



1

iOS

Présentation générale - Xcode - Structure d'un projet - Debugger - Interface Utilisateur - Données - Media - Divers



2

Présentation Générale

Présentation générale – UIKit Framework – Foundation Framework

Présentation générale

3

Objective-C



CocoaTouch

Media
Core Services

Core OS



Présentation générale

4

▣ **Core OS** (*OS X Kernel*)

- TCP/IP
- Sockets
- Power Management
- File System
- Security

Présentation générale

5

▣ Core Services

- Networking
- SQLite (*Database*)
- Core Location
- Threads
- Address Book
- FileAccess

Présentation générale

6

▣ Media

- OpenAL (*OpenAudio Library*)
- Quartz / Core Graphics
- CoreAnimation
- Open GL

Présentation générale

7

- **Cocoa** est l'API fournies par Apple pour le développement d'application Mac OS X
- **CocoaTouch** est l'API basée sur Cocoa, permettant le développement d'application mobile pour :
 - iPhone
 - iPad
 - iPod Touch...

Présentation générale

8

▣ CocoaTouch

■ UIKit (*UI**)

- UI Controls
- Event Handling
- Hardware APIs
- Accelerometer
- Camera

▣ CocoaTouch

■ Foundation (*NS**)

- Utility & Collection classes

UIKit Framework

9

- Utiliser lors de l'implémentation d'élément graphique.
 - Appuyer sur un bouton, redimensionner une image, etc...)
- Les éléments principaux :
 - UIApplication, Vues, Contrôles, Contrôleurs
- Généralement, tout ce qui est visible est une vue (*UIView*)
- L'interface utilisateur (*UI*) est géré par des contrôleurs de vue (*UIViewController*)

Foundation Framework

10

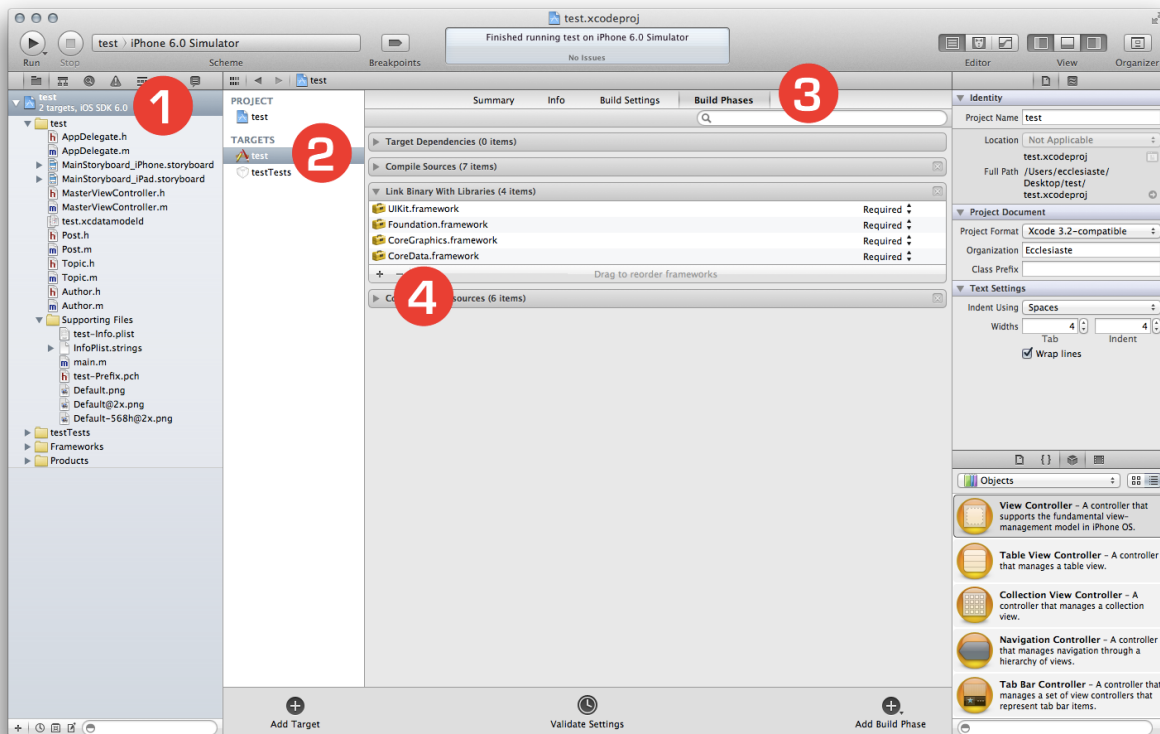
- ▣ Utilisation extrêmement fréquente de ces non-ui classes
- ▣ Exemples :
 - Collections :
 - ▣ NSArray, NSMutableArray, NSDictionary, NSMutableDictionary...
 - Values (*Wrappers*) :
 - ▣ NSNumber, NSInteger, NSValue, NSString...
 - Networking :
 - ▣ NSURL, NSURLRequest...

Framework

11

Ajouter un
framework au
projet

Après l'étape 4,
il ne vous reste
plus qu'à choisir
le(s)
framework(s) à
ajouter.





12

Xcode

Présentation de l'interface de développement (*IDE*) - Documentation

Xcode

13

- ▣ Interface de développement (**IDE**)
- ▣ Permet de gérer tous vos projets iPhone | iPad
- ▣ Généralement, un projet par application
- ▣ Permet entre autre de gérer :
 - Interface graphique
 - Code source
 - Ressources
 - Données
 - Localisation ...

