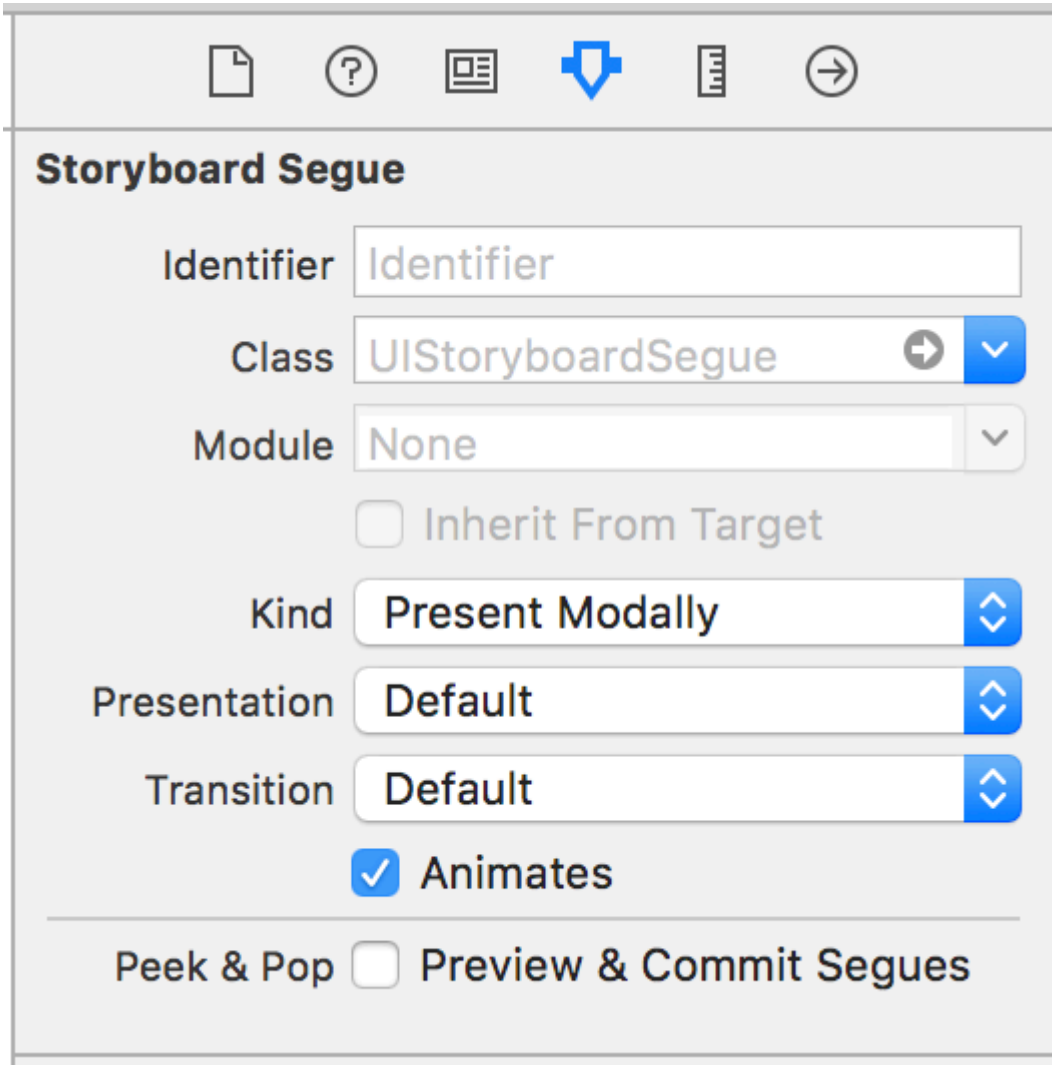
Ce sera une application très basique qui illustrera différents `ModalpresentationStyle` dans iOS. Selon la documentation trouvée ici , il existe 9 valeurs différentes pour `UIModalPresentationStyle` qui sont les suivantes,

1. `fullScreen`
2. `pageSheet`
3. `formSheet`
4. `currentContext`
5. `custom`
6. `overFullScreen`
7. `overCurrentContext`
8. `popover`
9. `none`

Pour configurer un projet, créez simplement un projet iOS normal et ajoutez 2 `ViewControllers` . Placez un `UIButton` dans votre `ViewController` initial et connectez-le à 2nd `ViewController` via un mécanisme `Target -> Action` . Pour distinguer les deux `ViewControllers` , définissez la propriété d'arrière-plan de `UIView` dans `ViewController` une autre couleur. Si tout se passe bien, votre Interface Builder devrait ressembler à ceci,


Button

(pour plus d'informations sur les raisons de l'iPad, reportez-vous à la section Remarques). Une fois que vous avez terminé la configuration de votre projet, sélectionnez-le et accédez à l' `attributes inspector` . Vous devriez pouvoir voir quelque chose comme ça,



Définissez la propriété aimable à `Present Modally` .

Maintenant, nous ne verrons pas tous les effets dans cet exemple car certains d'entre eux nécessitent peu de code.

Commençons par `fullscreen` . Cet effet est sélectionné par défaut lorsque vous sélectionnez l' `Present Modally` en `Kind` . Lorsque vous construisez et exécutez, le 2nd `ViewController` occupera tout l'écran de votre iPad.


iPad Air – iOS 10.3 (14E269)
Carrier
6:11 PM

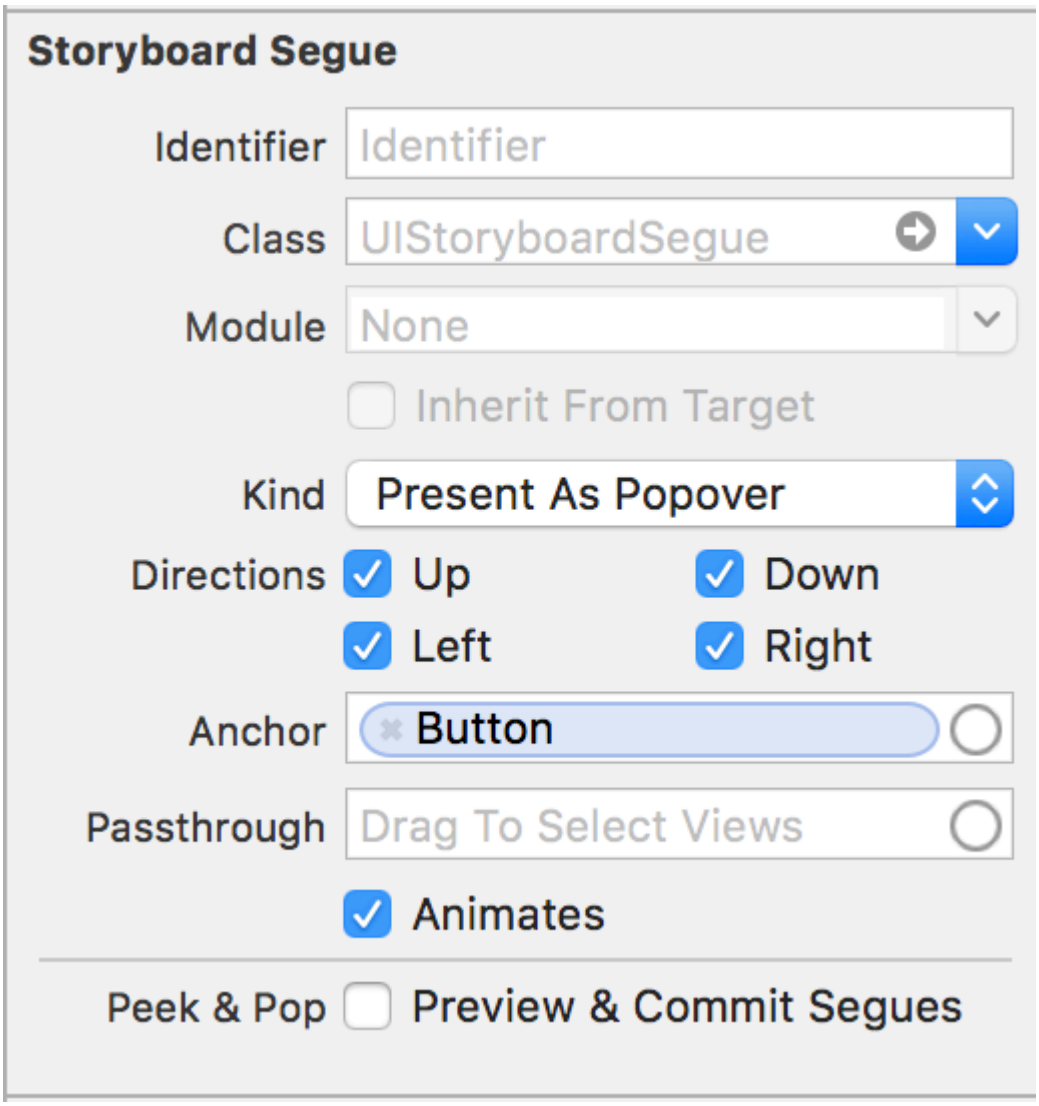
. Dans cette option, lorsque le périphérique est en mode portrait, le 2nd `ViewController` est similaire au plein écran, mais en mode paysage, 2nd `ViewController` réduit considérablement la largeur du périphérique. De plus, tout contenu non couvert par 2nd `ViewController` sera grisé.


iPad Air – iO
Carrier

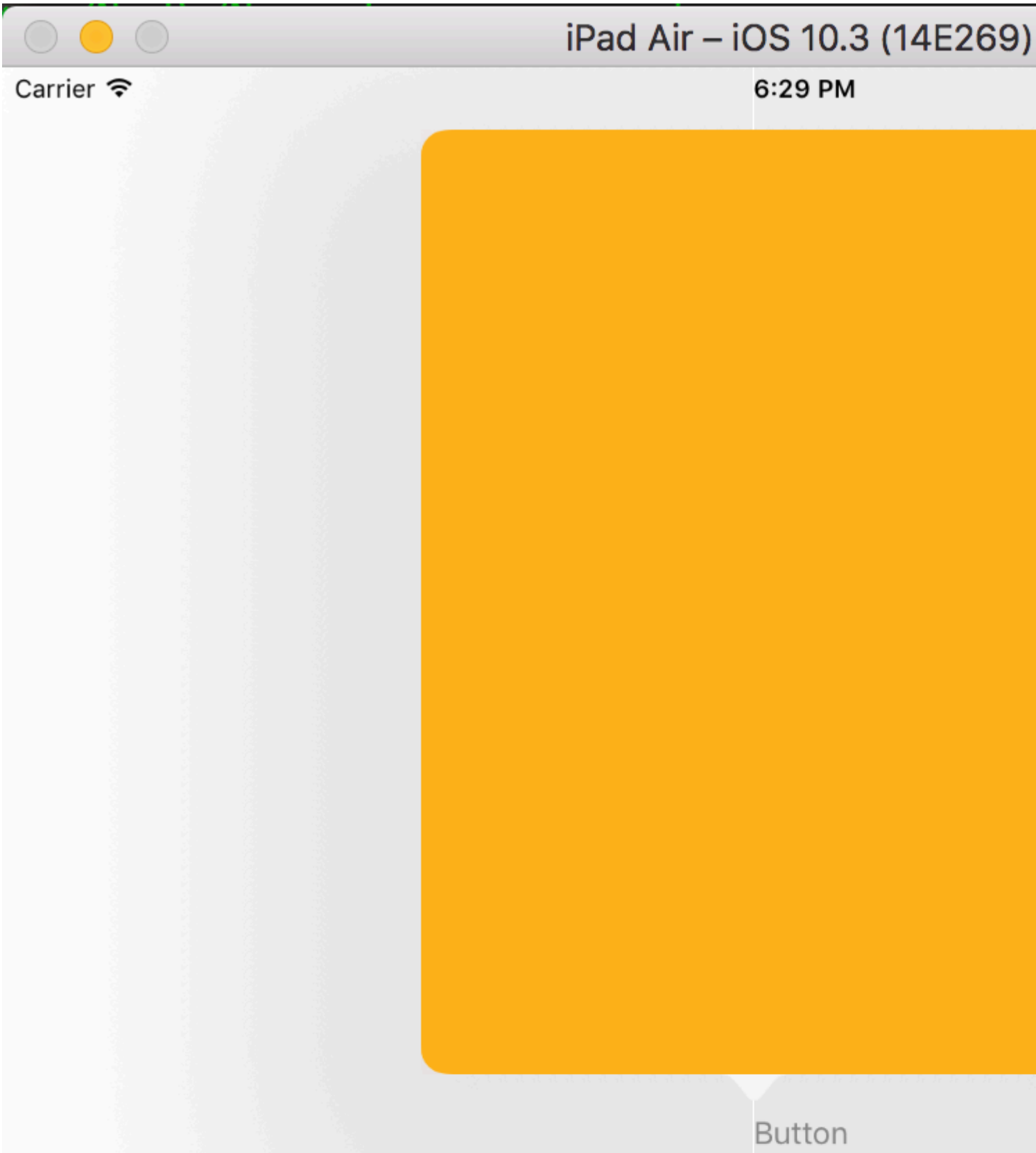
est placé au centre du périphérique et sa taille est inférieure à celle du périphérique. De même, lorsque l'appareil est en mode paysage et que le clavier est visible, la position de la vue est ajustée vers le haut pour afficher le `ViewController` .


iPad Air – iOS 10.3 (14E269)
Carrier
6:23 PM

`Present as Popover` onglet `Present as Popover` in `Kind` . Le 2nd `ViewController` est présenté sous la forme d'un petit popover (la taille peut être définie). Le contenu d'arrière-plan est grisé. Tout tapotement en dehors du popover rejetterait le popover. Votre `Attributes Inspector` devrait ressembler à ceci,



`Anchor` est l'élément d'interface utilisateur auquel vous souhaitez que la flèche de votre popover pointe. `Directions` sont les directions que vous autorisez votre `Anchor` popover à pointer.


iPad Air – iOS 10.3 (14E269)
Carrier
6:29 PM
Button

https://riptutorial.com/fr/ios/topic/10122/modelpresentationstyles

Chapitre 98: Modes d'arrière-plan

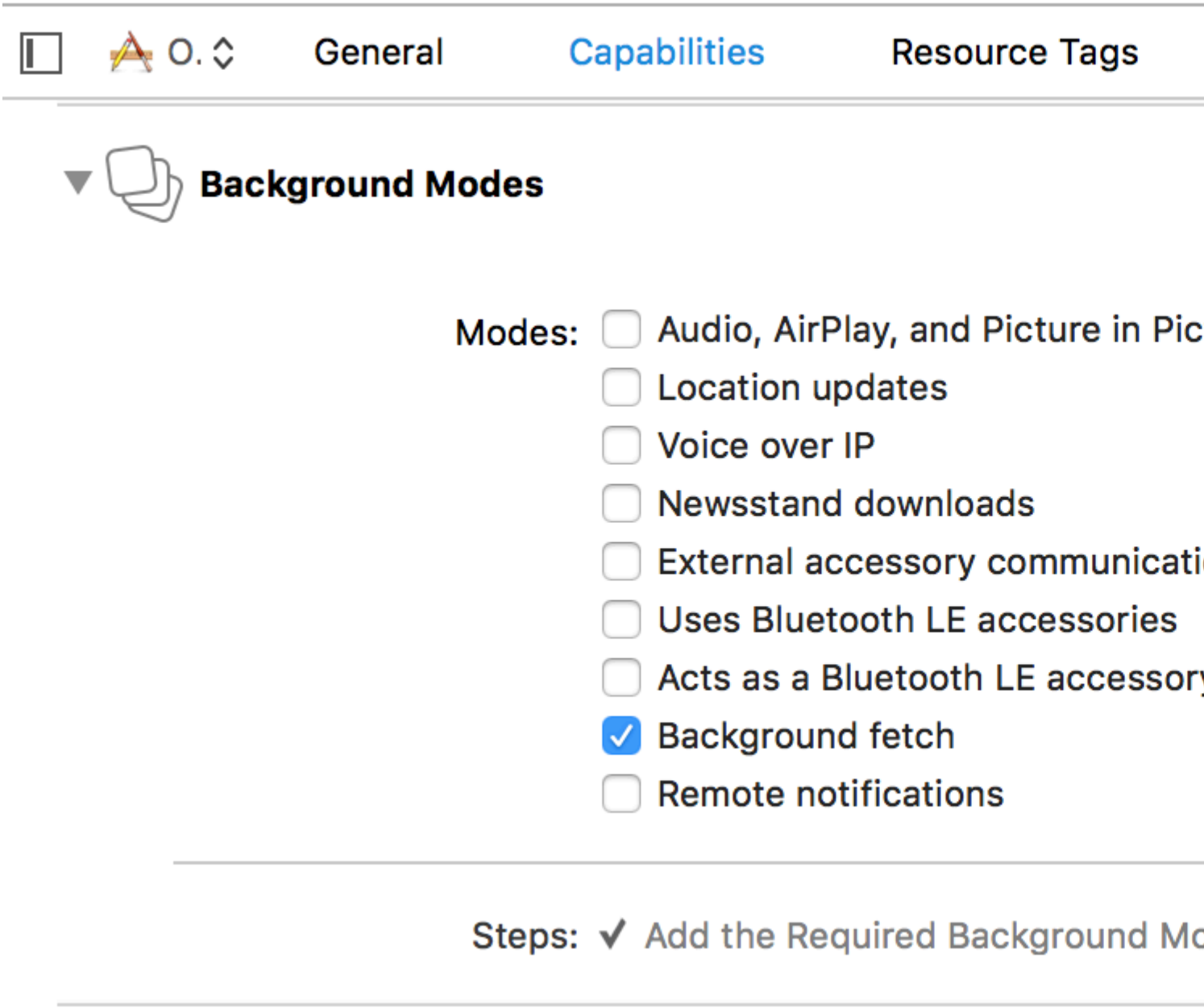
Introduction

Être réactif est un besoin pour chaque application. Les utilisateurs veulent avoir des applications dont le contenu est prêt à l'ouverture, de sorte que les développeurs doivent utiliser les modes d'arrière-plan pour rendre leurs applications plus conviviales.

Examples

Activation de la fonctionnalité Modes d'arrière-plan

1. Allez dans Xcode et ouvrez votre projet.
2. Dans la cible de votre application, accédez à l'onglet Capabilities.
3. Activer les modes d'arrière-plan.



Fetch de fond

La recherche en arrière-plan est un nouveau mode qui permet à votre application d'apparaître toujours à jour avec les dernières informations tout en minimisant l'impact sur la batterie. Vous pouvez télécharger des flux à intervalles fixes avec cette fonctionnalité.

Pour commencer:

1- Vérifier l'arrière-plan dans l'écran des capacités dans Xcode.

2- Dans la méthode `application(_:didFinishLaunchingWithOptions:)` dans `AppDelegate` , ajoutez:

Rapide

```
UIApplication.shared.setMinimumBackgroundFetchInterval(UIApplicationBackgroundFetchIntervalMinimum)
```

Objectif c

```
[[UIApplication shared]
setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMinimum]
```

> Au lieu de `UIApplicationBackgroundFetchIntervalMinimum` , vous pouvez utiliser n'importe `CGFloat` valeur `CGFloat` pour définir les intervalles d'extraction.

3- Vous devez implémenter l' `application(_:performFetchWithCompletionHandler:)` . Ajoutez ceci à votre `AppDelegate` :

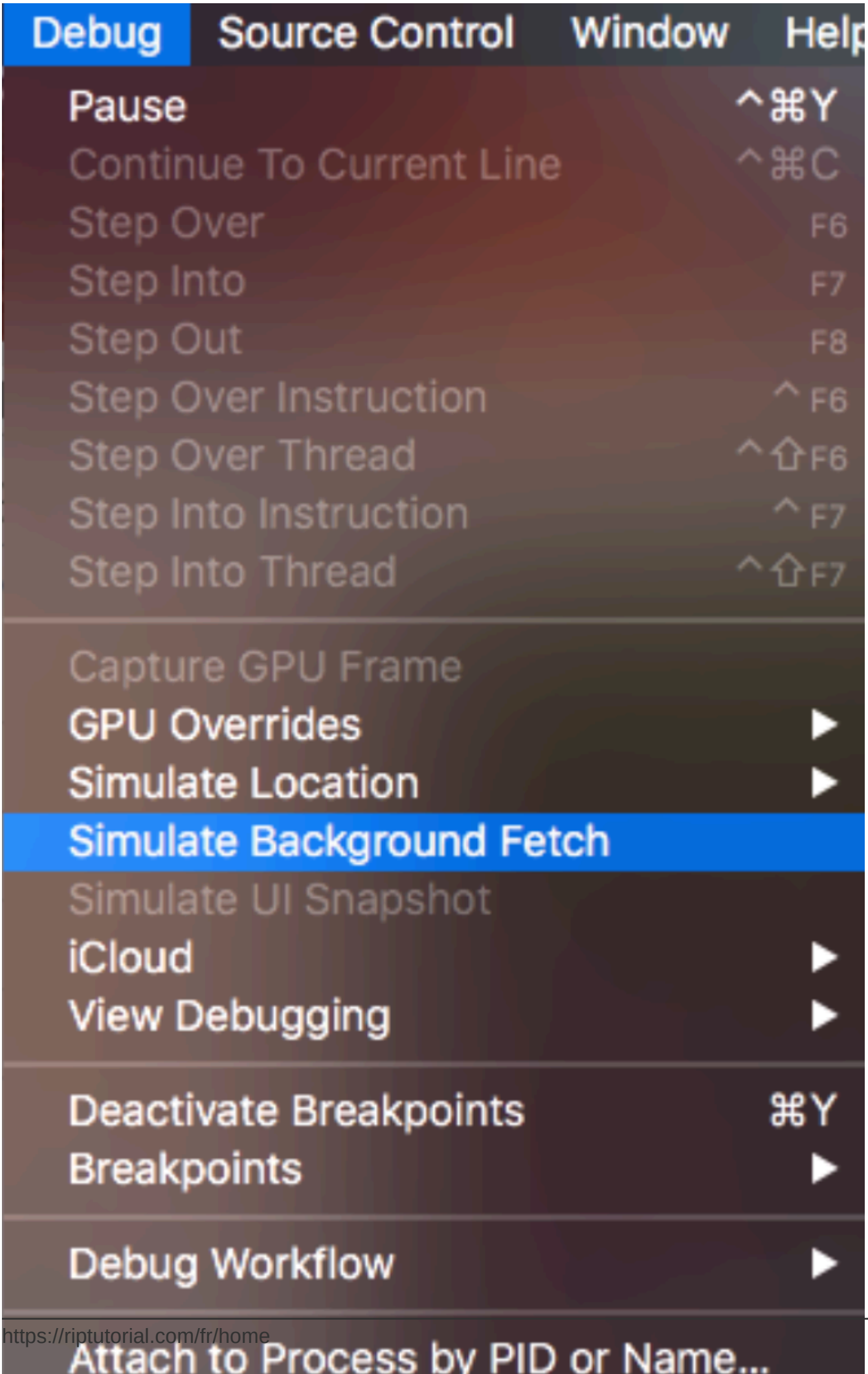
Rapide

```
func application(_ application: UIApplication, performFetchWithCompletionHandler
completionHandler: @escaping (UIBackgroundFetchResult) -> Void) {
    // your code here
}
```

Test de récupération en arrière-plan

1- Exécutez l'application sur un périphérique réel et joignez-la au débogueur Xcode.

2- Dans le menu Déboguer, sélectionnez **Simuler l'arrière-plan** :


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...

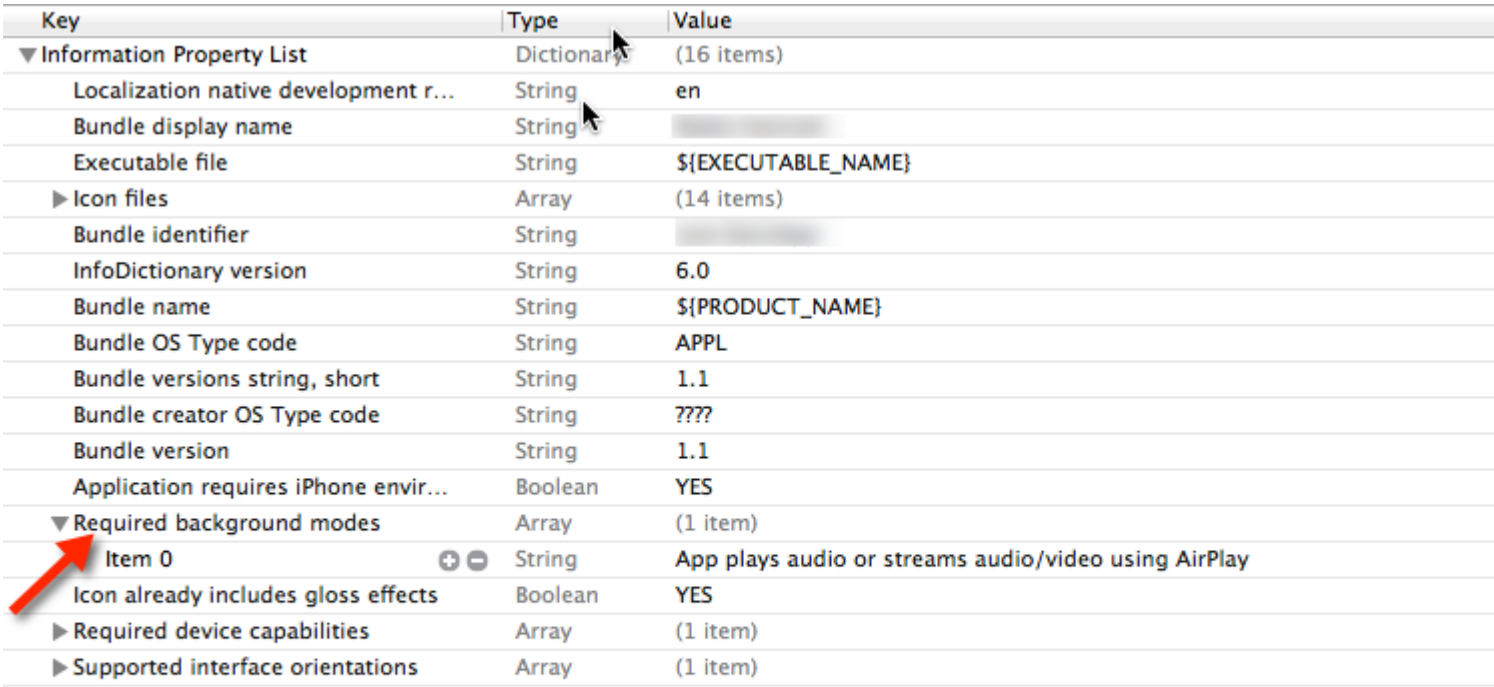
Chapitre 99: Modes de fond et événements

Examples

Jouer de l'audio en arrière-plan

Ajoutez une clé nommée **Modes d'arrière-plan requis** dans le fichier de liste de propriétés (.plist).

comme image suivante.



Et ajouter le code suivant dans

AppDelegate.h

```
#import <AVFoundation/AVFoundation.h>
#import <AudioToolbox/AudioToolbox.h>
```

AppDelegate.m

dans l'application didFinishLaunchingWithOptions

```
[[AVAudioSession sharedInstance] setDelegate:self];
[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPlayback error:nil];
[[AVAudioSession sharedInstance] setActive:YES error:nil];
```

```
[[UIApplication sharedApplication] beginReceivingRemoteControlEvents];

UInt32 size = sizeof(CFStringRef);
CFStringRef route;
AudioSessionGetProperty(kAudioSessionProperty_AudioRoute, &size, &route);
NSLog(@"route = %@", route);
```

Si vous souhaitez des modifications selon les événements, vous devez ajouter le code suivant dans 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;
        }
    }
}
```

Basé sur notification doit travailler dessus ..

Lire Modes de fond et événements en ligne: https://riptutorial.com/fr/ios/topic/3515/modes-de-fond-et-evenements

Chapitre 100: Mouvement de base

Examples

Accès au baromètre pour obtenir une altitude relative

Rapide

Importez la bibliothèque Core Motion:

```
import CoreMotion
```

Ensuite, nous devons créer un objet `CMAltimeter` , mais un écueil courant consiste à le créer dans `viewDidLoad()` . Si c'est le cas, l'altimètre ne sera pas accessible lorsque nous aurons besoin d'appeler une méthode. Néanmoins, continuez et créez votre objet `CMAltimeter` juste avant `viewDidLoad()` :

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

À présent:

1. Nous devons vérifier si `relativeAltitude` est même disponible avec la méthode suivante: `CMAltimeter.isRelativeAltitudeAvailable` .
2. Si cela retourne `true` , vous pouvez alors commencer à surveiller le changement d'altitude avec `startRelativeAltitudeUpdatesToQueue`
3. S'il n'y a pas d'erreurs, vous devriez pouvoir récupérer des données à partir des propriétés `relativeAltitude` et pression.

Ci-dessous la définition d'une action de bouton pour commencer la surveillance avec notre baromètre.

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

Lire Mouvement de base en ligne: https://riptutorial.com/fr/ios/topic/7636/mouvement-de-base

Chapitre 101: MPMediaPickerDelegate

Remarques

Veuillez consulter la documentation Apple pour plus d'informations sur la confidentialité.

Assurez-vous que l'application Music est disponible sur votre iPhone. Cela ne fonctionnera pas dans le simulateur.

Examples

Charger de la musique avec MPMediaPickerControllerDelegate et jouer avec AVAudioPlayer

Passez par les étapes:

- Ajoutez 'NSAppleMusicUsageDescription' à votre Info.plist pour l'autorité de confidentialité.
- Assurez-vous que votre musique est disponible sur votre iPhone. Cela ne fonctionnera pas dans le simulateur.

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)
    }

}
```

Lire MPMediaPickerDelegate en ligne:
https://riptutorial.com/fr/ios/topic/7299/mpmediapickerdelegate

Chapitre 102: MPVolumeView

Introduction

La classe MPVolumeView est la vue du volume pour présenter à l'utilisateur un contrôle de curseur permettant de définir le volume de sortie audio du système, ainsi qu'un bouton permettant de choisir l'itinéraire de sortie audio.

Remarques

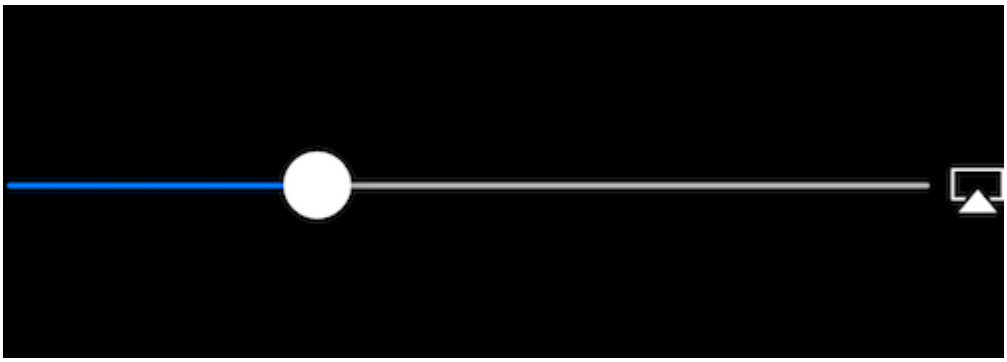
MPVolumeView ne s'affiche que lors de la construction et de l'exécution sur un appareil iOS réel et ne fonctionnera pas dans un simulateur.

Examples

Ajouter 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
```

!!! Une note très importante est que le MPVolumeView ne fonctionne que sur un appareil réel et non sur un simulateur.



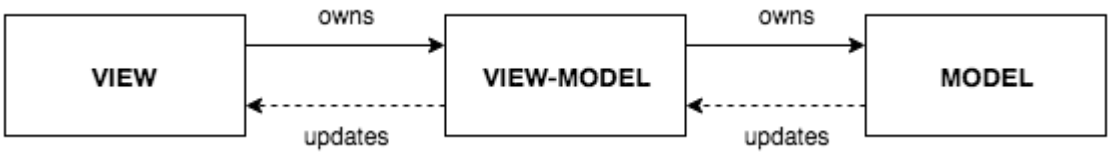
Lire MPVolumeView en ligne: https://riptutorial.com/fr/ios/topic/9038/mpvolumeview

Chapitre 103: MVVM

Examples

MVVM sans programmation réactive

Je commencerai par une brève explication de ce qui est et pourquoi utiliser le modèle de conception Model-View-ViewModel (MVVM) dans vos applications iOS. Lorsque iOS est apparu pour la première fois, Apple a suggéré d'utiliser MVC (Model-View-Controller) comme modèle de conception. Ils l'ont montré dans tous leurs exemples et tous les premiers développeurs ont été heureux de l'utiliser parce que cela séparait bien les préoccupations entre la logique métier et l'interface utilisateur. Au fur et à mesure que les applications devenaient plus grandes et plus complexes, un nouveau problème s'appelait de manière appropriée appelé MVC (Massive View Controllers). Étant donné que toute la logique métier a été ajoutée dans ViewController, avec le temps, ils devenaient généralement trop volumineux et complexes. Pour éviter le problème de MVC, un nouveau modèle de conception a été introduit dans le monde d'iOS: le modèle MVVM (Model-View-ViewModel).



Le diagramme ci-dessus montre à quoi ressemble MVVM. Vous disposez d'un ViewController + View standard (dans le storyboard, XIB ou Code), qui agit en tant que View MVVM (dans le texte suivant - View référencera View MVVM). Une vue fait référence à un ViewModel, où se trouve notre logique métier. Il est important de noter que ViewModel ne sait rien de la vue et n'a jamais de référence à la vue. ViewModel a une référence à un modèle.
Cela suffit avec une partie théorique du MVVM. Plus d'informations à ce sujet peuvent être lues ici .

L'un des **principaux problèmes liés à MVVM** est la mise à jour de View via ViewModel lorsque ViewModel ne contient aucune référence et ne connaît même pas la vue.

La partie principale de cet exemple est de montrer comment utiliser MVVM (plus précisément comment lier ViewModel et View) sans programmation réactive (ReactiveCocoa, ReactiveSwift ou RxSwif). Juste comme une note: si vous souhaitez utiliser la programmation réactive, encore mieux puisque les liaisons MVVM sont vraiment faciles à utiliser. Mais cet exemple concerne l'utilisation de MVVM sans programmation réactive.

Créons un exemple simple pour montrer comment utiliser MVVM.

Notre `MVVMExampleViewController` est un simple ViewController avec une étiquette et un bouton. Lorsque le bouton est enfoncé, le texte de l'étiquette doit être défini sur «Hello». Étant donné que décider de ce qu'il faut faire de l'interaction utilisateur-utilisateur fait partie de la logique métier, ViewModel devra décider quoi faire lorsque l'utilisateur appuie sur le bouton. La vue du MVVM ne

devrait faire aucune logique métier.

```
class MVVMExampleViewController: UIViewController {

    @IBOutlet weak var helloLabel: UILabel!

    var viewModel: MVVMExampleViewModel?

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

    @IBAction func sayHelloButtonPressed(_ sender: UIButton) {
        viewModel?.userTriggeredSayHelloButton()
    }
}
```

`MVVMExampleViewModel` est un simple ViewModel.

```
class MVVMExampleViewModel {

    func userTriggeredSayHelloButton() {
        // How to update View's label when there is no reference to the View??
    }
}
```

Vous pourriez vous demander comment définir la référence de ViewModel dans la vue. Je le fais habituellement lorsque ViewController est en cours d'initialisation ou avant son affichage. Pour cet exemple simple, je ferais quelque chose comme ça dans `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 vraie question est maintenant la suivante: comment mettre à jour View à partir de ViewModel sans donner de référence à View to ViewModel? (Rappelez-vous que nous n'utiliserons aucune des bibliothèques iOS de programmation réactive)

Vous pourriez penser à utiliser KVO, mais cela ne ferait que compliquer les choses. Certaines personnes intelligentes ont réfléchi à la question et ont créé la bibliothèque Bond . La bibliothèque peut sembler compliquée et difficile à comprendre au début, alors je vais en prendre une petite partie et rendre notre MVVM entièrement fonctionnel.

Présentons la classe `Dynamic` qui est au cœur de notre modèle MVVM simple mais totalement fonctionnel.

```
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` utilise Generics and Closures pour lier notre ViewModel à notre View. Je ne vais pas entrer dans les détails de cette classe, nous pouvons le faire dans les commentaires (pour raccourcir cet exemple). Nous allons maintenant mettre à jour nos `MVVMExampleViewController` et `MVVMExampleViewModel` pour utiliser ces classes.

Notre `MVVMExampleViewController` mis à jour

```
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()
    }
}
```

`MVVMExampleViewModel` mis à jour:

```
    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"
    }
}
```

C'est ça. Votre `ViewModel` est maintenant capable de mettre à jour `View` sans avoir de référence à la `View` .

Ceci est un exemple très simple, mais je pense que vous avez une idée de son potentiel. Je ne vais pas entrer dans les détails concernant les avantages du MVVM, mais une fois que vous passerez de MVC à MVVM, vous ne reviendrez pas en arrière. Essayez-le et voyez par vous-même.

Lire MVVM en ligne: https://riptutorial.com/fr/ios/topic/8775/mvvm

Chapitre 104: MyLayout

Introduction

MyLayout est un framework simple et facile d'objectif-c pour la disposition de vues iOS. MyLayout fournit des fonctions simples pour créer diverses interfaces complexes. Il intègre les fonctions suivantes: Autolayout et SizeClass d'iOS, cinq classes de disposition d'Android, float et flex-box et bootstrap de HTML / CSS. vous pouvez visiter de:

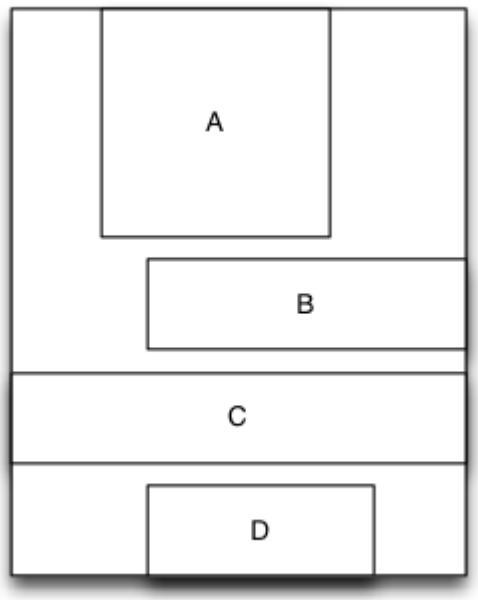
Objective-C: https://github.com/youngsoft/MyLinearLayout Swift: https://github.com/youngsoft/TangramKit

Examples

Une démo simple pour utiliser MyLayout

1. Il existe une vue de conteneur S dont la largeur est égale à 100 et la hauteur à la hauteur de toutes les sous-vues. il y a quatre sous-vues A, B, C, D disposées de haut en bas.
2. La marge gauche de la sous-vue A est la largeur de 20% de S, la marge droite est la largeur de 30% de S, la hauteur est égale à la largeur de A.
3. La marge gauche de la sous-vue B est de 40, la largeur de la largeur résiduelle de S, la hauteur de 40. La largeur de la sous-vue C est remplie par S, la hauteur est de 40.
4. La marge droite de la sous-vue D est 20, la largeur est 50% de la largeur de S, la hauteur est 40

comme ci-dessous figure:



```
    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];
```

Lire MyLayout en ligne: https://riptutorial.com/fr/ios/topic/9692/mylayout

Chapitre 105: Notifications enrichies

Introduction

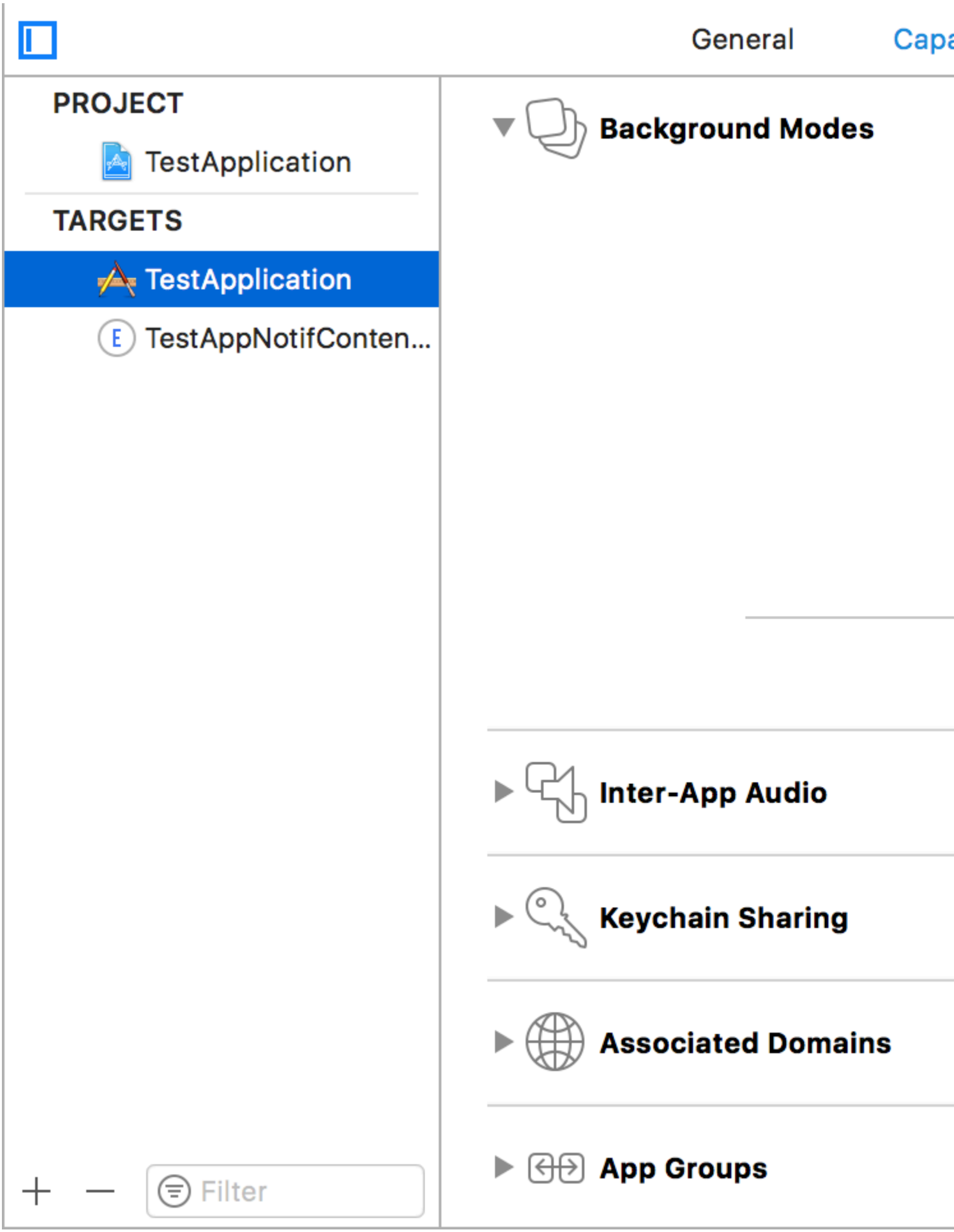
Les notifications enrichies vous permettent de personnaliser l'apparence des notifications locales et distantes lorsqu'elles apparaissent sur le périphérique de l'utilisateur. La notification rapide inclut principalement UNNotificationServiceExtension et UNNotificationContentExtension, c'est-à-dire affichant la notification normale de manière étendue.

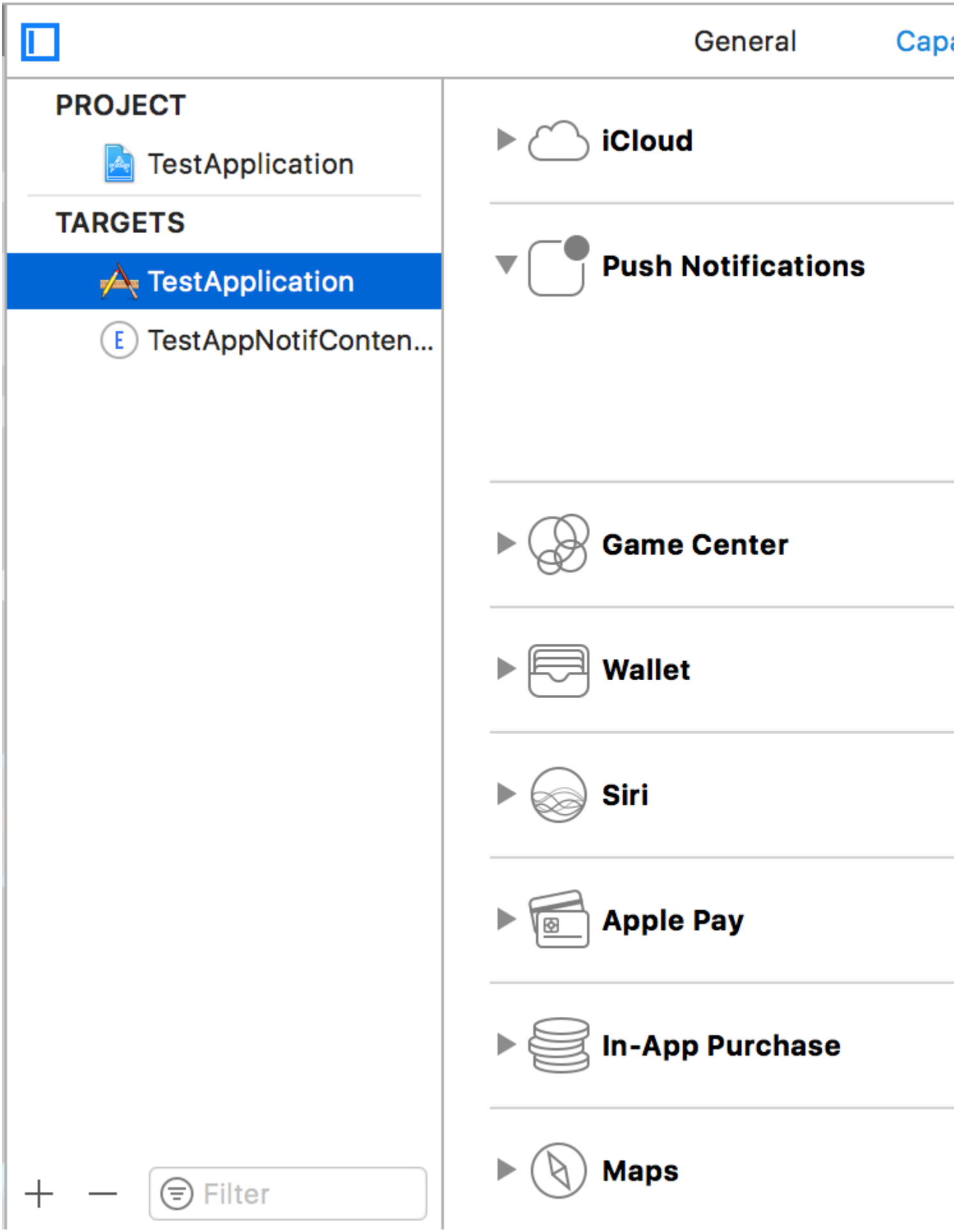
Examples

Créer un simple UNNotificationContentExtension

Étape 1

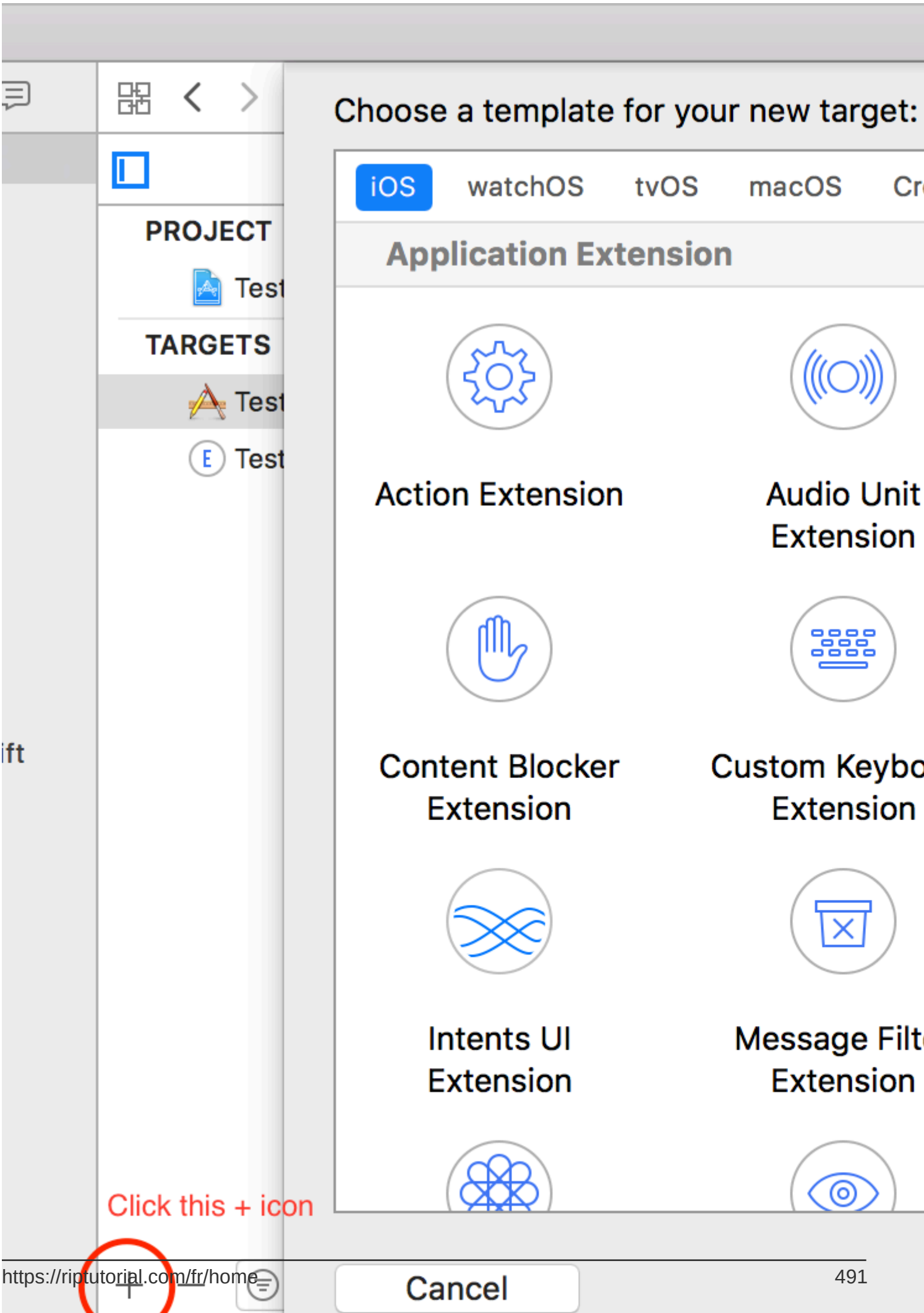
Rendre l'environnement adapté à la notification. Assurez-vous d'avoir activé les **modes d'arrière-plan** et la **notification push**


General
Capa
PROJECT
TestApplication
TARGETS
TestApplication
TestAppNotifConten...
Background Modes
Inter-App Audio
Keychain Sharing
Associated Domains
App Groups
Filter


General
Capa
PROJECT
TestApplication
TARGETS
TestApplication
TestAppNotifConten...
Filter
iCloud
Push Notifications
Game Center
Wallet
Siri
Apple Pay
In-App Purchase
Maps

Étape 2: création d'une extension UNNotificationContentExtension

Cliquez sur l'icône **+** en bas qui crée un modèle cible et sélectionnez Notification Content Extention -> next -> créer un nom pour l'extension de contenu -> finish


Choose a template for your new target:
iOS
watchOS
tvOS
macOS
Application Extension
Action Extension
Audio Unit Extension
Content Blocker Extension
Custom Keybo Extension
Intents UI Extension
Message Filt Extension
PROJECT
Test
TARGETS
Test
Test
Click this + icon
Cancel

- UNNotificationDefaultContentHidden: Ce booléen détermine si le corps par défaut de la notification doit être masqué ou non
- UNNotificationCategory: La catégorie est créée dans `UNUserNotificationCenter` dans votre application. Ici, il peut s'agir d'une chaîne ou d'un tableau de chaînes, de sorte que chaque catégorie peut donner différents types de données à partir desquelles nous pouvons créer différentes interfaces utilisateur. La charge utile que nous envoyons doit contenir le nom de la catégorie pour afficher cette extension particulière
  dans votre application. Ici, il peut s'agir d'une chaîne ou d'un tableau de chaînes, de sorte que chaque catégorie peut donner différents types de données à partir desquelles nous pouvons créer différentes interfaces utilisateur. La charge utile que nous envoyons doit contenir le nom de la catégorie pour afficher cette extension particulière
- UNNotificationExtensionInitialContentSizeRatio: La taille du contenu initial, c'est-à-dire lors de l'affichage de ContentExtension pour la première fois, la taille initiale par rapport à la largeur du périphérique. ici 1 indique que la hauteur sera égale à la largeur

Étape 4: Création de `UNNotificationAction` et `UNNotificationCategory` dans notre application

Dans la fonction `didFinishLaunchingWithOptions` votre application, la fonction `didFinishLaunchingWithOptions` ajoute

```
    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()
```

Nous avons créé deux `UNNotificationAction` avec les identifiants `ID1` et `ID2` et ajouté ces actions à `UNNotificationCategory` avec l'identifiant `CATID1` (l'ID de catégorie du fichier info.plist de ContentExtension est identique, ce que nous avons créé ici doit être utilisé dans le fichier payload et le fichier plist). Nous définissons la catégorie à `UNUserNotificationCenter` notre application et à la ligne suivante, nous enregistrons la notification qui appelle la fonction `didRegisterForRemoteNotificationsWithDeviceToken` où nous obtenons le jeton de périphérique

Remarque: n'oubliez pas d' `import UserNotifications` dans votre AppDelegate.swift et d'ajouter `UNUserNotificationCenterDelegate`

Étape 5: Exemple de charge utile pour le NotificationContent

```
'aps': {
   'badge': 0,
   'alert': {
       'title': "Rich Notification",
       'body': "Body of RICH NOTIFICATION",
       },
   'sound' : "default",
```

```
    'category': "CATID1",
    'mutable-content':"1",
    },
'attachment': "2"
```

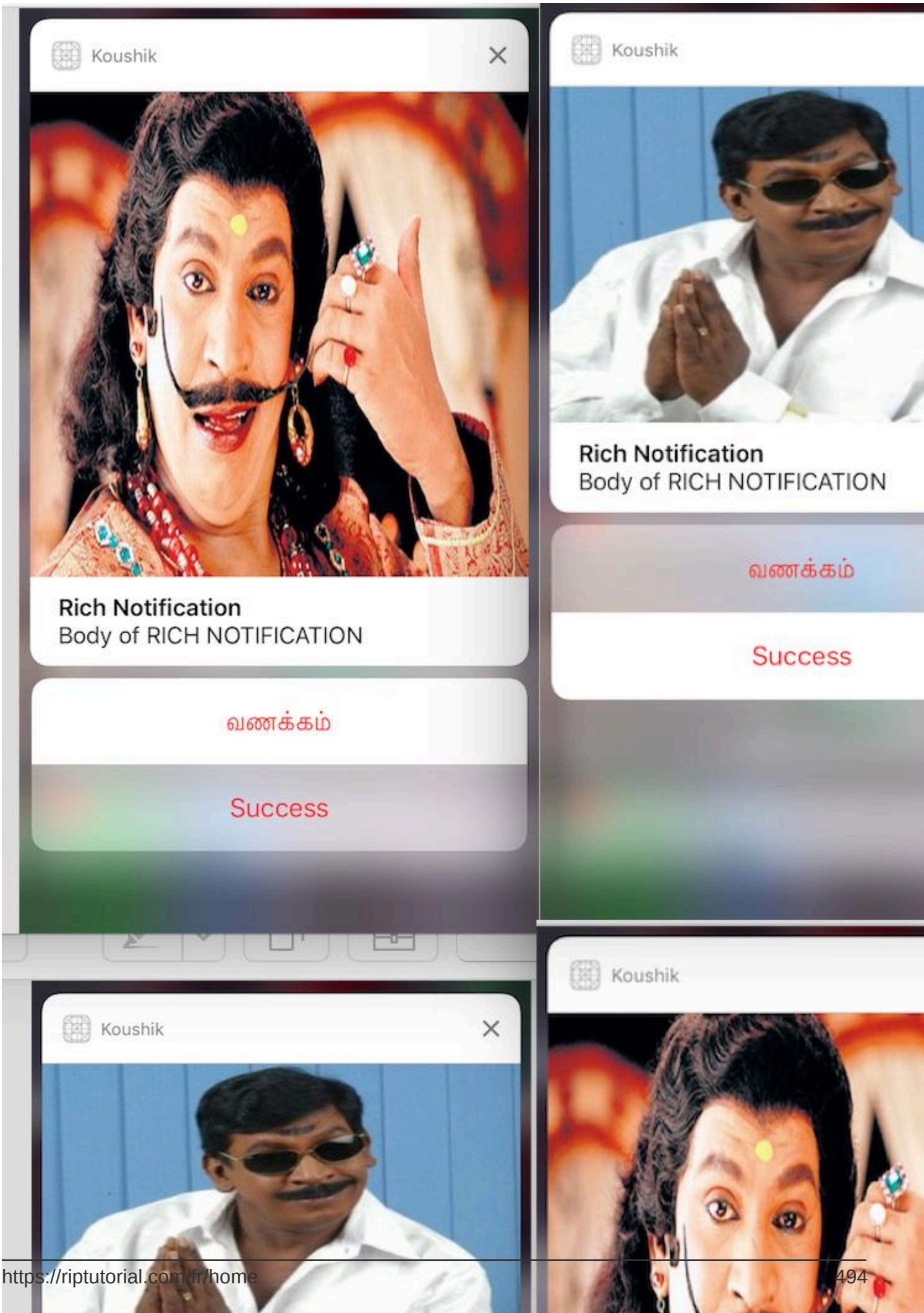
Étape 6: Configuration de ContentExtension

Les actions correspondantes pour la catégorie sont automatiquement affichées lorsque l'action de notification est effectuée. Permet de voir le code comment sa réalisation

```
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")
    }

    }
}
```

Étape 7: Résultat

Après avoir reçu et appuyé longuement / en cliquant sur la notification, la notification ressemble à ceci:


Koushik
Rich Notification
Body of RICH NOTIFICATION
வணக்கம்
Success
Koushik
Rich Notification
Body of RICH NOTIFICATION
வணக்கம்
Success
Koushik
Koushik

`UNNotificationExtensionOverrideDefaultTitle` comme YES. A l'étape 3, nous avons donné `UNNotificationExtensionDefaultContentHidden` comme NO si son OUI alors la notification ressemblera aux images 3 et 4.

Lire Notifications enrichies en ligne: https://riptutorial.com/fr/ios/topic/10769/notifications-enrichies

Chapitre 106: Notifications push

Syntaxe

- UIUserNotificationSettings.types: UIUserNotificationType // Un masque de bits des types de notification que votre application est autorisée à utiliser
- UIUserNotificationSettings.categories: Définir // Groupes d'actions enregistrés de l'application

Paramètres

Paramètre	La description
informations utilisateur	Un dictionnaire contenant des informations de notification à distance, comprenant éventuellement un numéro de badge pour l'icône de l'application, un son d'alerte, un message d'alerte, un identifiant de notification et des données personnalisées.

Examples

Enregistrement du périphérique pour les notifications Push

Pour enregistrer votre appareil pour les notifications push, ajoutez le code suivant à votre fichier `didFinishLaunchingWithOptions` dans la méthode `didFinishLaunchingWithOptions` :

Rapide

```
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
}
```

Objectif 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
         {}
    //-----------------------------------------------
```

Le code ci-dessus essaiera de communiquer avec le serveur APNs pour obtenir un jeton de périphérique (les prérequis sont-ils activés dans votre profil d'approvisionnement iOS).

Une fois qu'il établit une connexion fiable avec le serveur APN, le serveur vous fournit un jeton de périphérique.

Après avoir ajouté le code ci-dessus, ajoutez ces méthodes à la classe `AppDelegate` :

Rapide

```
func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
```

```
deviceToken: NSData) {
    print("DEVICE TOKEN = \(deviceToken)")
}

func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    print(error)
}
```

Objectif 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);
}
```

Les méthodes ci-dessus sont appelées en fonction du succès de l'enregistrement ou du scénario d'échec.

Appel de scénario de réussite:

Rapide

```
func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    print("DEVICE TOKEN = \(deviceToken)")
}
```

Dans 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)")
}
```

Objectif 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
}
}
```

Appel de scénario d'échec:

Rapide

```
func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    print(error)
}
```

Objectif c

```
- (void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error
```

Remarque

Si aucune des méthodes ci-dessus n'est appelée, votre appareil n'est pas en mesure de créer une connexion fiable avec le serveur APN, ce qui peut être dû à des problèmes d'accès à Internet.

Vérifier si votre application est déjà enregistrée pour Push Notification

Rapide

```
let isPushEnabled = UIApplication.sharedApplication().isRegisteredForRemoteNotifications()
```

Enregistrement pour la notification push (non interactive)

Il est recommandé d'ajouter la logique d'enregistrement de la notification push dans `AppDelegate.swift` car les fonctions de rappel (succès, échec) seront appelées leur. Pour vous inscrire, procédez comme suit:

```
let application = UIApplication.sharedApplication()
let settings = UIUserNotificationSettings(forTypes: [.Alert, .Badge, .Sound], categories: nil)
application.registerUserNotificationSettings(settings)
```

Ensuite, la fonction de rappel `didRegisterUserNotificationSettings` sera appelée et dans ce cas, il vous suffit de déclencher le registre comme ceci:

```
func application(application: UIApplication, didRegisterUserNotificationSettings
notificationSettings: UIUserNotificationSettings) {
    application.registerForRemoteNotifications()
}
```

Et dans ce cas, l'alerte système sera affichée pour demander la persécution pour recevoir une notification push. Une des fonctions de rappel suivantes sera appelée:

```
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)")

}
```

Dans de très rares cas, aucune fonction de rappel de réussite ou d'échec n'est appelée. Cela se produit lorsque vous avez des problèmes de connexion Internet ou que le Sandbox APNS est en panne. Le système effectue un appel API à APNS pour effectuer certaines vérifications, faute de quoi aucune des deux fonctions de rappel ne sera appelée. Visitez l' état du système Apple pour vous assurer que tout va bien.

Gestion des notifications push

Une fois que l'utilisateur clique sur une notification push, la fonction de rappel suivante sera appelée. Vous pouvez analyser le JSON pour obtenir toute information spécifique envoyée par le backend qui vous aidera à établir des liens profonds:

Rapide

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject
: AnyObject]) {
    print("Received notification: \(userInfo)")
}
```

Objectif 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);
}
```

Enregistrement de l'ID d'application à utiliser avec les notifications Push

Choses dont tu as besoin

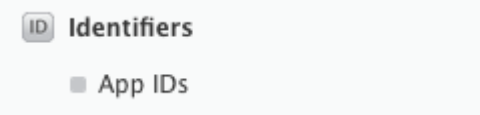
- Une adhésion payée au programme des développeurs Apple
- Un identifiant d'application et un identifiant valides pour votre application (comme com.example.MyApp) qui ne sont utilisés nulle part avant
- Accès à developer.apple.com et au Member Center
- Un périphérique iOS à tester (les notifications Push ne fonctionnent pas sur le simulateur)

Activation de l'accès APNs pour l'ID d'application dans Apple Developer Center

1- Connectez-vous à developer.apple.com Member Center (le lien du compte sur la page d'accueil)

2- Aller à "Certificats"

3- Sélectionnez "App ID" dans le panneau de gauche



4- Cliquez sur "+" en haut à droite

5- Ajouter un identifiant d’application avec l’option Push Notifications cochée

6- Cliquez sur l’ID d’application créé et sélectionnez Modifier

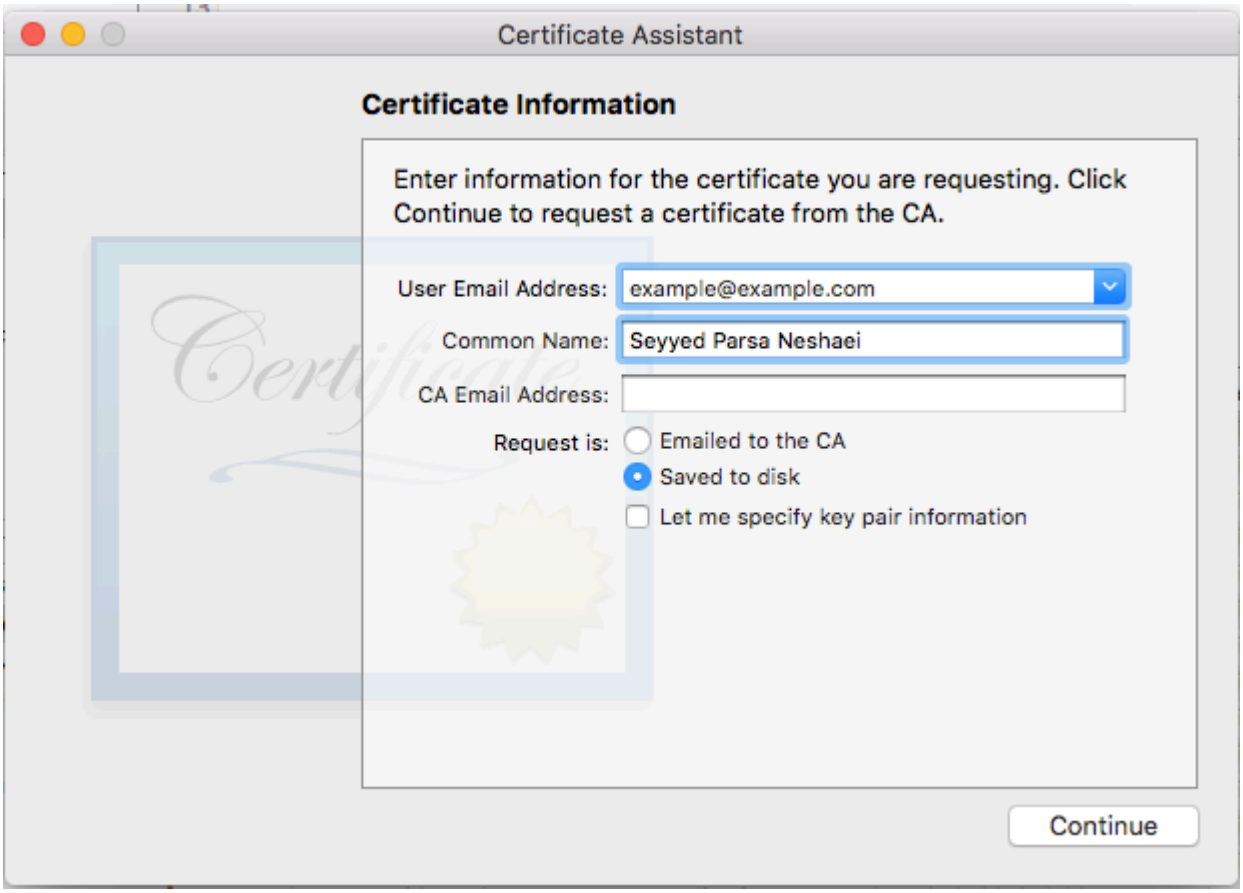
7- Cliquez sur Configurer dans le panneau Notifications push

8- Application Open Keychain Access sur votre Mac

9- Dans le menu Keychain Access, cliquez sur Certificate Assistant -> Demander un certificat à une autorité de certification

10- Entrez votre courrier dans le premier champ de texte

11- Entrez votre nom dans le deuxième champ de texte



12- Laisser l'adresse électronique de CA vide

13- Sélectionnez Enregistré sur le disque plutôt que envoyé par courrier électronique à l'autorité de certification

14- Cliquez sur Continuer et téléchargez le fichier généré

15- Télécharger le fichier généré par Apple et l'ouvrir pendant que Keychain Access est ouvert

Activer l'accès aux APN dans Xcode

1- Sélectionnez votre projet

2- Onglet Open Capabilities

3- Trouvez les notifications push et allumez-le

4-Trouver des modes d'arrière-plan et l'activer et vérifier les notifications à distance

Désinscription des notifications Push

Pour vous désinscrire des notifications à distance par programmation, vous pouvez utiliser

Objectif c

```
[[UIApplication sharedApplication] unregisterForRemoteNotifications];
```

Rapide

```
UIApplication.sharedApplication().unregisterForRemoteNotifications()
```

Cela revient à entrer dans les paramètres de votre téléphone et à désactiver manuellement les notifications pour l'application.

REMARQUE: il peut y avoir de rares cas où vous en auriez besoin (par exemple: lorsque votre application ne prend plus en charge les notifications push)

Si vous souhaitez simplement autoriser l'utilisateur à désactiver temporairement les notifications. Vous devez implémenter une méthode pour supprimer le jeton de périphérique dans la base de données sur votre serveur. Sinon, si vous ne désactivez que la notification localement sur votre appareil, votre serveur enverra toujours des messages.

Définition du numéro de badge de l'icône de l'application

Utilisez le morceau de code suivant pour définir le numéro de badge dans votre application (supposez que `someNumber` a déjà été déclaré):

Objectif c

```
[UIApplication sharedApplication].applicationIconBadgeNumber = someNumber;
```

Rapide

```
UIApplication.shared.applicationIconBadgeNumber = someNumber
```

Pour *supprimer* complètement le badge, il suffit de définir `someNumber = 0` .

Test des notifications push

Il est toujours judicieux de tester le fonctionnement des notifications push avant même que votre serveur ne soit prêt, juste pour vous assurer que tout est correctement configuré de votre côté. Il est assez facile de vous envoyer une notification push en utilisant un script PHP suivant.

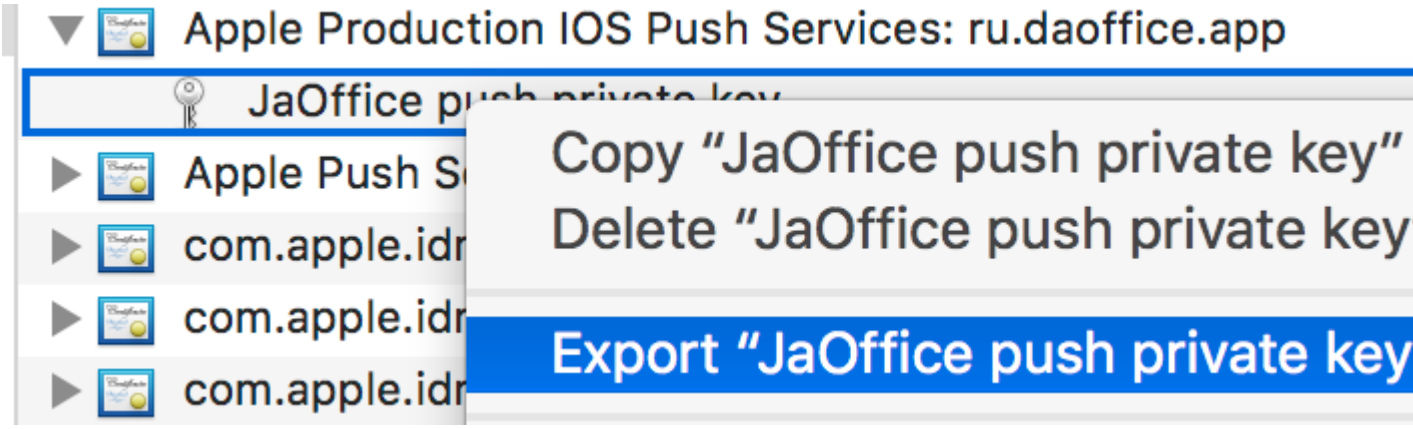
1. Enregistrez le script sous forme de fichier (send_push.php par exemple) dans le même dossier que votre certificat (développement ou production)
2. Modifiez-le pour placer votre jeton de périphérique, mot de passe du certificat
3. Choisissez le bon chemin pour ouvrir une connexion, dev_path ou prod_path (c'est ici que le script ouvre une connexion au serveur APNS)
4. cd dans le dossier du Terminal et lance la commande 'php send_push'
5. Recevoir la notification sur votre appareil

```
<?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);
```

Génération d'un certificat .pem à partir de votre fichier .cer pour le transmettre au développeur du serveur

1. Enregistrer aps.cer dans un dossier
2. Ouvrez "Keychain access" et exportez la clé qui se trouve sous ce certificat dans un fichier .p12 (appelez-la key.p12). Pour ce faire, cliquez dessus avec le bouton droit de la souris et choisissez Exporter. Enregistrez-le dans le même dossier que l'étape 1. À l'exportation, vous serez invité à saisir un mot de passe. Faites quelque chose et mémorisez-le.



3. cd dans ce dossier dans Terminal et exécutez les commandes suivantes:
4. Convertir .cer en certificat .pem

```
openssl x509 -in aps.cer -inform der -out aps.pem
```

5. Convertissez votre clé au format .pem. Pour ouvrir la clé, entrez le mot de passe avec lequel vous l'avez exporté à partir du trousseau, à l'étape 2. Ensuite, entrez un autre mot de passe qui protégera le fichier exporté. Vous serez invité à le saisir deux fois pour confirmation.

```
openssl pkcs12 -nocerts -out key.pem -in key.p12
```

6. Fusionner les fichiers dans un fichier final

```
cat key.pem aps.pem > final_cert.pem
```

7. Le final_cert.pem est le résultat final. Transmettez-le aux développeurs de serveur avec le mot de passe de l'étape 5 afin qu'ils puissent utiliser le certificat protégé.

Lire Notifications push en ligne: https://riptutorial.com/fr/ios/topic/3492/notifications-push

Chapitre 107: NSArray

Introduction

Voici quelques fonctions / méthodes utilitaires utiles qui peuvent être utilisées avec l'extension Array pour que le développeur puisse effectuer certaines opérations critiques sur un tableau à l'aide d'un code à une seule ligne.

Remarques

Une fois que le document actuel est approuvé, il ajoutera aussi beaucoup d'améliorations pour les autres utilisations de tableaux. C'est mon premier document et j'ai besoin de votre aide et de votre approbation dans mes efforts.

Examples

Convertir un tableau en chaîne json

Appelez cette fonction avec l'argument de paramètre sous forme de tableau avec le type 'any'. Il vous rendra json string. La chaîne Json est utilisée pour soumettre un tableau dans l'appel de service Web en tant que paramètre d'entrée de requête dans 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
    }
```

Lire NSArray en ligne: https://riptutorial.com/fr/ios/topic/9248/nsarray

Chapitre 108: NSAttributedString

Remarques

Définir la couleur de la police à l'aide de NSAttributedString

Examples

Création d'une chaîne comportant un crénage personnalisé (espacement des lettres)

`NSAttributedString` (et son frère modifiable `NSMutableAttributedString`) vous permet de créer des chaînes complexes dans leur apparence pour l'utilisateur.

Une application courante consiste à utiliser cette option pour afficher une chaîne et ajouter un crénage / espacement des lettres personnalisé.

Cela se ferait comme suit (où label est un `UILabel`), donnant un crénage différent pour le mot "kerning"

Rapide

```
var attributedString = NSMutableAttributedString("Apply kerning")
attributedString.addAttribute(attribute: NSKernAttributeName, value: 5, range: NSMakeRange(6,
7))
label.attributedText = attributedString
```

Objectif c

```
NSMutableAttributedString *attributedString;
attributedString = [[NSMutableAttributedString alloc] initWithString:@"Apply kerning"];
[attributedString addAttribute:NSKernAttributeName value:@5 range:NSMakeRange(6, 7)];
[label setAttributedText:attributedString];
```

Créer une chaîne avec un texte barré

Objectif c

```
NSMutableAttributedString *attributeString = [[NSMutableAttributedString alloc]
initWithString:@"Your String here"];
[attributeString addAttribute:NSStrikethroughStyleAttributeName
                    value:@2
                    range:NSMakeRange(0, [attributeString length])];
```

Rapide

```
let attributeString: NSMutableAttributedString =  NSMutableAttributedString(string: "Your
```

```
String here")
attributeString.addAttribute(NSStrikethroughStyleAttributeName, value: 2, range:
NSMakeRange(0, attributeString.length))
```

Ensuite, vous pouvez ajouter ceci à votre UILabel:

```
yourLabel.attributedText = attributeString;
```

Ajout de chaînes attribuées et de texte en gras dans 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())]))
```

Le résultat ressemble à:

La valeur est: **Quelque chose que l'utilisateur a entré**

Changer la couleur d'un mot ou d'une chaîne

Objectif 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;
```

Rapide

```
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
```

Note :

La principale consiste à utiliser un `NSMutableAttributedString` et le sélecteur `addAttribute:value:range` avec l'attribut `NSForegroundColorAttributeName` pour changer la couleur d'une plage de chaînes:

```
NSMutableAttributedString *attrsString =  [[NSMutableAttributedString alloc]
initWithAttributedString:label.attributedText];
[attrsString addAttribute:NSForegroundColorAttributeName value:color range:range];
```

Vous pouvez utiliser un autre moyen pour obtenir la plage, par exemple: NSRegularExpression.

Supprimer tous les attributs

Objectif c

```
NSMutableAttributedString *mutAttString = @"string goes here";
NSRange range = NSMakeRange(0, mutAttString.length);
[mutAttString setAttributes:@{} range:originalRange];
```

Conformément à la documentation Apple, nous utilisons `setAttributes` et pas `addAttribute` .

Rapide

```
mutAttString.setAttributes([:], range: NSRange(0..<string.length))
```

Lire NSAttributedString en ligne: https://riptutorial.com/fr/ios/topic/979/nsattributedstring

Chapitre 109: NSBundle

Examples

Obtenir le paquet principal

1. Obtenir une référence au bundle principal en utilisant Cocoa.

   Pour obtenir le faisceau principal dans l' application Cocoa, appelez la méthode de classe ***mainBundle*** de la classe **NSBundle.**

```
   NSBundle *mainBundle;
   // Get the main bundle for the app;
   mainBundle = [NSBundle mainBundle];
```

2. Obtenir une référence au bundle principal en utilisant Core Foundation.

   Utilisez la fonction ***CFBundleGetMainBundle*** pour récupérer le bundle principal de votre application basée sur C.

```
   CFBundleRef mainBundle;
   // Get the main bundle for the app
   mainBundle = CFBundleGetMainBundle();
```

Obtenir Bundle par chemin

1. Localisation d'un paquet de cacao en utilisant son chemin

   Pour obtenir le bundle sur un chemin spécifique en utilisant Cocoa, appelez la méthode ***bundleWithPath:*** class du **NSBundle**

```
   NSBundle *myBundle;
   // obtain a reference to a loadable bundle
   myBundle = [NSBundle bundleWithPath:@"/Library/MyBundle.bundle";
```

2. Localisation d'un bundle Cocoa Foundation en utilisant son Path

   Pour obtenir le bundle sur un chemin spécifique à l'aide de Core Foundation, appelez la fonction ***CFBundleCreate*** et utilisez le type **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);
```

Lire NSBundle en ligne: https://riptutorial.com/fr/ios/topic/5862/nsbundle

Chapitre 110: NSData

Remarques

Ressources utiles

Documentation Apple (NSData)

NSData.dataWithContentsOfFile ()

NSData.bytes

Examples

Création d'objets NSData

Utiliser un fichier

Rapide

```
let data = NSData(contentsOfFile: filePath) //assuming filePath is a valid path
```

Objectif c

```
NSData *data = [NSData dataWithContentsOfFile:filePath]; //assuming filePath is a valid path
```

Utiliser un objet String

Rapide

```
let data = (string as NSString).dataUsingEncoding(NSUTF8StringEncoding) //assuming string is a
String object
```

Objectif c

```
NSData *data = [string dataUsingEncoding:NSUTF8StringEncoding]; //assuming string is a String
object
```

Conversion de NSData en d'autres types

À ficeler

Rapide

```
let string = String(NSString(data: data, encoding: NSUTF8StringEncoding)) //assuming data is a valid NSData object
```

Objectif c

```
NSString *string = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
//assuming data is a valid NSData object
[string release];
```

Se ranger

Rapide

```
let array = data.bytes as! NSMutableArray //assuming data is a valid NSData object
```

Objectif c

```
NSMutableArray *array = (NSMutableArray *)[data bytes]; //assuming data is a valid NSData object
```

À Bytes Array

Rapide

```
let bytesArray = data.bytes as! UInt8 //assuming data is a valid NSData object
```

Objectif c

```
UInt8 *bytesArray = (UInt8 *)data.bytes; //assuming data is a valid NSData object
```

Conversion de NSData en chaîne HEX

`NSData`

peut être représenté sous forme de chaîne hexadécimale, similaire à ce qu'il affiche dans sa méthode de `description` .

Rapide

```
extension NSData {

    func hexString() -> String {
        return UnsafeBufferPointer<UInt8>(start: UnsafePointer<UInt8>(bytes), count: length)
            .reduce("") { $0 + String(format: "%02x", $1) }
    }

}
```

Objectif 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
```

Lire NSData en ligne: https://riptutorial.com/fr/ios/topic/5084/nsdata

Chapitre 111: NSDate

Syntaxe

- NSDate () // objet NSDate init à la date et l'heure actuelles
- NSDate (). TimeIntervalSince1970 // Date et heure actuelles en nombre de secondes entre 00:00:00 UTC le 1er janvier 1970.
- NSDate (). Compare (other: NSDate) // Retourne une comparaison de la date du jour à une autre date renvoie un `NSComparisonResult`

Remarques

Il existe différents types de format de date que vous pouvez définir: Voici leur liste complète.

Format	Signification / Description	Exemple 1	Exemple2
y	Une année avec au moins 1 chiffre.	175 après JC → “175”	2016 AD → “2016”
yy	Une année avec exactement 2 chiffres.	5 AD → “05”	2016 AD → “16”
yyy	Une année avec au moins 3 chiffres.	5 AD → “005”	2016 AD → “2016”
aaaa	Une année avec au moins 4 chiffres.	5 AD → “0005”	2016 AD → “2016”
M	Un mois avec au moins 1 chiffre.	Juillet → 7	"Novembre" → "11"
MM	Un mois avec au moins 2 chiffres.	Juillet → 07	"Novembre" → "11"
MMM	Abréviation de trois lettres par mois.	Juillet → “Jul”	"Novembre" → "Nov"
MMMM	Nom complet du mois.	Juillet → juillet	"Novembre" → "Novembre"
MMMMM	Abréviation d'un mois (janvier, juin et juillet, tous auront un «J»).	Juillet → “J”	"Novembre" → "N"
ré	Jour avec au moins un chiffre.	8 → “8”	29 → “29”
dd	Jour avec au moins deux chiffres.	8 → “08”	29 → “29”

Format	Signification / Description	Exemple 1	Exemple2
"E", "EE" ou "EEE"	Abréviation de jour en 3 lettres du nom du jour.	Lundi → “Mon”	Jeudi → “jeu”
EEEE	Nom de la journée complète	Lundi → lundi	Jeudi → jeudi
EEEEE	Abréviation de 1 lettre jour du nom du jour (le jeu et le mardi seront «T»)	Lundi → “M”	Jeudi → “T”
EEEEEE	Abréviation de jour en 2 lettres du nom du jour.	Lundi → “Mo”	Jeudi → “Th”
une	Période de la journée (AM / PM).	22 heures → «PM»	2 heures du matin → "AM"
h	Une heure basée sur 1-12 avec au moins 1 chiffre.	22h → “10”	2 heures → «2»
hh	Une heure basée sur 1-12 avec au moins 2 chiffres.	22h → 10h	2 heures → «02»
H	Une heure basée sur 0-23 avec au moins 1 chiffre.	22h → “14”	2 heures → «2»
HH	Une heure basée sur 0-23 avec au moins 2 chiffres.	22h → “14”	2 heures → «02»
m	Une minute avec au moins 1 chiffre.	7 → “7”	29 → “29”
mm	Une minute avec au moins 2 chiffres.	7 → “07”	29 → “29”
s	Une seconde avec au moins 1 chiffre.	7 → “7”	29 → “29”
ss	Une seconde avec au moins 2 chiffres.	7 → “07”	29 → “29”

Il y en a beaucoup d'autres, pour obtenir un temps différent en fonction de la zone (z), pour obtenir du temps avec des détails en millisecondes (S), etc.

Examples

Obtenir la date actuelle

Obtenir la date actuelle est très facile. Vous obtenez l'objet NSDate de la date actuelle en une seule ligne comme suit:

Rapide

```
var date = NSDate()
```

Swift 3

```
var date = Date()
```

Objectif c

```
NSDate *date = [NSDate date];
```

Obtenir l'objet NSDate N secondes à partir de la date actuelle

Le nombre de secondes à partir de la date et de l'heure actuelles pour la nouvelle date. Utilisez une valeur négative pour spécifier une date avant la date actuelle.

Pour ce faire, nous avons une méthode nommée `dateWithTimerIntervalSinceNow(seconds: NSTimeInterval) -> NSDate` (Swift) ou `+ (NSDate*)dateWithTimeIntervalSinceNow:(NSTimeInterval)seconds` (Objective-C).

Maintenant, par exemple, si vous avez besoin d'une date d'une semaine à partir de la date actuelle et d'une semaine à la date actuelle, nous pouvons le faire comme.

Rapide

```
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)
```

Objectif 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);
```

Comparaison de date

Il existe 4 méthodes pour comparer les dates:

Rapide

- `isEqualToDate(anotherDate: NSDate) -> Bool`
- `earlierDate(anotherDate: NSDate) -> NSDate`
- `laterDate(anotherDate: NSDate) -> NSDate`
- `compare(anotherDate: NSDate) -> NSComparisonResult`

Objectif c

- `- (BOOL)isEqualToDate:(NSDate *)anotherDate`
- `- (NSDate *)earlierDate:(NSDate *)anotherDate`
- `- (NSDate *)laterDate:(NSDate *)anotherDate`
- `- (NSComparisonResult)compare:(NSDate *)anotherDate`

Disons que nous avons 2 dates:

Rapide

```
let date1: NSDate = ... // initialized as  July 7, 2016 00:00:00
let date2: NSDate = ... // initialized as  July 2, 2016 00:00:00
```

Objectif c

```
NSDate *date1 = ... // initialized as  July 7, 2016 00:00:00
NSDate *date2 = ... // initialized as  July 2, 2016 00:00:00
```

Ensuite, pour les comparer, nous essayons ce code:

Rapide

```
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
}
```

Objectif 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
}
```

Si vous voulez comparer les dates et gérer les secondes, les semaines, les mois et les années:

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")
```

Ou vous pouvez l'utiliser pour chaque composant:

```
// 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
```

Obtenez le temps Unix Epoch

Pour obtenir l' époque Unix , utilisez la constante `timeIntervalSince1970` :

Rapide

```
let date = NSDate() // current date
let unixtime = date.timeIntervalSince1970
```

Objectif c

```
NSDate *date = [NSDate date]; // current date
int unixtime = [date timeIntervalSince1970];
```

NSDateFormatter

La conversion d'un objet `NSDate` en chaîne ne `NSDate` que 3 étapes.

1. Créez un objet `NSDateFormatter`

Rapide

```
let dateFormatter = NSDateFormatter()
```

Swift 3

```
let dateFormatter = DateFormatter()
```

Objectif c

```
NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
```

2. Définissez le format de date dans lequel vous voulez que votre chaîne

Rapide

```
dateFormatter.dateFormat = "yyyy-MM-dd 'at' HH:mm"
```

Objectif c

```
dateFormatter.dateFormat = @"yyyy-MM-dd 'at' HH:mm";
```

3. Obtenez la chaîne formatée

Rapide

```
let date = NSDate() // your NSDate object
let dateString = dateFormatter.stringFromDate(date)
```

Swift 3

```
let date = Date() // your NSDate object
let dateString = dateFormatter.stringFromDate(date)
```

Objectif c

```
NSDate *date = [NSDate date]; // your NSDate object
NSString *dateString = [dateFormatter stringFromDate:date];
```

Cela donnera quelque chose comme ceci: `2001-01-02 at 13:00`

Remarque

La création d'une instance `NSDateFormatter` est une opération coûteuse. Il est donc recommandé de la créer une fois et de la réutiliser si possible.

Extension utile pour convertir la date en chaîne.

```
extension Date {
        func toString() -> String {
            let dateFormatter = DateFormatter()
            dateFormatter.dateFormat = "MMMM dd yyyy"
            return dateFormatter.string(from: self)
        }
}
```

Des liens utiles pour une formation rapide des dates rapidement-get-lisible par l'homme-date-nsdateformatter .

Pour construire des formats de date, voir les modèles de format de date .

Convertir NSDate composé d'heures et de minutes (uniquement) en un NSDate complet

Il y a de nombreux cas où l'on a créé un NSDate uniquement à partir d'un format heure et minute, à savoir: 08:12 qui renvoie depuis un serveur en tant que chaîne et que vous initiez une instance NSDate uniquement avec ces **valeurs.**

L'inconvénient de cette situation est que votre NSDate est presque complètement "nu" et que vous devez créer: jour, mois, année, seconde et fuseau horaire pour que cet objet "joue" avec d'autres types de NSDate.

Pour l'exemple, supposons que hourAndMinute est le type NSDate composé du format heure et minute:

Objectif 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];
```

Maintenant, votre objet est le contraire total d'être "nu".

UTC Time offset à partir de NSDate avec TimeZone

Ici, le temps `UTC` est calculé à partir des données actuelles dans le fuseau horaire souhaité.

```
+(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;
}
```

Obtenir le type de cycle horaire (12 heures ou 24 heures)

Vérifier si la date actuelle contient le symbole pour AM ou PM

Objectif 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);
```

Demander le type de cycle temporel à partir de `NSDateFormatter`

Objectif c

```
NSString *formatStringForHours = [NSDateFormatter dateFormatFromTemplate:@"j" options:0
```

```
locale:[NSLocale currentLocale]];
NSRange containsA = [formatStringForHours rangeOfString:@"a"];
BOOL is24h = containsA.location == NSNotFound;
```

Cela utilise une chaîne de modèle de date spéciale appelée "j" qui, selon les spécifications ICU ...

> [...] demande le format d'heure préféré pour les paramètres régionaux (h, H, K ou k), déterminé par l'attribut préféré de l'élément hours dans les données supplémentaires. [...] Notez que l'utilisation de 'j' dans un squelette transmis à une API est la seule façon de demander à un squelette le type de cycle temporel préféré de l'environnement local (12 heures ou 24 heures).

Cette dernière phrase est importante. Il "est le seul moyen de demander à un squelette le type de cycle temporel préféré de l'environnement local". Puisque `NSDateFormatter` et `NSCalendar` sont construits sur la bibliothèque ICU, la même chose est vraie ici.

Référence

La deuxième option est dérivée de cette réponse .

Obtenir NSDate à partir du format de date JSON "/ Date (1268123281843) /"

Avant Json.NET 4.5, les dates étaient écrites au format Microsoft: "/ Date (1198908717056) /". Si votre serveur envoie la date dans ce format, vous pouvez utiliser le code ci-dessous pour le sérialiser à NSDate:

Objectif 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;
}
```

Obtenez le temps historique de NSDate (ex: 5 il y a, il y a 2 heures, 3 heures)

Cela peut être utilisé dans diverses applications de discussion, flux RSS et applications sociales où vous devez disposer des derniers flux avec horodatage:

Objectif 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;
}
```

Lire NSDate en ligne: https://riptutorial.com/fr/ios/topic/1502/nsdate

Chapitre 112: NSHTTPCookieStorage

Examples

Stockez et lisez les cookies de 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")
    }
}

}
```

Lire NSHTTPCookieStorage en ligne: https://riptutorial.com/fr/ios/topic/7312/nshttpcookiestorage

Chapitre 113: NSInvocation

Examples

NSInvocation Objective-C

Reportez-vous à ce message original par e.James

Selon la référence de classe NSInvocation d' Apple :

> Un `NSInvocation` est un message Objective-C rendu statique, c'est-à-dire une action transformée en objet.

Et, un *peu* plus en détail:

Le concept de message est au cœur de la philosophie objectif-c. Chaque fois que vous appelez une méthode ou accédez à une variable d'un objet, vous lui envoyez un message. `NSInvocation` est pratique lorsque vous souhaitez envoyer un message à un objet à un autre moment ou envoyer le même message plusieurs fois. `NSInvocation` vous permet de *décrire* le message que vous allez envoyer, puis de l' *invoquer* (l'envoyer effectivement à l'objet cible) ultérieurement.

Par exemple, supposons que vous souhaitiez ajouter une chaîne à un tableau. Vous devriez normalement envoyer le message `addObject:` comme suit:

```
[myArray addObject:myString];
```

Maintenant, supposons que vous souhaitiez utiliser `NSInvocation` pour envoyer ce message à un autre moment:

Tout d'abord, vous devez préparer un objet `NSInvocation` à utiliser avec `addObject:` selector de `NSMutableArray` :

```
NSMethodSignature * mySignature = [NSMutableArray
    instanceMethodSignatureForSelector:@selector(addObject:)];
NSInvocation * myInvocation = [NSInvocation
    invocationWithMethodSignature:mySignature];
```

Ensuite, vous spécifiez quel objet envoyer le message à:

```
[myInvocation setTarget:myArray];
```

Spécifiez le message que vous souhaitez envoyer à cet objet:

```
[myInvocation setSelector:@selector(addObject:)];
```

Et remplissez tous les arguments pour cette méthode:

```
[myInvocation setArgument:&myString atIndex:2];
```

Notez que les arguments d'objet doivent être passés par un pointeur. Merci à Ryan McCuaig de l' avoir signalé, et veuillez consulter la documentation d' Apple pour plus de détails.

À ce stade, `myInvocation` est un objet complet, décrivant un message pouvant être envoyé. Pour envoyer le message, vous devez appeler:

```
[myInvocation invoke];
```

Cette dernière étape entraînera l'envoi du message, en exécutant essentiellement `[myArray addObject:myString]; .`

Pensez-y comme envoyer un email. Vous ouvrez un nouvel email (objet `NSInvocation`), `NSInvocation` l'adresse de la personne (objet) à qui vous voulez l'envoyer, tapez un message pour le destinataire (spécifiez un `selector` et des arguments), puis cliquez sur "envoyer" (appel `invoke`).

Voir Utilisation de NSInvocation pour plus d'informations.

`NSUndoManager` utilise des objets `NSInvocation` pour pouvoir *inverser les* commandes. Essentiellement, vous créez un objet `NSInvocation` pour dire: "Hé, si vous voulez annuler ce que je viens de faire, envoyez ce message à cet objet, avec ces arguments". Vous donnez l'objet `NSInvocation` au `NSUndoManager` et il ajoute cet objet à un tableau d'actions annulables. Si l'utilisateur appelle "Annuler", `NSUndoManager` recherche simplement l'action la plus récente du tableau et appelle l'objet `NSInvocation` stocké pour effectuer l'action nécessaire.

Voir Enregistrement des opérations d'annulation pour plus de détails.

Lire NSInvocation en ligne: https://riptutorial.com/fr/ios/topic/8276/nsinvocation

Chapitre 114: NSNotificationCenter

Introduction

Les notifications iOS sont un moyen simple et puissant d'envoyer les données de manière souple. En d'autres termes, l'expéditeur d'une notification n'a pas à se soucier de savoir qui (le cas échéant) reçoit la notification, il la publie simplement sur le reste de l'application et elle peut être extraite par beaucoup de choses ou par rien. l'état de votre application.

Source : - HACKING avec Swift

Paramètres

Paramètre	Détails
prénom	Le nom de la notification pour laquelle enregistrer l'observateur; c'est-à-dire que seules les notifications portant ce nom sont utilisées pour ajouter le bloc à la file d'attente des opérations. Si vous omettez, le centre de notification n'utilise pas le nom d'une notification pour décider d'ajouter le bloc à la file d'attente des opérations.
obj	L'objet dont l'observateur souhaite recevoir les notifications; c'est-à-dire que seules les notifications envoyées par cet expéditeur sont envoyées à l'observateur. Si vous ne transmettez aucune information, le centre de notification n'utilise pas l'expéditeur d'une notification pour décider de la remettre à l'observateur.
queue	La file d'attente d'opération à quel bloc devrait être ajoutée. Si vous passez nil, le bloc est exécuté de manière synchrone sur le thread de publication.
bloc	Le bloc à exécuter lors de la réception de la notification. Le bloc est copié par le centre de notification et (la copie) est conservé jusqu'à ce que l'enregistrement de l'observateur soit supprimé.

Remarques

Un objet NSNotificationCenter (ou simplement un centre de notification) fournit un mécanisme de diffusion d'informations dans un programme. Un objet NSNotificationCenter est essentiellement une table de distribution des notifications.

Pour plus d'informations, consultez la documentation Apple ici

NSNotification & NSNotificationCenter dans Swift

Examples

Ajouter un observateur

Convention de nommage

Les notifications sont identifiées par des objets NSString globaux dont les noms sont composés de cette manière:

`Name of associated class` + `Did | Will` + `UniquePartOfName` + `Notification`

Par exemple:

- 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)
```

Objectif c

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                      selector:@selector(testNotification:)
                                      name:@"TestNotification"
                                      object:nil];
```

PS: Il est également intéressant de noter que le nombre de fois qu'un observateur a été ajouté doit être exactement le nombre de fois que l'observateur est supprimé. Une erreur de débutant

consiste à ajouter un observateur dans `viewWillAppear:` d'un UIViewController, mais supprimer l'observateur dans `viewDidUnload:` entraînera un nombre irrégulier de poussées et donc une fuite de l'observateur et le rappel du sélecteur de notification de manière superflue.

Supprimer des observateurs

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)
```

Objectif c

```
//Remove observer for single notification
[[NSNotificationCenter defaultCenter] removeObserver:self name:@"TestNotification"
object:nil];

//Remove observer for all notifications
[[NSNotificationCenter defaultCenter] removeObserver:self];
```

Poster une notification

Rapide

```
NSNotificationCenter.defaultCenter().postNotificationName("TestNotification", object: self)
```

Objectif c

```
[[NSNotificationCenter defaultCenter] postNotificationName:@"TestNotification" object:nil];
```

Publication d'une notification avec des données

Rapide

```
let userInfo: [String: AnyObject] = ["someKey": myObject]
NSNotificationCenter.defaultCenter().postNotificationName("TestNotification", object: self,
userInfo: userInfo)
```

Objectif c

```
NSDictionary *userInfo = [NSDictionary dictionaryWithObject:myObject forKey:@"someKey"];
[[NSNotificationCenter defaultCenter] postNotificationName: @"TestNotification" object:nil
userInfo:userInfo];
```

Observation d'une notification

Rapide

```
func testNotification(notification: NSNotification) {
    let userInfo = notification.userInfo
    let myObject: MyObject = userInfo["someKey"]
}
```

Objectif c

```
- (void)testNotification:(NSNotification *)notification {
    NSDictionary *userInfo = notification.userInfo;
    MyObject *myObject = [userInfo objectForKey:@"someKey"];
}
```

Ajout / Suppression d'un observateur avec un bloc

Au lieu d'ajouter un observateur avec un sélecteur, un bloc peut être utilisé:

```
id testObserver = [[NSNotificationCenter defaultCenter] addObserverForName:@"TestNotification"
                                                                    object:nil
                                                                     queue:nil
                                                                usingBlock:^(NSNotification*
notification) {
    NSDictionary *userInfo = notification.userInfo;
    MyObject *myObject = [userInfo objectForKey:@"someKey"];
}];
```

L'observateur peut alors être retiré avec:

```
[[NSNotificationCenter defaultCenter] removeObserver:testObserver
                                                name:@"TestNotification"
                                              object:nil];
```

Ajouter et supprimer un observateur pour le nom

```
// 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)
```

Lire NSNotificationCenter en ligne: https://riptutorial.com/fr/ios/topic/1601/nsnotificationcenter

Chapitre 115: NSPredicate

Syntaxe

- Substitions au format prédicat
  - Spécificateurs de chaîne au format C:% d,% s,% f, etc.
  - Substitution d'objet:% @
  - Substitution clavier:% K
- Opérateurs de comparaison de prédicats
  - =, ==: l'expression de gauche est égale à l'expression de droite
  - > =, =>: L'expression à gauche est supérieure ou égale à l'expression à droite
  - <=, = <: L'expression de gauche est inférieure ou égale à l'expression de droite
  - >: L'expression à gauche est plus grande que l'expression à droite
  - <: L'expression de gauche est moins que l'expression de droite
  - ! =, <>: L'expression de gauche n'est pas égale à l'expression de droite
  - BETWEEN: L'expression de gauche est comprise entre ou égale à l'une des valeurs de l'expression de droite, ce qui spécifie les limites inférieure et supérieure - ex: BETWEEN {0, 5}
- Opérateurs composés de prédicats
  - AND, &&: logique ET
  - OU, ||: logique OU
  - PAS,!: Logique PAS
- Opérateurs de comparaison de chaînes de prédicats
  - BEGINSWITH: L'expression de la main gauche commence par l'expression de la main droite
  - ENDSWITH: L'expression de la main gauche se termine par l'expression de la main droite
  - CONTIENT: l'expression de gauche contient l'expression de droite
  - LIKE: l'expression de gauche est égale à l'expression de droite, avec substitution de caractère générique
    - *: Correspond à zéro ou plusieurs caractères
    - ?: Correspond à un caractère

Examples

Créer un NSPredicate en utilisant predicateWithBlock

Objectif c

```
NSPredicate *predicate = [NSPredicate predicateWithBlock:^BOOL(id item,
                                                               NSDictionary *bindings) {
    return [item isKindOfClass:[UILabel class]];
}];
```

Rapide

```
let predicate = NSPredicate { (item, bindings) -> Bool in
    return item.isKindOfClass(UILabel.self)
}
```

Dans cet exemple, le prédicat correspond aux éléments de la classe `UILabel` .

Créer un NSPredicate en utilisant predicateWithFormat

Objectif c

```
NSPredicate *predicate = [NSPredicate predicateWithFormat: @"self[SIZE] = %d", 5)];
```

Rapide

```
let predicate = NSPredicate(format: "self[SIZE] >= %d", 5)
```

Dans cet exemple, le prédicat correspond à des éléments de tableaux ayant au moins 5 de longueur.

Création d'un NSPredicate avec des variables de substitution

Un `NSPredicate` peut utiliser des variables de substitution pour permettre de lier les valeurs à la volée.

Objectif c

```
NSPredicate *template = [NSPredicate predicateWithFormat: @"self BEGINSWITH $letter"];
NSDictionary *variables = @{ @"letter": @"r" };
NSPredicate *beginsWithR = [template predicateWithSubstitutionVariables: variables];
```

Rapide

```
let template = NSPredicate(format: "self BEGINSWITH $letter")
let variables = ["letter": "r"]
let beginsWithR = template.predicateWithSubstitutionVariables(variables)
```

Le prédicat de modèle n'est pas modifié par `predicateWithSubstitutionVariables` . Au lieu de cela, une copie est créée et cette copie reçoit les variables de substitution.

Utiliser NSPredicate pour filtrer un tableau

Objectif 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"]
```

Rapide

```
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"]
```

Validation du formulaire à l'aide de 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 avec la condition `AND`,` OR` et `NOT`

Le prédicat conditionnel sera plus propre et plus sûr en utilisant la classe `NSCompoundPredicate` qui fournit des opérateurs booléens de base pour les prédicats donnés.

Objectif c

ET - Condition

```
  NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
  NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
  NSPredicate *combinedPredicate = [NSCompoundPredicate andPredicateWithSubpredicates:
@[predicate,anotherPredicate]];
```

OU - Condition

```
 NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
 NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
 NSPredicate *combinedPredicate = [NSCompoundPredicate orPredicateWithSubpredicates:
@[predicate,anotherPredicate]];
```

NON - Condition

```
 NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
 NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
 NSPredicate *combinedPredicate = [NSCompoundPredicate notPredicateWithSubpredicate:
@[predicate,anotherPredicate]];
```

Lire NSPredicate en ligne: https://riptutorial.com/fr/ios/topic/5796/nspredicate

Chapitre 116: NSTimer

Paramètres

Paramètre	Détails
`interval`	Le temps, en secondes, à attendre avant de déclencher la minuterie; ou, en répétant les minuteries, le temps entre les tirs.
`target`	L'objet sur lequel appeler le `selector`
`selector`	Dans Swift, un objet `Selector` spécifiant la méthode à utiliser sur la `target`
`repeats`	Si la valeur est `false` , déclenchez la minuterie une seule fois. Si `true` , le feu de la minuterie chaque `interval` secondes.

Remarques

Un `NSTimer` vous permet d'envoyer un message à une cible après une période de temps spécifiée.

Examples

Créer une minuterie

Cela créera une minuterie pour appeler la méthode `doSomething` sur lui- `self` en 5 secondes.

Rapide

```
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)
```

Objectif c

```
NSTimer *timer = [NSTimer scheduledTimerWithTimeInterval:5.0 target:self
selector:@selector(doSomething) userInfo:nil repeats:NO];
```

Régler les répétitions sur `false/NO` indique que nous voulons que le minuteur ne se déclenche qu'une seule fois. Si nous définissons cette valeur sur `true/YES` , elle se déclenche toutes les cinq secondes jusqu'à ce qu'elle soit invalidée manuellement.

Lancer manuellement une minuterie

Rapide

```
timer.fire()
```

Objectif c

```
[timer fire];
```

L'appel de la méthode d' `fire` provoque l'exécution par NSTimer de la tâche qu'il aurait normalement effectuée sur une planification.

Dans un **minuteur non répétitif** , cela invalidera automatiquement le minuteur. En d'autres termes, le fait d'appeler le `fire` avant l'intervalle de temps ne produira qu'une seule invocation.

Dans un compte à **rebours** , cela invoquera simplement l'action sans interrompre le programme habituel.

Invalider une minuterie

Rapide

```
timer.invalidate()
```

Objectif c

```
[timer invalidate];
```

Cela empêchera la minuterie de se déclencher. **Doit être appelé à partir du thread dans lequel le minuteur a été créé,** voir les notes d'Apple :

> Vous devez envoyer ce message à partir du thread sur lequel la minuterie a été installée. Si vous envoyez ce message à partir d'un autre thread, la source d'entrée associée à la minuterie peut ne pas être supprimée de sa boucle d'exécution, ce qui peut empêcher la sortie du thread correctement.

Notes: Une fois le minuteur désactivé, il est impossible de déclencher le même minuteur invalidé. Au lieu de cela, vous devez initialiser à nouveau le minuteur invalidé et déclencher la méthode de tir.

Options de fréquence de minuterie

Minuterie répétée

Rapide

```
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()
        }
    }
```

Doit être invalidé manuellement si vous le souhaitez.

Rapide

Événement retardé non répété

```
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)
```

La minuterie sera déclenchée une fois, 3 secondes après le moment de l'exécution. Sera invalidé automatiquement, une fois tiré.

Transmission de données à l'aide de la minuterie

Si vous souhaitez transmettre des données avec la temporisation, vous pouvez le faire avec le paramètre `userInfo` .

Voici l'approche simple qui donne une brève idée de la manière dont vous pouvez transmettre les données à la méthode déclenchée à partir du minuteur.

[*Swift 3*]

```
Timer.scheduledTimer(timeInterval: 1.0, target: self, selector:#selector(iGotCall(sender:)),
userInfo: ["Name": "i am iOS guy"], repeats:true)
```

[*Objectif - C*]

```
NSTimer* timer = [NSTimer scheduledTimerWithTimeInterval:1.0
                                                  target:self
                                                selector:@selector(iGotCall:)
                                                userInfo:@"i am iOS guy" repeats:YES];
```

La ligne de code ci-dessus qui passe `["Name": "i am iOS guy"]` dans le `userInfo` . Donc, maintenant, lorsque l'appel `iGotCall` reçu, vous pouvez obtenir la valeur transmise comme ci-dessous.

[*Swift 3*]

```
func iGotCall(sender: Timer) {
        print((sender.userInfo)!)
    }
```

[*Objectif - C*]

```
- (void)iGotCall:(NSTimer*)theTimer {
    NSLog (@"%@", (NSString*)[theTimer userInfo]);
}
```

Lire NSTimer en ligne: https://riptutorial.com/fr/ios/topic/2624/nstimer

Chapitre 117: NSURL

Examples

Comment obtenir le dernier composant de chaîne de NSURL String.

```
NSURL *url = [NSURL URLWithString:@"http://www.example.com/images/apple-tree.jpg"];
NSString *fileName = [url lastPathComponent];
// fileName = "apple-tree.jpg"
```

Comment obtenir le dernier composant de chaîne de l'URL (NSURL) dans 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
```

Lire NSURL en ligne: https://riptutorial.com/fr/ios/topic/4610/nsurl

Chapitre 118: NSURLConnection

Examples

Méthodes de délégation

// conforme le protocole NSURLConnectionDelegate.

```
@interface ViewController : UIViewController<NSURLConnectionDelegate>
{
    NSMutableData *_responseData;
}
```

// Implémentation des méthodes du protocole 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
}
```

Demande synchrone

```
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
}
```

Demande asynchrone

```
// 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];
```

Lire NSURLConnection en ligne: https://riptutorial.com/fr/ios/topic/6004/nsurlconnection

Chapitre 119: NSURLSession

Remarques

La classe **NSURLSession** et les classes associées fournissent une API pour télécharger du contenu. Cette API fournit un ensemble complet de méthodes de délégation pour prendre en charge l'authentification et permet à votre application d'effectuer des téléchargements en arrière-plan lorsque votre application n'est pas en cours d'exécution ou, sous iOS, lorsque votre application est suspendue.

À un niveau élevé, **NSURLSession** est basé sur le concept de sessions et de tâches. Une tâche représente une requête unique pour une seule URL (ou un seul téléchargement vers une seule URL). Une session est un groupe de requêtes associées.

Le système d'exploitation fournit une seule session préexistante - la session partagée, qui fonctionne essentiellement comme NSURLConnection. De plus, vous pouvez créer vos propres sessions dans votre application si nécessaire.

Différentes applications utilisent des sessions de différentes manières. De nombreuses applications créent une seule session au lancement et continuent de la réutiliser. Les autres applications peuvent être en mesure d'annuler un groupe de tâches associées (par exemple, un navigateur Web annulant toutes les requêtes en attente lorsque vous fermez un onglet) et créer ainsi une session pour chaque groupe de requêtes associées.

La première étape lorsque vous utilisez NSURLSession consiste à créer un objet de configuration de session. L'objet (généralement) réutilisable contient divers paramètres de session que vous pouvez modifier pour vos besoins particuliers, tels que l'accès simultané maximal, des en-têtes supplémentaires à envoyer avec chaque requête, l'autorisation d'envoi de requêtes via la radio cellulaire (iOS uniquement), stockage des informations d'identification, version minimale de TLS et même paramètres de proxy.

Il existe trois types de configurations de session, en fonction de la manière dont vous souhaitez que la session résultante se comporte:

- **Les configurations par défaut** créent des sessions qui fonctionnent comme NSURLConnection.
- **Les configurations d'arrière-plan** créent des sessions dans lesquelles les demandes sont exécutées hors processus, permettant aux téléchargements de continuer même lorsque l'application n'est plus en cours d'exécution.
- **Les configurations éphémères** créent des sessions qui ne mettent rien en mémoire cache sur le disque, ne stockent pas les cookies sur le disque, etc.

Lorsque vous créez une configuration d'arrière-plan, vous devez fournir un identifiant de session qui vous permet de réassocier la session d'arrière-plan ultérieurement (si votre application se ferme ou est suspendue ou fermée par le système d'exploitation). Vous ne devez pas avoir plus d'une instance d'une session avec le même identifiant actif dans votre application. En règle

générale, ces configurations ne sont pas réutilisables. Toutes les autres configurations de session peuvent être réutilisées pour créer autant de sessions que vous le souhaitez. Donc, si vous devez créer plusieurs sessions avec des paramètres similaires, vous pouvez créer la configuration une fois et la réutiliser chaque fois que vous créez une nouvelle session.

Après avoir créé une session, vous pouvez créer des tâches dans cette session. Il existe trois types de tâches:

- **Les tâches de** données renvoient des données en tant qu'objet **NSData** . Celles-ci conviennent à un usage général, mais ne sont pas prises en charge dans les sessions en arrière-plan.
- **Les tâches de téléchargement** renvoient des données sous forme de fichier sur le disque. Celles-ci conviennent aux requêtes plus importantes ou aux sessions en arrière-plan.
- **Télécharger des tâches pour** télécharger des données depuis un objet NSData ou depuis un fichier sur le disque. Vous fournissez un objet ou un fichier de données qui fournit le corps POST. Le fichier / données de corps que vous fournissez dans la tâche remplace tous les fichiers / données de corps fournis dans l'objet NSURLRequest (le cas échéant).

Chacun de ces types vous permet d'obtenir les données de réponse de différentes manières, soit en utilisant des rappels basés sur des blocs, soit en fournissant un délégué sur la session et en implémentant des méthodes déléguées.

De plus, NSURLSession vous permet de fournir des méthodes de délégation pour la gestion de l'authentification, la gestion personnalisée des certificats TLS (à la fois pour les certificats clients et la validation du serveur), la modification du comportement de mise en cache, etc.

Examples

Demande GET simple

```
    // 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 crée une tâche de session et de données

```
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];
```

Configuration de la configuration d'arrière-plan

Pour créer une session d'arrière-plan

```
 // 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]
```

De plus, dans iOS, vous devez configurer le support pour gérer le relancement des applications en arrière-plan. Lorsque l'application de votre `application:handleEventsForBackgroundURLSession:completionHandler:` method (Objective-C) ou `application(_:handleEventsForBackgroundURLSession:completionHandler:)` méthode (Swift) est appelée, cela signifie que votre application a été relancée en arrière-plan pour gérer l'activité sur une session.

