DÉVELOPPEMENT SOUS IOS

Centre de Recherche Neurosciences Lyon

PREMIÈRE APPLICATION

SWIFT / OBJECTIVE-C

T1.A04

JDEV2017

- **Introduction**
  - Installation de XCode
  - Environnement iOS
  - Licences
  - Distribution
- **Environnement de développement**
  - Découverte de XCode
- **Objective-C / Swift**
  - Les classes
  - Frameworks
  - Création d'une interface graphique
  - Connexion à un capteur externe via BLE

INTRODUCTION

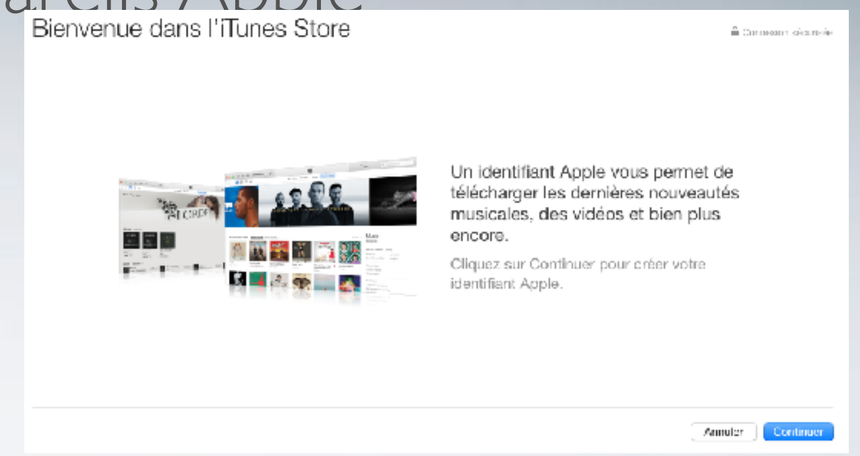
ETAPES

- Ordinateur Apple
- Création d'un Apple ID
- Enregistrement compte développeur
- Chargement et installation de XCode

CRÉATION D'UN APPLE ID (1)

- Web (à préférer)
- iTunes (nécessite la saisie d'une carte pour c
- Indispensable pour l'utilisation des services et appareils Apple

https://appleid.apple.com



iTunes

CRÉATION D'UN APPLE ID (2)

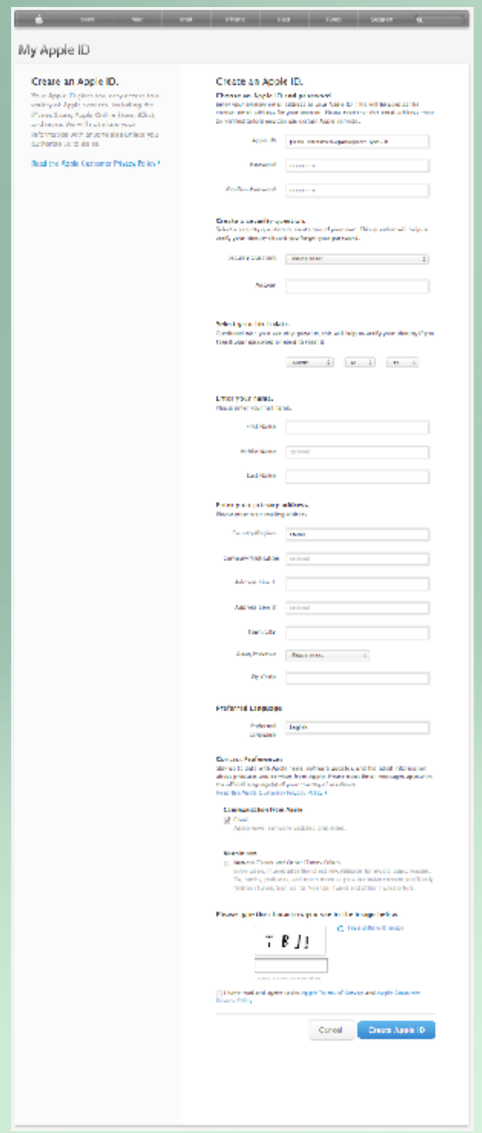
Web

iTunes

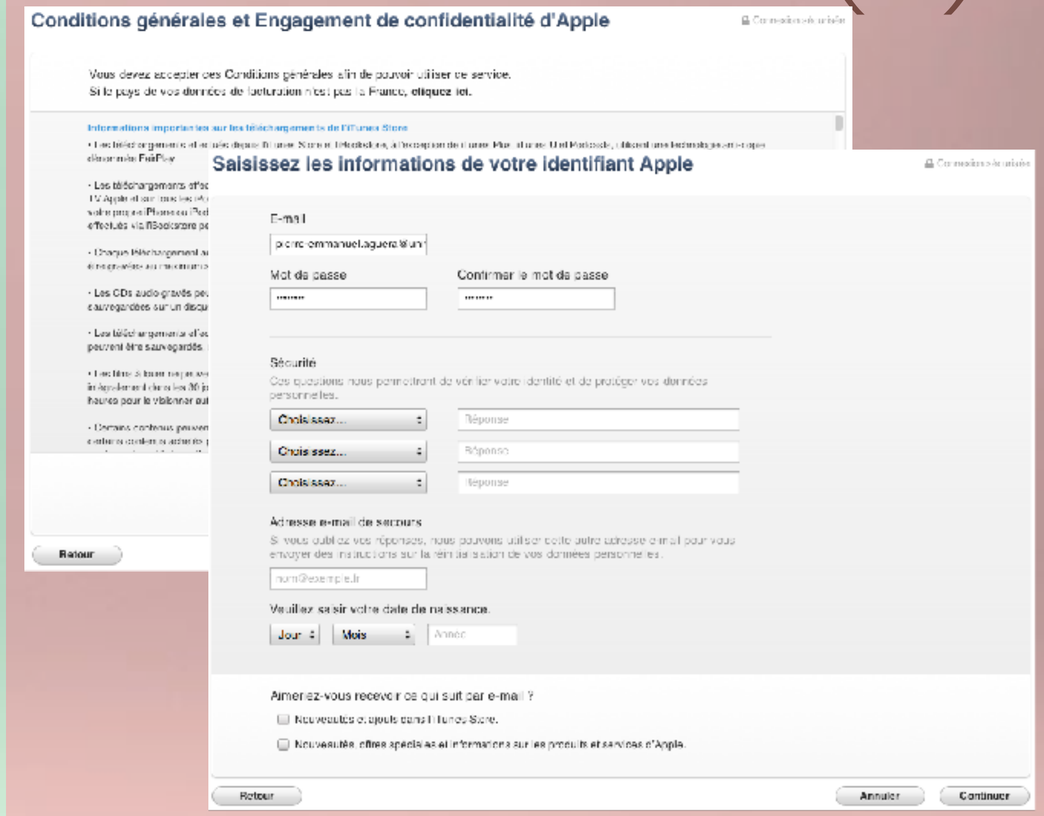
CRÉATION D'UN APPLE ID (2)

Web iTunes

CRÉATION D'UN APPLE ID (2)

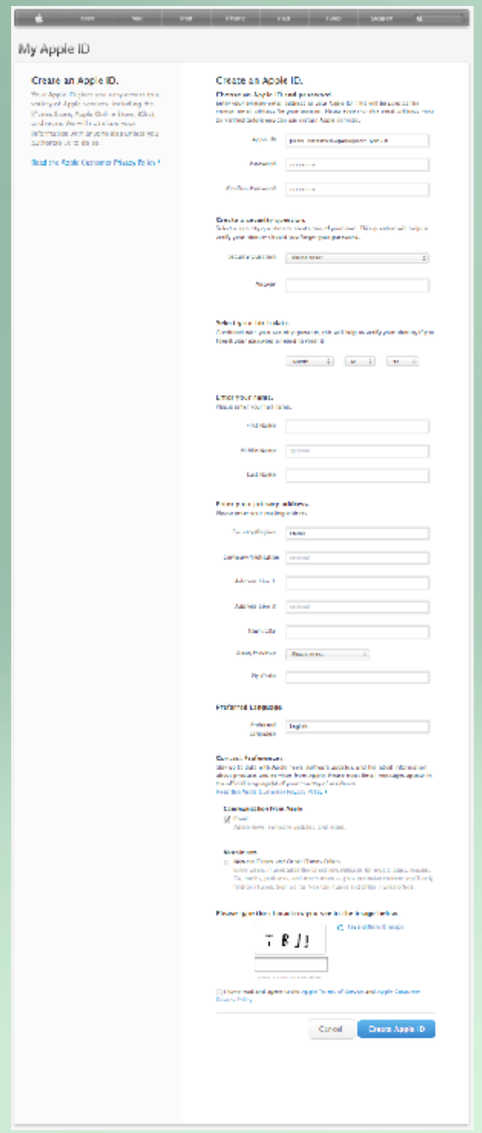


Web



iTunes

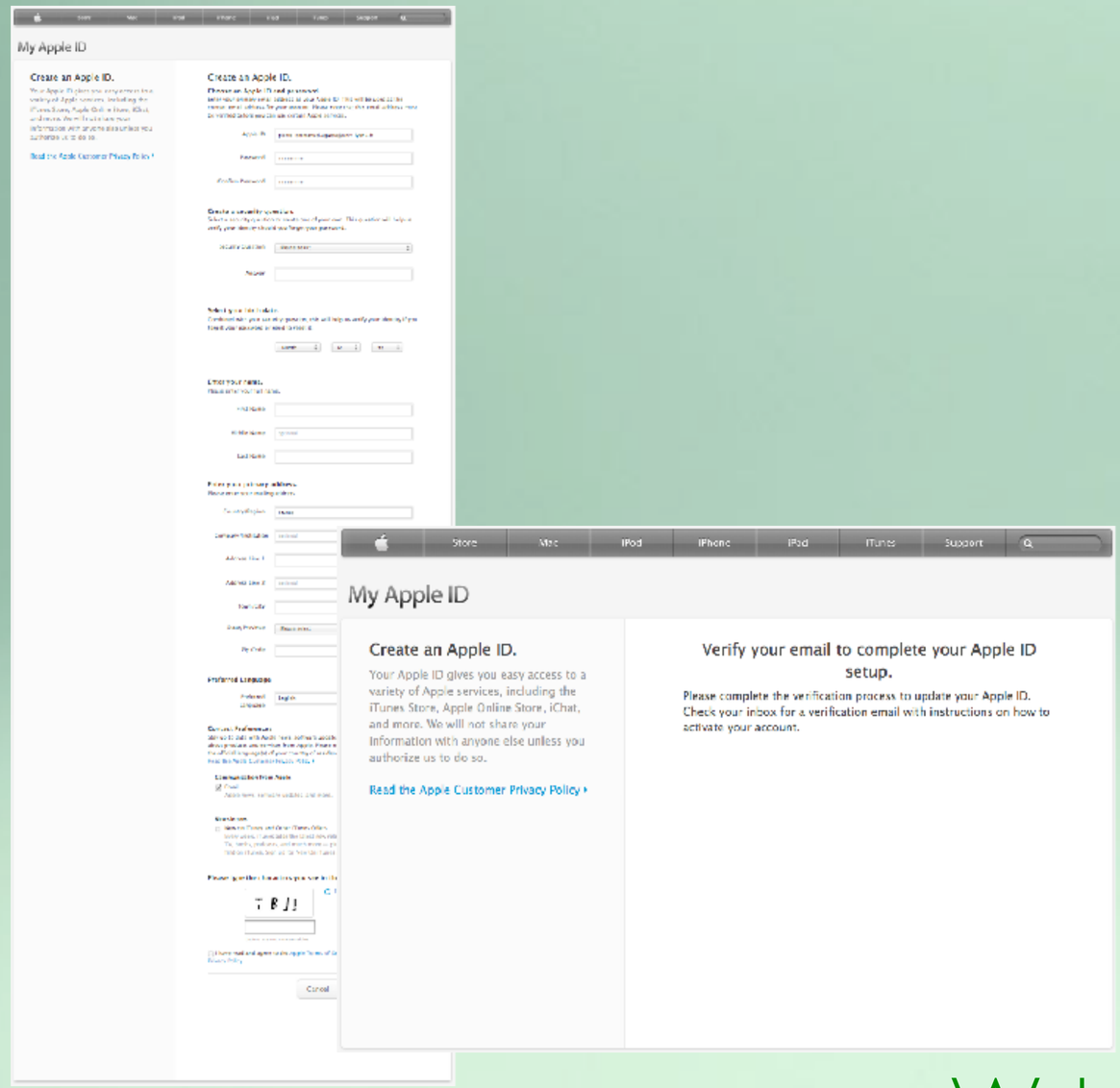
CRÉATION D'UN APPLE ID (2)



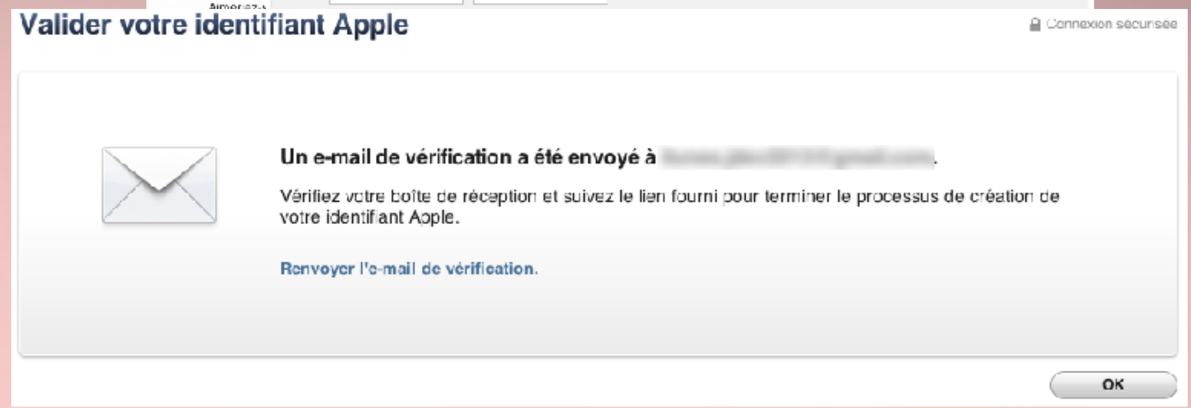
Web

iTunes

CRÉATION D'UN APPLE ID (2)



Web



iTunes

CRÉATION D'UN APPLE ID (3)

CRÉATION D'UN APPLE ID (3)

Chèr(e) Pierre-Emmanuel Aguera,

Vous avez saisi [redacted] comme adresse électronique de contact pour votre identifiant Apple ID. Pour terminer le processus, nous devons vérifier qu'il s'agit bien de votre adresse électronique. Cliquez simplement sur le lien ci-dessous et ouvrez une session à l'aide de votre Apple ID et de votre mot de passe.

Vérifiez maintenant >

Pourquoi ce courrier électronique vous a-t-il été envoyé ?
L'envoi de ce courrier électronique s'applique lorsqu'une personne ajoute ou modifie une adresse électronique de contact pour un compte Apple ID. Si cela ne vous concerne pas, ne vous inquiétez pas. Personne ne peut utiliser votre adresse électronique comme adresse de contact pour un Apple ID sans votre vérification.