Xcode

14

Créer un
nouveau projet



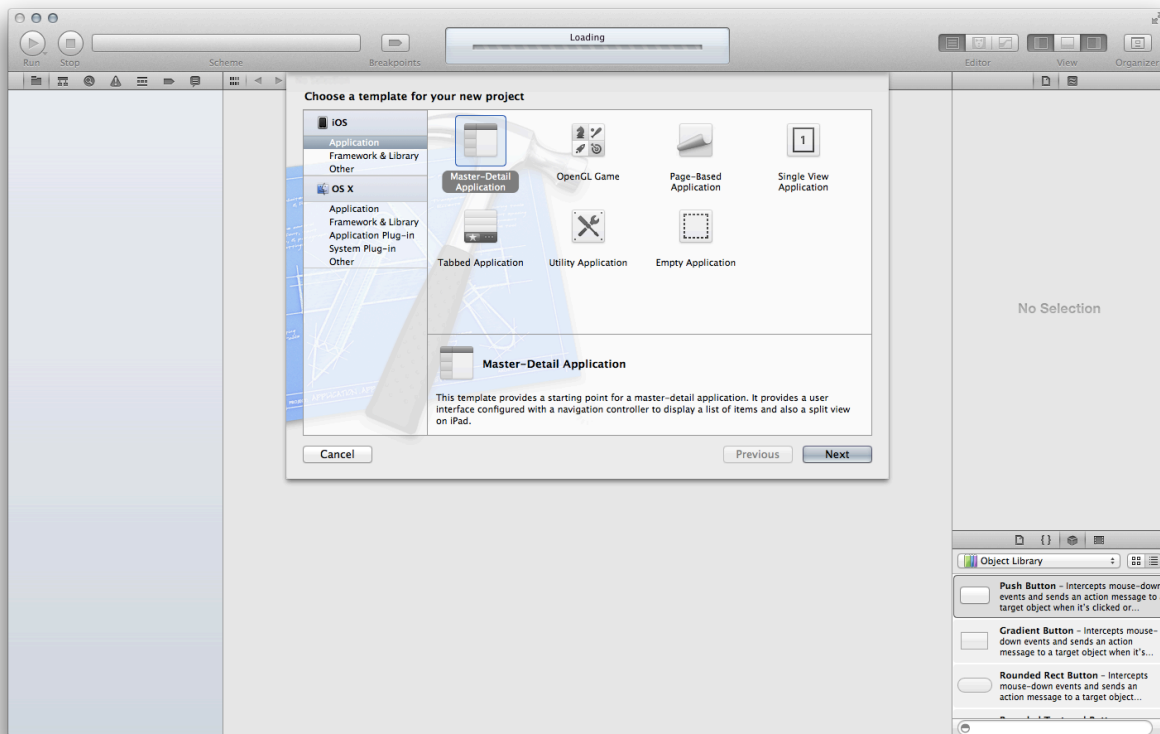
Xcode

15

Choisir un template

*Un template est un
modèle type
d'application.*

*Vous utiliserez
généralement
**Master-Detail
Application***

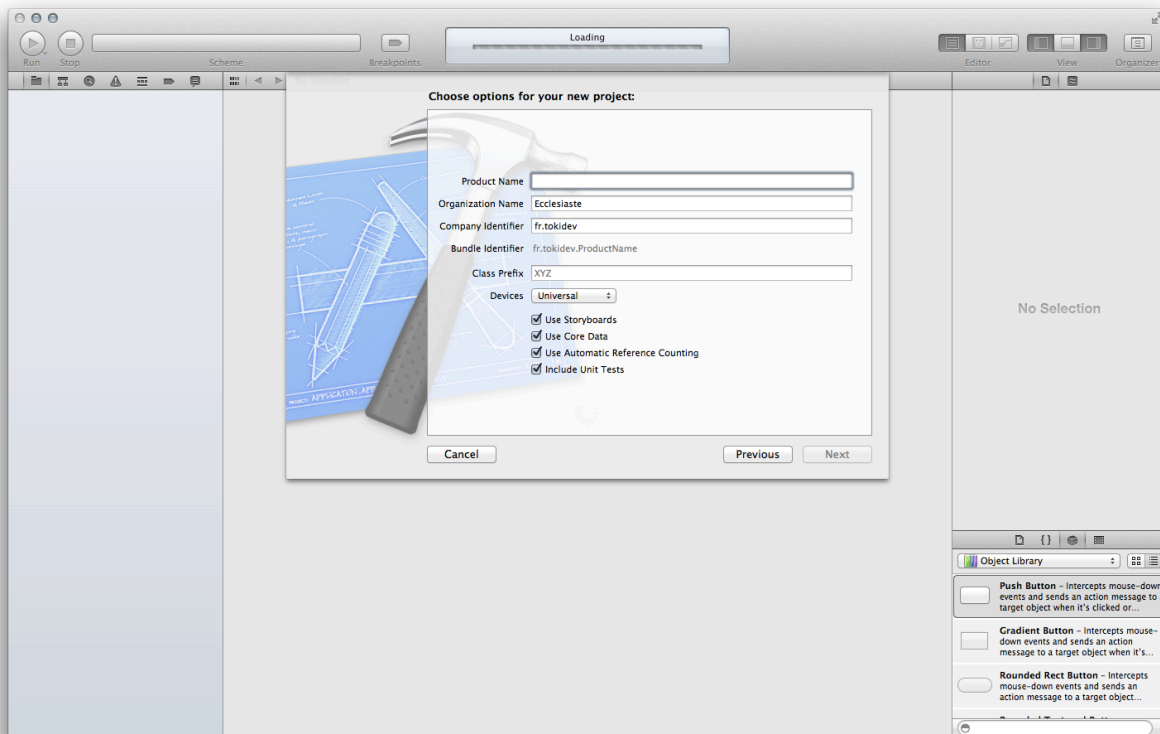


Xcode

16

Configuration
principale du
projet