Dans cette méthode, vous devez créer une nouvelle session avec l'identifiant fourni et le

configurer avec un délégué pour gérer les événements comme vous le feriez normalement au premier plan. En outre, vous devez stocker le gestionnaire d'achèvement fourni dans un dictionnaire, en utilisant la session comme clé.

Lorsque la `URLSessionDidFinishEventsForBackgroundURLSession:` (Obj-C) / `URLSessionDidFinishEventsForBackgroundURLSession` (Swift) du délégué est appelée pour vous informer qu'il n'y a plus d'événements à gérer, votre application doit rechercher le gestionnaire d'achèvement pour cette session, supprimer la session du dictionnaire et appeler le gestionnaire d'achèvement, indiquant ainsi au système d'exploitation que vous n'avez plus aucun traitement en attente lié à la session. (Si vous faites toujours quelque chose pour une raison quelconque lorsque vous recevez cet appel de délégué, attendez d'avoir terminé.) Dès que vous appelez cette méthode, la session d'arrière-plan est immédiatement invalidée.

Si votre application reçoit ensuite une `application:application:didFinishLaunchingWithOptions:` call (indiquant probablement que l'utilisateur a mis au premier plan votre application alors que vous étiez en train de traiter des événements en arrière-plan), créer une session en arrière avec ce même identifiant l'identifiant n'existe plus.

Si vous êtes curieux des détails, à un niveau élevé, lorsque vous créez une session d'arrière-plan, vous faites deux choses:

- Création d'une session dans un démon externe (nsurlsessiond) pour gérer les téléchargements
- Créer une session dans votre application qui communique avec ce démon externe via NSXPC

Normalement, il est dangereux de créer deux sessions avec le même ID de session dans un seul lancement de l'application, car elles tentent toutes deux de communiquer avec la même session dans le démon d'arrière-plan. C'est pourquoi la documentation officielle dit de ne jamais créer plusieurs sessions avec le même identifiant. Toutefois, si la première session était une session temporaire créée dans le cadre d'un appel `handleEventsForBackgroundURLSession` , l'association entre la session in-app maintenant invalidée et la session du démon d'arrière-plan n'existe plus.

Envoi d'une requête POST avec des arguments à l'aide de NSURLSession dans Objective-C

Il existe deux méthodes courantes pour coder un corps de requête POST: le codage d'URL (application / x-www-form-urlencoded) et les données de formulaire (multipart / form-data). Une grande partie du code est similaire, mais la façon dont vous construisez les données du corps est différente.

Envoi d'une requête à l'aide du codage URL

Que vous ayez un serveur pour votre petite application ou que vous travailliez en équipe avec un ingénieur back-end complet, vous voudrez parler à ce serveur à un moment donné avec votre application iOS.

Dans le code suivant, nous allons composer une chaîne d'arguments que le script du serveur de

destination utilisera pour faire quelque chose qui change en fonction de votre cas. Par exemple, nous pouvons vouloir envoyer la chaîne:

> name = Brendon & password = abcde

Pour le serveur lorsqu'un utilisateur s'inscrit à votre application, le serveur peut stocker ces informations dans une base de données.

Commençons. Vous souhaiterez créer une requête NSURLSession POST avec le code suivant.

```
// 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];
```

Le code ci-dessus vient d'être créé et a déclenché la requête POST sur le serveur. N'oubliez pas que l'URL du script et la chaîne de données POST changent en fonction de votre situation. Si vous lisez ceci, vous saurez comment remplir ces variables.

Vous devrez également ajouter une petite méthode qui effectue le codage de l'URL:

```
- (NSString *)URLEncode:(NSString *)originalString encoding:(NSStringEncoding)encoding
{
    return (__bridge_transfer NSString *)CFURLCreateStringByAddingPercentEscapes(
        kCFAllocatorDefault,
        (__bridge CFStringRef)originalString,
        NULL,
        CFSTR(":/?#[]@!$&'()*+,;="),
        CFStringConvertNSStringEncodingToEncoding(encoding));
}
```

Ainsi, lorsque le serveur aura fini de traiter ces données, il enverra un retour à votre application iOS. Nous devons donc traiter ce retour, mais comment?

Nous utilisons une programmation pilotée par les événements et utilisons les méthodes de délégué de NSURLSession. Cela signifie que lorsque le serveur renvoie une réponse, ces méthodes commenceront à se déclencher. Les 5 méthodes suivantes sont celles qui seront

déclenchées au cours de la requête ENTIRE, à chaque fois:

```
- (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;
```

Vous trouverez ci-dessous les méthodes ci-dessus utilisées en contexte. Chacun de leurs objectifs est assez explicite grâce à Apple, mais j'ai commenté leurs utilisations de toute façon:

```
// 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);
        }
    }
}
```

Alors c'est tout! C'est tout le code dont vous avez besoin pour envoyer, recevoir et analyser une demande pour une API dans iOS 9! D'accord ... c'était du code. Mais si implémenté comme ci-dessus, ce sera sûr! Assurez-vous de toujours traiter les erreurs suggérées ci-dessus.

Envoi d'une demande à l'aide du codage de formulaire

Le codage d'URL est un moyen largement compatible d'encoder des données arbitraires. Cependant, il est relativement inefficace de télécharger des données binaires (telles que des photos), car chaque octet non-ASCII se transforme en un code à trois caractères. Il ne prend pas non plus en charge les pièces jointes, vous devrez donc transmettre les noms de fichiers et les données de fichier en tant que champs distincts.

Supposons que nous voulions télécharger une photo de manière efficace et ressembler à un fichier du côté du serveur. Une façon de procéder consiste à utiliser le codage de formulaire à la place. Pour ce faire, modifiez le code qui crée la NSURLSession comme suit:

```
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];
```

Cela crée et déclenche la requête NSURLSession comme auparavant et, par conséquent, les méthodes déléguées se comportent exactement de la même manière. Assurez-vous que le script auquel l'image est envoyée (situé dans l'URL de la variable `url`) attend une image et peut l'analyser correctement.

Lire NSURLSession en ligne: https://riptutorial.com/fr/ios/topic/2009/nsurlsession

Chapitre 120: NSUserActivity

Introduction

Un objet NSUserActivity peut être utilisé pour coordonner des événements significatifs dans une application avec le système. C'est la base du transfert entre différents appareils sous iOS et macOS. En outre, il peut également être utilisé pour améliorer l'indexation publique et augmenter ou créer des résultats de recherche Spotlight pour une application. À partir d'iOS 10, il peut également être utilisé pour coordonner les interactions entre votre application et Siri à l'aide de SiriKit.

Remarques

Types d'activité

Les types d'activité pris en charge doivent être définis dans le fichier `Info.plist` votre application sous la clé `NSUserActivityTypes` . Les activités sont liées à votre ID d'équipe de développeur, ce qui signifie que la coordination des activités est restreinte entre les applications qui ont le même ID d'équipe (par exemple, "Safari" ne peut accepter une activité de transfert de Chrome).

Devenir / Résigner l'activité en cours

Marquer une activité comme étant en cours à l'aide de `becomeCurrent` rend disponible pour le transfert ou l'indexation Spotlight. Une seule activité peut être en cours à la fois. Vous pouvez marquer une activité comme inactive sans l'invalider en appelant `resignCurrent` .

Si vous `invalidate` une activité, la même instance peut ne plus redevenir active.

Ne marquez pas une activité comme actuelle lorsque vous la fournissez à SiriKit .

Indexation de recherche

Les activités **ne** doivent **pas** être utilisées comme mécanisme d'indexation à usage général dans votre application. Au lieu de cela, ils ne doivent être utilisés qu'en réponse à des actions initiées par l'utilisateur. Pour indexer tout le contenu de votre application, utilisez CoreSpotlight.

Examples

Créer un NSUserActivity

Pour créer un objet `NSUserActivity` , votre application doit déclarer les types d'activités qu'elle

prend en charge dans son fichier `Info.plist` . Les activités prises en charge sont définies par votre application et doivent être uniques. Une activité est définie à l'aide d'un schéma d'attribution de noms de domaine inversé (par exemple, "com.companyName.productName.activityName"). Voici à quoi peut ressembler une entrée dans votre Info.plist:

Clé	Valeur
NSUserActivityTypes	[Array]
- item0	com.companyName.productName.activityName01
- objet 1	com.companyName.productName.activityName02

Une fois que vous avez défini tous les types d'activité pris en charge, vous pouvez commencer à les utiliser et à les utiliser dans le code de votre application.

Pour créer un objet `NSUserActivity` , vous devez effectuer les opérations suivantes

```
// 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];
```

Après cela, l'activité ci-dessus devrait être disponible pour Handoff (bien qu'un travail supplémentaire soit nécessaire pour gérer correctement le "transfert").

Lire NSUserActivity en ligne: https://riptutorial.com/fr/ios/topic/10716/nsuseractivity

Chapitre 121: NSUserDefaults

Syntaxe

- `UserDefaults.standard.set(dic, forKey: "LoginSession") //Save value inside userdefaults`
    - `UserDefaults.standard.object(forKey: "LoginSession") as? [String:AnyObject] ?? [:] //Get value from UserDefaults`

Remarques

NSUserDefault qui est utilisé pour stocker tout type de DataType, et vous pouvez obtenir sa valeur partout dans la classe de l'application. **NSUserDéfault**

Examples

Réglage des valeurs

Pour définir une valeur dans `NSUserDefaults` , vous pouvez utiliser les fonctions suivantes:

Swift <3

```
setBool(_:forKey:)
setFloat(_:forKey:)
setInteger(_:forKey:)
setObject(_:forKey:)
setDouble(_:forKey:)
setURL(_:forKey:)
```

Swift 3

Dans Swift 3, les noms de fonction sont modifiés pour `set` insted de `set` suivi par le type.

```
set(_:forKey:)
```

Objectif 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;
```

Exemple d'utilisation serait:

Swift <3

```
NSUserDefaults.standardUserDefaults.setObject("Netherlands", forKey: "HomeCountry")
```

Swift 3

```
UserDefaults.standard.set("Netherlands", forKey: "HomeCountry")
```

Objectif c

```
[[NSUserDefaults standardUserDefaults] setObject:@"Netherlands" forKey:@"HomeCountry"];
```

Objets personnalisés

Pour enregistrer des objets personnalisés dans le `NSUserDefaults`, vous devez faire en sorte que votre CustomClass confirme le protocole de` NSCoding`. Vous devez implémenter les méthodes suivantes:

Rapide

```
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
}
```

Objectif 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"];
```

```
}
```

Obtenir des valeurs par défaut

Pour obtenir une valeur dans NSUserDefaults, vous pouvez utiliser les fonctions suivantes:

Rapide

```
arrayForKey(_:)
boolForKey(_:)
dataForKey(_:)
dictionaryForKey(_:)
floatForKey(_:)
integerForKey(_:)
objectForKey(_:)
stringArrayForKey(_:)
stringForKey(_:)
doubleForKey(_:)
URLForKey(_:)
```

Objectif 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;
```

Exemple d'utilisation serait:

Rapide

```
let homeCountry = NSUserDefaults.standardUserDefaults().stringForKey("HomeCountry")
```

Objectif c

```
NSString *homeCountry = [[NSUserDefaults standardUserDefaults] stringForKey:@"HomeCountry"];
```

Sauvegarder les valeurs

`NSUserDefaults` est écrit sur le disque régulièrement par le système, mais il `NSUserDefaults` que vous souhaitiez que vos modifications soient enregistrées immédiatement, par exemple lorsque

l'application passe en arrière-plan. Cela se fait en appelant `synchronize` .

Rapide

```
NSUserDefaults.standardUserDefaults().synchronize()
```

Objectif c

```
[[NSUserDefaults standardUserDefaults] synchronize];
```

Utiliser les gestionnaires pour enregistrer et lire des données

Bien que vous puissiez utiliser les méthodes `NSUserDefaults` n'importe où, il est parfois préférable de définir un gestionnaire qui enregistre et lit à partir de `NSUserDefaults` , puis utilise ce gestionnaire pour lire ou écrire vos données.

Supposons que nous voulions enregistrer le score d'un utilisateur dans `NSUserDefaults` . Nous pouvons créer une classe comme celle ci-dessous qui a deux méthodes: `setHighScore` et `highScore` . Partout où vous voulez accéder aux meilleurs scores, créez une instance de cette classe.

Rapide

```
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;
            }
        }
    }
}
```

Objectif 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
```

Les avantages sont que:

1. L'implémentation de votre processus de lecture et d'écriture est seulement à un endroit et vous pouvez le changer (par exemple passer de `NSUserDefaults` à Core Data) quand vous le voulez et ne pas vous soucier de changer tous les endroits avec lesquels vous travaillez.
2. Appelez simplement une seule méthode lorsque vous souhaitez accéder au score ou l'écrire.
3. Déboguez-le simplement quand vous voyez un bug ou quelque chose comme ça.

> **Remarque**
>
> Si vous vous inquiétez de la synchronisation, il est préférable d'utiliser une classe singleton qui gère la synchronisation.

Effacer NSUserDefaults

Rapide

```
let bundleIdentifier = NSBundle.mainBundle().bundleIdentifier()

NSUserDefaults.standardUserDefaults().removePersistentDomainForName(bundleIdentifier)
```

Objectif c

```
NSString *bundleIdentifier = [[NSBundle mainBundle] bundleIdentifier];

[[NSUserDefaults standardUserDefaults] removePersistentDomainForName: bundleIdentifier];
```

UserDefaults utilise dans Swift 3

Chaque application nécessaire pour stocker les détails liés à la session utilisateur ou à l'utilisateur dans l'application dans UserDefaults. Nous avons donc intégré toute une logique à une classe pour mieux gérer UserDefaults.

Swift 3

```
import Foundation

public struct Session {

    fileprivate static let defaults = UserDefaults.standard

    enum userValues: String {
        case auth_token
        case email
        case fname
        case mobile
        case title
        case userId
        case userType
        case OTP
        case isApproved
    }


    //MARK: - Getting here User Details
    static func getUserSessionDetails()->[String:AnyObject]? {
        let dictionary = defaults.object(forKey: "LoginSession") as? [String:AnyObject]
        return  dictionary
    }

    //MARK: - Saving Device Token
    static func saveDeviceToken(_ token:String){
        guard (gettingDeviceToken() ?? "").isEmpty else {
            return
        }
        defaults.removeObject(forKey: "deviceToken")
        defaults.set(token, forKey: "deviceToken")
        defaults.synchronize()
    }
```

```
    //MARK: - Getting Token here
    static func gettingDeviceToken()->String?{
        let token = defaults.object(forKey: "deviceToken") as? String
        if token == nil{
            return ""
        }else{ return token}
    }

    //MARK: - Setting here User Details
    static func setUserSessionDetails(_ dic :[String : AnyObject]){
        defaults.removeObject(forKey: "LoginSession")
        defaults.set(dic, forKey: "LoginSession")
        defaults.synchronize()
    }

    //MARK:- Removing here all Default Values
    static func userSessionLogout(){
        //Set Activity
        defaults.removeObject(forKey: "LoginSession")
        defaults.synchronize()
    }

    //MARK: - Get value from session here
    static func getUserValues(value: userValues) -> String? {
        let dic = getUserSessionDetails() ?? [:]
        guard let value = dic[value.rawValue] else{
            return ""
        }
        return value as? String
    }

}
```

Utilisation de la classe UserDefaults

```
//Saving user Details
Session.setUserSessionDetails(json ?? [:])

//Retriving user Details
let userId = Session.getUserValues(value: .userId) ?? ""
```

Lire NSUserDefaults en ligne: https://riptutorial.com/fr/ios/topic/3150/nsuserdefaults

Chapitre 122: Objective-C Objets associés

Introduction

Introduits pour la première fois dans iOS 3.1 dans le cadre du runtime Objective-C, les objets associés permettent d'ajouter des variables d'instance à un objet de classe existant (avec sous-classement).

Cela signifie que vous pourrez attacher n'importe quel objet à tout autre objet sans sous-classer.

Syntaxe

- void objc_setAssociatedObject (objet id, void * key, id value, objc_AssociationPolicy policy)
- id objc_getAssociatedObject (objet id, void * key)
- void objc_removeAssociatedObjects (objet id)

Paramètres

Param	Détails
objet	L'objet existant que vous souhaitez modifier
clé	Cela peut être fondamentalement n'importe quel pointeur qui a une adresse mémoire constante, mais une bonne pratique est d'utiliser ici une propriété calculée (getter)
valeur	L'objet que vous souhaitez ajouter
politique	La politique de mémoire pour cette nouvelle `value` c.-à-d. Si elle doit être conservée / attribuée, copiée, etc., comme toute autre propriété que vous déclarez

Remarques

Plus de détails ici:

NSHipster

@kostiakoval

kingscocoa

Examples

Exemple d'objet associé de base

Supposons que nous ayons besoin d'ajouter un objet NSString à `SomeClass` (nous ne pouvons pas sous-classe).

Dans cet exemple, nous créons non seulement un objet associé, mais également l'enveloppons dans une propriété calculée dans une catégorie pour une plus grande netteté.

```
#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));
}
```

Maintenant, il serait aussi facile que cela d'utiliser la propriété

```
SomeClass *instance = [SomeClass alloc] init];
instance.associated = @"this property is an associated object under the hood";
```

Lire Objective-C Objets associés en ligne: https://riptutorial.com/fr/ios/topic/9102/objective-c-objets-associes

Chapitre 123: OpenGL

Introduction

OpenGL ES est une bibliothèque graphique utilisée par iOS pour effectuer le rendu 3D.

Examples

Exemple de projet

Un exemple de projet (repo Git) qui peut être utilisé comme point de départ pour effectuer un rendu 3D. Le code pour configurer OpenGL et les shaders est assez long et fastidieux, donc il ne conviendra pas sous cet exemple. Des parties ultérieures de celui-ci peuvent être présentées dans des exemples séparés détaillant ce qui se passe exactement avec chaque morceau de code, mais pour l'instant, voici le projet Xcode.

Lire OpenGL en ligne: https://riptutorial.com/fr/ios/topic/9324/opengl

Chapitre 124: Opérations étendues

Examples

Obtenez le meilleur UIViewController

Une approche courante pour obtenir le meilleur `UIViewController` consiste à obtenir le RootViewController de votre `UIWindow` active. J'ai écrit une extension pour cela:

```
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!
}
```

Événements du système d'interception

En utilisant le NotificationCenter d'iOS, qui peut être très puissant, vous pouvez intercepter certains événements à l'échelle de l'application:

```
NotificationCenter.default.addObserver(
        self,
        selector: #selector(ViewController.do(_:)),
        name: NSNotification.Name.UIApplicationDidBecomeActive,
        object: nil)
```

Vous pouvez vous inscrire à beaucoup d'autres événements, regardez simplement https://developer.apple.com/reference/foundation/nsnotification.name .

Lire Opérations étendues en ligne: https://riptutorial.com/fr/ios/topic/7188/operations-etendues

Chapitre 125: plist iOS

Introduction

Plist est utilisé pour stocker des données dans l'application iOS. Plist enregistrer les données sous forme de tableau et de dictionnaires. En résumé, nous pouvons enregistrer des données en tant que: 1. Données statiques à utiliser dans l'application. 2. Les données qui proviendront du serveur.

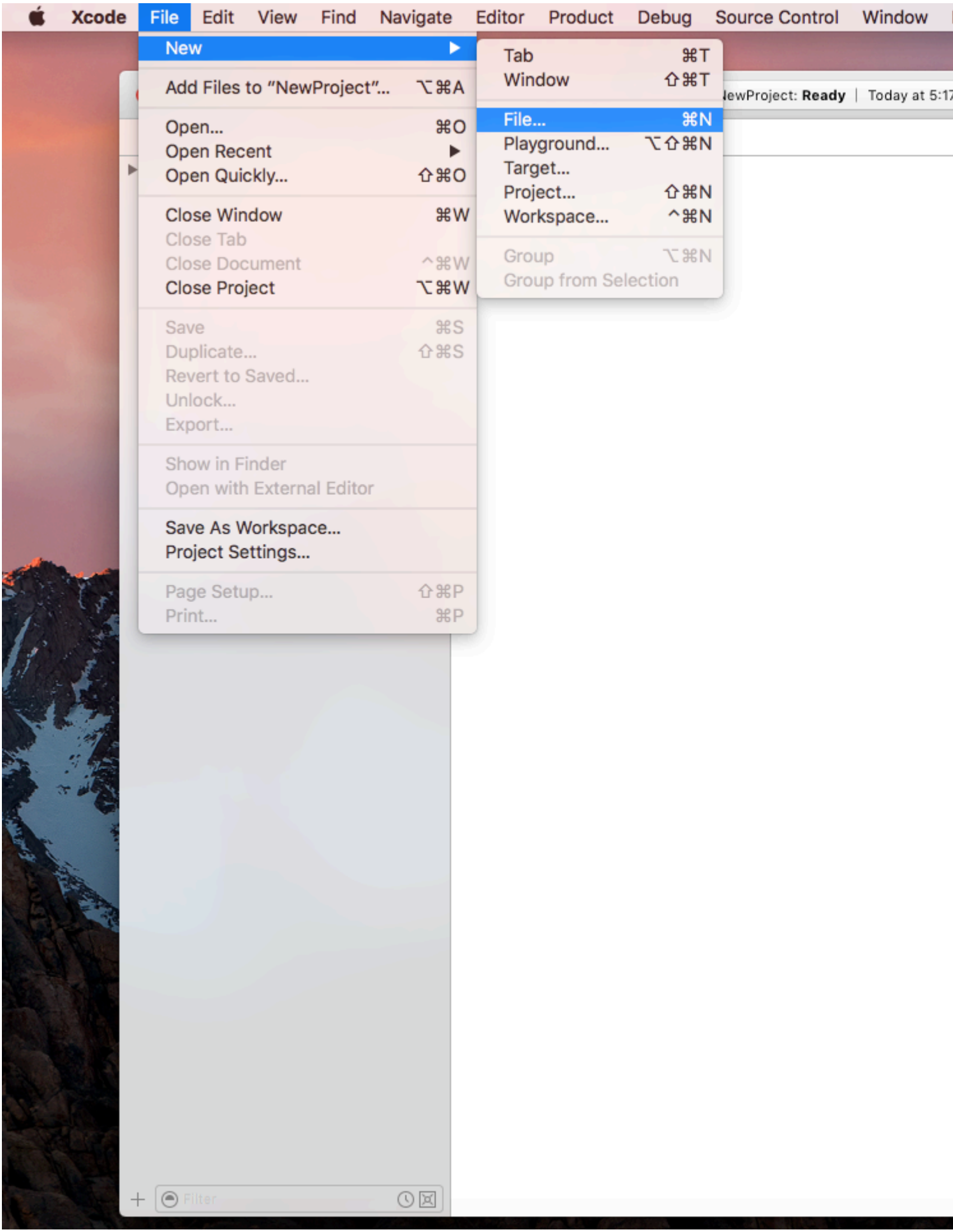
Examples

Exemple:

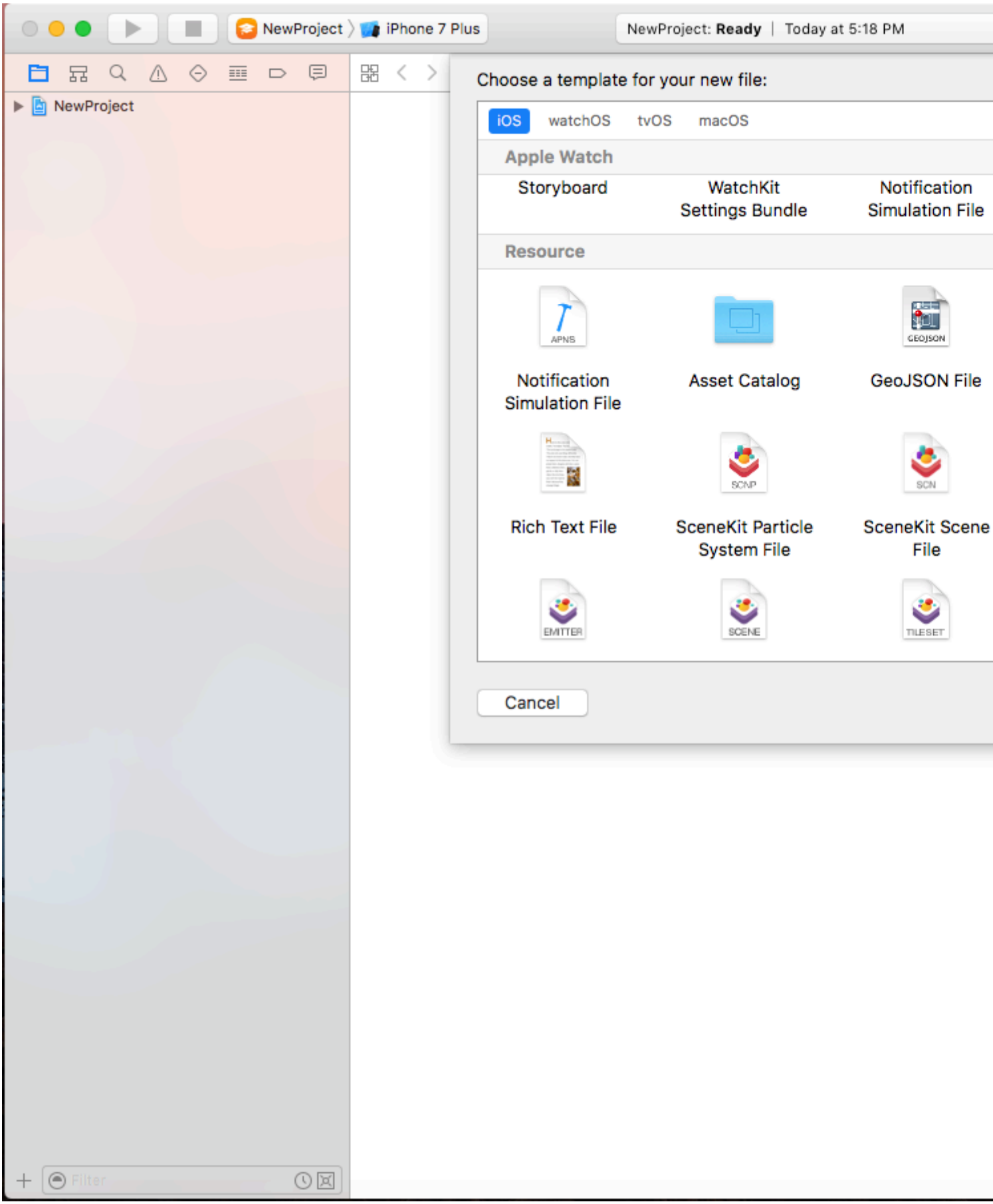
1. Données statiques à utiliser dans l'application.

Pour enregistrer des données statiques dans plist, procédez comme suit:

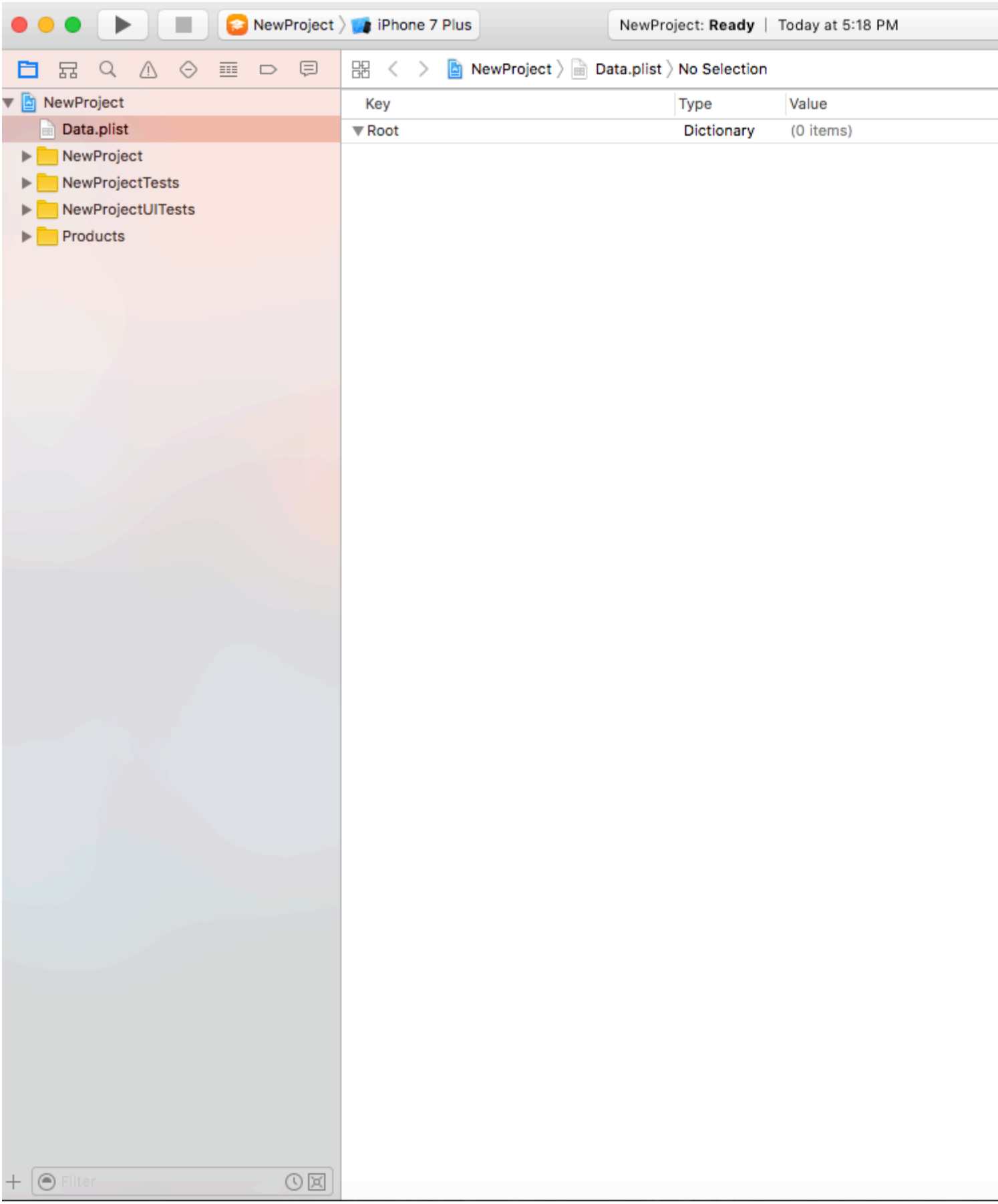
a) Ajouter un nouveau fichier


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

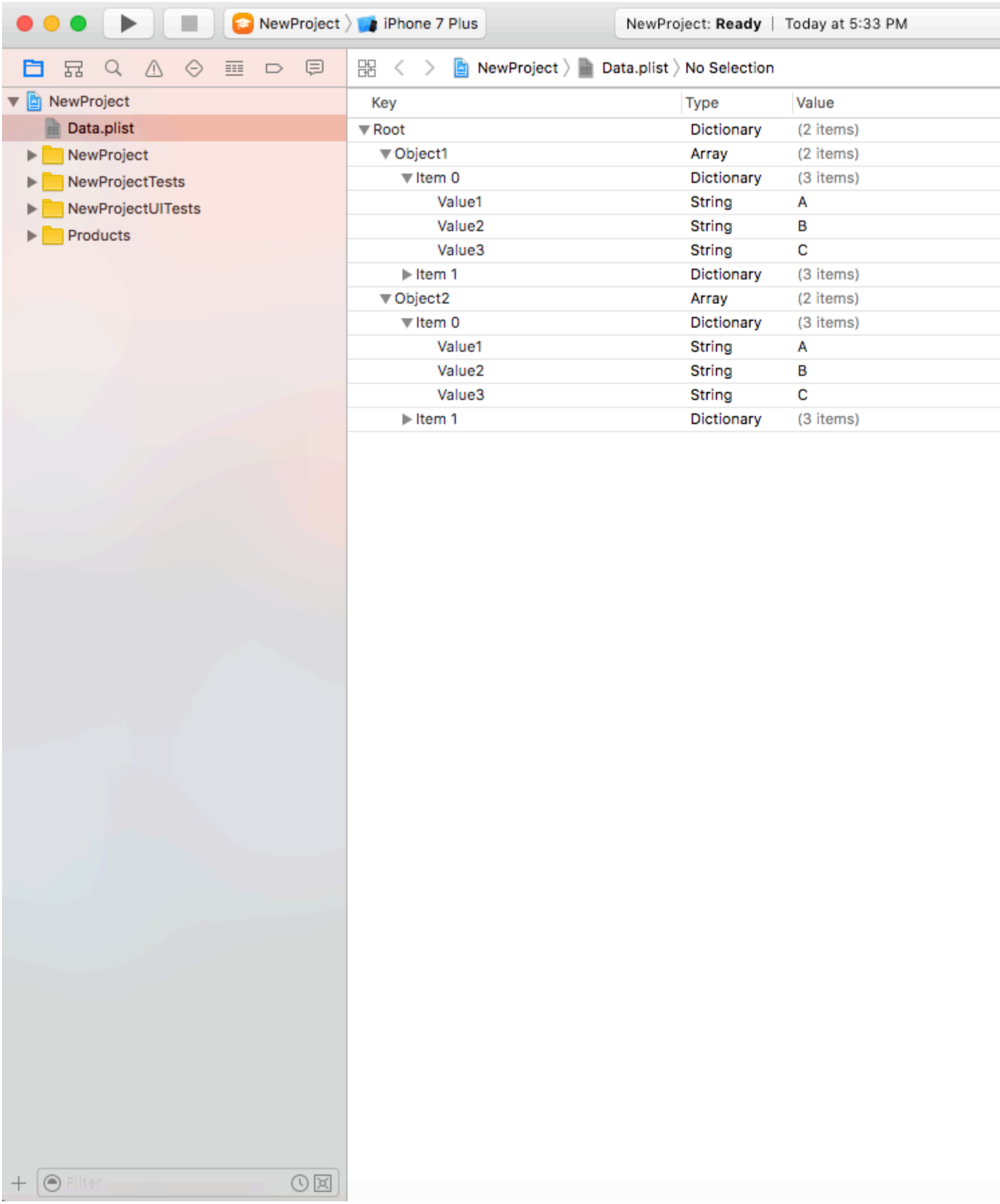
b) Cliquez sur Liste de propriétés dans Ressources



c) Nommez la propriété et un fichier sera créé comme (data.plist ici)



d) Vous pouvez créer une liste de tableaux et de dictionnaires comme suit:



// Lit plist à partir de bundle et récupère le dictionnaire racine

```
NSDictionary *dictRoot = [NSDictionary dictionaryWithContentsOfFile:[[NSBundle mainBundle]
pathForResource:@"Data" ofType:@"plist"]];
```

// Votre dictionnaire contient un tableau de dictionnaire // Retirez maintenant un tableau.

```
NSArray *arrayList = [NSArray arrayWithArray:[dictRoot objectForKey:@"Object1"]];

for(int i=0; i< [arrayList count]; i++)
{
    NSMutableDictionary *details=[arrayList objectAtIndex:i];
}
```

Enregistrer et éditer / supprimer des données de Plist

Vous avez déjà créé un plist. Ce plist restera le même dans app. Si vous souhaitez modifier les données dans cette liste, ajouter de nouvelles données dans plist ou supprimer des données de plist, vous ne pouvez pas apporter de modifications à ce fichier.

À cette fin, vous devrez stocker votre liste dans Répertoire de documents. Vous pouvez modifier votre pliste enregistré dans le répertoire du document.

Enregistrez plist dans le répertoire du document en tant que:

```
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
{

}
```

Récupérer les données de Plist en tant que:

```
    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"];
```

Vous pouvez modifier supprimer, ajouter de nouvelles données selon vos besoins et enregistrer le plist à nouveau dans le répertoire de documents.

Lire plist iOS en ligne: https://riptutorial.com/fr/ios/topic/8141/plist-ios

Chapitre 126: Polices personnalisées

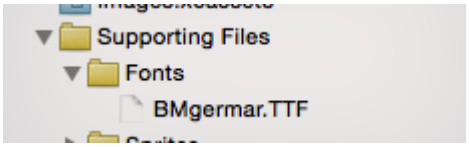
Examples

Incorporation de polices personnalisées

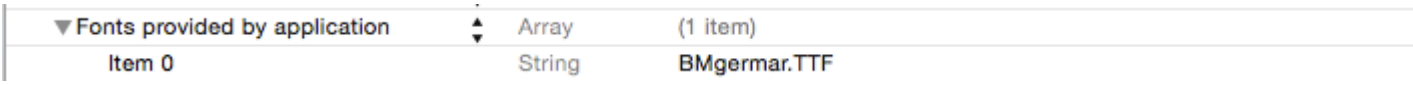
> **Prise en charge des polices personnalisées**
> Les applications qui souhaitent utiliser des polices personnalisées peuvent désormais inclure ces polices dans leur regroupement d'applications et les enregistrer avec le système en incluant la clé UIAppFonts dans leur fichier Info.plist. La valeur de cette clé est un tableau de chaînes identifiant les fichiers de polices dans le bundle de l'application. Lorsque le système détecte la clé, il charge les polices spécifiées et les met à la disposition de l'application.

Une fois les polices définies dans `Info.plist` , vous pouvez utiliser vos polices personnalisées comme toute autre police dans IB ou par programme.

1. Faites glisser et déposez votre police dans le dossier Xcode Supporting Files. N'oubliez pas de marquer votre application dans la section "Ajouter aux cibles". A partir de ce moment, vous pouvez utiliser cette police dans IB et la choisir dans la palette de polices.



2. Pour rendre cette police disponible sur le périphérique, ouvrez `Info.plist` et ajoutez des `Fonts provided by application key` (UIAppFonts). Ajoutez le nom de la police en tant que valeur à la clé Item 0. Remarque: le nom de la police peut varier de votre nom de fichier de police.



3. Obtenez le nom de police ajouté personnalisé en utilisant l'extrait ci-dessous

[*Swift 3*]

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

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

[*Objectif - C*]

```
for (NSString *familyName in [UIFont familyNames]){
```

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

Polices personnalisées avec storyboard

Les polices personnalisées pour les composants d'interface utilisateur du storyboard peuvent être facilement obtenues avec les attributs d'exécution définis par l' utilisateur dans le storyboard et les catégories .

Les avantages sont comme,

- Pas besoin de définir des sorties pour l'élément d'interface utilisateur
- Pas besoin de définir la police pour les éléments par programmation.

Étapes à suivre

1. **Fichier de polices:** ajoutez le fichier de polices (.ttf) au groupe d'applications et ajoutez l'entrée pour la police dans Info.plist sous ***Police fournie par l'application,*** comme dans cette documentation de polices personnalisées.

2. **Définir des catégories:** Ajoutez un fichier comme ***UIKit + IBExtensions*** et ajoutez les catégories pour les éléments d'interface utilisateur tels que UILabel, UIButton, etc. pour lesquels vous souhaitez définir une police personnalisée. Toutes les catégories auront une propriété personnalisée, par exemple **fontName** . Cela sera utilisé depuis le storyboard plus tard pour définir la police personnalisée (comme à l'étape 4).

UIKit + IBExtensions.h

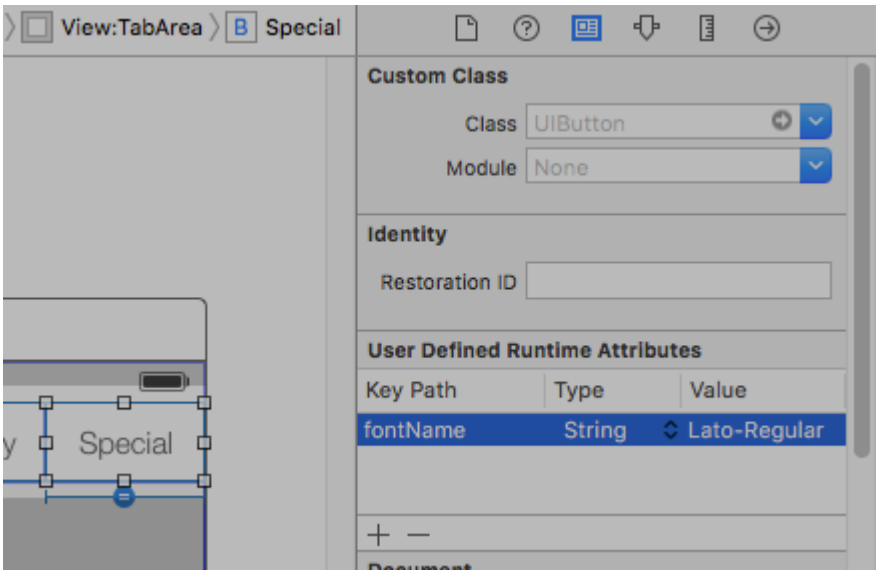
```
#import <UIKit/UIKit.h>

//Category extension for UILabel
@interface UILabel (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end

// Category extension for UITextField
@interface UITextField (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end

// Category extension for UIButton
@interface UIButton (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end
```

3. **Getters et Setters:** Définissez les getters et les setters pour la propriété fontName vers chaque catégorie ajoutée.

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. **Définition de la police dans le storyboard:** Ajoutez une entrée dans Attributs d'exécution définis par l'utilisateur avec ***fontName en*** tant que KeyPath et le nom de votre ***police personnalisée en*** tant que valeur avec le type String comme indiqué.



Cela va définir votre police personnalisée lors de l'exécution de l'application.

Remarques:

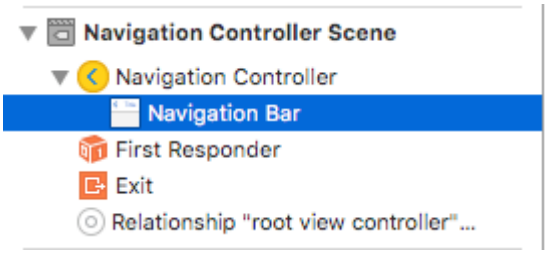
- Lato-Regular est la police personnalisée que j'ai utilisée.
- Le même nom dans le fichier ***.ttf*** ajouté dans le bundle doit être utilisé sans extension dans le storyboard.
- La taille de la police sera la même que celle définie dans l'inspecteur d'attribut de l'élément de l'interface utilisateur.

Application de polices personnalisées aux contrôles dans un Storyboard

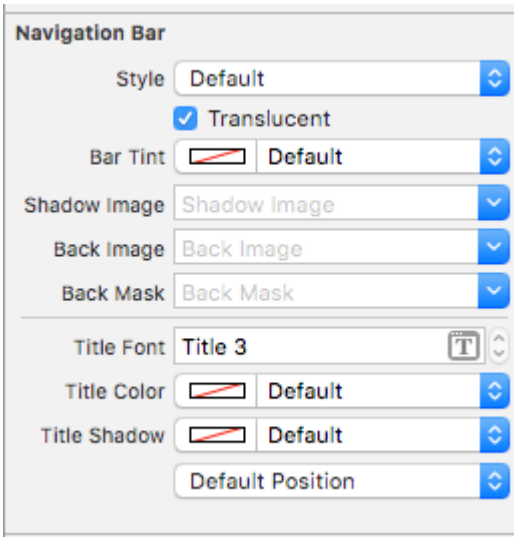
L'exemple suivant montre comment appliquer des polices personnalisées à une barre de navigation et inclut des correctifs pour certains comportements inhabituels trouvés dans Xcode. Il est également possible d'appliquer les polices personnalisées à **tout autre contrôle UIC** tel que **UILabels** , **UIButtons** , etc. en utilisant l'inspecteur d'attributs après l'ajout de la police personnalisée au projet. Veuillez noter les liens externes vers des échantillons de travail et des vidéos en bas.

1. Sélectionnez votre barre de navigation dans votre contrôleur de navigation

---



2. Modifier la police de titre dans l'inspecteur d'attributs

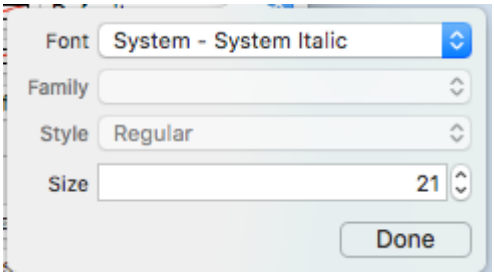


(Vous devrez probablement basculer la barre de teinte pour la barre de navigation avant que Xcode ne décroche la nouvelle police)

Notes (mises en garde)

Vérifié que cela fonctionne sur Xcode 7.1.1+. (**Voir les exemples ci-dessous**)

1. Vous devez basculer la teinte de la barre de navigation avant que la police ne prenne effet (cela ressemble à un bogue dans Xcode; vous pouvez le rétablir par défaut et la police va coller)
2. Si vous choisissez une police système, assurez-vous que la taille n'est pas 0.0 (sinon la nouvelle police sera ignorée)



3. Cela semble fonctionner sans problème quand une seule barre de navigation est dans la hiérarchie des vues. Il semble que les NavBars secondaires dans la même pile soient ignorées. (Notez que si vous affichez la barre de navigation du contrôleur de navigation maître, tous les autres paramètres de la barre de navigation personnalisée sont ignorés).

Gotchas (deux)

Certains d'entre eux sont répétés, ce qui signifie qu'ils méritent d'être notés.

1. Parfois, le storyboard xml est corrompu. Cela nécessite que vous examiniez la structure dans Storyboard en mode Code source (cliquez avec le bouton droit sur le fichier storyboard> Ouvrir en tant que ...)
2. Dans certains cas, la balise navigationItem associée à l'attribut runtime défini par l'utilisateur

a été définie comme enfant xml de la balise view au lieu de la balise du contrôleur de vue. Si oui, retirez-le entre les balises pour un fonctionnement correct.
3. Activez la teinte NavBar pour vous assurer que la police personnalisée est utilisée.
4. Vérifiez le paramètre de taille de la police à moins d'utiliser un style de police dynamique
5. La hiérarchie des vues remplace les paramètres. Il semble qu'une police par pile soit possible.

Résultat

Des échantillons

- Vidéo montrant plusieurs polices dans un projet avancé
- Téléchargement de source simple
- Advanced Project Download ~ affiche plusieurs polices NavBar et contournement personnalisé des polices
- Vidéo montrant plusieurs polices et polices personnalisées

Gestion des polices personnalisées

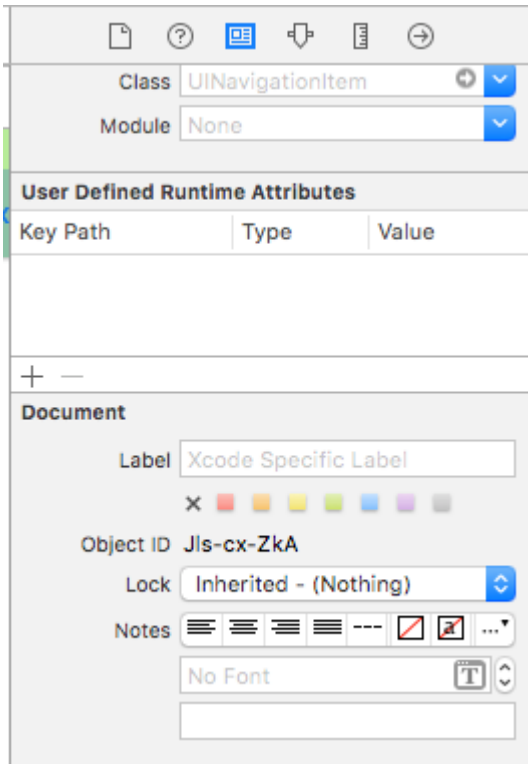
Remarque ~ Vous pouvez trouver une liste de contrôle agréable à partir du site Web Code With Chris et vous pouvez voir le projet de téléchargement exemple.

Si vous avez votre propre police et que vous souhaitez l'utiliser dans votre storyboard, il existe un ensemble correct de réponses à la question SO suivante. Une réponse identifie ces étapes.

1. Obtenez votre fichier de police personnalisé (.ttf, .ttc)
2. Importer les fichiers de police dans votre projet Xcode
3. Dans app-info.plist, ajoutez une clé nommée Fonts fournie par application.Il s'agit d'un type de tableau, ajoutez tous les noms de fichiers de polices au tableau, notez: y compris l'extension du fichier.
4. Dans le storyboard, dans la barre de navigation, accédez à l'inspecteur d'attributs, cliquez sur le bouton icône droit de la zone de sélection de polices. Dans le panneau contextuel, choisissez Police à personnaliser et choisissez la famille de votre nom de police incorporé.

Solution de contournement de police personnalisée

Donc, Xcode semble naturellement pouvoir gérer les polices personnalisées sur UINavigationItem, mais cette fonctionnalité ne se met pas à jour correctement (la police sélectionnée est ignorée).



Pour contourner ce problème:

L'une des méthodes consiste à utiliser le storyboard et à ajouter une ligne de code: ajoutez d'abord un UIView (UIButton, UILabel ou une autre sous-classe UIView) au View Controller (pas l'élément de navigation ... Xcode ne permet actuellement pas de le faire) cette). Après avoir ajouté le contrôle, vous pouvez modifier la police dans le storyboard et ajouter une référence en tant que prise à votre View Controller. Attribuez simplement cette vue à UINavigationItem.titleView. Vous pouvez également définir le nom du texte dans le code si nécessaire. Bug signalé (23600285).

```
@IBOutlet var customFontTitleView: UIButton!

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

Remarque - Cet exemple est dérivé d'une réponse que j'ai publiée sur SO (ici).

Lire Polices personnalisées en ligne: https://riptutorial.com/fr/ios/topic/1504/polices-personnalisees

Chapitre 127: Porte-clés

Syntaxe

- kSecClassGenericPassword // Une clé de valeur représentant un mot de passe non Internet
- kSecClassInternetPassword // Une clé de valeur représentant un mot de passe Internet
- kSecClassCertificate // Une clé de valeur représentant un certificat
- kSecClassCertificate // Une clé de valeur représentant une clé
- kSecClassIdentity // Une clé de valeur représentant une identité, qui est un certificat plus une clé

Remarques

iOS stocke des informations privées telles que des mots de passe, des clés de chiffrement, des certificats et des identités dans une zone de stockage sécurisée appelée porte-clé. Cette zone de stockage est entièrement gérée par un co-processeur appelé Secure Enclave, qui est intégré au processeur d'application. Étant donné que le trousseau est mis en sandbox sur iOS, les éléments de trousseau ne peuvent être récupérés que par l'application qui les a placés en premier lieu.

Dans certains cas, vous devez activer le partage de trousseau dans les fonctionnalités Xcode afin d'éviter les erreurs.

Pour interagir avec le trousseau, nous utilisons le framework ac appelé Keychain Services. Pour plus d'informations, consultez le Guide de programmation des services de trousseau d'Apple .

Étant donné que les services de trousseau sont inférieurs au niveau `Foundation` , ils sont limités à l'utilisation des types `CoreFoundation` . Par conséquent, la plupart des objets sont représentés en interne en tant que `CFDictionary` qui `CFString` conserver `CFString` s en tant que leurs clés et divers types de `CoreFoundation` tant que valeurs.

Bien que les services de trousseau soient inclus dans le cadre de la `Security` , l'importation de `Foundation` est généralement une bonne option, car elle inclut des fonctions d'assistance dans le backend.

De plus, si vous ne souhaitez pas gérer directement les services de trousseau, Apple fournit le projet exemple Generic Keychain Swift qui fournit les types Swift qui utilisent les services de trousseau dans les coulisses.

Examples

Ajouter un mot de passe au trousseau

Chaque élément de trousseau est le plus souvent représenté comme un `CFDictionary` . Vous pouvez, cependant, simplement utiliser `NSDictionary` dans Objective-C et profiter du pontage ou, dans Swift, vous pouvez utiliser `Dictionary` et le `CFDictionary` explicitement dans `CFDictionary` .

Vous pouvez créer un mot de passe avec le dictionnaire suivant:

Rapide

```
var dict = [String : AnyObject]()
```

Tout d'abord, vous avez besoin d'une paire clé / valeur permettant au porte-clés de savoir qu'il s'agit d'un mot de passe. Notez que parce que notre clé dict est une `String` nous devons `CFString` tout `CFString` en `String` explicitement dans Swift 3. CFString ne peut pas être utilisé comme clé dans un dictionnaire Swift car il n'est pas lavable.

Rapide

```
dict[kSecClass as String] = kSecClassGenericPassword
```

Ensuite, notre mot de passe peut avoir une série d'attributs pour le décrire et nous aider à le retrouver plus tard. Voici une liste d'attributs pour les mots de passe génériques .

Rapide

```
// 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
```

Enfin, nous avons besoin de nos données privées réelles. Veillez à ne pas garder cela en mémoire trop longtemps. Ce doit être `CFData` .

Rapide

```
dict[kSecValueData as String] = "my_password!!".data(using: .utf8) as! CFData
```

Enfin, la fonction d'ajout de Keychain Services veut savoir comment renvoyer l'élément de trousseau nouvellement construit. Comme vous ne devriez pas conserver les données très longtemps en mémoire, voici comment vous ne pouvez retourner que les attributs:

Rapide

```
dict[kSecReturnAttributes as String] = kCFBooleanTrue
```

Maintenant, nous avons construit notre article. Ajoutons-le:

Rapide

```
var result: AnyObject?
let status = withUnsafeMutablePointer(to: &result) {
    SecItemAdd(dict as CFDictionary, UnsafeMutablePointer($0))
}
let newAttributes = result as! Dictionary<String, AnyObject>
```

Cela place les nouveaux attributs dans le `result` . `SecItemAdd` prend en compte le dictionnaire que nous avons `SecItemAdd` , ainsi qu'un pointeur vers l'endroit où nous souhaiterions obtenir notre résultat. La fonction retourne alors un `OSStatus` indiquant le succès ou un code d'erreur. Les codes de résultat sont décrits ici .

Trouver un mot de passe dans le trousseau

Pour construire une requête, nous devons la représenter en tant que `CFDictionary` . Vous pouvez également utiliser `NSDictionary` dans Objective-C ou `Dictionary` dans Swift et le `CFDictionary` en `CFDictionary` .

Nous avons besoin d'une clé de classe:

Rapide

```
var dict = [String : AnyObject]()
dict[kSecClass as String] = kSecClassGenericPassword
```

Ensuite, nous pouvons spécifier des attributs pour affiner notre recherche:

Rapide

```
// 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
```

Nous pouvons également spécifier des touches de modification de recherche spéciales décrites ici .

Enfin, nous devons dire comment nous aimerions que nos données soient retournées. Ci-

dessous, nous vous demanderons de ne retourner que le mot de passe privé en tant qu'objet `CFData` :

Rapide

```
dict[kSecReturnData as String] = kCFBooleanTrue
```

Maintenant, cherchons:

Rapide

```
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)!
```

`SecItemCopyMatching` prend ici un dictionnaire de requêtes et un pointeur vers l'emplacement `SecItemCopyMatching` . Il retourne un `OSStatus` avec un code de résultat. Voici les possibilités.

Mise à jour d'un mot de passe dans le trousseau

Comme d'habitude, nous avons d'abord besoin d'un `CFDictionary` pour représenter l'élément que nous voulons mettre à jour. Cela doit contenir toutes les anciennes valeurs de l'élément, y compris les anciennes données privées. Ensuite, il faut un `CFDictionary` de tous les attributs ou des données que vous souhaitez modifier.

Donc, tout d'abord, construisons une clé de classe et une liste d'attributs. Ces attributs peuvent restreindre notre recherche, mais vous devez inclure tous les attributs et les anciennes valeurs si vous les modifiez.

Rapide

```
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
```

Maintenant, nous devons ajouter les anciennes données:

Rapide

```
dict[kSecValueData as String] = "my_password!!".data(using: .utf8) as! CFData
```

Maintenant, créons les mêmes attributs, mais un mot de passe différent:

Rapide

```
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
```

Maintenant, nous passons juste à Keychain Services:

Rapide

```
let status = SecItemUpdate(dict as CFDictionary, newDict as CFDictionary)
```

`SecItemUpdate` renvoie un code d'état. Les résultats sont décrits ici .

Supprimer un mot de passe du trousseau

Nous n'avons besoin que d'une chose pour supprimer un élément du trousseau: un `CFDictionary` avec des attributs décrivant les éléments à supprimer. Tous les éléments correspondant au dictionnaire de requête seront supprimés de manière permanente. Si vous ne souhaitez supprimer qu'un seul élément, veillez à être spécifique à votre requête. Comme toujours, nous pouvons utiliser un `NSDictionary` en Objective-C ou, dans Swift, nous pouvons utiliser un `Dictionary` puis le `CFDictionary` en `CFDictionary` .

Un dictionnaire de requêtes, dans ce contexte, comprend exclusivement une clé de classe pour décrire l'objet et les attributs pour décrire des informations sur l'élément. L'inclusion de restrictions de recherche telles que `kSecMatchCaseInsensitive` n'est pas autorisée.

Rapide

```
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
```

Et maintenant, nous pouvons simplement le supprimer:

Rapide

```
let status = SecItemDelete(dict as CFDictionary)
```

`SecItemDelete` renvoie un `OSStatus` . Les codes de résultat sont décrits ici .

Keychain Ajouter, mettre à jour, supprimer et rechercher des opérations en utilisant un fichier.

Porte-clés.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
```

Porte-clés.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
```

Keychain Access Control (TouchID avec retour de mot de passe)

Le trousseau permet de sauvegarder des éléments avec un attribut spécial SecAccessControl qui permettra d'obtenir un élément du trousseau uniquement après que l'utilisateur aura été authentifié avec Touch ID (ou un mot de passe si un tel retour est autorisé). App est seulement averti si l'authentification a réussi ou non, l'interface utilisateur entière est gérée par iOS.

Tout d'abord, l'objet SecAccessControl doit être créé:

Rapide

```
let error: Unmanaged<CFError>?

guard let accessControl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly, .userPresence, &error) else {
    fatalError("Something went wrong")
}
```

Ensuite, ajoutez-la au dictionnaire avec la clé kSecAttrAccessControl (qui est mutuellement exclusive avec la clé kSecAttrAccessible que vous avez utilisée dans d'autres exemples):

Rapide

```
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
```

Et sauvegardez-le comme vous l'avez fait auparavant:

Rapide

```
let lastResultCode = SecItemAdd(query as CFDictionary, nil)
```

Pour accéder aux données stockées, interrogez simplement Keychain pour obtenir une clé. Keychain Services présentera la boîte de dialogue d'authentification à l'utilisateur et retournera des données ou des données nulles selon que l'empreinte digitale appropriée a été fournie ou que le code d'accès a été apparié.

En option, une chaîne d'invite peut être spécifiée:

Rapide

```
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))
}
```

Faites attention que le `status` sera `err` si l'utilisateur a refusé, annulé ou échoué l'autorisation.

Rapide

```
if status == noErr {
    let password = String(data: queryResult as! Data, encoding: .utf8)!
    print("Password: \(password)")
} else {
    print("Authorization not passed")
}
```

Lire Porte-clés en ligne: https://riptutorial.com/fr/ios/topic/6839/porte-cles

Chapitre 128: Processus de soumission d'applications

Introduction

Ce tutoriel couvre toutes les étapes nécessaires pour télécharger une application iOS sur l'App Store.

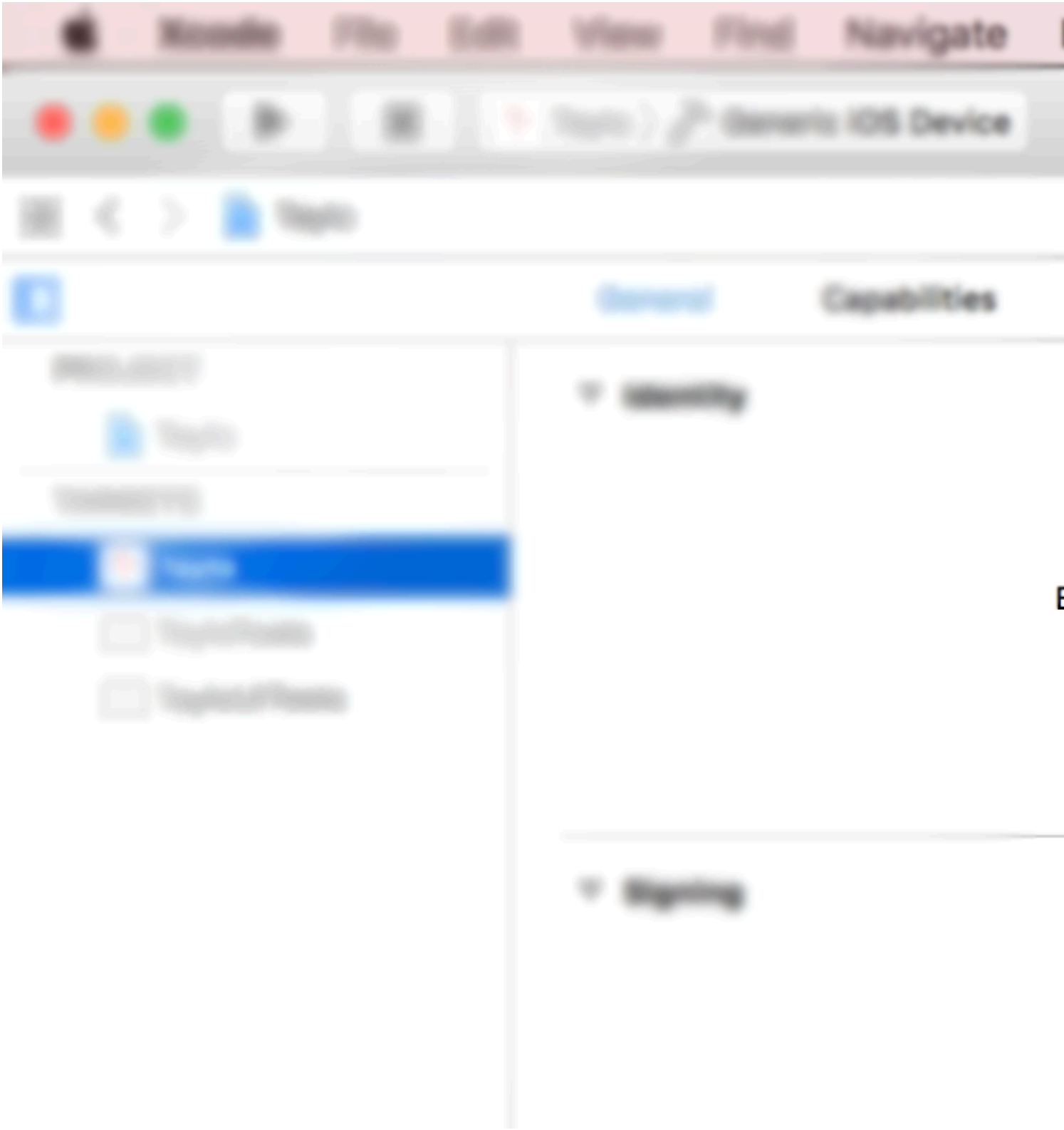
Examples

Configurer les profils d'approvisionnement

Dans les versions précédentes, la configuration des profils d'approvisionnement était effectuée manuellement. Vous générez un profil de distribution, vous le téléchargez et distribuez ensuite votre application. Cela devait être fait pour chaque machine de développement qui prenait beaucoup de temps. Cependant, dans la plupart des situations, Xcode 8 fera le plus gros de ce travail pour vous. Assurez-vous de vous connecter avec le compte qui sera utilisé pour distribuer l'application, puis sélectionnez simplement "Gérer automatiquement la signature du code" dans Cibles -> Général.

Archiver le code

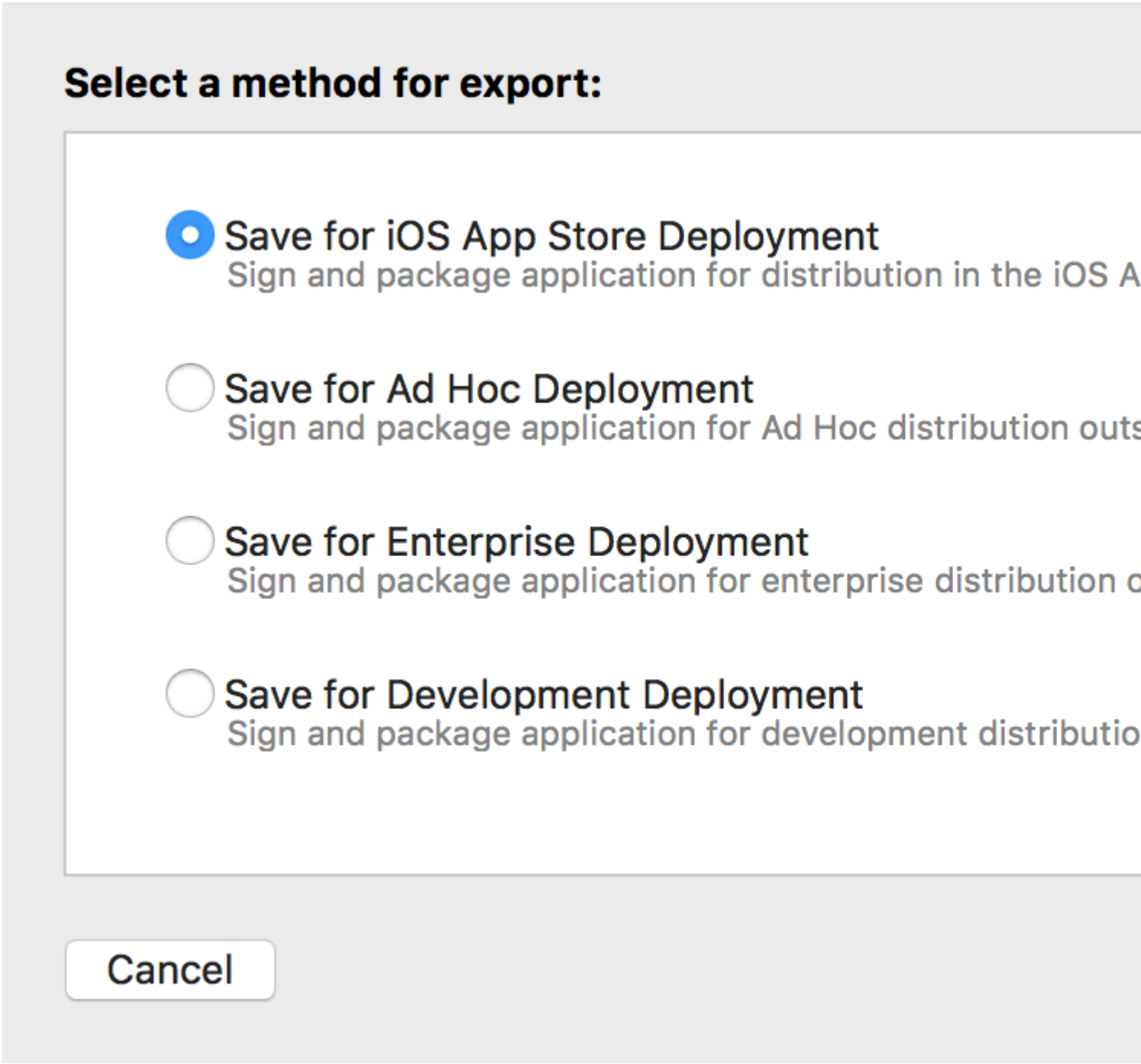
Une fois les profils de provisionnement définis, la prochaine étape du processus de soumission de l'application consiste à archiver votre code. Dans le menu déroulant des appareils et des simulateurs, sélectionnez l'option "Appareil iOS générique". Ensuite, sous le menu "Produit", sélectionnez l'option "Archiver".



Dans le cas où la soumission est une mise à jour de l'application existante sur le magasin, assurez-vous que le numéro de version est supérieur au nombre actuel et que le numéro de version est différent. Par exemple, l'application actuelle a le numéro de version 30 et l'étiquette de version 1.0. La prochaine mise à jour doit avoir au moins le numéro de version 31 et l'étiquette de version 1.0.1. Dans la plupart des cas, vous devez ajouter une troisième décimale à votre version en cas de corrections de bogues urgents ou de petits correctifs, la seconde décimale est principalement réservée aux mises à jour des fonctionnalités tandis que la première décimale est incrémentée en cas de mise à jour majeure.

Exporter un fichier IPA

Une fois cela fait, vous pouvez trouver votre archive dans l'organiseur Xcode. C'est ici que toutes vos versions précédentes et vos versions d'archive sont enregistrées et organisées si vous ne les supprimez pas. Vous remarquerez immédiatement un gros bouton bleu disant "Upload to App Store ..." mais dans 9/10 cas, cela ne fonctionnera pas pour diverses raisons (principalement les bogues Xcode). La solution consiste à exporter votre archive et à la télécharger en utilisant un autre outil Xcode appelé Application Loader. Cependant, puisque le chargeur d'application télécharge les fichiers IPA vers l'App Store, les archives doivent être exportées au format approprié. C'est une tâche triviale qui peut prendre environ une demi-heure. Cliquez sur le bouton "Exporter" dans le panneau latéral droit.

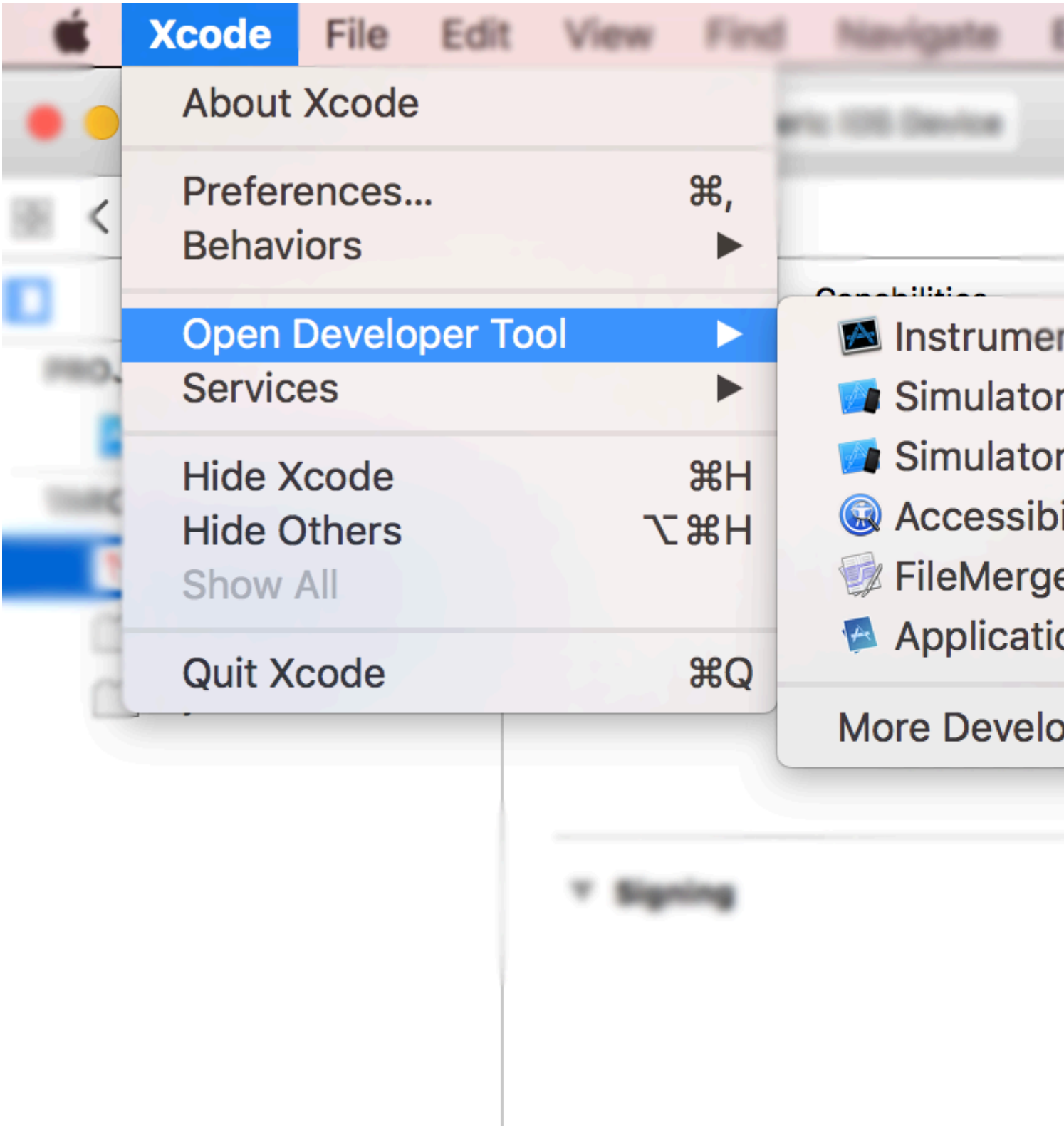


Si vous téléchargez une application sur l'App Store, sélectionnez la première option et cliquez sur

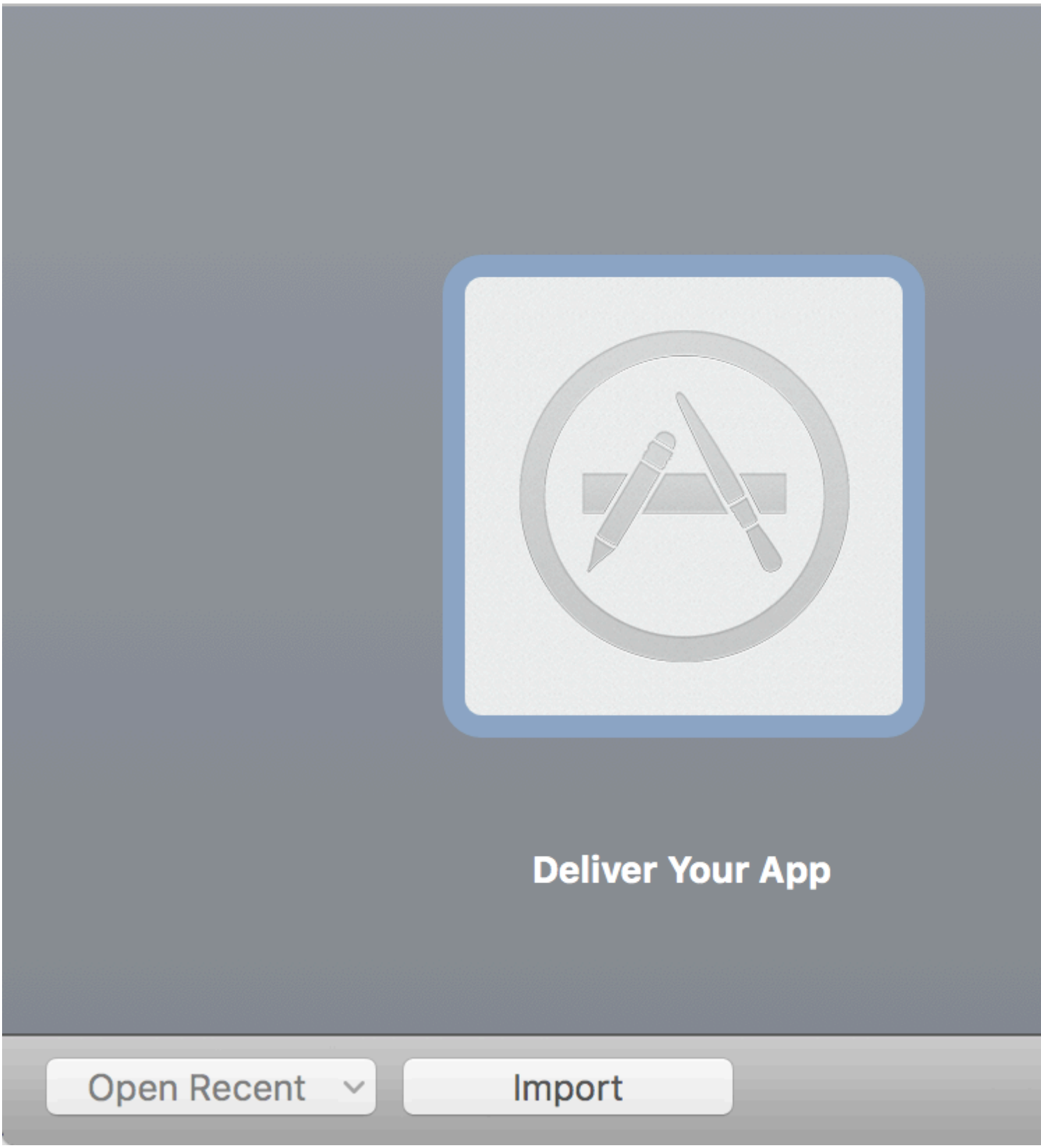
Suivant. Connectez-vous et validez à nouveau votre code et allez prendre une tasse de café. Une fois le processus d'exportation terminé, il vous sera demandé où enregistrer le fichier IPA généré. Habituellement, le bureau est le choix le plus pratique.

Télécharger un fichier IPA à l'aide d'Application Loader

Une fois le fichier IPA généré, ouvrez Xcode, accédez aux outils de développement et ouvrez Application Loader.



Si vous avez plusieurs comptes dans votre Xcode, il vous sera demandé de choisir. Choisissez naturellement celui que vous avez utilisé pour la signature du code dans la première étape. Choisissez "Livrer votre application" et téléchargez le code. Une fois le téléchargement terminé, l'affichage de votre liste de compilation sur iTunes Connect peut prendre jusqu'à une heure.



Lire Processus de soumission d'applications en ligne:
https://riptutorial.com/fr/ios/topic/8765/processus-de-soumission-d-applications

Chapitre 129: Profil avec instruments

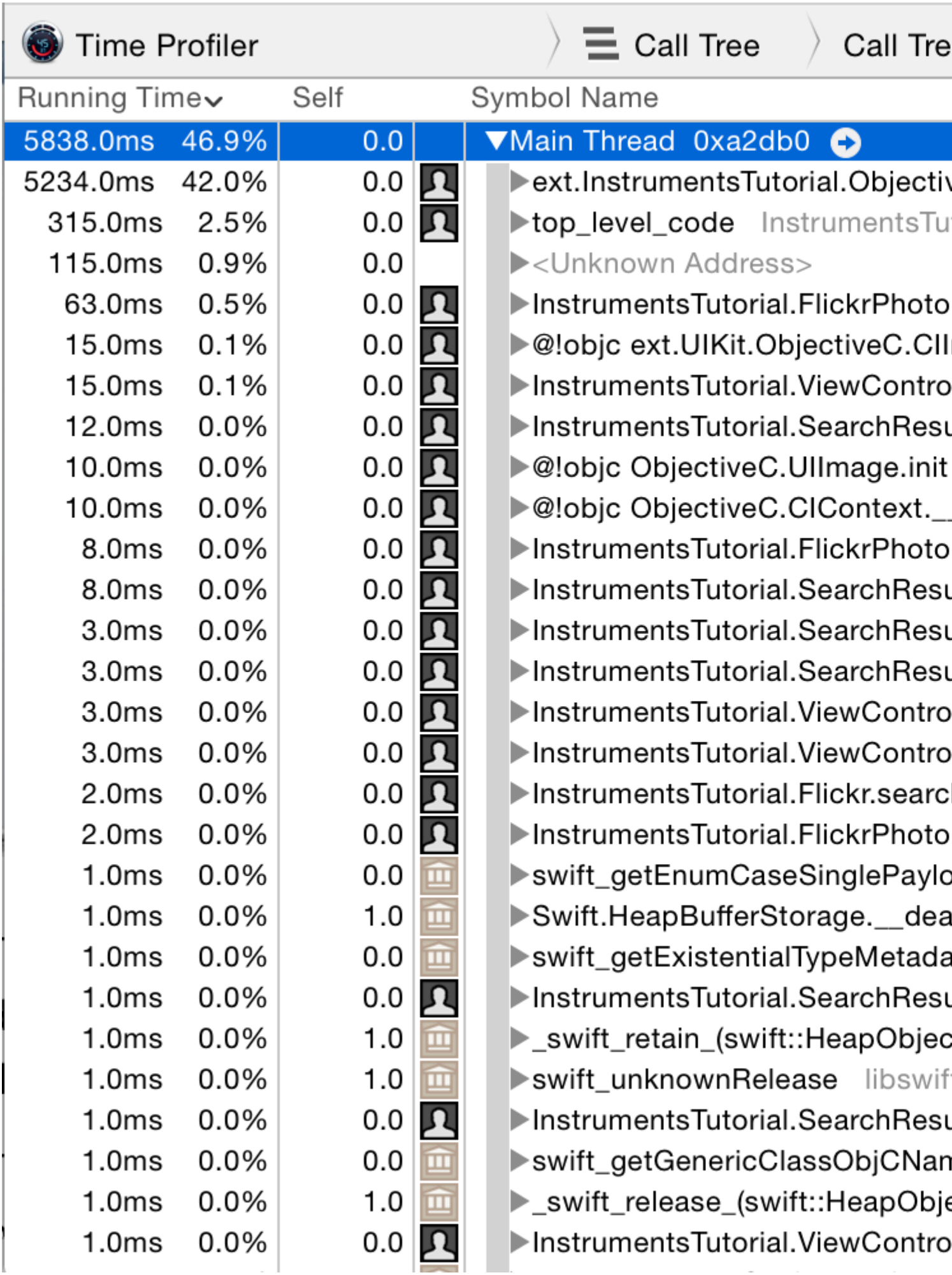
Introduction

Xcode inclut une application d'optimisation des performances appelée Instruments que vous pouvez utiliser pour profiler votre application en utilisant toutes sortes de mesures différentes. Ils ont des outils pour inspecter l'utilisation du processeur, l'utilisation de la mémoire, les fuites, l'activité de fichier / réseau et la consommation d'énergie, pour n'en nommer que quelques-uns. Il est vraiment facile de commencer à profiler votre application à partir de Xcode, mais il n'est pas toujours facile de comprendre ce que vous voyez lorsque vous profilez, ce qui dissuade certains développeurs de pouvoir utiliser cet outil à son plein potentiel.

Examples

Time Profiler

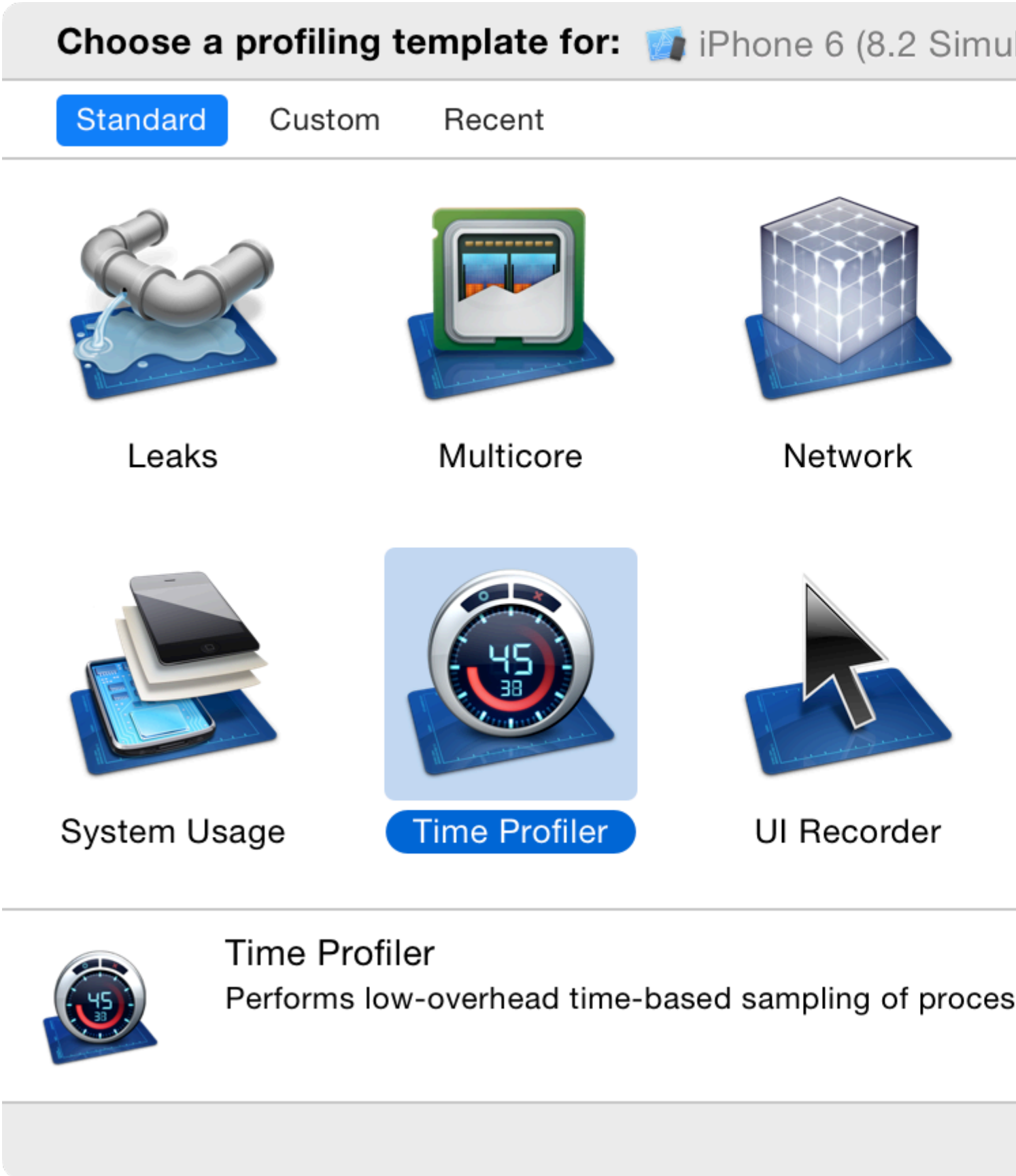
Le premier instrument que vous regardez est le `Time Profiler` . À des intervalles mesurés, Instruments arrêtera l'exécution du programme et effectuera une trace de pile sur chaque thread en cours d'exécution. Pensez-y en appuyant sur le bouton pause dans le débogueur de Xcode. Voici un aperçu du Time Profiler: -



Cet écran affiche l'`Call Tree`. L'`Call Tree`

indique le temps passé à exécuter différentes méthodes dans une application. Chaque ligne est une méthode différente du chemin d'exécution du programme. Le temps passé dans chaque méthode peut être déterminé à partir du nombre de fois où le profileur est arrêté dans chaque méthode. Par exemple, si **100** échantillons sont effectués à **intervalles de 1 milliseconde** et qu'une méthode particulière se trouve au sommet de la pile dans 10 échantillons, vous pouvez en déduire qu'environ **10%** du temps d'exécution total - **10 millisecondes** - a été utilisé. dans cette méthode. C'est une approximation assez grossière, mais ça marche!

Dans `Xcode's` barre de menus `Xcode's` , sélectionnez `Product\Profile` ou `press ⌘I` . Cela va construire l'application et lancer des instruments. Vous serez accueilli avec une fenêtre de sélection qui ressemble à ceci:



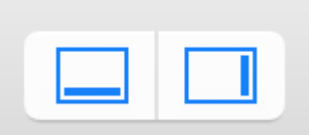
Ce sont tous des modèles différents fournis avec Instruments.

Sélectionnez l'instrument `Time Profiler` et cliquez sur Choisir. Cela ouvrira un nouveau document sur les instruments. Cliquez sur le **bouton d'enregistrement** rouge en haut à gauche pour lancer l'enregistrement et lancer l'application. On vous demandera peut-être votre mot de passe pour autoriser les **Instruments** à analyser d'autres processus - ne craignez rien, il est sécuritaire de

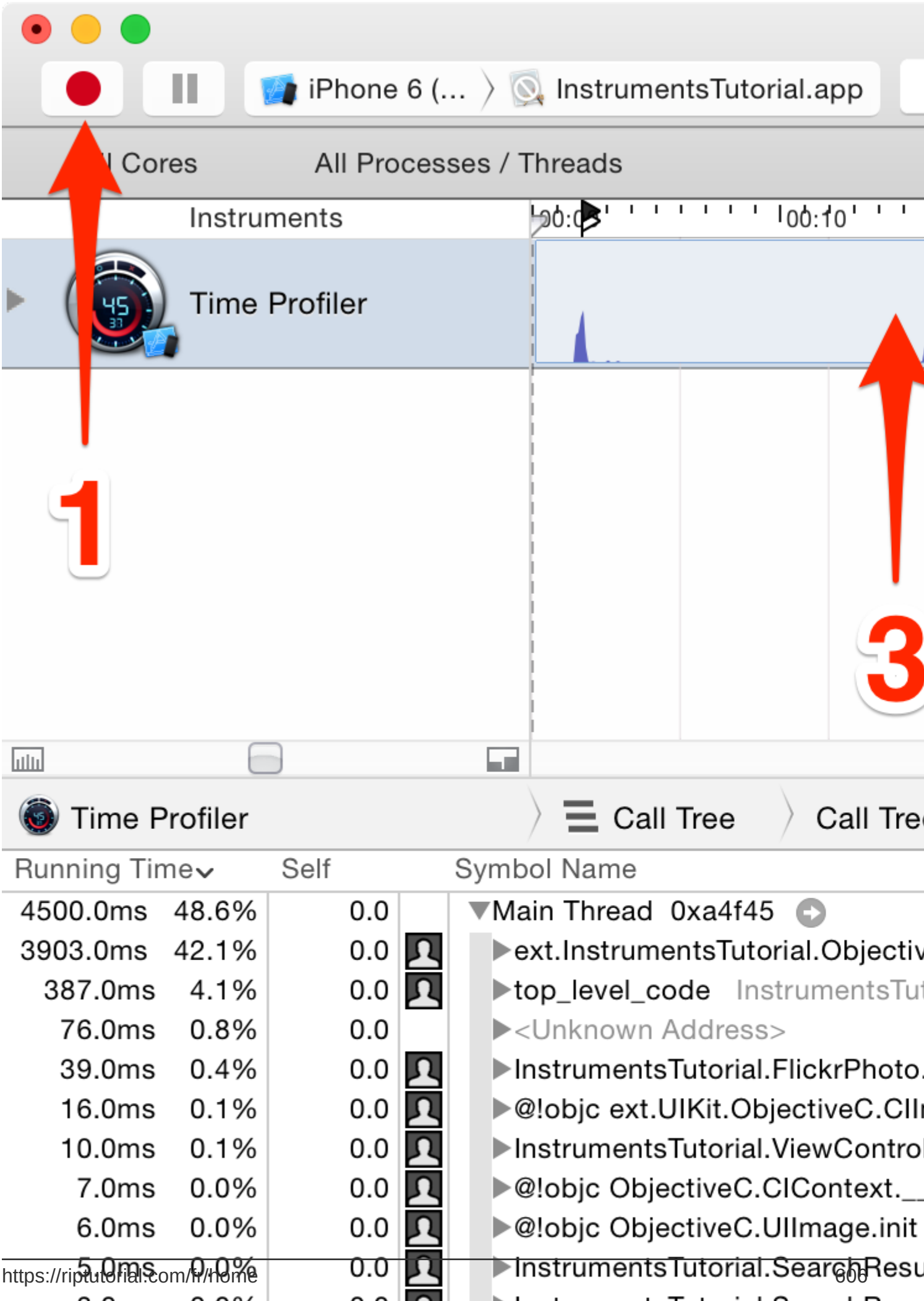
fournir ici! Dans la **fenêtre Instruments** , vous voyez le temps qui passe et une petite flèche se déplaçant de gauche à droite au-dessus du **graphique** au centre de l'écran. Cela indique que l'application est en cours d'exécution.

Maintenant, commencez à utiliser l'application. Recherchez des images et explorez un ou plusieurs résultats de recherche. Vous avez probablement remarqué que les résultats de recherche sont fastidieusement lents, et le fait de parcourir une liste de résultats de recherche est également incroyablement ennuyeux - c'est une application terriblement maladroite!

Eh bien, vous avez de la chance, car vous êtes sur le point de le réparer! Cependant, vous allez d'abord avoir un aperçu de ce que vous examinez dans **Instruments** . Tout d'abord, assurez-vous que le sélecteur de vue sur le côté droit de la barre d'outils a les deux options sélectionnées, comme ceci:



Cela garantira que tous les panneaux sont ouverts. Etudiez maintenant la capture d'écran ci-dessous et l'explication de chaque section en dessous:


iPhone 6 (...
InstrumentsTutorial.app
All Processes / Threads
Instrumentes
00:10
Time Profiler
1
3
Time Profiler
Call Tree
Running Time
Self
Symbol Name
4500.0ms 48.6% 0.0 Main Thread 0xa4f45
3903.0ms 42.1% 0.0 ext.InstrumentsTutorial.Objectiv
387.0ms 4.1% 0.0 top_level_code InstrumentsTut
76.0ms 0.8% 0.0 <Unknown Address>
39.0ms 0.4% 0.0 InstrumentsTutorial.FlickrPhoto.
16.0ms 0.1% 0.0 @!objc ext.UIKit.ObjectiveC.CIIr
10.0ms 0.1% 0.0 InstrumentsTutorial.ViewControl
7.0ms 0.0% 0.0 @!objc ObjectiveC.CIContext.__
6.0ms 0.0% 0.0 @!objc ObjectiveC.UIImage.init
5.0ms 0.0% 0.0 InstrumentsTutorial.SearchResu
3.0ms 0.0% 0.0 InstrumentsTutorial.SearchResu

. Le bouton rouge "enregistrer" arrête et démarre l'application en cours de profilage lorsque l'utilisateur clique dessus (il bascule entre une icône d'enregistrement et une icône d'arrêt). Le bouton de pause fait exactement ce que vous attendez et interrompt l'exécution en cours de l'application.

2. Ceci est la minuterie d'exécution. La minuterie compte combien de temps l'application est en cours d'exécution et combien de fois elle a été exécutée. Si vous arrêtez puis redémarrez l'application à l'aide des commandes d'enregistrement, cela déclencherait une nouvelle exécution et l'affichage afficherait alors l'exécution 2 de 2.

3. Ceci s'appelle une piste. Dans le cas du modèle Time Profiler que vous avez sélectionné, il n'y a qu'un seul instrument, il n'y a donc qu'une piste. Vous en apprendrez plus sur les spécificités du graphique présenté plus loin dans ce didacticiel.

4. Ceci est le panneau de détail. Il affiche les informations principales sur l'instrument que vous utilisez. Dans ce cas, il montre les méthodes les plus «chaudes», c'est-à-dire celles qui ont utilisé le plus de temps CPU. Si vous cliquez sur la barre en haut qui indique Call Tree (la main gauche) et sélectionnez Sample List, vous obtenez une vue différente des données. Cette vue montre chaque échantillon unique. Cliquez sur quelques exemples et vous verrez la trace de la pile capturée apparaître dans l'inspecteur des détails étendus.

5. Il s’agit du groupe d’inspecteurs. Il existe trois inspecteurs: Paramètres d'enregistrement, Paramètres d'affichage et Détails étendus. Vous en apprendrez plus sur certaines de ces options sous peu.

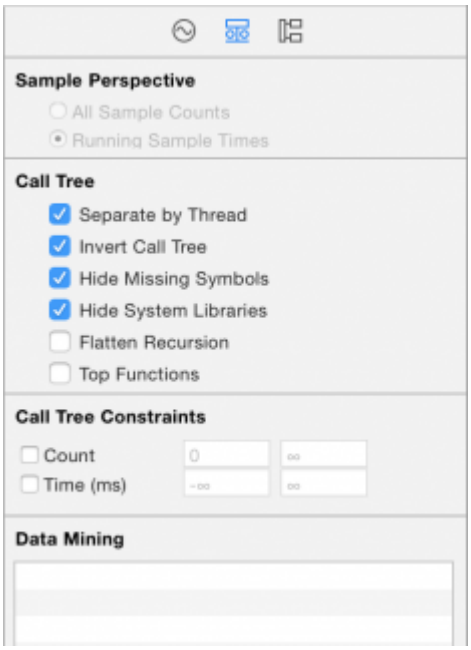
Forage profond

Effectuez une recherche d'image et explorez les résultats. Personnellement, j'aime bien chercher un «chien», mais choisissez ce que vous voulez - vous pourriez être l'un de ces chats!

Maintenant, faites défiler la liste vers le haut et le bas plusieurs fois pour avoir une bonne quantité de données dans `Time Profiler` . Vous devriez remarquer que les chiffres au milieu de l'écran changent et que le **graphique est** rempli. Cela vous indique que **les cycles du processeur** sont utilisés.

Vous ne vous attendriez vraiment pas à ce qu'une interface utilisateur soit aussi maladroite car cette `table view` est prête à être expédiée jusqu'à ce qu'elle défile comme du beurre! Pour identifier le problème, vous devez définir certaines options.

Sur le côté droit, sélectionnez l' **inspecteur Paramètres d'affichage** `(or press ⌥+2)` . Dans l' **inspecteur** , sous la section `Call Tree` , sélectionnez Séparer par **thread** , Inverser l' `Call Tree` , Masquer les symboles manquants et Masquer les bibliothèques système. Il ressemblera à ceci:



Voici ce que chaque option fait pour les données affichées dans le tableau à gauche:

Separate by Thread: Chaque thread doit être considéré séparément. Cela vous permet de comprendre quels threads sont responsables de la plus grande utilisation du **processeur** .

Inverser l'arborescence des appels: avec cette option, la `stack trace` la `stack trace` est considérée de haut en bas. C'est généralement ce que vous voulez, car vous voulez voir les méthodes les plus profondes où le **processeur** passe son temps.

Masquer les symboles manquants: Si le fichier `dSYM` est introuvable pour votre application ou une `system framework` , au lieu de voir les noms de méthode (symboles) dans la table, vous ne verrez que les valeurs hexadécimales correspondant aux adresses contenues dans le binaire. Si cette option est sélectionnée, seuls les symboles entièrement résolus sont affichés et les valeurs **hexadécimales** non résolues sont masquées. Cela permet de désencombrer les données présentées.

Masquer les bibliothèques système: Lorsque cette option est sélectionnée, seuls les symboles de votre propre application sont affichés. Il est souvent utile de sélectionner cette option, car vous ne vous souciez généralement pas de savoir où le **processeur** passe du temps dans votre propre code - vous ne pouvez pas faire grand chose sur la quantité de **CPU** `system libraries` du `system libraries` !

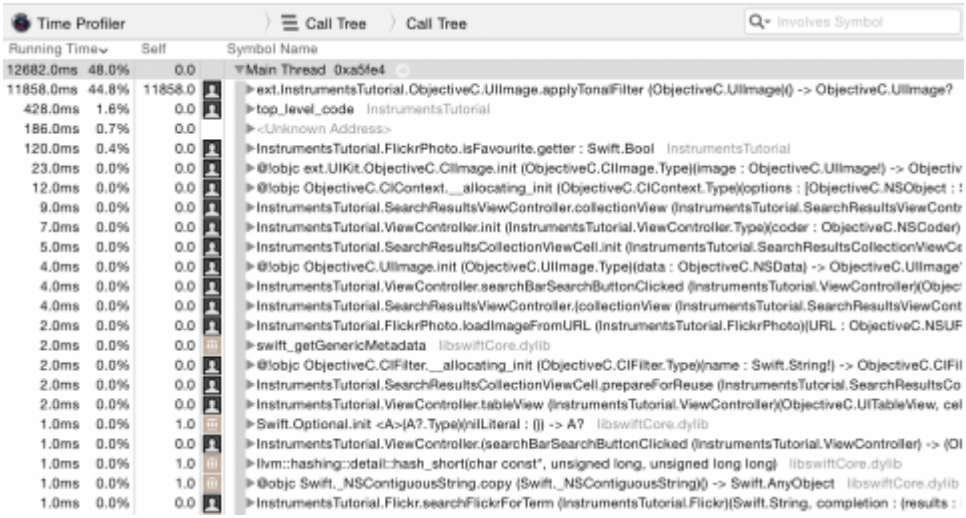
Aplatir la récursivité: cette option traite **les** fonctions **récursives** (celles qui s'appellent elles-mêmes) comme une seule entrée dans chaque `stack trace` , plutôt que comme une seule entrée.

Fonctions principales: L' activation de cette option fait que les `Instruments` compte le temps total passé dans une fonction comme la somme du temps directement dans cette fonction, ainsi que le temps passé dans les fonctions appelées par cette fonction.

Donc, si la fonction A appelle B, alors le temps de A est indiqué comme le temps passé dans A PLUS le temps passé en B. Cela peut être très utile, car il vous permet de choisir le plus grand chiffre à chaque descente dans la pile d'appels. sur vos méthodes les plus chronophages.

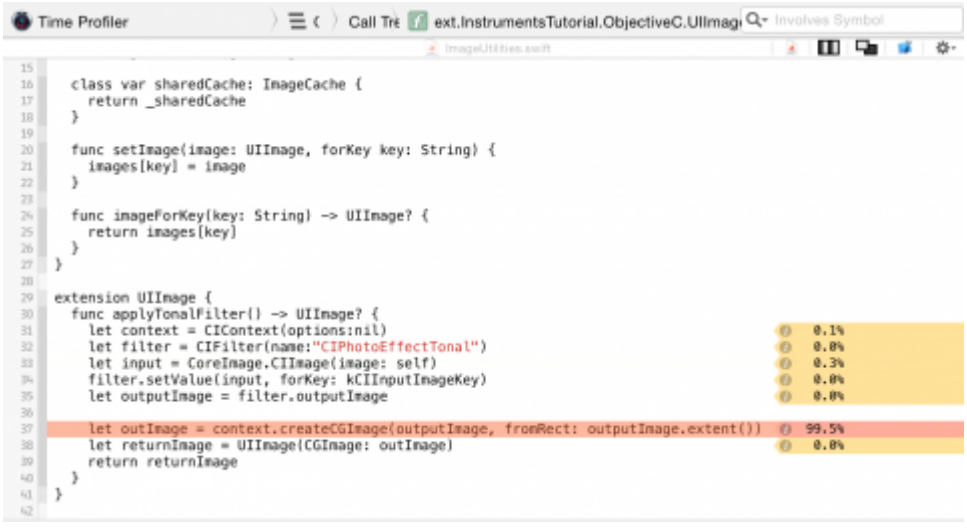
Si vous exécutez une application `Objective-C` , Show **Obj-C Only est également disponible** : Si cette option est sélectionnée, seules les méthodes `Objective-C` sont affichées, et non les fonctions `C` ou `C++` . Il n'y en a pas dans votre programme, mais si vous regardez une application `OpenGL` , cela peut avoir du `C++` , par exemple.

Bien que certaines valeurs puissent être légèrement différentes, l'ordre des entrées doit être similaire à celui du tableau ci-dessous une fois que vous avez activé les options ci-dessus:



Eh bien, cela n'a certainement pas l'air trop beau. La grande majorité du temps est consacrée à la méthode qui applique le filtre «tonal» aux vignettes. Cela ne devrait pas vous choquer trop, car le chargement et le défilement de la table étaient les éléments les plus clairs de l'interface utilisateur, et c'est à ce moment que les cellules de la table sont constamment mises à jour.

Pour en savoir plus sur ce qui se passe dans cette méthode, double-cliquez sur sa ligne dans la table. Cela fera apparaître la vue suivante:

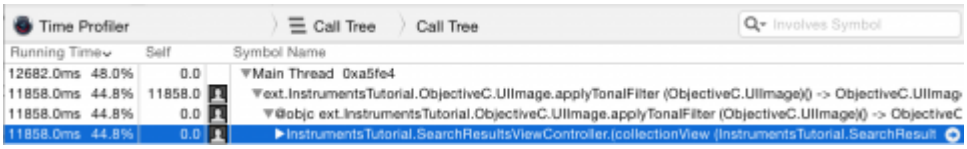


C'est intéressant, n'est-ce pas? `applyTonalFilter()` est une méthode ajoutée à `UIImage` dans une extension et presque **100** % du temps passé dans celle-ci est consacré à la création de la sortie CGImage après l'application du filtre d'image.

Il n'y a pas grand chose à faire pour accélérer les choses: la création de l'image est un processus assez intensif et prend autant de temps que nécessaire. Essayons de reculer et de voir où

`applyTonalFilter()` est appelé. **Cliquez sur** `Call Tree` dans le fil d'Ariane en haut de la vue de code pour revenir à l'écran précédent:

Cliquez maintenant sur la petite flèche à gauche de la ligne applyTonalFilter en haut de la table. Cela dépliera l'arbre d'appel pour afficher l'appelant de applyTonalFilter. Vous devrez peut-être déployer la prochaine rangée également; Lors du profilage de Swift, il y aura parfois des lignes en double dans l'arborescence des appels, avec le préfixe @objc. Vous êtes intéressé par la première ligne préfixée par le nom cible de votre application (InstrumentsTutorial):

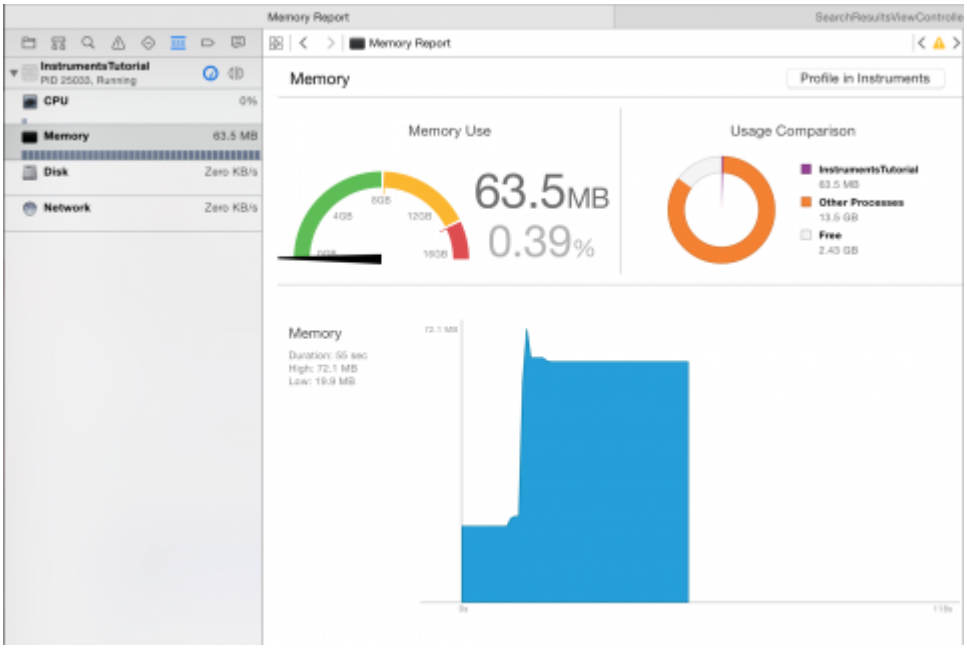


Dans ce cas, cette ligne fait référence à la vue de la collection de résultats `cellForItemAtIndexPath` . Double-cliquez sur la ligne pour afficher le code associé au projet.

Maintenant, vous pouvez voir quel est le problème. La méthode d'application du filtre tonal prend beaucoup de temps à exécuter et est appelée directement depuis cellForItemAtIndexPath, qui bloquera le `main thread` (et donc l'interface utilisateur entière) chaque fois qu'il demandera une image filtrée.

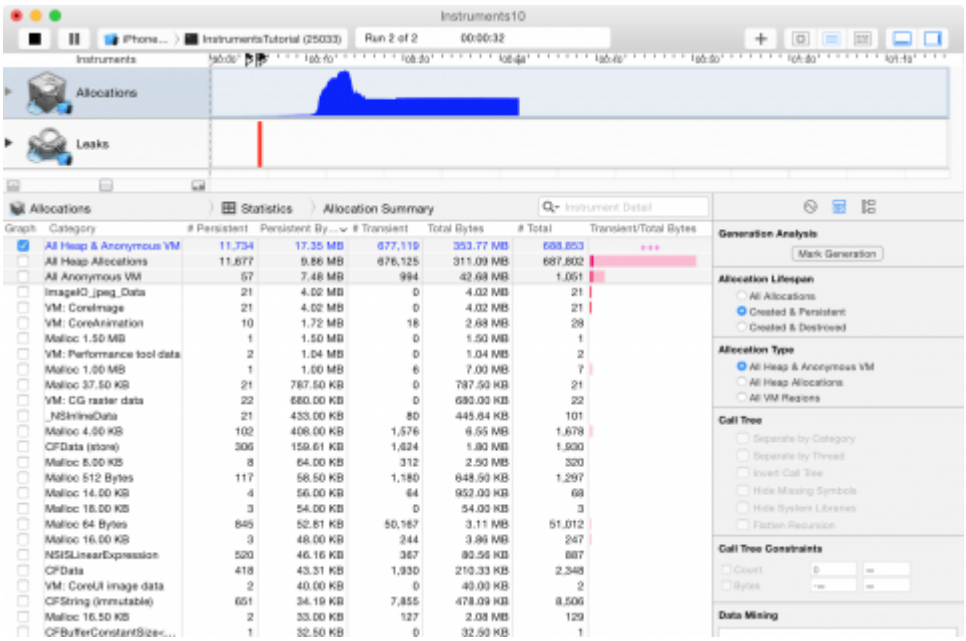
Allocations

Il existe des informations détaillées sur tous les **objets en** cours de création et la mémoire qui les sauvegarde. il montre également que vous `retain counts` pour chaque objet. Pour recommencer avec un nouveau `instruments profile` , quittez l'application Instruments. Cette fois, créez et exécutez l'application et ouvrez le navigateur de débogage dans la zone Navigateurs. Cliquez ensuite sur **Mémoire** pour afficher des graphiques sur l'utilisation de la mémoire dans la fenêtre principale:



Ces graphiques sont utiles pour avoir une idée rapide de la performance de votre application.

Mais vous allez avoir besoin d'un peu plus de puissance. Cliquez sur le bouton `Profile in Instruments` , puis sur Transférer pour amener cette session dans **Instruments** . L' **instrument d'allocation** démarrera automatiquement.



Cette fois, vous remarquerez deux pistes. L'un s'appelle Allocations et l'autre s'appelle Fuites. La piste Allocations sera discutée en détail plus tard; La piste Leaks est généralement plus utile dans Objective-C et ne sera pas abordée dans ce tutoriel. Alors, quel bug allez-vous traquer ensuite? Il y a quelque chose de caché dans le projet que vous ne savez probablement pas. Vous avez probablement entendu parler de fuites de mémoire. Mais ce que vous ne savez peut-être pas, c'est qu'il existe deux types de fuites:

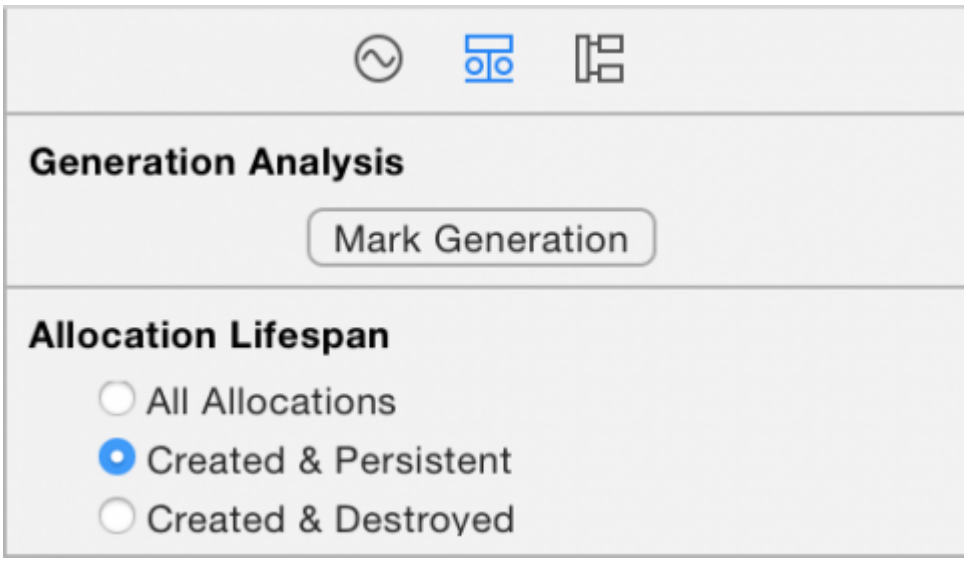
Les vraies fuites de mémoire sont celles où un objet n'est plus référencé par rien, mais toujours alloué - cela signifie que la mémoire ne peut jamais être réutilisée. Même si Swift et `ARC` aident à gérer la mémoire, le type de fuite de mémoire le plus courant est un `retain cycle or strong reference cycle` . C'est à ce moment que deux objets contiennent des références fortes les uns aux autres, de sorte que chaque objet empêche l'autre de se désallouer. Cela signifie que leur mémoire n'est jamais libérée!

La croissance de la mémoire sans limite est l'endroit où la mémoire continue à être allouée et n'a jamais la possibilité d'être **libérée** . Si cela continue pour toujours, alors `system's memory` sera remplie et vous aurez un gros problème de mémoire. Sous iOS, cela signifie que l'application sera détruite par le système.

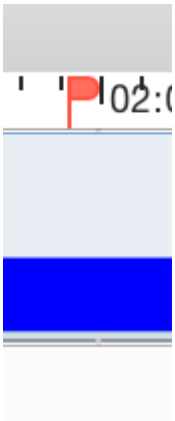
Avec **l'instrument** en cours d' exécution sur les allocations de l'application, faire cinq différentes recherches dans l'application , mais ne pas percer vers le bas dans les résultats encore. Assurez-vous que les recherches ont des résultats! Maintenant, laissez l'application régler un peu en attendant quelques secondes.

Vous devriez avoir remarqué que le **graphique** dans la piste Allocations a augmenté. Cela vous dit que la mémoire est allouée. C'est cette fonctionnalité qui vous guidera pour trouver `unbounded memory growth` .

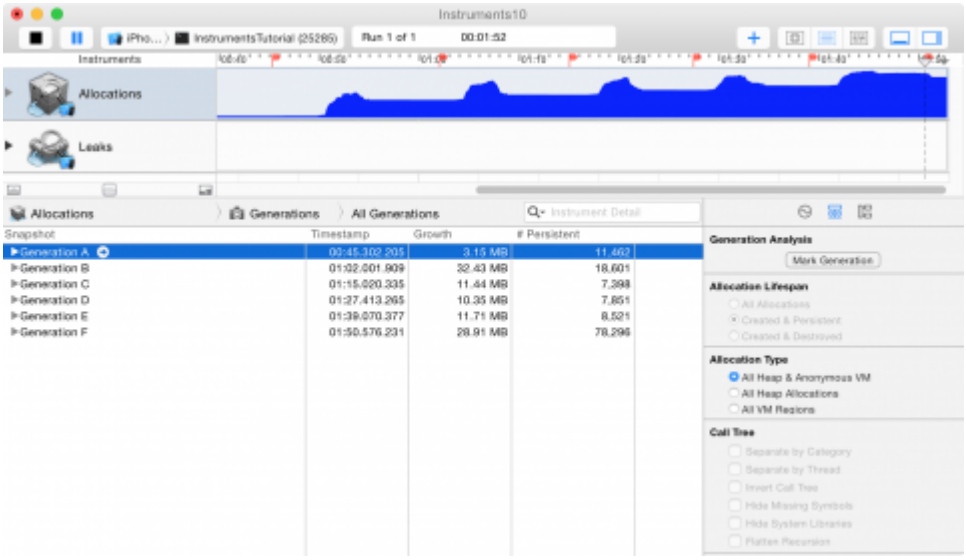
Vous allez effectuer une `generation analysis` . Pour ce faire, appuyez sur le bouton appelé Mark Generation. Vous trouverez le bouton en haut de l'inspecteur Paramètres d'affichage:



Appuyez dessus et vous verrez un drapeau rouge apparaître sur la piste, comme ceci:



Le but de l' `generation analysis` de `generation analysis` est d'effectuer une action plusieurs fois et de voir si la mémoire croît de `unbounded fashion` . **Effectuez** une recherche, attendez quelques secondes que les images soient chargées, puis revenez à la page principale. Puis marquez à nouveau la génération. Répétez cette opération pour différentes recherches. Après un **forage** dans quelques recherches, **instruments** ressemblera à ceci:



À ce stade, vous devriez être suspect. Remarquez comment le graphique bleu monte à chaque recherche dans laquelle vous **explorez** . Eh bien, ce n'est certainement pas bon. Mais attendez, qu'en est-il `memory warnings?` Vous en savez à propos de ça, non? `Memory warnings` sont la manière dont iOS indique à une application que les choses se resserrent dans le service de mémoire et que vous devez effacer de la mémoire.

Il est possible que cette croissance ne soit pas uniquement due à votre application. cela pourrait être quelque chose dans les profondeurs d' `UIKit` qui retient la mémoire. Donnez aux frameworks système et à votre application la possibilité d'éclaircir la **mémoire** avant de pointer un doigt vers l'une ou l'autre.

Simulez un `memory warning` en sélectionnant `Instrument\Simulate Memory Warning` dans la barre de menu de l'instrument ou `Hardware\Simulate Memory Warning` de `Hardware\Simulate Memory Warning` dans la barre de menus `simulator's` . Vous remarquerez que l'utilisation de la mémoire diminue légèrement, voire pas du tout. Certainement pas revenir là où ça devrait être. Donc, il y a encore **une croissance de la mémoire illimitée** qui se passe quelque part.

Le fait de marquer une génération après chaque itération de l'exploration d'une recherche permet de voir quelle **mémoire** a été allouée entre chaque génération. Jetez un œil dans le panneau de détails et vous verrez un tas de générations.

Lire Profil avec instruments en ligne: https://riptutorial.com/fr/ios/topic/9629/profil-avec-instruments

Chapitre 130: Redimensionner UIImage

Paramètres

CGInterpolationQualité	Niveaux d'interpolation de qualité pour le rendu d'une image.
La qualité d'interpolation est un paramètre d'état graphique	typedef enum CGInterpolationQuality CGInterpolationQuality;

Examples

Redimensionner une image en fonction de sa taille et de sa qualité

```
- (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;
}
```

Lire Redimensionner UIImage en ligne: https://riptutorial.com/fr/ios/topic/6422/redimensionner-uiimage

Chapitre 131: Référence CGContext

Remarques

Le type opaque **CGContextRef** représente une destination de dessin 2D Quartz. Un contexte graphique contient des paramètres de dessin et toutes les informations spécifiques au périphérique nécessaires pour rendre la peinture sur une page vers la destination, que la destination soit une fenêtre dans une application, une image bitmap, un document PDF ou une imprimante.

Examples

Dessiner une ligne

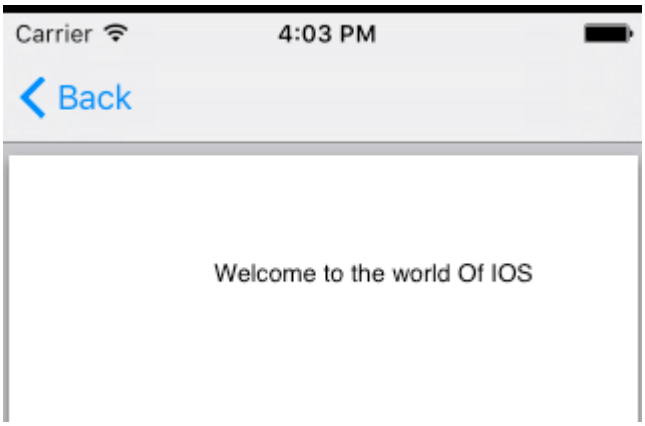
```
CGContextRef context = UIGraphicsGetCurrentContext();

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

Dessiner du texte

Draw To requiert que le **framework Core Text** soit ajouté dans la phase de construction

```
[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);
```



Lire Référence CGContext en ligne: https://riptutorial.com/fr/ios/topic/2664/reference-cgcontext

Chapitre 132: Rendre les coins sélectifs UIView arrondis

Examples

Objectif code C pour arrondir le coin sélectionné d'un UiView

Commencez par **importer** `#import <QuartzCore/QuartzCore.h>` dans votre classe ViewController. Voici comment je définis mon point de vue dans le code

```
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];
```

Voici la fonction qui fait le levage lourd et arrondit les arêtes sélectionnées qui sont le bord inférieur droit et le bord supérieur droit dans notre cas

```
- (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;
}
```

Lire Rendre les coins sélectifs UIView arrondis en ligne:
https://riptutorial.com/fr/ios/topic/7224/rendre-les-coins-selectifs-uiview-arrondis

Chapitre 133: Runtime en Objective-C

Examples

Utiliser des objets associés

Les objets associés sont utiles lorsque vous souhaitez ajouter des fonctionnalités à des classes existantes nécessitant un état de conservation.