Pour plus d'informations, consultez la rubrique Questions et réponses.

Merci,
L'assistance à la clientèle Apple

CRÉATION D'UN APPLE ID (3)

Chèr(e) Pierre-Emmanuel Aguera,

Vous avez saisi
votre identifiant Apple ID. Pour ter
s'agit bien de votre adresse électr
et ouvrez une session à l'aide de v

Vérifiez maintenant >

Pourquoi ce courrier électroniqu
L'envoi de ce courrier électronique
modifie une adresse électronique
vous concerne pas, ne vous inquié
électronique comme adresse de cc

Pour plus d'informations, consulte

Merci,
L'assistance à la clientèle Apple

TM et copyright © 2013 Apple Inc. Apple Sale
Numéro d'immatriculation de la société : 157
Tous droits réservés / Tenez-moi informé / E

Si vous ne souhaitez plus recevoir d'e-mails d
cliquer ici.

Store Mac iPod iPhone iPad iTunes Assistance

Mon identifiant Apple

Vérifiez votre adresse électronique.

Veuillez vérifier l'adresse électronique associée à votre identifiant Apple.

Connectez-vous pour vérifier votre adresse électronique.

Pour vérifier l'adresse électronique , veuillez vous connecter avec votre identifiant Apple et votre mot de passe.

Identifiant Apple

Vous avez oublié votre identifiant Apple ?

Mot de passe

Vous avez oublié votre mot de passe ?

Vérifier l'adresse

CRÉATION D'UN APPLE ID (3)

Chèr(e) Pierre-Emmanuel Aguera,

Vous avez saisi
votre identifiant Apple ID. Pour ter
s'agit bien de votre adresse électr
et ouvrez une session à l'aide de v

Vérifiez maintenant >

Pourquoi ce courrier électroniqu
L'envoi de ce courrier électronique
modifie une adresse électronique
vous concerne pas, ne vous inquié
électronique comme adresse de cc

Pour plus d'informations, consulte

Merci,
L'assistance à la clientèle Apple

TM et copyright © 2013 Apple Inc. Apple Sale
Numéro d'immatriculation de la société : 157
Tous droits réservés / Tenez-moi informé / E

Si vous ne souhaitez plus recevoir d'e-mails d
cliquer ici.

Store Mac iPod iPhone iPad iTunes Assistance

Mon identifiant Apple

Vérifiez votre adresse électronique.

Veuillez vérifier l'adresse électronique associée à votre identifiant Apple.

Store Mac iPod iPhone iPad iTunes Assistance

Mon identifiant Apple

Vérifiez votre adresse électronique.

Adresse électronique vérifiée.

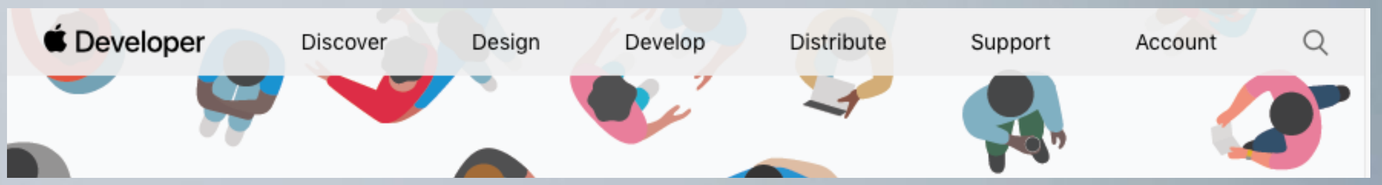
Merci d'avoir vérifié votre adresse. [redacted] est maintenant associé à votre identifiant Apple.

Vérification de l'adresse électronique

CRÉATION D'UN COMPTE DÉVELOPPEUR (1)

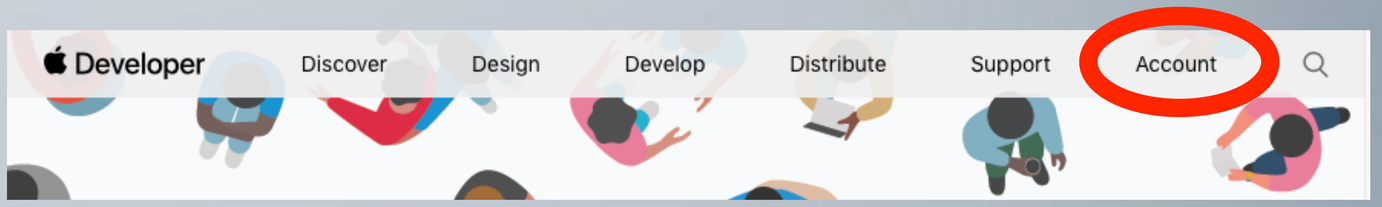
CREATION D'UN COMPTE DÉVELOPPEUR (1)

- https://developer.apple.com



CREATION D'UN COMPTE DÉVELOPPEUR (1)

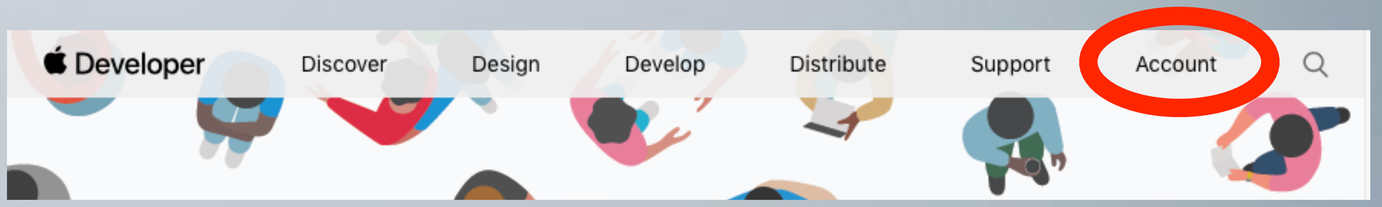
- https://developer.apple.com



- Account

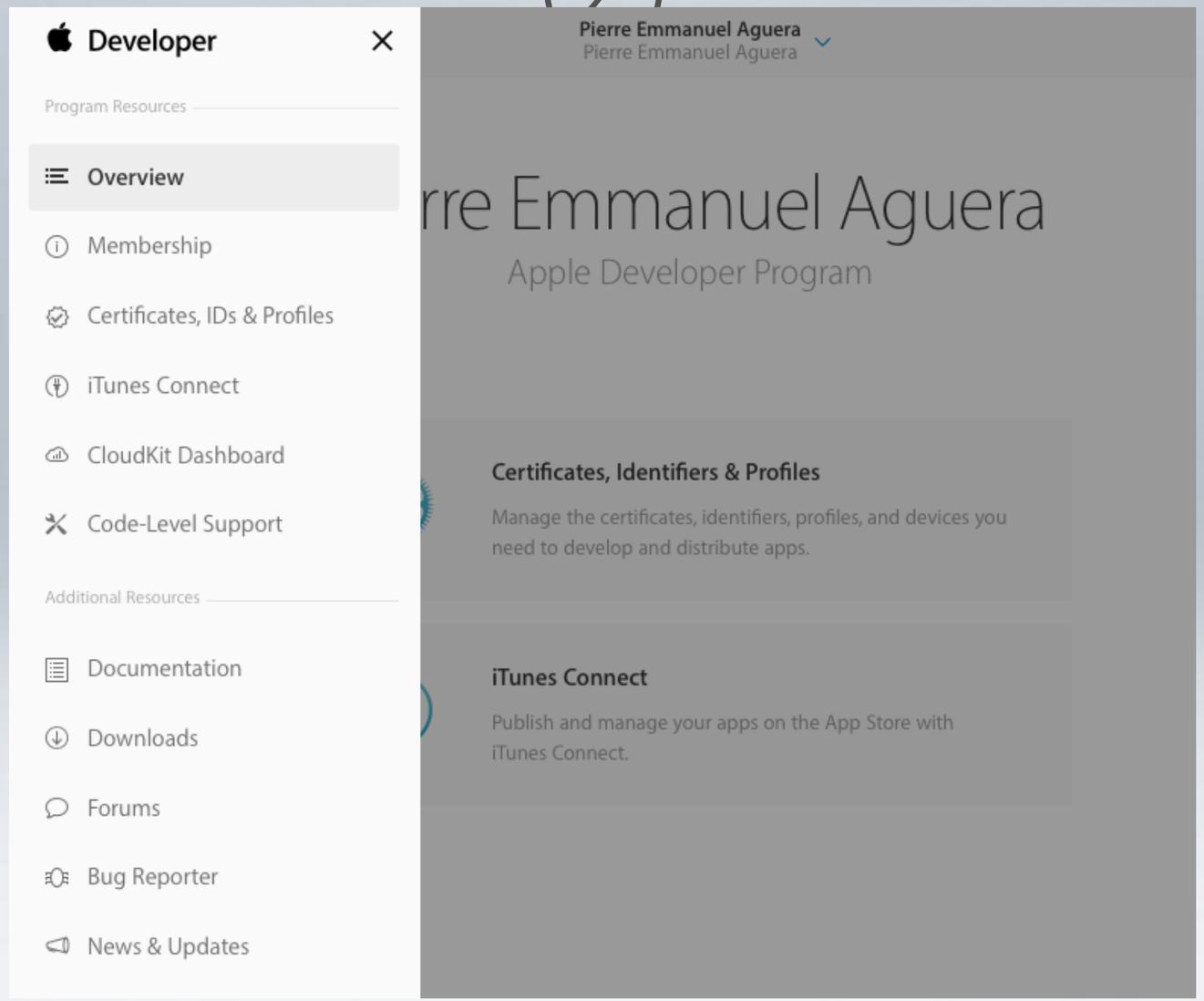
CREATION D'UN COMPTE DÉVELOPPEUR (1)

- https://developer.apple.com



- Account
- Utilisation de l'AppleID

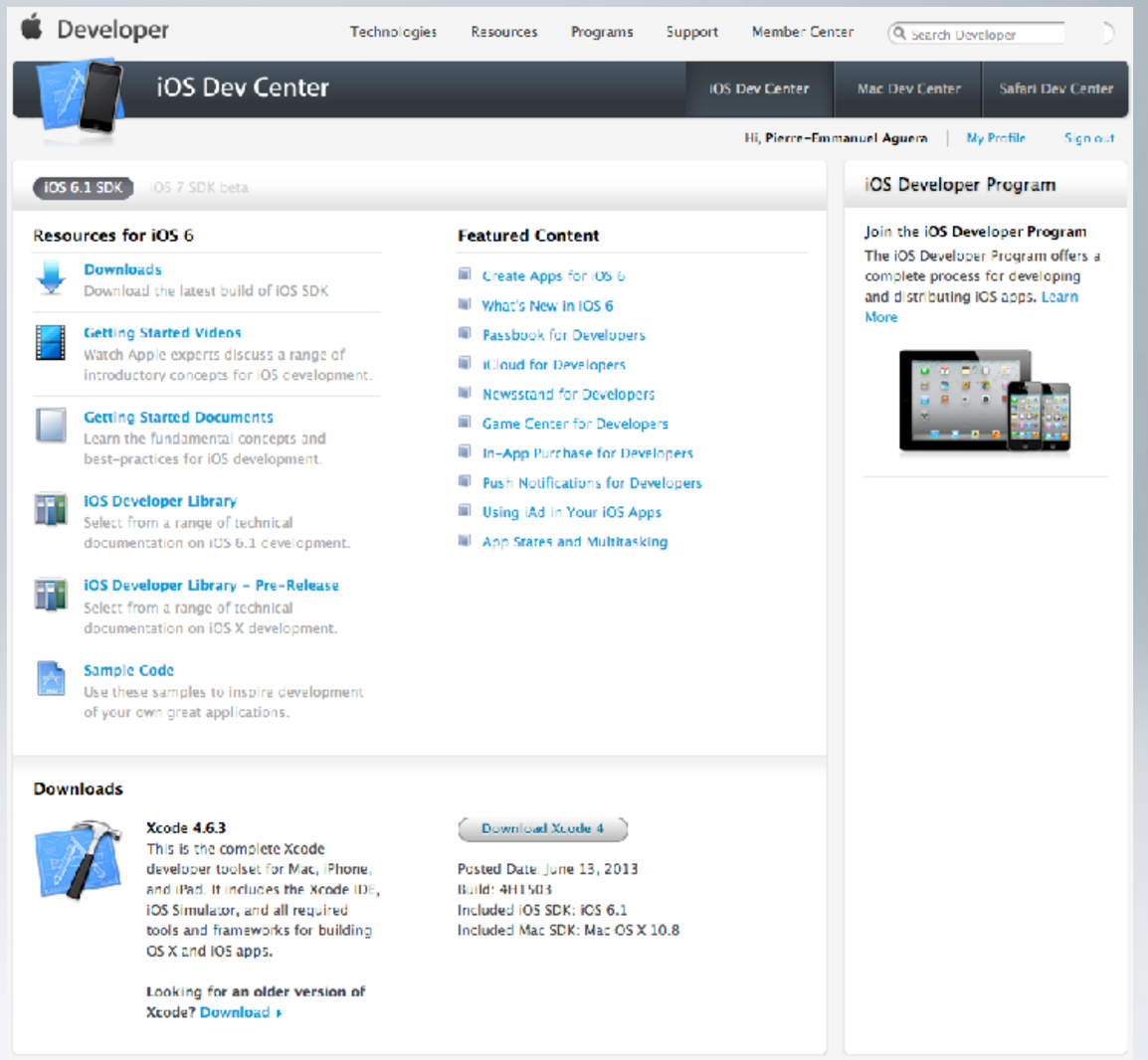
CREATION D'UN COMPTE DÉVELOPPEUR (2)



