Introduction à la programmation iPhone/iPad/iWatch/iTV,... (iOS/Objective C)

UV AMSE
Anthony Fleury - CERI SN (anthony.fleury@imt-lille-douai.fr)
IMT Lille Douai

Organisation du cours

- Cours sur développement mobiles : 2 plateformes - iOS et Android.

- Pour iOS :
  - Module de 20h : Cours + TD + TP
  - Programmation et test possible sur vos smartphone/tablettes
  - Un TP sur Apple Watch ?

Pré-requis et Objectifs du cours

- Pré-requis :
  - Vos cours de Java de l'an dernier (et d'algorithmique)
  - Cours d'objet
  - Langage C
- Introduction à un nouveau langage de programmation : **Objective C** !
- Les objectifs :
  - Vous présenter la programmation sur plateformes mobiles,
  - Présenter les spécificités de la plateforme Apple,
  - Vous présenter les bases de la programmation en Objective C,
  - Que vous soyez capables de réaliser une application basique sur iPhone/iPad et la déployer sur un device iOS.

Plan du cours

1. Introduction : la programmation sur mobile et la programmation sur plateforme Apple
2. Xcode, l'outil de développement
3. Premiers pas avec Objective C : classes et concepts de base !
4. Le SDK iOS : rapide présentation
5. Concepts et données dans les applications

INTRODUCTION

iOS Certificates – Apple Dc

Apple Inc. [US] https://developer.apple.com/account/ios/certificate/certificateList.action

Developer

Technologies Resources Programs Support Member Center Search Developer

Certificates, Identifiers & Profiles

Anthony Fleury

iOS Apps

Certificates
- All
- Pending
- Development
- Production

Identifiers

Devices

Provisioning Profiles

iOS Certificates

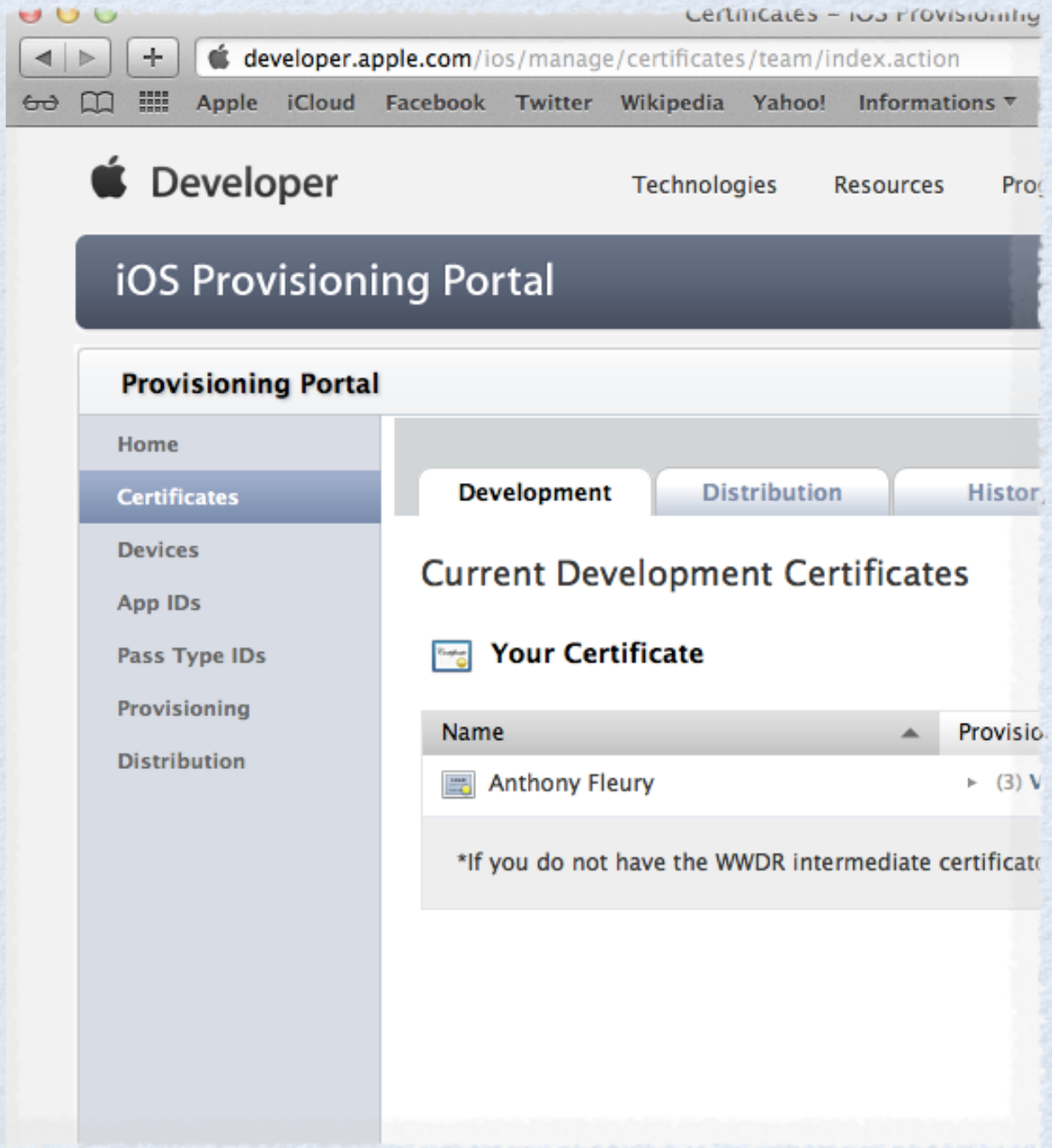
2 Certificates Total

Name	Type	Expires
Anthony Fleury	iOS Distribution	Feb 11, 2014
Anthony Fleury	iOS Development	Jun 21, 2014

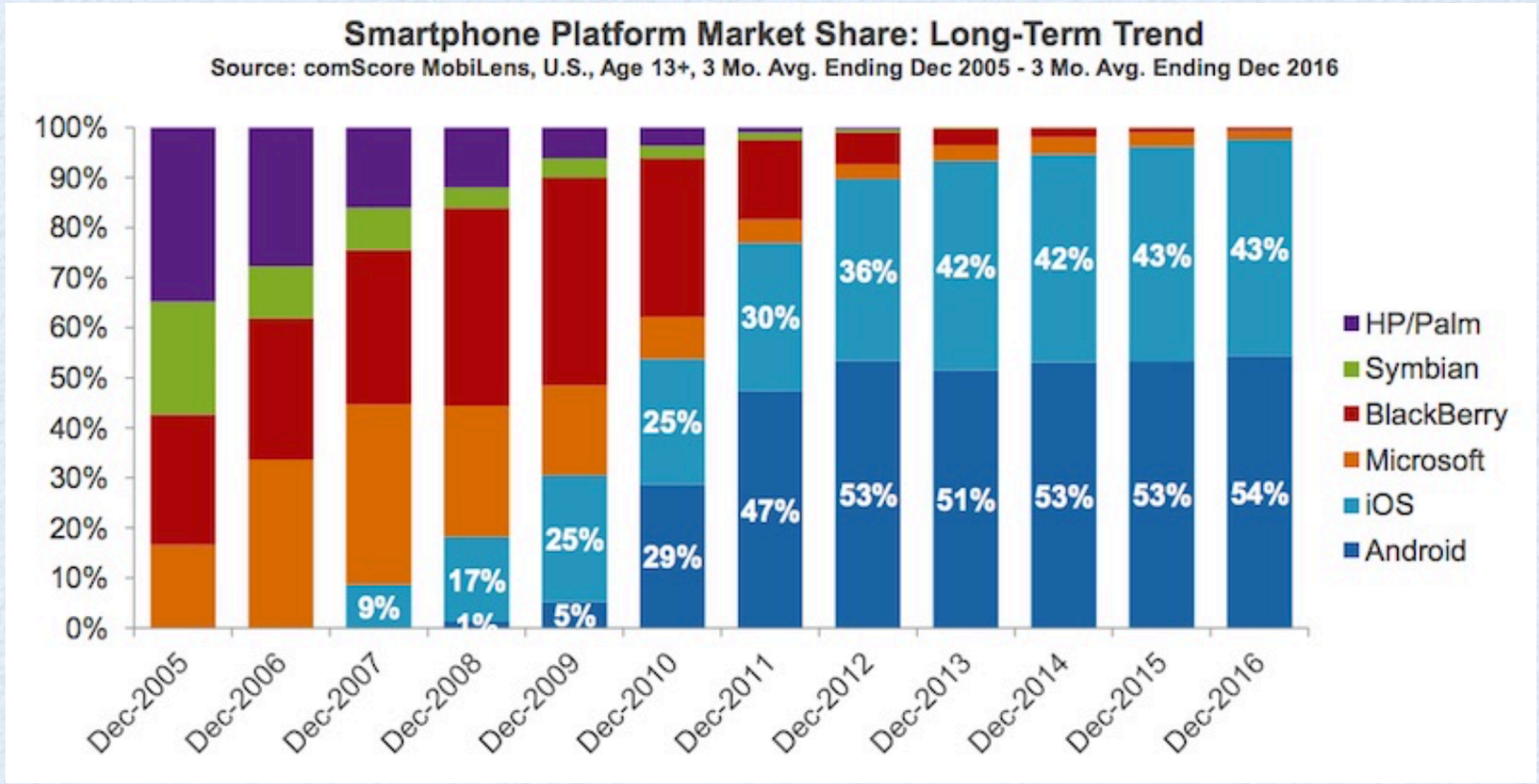
Copyright © 2013 Apple Inc. All rights reserved. Terms of Use | Privacy Policy