Par exemple, ajouter un indicateur d'activité à chaque UIView:

Implémentation 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
```

Vous pouvez également accéder à l'environnement d'exécution Objective-C via Swift:

Code rapide

```
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()
    }
}
```

Lire Runtime en Objective-C en ligne: https://riptutorial.com/fr/ios/topic/10120/runtime-en-objective-c

Chapitre 134: Scanner de code QR

Introduction

Les codes QR (Quick Response) sont des codes à barres bidimensionnels largement utilisés sur les étiquettes optiques lisibles par une machine. iOS fournit un moyen de lire les codes QR à l'aide de l' `AVFoundation` partir d'iOS 7 et suivants. Ce framework fournit un ensemble d'API pour configurer / ouvrir la caméra et lire les codes QR à partir du flux de la caméra.

Examples

UIViewController recherchant QR et affichant une entrée vidéo

```
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` - sera appelée sur une analyse réussie `initCaptureSession` - initialiser le balayage pour QR et entrée de la caméra `dismissCaptureSession` - cacher l'entrée de la caméra et arrêter le balayage

Scanner le code QR avec le framework AVFoudation

Avant iOS 7, lorsque vous souhaitiez numériser un code QR, nous devions peut-être utiliser des structures tierces ou des bibliothèques telles que zBar ou zXing . Mais Apple a introduit `AVCaptureMetaDataOutput` d'iOS 7 pour la lecture des codes à barres.

Pour lire le code QR en utilisant `AVFoundation` nous devons configurer / créer `AVCaptureSession` et utiliser `captureOutput:didOutputMetadataObjects:fromConnection:` méthode delegate.

Étape 1

Importer le framework `AVFoundation` et confirmer le protocole `AVCaptureMetadataOutputObjectsDelegate`

```
import AVFoundation
class ViewController: UIViewController, AVCaptureMetadataOutputObjectsDelegate
```

Étape 2

La lecture du code QR est totalement basée sur la capture vidéo. Ainsi, pour capturer des vidéos en continu, créez une `AVCaptureSession` et configurez les entrées et les sorties des périphériques. Ajoutez le code ci-dessous dans la méthode `viewDidLoad` contrôleur de vue

```
// 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
}
```

Étape 3

Implémenter la méthode déléguée `AVCaptureMetadataOutputObjectsDelegate` pour lire le code 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
        }
    }
}
```

Ici, les objets de métadonnées peuvent également vous fournir les limites du code QR lu sur le fil de la caméra. Pour obtenir les limites, transmettez simplement l'objet de métadonnées à la méthode `transformedMetadataObject videoPreviewLayer` comme ci-dessous.

```
let barCodeObject = videoPreviewLayer?.transformedMetadataObject(for: metadataObj)
        qrCodeFrameView?.frame = barCodeObject!.bounds
```

Lire Scanner de code QR en ligne: https://riptutorial.com/fr/ios/topic/7963/scanner-de-code-qr

Chapitre 135: Sécurité

Introduction

La sécurité dans iOS est liée à la sécurité des données, à la sécurité du transport, à la sécurité du code, etc.

Examples

Transport Security utilisant SSL

Les applications iOS doivent être écrites de manière à sécuriser les données transportées sur le réseau.
SSL est le moyen le plus courant de le faire.

Chaque fois que l'application tente d'appeler des services Web pour extraire ou transférer des données vers des serveurs, elle doit **utiliser le protocole SSL sur HTTP, c'est-à-dire HTTPS** .
Pour ce faire, l' **application doit appeler** `https://server.com/part` tels services Web et non `http://server.com/part` .
Dans ce cas, l'application doit faire confiance au serveur `server.com` aide du certificat SSL.

Voici l'exemple de la validation de la confiance du serveur

Implémenter `URLSessionDelegate` tant que:

```
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)
}
```

Voici le gestionnaire de confiance: (impossible de trouver le code 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
```

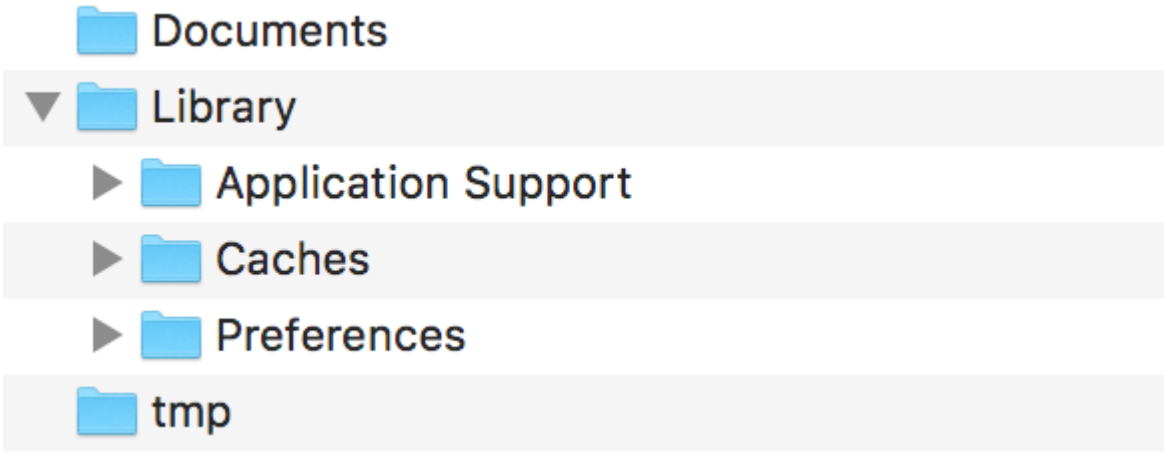
Server_Public_SSL_Cert.der est la clé SSL publique des serveurs.

En utilisant cette approche, notre application peut s’assurer qu’elle communique avec le serveur prévu et que personne n’intercepte la communication avec le serveur d’application.

Sécurisation des données dans les sauvegardes iTunes

Si nous voulons que nos données d'application soient protégées contre les sauvegardes iTunes, nous devons ignorer les données de notre application pour les sauvegarder dans iTunes. Chaque fois qu'un appareil iOS est sauvegardé à l'aide d'iTunes sur macOS, toutes les données stockées par toutes les applications sont copiées dans cette sauvegarde et stockées sur l'ordinateur de sauvegarde.

Mais nous pouvons exclure nos données d'application de cette sauvegarde en utilisant la clé `URLResourceKey.isExcludedFromBackupKey` .
Voici la structure de répertoire de notre application:



Remarque: les données généralement sensibles sont stockées dans le répertoire "Application Support".

Par exemple, si nous voulons exclure toutes nos données stockées dans le répertoire **Application Support** , nous pouvons utiliser les clés mentionnées ci-dessus comme suit:

```
    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
    }
```

De nombreux outils sont disponibles pour afficher les sauvegardes iTunes de toutes les données sauvegardées afin de vérifier si l’approche ci-dessus fonctionne ou non.
iExplorer est un bon outil pour explorer les sauvegardes iTunes.

Lire Sécurité en ligne: https://riptutorial.com/fr/ios/topic/9999/securite

Chapitre 136: Segues

Examples

Un aperçu

De la documentation Apple:

> Un objet UIStoryboardSegue est chargé d' **effectuer la transition visuelle entre deux contrôleurs de vue** . De plus, les objets segue sont utilisés pour préparer la transition d'un contrôleur de vue à un autre. **Les objets Segue contiennent des informations sur les contrôleurs de vue impliqués dans une transition** . Lorsqu'un segue est déclenché, mais avant la transition visuelle, le runtime du storyboard appelle la méthode prepareForSegue: sender: du contrôleur de vue en cours pour qu'il puisse transmettre les données nécessaires au contrôleur de vue sur le point d'être affiché.

Les attributs

Rapide

```
sourceViewController: UIViewController {get}
destinationViewController: UIViewController {get}
identifier: String? {get}
```

Les références:

- Référence de la classe UIViewController
- Référence de la classe UIStoryBoardSegue

Préparer votre contrôleur de vue avant le déclenchement d'une Segue

PrepareForSegue :

```
func prepareForSegue(_ segue:UIStoryboardSegue, sender sender:AnyObject?)
```

> Notifie le contrôleur de vue qu'une suite est sur le point d'être effectuée

Paramètres

segue : l'objet segue.

expéditeur : objet qui a initialisé le segue.

Exemple dans Swift

Effectuer une tâche si l'identifiant de la segue est "SomeSpecificIdentifier"

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
    if segue.identifier == "SomeSpecificIdentifier" {
        //- Do specific task
    }
}
```

Décider si une Segue appelée doit être effectuée.

ShouldPerformSegueWithIdentifier :

```
func shouldPerformSegueWithIdentifier(_ identifier:String, sender sender:AnyObject?) -> Bool
```

Détermine si le lien avec l'identificateur spécifié doit être effectué.

Paramètres

Identifiant : Chaîne qui identifie le segue déclenché

Sender : objet qui a initialisé le segue.

Exemple dans Swift

N'effectuez que segue si l'identifiant est "SomeSpecificIdentifier"

```
override func shouldPerformSegueWithIdentifier(identifier:String, sender:AnyObject?) -> Bool {
    if identifier == "SomeSpecificIdentifier" {
        return true
    }
    return false
}
```

Utiliser Segues pour naviguer en arrière dans la pile de navigation

---

Déroulez Segues

Sewind Segues vous permet de «dérouler» la pile de navigation et de spécifier la destination à suivre. La signature de cette fonction est la clé de la reconnaissance par Interface Builder. **Il doit avoir une valeur de retour de IBAction et prendre un paramètre de UIStoryboardSegue** . Le nom de la fonction n'a pas d'importance. En fait, la fonction n'a même rien à faire. C'est juste comme un marqueur dont UIViewController est la destination de la Segue Dérouler. [source] [1]

Signature obligatoire d'un seunding

Objectif c:

```
-(IBAction)prepareForUnwind:(UIStoryboardSegue *)segue {
}
```

Rapide:

```
@IBAction func prepareForUnwind(segue: UIStoryboardSegue) {
}
```

Segue déclencheur par programmation

PerformSegueWithIdentifier:

```
 func performSegueWithIdentifier(_ identifier:String, sender sender:AnyObject?)
```

> Lance le lien avec l'identifiant spécifié à partir du fichier de storyboard du contrôleur de vue en cours

Paramètres

Identifiant : Chaîne qui identifie le segue déclenché

Sender : objet qui initiera le segue.

Exemple dans Swift

Exécution d'un lien avec l'identifiant "SomeSpecificIdentifier" à partir d'une sélection de ligne de la vue de table:

```
func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
    performSegueWithIdentifier("SomeSpecificIdentifier", sender: indexPath.item)
}
```

Lire Segues en ligne: https://riptutorial.com/fr/ios/topic/5575/segues

Chapitre 137: Services Safari

Examples

Implémenter SFSafariViewControllerDelegate

Vous devez implémenter `SFSafariViewControllerDelegate` afin que votre classe soit avertie lorsque l'utilisateur appuie sur le bouton Done de SafariViewController et que vous pouvez également le supprimer.

D'abord, déclarez votre classe pour implémenter le protocole.

```
class MyClass: SFSafariViewControllerDelegate {

}
```

Implémentez la méthode des délégués à notifier en cas de licenciement.

```
func safariViewControllerDidFinish(controller: SFSafariViewController) {
    // Dismiss the SafariViewController when done
    controller.dismissViewControllerAnimated(true, completion: nil)
}
```

N'oubliez pas de définir votre classe en tant que délégué SafariViewController.

```
let safariVC = SFSafariViewController(URL: yourURL)
safariVC.delegate = self
```

Les méthodes de délégation supplémentaires que vous pouvez implémenter sont les suivantes:

```
// 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] { }
```

Ajouter des articles à la liste de lecture Safari

Vous pouvez ajouter des éléments à la liste de lecture d'un utilisateur dans Safari en appelant la méthode `addItem` sur le singleton `SSReadingList` .

```
let readingList = SSReadingList.default()
readingList?.addItem(with: yourURL, title: "optional title", previewText: "optional preview
text")
```

La liste de lecture par défaut peut être `nil` si l'accès à la liste de lecture n'est pas autorisé.

De plus, vous pouvez vérifier si la liste de lecture prend en charge une URL en appelant `supportsURL` .

```
SSReadingList.default().supportsURL(URL(string: "https://example.com")!)
```

Cela renverra `true` ou `false` indiquer si l'URL donnée est prise en charge par la liste de lecture Safari. Utilisez ceci par exemple pour déterminer s'il faut afficher un bouton pour ajouter une URL à la liste de lecture.

Ouvrir une URL avec SafariViewController

N'oubliez pas d'importer le framework nécessaire en premier.

```
import SafariServices
//Objective-C
@import SafariServices;
```

Instanciez une instance `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];
```

Si vous le souhaitez, vous pouvez également indiquer à SafariViewController si possible une fois le chargement terminé.

```
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];
```

Présenter le contrôleur de vue.

```
present(safariVC, animated: true, completion: nil)
//Objective-C
[self presentViewController:sfvc animated:YES completion:nil];
```

Lire Services Safari en ligne: https://riptutorial.com/fr/ios/topic/1371/services-safari

Chapitre 138: Signature de code

Examples

Profils d'approvisionnement

Pour créer un fichier IPA dans XCode, vous devez signer votre application avec un certificat et un profil d'approvisionnement. Ceux-ci peuvent être créés à l' adresse https://developer.apple.com/account/ios/profile/create

Types de profil d'approvisionnement

Les profils de provisioning sont divisés en deux types, Développement et Distribution:

Développement

- Développement d'applications iOS / développement d'applications tvOS - utilisé en développement pour installer votre application sur un périphérique de test.

Distribution

- App Store / tvOS App Store - utilisé pour signer votre application pour le téléchargement de l'app store.
- In House - Utilisé pour la distribution Enterprise de votre application, sur les appareils de votre entreprise.
- Ad Hoc / tvOS Ad Hoc - Utilisé pour distribuer votre application à un nombre limité de périphériques spécifiques (par exemple, vous devez connaître les UDID des périphériques sur lesquels vous souhaitez installer votre application).

Lire Signature de code en ligne: https://riptutorial.com/fr/ios/topic/6055/signature-de-code

Chapitre 139: Simulateur

Introduction

Les simulateurs iOS, watchOS et tvOS sont d'excellents moyens de tester vos applications sans utiliser d'appareil. Ici, nous allons parler de travailler avec des simulateurs.

Remarques

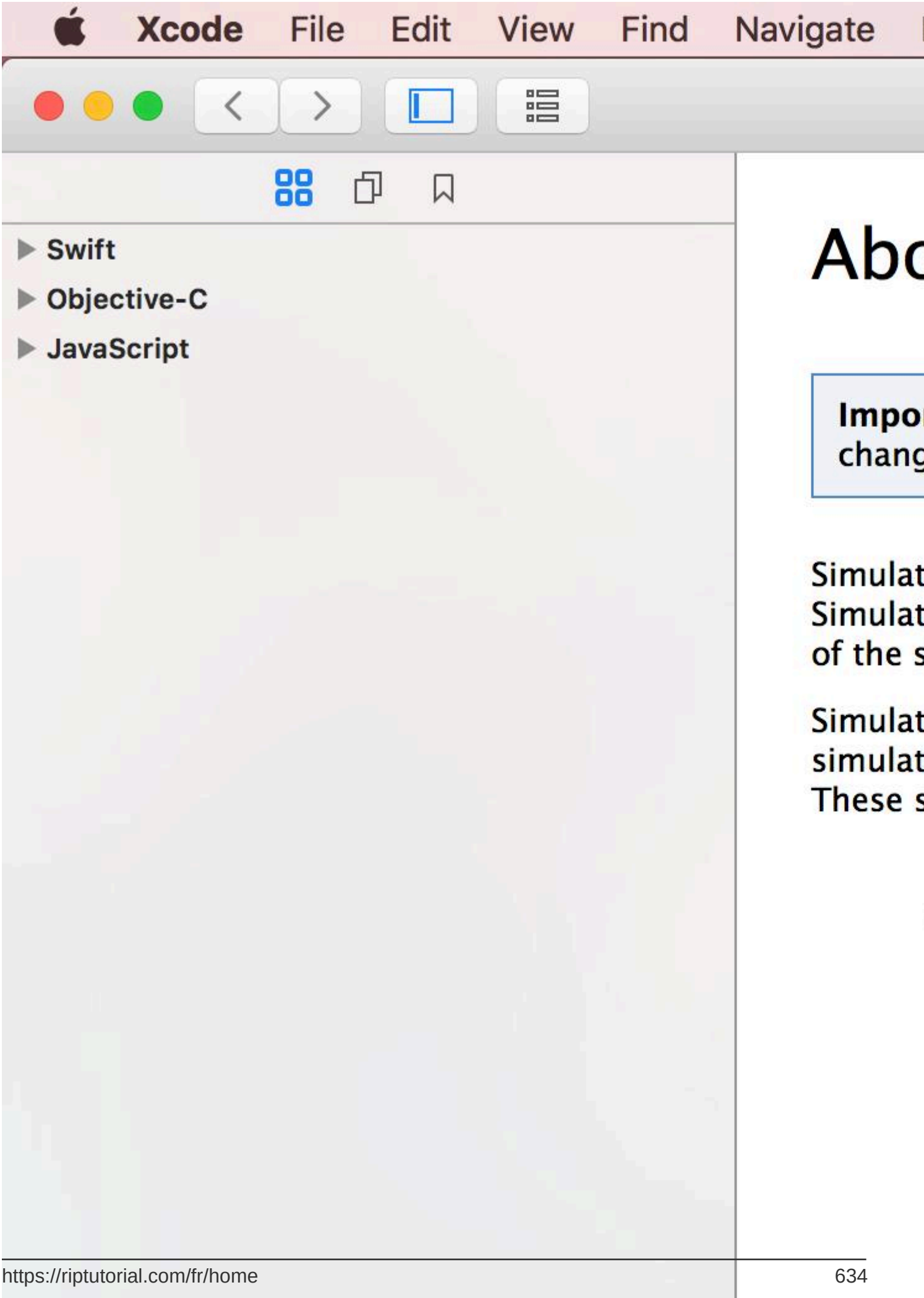
Différents types de simulateurs

- Simulateur iOS
- watchOS simulateur
- simulateur tvOS
- Simulateur de barre tactile

  Il n'y a pas de simulateur pour macOS, car Xcode est exécuté sur MacOS et chaque fois que nécessaire, il lancera les applications natives.

Obtenir de l'aide

Vous pouvez toujours consulter l'aide de Simulator dans Aide -> Aide du simulateur:


Xcode File Edit View Find Navigate
Swift
Objective-C
JavaScript
Abo
Impo
chang
Simulat
Simulat
of the s
Simulat
simulat
These s

Chapitre 140: Simulateur construit

Introduction

Où trouver la construction du simulateur?

Allez dans ~ / Library / Developer / CoreSimulator / Devices /

Vous trouverez des répertoires avec des noms alphanumériques

puis cliquez sur l'un des répertoires et effectuez la sélection suivante

Données / Conteneurs / Bundle / Application /

Encore une fois, vous trouverez des répertoires avec des noms alphanumériques si vous cliquez sur ce que vous trouverez

Simulateur construit là-bas

Remarque:

L'installation d'un appareil iOS sur un simulateur ne fonctionnera pas.

simulateur iPhone utilise l'architecture i386 simulateur iPad construit utilise x8

Examples

Installation de la construction manuellement sur simulateur

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

Lire Simulateur construit en ligne: https://riptutorial.com/fr/ios/topic/9813/simulateur-construit

Chapitre 141: Simulation de l'emplacement à l'aide de fichiers GPX iOS

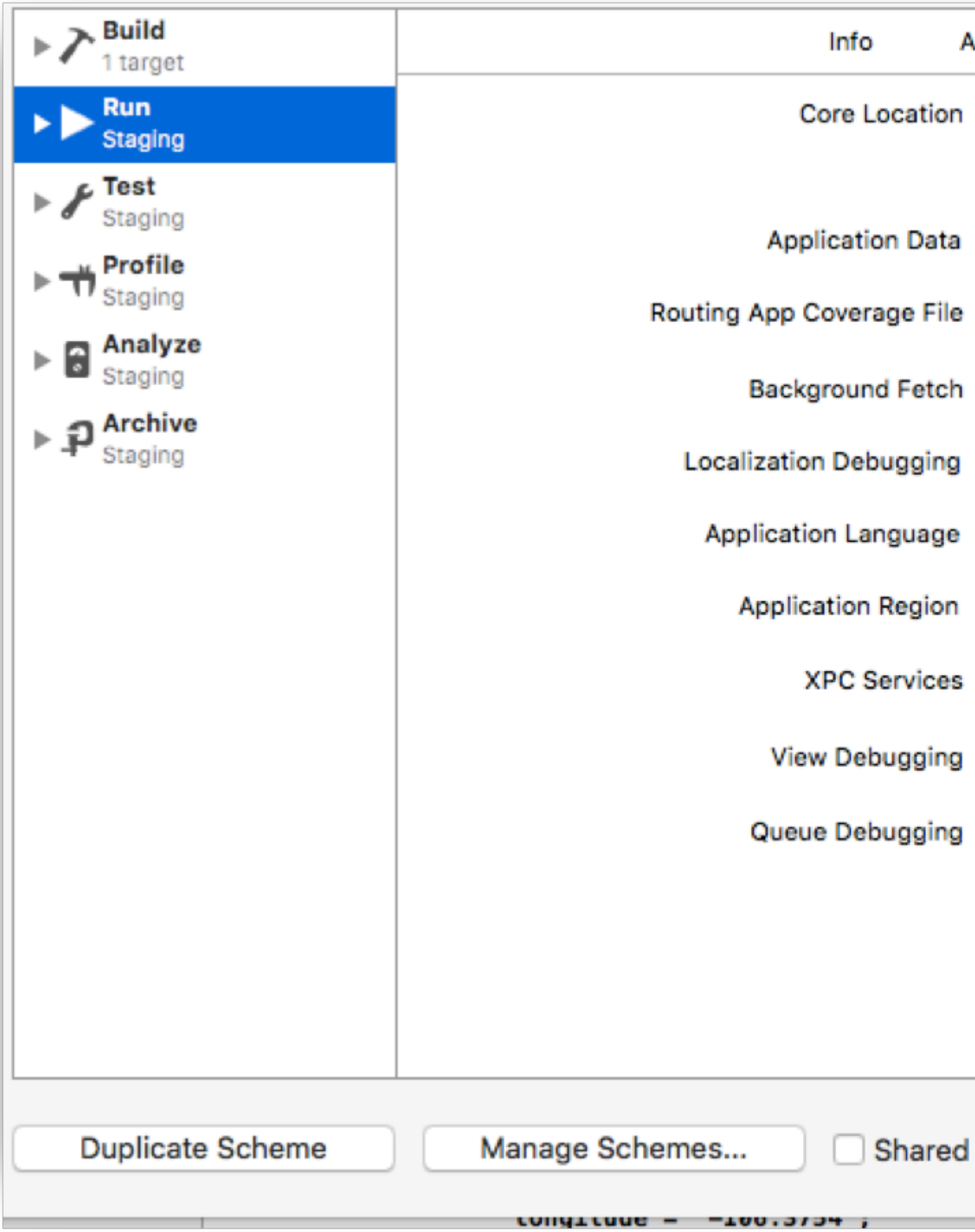
Examples

Votre fichier .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>
```

Pour définir cet emplacement:

1. Accédez à Edit Scheme.
2. Sélectionnez Exécuter -> Options.
3. Cochez "Autoriser la simulation de localisation".
4. Sélectionnez le nom de fichier * .GPX dans la liste déroulante "Emplacement par défaut".



Lire Simulation de l'emplacement à l'aide de fichiers GPX iOS en ligne:
https://riptutorial.com/fr/ios/topic/9883/simulation-de-l-emplacement-a-l-aide-de-fichiers-gpx-ios

Chapitre 142: SiriKit

Remarques

Différents types de requêtes Siri

- Réservation de tour (par exemple, faites-moi un tour à New York via MyApp)
- Messagerie (par exemple, envoyer un texte à John en utilisant MyApp)
- Recherche de photos (par exemple, recherche de photos de plage prises l'été dernier dans MyApp)
- Paiements (p. Ex., Envoi de 20 $ à John pour dîner la nuit dernière en utilisant MyApp)
- Appels VoIP (par exemple, appeler Mike sur mon MyApp)
- Entraînements (par exemple, démarrer mon entraînement quotidien à partir de MyApp)
- Climat et radio (spécialement conçus pour CarPlay, par exemple, réglez le chauffage à 72 degrés)

Examples

Ajout de l'extension Siri à l'application

Pour intégrer les fonctionnalités Siri dans votre application, vous devez ajouter une extension comme vous le feriez lors de la création d'un widget iOS 10 (ancienne extension de vue Today) ou d'un clavier personnalisé.

Ajout de capacité

1- Dans les paramètres du projet, sélectionnez la cible de votre application iOS et accédez à l'onglet Capabilities.

2- Activer la capacité Siri

Ajout de l'extension

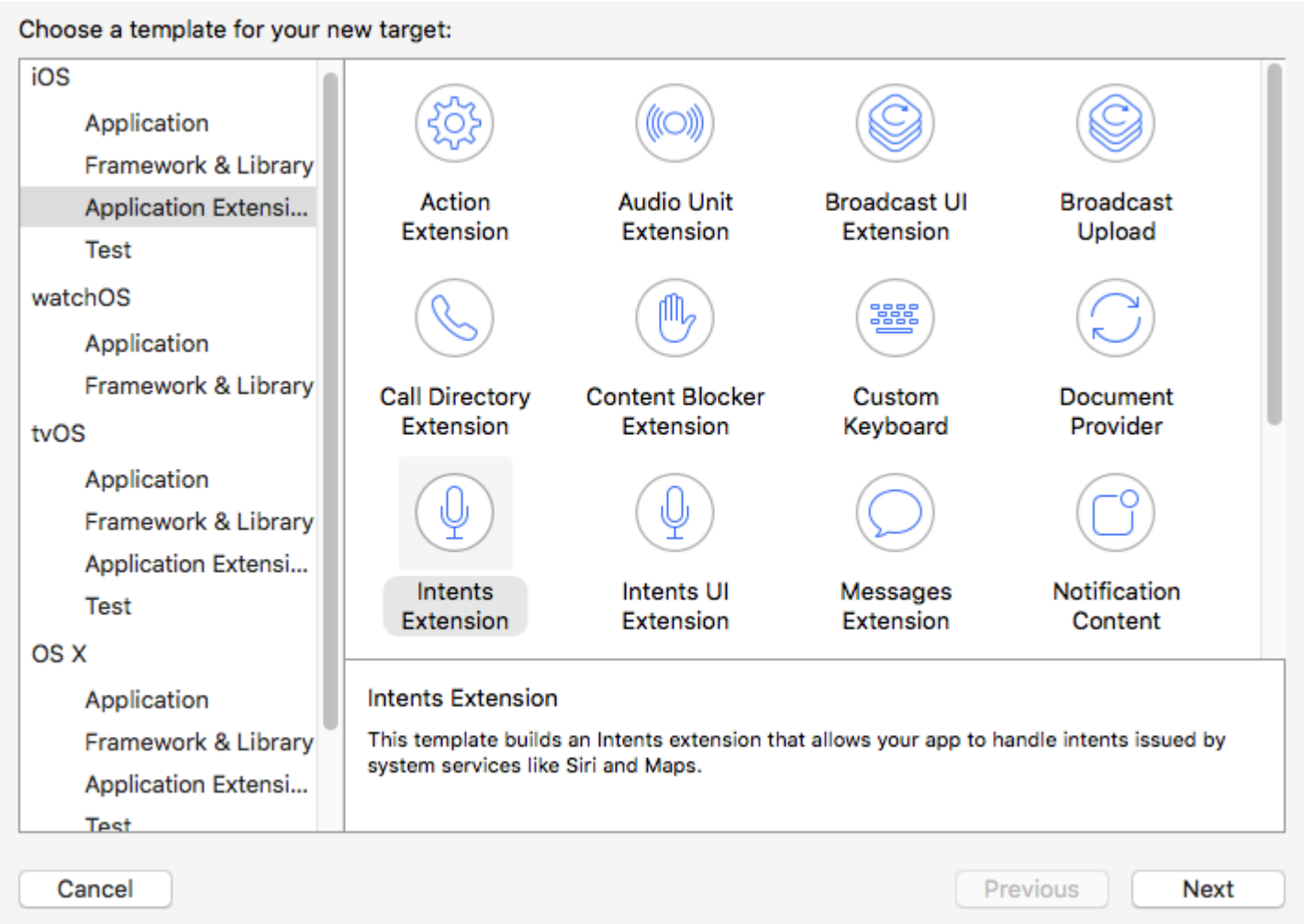
1- Allez dans Fichier -> Nouveau -> Cible ...

2- Sélectionnez iOS -> Extension d'application dans le volet gauche

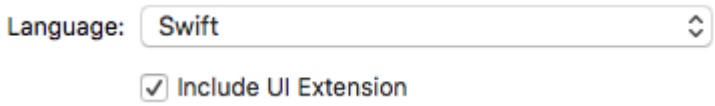
3- Double-cliquez sur Intentions Extension depuis la droite

Selon Apple:

Intents Extension template construit une extension Intents qui permet à votre application de gérer les intentions émises par les services système tels que Siri et Maps.



4- Choisissez un nom et assurez-vous de cocher "Inclure l'extension UI"



En procédant ainsi, deux nouvelles cibles (extension des intentions et extension d'interface utilisateur) sont créées et, par défaut, elles contiennent un code d'intention d'entraînement. Pour différents types de requêtes Siri, voir Remarques.

Remarque

Chaque fois que vous souhaitez déboguer votre extension, sélectionnez simplement le schéma d'intention parmi les schémas disponibles.

Remarque

Vous ne pouvez pas tester les applications SiriKit dans le simulateur. Au lieu de cela, vous avez besoin d'un appareil réel.

Lire SiriKit en ligne: https://riptutorial.com/fr/ios/topic/5869/sirikit

Chapitre 143: SLComposeViewController

Examples

SLComposeViewController pour Twitter, Facebook, SinaWelbo et TencentWelbo

Objectif c

Ajoutez d'abord le `Social Framework` au projet XCode.

Importez la classe `#import "Social/Social.h"` dans le ViewController requis

Twitter avec texte, image et lien

```
//- - 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 avec texte, image et lien

```
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeFacebook])
{
    SLComposeViewController *fbVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeFacebook];
    [fbVC setInitialText:text];
    //To send link together with text
    [fbVC addURL:[NSURL URLWithString:@"https://twitter.com/IbrahimH_ss_n"]];
    //To add a photo to a link
```

```
    [fbVC addImage:[UIImage imageNamed:@"image"]];
    [self presentViewController:fbVC animated:YES completion:nil];
}
else
{//Shows alert if twitter is not signed in
       UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to twitter."preferredStyle:UIAlertControllerStyleAlert];
        [self presentViewController:alertCont animated:YES completion:nil];
        UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay"
style:UIAlertActionStyleDefault handler:nil];
        [alertCont addAction:okay];
}
```

SinaWeibo

```
//- - SinaWeibo - -
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeSinaWeibo]){

    SLComposeViewController *SinaWeiboVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeSinaWeibo];
    [SinaWeiboVC setInitialText:text];

    [self presentViewController:SinaWeiboVC animated:YES completion:nil];
}
else
{
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to SinaWeibo."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay" style:UIAlertActionStyleDefault
handler:nil];
    [alertCont addAction:okay];
}
```

TencentWeibo

```
//- -TencentWeibo text share
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeTencentWeibo])
{
    SLComposeViewController *tencentWeiboVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTencentWeibo];
    [tencentWeibo setInitialText:text];
    [self presentViewController:tencentWeibo animated:YES completion:nil];
}
else
{
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to SinaWeibo."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay" style:UIAlertActionStyleDefault
handler:nil];
    [alertCont addAction:okay];
}
```

Lire SLComposeViewController en ligne:
https://riptutorial.com/fr/ios/topic/7366/slcomposeviewcontroller

Chapitre 144: StoreKit

Examples

Obtenez des informations sur les produits localisés à partir de l'App Store

Obtenez des informations de produit localisées à partir d'un ensemble de chaînes d'identificateurs de produit à l'aide de `SKProductsRequest` :

```
import StoreKit

let productIdentifierSet = Set(["yellowSubmarine", "pennyLane"])
let productsRequest = SKProductsRequest(productIdentifiers: productIdentifierSet)
```

Afin de traiter les produits de `productsRequest` , nous devons affecter un délégué à la requête qui gère la réponse. Le délégué doit se conformer au protocole `SKProductsRequestDelegate` , ce qui signifie qu'il doit hériter de `NSObject` (c.-à-d. Tout objet `Foundation`) et implémenter la méthode `productsRequest` :

```
class PaymentManager: NSObject, SKProductsRequestDelegate {

    var products: [SKProduct] = []

    func productsRequest(request: SKProductsRequest,
                         didReceiveResponse response: SKProductsResponse) {

        products = response.products

    }

}
```

Pour initier les `productsRequest` nous `PaymentManager` comme `PaymentManager` de la demande de produits et appelons la méthode `start()` sur la demande:

```
let paymentManager = PaymentManager()
productsRequest.delegate = paymentManager
productsRequest.start()
```

Si les demandes réussissent, les produits seront dans le `paymentManager.products` .

Lire StoreKit en ligne: https://riptutorial.com/fr/ios/topic/6025/storekit

Chapitre 145: Storyboard

Introduction

Normalement, les contrôleurs de vue dans un storyboard sont instanciés et créés automatiquement en réponse aux actions définies dans le storyboard même. Toutefois, vous pouvez utiliser un objet storyboard pour instancier le contrôleur de vue initial dans un fichier de storyboard ou instancier d'autres contrôleurs de vue que vous souhaitez présenter par programme. Vous trouverez ci-dessous des exemples des deux cas d'utilisation.

Examples

Initialiser

```
//Swift
let storyboard = UIStoryboard(name: "Main", bundle: NSBundle.mainBundle())

//Objective-c
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:[NSBundle
mainBundle]];
```

Récupérer le ViewController initial

```
//Swift
let initialScreen = storyboard.instantiateInitialViewController()

//Objective-c
UIViewController *initailScreen = [storyboard instantiateInitialViewController];
```

Récupérer ViewController

```
//Swift
let viewController = storyboard.instantiateViewControllerWithIdentifier("identifier")

//Objective-c
UIViewController *viewController = [storyboard
instantiateViewControllerWithIdentifier:@"identifier"];
```

Lire Storyboard en ligne: https://riptutorial.com/fr/ios/topic/3514/storyboard

Chapitre 146: Swift: Modifier le rootViewController dans AppDelegate pour présenter le flux principal ou de connexion / d'intégration

Introduction

Il est souvent utile de présenter une première expérience aux nouveaux utilisateurs de votre application. Cela peut être pour un certain nombre de raisons, telles que les inviter à se connecter (si requis pour votre situation), expliquer comment utiliser l'application ou simplement les informer des nouvelles fonctionnalités d'une mise à jour (comme le font Notes, Photos et Musique). iOS11).

Remarques

Tout d'abord, comme vous traitez de multiples flux, c'est là que les Storyboards peuvent être utilisés efficacement. Par défaut, votre application utilise `Main.storyboard` pour votre flux principal. Votre flux d'intégration / alternatif peut être contenu dans un storyboard secondaire, par exemple. `Onboarding.storyboard`

Cela présente de nombreux avantages:

- dans une équipe de développeurs, le travail sur chaque flux utilisateur peut être séparé
- contrôle de source plus clair (git)
- séparation des préoccupations

Lorsque votre application se lance, vous pouvez déterminer quel flux doit être présenté. La logique pour cela peut être contenue dans votre AppDelegate:

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    let isFirstRun = true // logic to determine goes here
    if isFirstRun {
        showOnboarding()
    }
    return true
}
```

Pour afficher le flux d'intégration, il est utile de se demander comment gérer l'expérience de le rejeter une fois que le client l'a terminé et qui est sémantiquement correct pour ce que vous essayez de créer.

Approches:

Les deux approches principales sont:

1. Permuter le contrôleur de vue racine de la fenêtre principale de l'application
2. Présenter le flux d'intégration en tant que parcours modal, chevauchant le flux principal.

L'implémentation de ceci devrait être contenue dans une extension à AppDelegate.

Examples

Option 1: permuter le contrôleur de vue racine (bon)

Il y a des avantages à changer de contrôleur de vue racine, même si les options de transition sont limitées à celles prises en charge par `UIViewAnimationOptions` .

Vous pouvez afficher le flux d'intégration en définissant simplement `UIApplication.shared.keyWindow.rootViewController`

Le licenciement est géré en utilisant `UIView.transition(with:)` et en transmettant le style de transition sous la forme d'un `UIViewAnimationOptions` , en l'occurrence Cross Dissolve. (Flips et Curls sont également supportés).

Vous devez également définir le cadre de la vue principale avant d'y revenir, car vous l'instanciez pour la première fois.

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

Option 2: Flux alternatif actuel modéré (meilleur)

Dans l'implémentation la plus simple, le flux d'intégration peut simplement être présenté dans un contexte modal, puisque l'utilisateur est sémantiquement sur un seul trajet.

> [Directives relatives à l'interface utilisateur Apple - Modalité] [1]:
>
> Pensez à créer un contexte modal uniquement lorsqu'il est essentiel d'attirer l'attention de quelqu'un, lorsqu'une tâche doit être terminée ou abandonnée pour continuer à utiliser l'application ou pour enregistrer des données importantes.

La présentation modale permet la simple option de licenciement en fin de parcours, avec peu de ressources sur les contrôleurs de permutation.

Les transitions personnalisées sont également prises en charge de manière standard, car cela utilise 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)
        }
    }
}
```

Lire Swift: Modifier le rootViewController dans AppDelegate pour présenter le flux principal ou de connexion / d'intégration en ligne: https://riptutorial.com/fr/ios/topic/10880/swift--modifier-le-rootviewcontroller-dans-appdelegate-pour-presenter-le-flux-principal-ou-de-connexion---d-integration

Chapitre 147: SWRevealViewController

Remarques

L'utilisation de la classe SWRevealViewController en tant que navigation principale peut ne pas toujours aboutir à la meilleure expérience utilisateur. Si la barre latérale ne contient que 5 entrées ou moins (ou si le contenu peut être compressé en 5 entrées ou moins), vous devriez envisager d'utiliser la barre d'onglets par défaut.

La barre d'onglets est intuitive et permet à l'utilisateur de changer rapidement entre les vues / contextes. D'autre part, la navigation dans la barre latérale peut effectuer plus d'actions que de changer de vue / contexte et utilise moins d'espace lorsqu'elle est réduite.

Pour plus d'informations, consultez les directives d'interface utilisateur iOS d'Apple.

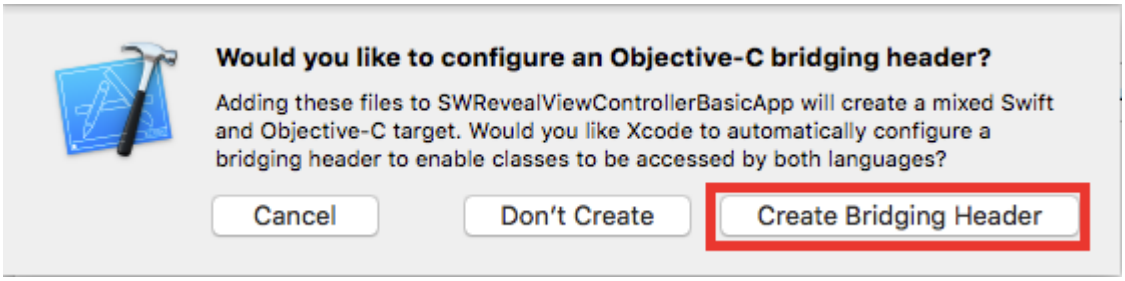
Examples

Configurer une application de base avec SWRevealViewController

Créer une application de base avec un modèle d'application à vue unique avec swift as language

Ajoutez `SWRevealViewController.h` et `SWRevealViewController.m`

puis cliquez sur le bouton Créer un en-tête de pontage

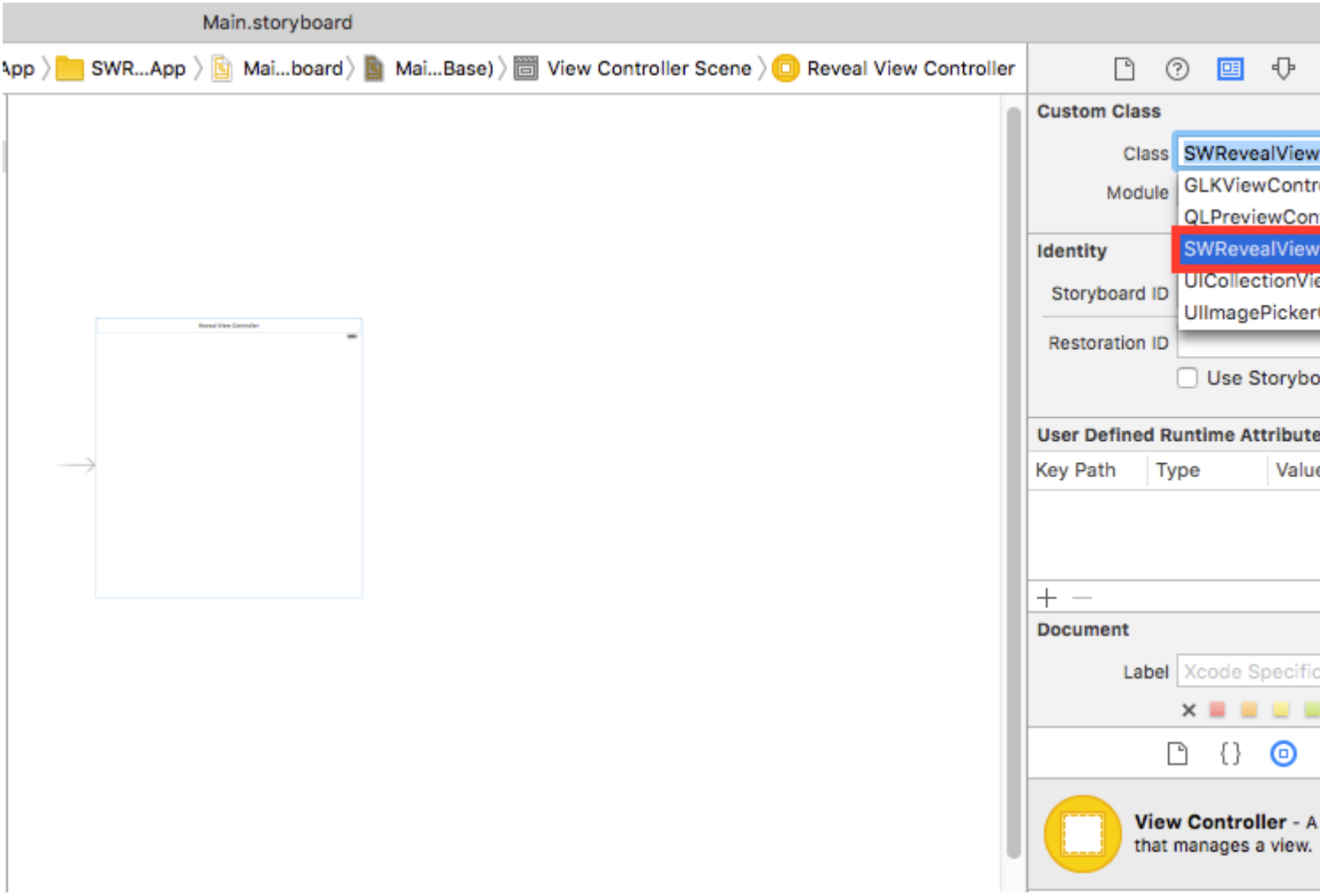


et ajouter

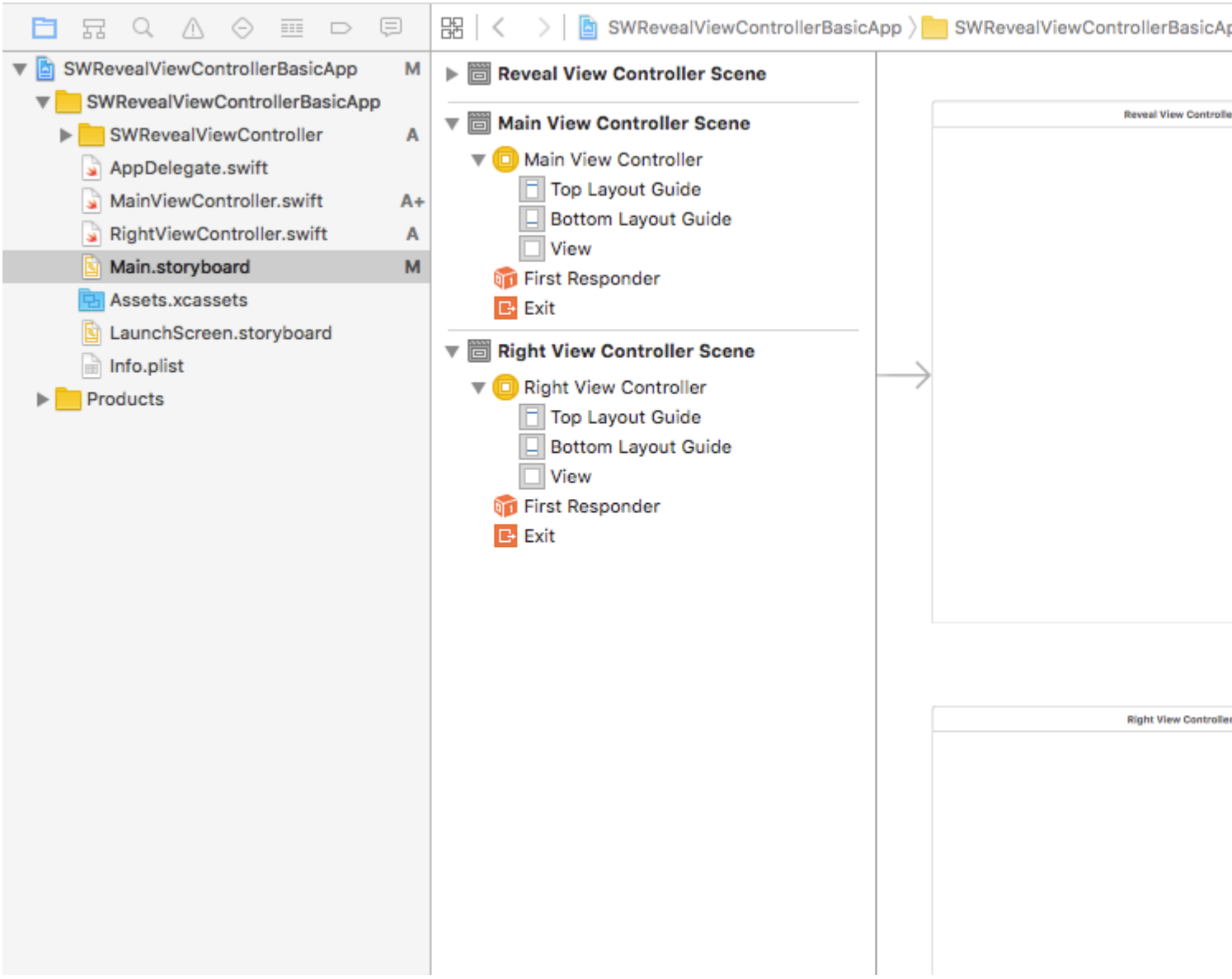
```
#import "SWRevealViewController.h"
```

sur l'en-tête de pontage

Puis sélectionnez viewController sur le storyboard et changez de classe en `SWRevealViewController`

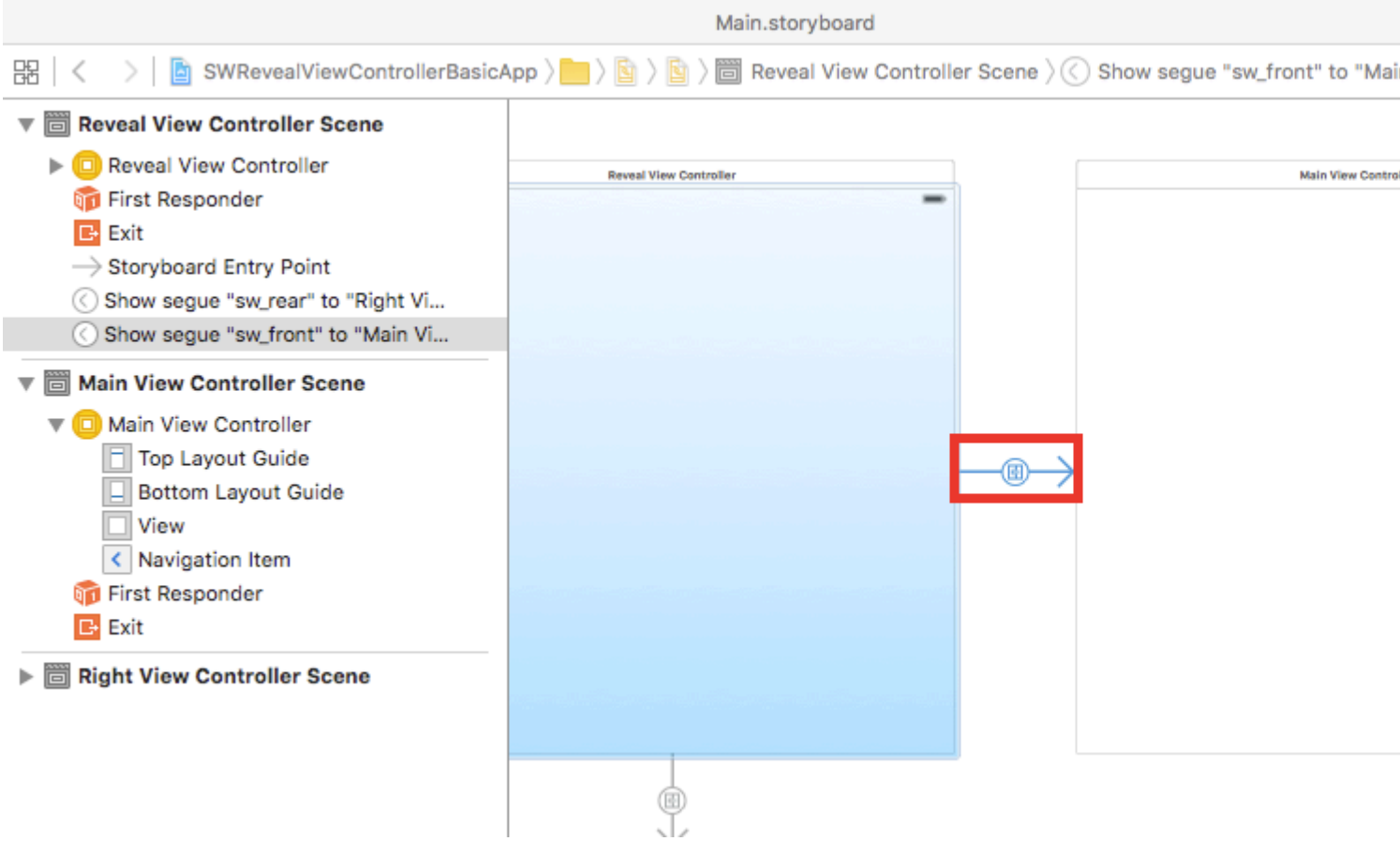


Ensuite, renommez le viewController des fichiers en MainViewController et ajoutez un nouveau ViewController avec le nom RightViewController



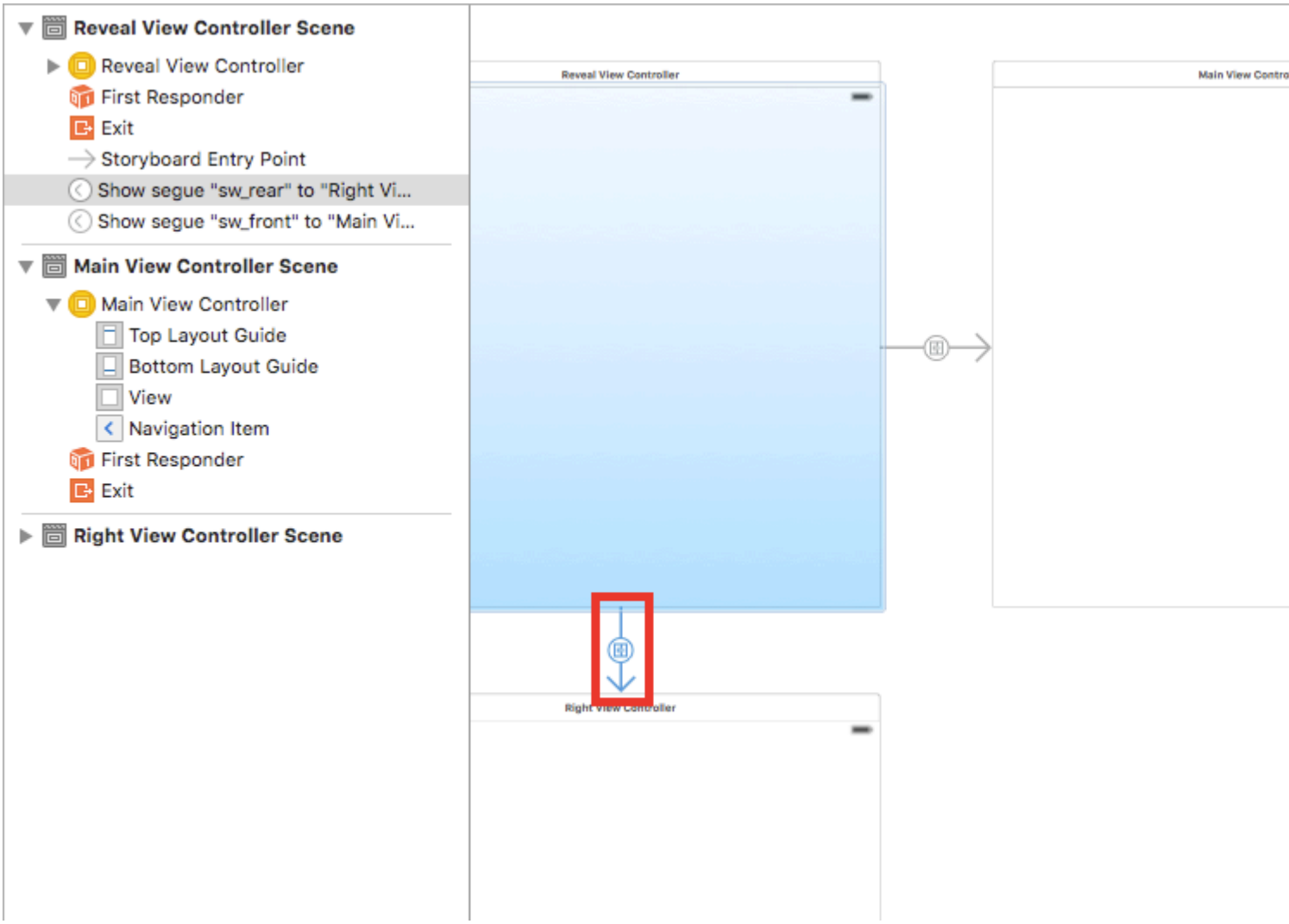
puis nous ajoutons deux segues de SWRevealViewController à MainViewController et de SWRevealViewController à RightViewController, puis nous devons sélectionner le premier (de SWRevealViewController à MainViewController) et éditer les propriétés

sur l'identifiant set `sw_front` sur le jeu de classes `SWRevealViewControllerSegueSetController`



après cela, nous devons faire la même chose avec le segue (de SWRevealViewController à RightViewController)

sur l'identifiant set `sw_rear` sur le jeu de classes `SWRevealViewControllerSegueSetController`



puis sur MainViewController ajoutez cette ligne à la méthode `viewDidLoad`

```
self.view.addGestureRecognizer(self.revealViewController().panGestureRecognizer());
```

Et c'est tout, vous avez une application de base avec SWRevealViewController intégré, vous pouvez glisser à droite pour afficher `RightViewController` tant que menu latéral

Lire SWRevealViewController en ligne:
https://riptutorial.com/fr/ios/topic/4614/swrevealviewcontroller

Chapitre 148: Test de l'interface utilisateur

Syntaxe

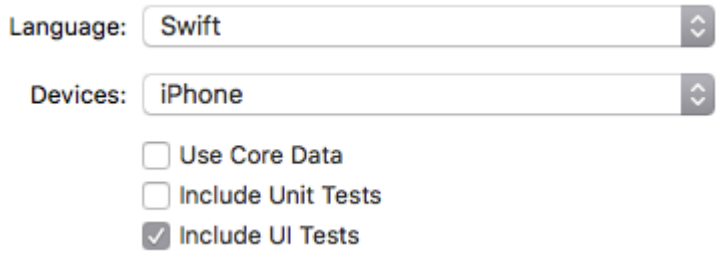
- XCUIApplication () // Proxy pour une application. Les informations identifiant l'application sont spécifiées dans les paramètres cibles de Xcode comme "Application cible".
- XCUIElement () // Un élément d'interface utilisateur dans une application.

Examples

Ajout de fichiers de test à un projet Xcode

Lors de la création du projet

Vous devez cocher "Inclure les tests d'interface utilisateur" dans la boîte de dialogue de création de projet.

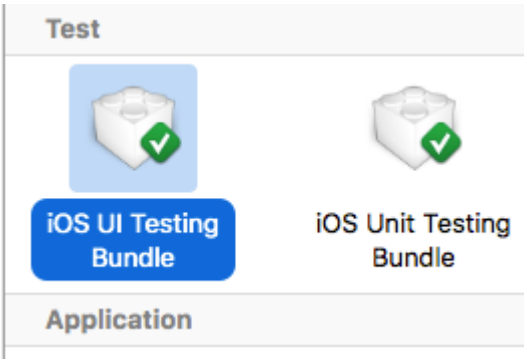


Après avoir créé le projet

Si vous n'avez pas vérifié la `UI target` lors de la création du projet, vous pouvez toujours ajouter une cible de test ultérieurement.

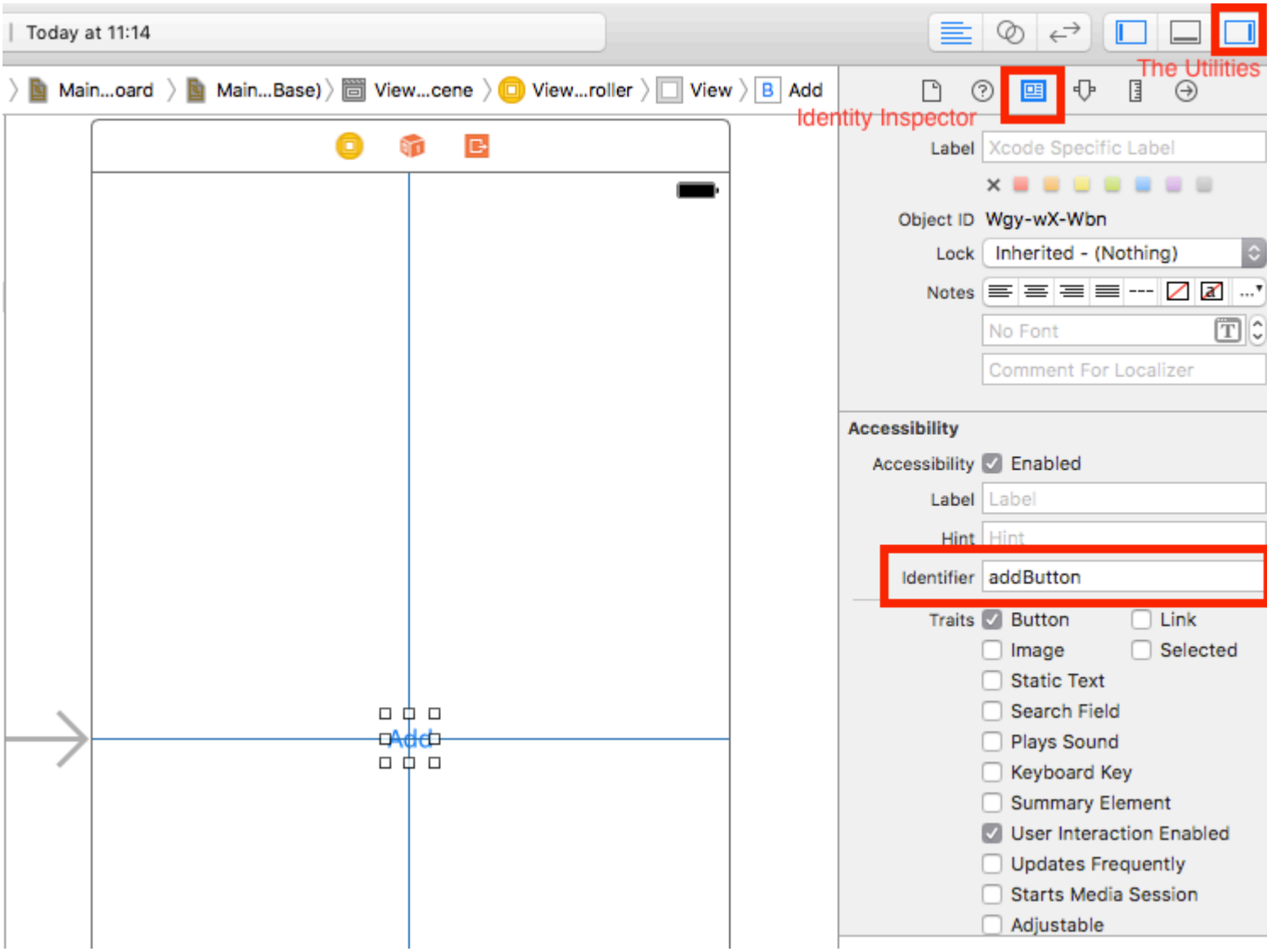
Setps:

- Alors que le projet est ouvert, allez dans `File` -> `New` -> `Target`
- Rechercher un ensemble de `iOS UI Testing Bundle`



Lorsque l'accessibilité est activée dans les utilitaires

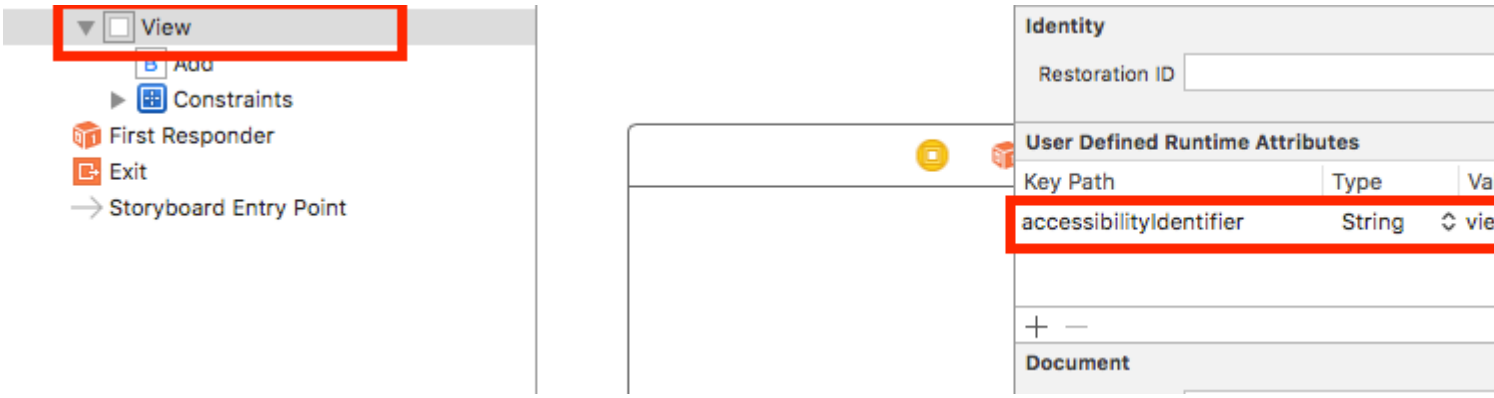
- Sélectionnez le `storyboard` .
- Développez `the Utilities`
- Sélectionner un `Identity Inspector`
- Sélectionnez votre élément sur le storyboard
- Ajouter un nouvel identifiant d'accessibilité (par exemple `addButton`)



Lorsque l'accessibilité est désactivée dans les utilitaires

- Sélectionnez le `storyboard` .
- Développez `the Utilities`
- Sélectionner un `Identity Inspector`
- Sélectionnez votre élément sur le storyboard
- Ajouter un attribut dans les attributs `User Defined Runtime Attributes`
- Pour le type de `Key Path` - `accessibilityIdentifier`

- Pour le `Type` - `String
- For `Value` - nouvel identifiant d'accessibilité pour votre élément (dans la `view` exemple)



Configuration dans le fichier 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()
    }
}
```

Dans `[ ]` ajoutez un identifiant d'accessibilité pour l'élément.

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"]
```

Éléments 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

- UITextField normal

```
let textField = app.textFields["textField"]
```

- mot de passe UITextField

```
let passwordTextField = app.secureTextFields["passwordTextField"]
```

UITextView

```
let textView = app.textViews["textView"]
```

UISwitch

```
let switch = app.switches["switch"]
```

Des alertes

```
let alert = app.alerts["About yourself"] // Title of presented alert
```

Désactiver les animations lors des tests de l'interface utilisateur

Dans un test, vous pouvez désactiver les animations en ajoutant dans `setUp` :

```
    app.launchEnvironment = ["animations": "0"]
```

Où `app` est l'instance de XCUIApplication.

Déjeuner et terminer l'application pendant l'exécution

Demande de déjeuner pour les tests

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

    let app = XCUIApplication()

    app.launch()
}
```

Application de terminaison

```
func testStacOverFlowApp() {

    app.terminate()
}
```

Faire pivoter les appareils

Le périphérique peut être `XCUIDevice.shared().orientation` en modifiant l' `orientation` dans

`XCUIDevice.shared().orientation`

```
XCUIDevice.shared().orientation = .landscapeLeft
XCUIDevice.shared().orientation = .portrait
```

Lire Test de l'interface utilisateur en ligne: https://riptutorial.com/fr/ios/topic/7526/test-de-l-interface-utilisateur

Chapitre 149: Texte UILabel souligné

Examples

Souligner un texte dans un UILabel en utilisant 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];
```

Souligner un texte dans UILabel en utilisant 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
```

Lire Texte UILabel souligné en ligne: https://riptutorial.com/fr/ios/topic/7219/texte-uilabel-souligne

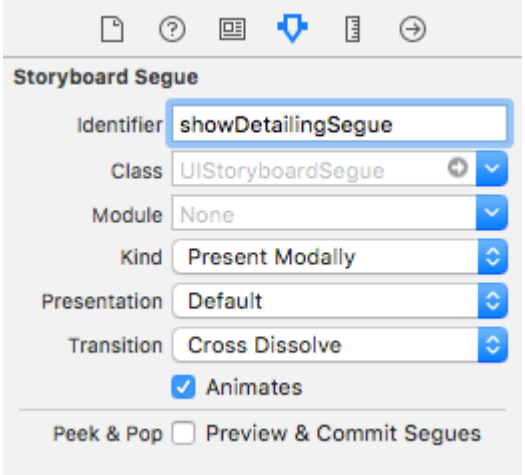
Chapitre 150: Transmission de données entre les contrôleurs de vue

Examples

Utilisation de Segues (transmission de données en avant)

Pour transmettre des données du contrôleur de vue actuel au nouveau contrôleur de vue suivant (pas à un contrôleur de vue précédent) à l'aide de segues, créez d'abord un lien avec un identifiant dans le storyboard correspondant. Remplacez la méthode `prepareForSegue` votre contrôleur d'affichage actuel. À l'intérieur de la méthode, vérifiez le segment que vous venez de créer grâce à son identifiant. Lancez le contrôleur de vue de destination et transmettez-lui les données en définissant les propriétés sur le contrôleur de la vue réduite.

Définition d'un identifiant pour un segue:



Les segues peuvent être exécutées par programme ou en utilisant les actions de bouton définies dans le storyboard par Ctrl + Glisser vers le contrôleur de vue de destination. En cas de besoin, vous pouvez appeler un segue par programme en utilisant l'identifiant de segue dans le contrôleur de vue:

Objectif c

```
- (void)showDetail {
    [self performSegueWithIdentifier:@"showDetailingSegue" sender:self];
}
```

Rapide

```
func showDetail() {
    self.performSegue(withIdentifier: "showDetailingSegue", sender: self)
}
```

Vous pouvez configurer la charge utile segue dans la version surchargée de la méthode `prepareForSegue` . Vous pouvez définir les propriétés requises avant le chargement du contrôleur de vue de destination.

Objectif c

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if([segue.identifier isEqualToString:@"showDetailingSegue"]){
        DetailViewController *controller = (DetailViewController
*)segue.destinationViewController;
        controller.isDetailingEnabled = YES;
    }
}
```

Rapide

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "showDetailingSegue" {
        let controller = segue.destinationViewController as! DetailViewController
        controller.isDetailingEnabled = true
    }
}
```

`DetailViewController` est le nom du second contrôleur de vue et `isDetailingEnabled` est une variable publique dans ce contrôleur de vue.

Pour développer ce modèle, vous pouvez traiter une méthode publique sur `DetailViewController` tant que pseudo-initialiseur, afin d'initialiser les variables requises. Cela va auto-documenter les variables qui doivent être définies sur `DetailViewController` sans avoir à lire son code source. C'est aussi un endroit pratique pour mettre des défauts.

Objectif c

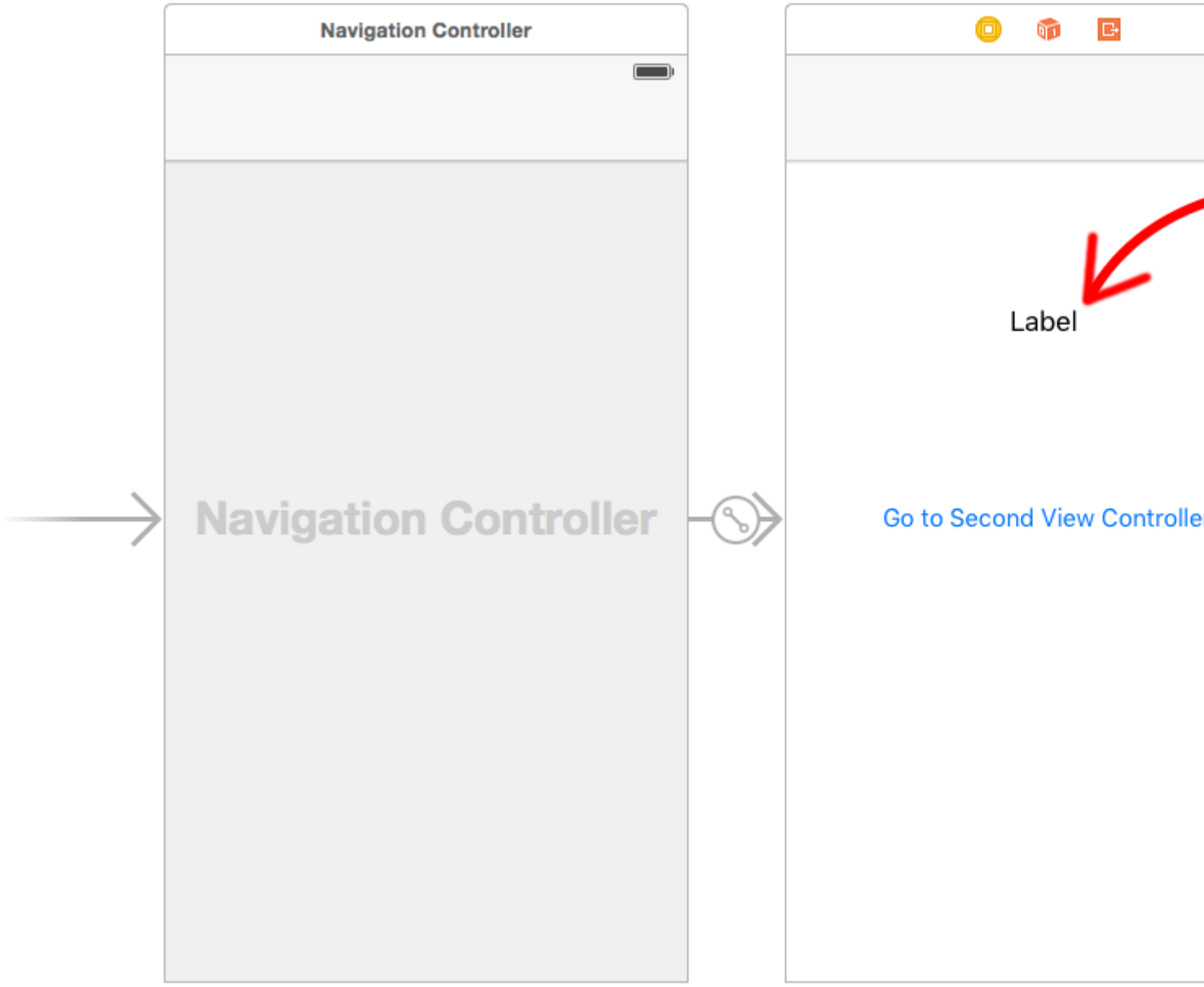
```
- (void)initVC:(BOOL *)isDetailingEnabled {
    self.isDetailingEnabled = isDetailingEnabled
}
```

Rapide

```
func initVC(isDetailingEnabled: Bool) {
    self.isDetailingEnabled = isDetailingEnabled
}
```

Utilisation du modèle de délégué (retour des données)

Pour renvoyer des données du contrôleur de vue actuel au contrôleur de vue précédent, vous pouvez utiliser le modèle de délégué.



Cet exemple suppose que vous avez effectué un segue dans le `showSecondViewController` interface et que vous définissez l'identificateur de segue sur `showSecondViewController` . Les points de vente et les actions doivent également être reliés aux noms dans le code suivant.

Premier contrôleur d'affichage

Le code du contrôleur First View est

Rapide

```
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)
    }
}
```

Objectif 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
```

Notez l'utilisation de notre protocole personnalisé `DataEnteredDelegate` .

Second View Controller et Protocole

Le code du second contrôleur de vue est

Rapide

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

Objectif 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
```

Notez que le `protocol` est en dehors de la classe View Controller.

Passer des données en arrière en utilisant le déroulement pour se

Contrairement à segue qui vous permet de transmettre des données "en avant" du contrôleur de vue actuel au contrôleur de vue de destination:

(VC1) **->** *(VC2)*

En utilisant "dérouler", vous pouvez faire le contraire, passer des données de la destination ou du contrôleur de vue actuel à son contrôleur de vue présentateur:

(VC1) **<-** *(VC2)*

REMARQUE : faites attention à ce que l'utilisation de dérouler vous permette de transmettre les données en premier, puis le contrôleur de vue actuel (VC2) sera libéré.

Voici comment procéder:

Tout d'abord, vous devrez ajouter la déclaration suivante au contrôleur de vue de présentation (VC1), qui est le contrôleur de vue que vous souhaitez transmettre aux données:

```
@IBAction func unwindToPresentingViewController(segue:UIStoryboardSegue)
```

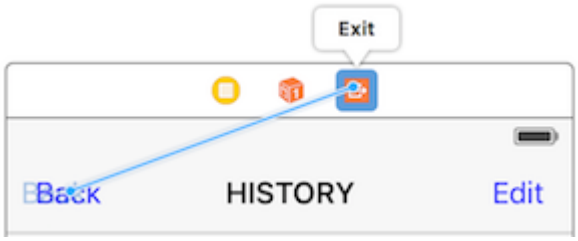
L'important est d'utiliser le préfixe `unwind` , cela "informe" Xcode qu'il s'agit d'une méthode de déroulage vous donnant la possibilité de l'utiliser également dans le storyboard.

Ensuite, vous devrez implémenter la méthode, elle ressemble presque à un segue réel:

```
@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
        }
```

Vous avez maintenant 2 options pour appeler les appels à la fin:

1. Vous pouvez « coder en dur » invoquer le: `self.performSegueWithIdentifier("YourCustomIdentifier", sender: self)` qui fera le dérouleur pour vous à chaque fois que vous voulez `performSegueWithIdentifier` .
2. Vous pouvez lier la méthode de déroulement en utilisant le `storyboard` à votre objet "Exit": ctrl + faites glisser le bouton que vous voulez appeler la méthode déroulante vers l'objet "Exit":



Relâchez et vous aurez la possibilité de choisir votre méthode de déroulement personnalisée:

Transmission de données à l'aide de fermetures (retour de données)

Au lieu d'utiliser le **modèle de délégué** , qui divise l'implémentation dans différentes parties de la classe `UIViewController` , vous pouvez même utiliser des `closures` pour renvoyer des données en avant et en arrière. En supposant que vous utilisez le `UIStoryboardSegue` , dans la méthode `prepareForSegue` , vous pouvez facilement configurer le nouveau contrôleur en une seule étape

```
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)
        }
    }
}
```

Ceci est un exemple d'utilisation et il est préférable d'utiliser Swift, la syntaxe du bloc Objective-C n'est pas si facile de rendre le code plus lisible

Utiliser la fermeture de rappel (blocage) pour transmettre des données

ce sujet est un problème classique dans le développement iOS, et sa solution est diverse comme d'autres exemples déjà présentés. Dans cet exemple, je vais vous montrer une autre utilisation quotidienne commune: transmettre des données à l'aide de la `closure` en adaptant l'exemple de `delegate pattern` sur cette page en `closure` rappel!

Une chose que cette méthode est supérieure à `delegate pattern` est au lieu de diviser le code de configuration en deux endroits différents (regardez l'exemple de délégué sur cette page, `prepareForSegue` , `userDidEnterInformation`) plutôt que de les rassembler (seulement dans `prepareForSegue` , je le montrerai)

Démarrer à partir de Second View Controller

nous devons comprendre comment utiliser le callback, puis pouvons-nous l'écrire, c'est pourquoi nous commençons à partir du second contrôleur de vue car nous utilisons callback: lorsque nous avons la nouvelle entrée de texte, nous appelons notre callback, en **utilisant le paramètre callback** comme support pour renvoyer les données au premier ViewController, notez que j'ai dit en utilisant le paramètre callback, c'est très important, les novices (comme je l'étais) oublient toujours cela et ne savent pas par où commencer pour écrire la fermeture de rappel correctement

donc dans ce cas, nous savons que notre callback ne prend qu'un seul paramètre: text et que son type est `String` , déclarons-le et en faisons la propriété puisque nous avons besoin de remplir notre premier contrôleur de vue

Je ne fais que commenter la partie `delegate` et la conserver pour la comparer

```
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)
    }
}
```

Terminer le contrôleur de la première vue

tout ce que vous avez à faire est de passer la fermeture de rappel, et nous avons terminé, la fermeture fera le travail futur pour nous puisque nous l'avons déjà configuré dans le contrôleur de deuxième vue

regardez comment cela rend notre code plus court comparé au `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
    //}
}
```

et dans le dernier cas, peut-être quelqu'un d'entre vous sera-t-il confus en pensant que nous ne transmettons que les données (fermeture dans ce cas) d'une seule manière, considère-le comme un outil de communication? peut-être devriez-vous vraiment l'exécuter et le prouver vous-même, tout ce que je dirai, c'est le **paramètre** , c'est **le paramètre de la fermeture de rappel** qui renvoie les données!

En attribuant une propriété (transmission de données vers l'avant)

Vous pouvez transmettre des données directement en assignant la propriété du contrôleur de vue suivant avant de le pousser ou de le présenter.

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

Lire Transmission de données entre les contrôleurs de vue en ligne:
https://riptutorial.com/fr/ios/topic/434/transmission-de-donnees-entre-les-controleurs-de-vue

Chapitre 151: Transmission de données entre les contrôleurs de vue (avec MessageBox-Concept)

Introduction

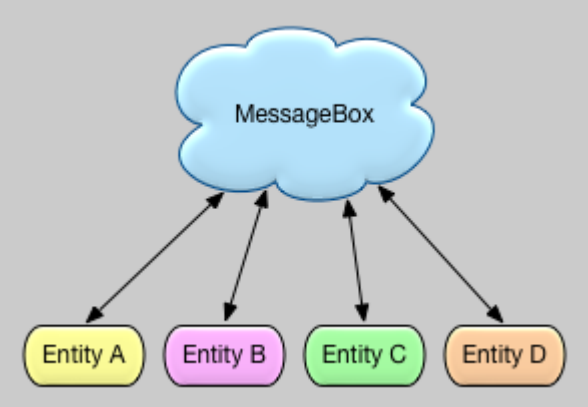
MessageBox est un concept simple pour découpler les entités.

Par exemple, l'entité A peut placer un message que l'entité B peut lire, le cas échéant.

Un contrôleur de vue souhaite parler à un autre contrôleur de vue, mais vous ne souhaitez pas créer de relation forte ou faible.

Examples

Exemple d'utilisation simple




```
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)
```

Lire Transmission de données entre les contrôleurs de vue (avec MessageBox-Concept) en ligne: https://riptutorial.com/fr/ios/topic/9118/transmission-de-donnees-entre-les-controleurs-de-vue--avec-messagebox-concept-

Chapitre 152: Tutoriel AirPrint sur iOS

Examples

Impression AirPrint Banner Text

Objectif c

Ajoutez le délégué et un formateur de `ViewController.h` fichier `ViewController.h`

```
@interface ViewController : UIViewController <UIPrintInteractionControllerDelegate> {
    UISimpleTextPrintFormatter *_textFormatter;
}
```

Dans le fichier `ViewController.m` , définissez les constantes suivantes

```
#define DefaultFontSize 48
#define PaddingFactor 0.1f
```

La fonction qui imprime le texte est la suivante: -

```
-(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 fonction de délégué qui configure la page d'impression: -

```
- (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;
}
```

Lire Tutoriel AirPrint sur iOS en ligne: https://riptutorial.com/fr/ios/topic/7395/tutoriel-airprint-sur-ios

Chapitre 153: Tweak de CydiaSubstrate

Introduction

Apprenez à créer des réglages de substrat cydia pour les iPhones jailbreakés.

Ces ajustements vous permettront de modifier le comportement du système d'exploitation pour agir comme vous le souhaitez.

Remarques

Installer Theos

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

Examples

Créer un nouveau tweak en utilisant Theos

Utilisez nic pour créer un nouveau projet

Entrez cette commande dans votre terminal

```
$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):
```

Choisissez le modèle `[11.] iphone/tweak`

Remplissez les détails et vous obtiendrez les fichiers suivants créés:

```
-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
```

Remplacez la méthode de sauvegarde des captures d'écran iOS

Ouvrez le fichier `Tweak.xm` en utilisant votre éditeur de code préféré.

Accrocher à une certaine méthode du système d'exploitation.

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

Notez que vous pouvez choisir si la fonction d'origine doit être appelée ou non, par exemple:

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

remplacera la fonction sans appeler l'original, les captures d'écran du boîtier n'étant pas enregistrées.

Lire Tweak de CydiaSubstrate en ligne: https://riptutorial.com/fr/ios/topic/10533/tweak-de-cydiasubstrate

Chapitre 154: Type dynamique

Remarques

```
// Content size category constants
UIContentSizeCategoryExtraSmall
UIContentSizeCategorySmall
UIContentSizeCategoryMedium
UIContentSizeCategoryLarge
UIContentSizeCategoryExtraLarge
UIContentSizeCategoryExtraExtraLarge
UIContentSizeCategoryExtraExtraExtraLarge

// Accessibility sizes
UIContentSizeCategoryAccessibilityMedium
UIContentSizeCategoryAccessibilityLarge
UIContentSizeCategoryAccessibilityExtraLarge
UIContentSizeCategoryAccessibilityExtraExtraLarge
UIContentSizeCategoryAccessibilityExtraExtraExtraLarge
```

Examples

Obtenir la taille du contenu actuel

Rapide

```
UIApplication.sharedApplication().preferredContentSizeCategory
```

Objectif c

```
[UIApplication sharedApplication].preferredContentSizeCategory;
```

Cela renvoie une constante de catégorie de taille de contenu ou une constante de catégorie de taille de contenu d'accessibilité.

Notification de modification de la taille du texte

Vous pouvez vous inscrire pour recevoir des notifications lorsque la taille du texte de l'appareil est modifiée.

Rapide

```
NSNotificationCenter.defaultCenter().addObserver(self, selector: #selector(updateFont), name:
```

```
name:UIContentSizeCategoryDidChangeNotification, object: nil)
```

Objectif c

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(updateFont)
name:UIContentSizeCategoryDidChangeNotification object:nil];
```

L'objet `userInfo` notification contient la nouvelle taille sous `UIContentSizeCategoryNewValueKey` .

Correspondance de la taille de la police de type dynamique dans WKWebView

WKWebView redimensionne les polices sur le contenu Web de sorte qu'une page Web pleine grandeur s'adaptera au facteur de forme de l'appareil. Si vous souhaitez que la taille du texte Web dans Portrait et Landscape soit identique à la taille de lecture préférée de l'utilisateur, vous devez la définir explicitement.

Rapide

```
// 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
}
```

Gestion de la modification de taille de texte préférée sans notifications sur iOS 10

`UILabel` , `UITextField` et `UITextView` ont une nouvelle propriété à partir d'iOS 10 pour redimensionner automatiquement leur police lorsqu'un utilisateur modifie sa taille de lecture préférée nommée `adjustsFontForContentSizeCategory` .

Rapide

```
@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.
}
```

Lire Type dynamique en ligne: https://riptutorial.com/fr/ios/topic/4466/type-dynamique

Chapitre 155: UIAapparence

Examples

Définir l'apparence de toutes les instances de la classe

Pour personnaliser l'apparence de toutes les instances d'une classe, accédez au proxy d'apparence de la classe souhaitée. Par exemple:

Définir la couleur de teinte UIButton

Rapide:

```
UIButton.appearance().tintColor = UIColor.greenColor()
```

Objectif c:

```
[UIButton appearance].tintColor = [UIColor greenColor];
```

Définir la couleur de fond UIButton

Rapide:

```
UIButton.appearance().backgroundColor = UIColor.blueColor()
```

Objectif c:

```
[UIButton appearance].backgroundColor = [UIColor blueColor];
```

Définir la couleur du texte UILabel

Rapide:

```
UILabel.appearance().textColor = UIColor.redColor()
```

Objectif c:

```
[UILabel appearance].textColor = [UIColor redColor];
```

Définir la couleur d'arrière-plan de UILabel

Rapide:

```
UILabel.appearance().backgroundColor = UIColor.greenColor()
```

Objectif c:

```
[UILabel appearance].backgroundColor = [UIColor greenColor];
```

Définir la couleur de teinte UINavigationBar

Rapide:

```
UINavigationBar.appearance().tintColor = UIColor.cyanColor()
```

Objectif c:

```
[UINavigationBar appearance].tintColor = [UIColor cyanColor];
```

Définir la couleur d'arrière-plan d'UINavigationBar

Rapide:

```
UINavigationBar.appearance().backgroundColor = UIColor.redColor()
```

Objectif c:

```
[UINavigationBar appearance].backgroundColor = [UIColor redColor];
```

Apparence pour la classe lorsqu'elle est contenue dans une classe de conteneur

Utilisez `appearanceWhenContainedInInstancesOfClasses:` pour personnaliser l'apparence d'une instance lorsqu'elle est contenue dans une instance de classe de conteneur. Par exemple la personnalisation des `UILabel` de `textColor` et `backgroundColor` au sein `ViewController` classe ressemblera à ceci:

Définir la couleur du texte UILabel

Rapide:

```
UILabel.appearanceWhenContainedInInstancesOfClasses([ViewController.self]).textColor =
UIColor.whiteColor()
```

Objectif c:

```
[UILabel appearanceWhenContainedInInstancesOfClasses:@[[ViewController class]]].textColor =
[UIColor whiteColor];
```

Définir la couleur d'arrière-plan de UILabel

Rapide:

```
UILabel.appearanceWhenContainedInInstancesOfClasses([ViewController.self]).backgroundColor =
UIColor.blueColor()
```

Objectif c:

```
[UILabel appearanceWhenContainedInInstancesOfClasses:@[[ViewController
class]]].backgroundColor = [UIColor blueColor];
```

Lire UIAapparence en ligne: https://riptutorial.com/fr/ios/topic/3422/uiaapparence

Chapitre 156: UIActivityViewController

Paramètres

Le nom du paramètre	La description
activitéItems	Contient un tableau d'objet pour effectuer l'activité. Ce tableau ne doit pas être nul et doit contenir au moins un objet.
applicationActivités	Tableau d'objets UIActivity représentant les services personnalisés pris en charge par votre application. Ce paramètre peut être nul.

Examples

Initialisation du contrôleur de vue d'activité

Objectif 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];
```

Rapide

```
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)
```

Lire UIActivityViewController en ligne:
https://riptutorial.com/fr/ios/topic/2889/uiactivityviewcontroller

Chapitre 157: UIAlertController

Remarques

> Un objet `UIAlertController` affiche un message d'alerte à l'utilisateur. Cette classe remplace les classes `UIActionSheet` et `UIAlertView` pour l'affichage des alertes. Après avoir configuré le contrôleur d'alerte avec les actions et le style souhaités, présentez-le à l'aide de la méthode `presentViewController:animated:completion:` method.

De la documentation d'Apple

UIAlertController dans Swift

Examples

AlertViews avec UIAlertController

`UIAlertView` et `UIActionSheet` sont `UIActionSheet` dans `iOS 8` et `UIActionSheet` ultérieures. Ainsi, Apple a introduit un nouveau contrôleur pour `AlertView` et `ActionSheet` appelé `UIAlertController` , modifiant le `UIAlertController` `preferredStyle` , vous pouvez basculer entre `AlertView` et `ActionSheet` . Il n'y a pas de méthode de délégué car tous les événements de bouton sont traités dans leurs blocs.

Simple AlertView

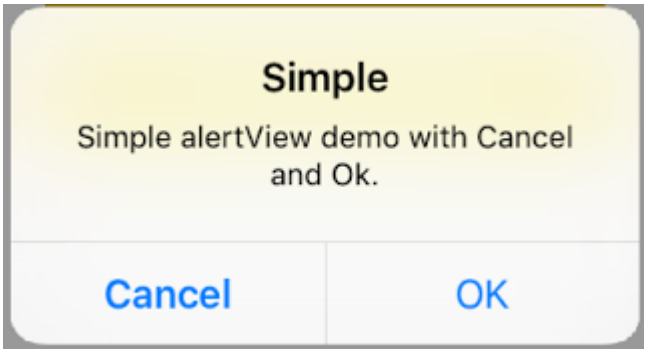
Rapide:

```
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)
```

Objectif 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];
```



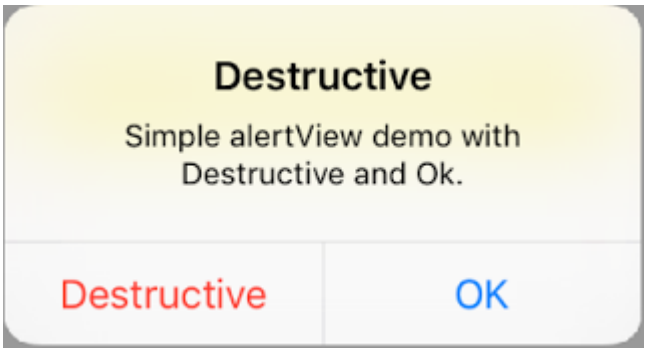
Alerte destructiveVoir

Rapide:

```
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)
```

Objectif 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];
```



Pop up temporaire en forme de pain grillé

Bon pour les notifications rapides qui ne nécessitent pas d'interaction.

Rapide

```
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)
     }
}
```

Ajouter un champ de texte dans UIAlertController comme une boîte d'invite

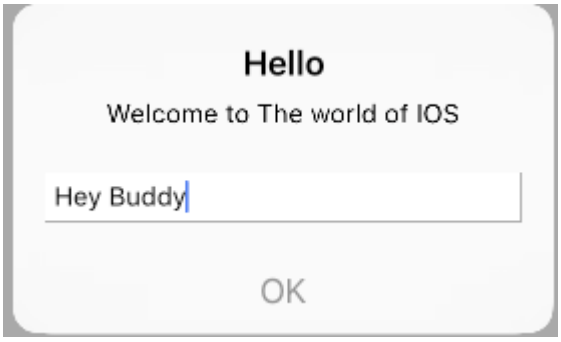
Rapide

```
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)
```

Objectif 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];
```



Feuilles d'action avec UIAlertController

Avec `UIAlertController` , des feuilles d'action telles que la `UIActionSheet` sont créées avec la même API que celle utilisée pour AlertViews.