INSTALLATION DE XCODE

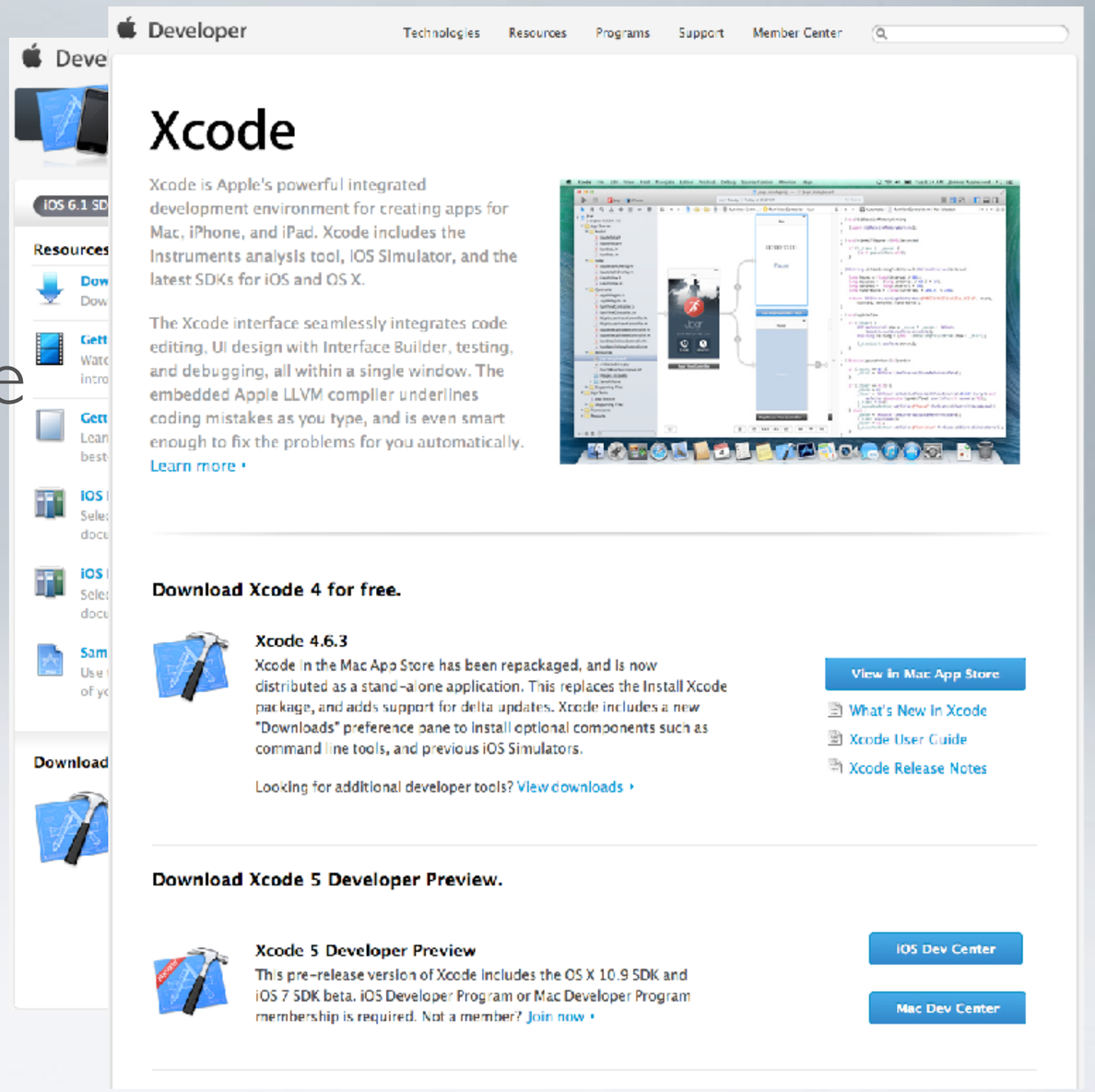
INSTALLATION DE XCODE

- Download XCode



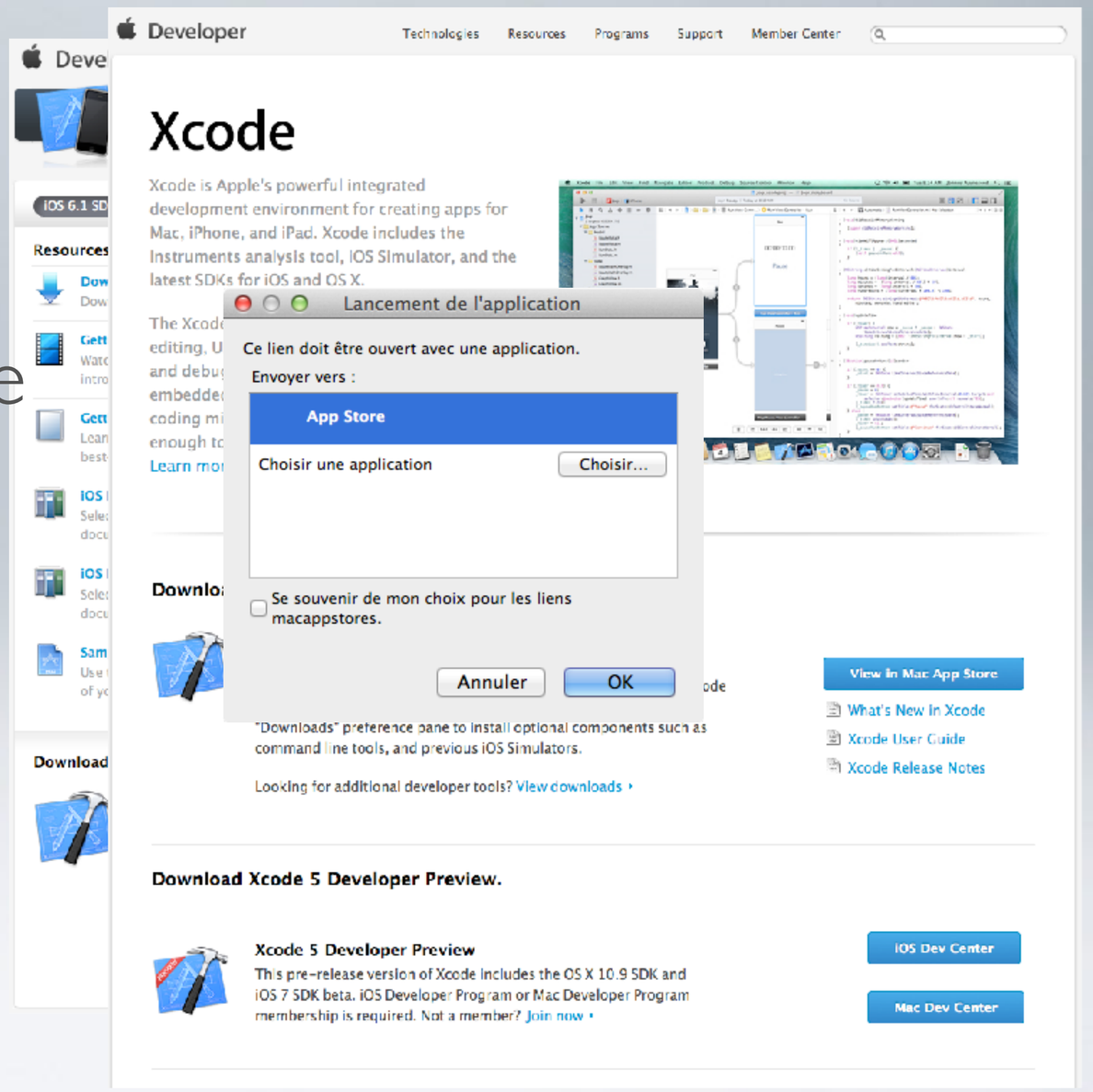
INSTALLATION DE XCODE

- Download XCode
- Renvoi vers Mac App Store



INSTALLATION DE XCODE

- Download XCode
- Renvoi vers Mac App Store
- Ouvrir avec l'App Store



INSTALLATION DE XCODE

- Download XCode
- Renvoi vers Mac App Store
- Ouvrir avec l'App Store
- Cliquer sur Gratuit

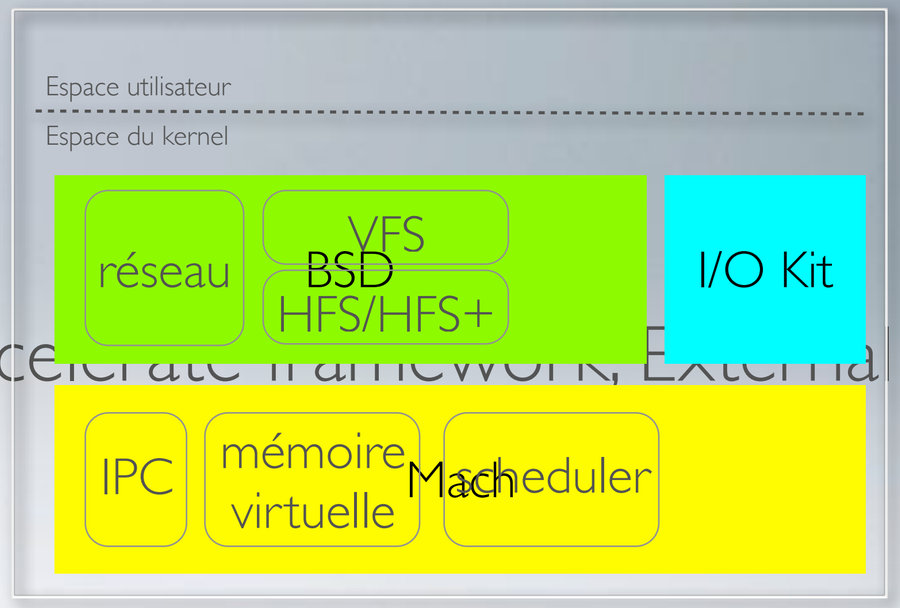
INSTALLATION DE XCODE

- Download XCode
- Renvoi vers Mac App Store
- Ouvrir avec l'App Store
- Cliquer sur Gratuit
- Et installer l'app

CARACTÉRISTIQUES LOGICIELLES

Kernel

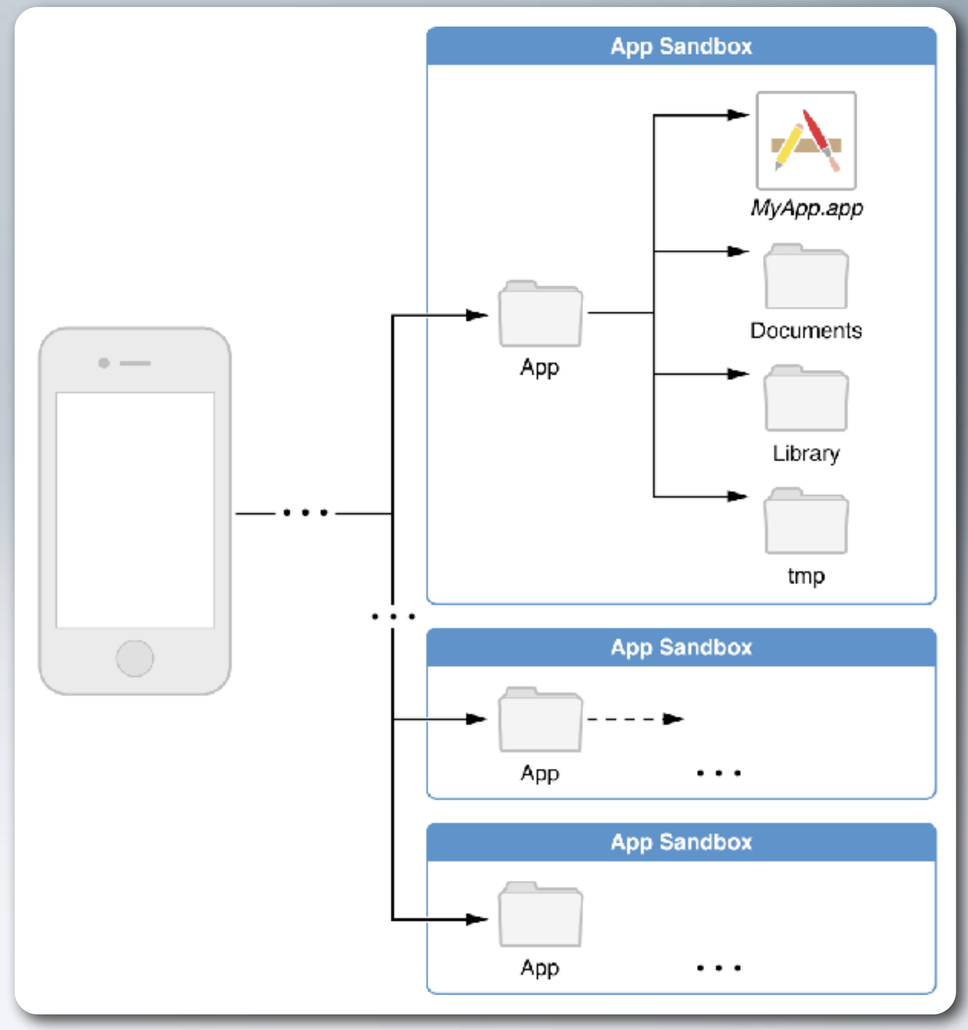
- iOS dérivé de Mac OS X
- Coeur Darwin
- Kernel XNU (X is Not UNIX)
- API spécifiques (SpringBoard, Cocoa Touch, Accelerate framework, External Accessory Framework, ...)



APPLICATIONS

Limitations

- Mémoire physique limitée
  - ➡ pas de swap
  - ➡ processus de libération de la mémoire
- Gestion de l'énergie :
  - ➡ systèmes de veille
  - ➡ multitâche
- Performances matérielles limitées



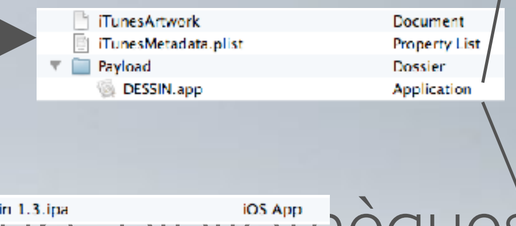
CARACTÉRISTIQUES LOGICIELLES

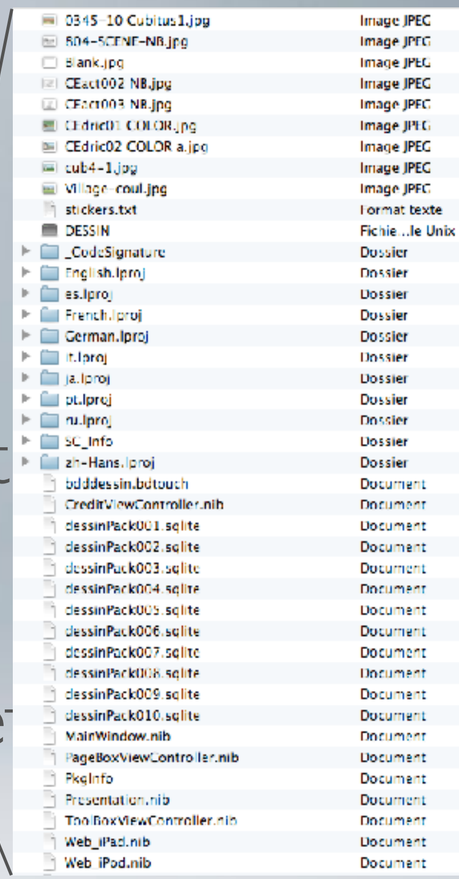
Bundles, applications et frameworks

- Bundle : structure hiérarchique standardisée contenant du code et les ressources utilisées par ce code

- Application : packagée dans un bundle contenant l'exécutable et ressources

- Framework : bundle contenant une ou plusieurs bibliothèques et fichiers






```
CodeResources/        Symbolic link to Code Signature/CodeResources plist
    Headers/          Symbolic link to Miscellaneous .h files provided by this
framework
    Resources/        .nib files (GUI), .lproj files, or other files required by
framework
    Versions/         Subdirectory to allow versioning
        A/            Letter directories denoting version of this framework
        Current/      Symbolic link to preferred framework version
    Framework -name   Symbolic link to framework binary, in preferred version
```

LICENCES DE DÉVELOPPEMENT

Programmes

https://developer.apple.com/support/compare-memberships/

Benefits and Resources

	Sign in with Apple ID	Individual	Organization	Enterprise Program
Xcode Developer Tools	●	●	●	●
Xcode Beta Releases	●	●	●	●
Developer Forums	●	●	●	●
Bug Reporter	●	●	●	●
Test on Device	●	●	●	●
Beta OS Releases		●	●	●
Advanced App Capabilities		●	●	●
App Store Distribution		●	●	
In-house App Distribution				●
Safari Extensions		●	●	
Developer ID		●	●	●
Technical Support Incidents		●	●	●
Team Management			●	●
TestFlight Beta Testing		●	●	
App Analytics		●	●	
Cost	Free	99 USD*	99 USD*	299 USD**
Requirement	13+	18+	DUNS Number	DUNS Number

Introduction

RÔLE

https://developer.apple.com/support/roles/

	Agent	Admin	Member
Accept Legal Agreements	●		
Renew Membership	●		
Create Developer ID Certificates	●		
Invite Members and Assign Roles (organization only)	●	●	
Create Provisioning Profiles	●	●	○
Approve Certificate Signing Requests	●	●	
Create and Remove Development Certificates	●	●	●
Add and Disable UDIDs	●	●	○
Register and Configure App IDs	●	●	○
Delete App IDs	●	●	
Create an iOS Distribution Certificate and Distribution Provisioning Profiles	●	●	
Create Certificates for Apple Push Notification service and Pass Type IDs	●	●	
Create Mac App Distribution and Mac Installer Distribution Certificates	●	●	
Purchase and Submit Technical Support Incidents	●	●	●
Post in Developer Forums	●	●	●
Download Beta Software	●	●	●
Download Provisioning Profiles	●	●	●
Submit Certificate Signing Requests	●	●	●
Create Safari Extension Certificates	●	●	●
Submit Safari Extensions to the Safari Extensions Gallery	●	●	●

○ Access requires Xcode 7 or later.