Introduction

- Plateforme «fermée»
- Développement d'une application : obligation d'avoir un certificat développeur
- Soumission à la plateforme Apple : vérification du respect des «règles» Apple !
- Politique de contrôle des ventes (obligations d'achats intégrés par leur plateforme)

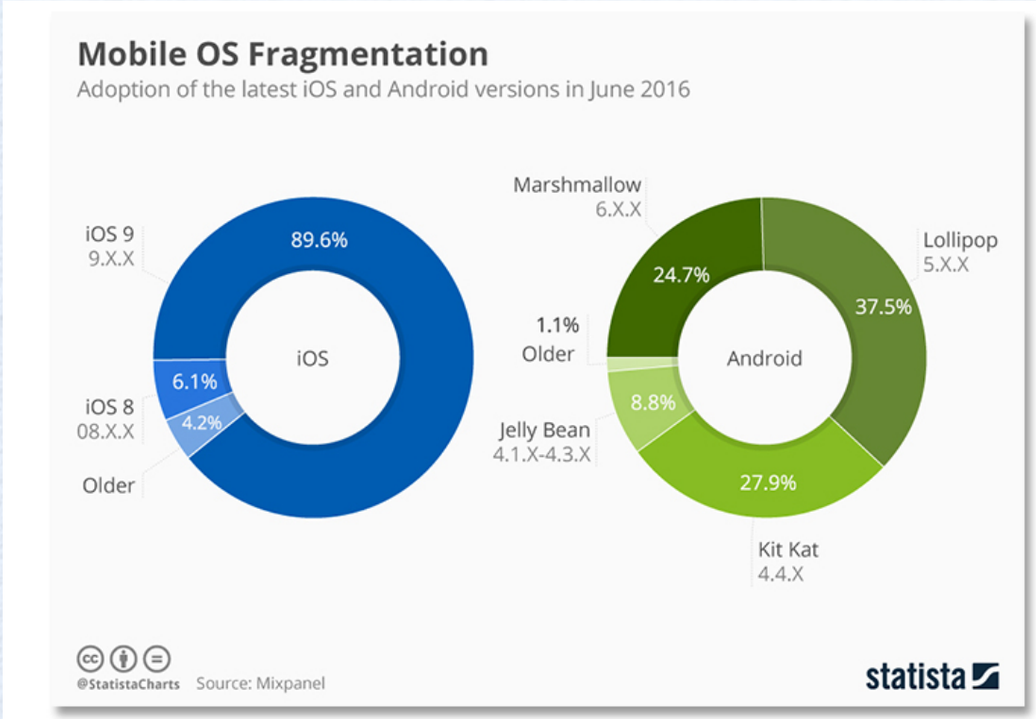


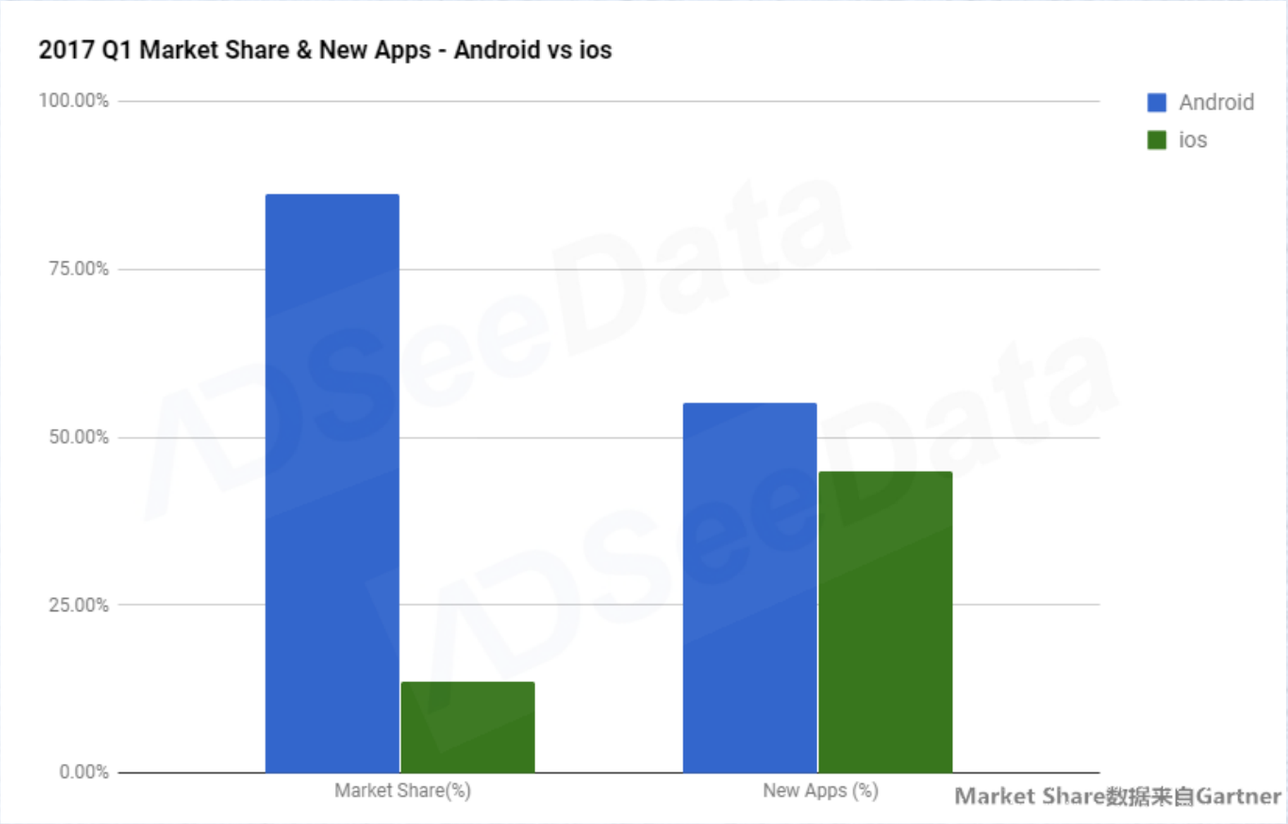
Android vs iOS



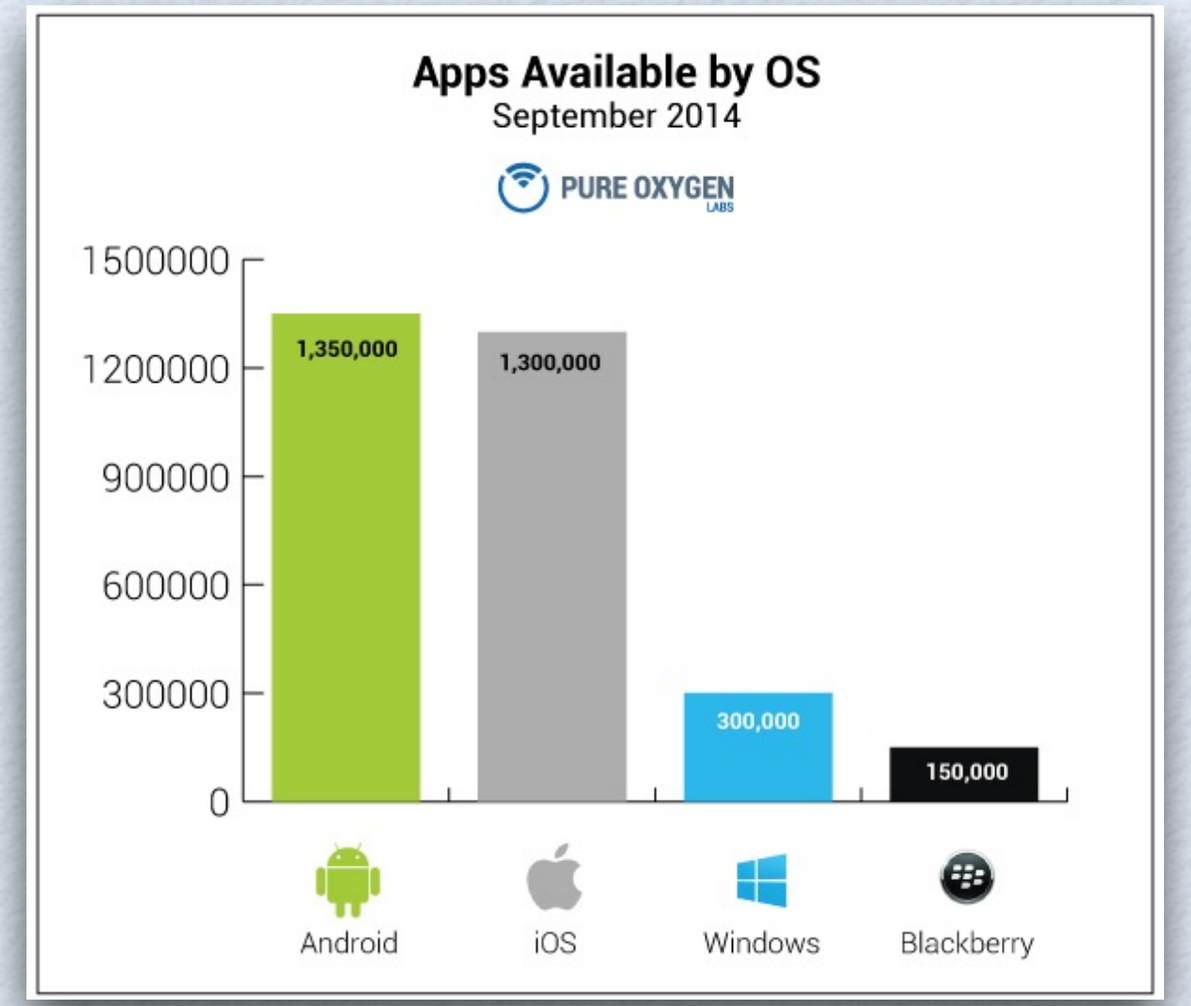
- Avantages / désavantages :
  - iOS sur une seule suite de matériel
  - Android plus ouvert

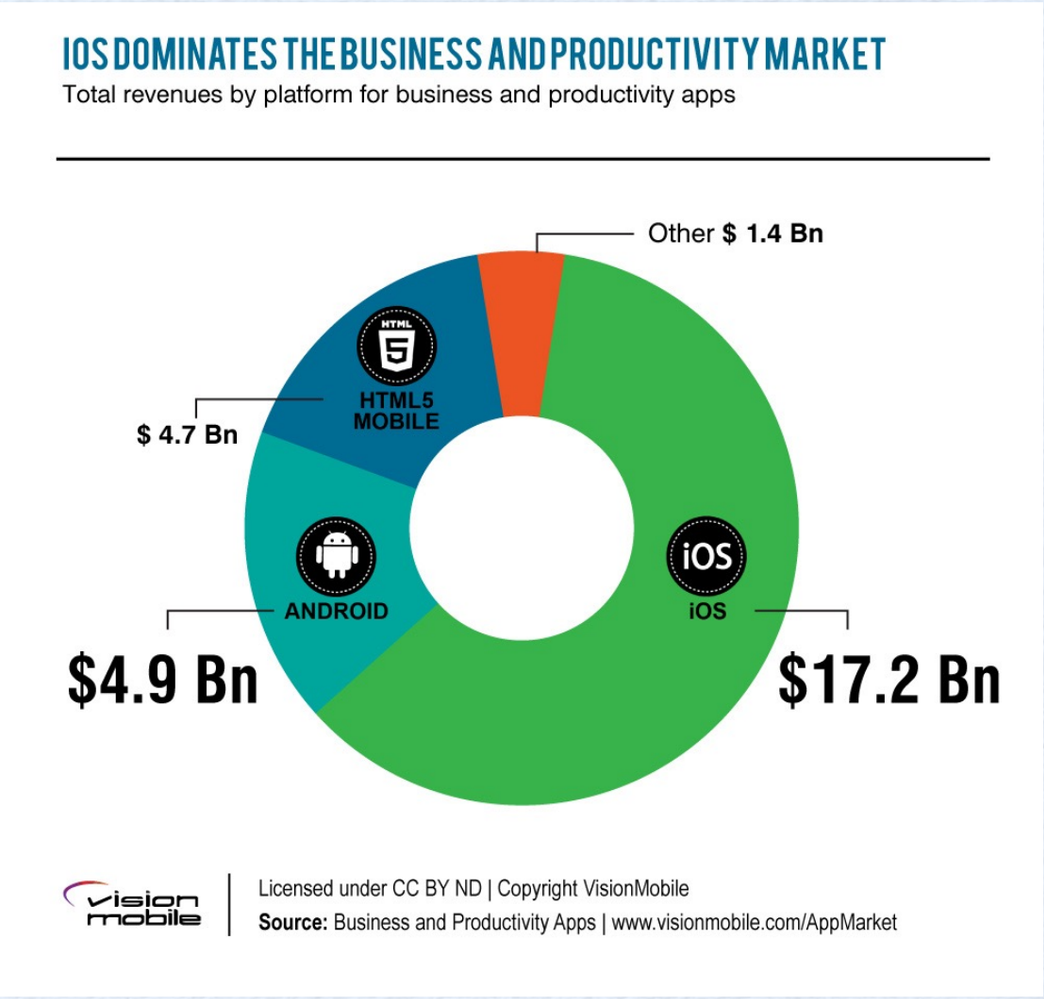
Android vs iOS





Android vs iOS





Un modèle économique qui s'est construit avec l'essor des smartphones :

- **Vente d'applications dans tous les domaines, créant un modèle inverse à celui connu en informatique jusqu'à maintenant…**
- **Recettes dues aux publicités pour les apps gratuites**

C'est un marché qui ne peut plus être ignoré —> Gros besoin de developpeurs.

Compte développeur

	Apple Developer	University Program	Standard Program Particulier	Standard Program Entreprise	Enterprise Program
Accès au site Dev Center	✓	✓	✓	✓	✓
iPhone SDK	✓	✓	✓	✓	✓
Accès aux versions Beta	✗	✗	✓	✓	✓
Création d'une équipe de développement	✗	✓	✗	✓	✓
Forums de discussions	✗	✓	✓	✓	✓
Déclaration d'incidents au support	✗	✗	2 par année d'adhésion	2 par année d'adhésion	2 par année d'adhésion
Déploiement sur iPhone, iPad et iPod Touch	✗	✓	✓	✓	✓
Distribution Ad Hoc	✗	✗	✓	✓	✓
Distribution In-House	✗	✗	✗	✗	✓
Distribution AppStore	✗	✗	✓	✓	✗
Prix d'adhésion	**Gratuit**	**Gratuit**	**$99/an**	**$99/an**	**$299/an**

Développer une Application

iOS Certificates – Apple Dc

Apple Inc. [US] https://developer.apple.com/account/ios/certificate/certificateList.action