Feuille d'action simple avec deux boutons

Rapide

```
let alertController = UIAlertController(title: "Demo", message: "A demo with two buttons",
preferredStyle: UIAlertControllerStyle.actionSheet)
```

Objectif c

```
UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"Demo"
message:@"A demo with two buttons" preferredStyle:UIAlertControllerStyleActionSheet];
```

Créez les boutons "Annuler" et "OK"

Rapide

```
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
}
```

Objectif 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
    }];
```

Et ajoutez-les à la fiche d'action:

Rapide

```
alertController.addAction(cancelAction)
alertController.addAction(okAction)
```

Objectif c

```
[alertController addAction:cancelAction];
[alertController addAction:okAction];
```

Présentez maintenant le `UIAlertController` :

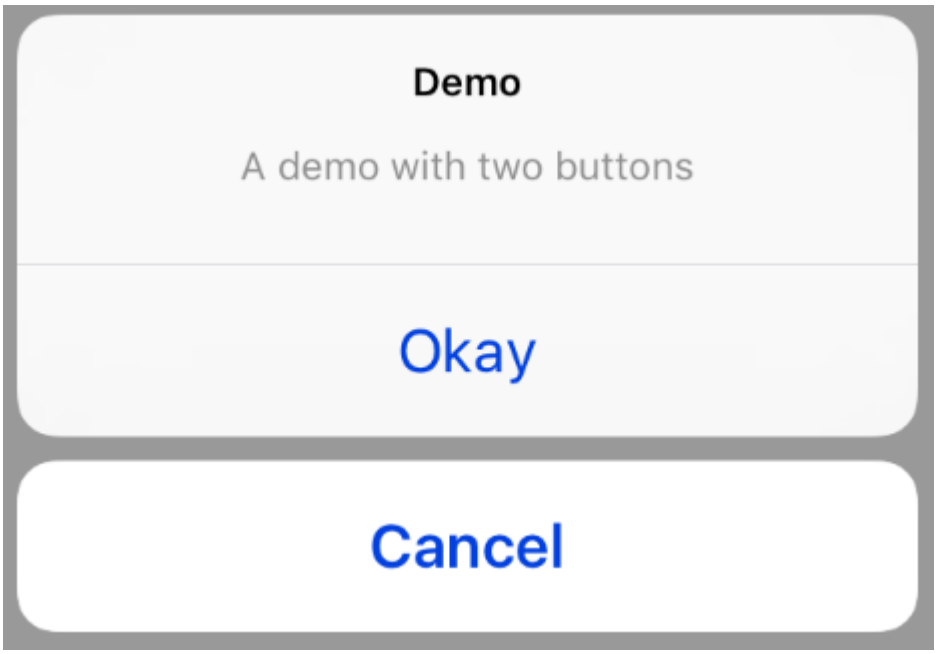
Rapide

```
self.present(alertController, animated: true, completion: nil)
```

Objectif c

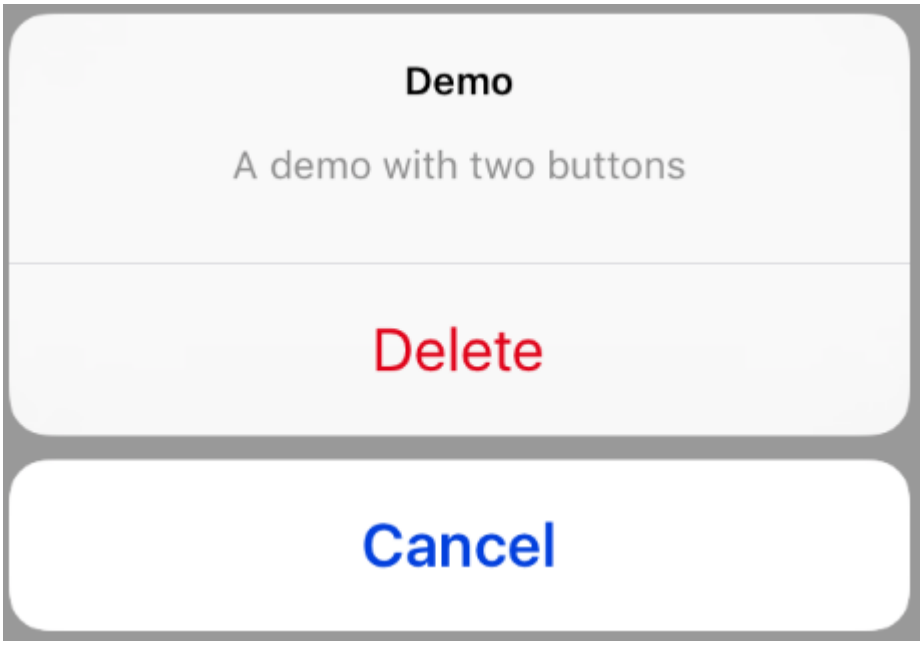
```
[self presentViewController:alertController animated: YES completion: nil];
```

Cela devrait être le résultat:



Feuille d'action avec bouton destructeur

En utilisant le `UIAlertActionStyle .destructive` pour `UIAlertAction` , vous créez un bouton avec une couleur rouge.



Pour cet exemple, `okAction` ci-dessus a été remplacé par cette `UIAlertAction` :

Rapide

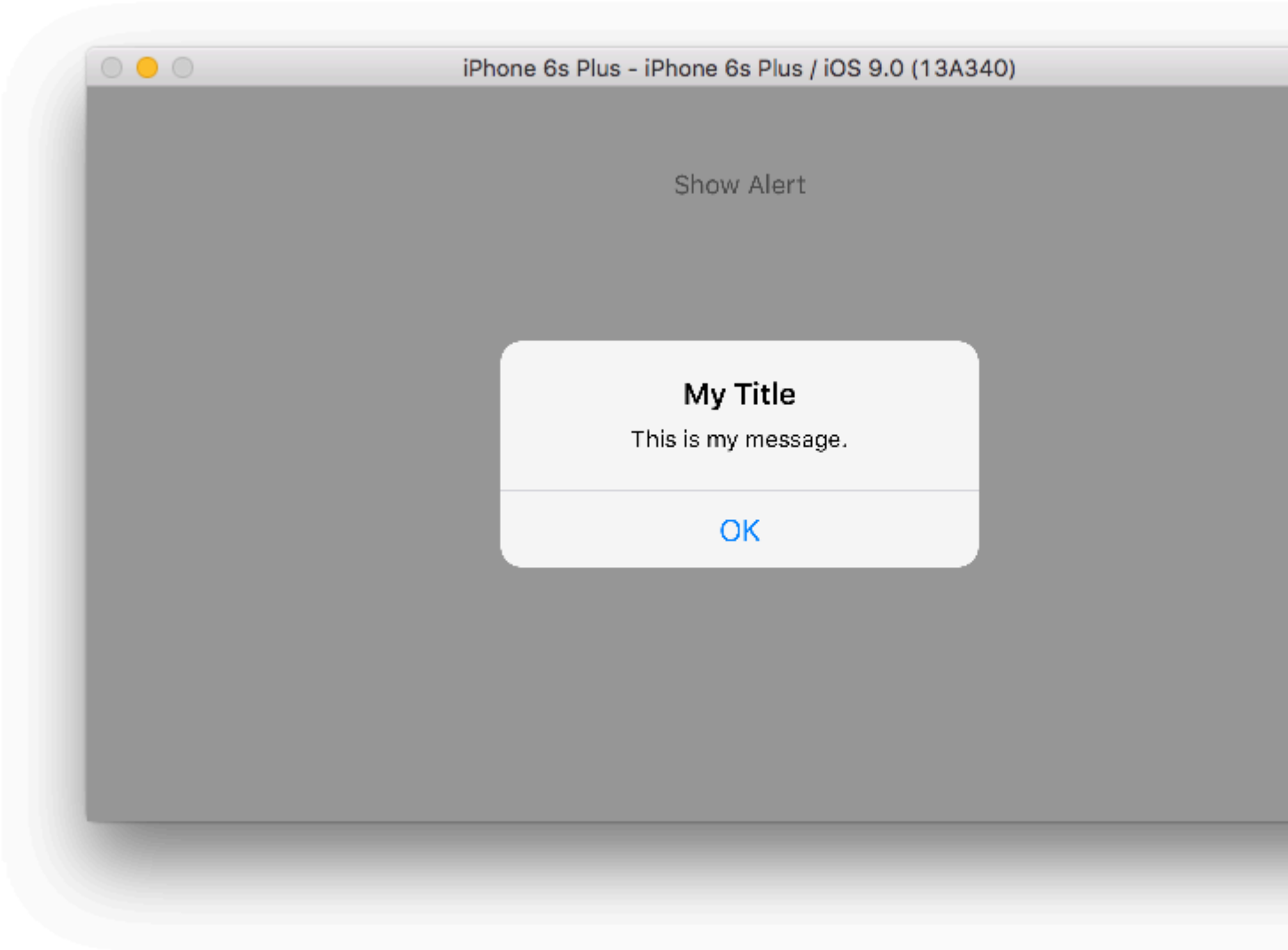
```
let destructiveAction = UIAlertAction(title: "Delete", style: .destructive) { (result :
UIAlertAction) -> Void in
    //action when pressed button
}
```

Objectif c

```
UIAlertAction * destructiveAction = [UIAlertAction actionWithTitle:@"Delete"
style:UIAlertActionStyleDestructive handler:^(UIAlertAction * action) {
            //action when pressed button
        }];
```

Affichage et traitement des alertes

Un bouton

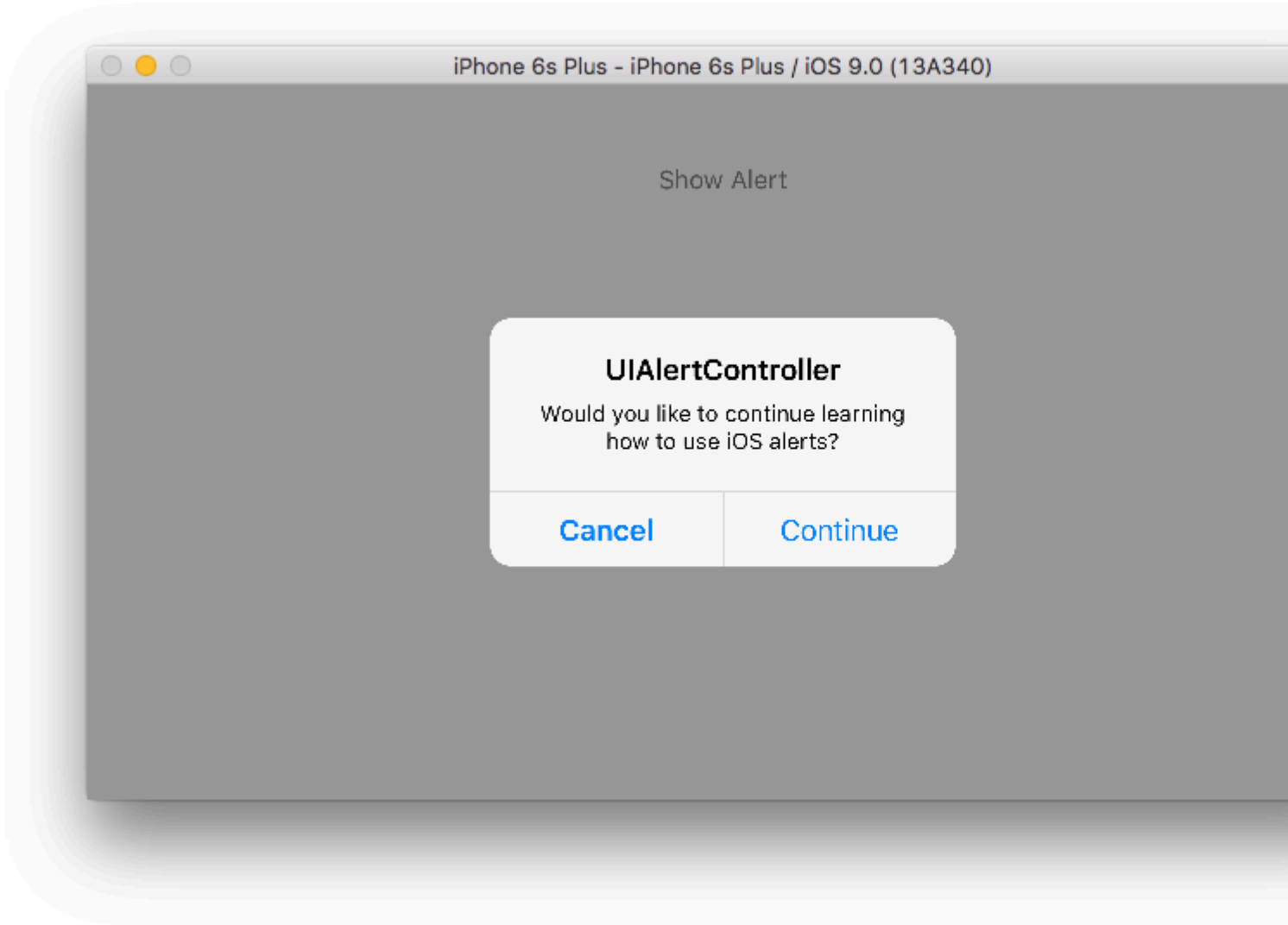


Rapide

```
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)
    }
}
```

Deux boutons

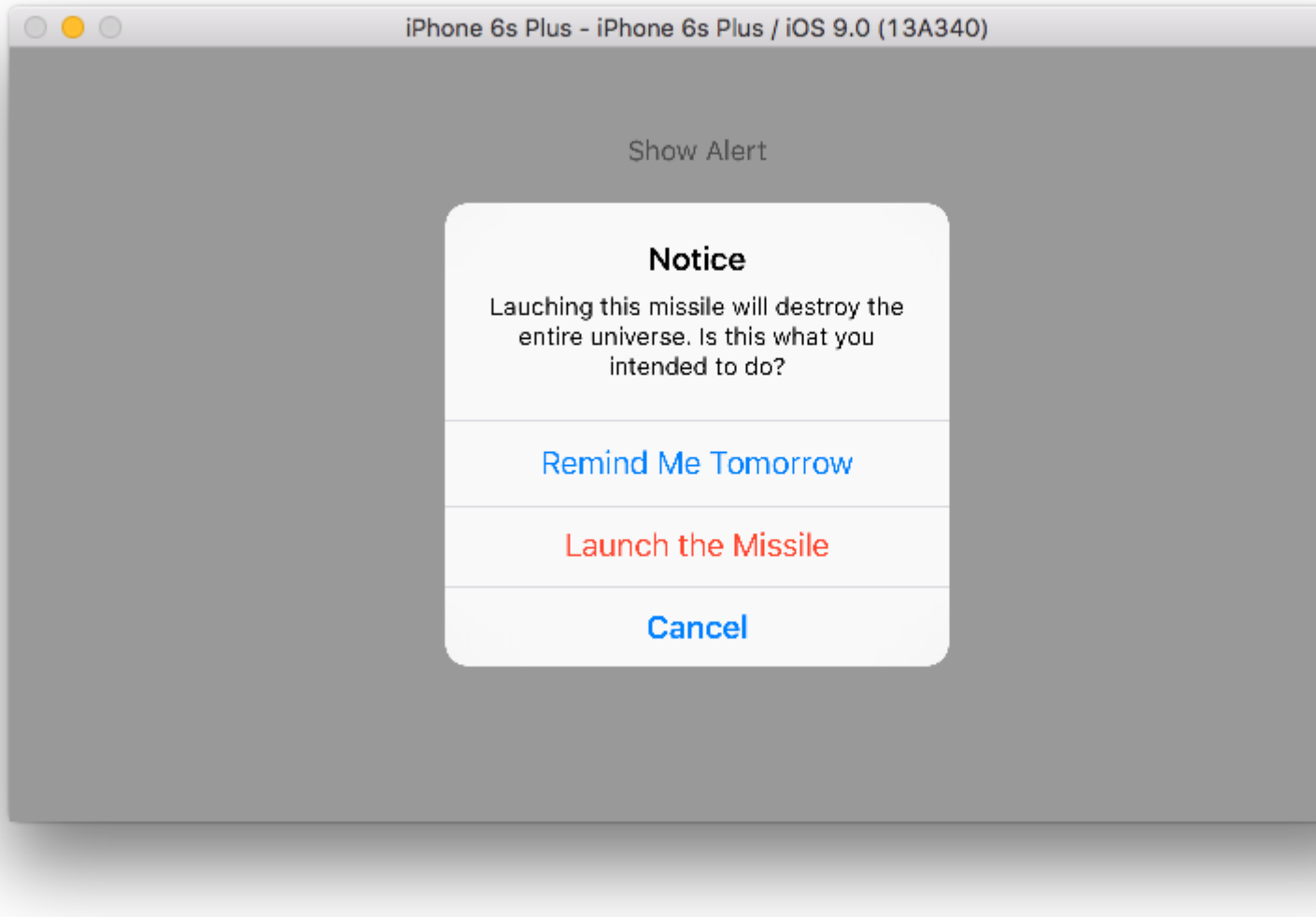


Rapide

```
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)
    }
}
```

Trois boutons



Rapide

```
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)
    }
}
```

Manipulation des boutons

Le `handler` était `nil` dans les exemples ci-dessus. Vous pouvez remplacer `nil` par une fermeture pour faire quelque chose lorsque l'utilisateur appuie sur un bouton, comme dans l'exemple ci-dessous:

Rapide

```
alert.addAction(UIAlertAction(title: "Launch the Missile", style:
UIAlertActionStyle.Destructive, handler: { action in

    // do something like...
    self.launchMissile()

}))
```

Remarques

- Plusieurs boutons n'ont pas nécessairement besoin d'utiliser différents types `UIAlertActionStyle` . Ils pourraient tous être `.Default` .
- Pour plus de trois boutons, envisagez d'utiliser une feuille d'action. La configuration est très similaire. Voici un exemple.

Mettre en évidence un bouton d'action

Le contrôleur d'alertes possède une propriété qui permet de mettre l'accent sur une action ajoutée dans le contrôleur d'alertes. Cette propriété peut être utilisée pour mettre en évidence une action particulière à l'attention de l'utilisateur. Pour l'objectif C;

```
@property(nonatomic, strong) UIAlertAction *preferredAction
```

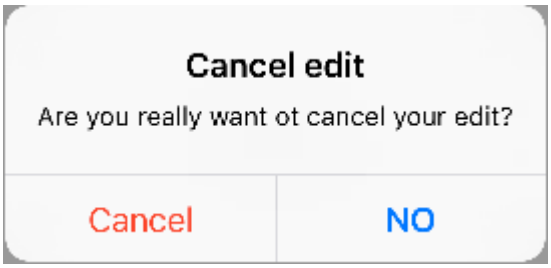
Une action **déjà ajoutée dans le contrôleur d'alertes** peut être affectée à cette propriété. Le contrôleur d'alertes mettra en évidence cette action.

Cette propriété ne peut être utilisée qu'avec UIAlertControllerStyleAlert.

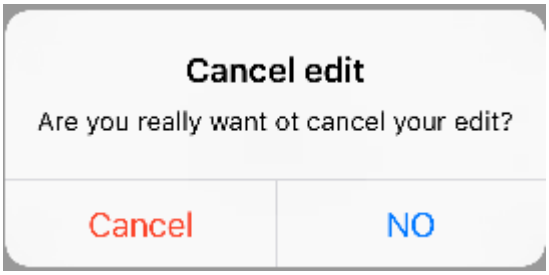
L'exemple suivant montre comment l'utiliser.

```
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 avec le **jeu d'action préféré** . Le bouton **NO** est en surbrillance.



Alert Controller avec l' **action préférée non définie** . Le bouton **NO** n'est pas en surbrillance.



Lire UIAlertController en ligne: https://riptutorial.com/fr/ios/topic/874/uialertcontroller

Chapitre 158: UIBarButtonItem

Paramètres

Paramètre	La description
Titre	Le titre UIBarButtonItem
style	Le style de l'UIBarButtonItem
cible	L'objet devant recevoir l'action UIBarButtonItem
action	Le sélecteur (méthode) à exécuter lorsque l'utilisateur appuie sur UIBarButtonItem

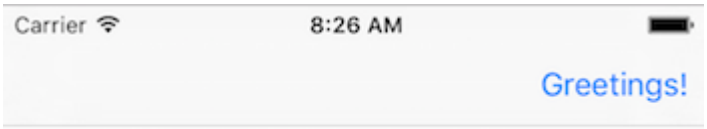
Remarques

La référence à `self.navigationItem` suppose que UIViewController est intégré à un UINavigationController.

Examples

Créer 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;
```

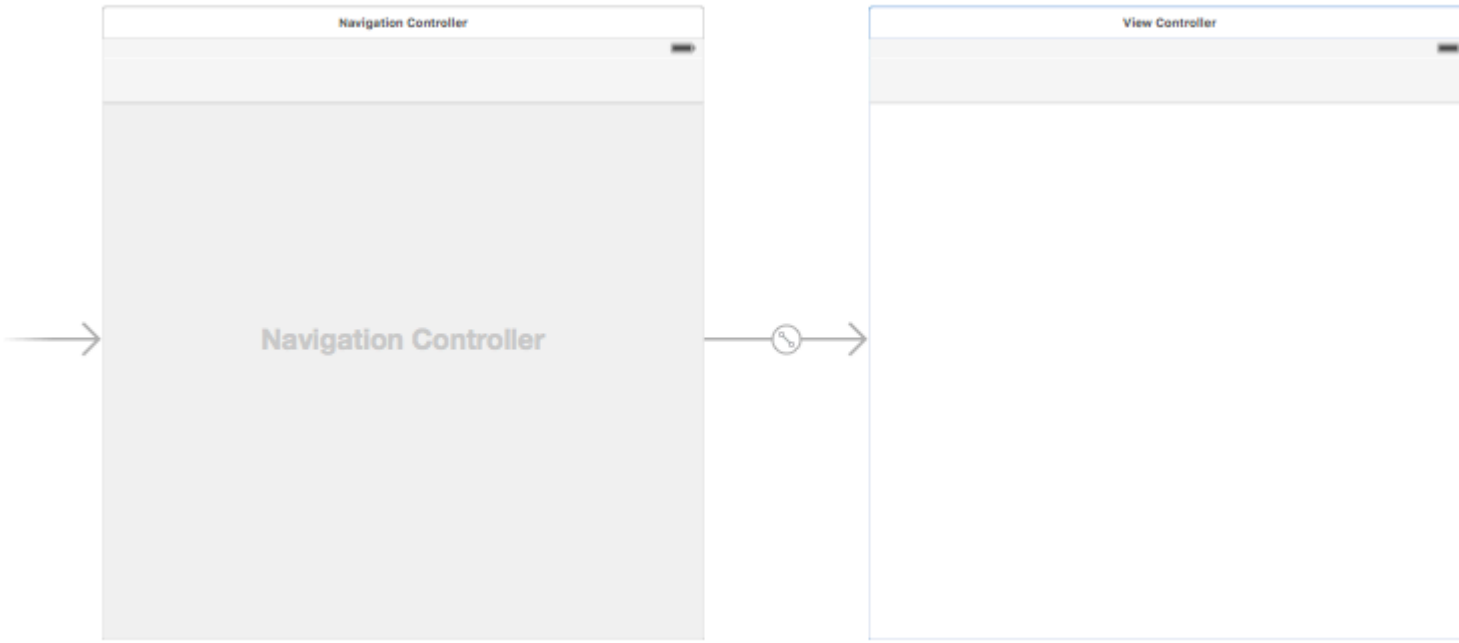


Création d'un UIBarButtonItem dans Interface Builder

L'exemple ci-dessous montre comment ajouter un bouton de barre de navigation (appelé `UIBarButtonItem`) dans Interface Builder.

Ajouter un contrôleur de navigation à votre storyboard

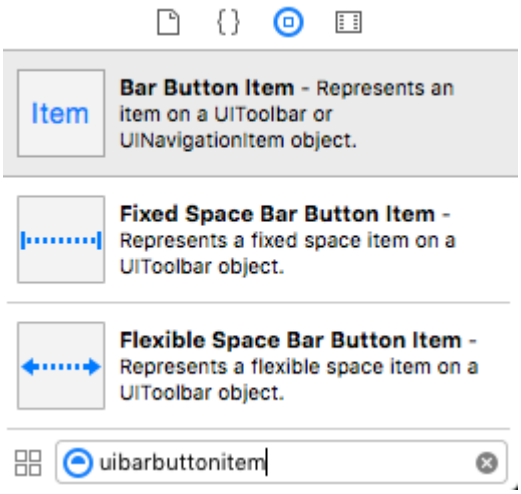
Sélectionnez votre View Controller, puis dans le menu Xcode, choisissez **Editor> Embed In> Navigation Controller** .



Vous pouvez également ajouter un `UINavigationBar` partir de la bibliothèque d'objets.

Ajouter un élément de bouton de barre

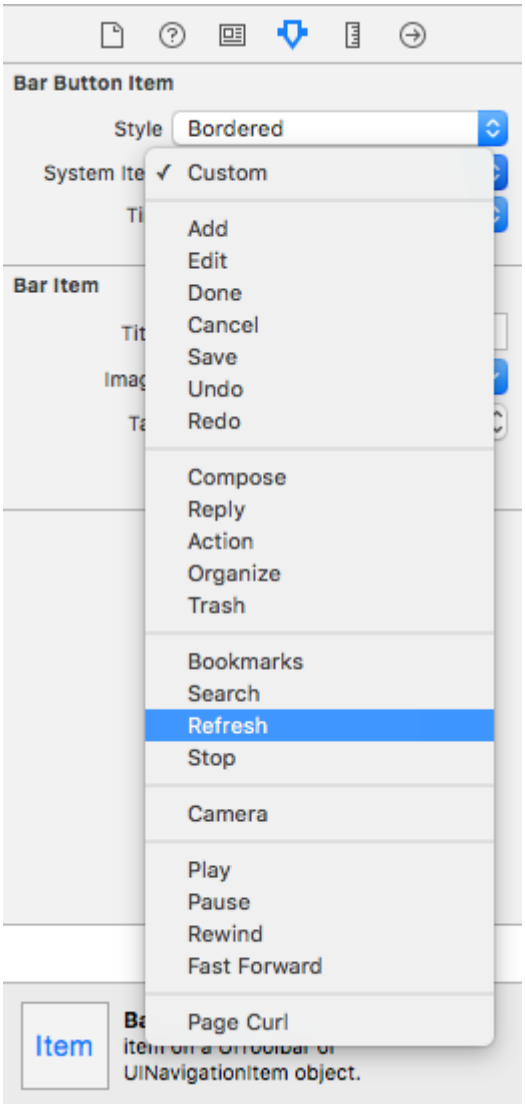
Faites glisser un `UIBarButtonItem` de la bibliothèque d'objets vers la barre de navigation supérieure.



Ça devrait ressembler à ça:

Définir les attributs

Vous pouvez double-cliquer sur «Item» pour changer le texte en quelque chose comme «Actualiser», mais vous pouvez utiliser une icône réelle pour *Actualiser* . Sélectionnez simplement l'inspecteur d'attributs pour `UIBarButtonItem` et, pour l' **élément du système, sélectionnez Actualiser** .



Cela vous donnera l'icône de rafraîchissement par défaut.

Ajouter une action IB

Contrôlez le glissement depuis `UIBarButtonItem` vers View Controller pour ajouter un `@IBAction` .

```
class ViewController: UIViewController {

    @IBAction func refreshBarButtonItemTap(sender: UIBarButtonItem) {

        print("How refreshing!")
    }

}
```

C'est tout.

Remarques

- Cet exemple provient à l'origine de cette réponse Stack Overflow .

Bar Button Item Image originale sans couleur de teinte

À condition que `barButtonItem` ait une propriété d'image non nulle (par exemple, définie dans Interface Builder).

Objectif c

```
   barButtonItem.image = [barButtonItem.image
imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
```

Lire UIBarButtonItem en ligne: https://riptutorial.com/fr/ios/topic/1543/uibarbuttonitem

Chapitre 159: UIBezierPath

Examples

Comment appliquer un rayon de coin à des rectangles dessinés par UIBezierPath

Rayon d'angle pour les 4 arêtes:

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) cornerRadius: 11];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Rayon d'angle pour le bord supérieur gauche:

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopLeft cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Rayon d'angle pour le bord supérieur droit:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopRight cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

rayon d'angle pour le bord inférieur gauche:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomLeft cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

rayon d'angle pour le bord inférieur droit:

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomRight cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

rayon d'angle pour les bords inférieurs:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomLeft |
UIRectCornerBottomRight cornerRadii: CGSizeMake(11, 11)];
```

```
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

rayon d'angle pour les bords supérieurs:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopLeft | UIRectCornerTopRight
cornerRadii: CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Comment créer une forme simple en utilisant UIBezierPath

Pour un simple cercle:

```
UIBezierPath* ovalPath = [UIBezierPath bezierPathWithOvalInRect: CGRectMake(0,0,50,50)];
[UIColor.grayColor setFill];
[ovalPath fill];
```

Rapide:

```
let ovalPath = UIBezierPath(ovalInRect: CGRect(x: 0, y: 0, width: 50, height: 50))
UIColor.grayColor().setFill()
ovalPath.fill()
```

Pour un rectangle simple:

```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect: CGRectMake(0,0,50,50)];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Rapide:

```
let rectanglePath = UIBezierPath(rect: CGRect(x: 0, y: 0, width: 50, height: 50))
UIColor.grayColor().setFill()
rectanglePath.fill()
```

Pour une ligne simple:


```
UIBezierPath* bezierPath = [UIBezierPath bezierPath];
[bezierPath moveToPoint: CGPointMake(x1,y1)];
[bezierPath addLineToPoint: CGPointMake(x2,y2)];
[UIColor.blackColor setStroke];
bezierPath.lineWidth = 1;
[bezierPath stroke];
```

Rapide:

```
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()
```

Pour un demi-cercle:

```
 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];
```

Rapide:

```
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()
```

Pour un simple triangle:

```
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];
```

Rapide:

```
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 + Mise en forme automatique

Pour que le chemin de bezier soit redimensionné en fonction du cadre de la vue, remplacez le

tracé de l'image que vous tracez le chemin de la bezier:

```
- (void)drawRect:(CGRect)frame
{
    UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect:
CGRectMake(CGRectGetMinX(frame), CGRectGetMinY(frame), CGRectGetWidth(frame),
CGRectGetHeight(frame))];
    [UIColor.grayColor setFill];
    [rectanglePath fill];
}
```

Comment appliquer des ombres à UIBezierPath

Considérons un simple rectangle qui est dessiné par le chemin du bezier.

```
 UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect:
CGRectMake(x,y,width,height)];
 [UIColor.grayColor setFill];
 [rectanglePath fill];
```

Ombre de remplissage externe de 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);
```

Basic ombre de remplissage interne:

```
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);
```

Concevoir et dessiner un chemin de Bézier

Cet exemple montre le processus de conception de la forme à dessiner dans une vue. Une forme spécifique est utilisée mais les concepts que vous apprenez peuvent être appliqués à n'importe quelle forme.

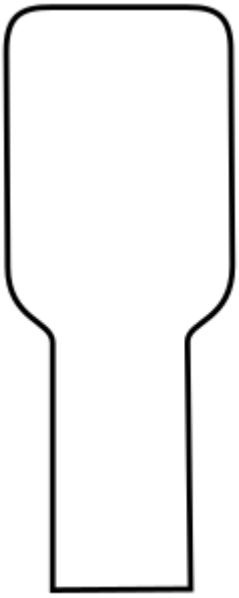
Comment dessiner un chemin Bézier dans une vue personnalisée

Ce sont les principales étapes:

1. Concevez le contour de la forme que vous voulez.
2. Diviser le tracé en segments de lignes, d'arcs et de courbes.
3. Construisez ce chemin par programmation.
4. Dessinez le chemin dans `drawRect` ou en utilisant un `CAShapeLayer` .

Contour de forme

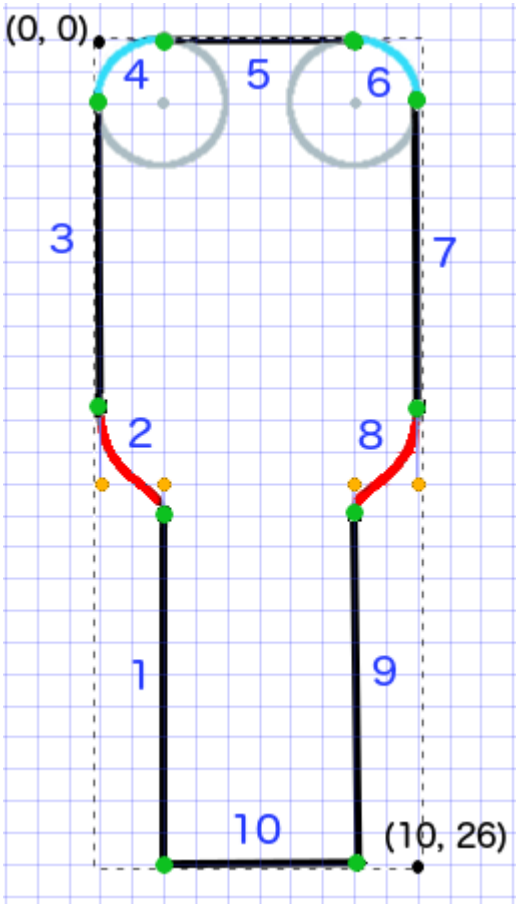
Vous pourriez faire n'importe quoi, mais par exemple, j'ai choisi la forme ci-dessous. Cela pourrait être une touche popup sur un clavier.



Diviser le chemin en segments

Examinez votre conception de forme et décomposez-la en éléments simples de lignes (pour les lignes droites), en arcs (pour les cercles et les coins arrondis) et en courbes (pour toute autre chose).

Voici à quoi ressemblerait notre exemple:



- Les noirs sont des segments de ligne
- Le bleu clair est un segment d'arc
- Rouge sont des courbes
- Les points orange sont les points de contrôle des courbes
- Les points verts sont les points entre les segments de chemin
- Les lignes pointillées indiquent le rectangle englobant
- Les numéros en bleu foncé sont les segments dans l'ordre où ils seront ajoutés par programmation

Construire le chemin par programmation

Nous commencerons arbitrairement dans le coin inférieur gauche et travaillerons dans le sens des aiguilles d'une montre. Je vais utiliser la grille dans l'image pour obtenir les valeurs x et y pour les points. Je vais tout coder ici, mais bien sûr, vous ne le feriez pas dans un vrai projet.

Le processus de base est le suivant:

1. Créer un nouveau `UIBezierPath`
2. Choisissez un point de départ sur le chemin avec `moveToPoint`
3. Ajouter des segments au chemin

- ligne: `addLineToPoint`
- arc: `addArcWithCenter`
- courbe: `addCurveToPoint`

4. Ferme le chemin avec `closePath`

Voici le code pour faire le chemin dans l'image ci-dessus.

```
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
}
```

Remarque: Une partie du code ci-dessus peut être réduite en ajoutant une ligne et un arc dans une seule commande (l'arc ayant un point de départ implicite). Voir ici pour plus de détails.

Dessine le chemin

Nous pouvons dessiner le chemin dans une couche ou dans `drawRect` .

Méthode 1: dessiner un chemin dans une couche

Notre classe personnalisée ressemble à ceci. Nous ajoutons notre chemin Bézier à un nouveau `CAShapeLayer` lorsque la vue est initialisée.

```
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
    }
}
```

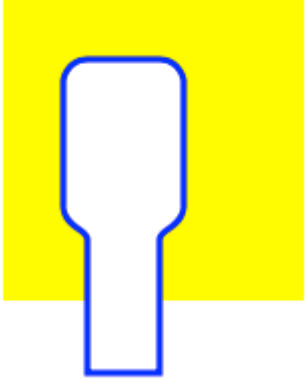
Et créer notre vue dans le View Controller comme ceci

```
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)

}
```

On a...

Hmm, c'est un peu petit parce que j'ai codé tous les nombres. Je peux augmenter la taille du chemin, comme ceci:

```
let path = createBezierPath()
let scale = CGAffineTransformMakeScale(2, 2)
path.applyTransform(scale)
shapeLayer.path = path.CGPath
```



Méthode 2: dessiner un chemin dans `drawRect`

Utiliser `drawRect` est plus lent que dessiner sur le calque, donc ce n'est pas la méthode recommandée si vous n'en avez pas besoin.

Voici le code révisé pour notre vue personnalisée:

```
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
    }
}
```

ce qui nous donne le même résultat ...

Une étude plus approfondie

D'excellents articles pour comprendre les chemins de Bézier.

- Penser comme un chemin de Bézier (Tout ce que j'ai lu de cet auteur est bon et l'inspiration pour mon exemple ci-dessus vient de là)

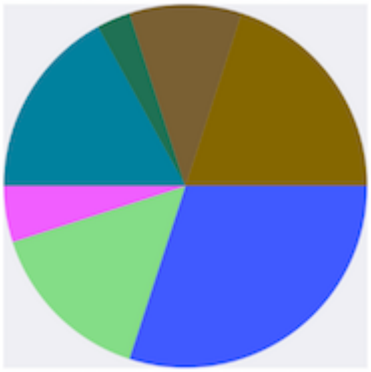
- Coding Math: Episode 19 - Courbes de Bézier (illustrations visuelles amusantes)
- Courbes de Bézier (comment elles sont utilisées dans les applications graphiques)
- Courbes de Bézier (bonne description de la manière dont les formules mathématiques sont dérivées)

Remarques

- Cet exemple provient à l'origine de cette réponse Stack Overflow .
- Dans vos projets réels, vous ne devriez probablement pas utiliser de numéros codés en dur, mais plutôt obtenir les tailles à partir des limites de votre vue.

vue de tarte et vue de colonne avec UIBezierPath

- vue de tarte

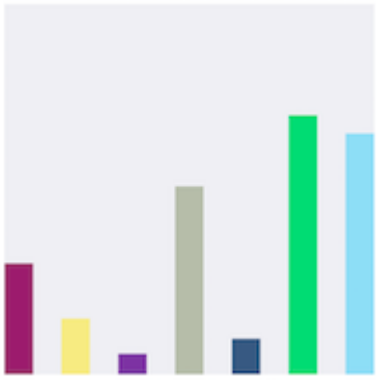


```
- (void)drawRect:(CGRect)rect {

    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
    }
  }
```

- vue colonne



```
- (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()
    }
  }
```

Lire UIBezierPath en ligne: https://riptutorial.com/fr/ios/topic/3186/uibezierpath

Chapitre 160: UIButton

Introduction

UIButton : UIControl intercepte les événements tactiles et envoie un message d'action à un objet cible lorsqu'il est tapé. Vous pouvez définir le titre, l'image et les autres propriétés d'apparence d'un bouton. De plus, vous pouvez spécifier une apparence différente pour chaque état du bouton.

Remarques

Types de boutons

Le type d'un bouton définit son apparence et son comportement de base. Après avoir créé un bouton, vous ne pouvez pas changer son type. Les types de boutons les plus couramment utilisés sont les types Personnalisé et Système, mais utilisez les autres types le cas échéant.

- 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.
```

- UIButtonTypeContactAjouter

```
A contact add button.
```

Lors de la création d'un bouton personnalisé (c'est-à-dire un bouton avec le type personnalisé), le cadre du bouton est défini sur (0, 0, 0, 0) au départ. Avant d'ajouter le bouton à votre interface, vous devez mettre à jour le cadre à une valeur plus appropriée.

Examples

Créer un UIButton

Les UIBtons peuvent être initialisés dans un cadre:

Rapide

```
let button = UIButton(frame: CGRect(x: x, y: y, width: width, height: height)
```

Objectif c

```
UIButton *button = [[UIButton alloc] initWithFrame:CGRectMake(x, y, width, height)];
```

Un type spécifique d'UIButton peut être créé comme ceci:

Rapide

```
let button = UIButton(type: .Custom)
```

Objectif c

```
UIButton *button = [UIButton buttonWithType:UIButtonTypeCustom];
```

où `type` est un `UIButtonType` :

```
enum UIButtonType : Int {
    case Custom
    case System
    case DetailDisclosure
    case InfoLight
    case InfoDark
    case ContactAdd
    static var RoundedRect: UIButtonType { get }
}
```

Définir le titre

Rapide

```
button.setTitle(titleString, forState: controlState)
```

Objectif c

```
[button setTitle:(NSString *) forState:(UIControlState)];
```

Pour définir le titre par défaut sur "Hello, World!"

Rapide

```
button.setTitle("Hello, World!", forState: .normal)
```

Objectif c

```
[button setTitle:@"Hello, World!" forControlState:UIControlStateNormal];
```

Définir la couleur du titre

```
//Swift
button.setTitleColor(color, forControlState: controlState)

//Objective-C
[button setTitleColor:(nullable UIColor *) forState:(UIControlState)];
```

Pour définir la couleur du titre en bleu

```
//Swift
button.setTitleColor(.blue, for: .normal)

//Objective-C
[button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal]
```

Alignement horizontal des contenus

Rapide

```
//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
```

Objectif 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;
```

Obtenir l'étiquette du titre

L’étiquette de titre sous-jacente, s’il en existe, peut être récupérée à l’aide de

Rapide

```
var label: UILabel? = button.titleLabel
```

Objectif c

```
UILabel *label = button.titleLabel;
```

Cela peut être utilisé pour définir la police de l'étiquette du titre, par exemple

Rapide

```
button.titleLabel?.font = UIFont.boldSystemFontOfSize(12)
```

Objectif c

```
button.titleLabel.font = [UIFont boldSystemFontOfSize:12];
```

Désactiver un UIButton

Un bouton peut être désactivé par

Rapide

```
myButton.isEnabled = false
```

Objectif c:

```
myButton.enabled = NO;
```

Le bouton deviendra gris:

Si vous ne souhaitez pas que l'apparence du bouton change lorsque désactivé, définissez `adjustsImageWhenDisabled` sur `false` / `NO`

Ajout d'une action à un UIButton via Code (par programmation)

Pour ajouter une méthode à un bouton, créez d'abord une méthode d'action:

Objectif c

```
-(void)someButtonAction:(id)sender {
  // sender is the object that was tapped, in this case its the button.
    NSLog(@"Button is tapped");
}
```

Rapide

```
func someButtonAction() {
    print("Button is tapped")
}
```

Maintenant, pour ajouter cette méthode d'action à votre bouton, vous devez écrire la ligne de code suivante:

Objectif c

```
[yourButtonInstance addTarget:self action:@selector(someButtonAction)
forControlEvents:UIControlEventTouchUpInside];
```

Rapide

```
yourButtonInstance.addTarget(self, action: #selector(someButtonAction), forControlEvents:
.TouchUpInside)
```

Pour le paramètre ControlEvents, tous les membres d' `ENUM` UIControlEvents sont valides.

Définition de la police

Rapide

```
myButton.titleLabel?.font =  UIFont(name: "YourFontName", size: 20)
```

Objectif c

```
myButton.titleLabel.font = [UIFont fontWithName:@"YourFontName" size:20];
```

Joindre une méthode à un bouton

Pour ajouter une méthode à un bouton, créez d'abord une méthode d'action:

Objectif c

```
-(void) someButtonAction{
    NSLog(@"Button is tapped");

}
```

Rapide

```
func someButtonAction() {
        print("Button is tapped")
    }
```

Maintenant, pour ajouter cette méthode d'action à votre bouton, vous devez écrire la ligne de code suivante:

Objectif c

```
[yourButtonInstance addTarget:self action:@selector(someButtonAction)
forControlEvents:UIControlEventTouchUpInside];
```

Rapide

```
yourButtonInstance.addTarget(self, action: #selector(someButtonAction), forControlEvents:
.touchUpInside)
```

Pour ControlEvents, tous les membres d' `ENUM` UIControlEvents sont valides.

Obtenir la taille de UIButton strictement basée sur son texte et sa police

Pour obtenir la taille exacte du texte d'un UIButton en fonction de sa police, utilisez la fonction `intrinsicContentSize` .

Rapide

```
button.intrinsicContentSize.width
```

Objectif c

```
button.intrinsicContentSize.width;
```

Définir une image

Rapide

```
button.setImage(UIImage(named:"test-image"), forState: .normal)
```

Objectif c

```
[self.button setImage:[UIImage imageNamed:@"test-image"] forState:UIControlStateNormal];
```

États de contrôle multiples

Vous pouvez également définir une image pour plusieurs `UIControlStates` , par exemple pour définir la même image pour l'état `Selected` et `Highlighted` :

Rapide

```
button.setImage(UIImage(named:"test-image"), forState:[.selected, .highlighted])
```

Objectif c

```
[self.button setImage:[UIImage imageNamed:@"test-image"]
forState:UIControlStateSelected|UIControlStateHighlighted];
```

Lire UIButton en ligne: https://riptutorial.com/fr/ios/topic/516/uibutton

Chapitre 161: UICollectionView

Examples

Créer une vue de collection par programme

Rapide

```
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)
}
```

Objectif 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
    }
}
```

Créer une UICollectionView

Initialiser un `UICollectionView` avec un cadre `CGRect` :

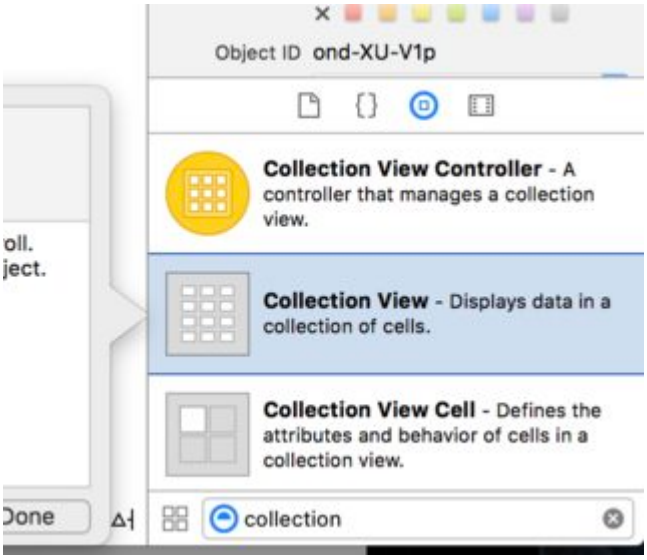
Rapide:

```
let collection = UICollectionView(frame: CGRect(x: 0, y: 0, width: 200, height: 21))
```

Objectif c:

```
UICollectionView *collection = [[UICollectionView alloc] initWithFrame:CGRectMake(0, 0, 200,
21)];
```

Vous pouvez également créer un `UICollectionView` dans Interface Builder



UICollectionView - Source de données

Chaque vue de collection doit avoir un objet `Datasource` . L'objet `Datasource` est le contenu que votre application affichera dans `UICollectionView` . Au minimum, tous les objets `Datasource` doivent implémenter `collectionView:numberOfItemsInSection:` et `collectionView:cellForItemAtIndexPath:` méthodes.

Méthodes requises

Rapide

```
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
}
```

Objectif 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;
}
```

Exemple de base Swift d'une vue de collection

Créer un nouveau projet

Il ne peut s'agir que d'une application à vue unique.

Ajouter le code

Créez un nouveau fichier Cocoa Touch Class (Fichier> Nouveau> Fichier ...> iOS> Classe Cocoa Touch). Nommez-le `MyCollectionViewCell` . Cette classe contiendra les sorties pour les vues que vous ajoutez à votre cellule dans le storyboard.

```
import UIKit
class MyCollectionViewCell: UICollectionViewCell {

    @IBOutlet weak var myLabel: UILabel!
}
```

Nous allons connecter cette prise plus tard.

Ouvrez ViewController.swift et assurez-vous d'avoir le contenu suivant:

```
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)!")
    }
}
```

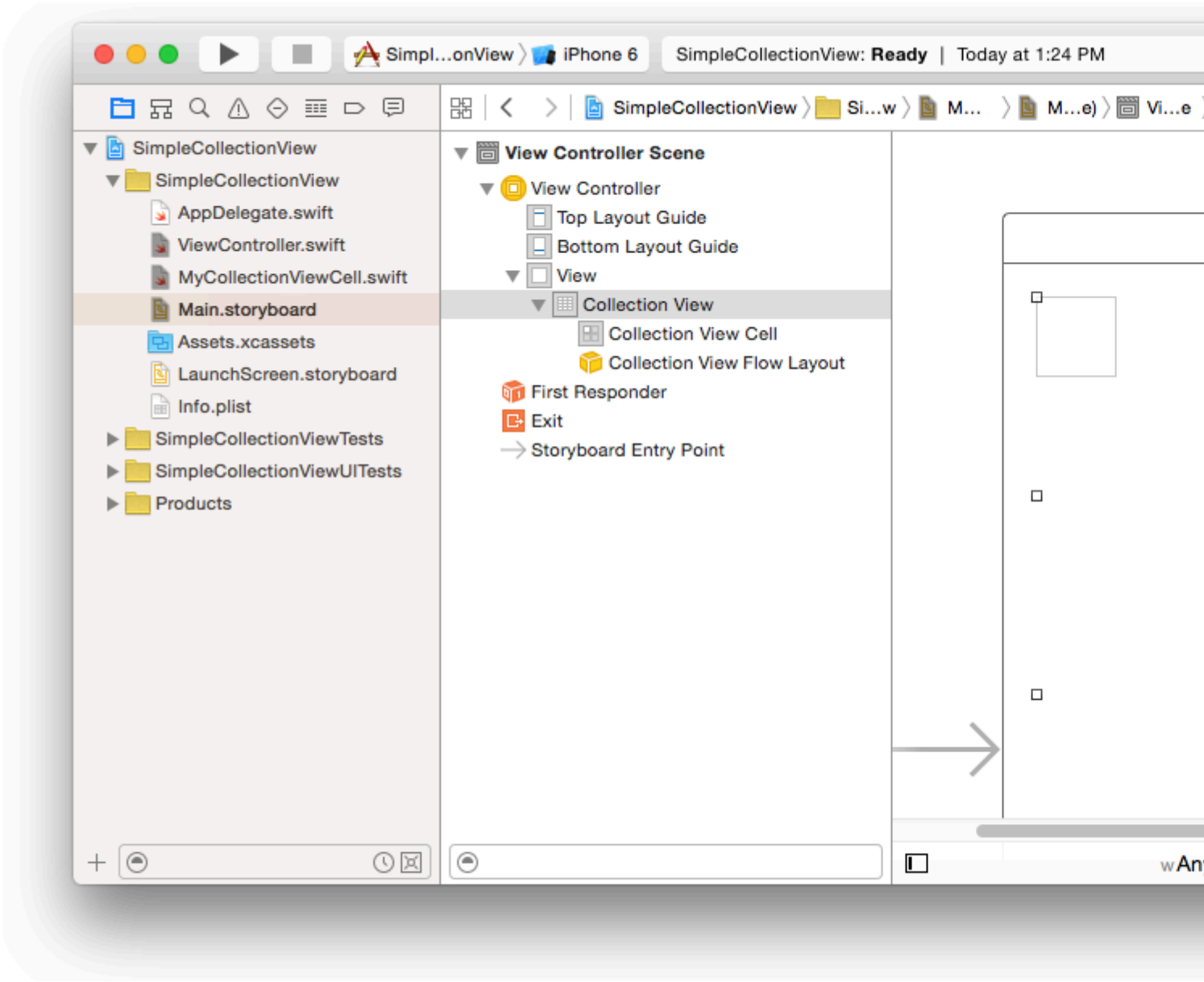
Remarques

- `UICollectionViewDataSource` et `UICollectionViewDelegate` sont les protocoles que la vue de collection suit. Vous pouvez également ajouter le protocole `UICollectionViewDelegateFlowLayout` pour modifier la taille des vues par programmation, mais cela n'est pas nécessaire.
- Nous mettons simplement des chaînes simples dans notre grille, mais vous pouvez certainement faire des images plus tard.

Configurer le storyboard

Faites glisser une vue de collection vers le contrôleur de vue de votre storyboard. Vous pouvez

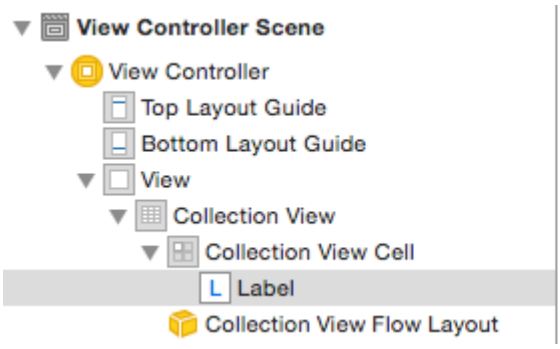
ajouter des contraintes pour le faire remplir la vue parent si vous le souhaitez.



Assurez-vous que vos valeurs par défaut dans l'inspecteur d'attributs sont également

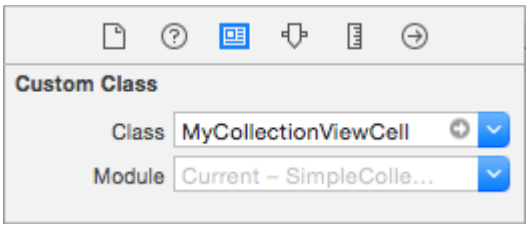
- Articles: 1
- Mise en page: flux

La petite case en haut à gauche de la vue Collection est une cellule de vue de collection. Nous allons l'utiliser comme cellule prototype. Faites glisser une étiquette dans la cellule et centrez-la. Vous pouvez redimensionner les bordures de la cellule et ajouter des contraintes pour centrer l'étiquette si vous le souhaitez.



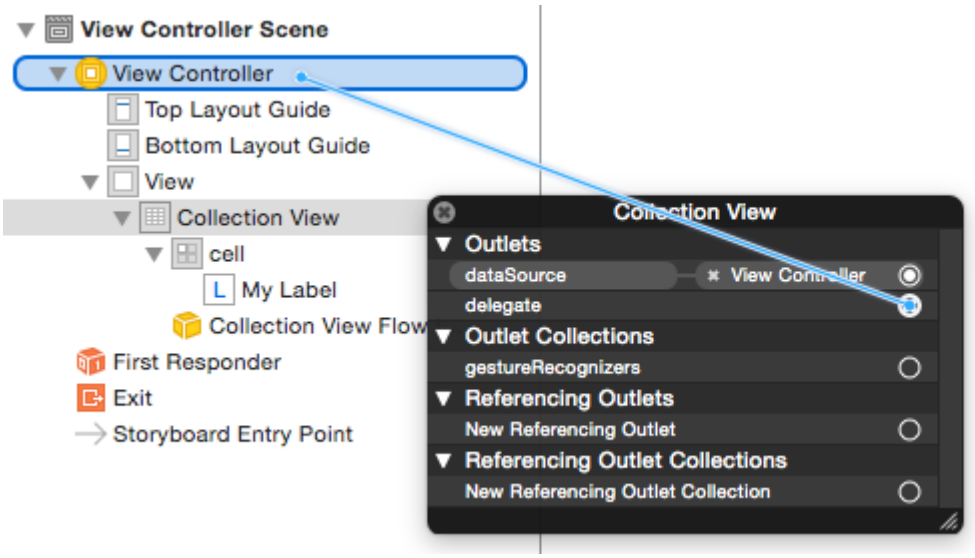
Ecrivez "cellule" (sans guillemets) dans la zone Identifiant de l'inspecteur d'attributs pour la cellule de vue de collection. Notez que c'est la même valeur que `let reuseIdentifier = "cell"` dans ViewController.swift.

Et dans l'inspecteur d'identité pour la cellule, définissez le nom de la classe sur `MyCollectionViewCell` , notre classe personnalisée que nous avons `MyCollectionViewCell` .



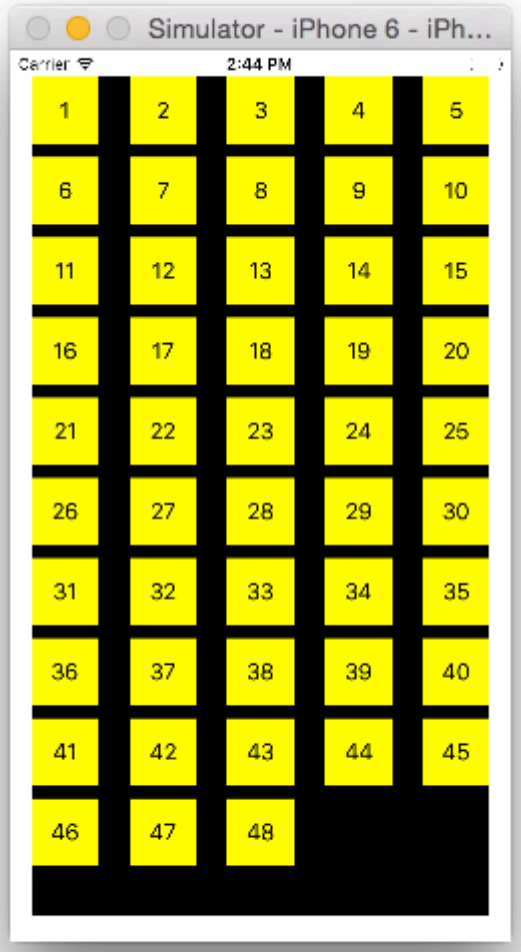
Brancher les sorties

- Accrochez l'étiquette dans la cellule de collection à `myLabel` dans la classe `MyCollectionViewCell` . (Vous pouvez faire glisser le curseur .)
- Accrochez le `delegate` et le `dataSource` du View View au View Controller. (Cliquez avec le bouton droit de la souris sur la vue Collection dans la structure du document. Cliquez ensuite sur la flèche et faites-la glisser jusqu'au View Controller.)



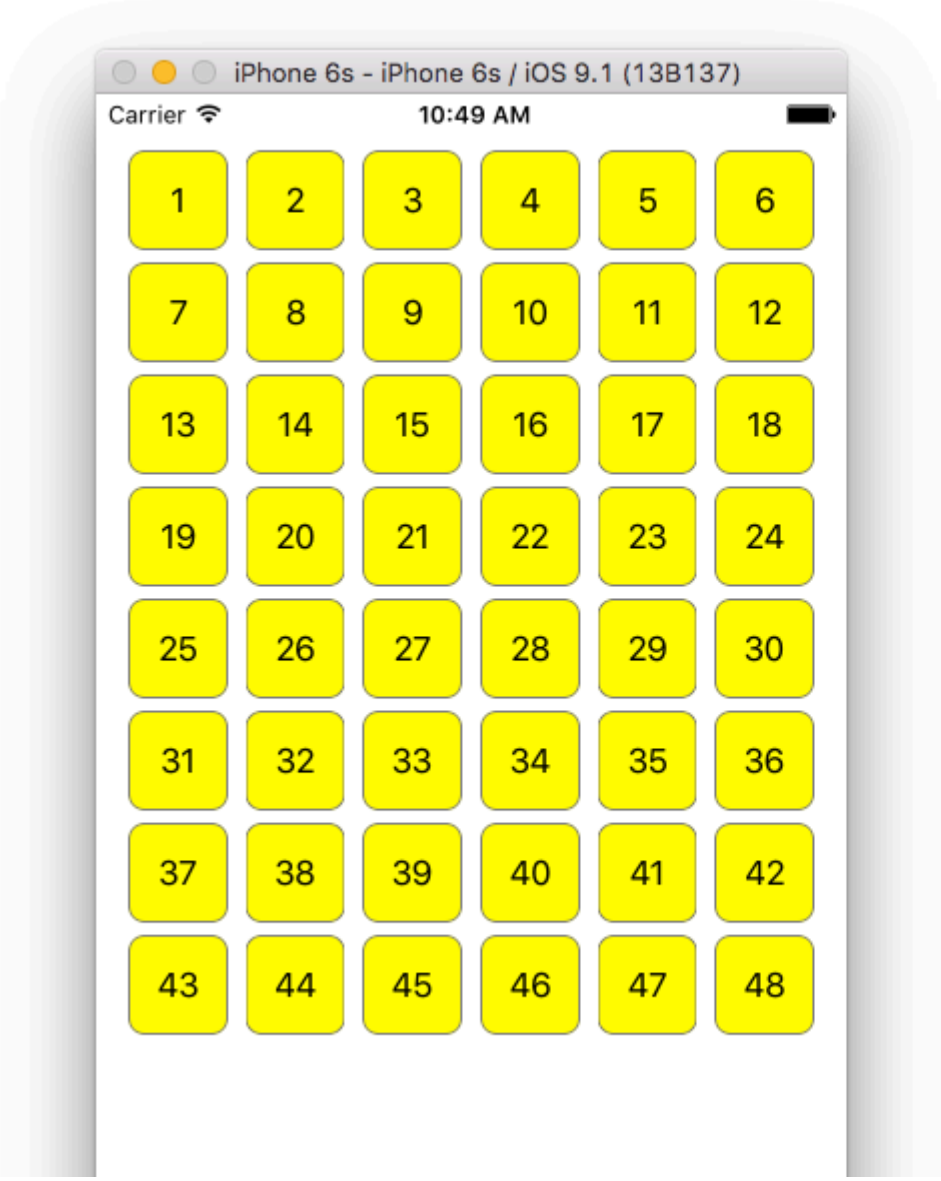
Fini

Voici à quoi cela ressemble après l'ajout de contraintes pour centrer l'étiquette dans la cellule et épingler la vue de collection aux murs du parent.



Faire des améliorations

Si vous souhaitez apporter des améliorations à l'apparence, reportez-vous à la publication d'origine de cet exemple .



Une étude plus approfondie

- Un didacticiel UICollectionView simple
- Partie 1 du didacticiel UICollectionView: Mise en route
- UICollectionView Tutorial Partie 2: Vues réutilisables et sélection de cellules

Effectuer des mises à jour par lots

Vous pouvez animer des modifications complexes dans votre vue de collection à l'aide de la méthode `performBatchUpdates` . A l'intérieur du bloc de mise à jour, vous pouvez spécifier plusieurs

modifications pour les animer en une seule fois.

```
collecitonView.performBatchUpdates({
    // Perform updates
}, nil)
```

Dans le bloc de mise à jour, vous pouvez effectuer des insertions, des suppressions, des déplacements et des rechargements. Voici comment déterminer quel indexPath utiliser:

Type	NSIndexPath
Insertion	Index dans un nouveau tableau
Effacement	Index dans l'ancien tableau
Bouge toi	de: ancien tableau, à: nouveau tableau
Recharger	soit nouveau ou ancien tableau (cela ne devrait pas avoir d'importance)

Vous ne devez appeler que le rechargement sur les cellules qui n'ont pas été déplacées, mais leur contenu a changé. Il est important de noter qu'un déplacement ne rafraîchit pas le contenu d'une cellule, mais déplace uniquement son emplacement.

Pour vérifier que votre mise à jour par lots sera effectuée correctement, assurez-vous que l'ensemble d'indexPaths pour la `deletion` , le `move-from` et le `reload` est unique et que l'ensemble d'indexPaths pour l' `insertion` , le `move-to` et le `reload` est unique.

Voici un exemple de mise à jour de lot correcte:

```
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)
```

Configuration de UICollectionViewDelegate et sélection des éléments

Parfois, si une action doit être liée à la sélection de cellules d'une vue de collection, vous devez implémenter le protocole `UICollectionViewDelegate` .

Supposons que la vue de collection se trouve dans un `UIViewController MyViewController` .

Objectif c

Dans votre *MyViewController.h* déclare qu'il implémente le protocole `UICollectionViewDelegate` , comme ci-dessous

```
@interface MyViewController : UIViewController <UICollectionViewDelegate, .../* previous
existing delegate, as UICollectionDataSource *>
```

Rapide

Dans votre *MyViewController.swift,* ajoutez ce qui suit

```
class MyViewController : UICollectionViewDelegate {
}
```

La méthode à mettre en œuvre est

Objectif c

```
-(void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath
*)indexPath
{
}
```

Rapide

```
func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath:
NSIndexPath)
{
}
```

À titre d'exemple, nous pouvons définir la couleur d'arrière-plan de la cellule sélectionnée sur verte.

Objectif c

```
-(void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath
*)indexPath
{
    UICollectionViewCell* cell = [collectionView cellForItemAtIndexPath:indexPath];
    cell.backgroundColor = [UIColor greenColor];
}
```

Rapide

```
class MyViewController : UICollectionViewDelegate {
    func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath:
NSIndexPath)
    {
        var cell : UICollectionViewCell = collectionView.cellForItemAtIndexPath(indexPath)!
        cell.backgroundColor = UIColor.greenColor()
    }
}
```

Gérer plusieurs vues de collection avec DataSource et Flowlayout

Ici, nous gérons plusieurs méthodes de délégation de collections avec des événements de Didselect.

```
extension ProductsVC: UICollectionViewDelegate, UICollectionViewDataSource{

        // MARK: - UICollectionViewDataSource
        func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection
section: Int) -> Int {
            guard collectionView == collectionCategory else {
                return arrOfProducts.count
            }
            return arrOfCategory.count
        }

        func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath:
IndexPath) -> UICollectionViewCell {

            guard collectionView == collectionProduct else {
                  let cell = collectionView.dequeueReusableCell(withReuseIdentifier:
"ProductCategoryCell", for: indexPath) as! ProductCategoryCell
                  cell.viewBackground.layer.borderWidth = 0.5
                  //Do some thing as per use
                  return cell
            }

            let cell = collectionView.dequeueReusableCell(withReuseIdentifier: cellIdentifier,
for: indexPath) as! ProductCell
            cell.contentView.layer.borderWidth = 0.5
            cell.contentView.layer.borderColor = UIColor.black.cgColor
            let json = arrOfProducts[indexPath.row]
            //Do something as per use

            return cell
        }

        func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath:
IndexPath) {
            guard collectionView == collectionCategory else {
                let json = arrOfProducts[indexPath.row]
                // Do something for  collectionProduct here
                return
            }
            let json = arrOfCategory[indexPath.row] as [String: AnyObject]
            let id = json["cId"] as? String ?? ""
            // Do something
        }
    }

    extension ProductsVC: UICollectionViewDelegateFlowLayout{

        // MARK: - UICollectionViewDelegateFlowLayout
        func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {

            let collectionWidth = collectionView.bounds.width
            guard collectionView == collectionProduct else {
                var itemWidth = collectionWidth / 4 - 1;
```

```
                if(UI_USER_INTERFACE_IDIOM() == .pad) {
                    itemWidth = collectionWidth / 4 - 1;
                }
                return CGSize(width: itemWidth, height: 50)
            }

            var itemWidth = collectionWidth / 2 - 1;
            if(UI_USER_INTERFACE_IDIOM() == .pad) {
                itemWidth = collectionWidth / 4 - 1;
            }
            return CGSize(width: itemWidth, height: 250);
        }

        func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumInteritemSpacingForSectionAt section: Int) -> CGFloat {
            return 1
        }

        func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumLineSpacingForSectionAt section: Int) -> CGFloat {
            return 1
        }

    }
```

Lire UICollectionView en ligne: https://riptutorial.com/fr/ios/topic/2399/uicollectionview

Chapitre 162: UIColor

Examples

Créer un UIColor

Vous pouvez créer un `UIColor` plusieurs manières:

Rapide

- Utiliser l'une des couleurs prédéfinies:

```
let redColor = UIColor.redColor()
let blueColor: UIColor = .blueColor()

// In Swift 3, the "Color()" suffix is removed:
let redColor = UIColor.red
let blueColor: UIColor = .blue
```

  Si le compilateur sait déjà que la variable est une instance de `UIColor` vous pouvez ignorer le type tous ensemble:

```
let view = UIView()
view.backgroundColor = .yellowColor()
```

- En utilisant la valeur de niveaux de gris et l'alpha:

```
let grayscaleColor = UIColor(white: 0.5, alpha: 1.0)
```

- En utilisant la teinte, la saturation, la luminosité et l'alpha:

```
let hsbColor = UIColor(
    hue: 0.4,
    saturation: 0.3,
    brightness: 0.7,
    alpha: 1.0
)
```

- En utilisant les valeurs RGBA:

```
let rgbColor = UIColor(
    red: 30.0 / 255,
    green: 70.0 / 255,
    blue: 200.0 / 255,
    alpha: 1.0
)
```

- En utilisant une image de motif:

```
let patternColor = UIColor(patternImage: UIImage(named: "myImage")!)
```

Objectif c

- Utiliser l'une des couleurs prédéfinies:

```
UIColor *redColor = [UIColor redColor];
```

- En utilisant la valeur de niveaux de gris et l'alpha:

```
UIColor *grayscaleColor = [UIColor colorWithWhite: 0.5 alpha: 1.0];
```

- En utilisant la teinte, la saturation, la luminosité et l'alpha:

```
UIColor *hsbColor = [UIColor
    colorWithHue: 0.4
    saturation: 0.3
    brightness: 0.7
    alpha: 1.0
];
```

- En utilisant les valeurs RGBA:

```
UIColor *rgbColor = [UIColor
    colorWithRed: 30.0 / 255.0
    green: 70.0 / 255.0
    blue: 200.0 / 255.0
    alpha: 1.0
];
```

- En utilisant une image de motif:

```
UIColor *pattenColor = [UIColor colorWithPatternImage:[UIImage
imageNamed:@"myImage.png"]];
```

Méthodes non documentées

Il existe une variété de méthodes non documentées sur `UIColor` qui exposent des couleurs ou des fonctionnalités alternatives. Ceux-ci peuvent être trouvés dans le fichier d'en-tête privé `UIColor` . Je vais documenter l'utilisation de deux méthodes privées, `styleString()` et `_systemDestructiveTintColor()` .

`styleString`

Depuis iOS 2.0, il existe une méthode d'instance privée sur `UIColor` appelée `styleString` qui renvoie une représentation de la chaîne RVB ou RGBA, même pour des couleurs comme `whiteColor` dehors de l'espace RVB.

Objectif 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)
```

Dans Swift, vous pouvez utiliser un en-tête de pontage pour exposer l'interface. Avec pure Swift, vous devrez créer un protocole `@objc` avec la méthode privée et `unsafeBitCast` `UIColor` avec le protocole:

```
@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()`

Il existe une méthode de classe non documentée sur `UIColor` appelée `_systemDestructiveTintColor` qui renvoie la couleur rouge utilisée par les boutons système destructeurs:

```
let red = UIColor.performSelector("_systemDestructiveTintColor").takeUnretainedValue()
```

Il retourne un objet non géré, que vous devez appeler `.takeUnretainedValue()` car la propriété de couleur n'a pas été transférée à notre propre objet.

Comme avec toute API non documentée, vous devez faire attention lorsque vous essayez d'utiliser cette méthode:

```
if UIColor.respondsToSelector("_systemDestructiveTintColor") {
    if let red = UIColor.performSelector("_systemDestructiveTintColor").takeUnretainedValue()
as? UIColor {
        // use the color
    }
}
```

ou en utilisant un protocole:

```
@objc protocol UIColorPrivateStatic {
    func _systemDestructiveTintColor() -> UIColor
}

let privateClass = UIColor.self as! UIColorPrivateStatic
privateClass._systemDestructiveTintColor() // UIDeviceRGBColorSpace 1 0.231373 0.188235 1
```

Couleur avec composant Alpha

Vous pouvez définir l'opacité sur un certain `UIColor` sans en créer un en utilisant l' `init(red:_,green:_,blue:_,alpha:_)` .

Rapide

```
let colorWithAlpha = UIColor.redColor().colorWithAlphaComponent(0.1)
```

Swift 3

```
//In Swift Latest Version
_ colorWithAlpha = UIColor.red.withAlphaComponent(0.1)
```

Objectif c

```
UIColor * colorWithAlpha = [[UIColor redColor] colorWithAlphaComponent:0.1];
```

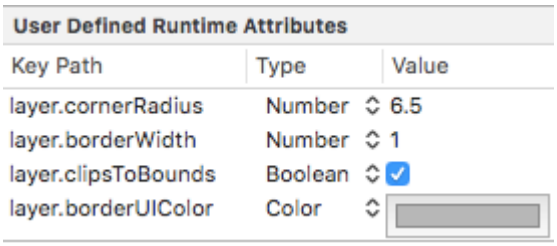
Faire en sorte que les attributs définis par l'utilisateur appliquent le type de données CGColor

Par défaut, Interface Builder n'accepte pas le `CGColor` données `CGColor` , de sorte que vous `CGColor` ajouter un `CGColor` aide des attributs définis par l'utilisateur dans le générateur d'interface. on peut vouloir utiliser une extension comme ceci:

Extension rapide:

```
extension CALayer {
    func borderUIColor() -> UIColor? {
        return borderColor != nil ? UIColor(CGColor: borderColor!) : nil
    }

    func setBorderUIColor(color: UIColor) {
        borderColor = color.CGColor
    }
}
```

Le nouvel attribut défini par l'utilisateur (borderUIColor) sera reconnu et appliqué sans problème.



Créer un UIColor à partir d'un nombre hexadécimal ou d'une chaîne

Vous pouvez créer une `UIColor` partir d'un nombre hexadécimal ou d'une chaîne, par exemple 0xff00cc, "#FFFFFF"

Rapide

Valeur 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)
    }
}
```

Exemple:

```
let color = UIColor(hex: 0xff00cc, alpha: 1.0)
```

Notez que pour `alpha` la valeur par défaut de `1.0` est fournie, elle peut donc être utilisée comme suit:

```
let color = UIColor(hex: 0xff00cc)
```

Valeur de chaîne

```
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)
    }
}
```

Exemple d'utilisation:

Hex avec alpha

```
let color = UIColor("#80FFFFFF")
```

Hex sans alpha (la `color` alpha sera égale à 1,0)

```
let color = UIColor("#FFFFFF")
let color = UIColor("#FFF")
```

Objectif c

Valeur 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
```

Exemple:

```
UIColor *color = [UIColor colorWithHex:0xff00cc alpha:1.0];
```

Valeur de chaîne

```
- (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];
}
```

Exemple d'utilisation:

Hex avec alpha

```
UIColor* color = [self hex:@"#80FFFFFF"];
```

Hex sans alpha (la `color` alpha sera égale à 1)

```
UIColor* color = [self hex:@"#FFFFFF"];
UIColor* color = [self hex:@"#FFF"];
```

Luminosité de couleur ajustée de UIColor

L'exemple de code ci-dessous vous donnera une version ajustée de cette couleur où un pourcentage plus élevé sera plus lumineux et un pourcentage plus faible plus sombre.

Objectif 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;
}
```

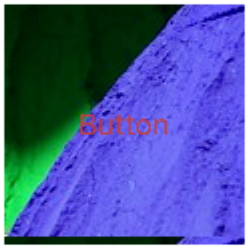
Rapide

```
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 à partir d'un motif d'image

Vous pouvez créer un objet `UIColor` aide d'un motif d'image à l'aide de la `UIColor(patternImage:_)` .

```
btn.backgroundColor = UIColor(patternImage: UIImage(named: "image")!)
```



Ombre plus claire et plus foncée d'une couleur UIC donnée

L'exemple de code ci-dessous montre comment obtenir une nuance plus claire et plus foncée d'une couleur donnée, utile dans les applications ayant des thèmes dynamiques

Pour une **couleur plus foncée**

```
+ (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;
}
```

Pour une **couleur plus claire**

```
+ (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;
}
```

Voir Différences visuelles ci-dessous, en considérant que la couleur donnée est `[UIColor orangeColor]`

Lire UIColor en ligne: https://riptutorial.com/fr/ios/topic/956/uicolor

Chapitre 163: UIControl - Gestion des événements avec des blocs

Examples

introduction

Généralement, lorsque vous utilisez `UIControl` ou `UIButton` , nous ajoutons un `selector` tant qu'action de rappel lorsqu'un événement se produit sur un bouton ou un contrôle, par exemple lorsque l'utilisateur appuie sur le bouton ou touche le contrôle.

Par exemple, nous ferions ce qui suit:

```
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")
    }
}
```

En ce qui concerne le `selector` , le compilateur doit seulement savoir qu'il existe. Cela peut être fait via un `protocol` et ne pas être implémenté.

Par exemple, ce qui suit planterait votre application:

```
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()
    }
}
```

C'est parce que votre application `onButtonPress` PAS la fonction `onButtonPress` .

Maintenant, si vous pouviez faire tout cela en même temps que l'initialisation du bouton? Que faire si vous n'aviez pas à spécifier de rappels et que vous pouviez spécifier des blocs pouvant être ajoutés et supprimés à tout moment? Pourquoi se soucier de la mise en œuvre des sélecteurs?

Solution

```
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
    }
}
```

Ce qui précède est une simple extension sur `UIControl` . Il ajoute une classe privée interne qui a une `func run(_ control: UIControl)` rappel `func run(_ control: UIControl)` utilisée comme action des événements.

Ensuite, nous utilisons l' `object association` pour ajouter et supprimer la cible car elle ne sera pas conservée par `UIControl` .

La fonction de gestionnaire d'événements renvoie un `Protocol` afin de masquer le fonctionnement interne de la classe `Target` , mais également pour vous `enable` d' `enable` et de `disable` la cible à tout moment.

Exemple d'utilisation:

```
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()
    }
}
```

Lire UIControl - Gestion des événements avec des blocs en ligne:
https://riptutorial.com/fr/ios/topic/3180/uicontrol---gestion-des-evenements-avec-des-blocs

Chapitre 164: UIDatePicker

Remarques

`UIDatePicker` n'hérite pas d' `UIPickerView` , mais gère un objet de vue de sélecteur personnalisé en tant que sous-vue.

Examples

Créer un sélecteur de date

Rapide

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
```

Objectif c

```
UIDatePicker *datePicker = [[UIDatePicker alloc] initWithFrame:CGRectMake(x: 0, y: 0, width:
320, height: 200)];
```

Réglage de la date minimum-maximum

Vous pouvez définir la date minimale et maximale que UIDatePicker peut afficher.

Date minimum

```
[datePicker setMinimumDate:[NSDate date]];
```

Date maximale

```
[datePicker setMaximumDate:[NSDate date]];
```

Modes

UIDatePicker dispose de plusieurs modes de sélection.

```
enum UIDatePickerMode : Int {
    case Time
    case Date
    case DateAndTime
    case CountDownTimer
}
```

- `Time` - Le sélecteur de date affiche les heures, les minutes et (éventuellement) une désignation AM / PM.
- `Date` - Le sélecteur de date affiche les mois, les jours du mois et les années.
- `DateAndTime` - Le sélecteur de date affiche les dates (en tant que jour unifié de la semaine, le mois et le jour du mois) plus les heures, les minutes et (éventuellement) une désignation AM / PM.
- `CountDownTimer` - Le sélecteur de date affiche les valeurs des heures et des minutes, par exemple [1 | 53]. L'application doit définir une minuterie pour déclencher à l'intervalle approprié et définir le sélecteur de date que les secondes cochez.

Définition de la propriété datePickerMode

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.datePickerMode = .Date
```

Réglage de l'intervalle des minutes

Vous pouvez modifier la propriété `minuteInterval` pour définir l'intervalle affiché par la roue des minutes. La valeur par défaut est 1, la valeur maximale est 30.

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.minuteInterval = 15
```

Durée de compte à rebours

La valeur `NSTimeInterval` de cette propriété indique les secondes à partir desquelles le sélecteur de date en mode compte à rebours décompte. Si le mode du sélecteur de date n'est pas `CountDownTimer` , cette valeur est ignorée. La valeur maximale est 86,399 secondes (23:59)

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.countDownDuration = 60 * 60
```

Lire UIDatePicker en ligne: https://riptutorial.com/fr/ios/topic/5643/uidatepicker

Chapitre 165: UIDevice

Paramètres

Propriété	La description
prénom	Le nom identifiant le périphérique.
systemName: String	Nom du système d'exploitation exécuté sur le périphérique représenté par le récepteur.
modèle: String	Le modèle de l'appareil.
systemVersion: String	La version actuelle du système d'exploitation.

Remarques

La classe UIDevice fournit une instance Singleton représentant le périphérique en cours. A partir de cette instance, vous pouvez obtenir des informations sur le périphérique, telles que le nom, le modèle de périphérique et le nom et la version du système d'exploitation.

Examples

Obtenir le nom du modèle d'appareil 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
}
```

Obtenir l'état de la batterie et le niveau de la batterie

```
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%, ...)
}
```

Identification de l'appareil et fonctionnement

```
 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
```

Obtenir l'orientation du périphérique

```
UIDevice *deviceInfo = [UIDevice currentDevice];
int d = deviceInfo.orientation;
```

`deviceInfo.orientation` renvoie une valeur UIDeviceOrientation indiquée ci-dessous:

```
UIDeviceOrientationUnknown 0
UIDeviceOrientationPortrait 1
UIDeviceOrientationPortraitUpsideDown 2
UIDeviceOrientationLandscapeLeft 3
UIDeviceOrientationLandscapeRight 4
UIDeviceOrientationFaceUp 5
UIDeviceOrientationFaceDown 6
```

Écoute des changements d'orientation du périphérique dans un View Controller:

```
- (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
}
```

Pour désactiver la vérification de tout changement d'orientation:

```
- (void)viewWillDisappear:(BOOL)animated {
    [super viewWillDisappear:animated];
    [[UIDevice currentDevice] endGeneratingDeviceOrientationNotifications];
}
```

Obtenir l'état de la batterie du périphérique

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

Utilisation du capteur de proximité

```
//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.");
    }
}
```

Lire UIDevice en ligne: https://riptutorial.com/fr/ios/topic/4878/uidevice

Chapitre 166: UIFeedbackGenerator

Introduction

`UIFeedbackGenerator` et ses sous-classes offrent une interface publique à Taptic Engine® présente sur les appareils iOS à partir de l'iPhone 7. Haptics, de la marque Taptics, fournit des informations tactiles pour les événements à l'écran. Alors que de nombreux contrôles système fournissent des haptiques `UIFeedbackGenerator` emploi, les développeurs peuvent utiliser les sous-classes `UIFeedbackGenerator` pour ajouter des haptiques aux contrôles personnalisés et autres événements. `UIFeedbackGenerator` est une classe abstraite qui ne doit pas être utilisée directement. Les développeurs utilisent plutôt une de ses sous-classes.

Examples

Effet déclencheur Haptic

L'exemple montre comment déclencher une haptique d'impact en utilisant `UIImpactFeedbackGenerator` après un `UIImpactFeedbackGenerator` sur un bouton.

Rapide

```
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()
    }
}
```

Objectif 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
```

Lire UIFeedbackGenerator en ligne: https://riptutorial.com/fr/ios/topic/10048/uifeedbackgenerator

Chapitre 167: UIFont

Introduction

UIFont est une classe utilisée pour obtenir et définir les informations relatives aux polices. Il hérite de `NSObject` et est conforme à `Hashable` , `Equatable` , `CVarArg` et `NSCopying` .

Examples

Déclarer et initialiser UIFont

Vous pouvez déclarer un `UIFont` comme suit:

```
var font: UIFont!
```

`UIFont` a plus de méthodes `init()` :

- `UIFont.init(descriptor: UIFontDescriptor, size: CGFloat)`
- `UIFont.init(name: String, size: CGFloat)`

Par conséquent, vous pouvez initialiser un `UIFont` comme ceci:

```
let font = UIFont(name: "Helvetica Neue", size: 15)
```

La police par défaut est `System` , taille `17` .

Changer la police d'une étiquette

Pour modifier la police de texte d'une étiquette, vous devez accéder à sa propriété de `font` :

```
label.font = UIFont(name:"Helvetica Neue", size: 15)
```

Le code ci-dessus changera la police de l'étiquette en `Helvetica Neue` , taille `15` . Attention, vous devez épeler le nom de la police correctement, sinon, cette erreur sera lancée, car la valeur initialisée ci-dessus est facultative et peut donc être nulle:

> Inopinément trouvé nil en déroulant une valeur optionnelle

Lire UIFont en ligne: https://riptutorial.com/fr/ios/topic/9792/uifont

Chapitre 168: UIGestureRecognizer

Examples

UITapGestureRecognizer

Initialisez `UITapGestureRecognizer` avec une cible, `self` dans ce cas, et une `action` qui est une méthode avec un seul paramètre: un `UITapGestureRecognizer` .

Après l'initialisation, ajoutez-le à la vue pour qu'il reconnaisse les taps.

Rapide

```
override func viewDidLoad() {
    super.viewDidLoad()
    let recognizer = UITapGestureRecognizer(target: self,
                                            action: #selector(handleTap(_:)))
    view.addGestureRecognizer(recognizer)
}

func handleTap(recognizer: UITapGestureRecognizer) {

}
```

Objectif c

```
- (void)viewDidLoad {
    [super viewDidLoad];
    UITapGestureRecognizer *recognizer =
        [[UITapGestureRecognizer alloc] initWithTarget:self
                                                action:@selector(handleTap:)];
    [self.view addGestureRecognizer:recognizer];
}

- (void)handleTap:(UITapGestureRecognizer *)recognizer {

}
```

Exemple de renvoi au clavier via UITapGestureRecognizer:

Tout d'abord, vous créez la fonction de rejet du clavier:

```
func dismissKeyboard() {
        view.endEditing(true)
    }
```

Ensuite, vous ajoutez un identificateur de geste dans votre contrôleur de vue, en appelant la méthode que nous venons de créer.

```
let tap: UITapGestureRecognizer = UITapGestureRecognizer(target: self, action:
"dismissKeyboard")
        view.addGestureRecognizer(tap)
```

Exemple d'obtention de l'emplacement de geste UITapGestureRecognizer (Swift 3):

```
func handleTap(gestureRecognizer: UITapGestureRecognizer) {
print("tap working")
if gestureRecognizer.state == UIGestureRecognizerState.recognized
   {
     print(gestureRecognizer.location(in: gestureRecognizer.view))
   }
}
```

UIPanGestureRecognizer

Les détecteurs de mouvements de panoramique détectent les mouvements de déplacement. L'exemple suivant ajoute une image à un contrôleur de vue et permet à l'utilisateur de la faire glisser sur l'écran.

Objectif 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];
}
```

Rapide

```
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)
}
```

> **Remarque:** Bien que `UIPanGestureRecognizer` soit utile pour détecter les `UIPanGestureRecognizer` de glissement, si vous souhaitez simplement détecter un geste de base tel que faire glisser un doigt de gauche à droite ou de haut en bas, utilisez `UISwipeGestureRecognizer` . `UIPanGestureRecognizer` est un meilleur choix si vous avez besoin d'accéder à des méthodes telles que `translationInView:` ou `velocityInView:` `UIPanGestureRecognizer`

UITapGestureRecognizer (Double Tap)

Le double tapotement, comme un simple tapotement, utilise également le `UITapGestureRecognizer` . Vous définissez simplement le `numberOfTapsRequired` sur `2` .

Rapide

```
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"
}
```

Remarques

- Un exemple de projet peut être trouvé ici .
- Vous pouvez reconnaître un triple tap en définissant le `numberOfTapsRequired` sur `3` .

UILongPressGestureRecognizer

`UILongPressGestureRecognizer` vous permet d'écouter un `UILongPressGestureRecognizer` long sur une vue. Vous pouvez définir la durée du délai avant l'appel de la méthode d'action.

Rapide

```
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"
    }
}
```

Remarques

- Un exemple de projet plus complet peut être trouvé ici .
- Changez le `minimumPressDuration` pour définir la longueur de l' `minimumPressDuration` long.

UISwipeGestureRecognizer

Les gestes de balayage vous permettent d'écouter l'utilisateur déplacer rapidement son doigt sur l'écran dans une certaine direction.

Rapide

```
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"
    }
}
```

Objectif 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");
    }

}
```

Remarques

- Un exemple de projet plus complet peut être trouvé ici .

UIPinchGestureRecognizer

Les pincements sont un geste à deux doigts où les doigts se rapprochent ou s'éloignent les uns des autres. Ce geste est généralement utilisé pour redimensionner une vue.

Rapide

```
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
    }
}
```

Remarques

- Un exemple de projet plus complet peut être trouvé ici .

UIRotationGestureRecognizer

Deux doigts qui tournent autour d'un centre peuvent être écoutés avec `UIRotationGestureRecognizer` . Ceci est généralement utilisé pour faire pivoter une vue.

Rapide

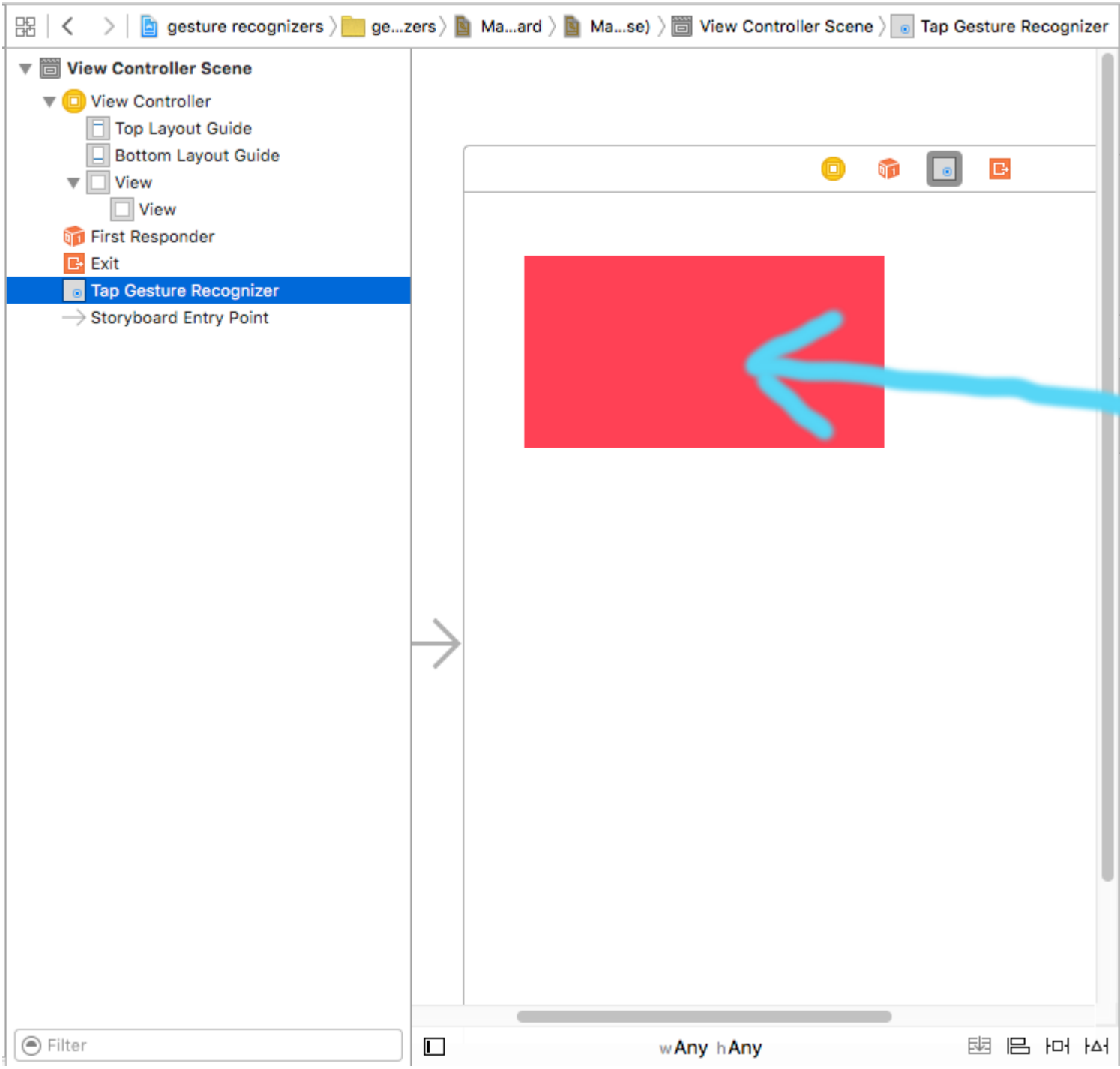
```
override func viewDidLoad() {
    super.viewDidLoad()

    // 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
    }
}
```

Remarques

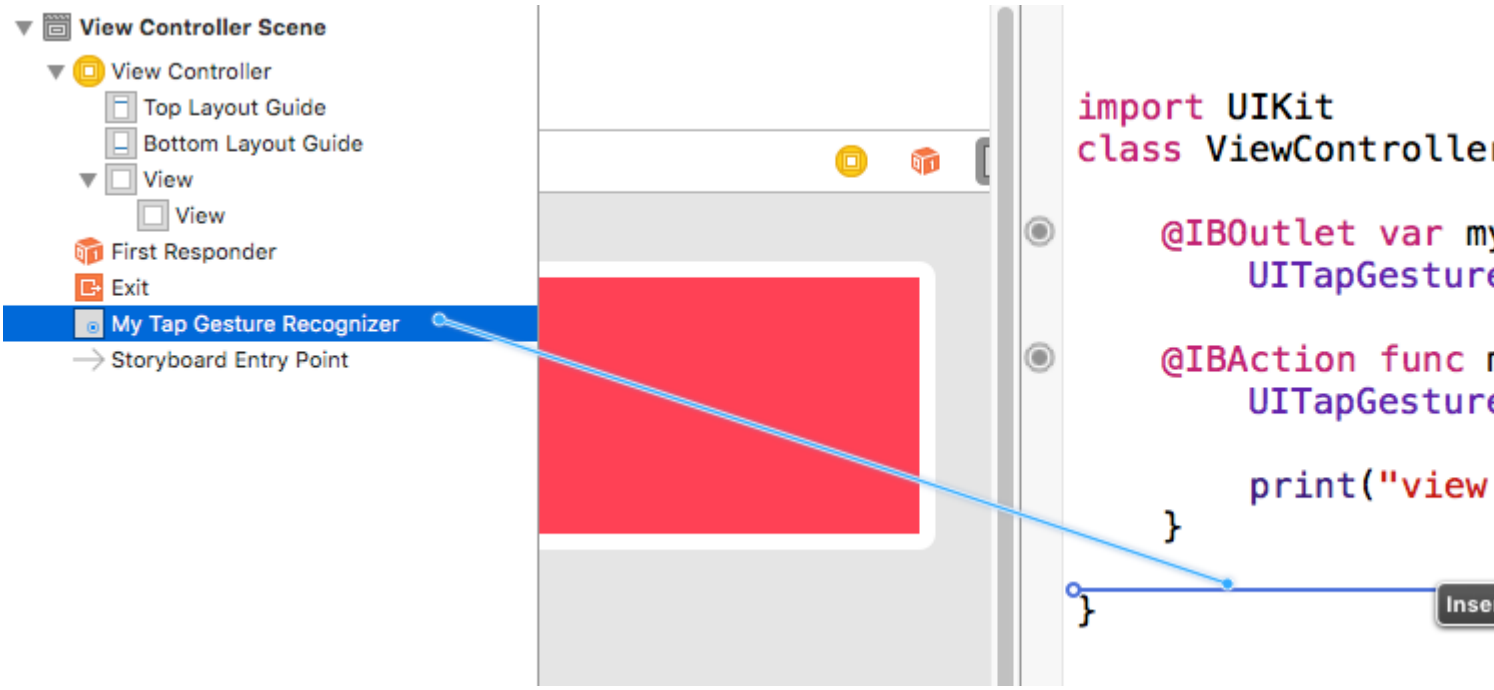
- Un exemple de projet peut être trouvé ici .

Ajout d'un dispositif de reconnaissance de mouvements dans Interface Builder

Faites glisser un identificateur de geste de la bibliothèque d'objets sur votre vue.



Contrôlez le glissement du geste dans la structure du document vers votre code View Controller afin de créer une sortie et une action.



Remarques

- Cet exemple provient de cet exemple de projet plus complet démontrant des identificateurs de gestes.

Lire UIGestureRecognizer en ligne: https://riptutorial.com/fr/ios/topic/1289/uigesturerecognizer

Chapitre 169: UIImage

Remarques

Sujet développeur Apple pour UIImage

Examples

Créer UIImage

Avec image locale

Rapide

```
let image = UIImage(named: "imageFromBundleOrAsset")
```

Objectif c

```
UIImage *image = [UIImage imageNamed:@"imageFromBundleOrAsset"];
```

Remarque

La méthode `imageNamed` met en cache le contenu de l'image en mémoire. Le chargement de nombreuses images volumineuses de cette manière peut entraîner des avertissements de mémoire insuffisante, ce qui peut entraîner la fermeture de l'application. Cela peut être corrigé en utilisant la méthode `imageWithContentsOfFile` de `UIImage` , qui n'utilise pas la mise en cache.

Avec NSData

Rapide

```
let imageData = Data(base64Encoded: imageString, options:
Data.Base64DecodingOptions.ignoreUnknownCharacters)

let image = UIImage(data: imageData!)
```

Avec UIColor

Rapide

```
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()
```

Objectif 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();
```

Avec le contenu du fichier

Objectif c

Exemple:

```
UIImage *image = [UIImage imageWithContentsOfFile:[[NSBundle mainBundle]
pathForResource:[cellCountry objectForKey:@"Country_Flag"] ofType:nil]];
```

En utilisant un tableau:

Exemple:

```
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;
    }
}
```

// Utiliser un tableau d'images pour les animations ici:

```
self.myImageView.animationImages = imageArray;
```

Création et initialisation d'objets image avec le contenu du fichier

Création et retour d'un objet image en chargeant les données d'image à partir du fichier dans le chemin spécifié.

Exemple:

```
 UIImage *image = [UIImage imageWithContentsOfFile:[[NSBundle mainBundle]
pathForResource:[cellCountry objectForKey:@"Country_Flag"] ofType:nil]];
```

En utilisant un tableau:

Exemple

```
 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;
```

Image redimensionnable avec majuscules

Dans l'exemple d'une bulle de message illustrée ci-dessous: les coins de l'image doivent rester inchangés, ce qui est spécifié par `UIEdgeInsets` , mais les bordures et le centre de l'image doivent être étendus pour couvrir la nouvelle taille.

```
let insets = UIEdgeInsetsMake(12.0, 20.0, 22.0, 12.0)
let image = UIImage(named: "test")
image?.resizableImageWithCapInsets(insets, resizingMode: .Stretch)
```

Comparer des images

> La méthode `isEqual:` est le seul moyen fiable de déterminer si deux images contiennent les mêmes données d'image. Les objets image que vous créez peuvent être différents les uns des autres, même lorsque vous les initialisez avec les mêmes données d'image mises en cache. La seule façon de déterminer leur égalité est d'utiliser la méthode `isEqual:` qui compare les données d'image réelles. Le listing 1 illustre les manières correctes et incorrectes de comparer les images.
>
> Source: Documentation Apple

Rapide

```
// 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.
}
```

Objectif 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.
}
```

Créer UIImage avec UIColor

Rapide

```
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()
```

Objectif c:

Ajoutez cette méthode en tant qu'extension de `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;
}
```

Image dégradée avec couleurs

Création d'un dégradé `UIImage` avec des couleurs dans `CGRect`

Rapide:

```
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!
    }
}
```

Usage:

```
let image = UIImage.gradientImageWithBounds(CGRect(x: 0, y: 0, width: 200, height: 200),
colors: [UIColor.yellowColor().CGColor, UIColor.blueColor().CGColor])
```

Objectif 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;
}
```

Couche de fond dégradé pour les limites

```
+ (CALayer *)gradientBGLayerForBounds:(CGRect)bounds colors:(NSArray *)colors
{
    CAGradientLayer * gradientBG = [CAGradientLayer layer];
    gradientBG.frame = bounds;
    gradientBG.colors = colors;
    return gradientBG;
}
```

Convertir UIImage vers / depuis l'encodage base64

Codage

```
//convert the image to NSData first
let imageData:NSData = UIImagePNGRepresentation(image)!
// convert the NSData to base64 encoding
let strBase64:String =
imageData.base64EncodedStringWithOptions(.Encoding64CharacterLineLength)
```

Décodage

```
let dataDecoded:NSData = NSData(base64EncodedString: strBase64, options:
NSDataBase64DecodingOptions(rawValue: 0))!
let decodedimage:UIImage = UIImage(data: dataDecoded)!
```

Prenez un instantané d'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
```

Appliquer UIColor à UIImage

Utilisez la même application UIImage avec plusieurs bases de thème en appliquant simplement UIColor à l'instance UIImage comme suit.

```
// *** 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
```

Maintenant, disons que vous voulez faire la même chose avec UIImageView, puis utilisez le code suivant

```
[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.
```

Changer la couleur UIImage

Swift Ajoutez cette extension à 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
        }
    }
}
```

Ensuite, pour changer la couleur de votre UIImage

```
my_image.maskWithColor(UIColor.blueColor())
```

Trouvé sur ce lien

Lire UIImage en ligne: https://riptutorial.com/fr/ios/topic/1409/uiimage

Chapitre 170: UIImagePickerController

Introduction

UIImagePickerController fournit une solution quasiment prête à l'emploi pour permettre à l'utilisateur de sélectionner une image à partir de son périphérique ou de prendre une photo avec l'appareil photo, puis de présenter cette image. En vous conformant à UIImagePickerControllerDelegate, vous pouvez créer une logique qui spécifie dans votre application comment présenter l'image et comment l'utiliser (en utilisant didFinishPickingMediaWithInfo) et si l'utilisateur refuse de sélectionner une image ou de la prendre (en utilisant imagePickerControllerDidCancel)).

Examples

Utilisation générique de UIImagePickerController

Étape 1: Créez le contrôleur, définissez le délégué et conformez-vous au protocole

```
//Swift
class ImageUploadViewController: UIViewController, UIImagePickerControllerDelegate,
UINavigationControllerDelegate {

    let imagePickerController = UIImagePickerController()

    override func viewDidLoad() {
        super.viewDidLoad()
        imagePickerController.delegate = self
    }
}

//Objective-C
@interface ImageUploadViewController : UIViewController
<UIImagePickerControllerDelegate,UINavigationControllerDelegate> {

    UIImagePickerController *imagePickerController;

}

@end

@implementation ImageUploadViewController

- (void)viewDidLoad {

    [super viewDidLoad];

    imagePickerController.delegate = self;

}

@end
```

note: En réalité, nous `UINavigationControllerDelegate` rien de défini dans `UINavigationControllerDelegate` , mais `UIImagePickerController` hérite de `UINavigationController` et modifie le comportement de `UINavigationController` . Par conséquent, nous devons encore dire que notre contrôleur de vue est conforme à `UINavigationControllerDelegate` .

Étape 2: Chaque fois que vous devez montrer `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];
```

Étape 3: implémentez les méthodes de délégué:

```
//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];

}
```

Lire UIImagePickerController en ligne:
https://riptutorial.com/fr/ios/topic/3023/uiimagepickercontroller

Chapitre 171: UIImageView

Examples

Créer un UIImageView

Pour créer un `UIImageView` programmation, il vous suffit de créer une instance de `UIImageView` :

```
//Swift
let imageView = UIImageView()

//Objective-C
UIImageView *imageView = [[UIImageView alloc] init];
```

Vous pouvez définir la taille et la position du `UIImageView` avec un `CGRect` :

```
//Swift
imageView.frame = CGRect(x: 0, y: 0, width: 200, height: 200)

//Objective-C
imageView.frame = CGRectMake(0,0,200,200);
```

Ou vous pouvez définir la taille lors de l'initialisation:

```
//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;
```

> Remarque: Vous devez importer `UIKit` pour utiliser un `UIImageView` .

Assigner une image à un UIImageView

Vous pouvez affecter une image à un `UIImageView` lors de l'initialisation ou ultérieurement à l'aide de la propriété `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"];
```

Animation d'un UIImageView

Vous pouvez animer un `UIImageView` en affichant rapidement les images dans une séquence en utilisant les propriétés d'animation de `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 propriété `animationImages` est un `Array` de `UIImages` qui est exécuté de haut en bas lorsque l'animation est déclenchée.

La propriété `animationDuration` est un `Double` indiquant combien de secondes l'animation sera exécutée.

La propriété `animationRepeatCount` est un objet `Int` qui indique combien de fois l'animation sera exécutée.

Pour démarrer et arrêter l'animation, vous pouvez appeler les méthodes appropriées:

```
imageView.startAnimating()
imageView.stopAnimating()
```

Il existe une méthode `isAnimating()` qui renvoie une valeur `Boolean` indiquant si l'animation est en cours d'exécution ou non.

S'il vous plaît noter que ce n'est pas un moyen très efficace pour créer des animations: c'est assez lent et consommant beaucoup de ressources. Envisagez d'utiliser des calques ou des sprites pour de meilleurs résultats

Faire une image dans un cercle ou arrondi

Cet exemple montre comment créer un `UIView` ou un `UIImageView` , arrondi avec un rayon comme celui-ci:

Objectif c

```
someImageView.layer.cornerRadius = CGRectGetHeight(someImageView.frame) / 2;
someImageView.clipsToBounds = YES;
```

Rapide

```
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)
```

Il est suggéré que si vous utilisez autolayout que vous mettez le `someImageView.layer.cornerRadius` code `viewDidLayoutSubviews` . Cela permettra à `cornerRadius` l'image de se mettre à jour si l'image change de taille.

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

```
    someImageView.layer.cornerRadius = someImageView.frame.size.width/2
    someImageView.layer.masksToBounds = true
}
```

UIImage masqué avec étiquette

Cela rend l'image masquée à la forme des lettres de l'étiquette:

Objectif c

```
self.maskImage.layer.mask = self.maskLabel.layer;
self.maskImage.layer.masksToBounds = YES;
```

Swift 3

```
maskImageView.mask = maskLabel
maskImageView.masksToBounds = true
```

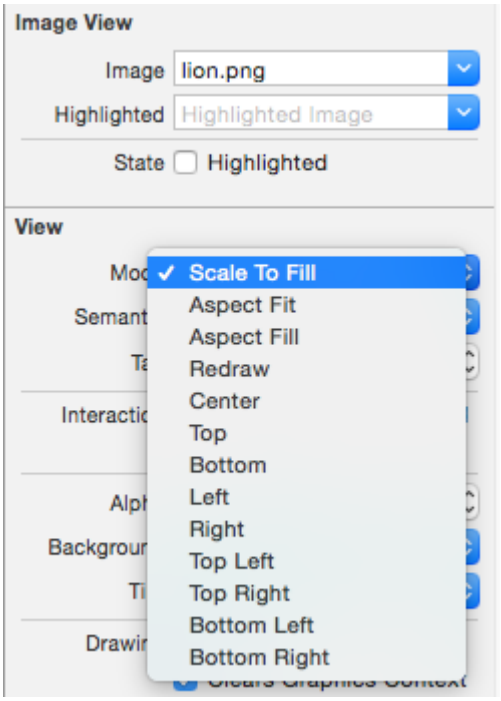
Voici le résultat:

Changer la couleur d'une image

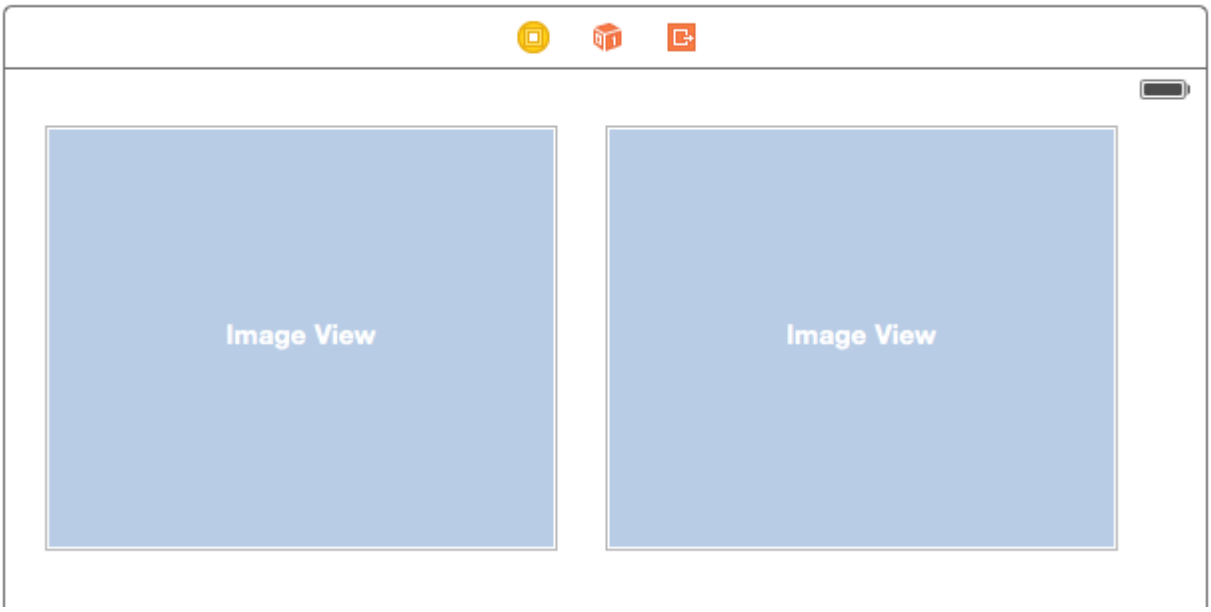
```
//Swift
imageView.tintColor = UIColor.redColor()
imageView.image = imageView.image?.imageWithRenderingMode(.AlwaysTemplate)

//Swift 3
imageView.tintColor = UIColor.red
imageView.image = imageView.image?.withRenderingMode(.alwaysTemplate)

//Objective-C
imageView.tintColor = [UIColor redColor];
imageView.image = [imageView.image imageWithRenderingMode:UIImageRenderingModeAlwaysTemplate]
```

Comment la propriété Mode affecte une image

La propriété mode de contenu d'une vue indique comment son contenu doit être présenté. Dans Interface Builder, les différents modes peuvent être sélectionnés dans l'inspecteur d'attributs.



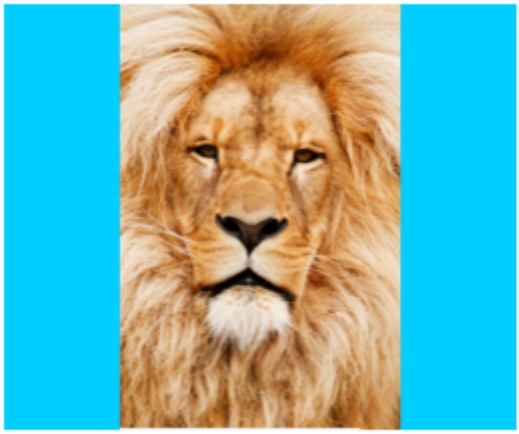
Utilisons deux vues d'image pour voir comment fonctionnent les différents modes.



Echelle à remplir


Les hauteurs et les largeurs d'image sont étirées pour correspondre à la taille de `UIImageView` .

Aspect Fit




Le côté le plus long (hauteur ou largeur) de l'image est étiré pour correspondre à la vue. Cela rend l'image aussi grande que possible tout en affichant l'image entière sans déformer la hauteur ou la largeur. (J'ai défini le fond `UIImageView` sur bleu pour que sa taille soit claire.)

Remplissage d'aspect


Le côté le plus court (hauteur ou largeur) de l'image est étiré pour correspondre à la vue. Comme "Aspect Fit", les proportions de l'image ne sont pas déformées par rapport à leur format d'origine.

Redessiner


Redraw est uniquement pour les vues personnalisées qui doivent effectuer leur propre redimensionnement et redimensionnement. Nous n'utilisons pas de vue personnalisée, nous ne devrions donc pas utiliser Redraw. Notez qu'ici, `UIImageView` nous donne juste le même résultat que Scale to Fill, mais il fait plus de travail en coulisse.

À propos de Redraw, la documentation Apple indique:

> Les modes de contenu permettent de recycler le contenu de votre vue, mais vous pouvez également définir le mode de contenu sur la valeur `UIViewContentModeRedraw` lorsque vous souhaitez que vos vues personnalisées se redessinent lors des opérations de redimensionnement et de redimensionnement. La définition du mode de contenu de votre vue sur cette valeur force le système à appeler la méthode `drawRect:` votre vue en réponse aux modifications de géométrie. En général, vous devriez éviter d'utiliser cette valeur autant que possible, et vous ne devriez certainement pas l'utiliser avec les vues système standard.

Centre

L'image est centrée dans la vue mais la longueur et la largeur de l'image ne sont pas étirées.

Haut

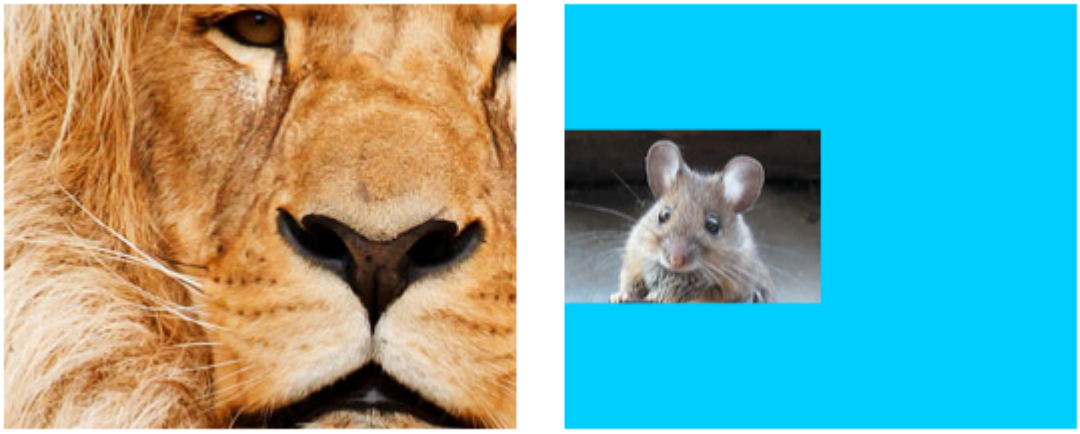
Le bord supérieur de l'image est centré horizontalement en haut de la vue et la longueur et la largeur de l'image ne sont pas étirées.

Bas

Le bord inférieur de l'image est centré horizontalement au bas de la vue et la longueur et la

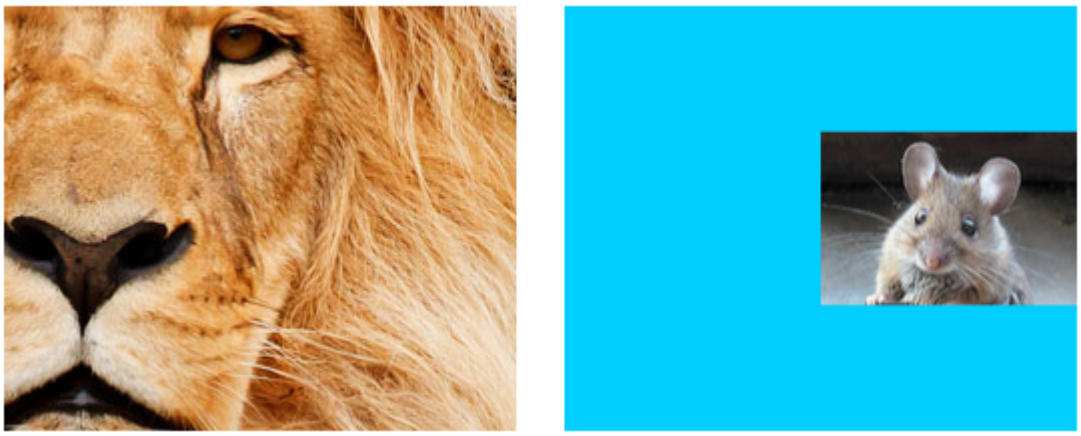
largeur de l'image ne sont pas étirées.

La gauche



Le bord gauche de l'image est centré verticalement à gauche de la vue et la longueur et la largeur de l'image ne sont pas étirées.

Droite



Le bord droit de l'image est centré verticalement à droite de la vue et la longueur et la largeur de l'image ne sont pas étirées.

En haut à gauche

Le coin supérieur gauche de l'image est placé dans le coin supérieur gauche de la vue. La longueur et la largeur de l'image ne sont pas étirées.

En haut à droite

Le coin supérieur droit de l'image est placé dans le coin supérieur droit de la vue. La longueur et la largeur de l'image ne sont pas étirées.

En bas à gauche

Le coin inférieur gauche de l'image est placé dans le coin inférieur gauche de la vue. La longueur et la largeur de l'image ne sont pas étirées.

En bas à droite

Le coin inférieur droit de l'image est placé dans le coin inférieur droit de la vue. La longueur et la largeur de l'image ne sont pas étirées.