DISTRIBUTION D'APPLICATION

Modes de distribution

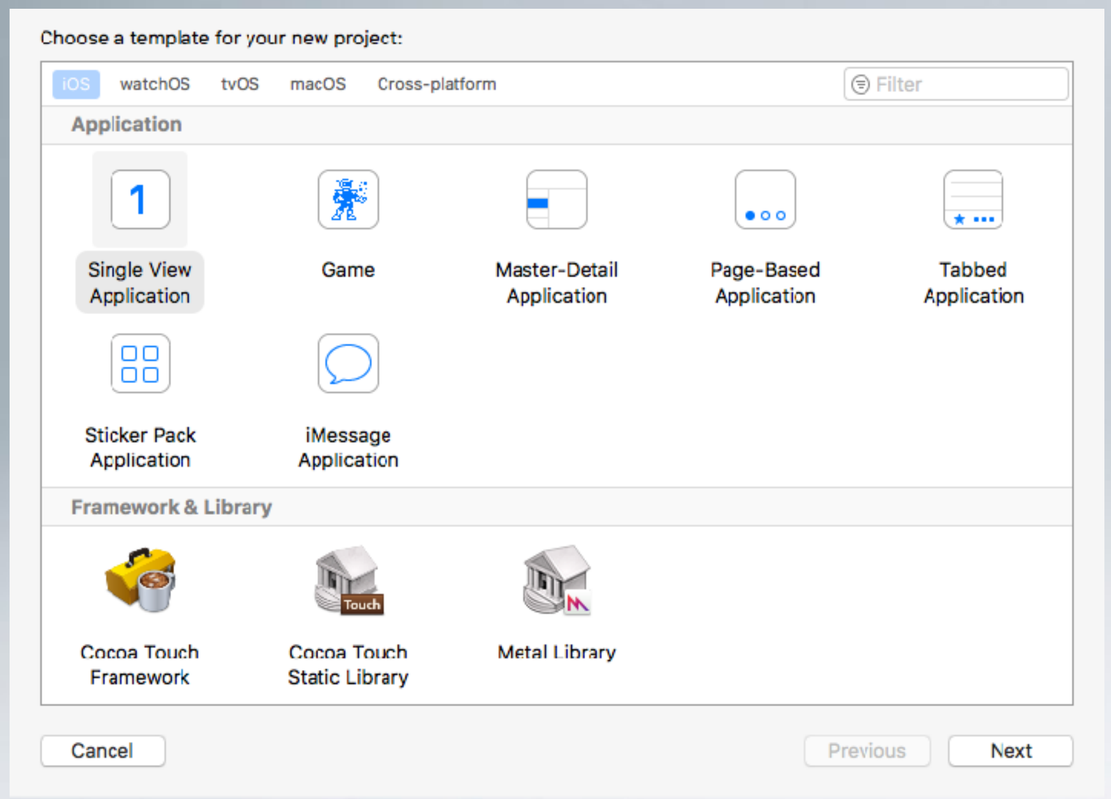
- 4 façons de distribuer les applications
  - AppStore
  - B2B
  - Ad Hoc
  - In House

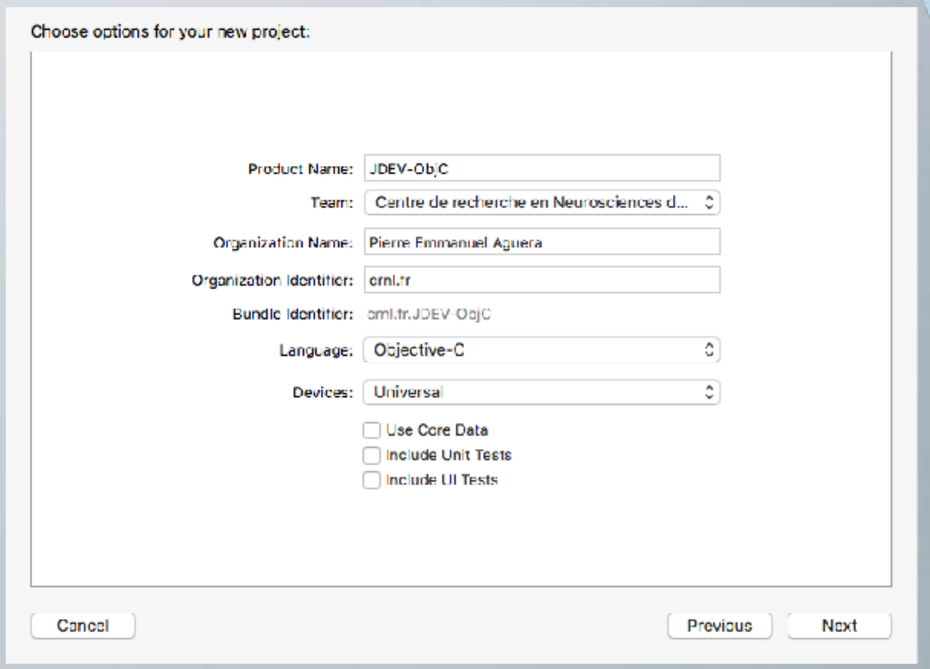
LANCEMENT

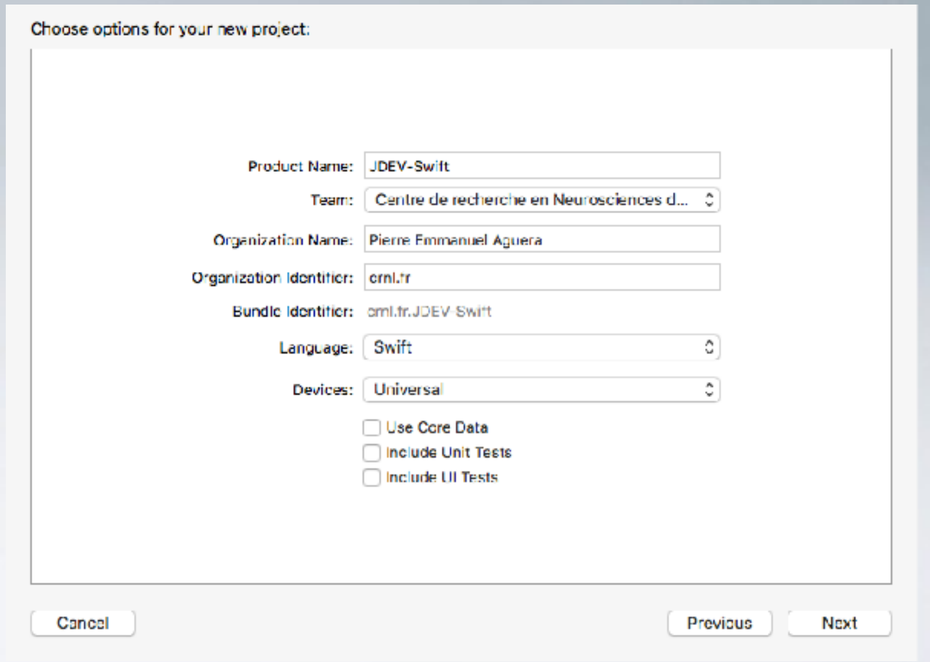

Welcome to Xcode
Version 8.3.3 (8E3004b)
Get started with a playground
Explore new ideas quickly and easily.
Create a new Xcode project
Create an app for iPhone, iPad, Mac, Apple Watch or Apple TV.
Check out an existing project
Start working on something from an SCM repository.
Show this window when Xcode launches
Arduino_Servo
~/Downloads/Arduino_Servo_Starter-Swift3
TestBeeWi
~/Documents/Devel/Apple/ios
Questionnaire
~/Documents/Devel/Apple/ios
MyPlayground
~/Documents/Devel/Apple/ios
LocaStims
~/Documents/Devel/Apple/ios
LocaStims
...uments/Devel/Apple/ios/LocaStims.Master
LocaStims
...uments/Devel/Apple/ios/LocaStims.Master
Screen&Touch Time
.../ios/Screen&Touch Time/ScreenTouchTime
Stabilo
~/Documents/Devel/Apple/ios
Open another project...

Environnement de développement

CRÉATION







CRÉATION PROJET

navigateur

éditeur

inspecteur

bibliothèques

debugger

navigateur

éditeur

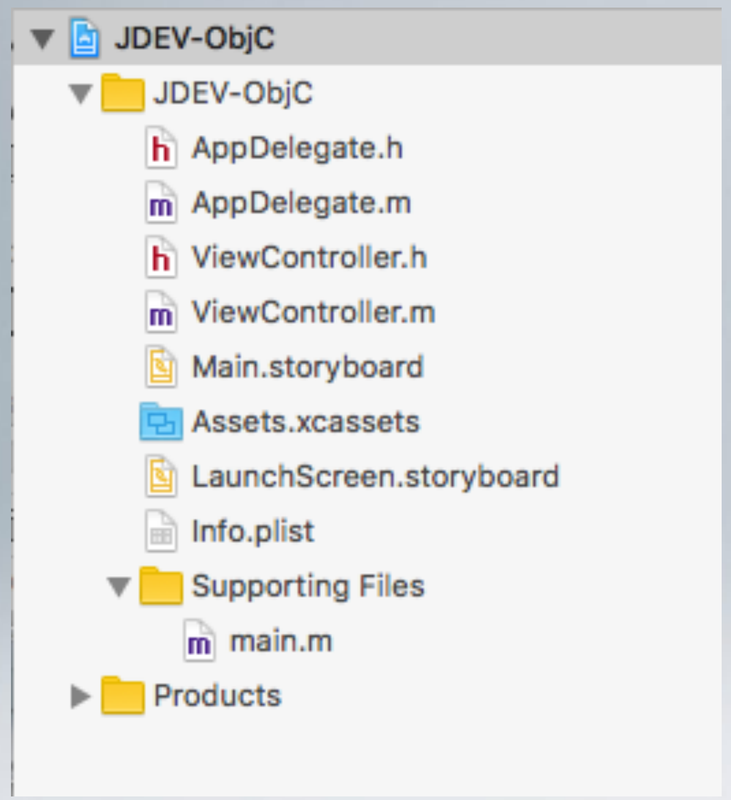
inspecteur

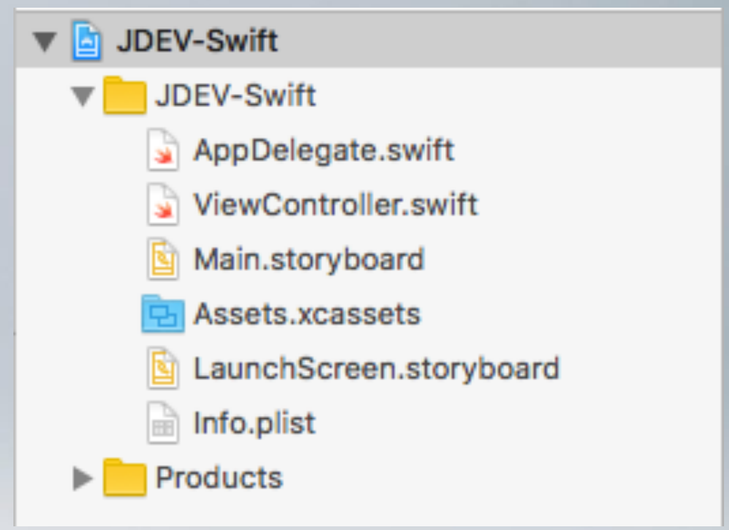
bibliothèques

debugger

[ObjC]

DESCRIPTION





Langages

1983

basé sur C et Smalltalk

orienté objet

v3.1

première version 2014
version 4.0 en développement

Objective-C, C#, CLU,

Haskell, Python, Racket,
Ruby, Rust et langage D

ELÉMENTS DE SYNTAXE

- Surcouche du C
- Objets Objective-C sont référencés par des pointeurs
- Certaines classes pour objets mutables ou non (NSMutableString, NSString, NSMutableArray, NSArray, …)
- Chaîne préfixées par @ (@"ma chaîne") (classe NSString)
- Déclaration de méthode:

  - (int)maMethode:(int)n;
- appel de méthode:

  [monObjet methode:param];

- Langage fonctionnel et objet
- Pas de séparateur d'instruction (comme ; en C)
- Types de base: String, Int, Float, Double, Bool
- 2 modes pour les variables: immuable (let) ou modifiable (var)
  - let a=2 var b=3
- Déclaration de méthode:
  - func maMethode(n:Int)->Int
  - func maMethode()
- Appel de méthode:

  monObjet.maMethode(param)

LES CLASSES

Création:
(alloc) init

Destruction:
dealloc

Propriétés:
getter et setter
automatiquement générés
(setter = setProperty, getter = property)