Developer

Technologies Resources Programs Support Member Center Search Developer

Certificates, Identifiers & Profiles

Anthony Fleury

iOS Apps

Certificates
- All
- Pending
- Development
- Production

Identifiers

Devices

Provisioning Profiles

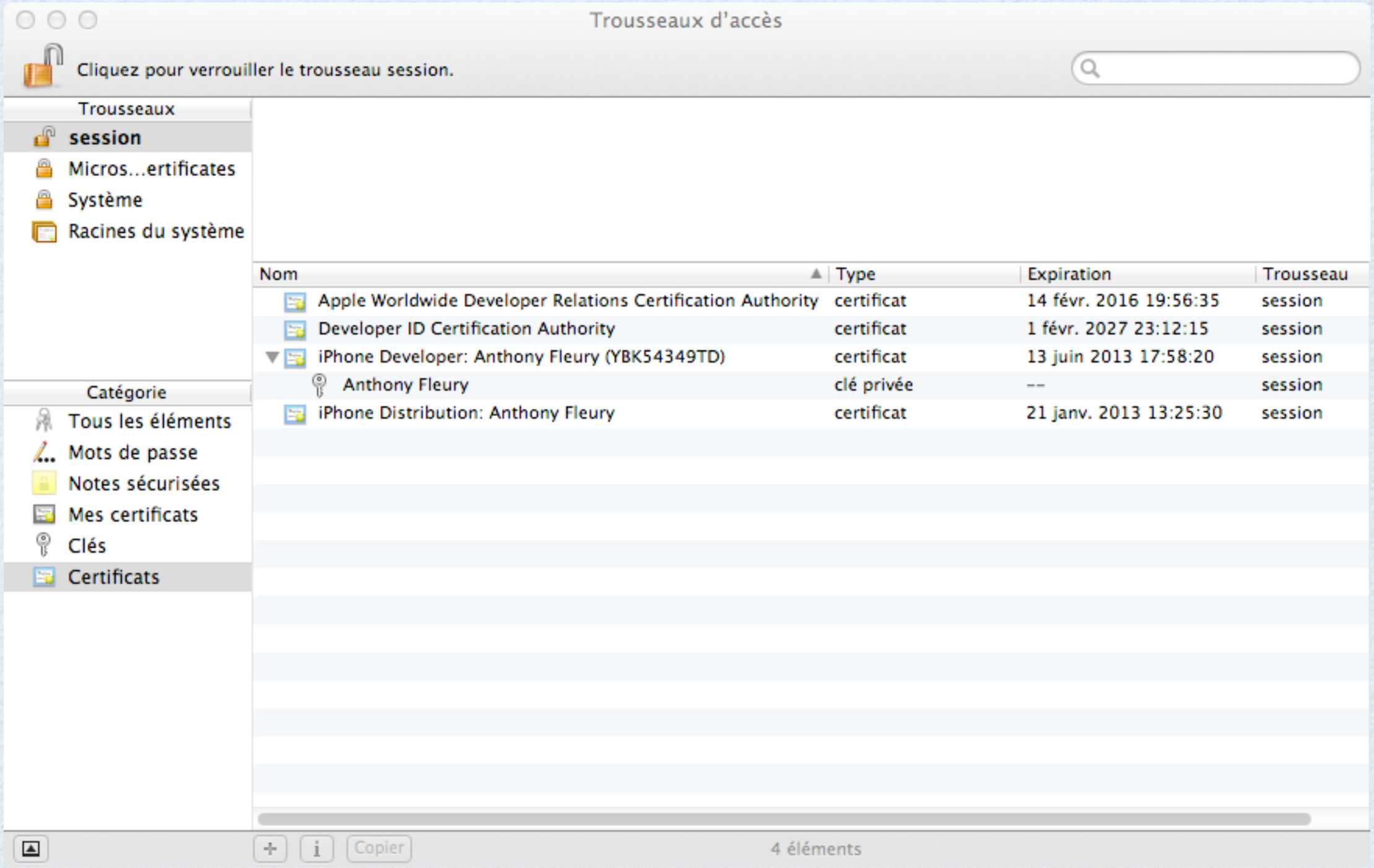
iOS Certificates

2 Certificates Total

Name	Type	Expires
Anthony Fleury	iOS Distribution	Feb 11, 2014
Anthony Fleury	iOS Development	Jun 21, 2014

Copyright © 2013 Apple Inc. All rights reserved. Terms of Use | Privacy Policy

Développer une Application



Développer une Application

Developer

Technologies Resources Programs Support Member Center Search Developer

iOS Provisioning Portal

Welcome, Anthony Fleury | Edit Profile | Log out

Provisioning Portal

Go to iOS Dev Center

- Home
- Certificates
- Devices
- App IDs
- Pass Type IDs
- Provisioning
- Distribution

Development | Distribution | History | How To

Development Provisioning Profiles

New Profile

Provisioning Profile	App ID	Status	Actions
DeveloppementsEtudiants	UEWZ786ANK.*	Active	Download Edit
ProvisionningActim	666YMGKRU2.com.minesdedouai.Ac...	Active	Download Edit
iOS Team Provisioning Profile:...	UEWZ786ANK.*	Active • Managed by Xcode	Download

Remove Selected

Développer une Application

Organizer – Devices

Devices | Repositories | Projects | Archives | Documentation

LIBRARY
- Provisioning Profiles
- Software Images
- Device Logs
- Screenshots

TEAMS
- Anthony Fleury

DEVICES
- My Mac 10.7.5 (11G63b)
  - Provisioning Profiles
- iPad afleury 5.1.1 (9B206)
  - Device Logs
  - Screenshots
- iPhone de afleury 6.0.1 (10A523)
  - Device Logs
  - Screenshots

Profile Name

Name	Platform	Creation	Expiration	App Identifier	Team	Status
DeveloppementsE...	iOS Profile	10/07/12	10/07/13	UEWZ786ANK.*	Anthony Fleury	Valid profile
ProvisionningActim	iOS Profile	10/07/12	10/07/13	666YMGKRU2.co...	Anthony Fleury	Valid profile
iOS Team Provisio...	iOS Profile	15/07/12	15/07/13	UEWZ786ANK.*	Anthony Fleury	Valid profile
iOS Team Provisio...	iOS Profile	09/08/12	09/08/13	UEWZ786ANK.*	Unknown	Valid profile

New | Import | Export | Refresh

Développer une Application

Développer une Application

Actimetry4iPhone.xcodeproj

Run | Stop | Actimetry4iPhone 〉 iOS Device (Scheme) | Breakpoints | Build Succeeded | 03/08/12 at 08:57 | Project ⚠3 | Editor | View | Organizer

Actimetry4iPhone
1 target, iOS SDK 6.0

- Classes
  - ActimetryAiPhoneAppDelegate.h
  - ActimetryAiPhoneAppDelegate.m
  - Views
    - FeedbackDetails.h
    - FeedbackDetails.m
    - Graphs.m
    - Graphs.h
    - Experimentation.m
    - Experimentation.h
    - ConfigureApp.h
    - ConfigureApp.m
    - acquisitionView.h
    - acquisitionView.m
    - FeedbackExperimentation.h
    - FeedbackExperimentation.m
    - FeedbackConf.h
    - FeedbackConf.m
    - feedBackExperiments.h
    - feedBackExperiments.m
  - AcquisitionHandler.h
  - AcquisitionHandler.m
  - AcqusitionWriter.h
  - AcqusitionWriter.m
  - ConfigHandler.h
  - ConfigHandler.m
  - FeedbackHandler.h
  - FeedbackHandler.m
  - SoundPlayer.h
  - SoundPlayer.m
  - RemoteControlAndView.h
  - RemoteControlAndView.m
- Other Sources
- Resources
  - Actimetry4iPhone-Info.plist
  - left_sound.aiff
  - biofeedback.png

PROJECT
- Actimetry4iPhone

TARGETS
- Actimetry4iPhone

Info | Build Settings

Basic | All | Combined | Levels

Setting	Resolved	Actimetry4iPhone	iOS Default
Architectures			
Architectures	Standard (armv7,	Standard (armv7,	Standard (armv7,
Base SDK	Latest iOS (iOS 6.0)	Latest iOS (iOS 6.0)	iOS 6.0
Code Signing			
Code Signing Identity	<Multiple values>		
Debug	iPhone Developer		
Any iOS SDK	iPhone Developer		
Distribution	iPhone Distribution		
Any iOS SDK	iPhone Distribution		
Release	iPhone Developer		
Any iOS SDK	iPhone Developer		
Deployment			
Installation Directory			
Strip Linked Product	Yes		
Targeted Device Family	iPhone		
iOS Deployment Target	iOS 5.0		
Packaging			
Info.plist File			
Product Name			
Apple LLVM compiler 4.1 – Language			
C Language Dialect	C99 [-std=c99]	C99 [-std=c99]	Compiler Default
Other C Flags	<Multiple values>	<Multiple values>	
Debug			
Distribution	-DNS_BLOCK_ASSERTIO...	-DNS_BLOCK_ASSERTIO...	
Release	-DNS_BLOCK_ASSERTIO...	-DNS_BLOCK_ASSERTIO...	
Apple LLVM compiler 4.1 – Warnings – All languages			
Mismatched Return Type	Yes	Yes	No
Missing Function Prototypes	Yes	Yes	No
Unused Variables	Yes	Yes	No