*Vous pourrez
choisir ici
d'utiliser les
storyboards et
Core Data*

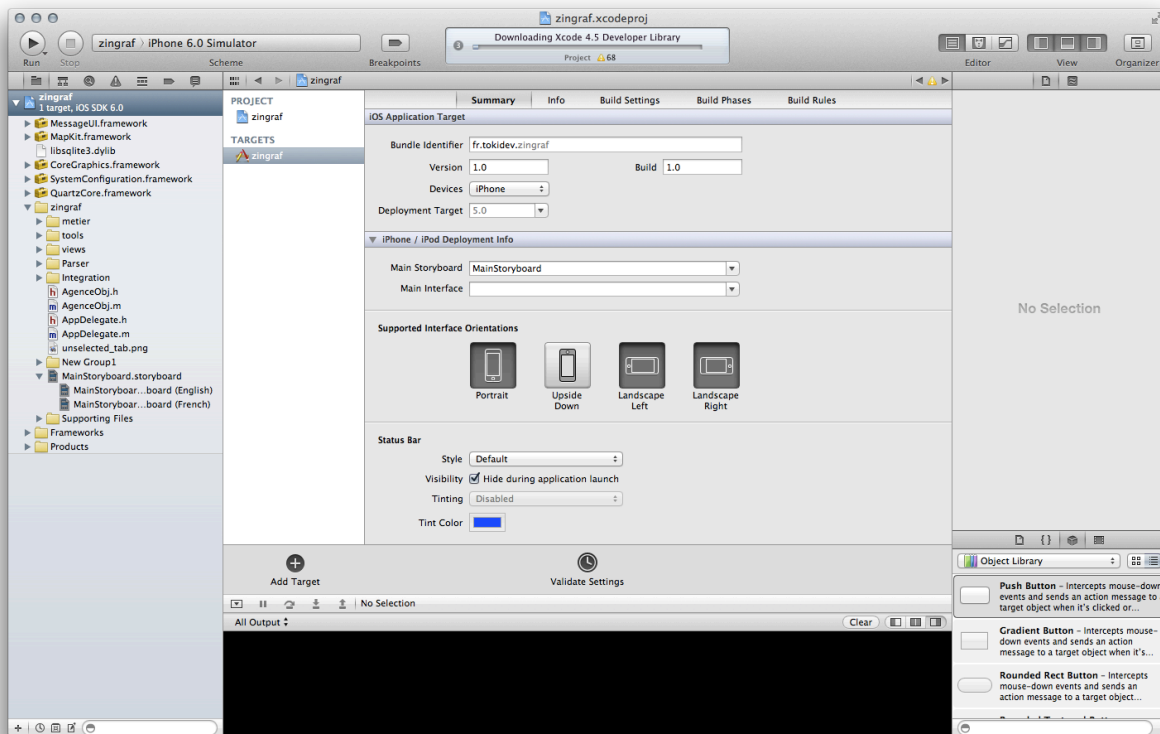


Xcode

17

**Vue d'ensemble
de l'interface**

*Ecran de
configuration du
projet*

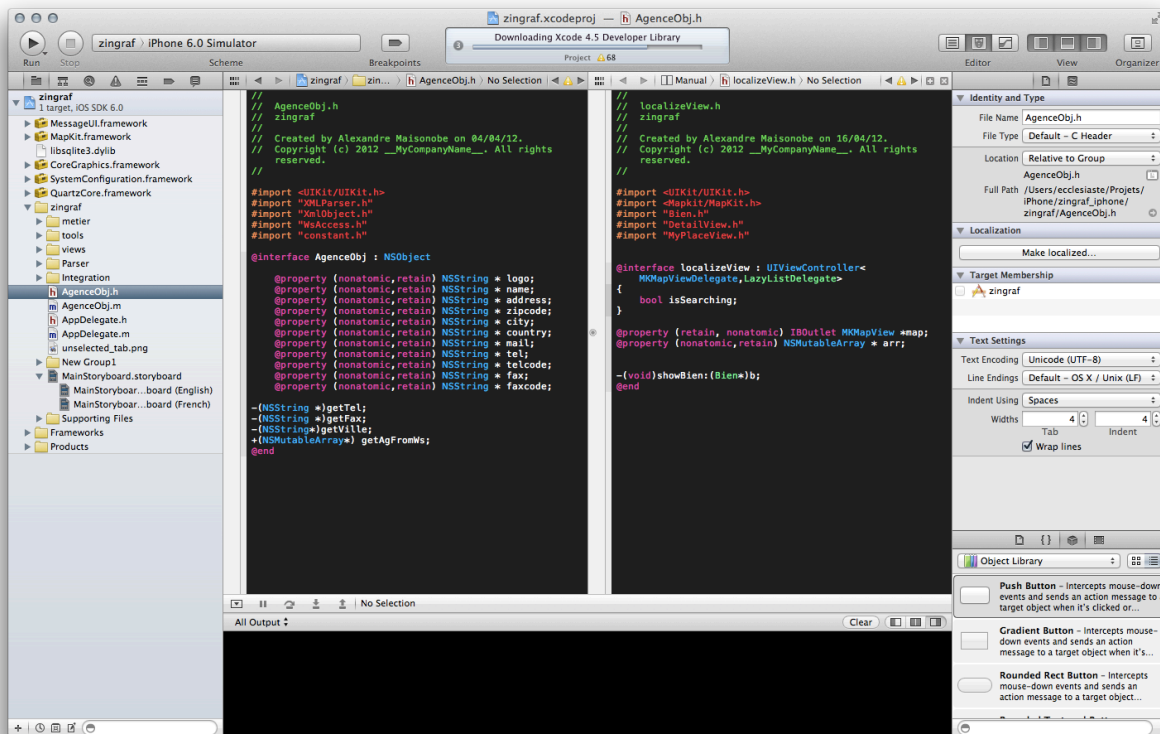


Xcode

18

Vue d'ensemble
de l'interface

Exemple de
code avec écran
partagé (.h et
.m)

