Cocoa : Les bases

Let’s talk about Foundation...

Cocoa : Les bases

- Mutable et immutable
- Objet racine
- Les chaînes de caractères
- Les collections
- Les nombres
- Aller plus loin avec la documentation
- Lien avec l’interface

Mutable et immutable

Mutable et immutable

- NSGizmo & NSMutableGizmo
- Est-ce que le contenu peut changer ?
  - La valeur de la chaine de caractères
  - Les objets contenus dans une collection
  - ...

- NSGizmo
  - Un bon choix en général
- NSMutableGizmo
  - Si le contenu évolue souvent et incrémentalement

Objet racine

Objet racine

- NSObject
  - Classe racine
  - Fourni les mécanismes de création/destruction d’objet
  - Permet la gestion de la mémoire
  - Permet l’introspection

Objet racine

- Création
  - `+(id) alloc;`
  - `-(id) init;`
- En pratique
  - `NSObject *newObject = [[NSObject alloc] init];`

Les chaînes de caractères

- NSString
  - Gestion de chaînes de caractères Unicode
  - Immutable
- NSMutableString
  - Sous-classe mutable de NSString
- Il existe aussi NSAttributedString et NSMutableAttributedString
  - Des chaines avec « attributs » (polices, tailles, graisses, etc.)

- Création simplifiée

```
NSString *aString = @"This is a string"
```

- Les formats

```
@"Une NSString %@, un int %d"
```

- Affichage console

```
NSLog(@"Error: %d", errorCode)
```

- NSString

```
-(id)initWithFormat:(NSString *)format...

-(NSUInteger)length

-(NSArray *)pathComponents

-(BOOL)isEqualToString:(NSString *)aString

...
```

- NSMutableString

```
-(void)setString:(NSString *)aString

-(void)replaceCharactersInRange:(NSRange)aRange withString:
(NSString *)aString

-(void)appendFormat:(NSString *)format ...

...
```

Les collections

- Les tableaux : **NSArray**
  - Collection d’objets ordonnés et indexés
- Les dictionnaires : **NSDictionary**
  - Collection d’objets associés à des clés
- Les ensembles : **NSSet**
  - Collection d’objets non ordonnés
- Et leurs sous-classes mutables !

- Les collections Objective-C stockent des références vers de objets
  - Pas de type primitifs dans un tableau !

Tableaux

- NSArray

```
- (id)initWithObjects:(id)firstObj,…,nil

- (NSUInteger)count

- (BOOL)containsObject:(id)anObject

- (NSUInteger)indexOfObject:(id)anObject

...
```

Tableaux

- NSMutableArray

```
- (void)addObject:(id)anObject

- (void)removeLastObject

- (void)insertObject:(id)anObject atIndex:(NSUInteger)index

...
```

Tableaux

```objc
NSArray *array = [NSArray alloc] initWithObjects:obj1, @"toto", obj2, nil];
//Syntaxe "originale"

NSArray *array = @[obj1, @"toto", obj2];
//Syntaxe moderne

NSArray <NSString *> *stringArray = @[@"String 1", @"String 2", @"String 3"];
//Syntaxe avec annotation de type (Xcode 7)
```

Dictionnaires

- NSDictionary

```
-(id)initWithObjectsAndKeys:(id)firstObject,…,nil

- (id)valueForKey:(NSString *)key

- (NSUInteger)count

- (NSArray *)allKeys

...
```

Dictionnaires

- NSMutableDictionary

```
- (void)setValue:(id)value forKey:(NSString *)key

- (void)removeObjectForKey:(id)aKey

- (void)removeAllObjects

...
```

Dictionnaires

```objc
NSDictionary *dict = [[NSDictionary alloc] initWithObjectsAndKeys:array, @"key1", stringArray, @"key2", nil];
//Syntaxe "originale"

NSDictionary *dict = @{@"key1":obj1, @"key2":obj2};
//Syntaxe moderne

NSDictionary <NSString *, Vehicule *> *typedDict = @{@"key1" : vehicule1, @"key2" : vehicule2};
//Avec annotation de types
```

Ensembles

- NSSet

```
- (id)initWithObjects:(id)firstObj, ...
- (NSUInteger)count
- (BOOL)containsObject:(id)anObject
- (id)anyObject
```

-

Ensembles

- NSMutableSet

```
- (void)addObject:(id)anObject

- (void)addObjectFromArray:(NSArray *)anArray

- (void)intersectSet:(NSSet *)aSet

...
```

Les nombres

- NSNumber
  - “Emballage objet” pour des types primitif
  - Permet de mettre une valeur dans une collection