Variable d'instance
accède par préfixe _

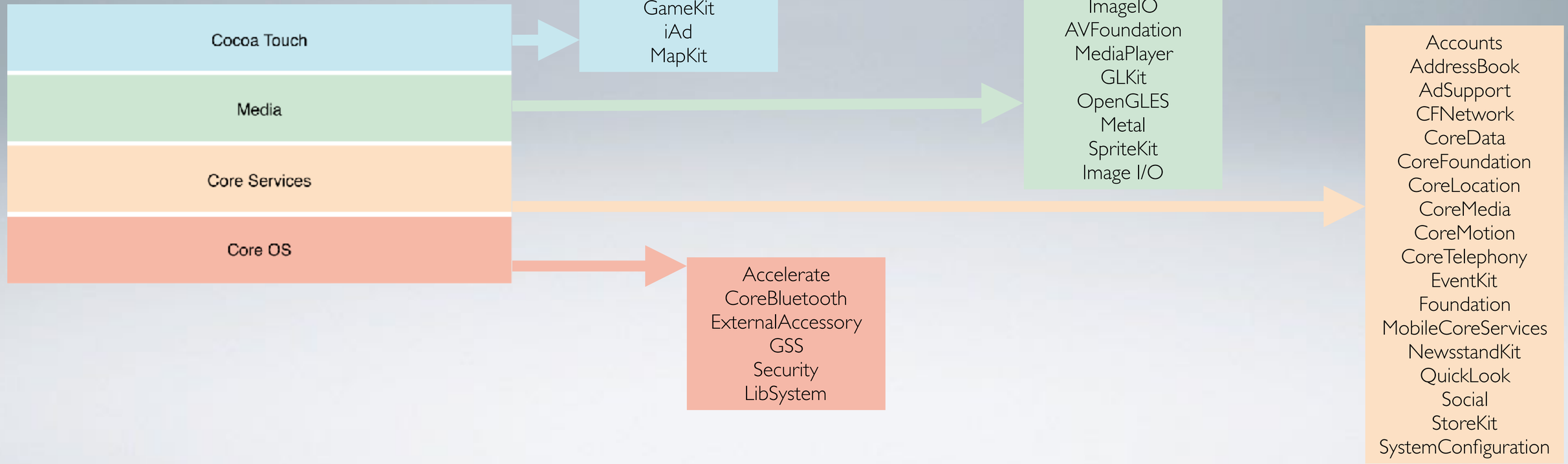
[ObjC]

Création:
init

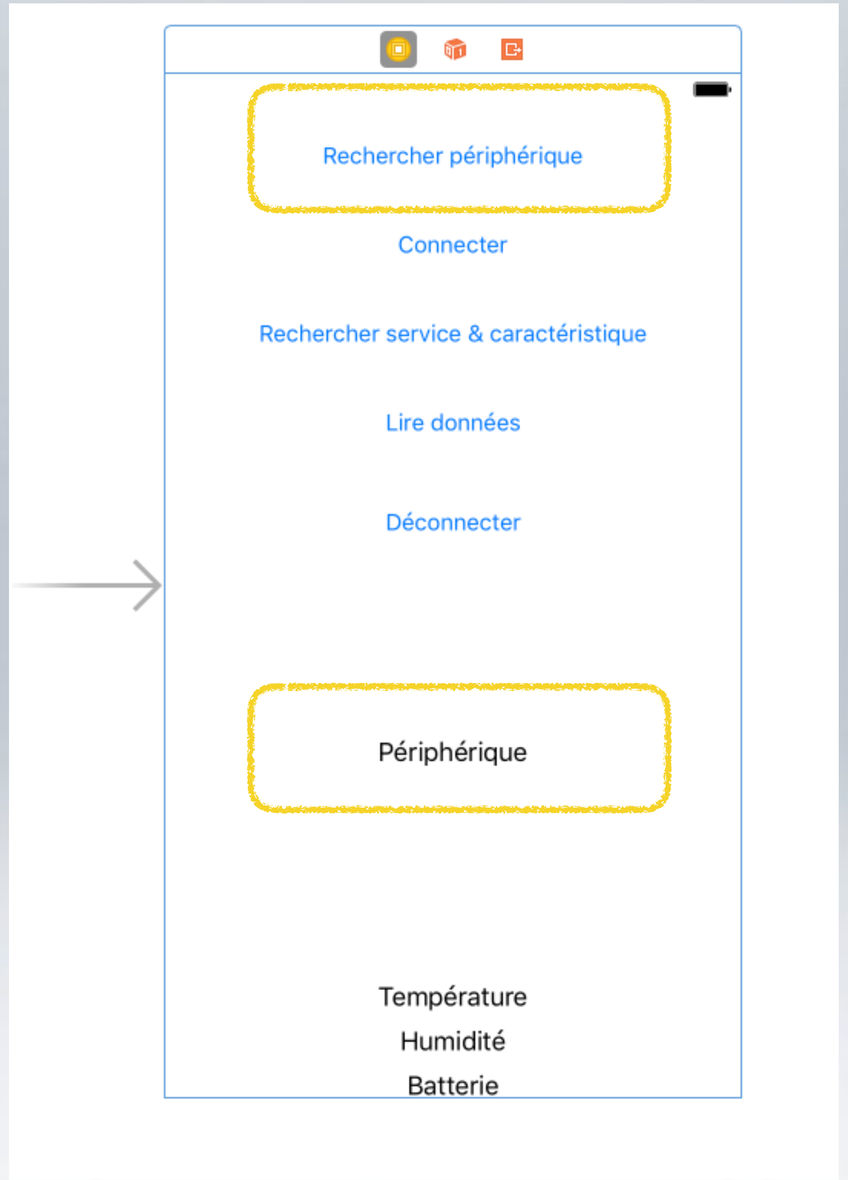
Destruction:
deinit

Propriétés:
accès direct

FRAMEWORKS


Cocoa Touch
Media
Core Services
Core OS
UIKit
AddressBookUI
EventKitUI
MessageUI
GameKit
iAd
MapKit
CoreAudio
AudioToolbox
AudioUnit
CoreMidi
MediaToolbox
OpenAL
QuartzCore
CoreGraphics
CoreImage
CoreText
CoreVideo
ImageIO
AVFoundation
MediaPlayer
GLKit
OpenGLES
Metal
SpriteKit
Image I/O
Accounts
AddressBook
AdSupport
CFNetwork
CoreData
CoreFoundation
CoreLocation
CoreMedia
CoreMotion
CoreTelephony
EventKit
Foundation
MobileCoreServices
NewsstandKit
QuickLook
Social
StoreKit
SystemConfiguration
Accelerate
CoreBluetooth
ExternalAccessory
GSS
Security
LibSystem

Interface


Rechercher périphérique
Connecter
Rechercher service & caractéristique
Lire données
Déconnecter
Périphérique
Température
Humidité
Batterie

Code

```objc
//
//  ViewController.m
//  JDEV-ObjC
//
//  Created by Pierre Emmanuel Aguera on 23/06/2017.
//  Copyright © 2017 Pierre Emmanuel Aguera. All rights reserved.
//

#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
 [super viewDidLoad];
 // Do any additional setup after loading the view, typically from a
nib.
}


- (void)didReceiveMemoryWarning {
 [super didReceiveMemoryWarning];
 // Dispose of any resources that can be recreated.
}


@end
```

```swift
//
//  ViewController.swift
//  JDEV-Swift
//
//  Created by Pierre Emmanuel Aguera on 23/06/2017.
//  Copyright © 2017 Pierre Emmanuel Aguera. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

 override func viewDidLoad() {
  super.viewDidLoad()
  // Do any additional setup after loading the view, typically from a
nib.
 }

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


}
```

[ObjC]

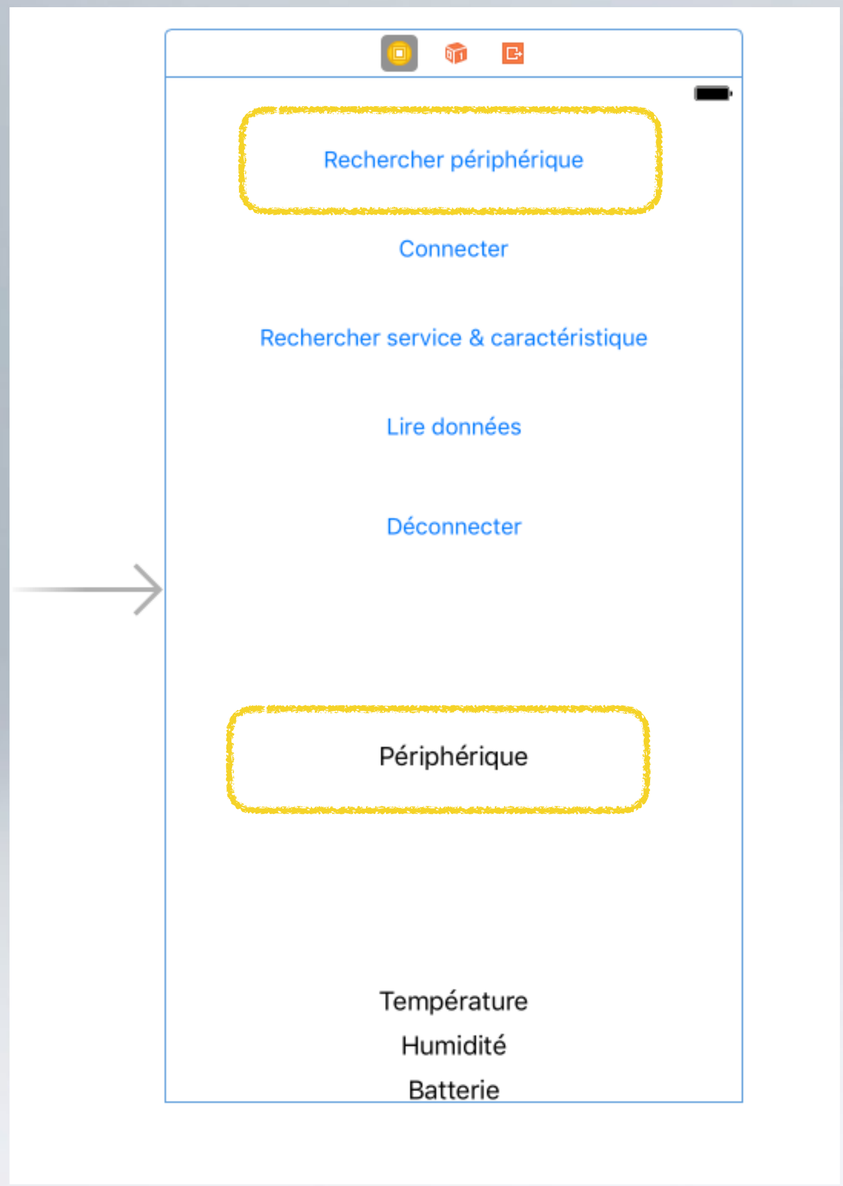
Code

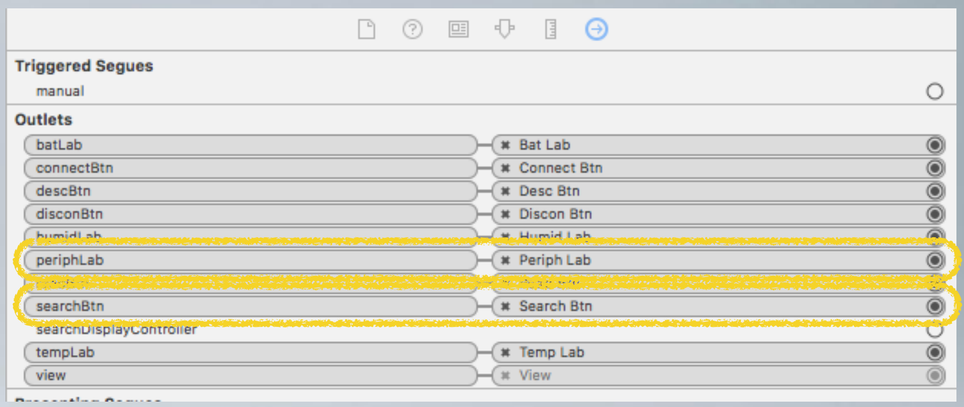
```objc
//
//  ViewController.m
//  JDEV-ObjC
//
//  Created by Pierre Emmanuel Aguera on 23/06/2017.
//  Copyright © 2017 Pierre Emmanuel Aguera. All rights reserved.
//

#import "ViewController.h"

#define NO_PERIPH @"Pas de périphérique"

@interface ViewController ()
@property IBOutlet UIButton *searchBtn;
@property IBOutlet UILabel *periphLab;
@property NSString *periphName;

@end

@implementation ViewController

- (id)initWithCoder:(NSCoder *)aDecoder {
 self = [super initWithCoder:aDecoder];

 _periphName = NO_PERIPH;
 return self;
}

- (void)viewDidLoad {
 [super viewDidLoad];
 // Do any additional setup after loading the view, typically from a
nib.
}


- (void)didReceiveMemoryWarning {
 [super didReceiveMemoryWarning];
 // Dispose of any resources that can be recreated.
}


@end
```

[ObjC]

```swift
//
//  ViewController.swift
//  JDEV-Swift
//
//  Created by Pierre Emmanuel Aguera on 23/06/2017.
//  Copyright © 2017 Pierre Emmanuel Aguera. All rights reserved.
//

import UIKit

class ViewController: UIViewController {
 @IBOutlet var searchBtn: UIButton?
 @IBOutlet var periphLab: UILabel?
 private var periphName: String

 private let NO_PERIPH = "Pas de périphérique"

 required init?(coder aDecoder: NSCoder) {

  periphName = NO_PERIPH;

  super.init(coder: aDecoder)
 }

 override func viewDidLoad() {
  super.viewDidLoad()
  // Do any additional setup after loading the view, typically from a
nib.
 }

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


}
```

Interface


Rechercher périphérique
Connecter
Rechercher service & caractéristique
Lire données
Déconnecter
Périphérique
Température
Humidité
Batterie


Triggered Segues
manual
Outlets
batLab
Bat Lab
connectBtn
Connect Btn
descBtn
Desc Btn
disconBtn
Discon Btn
periphLab
Periph Lab
searchBtn
Search Btn
searchDisplayController
tempLab
Temp Lab
view
View

Code

```objc
typedef enum {UndefinedState} appState_t;
```

```objc
@property appState_t appState;
```

```objc
_appState = UndefinedState;
```

```objc
- (void)updateInterface {
 switch (_appState) {
  case UndefinedState:
   [_searchBtn setEnabled:YES];
   _periphName = @"Pas de périphérique";
   [_periphLab setText:_periphName];
   [_periphLab setTextColor:[UIColor darkGrayColor]];
   break;
 }
}
```

```swift
enum AppState {case undefinedState}
```

```swift
private var appState: AppState
```

```swift
appState = .undefinedState;
```