Don't Code Sign
Automatic Profile Selector (Recommended)
iPhone Developer
✓ iPhone Distribution
iOS Team Provisioning Profile: * – 9 août 2012 16:42:10 (for bundle identifiers '*')
iPhone Developer: Anthony Fleury (YBK54349TD)
iOS Team Provisioning Profile: * – 15 juil. 2012 18:59:45 (for bundle identifiers '*')
iPhone Developer: Anthony Fleury (YBK54349TD)
ProvisionningActim (for bundle identifiers 'com.minesdedouai.Actim9d4iphone')
iPhone Developer: Anthony Fleury (YBK54349TD)
DeveloppementsEtudiants (for bundle identifiers '*')
iPhone Developer: Anthony Fleury (YBK54349TD)
Other...

Identity
Project Name: Actimetry4iPhone
Location: Not Applicable
Actimetry4iPhone.xcodeproj
Full Path: /Users/fleury_anthony/...

Object Library
- Push Button – Intercepts mouse-down events and sends an action message to a target object when...
- Gradient Button – Intercepts mouse-down events and sends an action message to a target object...
- Rounded Rect Button – Intercepts mouse-down events and sends an action message to a target object...

Add Target | Validate Settings | Add Build Setting

Distribuer une Application

- Plusieurs méthodes de distribution pour entreprises (ad-hoc et App store)
- Pour les particuliers et entreprises passant par App Store :
  - Développement de l'application avec l'ensemble de ses fonctionnalités
  - Debug de l'application en mode distribution sur un ensemble de devices
  - Envoie à Apple pour validation :
    - Validation du contenu de l'application (certains types d'applications sont interdites, au choix d'Apple)
    - Validation du comportement de l'application (attention à ce qui est fait avec le carnet d'adresse, les messages etc.)
    - Validation de l' « expérience utilisateur » : pas de bug, une bonne présentation de l'interface, une navigation « logique » et conforme à ce qui se fait dans les autres applications etc.
    - Respect des règles et utilisation des outils Apple (achat in-app etc.)
    - Non contournement et non utilisation de parties privées de l'API Apple ou autre...

Vendre une Application

- Une fois accepté, l'application est distribué sur le store soit gratuitement, soit payante.
- Pour les applications payantes :
  - Choix du prix par le développeur
  - Demande d'application de taxe (formulaires à remplir...)
  - Le développeur touchera environ 60% du prix de vente pour chaque application (le reste ira à Apple et au fisc américain)
  - Possibilité aussi de rajouter de la pub dans les applications : rétribution dépendant alors de la régie publicitaire utilisée et de ses conditions de paiement.

XCode : l'écriture de l'App !

Fichiers de votre projet

Votre code

Propriétés de l'objet en cours

Actimetry4iPhone.xcodeproj — Actimetry4iPhoneAppDelegate.m

Build Succeeded | 03/08/12 at 08:57

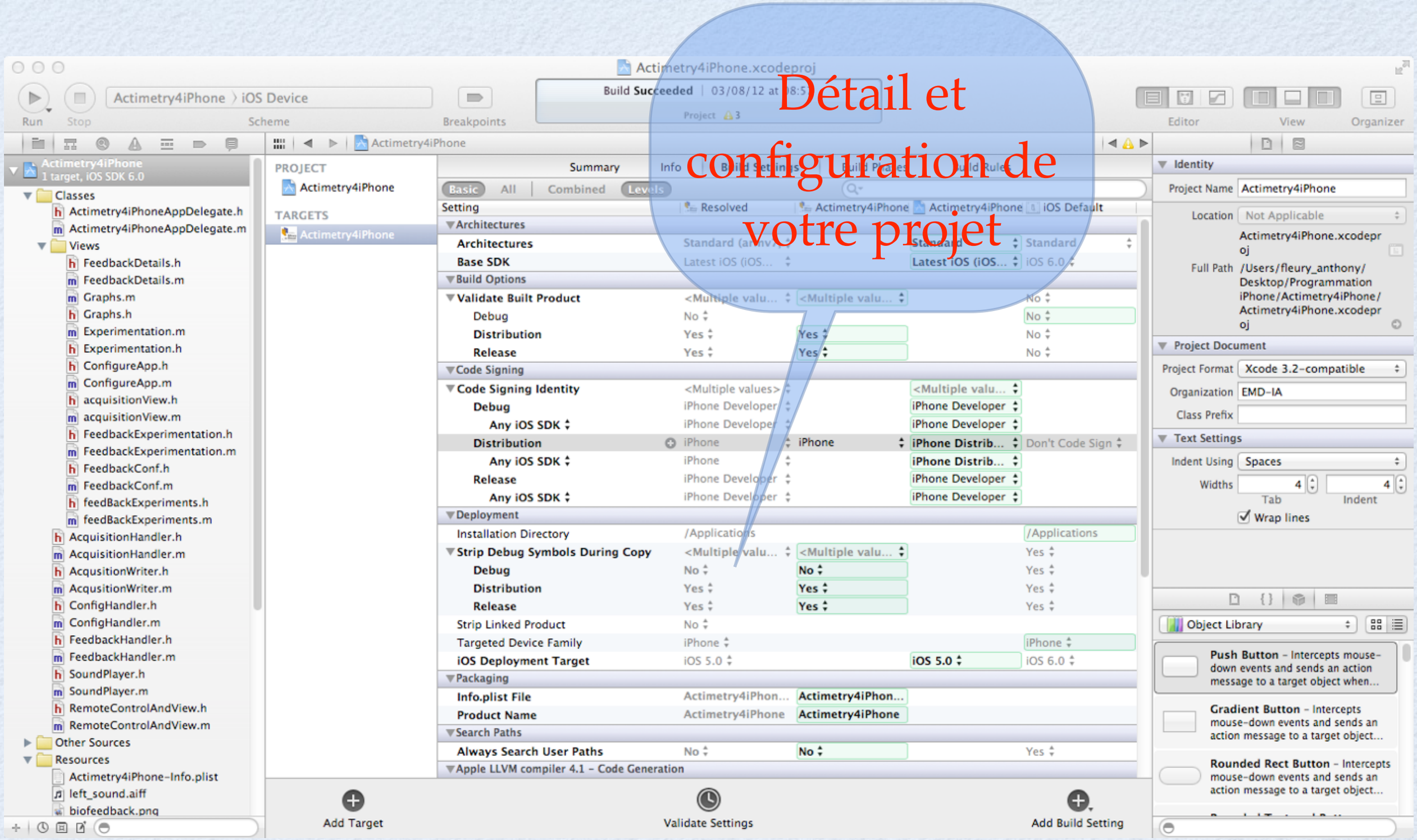
Actimetry4iPhone > Classes > Actimetry4iPhoneAppDelegate.m > No Selection

Actimetry4iPhoneAppDelegate.m
Views
FeedbackDetails.h
FeedbackDetails.m
Graphs.m
Graphs.h
Experimentation.m
Experimentation.h
ConfigureApp.h
ConfigureApp.m
acquisitionView.h
acquisitionView.m
FeedbackExperimentation.h
FeedbackExperimentation.m
FeedbackConf.h
FeedbackConf.m
feedBackExperiments.h
feedBackExperiments.m
AcquisitionHandler.h
AcquisitionHandler.m
AcqusitionWriter.h
AcqusitionWriter.m
ConfigHandler.h
ConfigHandler.m
FeedbackHandler.h
FeedbackHandler.m
SoundPlayer.h
SoundPlayer.m
RemoteControlAndView.h
RemoteControlAndView.m
Other Sources
Resources
Actimetry4iPhone-Info.plist
left_sound.aiff
biofeedback.png
Engrenages.png
running.png
Icon.png

```
//
//  Actimetry4iPhoneAppDelegate.m
//  Actimetry4iPhone
//
//  Created by Anthony Fleury (Anthony.Fleury@mines-douai.fr) on 5/19/11.
//  Copyright 2011 Ecole des Mines de Douai - Dpt IA. All rights reserved.
//
//  Application delegate.

#import "Actimetry4iPhoneAppDelegate.h"

@implementation Actimetry4iPhoneAppDelegate

@synthesize window;
@synthesize tabBarController;


#pragma mark -
#pragma mark Application lifecycle

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    self.window.rootViewController = self.tabBarController;
    [self.window makeKeyAndVisible];

    return YES;
}


- (void)applicationWillResignActive:(UIApplication *)application {
}


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


- (void)applicationWillEnterForeground:(UIApplication *)application {
}


- (void)applicationDidBecomeActive:(UIApplication *)application {
}


- (void)applicationWillTerminate:(UIApplication *)application {
}


#pragma mark -
#pragma mark UITabBarControllerDelegate methods

#pragma mark -
```