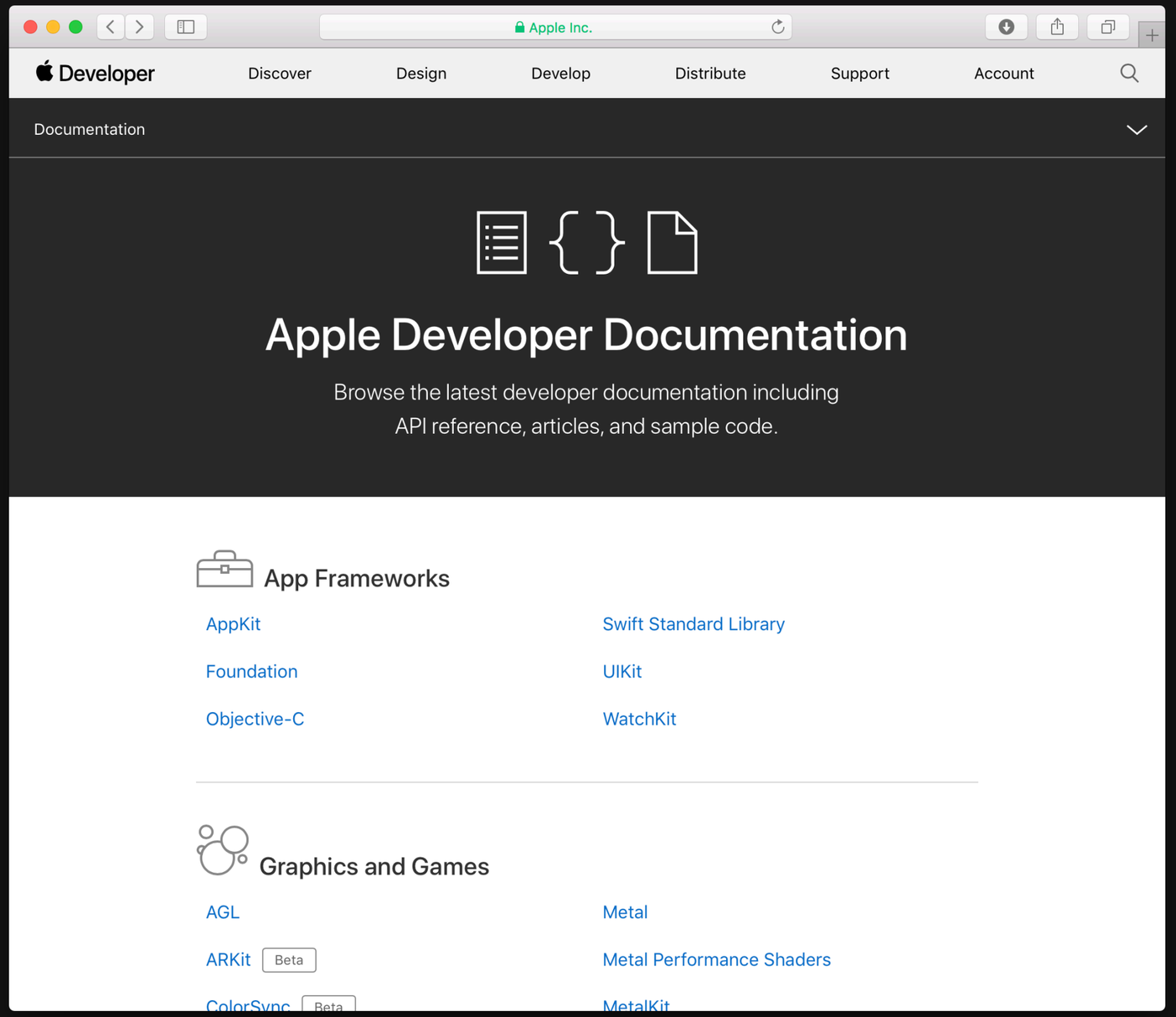
- Créer un NSNumber
  - `+ (NSNumber *)numberWithInt:(int)value`
  - `+ (NSNumber *)numberWithFloat:(float)value`
  - …
- Récupérer un type primitif
  - `- (int)intValue`

Les nombres

```objc
NSNumber *one = [NSNumber numberWithInt:1];

NSNumber *two = @2;

NSNumber *twoAndHalf = @2.5;

float f = [twoAndHalf floatValue];
```