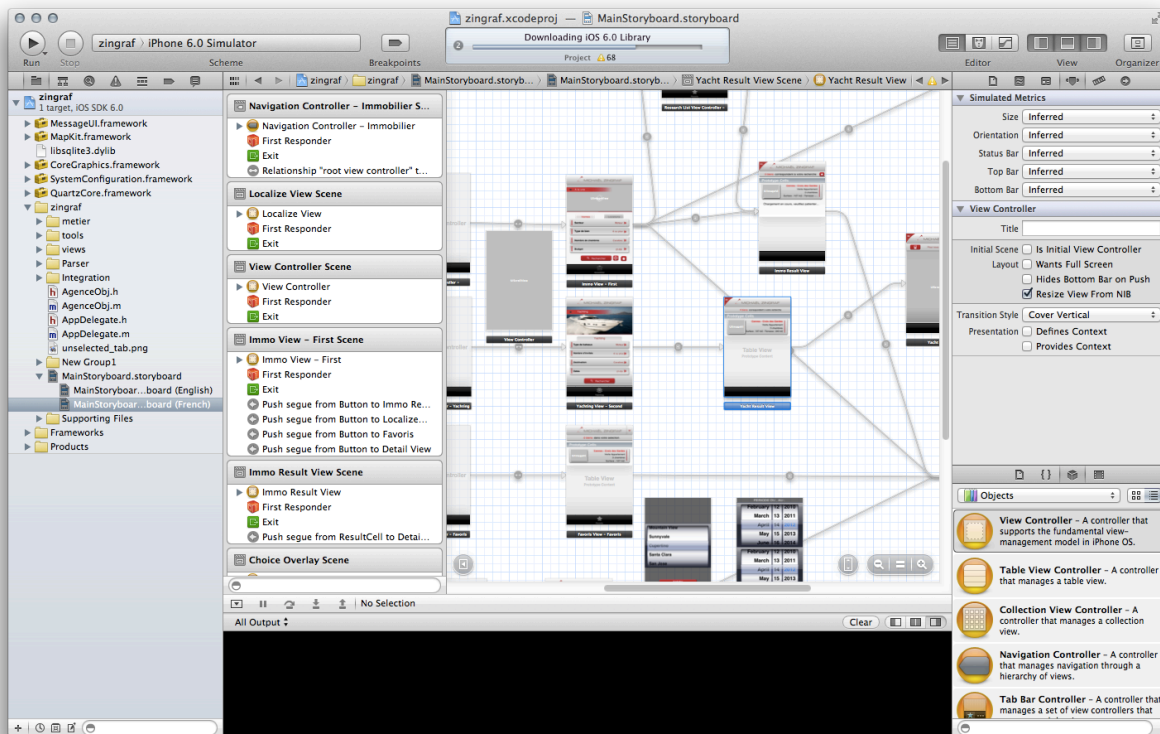


Xcode

19

Vue d'ensemble
de l'interface

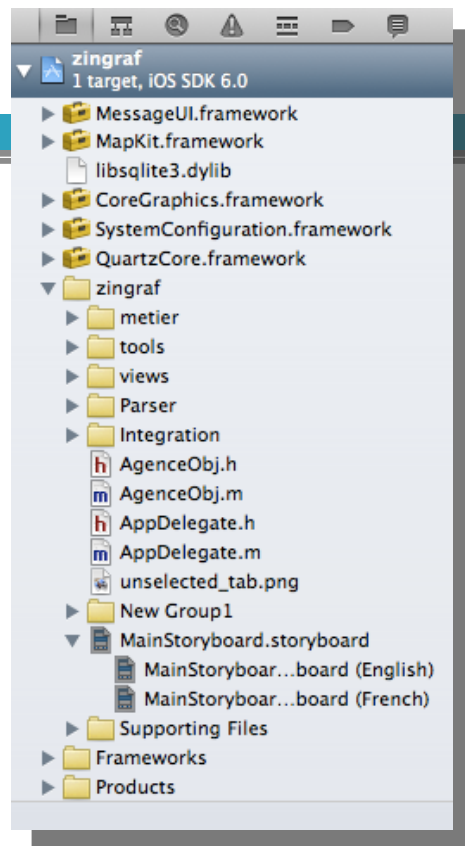
Storyboard
dezoomé



Xcode

20

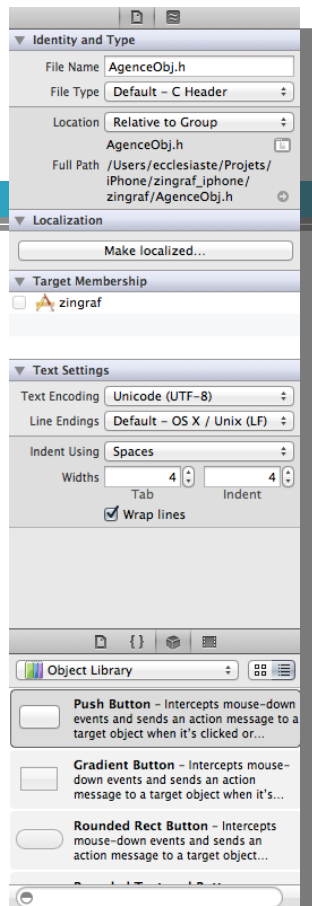
Panneau des
fichiers du
projet



Xcode

21

Panneau des
propriétés

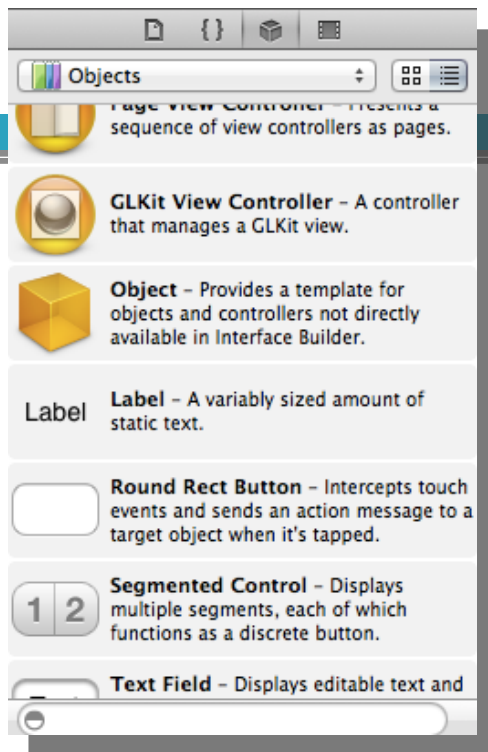


Xcode

22

Panneau des
composants
graphiques

*Dans l'interface
builder (IB) ou
le storyboard*



Documentation

23

The screenshot shows the iPhone Dev Center documentation page for the UIAlertView Class Reference. The interface includes a navigation sidebar on the left with sections like API, Title, and Full Text. The main content area is titled 'UIAlertView Class Reference' and contains a table of contents with links to Overview, Tasks, Properties, Instance Methods, Revision History, and Index. Below the table of contents, there is a table with key information about the class, including its inheritance, conformance, framework, availability, declared in, and related sample code. The 'Overview' section below the table provides a brief description of the class and its usage.

iPhone Dev Center: UIAlertView Class Reference

Back/Forward Home Bookmarks

Contains Prefix Exact All Doc Sets All Languages

UIAlertView Class Reference

Jump To...

UIAlertView Class Reference

Table of Contents

- Overview
- Tasks
- Properties
- Instance Methods
- Revision History
- Index

Inherits from	UIView : UIResponder : NSObject
Conforms to	NSCoding (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/UIKit.framework
Availability	Available in iPhone OS 2.0 and later.
Declared in	UIAlertView.h