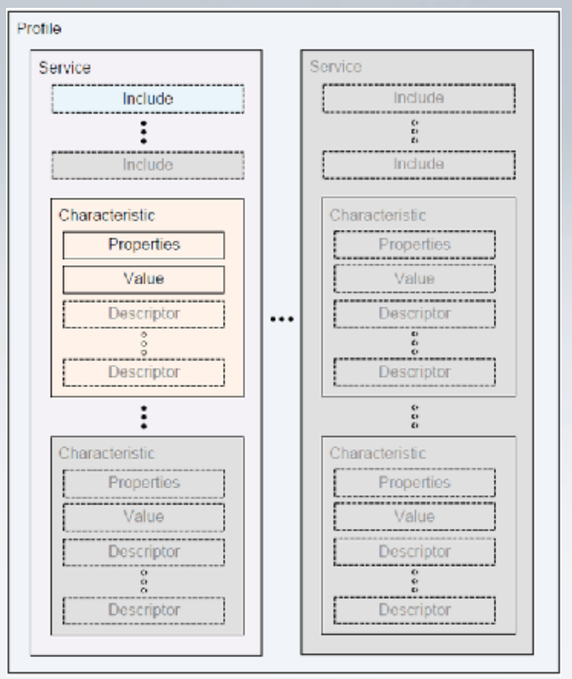
```swift
func updateInterface() {
 switch appState {
 case .undefinedState,
  searchBtn?.isEnabled = true
  periphName = "Pas de périphérique"
  periphLab?.text = periphName
  periphLab?.textColor = UIColor.darkGray
 }
}
```

Bluetooth Low Energy

https://www.bluetooth.com/specifications/gatt/services

Central

Périphérique



Code

Controller.h

```objc
@import CoreBluetooth;

@interface ViewController : UIViewController <CBCentralManagerDelegate>
```

Controller.m

```objc
@property CBCentralManager *centralManager;

_centralManager = [[CBCentralManager alloc] initWithDelegate:self
queue:nil];

#pragma mark - CBCentralManagerDelegate
- (void)centralManager:(CBCentralManager *)central
didConnectPeripheral:(CBPeripheral *)peripheral {
 NSLog(@"Central manager did connect to peripheral");
}

- (void)centralManager:(CBCentralManager *)central
didFailToConnectPeripheral:(CBPeripheral *)peripheral error:(NSError
*)error {
 NSLog(@"Central manager failed to connect to peripheral");
}


- (void)centralManager:(CBCentralManager *)central
didDisconnectPeripheral:(CBPeripheral *)peripheral error:(NSError
*)error {
 NSLog(@"Central manager did disconnect from peripheral");
}

- (void)centralManager:(CBCentralManager *)central
didDiscoverPeripheral:(CBPeripheral *)peripheral advertisementData:
(NSDictionary *)advertisementData RSSI:(NSNumber *)RSSI {
}

// method called whenever the device state changes.
- (void)centralManagerDidUpdateState:(CBCentralManager *)central {
}
```

1

```swift
import CoreBluetooth
```

4

```swift
class ViewController: UIViewController, CBCentralManagerDelegate {
```

2

```swift
 private var centralManager: CBCentralManager?
```

3

```swift
centralManager = CBCentralManager.init(delegate:self, queue:nil)
```

5

```swift
 // MARK: - CBCentralManagerDelegate
 func centralManager(_ central: CBCentralManager, didConnect
peripheral: CBPeripheral) {
   print("Central manager did connect to peripheral")
 }

 func centralManager(_ central: CBCentralManager,
didDisconnectPeripheral peripheral: CBPeripheral, error: Error?) {
   print("Central manager did disconnect from peripheral")
 }

   func centralManager(_ central: CBCentralManager, didDiscover
peripheral: CBPeripheral, advertisementData: [String : Any], rssi RSSI:
NSNumber) {
 }

   func centralManagerDidUpdateState(_ central: CBCentralManager) {
 }
```

[ObjC]

Protocol

CBPeripheralDelegate

Language

Swift
Objective-C

SDKs

iOS 5.0+
macOS 10.7+
tvOS 9.0+

On This Page

The delegate of a CBPeripheral object must adopt the CBPeripheralDelegate protocol. The delegate uses this protocol's methods to monitor the discovery, exploration, and interaction of a remote peripheral's services and properties. There are no required methods in this protocol.

Symbols

Discovering Services

func peripheral(CBPeripheral, didDiscoverServices: Error?)
Invoked when you discover the peripheral's available services.
func peripheral(CBPeripheral, didDiscoverIncludedServicesFor: CBService, error: Error?)
Invoked when you discover the included services of a specified service.

Discovering Characteristics and Characteristic Descriptors

func peripheral(CBPeripheral, didDiscoverCharacteristicsFor: CBService, error: Error?)
Invoked when you discover the characteristics of a specified service.
func peripheral(CBPeripheral, didDiscoverDescriptorsFor: CBCharacteristic, error: Error?)
Invoked when you discover the descriptors of a specified characteristic.

Retrieving Characteristic and Characteristic Descriptor Values

func peripheral(CBPeripheral, didUpdateValueFor: CBCharacteristic, error: Error?)
Invoked when you retrieve a specified characteristic's value, or when the peripheral device notifies your app that the characteristic's value has changed.
func peripheral(CBPeripheral, didUpdateValueFor: CBDescriptor, error: Error?)
Invoked when you retrieve a specified characteristic descriptor's value.

Writing Characteristic and Characteristic Descriptor Values

func peripheral(CBPeripheral, didWriteValueFor: CBCharacteristic, error: Error?)
Invoked when you write data to a characteristic's value.
func peripheral(CBPeripheral, didWriteValueFor: CBDescriptor, error: Error?)
Invoked when you write data to a characteristic descriptor's value.

Managing Notifications for a Characteristic's Value

func peripheral(CBPeripheral, didUpdateNotificationStateFor: CBCharacteristic, error: Error?)
Invoked when the peripheral receives a request to start or stop providing notifications for a specified characteristic's value.

Retrieving a Peripheral's Received Signal Strength Indicator (RSSI) Data

func peripheralDidUpdateRSSI(CBPeripheral, error: Error?)
Invoked when you retrieve the value of the peripheral's current RSSI while it is connected to the central manager.
func peripheral(CBPeripheral, didReadRSSI: NSNumber, error: Error?)
Invoked after you call readRSSI() to retrieve the value of the peripheral's current RSSI while it is connected to the central manager.

Monitoring Changes to a Peripheral's Name or Services

func peripheralDidUpdateName(CBPeripheral)
Invoked when a peripheral's name changes.
func peripheral(CBPeripheral, didModifyServices: [CBService])
Invoked when a peripheral's services have changed.

Code

```objc
typedef enum {UndefinedState, InitializedState} appState_t;
```

```objc
- (void)updateInterface {
 switch (_appState) {
  case UndefinedState:
  case InitializedState:
   [_searchBtn setEnabled:YES];
   _periphName = @"Pas de périphérique";
   [_periphLab setText:_periphName];
   [_periphLab setTextColor:[UIColor darkGrayColor]];
   break;
 }
}
```

```objc
 switch ([central state]) {
  case CBManagerStatePoweredOn:
   NSLog(@"CoreBluetooth BLE hardware is powered on and ready");
   _appState = InitializedState;
   break;
  case  CBManagerStatePoweredOff:
   NSLog(@"CoreBluetooth BLE hardware is powered off");
   _appState = UndefinedState;
   break;
  case CBManagerStateUnauthorized:
   NSLog(@"CoreBluetooth BLE state is unauthorized");
   _appState = UndefinedState;
   break;
  case CBManagerStateUnknown:
   NSLog(@"CoreBluetooth BLE state is unknown");
   _appState = UndefinedState;
   break;
  case CBManagerStateUnsupported:
   NSLog(@"CoreBluetooth BLE hardware is unsupported on this platform");
   _appState = UndefinedState;
   break;
  case CBManagerStateResetting:
   NSLog(@"CoreBluetooth BLE hardware is resetting");
   _appState = UndefinedState;
   break;
 }
 // Update interface
 [self updateInterface];
```

2

```swift
 enum AppState {case undefinedState, initializedState}
```

3

```swift
 func updateInterface() {
  switch appState {
  case .undefinedState,
      .initializedState:
   searchBtn?.isEnabled = true
   periphName = "Pas de périphérique"
   periphLab?.text = periphName
   periphLab?.textColor = UIColor.darkGray
  }
}
```

```swift
  switch central.state {
  case .poweredOn:
   print("CoreBluetooth BLE hardware is powered on and ready")
   appState = .initializedState
  case .poweredOff:
   print("CoreBluetooth BLE hardware is powered off")
   appState = .undefinedState
  case .unauthorized:
   print("CoreBluetooth BLE state is unauthorized")
   appState = .undefinedState
  case .unknown:
   print("CoreBluetooth BLE state is unknown")
   appState = .undefinedState
  case .unsupported:
   print("CoreBluetooth BLE hardware is unsupported on this platform")
   appState = .undefinedState
  case .resetting:
   print("CoreBluetooth BLE hardware is resetting")
   appState = .undefinedState
  }
 // Update interface
 updateInterface()
```

[ObjC]

Code

```objc
typedef enum {UndefinedState, InitializedState, SearchingState}
appState_t;
```

2

```swift
enum AppState {case undefinedState, initializedState, searchingState}
```

```objc
  case SearchingState:
   [_searchBtn setEnabled:YES];
   [_connectBtn setEnabled:NO];
   [_descBtn setEnabled:NO];
   [_readBtn setEnabled:NO];
   [_disconBtn setEnabled:NO];

   [_periphLab setText:_periphName];
   [_periphLab setTextColor:[UIColor darkGrayColor]];
   break;
```

3

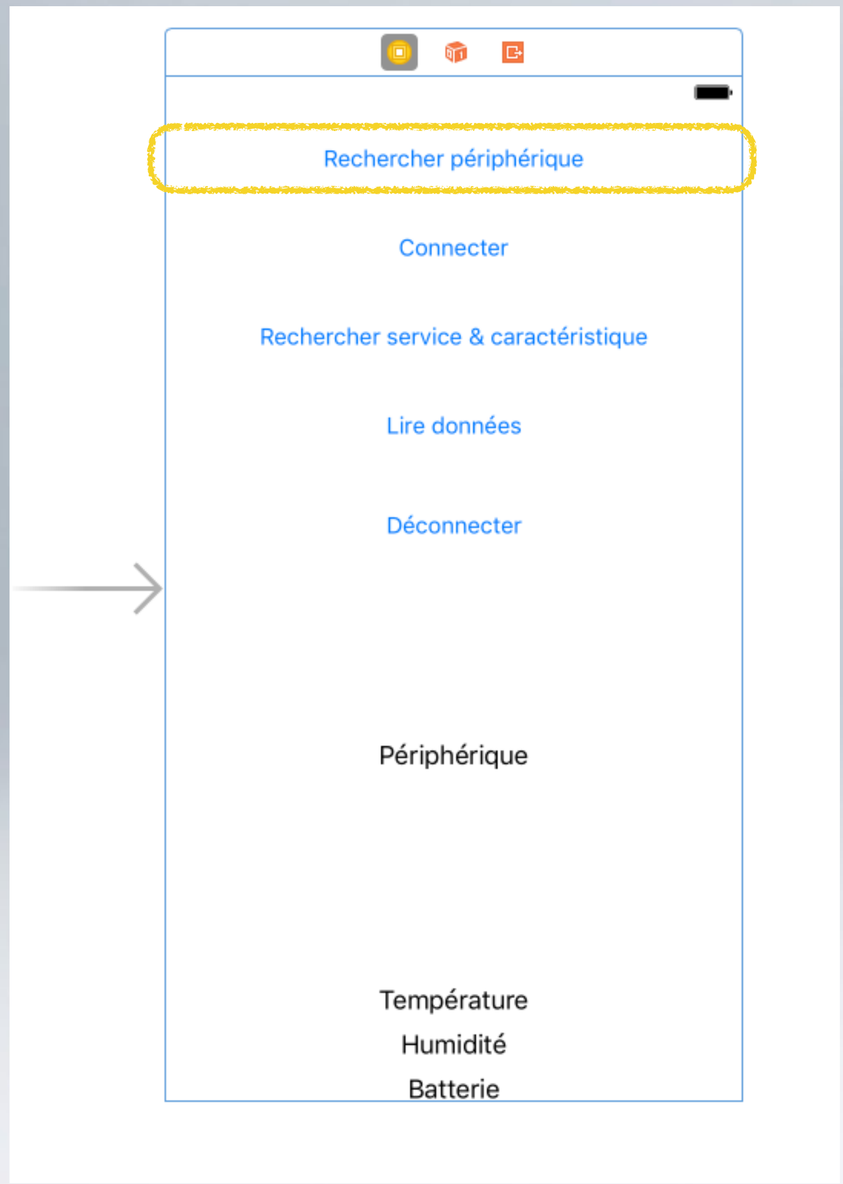
```swift
  case .searchingState:
   searchBtn?.isEnabled = true
   periphLab?.text = periphName
   periphLab?.textColor = UIColor.darkGray
```

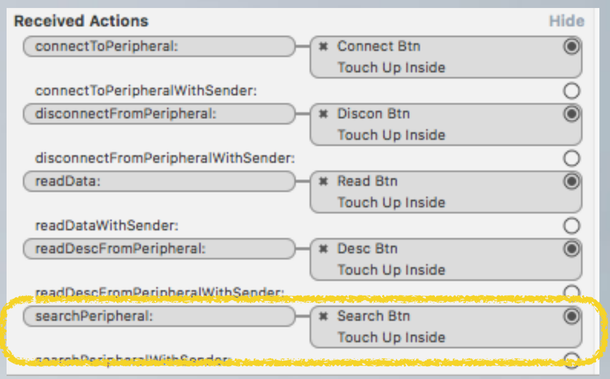
```objc
#pragma mark - IBActions
- (IBAction)searchPeripheral:(id)sender {
 // Doesn't advertise for service
 [_centralManager scanForPeripheralsWithServices:nil options:nil];

 _appState = SearchingState;
 // Update interface
 [self updateInterface];
}
```

```swift
// MARK: - Actions
@IBAction func searchPeripheral(sender: UIButton) {
 // Doesn't advertise for service
 centralManager?.scanForPeripherals(withServices: nil,  options:nil)

 appState = .searchingState
 // Update interface
 updateInterface()
}
```

[ObjC]

Interface


Rechercher périphérique
Connecter
Rechercher service & caractéristique
Lire données
Déconnecter
Périphérique
Température
Humidité
Batterie


Received Actions
Hide
connectToPeripheral:
Connect Btn
Touch Up Inside
connectToPeripheralWithSender:
disconnectFromPeripheral:
Discon Btn
Touch Up Inside
disconnectFromPeripheralWithSender:
readData:
Read Btn
Touch Up Inside
readDataWithSender:
readDescFromPeripheral:
Desc Btn
Touch Up Inside
searchPeripheral:
Search Btn
Touch Up Inside

Code

```
@property CBPeripheral *thermPeripheral;
```

```
 _thermPeripheral = nil;
```

```
 NSString *localName = [advertisementData
objectForKey:CBAdvertisementDataLocalNameKey];
 if ([localName length] > 0) {
  NSLog(@"Found the device: %@", localName);
  if ([localName compare:@"BeeWi SmartClim"] == NSOrderedSame) {
   _periphName = localName;

   // End scanning
   [_centralManager stopScan];

   // Retain peripheral
   _thermPeripheral = peripheral;

   _appState = PeriphFoundState;

   // Update interface
   [self updateInterface];
  }
```