La documentation

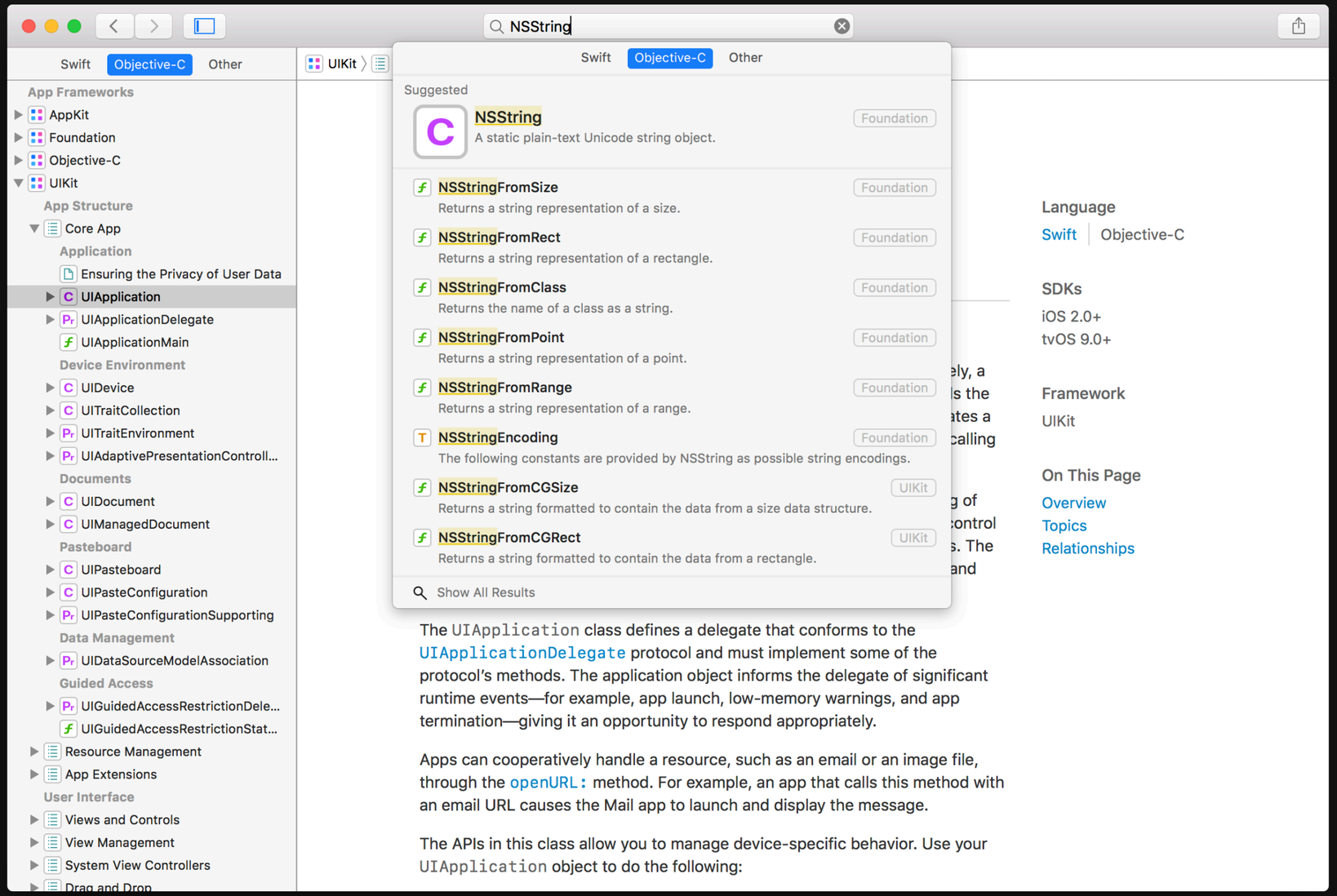
La documentation

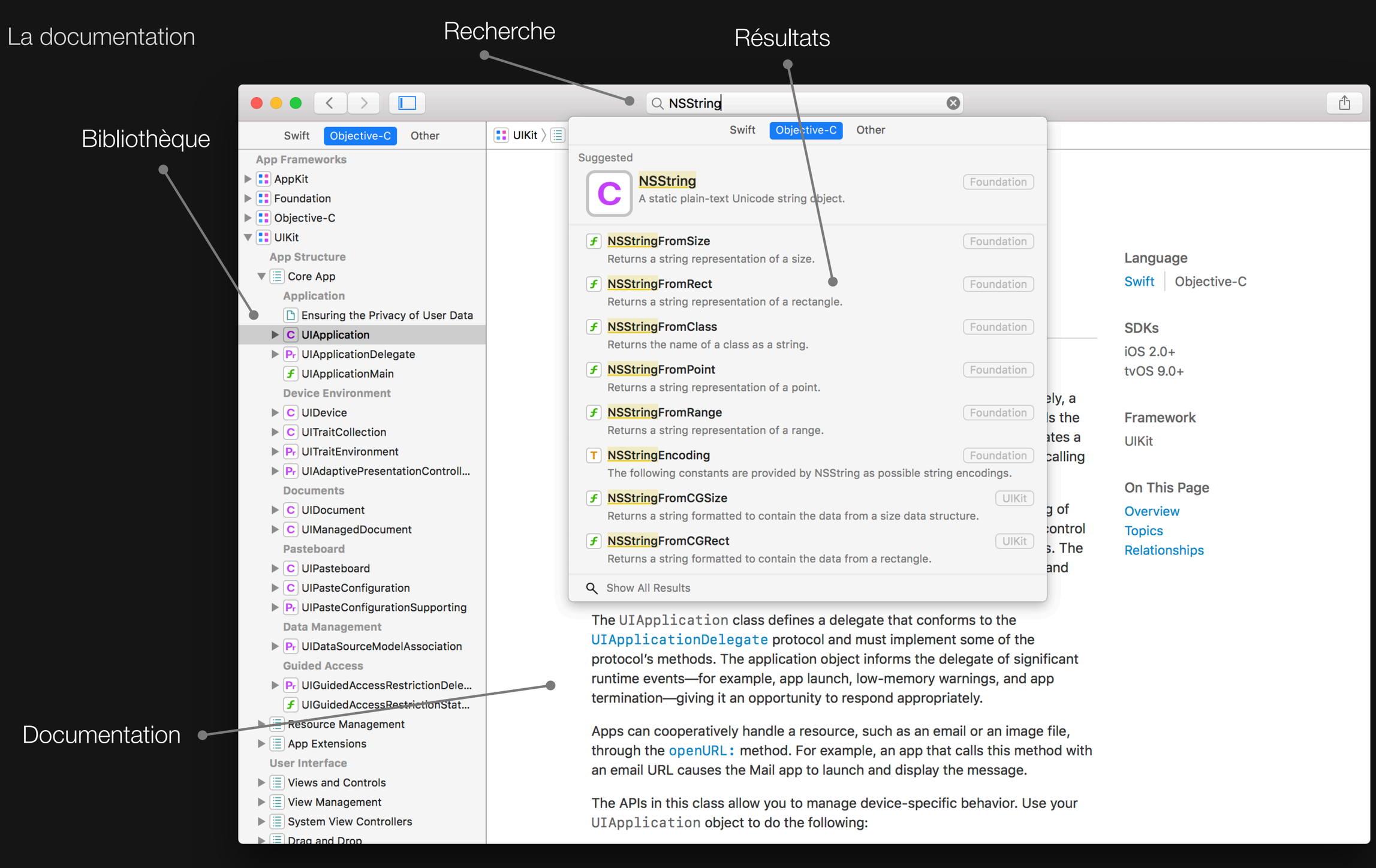


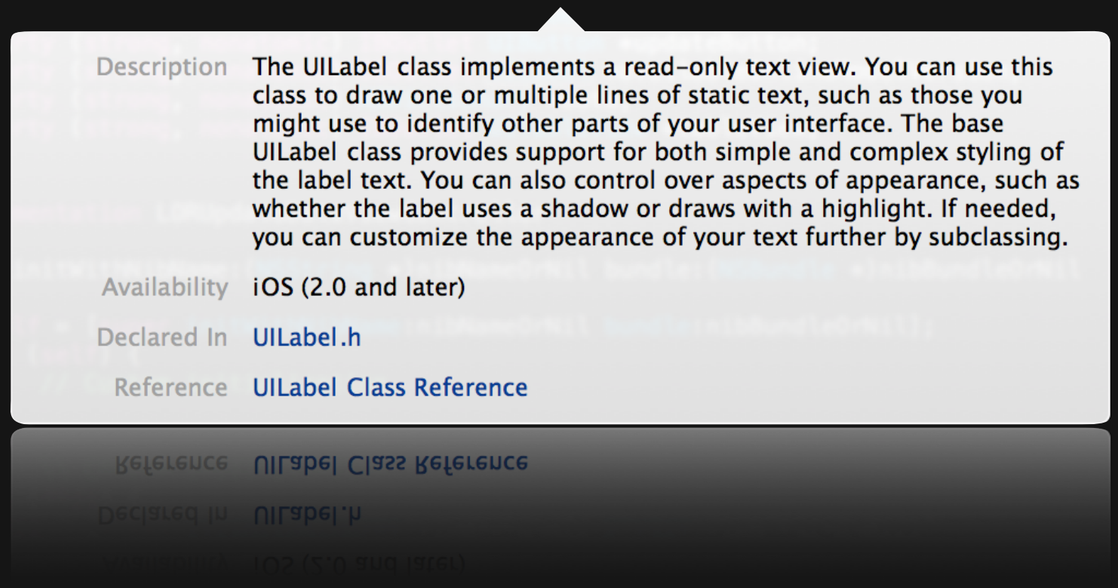
Web : developer.apple.com/documentation

La documentation

Xcode : ⌘⇧0




La documentation
Recherche
Résultats
Bibliothèque
Documentation
NSString
Swift
Objective-C
Other
App Frameworks
AppKit
Foundation
Objective-C
UIKit
App Structure
Core App
Application
Ensuring the Privacy of User Data
UIApplication
UIApplicationDelegate
UIApplicationMain
Device Environment
UIDevice
UITraitCollection
UITraitEnvironment
UIAdaptivePresentationControll...
Documents
UIDocument
UIManagedDocument
Pasteboard
UIPasteboard
UIPasteConfiguration
UIPasteConfigurationSupporting
Data Management
UIDataSourceModelAssociation
Guided Access
UIGuidedAccessRestrictionDele...
UIGuidedAccessRestrictionStat...
Resource Management
App Extensions
User Interface
Views and Controls