Identity and Type
File Name: Actimetry4iPhoneAppDelegate.m
File Type: Default - Objective-C...
Location: Relative to Group
Actimetry4iPhoneAppDelegate.m
Full Path: /Users/fleury_anthony/Desktop/Programmation iPhone/Actimetry4iPhone/Classes/Actimetry4iPhoneAppDelegate.m
Localization
Make localized...
Target Membership
Actimetry4iPhone
Text Settings
Text Encoding: Unicode (UTF-8)
Object Library
Push Button - Intercepts mouse-down events and sends an action message to a target object when...
Gradient Button - Intercepts mouse-down events and sends an action message to a target object...
Rounded Rect Button - Intercepts mouse-down events and sends an action message to a target object...

XCode : L'écriture de l'App !

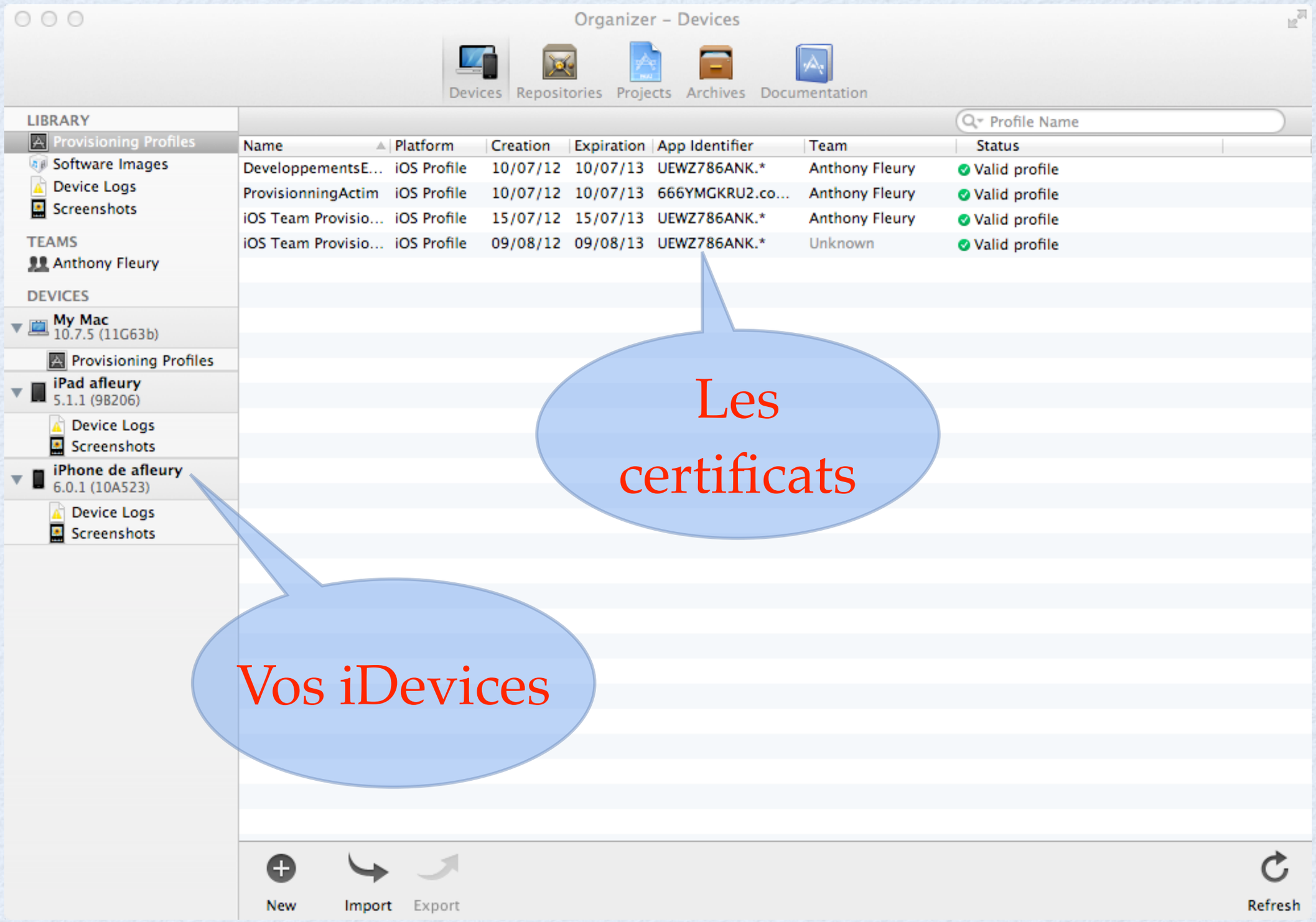


Détail et configuration de votre projet

XCode : L'écriture de l'App !

XCode : l'écriture de l'App !

XCode : L'écriture de l'App !



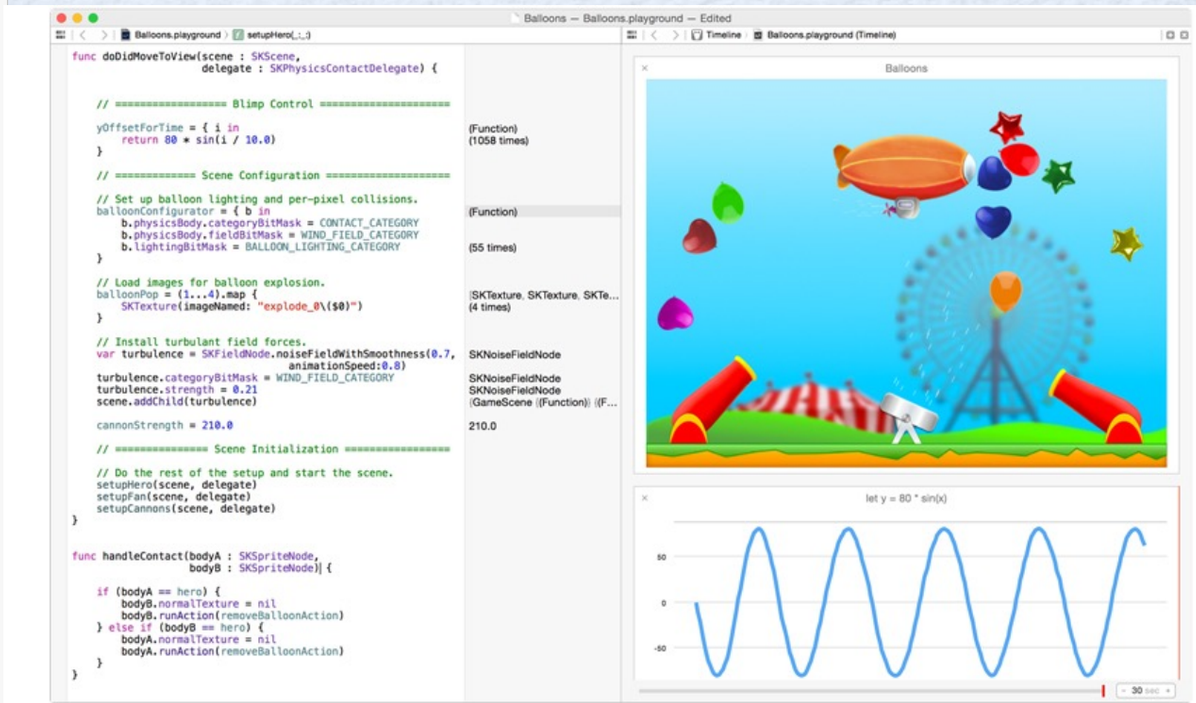
Les alternatives à XCode

- Faire des applications Web (cf. cours de Luc Fabresse)
- Utiliser des SDKs alternatifs (Xamarin, Marmelade par exemple), qui sont en général compatibles avec plusieurs plateformes (et permettent des développements en différents langages)
- Utiliser des outils portant des applications Web en native (PhoneGap)