Remarques

- Cet exemple vient d' ici .
- Si le contenu (dans notre cas, l'image) a la même taille que la vue (dans notre cas, la `UIImageView`), alors changer le mode de contenu ne fera aucune différence notable.
- Voir ceci et cette question pour une discussion sur les modes de contenu pour les vues autres que `UIImageView` .
- Dans Swift, pour définir le mode de contenu par programmation, procédez comme suit:

```
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
```

Lire UIImageView en ligne: https://riptutorial.com/fr/ios/topic/695/uiimageview

Chapitre 172: UIKit Dynamics avec UICollectionView

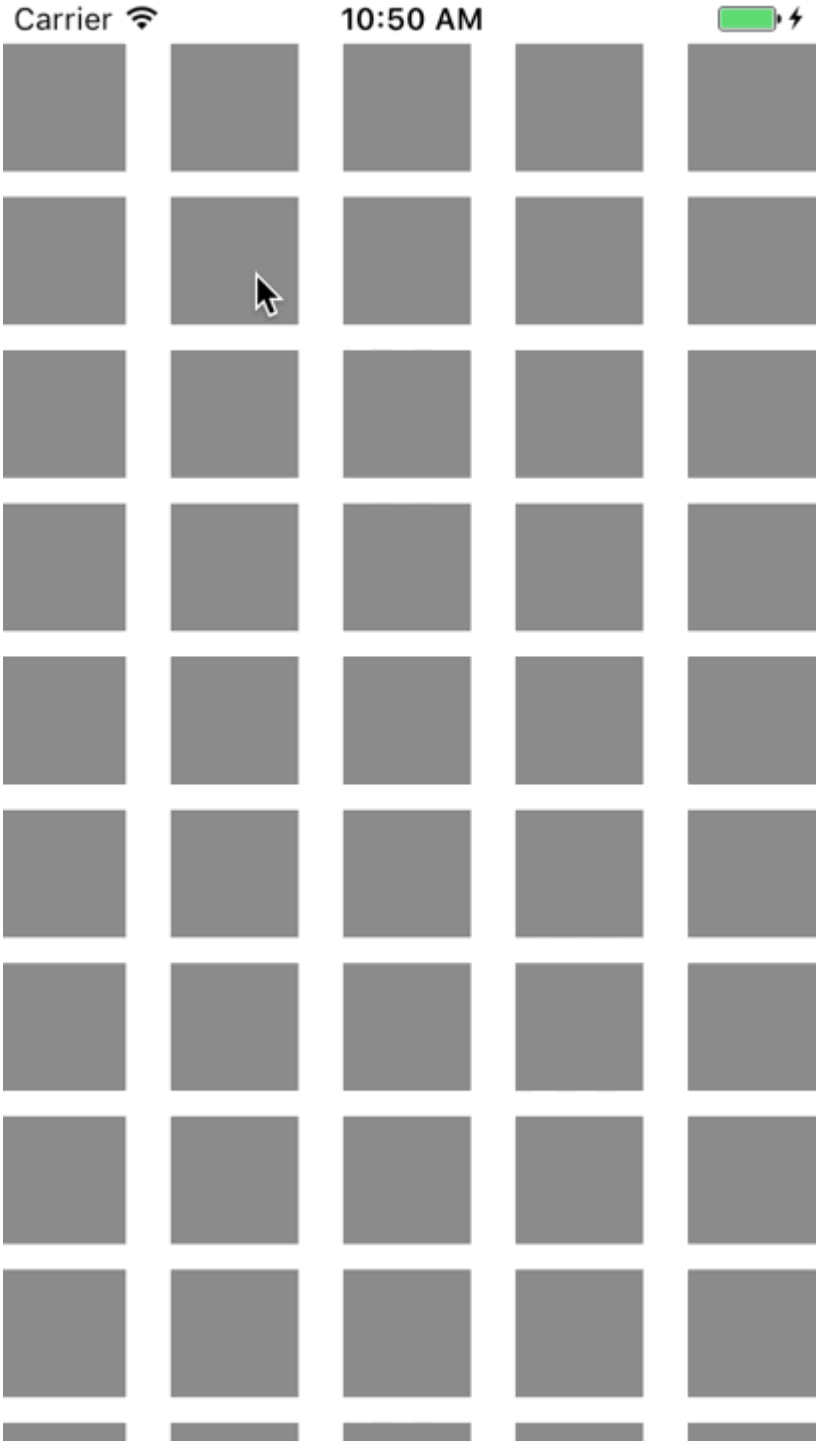
Introduction

UIKit Dynamics est un moteur physique intégré à UIKit. UIKit Dynamics propose un ensemble d'API offrant une interopérabilité avec `UICollectionView` et `UICollectionViewLayout`

Examples

Création d'un comportement de glisser personnalisé avec UIDynamicAnimator

Cet exemple montre comment créer un comportement de glissement personnalisé en sous-`UIDynamicBehavior` et en sous- `UICollectionViewFlowLayout` . Dans l'exemple, nous avons `UICollectionView` qui permet la sélection de plusieurs éléments. Puis, avec un `UIDynamicAnimator` long, ces éléments peuvent être déplacés dans une animation élastique et élastique pilotée par un `UIDynamicAnimator` .



Le comportement de glissement est généré en combinant un comportement de bas niveau qui ajoute un `UIAttachmentBehavior` aux angles pour un `UIDynamicItem` et un comportement de haut niveau qui gère le comportement de bas niveau pour un certain nombre de `UIDynamicItems` .

Nous pouvons commencer par créer ce comportement de bas niveau, nous appellerons `RectangleAttachmentBehavior`

Rapide

```
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
    }
}
```

Objectif 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
```

Ensuite, nous pouvons créer le comportement de haut niveau qui combinera un certain nombre de `RectangleAttachmentBehavior` .

Rapide

```
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) }
    }
}
```

Objectif 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
```

Maintenant, avec nos comportements en place, l'étape suivante consiste à les ajouter à notre vue de collection lorsque. Comme nous voulons normalement une disposition de grille standard, nous pouvons sous- `UICollectionViewFlowLayout` et modifier uniquement les attributs lors du glissement. Nous le faisons principalement en `layoutAttributesForElementsInRect` et en utilisant la `UIDynamicAnimator's` commodité `itemsInRect` .

Rapide

```
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()
    }
}
```

Objectif 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
```

Enfin, nous allons créer un contrôleur de vue qui créera notre `UICollectionView` et gérera notre long geste d' `UICollectionView` .

Rapide

```
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)!)
        }
    }
}
```

Objectif 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
```

Pour plus d'informations Session 2013 de la WWDC "Techniques avancées avec dynamique UIKit"

Lire UIKit Dynamics avec UICollectionView en ligne: https://riptutorial.com/fr/ios/topic/10079/uikit-dynamics-avec-uicollectionview

Chapitre 173: UILabel

Introduction

La classe UILabel implémente une vue texte en lecture seule. Vous pouvez utiliser cette classe pour dessiner une ou plusieurs lignes de texte statique, telles que celles que vous pourriez utiliser pour identifier d'autres parties de votre interface utilisateur. La classe UILabel de base prend en charge le style simple et complexe du texte d'étiquette. Vous pouvez également contrôler les aspects de l'apparence, par exemple si l'étiquette utilise une ombre ou dessine avec une surbrillance. Si nécessaire, vous pouvez personnaliser l'apparence de votre texte en procédant à une sous-classification.

Syntaxe

- UILabel.numberOfLines: Int // obtient ou définit le nombre maximum de lignes que l'étiquette peut avoir. 0 est illimité
- UILabel.text: String? // récupère ou définit le texte affiché par l'étiquette
- UILabel.textColor: UIColor! // obtenir ou définir la couleur du texte sur l'étiquette
- UILabel.tintColor: UIColor! // obtenir ou définir la couleur de teinte de l'étiquette
- UILabel.attributedText: NSAttributedString? // récupère ou définit le texte attribué de l'étiquette
- UILabel.font: UIFont! // récupère ou définit la police du texte sur l'étiquette
- UILabel.textAlignment: NSTextAlignment // obtient ou définit l'alignement du texte

Remarques

Les UILabels sont des vues pouvant être utilisées pour afficher une ou plusieurs lignes de texte. Il contient plusieurs manières de styliser du texte, telles que des ombres, des couleurs de texte et des polices.

Les UILabels peuvent également afficher des chaînes attribuées, qui sont du texte + balisage en ligne pour appliquer des styles à des parties du texte.

UILabel n'est pas conforme au protocole UIAppearance. Par conséquent, vous ne pouvez pas utiliser les méthodes proxy UIAppearance pour personnaliser l'apparence de UILabels. Voir cette discussion pour plus.

Référence du développeur Apple ici

Examples

Modification du texte dans une étiquette existante

`UILabel` pouvez modifier le texte d'un `UILabel` existant en accédant et en modifiant la propriété `text`

de l' `UILabel` . Cela peut être fait directement en utilisant des littéraux `String` ou indirectement en utilisant des variables.

Définition du texte avec des littéraux de `String`

Rapide

```
label.text = "the new text"
```

Objectif c

```
// Dot Notation
label.text = @"the new text";

// Message Pattern
[label setText:@"the new text"];
```

Définition du texte avec une variable

Rapide

```
let stringVar = "basic String var"
label.text = stringVar
```

Objectif c

```
NSString * stringVar = @"basic String var";

// Dot Notation
label.text = stringVar;

// Message Pattern
[label setText: stringVar];
```

Couleur du texte

Vous pouvez utiliser la propriété `textColor` l'étiquette pour appliquer une couleur de texte au texte entier de l'étiquette.

Rapide

```
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)
```

Objectif 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];
```

Appliquer une couleur de texte à une partie du texte

Vous pouvez également varier la couleur du texte (ou d'autres attributs) de parties du texte en utilisant `NSAttributedString` :

Objectif 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;
```

Rapide

```
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
```

Alignement du texte

Rapide

```
label.textAlignment = NSTextAlignment.left
//or the shorter
label.textAlignment = .left
```

Toute valeur dans le `NSTextAlignment` ENUM est valide: `.left` , `.center` , `.right` , `.justified` , `.natural`

Objectif c

```
label.textAlignment = NSTextAlignmentLeft;
```

Toute valeur de l'énumération `NSTextAlignment` est valide: `NSTextAlignmentLeft` , `NSTextAlignmentCenter` , `NSTextAlignmentRight` , `NSTextAlignmentJustified` , `NSTextAlignmentNatural`

L'alignement vertical dans `UILabel` n'est pas pris en charge: alignez verticalement le texte sur le dessus dans un UILabel

Créer un UILabel

Avec un cadre

Lorsque vous connaissez les dimensions exactes que vous souhaitez définir pour votre étiquette, vous pouvez initialiser une `UILabel` avec un cadre `CGRect` .

Rapide

```
let frame = CGRect(x: 0, y: 0, width: 200, height: 21)
let label = UILabel(frame: frame)
view.addSubview(label)
```

Objectif c

```
CGRect frame = CGRectMake(0, 0, 200, 21);
UILabel *label = [[UILabel alloc] initWithFrame:frame];
[view addSubview:label];
```

Avec mise en page automatique

Vous pouvez ajouter des contraintes sur un `UILabel` lorsque vous souhaitez `UILabel` calcule dynamiquement son cadre à l'exécution.

Rapide

```
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)
])
```

Objectif c

```
UILabel *label = [[UILabel alloc] init];
```

Avec 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 documentation de VFL peut être trouvée ici

Une fois l'étiquette créée, veillez à définir les dimensions via Mise en forme automatique. Xcode affichera les erreurs s'il est mal fait.

Avec Interface Builder

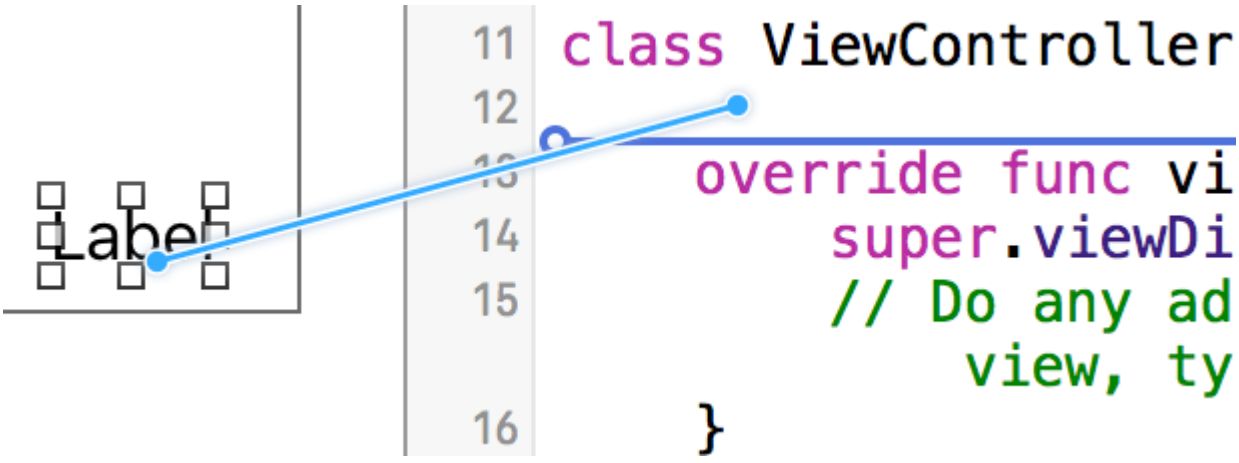
Vous utilisez également Interface Builder pour ajouter un `UILabel` à votre fichier `Storyboard` ou `.xib` en faisant glisser une `Label` du panneau Bibliothèque d'objets et en la déposant dans une vue du canevas:

Au lieu de spécifier un cadre (position et taille) pour un `UILabel` programme, un `Storyboard` ou un `.xib` vous permet d'utiliser Auto Layout pour ajouter des contraintes au contrôle.

Pour accéder à cette étiquette créée à partir de `storyboard` ou de `xib` créez un IBOutlet de cette étiquette.

Liaison entre Interface Builder et View Controller

Une fois que vous avez ajouté un `UILabel` à votre `Storyboard` ou `.xib` le fichier, vous pouvez le lier à votre code en appuyant sur `Control ^` , puis en faisant glisser la souris entre `UILabel` et votre `ViewController` . ont le même effet.



Dans la boîte de dialogue des propriétés, vous pouvez définir le nom de `UILabel` et le définir comme `strong` ou `weak` . Pour plus d'informations sur `strong` et `weak` , voir ceci ,

L'autre solution consiste à faire en sorte que le point de vente soit programmé comme suit:

Rapide

```
@IBOutlet weak var nameLabel : UILabel!
```

Objectif c

```
@property (nonatomic, weak) IBOutlet UILabel *nameLabel;
```

Définir la police

Rapide

```
let label = UILabel()
```

Objectif c

```
UILabel *label = [[UILabel alloc] init];
or
UILabel *label = [UILabel new]; // convenience method for calling alloc-init
```

Changer la taille de la police par défaut

Rapide

```
label.font = UIFont.systemFontOfSize(17)
```

Swift 3

```
label.font = UIFont.systemFont(ofSize: 17)
```

Objectif c

```
label.font = [UIFont systemFontOfSize:17];
```

Utiliser un poids de police spécifique

iOS 8.2

Rapide

```
label.font = UIFont.systemFontOfSize(17, weight: UIFontWeightBold)
```

Swift3

```
label.font = UIFont.systemFont(ofSize: 17, weight: UIFontWeightBold)
```

Objectif c

```
label.font = [UIFont systemFontOfSize:17 weight:UIFontWeightBold];
```

iOS 8.2

Rapide

```
label.font = UIFont.boldSystemFontOfSize(17)
```

Swift3

```
label.font = UIFont.boldSystemFont(ofSize: 17)
```

Objectif c

```
label.font = [UIFont boldSystemFontOfSize:17];
```

Utilisez un style de texte de type dynamique.

La taille de la police et du point dépendra de la taille de lecture préférée de l'utilisateur.

Rapide

```
label.font = UIFont.preferredFontForTextStyle(UIFontTextStyleBody)
```

Swift 3

```
label.font = UIFont.preferredFont(forTextStyle: .body)
```

Objectif c

```
label.font = [UIFont preferredFontForTextStyle:UIFontTextStyleBody];
```

Utilisez une police différente tout à fait

Rapide

```
label.font = UIFont(name: "Avenir", size: 15)
```

Objectif c

```
label.font = [UIFont fontWithName:@"Avenir" size:15];
```

Remplacer la taille de la police

Un moyen de définir la taille de la police sans connaître la famille de polices consiste à utiliser la propriété **font** de `UILabel` .

Rapide

```
label.font = label.font.fontWithSize(15)
```

Swift 3

```
label.font = label.font.withSize(15)
```

Objectif c

```
label.font = [label.font fontWithSize:15];
```

Utilisez la police personnalisée Swift

Reportez-vous à ce lien

Nombre de lignes

Lorsque vous créez une étiquette et définissez son texte pour qu'il soit plus qu'une seule ligne pouvant être affichée, celle-ci sera tronquée et vous ne verrez qu'une seule ligne de texte se terminant par trois points (...). En effet, une propriété appelée `numberOfLines` est définie sur 1 et, par conséquent, une seule ligne sera affichée. C'est une erreur courante dans le traitement de `UILabel` , et beaucoup de gens le considèrent comme un bogue, ou ils peuvent utiliser plus d'une étiquette pour afficher plus qu'une ligne de texte, mais en éditant cette propriété, on peut dire à un

`UILabel` accepter jusqu'au nombre de lignes spécifié. Par exemple, si cette propriété est définie sur 5, l'étiquette peut afficher 1, 2, 3, 4 ou 5 lignes de données.

Définition de la valeur par programmation

Pour définir cette propriété, affectez-lui simplement un nouvel entier:

Rapide

```
label.numberOfLines = 2
```

Objectif c

```
label.numberOfLines = 2;
```

Remarque

Il est possible de définir cette propriété sur 0. Cependant, cela ne signifie pas qu'elle n'acceptera aucune ligne, mais que l'étiquette peut avoir autant de lignes que nécessaire (aka "Infinity"):

Rapide

```
label.numberOfLines = 0
```

Objectif c

```
label.numberOfLines = 0;
```

Remarque

Si l'étiquette a une contrainte de hauteur, la contrainte sera respectée. Dans ce cas, `label.numberOfLines = 0` peut ne pas fonctionner comme prévu.

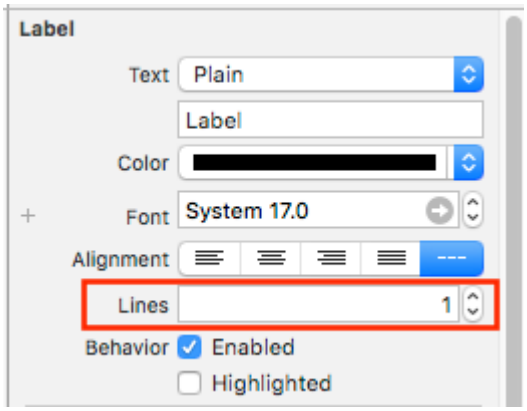
Remarque

Pour un texte multi-lignes plus complexe, UITextView peut être plus adapté. *

Définition de la valeur dans le générateur d'interface

Au lieu de définir `numberOfLines` programmation, vous pouvez utiliser un `Storyboard` ou un `.xib` et définir la propriété `numberOfLines` . De cette façon, nous obtenons les mêmes résultats que le code ci-dessus.

Comme ci-dessous:



Taille pour s'adapter

Supposons que vous ayez un `UILabel` sur votre `storyboard` et que vous ayez créé un `IBOutlet` dans `ViewController.swift` / `ViewController.m` et l' `ViewController.m` nommé `labelOne` .

Pour rendre les modifications facilement visibles, modifiez les `textColor` `backgroundColor` et `textColor` de labelOne dans la méthode `viewDidLoad` :

La fonction `sizeToFit` est utilisée lorsque vous souhaitez redimensionner automatiquement une étiquette en fonction du contenu stocké.

Rapide

```
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()
```

Objectif c

```
labelOne.backgroundColor = [UIColor blueColor];
labelOne.textColor = [UIColor whiteColor];
labelOne.text = @"Hello, World!";
[labelOne sizeToFit];
```

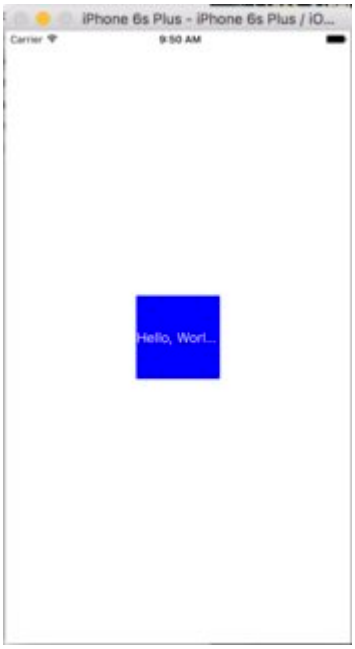
La sortie du code ci-dessus est:

Comme vous pouvez le voir, il n'y a pas de changement car le texte est parfaitement adapté à labelOne. sizeToFit ne modifie que le cadre de l'étiquette.

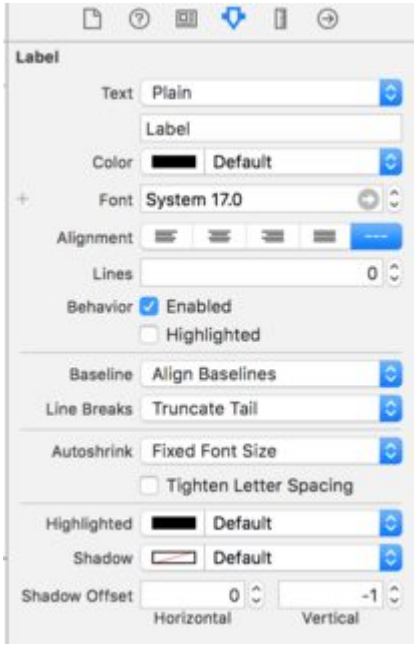
Changeons le texte en un peu plus long:

```
labelOne.text = "Hello, World! I’m glad to be alive!"
```

Maintenant, labelOne ressemble à ceci:



Même appeler `sizeToFit` ne change rien. En effet, par défaut, le NumberOfLines indiqué par UILabel est défini sur 1. Changeons-le en zéro sur le storyboard:



Cette fois, quand nous exécutons l'application, labelOne apparaît correctement:

La propriété `numberOfLines` peut également être modifiée dans le fichier `ViewController` :

```
// Objective-C
labelOne.numberOfLines = 0;

// Swift
labelOne.numberOfLines = 0
```

Couleur de fond

Rapide

```
label.backgroundColor = UIColor.redColor()

label.backgroundColor = .redColor()
```

Swift 3

```
label.backgroundColor = UIColor.red
```

Objectif c

```
label.backgroundColor = [UIColor redColor];
```

Ajouter des ombres au texte

Rapide

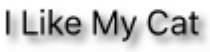
```
label1.layer.shadowOffset = CGSize(width: 3, height: 3)
label1.layer.shadowOpacity = 0.7
label1.layer.shadowRadius = 2
```

Swift 3

```
label1.layer.shadowOffset = CGSize(width: 3, height: 3)
label1.layer.shadowOpacity = 0.7
label1.layer.shadowRadius = 2
```

Objectif c

```
label1.layer.shadowOffset = CGSizeMake(3, 3);
label1.layer.shadowOpacity = 0.7;
label1.layer.shadowRadius = 2;
```



Hauteur variable en utilisant des contraintes

Vous pouvez créer une `UILabel` avec une hauteur dynamique en utilisant la mise en page automatique.

Vous devez définir le `numberOfLines` à zéro (0) et ajouter une hauteur minimale en définissant une contrainte avec une relation de type `.GreaterThanOrEqual` sur l'attribut `.Height`

iOS 6

Rapide

```
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

Rapide

```
label.numberOfLines = 0
label.translatesAutoresizingMaskIntoConstraints = false
label.heightAnchor.constraintGreaterThanOrEqualToConstant(20).active = true
```

LineBreakMode

En utilisant du code

```
UILabel.lineBreakMode: NSLineBreakMode
```

Rapide

```
label.lineBreakMode = .ByTruncatingTail
```

- .ByWordWrapping
- .ByCharWrapping
- .ByClipping
- .ByTruncatingHead
- .ByTruncatingTail
- .ByTruncatingMiddle

Swift 3

```
label.lineBreakMode = .byTruncatingTail
```

- `.byWordWrapping`
- `.byCharWrapping`
- `.byClipping`
- `.byTruncatingHead`
- `.byTruncatingTail`
- `.byTruncatingMiddle`

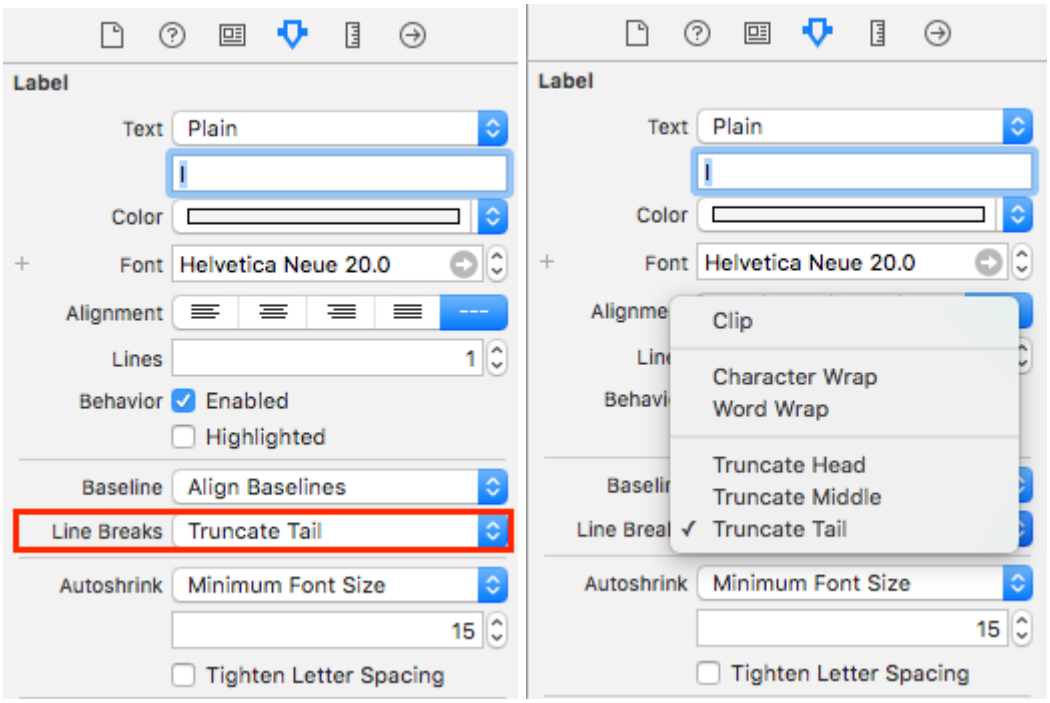
Objectif c

```
[label setLineBreakMode:NSLineBreakByTruncatingTail];
```

- `NSLineBreakByWordWrapping`
- `NSLineBreakByCharWrapping`
- `NSLineBreakByClipping`
- `NSLineBreakByTruncatingHead`
- `NSLineBreakByTruncatingTail`
- `NSLineBreakByTruncatingMiddle`

Utiliser le storyboard

Cela peut également être défini dans l'inspecteur d'attributs d'une étiquette UIL:



Les constantes

- Word Wrapping - le retour à la ligne se produit aux limites des mots, à moins que le mot lui-même ne tienne sur une seule ligne
- Char Wrapping - l'emballage commence avant le premier caractère qui ne correspond pas

- Clipping - les lignes ne sont simplement pas dessinées au-delà du bord du conteneur de texte
- Troncature de la tête - la ligne est affichée de sorte que la fin s'insère dans le conteneur et que le texte manquant au début de la ligne est indiqué par un glyphe de points de suspension
- Tronquer la queue - la ligne est affichée de sorte que le début s'insère dans le conteneur et le texte manquant à la fin de la ligne est indiqué par un glyphe de points de suspension
- Troncature du milieu - la ligne est affichée de sorte que le début et la fin tiennent dans le conteneur et que le texte manquant au milieu soit indiqué par un glyphe à points de suspension

Calculer les limites de contenu (c.-à-d. Hauteurs de cellules dynamiques)

Un cas d'utilisation courant pour vouloir calculer la trame qu'une étiquette va prendre est de dimensionner les cellules de vue de table de manière appropriée. La méthode recommandée consiste à utiliser la méthode `NSString boundingRectWithSize:options:attributes:context:`

`options` prend les `options` dessin de chaîne:

- `NSStringDrawingUsesLineFragmentOrigin` doit être utilisé pour les étiquettes comportant plusieurs lignes
- `NSStringDrawingTruncatesLastVisibleLine` devrait être ajouté en utilisant le `|` opérateur s'il y a un nombre maximum de lignes

`attributes` est un `NSDictionary` d'attributs qui `NSDictionary` chaînes attribuées (liste complète: Apple Docs), mais les facteurs qui affectent la hauteur sont les suivants:

- **NSFontAttributeName** : Très important, la taille et la famille de polices constituent une partie critique de la taille affichée de l'étiquette.

- **NSParagraphStyleAttributeName** : permet de personnaliser l'affichage du texte. Cela inclut l'interligne, l'alignement du texte, le style de troncature et quelques autres options. Si vous n'avez modifié explicitement aucune de ces valeurs, vous ne devriez pas avoir à vous soucier de cela, mais cela peut être important si vous modifiez certaines valeurs sur IB.

`context` doit être `nil` puisque le cas d'utilisation principal de `NSStringDrawingContext` permet à la police de redimensionner pour correspondre à un rect spécifié, ce qui ne devrait pas être le cas si nous calculons une hauteur dynamique.

Objectif 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
}
```

À l'inverse, si vous avez défini un nombre maximum de lignes, vous devrez d'abord calculer la hauteur d'une seule ligne pour vous assurer de ne pas obtenir une valeur supérieure à la taille autorisée:

```
    // 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;
```

Label cliquable

> **REMARQUE:** Dans la plupart des cas, il est préférable d'utiliser un `UIButton` au lieu de créer un `UILabel` vous pouvez appuyer. N'utilisez cet exemple que si vous êtes sûr que vous ne voulez pas utiliser un `UIButton` pour une raison quelconque.

1. Créer une étiquette
2. Activer l'interaction avec l'utilisateur
3. Ajouter `UITapGestureRecognizer`

La clé pour créer un `UILabel` cliquable `UILabel` à activer l'interaction de l'utilisateur.

Rapide

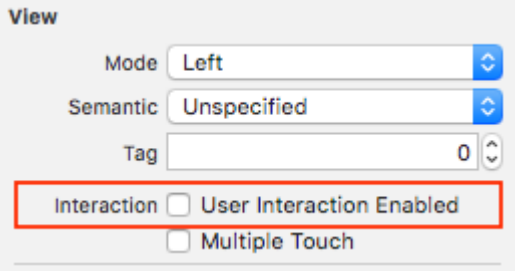
```
let label = UILabel()
label.userInteractionEnabled = true

let gesture = UITapGestureRecognizer(target: self, action: #selector(labelClicked(_:)))
label.addGestureRecognizer(gesture)
```

Objectif c

```
UILabel *label = [[UILabel alloc] init];
[label setUserInteractionEnabled:YES];

UITapGestureRecognizer* gesture = [[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(labelClicked:)];
[label addGestureRecognizer:gesture];
```

Définition de "userInteractionEnabled" dans l'inspecteur des attributs du storyboard

Au lieu d'utiliser le code, vous pouvez sélectionner le UILabel dans le storyboard et cocher l'option:



Cadre d'étiquette dynamique de longueur de texte inconnue

Parfois, nous devons redimensionner un UILabel en fonction d'un contenu dynamique dont la longueur du texte est inconnue. Dans cet exemple, la largeur de UILabel est fixée à 280 points et la hauteur est infinie, disons 9999. Estimation de l'image par rapport au style de texte et à maximumLabelSize.

Objectif 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;
```

Rapide

```
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)
```

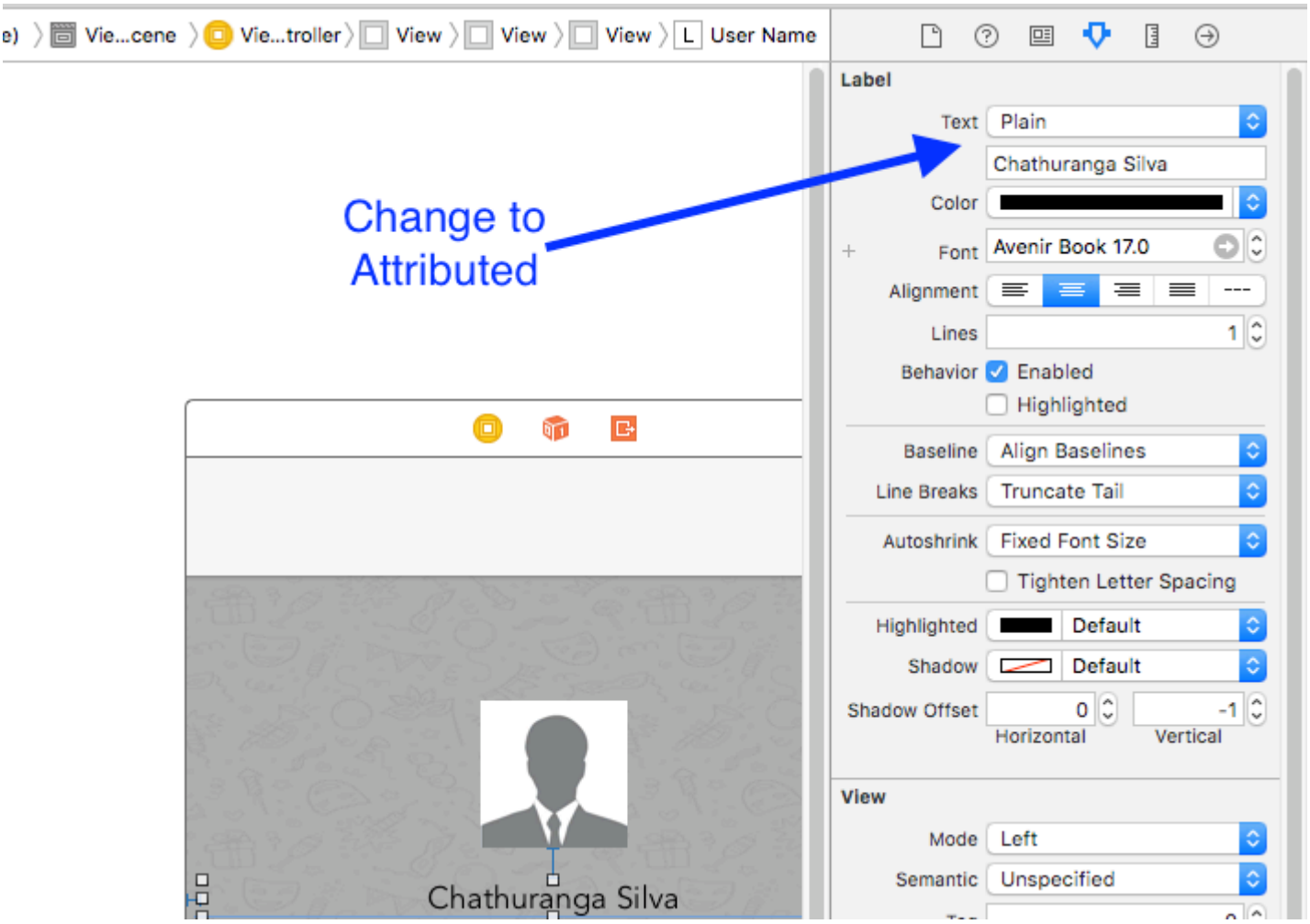
```
// create a frame that is filled with the UILabel frame data
var newFrame: CGRect = label.frame
// resizing the frame to calculated size
newFrame.size.height = expectedLabelSize.height
// put calculated frame into UILabel frame
label.frame = newFrame
```

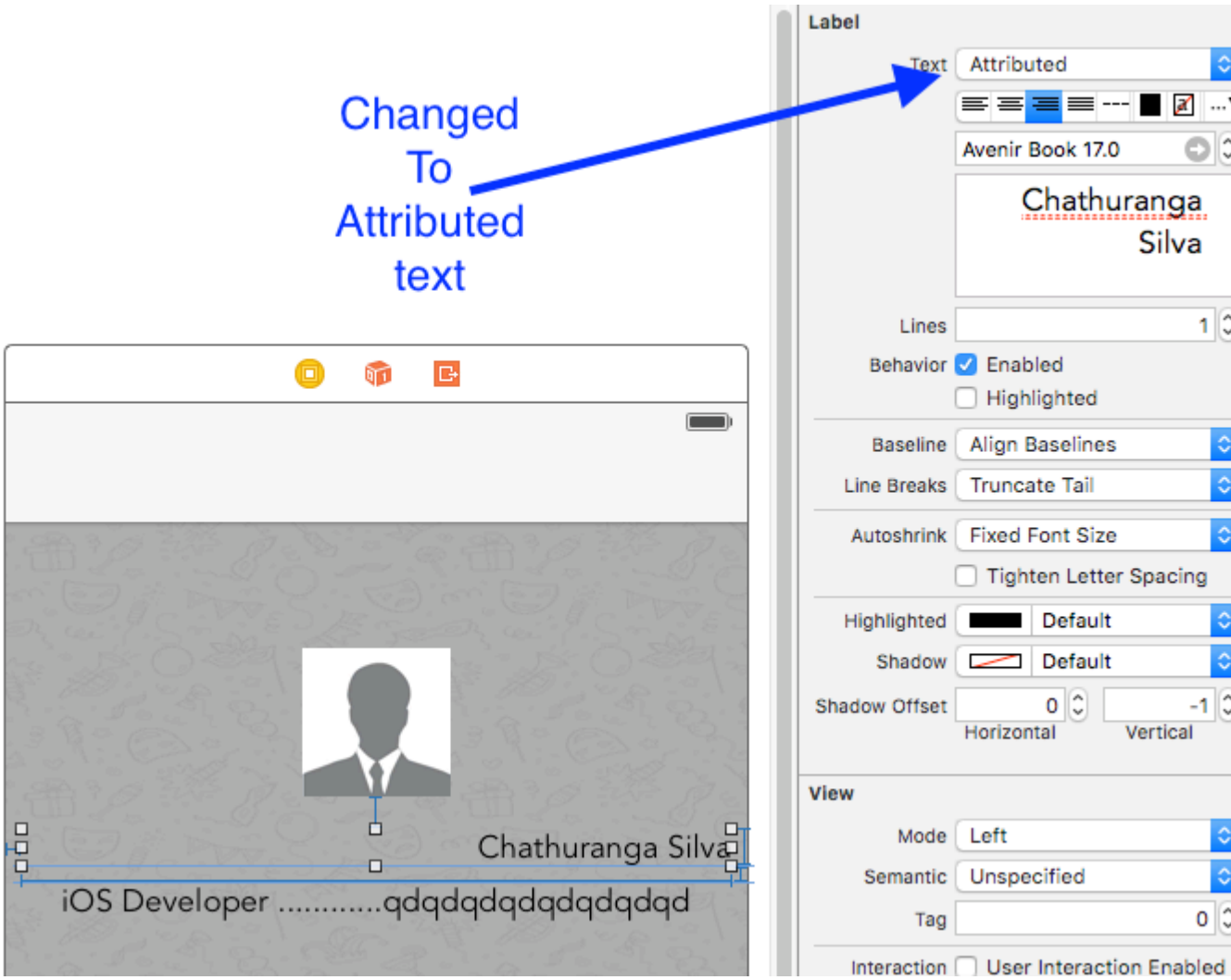
Texte d'attribut d'étiquette

01. Texte souligné: - Ligne simple / double, barré: - Ligne simple / double

Étape 1

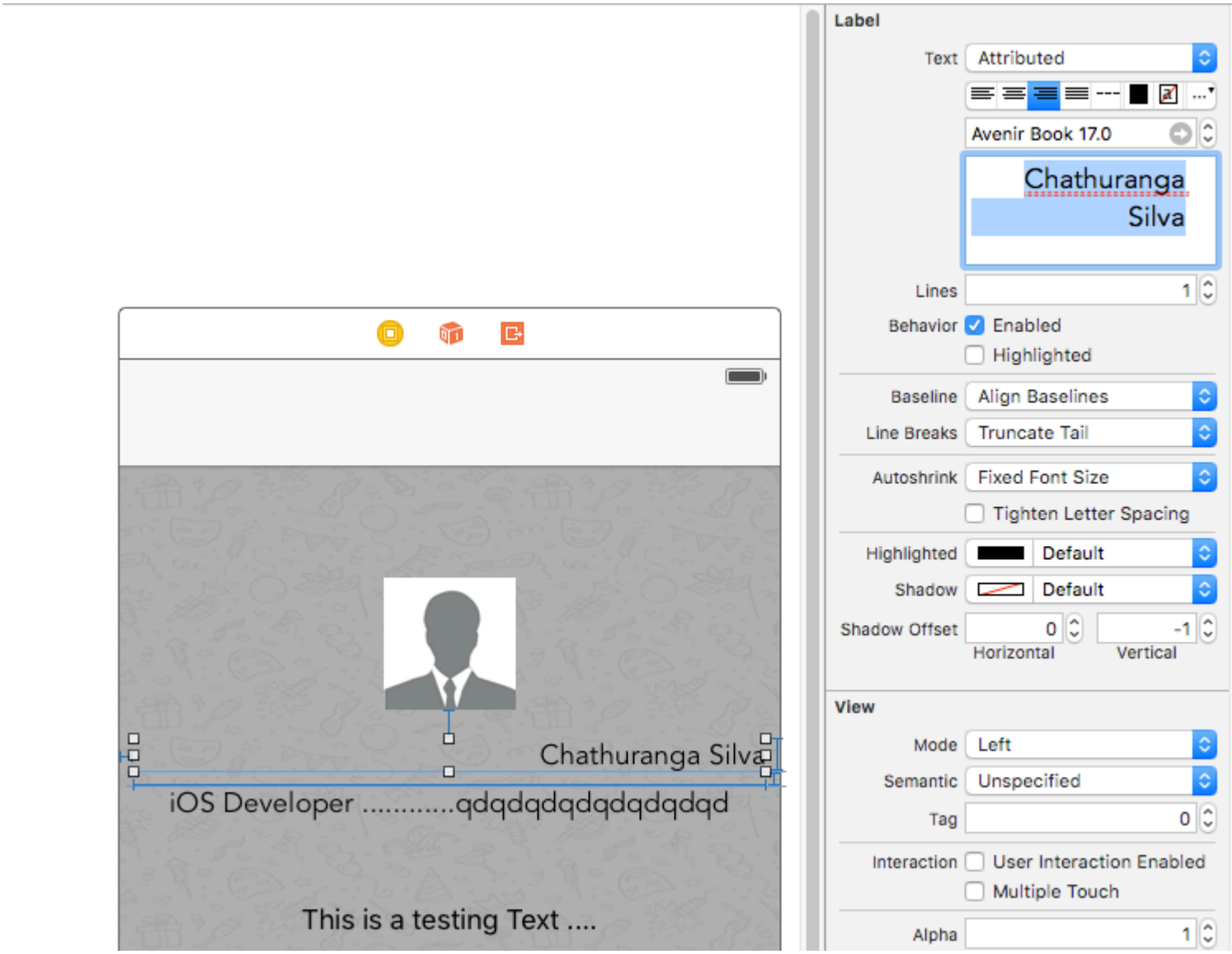
Sélectionnez l'étiquette et changez le type d'étiquette Plain en Attribué





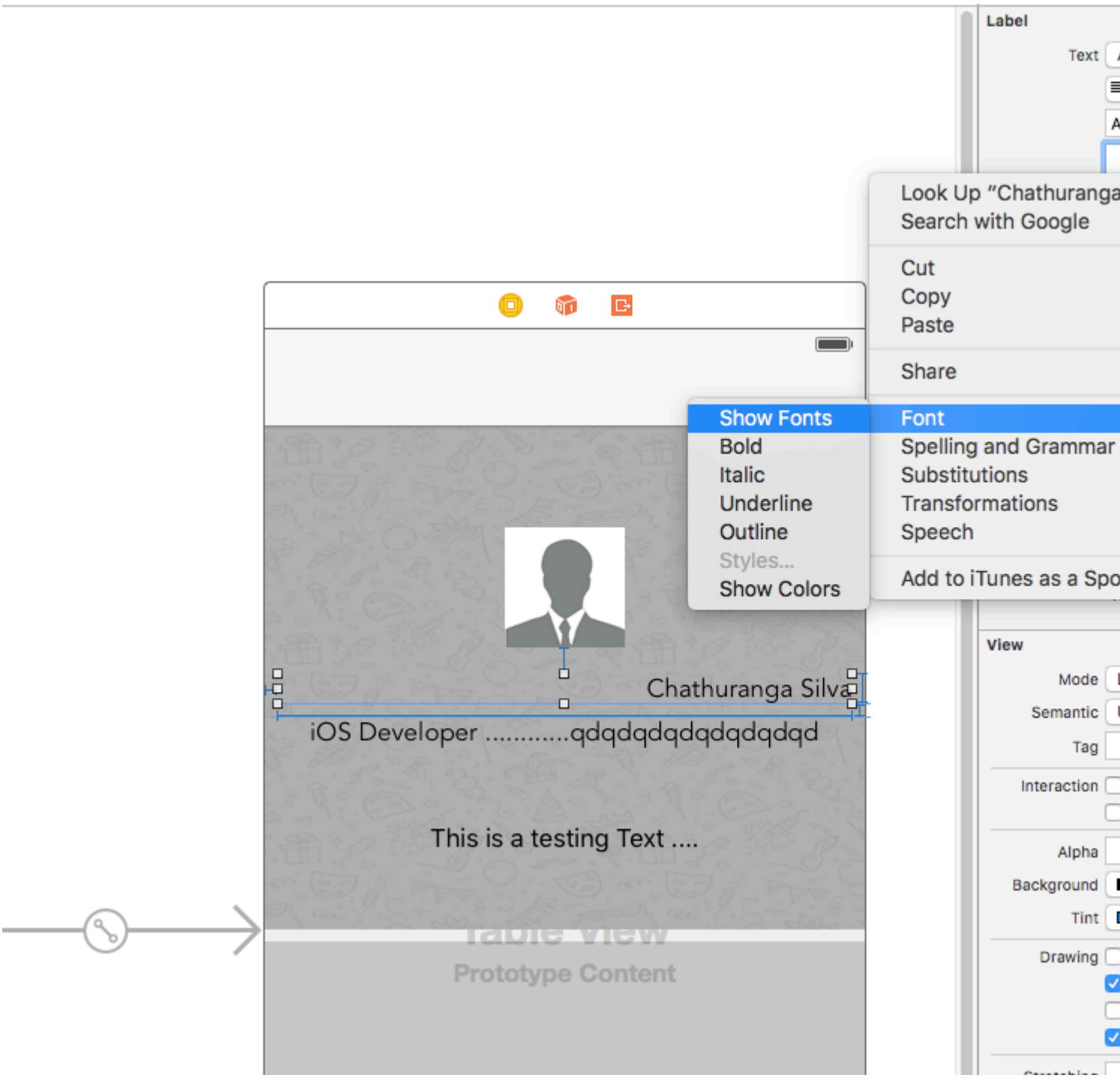
Étape 2

Cliquez sur le texte de l'étiquette et cliquez avec le bouton droit



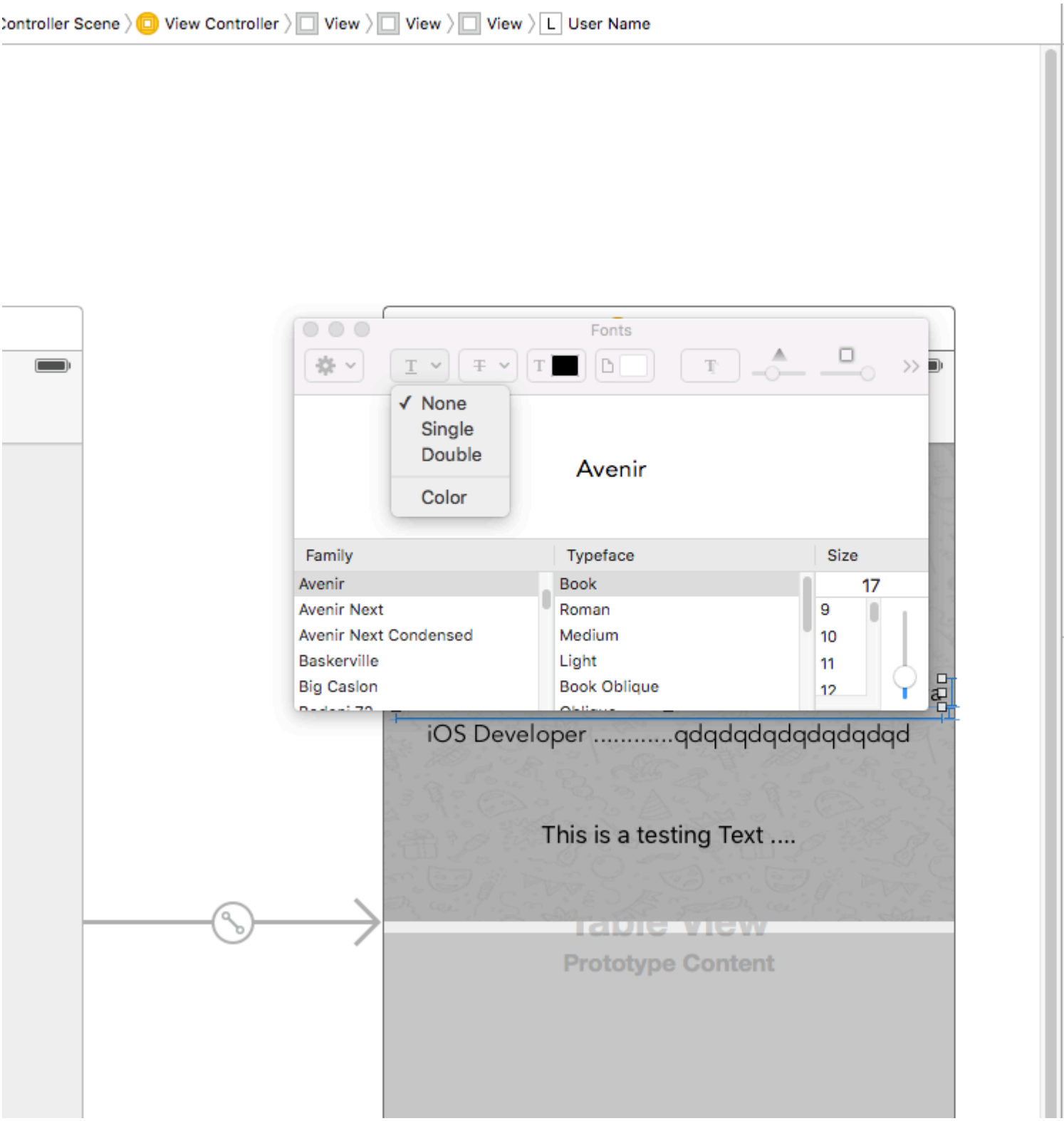
Étape 3

Puis cliquez sur Police -> Afficher les polices

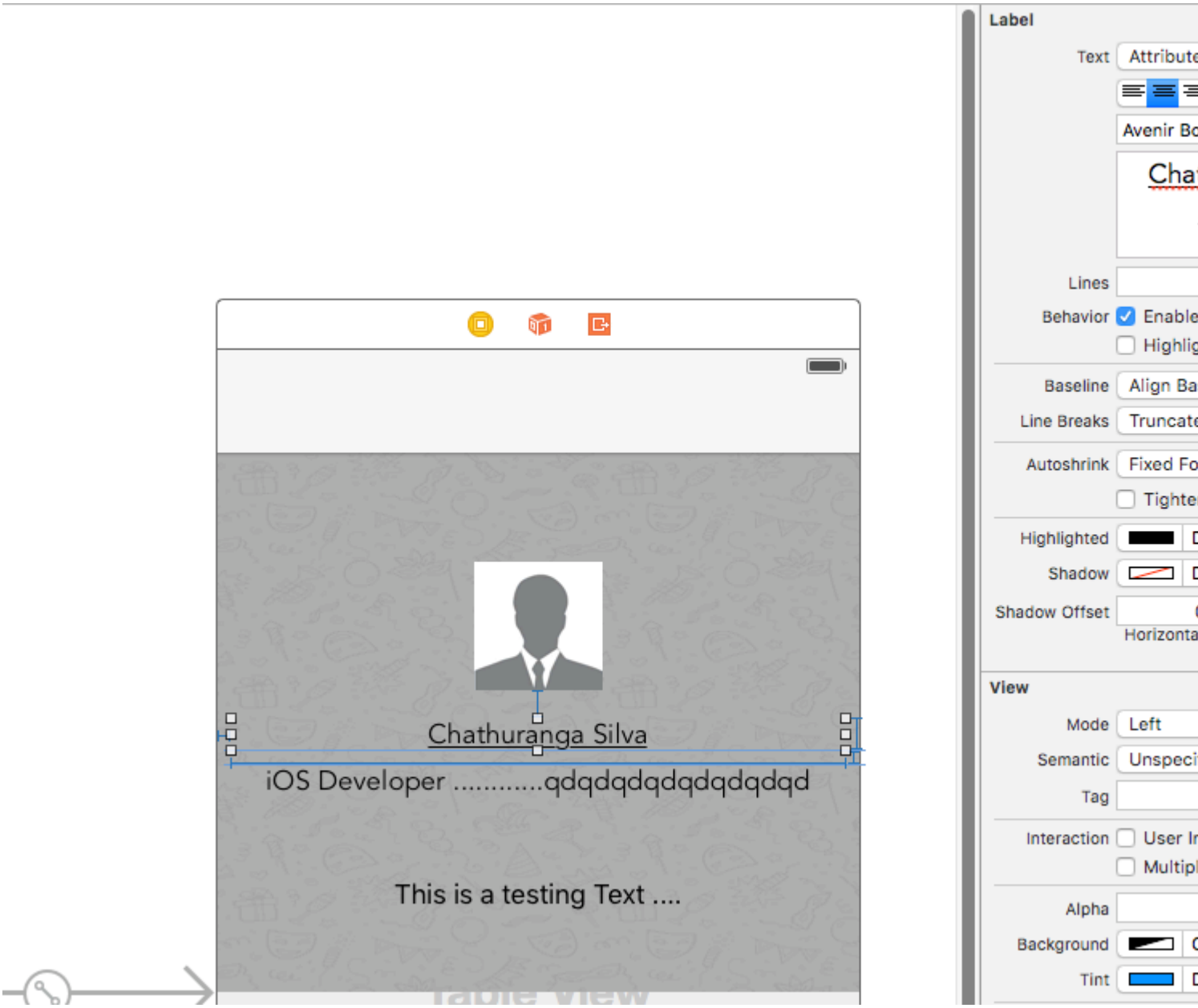


Étape 4

Ensuite, la vue de la police apparaîtra et cliquera sur le bouton de soulignement pour souligner le texte ou cliquer sur le bouton barré pour que le texte soit barré.Et sélectionnez une ligne simple ou double.

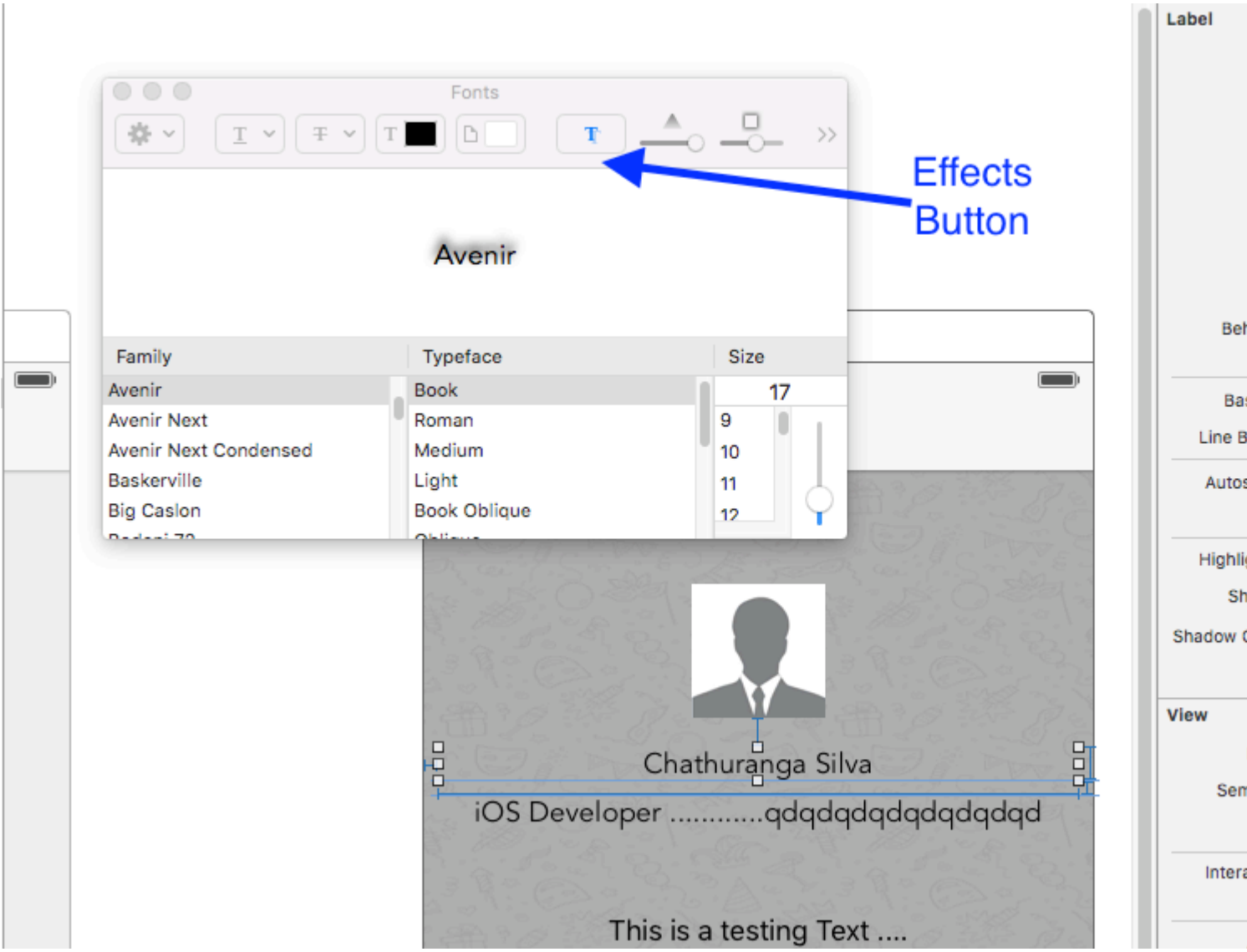


Enfin, cliquez sur Entrée et l'étiquette sera soulignée ou barrée selon votre sélection.

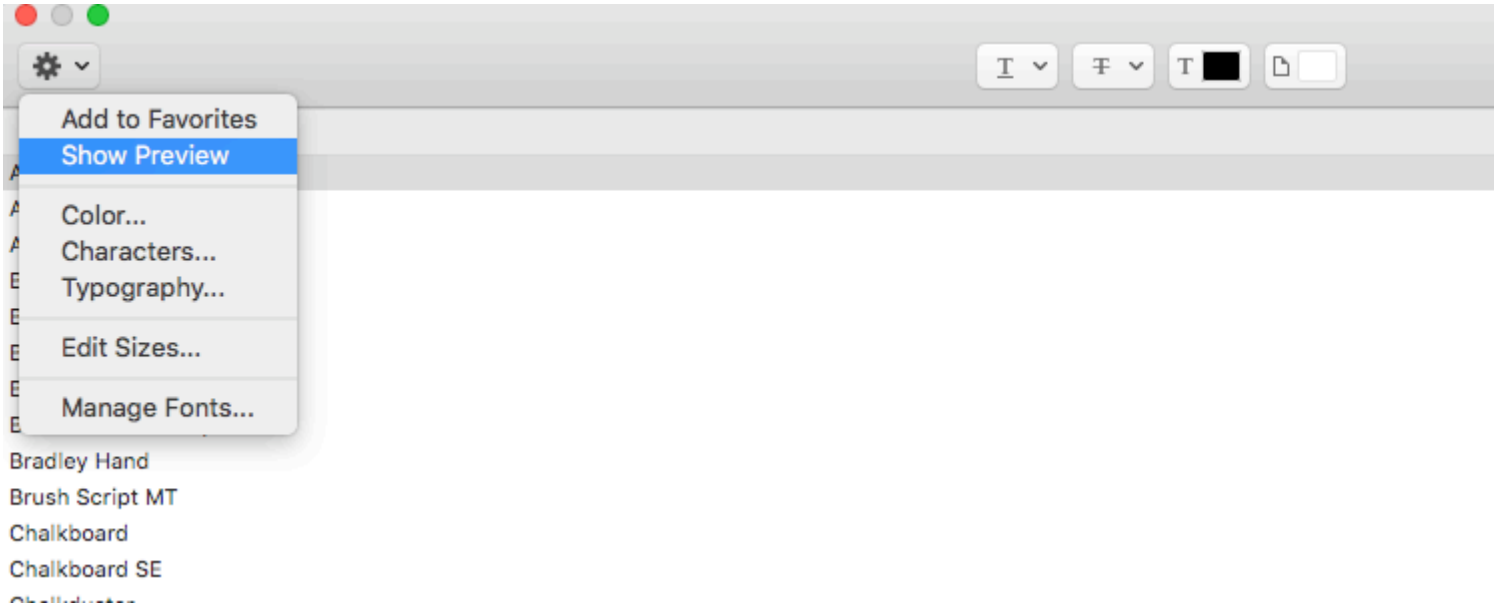


02. Ajouter du texte shaddow / effets de flou d'arrière-plan

Obtenez la vue Police comme décrit ci-dessus et cliquez sur le bouton Effets.



Si vous ne voyez pas l'aperçu, cliquez sur l'image affichée dans les paramètres



Enfin, changez shaddow et décalez selon vos préférences.

Justifier le texte

Rapide

```
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)
```

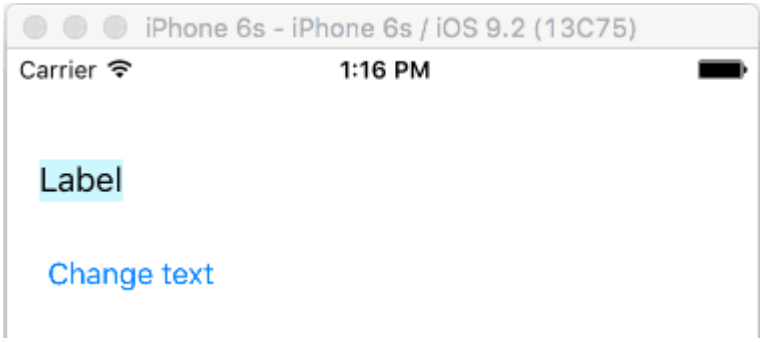
Objectif c

```
  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];
```

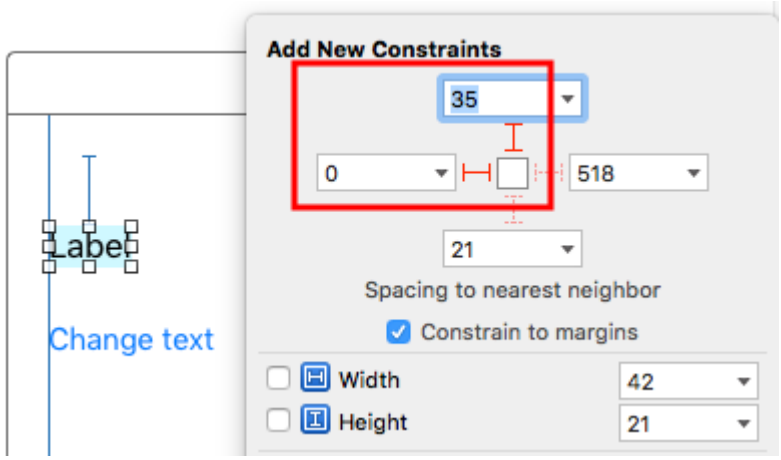
Etiquette de taille automatique pour adapter le texte

Cet exemple montre comment la largeur d'une étiquette peut être redimensionnée automatiquement lorsque le contenu du texte change.



Pin les bords gauche et supérieur

Utilisez simplement la mise en page automatique pour ajouter des contraintes à épingler les côtés gauche et supérieur de l'étiquette.



Après cela, il sera automatiquement redimensionné.

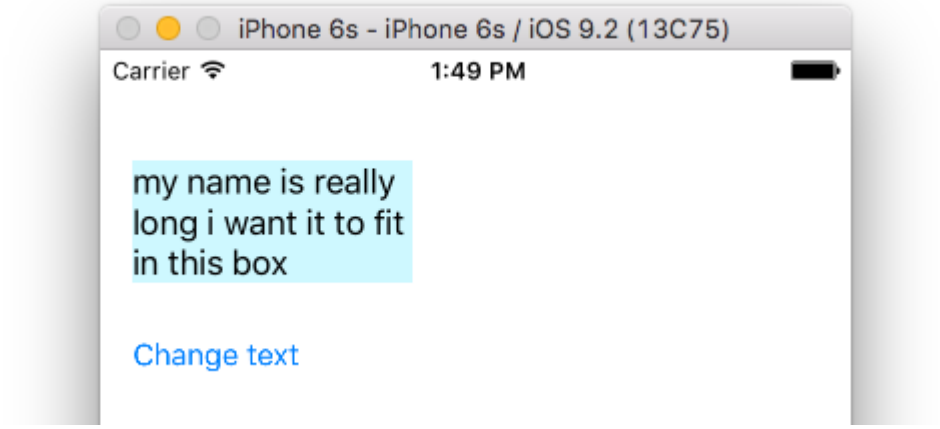
Remarques

- Cet exemple provient de cette réponse Stack Overflow .
- N'ajoutez pas de contraintes pour la largeur et la hauteur. Les étiquettes ont une taille *intrinsèque* basée sur leur contenu textuel.
- Pas besoin de définir `sizeToFit` lors de l'utilisation de la mise en page automatique. Le code complet pour l'exemple de projet est ici:

```
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"
    }
}
```

- Cette méthode peut également être utilisée pour espacer correctement plusieurs étiquettes horizontalement, comme dans cet exemple .

- Si vous souhaitez que votre étiquette soit alignée, définissez le nombre de lignes sur 0 dans IB et ajoutez `myLabel.preferredMaxLayoutWidth = 150 // or whatever` dans le code. (Le bouton est également épinglé au bas de l'étiquette pour qu'il se déplace vers le bas lorsque la hauteur de l'étiquette augmente.)



Obtenir la taille de UILabel strictement basée sur son texte et sa police

`NSString` fournit la méthode `boundingRectWithSize` qui peut être utilisée pour prédire la taille CGS résultante d'un `UILabel` fonction de son texte et de sa police sans qu'il soit nécessaire de créer un `UILabel`

Objectif c

```
[[text boundingRectWithSize:maxSize options:(NSStringDrawingTruncatesLastVisibleLine |
NSStringDrawingUsesLineFragmentOrigin) attributes:@{NSFontAttributeName: fontName}
context:nil] size];
```

Rapide

```
let nsText = text as NSString?
nsText?.boundingRectWithSize(maxSize, options: [.TruncatesLastVisibleLine,
.UsesLineFragmentOrigin], attributes: [NSFontAttributeName: fontName], context: nil).size
```

Rapide

Créer une étiquette et une étiquette Ajoutez le code ci-dessous où vous allez assigner du texte à l'étiquette.

```
@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
```

Note: "40" est l'espace des côtés gauche et droit de l'écran.

Couleur de texte surlignée et mise en évidence

Objectif c

```
UILabel *label = [[UILabel alloc] init];
label.highlighted = YES;
label.highlightedTextColor = [UIColor redColor];
```

Rapide

```
let label = UILabel()
label.highlighted = true
label.highlightedTextColor = UIColor.redColor()
```

Swift 3

```
let label = UILabel()
label.isHighlighted = true
label.highlightedTextColor = UIColor.red
```

Lire UILabel en ligne: https://riptutorial.com/fr/ios/topic/246/uilabel

Chapitre 174: UILocalNotification

Introduction

Les notifications locales permettent à votre application d'informer l'utilisateur sur le contenu qui ne nécessite pas l'utilisation d'un serveur.

Contrairement aux notifications à distance qui sont déclenchées à partir d'un serveur, les notifications locales sont planifiées et déclenchées dans une application. Les notifications en général visent à accroître l'interaction des utilisateurs avec l'application, en invitant ou en incitant l'utilisateur à ouvrir et à interagir avec l'application.

UILocalNotification est obsolète dans iOS 10. Utilisez plutôt le framework UserNotifications.

Remarques

Ne confondez pas UILocalNotification avec les notifications push. UILocalNotification est déclenché par votre appareil et, une fois programmé, est copié sur le système.

Liens:

- Référence de la classe UILocalNotification
- UILocalNotification sur le dépassement de pile

Examples

Planification d'une notification locale

Assurez-vous que vous voyez S'inscrire aux notifications locales pour que cela fonctionne:

Rapide

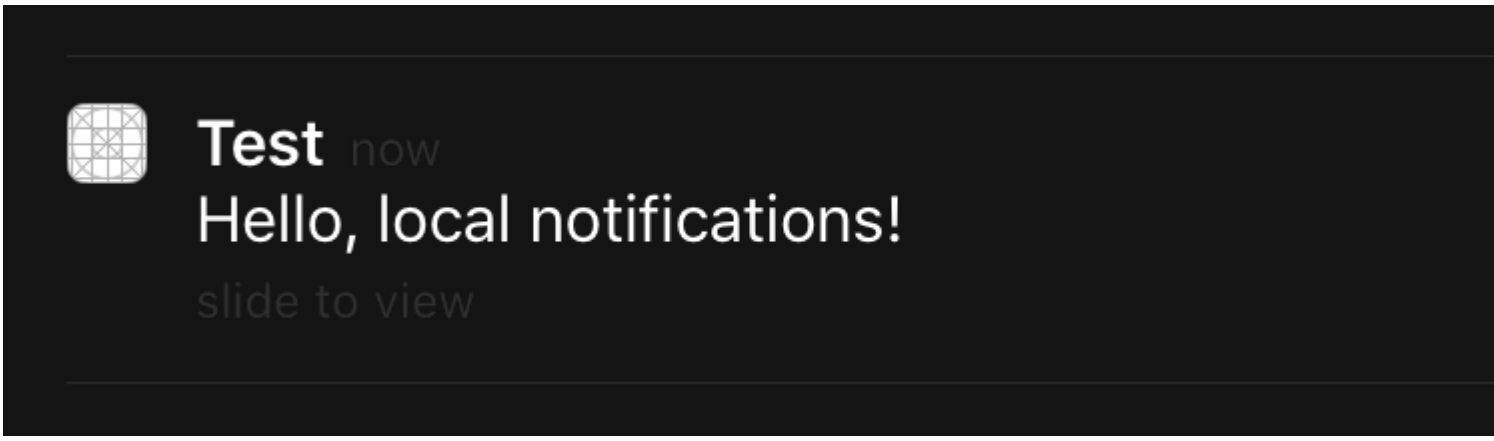
```
let notification = UILocalNotification()
notification.alertBody = "Hello, local notifications!"
notification.fireDate = NSDate().dateByAddingTimeInterval(10) // 10 seconds after now
UIApplication.sharedApplication().scheduleLocalNotification(notification)
```

Objectif c

```
UILocalNotification *notification = [[UILocalNotification alloc] init];
notification.alertBody = @"Hello, local notifications!";
notification.fireDate = [NSDate dateWithTimeIntervalSinceNow:10]; // 10 seconds after now
[[UIApplication sharedApplication] scheduleLocalNotification:notification];
```

Pour voir la notification dans le simulateur iOS, tapez ^⌘H (control-command-H) pour retourner à la maison, puis tapez ⌘L (commande-L) pour verrouiller le périphérique. Attendez quelques

secondes, et la notification doit apparaître (cette apparence varie en fonction du type de notification décrit dans "Enregistrement des notifications locales"):



Balayez la notification pour revenir à l'application (notez que si vous avez appelé ceci dans le `viewDidLoad` , `viewWillAppear` , `viewDidAppear` , etc. du premier contrôleur de vue, la notification sera à nouveau programmée).

Enregistrement pour les notifications locales

iOS 8

Pour présenter des notifications locales à l'utilisateur, vous devez enregistrer votre application avec l'appareil:

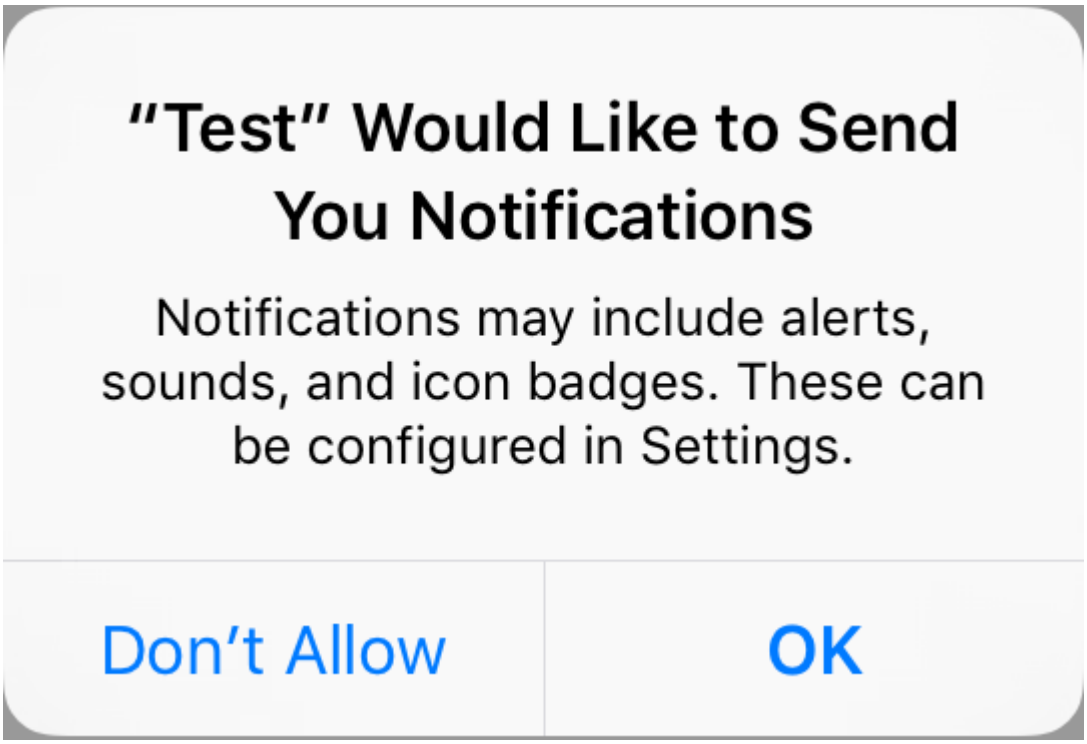
Rapide

```
let settings = UIUserNotificationSettings(forTypes: [.Badge, .Sound, .Alert], categories: nil)
UIApplication.sharedApplication().registerUserNotificationSettings(settings)
```

Objectif c

```
UIUserNotificationSettings *settings = [UIUserNotificationSettings
settingsForTypes:(UIUserNotificationTypeBadge | UIUserNotificationTypeSound |
UIUserNotificationTypeAlert) categories:nil];
[[UIApplication sharedApplication] registerUserNotificationSettings:settings];
```

Ceci présentera une alerte la première fois qu'il s'appelle:



Indépendamment de ce que l'utilisateur choisit, l'alerte ne réapparaîtra plus et l'utilisateur devra modifier les modifications dans Paramètres.

Répondre à une notification locale reçue

IMPORTANT: cette méthode déléguée est uniquement appelée au premier plan.

Rapide

```
func application(application: UIApplication, didReceiveLocalNotification notification:
UILocalNotification) {

}
```

Objectif c

```
- (void)application:(UIApplication *)application
didReceiveLocalNotification:(UILocalNotification *)notification {

}
```

Cette méthode est généralement remplacée dans AppDelegate, qui est conforme au protocole UIApplicationDelegate.

Gestion des notifications locales à l'aide d'UUID

Souvent, vous devrez être capable de gérer vos notifications, en étant en mesure de les suivre et de les annuler.

Suivre une notification

Vous pouvez affecter un UUID (identificateur unique universel) à une notification afin de pouvoir la suivre:

Rapide

```
let notification = UILocalNotification()
let uuid = NSUUID().uuidString
notification.userInfo = ["UUID": uuid]
UIApplication.shared.scheduleLocalNotification(notification)
```

Objectif c

```
UILocalNotification *notification = [[UILocalNotification alloc] init];
NSString *uuid = [[NSUUID UUID] UUIDString];
notification.userInfo = @{ @"UUID": uuid };
[[UIApplication sharedApplication] scheduleLocalNotification:notification];
```

Annuler une notification

Pour annuler une notification, nous obtenons d'abord une liste de toutes les notifications, puis nous trouvons celle avec un UUID correspondant. Enfin, nous l'annulons.

Rapide

```
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)
}
```

Objectif 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;
    }
}
```

Vous voudrez probablement stocker tous ces UUID dans Core Data ou Realm.

Présenter une notification locale immédiatement

Si vous souhaitez afficher la notification locale immédiatement, vous devez appeler:

Swift 3

```
UIApplication.shared.presentLocalNotificationNow(notification)
```

Swift 2

```
UIApplication.sharedApplication().presentLocalNotificationNow(notification)
```

Objectif c

```
[[UIApplication sharedApplication] presentLocalNotificationNow:notification];
```

Un avantage de cette utilisation est que vous n'aurez pas à définir les propriétés `fireDate` et `timeZone` de votre objet `UILocalNotification` .

Son de notification

Des sons personnalisés peuvent être fournis pour les notifications générées par votre application. Lorsque le système affiche une alerte pour une notification locale ou insère une icône d'application, il émet ce son (tant que l'utilisateur n'a pas désactivé les sons de notification).

La valeur par défaut est nil, ce qui signifie qu'aucun son n'est joué pour votre notification.

Pour fournir un son personnalisé, ajoutez un fichier `.caf` , `.wav` ou `.aiff` à votre bundle d'applications. Les sons qui durent plus de 30 secondes ne sont pas pris en charge. Fournir un son qui ne répond pas à ces exigences entraînera la lecture du son par défaut (`UILocalNotificationDefaultSoundName`).

Objectif c

```
UILocalNotification *notification = [UILocalNotification new];
notification.soundName = @"nameOfSoundInBundle.wav"; // Use
UILocalNotificationDefaultSoundName for the default alert sound
```

Rapide

```
let notification = UILocalNotification()
notification.soundName = "nameOfSoundInBundle.wav"
```

Enregistrer et planifier des notifications locales dans Swift 3.0 (iOS 10)

enregistrement

dans **AppDelegate**

```
import UserNotifications
```

dans **didFinishLaunchingWithOptions** ,

```
UNUserNotificationCenter.current().requestAuthorization(options: [.alert,.sound,.badge]) {
(granted, error) in

// Here you can check Request is Granted or not.

}
```

Créer et planifier une notification.

```
    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)
```

Si jamais cette partie de code est déclenchée, si vous avez autorisé la permission de notification, vous recevrez une notification.

Pour le tester correctement, assurez-vous que votre application est en mode Arrière-plan.

quoi de neuf dans UILocalNotification avec iOS10

Vous pouvez utiliser `UILocalNotification` , les anciennes API fonctionnent également correctement avec iOS10, mais il vaut mieux utiliser les API dans le cadre des notifications utilisateur à la place. Il existe également de nouvelles fonctionnalités que vous ne pouvez utiliser qu'avec le cadre iOS10 des notifications utilisateur.

Cela arrive également à la notification à distance, pour plus d'informations: Ici .

Nouvelles fonctionnalités:

1. Vous pouvez maintenant présenter des alertes, des sons ou augmenter le badge lorsque l'application est au premier plan avec iOS 10
2. Vous pouvez désormais gérer tous les événements au même endroit lorsque l'utilisateur appuie (ou fait glisser) le bouton d'action, même si l'application a déjà été supprimée.
3. Soutenir le toucher 3D au lieu de faire glisser le geste.
4. Maintenant, vous pouvez supprimer la notification locale spécifique uniquement par un code de ligne.
5. Prise en charge des notifications enrichies avec une interface utilisateur personnalisée.

Il est vraiment facile pour nous de convertir les API `UILocalNotification` API API iOS10, elles sont

très similaires.

J'écris ici une démo pour montrer comment utiliser simultanément les anciennes et les nouvelles API: **iOS10AdaptationTips** .

Par exemple,

Avec la mise en œuvre rapide:

1. import UserNotifications

```
///    Notification become independent from UIKit
import UserNotifications
```

2. demande d'autorisation pour localNotification

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

3. planifier localNotification

4. mettre à jour le numéro de badge de l'icône de l'application

```
@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"])
}
```

Implémentation Objective-C:

1. import UserNotifications

```
// Notifications are independent from UIKit
#import <UserNotifications/UserNotifications.h>
```

2. demande d'autorisation pour 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. planifier localNotification

4. mettre à jour le numéro de badge de l'icône de l'application

```
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!");
    }
}];
```

Allez ici pour plus d'informations: **iOS10AdaptationTips** .

#actualisé

> Application de terminaison en raison d'une exception non capturée "NSInternalInconsistencyException", raison: "l'intervalle de temps doit être d'au moins 60 s'il est répété"

```
let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 60, repeats: true)
```

Lire UILocalNotification en ligne: https://riptutorial.com/fr/ios/topic/635/uilocalnotification

Chapitre 175: UINavigationController

Remarques

De la documentation :

> La classe UINavigationController implémente un contrôleur de vue spécialisé qui gère la navigation du contenu hiérarchique. Cette interface de navigation permet de présenter efficacement vos données et facilite la navigation de l'utilisateur dans ce contenu. Vous utilisez généralement cette classe en l'état, mais vous pouvez également sous-classer pour personnaliser le comportement de la classe.

Examples

Popping dans un contrôleur de navigation

Vers le contrôleur de vue précédent

Pour revenir à la page précédente, vous pouvez le faire:

Rapide

```
navigationController?.popViewControllerAnimated(true)
```

Objectif c

```
[self.navigationController popViewControllerAnimated:YES];
```

Contrôleur de vue racine

Pour accéder à la racine de la pile de navigation, vous pouvez le faire:

Rapide

```
navigationController?.popToRootViewControllerAnimated(true)
```

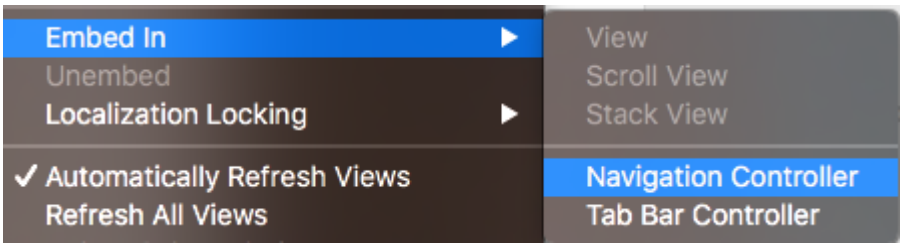
Objectif c

```
[self.navigationController popToRootViewControllerAnimated:YES];
```

Créer un contrôleur de navigation

Dans votre storyboard, sélectionnez le ViewController que vous souhaitez incorporer dans un contrôleur de navigation.

Puis naviguez vers l'éditeur> Intégrer dans> Contrôleur de navigation



Et cela va créer votre contrôleur de navigation

Intégrer un contrôleur de vue dans un contrôleur de navigation par programmation

Rapide

```
//Swift
let viewController = UIViewController()
let navigationController = UINavigationController(rootViewController: viewController)

//Objective-C
UIViewController *viewController = [[UIViewController alloc] init];
UINavigationController *navigationController = [[UINavigationController alloc]
initWithRootViewController:viewController];
```

Pousser un contrôleur de vue sur la pile de navigation

```
//Swift
let fooViewController = UIViewController()
navigationController?.pushViewController(fooViewController, animated: true)
```

```
//Objective-C
UIViewController *fooViewController = [[UIViewController alloc] init];
[navigationController pushViewController:fooViewController animated:YES];
```

Objectif

`UINavigationController` est utilisé pour former une hiérarchie arborescente de contrôleurs de vue, appelée `navigation stack` .

Du point de vue des développeurs:

Vous pouvez connecter un contrôleur indépendant et bénéficier de tous les avantages d'un gestionnaire de hiérarchie gratuit et d'un présentateur d'interface utilisateur commun gratuit. `UINavigationController` anime la transition vers les nouveaux contrôleurs et fournit automatiquement les fonctionnalités de retour. `UINavigationController` donne également accès à tous les autres contrôleurs de la `navigation stack` ce qui peut faciliter l'accès à certaines fonctionnalités ou données.

Du point de vue de l'utilisateur:

`UINavigationController` permet de se rappeler où se trouve l'utilisateur à l'heure actuelle (titre de la barre de navigation) et comment il peut revenir (bouton arrière intégré) à l'un des écrans précédents.

Lire UINavigationController en ligne: https://riptutorial.com/fr/ios/topic/1079/uinavigationcontroller

Chapitre 176: UIPageViewController

Introduction

UIPageViewController permet aux utilisateurs de faire facilement la transition entre plusieurs vues en utilisant un geste de balayage. Pour créer un UIPageViewController, vous devez implémenter les méthodes UIPageViewControllerDataSource. Celles-ci incluent des méthodes pour retourner à la fois le UIPageViewController avant et après le UIPageViewController actuel avec les méthodes presentationCount et presentationIndex.

Syntaxe

1. UIPageViewControllerTransitionStyle
2. UIPageViewControllerNavigationOrientation
3. UIPageViewControllerSpineLocation
4. UIPageViewControllerNavigationDirection

Remarques

Référence du développeur Apple ici

Examples

Créez un UIPageViewController de pagination horizontale par programmation

1. Init tableau des contrôleurs de vue qui seront gérés par UIPageViewController. Ajoutez une classe de contrôleur de vue de base qui possède un `identifier` propriété qui sera utilisé pour identifier les contrôleurs de vue lors de l'utilisation des méthodes de source de données UIPageViewController. Laissez les contrôleurs de vue hériter de cette classe de 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. Créez l'instance UIPageViewController.

```
UIPageViewController *pageViewController = [[UIPageViewController alloc]
initWithTransitionStyle:UIPageViewControllerTransitionStyleScroll

navigationOrientation:UIPageViewControllerNavigationOrientationHorizontal

options:nil];
```

3. La source de données est la classe en cours qui doit implémenter le protocole

`UIPageViewControllerDataSource` .

```
pageViewController.dataSource = self;
```

4. `setViewControllers` ajoutera uniquement le premier contrôleur de vue, ensuite sera ajouté à la pile à l'aide de méthodes de source de données

```
if (viewControllers.count) {
    [pageViewController setViewControllers:@[[viewControllers objectAtIndex:0]]
                                 direction:UIPageViewControllerNavigationDirectionForward
                                  animated:NO
                                completion:nil];
}
```

5. Ajoutez UIPageViewController en tant que contrôleur de vue enfant pour qu'il reçoive des événements d' `appearance` et de `rotation` contrôleur de la vue parent.

```
 [self addChildViewController:pageViewController];
 pageViewController.view.frame = self.view.frame;
 [self.view addSubview:pageViewController.view];
 [pageViewController didMoveToParentViewController:self];
```

6. Implémentation des méthodes 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;
}
```

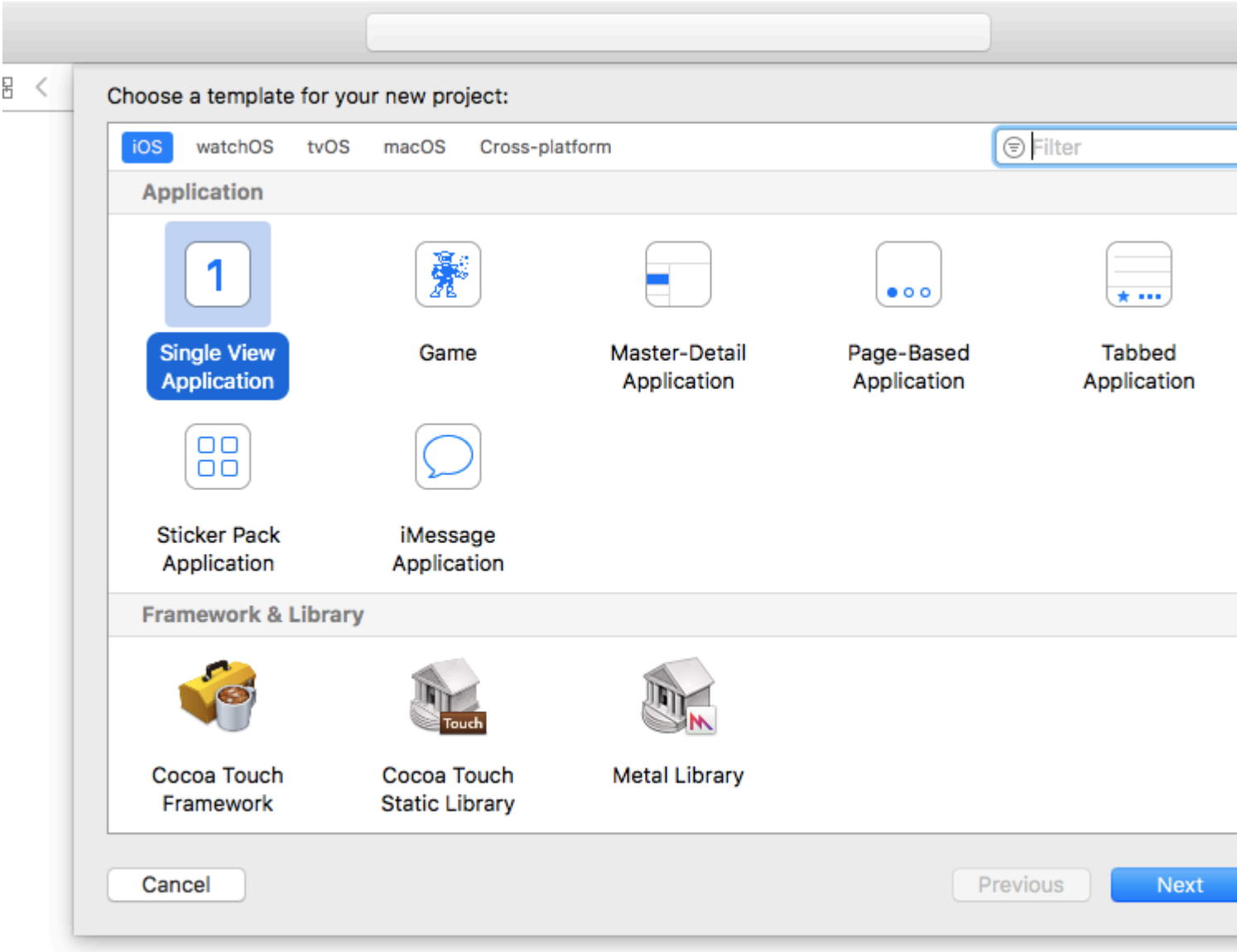
7. Méthode utilitaire qui renvoie un contrôleur de vue à l'aide d'un index, si l'index est hors limites, il renvoie nil.

```
- (UIViewController *)childViewControllerAtIndex:(NSInteger)index
```

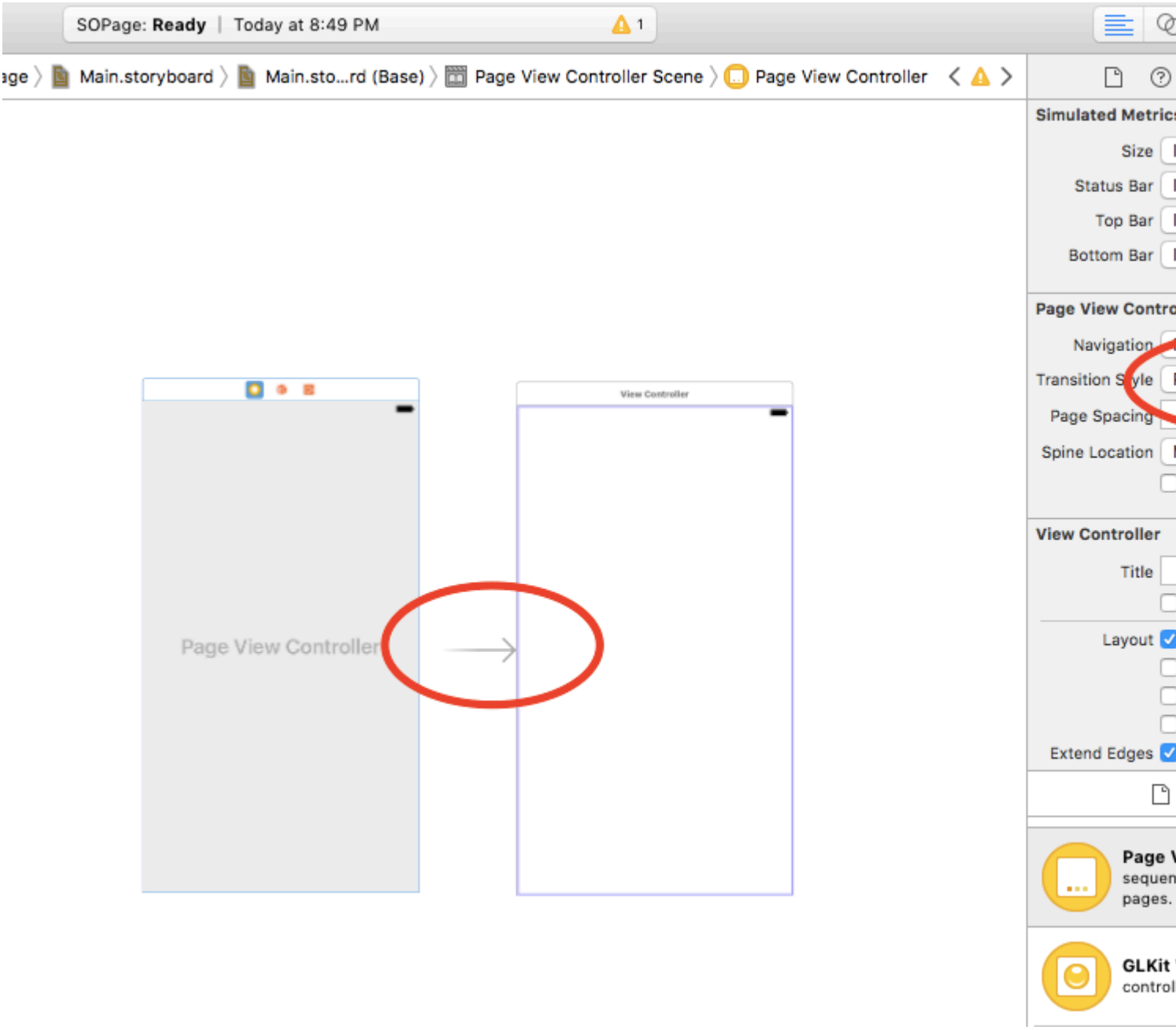
```
{
    if (index <= ([viewControllers count] - 1)) {
        return [viewControllers objectAtIndex:index];
    } else {
        return nil;
    }
}
```

Un moyen simple de créer des contrôleurs de vue de page horizontaux (pages infinies)

1. Créons un nouveau projet, je choisis l'application Single View pour une meilleure démonstration



2. Faites glisser un contrôleur de vue de page sur le storyboard, il y a 2 choses à changer après cela:
    1. Définir le contrôleur de vue de page en tant que contrôleur de vue initial
    2. Changer le style de transition pour faire défiler



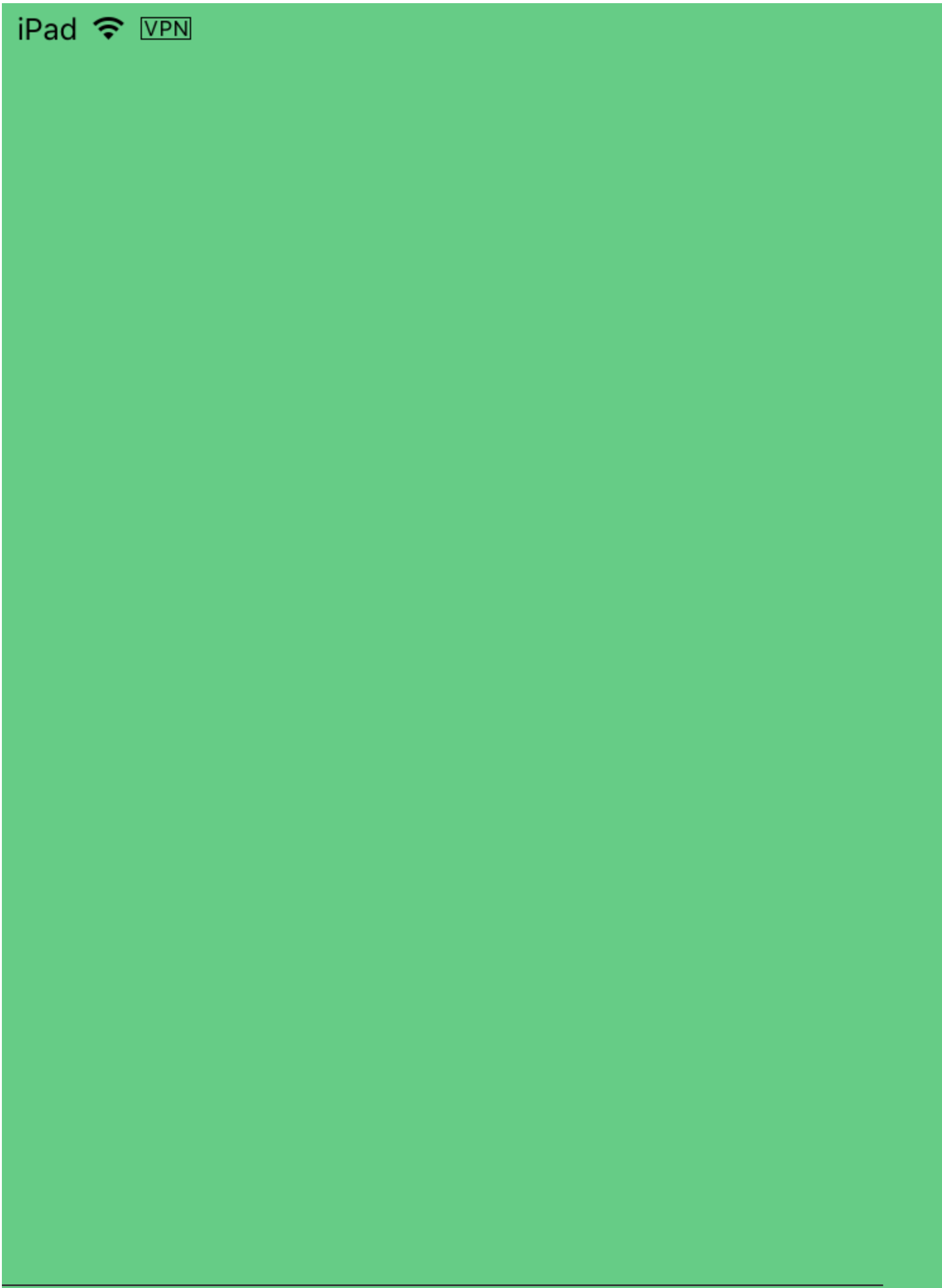
3. Et vous devez créer une classe UIPageViewController, puis la définir comme classe personnalisée du contrôleur de vue de page sur le storyboard

4. Collez ce code dans votre classe UIPageViewController, vous devriez obtenir une application paginée infinie colorée :)

```
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
    }
}
```

Voici à quoi ressemble le projet final: vous obtenez un contrôleur de vue de couleur différente avec chaque rouleau:


iPad
VPN

Chapitre 177: UIPheonix - framework d'interface utilisateur simple, flexible, dynamique et hautement évolutif

Introduction

Inspiré par le développement de jeux UIPheonix est un concept d'interface utilisateur + super simple, flexible, dynamique et hautement évolutif pour la création d'applications réutilisables pilotées par des composants / contrôles pour macOS, iOS et tvOS. La même API s'applique au développement multiplate-forme! Pensez-y comme en utilisant des blocs Lego, vous pouvez utiliser des blocs similaires et les déplacer facilement comme bonjour.

https://github.com/MKGitHub/UIPheonix

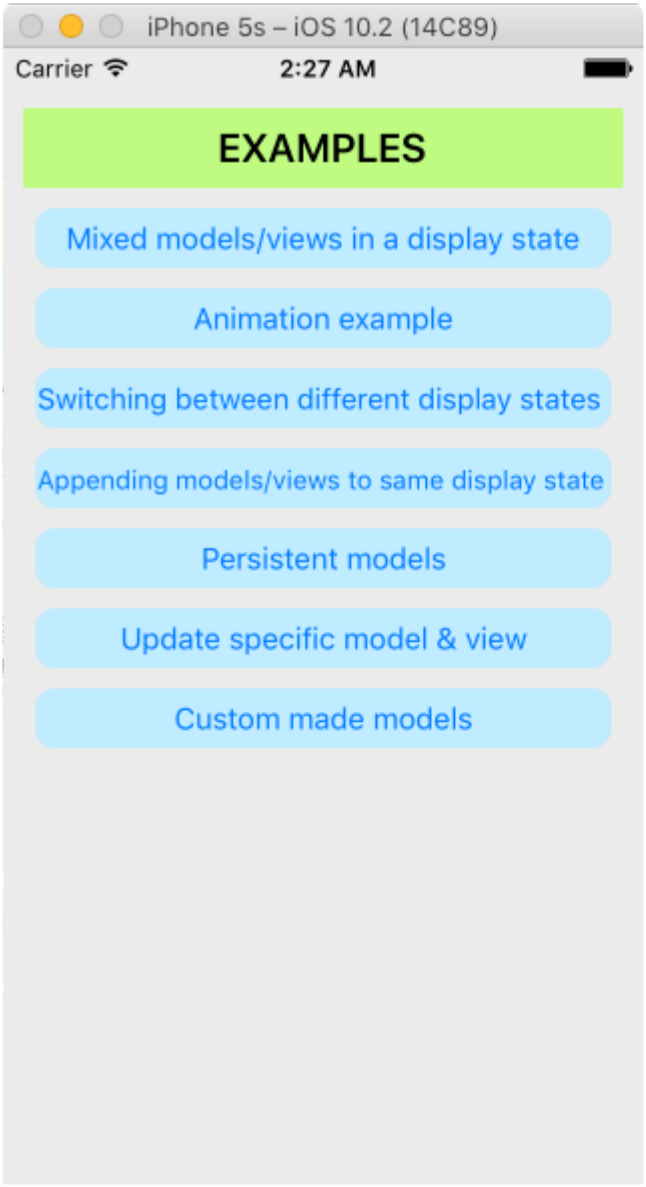
Remarques

- Oubliez les mises en page statiques, les problèmes de contrainte et les explosions d'avertissement dans la console.
- Oubliez tout le code de la colle, tout le code passe-partout et toute la pile de codes inutiles dans vos applications.
- Créez et apportez rapidement des modifications à votre interface utilisateur.
- Rendez votre interface utilisateur réutilisable.
- Concentrez-vous sur la création de votre application, et non sur les problèmes de mise en page.
- Configuration minimale, impact minimal sur votre application, légèreté, pas de dépendances, pas de douleur mais beaucoup de gain!
- S'appuie sur les vues de collection et les vues de table, ce qui vous permet de les combiner facilement.
- Ne remplace pas les technologies Apple par des implémentations personnalisées, vous serez donc toujours en sécurité et à jour et vous pourrez facilement revenir à tout moment.
- Des applications de démonstration fournies pour macOS, iOS et tvOS (Kung Fu!)

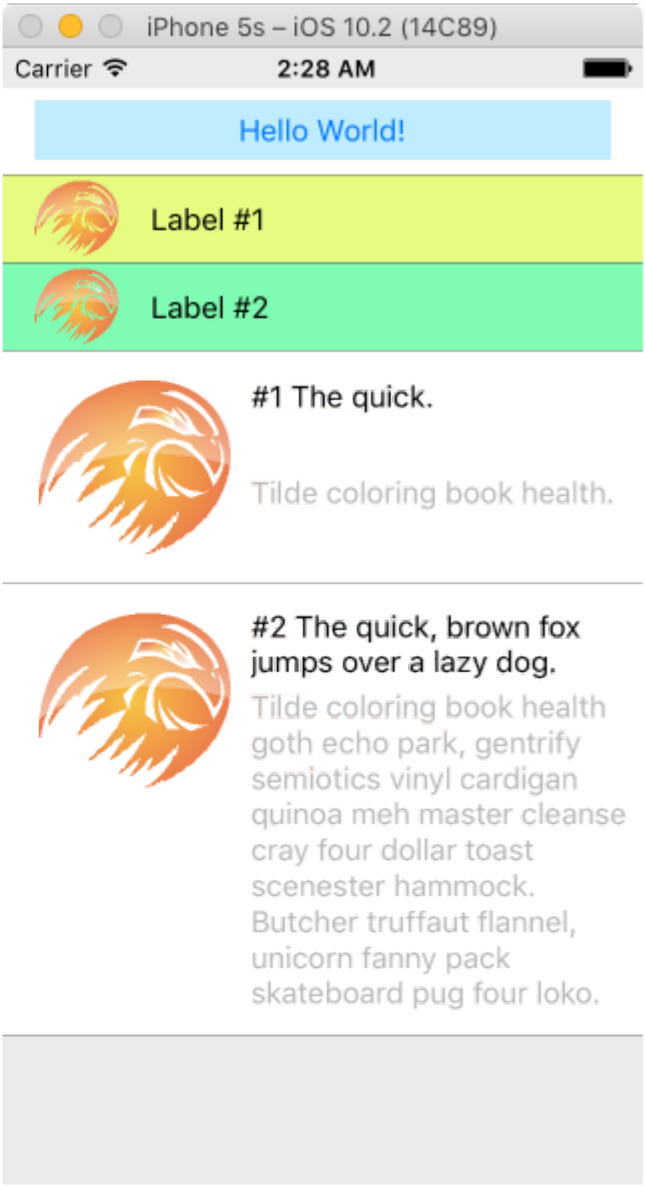
Examples

Exemple de composants d'interface utilisateur

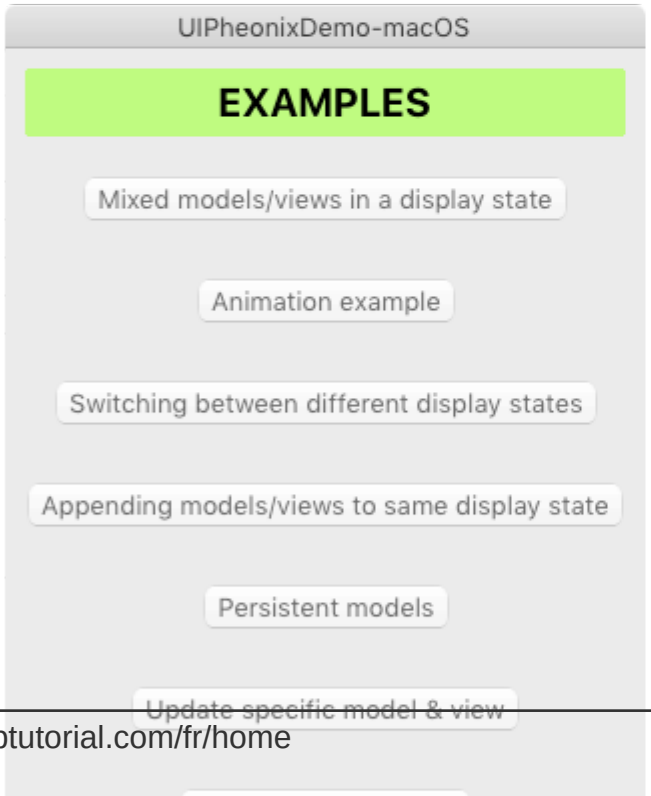
iOS - Collection View



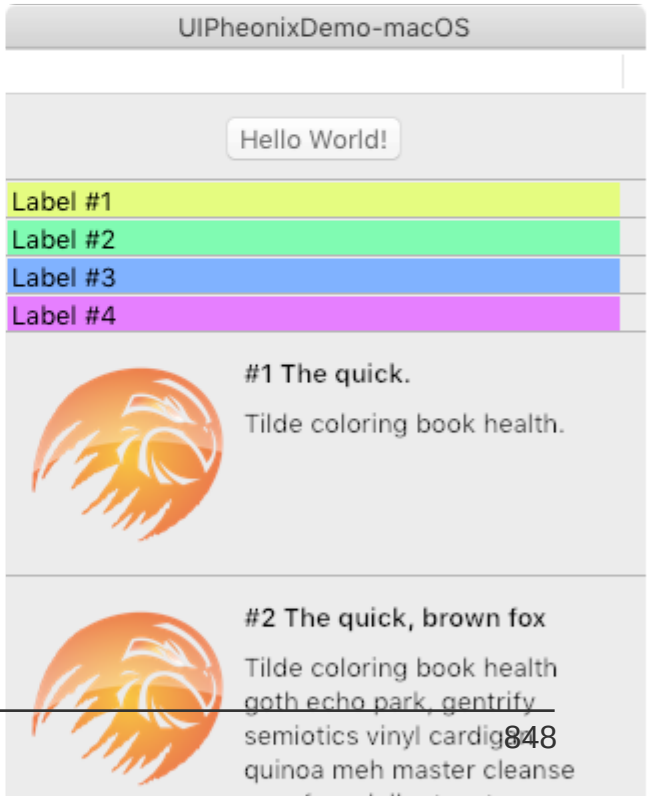
iOS - Table View



macOS - Collection View



macOS - Table View



Chapitre 178: UIPickerView

Examples

Exemple de base

Rapide

```
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
    }

}
```

Objectif 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];
}
```

Changement de sélecteurCouleur de fond et couleur du texte

Objectif 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"];
```

Rapide

```
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")
```

Lire UIPickerView en ligne: https://riptutorial.com/fr/ios/topic/4242/uipickerview

Chapitre 179: UIRefreshControl TableView

Introduction

Un objet UIRefreshControl fournit un contrôle standard qui peut être utilisé pour lancer l'actualisation du contenu d'une vue de table. Vous liez un contrôle d'actualisation à une table via un objet contrôleur de vue de table associé. Le contrôleur de vue de table gère le travail d'ajout du contrôle à l'apparence visuelle du tableau et de gestion de l'affichage de ce contrôle en réponse aux gestes utilisateur appropriés.

Examples

Exemple d'objectif-C

Déclarez d'abord une propriété comme celle-ci dans ViewController

```
@property (nonatomic) UIRefreshControl *refreshControl;
```

Plus tard dans `viewDidLoad()` configurez le refreshControl comme indiqué ci-dessous:

```
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;
```

Maintenant, la fonction `refreshTable` est définie comme suit:

```
- (void)refreshTable {
    //TODO: refresh your data
    [self.refreshControl endRefreshing];
    [self.refreshControl beginRefreshing];
    [self.tableView reloadData];
    [self.refreshControl endRefreshing];
}
```

Configurez refreshControl sur 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];
    });
});
```

}

Lire UIRefreshControl TableView en ligne: https://riptutorial.com/fr/ios/topic/8278/uirefreshcontrol-tableview

Chapitre 180: UIScrollView

Examples

Créer un UIScrollView

Créez une instance de `UIScrollView` avec un `CGRect` tant `CGRect` .

Rapide

```
let scrollview = UIScrollView.init(frame: CGRect(x: 0, y: 0, width: 320, height: 400))
```

Objectif c

```
UIScrollView *scrollview = [[UIScrollView alloc] initWithFrame:CGRectMake(0, 0, 320, 400)];
```

Défilement Afficher la taille du contenu

La propriété `contentSize` doit être définie sur la taille du contenu défilant. Ceci spécifie la taille de la zone déroulante. Le défilement est visible lorsque la zone défilante, c.-à-d. `contentSize` est plus grande que la taille de l'image `UIScrollView` .

Avec Autolayout:

Lorsque le contenu de la vue de défilement est configuré avec autolayout, il doit être explicitement dimensionné verticalement et horizontalement et les 4 arêtes doivent être épinglées à la vue de défilement contenant. De cette façon, `contentSize` est calculé automatiquement en fonction du contenu de la vue de défilement et est également mis à jour lorsque la disposition du contenu est modifiée.