View Management
System View Controllers
Drag and Drop
UIKit
Suggested
NSString
A static plain-text Unicode string object.
Foundation
NSStringFromSize
Returns a string representation of a size.
NSStringFromRect
Returns a string representation of a rectangle.
NSStringFromClass
Returns the name of a class as a string.
NSStringFromPoint
Returns a string representation of a point.
NSStringFromRange
Returns a string representation of a range.
NSStringEncoding
The following constants are provided by NSString as possible string encodings.
NSStringFromCGSize
Returns a string formatted to contain the data from a size data structure.
NSStringFromCGRect
Returns a string formatted to contain the data from a rectangle.
Show All Results
The UIApplication class defines a delegate that conforms to the UIApplicationDelegate protocol and must implement some of the protocol's methods. The application object informs the delegate of significant runtime events—for example, app launch, low-memory warnings, and app termination—giving it an opportunity to respond appropriately.
Apps can cooperatively handle a resource, such as an email or an image file, through the openURL: method. For example, an app that calls this method with an email URL causes the Mail app to launch and display the message.
The APIs in this class allow you to manage device-specific behavior. Use your UIApplication object to do the following:
Language
Swift
Objective-C
SDKs
iOS 2.0+
tvOS 9.0+
Framework
UIKit
On This Page
Overview
Topics
Relationships



XCODE QUICK HELP

⌥ clic

Lien avec l'interface

Lien avec l’interface

- Action : L’interface graphique appelle une méthode sur un objet
- Outlet : Variable d’instance, faisant référence à un élément graphique

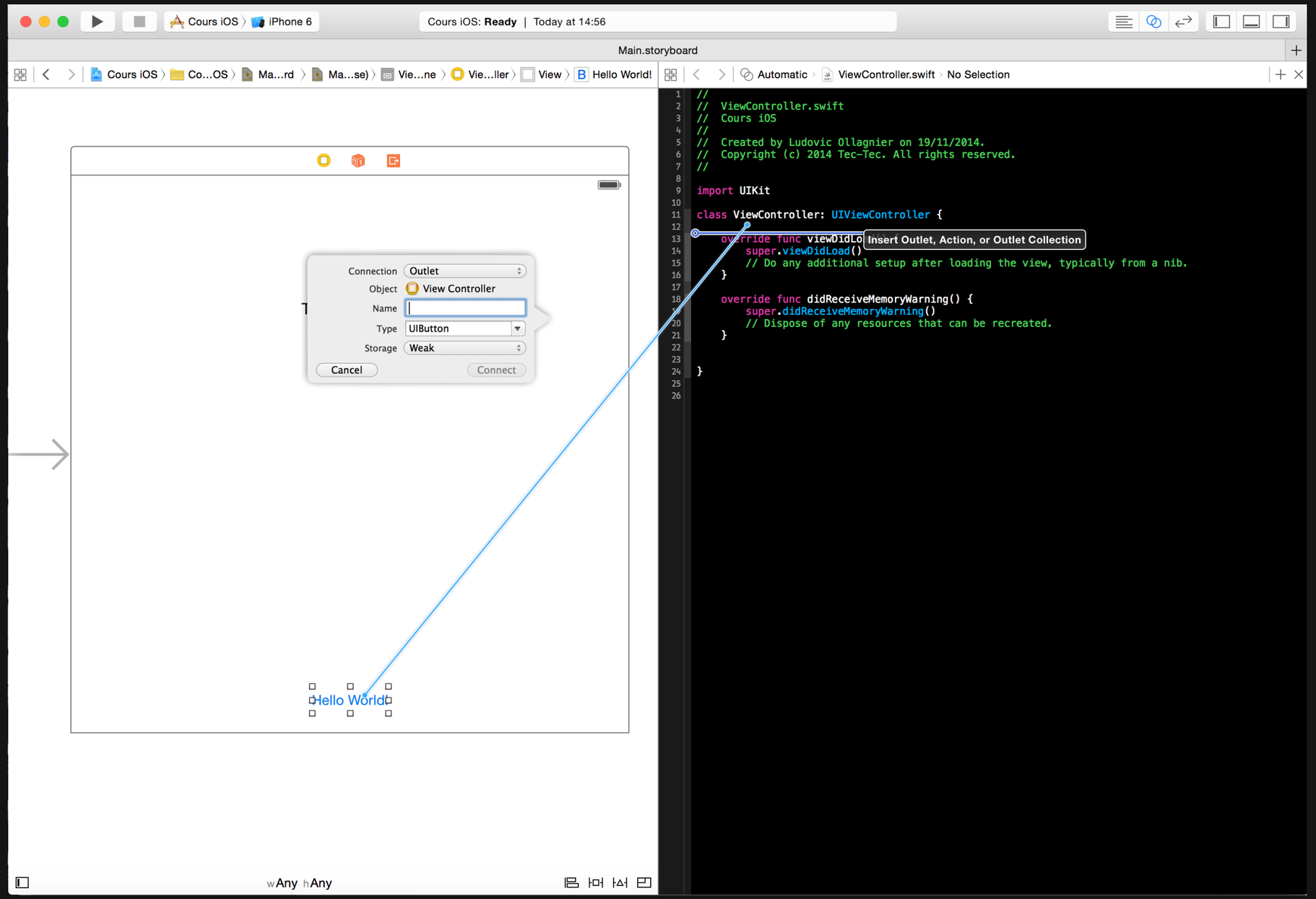
- Déclaration d’une action

```
- (IBAction)doSomething:(id)sender;
```