Related sample code	GKTank LaunchMe UICatalog URLCache WITap

Overview

Use the UIAlertView class to display an alert message to the user. An alert view is a subclass of UIWindow but differs in appearance from an action sheet (an instance of UICollectionViewController).

Use the properties and methods defined in this class to set the title, message, and alert view and configure the buttons. You must set a delegate if you add custom buttons. The delegate should conform to the UIAlertViewDelegate protocol. Use the show method to display an alert view once it is configured.

Documentation

24

- La documentation officielle sera votre première source d'information.
- Google
- <http://stackoverflow.com/>
- <http://www.iphonedevsdk.com/forum/>
- Apple Developer Forums



25

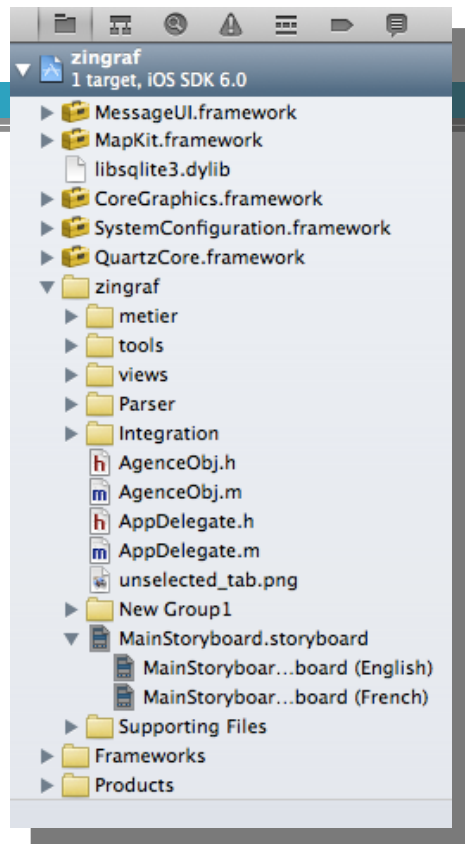
Structure d'un projet

Fichiers principaux - Suggestions

Fichiers principaux

26

La plupart des
projets
d'application
iPhone / iPad
possède des
fichiers types
standards.



Fichiers principaux

27

- nomProjet-Info.plist
 - Contient un certain nombre de paramètres du projet.
- Storyboard (*Extension .storyboard*)
 - Regroupe l'intégralité des écrans de votre application.
- Core Data (*Extension .xcdatamodel*)
 - Fichier contenant votre modèle (*ORM CoreData*)

Suggestions

28

- Créer des dossiers !
 - Framework : Regroupant les frameworks de CocoaTouch que vous ajouterez.
 - Ressources : Regroupant toutes les images, vidéos ou sons que vous utiliserez.
 - Sources : Regroupant toutes les classes de votre projet.
- Vous aurez ainsi un projet plus propre, plus compréhensible et plus facile à maintenir !