Manuellement:

Rapide

```
scrollview.contentSize = CGSize(width: 640, height: 800)
```

Objectif c

```
scrollview.contentSize = CGSizeMake(640, 800);
```

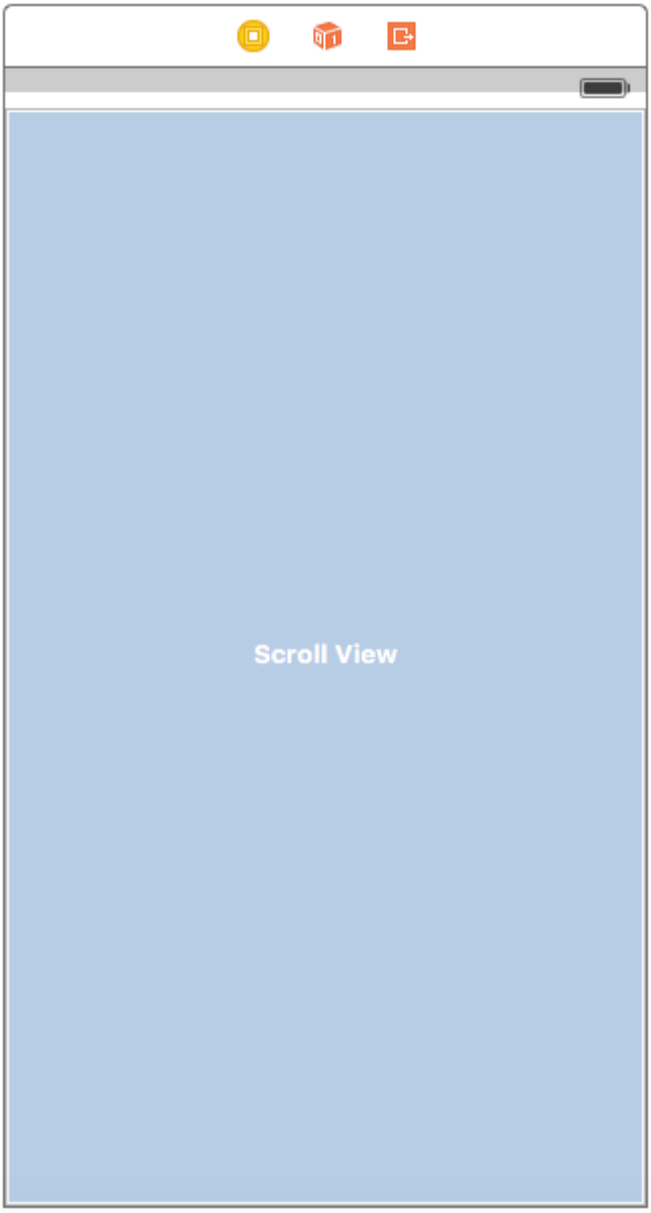
ScrollView avec mise en forme automatique

Étapes simples pour utiliser scrollview avec autolayout.

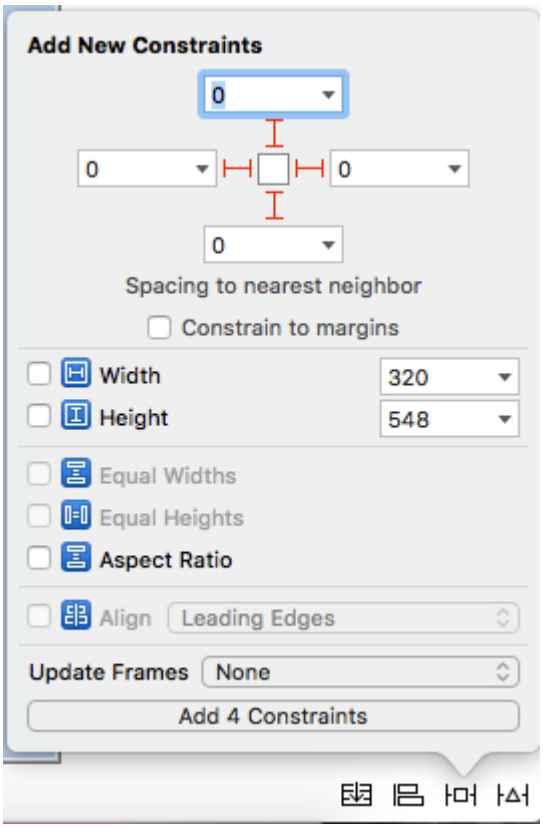
- Créer un nouveau projet avec une application à vue unique
- Sélectionnez le viewcontroller par défaut et changez sa taille d'écran en iPhone-4inch à

partir de l'inspecteur d'attributs.

- Ajoutez une vue de défilement à la vue de votre viewcontroller comme suit et définissez la couleur d'arrière-plan sur bleu



- Ajoutez des contraintes comme indiqué ci-dessous image



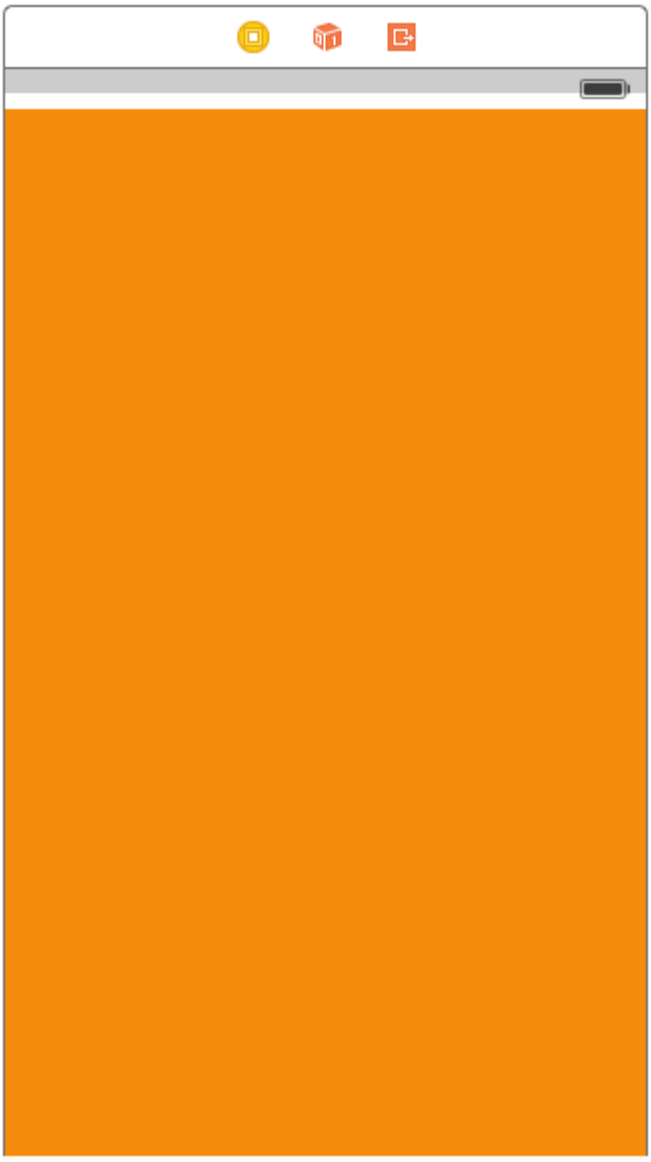
Qu'est-ce que cela va faire, il suffit de coller tous les bords de scrollview à la vue de viewcontroller

Scénario 1:

Maintenant, disons que notre contenu est énorme, et nous voulons qu'il défile horizontalement aussi bien que verticalement.

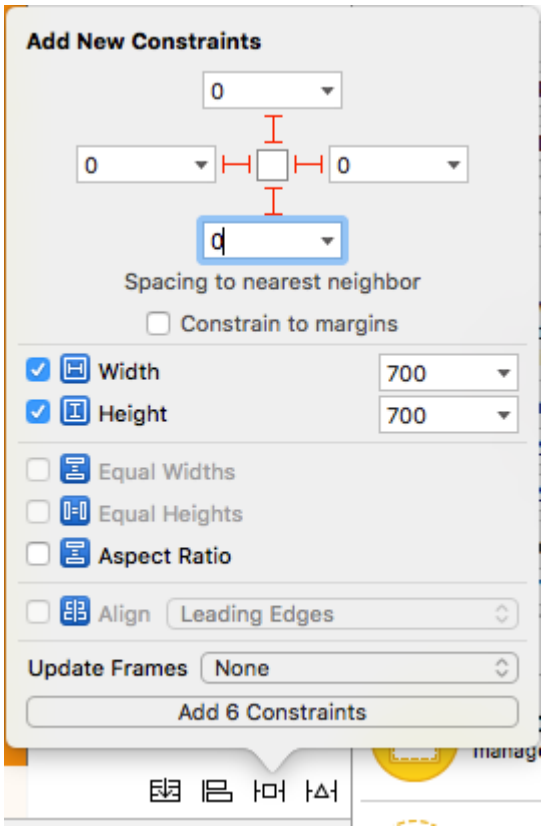
Pour ça,

- Ajoutez un UIView à la vue de défilement du cadre (0,0,700,700). Donnez-lui une couleur de fond orange pour l'identifier différemment.



Vient ensuite la partie importante, il faut la faire défiler horizontalement et verticalement.

- Sélectionnez la vue orange et ajoutez les contraintes suivantes



Laissez-moi vous expliquer ce que nous avons fait dans l'étape ci-dessus.

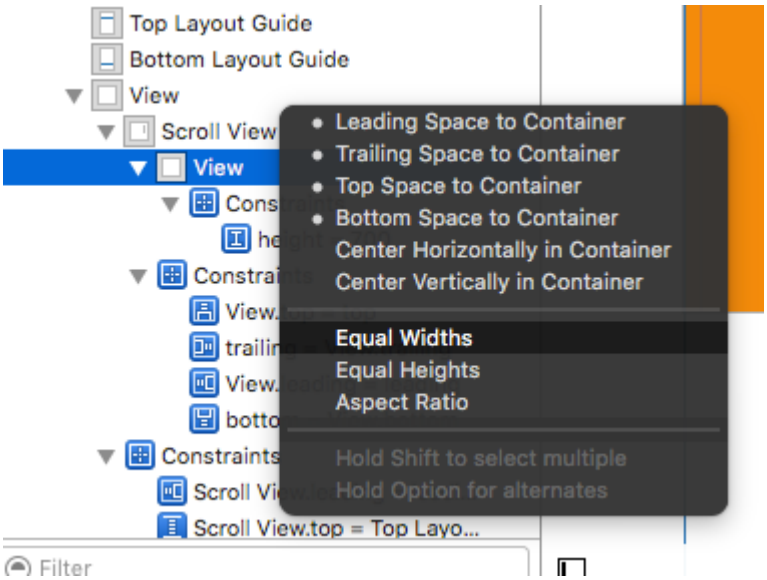
- Nous avons fixé la hauteur et la largeur à 700.
- Nous définissons un espace à scrollview = 0 qui indique à la vue de défilement que le contenu est défilable horizontalement.
- Nous définissons l'espace du bas sur scrollview = 0 qui indique à la vue de défilement que le contenu est défilable verticalement.

Maintenant, lancez le projet et vérifiez.

Scénario 2: *considérons un scénario où nous savons que la largeur du contenu sera identique à la largeur de défilement, mais que la hauteur est supérieure à celle de scrollview.*

Suivez les étapes pour faire défiler le contenu verticalement.

- Supprimez la contrainte de largeur dans le cas ci-dessus.
- Modifiez la largeur de la vue orange pour qu'elle corresponde à la largeur de la vue de défilement.
- Faites glisser Ctrl depuis la vue orange pour faire défiler la vue et ajouter **une** contrainte de **largeur égale** .



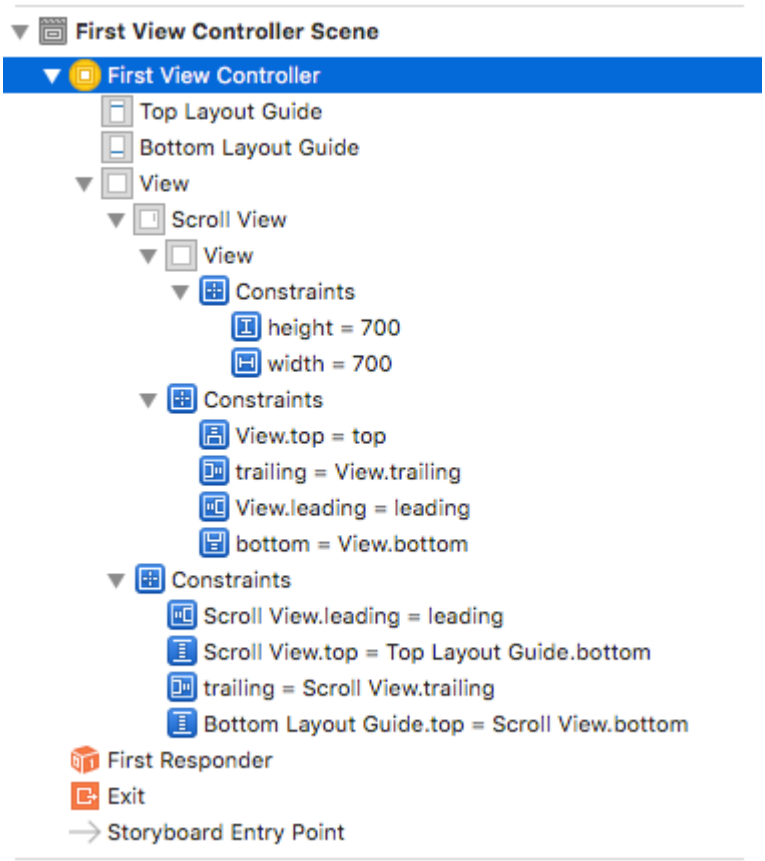
- Et fait!!! Il suffit de lancer et de vérifier s'il défile verticalement

Scénario 3:

Maintenant, nous voulons faire défiler uniquement horizontalement et non verticalement.

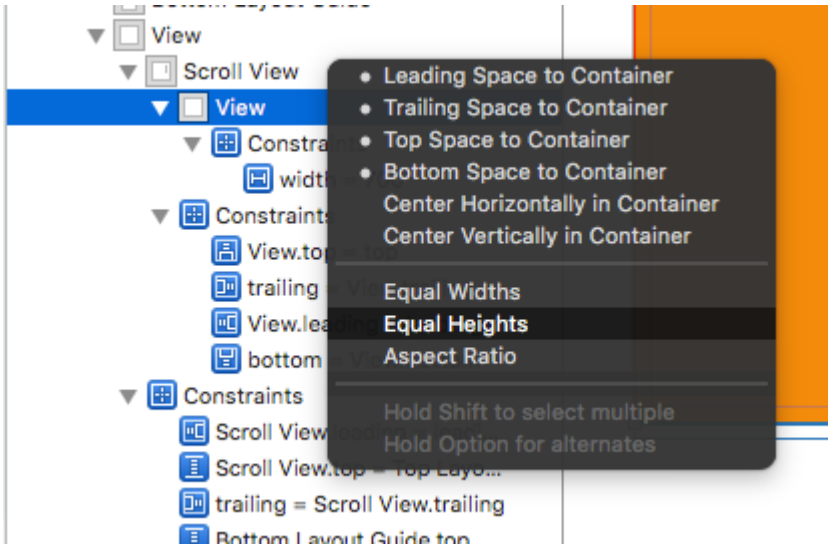
Suivez les étapes pour faire défiler horizontalement le contenu.

- Annuler toutes les modifications pour obtenir des contraintes comme ci-dessous (c.-à-d. **Restaurer les contraintes d'origine qui ont atteint le défilement vertical et horizontal**)



- Vérifiez l'image de la vue orange, qui devrait être (0,0,700,700)

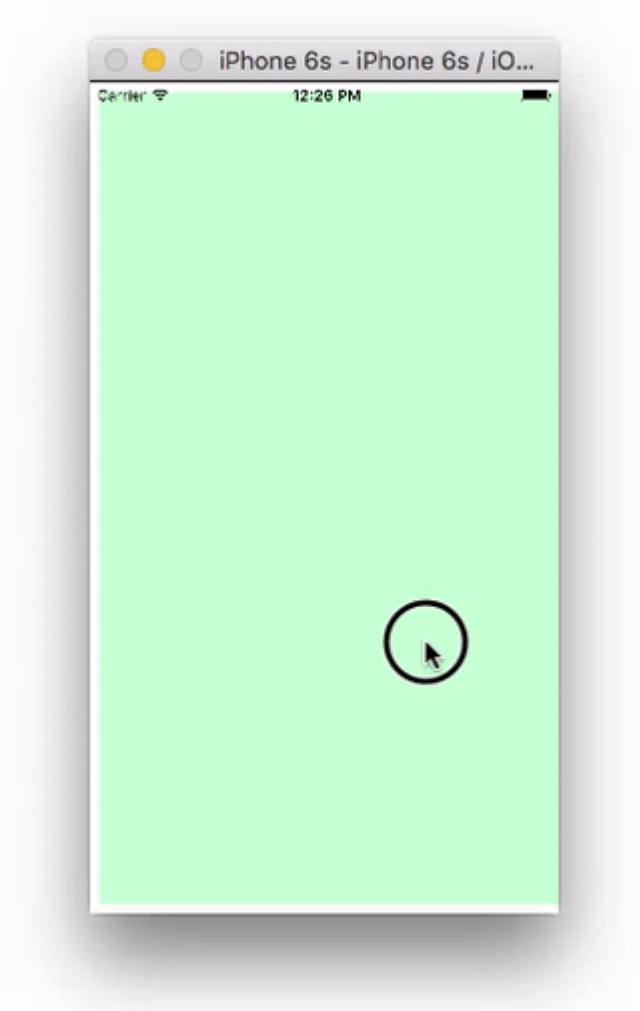
- Supprimer la contrainte de hauteur de la vue orange.
- Modifiez la hauteur de la vue orange pour qu'elle corresponde à la hauteur de défilement.
- Tout en maintenant la touche Ctrl enfoncée, faites glisser la vue orange pour faire défiler la vue et ajouter **une** contrainte de **hauteur égale** .



- Et fait!!! Il suffit de lancer et de vérifier s'il défile verticalement

Défilement du contenu avec mise en forme automatique activée

Ce projet est un exemple autonome entièrement réalisé dans Interface Builder. Vous devriez pouvoir le parcourir en 10 minutes ou moins. Ensuite, vous pouvez appliquer les concepts que vous avez appris à votre propre projet.



Ici, j'utilise juste `UIView` s mais ils peuvent représenter n'importe quelle vue que vous aimez (c.-à-d. Bouton, étiquette, etc.). J'ai également choisi le défilement horizontal car les captures d'écran du storyboard sont plus compactes pour ce format. Les principes sont les mêmes pour le défilement vertical.

Concepts clés

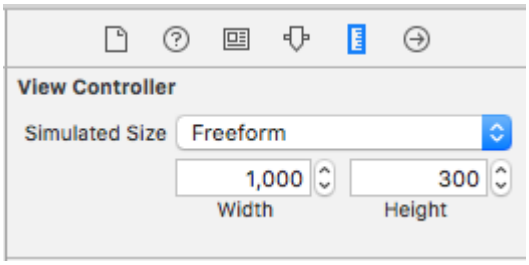
- `UIScrollView` ne doit utiliser qu'une seule sous-vue. Ceci est un 'UIView' qui sert de vue de contenu pour contenir tout ce que vous souhaitez faire défiler.
- Rendre la vue du contenu et le *parent de* la vue de défilement avoir des hauteurs égales pour le défilement horizontal. (Largeurs égales pour le défilement vertical)
- Assurez-vous que tout le contenu défilable a une largeur définie et est épinglé de tous les côtés.

Lancer un nouveau projet

Il ne peut s'agir que d'une application à vue unique.

Storyboard

Dans cet exemple, nous allons faire un défilement horizontal. Sélectionnez le View Controller, puis choisissez Freeform dans l'inspecteur de taille. Faites la largeur `1,000` et la hauteur `300` . Cela nous donne juste la place dans le storyboard pour ajouter du contenu qui défile.

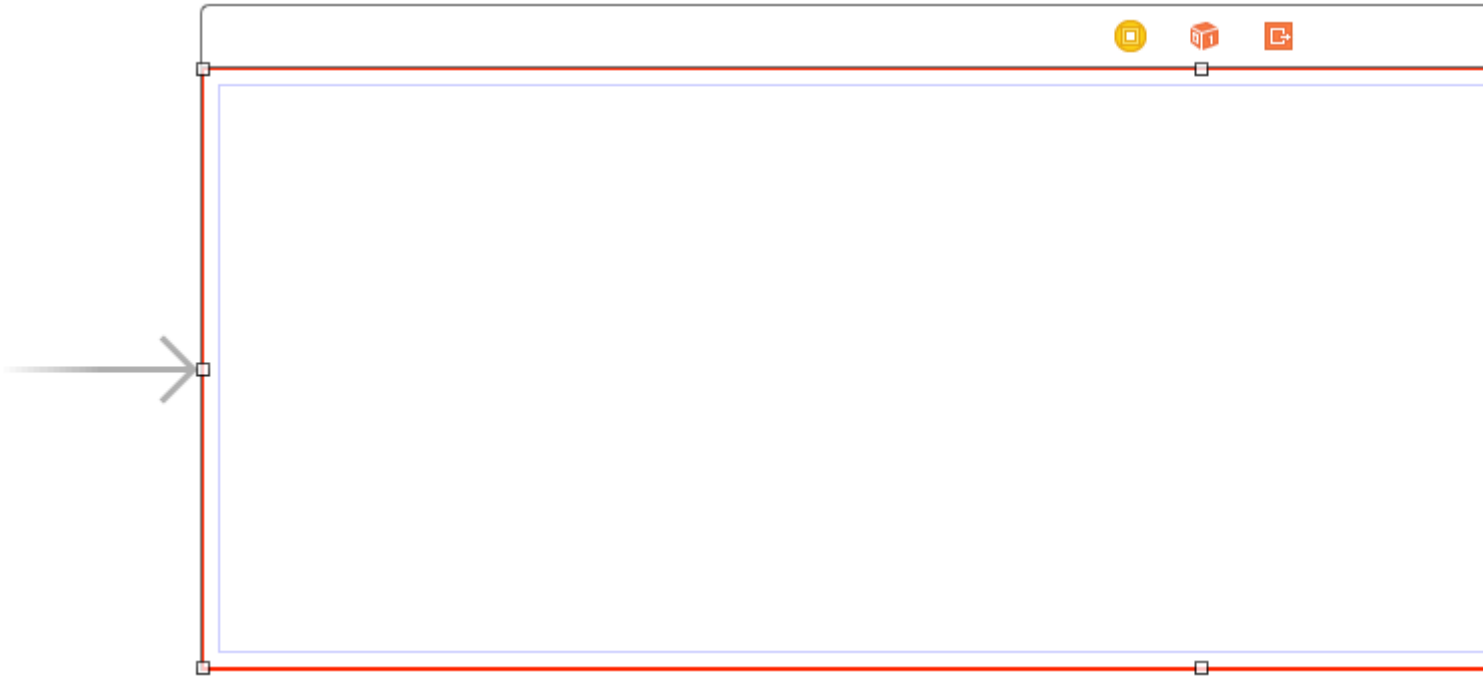


Ajouter une vue de défilement

Ajoutez un `UIScrollView` et épinglez les quatre côtés à la vue racine du contrôleur de vue.

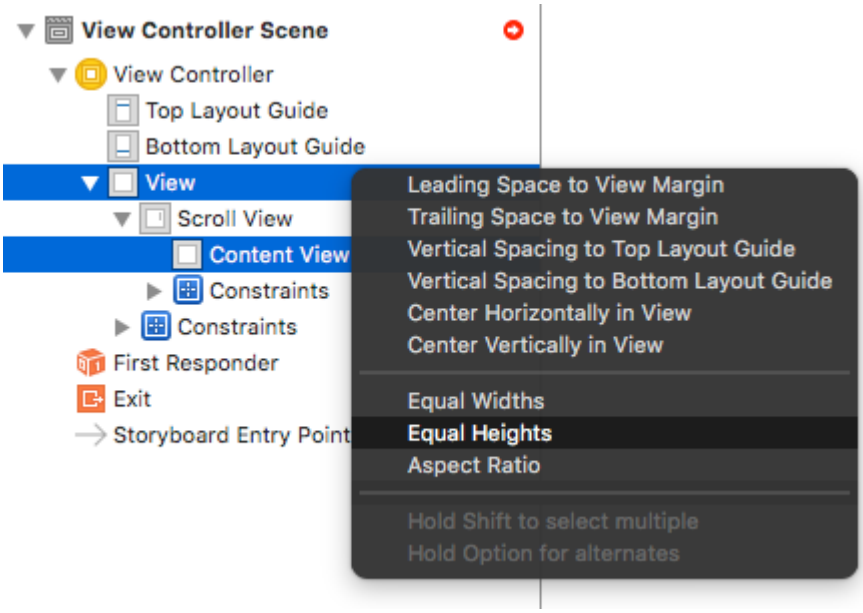
Ajouter une vue de contenu

Ajoutez un `UIView` tant que sous-vue à la vue de défilement. *Ceci est la clé.* N'essayez pas d'ajouter beaucoup de sous-vues à la vue de défilement. Ajoutez simplement un seul `UIView` . Ce sera votre affichage de contenu pour les autres vues que vous souhaitez faire défiler. Épinglez la vue de contenu à la vue de défilement sur les quatre côtés.



Hauteurs égales

Désormais, dans la structure du document, la `commande` clique à la fois sur la vue du contenu et sur la vue *parente de* la vue de défilement afin de les sélectionner toutes les deux. Ensuite, définissez les hauteurs comme étant égales (contrôlez </ kbd faites glisser de l'affichage de contenu vers l'affichage de défilement>). *C'est aussi la clé.* Comme nous faisons défiler horizontalement, la vue du contenu de la vue de défilement ne saura pas à quelle hauteur elle doit être, sauf si nous la configurons de cette manière.

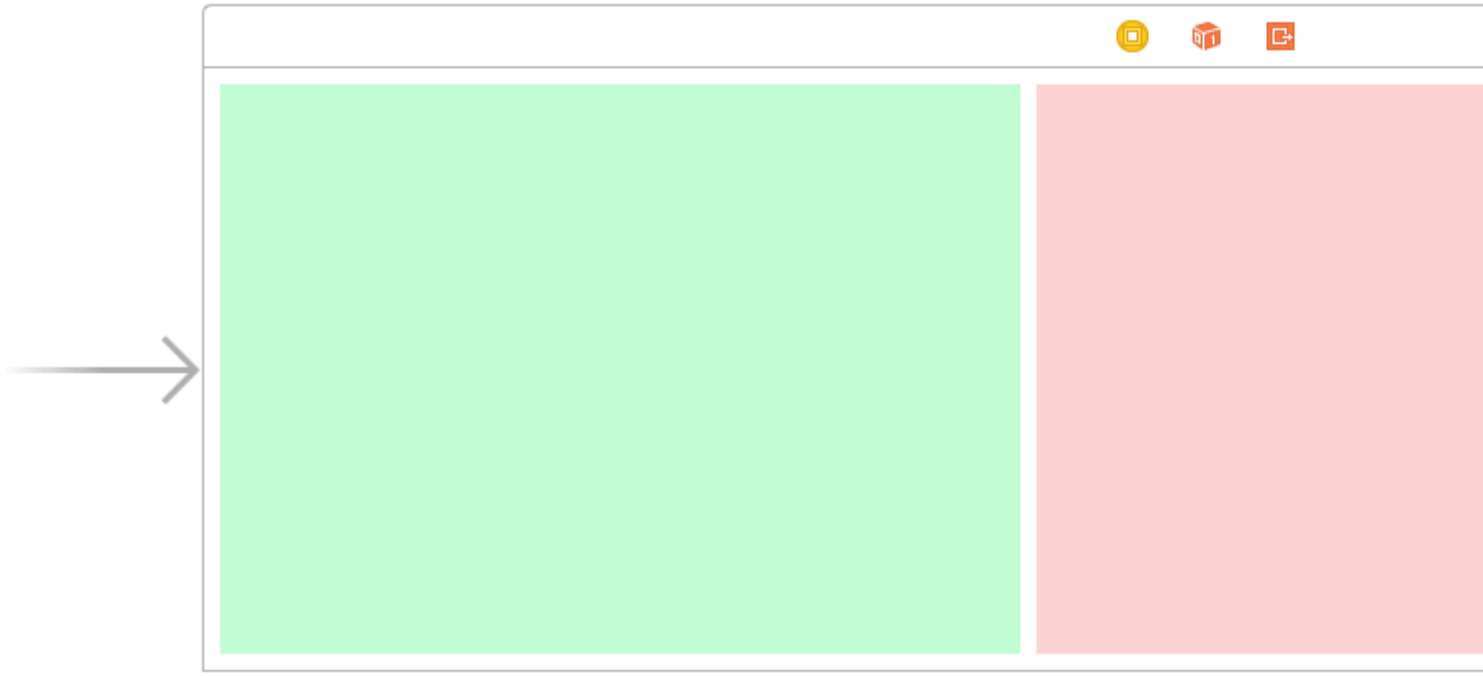


Remarque:

- Si nous faisions défiler le contenu verticalement, nous définirions la largeur de la vue de contenu comme étant égale à la largeur du parent de la vue de défilement.

Ajouter du contenu

Ajoutez trois `UIView` s et donnez-leur toutes les contraintes. J'ai utilisé 8 marges de points pour tout.



Contraintes:

- Vue verte: épinglez les bords supérieur, gauche et inférieur. Faites la largeur 400.
- Vue rouge: épinglez les bords supérieur, gauche et inférieur. Faites la largeur 300.
- Vue violette: épinglez les quatre bords. Faites la largeur quel que soit l'espace restant (268 dans ce cas).

La définition des contraintes de largeur est également la clé pour que la vue de défilement connaisse la largeur de son affichage de contenu.

Fini

C'est tout. Vous pouvez exécuter votre projet maintenant. Il devrait se comporter comme l'image défilante en haut de cette réponse.

Une étude plus approfondie

- iOS: Comment faire fonctionner la mise en forme automatique sur un ScrollView
- Comment configurer un UIScrollView avec mise en forme automatique dans Interface Builder
- Tutoriel vidéo YouTube: UIScrollView - Comment garder vos vues à l'écran

Activer / Désactiver le défilement

La propriété `scrollEnabled` stocke une valeur `Boolean` qui détermine si le défilement est activé ou non.
Si la valeur de cette propriété est true / YES, le défilement est activé, sinon pas.La valeur par défaut est `true`

Rapide

```
scrollview.isScrollEnabled = true
```

Objectif c

```
scrollview.scrollEnabled = YES;
```

Zoom avant / arrière UIImageView

Créer une instance UIScrollView

```
let scrollview = UIScrollView.init(frame: self.view.bounds)
```

Et puis définissez ces propriétés:

```
scrollView.minimumZoomScale = 0.1
scrollView.maximumZoomScale = 4.0
scrollView.zoomScale = 1.0
scrollview.delegate = self as? UIScrollViewDelegate
```

Pour effectuer un zoom avant ou arrière sur l'image, vous devez spécifier la quantité de zoom avant et arrière que l'utilisateur peut utiliser. Nous faisons cela en définissant les valeurs des propriétés `minimumZoomScale` et `maximumZoomScale` de la vue de `minimumZoomScale` . Les deux sont définis sur 1.0 par défaut.

Et `zoomScale` à 1.0 qui spécifie le facteur de zoom pour le zoom minimum et maximum.

Pour prendre en charge le zoom, vous devez définir un délégué pour votre vue défilement. L'objet délégué doit être conforme au protocole `UIScrollViewDelegate` . Cette classe de délégué doit implémenter la méthode `viewForZoomingInScrollView()` et renvoyer la vue pour zoomer.

Modifiez votre ViewController comme indiqué

```
class ViewController: UIViewController, UIScrollViewDelegate
```

Ajoutez ensuite la fonction de délégué suivante à la classe.

```
func viewForZoomingInScrollView(scrollView: UIScrollView) -> UIView? {
    return imageView
}
```

Maintenant, créez une instance UIImageView

Faire de cette variable une variable de classe

```
var imageView:UIImageView = UIImageView.init(image: UIImage.init(named: "someImage.jpg"))
```

Et puis l'ajouter à scrollview

```
scrollView?.addSubview(imageView)
```

Référence

- Guide de programmation de l'affichage du défilement pour iOS
- Tutoriel UIScrollView

Détection du moment où UIScrollView a fini de défiler avec les méthodes déléguées

scrollViewDidEndDecelerating: indique au délégué que la vue de défilement a **cessé de** ralentir le mouvement de défilement.

Objectif c:

```
- (void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView {
    [self stoppedScrolling];
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate {
    if (!decelerate) {
        [self stoppedScrolling];
    }
}

- (void)stoppedScrolling {
    // done, do whatever
}
```

Rapide:

```
func scrollViewDidEndDragging(scrollView: UIScrollView, willDecelerate decelerate: Bool) {
    if !decelerate {
        stoppedScrolling()
    }
}

func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
    stoppedScrolling()
}
```

```
func stoppedScrolling() {
    // done, do whatever
}
```

Restreindre le sens du défilement

Vous pouvez restreindre les instructions que l'utilisateur peut faire défiler à l'aide du code suivant:

```
func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if scrollView.contentOffset.x != 0 {
        scrollView.contentOffset.x = 0
    }
}
```

Chaque fois que l'utilisateur fait défiler l'axe des x, le décalage du contenu de scrollView est remis à 0.
Vous pouvez évidemment changer les `x` s en `y` s et verrouiller ainsi la direction pour être horizontale uniquement.

Vous devez également vous assurer de placer ce code dans la méthode déléguée `scrollViewDidScroll(_ scrollView: UIScrollView)` . Sinon, vous ne pourrez pas le faire fonctionner.

Veillez également à avoir importé `UIScrollViewDelegate` dans votre déclaration de classe, comme ceci:

```
class ViewController: UIViewController, UIScrollViewDelegate
```

... et définissez le délégué de scrollView sur une méthode comme `viewDidLoad(_:)`

```
scrollView.delegate = self
```

Lire UIScrollView en ligne: https://riptutorial.com/fr/ios/topic/1575/uiscrollview

Chapitre 181: UIScrollView avec enfant StackView

Examples

Un exemple complexe de StackView dans Scrollview

Voici un exemple de ce qui peut être fait avec StackViews imbriqués, donnant à l'utilisateur l'impression d'une expérience de défilement continu utilisant des éléments d'interface utilisateur ou des alignements complexes.

Prévention de la mise en page ambiguë

Une question fréquente à propos de StackViews dans Scrollviews provient des alertes ambiguës avec / heigh sur le générateur d'interface. Comme l'explique cette réponse , il est nécessaire de:

1. Ajouter dans UIScrollView un UIView (le contentScrollView);
2. Dans ce contentScrollView, définissez les marges supérieure, inférieure, gauche et droite sur 0
3. Définir également aligner le centre horizontalement et verticalement;

Faire défiler le contenu à l'intérieur de StackViews imbriquées

Le gros truc sur le défilement consiste à déterminer le décalage nécessaire pour présenter (par exemple) un **Textfield dans un StackView avec est à l'intérieur de ScrollView** .

Si vous essayez d'obtenir **la position de `Textfield.frame.minY` peut être 0** , car le frame minY ne prend en compte que la distance entre l'élément et le sommet du StackView. Donc, vous devez tenir **compte de toutes les autres vues / vues de la pile parente.**

Une bonne solution consiste à:

1 - Implémenter l'extension 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)
    }

}
```

Cela prendra en compte toutes les vues parentes et la somme des décalages nécessaires pour la vue de défilement présente la vue nécessaire à l'écran (par exemple, un champ de texte qui ne peut pas rester derrière le clavier de l'utilisateur)

Exemple d'utilisation:

```
func textViewDidBeginEditing(_ textView: UITextView) {
    self.contentScrollView.scrollToShowView(view: textView)
}
```

Lire UIScrollView avec enfant StackView en ligne:
https://riptutorial.com/fr/ios/topic/9404/uiscrollview-avec-enfant-stackview

Chapitre 182: UISearchController

Syntaxe

- UISearchController (searchResultsController: UIViewController?) // Transmet nil comme paramètre si le contrôleur de mise à jour de recherche affiche également le contenu pouvant faire l'objet d'une recherche.
- func updateSearchResults (pour searchController: UISearchController) // Méthode requise pour implémenter lors de l'adoption du protocole UISearchResultsUpdating

Paramètres

Paramètre	Détails
`UISearchController.searchBar`	La barre de recherche à installer dans votre interface. *(lecture seulement)*
`UISearchController.searchResultsUpdater`	L'objet responsable de la mise à jour du contenu du contrôleur de résultats de recherche.
`UISearchController.isActive`	L'état présenté de l'interface de recherche.
`UISearchController.obscuresBackgroundDuringPresentation`	Un booléen indiquant si le contenu sous-jacent est masqué lors d'une recherche.
`UISearchController.dimsBackgroundDuringPresentation`	Un booléen indiquant si le contenu sous-jacent est estompé lors d'une recherche.
`UISearchController.hidesNavigationBarDuringPresentation`	Un booléen indiquant si la barre de navigation doit être masquée lors de la recherche.
`UIViewController.definesPresentationContext`	Valeur booléenne indiquant si la vue de ce contrôleur de vue est couverte lorsque le contrôleur de vue ou l'un de ses descendants présente un contrôleur de vue.
`UIViewController.navigationItem.titleView`	Une vue personnalisée affichée au centre de la barre de navigation lorsque le récepteur est le premier

Paramètre	Détails
	élément dans lequel une barre de recherche peut être placée.
`UITableViewController.tableView.tableHeaderView`	Renvoie une vue d'accessoire affichée au-dessus de la table dans laquelle une barre de recherche peut être placée.

Remarques

Référence du cadre UIKit:

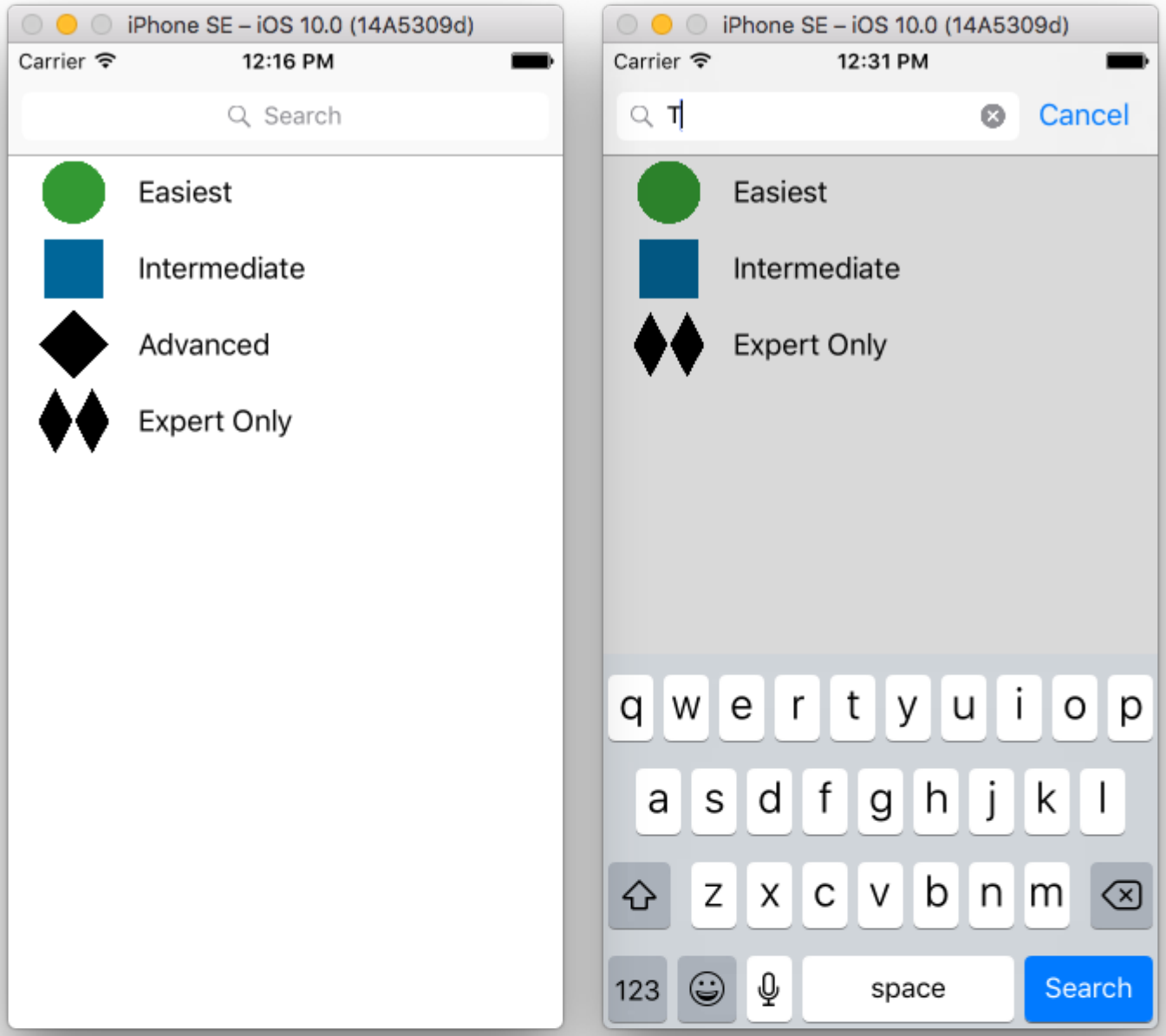
UISearchController

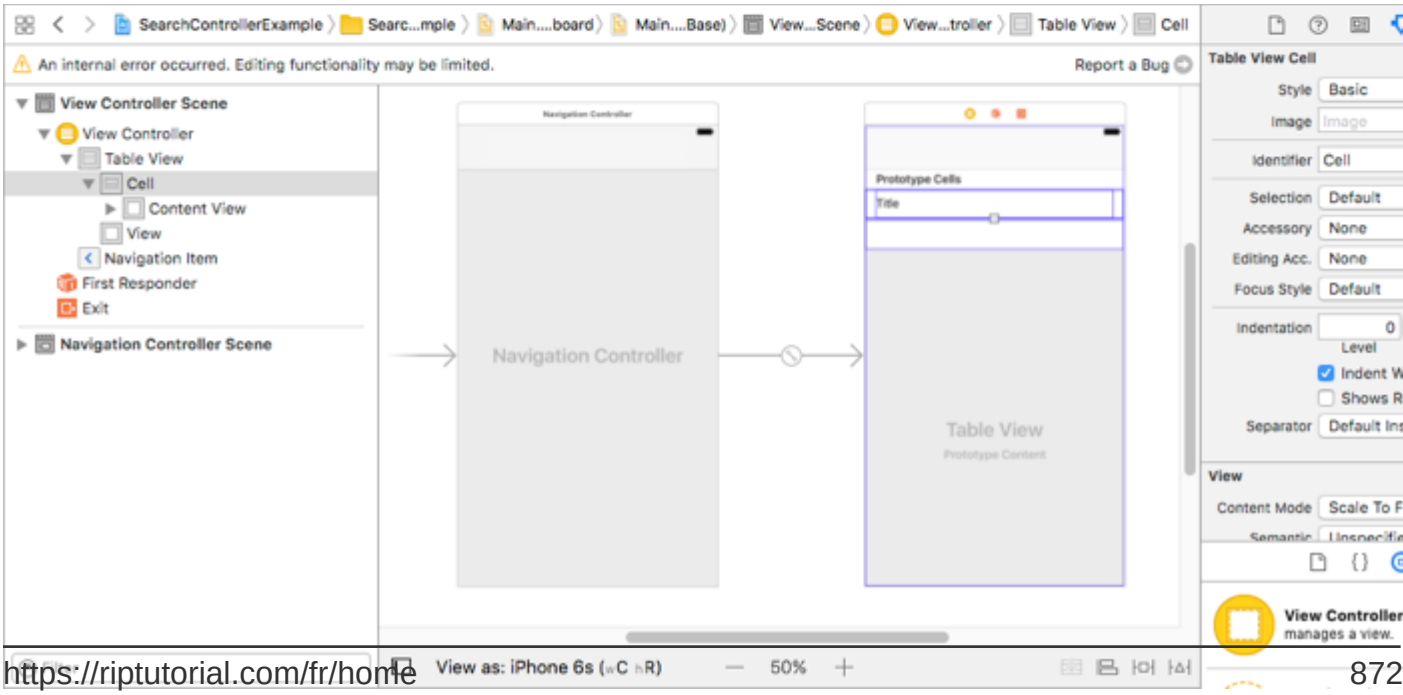
UISearchResultsUpdating

Examples

Barre de recherche dans le titre de la barre de navigation

Cet exemple utilise un contrôleur de recherche pour filtrer les données dans un contrôleur de vue de table. La barre de recherche est placée dans la barre de navigation dans laquelle la vue de table est incorporée.


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
Indent While Editing
Separator
Default Insets
View
Content Mode
Scale To Fill
View Controller
manages a view.
View as: iPhone 6s
50%

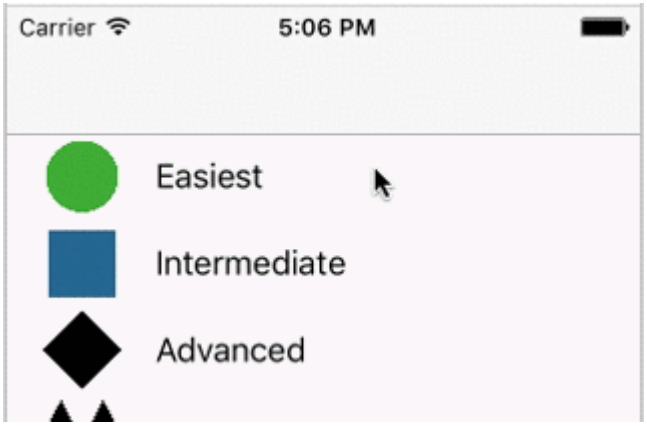
`UINavigationItem` (qui contient la barre de navigation). Ensuite, définissez notre classe ViewController personnalisée pour qu'elle hérite de `UITableViewController` et adopte le protocole `UISearchResultsUpdating` .

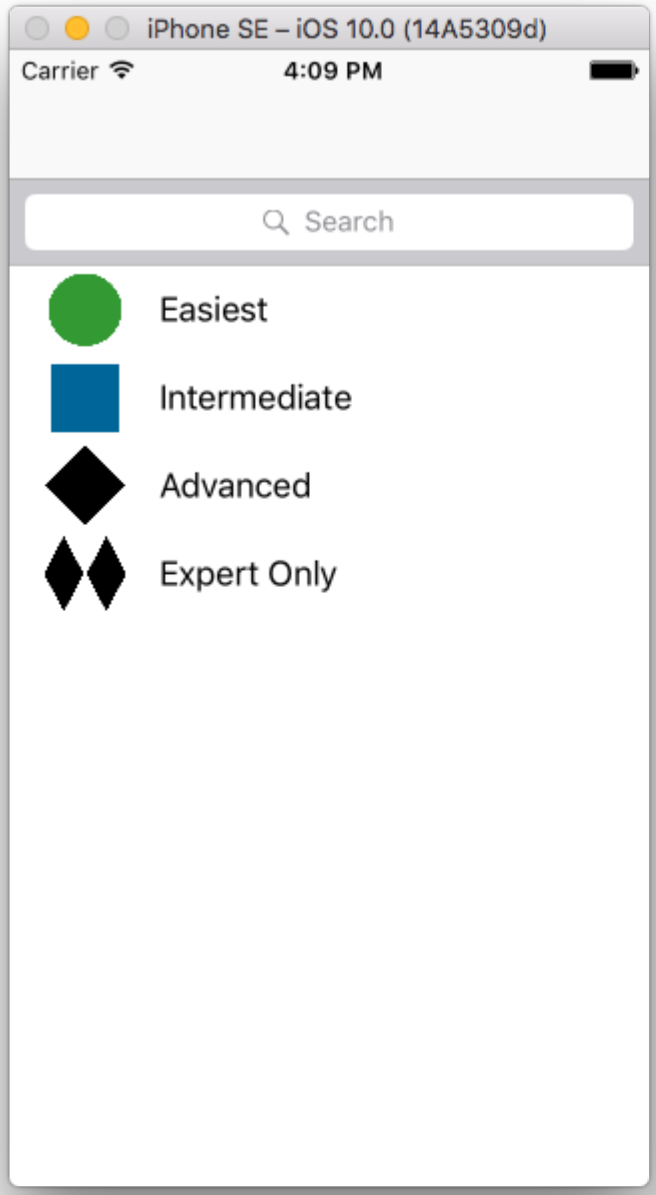
```
class ViewController: UITableViewController, UISearchResultsUpdating {

    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
    }
```

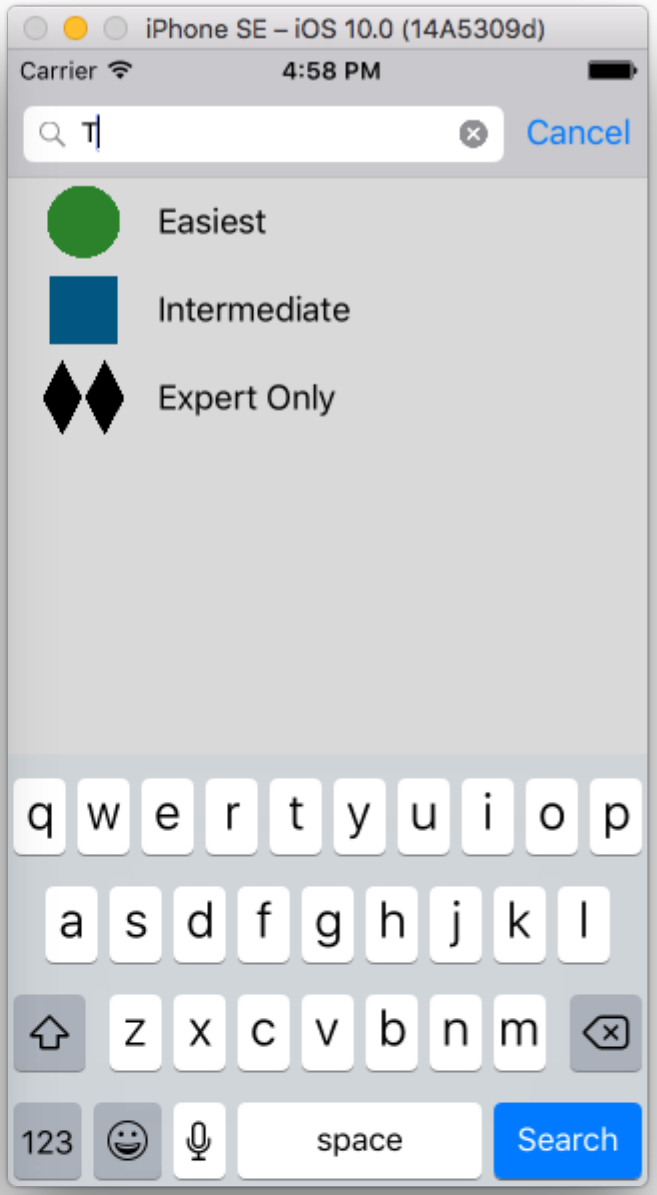
```
    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        // If the search bar is active, use the searchResults data.
        let entry = searchController.isActive ?
                    searchResults[indexPath.row] : entries[indexPath.row]

        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
        cell.textLabel?.text = entry.title
        cell.imageView?.image = UIImage(named: entry.image)
        return cell
    }
}
```

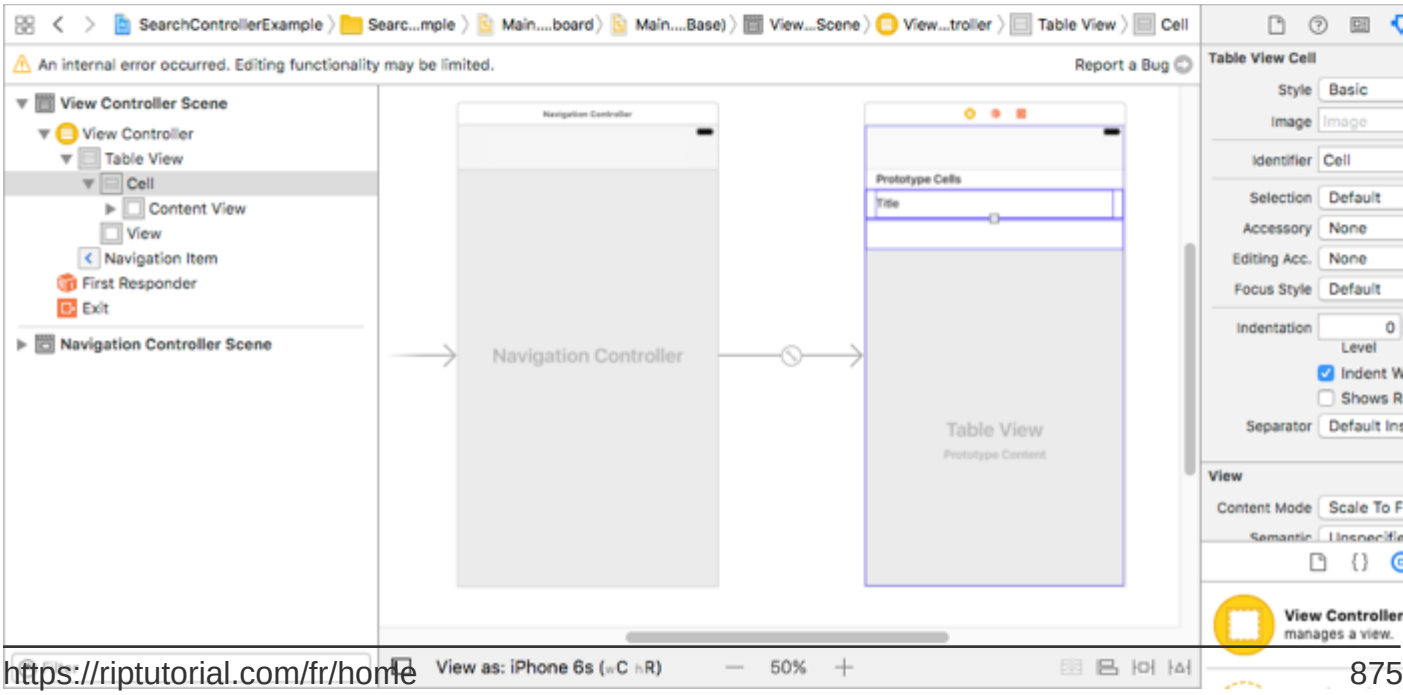
Barre de recherche dans l'en-tête de vue de table

Cet exemple utilise un contrôleur de recherche pour filtrer les cellules dans un contrôleur de vue de table. La barre de recherche est placée dans la vue d'en-tête de la vue de table. Le contenu de la vue de table est décalé à la même hauteur que la barre de recherche, de sorte que la barre de recherche est masquée au début. Lors du défilement vers le haut du bord supérieur de la vue de la table, la barre de recherche est révélée. Ensuite, lorsque la barre de recherche devient active, elle masque la barre de navigation.




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 - A controller manages a view.
View as: iPhone 6s
50%

qui provient du protocole `UISearchResultsUpdating` :

```
func updateSearchResultsForSearchController(searchController: UISearchController) {

}
```

UISerachController dans 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.
}
```

Lire UISearchController en ligne: https://riptutorial.com/fr/ios/topic/2813/uisearchcontroller

Chapitre 183: UISegmentedControl

Introduction

Un objet UISegmentedControl est un contrôle horizontal composé de plusieurs segments, chaque segment fonctionnant comme un bouton discret. Un contrôle segmenté offre un moyen compact de regrouper un certain nombre de contrôles.

Examples

Créer UISegmentedControl via le code

1. Créez une nouvelle instance de UISegmentedControl contenant 3 éléments (segments):

```
let mySegmentedControl = UISegmentedControl (items: ["One", "Two", "Three"])
```

2. Cadre de configuration;

```
mySegmentedControl.frame = CGRect(x: 0.0, y: 0.0, width: 300, height: 50)
```

3. Effectuez la sélection par défaut (les segments ne sont pas indexés par 0):

```
mySegmentedControl.selectedSegmentIndex = 0
```

4. Configurez la cible:

```
mySegmentedControl.addTarget(self, action: #selector(segmentedValueChanged(_:)), for:
.valueChanged)
```

5 La valeur de la poignée a changé:

```
func segmentedValueChanged(_ sender:UISegmentedControl!) {
    print("Selected Segment Index is : \(sender.selectedSegmentIndex)")
}
```

6. Ajouter UISegmentedControl à la hiérarchie des vues

```
yourView.addSubview(mySegmentedControl)
```

Lire UISegmentedControl en ligne: https://riptutorial.com/fr/ios/topic/9963/uisegmentedcontrol

Chapitre 184: UISlider

Examples

UISlider

Objectif c

Déclarez une propriété de curseur dans `ViewController.h` ou dans l'interface de `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];
```

Traiter l'événement de changement de curseur

```
- (IBAction)sliderAction:(id)sender {
    NSLog(@"Slider Value %f", sender.value);
}
```

Exemple 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)
```

Gestion de l'événement de changement de curseur

```
 func sliderAction(sender:UISlider!)
 {
    print("value--\(sender.value)")
```

```
}
```

Ajouter une image personnalisée

Pour ajouter une image personnalisée pour le curseur du curseur, appelez simplement la méthode setThumbImage avec votre image personnalisée:

Swift 3.1:

```
let slider = UISlider()
let thumbImage = UIImage
slider.setThumbImage(thumbImage, for: .normal)
```

Lire UISlider en ligne: https://riptutorial.com/fr/ios/topic/7402/uislider

Chapitre 185: UISplitViewController

Remarques

`UISplitViewController` est une classe de conteneur comme `UITabViewController` , `UINavigationController` . Il sépare la vue principale en deux contrôleurs de vue masterViewController (PrimaryViewController) et detailViewController (SecondaryViewController). nous pouvons envoyer un tableau avec deux contrôleurs de vue et Apple recommande à `UISplitViewController` tant que rootviewcontroller pour votre application. Pour interagir entre les contrôleurs de vue, j'utilise `NSNotificationCenter` .

Examples

Interaction entre la vue maître et la vue détaillée à l'aide des délégués de l'objectif C

`UISplitViewController` doit être le rootViewController de votre application.

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;
}
```

Créez simplement un objet pour `UISplitViewController` et définissez-le en tant que rootviewcontroller pour votre application.

SplitViewController.h

```
#import <UIKit/UIKit.h>
#import "MasterViewController.h"
#import "DetailViewController.h"
@interface ViewController : UISplitViewController
{
DetailViewController *detailVC;
MasterViewController *masterVC;
NSMutableArray *array;
}
@end
```

`MasterViewController` est toujours sur le côté gauche de l'appareil , vous pouvez régler la largeur `UISplitViewCOntroller` méthodes de délégué et `DetailViewController` est sur le côté droit de l'application

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;
}
```

Les contrôleurs ViewControllers maîtres et de détails créés sont ajoutés à un tableau défini sur `self.viewControllers` dans `UISplitViewController` . `self.preferredDisplayMode` est le jeu de modes pour l'affichage de la documentation Apple `DetailViewController` et `DetailViewController` pour DisplayMode . `self.presentsWithGesture` active le geste de balayage pour afficher `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
```

Créez un délégué `DetailViewDelegate` avec la `sendSelectedNavController:(UIViewController *)viewController` pour envoyer le `UIViewController` au `DetailViewcontroller` . Ensuite, dans `MasterViewController` le mainTableView est la tableview dans le gauche. Le viewControllerArray contient tous les `UIViewControllers` à afficher dans `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
```

Créez des `UIViewControllers` et `UIViewControllers` -les à un tableau. La vue Table est initialisée puis sur la méthode `didSelectRowAtIndexPath` J'envoie un `UIViewController` au `DetailViewController` utilisant `detailDelegate` avec le `UIViewController` correspondant dans le tableau

DetailViewController.h

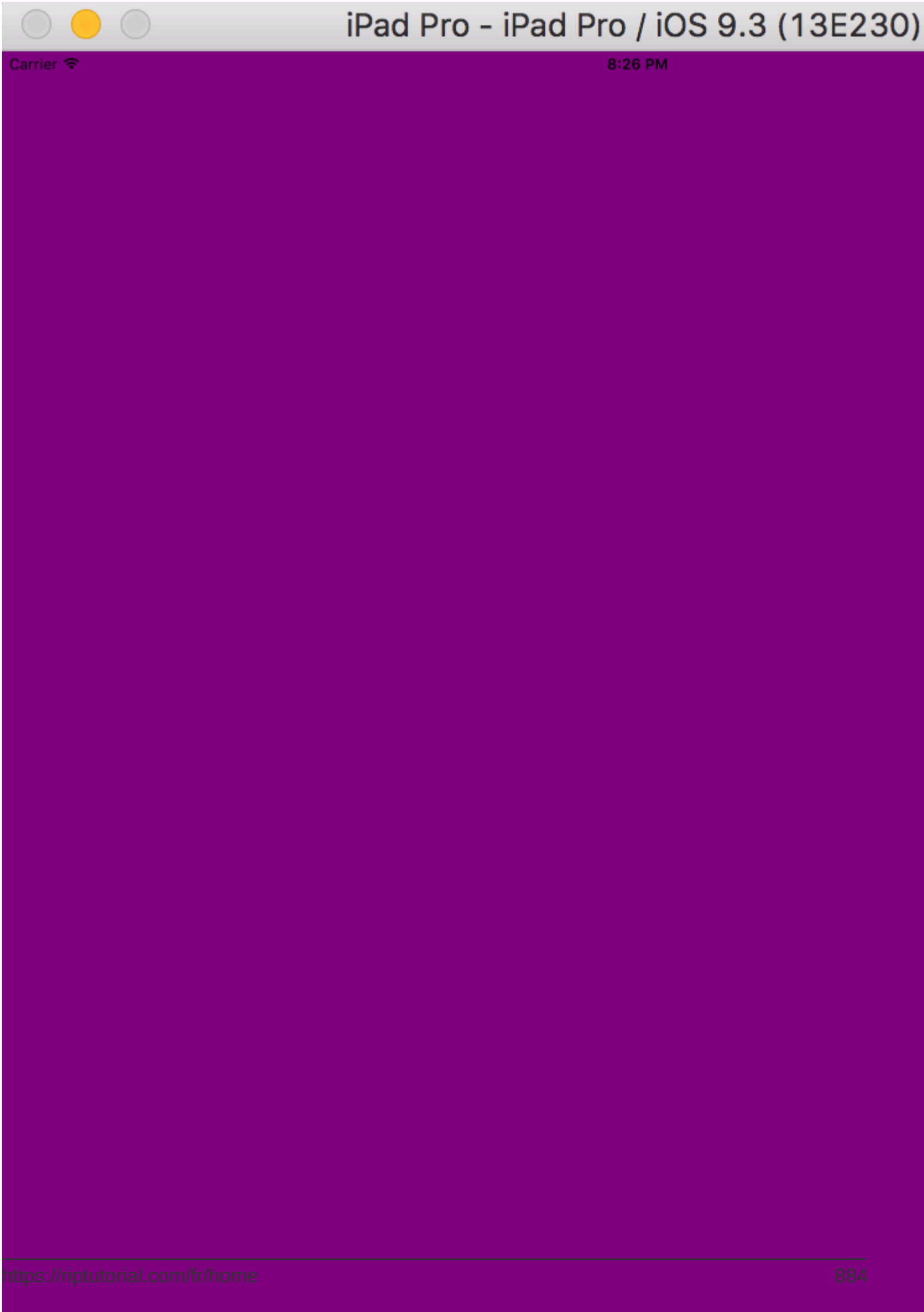
```
#import <UIKit/UIKit.h>

@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
```

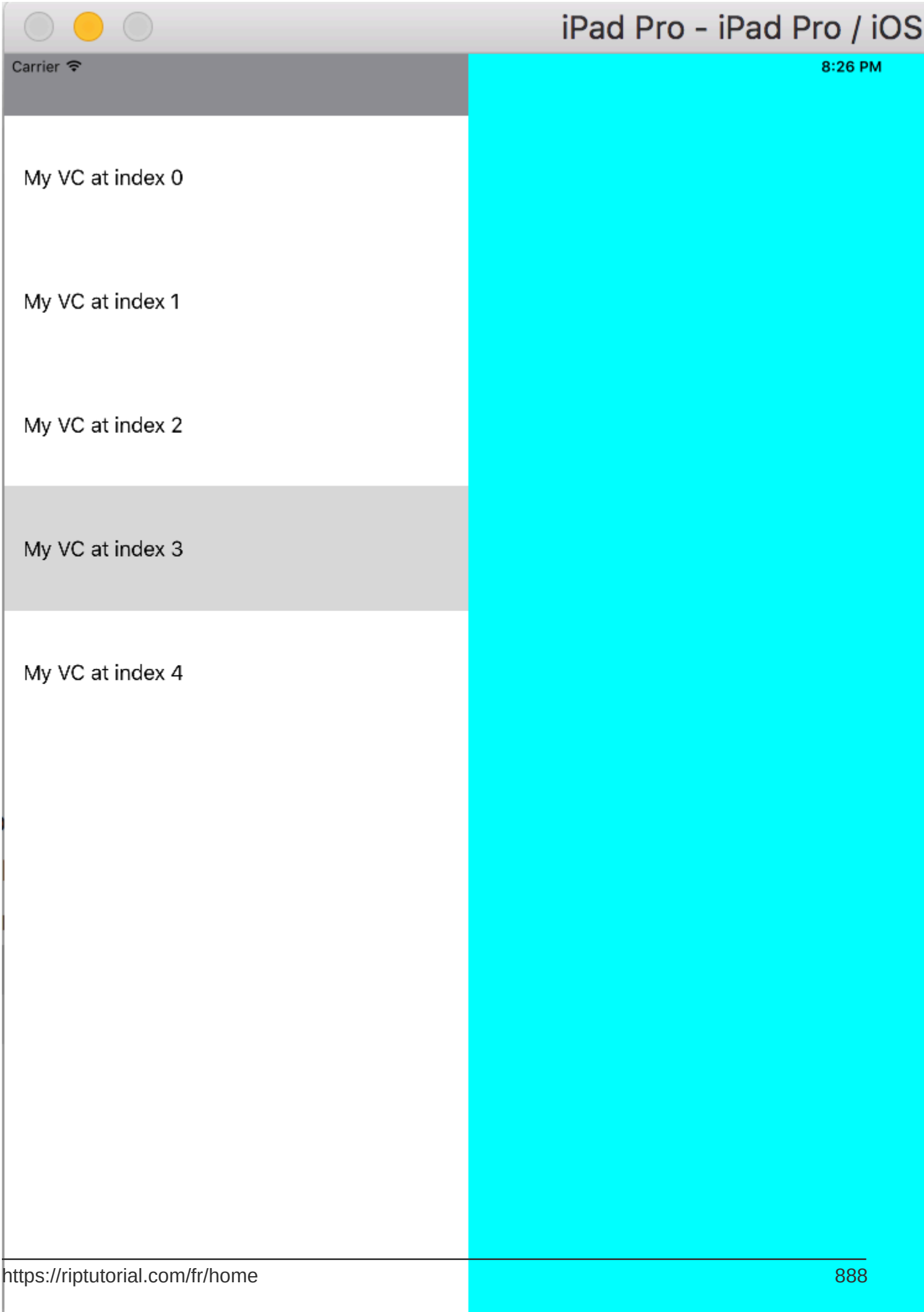
`sendSelectedNavController` est déclaré ici avec la suppression de toutes les vues dans le `DetailViewController` et l'ajout du `UIViewController` passé à `UIViewController` de `MasterViewController`

Ajout de quelques captures d'écran de l'application


iPad Pro - iPad Pro / iOS 9.3 (13E230)
Carrier
8:26 PM

`preferredDisplayMode` comme automatique glisser l'écran , nous obtenons le `MasterViewController` comme attaché à l'image ci - dessous , mais en mode paysage que nous obtenons à la fois le `MasterViewController` et `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

Chapitre 186: UISplitViewController

Remarques

Dans iOS 8 et versions ultérieures, vous pouvez utiliser la classe `UISplitViewController` sur tous les périphériques iOS. Dans les versions précédentes d'iOS, la classe est uniquement disponible sur iPad. `UISplitViewController` est une classe de conteneur comme `UITabViewController` , `UINavigationController` . Il sépare la vue principale en deux `UIViewControllers` `masterViewController` (PrimaryViewController) et `detailViewController` (SecondaryViewController). nous pouvons envoyer un `NSArray` avec deux `UIViewControllers` et Apple recommande `UISplitViewController` comme rootviewcontroller pour votre application. Pour interagir entre les `UIViewControllers` j'utilise `NSNotificationCenter` .

Examples

Interaction entre la vue maître et la vue détaillée à l'aide des délégués dans l'objectif C

`UISplitViewController` besoin du contrôleur de vue racine de la fenêtre de votre application

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;
}
```

Créez simplement un objet pour votre `UISplitVIewController` et définissez-le comme rootViewController pour votre application.

SplitViewController.h

```
#import <UIKit/UIKit.h>
#import "MasterViewController.h"
#import "DetailViewController.h"
@interface ViewController : UISplitViewController
{
DetailViewController *detailVC;
MasterViewController *masterVC;
NSMutableArray *array;
}
```

```
@end
```

`MasterViewController` est un `UIViewController` qui est situé sur le côté gauche de l'appareil , vous pouvez régler la largeur en `UISplitViewController` utilisant `maximumPrimaryColumnWidth` et `DetailViewController` est sur le côté droit

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;
}
```

Les maîtres et détails `UIViewController` sont ajoutés à un `NSArray` défini sur `self.viewControllers` . `self.preferredDisplayMode` est le mode défini pour l'affichage de `MasterViewController` et `DetailViewController` . `self.presentsWithGesture` active le geste de glissement pour afficher `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
```

Créez une `DetailViewDelegate` Delegate avec `sendSelectedNavController` pour envoyer les `UIViewControllers` au `DetailViewController` . Ensuite, dans `MasterViewController` un `UITableView` est créé. `ViewControllerArray` contient tous les `UIViewControllers` à afficher dans `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
```

Créé un `UIViewController` et l'a ajouté à un `NSMutableArray` . Le `UITableView` est initialisé puis sur `didselectrowatindexpath` méthode que j'envoie un `UIViewController` au `DetailViewController` en utilisant `detailDelegate` délégué avec le correspondant `UIViewController` dans le `NSMutableArray` en tant que paramètre

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
```

Le `sendSelectedNavController` est déclaré ici avec la suppression de tous les `UIView` dans le `DetailViewController` et en ajoutant le `UIViewController` passé à `UIViewController` de `MasterViewController` .

Lire UISplitViewController en ligne: https://riptutorial.com/fr/ios/topic/4844/uisplitviewcontroller

Chapitre 187: UIStackView

Examples

Créer une vue de pile horizontale par programme

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.
```

Rapide

```
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.
```

Objectif 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.
```

Créer une vue de pile verticale par programme

Rapide

```
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.
```

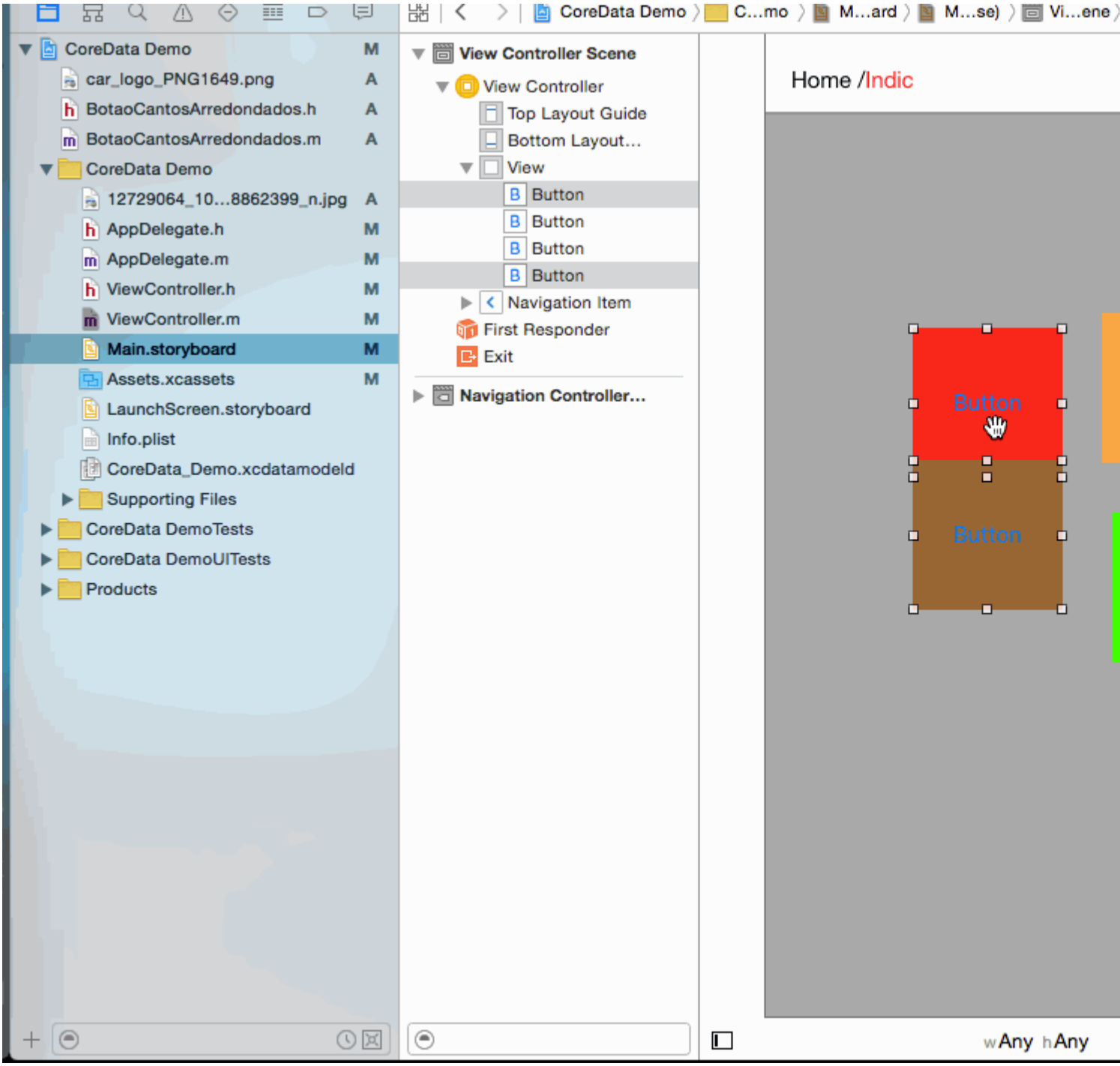
Objectif c

```
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.
```

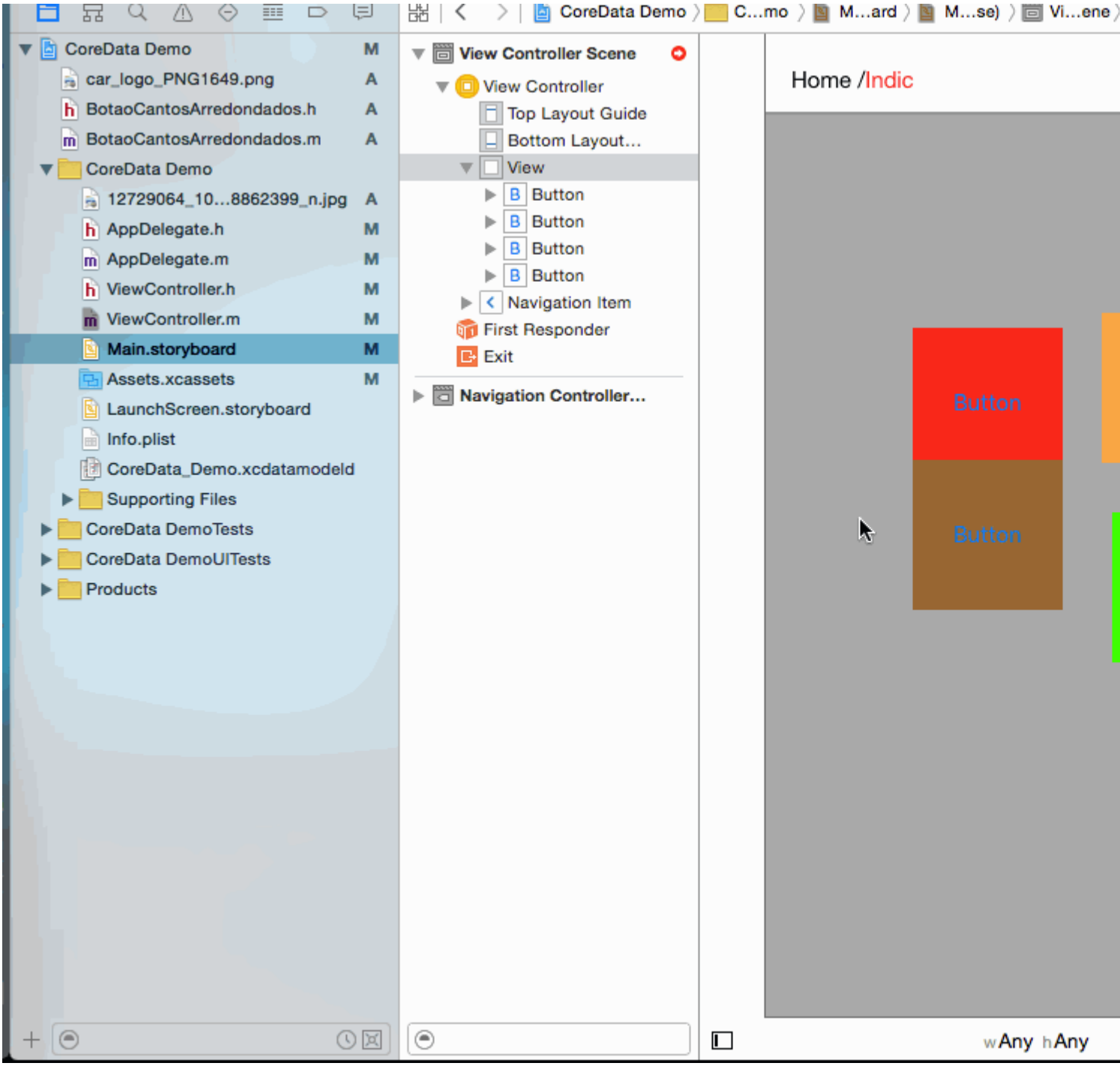
Boutons centraux avec UIStackview

Étape 1: - Prenez 4 boutons dans votre Storyboard. Button1, Button2, Button 3, Button4

Étape 2: - Donner la hauteur et la largeur fixes à tous les boutons.

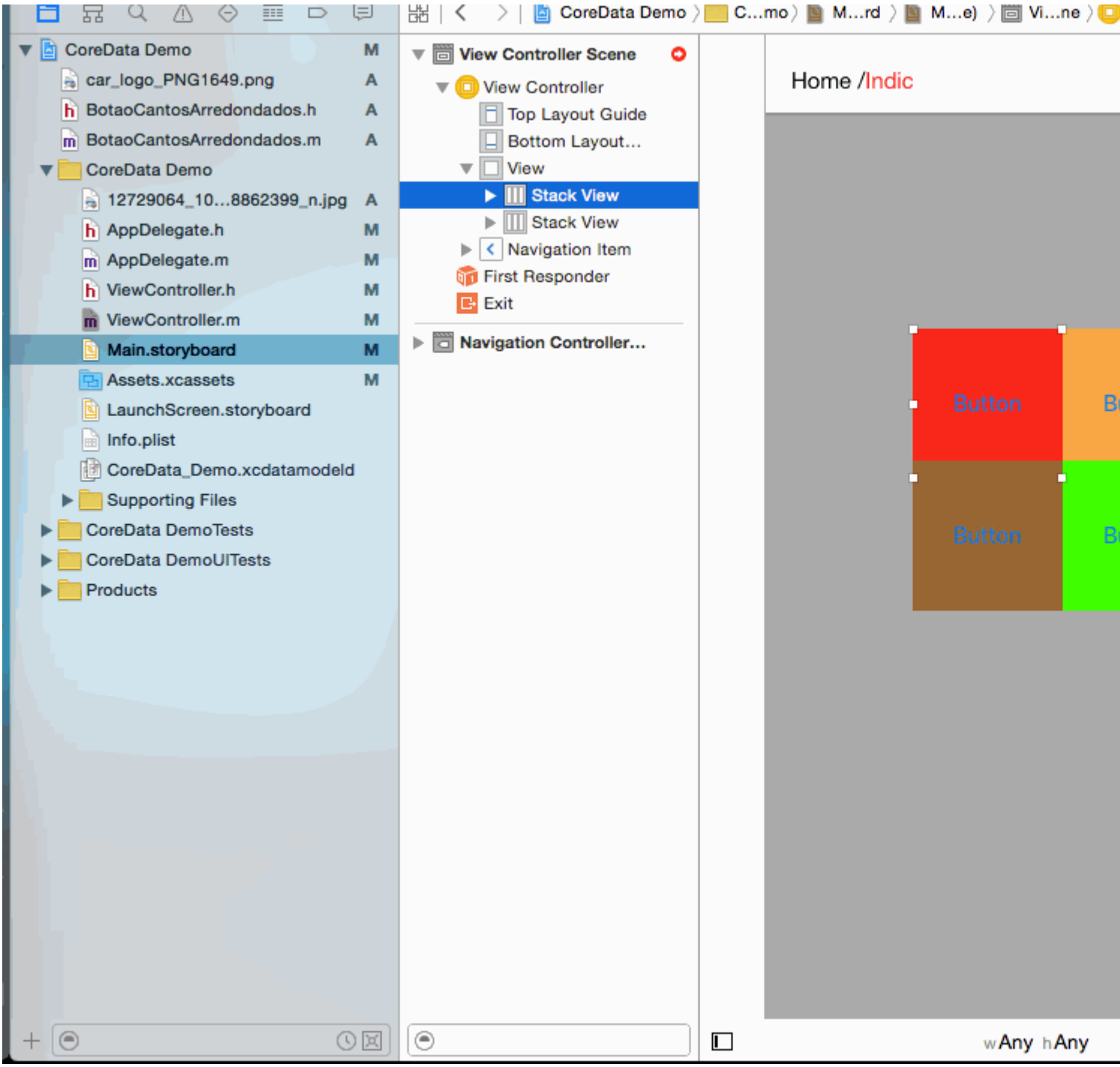


Étape 3: - Toutes les paires de boutons 2 - 2 dans la vue 2 piles.

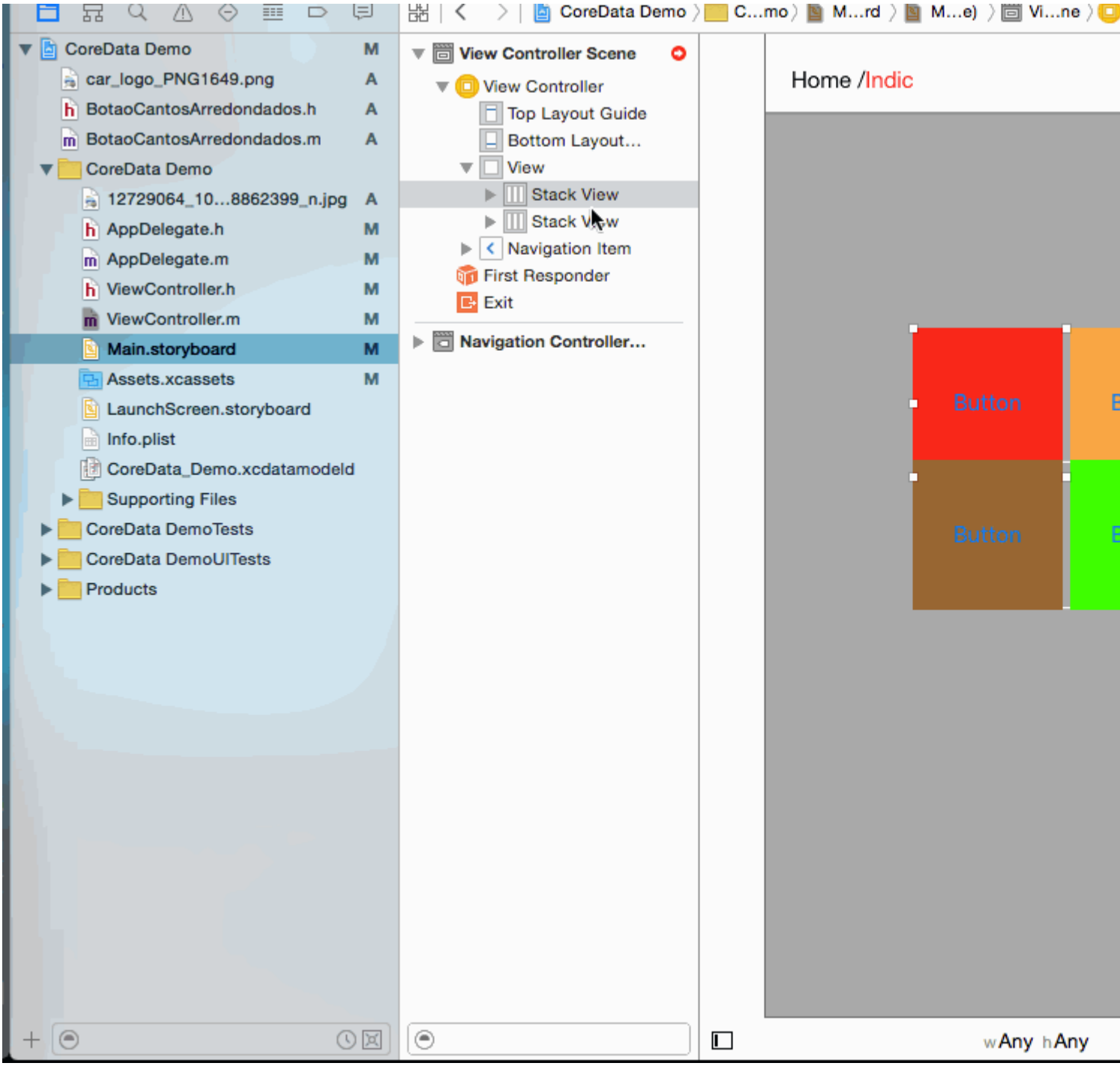


Étape 4: - Définissez la propriété UIStackview pour les deux.

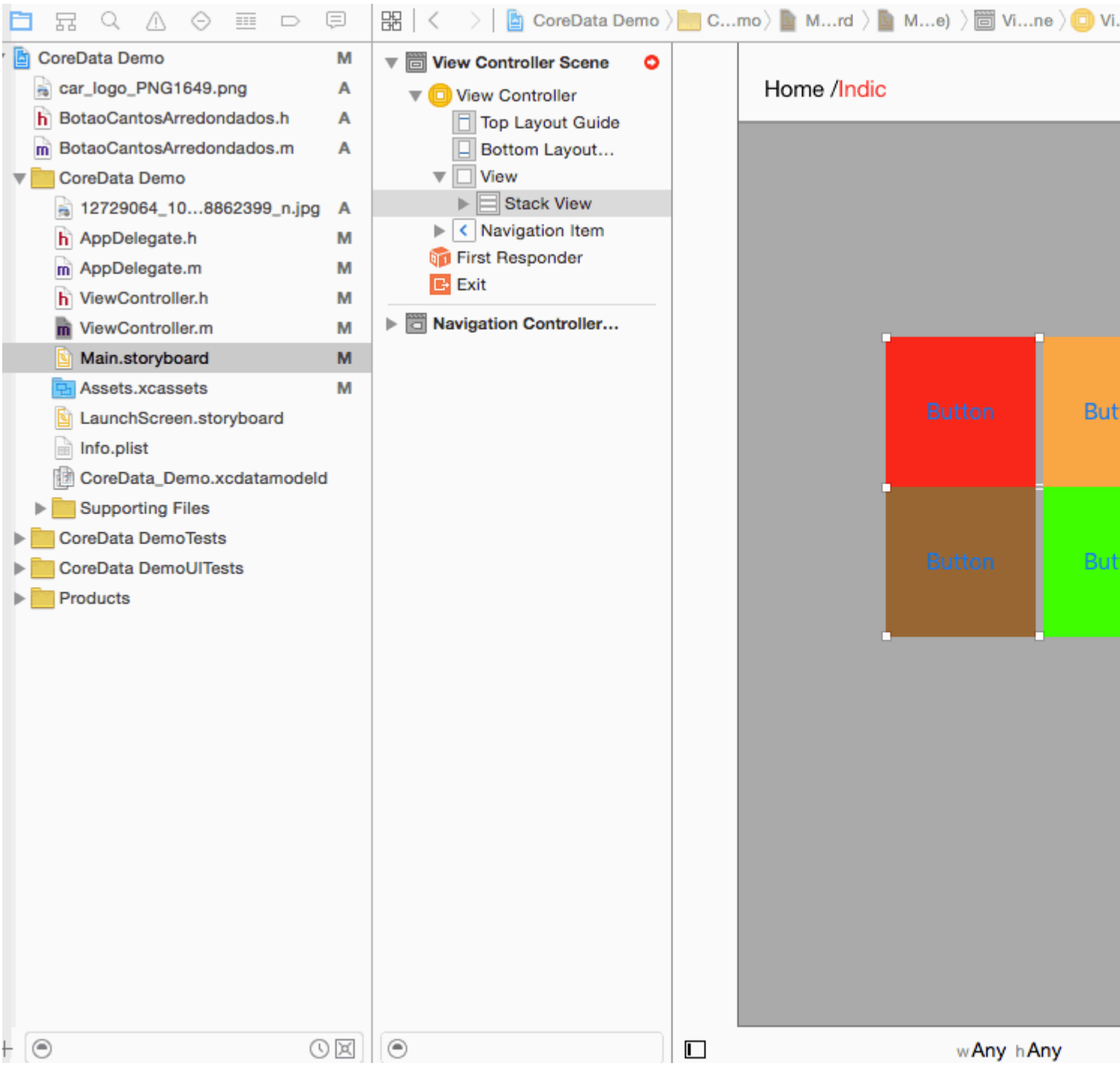
```
    Distribution -> Fill Equally
         Spacing -> 5 (as per your requirement)
```



Étape 5: - Ajouter les deux Stackview dans un Stackview

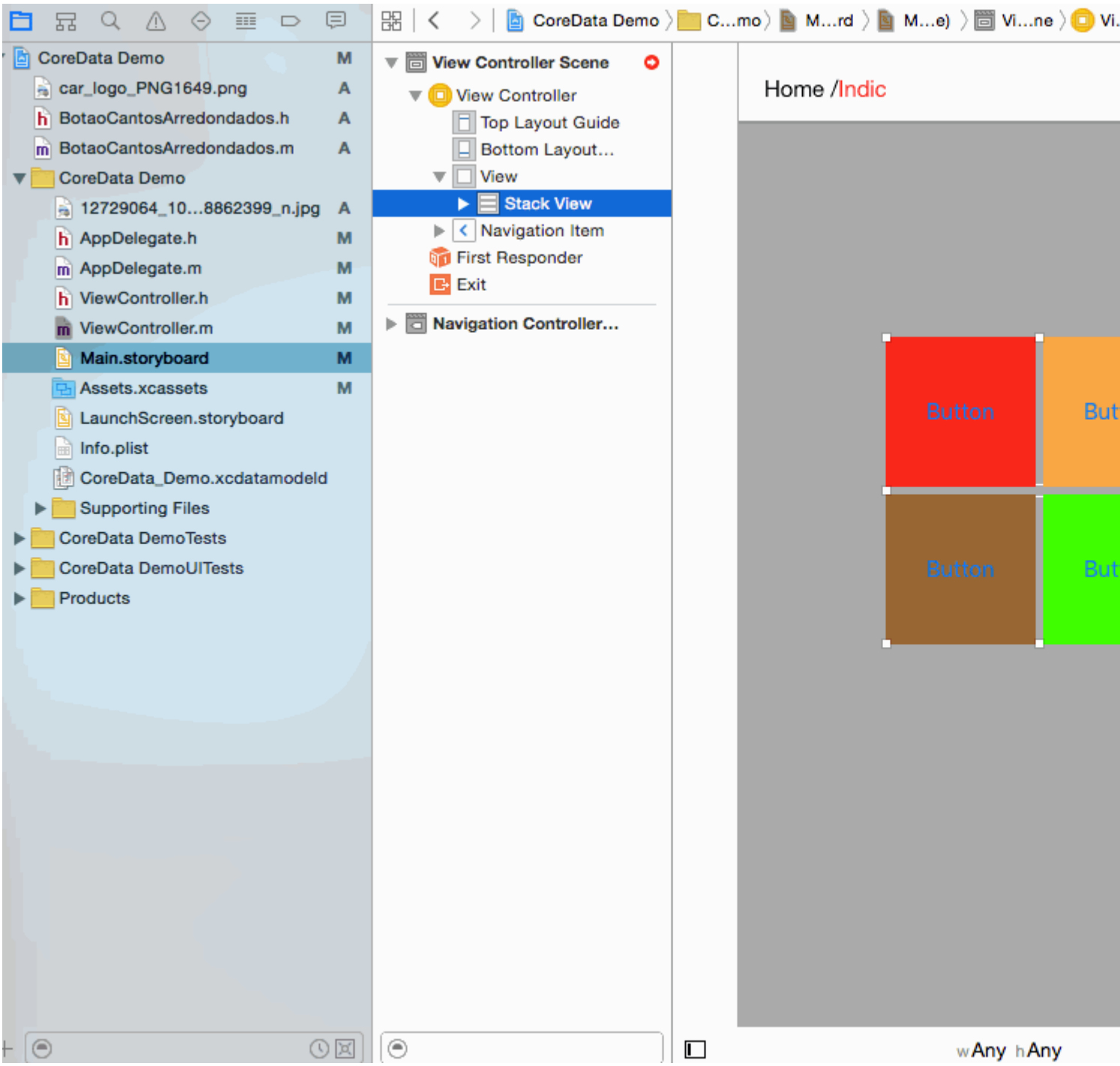


Étape 6: - Définir la `Distribution = Fill equally Spacing =5` dans la vue principale (définie selon vos besoins)

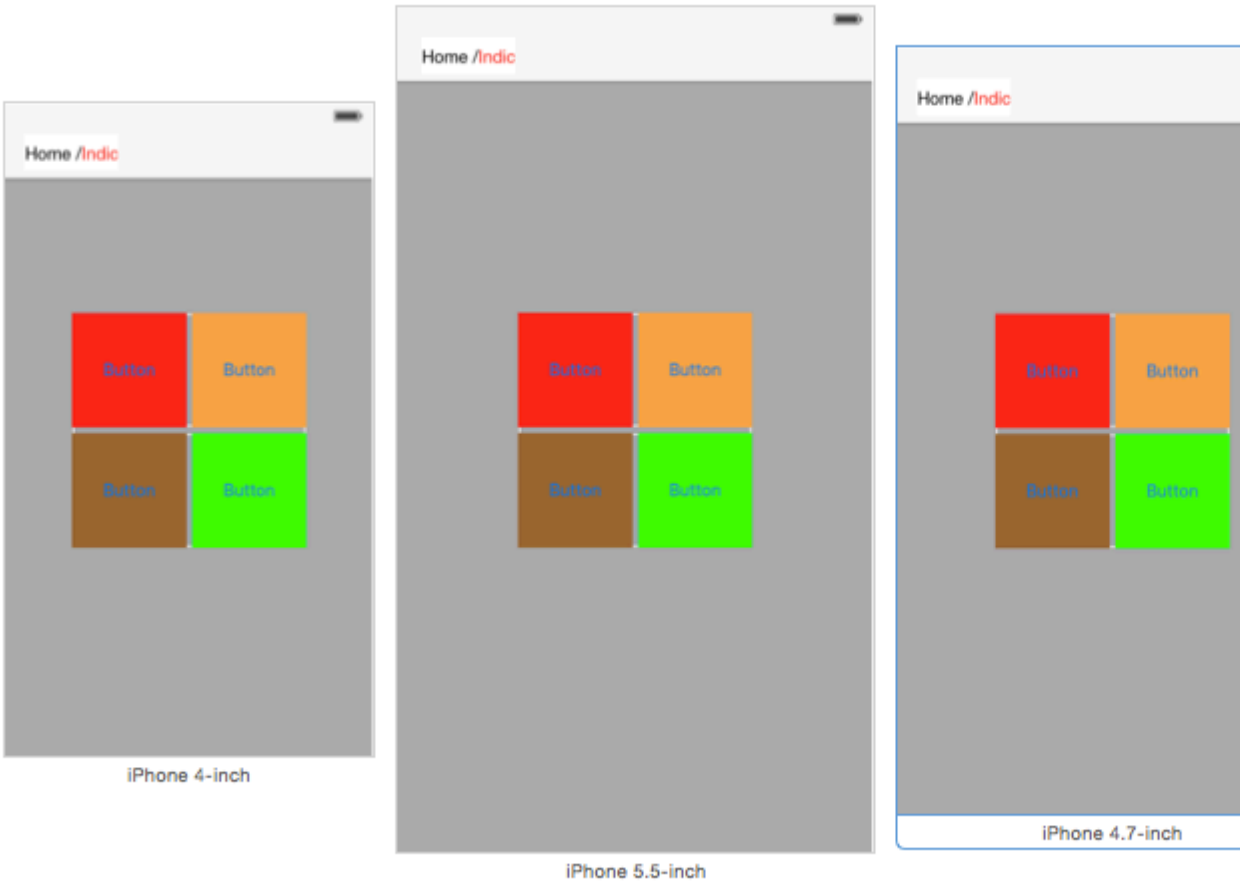


Étape 7: - Maintenant, définissez Constrain sur la pile principale

```
    center Horizontally in container

    center vertically in container

      and select Update Frame.
```



Étape 8: - Il est temps pour le périphérique Output for All.



Lire UIStackView en ligne: https://riptutorial.com/fr/ios/topic/1390/uistackview

Chapitre 188: UIStoryboard

Introduction

Un objet UIStoryboard encapsule le graphique du contrôleur de vue stocké dans un fichier de ressources de storyboard Interface Builder. Ce graphique de contrôleur de vue représente les contrôleurs de vue pour tout ou partie de l'interface utilisateur de votre application.

Examples

Obtenir une instance de UIStoryboard par programmation

RAPIDE:

Vous pouvez obtenir une instance de **UIStoryboard par** programmation comme suit:

```
    let storyboard = UIStoryboard(name: "Main", bundle: nil)
```

où:

- **name** => le nom du storyboard sans extension
- **bundle** => l'ensemble contenant le fichier de storyboard et ses ressources associées. Si vous spécifiez nil, cette méthode examine le bundle principal de l'application en cours.

Par exemple, vous pouvez utiliser l'instance créée ci-dessus pour accéder à un certain **UIViewController** instancié dans ce storyboard:

```
   let viewController = storyboard.instantiateViewController(withIdentifier: "yourIdentifier")
```

OBJECTIF C:

Vous pouvez obtenir une instance de **UIStoryboard** dans Objective-C comme suit:

```
 UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"MainStoryboard" bundle:nil];
```

Exemple d'accès à **UIViewController** instancié dans ce storyboard:

```
MyViewController *myViewController = [storyboard
instantiateViewControllerWithIdentifier:@"MyViewControllerIdentifier"];
```

Ouvrir un autre storyboard

```
let storyboard = UIStoryboard(name: "StoryboardName", bundle: nil)
let vc = storyboard.instantiateViewController(withIdentifier: "ViewControllerID") as
YourViewController
self.present(vc, animated: true, completion: nil)
```

Lire UIStoryboard en ligne: https://riptutorial.com/fr/ios/topic/8795/uistoryboard

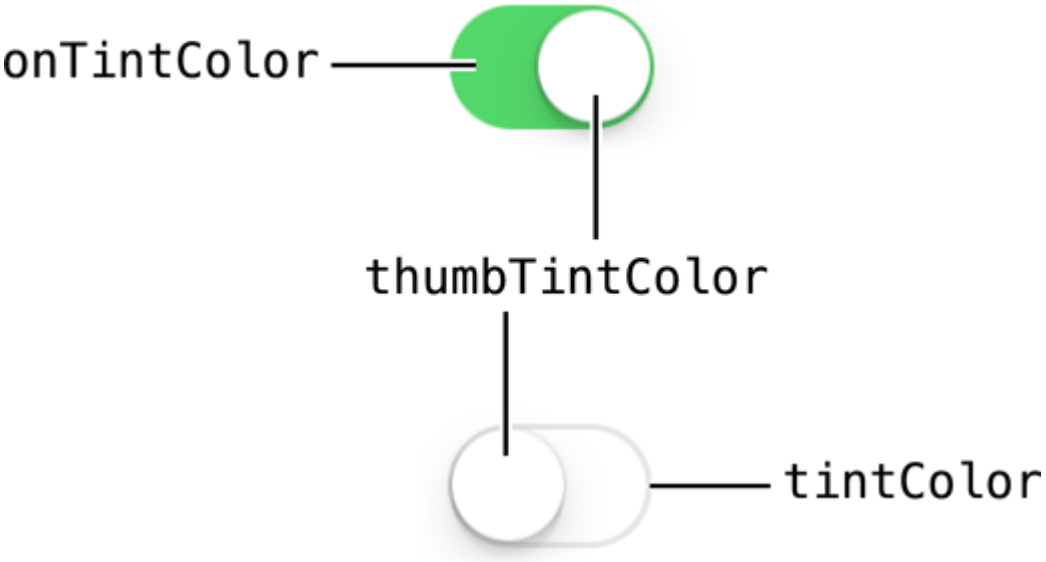
Chapitre 189: UISwitch

Syntaxe

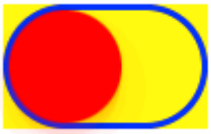
- (instancetype) initWithFrame: (CGRect) frame;
- (void) setOn: (BOOL) sur animé: (BOOL) animé;
- (instancetype nullable) initWithCoder: (NSCoder *) aDecoder;

Remarques

1. Référence UISwitch: Documentation Apple



2. Une autre référence donnée par: Enoch Huang



Examples

Mettre en / hors service

Objectif c

```
[mySwitch setOn:YES];
//or
[mySwitch setOn:YES animated:YES];
```

Rapide

```
mySwitch.setOn(false)
//or
mySwitch.setOn(false, animated: false)
```

Définir la couleur de fond

Objectif 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];
```

Rapide

```
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)
```

Définir la couleur de la teinte

Objectif c

```
//for off-state
mySwitch.tintColor = [UIColor blueColor];
[mySwitch setTintColor: [UIColor blueColor]];

//for on-state
mySwitch.onTintColor = [UIColor cyanColor];
[mySwitch setOnTintColor: [UIColor cyanColor]];
```

Rapide

```
//for off-state
mySwitch.tintColor = UIColor.blueColor()

//for on-state
mySwitch.onTintColor = UIColor.cyanColor()
```

Définir l'image pour l'état On / Off

Objectif 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"]];
```

Rapide

```
//set off-image
mySwitch.offImage = UIImage(named: "off_image")

//set on-image
mySwitch.onImage = UIImage(named: "on_image")
```

Lire UISwitch en ligne: https://riptutorial.com/fr/ios/topic/2182/uiswitch

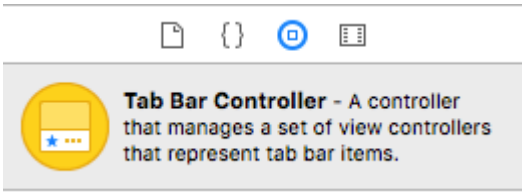
Chapitre 190: UITabBarController

Examples

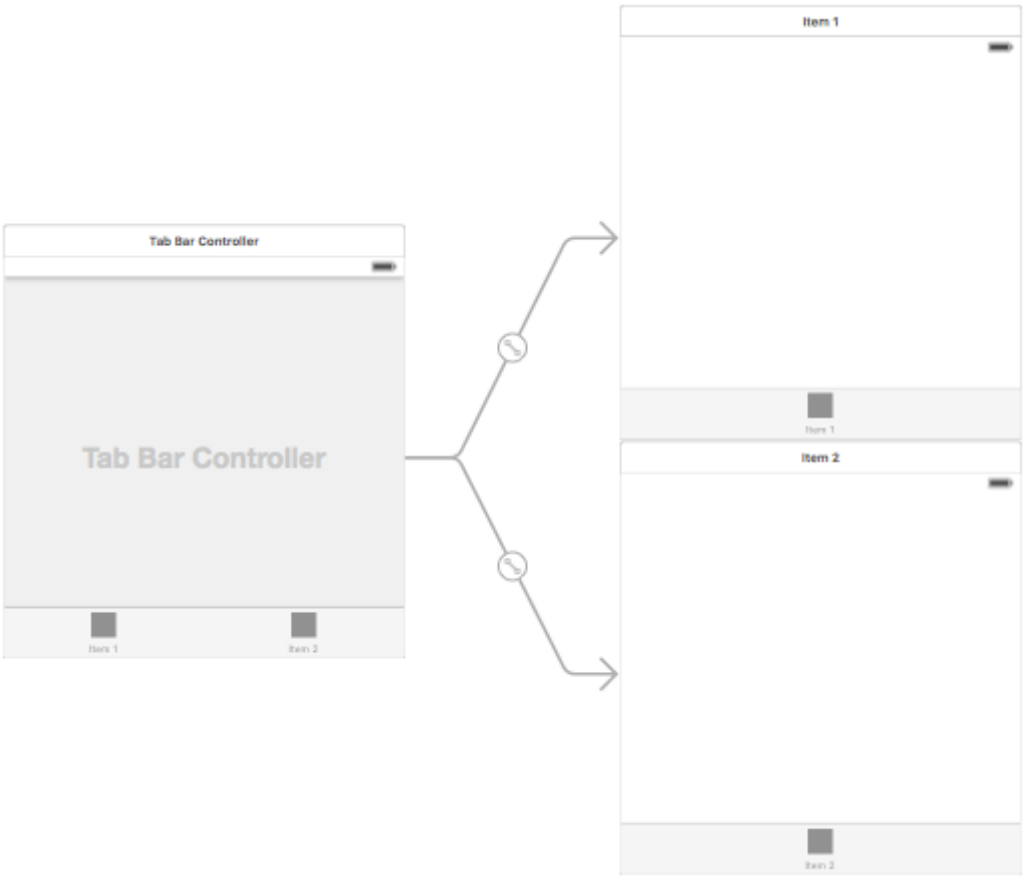
Créer une instance

Une «barre d'onglets» se trouve généralement dans la plupart des applications iOS et est utilisée pour présenter des vues distinctes dans chaque onglet.

Pour créer un contrôleur de barre d'onglets à l'aide du générateur d'interface, faites glisser un contrôleur de barre d'onglets de la bibliothèque d'objets dans le canevas.



Par défaut, un contrôleur de barre de tabulation est livré avec deux vues. Pour ajouter des vues supplémentaires, contrôlez le glisser depuis le contrôleur de la barre d'onglets vers la nouvelle vue et sélectionnez «Contrôler les contrôleurs» dans la liste déroulante.



Modification du titre de la barre d'onglets et de l'icône

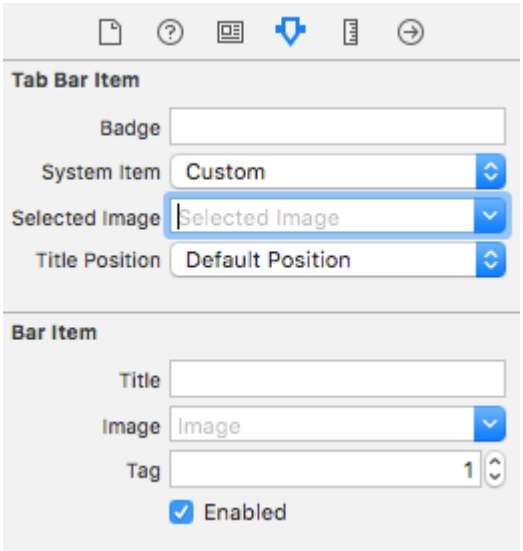
Utiliser le Story Board:

Sélectionnez l'élément de la barre d'onglets dans le contrôleur de vue correspondant et accédez à l'inspecteur d'attributs

Si vous souhaitez une icône et un titre intégrés, définissez l'option «Élément système» sur la valeur correspondante.

Pour une icône personnalisée, ajoutez les images requises au dossier des ressources et définissez l'élément «Élément système» de l'ancien à «personnalisé».

Maintenant, définissez l'icône à afficher lorsque l'onglet est sélectionné dans le menu déroulant "Image sélectionnée" et l'icône de l'onglet par défaut dans le menu déroulant "Image". Ajoutez le titre correspondant dans le champ "titre".



Par programme:

Dans la méthode `viewDidLoad()` du contrôleur de vue, ajoutez le code suivant:

Objectif c:

```
self.title = @"item";

self.tabBarItem.image = [UIImage imageNamed:@"item"];
self.tabBarItem.selectedImage = [UIImage imageNamed:@"item_selected"];
```

Rapide:

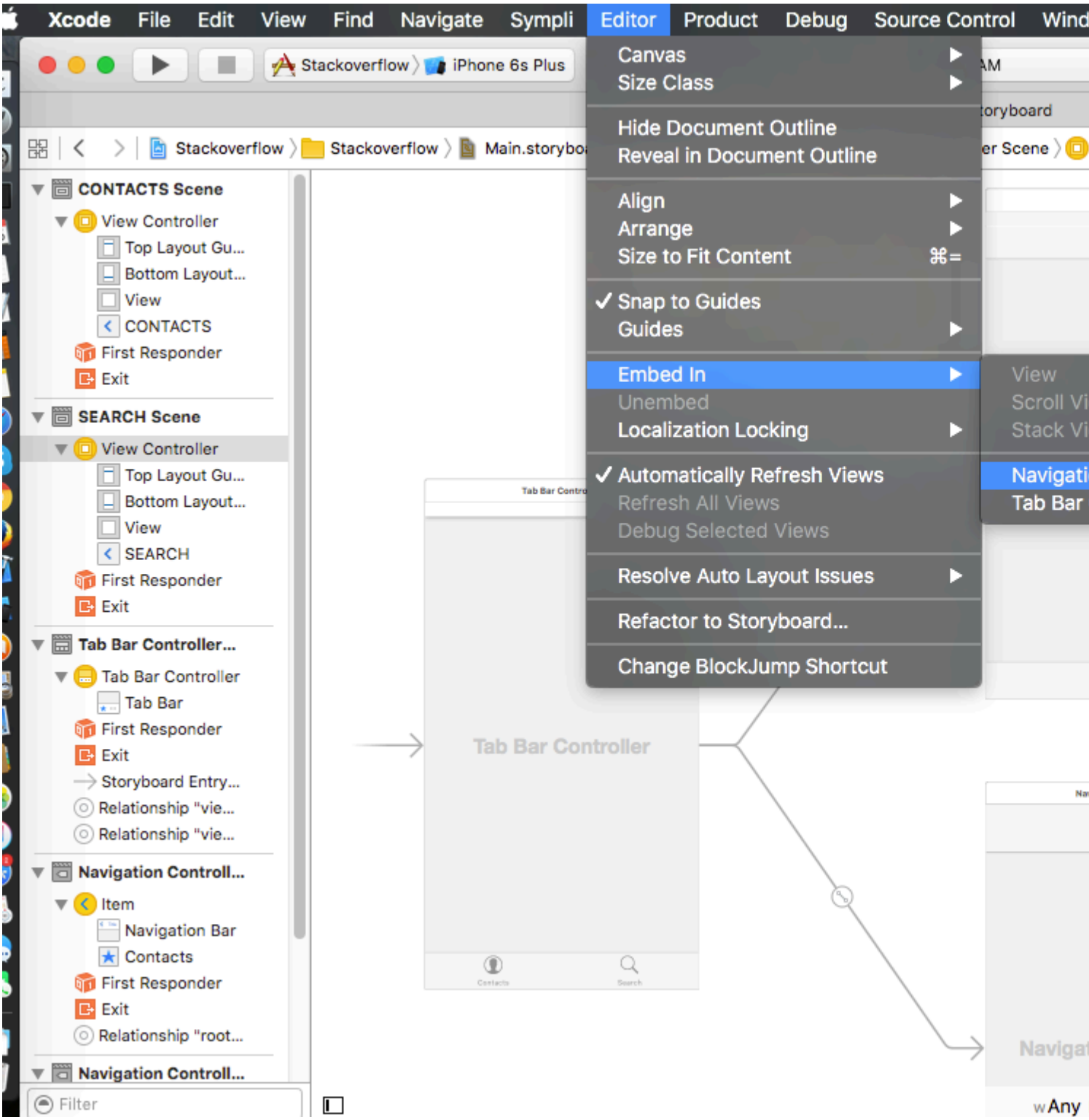
```
self.title = "item"
self.tabBarItem.image = UIImage(named: "item")
self.tabBarItem.selectedImage = UIImage(named: "item_selected")
```

Contrôleur de navigation avec TabBar

Le contrôleur de navigation peut être intégré dans chaque onglet en utilisant le storyboard lui-même. Cela peut être comme dans la capture d'écran ajoutée.

Pour ajouter un contrôleur de navigation à un contrôleur de connexion se connectant depuis le contrôleur de la barre d’onglet, voici le flux

- Sélectionnez le contrôleur de vue pour lequel vous devez ajouter un contrôleur de navigation. Ici, laissez-le être le contrôleur de vue de recherche comme affichage de sélection.
- Dans le menu **Editeur** du Xcode, sélectionnez l'option **Intégrer dans -> Contrôleur de navigation**



Personnalisation de la couleur de la barre d'onglets

```
[[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];
```

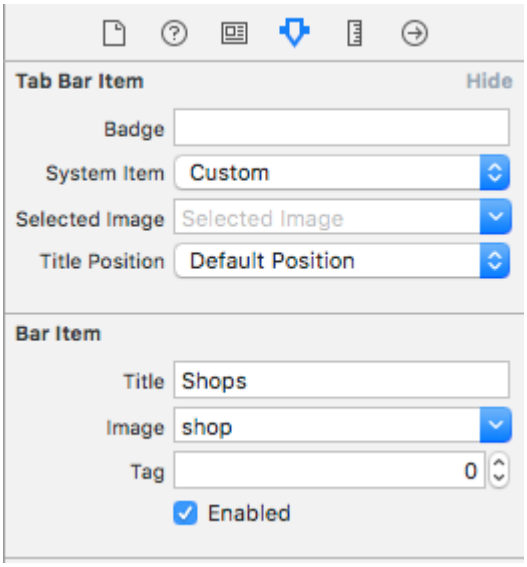
UITabBarController avec sélection de couleurs personnalisée

UITabBarController dans `Swift 3` Modifiez la couleur et le titre de l'image en fonction de la sélection en modifiant la couleur de l'onglet sélectionné.