Les alternatives à Objective C

- Depuis Septembre 2014, nouveau langage de programmation : SWIFT. Inférence de type, génériques, surcharge d'opérateurs, notions avancées de blocs, …
- Playground + REPL —> Xcode > 6.1

```
var str = "Hello, playground"                    "Hello, playground"


let triple: Int -> Int = {                       <opaque>
    (number: Int) in
    let result = 3 * number                      9
    number                                       3
    return result                                9
}
triple(3)                                        9


let listOfNumbers = 1..5                         {_startIndex 1, _endIndex 5}
var sum = 0                                      0
for n in listOfNumbers {                         (4 times)
    sum += n
}


var j = 2                                        2
for var i = 0; i < 5; ++i
{
    j * i                                        (5 times)

}
```



Avantage des Applications Natives

- Accès à toutes les ressources (carnet d'adresses, localisation, calendrier...) et à tous les capteurs (appareil photo, caméra, capteur de proximité...).
- Avec le SDK iOS : look & feel «Apple» (les applications iOS sont toutes sur un même ensemble de modèles)

OBJECTIVE C

Présentation du langage - Différences avec Java et C

Historique

- Langage mis au point début des années 1980 par Brad Cox
- Créé un langage orienté objet à partir de C et de Smalltalk 80
- Utilisé par Steve Jobs en 1985 pour l'OS NeXTStep.
- En 2007, Apple sort Mac OS X 10.5 qui remet au gout du jour le langage Objective C avec notamment un garbage collector, la notion de blocks...
- Cocoa est l'API (non libre) pour Objective C développée par Apple. GNUStep est une API libre pour ce langage.

Objective-C : Organisation du code

- En C :
  - Fichiers .h pour les déclarations des fonctions, des structures, etc.
  - Fichier .c pour la définition des fonctions.
- En Java :
  - Toute la classe (déclaration, définition des méthodes) sont dans le même .java.
- En Objective C :
  - Fichiers .h pour la déclaration des classes (comprenant les variables membres et les méthodes)
  - Fichiers .m pour la définition des méthodes de chaque classe et aussi des fonctions comme la fonction main().
- En C et en Objective C :
  - Les fichiers .c ou .m incluent les fichiers .h déclarant les classes et les fonctions utilisées.

Objective-C : Les classes (1/5)

- Comme en Java, une classe est la modélisation d'un concept avec :
  - Des données membres
  - Des méthodes associées
    - Méthodes de classe pour une méthode s'appliquant à la classe
    - Méthodes d'instances pour une méthode s'appliquant à une instance de la classe.

Objective-C : Les classes (2/5)

- Toute classe dérive, en Objectice-C, de la classe NSObject (la classe «racine»).
- Lorsque l'on déclare une instance, on ne peut déclarer qu'un pointeur ! (pas comme en Java)
- NSObject offre par défaut un ensemble de méthodes (allocation, initialisation) que l'utilisateur peut réécrire pour sa classe.
- Possède un **pointeur** sur l'instance en cours nommé self (accès au membre avec ->)
- nil représente le pointeur adresse «invalide»

Objective-C : Les classes (3/5)

fichier livre.h :

```
@interface Livre : NSObject {
   NSString* titre;
   NSString* auteur;
   int anneeParution;
}

- (id) init;
-  (id)  initWithTitre:(NSString*)  titre
auteur:(NSString*)  auteur  annee:  (int)
annee;
- (void) dealloc;
@end
```

fichier main.m :

```
#import «livre.h»

int main(int argc, const char* argv[]) {
   Livre* l = [[Livre alloc] init];

   Livre*  l2  =  [[Livre  alloc]
   initWithTitre:@«Titre»
   auteur:@«auteur» annee:2012];
}
```

fichier livre.m :

```
#import «livre.h»
@implementation Livre
- (id) init
{
   self = [super init];
   if(self) {
      titre = [[NSString alloc] init];
      auteur = [[NSString alloc] init];
      anneeParution = 2012;
   }

   return self;
}

-  (id)  initWithTitre:(NSString*)  titre  auteur:
(NSString*) auteur annee: (int) annee
{
   self = [super init];
   if(self) {
      self->titre = titre;
      self->auteur = auteur;
      self->anneeParution = annee;
   }

   return self;
}

- (void) dealloc
{
   [super dealloc];
}
@end
```

Type « générique », peut désigner tout type…

Objective-C : Les classes (3/5)

fichier main.m :

```
#import «livre.h»

int main(int argc, const char* argv[]) {
   Livre* l = [[Livre alloc] init];

   Livre* l2 = [[Livre alloc] initWithTitre:@«Titre» auteur:@«auteur» annee:2012];
}
```

- En Objective-C, passage de message par utilisation des [] : [Livre alloc] appelle la méthode alloc de la classe Livre (méthode de classe).
- [l description] appelle la méthode d'instance description sur l'instance l.

Objective-C : Les classes (3/5)

fichier main.m :

```
#import «livre.h»

int main(int argc, const char* argv[]) {
   Livre* l = [[Livre alloc] init];

   Livre* l2 = [[Livre alloc] initWithTitre:@«Titre» auteur:@«auteur» annee:2012];
}
```

- Les méthodes déclarent leurs paramètres d'entrée avec des « noms » les décrivant. Ainsi, ces étiquettes sont réutilisées dans le passage de message suivi d'un « : » et de la valeur.
- Idem pour les constructeurs, initWith… est le format standard en Objective-C.

Objective-C : Les classes (4/5)

fichier livre.h :

```
@interface Livre : NSObject {
   NSString* titre;
   NSString* auteur;
   int anneeParution;
}

- (id) init;
-  (id)  initWithTitre:(NSString*)  titre
auteur:(NSString*)  auteur  annee:  (int)
annee;

// getters
- (int) anneeParution;
- (NSString*) titre;
- (NSString*) auteur;
// setters
- (void) setAnneeParution: (int) annee;
- (void) setTitre: (NSString*) titre;
- (void) setAuteur: (NSString*) auteur;

@end
```

fichier livre.m :

```
#import «livre.h»
@implementation Livre
- (id) init {
   //...
}
-  (id)  initWithTitre:(NSString*)  titre  auteur:
(NSString*) auteur annee: (int) annee {
   //...
}
- (void) dealloc {
   //...
}
- (NSString*) titre {
   return titre;
}
- (NSString*) auteur {
   return auteur;
}
- (NSString*) anneeParution {
   return anneeParution;
}
- (void) setAnneeParution: (int) annee {
   anneeParution = annee;
}
- (void) setTitre: (NSString*) titre {
   self->titre = titre;
}
- (void) setAnneeParution: (NSString*) auteur {
   self->auteur = auteur;
}
@end
```

Objective-C : Les classes (5/5)

fichier livre.h :

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

@interface Livre : NSObject

- init;
- initWithTitre:(NSString*) titre auteur:
(NSString*) auteur annee: (int) annee;
- (void) dealloc;

@property (nonatomic, strong) NSString*
titre;
@property (nonatomic, strong) NSString*
auteur;
@property (nonatomic) int anneeParution;
@end
```

fichier livre.m :

```
#import «livre.h»

@implementation Livre

- (id) init {
   //...
}
- (id) initWithTitre:(NSString*) titre
auteur:(NSString*) auteur annee: (int)
annee {
   //...
}
- (void) dealloc {
   //...
}
@end
```

Même comportement : génère les accesseurs !!!

Objective-C : Public ? Private ? Protected ?

- Mots clés @public, @private, @protected
- Même sens qu'en Java
- @protected par défaut.

fichier livre.h :

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

@interface Livre : NSObject {
   @public:
   @private:
   @protected:
      NSString* titre;
      NSString* auteur;
      int anneeParution;
}

- (id) init;
- (id) initWithTitre:(NSString*) titre
auteur:(NSString*) auteur annee: (int)
annee;
- (void) dealloc;

@property (nonatomic, strong) NSString*
titre;
@property (nonatomic, strong) NSString*
auteur;
@property (nonatomic) int anneeParution;
@end
```

Gestion mémoire

- Mémoire gérée par « Automatic Reference Counting ».
- S'occupe de savoir si un objet est encore référencé dans le code, et donc si sa valeur est accessible par une partie du code.
- Les weak et strong des propriétés sont là pour gérer ceci dans les accesseurs.