29

Debugger

Console/NSLog – Compiler Warnings – Debugger - Exceptions

Console/NSLog

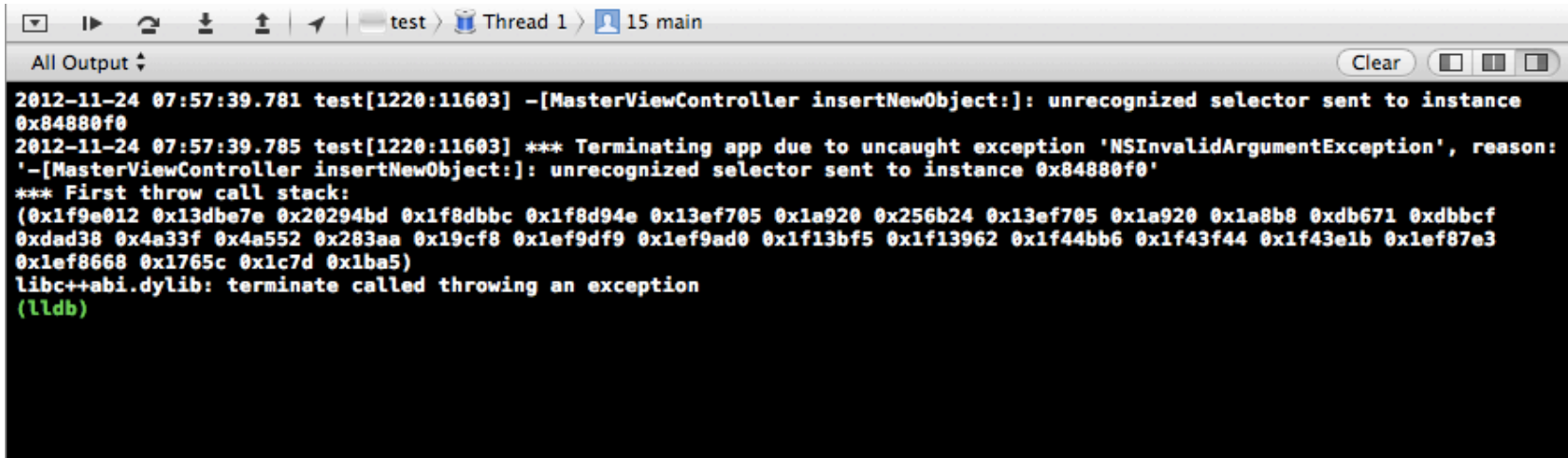
30

- Le premier outil utilisable pour vous aider à débbugger une application est la méthode **NSLog**.
- Celle-ci écrit directement dans la console.
 - Equivalent de : printf, println, echo ...
- NSLog fonctionne selon la syntaxe du printf (en C)
 - Exemple :
 - NSLog("Ceci affiche la string : %@ ", maString);
 - NSLog("Ceci affiche l'entier : %d", monEntier);

Compiler Warnings

31

- Comme pour tout langage compilé, vous aurez la possibilité de voir, dans la console, **les erreurs ou avertissements** renvoyées par le compilateur.



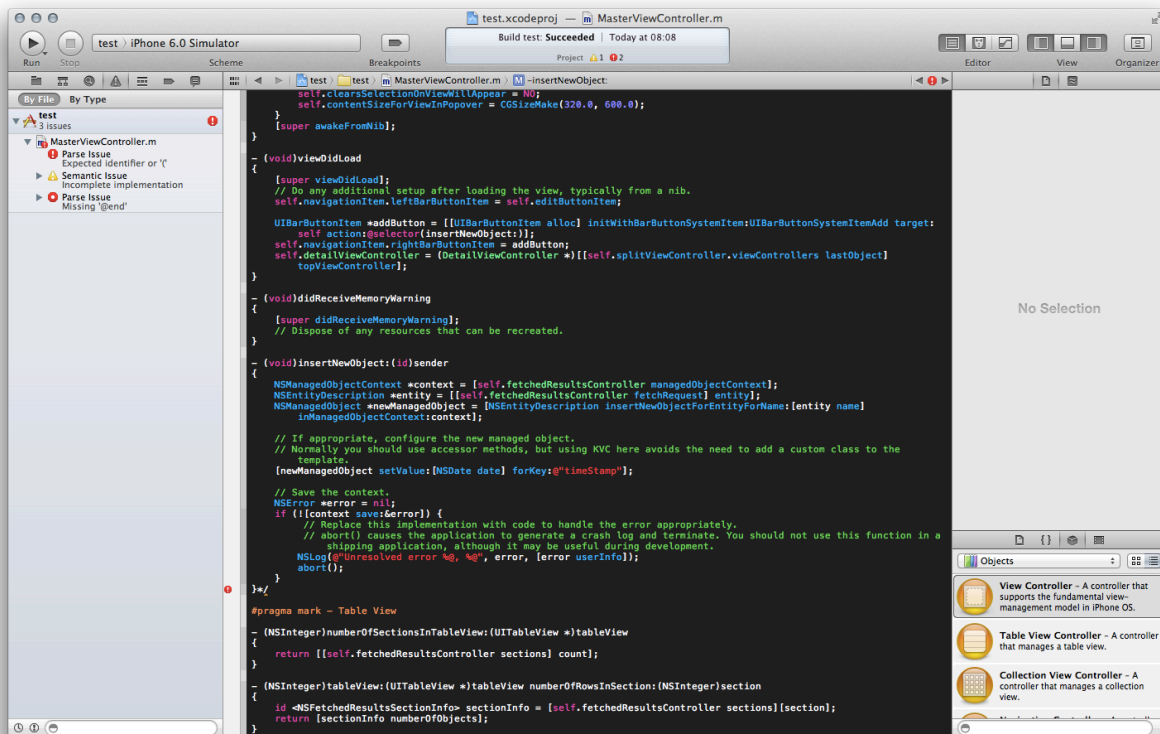
The screenshot shows a debugger's 'All Output' window. The title bar indicates the current context is 'test > Thread 1 > 15 main'. The output text is as follows:

```
2012-11-24 07:57:39.781 test[1220:11603] -[MasterViewController insertNewObject:]: unrecognized selector sent to instance 0x84880f0
2012-11-24 07:57:39.785 test[1220:11603] *** Terminating app due to uncaught exception 'NSInvalidArgumentException', reason: '-[MasterViewController insertNewObject:]: unrecognized selector sent to instance 0x84880f0'
*** First throw call stack:
(0x1f9e012 0x13dbe7e 0x20294bd 0x1f8dbbc 0x1f8d94e 0x13ef705 0x1a920 0x256b24 0x13ef705 0x1a920 0x1a8b8 0xdb671 0xdbbcf
0xdad38 0x4a33f 0x4a552 0x283aa 0x19cf8 0x1ef9df9 0x1ef9ad0 0x1f13bf5 0x1f13962 0x1f44bb6 0x1f43f44 0x1f43e1b 0x1ef87e3
0x1ef8668 0x1765c 0x1c7d 0x1ba5)
libc++abi.dylib: terminate called throwing an exception
(lldb)
```

Debugger

32

Après chaque Build, vous pourrez voir le nombre d'erreurs et avertissement en haut de votre interface.