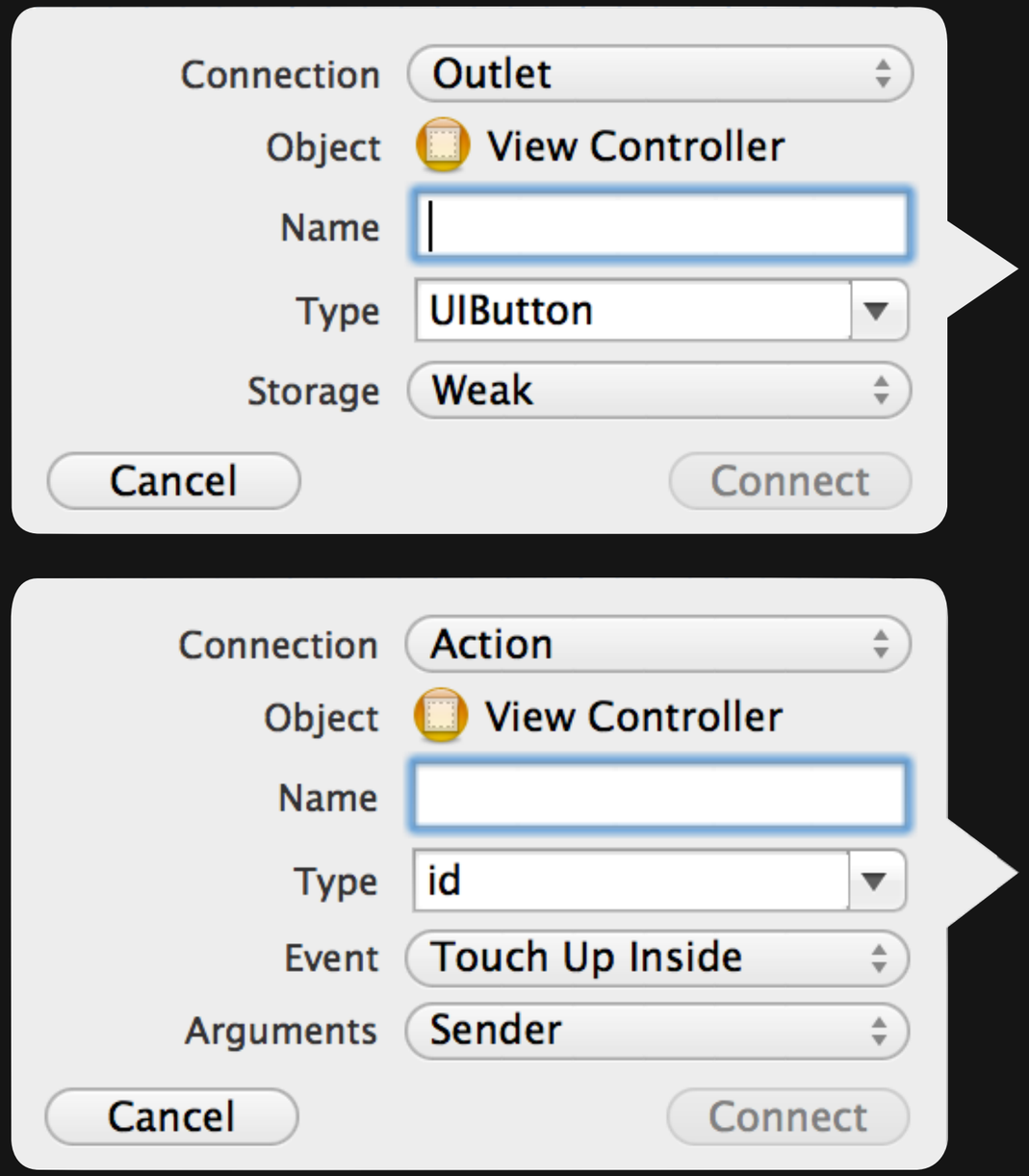
- Déclaration d’un outlet

```
IBOutlet id myButton;
```