Objective-C : L'héritage

- L'héritage s'écrit avec
  - @interface Nom : ClasseDeBase
- Idem qu'en Java, les membres public peuvent être accéder « de partout », les protected sont limités à la classe et à ses sous classes, les private à la classe
- Accès à un pointeur sur la classe mère : super.
- Possibilité de remplacer (ou non) les méthodes de la classe de base.
- Ajout de membres et de méthodes pour spécifier la nouvelle entité.

Objective-C : Les Catégories

- Une alternative à l'héritage.
- Ne permet de spécifier QUE des méthodes.
- Notation : @interface NomClasse (NomCategorie)
- Convention : fichier .h nommé «NomClasse+NomCategorie.h»
- Le nom de la classe reste NomClasse (l'inclusion du second .h avec la catégorie spécifie que l'on utilise la catégorie).

Objective-C : Les sélecteurs (1/2)

- Un sélecteur est un objet définissant un message à envoyer à un objet.
- On peut définir ce sélecteur à partir de plusieurs choses :
  - une chaîne de caractère
  - une variable (permet ainsi d'exécuter des méthodes dont on ne connaît même pas le nom à la compilation)
- Possibilité aussi de vérifier qu'un objet répond à un sélecteur (et donc à un message avec un certain nombre de paramètres).

Objective-C : Les sélecteurs (2/2)

fichier livre.h :

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

@interface Livre : NSObject {
     NSString* titre;
     NSString* auteur;
     int anneeParution;
}

- (id) init;
- (id) initWithTitre:(NSString*) titre
auteur:(NSString*) auteur annee: (int)
annee;
- (void) dealloc;

- (void) afficheLivreLegende:(NSString*)
s;
- (void) afficheLivre;

@property (nonatomic, strong) NSString*
titre;
@property (nonatomic, strong) NSString*
auteur;
@property (nonatomic) int anneeParution;
@end
```

fichier main.m :

```
#import «livre.h»

int main(int argc, const char* argv[]) {
   Livre* l = [[Livre alloc]
   initWithTitre:@«Test»
   auteur:@«L'auteur» annee:2012];

   SEL selecteurPourAffichageAvecLegende
   = @selector(afficheLivreLegende:);

   SEL selecteurPourAffichage =
   @selector(afficheLivre);

   [l
   performSelector:selecteurPourAffichage
   AvecLegende withObject:@ «Le livre
   suivant est :»];

   if ([l
   respondsToSelector:selecteurPourAffich
   age]) {
      [l
      performSelector:selecteurPourAfficha
      ge];
   }
}
```

Objective-C : Les Protocoles (1/2)

- Permet de spécifier qu'une classe implémente un certain nombre de méthodes (sans pour autant dériver et donc avoir accès aux variables membres).
- Utilisé notamment dans les interfaces graphiques sous Cocoa (délégation).

Objective-C : Les Protocoles (2/2)

fichier affichage.h :

```
#import <Founation/Foundation.h>
@protocole affichage <NSObject>
- (void) afficher;
@required:
@optional:
@end
```

fichier livre.h :

```
#import «affichage.h»
#import <Founation/Foundation.h>
@interface Livre : NSObject <affichage> {
   NSString* titre;
   NSString* auteur;
   int anneeParution;
}

- (id) init;
-  (id)  initWithTitre:(NSString*)  titre
auteur:(NSString*)  auteur  annee:  (int)
annee;
- (void) dealloc;
- (void) afficher;

@property  (nonatomic,  strong)  NSString*
titre;
@property  (nonatomic,  strong)  NSString*
auteur;
@property (nonatomic) int anneeParution;
@end
```

fichier livre.m :

```
#import «livre.h»

@implementation Livre

   //...

- (void) afficher {
   // ...
   // implémente la méthode affichage du
   // protocole affichage.
}
@end
```

fichier main.m :

```
#import «livre.h»
#import «affichage.h»
int main(int argc, const char* argv[]) {
   Livre* l = [[Livre alloc] init];
   id<affichage> afficheur = l;
   [afficheur afficher];
}
```

Objective-C : Quelques classes utiles... NSArray et NSMutableArray (1/2)

- En C : tableau avec [], sur des éléments de types déterminés.
- En Objective C, possibilité de faire ces tableaux, y compris sur des types de base.
- Classe NSArray (plus pratique et plus facile à utiliser) avec des objets de tous types et NSMutableArray avec des objets de tout types et de taille non fixe.

Objective-C : Quelques classes utiles... NSArray et NSMutableArray (1/2)

fichier main.m :

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

int main(int argc, const char* argv[]) {
  NSMutableArray * matieres = [[NSMutableArray alloc] init];

  [matieres addObject:@«Cours iOS»];
  [matieres addObject:@«Cours Android»];

    NSLog(«%d Objet», [matieres count]);

  for(int i = 0; i < [matieres count]; i++) {
    NSLog(«Cours %d : %@», i, [matieres objectAtIndex:i]);
  }

  NSArray* matieres2 = [[NSArray alloc] initWithObjects:@«Cours iOS», @«Cours
  Android», nil];

  [matieres2 makeObjectsPerformSelector:@selector(actionARealiser)];
}
```

La version «mutable» permet de faire varier la taille ou les éléments !

Objective-C : Quelques classes utiles... NSString (1/2)

- La classe NSString permet de gérer facilement des chaînes de caractères.
- Les chaînes Objective C comme en C sont entre guillemets, mais avec un @ avant. (ex. @«Hello World».
- Offre des possibilités de recherche, d'ajout de chaînes, de constructions à partir de formattage « à la printf », etc.

Objective-C : Quelques classes utiles... NSString (2/2)

fichier main.m :

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

int main(int argc, const char* argv[]) {
    NSString * hello = [[NSString alloc] initWithString:@«Hello»];
    NSString * world = @« World !»;

    NSString * helloWorld = [hello stringByAppendingString:world];

    int a = 0; double b = 1;

    NSString * formattage = [[NSString alloc] initWithFormat:@«a vaut : %d, b vaut :
    %lf ! %@», a, b, helloWorld];

    NSLog(@«%@», helloWorld);
    NSLog(@«%@», formattage);
}
```

Objective-C : Quelques classes utiles... NSDictionnary et NSMutableDictionnary (1/2)

- Un dictionnaire est un tableau qui va associer une valeur à une clé.
- L'objet (quelque soit son type) est associé à une clé (une chaîne de caractère.
- Très utilisé dans les configurations notamment (pour sauvegarder des paramètres).

Objective-C : Quelques classes utiles... NSDictionnary et NSMutableDictionnary (2/2)

fichier main.m :

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

int main(int argc, const char* argv[]) {
    NSArray* Noms = [[NSArray alloc] initWithObjects:@«Def», @«Info», nil];
    NSArray* Tels = [[NSArray alloc] initWithObjects:[NSNumber numberWithInt:2044],
    [NSNumber numberWithInt:2381], nil];
    NSMutableDictionnary* annuaire = [NSDictionnary dictionaryWithObjects:Tels
    forKeys:Noms];

    NSDictionnary* annuaire2 = [NSDictionnary dictionaryWithObjectsAndKeys:[NSNumber
    numberWithInt:2044], @«Def», [NSNumber numberWithInt:2381], @«Info», nil];

    NSLog(@«Numéro pour Info : %d», [[annuaire objectForKey:@«Info»] intValue]);

    [annuaire setObject:[NSNumber numberWithInt:2345], forKey:@«autreNom»];
}
```

Attention : on ne manipule que des objets dans un NSArray comme dans un dictionnaire, d'où l'utilisation de NSNumber que l'on re-transforme ensuite en entier. Idem, NSMutableDictionnary va permettre d'avoir des dictionnaires de taille non fixe avec des objets modifiables.

Objective-C : Gestion mémoire

- L'objet NSAutoreleasePool s'occupe de gérer les variables mises en autorelease dans le programme.

fichier main.m :

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

int main(int argc, const char* argv[]) {
   NSAutoreleasePool* pool = [[NSAutoreleasePool alloc] init];

   // Votre code...

   [pool drain]; // [pool release] moins specifique
}
```