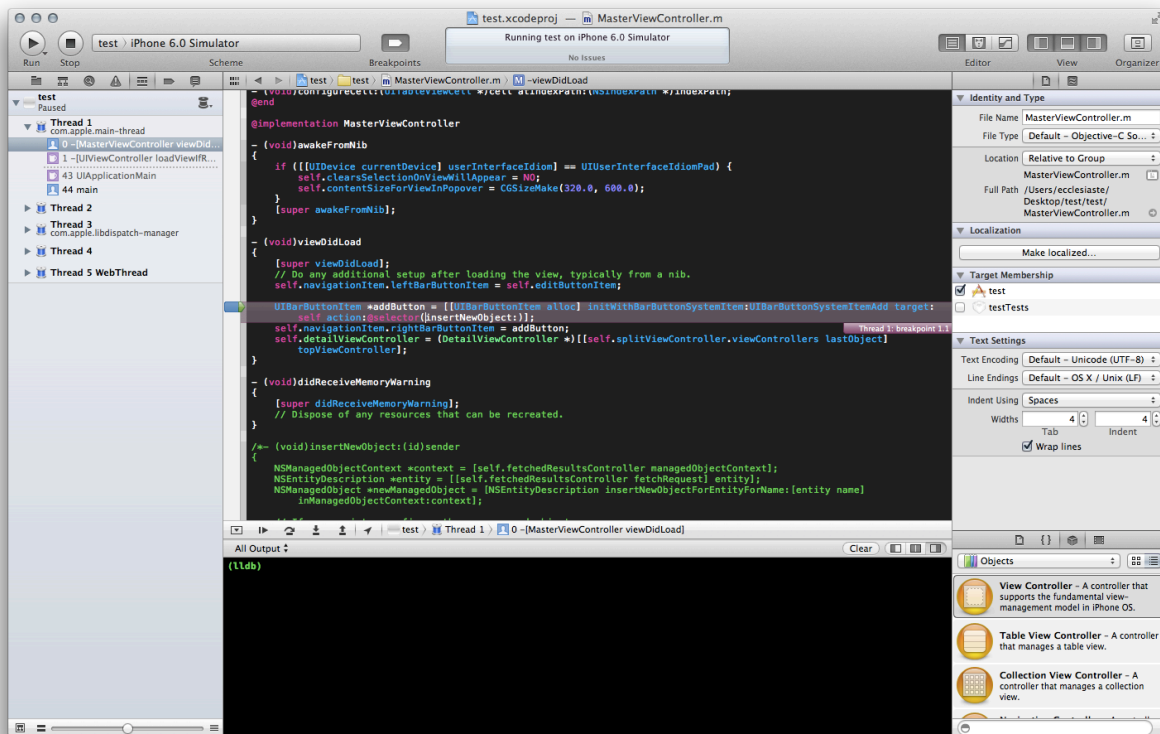


Debugger

33

Comme tout IDE,
Xcode possède
son debugger.

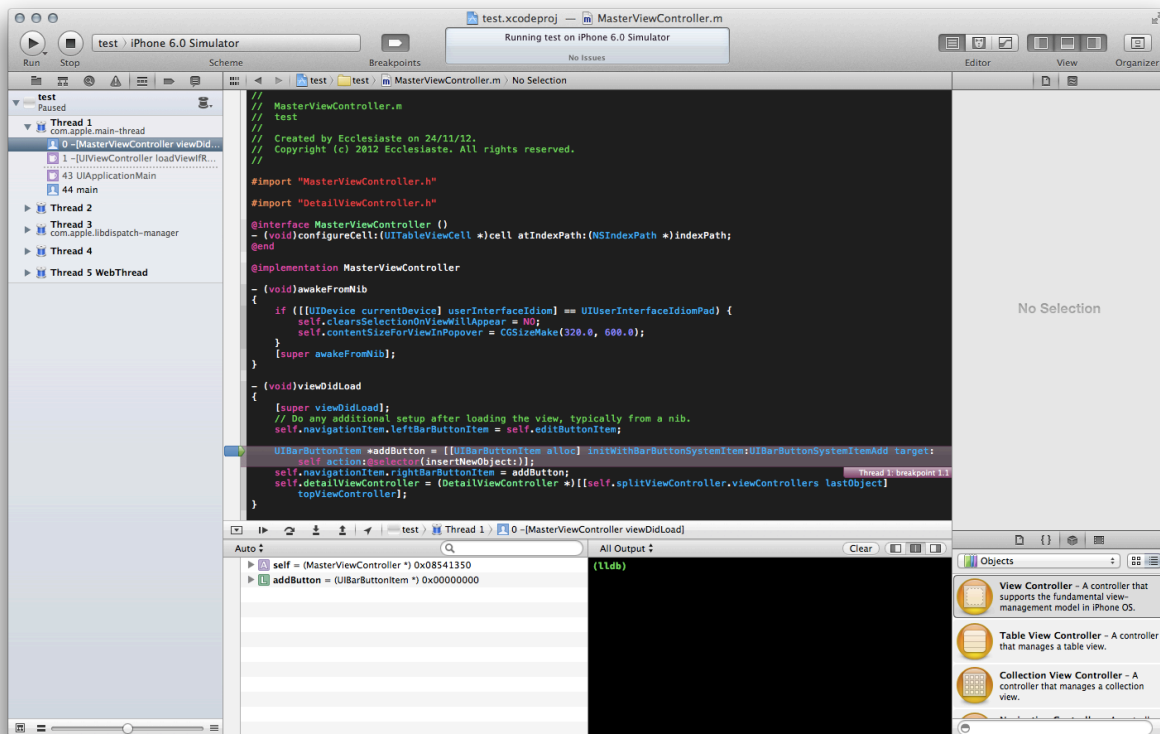
Celui-ci vous
permet de placer
des **Breakpoints**
afin de pouvoir
suivre les étapes
de l'exécution de
l'application.



Debugger

34

Une fois
l'application
lancée en mode
debug
(breakpoints),
vous pourrez
suivre son
exécution ainsi
que la valeur
des variables.