2

```
 private var thermPeripheral: CBPeripheral?
```

3

```
  thermPeripheral = nil;
```

```
  let localName: String? =
advertisementData[CBAdvertisementDataLocalNameKey] as! String?
  if (localName != nil) && (strlen(localName)) > 0 {
   print("Found the device: \(String(describing: localName))")
   if (localName?.compare("BeeWi SmartClim") == .orderedSame) {
    periphName = localName!

    // End scanning
    centralManager?.stopScan()

    // Retain peripheral
    thermPeripheral = peripheral

    appState = .periphFoundState

    // Update interface
    updateInterface()
   }
  }
```

[ObjC]

Code

```
typedef enum {UndefinedState, InitializedState, SearchingState,
PeriphFoundState} appState_t;
```

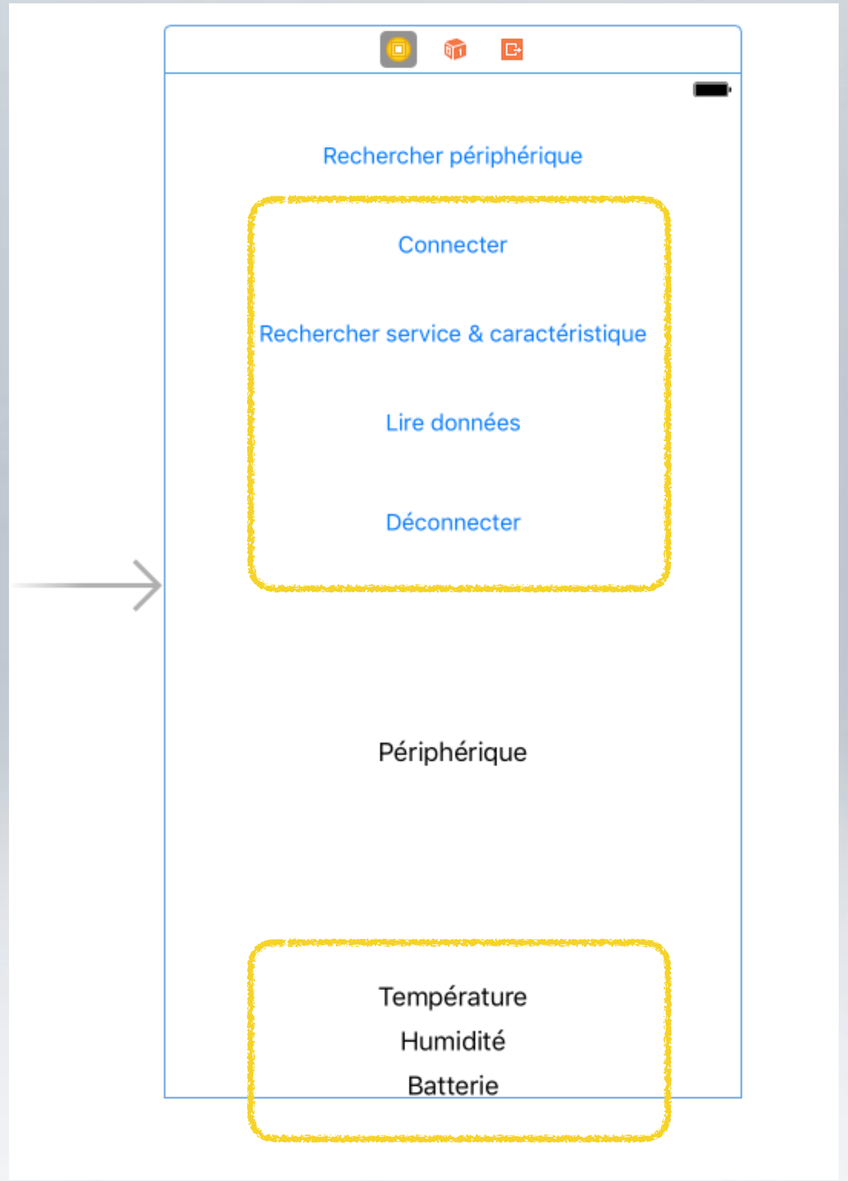
```
 enum AppState {case undefinedState, initializedState, searchingState,
periphFoundState}
```

```
  case PeriphFoundState:
   [_searchBtn setEnabled:YES];
   [_periphLab setText:_periphName];
   [_periphLab setTextColor:[UIColor redColor]];
   break;
```

2

```
  case .periphFoundState:
   searchBtn?.isEnabled = true
   periphLab?.text = periphName
   periphLab?.textColor = UIColor.red
```

Interface


Rechercher périphérique
Connecter
Rechercher service & caractéristique
Lire données
Déconnecter
Périphérique
Température
Humidité
Batterie

Code

```objc
@property IBOutlet UIButton *connectBtn;
@property IBOutlet UIButton *descBtn;
@property IBOutlet UIButton *readBtn;
@property IBOutlet UIButton *disconBtn;

@property IBOutlet UILabel *tempLab;
@property IBOutlet UILabel *humidLab;
@property IBOutlet UILabel *batLab;
```

```objc
  case UndefinedState:
  case InitializedState:
   [_searchBtn setEnabled:YES];
   [_connectBtn setEnabled:NO];
   [_descBtn setEnabled:NO];
   [_readBtn setEnabled:NO];
   [_disconBtn setEnabled:NO];

   _periphName = @"Pas de périphérique";
   [_periphLab setText:_periphName];
   [_periphLab setTextColor:[UIColor darkGrayColor]];
   break;
  case SearchingState:
   [_searchBtn setEnabled:YES];
   [_connectBtn setEnabled:NO];
   [_descBtn setEnabled:NO];
   [_readBtn setEnabled:NO];
   [_disconBtn setEnabled:NO];

   [_periphLab setText:_periphName];
   [_periphLab setTextColor:[UIColor darkGrayColor]];
   break;
  case PeriphFoundState:
   [_searchBtn setEnabled:YES];
   [_connectBtn setEnabled:YES];
   [_descBtn setEnabled:NO];
   [_readBtn setEnabled:NO];
   [_disconBtn setEnabled:NO];

   [_periphLab setText:_periphName];
   [_periphLab setTextColor:[UIColor redColor]];
   break;
```

2

```swift
@IBOutlet var connectBtn: UIButton?
@IBOutlet var descBtn: UIButton?
@IBOutlet var readBtn: UIButton?
@IBOutlet var disconBtn: UIButton?

@IBOutlet var tempLab: UILabel?
@IBOutlet var humidLab: UILabel?
@IBOutlet var batLab: UILabel?
```

```swift
 case .undefinedState,
    .initializedState:
  searchBtn?.isEnabled = true
  connectBtn?.isEnabled = false
  descBtn?.isEnabled = false
  readBtn?.isEnabled = false
  disconBtn?.isEnabled = false

  periphName = "Pas de périphérique"
  periphLab?.text = periphName
  periphLab?.textColor = UIColor.darkGray
 case .searchingState:
  searchBtn?.isEnabled = true
  connectBtn?.isEnabled = false
  descBtn?.isEnabled = false
  readBtn?.isEnabled = false
  disconBtn?.isEnabled = false

  periphLab?.text = periphName
  periphLab?.textColor = UIColor.darkGray
 case .periphFoundState:
  searchBtn?.isEnabled = true
  connectBtn?.isEnabled = true
  descBtn?.isEnabled = false
  readBtn?.isEnabled = false
  disconBtn?.isEnabled = false

  periphLab?.text = periphName
  periphLab?.textColor = UIColor.red
```

[ObjC]

Code

```objc
@property float temperature;
@property int humidity;
@property int battery;
```

2

```swift
private var temperature: Float
private var humidity: Int
private var battery: Int
```

```objc
_temperature = 0.0;
_humidity = 0;
_battery = 0;
```

3

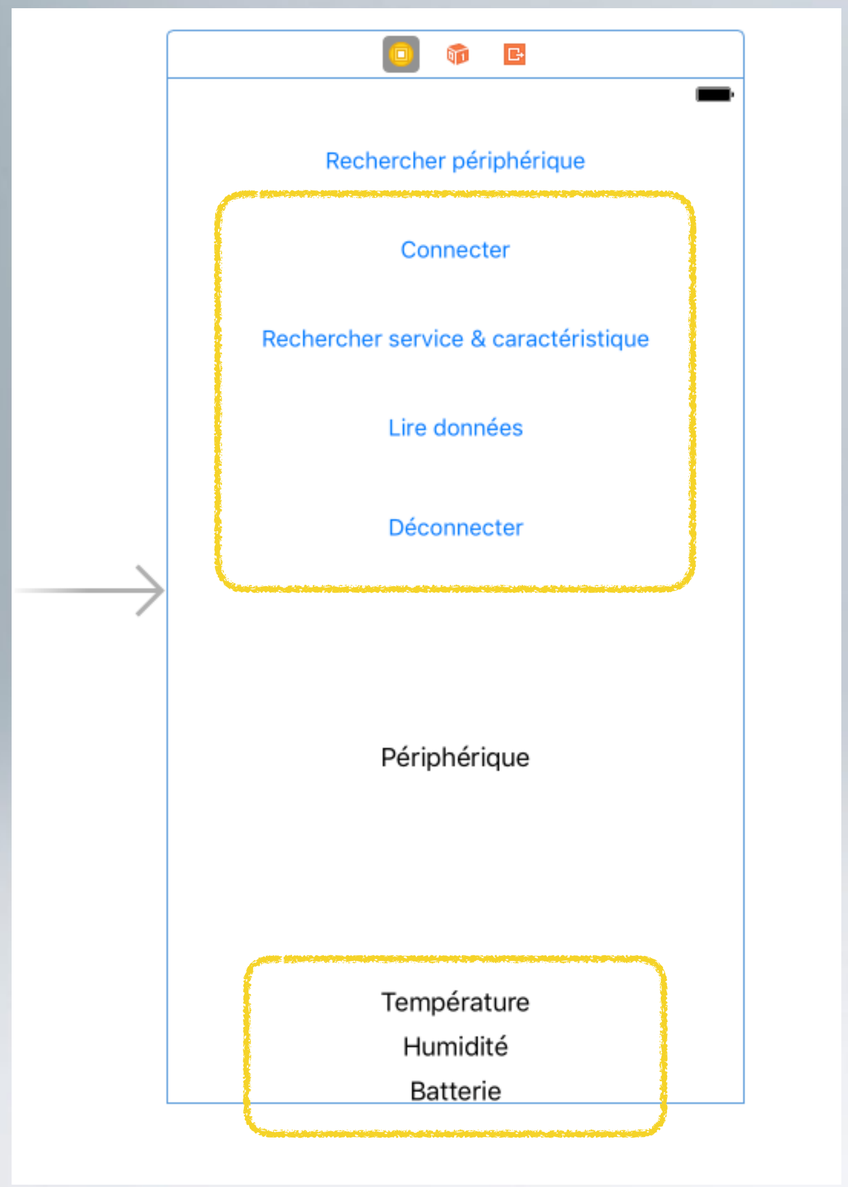
```swift
temperature = 0.0
humidity = 0
battery = 0
```

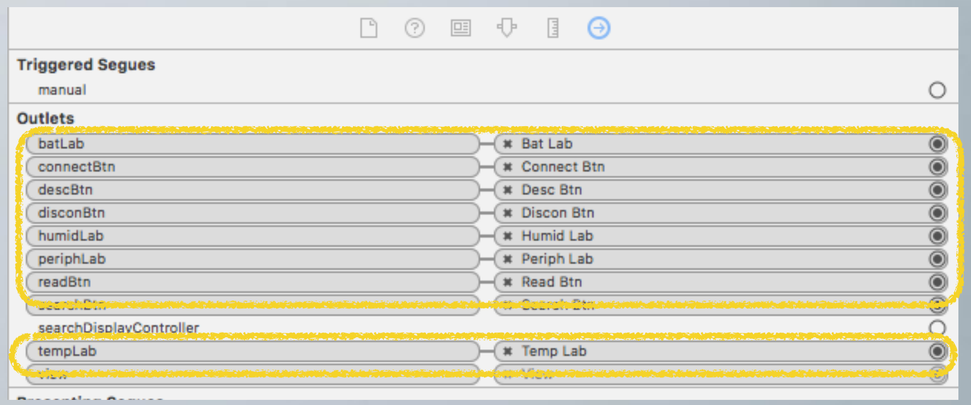
```objc
- (IBAction)connectToPeripheral:(id)sender {
 // Connect
}

- (IBAction)readDescFromPeripheral:(id)sender {
 // Read service and characteristic description
}

- (IBAction)readData:(id)sender {
}

- (IBAction)disconnectFromPeripheral:(id)sender {
 // Disconnect
}
```

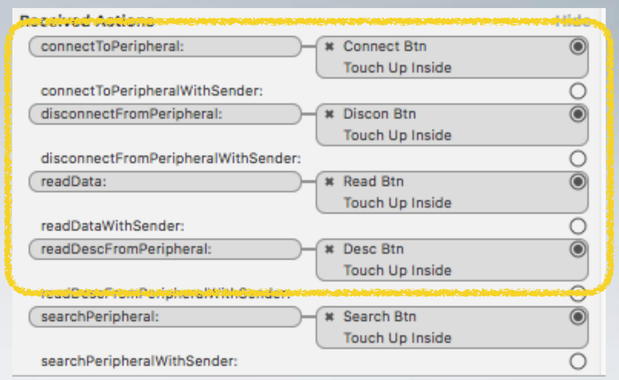
```swift
@IBAction func connectToPeripheral(sender: UIButton) {
 // Connect
}

@IBAction func readDescFromPeripheral(sender: UIButton) {
 // Read service and characteristic description
}

 @IBAction func readData(sender: UIButton) {
}

@IBAction func disconnectFromPeripheral(sender: UIButton) {
 // Disconnect
}
```

[ObjC]

Interface


Rechercher périphérique
Connecter
Rechercher service & caractéristique
Lire données
Déconnecter
Périphérique
Température
Humidité
Batterie


Triggered Segues
manual
Outlets
batLab
Bat Lab
connectBtn
Connect Btn
descBtn
Desc Btn
disconBtn
Discon Btn
humidLab
Humid Lab
periphLab
Periph Lab
readBtn
Read Btn
searchDisplayController
tempLab
Temp Lab


connectToPeripheral:
Connect Btn
Touch Up Inside
connectToPeripheralWithSender:
disconnectFromPeripheral:
Discon Btn
Touch Up Inside
disconnectFromPeripheralWithSender:
readData:
Read Btn
Touch Up Inside
readDataWithSender:
readDescFromPeripheral:
Desc Btn
Touch Up Inside
searchPeripheral:
Search Btn
Touch Up Inside
searchPeripheralWithSender:

Code

Controller.h

```
@interface ViewController : UIViewController <CBCentralManagerDelegate,
CBPeripheralDelegate>
```

2

```
class ViewController: UIViewController, CBCentralManagerDelegate,
CBPeripheralDelegate {
```

Controller.m

```
#pragma mark - CBPeripheralDelegate

// CBPeripheralDelegate - Invoked when you discover the peripheral's
available services.
- (void)peripheral:(CBPeripheral *)peripheral didDiscoverServices:
(NSError *)error {
}

// Invoked when you discover the characteristics of a specified
service.
- (void)peripheral:(CBPeripheral *)peripheral
didDiscoverCharacteristicsForService:(CBService *)service error:
(NSError *)error {
}

// Invoked when you retrieve a specified characteristic's value, or
when the peripheral device notifies your app that the characteristic's
value has changed.
- (void)peripheral:(CBPeripheral *)peripheral
didUpdateValueForCharacteristic:(CBCharacteristic *)characteristic
error:(NSError *)error {
}
```

3

```
 // MARK: - CBPeripheralDelegate

 // Invoked when you discover the peripheral's available services.
 func peripheral(_ peripheral: CBPeripheral, didDiscoverServices
error: Error?) {
 }

 // Invoked when you discover the characteristics of a specified
service.
 func peripheral(_ peripheral: CBPeripheral,
didDiscoverCharacteristicsFor service: CBService, error: Error?) {
 }

 // Invoked when you retrieve a specified characteristic's value, or
when the peripheral device notifies your app that the characteristic's
value has changed.
 func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor
characteristic: CBCharacteristic, error: Error?) {
 }
```

```
- (IBAction)connectToPeripheral:(id)sender {
 // Connect
 _thermPeripheral.delegate = self;

 [_centralManager connectPeripheral:_thermPeripheral options:nil];
}
```

```
@IBAction func connectToPeripheral(sender: UIButton) {
 // Connect
 thermPeripheral?.delegate = self

 centralManager?.connect(thermPeripheral!, options:nil)
}
```

[ObjC]

Protocol

CBPeripheralDelegate

Language

Swift
Objective-C

SDKs

iOS 5.0+
macOS 10.7+
tvOS 9.0+

On This Page

The delegate of a CBPeripheral object must adopt the CBPeripheralDelegate protocol. The delegate uses this protocol's methods to monitor the discovery, exploration, and interaction of a remote peripheral's services and properties. There are no required methods in this protocol.