```
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
    }
```

```
}
```

Choisir l'image pour la barre d'onglets et définir le titre de l'onglet ici



Sélection d'un autre onglet

Créer un contrôleur de barre d'onglets par programmation sans 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
    }
```

Lire UITabBarController en ligne: https://riptutorial.com/fr/ios/topic/2763/uitabbarcontroller

Chapitre 191: UITableView

Introduction

Une vue simple, largement utilisée, mais très puissante, capable de présenter des données sous forme de liste à l'aide de lignes et d'une seule colonne. Les utilisateurs peuvent faire défiler verticalement les éléments d'une vue de table, et éventuellement manipuler et sélectionner du contenu.

Syntaxe

- - (CGFloat) tableView: (UITableView *) tableView heightForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (CGFloat) tableView: (UITableView *) tableView heightForHeaderInSection: (NSInteger) section;
- - (CGFloat) tableView: (UITableView *) tableView heightForFooterInSection: (NSInteger) section;
- - (UIView *) tableView: (UITableView *) tableView viewForHeaderInSection: (NSInteger) section;
- - (UIView *) tableView: (UITableView *) tableView viewForFooterInSection: (NSInteger) section;
- - (UITableViewCellAccessoryType) tableView: (UITableView *) tableView accessoireTypeForRowWithIndexPath: (NSIndexPath *) indexPath
- - (void) tableView: (UITableView *) tableView accessoireButtonTappedForRowWithIndexPath: (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 *) proposalDestinationIndexPath;
- - (NSInteger) tableView: (UITableView *) tableView indentationLevelForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSInteger) tableView: (UITableView *) tableView numberOfRowsInSection: section (NSInteger);
- - (UITableViewCell *) tableView: (UITableView *) tableView cellForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSInteger) numberOfSectionsInTableView: (UITableView *) tableView;
- - (NSString *) tableView: (UITableView *) tableView titleForHeaderInSection: section (NSInteger); // style de police fixe. utiliser la vue personnalisée (UILabel) si vous voulez quelque chose de différent
- - (NSString *) tableView: (UITableView *) tableView titleForFooterInSection: section (NSInteger);
- - (BOOL) tableView: (UITableView *) tableView canEditRowAtIndexPath: (NSIndexPath *) indexPath;
- - (BOOL) tableView: (UITableView *) tableView canMoveRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSArray *) sectionIndexTitlesForTableView: (UITableView *) tableView;
- - (NSInteger) tableView: (UITableView *) tableView sectionForSectionIndexTitle: (NSString *) title atIndex: (NSInteger) index;
- - (void) tableView: (UITableView *) tableView commitEditingStyle: (UITableViewCellEditingStyle) ÉditionStyle forRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView moveRowAtIndexPath: (NSIndexPath *) sourceIndexPath toIndexPath: (NSIndexPath *) destinationIndexPath;

Remarques

`UITableView` est une sous-classe de `UIScrollView` . Les classes qui suivent le protocole `UITableViewDelegate` suivent également le protocole `UIScrollViewDelegate` . `UITableView` peut être utile pour afficher des listes longues ou indéterminées dans ses cellules, alors `UIScrollView` convient mieux lorsque la taille des vues à afficher est connue à l'avance.

Examples

Cellules auto-calibrantes

Dans iOS 8, Apple a introduit la cellule auto-dimensionnante. Disposez explicitement vos UITableViewCells avec Autolayout et UITableView s'occupe du reste pour vous. Hauteur de la ligne est calculée automatiquement, par défaut `rowHeight` valeur est UITableViewAutomaticDimension.

UITableView propriété `estimatedRowHeight` est utilisé lorsque la cellule d' auto-calibrage calcule.

> Lorsque vous créez une cellule de vue de table à redimensionnement automatique, vous devez définir cette propriété et utiliser des contraintes pour définir la taille de la cellule.
>
> *- Apple, Documentation UITableView*

```
self.tableView.estimatedRowHeight = 44.0
```

Notez que la propriété heightForRowAtIndexPath du délégué `heightForRowAtIndexPath` est *inutile* si vous souhaitez avoir une hauteur dynamique pour toutes les cellules. Définissez simplement la propriété ci-dessus si nécessaire et avant de recharger ou de charger la vue de table. Cependant, vous pouvez définir la hauteur de cellules spécifiques tout en ayant d'autres dynamiques via la fonction suivante:

Rapide

```
override func tableView(tableView: UITableView, heightForRowAtIndexPath indexPath:
NSIndexPath) -> CGFloat {
    switch indexPath.section {
    case 1:
        return 60
    default:
        return UITableViewAutomaticDimension
    }
}
```

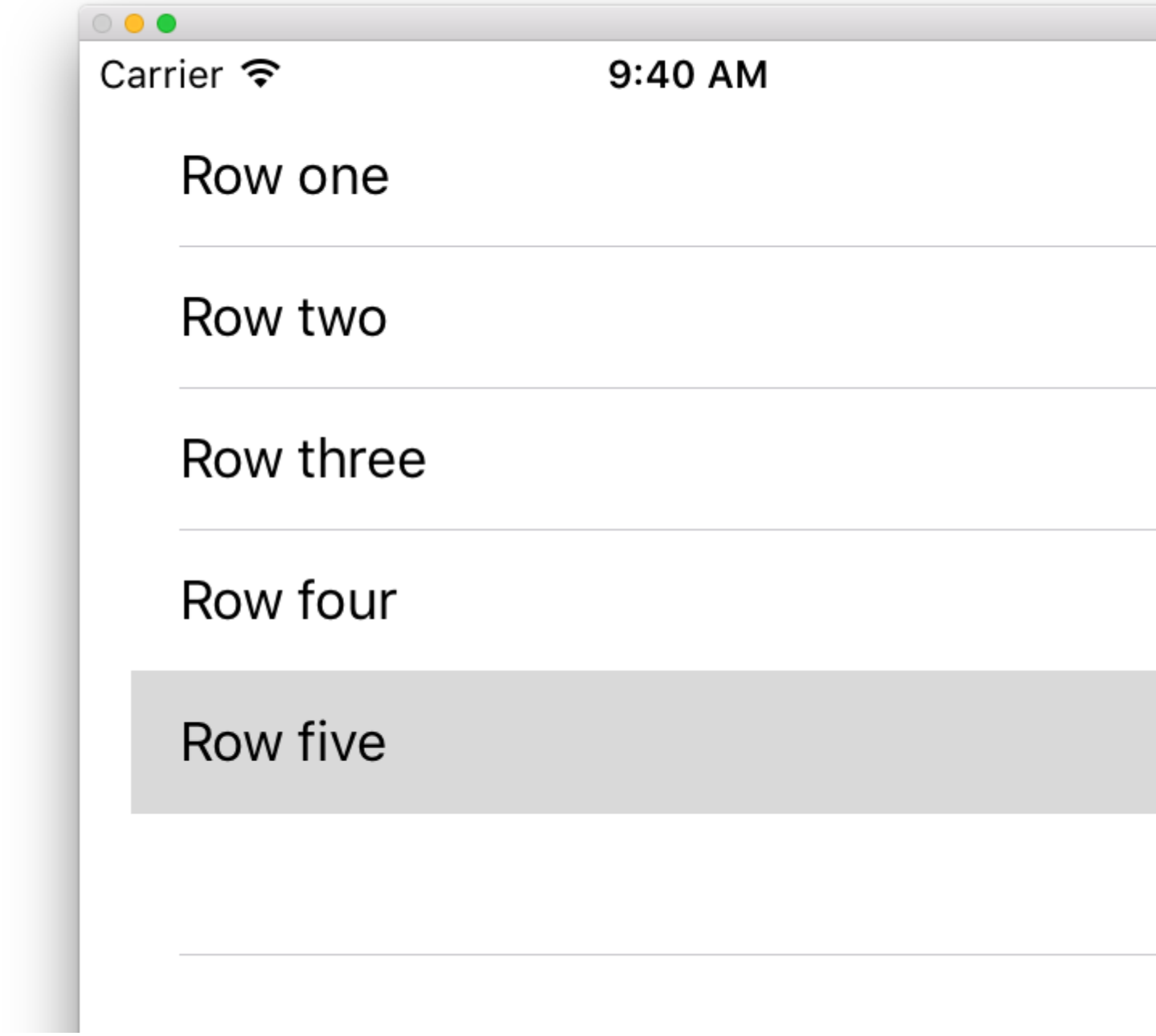
Objectif c

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
  switch (indexPath.section) {
```

```
        case 1:
          return 60;
        default:
          return UITableViewAutomaticDimension;
    }
}
```

Créer un UITableView

Une vue de tableau est une liste de lignes pouvant être sélectionnées. Chaque ligne est alimentée par une source de données. Cet exemple crée une vue de table simple dans laquelle chaque ligne est une seule ligne de texte.



Ajouter un UITableView à votre storyboard

Bien qu'il existe plusieurs manières de créer une `UITableView` , l'une des plus faciles consiste à en ajouter une à un storyboard. Ouvrez votre storyboard et faites glisser un `UITableView` sur votre `UIViewController` . Veillez à utiliser la mise en page automatique pour aligner correctement la table (broche sur les quatre côtés).

Remplir votre table avec des données

Afin d'afficher le contenu de manière dynamique (c.-à-d. Le charger à partir d'une source de données comme un tableau, un modèle Core Data, un serveur en réseau, etc.) dans votre tableau, vous devez configurer la source de données.

Créer une source de données simple

Une source de données pourrait, comme indiqué ci-dessus, être n'importe quoi avec des données. C'est entièrement à vous de savoir comment le formater et ce qu'il contient. La seule exigence est que vous devez être en mesure de le lire plus tard afin de pouvoir remplir chaque ligne de votre table en cas de besoin.

Dans cet exemple, nous allons simplement définir un tableau avec des chaînes (texte) comme source de données:

Rapide

```
let myDataArray: [String] = ["Row one", "Row two", "Row three", "Row four", "Row five"]
```

Objectif 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"];
```

Configuration de votre source de données dans votre View Controller

Assurez-vous que votre contrôleur de vue est conforme au protocole `UITableViewDataSource` .

Rapide

```
class ViewController: UIViewController, UITableViewDataSource {
```

Objectif c

```
@interface ViewController : UIViewController <UITableViewDataSource>
```

Dès que votre contrôleur de vue a déclaré qu'il sera **conforme** à la `UITableViewDataSource` (c'est ce

que nous venons de faire ci - dessus), vous *devez* mettre en œuvre au moins les méthodes suivantes dans votre vue classe contrôleur:

- `tableView:numberOfRowsInSection` , cela vous demande combien de lignes votre vue doit avoir.

```
// Swift

func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
     return self.myDataArray.count
}
```

- `tableView:cellForRowAtIndexPath` , vous demande de créer et de renvoyer une cellule pour chaque ligne que vous avez spécifiée dans `tableView:numberOfRowsInSection` . Donc, si vous avez dit avoir besoin de 10 lignes, cette méthode sera appelée dix fois pour chaque ligne et vous devrez créer une cellule pour chacune de ces lignes.

```
// 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
 }
```

AVERTISSEMENT: Vous **ne** pouvez **PAS** retourner zéro pour toutes les cellules dans `cellForRowAtIndexPath:` . Cela provoquera le crash de votre application et vous verrez l'erreur suivante dans la console:

```
Uncaught exception 'NSInternalInconsistencyException', reason: 'UITableView
dataSource must return a cell from tableView:cellForRowAtIndexPath:'
```

Connexion de la source de données de la vue de table à votre contrôleur de vue

Vous pouvez soit le faire via le code en définissant la propriété `dataSource` votre table sur `self` sur votre contrôleur de vue. Ou vous pouvez sélectionner votre vue de table dans votre storyboard, ouvrir l'inspecteur d'attributs, sélectionner le panneau "Outlets" et faire glisser de `dataSource` vers votre contrôleur de vue (**REMARQUE** : assurez-vous de vous connecter à UIViewCONTROLLER, **pas** à UIView ou à un autre objet *de* votre UIViewController).

Gestion des sélections de lignes

Lorsqu'un utilisateur appuie sur une ligne dans la vue de la table, vous devez généralement faire quelque chose pour y répondre. Dans de nombreuses applications, lorsque vous appuyez sur une ligne, plus d'informations sur cet élément sont affichées. Pensez à l'application Messages: lorsque vous appuyez sur la ligne affichant l'un de vos contacts, la conversation avec cette personne est alors affichée à l'écran.

Dans orer pour ce faire, vous devez vous conformer au protocole `UITableViewDelegate` . Cela est similaire à la conformité au protocole de source de données. Cette fois, cependant, vous l'ajouterez juste à côté de `UITableViewDataSource` et séparez-le par une virgule. Donc, ça devrait ressembler à ceci:

Rapide

```
class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {
```

Objectif c

```
@interface ViewController : UIViewController <UITableViewDataSource, UITableViewDelegate>
```

Il n'y a aucune méthode requise pour implémenter le délégué de la vue de table. Cependant, pour gérer les sélections de lignes, vous devez utiliser la méthode suivante:

- `tableView:didSelectRowAtIndexPath` , ceci est appelé chaque fois qu'une ligne est tapée, ce qui vous permet de faire quelque chose en réponse. Pour notre exemple, nous allons simplement imprimer une déclaration de confirmation dans le journal 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 solution finale

Voir ci-dessous pour la configuration complète avec juste du code, aucune explication.

Rapide

```
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).")
    }
}
```

Objectif 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
```

Délégué et source de données

`UITableViewDelegate` est utilisé pour contrôler l'affichage de la table et `UITableViewDataSource` pour définir les `UITableView` de `UITableView` . Il existe deux méthodes obligatoires et de nombreuses méthodes facultatives pouvant être utilisées pour personnaliser la taille, les sections, les en-têtes et les cellules dans `UITableView` .

UITableViewDataSource

Méthodes requises

`numberOfRowsInSection:` cette méthode définit le nombre de cellules à afficher dans chaque section de la tableview.

Objectif 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:` Cette méthode est l'endroit où les `UITableView` de `UITableView` sont créées et configurées. Devrait retourner soit une `UITableViewCell` ou une sous-classe personnalisée.

Remarque: À l'aide de `dequeueReusableCellWithIdentifier:forIndexPath:` requiert que la classe ou le nib ait été enregistré pour cet identifiant à l'aide des `UITableView registerClass:forCellReuseIdentifier:` ou `registerNib:forCellReuseIdentifier:` Habituellement, cela se fera dans le `UIViewController` de `viewDidLoad` méthode.

Objectif 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
}
```

Méthodes optionnelles

`titleForHeaderInSection:` Définit une chaîne comme titre pour chaque en-tête de section dans la vue de table. Cette méthode permet uniquement de modifier le titre, une personnalisation supplémentaire peut être effectuée en définissant la vue de l'en-tête.

Objectif 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:` définit une chaîne comme titre pour chaque en-tête de section dans la vue de table.

Objectif 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:` utilisé pour déterminer si l'interface utilisateur de modification doit être affichée pour la ligne spécifiée. Devrait retourner `YES` si la ligne spécifiée peut être supprimée ou ajoutée.

Objectif c

```
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath {
    return YES;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, canEditRowAtIndexPath indexPath: NSIndexPath) -> Bool
{
    return true
}
```

`commitEditingStyle:forRowAtIndexPath` Effectuer le travail requis pour gérer l'ajout ou la suppression de la ligne spécifiée. Par exemple, supprimez la cellule de `UITableView` avec animation et supprimez l'objet associé du modèle de données de la table.

Objectif 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` Permet d'ajouter des actions ou des boutons supplémentaires au mode d'édition d'une ligne dans une vue `UITableview` . Par exemple, si vous vouliez deux boutons, un bouton d'édition et de suppression lorsque l'utilisateur essaie de modifier la ligne, vous utiliseriez cette méthode.

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

Toutes les méthodes de `UITableViewDelegate` sont facultatives, mais un délégué les implémentant `UITableView` des fonctionnalités supplémentaires pour `UITableView` .

`numberOfSectionsInTableView:` Par défaut, cette option renvoie 1, mais la prise en charge de plusieurs sections est activée en renvoyant un nombre différent de sections.

Objectif c

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return self.numSections;
}
```

Swift 3

```
func numberOfSectionsInTableView(_ tableView: UITableView) -> Int {
    return self.numSections
}
```

`viewForHeaderInSection` Permet la configuration d'une vue personnalisée en tant qu'en-tête pour la section.

Objectif 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:` Définit la hauteur de chaque cellule dans la vue de tableau.

Objectif 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` Définit la hauteur de l'en-tête et du pied de page de chaque section dans la vue de table

Objectif c

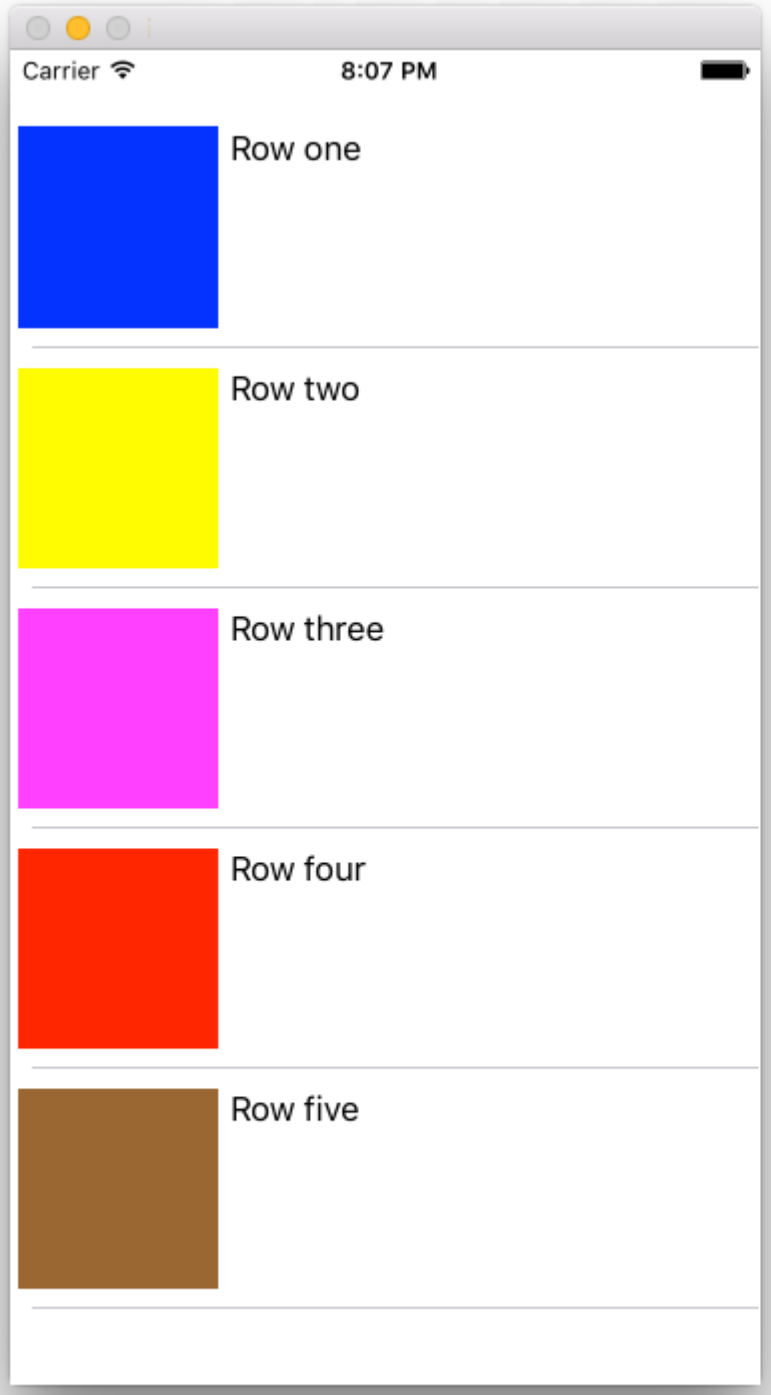
```
- (CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section {
    return 33;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
    return 33
}
```

Cellules personnalisées

Personnaliser un `UITableViewCell` peut permettre des interfaces très puissantes, dynamiques et réactives. Grâce à une personnalisation étendue et à d'autres techniques, vous pouvez: mettre à jour des propriétés spécifiques ou des éléments d'interface pour modifier, animer ou dessiner des éléments, charger des vidéos de manière efficace ou même télécharger des images depuis un ordinateur. réseau. Les possibilités ici sont presque infinies. Vous trouverez ci-dessous un exemple simple de ce à quoi une cellule personnalisée peut ressembler.



Cette section couvre les bases, et nous espérons qu'elle sera élargie pour détailler des processus plus complexes tels que ceux décrits ci-dessus.

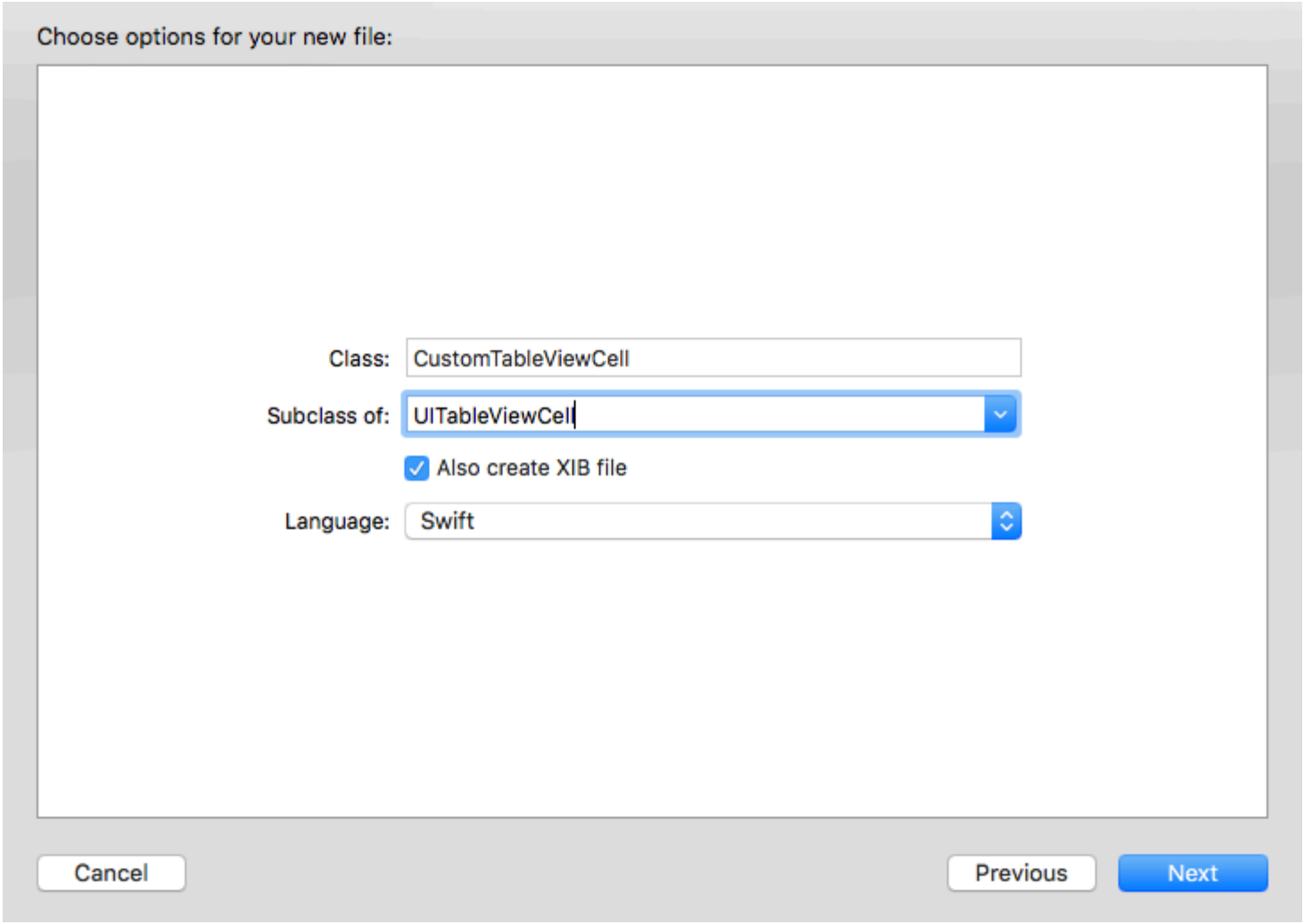
Créer votre cellule personnalisée

Tout d'abord, créez une nouvelle sous-classe de `UITableViewCell` (créez une nouvelle classe Cocoa Touch dans Xcode et définissez `UITableViewCell` comme super-classe). Voici ce que votre code peut ressembler après le sous-classement.

Rapide

```
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)
    }
}
```

Si vous le souhaitez, cochez la case "Créer également un fichier XIB" lors de la création de votre nouveau fichier à personnaliser à l'aide d'Interface Builder. Si vous le faites, connectez `customLabel` tant que `@IBOutlet`



Dans un `UIViewController` contenant la `tableView` , enregistrez la classe de la nouvelle cellule personnalisée (voir ci-dessous). *Notez que cela n'est nécessaire que si vous ne concevez pas la*

cellule avec un Storyboard dans l'interface de votre vue de table.

Rapide

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

    // Register Cell Class
    tableView.register(CustomTableViewCell.self, forCellReuseIdentifier:
CustomTableViewCell.identifier)
}
```

Si vous avez choisi d'utiliser un fichier XIB, `registerNib` plutôt:

Rapide

```
// Register Nib
tableView.register(UINib(nibName: CustomTableViewCell.identifier, bundle: nil),
forCellReuseIdentifier: CustomTableViewCell.identifier)
```

Maintenant que votre `tableView` connaît votre cellule personnalisée, vous pouvez la `cellForRowAtIndexPath` dans `cellForRowAtIndexPath` :

Rapide

```
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
}
```

Développer et réduire UITableViewCells

Dans votre Storyboard, ajoutez un objet UITableView sur votre UIViewController et laissez-le couvrir l'intégralité de la vue. Configurez les `UITableviewDataSource` et `UITableviewDelegate` .

Objectif c

Dans votre fichier `.h`

```
NSMutableArray *arrayForBool;
NSMutableArray *sectionTitleArray;
```

Dans votre fichier `.m`

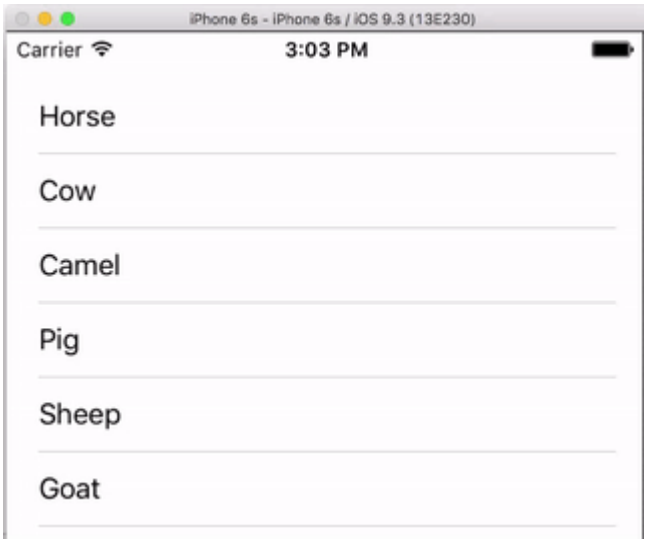
```
- (void)viewDidLoad  {
    [super viewDidLoad];

    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];

   }
 }
```

Glisser pour supprimer les lignes

Je pense toujours qu'il est bon d'avoir un exemple très simple et autonome pour que rien ne soit supposé lorsque j'apprends une nouvelle tâche. Cette réponse est celle de la suppression des lignes `UITableView` . Le projet se comporte comme ceci:



Ce projet est basé sur l' exemple UITableView pour Swift .

Ajouter le code

Créez un nouveau projet et remplacez le code ViewController.swift par le suivant.

```
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.
        }
    }

}
```

La méthode de clé unique dans le code ci-dessus qui permet la suppression de lignes est la dernière. Ici c'est encore pour l'emphase:

```
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

Ajoutez un `UITableView` au View Controller dans le storyboard. Utilisez la mise en page automatique pour épingler les quatre côtés de la vue de table aux bords du View Controller. Contrôlez le glissement de la vue de table dans le storyboard vers `@IBOutlet var tableView: UITableView!` ligne dans le code.

Fini

C'est tout. Vous devriez pouvoir exécuter votre application maintenant et supprimer des lignes en glissant vers la gauche et en appuyant sur "Supprimer".

Remarques

- Ceci est uniquement disponible sur iOS 8. Voir cette réponse pour plus de détails.
- Si vous devez modifier le nombre de boutons affichés ou le texte du bouton, consultez cette réponse pour plus de détails.

Lectures complémentaires

- Comment faire pour créer une cellule de vue de tableau pouvant être déplacée avec des actions - sans écraser les vues de défilement
- Documentation Apple

Lignes de séparation

Modification de la largeur des lignes de séparation

Vous pouvez définir que les lignes de séparation de votre vue de tableau étendent les différentes largeurs sur la table en modifiant la propriété `layoutMargins:` de vos cellules. Cela peut être réalisé de plusieurs manières.

Modification des lignes de séparation pour des cellules spécifiques

Dans la méthode `willDisplayCell:` votre source de données de `cellForRowAtIndexPath:` , *ou dans* la méthode `willDisplayCell:` définissez la propriété `layoutMargins:` la cellule sur `UIEdgeInsetsZero` (s'étend sur toute la largeur de la table) ou sur tout ce que vous pouvez souhaiter ici.

Objectif c

```
[cell setLayoutMargins:UIEdgeInsetsZero];

// May also use separatorInset
[cell setSeparatorInset:UIEdgeInsetsZero];
```

Rapide

```
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
}
```

Supprimer toutes les lignes de séparation

Les fines lignes grises entre chaque cellule peuvent ne pas correspondre exactement à ce que vous recherchez. Il est assez simple de les cacher de la vue.

Dans votre englobant `UIViewController` de `viewDidLoad:` méthode ajoutez le code suivant. Vous pouvez également définir cette propriété à tout moment avant de charger ou de recharger la vue de table (il n'est pas nécessaire que la méthode `viewDidLoad:` utilisée).

Rapide:

```
 tableView.separatorStyle = .None
```

Objectif c:

```
tableView.separatorStyle = UITableViewCellSeparatorStyleNone;
```

Vous pouvez également modifier la propriété dans votre Storyboard ou XIB en sélectionnant votre tableView et en définissant le `separator` (sous l'inspecteur d'attributs) sur `None` .

Masquer les lignes de séparation en excès

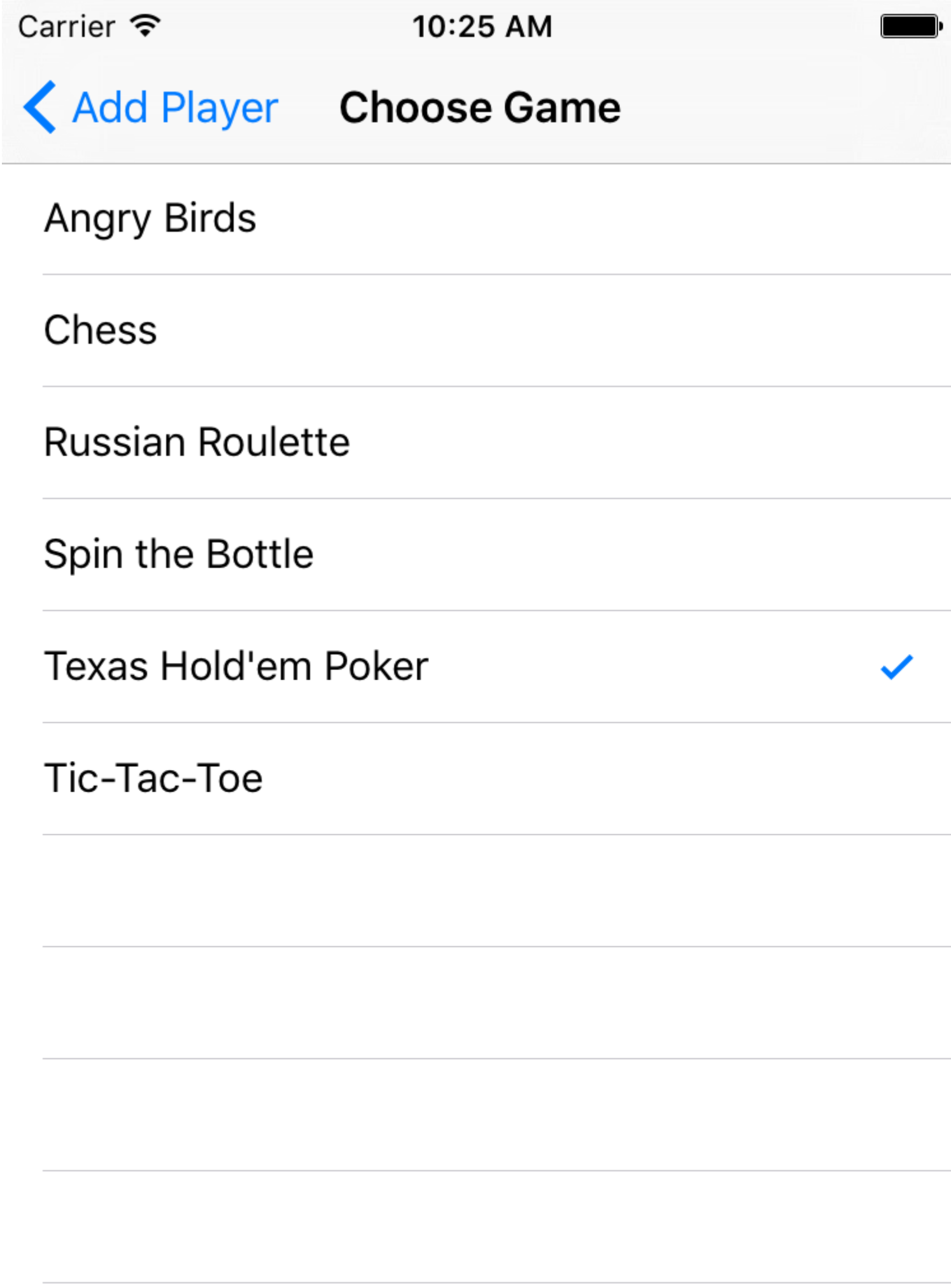
Vous pouvez masquer les lignes de séparation `UITableViewCell` pour les cellules vides en définissant une vue de pied de page vide au bas d'un UITableView:

Rapide

```
tableView.tableFooterView = UIView()
```

Objectif 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

Chapitre 192: UITableViewCell

Introduction

Charger le fichier xib de cellule personnalisée utilise la classe de catégorie de cellule, pas besoin d'enregistrer le fichier nib

Examples

Fichier Xib de UITableViewCell

Créez une classe de catégorie de cellule `UITableView` .

Fichier UITableViewCell + RRCell.h

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

@interface UITableViewCell (RRCell)

-(id)initWithOwner:(id)owner;

@end
```

Fichier UITableViewCell + RRCell.m

```
#import "UITableViewCell+RRCell.h"

@implementation UITableViewCell (RRCell)

#pragma clang diagnostic push
#pragma clang diagnostic ignored "-Wobjc-designated-initializers"

-(id)initWithOwner:(id)owner {

    if (self = [super init]) {

        NSArray *nib = [[NSBundle mainBundle]loadNibNamed:NSStringFromClass([self class])
owner:self options:nil];
        self = [nib objectAtIndex:0];
    }
    return self;
}

#pragma clang diagnostic pop



@end
```

Importer une classe de catégorie de cellule pour utiliser cette méthode dans la méthode `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;
}
```

Lire UITableViewCell en ligne: https://riptutorial.com/fr/ios/topic/10101/uitableviewcell

Chapitre 193: UITableViewController

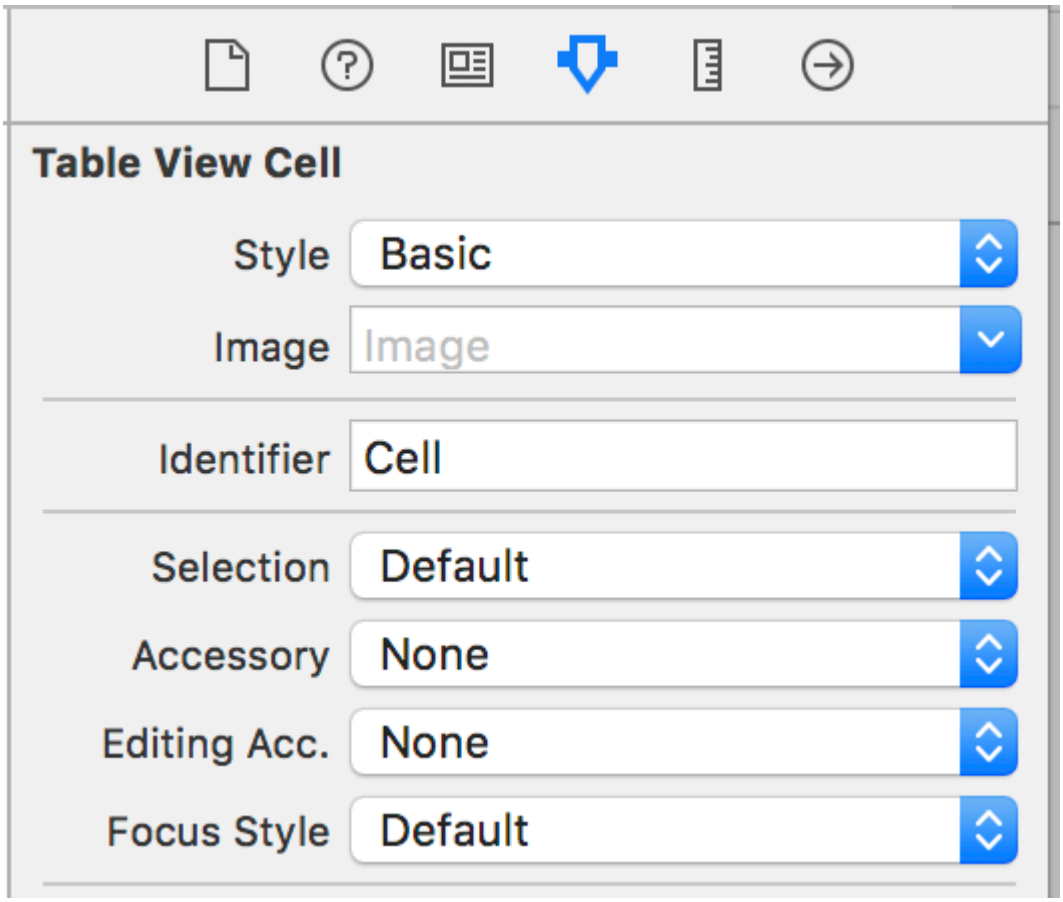
Introduction

Objet de contrôleur UITableViewController qui gère une vue de table. Pour certains scénarios, il sera recommandé d'utiliser UITableViewController, par exemple si vous avez beaucoup de cellules et que certaines ont UITextfield.

Examples

TableView avec propriétés dynamiques avec 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 avec cellule personnalisée

Pour une cellule de table personnalisée, vous avez besoin d'une classe qui est une sous-classe de `UITableViewCell` , un exemple de classe que vous pouvez voir ci-dessous.

```
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
}

}
```

Vos délégués de table

```
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
}
```

Lire UITableViewController en ligne: https://riptutorial.com/fr/ios/topic/10953/uitableviewcontroller

Chapitre 194: UITextField

Introduction

UITextField fait partie du framework UIKit et permet d'afficher une zone pour collecter les entrées de texte de l'utilisateur à l'aide du clavier à l'écran

Syntaxe

- UITextField.text: String // obtient ou définit le texte affiché par le champ.
- UITextField.attributedText: NSAttributedString // récupère ou définit le texte attribué affiché par le champ.
- UITextField.textColor: UIColor // récupère ou définit la couleur du texte sur le champ
- UITextField.font: UIFont // récupère ou définit la police du texte sur le champ
- UITextField.textAlignment: NSTextAlignment // la valeur par défaut est NSLeftTextAlignment
- UITextField.borderStyle: UITextBorderStyle // La valeur par défaut est UITextBorderStyleNone. Si défini sur UITextBorderStyleRoundedRect, les images d'arrière-plan personnalisées sont ignorées.
- UITextField.placeholder: String // la valeur par défaut est nil. la ficelle est dessinée à 70% de gris
- UITextField.attributedPlaceholder: NSAttributedString // récupère ou définit l'espace réservé attribué au champ
- UITextField.clearsOnBeginEditing: Bool // default est NO qui déplace le curseur à l'emplacement cliqué. si OUI, tout le texte est effacé
- UITextField.adjustsFontSizeToFitWidth: Bool // default est NO. si OUI, le texte sera réduit à minFontSize le long de la ligne de base
- UITextField.minimumFontSize: CGFloat // la valeur par défaut est 0.0. réel min peut être épinglé à quelque chose de lisible. utilisé si adjustsFontSizeToFitWidth est OUI
- UITextField.delegate: UITextFieldDelegate? // la valeur par défaut est nil. référence faible
- UITextField.clearButtonMode: UITextFieldViewMode // Définit lorsque le bouton Effacer apparaît. la valeur par défaut est UITextFieldViewModeNever
- UITextField.leftView: UIView? // p.ex. loupe
- UITextField.leftViewMode: UITextFieldViewMode // se définit lorsque la vue de gauche apparaît. la valeur par défaut est UITextFieldViewModeNever
- UITextField.rightView: UIView? // par exemple le bouton de favoris
- UITextField.rightViewMode: UITextFieldViewMode // se définit lorsque la vue de droite apparaît. la valeur par défaut est UITextFieldViewModeNever
- UITextField.inputView: UIView? // Présenté lorsque l'objet devient le premier répondant. Si défini sur nil, revient à la chaîne de répondeur suivante. S'il est défini lors du premier répondeur, il ne prendra effet qu'après l'appel de reloadInputViews.
- UITextField.inputAccessoryView: UIView?
- UITextField.isSecureTextEntry: Bool // Par exemple, si le champ contient une entrée confidentielle telle qu'un mot de passe ou un numéro de carte

Examples

Initialiser le champ de texte

Rapide

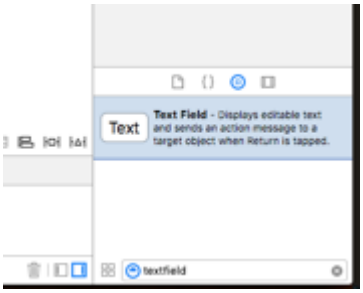
```
let frame = CGRect(x: 0, y: 0, width: 100, height: 100)
let textField = UITextField(frame: frame)
```

Objectif c

```
CGRect *frame = CGRectMake(0, 0, 100, 100);
UITextField *textField = [[UITextField alloc] initWithFrame:frame];
```

Interface Builder

Vous pouvez également ajouter un `UITextField` à un storyboard en le faisant glisser depuis la bibliothèque d'objets.



Vue accessoire d'entrée (barre d'outils)

Ajoutez une vue accessoire au-dessus du clavier. Ceci est couramment utilisé pour ajouter des boutons suivant / précédent, ou des boutons supplémentaires comme Terminé / Soumettre (en particulier pour les types de clavier nombre / téléphone / pad décimal qui n'ont pas de touche de retour intégrée).

Rapide

```
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
```

Objectif 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;
```

Auto-capitalisation

Rapide

```
textField.autocapitalizationType = .None
```

Objectif c

```
textField.autocapitalizationType = UITextAutocapitalizationTypeNone;
```

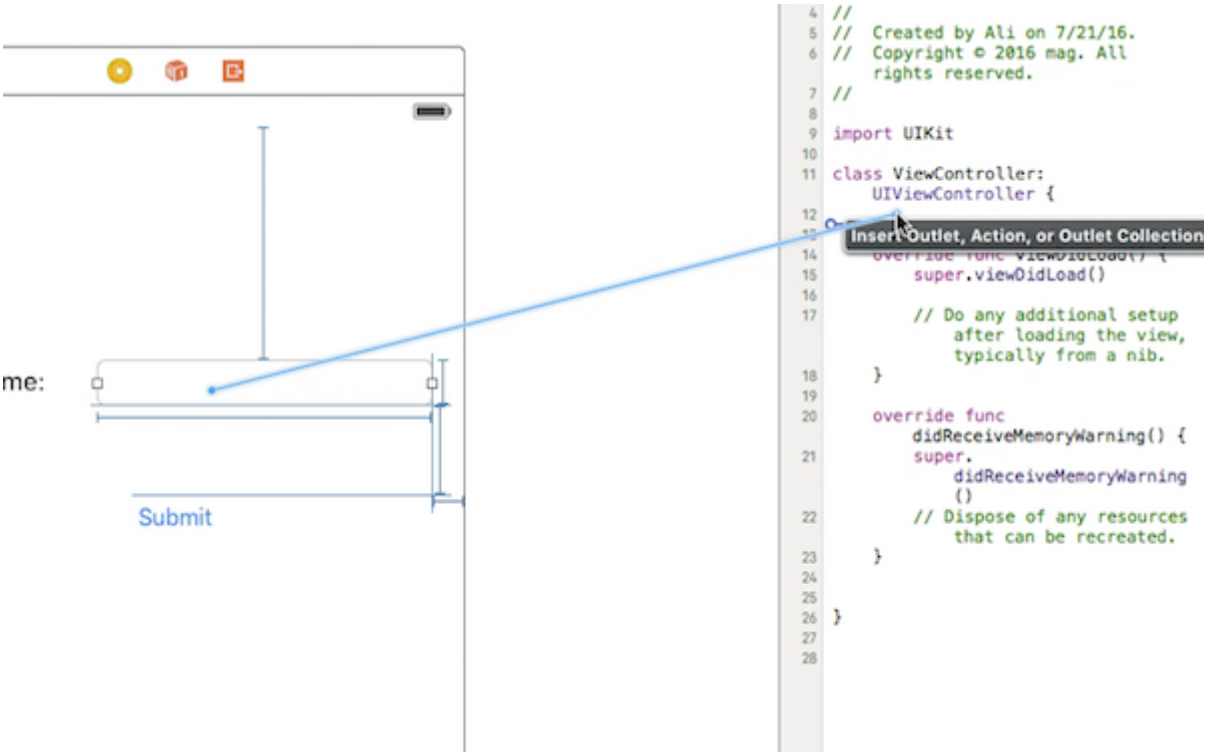
Toutes les options:

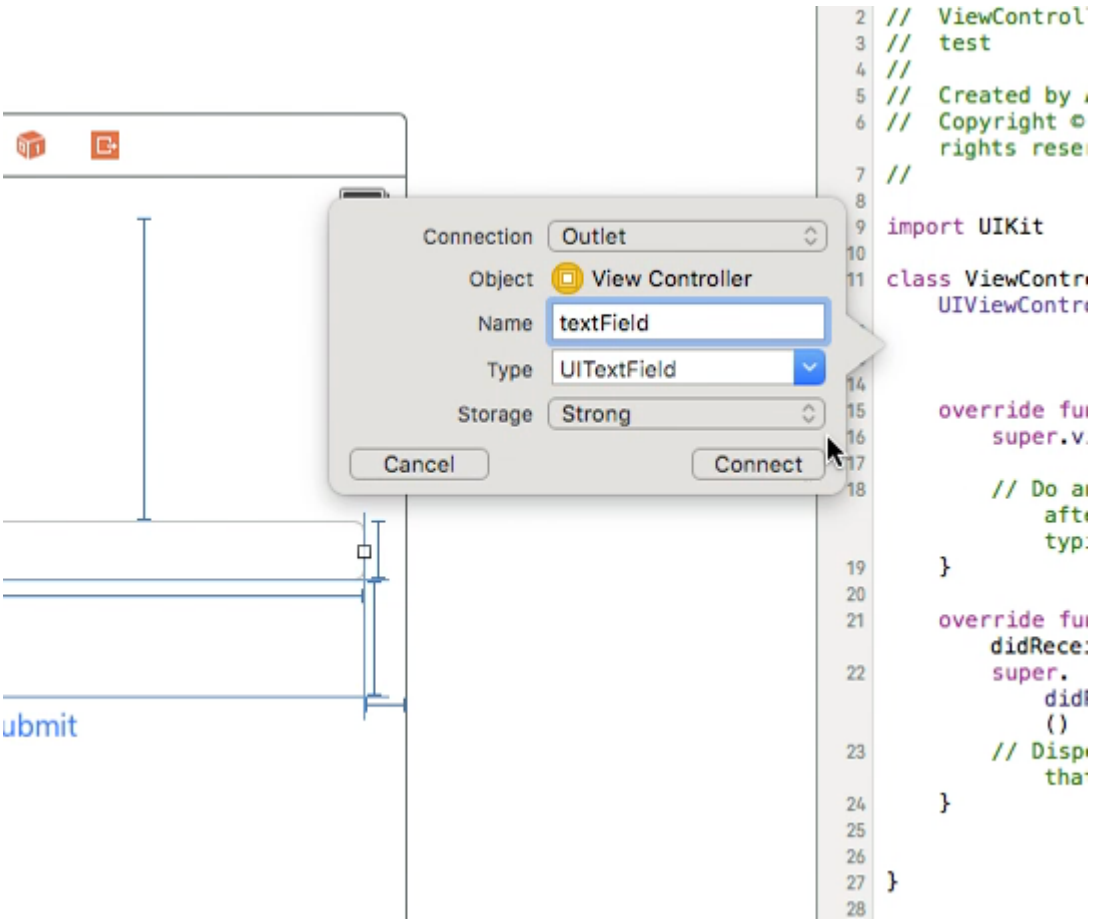
- `.None` \ `UITextAutocapitalizationTypeNone` : Ne rien autocapitaliser
- `.Words` \ `UITextAutocapitalizationTypeWords` : Autocapitalise chaque mot
- `.Sentences` \ `UITextAutocapitalizationTypeSentences` : Autocapitalise le premier mot d'une phrase
- `.AllCharacters` \ `UITextAutocapitalizationTypeAllCharacters` : Autocapitalise chaque lettre (c.-à- `UITextAutocapitalizationTypeAllCharacters` . Majuscule)

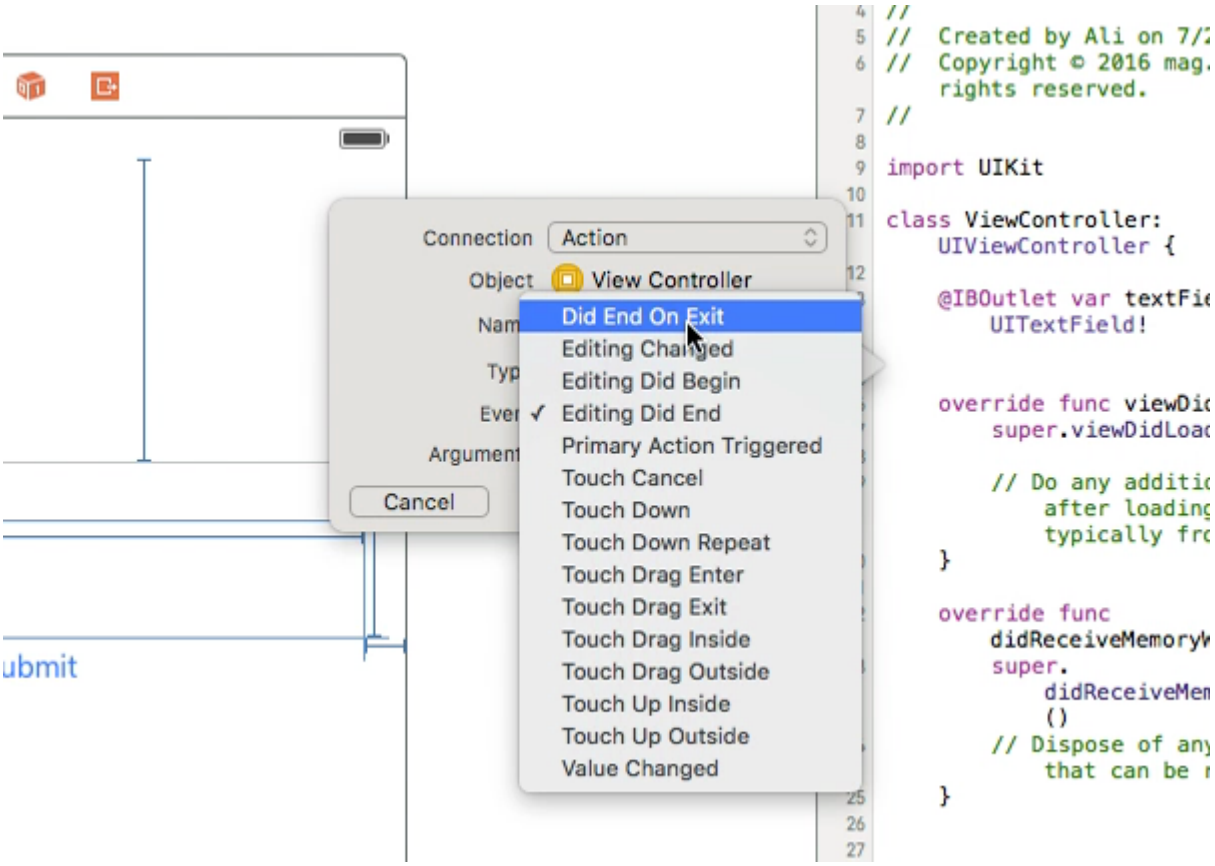
Rejeter le clavier

Rapide

Ctrl + Faites glisser depuis le champ UItext de MainStoryboard vers la classe ViewController et créez une sortie UITextField







Après cela, sélectionnez à nouveau UItextField et Ctrl + faites glisser dans la classe ViewController, mais cette fois-ci, sélectionnez **Action** connection et, au stockage, sélectionnez **Did End On Exit,** puis cliquez sur Connect.

dans l'action que vous venez de créer, tapez le nom de votre UItextField `.resignFirstResponder()`

```
   @IBAction func textFieldResign(sender: AnyObject) {
        yourTextFieldName.resignFirstResponder()
    }
```

Cela permettra de cacher le clavier en appuyant sur la touche retour du clavier.

Un autre exemple de masquage du clavier lorsque la touche retour est pressée:

nous ajoutons le protocole `UITextFieldDelegate` côté de `UIViewController`

dans la fonction vieDidLoad, nous ajoutons `self.yourTextFieldName.delegate = self`

Et enfin nous ajoutons ceci

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
                yourTextFieldName.resignFirstResponder()
                return true
            }
```

Le code final est le suivant:

```
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
        }
```

}

Objectif c

```
[textField resignFirstResponder];
```

Définir l'alignement

Rapide

```
textField.textAlignment = .Center
```

Objectif c

```
[textField setTextAlignment: NSTextAlignmentCenter];
```

Dans l'exemple, nous avons défini le `NSTextAlignment` sur center. Vous pouvez également définir sur `.Left` , `.Right` , `.Justified` et `.Natural` .

`.Natural` est l'alignement par défaut pour la localisation en cours. Cela signifie que pour les langues de gauche à droite (par exemple, l'anglais), l'alignement est `.Left` ; pour les langues de droite à gauche, c'est: `.Right` .

Type de clavier

Pour modifier l'apparence du clavier, les types suivants peuvent être définis individuellement sur chaque propriété `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
};
```

Déplacement du défilement lorsque UITextView devient le premier répondant

Observez les notifications `UIKeyboardWillShowNotification` et `UIKeyboardWillHideNotification`, mettez à jour les `scrollView` contenu scrollView en fonction de la hauteur du clavier, puis accédez au contrôle ciblé.

```
- (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];
}
```

Obtenir le clavier et masquer le clavier

Se concentrer

Rapide

```
textField.becomeFirstResponder()
```

Objectif c

```
[textField becomeFirstResponder];
```

Démissionner

Rapide

```
textField.resignFirstResponder()
```

Objectif c

```
[textField resignFirstResponder];
```

Remplacer le clavier par UIPickerView

Dans certains cas, vous voulez montrer à vos utilisateurs un `UIPickerView` avec un contenu prédéfini pour un `UITextField` au lieu d'un clavier.

Créer un UIPickerView personnalisé

Au début, vous avez besoin d'une classe d'encapsulation personnalisée pour `UIPickerView` conforme aux protocoles `UIPickerViewDataSource` et `UIPickerViewDelegate` .

```
class MyPickerView: UIPickerView, UIPickerViewDataSource, UIPickerViewDelegate
```

Vous devez implémenter les méthodes suivantes pour le DataSource et le délégué:

```
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 ""
    }
}
```

Pour gérer les données, `MyPickerView` besoin des propriétés `data` , `selectedValue` et `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 ""
        }
    }
}
```

Préparez votre ViewController

Le `ViewController` qui contient votre textField doit avoir une propriété pour votre `UIPickerView` personnalisé. (En supposant que vous ayez déjà une autre propriété ou `@IBOutlet` contenant votre textField)

```
/**
 The picker view to present as keyboard
 */
var picker: MyPickerView?
```

Dans votre `viewDidLoad()` , vous devez initialiser le `picker` et le configurer un peu:

```
picker = MyPickerView()
picker?.autoresizingMask = [.flexibleHeight, .flexibleWidth]
picker?.backgroundColor = UIColor.white()

picker?.data = ["One", "Two", "Three", "Four", "Five"] //The data shown in the picker
```

Maintenant, vous pouvez ajouter le `MyPicker` comme `inputView` de votre `UITextField` :

```
textField.inputView = picker
```

Rejeter le clavier du sélecteur

Maintenant, vous avez remplacé le clavier par un `UIPickerView` , mais il est impossible de le `UIPickerView` . Cela peut être fait avec un `.inputAccessoryView` personnalisé:

Ajoutez la propriété `pickerAccessory` à votre `ViewController` .

```
/**
 A toolbar to add to the keyboard when the `picker` is presented.
 */
```

```
var pickerAccessory: UIToolbar?
```

Dans `viewDidLoad()` , vous devez créer une `UIToolbar` pour le `inputAccessoryView` :

```
pickerAccessory = UIToolbar()
pickerAccessory?.autoresizingMask = .flexibleHeight

//this customization is optional
pickerAccessory?.barStyle = .default
pickerAccessory?.barTintColor = UIColor.red()
pickerAccessory?.backgroundColor = UIColor.red()
pickerAccessory?.isTranslucent = false
```

Vous devez définir le cadre de votre barre d'outils. Pour s'adapter à la conception d'iOS, il est recommandé d'utiliser une hauteur de `44.0` :

```
var frame = pickerAccessory?.frame
frame?.size.height = 44.0
pickerAccessory?.frame = frame!
```

Pour une bonne expérience utilisateur, vous devriez ajouter deux boutons ("Terminé" et "Annuler"), mais cela fonctionnerait aussi avec un seul qui renvoie le clavier.

```
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]
```

Maintenant, vous pouvez ajouter la barre d'outils en tant que `inputAccessoryView`

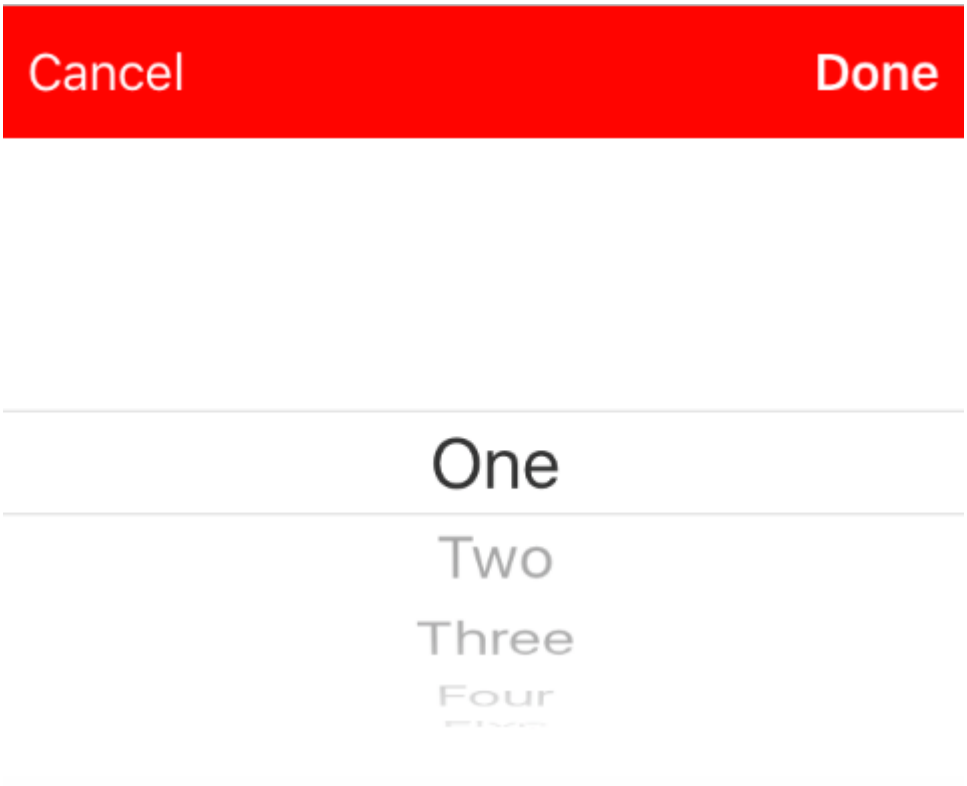
```
textField.inputAccessoryView = pickerAccessory
```

Avant de pouvoir construire votre projet, vous devez implémenter les méthodes, les boutons appellent:

```
/**
 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
}
```

Exécutez votre projet, appuyez sur le `textField` et vous devriez voir un sélecteur comme celui-ci au lieu du clavier:



Sélectionnez une valeur par programmation (facultatif)

Si vous ne souhaitez pas que la première ligne soit automatiquement sélectionnée, vous pouvez définir la ligne sélectionnée comme dans `UIPickerView` :

```
picker?.selectRow(3, inComponent: 0, animated: false) //Will select the row at index 3
```

Rejeter le clavier lorsque l'utilisateur appuie sur le bouton de retour

Configurez votre contrôleur de vue pour gérer la modification du texte du champ de texte.

```
class MyViewController: UITextFieldDelegate {

    override viewDidLoad() {
        super.viewDidLoad()

        textField.delegate = self
    }

}
```

`textFieldShouldReturn` est appelé chaque fois que le bouton de retour du clavier est pressé.

Rapide:

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
    textField.resignFirstResponder()
    return true;
}
```

Objectif c:

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField {
   [textField resignFirstResponder];
   return true;
}
```

Obtenir et définir la position du curseur

Informations utiles

Le tout début du texte du champ de texte:

```
let startPosition: UITextPosition = textField.beginningOfDocument
```

La fin du texte du champ de texte:

```
let endPosition: UITextPosition = textField.endOfDocument
```

La gamme actuellement sélectionnée:

```
let selectedRange: UITextRange? = textField.selectedTextRange
```

Obtenir la position du curseur

```
if let selectedRange = textField.selectedTextRange {

    let cursorPosition = textField.offsetFromPosition(textField.beginningOfDocument,
toPosition: selectedRange.start)

    print("\(cursorPosition)")
}
```

Définir la position du curseur

Pour définir la position, toutes ces méthodes définissent une plage avec les mêmes valeurs de début et de fin.

Au début

```
let newPosition = textField.beginningOfDocument
textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

Jusqu'à la fin

```
let newPosition = textField.endOfDocument
textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

À une position à gauche de la position actuelle du curseur

```
// 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)
    }
}
```

À une position arbitraire

Commencez par le début et déplacez 5 caractères vers la droite.

```
let arbitraryValue: Int = 5
if let newPosition = textField.positionFromPosition(textField.beginningOfDocument,
inDirection: UITextLayoutDirection.Right, offset: arbitraryValue) {

    textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
}
```

en relation

Sélectionner tout le texte

```
textField.selectedTextRange = textField.textRangeFromPosition(textField.beginningOfDocument,
toPosition: textField.endOfDocument)
```

Sélectionnez une plage de texte

```
// 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!)
}
```

Insérer du texte à la position actuelle du curseur

```
textField.insertText("Hello")
```

Remarques

- Cet exemple provient à l'origine de cette réponse Stack Overflow .
- Cette réponse utilise un champ de texte, mais les mêmes concepts s'appliquent à `UITextView` .
- Utilisez `textField.becomeFirstResponder()` pour mettre en évidence le champ de texte et faire apparaître le clavier.
- Voir cette réponse pour savoir comment obtenir le texte à une certaine distance.

en relation

- Comment créer une plage dans Swift (traite indirectement de la question de savoir pourquoi nous devons utiliser `selectedTextRange` ici plutôt que de simplement `selectedRange`)

Masquer le caret clignotant

Pour masquer le caret clignotant, vous devez remplacer caretRectForPosition d'un objet UITextField et renvoyer 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
}
```

Objectif c

```
- (CGRect) caretRectForPosition:(UITextPosition*) position{
return CGRectZero;
}
```

Modifier la couleur et la police de l'espace réservé

Nous pouvons changer le style de l'espace réservé en définissant `attributedPlaceholder` (une `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
}
```

Dans cet exemple, nous ne modifions que la `color` et la `font` . Vous pouvez modifier d'autres propriétés, telles que le soulignement ou le style barré. Reportez-vous à `NSAttributedString` pour `NSAttributedString` les propriétés pouvant être modifiées.

Créer un champ UITextField

Initialisez `UITextField` avec un CGRect en tant qu'image:

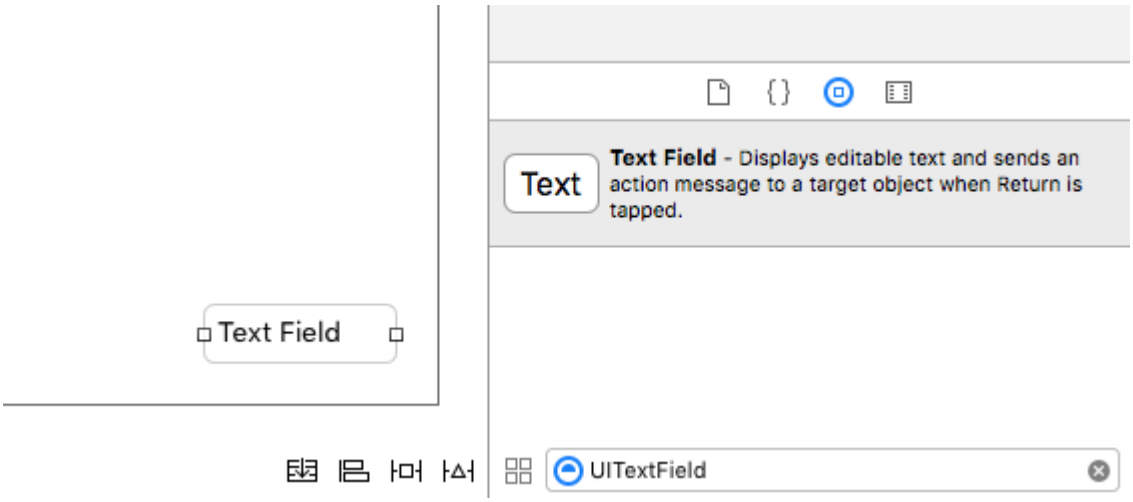
Rapide

```
let textfield = UITextField(frame: CGRect(x: 0, y: 0, width: 200, height: 21))
```

Objectif c

```
UITextField *textField = [[UITextField alloc] initWithFrame:CGRectMake(0, 0, 200, 21)];
```

Vous pouvez également créer un `UITextField` dans Interface Builder:



Lire UITextField en ligne: https://riptutorial.com/fr/ios/topic/1630/uitextfield

Chapitre 195: UITextField Délégué

Examples

UITextField - Limite le champ de texte à certains caractères

Si vous souhaitez effectuer une validation de saisie de votre champ de texte par l'utilisateur, utilisez l'extrait de code suivant:

```
// 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
    }
}
```

Objectif 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];
}
```

De plus, vous pouvez également utiliser les jeux de caractères fournis par Apple pour effectuer la validation:

Jetez un oeil à https://developer.apple.com/reference/foundation/nscharacterset

```
let allowedCharacters = CharacterSet.alphanumerics.inverted
let allowedCharacters = CharacterSet.capitalizedLetters.inverted
```

Trouver le prochain tag et gérer le clavier

Le champ de texte appelle différentes méthodes de délégation (uniquement si des délégués sont définis) Une des méthodes déléguées appelée par textfield est * **- (BOOL) textFieldShouldReturn: (UITextField *) *textField***

Cette méthode est appelée chaque fois que l'utilisateur appuie sur le bouton de retour. En utilisant cette méthode, nous pouvons implémenter tout comportement personnalisé.

Par exemple,

> Dans l'exemple ci-dessous, le prochain répondeur sera trouvé sur la base du tag et gérera le clavier. Ici 20 est la constante, comme tag assigné à textfield sont comme ceci 50,70,90 etc.

Ici, pour trouver un nouvel objet textfield en tant que répondeur, il créera le champ de texte actuel comme nouveau répondeur et ouvrira le clavier en conséquence.

```
- (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;
            }
```

Actions lorsqu'un utilisateur a commencé / a fini d'interagir avec un champ de texte

Pour Swift 3.1:

Dans le premier exemple, on peut voir comment intercepter l'utilisateur interagissant avec un champ de texte lors de l'écriture. De même, certaines méthodes de UITextFieldDelegate sont appelées lorsqu'un utilisateur a démarré et a mis fin à son interaction avec un objet TextField.

Pour pouvoir accéder à ces méthodes, vous devez vous conformer au protocole UITextFieldDelegate et, pour chaque champ de texte à notifier, affectez la classe parente en tant que déléguée:

```
class SomeClass: UITextFieldDelegate {

    @IBOutlet var textField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        textField.delegate = self
    }

}
```

Vous pourrez maintenant implémenter toutes les méthodes UITextFieldDelegate.

Pour être averti lorsqu'un utilisateur a commencé à modifier un champ de texte, vous pouvez implémenter la méthode textFieldDidBeginEditing (_ :) comme suit:

```
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
    }
}
```

De même, être averti si un utilisateur a mis fin à une interaction avec un champ de texte, vous pouvez utiliser la méthode textFieldDidEndEditing (_ :) comme ceci:

```
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
    }
}
```

Si vous souhaitez contrôler si un objet TextField doit commencer / finir l'édition, les méthodes textFieldShouldBeginEditing (_ :) et textFieldShouldEndEditing (_ :) peuvent être utilisées en renvoyant true / false en fonction de la logique requise.

Lire UITextField Délégué en ligne: https://riptutorial.com/fr/ios/topic/7185/uitextfield-delegue

Chapitre 196: UITextField personnalisé

Introduction

En utilisant UITextField personnalisé, nous pouvons manipuler le comportement du champ de texte!

Examples

UITextField personnalisé pour le filtrage du texte d'entrée

Voici un exemple de `UITextField` personnalisé qui ne prend que du texte et rejette tous les autres.

REMARQUE: pour l'iPhone, il est facile de le faire en utilisant le clavier numérique, mais pour l'iPad, il n'existe pas de clavier avec Nombres uniquement.

```
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: "")
}
}
```

Donc, partout où nous voulons un champ de texte qui ne prendrait que des nombres comme texte

d’entrée, alors nous pouvons utiliser cet objet UITextField personnalisé.

UITextField personnalisé pour interdire toutes les actions telles que copier, coller, etc.

Si nous voulons désactiver toutes les actions comme Copier, Coller, Remplacer, Sélectionner, etc. à partir de `UITextField` nous pouvons utiliser les champs de texte personnalisés suivants:

```
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
}
}
```

En `enableLongPressActions` propriété `enableLongPressActions` , nous pouvons activer toutes les actions à tout moment ultérieurement, si nécessaire.

Lire UITextField personnalisé en ligne: https://riptutorial.com/fr/ios/topic/9997/uitextfield-personnalise

Chapitre 197: UITextView

Examples

Changer le texte

Rapide

```
textView.text = "Hello, world!"
```

Objectif c:

```
textView.text = @"Hello, world!";
```

Définir le texte attribué

```
// 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
```

Changer l'alignement du texte

Rapide

```
textView.textAlignment = .left
```

Objectif c

```
textView.textAlignment = NSTextAlignmentLeft;
```

Méthodes UITextViewDelegate

Répondre aux notifications de modification

- `textViewShouldBeginEditing(_:)`
- `textViewDidBeginEditing(_:)`
- `textViewShouldEndEditing(_:)`
- `textViewDidEndEditing(_:)`

Répondre aux modifications de texte

- `textView(_:shouldChangeTextIn:replacementText:)`
- `textViewDidChange(_:)`

Répondre à l'URL

- `textView(_: UITextView, shouldInteractWithURL: NSURL, inRange: NSRange) -> Bool`

Changer la police

Rapide

```
//System Font
textView.font = UIFont.systemFont(ofSize: 12)

//Font of your choosing
textView.font = UIFont(name: "Font Name", size: 12)
```

Objectif c

```
//System Font
textView.font = [UIFont systemFontOfSize:12];

//Font of your choosing
textView.font = [UIFont fontWithName:@"Font Name" size:12];
```

Changer la couleur du texte

Rapide

```
textView.textColor = UIColor.red
```

Objectif c

```
textView.textColor = [UIColor redColor];
```

UITextView avec du texte 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];
```

Détection automatique de liens, adresses, dates et autres

`UITextView` a un support intégré pour détecter automatiquement une variété de données. Les données pouvant être détectées automatiquement comprennent actuellement:

```
enum {
   UIDataDetectorTypePhoneNumber   = 1 << 0,
   UIDataDetectorTypeLink          = 1 << 1,
   UIDataDetectorTypeAddress       = 1 << 2,
   UIDataDetectorTypeCalendarEvent = 1 << 3,
   UIDataDetectorTypeNone          = 0,
   UIDataDetectorTypeAll           = NSUIntegerMax
};
```

Activation de la détection automatique

```
// you may add as many as you like by using the `|` operator between options
textView.dataDetectorTypes = (UIDataDetectorTypeLink | UIDataDetectorTypePhoneNumber);
```

Si activé, le texte apparaîtra comme un lien hypertexte sur le `UITextView`

Données cliquables

Pour autoriser le clic sur le lien (ce qui entraînera différentes actions en fonction du type de données), vous devez vous assurer que `UITextView` est sélectionnable mais pas modifiable et que l'interaction de l'utilisateur est activée

```
textView.editable = NO;
textView.selectable = YES;
textView.userInteractionEnabled = YES; // YES by default
```

Vérifiez pour voir si vide ou nul

Rapide

```
if let text = self.textView.text where !text.isEmpty {
    // Do stuff for text
} else {
    // Do stuff for nil text or empty string
}
```

Objectif c

```
if (self.textView.text.length > 0){
    // Do stuff for text
}   else {
    // Do stuff for nil text or empty string
}
```

Obtenir et définir la position du curseur

Informations utiles

Le tout début du texte du champ de texte:

```
let startPosition: UITextPosition = textView.beginningOfDocument
```

La fin du texte du champ de texte:

```
let endPosition: UITextPosition = textView.endOfDocument
```

La gamme actuellement sélectionnée:

```
let selectedRange: UITextRange? = textView.selectedTextRange
```

Obtenir la position du curseur

```
if let selectedRange = textView.selectedTextRange {

    let cursorPosition = textView.offsetFromPosition(textView.beginningOfDocument, toPosition:
selectedRange.start)

    print("\(cursorPosition)")
}
```

Définir la position du curseur

Pour définir la position, toutes ces méthodes définissent une plage avec les mêmes valeurs de début et de fin.

Au début

```
let newPosition = textView.beginningOfDocument
textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

Jusqu'à la fin

```
let newPosition = textView.endOfDocument
textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

À une position à gauche de la position actuelle du curseur

```
// 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)
    }
}
```

À une position arbitraire

Commencez par le début et déplacez 5 caractères vers la droite.

```
let arbitraryValue: Int = 5
if let newPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: arbitraryValue) {

    textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
}
```

en relation

Sélectionner tout le texte

```
textView.selectedTextRange = textView.textRangeFromPosition(textView.beginningOfDocument,
toPosition: textView.endOfDocument)
```

Sélectionnez une plage de texte

```
// 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!)
}
```

Insérer du texte à la position actuelle du curseur

```
textView.insertText("Hello")
```

Remarques

- Cet exemple provient à l'origine d'une adaptation de cette réponse Stack Overflow .
- Cette réponse utilise un champ de texte, mais les mêmes concepts s'appliquent à `UITextView` .
- Utilisez `textView.becomeFirstResponder()` pour mettre en `textView.becomeFirstResponder()` le champ de texte et faire apparaître le clavier.
- Voir cette réponse pour savoir comment obtenir le texte à une certaine distance.

en relation

- Comment créer une plage dans Swift (traite indirectement de la question de savoir pourquoi nous devons utiliser `selectedTextRange` ici plutôt que de simplement `selectedRange`)

Supprimez les rembourrages supplémentaires pour les adapter à un texte mesuré avec précision.

`UITextView` a des `UITextView` supplémentaires par défaut. Parfois, c'est embêtant, surtout si vous voulez mesurer du texte sans instance de vue et le placer précisément dans une zone.

Faites ceci pour enlever de tels rembourrages.

```
messageTextView.textContainerInset = UIEdgeInsetsZero
messageTextView.textContainer.lineFragmentPadding = 0
```

Vous pouvez désormais mesurer la taille du texte à l'aide de `NSAttributedString.boundingRectWithSize(...)` et redimensionner un objet `UITextView` uniquement pour l'adapter au texte.

```
let budget = getSomeCGSizeBudget()
let text = getSomeAttributedString()
let textSize = text.boundingRectWithSize(budget, options: [.UsesLineFragmentOrigin,
.UsesFontLeading], context: nil).size
messageTextView.frame.size = textSize // Just fits.
```

Lire UITextView en ligne: https://riptutorial.com/fr/ios/topic/1043/uitextview

Chapitre 198: UIView

Syntaxe

1. // Objectif c
2. [UIView new] // Récupère un objet de vue alloué et initialisé
3. [[UIView alloc] initWithFrame: (Pass CGRect)] // Récupère la vue allouée et initialisée avec un cadre
4. [[UIView alloc] init] // Récupère un objet de vue alloué et initialisé
5. // Rapide
6. UIView () // Crée une instance UIView avec le cadre CGRect.zero
7. UIView (frame: CGRect) // Crée une instance UIView spécifiant le cadre
8. UIView.addSubview (UIView) // Ajout d'une autre instance UIView en tant que sous-vue
9. UIView.hidden // Récupère ou définit la visibilité de la vue
10. UIView.alpha // Récupère ou définit l'opacité de la vue
11. UIView.setNeedsLayout () // Force la vue à mettre à jour sa disposition

Remarques

La classe **UIView** définit une zone rectangulaire sur l'écran et les interfaces pour gérer le contenu de cette zone. Lors de l'exécution, un objet de vue gère le rendu de tout contenu dans sa zone et gère également les interactions avec ce contenu.

Examples

Créer un UIView

Objectif 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;
```

Rapide

```
let myFrame = CGRect(x: 0, y: 0, width: 320, height: 35)
let view = UIView(frame: myFrame)
```

Faire la vue arrondie

Pour un arrondi `UIView` , spécifiez un `cornerRadius` pour la vue de `layer` .

Cela applique également toute classe qui hérite de `UIView` , telle que `UIImageView` .

Par programme

Code rapide

```
someImageView.layoutIfNeeded()
someImageView.clipsToBounds = true
someImageView.layer.cornerRadius = 10
```

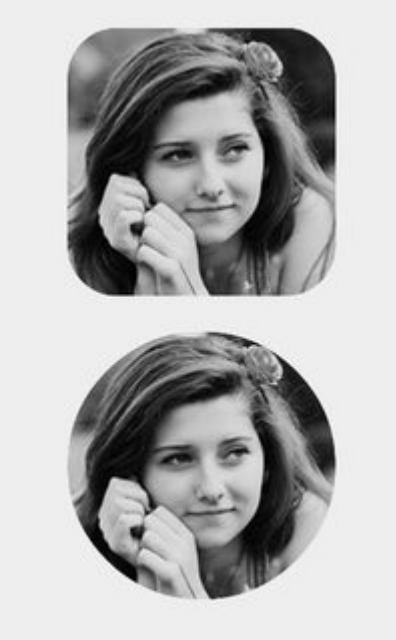
Code Objective-C

```
[someImageView layoutIfNeeded];
someImageView.clipsToBounds = YES;
someImageView.layer.cornerRadius = 10;
```

Exemple

```
//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;
```

Voici le résultat, montrant l'effet de la vue arrondie en utilisant le rayon de coin spécifié:



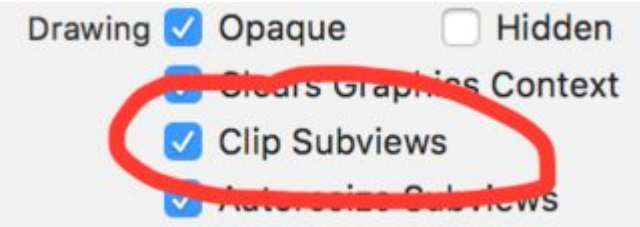
Remarque

Pour ce faire, vous devez inclure le framework QuartzCore.

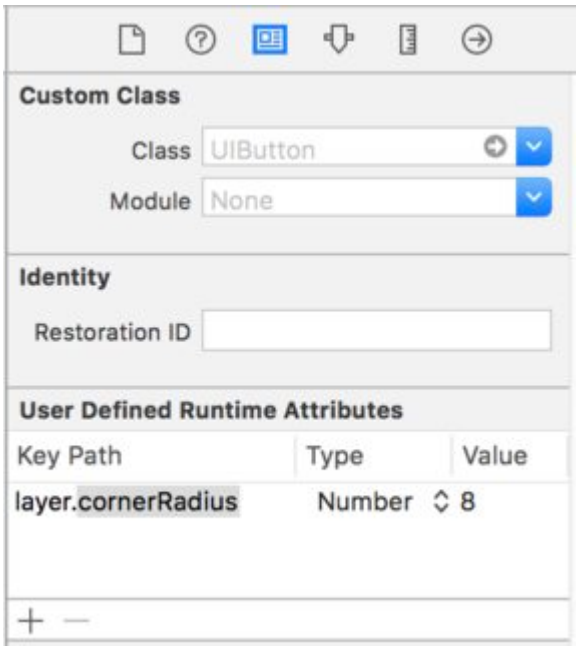
```
#import <QuartzCore/QuartzCore.h>
```

Configuration de storyboard

Un effet de vue arrondi peut également être obtenu de manière `non-programmatically` en définissant les propriétés correspondantes dans **Storyboard** .



Les propriétés des `layer` n'étant pas exposées dans Storyboard, vous devez modifier l'attribut `cornerRadius` via la section Attributs d'exécution définis par l'utilisateur.



Extension rapide

Vous pouvez utiliser cette extension pratique pour appliquer une vue arrondie tant qu'elle a la même largeur et la même hauteur.

```
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
    }
}
```

Pour l'utiliser:

```
yourView.setAsCircle()
```

Prendre un instantané

Vous pouvez prendre un instantané d'un `UIView` comme ceci:

Rapide

```
let snapshot = view.snapshotView(afterScreenUpdates: true)
```

Objectif c

```
UIView *snapshot = [view snapshotViewAfterScreenUpdates: YES];
```

Utilisation de IBInspectable et IBDesignable

`IBInspectable` propriétés `IBDesignable` et `IBDesignable` `UIView` sont une (ou deux) des nouvelles fonctionnalités les plus intéressantes des versions récentes de Xcode. Celles-ci n'ont rien à voir avec les fonctionnalités de votre application mais ont un impact sur l'expérience des développeurs dans Xcode. L'objectif est de pouvoir inspecter visuellement les vues personnalisées de votre application iOS sans l'exécuter. Supposons donc que vous ayez une vue personnalisée nommée `CustomView` qui hérite de `UIView` . Dans cette vue personnalisée, il affichera une chaîne de texte avec une couleur désignée. Vous pouvez également choisir de ne pas afficher de texte. Nous aurons besoin de trois propriétés:

```
var textColor: UIColor = UIColor.blackColor()
var text: String?
var showText: Bool = true
```

Nous pouvons alors remplacer la fonction `drawRect` dans la 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)!
            ])
    }
}
```

En supposant que la propriété `text` est définie, cela dessine une chaîne dans le coin supérieur gauche de la vue lorsque l'application est exécutée. Le problème est que nous ne saurons pas à quoi cela ressemble sans exécuter l'application. C'est là `IBInspectable` et `IBDesignable` . `IBInspectable` nous permet de définir visuellement les valeurs de propriété de la vue dans Xcode, comme avec les contrôles intégrés. `IBDesignable` nous montrera un aperçu visuel dans le storyboard. Voici comment la classe devrait ressembler:

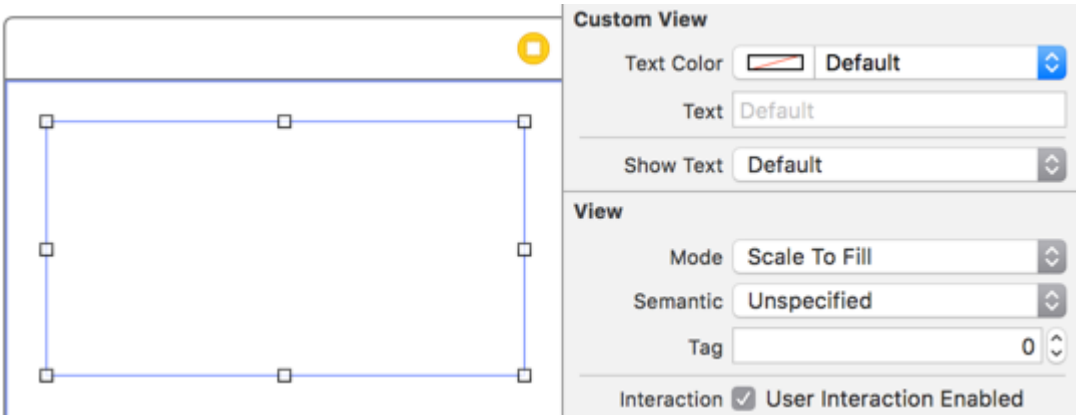
```
@IBDesignable
class CustomView: UIView {
    @IBInspectable var textColor: UIColor = UIColor.blackColor()
    @IBInspectable var text: String?
    @IBInspectable var showText: Bool = true

    override func drawRect(rect: CGRect) {
        // ...
    }
}
```

Ou dans l'objectif C:

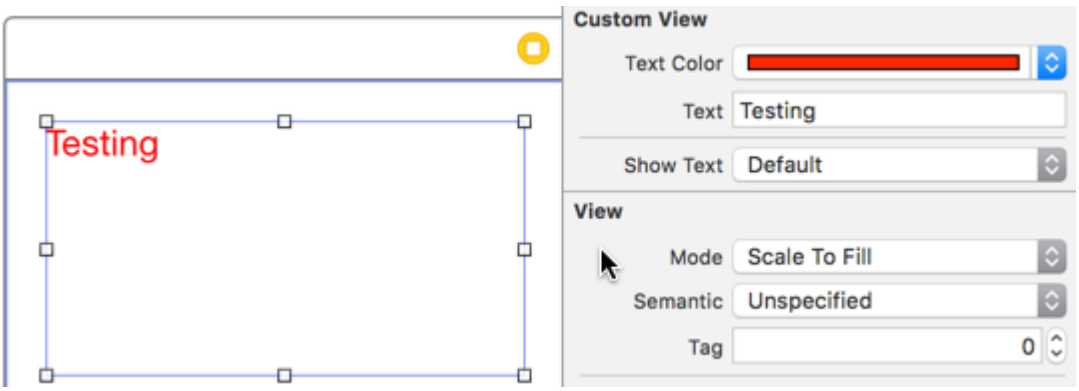
```
IB_DESIGNABLE
@interface CustomView: UIView

@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
```

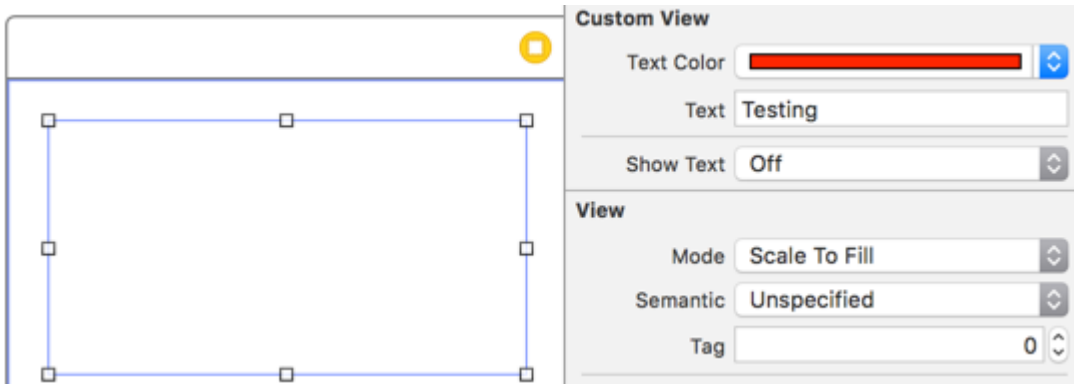
Les prochaines captures d'écran montrent ce qui se passe dans Xcode. Le premier est ce qui se passe après l'ajout de la classe révisée. Notez qu'il existe trois nouveaux éléments d'interface utilisateur pour les trois propriétés. La *couleur du texte* affiche un sélecteur de couleurs, le *texte* est juste une zone de saisie et *Afficher le texte* nous donne les options pour `Off` et `On` qui sont respectivement `false` et `true` .



Le suivant est après avoir changé la *couleur* du *texte* en rouge en utilisant le sélecteur de couleur. De plus, du texte a été fourni pour que la fonction `drawRect` l'affiche. Notez que la vue dans Interface Builder a également été mise à jour.

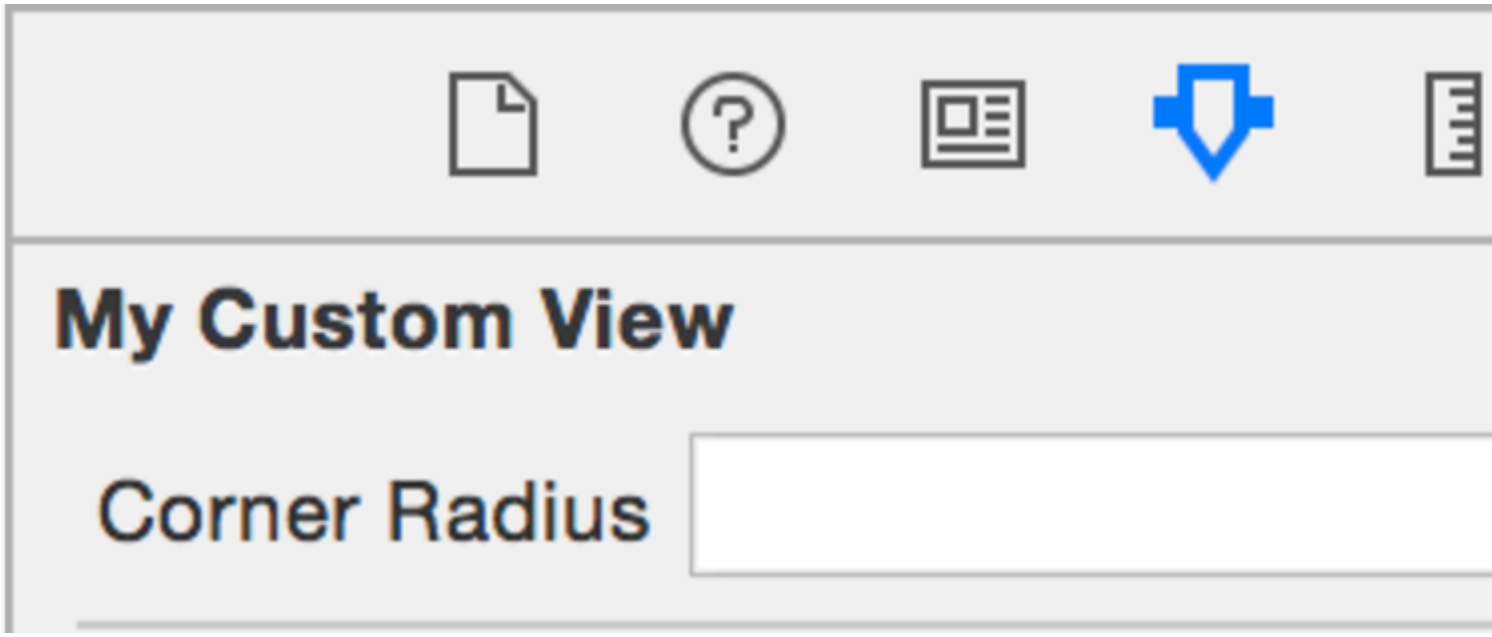


Enfin, la définition de l'option *Afficher le texte* sur `Off` dans l'inspecteur de propriétés fait disparaître l'affichage de texte dans Interface Builder.



Cependant, nous avons tous besoin de créer un `UIView` arrondi à plusieurs vues de votre `Storyboard` Au lieu de déclarer `IBDesignable` à chaque vue de `Storyboard` , il est préférable de créer une `Extension` de `UIView` et de créer une interface utilisateur `UIView` chaque `UIView` tout au long du projet pour créer une vue arrondie en définissant le rayon du coin. Un rayon de bordure configurable sur toute UIView que vous créez dans le storyboard.

```
extension UIView {

    @IBInspectable var cornerRadius:CGFloat {
        set {
            layer.cornerRadius = newValue
            clipsToBounds = newValue > 0
        }
        get {
            return layer.cornerRadius
        }
    }

}
```



Animation d'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()
}
```

Extension UIView pour les attributs de taille et de cadre

Si nous voulons obtenir la valeur d'origine de la vue, nous devons écrire comme suit:

```
view.frame.origin.x
```

Pour la largeur, il faut écrire:

```
view.frame.size.width
```

Mais si nous ajoutons une simple extension à un `UIView` , nous pouvons obtenir tous les attributs très simplement, comme:

```
view.x
view.y
view.width
view.height
```

Cela aidera également à définir ces attributs comme:

```
view.x = 10
view.y = 10
view.width = 100
view.height = 200
```

Et la simple extension serait:

```
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)
        }
    }
}
```

Nous devons ajouter ce fichier de classe dans un projet et il sera disponible pour être utilisé tout au long du projet!

Gestion par programmation de l'insertion et de la suppression de UIView dans et à partir d'un autre UIView

Supposons que vous ayez un `parentView` dans lequel vous voulez insérer un nouveau `subView` programmation (par exemple, lorsque vous voulez insérer un `UIImageView` dans la vue d'un `UIViewController`), que vous pouvez le faire comme ci-dessous.

Objectif c

```
[parentView addSubview:subView];
```

Rapide

```
parentView.addSubview(subView)
```

Vous pouvez également ajouter le subView sous un autre `subView2` , qui est déjà une sous-vue de parentView en utilisant le code suivant:

Objectif c

```
[parentView insertSubview:subView belowSubview:subView2];
```

Rapide

```
parentView.insertSubview(subView, belowSubview: subView2)
```

Si vous voulez l'insérer au-dessus de `subView2` vous pouvez le faire de cette façon:

Objectif c

```
[parentView insertSubview:subView aboveSubview:subView2];
```

Rapide

```
parentView.insertSubview(subView, aboveSubview: subView2)
```

Si quelque part dans votre code vous devez mettre en avant une certaine `subView` vue, donc au-dessus de toutes les sous- `parentView` de parentView, vous pouvez le faire comme ceci:

Objectif c

```
[parentView bringSubviewToFront:subView];
```

Rapide

```
parentView.bringSubviewToFront(subView)
```

Enfin, si vous souhaitez supprimer `subView` de `parentView` , vous pouvez faire comme ci-dessous:

Objectif c

```
[subView removeFromSuperview];
```

Rapide

```
subView.removeFromSuperview()
```

Créer UIView en utilisant 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];
```

Les fonctions

Fonction pour ajouter une vue avec une hauteur fixe à l'aide de contraintes d'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];

}
```

Fonction ajoute une contrainte de redimensionnement complète pour UIView créé.

```
-(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];
}
```

Utilisation de la taille du contenu intrinsèque

Lors de la création d'une sous-classe UIView, la taille du contenu intrinsèque permet d'éviter de définir des contraintes de hauteur et de largeur codées en dur

un aperçu de base sur la façon dont une classe peut utiliser cette

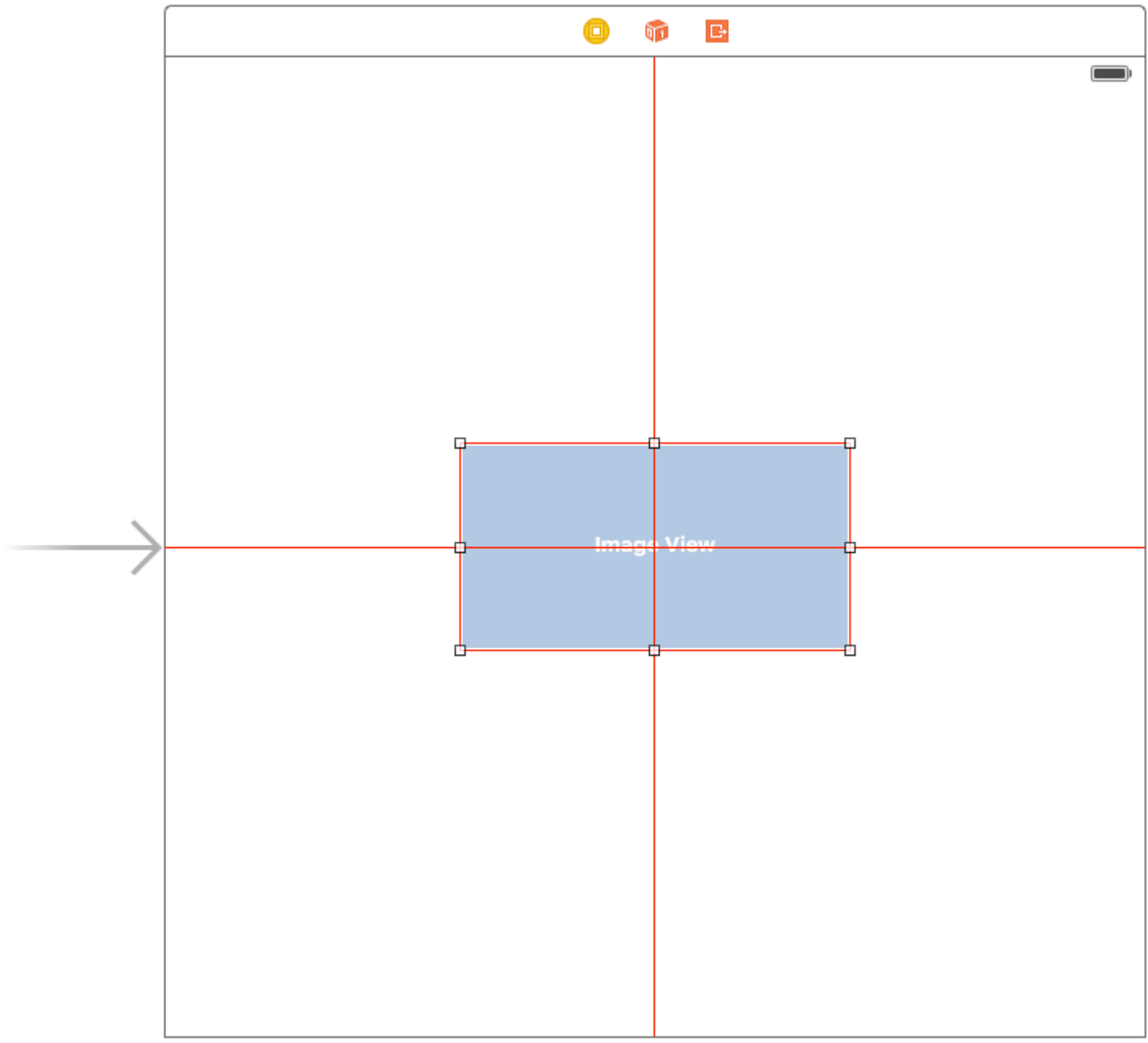
```
class ImageView: UIView {
    var image: UIImage {
        didSet {
            invalidateIntrinsicContentSize()
        }
    }
    // omitting initializers
    // convenience init(image: UIImage)

    override func intrinsicContentSize() -> CGSize {
        return CGSize(width: image.size.width, height: image.size.height)
    }
}
```

Si vous souhaitez uniquement fournir une taille intrinsèquement, vous pouvez fournir la valeur `UIViewNoIntrinsicMetric` pour la valeur que vous souhaitez ignorer.

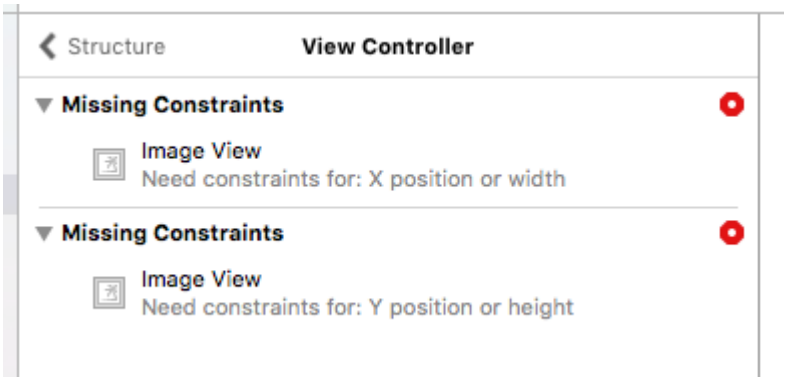
```
override func intrinsicContentSize() -> CGSize {
    return CGSize(width: UIViewNoIntrinsicMetric, height: image.size.width)
}
```

Avantages lors de l'utilisation avec AutoLayout et Interface Builder

On pourrait prendre cette ImageView (ou UIImageView) et définir l'alignement horizontal au centre X de superview et l'alignement vertical au centre Y de superview.

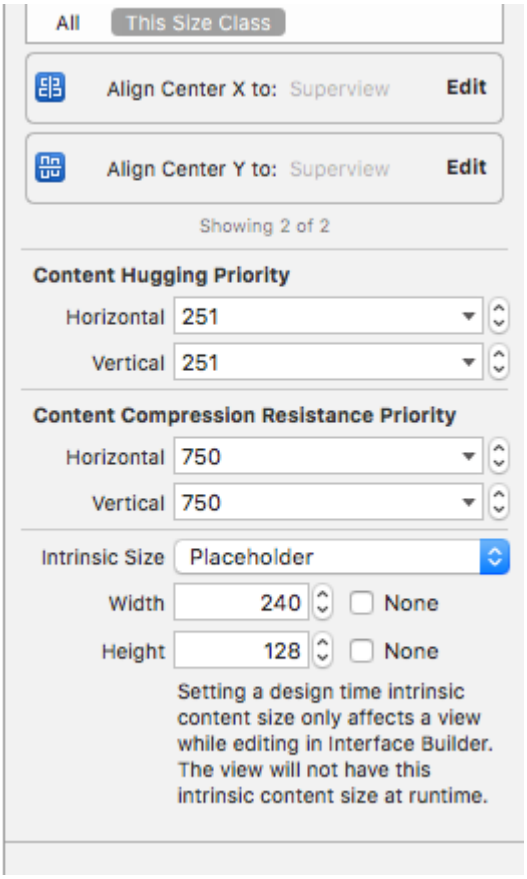


Le constructeur de l'interface se plaindra à ce stade en vous donnant l'avertissement suivant:



C'est ici que la `Placeholder Intrinsic Size` entre en jeu.

En entrant dans le panneau de l'inspecteur Taille et dans la liste déroulante Taille intrinsèque, vous pouvez basculer cette valeur de Par défaut à Espace réservé.



et maintenant, le générateur d'interface supprimera les avertissements précédents et vous pourrez utiliser cette taille pour avoir des vues de taille dynamique définies dans le générateur d'interface.

Secouer une vue

```
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")
    }
}
```

Cette fonction peut être utilisée pour attirer l'attention sur une vue spécifique en la secouant un peu.

Lire UIView en ligne: https://riptutorial.com/fr/ios/topic/858/uiview

Chapitre 199: UIViewController

Examples

Sous-classement

Sous- `UIControl` nous donne accès aux méthodes suivantes:

- `beginTrackingWithTouch` est appelé lorsque le doigt touche pour la première fois dans les limites du contrôle.
- `continueTrackingWithTouch` est appelé à plusieurs reprises lorsque le doigt glisse sur le contrôle et même en dehors des limites du contrôle.
- `endTrackingWithTouch` est appelé lorsque le doigt est retiré de l'écran.

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

Voici comment le contrôleur de vue est configuré pour être le délégué et répondre aux événements tactiles de notre contrôle personnalisé.

```
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"
    }
}
```

Remarques

- D'autres méthodes permettant d'obtenir le même résultat sans sous-classer incluent l'ajout d'une cible ou l'utilisation d'un identificateur de geste.
- Il n'est pas nécessaire d'utiliser un délégué avec ces méthodes si elles ne sont utilisées que dans le contrôle personnalisé lui-même. Nous aurions pu simplement ajouter une déclaration `print` pour montrer comment les événements sont appelés. Dans ce cas, le code serait simplifié pour

```
  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")
        }
    }
```

Créer une instance

Rapide

```
let viewController = UIViewController()
```

Objectif c

```
UIViewController *viewController = [UIViewController new];
```

Définir la vue par programmation

Rapide

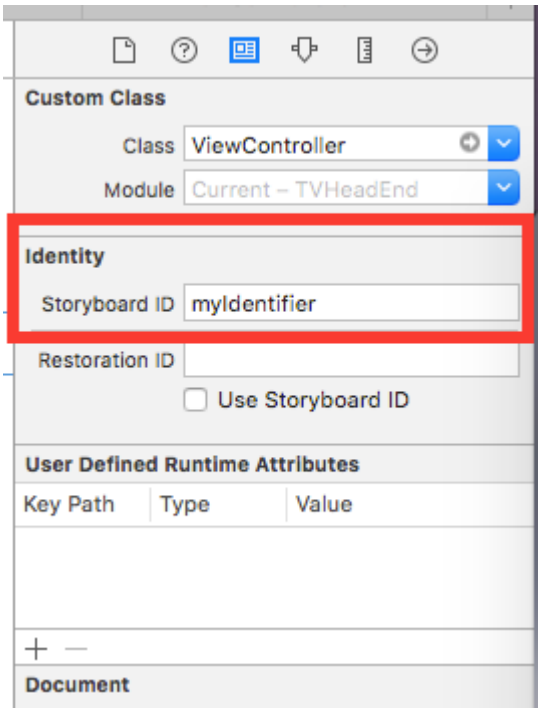
```
class FooViewController: UIViewController {

  override func loadView() {
    view = FooView()
  }

}
```

Instancier depuis un storyboard

```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
```

Avec un identifiant :

Donnez à la scène un ID Storyboard dans l'inspecteur d'identité du storyboard.