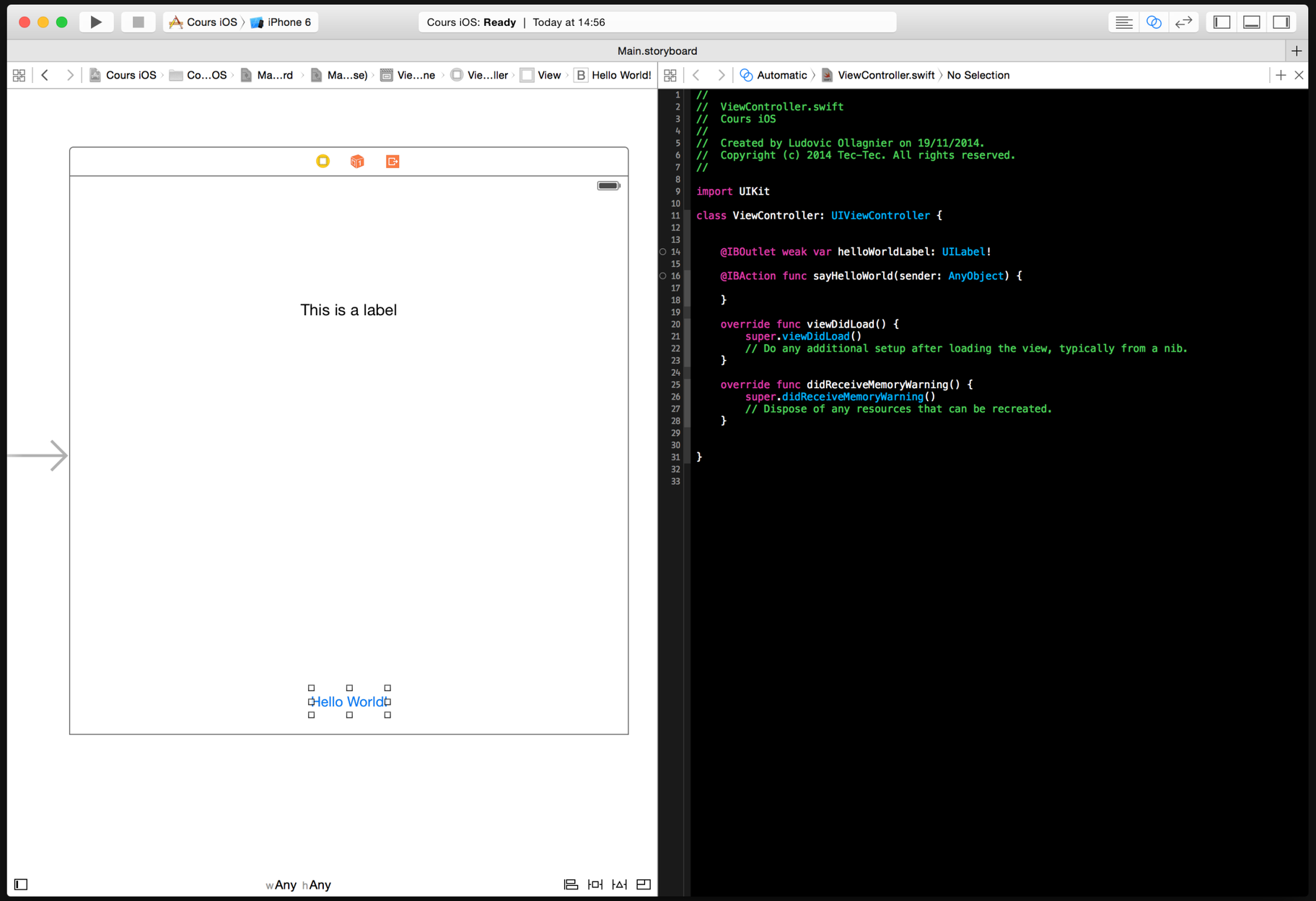
Lien avec l'interface



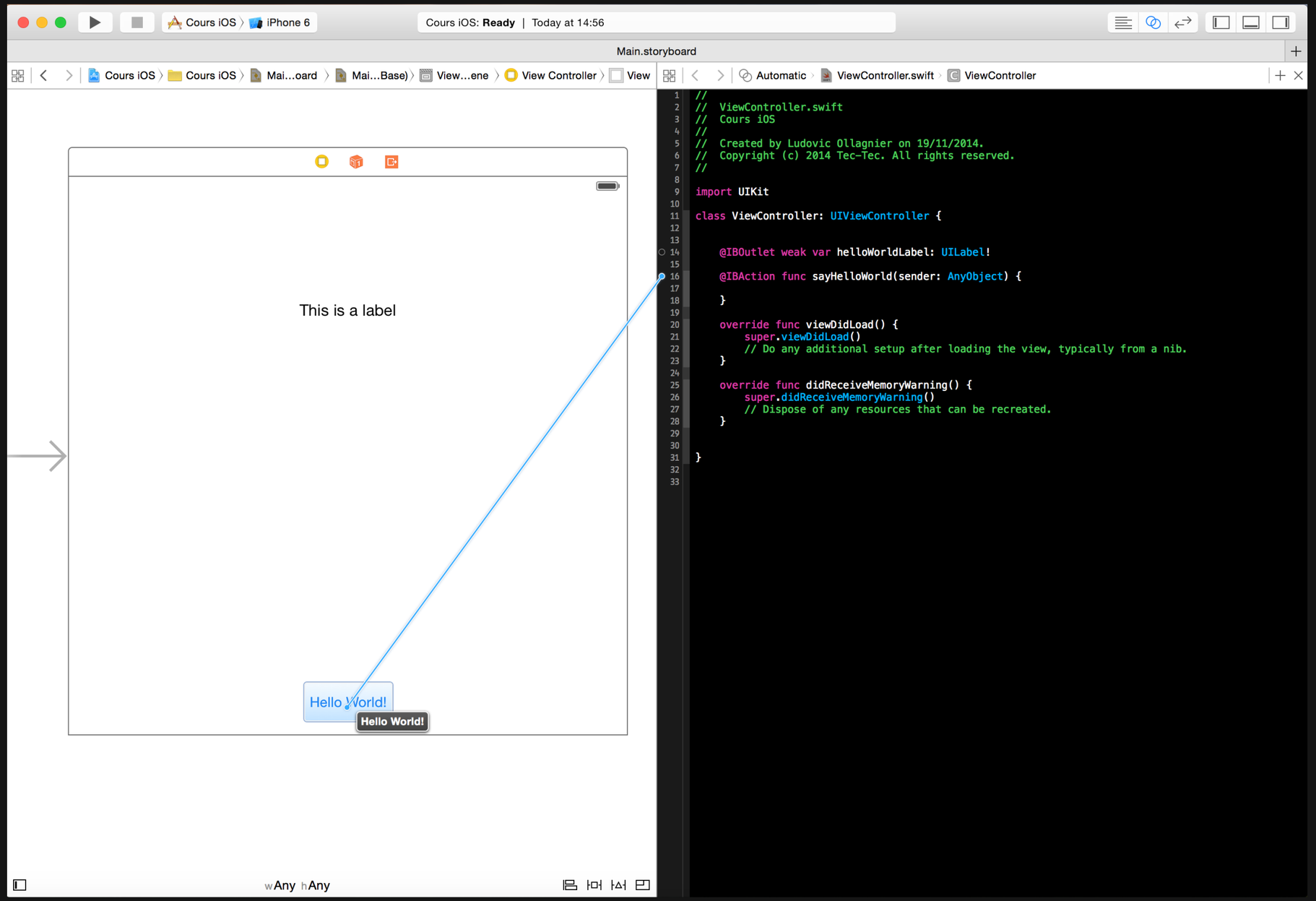
Lien avec l'interface



Lien avec l'interface



Lien avec l'interface



Pour aller plus loin…

- http://developer.apple.com
- iOS App Programming Guide