Symbols

Discovering Services

func peripheral(CBPeripheral, didDiscoverServices: Error?)
Invoked when you discover the peripheral's available services.
func peripheral(CBPeripheral, didDiscoverIncludedServicesFor: CBService, error: Error?)
Invoked when you discover the included services of a specified service.

Discovering Characteristics and Characteristic Descriptors

func peripheral(CBPeripheral, didDiscoverCharacteristicsFor: CBService, error: Error?)
Invoked when you discover the characteristics of a specified service.
func peripheral(CBPeripheral, didDiscoverDescriptorsFor: CBCharacteristic, error: Error?)
Invoked when you discover the descriptors of a specified characteristic.

Retrieving Characteristic and Characteristic Descriptor Values

func peripheral(CBPeripheral, didUpdateValueFor: CBCharacteristic, error: Error?)
Invoked when you retrieve a specified characteristic's value, or when the peripheral device notifies your app that the characteristic's value has changed.
func peripheral(CBPeripheral, didUpdateValueFor: CBDescriptor, error: Error?)
Invoked when you retrieve a specified characteristic descriptor's value.

Writing Characteristic and Characteristic Descriptor Values

func peripheral(CBPeripheral, didWriteValueFor: CBCharacteristic, error: Error?)
Invoked when you write data to a characteristic's value.
func peripheral(CBPeripheral, didWriteValueFor: CBDescriptor, error: Error?)
Invoked when you write data to a characteristic descriptor's value.

Managing Notifications for a Characteristic's Value

func peripheral(CBPeripheral, didUpdateNotificationStateFor: CBCharacteristic, error: Error?)
Invoked when the peripheral receives a request to start or stop providing notifications for a specified characteristic's value.

Retrieving a Peripheral's Received Signal Strength Indicator (RSSI) Data

func peripheralDidUpdateRSSI(CBPeripheral, error: Error?)
Invoked when you retrieve the value of the peripheral's current RSSI while it is connected to the central manager.
func peripheral(CBPeripheral, didReadRSSI: NSNumber, error: Error?)
Invoked after you call readRSSI() to retrieve the value of the peripheral's current RSSI while it is connected to the central manager.

Monitoring Changes to a Peripheral's Name or Services

func peripheralDidUpdateName(CBPeripheral)
Invoked when a peripheral's name changes.
func peripheral(CBPeripheral, didModifyServices: [CBService])
Invoked when a peripheral's services have changed.

Code

```objc
typedef enum {UndefinedState=0, InitializedState, SearchingState,
PeriphFoundState, ConnectedState} appState_t;
```

```swift
 enum AppState {case undefinedState, initializedState, searchingState,
periphFoundState, connectedState}
```

```objc
  case ConnectedState:
   [_searchBtn setEnabled:NO];
   [_connectBtn setEnabled:NO];
   [_descBtn setEnabled:YES];
   [_readBtn setEnabled:NO];
   [_disconBtn setEnabled:YES];

   [_periphLab setText:_periphName];
   [_periphLab setTextColor:[UIColor greenColor]];
   break;
```

3

```swift
  case .connectedState:
   searchBtn?.isEnabled = false
   connectBtn?.isEnabled = false
   descBtn?.isEnabled = true
   readBtn?.isEnabled = false
   disconBtn?.isEnabled = true

   periphLab?.text = periphName
   periphLab?.textColor = UIColor.green
```

```objc
_appState = ConnectedState;

// Update interface
[self updateInterface];
```

2

```swift
appState = .connectedState

// Update interface
updateInterface()
```

Code

```objc
#define BEEWI_SERVICE_1_String  @"A8B3FA04-4834-4051-89D0-3DE95CDDD318"
```

```swift
private let BEEWI_SERVICE_1_String = "A8B3FA04-4834-4051-89D0-3DE95CDDD318"
```

```objc
typedef enum {UndefinedState=0, InitializedState, SearchingState,
PeriphFoundState, ConnectedState, GettingDescState} appState_t;
```

```swift
 enum AppState {case undefinedState, initializedState, searchingState,
periphFoundState, connectedState, gettingDescState}
```

```objc
@property NSArray *services;
```

```swift
private var services: [CBUUID]=[]
```

```objc
 _services = @[[CBUUID UUIDWithString:BEEWI_SERVICE_1_String]];
```

```swift
  services.append(CBUUID(string:BEEWI_SERVICE_1_String))
```

```objc
 case GettingDescState:
  [_searchBtn setEnabled:NO];
  [_connectBtn setEnabled:NO];
  [_descBtn setEnabled:NO];
  [_readBtn setEnabled:NO];
  [_disconBtn setEnabled:YES];

  [_periphLab setText:_periphName];
  [_periphLab setTextColor:[UIColor greenColor]];
  break;
```

```swift
 case .gettingDescState:
  searchBtn?.isEnabled = false
  connectBtn?.isEnabled = false
  descBtn?.isEnabled = false
  readBtn?.isEnabled = false
  disconBtn?.isEnabled = true

  periphLab?.text = periphName
  periphLab?.textColor = UIColor.green
```

```objc
// Read service and characteristic description
[_thermPeripheral discoverServices:_services];

_appState = GettingDescState;
// Update interface
[self updateInterface];
```

```swift
// Read service and characteristic description
thermPeripheral?.discoverServices(services)

appState = .gettingDescState
// Update interface
updateInterface()
```

Code

```objc
for (CBService *service in peripheral.services) {
 NSLog(@"Discovered service %@", service);
 NSLog(@"Discovering characteristics for service %@", service);
  [peripheral discoverCharacteristics:nil forService:service];
}
```

```swift
for service in peripheral.services! {
 print("Discovered service \(service)")
 print("Discovering characteristics for service %@", service)
 peripheral.discoverCharacteristics(nil, for: service)
 }
```

[ObjC]

Code

```
private let BEEWI_DATA_Characteristic = "A8B3FB43-4834-4051-89D0-3DE95CDDD318"
```

```
#define BEEWI_DATA_Characteristic @"A8B3FB43-4834-4051-89D0-3DE95CDDD318"
```

1

```
typedef enum {UndefinedState=0, InitializedState, SearchingState,
PeriphFoundState, ConnectedState, GettingDescState, DescFoundState}
appState_t;
```

```
enum AppState {case undefinedState, initializedState, searchingState,
periphFoundState, connectedState, gettingDescState, descFoundState}
```

4

```
@property CBCharacteristic *dataChar;
```

```
private var dataChar: CBCharacteristic?
```

2

```
_dataChar = nil;
```

```
dataChar = nil;
```

3

```
   case DescFoundState:
    [_searchBtn setEnabled:NO];
    [_connectBtn setEnabled:NO];
    [_descBtn setEnabled:NO];
    [_readBtn setEnabled:YES];
    [_disconBtn setEnabled:YES];

    [_periphLab setText:_periphName];
    [_periphLab setTextColor:[UIColor greenColor]];
```

```
   case .descFoundState:
    searchBtn?.isEnabled = false
    connectBtn?.isEnabled = false
    descBtn?.isEnabled = false
    readBtn?.isEnabled = true
    disconBtn?.isEnabled = true

    periphLab?.text = periphName
    periphLab?.textColor = UIColor.green
```

5

```
 for (CBCharacteristic *characteristic in service.characteristics) {
  NSLog(@"Discovered characteristic %@", characteristic);
  if ([[characteristic UUID] UUIDString]
compare:BEEWI_DATA_Characteristic] == NSOrderedSame) {
    _dataChar = characteristic;

    _appState = DescFoundState;

    // Update GUI
    [self updateInterface];
  }
 }
```

```
   for characteristic in service.characteristics! {
    print("Discovered characteristic %@", characteristic)
    if characteristic.uuid.uuidString.compare(BEEWI_DATA_Characteristic)
== .orderedSame {
     dataChar = characteristic

     appState = .descFoundState

     // Update GUI
     updateInterface()
     }
    }
```

6

Code

```objc
_appState = PeriphFoundState;

// Update interface
[self updateInterface];
```

```objc
// Disconnect
[_centralManager cancelPeripheralConnection:_thermPeripheral];
_dataChar = nil;
```

[ObjC]

2

```swift
appState = .periphFoundState

// Update interface
updateInterface()
```

```swift
// Disconnect
centralManager?.cancelPeripheralConnection(thermPeripheral!)
dataChar = nil
```

Code

```
typedef enum {UndefinedState=0, InitializedState, SearchingState,
PeriphFoundState, ConnectedState, GettingDescState, DescFoundState,
ReadingState} appState_t;
```

```
enum AppState {case undefinedState, initializedState,
searchingState, periphFoundState, connectedState, gettingDescState,
descFoundState, readingState}
```

1

```
 if ((_thermPeripheral != nil) && (_dataChar != nil)) {
  [_tempLab setText:[NSString stringWithFormat:@"Température %.1f °C",
_temperature]];
  [_humidLab setText:[NSString stringWithFormat:@"Humidité %d %%",
_humidity]];
  [_batLab setText:[NSString stringWithFormat:@"Batterie %d %%",
_battery]];
 } else {
  [_tempLab setText:@"Température --.-"];
  [_humidLab setText:@"Humidité --"];
  [_batLab setText:@"Batterie --"];
 }
```

```
  if ((thermPeripheral != nil) && (dataChar != nil)) {
   tempLab?.text = String(format:"Température %.1f °C", temperature)
   humidLab?.text = String(format:"Humidité %ld %%", humidity)
   batLab?.text = String(format:"Batterie %ld %%", battery)
  } else {
   tempLab?.text = "Température --.-"
   humidLab?.text = "Humidité --"
   batLab?.text = "Batterie --"
  }
```

2

```
 case ReadingState:
  [_searchBtn setEnabled:NO];
  [_connectBtn setEnabled:NO];
  [_readBtn setEnabled:YES];
  [_disconBtn setEnabled:YES];

  [_periphLab setText:_periphName];
  [_periphLab setTextColor:[UIColor greenColor]];
  break;
```

```
 case .readingState:
  searchBtn?.isEnabled = false
  connectBtn?.isEnabled = false
  descBtn?.isEnabled = false
  readBtn?.isEnabled = true
  disconBtn?.isEnabled = true

  periphLab?.text = periphName
  periphLab?.textColor = UIColor.green
```

```
if ((_thermPeripheral != nil) && (_dataChar != nil)) {
 _appState = ReadingState;
 // Update interface
 [self updateInterface];
 [_thermPeripheral readValueForCharacteristic:_dataChar];
}
```

3

```
  if ((thermPeripheral != nil) && (dataChar != nil)) {
   appState = .readingState
   // Update interface
   updateInterface()
   thermPeripheral?.readValue(for: dataChar!)
  }
```

[ObjC]

Code

```objc
 if ([characteristic UUID] == [_dataChar UUID]) {
  NSData *data = characteristic.value;
  if ([data length] != 10) {
   NSLog(@"bad data length %d (should be 20)", (int)[data length]);
  } else {
/*# the temperature consists of 3 bytes
# Posivive value: byte 1 & 2 present the tenfold of the temperature
# Negative value: byte 2 - byte 3 present the tenfold of the
temperature */
    int temperature_buf[3];
    char *ptr_data_buf = (char *)[data bytes];
    temperature_buf[0] = ((int)ptr_data_buf[0]) & 0x000000ff;
    temperature_buf[1] = ((int)ptr_data_buf[1]) & 0x000000ff;
    temperature_buf[2] = ((int)ptr_data_buf[2]) & 0x000000ff;
    int temperatureInt = (temperature_buf[2]*255)+temperature_buf[1];
    _temperature = (float)temperatureInt / 10.0;

    char humidity_buf[1];
    humidity_buf[0] = ptr_data_buf[4];
    _humidity = (char)*humidity_buf;

    char battery_buf[1];
    battery_buf[0] = ptr_data_buf[9];
    _battery = (char)*battery_buf;

    _appState = DescFoundState;

    // Update GUI (labels)
    [self updateInterface];
  }
}
```

```swift
  if characteristic.uuid == dataChar?.uuid {
    let data : Data = characteristic.value!
    if data.count != 10 {
     print("bad data length \(data.count) (should be 20)")
    } else {
     /*# the temperature consists of 3 bytes
     # Posivive value: byte 1 & 2 present the tenfold of the
temperature
     # Negative value: byte 2 - byte 3 present the tenfold of the
temperature */
     let data_buf = [UInt8](data)
     var temperature_buf = [UInt8](repeating: 0, count: 3)
     temperature_buf[0] = data_buf[0] & 0x000000ff
     temperature_buf[1] = data_buf[1] & 0x000000ff
     temperature_buf[2] = data_buf[2] & 0x000000ff
     let temperatureInt : Int =
(Int(temperature_buf[2])*255)+Int(temperature_buf[1])
     temperature = Float(temperatureInt) / 10.0

     let humidity_buf: UInt8 = data_buf[4]
     humidity = Int(humidity_buf)

     let battery_buf : UInt8 = data_buf[9]
     battery = Int(battery_buf)

     appState = .descFoundState

     // Update GUI (labels)
     updateInterface()
    }
  }
```

RESSOURCES

ation Apple:
e.com/documentation/
ntent/documentation/Swift/Conceptual/
ming_Language/

oks:
Language (Swift 4 Edition)
rogramming-language/id1002622538?mt=11

C EDX:
OS (parties I et II)

n/specifications/gatt/services