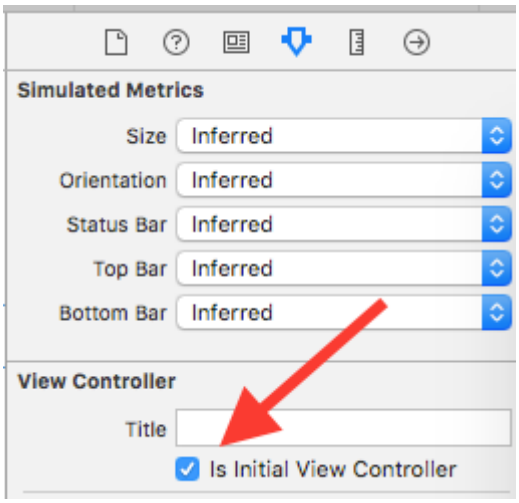


Instancier en code:

```
UIViewController *controller = [storyboard
instantiateViewControllerWithIdentifier:@"myIdentifier"];
```

Instancier un viewcontroller initial :

Dans le storyboard, sélectionnez le contrôleur de vue, puis sélectionnez l'inspecteur d'attribut, cochez la case "Est-ce que le contrôleur de vue initiale est".




```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
UIViewController *controller = [storyboard instantiateInitialViewController];
```

Accéder au contrôleur de vue conteneur

Lorsque le contrôleur de vue est présenté dans un contrôleur de barre d'onglets, vous pouvez accéder au contrôleur de la barre d'onglets comme suit:

Rapide

```
let tabBarController = viewController.tabBarController
```

Objectif c

```
UITabBarController *tabBarController = self.tabBarController;
```

Lorsque le contrôleur de vue fait partie d'une pile de navigation, vous pouvez accéder au contrôleur de navigation comme suit:

Rapide

```
let navigationController = viewController.navigationController
```

Objectif c

```
UINavigationController *navigationController = self.navigationController;
```

Ajout / suppression d'un contrôleur de vue enfant

Pour ajouter un contrôleur de vue enfant:

```
- (void)displayContentController:(UIViewController *)vc {
   [self addChildViewController:vc];
   vc.view.frame = self.view.frame;
   [self.view addSubview:vc.view];
   [vc didMoveToParentViewController:self];
}
```

Pour supprimer un contrôleur de vue enfant:

```
- (void)hideContentController:(UIViewController *)vc {
   [vc willMoveToParentViewController:nil];
   [vc.view removeFromSuperview];
   [vc removeFromParentViewController];
}
```

Lire UIViewController en ligne: https://riptutorial.com/fr/ios/topic/1956/uiviewcontroller

Chapitre 200: UIViews personnalisées à partir de fichiers XIB

Remarques

Depuis Apple: création d'une vue personnalisée qui rend dans Interface Builder

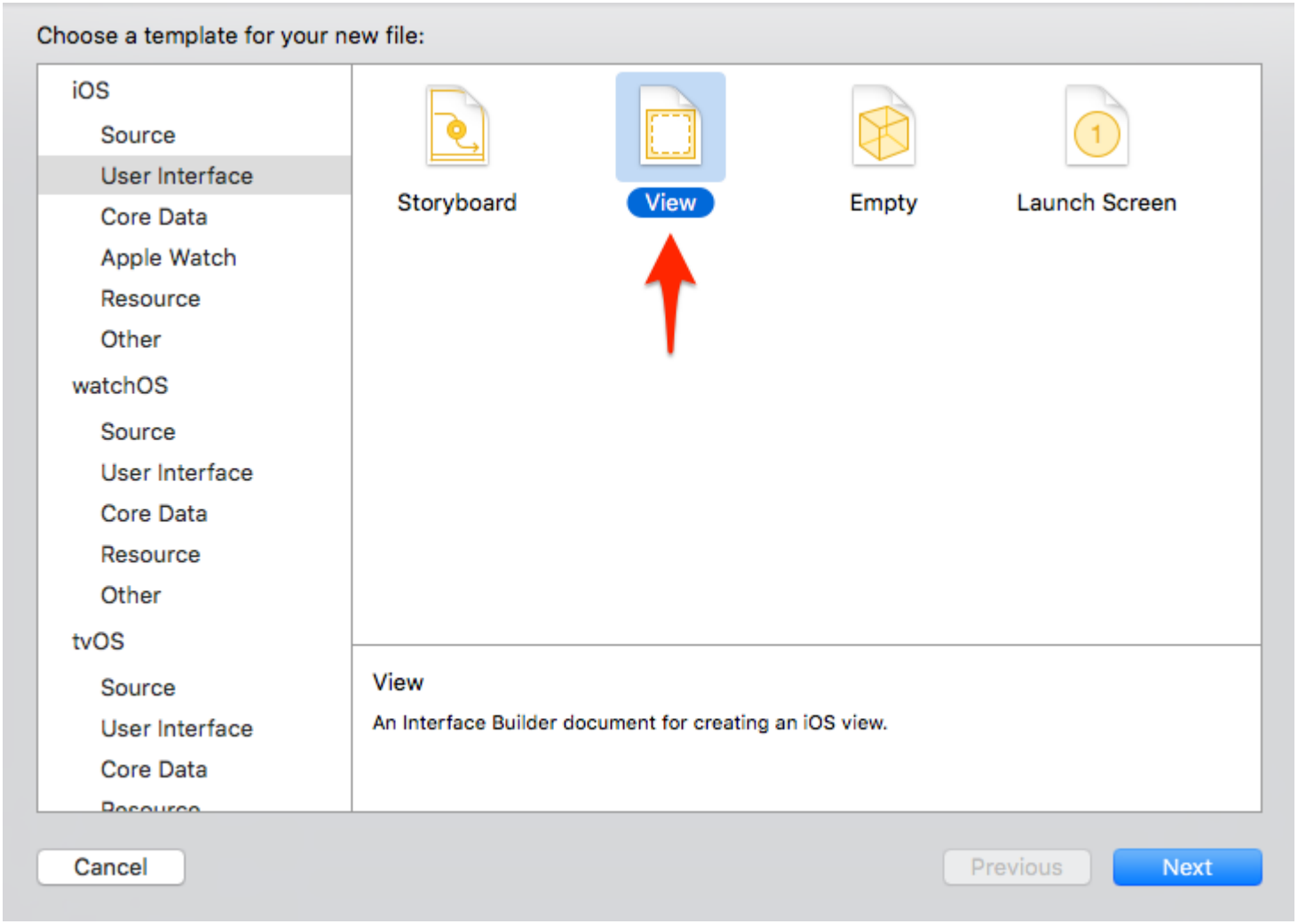
- Remarque: gardez à l'esprit que si vous utilisiez des polices personnalisées dans vos éléments XIB (tels UILabel, UITextField, etc.), le temps de chargement initial de votre XIB sera plus long en fonction de la police choisie et de la version du système.

Examples

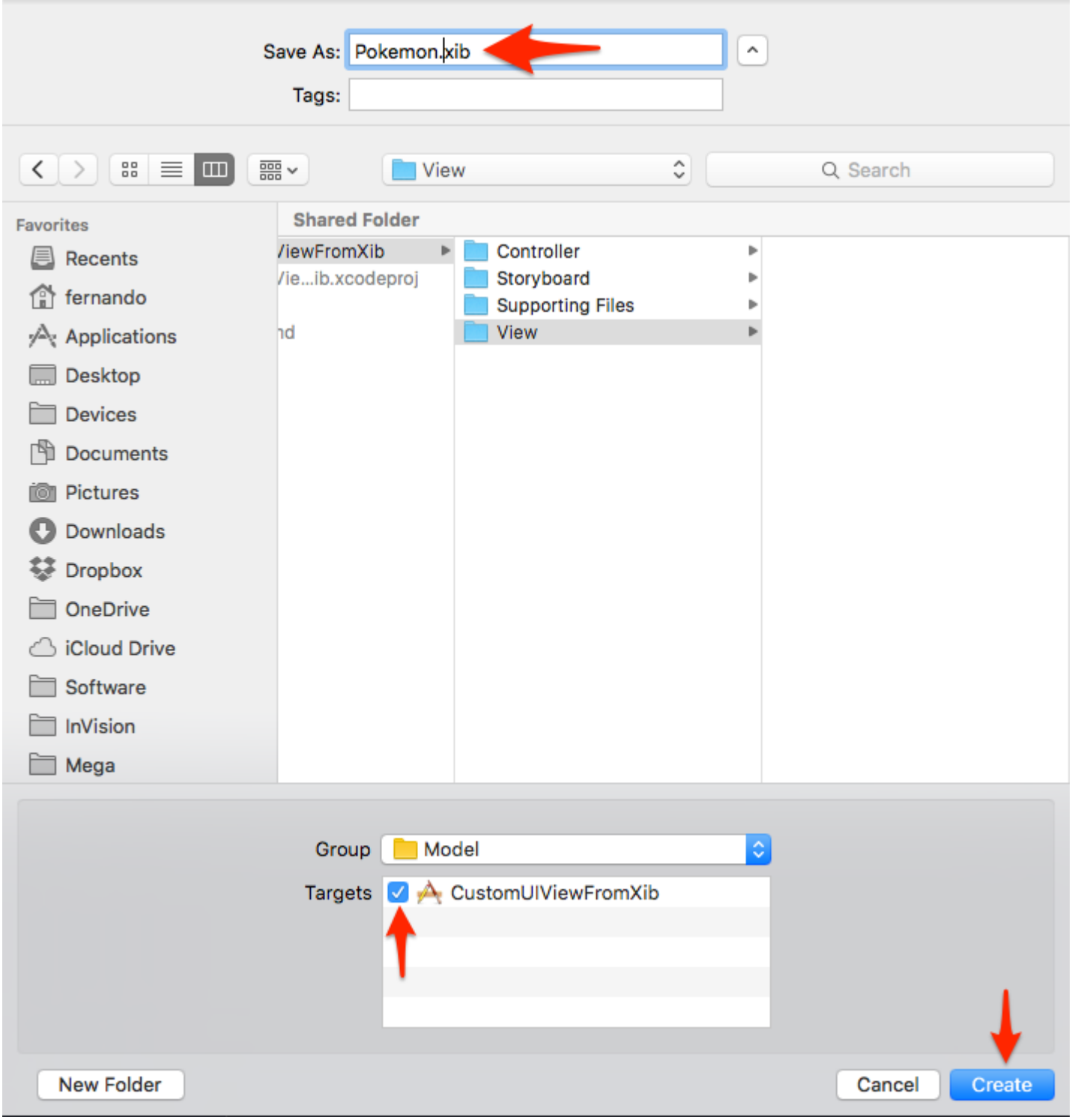
Éléments de câblage

Créer un fichier XIB

Barre de menus Xcode> Fichier> Nouveau> Fichier.
Sélectionnez iOS, Interface utilisateur puis "Afficher":



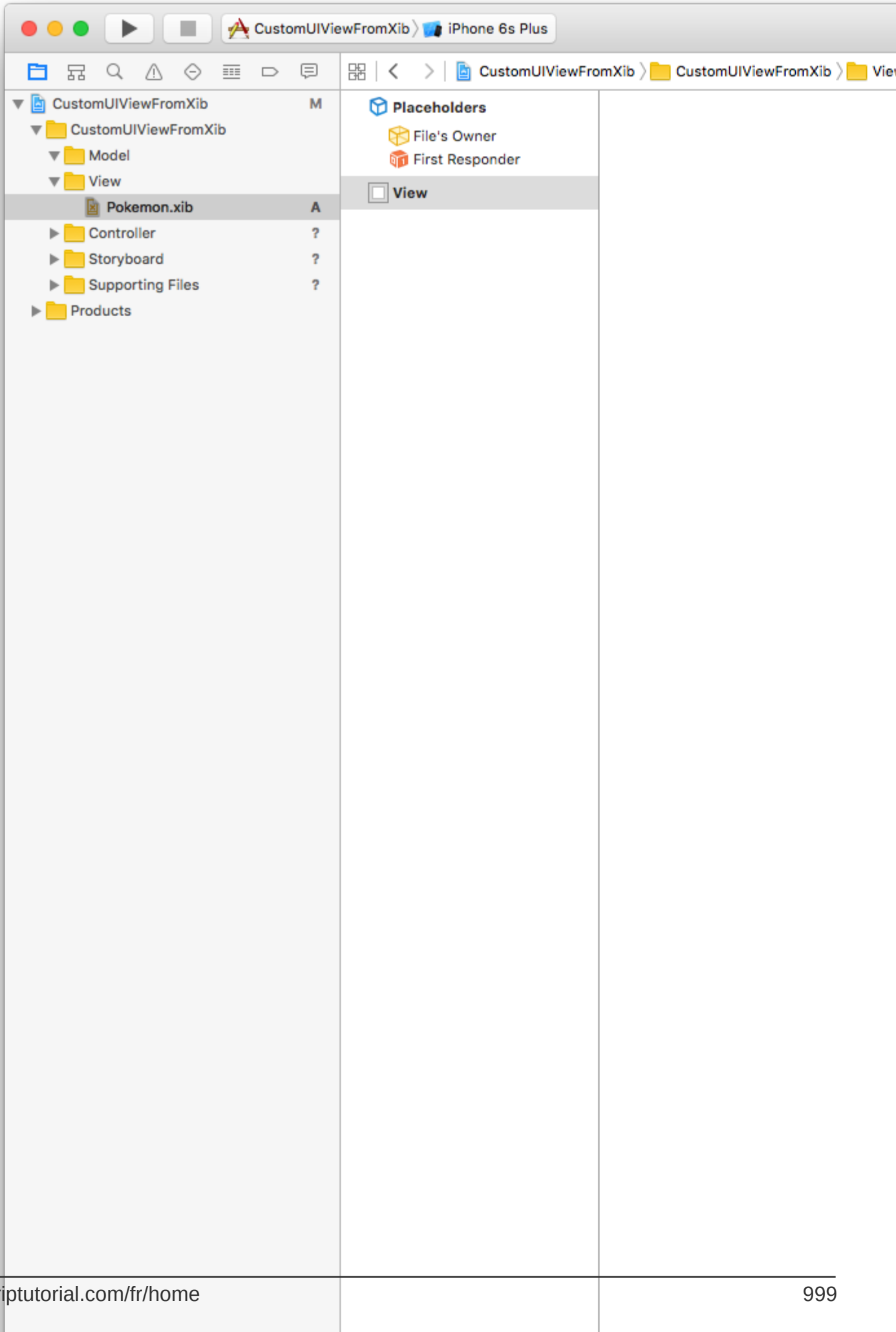
Donnez un nom à votre XIB (oui, nous faisons un exemple Pokemon).
N'oubliez pas de vérifier votre cible et appuyez sur "Créer".



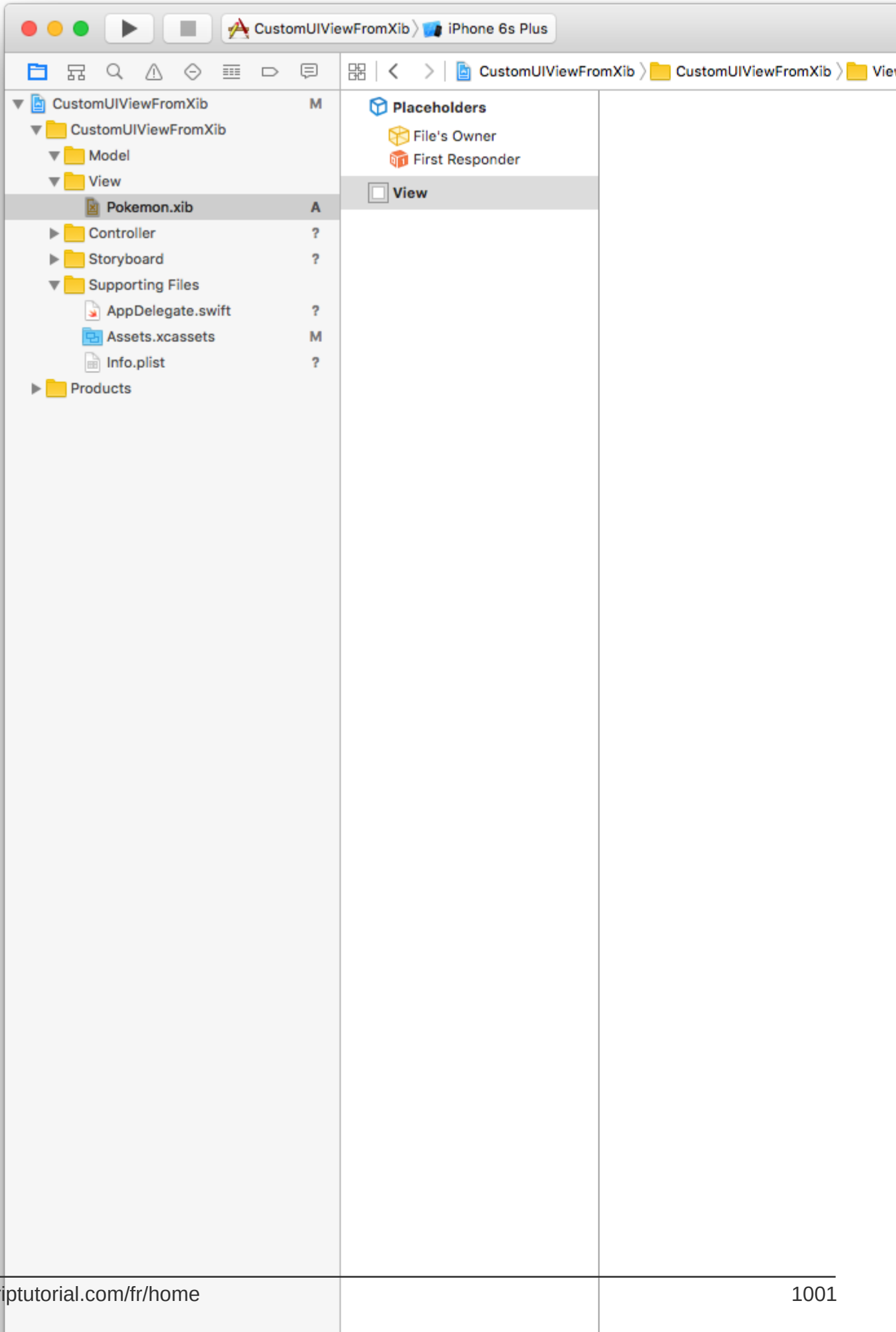
Concevez votre vue

Pour faciliter les choses, définissez:

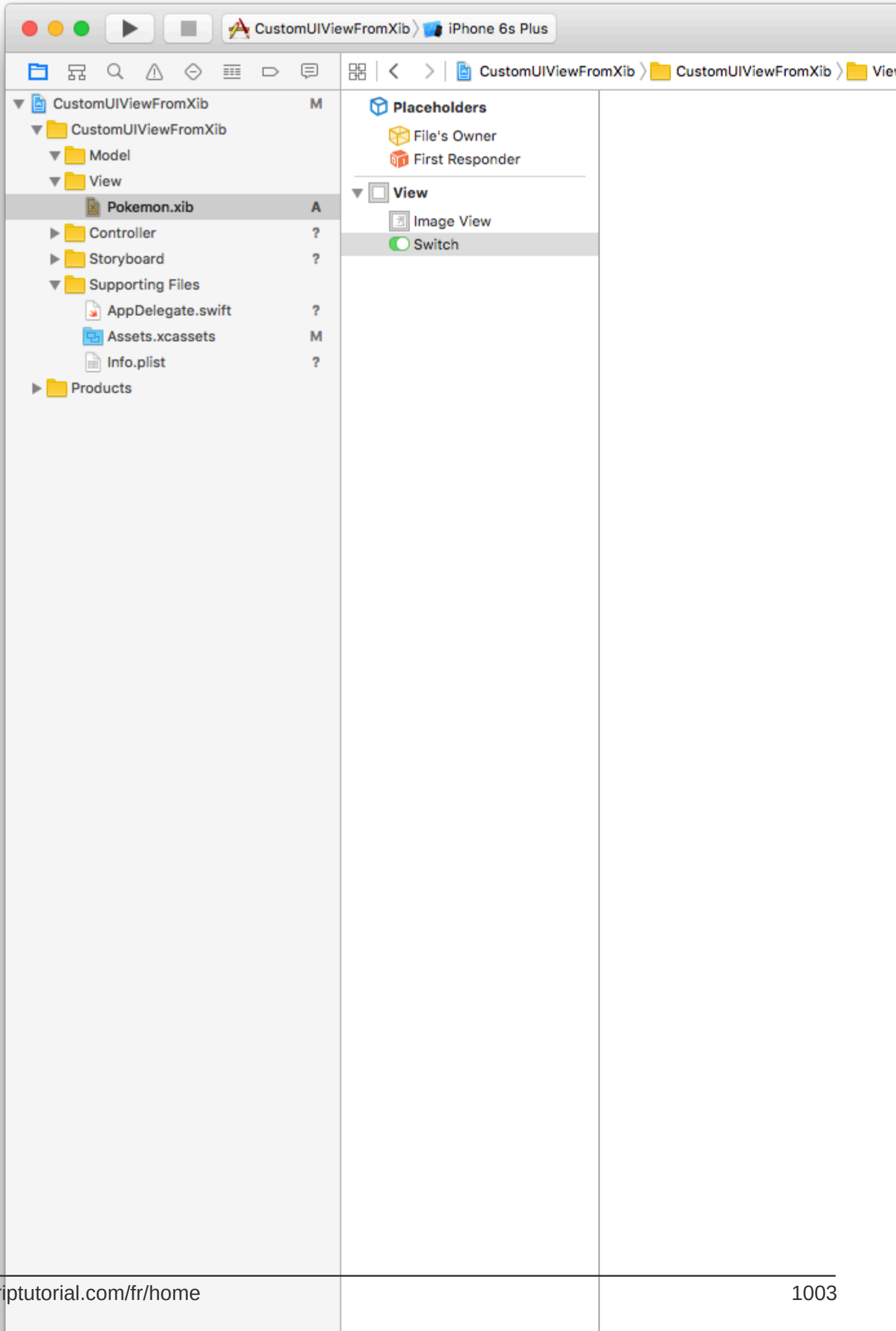
- Taille: Freeform
- Barre d'état: aucune
- Barre supérieure: aucune
- Barre inférieure: aucune


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

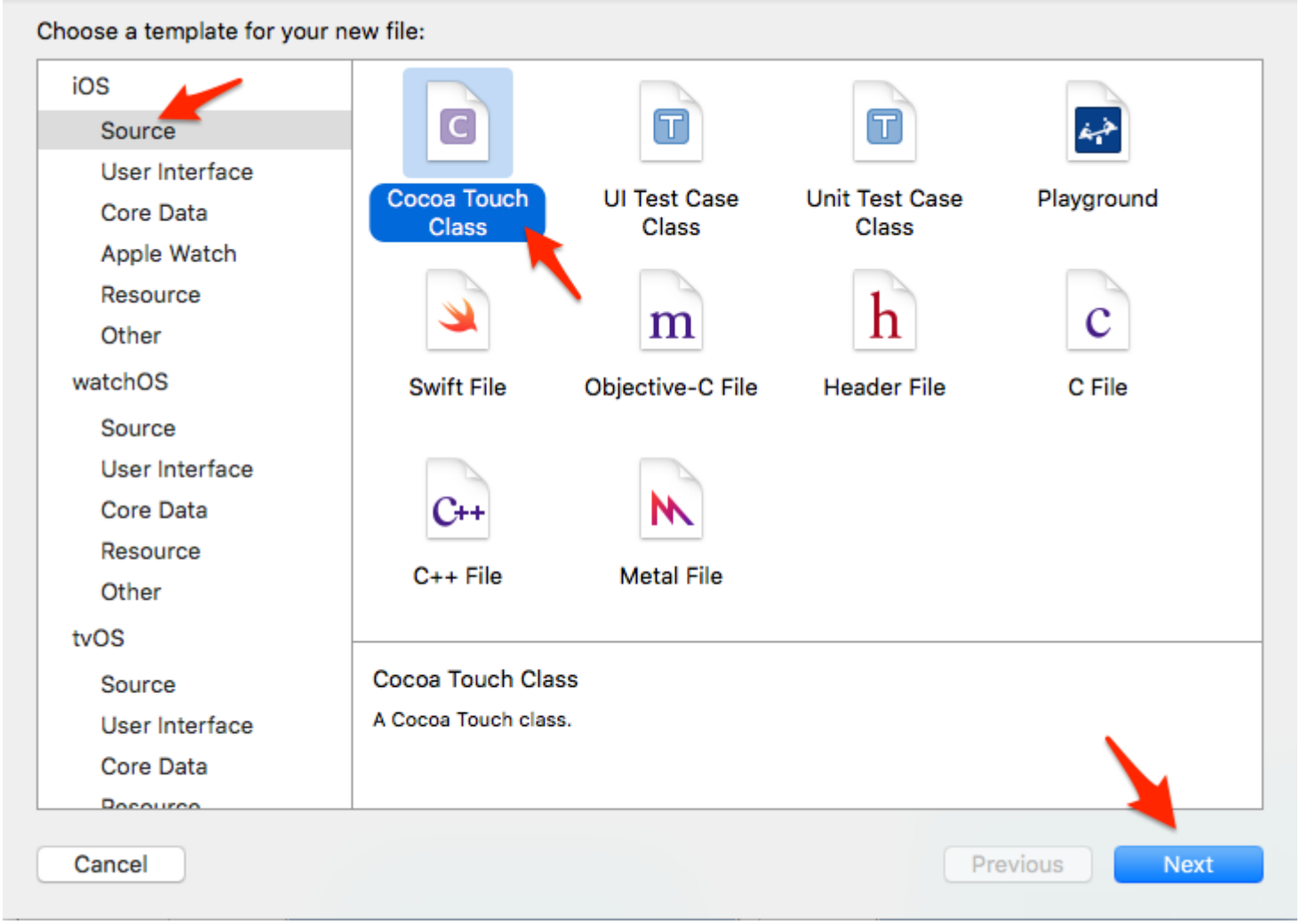
Pour cet exemple, nous utiliserons la largeur 321 et la hauteur 256.


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

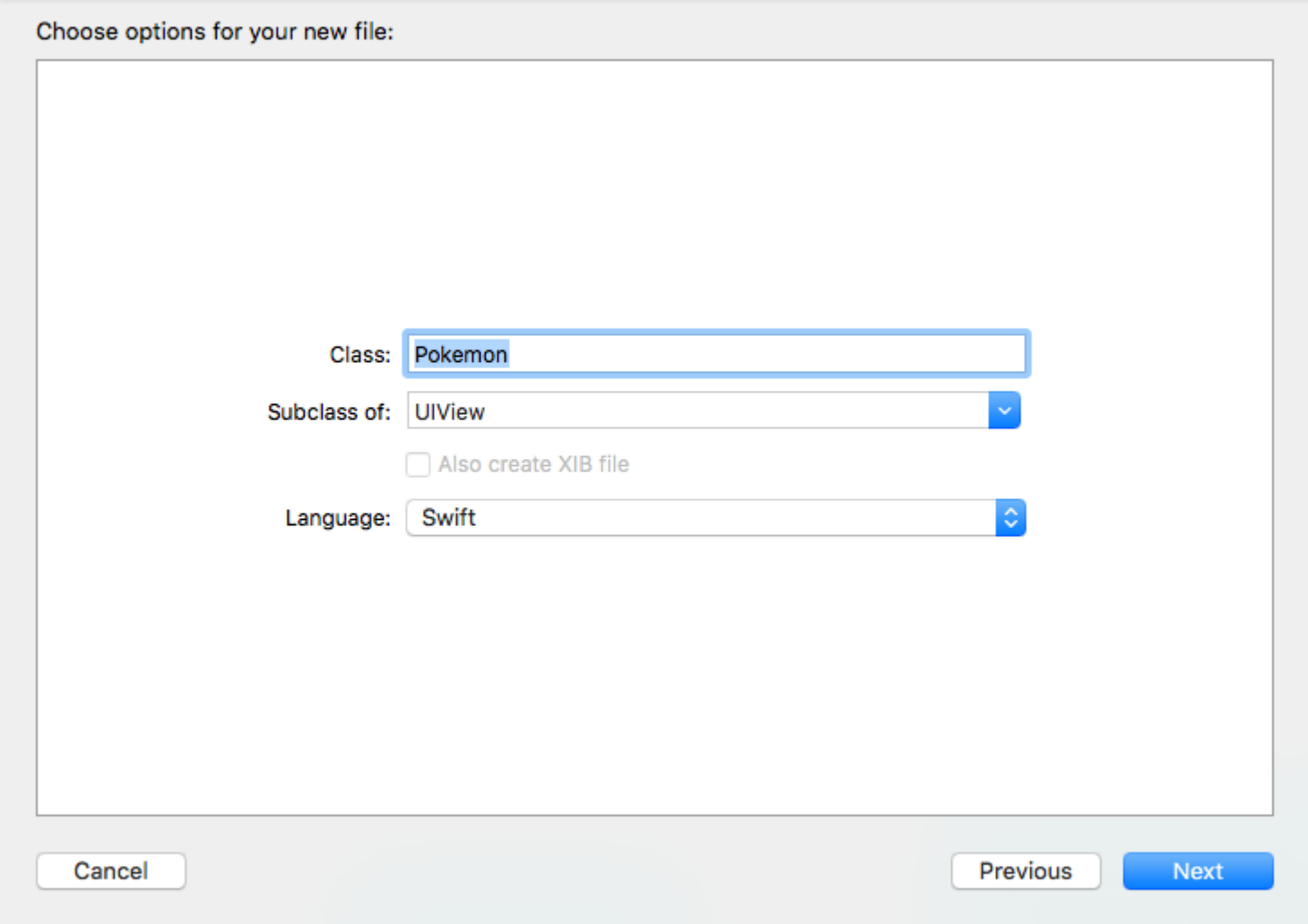
Ici, nous allons ajouter une **vue d'image** (256x256) et un **commutateur** .


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

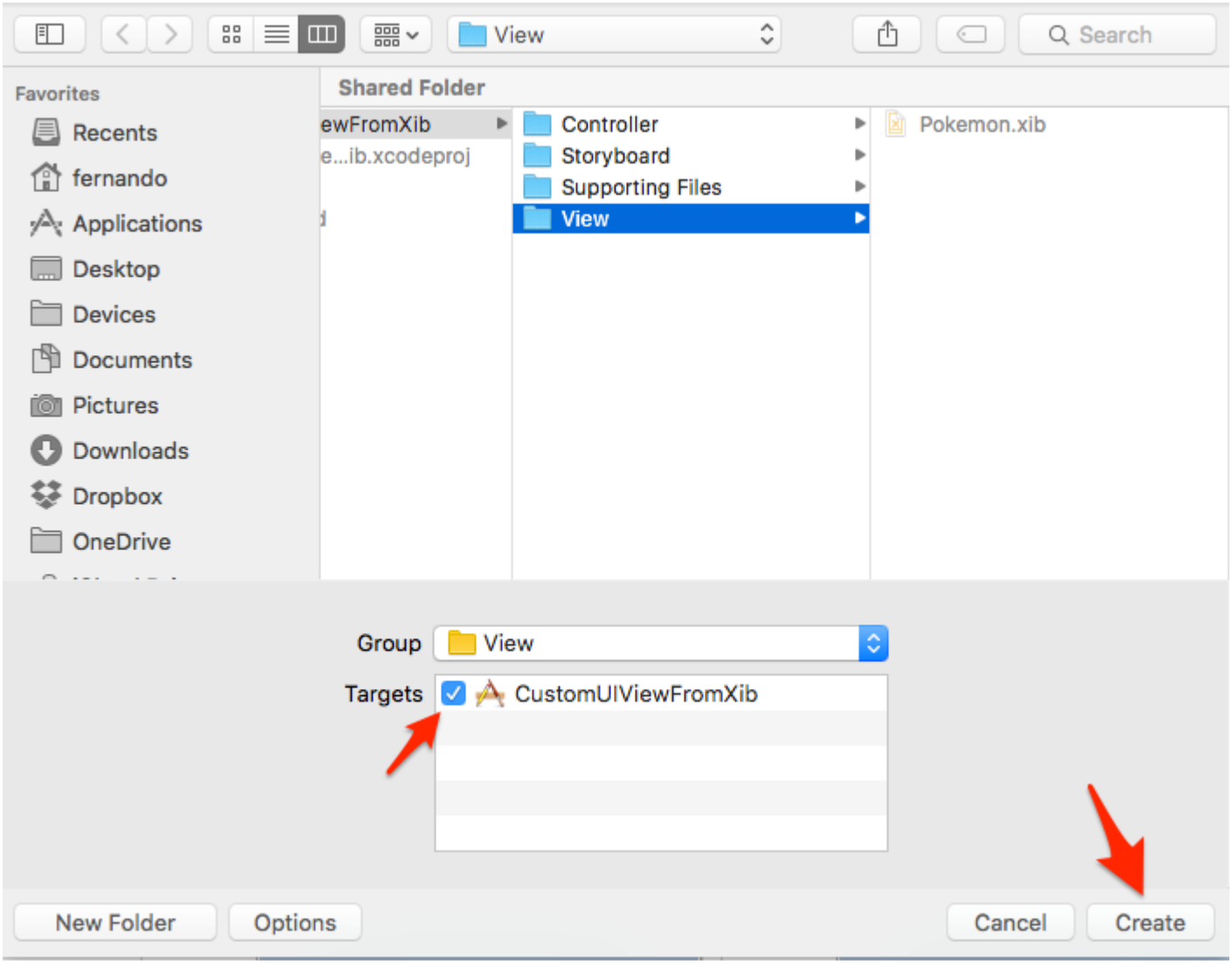
Barre de menus Xcode> Fichier> Nouveau> Fichier.
Sélectionnez iOS / Source / Cocoa Touch Class. Hit "Suivant".



Donnez un nom à la classe, qui doit porter le même nom que le fichier XIB (Pokemon).
Sélectionnez UIView comme type de sous-classe, puis cliquez sur "Suivant".

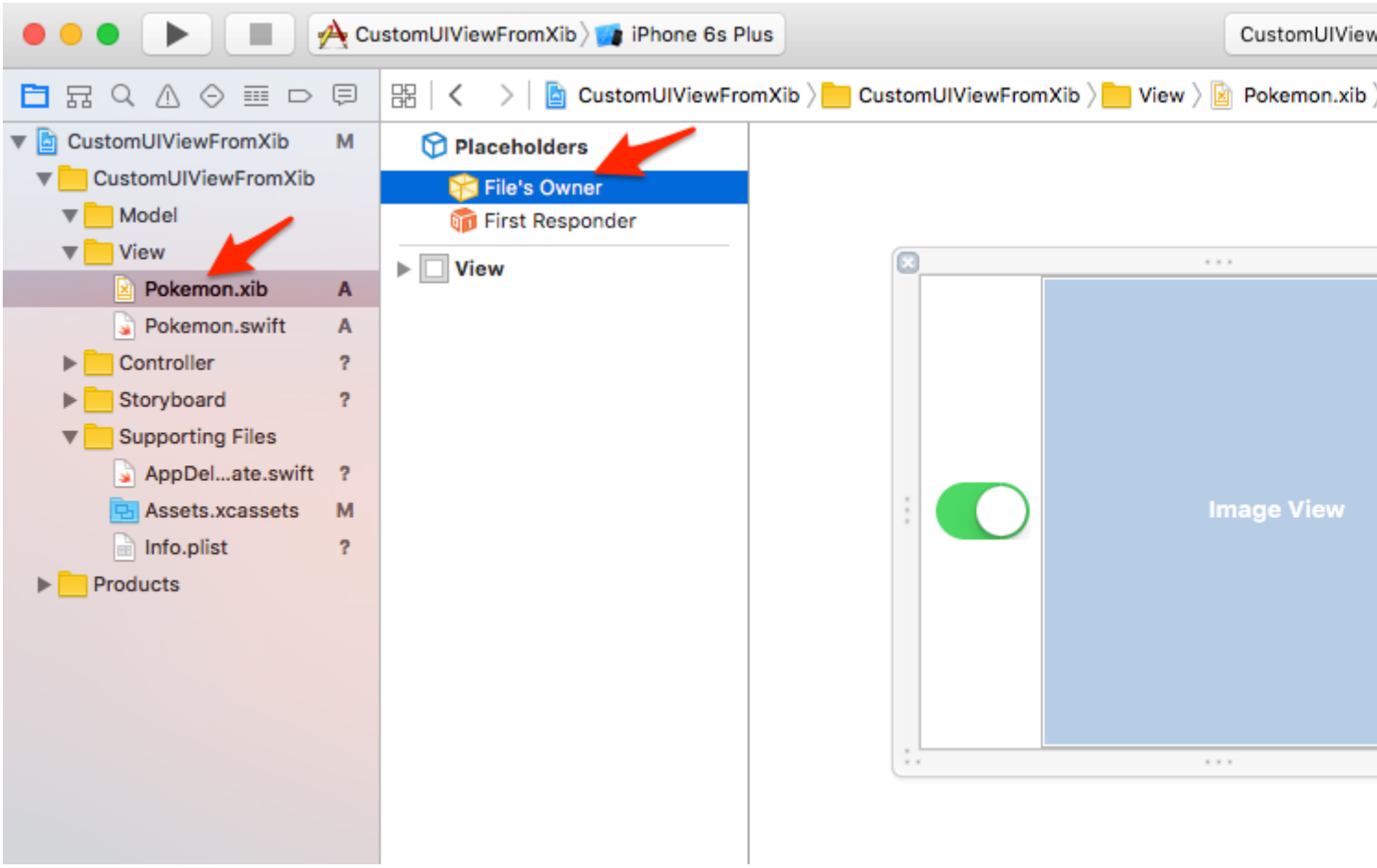


Dans la fenêtre suivante, sélectionnez votre cible et appuyez sur "Créer".



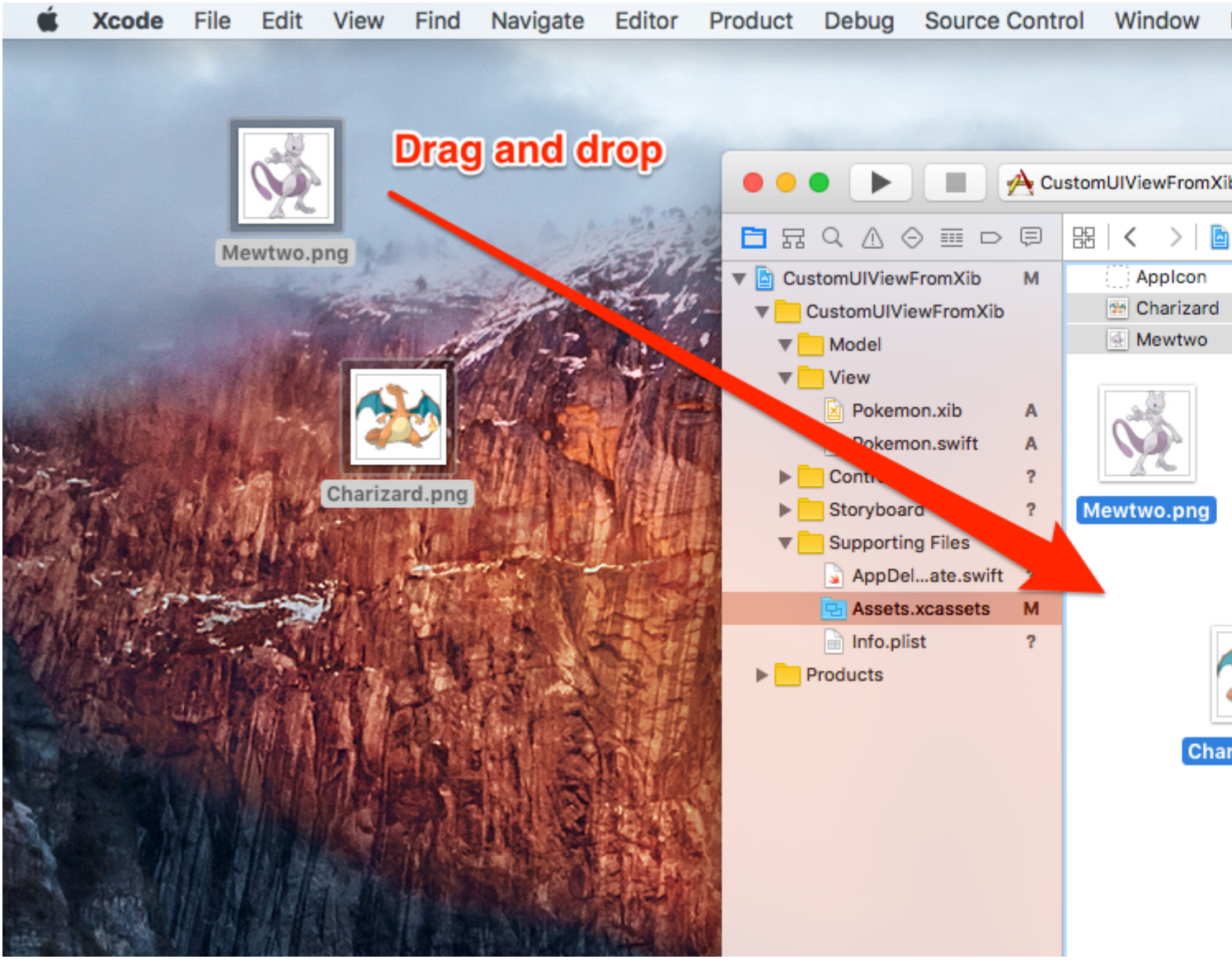
Connectez Pokemon.xib à Pokemon.swift via l'attribut "File's Owner"

Cliquez sur le fichier Pokemon.xib dans Xcode.
Cliquez sur le "Propriétaire du fichier".
Dans "Inspecteur d'identité" (en haut à droite), définissez la classe sur notre fichier Pokemon.swift récemment créé.



POKEMONS !!!

Oui! Glissez et déposez des Pokemons dans votre projet pour terminer notre "infrastructure". Nous ajoutons ici deux fichiers PGN, 256x256, transparents.



> Montrez-moi déjà le code.

D'accord, d'accord.
Il est temps d'ajouter du code à notre classe Pokemon.swift.

C'est en fait assez simple:

1. Implémenter les initialiseurs requis
2. Chargez le fichier XIB
3. Configurez la vue qui affichera le fichier XIB
4. Montrer la vue ci-dessus

Ajoutez le code suivant à la classe Pokemon.swift:

```
import UIKit

class Pokemon: UIView {

    // MARK: - Initializers

    override init(frame: CGRect) {
```

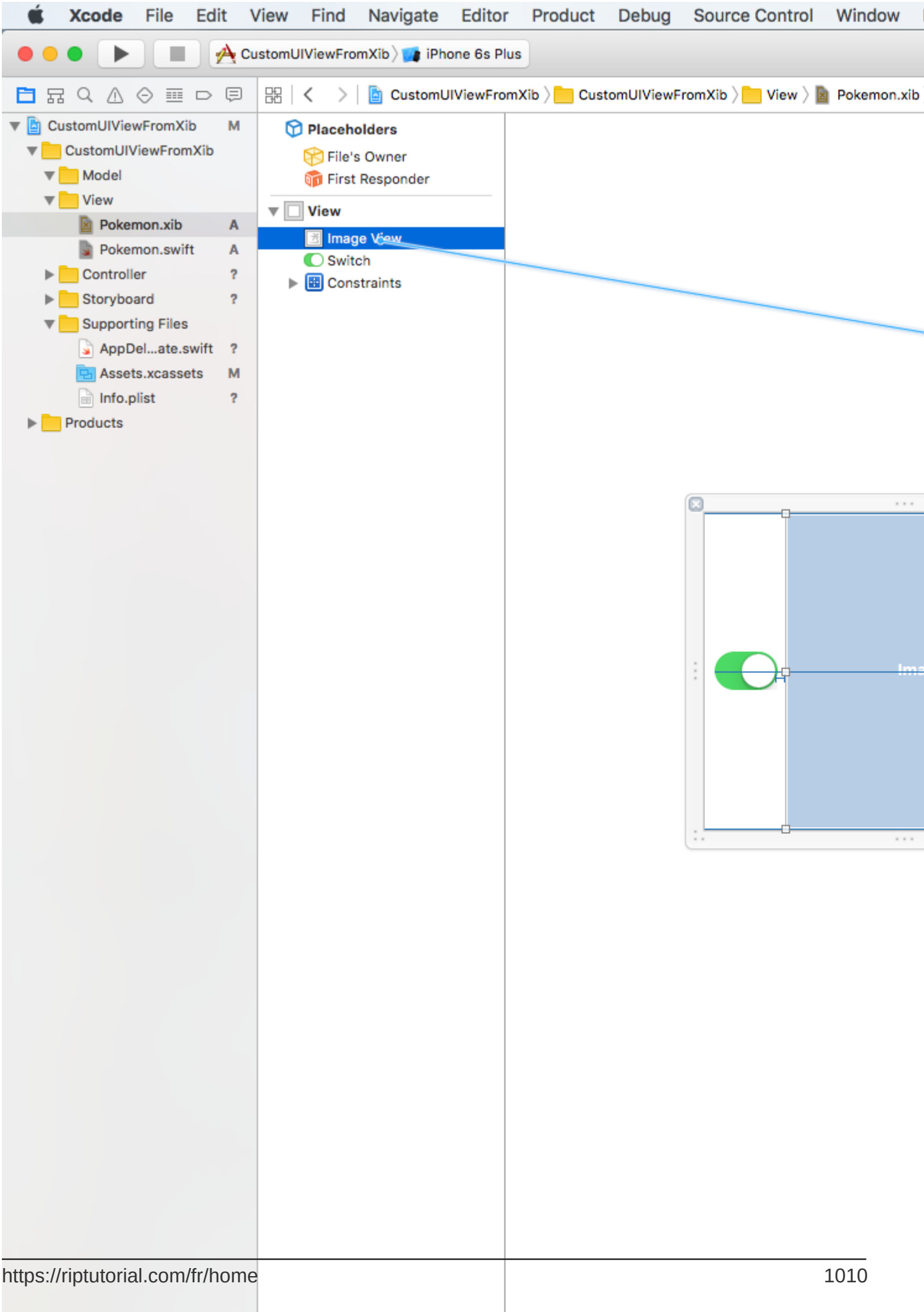
```
        super.init(frame: frame)
        setupView()
    }

    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 et @IBInspectable

En ajoutant `@IBDesignable` à votre classe, vous pouvez le rendre en direct dans Interface Builder. En ajoutant `@IBInspectable` aux propriétés de votre classe, vous pouvez voir vos vues personnalisées changer dans Interface Builder dès que vous modifiez ces propriétés.

Faisons la `Image View` de notre vue personnalisée "Inspectable".

Tout d'abord, connectez la `Image View` du fichier Pokemon.xib à la classe Pokemon.swift.

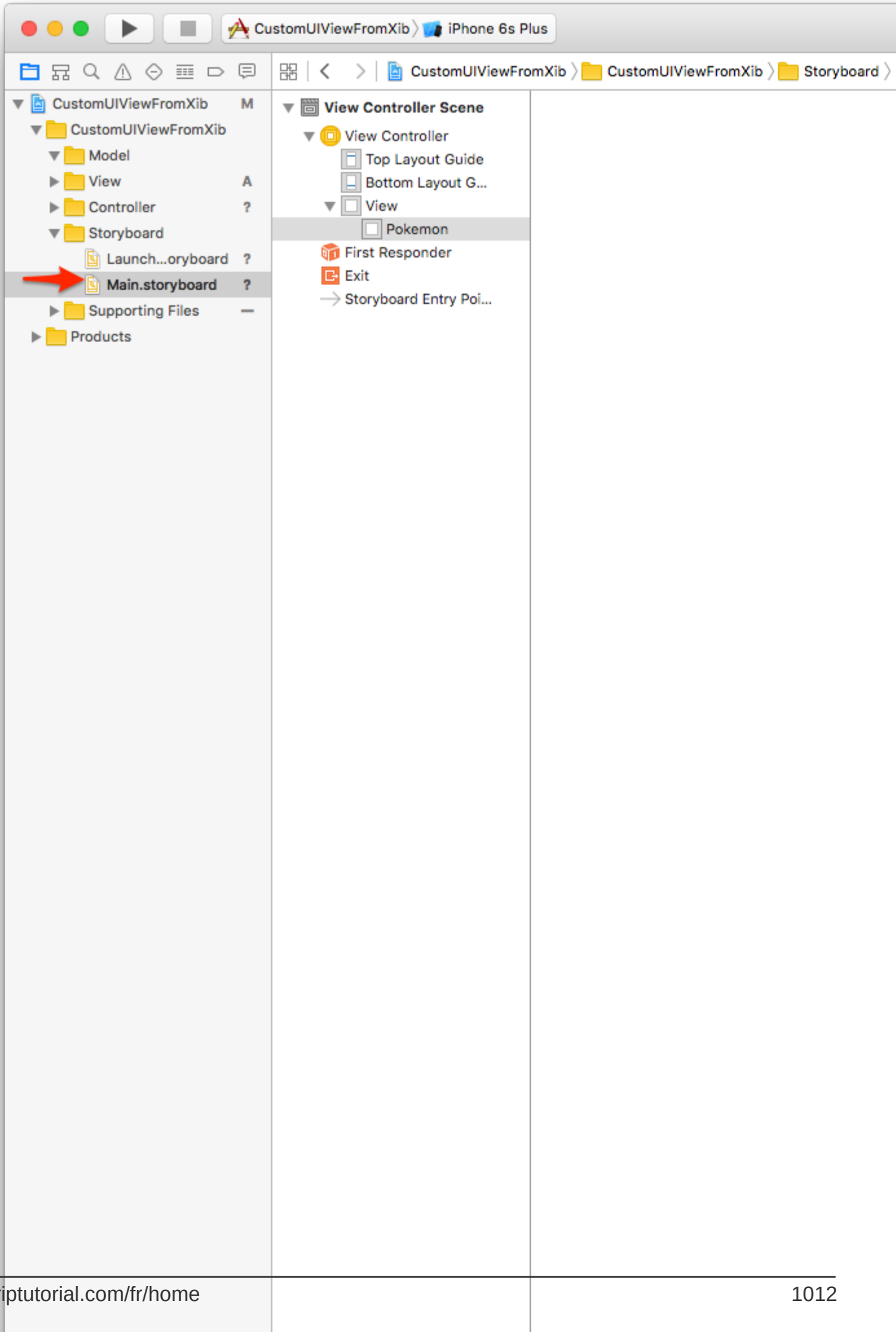

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window
CustomUIViewFromXib iPhone 6s Plus
CustomUIViewFromXib CustomUIViewFromXib View Pokemon.xib
CustomUIViewFromXib M
CustomUIViewFromXib
Model
View
Pokemon.xib A
Pokemon.swift A
Controller ?
Storyboard ?
Supporting Files
AppDel...ate.swift ?
Assets.xcassets M
Info.plist ?
Products
Placeholders
File's Owner
First Responder
View
Image View
Switch
Constraints

avant le nom de la classe):

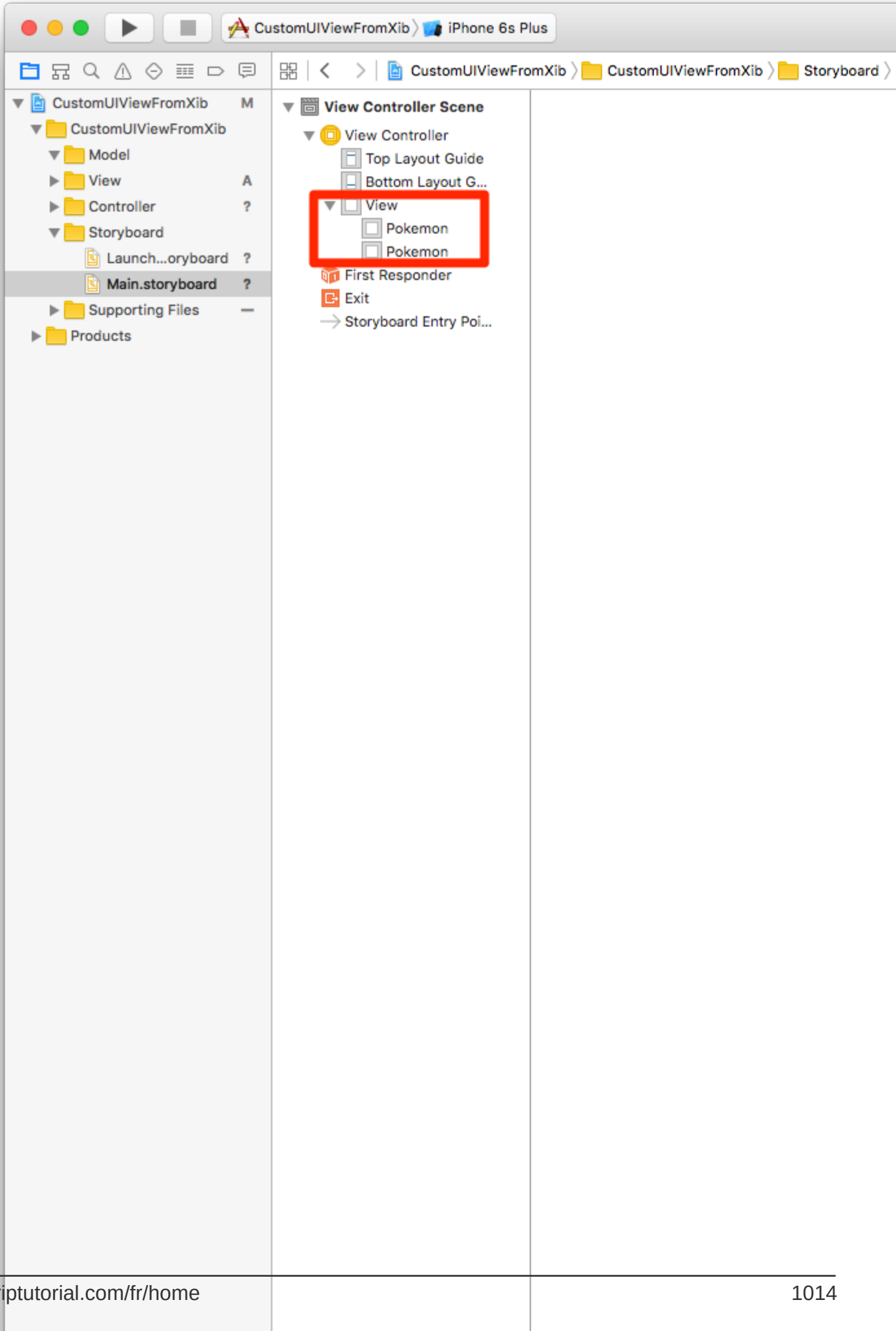
```
@IBDesignable class Pokemon: UIView {

    // MARK: - Properties

    @IBOutlet weak var imageView: UIImageView!

    @IBInspectable var image: UIImage? {
        get {
            return imageView.image
        }
        set(image) {
            imageView.image = image
        }
    }

    // MARK: - Initializers
    ...
```

Utiliser vos vues personnalisées

Entré dans votre fichier de scénario principal, faites glisser un UIView dans celui-ci.
Redimensionnez la vue pour, disons 200x200. Centraliser.
Accédez à l'inspecteur d'identité (en haut à droite) et définissez la classe sur 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...

Donnez-lui une taille différente, disons 150x150.
Choisissez une autre image Pokemon, observez:

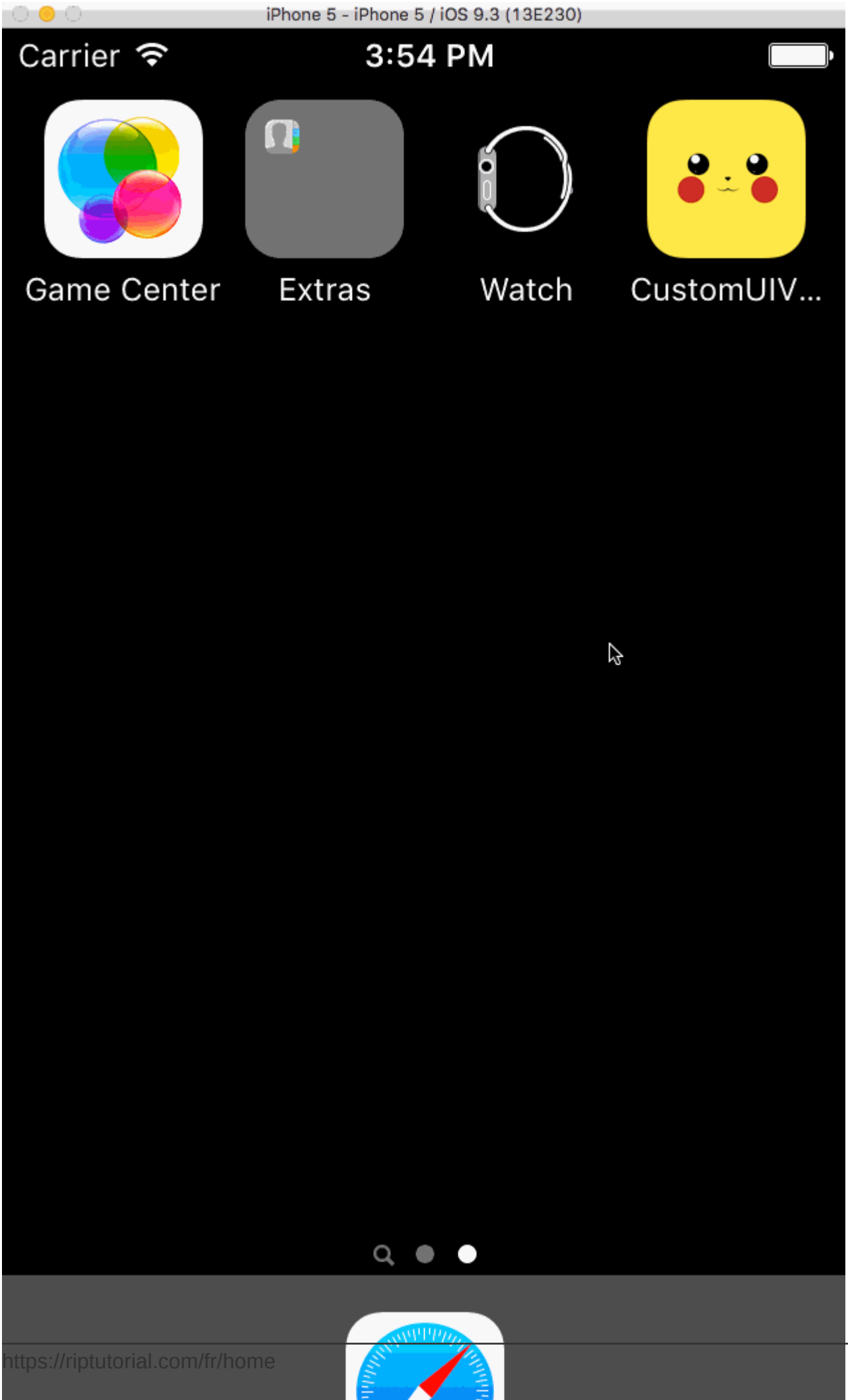

CustomUIViewFromXib
iPhone 6s Plus
CustomUIViewFromXib
CustomUIViewFromXib
Storyboard
CustomUIViewFromXib
M
CustomUIViewFromXib
Model
View
A
Controller
?
Storyboard
Launch...oryboard
?
Main.storyboard
?
Supporting Files
Products
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout G...
View
Pokemon
Pokemon
First Responder
Exit
Storyboard Entry Poi...

Le bouton permettra aux Pokemons d'être activés / désactivés.

Créez un `IBAction` du bouton Switch à la classe Pokemon.swift.
Appelez l'action quelque chose comme `switchTapped` .
Ajoutez-lui le code suivant:

```
// MARK: - Actions

@IBAction func switchTapped(sender: UISwitch) {
    imageView.alpha = sender.on ? 1.0 : 0.2
}

// MARK: - Initializers
...
```

Résultat final:


iPhone 5 - iPhone 5 / iOS 9.3 (13E230)
Carrier
3:54 PM
Game Center
Extras
Watch
CustomUIV...

Vous pouvez désormais créer des vues personnalisées complexes et les réutiliser partout où vous le souhaitez.
Cela augmentera la productivité tout en isolant le code dans des éléments d'interface utilisateur autonomes.

Le projet final peut être cloné dans Github.
(**Mis à jour pour Swift 3.1**)

Comment créer UIView réutilisable personnalisé à l'aide de XIB

L'exemple suivant montre les étapes impliquées dans l'initialisation d'une vue depuis XIB.

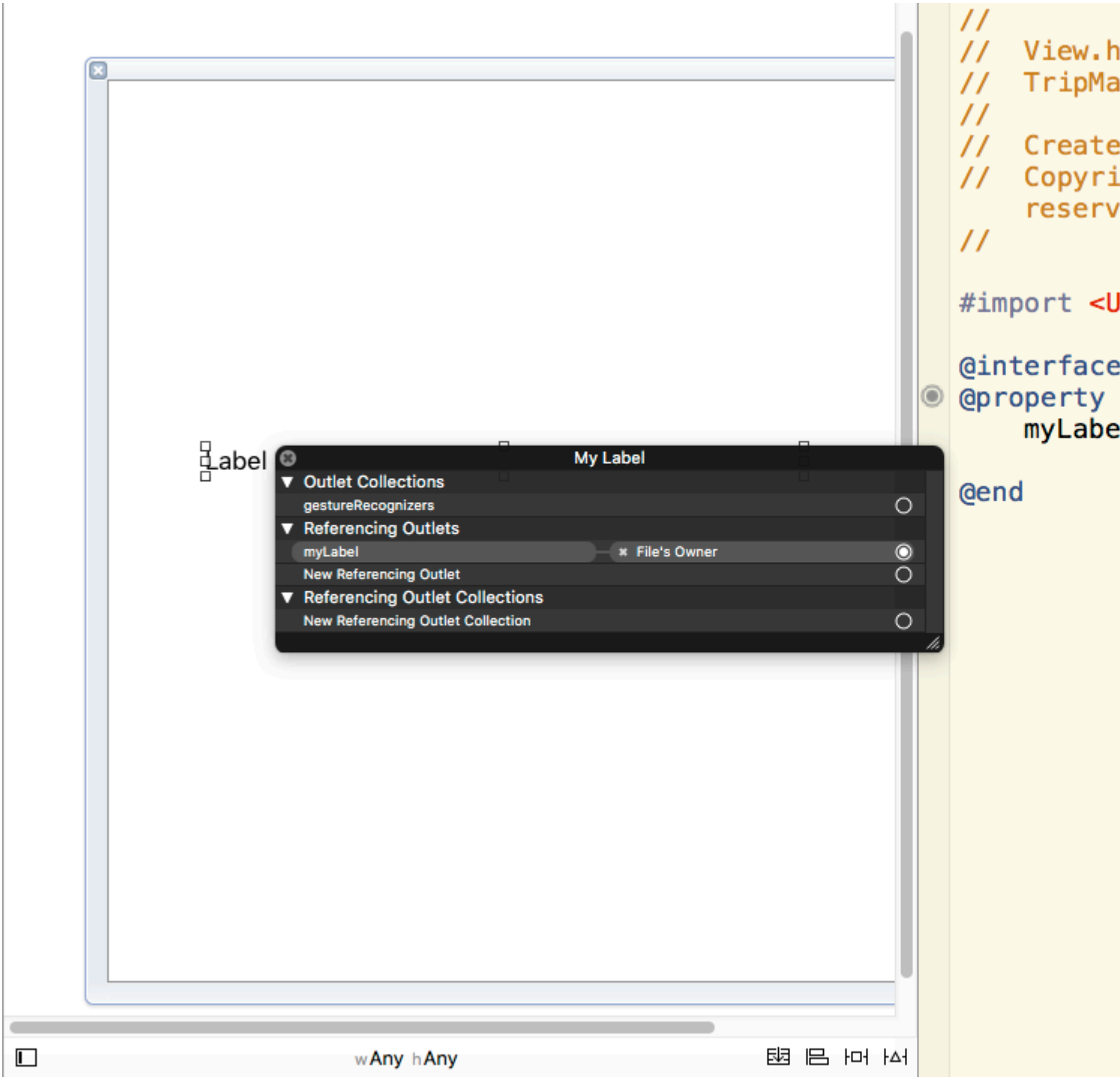
Ce n'est pas une opération complexe mais des étapes précises doivent être suivies pour le faire correctement dès le départ, en évitant les exceptions.

Comment fonctionne loadNibNamed

Les principales étapes sont:

1. Créer XIB
2. Créer la classe .h et .m
3. Définir les points de vente en .h
4. Connecter des prises entre .h et XIB

Voir la capture d'écran ci-jointe:



5. Appelez loadNibNamed dans la fonction initWithCoder du fichier .m. Ceci est nécessaire pour vous assurer que vous pouvez directement placer l'objet UIView dans le fichier storyboard / Parent UIView XIB et le définir comme vue personnalisée. Aucun autre code d'initialisation n'est nécessaire une fois que vous avez chargé le storyboard / parent XIB. Votre vue personnalisée peut être ajoutée à d'autres vues, tout comme les autres objets de vue Objective C intégrés donnés dans XCode.

Lire UIViews personnalisées à partir de fichiers XIB en ligne:
https://riptutorial.com/fr/ios/topic/1362/uiviews-personnalisees-a-partir-de-fichiers-xib

Chapitre 201: UIWebView

Remarques

Fonctions déléguées UIWebView: -

Décélérations Objective-C

```
- (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

Créer une instance UIWebView

Rapide

```
let webview = UIWebView(frame: CGRect(x: 0, y: 0, width: 320, height: 480))
```

Objectif 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;
```

Faire une demande d'URL

Charger le contenu dans WebView à partir de l' `url`

Rapide

```
webview.loadRequest(NSURLRequest(URL: NSURL(string: "http://www.google.com")!))
```

Objectif c

```
[webview loadRequest:[NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://www.google.com"]]];
```

Arrêtez de charger le contenu Web

La méthode `stopLoading()` arrête le processus de chargement en cours de la vue Web.

Rapide

```
webview.stopLoading()
```

Objectif c

```
[webview stopLoading];
```

Recharger le contenu Web actuel

Rapide

```
webview.reload()
```

Objectif c

```
[webview reload];
```

Détermination de la taille du contenu

Dans de nombreux cas, par exemple lors de l'utilisation de vues Web dans des cellules de vue tableau, il est important de déterminer la taille du contenu de la page HTML rendue. Après avoir chargé la page, cela peut être calculé dans la méthode déléguée `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);
}
```

Le code utilise une astuce supplémentaire pour régler rapidement la hauteur de la vue Web à 1 avant de mesurer la taille du raccord. Sinon, il rapporterait simplement la taille de l'image actuelle. Après la mesure, nous réglons immédiatement la hauteur à la hauteur réelle du contenu.

La source

Charger une chaîne HTML

Les vues Web sont utiles pour charger des chaînes HTML générées localement.

```
NSString *html = @"<!DOCTYPE html><html><body>Hello World</body></html>";
[webView loadHTMLString:html baseURL:nil];
```

Rapide

```
  let htmlString = "<h1>My First Heading</h1><p>My first paragraph.</p>"
  webView.loadHTMLString(htmlString, baseURL: nil)
```

Une URL de base locale peut être spécifiée. Ceci est utile pour référencer des images, des feuilles de style ou des scripts du lot d'applications:

```
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]]];
```

Dans ce cas, `style.css` est chargé localement à partir du répertoire de ressources de l'application. Bien sûr, il est également possible de spécifier une URL distante.

Charger JavaScript

Nous pouvons exécuter du code JavaScript personnalisé sur un `UIWebView` à l'aide de la méthode `stringByEvaluatingJavaScriptFromString()` . Cette méthode renvoie le résultat de l'exécution du script JavaScript transmis dans le paramètre de script, ou nul si le script échoue.

Rapide

Charger le script depuis la chaîne

```
webview.stringByEvaluatingJavaScriptFromString("alert('This is JavaScript!');")
```

Charger le script à partir du fichier local

```
//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)
        }
```

Objectif c

Charger le script depuis la chaîne

```
[webview stringByEvaluatingJavaScriptFromString:@"alert('This is JavaScript!');"];
```

Charger le script à partir du fichier local

```
//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];
```

Remarque La méthode `stringByEvaluatingJavaScriptFromString:` attend que l'évaluation de JavaScript soit terminée. Si vous chargez du contenu Web dont vous n'avez pas vérifié le code JavaScript, l'appel de cette méthode risque de bloquer votre application. La meilleure pratique consiste à adopter la classe `WKWebView` et à utiliser la `WKWebView evaluateJavaScript:completionHandler:` place. Mais `WKWebView` est disponible depuis iOS 8.0 et versions ultérieures.

Charger des fichiers de documents comme .pdf, .txt, .doc, etc.

Au lieu de pages Web, nous pouvons également charger les fichiers de document dans iOS WebView comme .pdf, .txt, .doc, etc. `loadData` méthode `loadData` est utilisée pour charger `NSData` dans Webview.

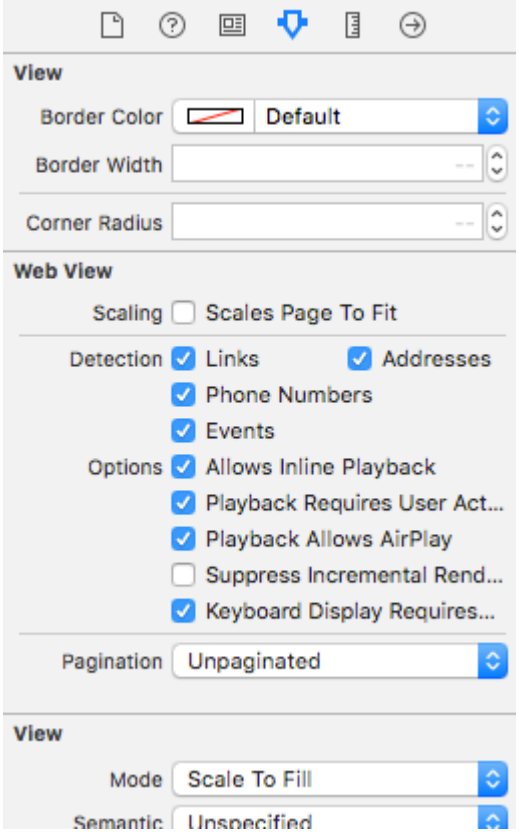
Rapide

```
//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())
```

Objectif 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]];
```

Créer des liens cliquables dans UIWebview



Dans vc.h

```
@interface vc : UIViewController<UIWebViewDelegate>
```

dans 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;

}
```

Charger le fichier HTML local dans webView

Tout d'abord, ajoutez le fichier HTML à votre projet (si vous êtes invité à choisir des options pour ajouter le fichier, sélectionnez *Copier les éléments si nécessaire*).

La ligne de code suivante charge le contenu du fichier HTML dans le WebView

```
webView.loadRequest(NSURLRequest(URL: NSURL(fileURLWithPath:
NSBundle.mainBundle().pathForResource("YOUR HTML FILE", ofType: "html")!})
```

- Si votre fichier HTML s'appelle index.html, remplacez votre fichier **HTML** par un **index**
- Vous pouvez utiliser ce code dans *viewDidLoad ()* ou *viewDidAppear ()* ou toute autre fonction

Lire UIWebView en ligne: https://riptutorial.com/fr/ios/topic/1452/uiwebview

Chapitre 202: Utilisation des séparateurs d'images

Introduction

Les éléments d'image permettent de gérer et d'organiser différents types d'éléments d'image dans notre application iOS à l'aide de Xcode.

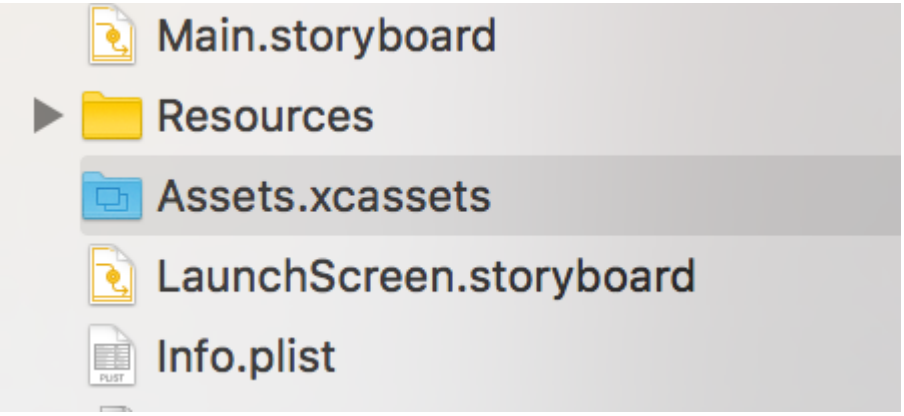
Ces ressources peuvent être des **icônes d'application, des images de lancement, des images utilisées dans l'application, des images en taille réelle, des images de taille aléatoire,** etc.

Examples

Icône de l'application utilisant des éléments d'image

Chaque fois que nous créons un nouveau projet dans Xcode pour notre nouvelle application, il nous donne différents dans les classes construites, des cibles, des tests, fichier plist, etc. De même , il nous donne aussi `Assets.xcassets` fichier, qui gère tous les actifs d'image dans notre projet.

Voici à quoi ressemble ce fichier dans le navigateur de fichiers:



Si on clique dessus, ça ressemblera à ceci:

AppIcon

Il suffit de **glisser-déposer l'** image correspondante sur chaque bloc carré vide. Chaque noir nous dira quelle taille doit avoir cette image, elle est écrite juste en dessous.
Après avoir glissé et déposé toutes les images dans tous les carrés, cela ressemblera à ceci:


AppIcon


AppIcon

car les iPads non rétiniens utiliseront deux `2x` images de lancement par mise à l'échelle

2 **12,9 "iPad Pro** : il n'y a pas de place pour cet iPad car cet iPad utilisera également `2x iPad` images `2x iPad` en les redimensionnant

3 **Retina HD 5.5 ": l'** iPad devrait avoir `1920x1080px` pour le portrait et `1080x1920px` pour le paysage, mais Xcode donnera des indications et l'image de lancement ne sera pas affichée sur ces appareils

4 **SplitView:** comme nous utilisons `LaunchImage Asset` au lieu de `LaunchScreen XIB` , notre application ne prend pas en charge `SplitView` sur iPad et iPhone 5.5 "

5 **Réinstallation:** si notre application est déjà installée sur l'appareil et que nous essayons de l'exécuter avec ces nouveaux éléments d'image de lancement, le périphérique n'affiche parfois pas les images de lancement lors du lancement de l'application. Dans ce cas, il suffit de supprimer l'application de l'appareil, nettoyer + construire le projet et l'exécuter, il affichera de nouvelles images de lancement

Lire Utilisation des séparateurs d'images en ligne:
https://riptutorial.com/fr/ios/topic/10087/utilisation-des-separateurs-d-images

Chapitre 203: UUID (Universally Unique Identifier)

Remarques

Pour enregistrer l'UUID, nous pouvons utiliser SSKeychainUtility . Vous trouverez un exemple sur la page Github

Examples

Générer UUID

UUID aléatoire

Rapide

```
func randomUUID() -> NSString{
    return NSUUID.UUID().UUIDString()
}
```

Objectif 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;
}
```

Identifiant du fournisseur

iOS 6

En une seule ligne, nous pouvons obtenir un UUID comme ci-dessous:

Rapide

```
let UDIDString = UIDevice.currentDevice().identifierForVendor?.UUIDString
```

Objectif c

```
NSString *UDIDString = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
```

`identifierForVendor` est un identifiant unique qui reste le même pour toutes les applications d'un fournisseur unique sur un seul périphérique, à moins que toutes les applications du fournisseur soient supprimées de cet appareil. Voir la documentation d'Apple sur le moment où cet `UUID` change.

IFA d'Apple contre IFV (identifiant Apple pour les annonceurs et identifiant pour les fournisseurs)

- Vous pouvez utiliser l'IFA pour mesurer les clics sur les annonces et l'IFV pour mesurer les installations d'applications.
- IFA dispose de mécanismes de confidentialité intégrés qui le rendent parfait pour la publicité. En revanche, l'IFV est destiné aux développeurs pour une utilisation interne afin de mesurer les utilisateurs qui installent leurs applications.

SI UN

- La classe ASIdentifierManager fournit
    - **advertisingIdentifier: UUID** : Chaîne alphanumérique unique à chaque périphérique, utilisée uniquement pour diffuser des publicités.
    - **isAdvertisingTrackingEnabled** : valeur booléenne indiquant si l'utilisateur a un suivi des annonces limité.

IFV

- La classe ASIdentifierManager fournit
    - **identificateurForVendor: UUID** : chaîne alphanumérique qui identifie de manière unique un périphérique auprès du fournisseur de l'application.

Trouvez votre appareil IFA et IFV ici .

Créer une chaîne UUID pour les appareils iOS

Ici, nous pouvons créer une `UUID String` avec une ligne.

Représente les chaînes UUID, qui peuvent être utilisées pour identifier de manière unique des types, des interfaces et d'autres éléments.

Swift 3.0

```
print(UUID().uuidString)
```

C'est très utile pour identifier plusieurs périphériques avec un identifiant unique.

Lire UUID (Universally Unique Identifier) en ligne: https://riptutorial.com/fr/ios/topic/3629/uuid--universally-unique-identifier-

Chapitre 204: Valeur clé Codage-Valeur Valeur Observation

Remarques

KVC : - Codage de la valeur clé

Normalement, les variables d'instance sont accessibles via des propriétés ou des accesseurs, mais KVC offre un autre moyen d'accéder aux variables sous forme de chaînes. De cette façon, votre classe agit comme un dictionnaire et le nom de votre propriété, par exemple «age», devient clé et la valeur que la propriété contient devient une valeur pour cette clé.

```
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 : - Observateur de la valeur-clé

Le mécanisme par lequel les objets sont notifiés en cas de changement de propriété est appelé KVO. Ex.: Notification clavier

> Par exemple, l'objet personne souhaite obtenir une notification lorsque la propriété accountBalance est modifiée dans l'objet BankAccount. Pour ce faire, Object Person doit s'enregistrer en tant qu'observateur de la propriété accountBalance de BankAccount en envoyant un addObserver: forKeyPath: options: context: message.

Examples

Utilisation du contexte pour l'observation KVO

```
-(void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object
change:(NSDictionary<NSString *,id> *)change context:(void *)context
```

Le contexte est important si vous expédiez votre classe pour que d'autres puissent l'utiliser. Le contexte permet à votre observateur de classe de vérifier que vous êtes son observateur appelé.

Le problème de ne pas dépasser un observateur est que si quelqu'un sous-classe votre classe et enregistre un observateur pour le même objet, la même clé et qu'il ne passe pas de contexte, alors l'observateur de super-classe peut être appelé plusieurs fois.

Une variable unique et interne pour votre utilisation est un bon contexte.

Pour plus d'informations.

importance et bon contexte

Observation d'une propriété d'une sous-classe NSObject

La plupart des fonctionnalités KVO et KVC sont déjà implémentées par défaut sur toutes les sous-classes `NSObject` .

Pour commencer à observer une propriété nommée `firstName` d'un objet nommé `personObject` faites-le dans la classe d'observation:

```
[personObject addObserver:self
               forKeyPath:@"firstName"
                  options:NSKeyValueObservingOptionNew
                  context:nil];
```

L'objet que l' `self` dans le code ci - dessus fait référence à recevra alors un `observeValueForKeyPath:ofObject:change:context:` un message chaque fois que les changements de chemin clés observées.

```
- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary<NSString *,id> *)change
                       context:(void *)context
{
    NSLog(@"new value of %@ is: %@", keyPath, change[NSKeyValueChangeNewKey]);
}
```

"Key path" est un terme KVC. `NSObject` sous-classes `NSObject` implémentent la fonctionnalité KVC par défaut.

Une variable d'instance nommée `_firstName` sera accessible par le chemin d'accès de la clé `@"firstName"` .

Une méthode de lecture du nom `firstName` sera appelé lors de l' accès au `@"firstName"` chemin de clé, quelle que soit l'existence d' une `_firstName` variable d'instance ou `setFirstName` méthode setter.

Lire Valeur clé Codage-Valeur Valeur Observation en ligne:
https://riptutorial.com/fr/ios/topic/3493/valeur-cle-codage-valeur-valeur-observation

Chapitre 205: Vérification de la connectivité réseau

Remarques

Le code source de `Reachability.h` et `Reachability.m` est disponible sur le site de documentation des développeurs d'Apple.

Mises en garde

Contrairement à d'autres plates-formes, Apple doit encore fournir un ensemble standard d'API pour déterminer l'état du réseau d'un périphérique iOS et proposer uniquement ces exemples de code liés ci-dessus. Le fichier source change au fil du temps, mais une fois importé dans un projet d'application, il est rarement mis à jour par les développeurs.

Pour cette raison, la plupart des développeurs d'applications ont tendance à utiliser l'une des nombreuses bibliothèques github / Cocoapod pour leur accessibilité.

Apple recommande également, pour les demandes faites à vous sur ordre de l'utilisateur, toujours tenter une connexion d' *abord,* avant d' utiliser joignabilité / SCNetworkReachability pour diagnostiquer l'échec ou d'attendre la connexion pour revenir .

Examples

Création d'un écouteur d'accessibilité

La classe d' accessibilité d'Apple vérifie périodiquement l'état du réseau et avertit les observateurs des modifications.

```
Reachability *internetReachability = [Reachability reachabilityForInternetConnection];
[internetReachability startNotifier];
```

Ajouter un observateur aux modifications du réseau

`Reachability` `NSNotification` utilise les messages `NSNotification` pour alerter les observateurs lorsque l'état du réseau a changé. Votre classe devra devenir un observateur.

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(reachabilityChanged:) name:kReachabilityChangedNotification object:nil];
```

Ailleurs dans votre classe, implémentez la signature de la méthode

```
- (void) reachabilityChanged:(NSNotification *)note {
    //code which reacts to network changes
```

```
}
```

Alerte lorsque le réseau devient indisponible

```
- (void)reachabilityChanged:(NSNotification *)note {
    Reachability* reachability = [note object];
    NetworkStatus netStatus = [reachability currentReachabilityStatus];

    if (netStatus == NotReachable) {
        NSLog(@"Network unavailable");
    }
}
```

Alerte lorsque la connexion devient un réseau WIFI ou cellulaire

```
- (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;
    }
}
```

Vérifier si est connecté au réseau

Rapide

```
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
}
```

Objectif c:

nous pouvons vérifier la connectivité du réseau en quelques lignes de code comme suit:

```
-(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;
    }
}
```

Lire Vérification de la connectivité réseau en ligne:
https://riptutorial.com/fr/ios/topic/704/verification-de-la-connectivite-reseau

Chapitre 206: Vérification de la version iOS

Examples

iOS 8 et versions ultérieures

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)
}
```

Comparer les versions

```
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;
}
```

Objectif 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 et ultérieur

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

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

Version iOS du périphérique

Cela donnera la version actuelle du système.

Objectif c

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

Rapide

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

Swift 3

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

Lire Vérification de la version iOS en ligne: https://riptutorial.com/fr/ios/topic/2194/verification-de-la-version-ios

Chapitre 207: Voie rapide

Examples

outils fastlane

fastlane est un outil d'automatisation de build open source pour Android et iOS pour les développeurs. Cela réduit votre temps de génération de build. C'est un outil de ligne de commande qui utilise Ruby . Vous avez donc besoin de Ruby sur votre ordinateur. La plupart des Mac ont déjà installé Ruby par défaut.

Installer fastlane

1. Ouvrez un terminal.
2. Exécuter `sudo gem install fastlane --verbose`
3. Si vous n'avez pas encore installé les outils de ligne de commande Xcode, exécutez `xcode-select --install` pour les installer.
4. Maintenant, `cd` dans votre dossier de projet (tapez `cd` [avec l'espace à la fin] et faites glisser votre dossier de projet dans le terminal)
5. Exécutez `fastlane init` pour obtenir une configuration rapide.
6. Vous pouvez maintenant utiliser tous les outils Fastlane:

Outils iOS

- livrer : Télécharger des captures d'écran, des métadonnées et votre application sur l'App Store
- instantané : automatisez la prise de captures d'écran localisées de votre application iOS sur chaque appareil
- frameit : placez rapidement vos captures d'écran dans les bons cadres
- pem : générer et renouveler automatiquement vos profils de notification push
- soupir : Parce que vous préférez passer votre temps à construire des choses que de lutter contre l'approvisionnement
- Produire : Créez de nouvelles applications iOS sur iTunes Connect et Dev Portal en utilisant la ligne de commande
- cert : créer et maintenir automatiquement des certificats de signature de code iOS
- gym : Construire vos applications iOS n'a jamais été aussi facile
- match : synchronisez facilement vos certificats et profils dans votre équipe en utilisant Git
- scan : le moyen le plus simple d'exécuter des tests pour vos applications iOS et Mac
- vaisseau spatial : bibliothèque Ruby pour accéder au Apple Dev Center et iTunes Connect

Outils iOS TestFlight

- pilot : le meilleur moyen de gérer vos testeurs et vos builds TestFlight à partir de votre

terminal
- embarquement : le moyen le plus simple d'inviter vos testeurs bêta TestFlight

Outils Android

- offre : téléchargez votre application Android et ses métadonnées sur Google Play
- screengrab : automatisez la prise de captures d'écran localisées de votre application Android sur chaque appareil

Lire Voie rapide en ligne: https://riptutorial.com/fr/ios/topic/3574/voie-rapide

Chapitre 208: WCSessionDelegate

Introduction

`WCSessionDelegate` fonctionne avec watch OS2 + en utilisant WatchConnectivity. var watchSession: WCSession? func startWatchSession () {if (WCSession.isSupported ()) {watchSession = WCSession.default () watchSession! .delegate = self watchSession! .activate ()} Implémenter la méthode requise: - `didReceiveApplicationContext`

Examples

Contrôleur de kit de montre (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")
    }
```

Lire WCSessionDelegate en ligne: https://riptutorial.com/fr/ios/topic/8289/wcsessiondelegate

Chapitre 209: WKWebView

Introduction

WKWebView est la pièce maîtresse de l'API WebKit moderne introduite dans iOS 8 et OS X Yosemite. Il remplace UIWebView dans UIKit et WebView dans AppKit, offrant une API cohérente sur les deux plates-formes.

Bénéficiant d'un défilement rapide de 60 images par seconde, de gestes intégrés, d'une communication rationalisée entre l'application et la page Web et du même moteur JavaScript que Safari, WKWebView est l'une des annonces les plus importantes de WWDC 2014.

Examples

Créer un navigateur Web simple

```
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)
```

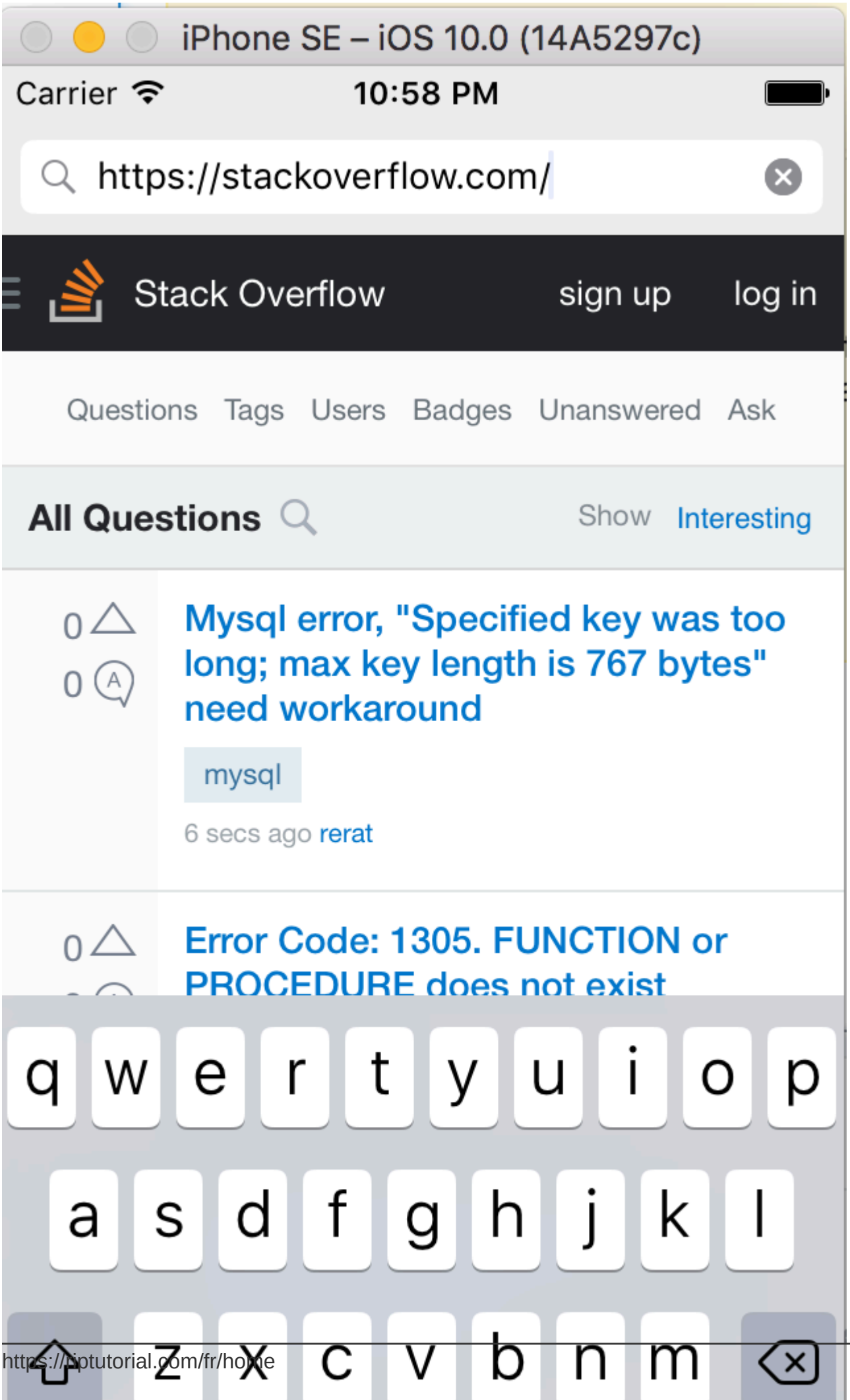
```
        //Setup which views will be constrained.

        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.
    }
}
```

Afficher le bouton GO personnalisé dans le clavier:


iPhone SE – iOS 10.0 (14A5297c)
Carrier
10:58 PM
https://stackoverflow.com/
Stack Overflow
sign up
log in
Questions
Tags
Users
Badges
Unanswered
Ask
All Questions
Show
Interesting
0
0
Mysql error, "Specified key was too long; max key length is 767 bytes" need workaround
mysql
6 secs ago rerat
0
Error Code: 1305. FUNCTION or PROCEDURE does not exist
q w e r t y u i o p
a s d f g h j k l
z x c v b n m

`.AtDocumentEnd`

Envoyer des messages à partir de JavaScript et les gérer du côté natif

Les messages peuvent être envoyés à partir de JavaScript en utilisant le code suivant

```
window.webkit.messageHandlers.{NAME}.postMessage()
```

Voici comment créer un gestionnaire de messages de script pour gérer les messages:

```
class NotificationScriptMessageHandler: NSObject, WKScriptMessageHandler {
    func userContentController(userContentController: WKUserContentController,
didReceiveScriptMessage message: WKScriptMessage!) {
        if message.name == "{NAME}" {
            // to be sure of handling the correct message
            print(message.body)
        }
    }
}
```

Voici comment configurer le gestionnaire de messages de script dans WKWebView:

```
let configuration = WKWebViewConfiguration()
let userContentController = WKUserContentController()
let handler = NotificationScriptMessageHandler()
userContentController.addScriptMessageHandler(handler, name: "{NAME}")
configuration.userContentController = userContentController
let webView = WKWebView(frame: self.view.bounds, configuration: configuration)
```

> **REMARQUE:** ajouter le même gestionnaire `"{NAME}"` avec `addScriptMessageHandler:name:` plus d'une fois, entraîne `NSInvalidArgumentException` exception `NSInvalidArgumentException` .

Lire WKWebView en ligne: https://riptutorial.com/fr/ios/topic/3602/wkwebview

Chapitre 210: Xcode Build & Archive à partir de la ligne de commande

Syntaxe

- `xcodebuild [-project name.xcodeproj] -scheme schemename [[-destination destinationspecifier] ...] [-destination-timeout value] [-configuration configurationname] [-sdk [sdkfullpath | sdkname]] [action ...] [buildsetting=value ...] [-userdefault=value ...]`

Paramètres

Option	La description
-projet	Générez le nom du projet.xcodeproj.
-schème	Obligatoire si vous construisez un espace de travail.
-destination	Utiliser le périphérique de destination
-configuration	Utiliser la configuration de construction
-sdk	SDK spécifié

Remarques

Exécutez `xcodebuild` partir du répertoire contenant votre projet pour créer un projet Xcode. Pour créer un espace de travail Xcode, vous devez passer les options **-workspace** et **-scheme** pour définir la génération. Les paramètres du schéma contrôlent les cibles créées et leur mode de construction, bien que vous puissiez passer d'autres options à xcodebuild pour remplacer certains paramètres du schéma.

Examples

Construire et archiver

Construire:

```
xcodebuild -exportArchive -exportFormat ipa \
    -archivePath "/Users/username/Desktop/MyiOSApp.xcarchive" \
    -exportPath "/Users/username/Desktop/MyiOSApp.ipa" \
    -exportProvisioningProfile "MyCompany Distribution Profile"
```

Archiver:

```
xcodebuild -project <ProjectName.xcodeproj>
    -scheme <ProjectName>
    -sdk iphonesimulator
    -configuration Debug
    -destination "platform=iOS Simulator,name=<Device>,OS=9.3"
    clean build
```

Lire Xcode Build & Archive à partir de la ligne de commande en ligne: https://riptutorial.com/fr/ios/topic/5027/xcode-build--amp--archive-a-partir-de-la-ligne-de-commande

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec iOS	Ali Beadle, Allan Burleson, Anand Nimje, Anatoliy, Ashutosh Dave, Bhadresh Kathiriya, bjtitus, Blachshma, bmike, Charlie H, Cin316, Community, Dair, dan, deyanm , Efraim Weiss, Erik Godard, FelixSFD, Fogmeister, Hudson Taylor, Irfan, J F, Jack Ngai, James, Josh Brown, jrf, Kampai, Kevin, Losiowaty, M. Galban, Maddyゾゾ, Matthew Cawley, Md. Ibrahim Hassan, Midhun MP, Miguel Cabezas, Muhammad Zohaib Ehsan, Pro Q, PSN, RamenChef, Sam Fischer, Seyyed Parsa Neshaei, shim, Skeleton Bow, Stephen Leppik, Steve Moser, Suragch, SuzGupta, The_Curry_Man, ThrowingSpoon, Undo, user3480295, user6939352, Vignan
2	3D Touch	4444, Harshal Bhavsar, LinusGeffarth, Md. Ibrahim Hassan, Onur Tuna, Stephen Leppik, tobeiosdeveloper
3	Accessibilité	Harshal Bhavsar, Justin, Ruby, Stephen Leppik, Zev Eisenberg
4	Achat in-app	Cyril Ivar Garcia, Martin, rigdonmr, WMios
5	AFNetworking	4444, Mayank Patel, OhadM, Ruby
6	AirDrop	FelixSFD, Md. Ibrahim Hassan
7	AJOUT D'UN EN-TÊTE DE PONT SWIFT	yogesh wadhwa
8	Alamofire	Alex Koshy, Josh Caswell, Sour LeangChhean, yogesh wadhwa
9	API de reconnaissance vocale iOS 10	rohit90, Stephen Leppik
10	API Google Adresses 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 (comptage	4444, Irfan, John Militer, Ketan P, Tricertops

	automatique des références)	
14	Architecture MVP	Oleh Zayats
15	attributTexte dans UILabel	vp2698
16	AVPlayer et AVPlayerViewController	Bonnie, Chirag Desai, Gazi Alankus, Harshal Bhavsar, Konda Yadav, Stephen Leppik
17	AVSpeechSynthesizer	Ali Beadle, Bhumit Mehta, Harshal Bhavsar, Midhun MP, Stephen Leppik
18	AWS SDK	OhadM
19	Barre de navigation	Md. Ibrahim Hassan, Mehul Chuahan
20	Bloc	4444, animuson, Joshua, Mehul Chuahan, Ruby, Tamarous, user459460
21	Blocs de chaîne dans une file d'attente (avec MKBlockQueue)	StackUnderflow
22	CAAnimation	Bhavin Ramani, James P, Mr. Xcoder, Narendra Pandey, Rahul, Rob, Undo
23	Cache les images en ligne	Md. Ibrahim Hassan
24	Cadre de contacts	Md. Ibrahim Hassan, Seyyed Parsa Neshaei
25	CAGradientLayer	Bhavin Ramani, Harshal Bhavsar, Sam Fischer, Stephen Leppik, Undo
26	CALayer	Alistra, Dunja Lalic, HariKrishnan.P, Harshal Bhavsar, ignotusverum, iOS BadBoy, Kamil Harasimowicz, Luiz Henrique Guimaraes, Stephen Leppik, Suragch, Viktor Simkó, william205
27	CAShapeLayer	Filip Radelic, HariKrishnan.P, Harshal Bhavsar, Narendra Pandey, Stephen Leppik
28	Catégories	Faran Ghani, simple_code
29	Changer la couleur de la barre d'état	Alex Rouse, danshevluk, Harshal Bhavsar, Mr. Xcoder, shim, Stephen Leppik, Steve Moser, william205, WMios
30	Charger des images asynchrones	J.Paravicini
31	Classements GameCenter	4444, Cyril Ivar Garcia, Harshal Bhavsar, Stephen Leppik

32	Classes de taille et adaptivité	Tim
33	Clavier personnalisé	Md. Ibrahim Hassan
34	CLLocation	amar, Duly Kinsky, FelixSFD, Siddharth Sunil, Sujania, That lazy iOS Guy , void, Zee
35	CloudKit	Seyyed Parsa Neshaei
36	Codable	Ashish Kakkad
37	Concurrence	Doc, Fonix, Juan Campa, Kevin DiTraglia, Tien
38	Configuration iOS de Carthage	Md. Ibrahim Hassan
39	Configurer les balises avec CoreBluetooth	Beto Caldas
40	Content Hugging / Compression de contenu dans Autolayout	Mehul Chuahan
41	Convertir le HTML en chaîne NSAttributed et vice versa	Md. Ibrahim Hassan
42	Convertir NSAttributedString en UIImage	Md. Ibrahim Hassan
43	Core SpotLight dans iOS	Md. Ibrahim Hassan
44	Couper un UIImage en cercle	Md. Ibrahim Hassan
45	Création PDF dans iOS	Mansi Panchal, Narendra Pandey
46	Créer un fichier .ipa à télécharger sur AppStore avec Applicationloader	Anuj Joshi
47	Créer un ID d'application	yogesh wadhwa
48	Créer une infrastructure personnalisée dans iOS	Saeed-rz
49	Créer une vidéo à partir d'images	Tiko

50	CTCallCenter	MANI, Md. Ibrahim Hassan, OhadM
51	Débogage des pannes	NobodyNada
52	Deep Linking dans iOS	bryanjclark, Dunja Lalic, FelixSFD, sanman
53	Définir l'arrière-plan de la vue	Adriana Carelli, Andreas, Bhadresh Kathiriya, Harshal Bhavsar, Md. Ibrahim Hassan, user459460
54	Délégués de multidiffusion	Rahul
55	Demande d'évaluation / révision	Abhijit
56	Détection de visage avec CoreImage / OpenCV	Md. Ibrahim Hassan
57	DispatchGroup	Brandon, Fonix
58	Domaine	subv3rsion
59	Données de base	Ankit chauhan, Md. Ibrahim Hassan
60	Dynamique UIKit	beyowulf, Mark Stewart, Md. Ibrahim Hassan
61	Emplacement central	Harshal Bhavsar, Mayuri R Talaviya, Mehul Chuahan, mtso, quant24, Stephen Leppik, sushant jagtap, william205
62	EventKit	Seyyed Parsa Neshaei
63	Extension pour notification Push enrichie - iOS 10.	Oleh Zayats
64	FacebookSDK	Brian, Harshal Bhavsar, Irfan, Mehul Chuahan, OhadM, Ravi Prakash Verma, Stephen Leppik
65	Fichier texte de base I / O	Idan
66	FileHandle	Nikhlesh Bagdiya
67	Filtres CoreImage	Md. Ibrahim Hassan
68	Framework XCTest - Tests unitaires	D4ttatraya, dasdom, Jan ATAC, Josh Brown, msohng, Raphael Silva, Seyyed Parsa Neshaei, Tarun Seera
69	GameplayKit	BennX, Seyyed Parsa Neshaei
70	GCD (Expédition Central Grand)	Andrea Antonioni, DS Dharma, Fonix, Md. Ibrahim Hassan, skyline75489

71	Gérer plusieurs environnements en utilisant la macro	Tien
72	Gestion des schémas d'URL	azimov, Brian, Dunja Lalic, Harshal Bhavsar, James P, Stephen Leppik
73	Gestion du clavier	Alexander Tkachenko, Greg, Harshal Bhavsar, Mr. Xcoder, Richard Ash, Shog9, slxl, Stephen Leppik, Steve Moser, Suragch, V1P3R, william205, WMios
74	Graphique (Coreplot)	MarmiK, Md. Ibrahim Hassan
75	Graphiques de base	Dunja Lalic, Josh Caswell, Seyyed Parsa Neshaei, Sunil Sharma, Unheilig
76	Guide pour choisir les meilleurs modèles d'architecture iOS	Phani Sai
77	Healthkit	Md. Ibrahim Hassan
78	iBeacon	amar, Arefly, Harshal Bhavsar, Stephen Leppik
79	IBOutlets	Fabio, SharkbaitWhohaha
80	Idiomes d'initialisation	Jano
81	Instantané de UIView	Bright Future, Darshit Shah, Md. Ibrahim Hassan, SpaceDog
82	Intégration SqlCipher	Nirav
83	Interopérabilité Swift et Objective-C	Harshal Bhavsar, njuri, Stephen Leppik
84	iOS - Implémentation de XMPP avec le framework Robbie Hanson	Saheb Roy
85	iOS TTS	Ali Abbas, Stephen Leppik
86	Liens universels	Harshal Bhavsar, Irfan, satheeshwaran, Stephen Leppik, Vineet Choudhary
87	Localisation	4444, animuson, Joshua, Ruby, WMios
88	Messagerie FCM dans Swift	Saeed-rz
89	Méthodes personnalisées	Kamil Harasimowicz

	de sélection de UITableViewCells	
90	Mise à jour dynamique d'un UIStackView	Harshal Bhavsar, Rahul, Stephen Leppik
91	Mise en forme automatique UIScrollView	Aaron, Brandon, Shrikant K
92	Mise en page automatique	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
93	MKDistanceFormatter	Harshal Bhavsar, Md. Ibrahim Hassan, Stephen Leppik, Undo
94	MKMapView	Arnon Rodrigues, Brian, FelixSFD, Harshal Bhavsar, Kosuke Ogawa, Mahesh, Mehul Thakkar, Ortwin Gentz, Reinier Melian, Stephen Leppik
95	ModelPresentationStyles	Dishant Kapadiya
96	Modes d'arrière-plan	Seyyed Parsa Neshaei
97	Modes de fond et événements	Ashish Kakkad
98	Mouvement de base	Md. Ibrahim Hassan, RamenChef
99	MPMediaPickerDelegate	FelixSFD, George Lee
100	MPVolumeView	lostAtSeaJoshua
101	MVVM	JPetric
102	MyLayout	
103	Notifications enrichies	Koushik
104	Notifications 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
105	NSArray	Krunal, user5553647
106	NSAttributedString	Bhavin Ramani, Harshal Bhavsar, Jinhuan Li, Kirit Modi,

		Luiz Henrique Guimaraes, Mansi Panchal, Stephen Leppik, Tim, Tim Ebenezer, Undo
107	NSBundle	wdywayne
108	NSData	Felipe Cypriano, maxkonovalov, Seyyed Parsa Neshaei
109	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
110	NSHTTPCookieStorage	balagurubaran
111	NSInvocation	Md. Ibrahim Hassan
112	NSNotificationCenter	Alex Kallam, Alex Koshy, Anand Nimje, Bence Pattogato, Bright Future, Ichthyocentaurs, Jacopo Penzo, James P, Kirit Modi, Tarun Seera
113	NSPredicate	Brendon Roberto, Joshua, Mehul Chuahan
114	NSTimer	AJ9, James P, Maddyゾゾ, Samuel Teferra, tfrank377, That lazy iOS Guy , Undo, william205
115	NSURL	Adnan Aftab, ApolloSoftware, tharkay
116	NSURLConnection	byJeevan
117	NSURLSession	bluey31, dasdom, dgatwood, Duly Kinsky, Harshal Bhavsar, Narendra Pandey, Otávio, R P, sage444, Stephen Leppik
118	NSUserActivity	Samuel Spencer
119	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
120	Objective-C Objets associés	Noam
121	OpenGL	Fonix
122	Opérations étendues	midori
123	plist iOS	SNarula

124	Polices personnalisées	Alexi, Dima Deplov, Harshal Bhavsar, Maddyツ, njuri, Stephen Leppik, Tommie C.
125	Porte-clés	abjurato, avojak, Matthew Seaman, Mehul Chuahan
126	Processus de soumission d'applications	Nermin Sehic
127	Profil avec instruments	Vinod Kumar
128	Redimensionner UIImage	Rahul
129	Référence CGContext	4444, Narendra Pandey
130	Rendre les coins sélectifs UIView arrondis	Md. Ibrahim Hassan
131	Runtime en Objective-C	halil_g
132	Scanner de code QR	Bluewings, Efraim Weiss
133	Sécurité	D4ttatraya
134	Segues	Daniel Ormeño
135	Services Safari	Arnon Rodrigues, Harshal Bhavsar, Kilian Koeltzsch, Md. Ibrahim Hassan, Stephen Leppik
136	Signature de code	HaemEternal
137	Simulateur	Seyyed Parsa Neshaei
138	Simulateur construit	Durai Amuthan.H
139	Simulation de l'emplacement à l'aide de fichiers GPX iOS	Uma
140	SiriKit	Seyyed Parsa Neshaei
141	SLComposeViewController	Md. Ibrahim Hassan
142	StoreKit	askielboe
143	Storyboard	Harshal Bhavsar, Kirit Vaghela, Stephen Leppik, Tommie C.
144	Swift: Modifier le rootViewController dans AppDelegate pour présenter le flux principal	cleverbit

	ou de connexion / d'intégration	
145	SWRevealViewController	Reinier Melian, tharkay
146	Test de l'interface utilisateur	P. Pawluś
147	Texte UILabel souligné	Md. Ibrahim Hassan
148	Transmission de données entre les contrôleurs de vue	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
149	Transmission de données entre les contrôleurs de vue (avec MessageBox-Concept)	StackUnderflow
150	Tutoriel AirPrint sur iOS	Md. Ibrahim Hassan
151	Tweak de CydiaSubstrate	gkpln3
152	Type dynamique	Alvin Abia, H. M. Madrone, Harshal Bhavsar, James P, Stephen Leppik
153	UIAapparence	azimov, Harshal Bhavsar, Stephen Leppik, Undo
154	UIActivityViewController	Amandeep, Harshal Bhavsar, Stephen Leppik, Vivek Molkar
155	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
156	UIBarButtonItem	Ahmed Khalaf, Dunja Lalic, hgwhittle, Suragch, william205
157	UIBezierPath	Bean, Igor Bidiniuc, Suragch, Teja Nandamuri
158	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
159	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
160	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
161	UIControl - Gestion des événements avec des blocs	Brandon
162	UIDatePicker	Pavel Gatilov
163	UIDevice	Bhavin Ramani, FelixSFD, Md. Ibrahim Hassan, Mehul Chuahan, Nef10, pableiros, Ramkumar chintala
164	UIFeedbackGenerator	beyowulf
165	UIFont	Mr. Xcoder
166	UIGestureRecognizer	Adam Preble, dannyzlo, Dunja Lalic, Harshal Bhavsar, John Leonardo, Josh Caswell, Md. Ibrahim Hassan, Ruby , Stephen Leppik, Sujania, Suragch, Undo
167	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
168	UIImagePickerController	Brian, stonybrooklyn, william205
169	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
170	UIKit Dynamics avec UICollectionView	beyowulf
171	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
172	UILocalNotification	Bhumit Mehta, Brian, Byte1518, D4ttatraya, David, ElonChan, Harshal Bhavsar, hgwhittle, kamwysoc, KrishnaCA, rajesh sukumaran, Rex, Samuel Spencer, themathsrobot, tktsubota, william205, Wolverine, Xenon
173	UINavigationController	dasdom, Oleh Zayats, sage444, Suragch, william205, WMios
174	UIPageViewController	azimov, Bright Future, Harshal Bhavsar, Mayuri R Talaviya, Stephen Leppik, stonybrooklyn, Victor M
175	UIPheonix - framework d'interface utilisateur simple, flexible, dynamique et hautement évolutif	StackUnderflow

176	UIPickerView	FelixSFD, Hasintha Janka, MCMatan, Md. Ibrahim Hassan, Moritz, NinjaDeveloper
177	UIRefreshControl TableView	Md. Ibrahim Hassan, Mohammad Rana
178	UIScrollView	Bhavin Ramani, LinusGeffarth, maxkonovalov, Rex, sanman, Sujania, Sunil Sharma, Suragch, tharkay, torinpitchers
179	UIScrollView avec enfant StackView	mourodrigo
180	UISearchController	Harshal Bhavsar, Mehul Chuahan, mtso, Stephen Leppik, Tarvo Mäesepp
181	UISegmentedControl	Kamil Harasimowicz
182	UISlider	Andreas, Md. Ibrahim Hassan
183	UISplitViewController	Cerbrus, Koushik
184	UIStackView	Anuj Joshi, danshevluk, Harshal Bhavsar, Kof, Lior Pollak , Sally, sasquatch, Stephen Leppik, william205
185	UIStoryboard	Adriana Carelli, Mr. Xcoder, Vignan
186	UISwitch	Bhavin Ramani, FelixSFD, Md. Ibrahim Hassan, Mr. Xcoder, RamenChef, Sujay
187	UITabBarController	Alexi, Anand Nimje, Cristina, Mehul Chuahan, Quantm, Srinija
188	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
189	UITableViewCell	Rahul
190	UITableViewController	Aju
191	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
192	UITextField Délégué	Andreas, animuson, Md. Ibrahim Hassan, midori, Ruby
193	UITextField personnalisé	D4ttatraya
194	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
195	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
196	UIViewController	dasdom, Dunja Lalic, shim, Suragch, tassinari, william205
197	UIViews personnalisées à partir de fichiers XIB	backslash-f, Code.Warrior, Harshal Bhavsar, idocode, Nirav Bhatt, Sharpkits Innovations, Stephen Leppik
198	UIWebView	Allan Burleson, dchar4life80X, iOS BadBoy, J F, Julian135, KANGKANG, Kevin DiTraglia, maxkonovalov, Md. Ibrahim Hassan, Ortwin Gentz, Ramkumar chintala, Sunil Sharma
199	Utilisation des séparateurs d'images	D4ttatraya
200	UUID (Universally Unique Identifier)	Anand Nimje, FelixSFD, Harshal Bhavsar, James P, Mehul Chuahan, Rahul Vyas, Seyyed Parsa Neshaei, shim, Stephen Leppik, sushant jagtap
201	Valeur clé Codage-Valeur Valeur Observation	D4ttatraya, Harshal Bhavsar, Mehul Chuahan, Mihriban Minaz, Mithrandir, Muhammad Zohaib Ehsan, Pärserk, sanman
202	Vérification de la connectivité réseau	ajmccall, breakingobstacles, Mick MacCallum, pableiros, sushant jagtap
203	Vérification de la version iOS	Bhavin Ramani, byJeevan, James P, Joshua, njuri, Samuel Teferra, Sandy
204	Voie rapide	J F, KrauseFx, SM18, tharkay

205	WCSessionDelegate	pkc456
206	WKWebView	Brandon, byJeevan, Mahmoud Adam, Yevhen Dubinin
207	Xcode Build & Archive à partir de la ligne de commande	Kyle Decot, Shardul