Exceptions

35

- Comme en Java ou C#, il existe des exceptions en Objective-C.

`NSException raise NSInvalidArgumentException`
`Foo must not be nil`

- Vous pouvez également, utiliser les exceptions de cette façon :

```
@try  
    // Code normal, avec exception possible  
  
@catch (NSException)  
    // Gérer l'exception  
  
@finally  
    // Nettoyage obligatoire
```



36

Interface Utilisateur

MVC - Storyboard - Vues & Conteneur – Delegate – Notification Center - Composants & Événements

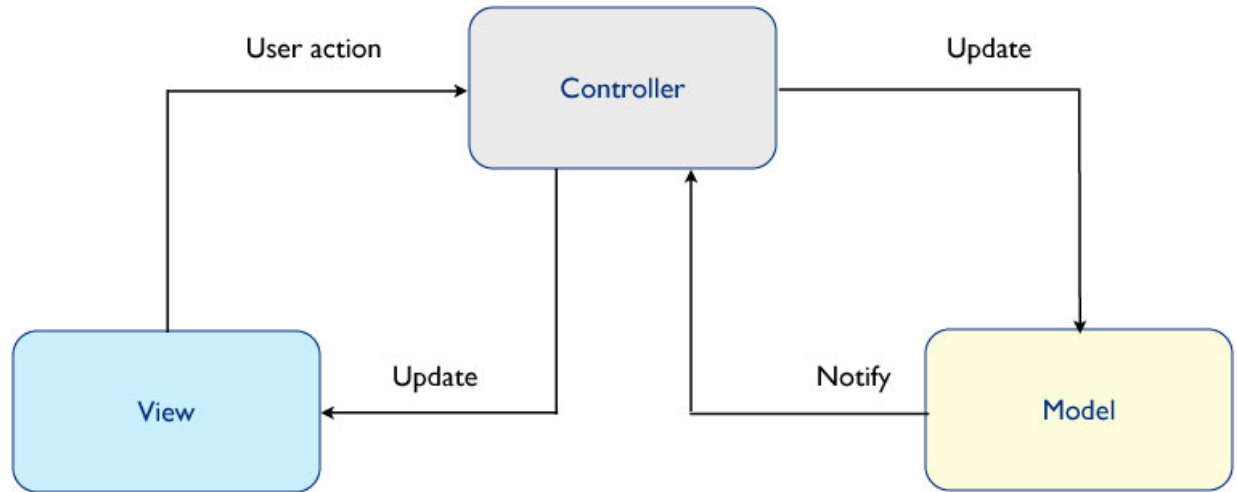
MVC

37

Modèle

Vue

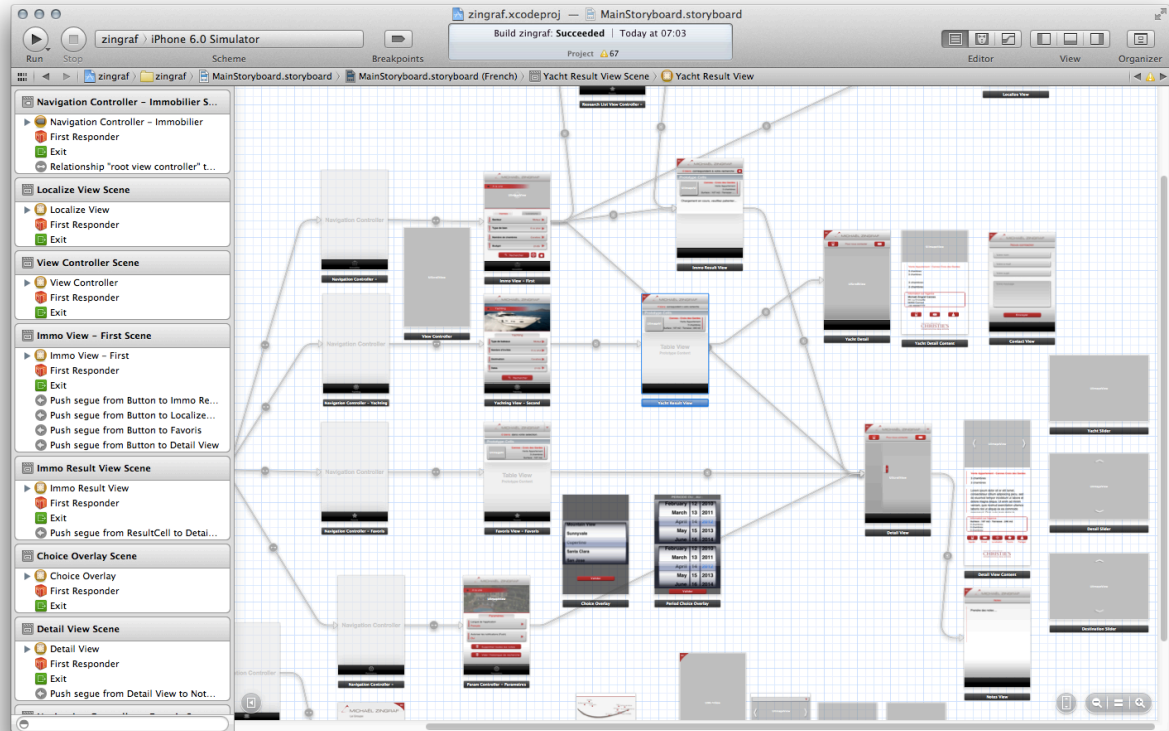
Contrôleur



Storyboard

38

Représente
l'intégralité des
écrans ainsi que
leurs
enchainements



Storyboard

39

- Un storyboard commence toujours par un point d'entrée, représenté par une flèche.
- A partir de ce point, deux catégories d'éléments peuvent être ajoutés :
 - Les conteneurs
 - Les vues

Vues & Conteneurs

40

- Un conteneur va, comme son nom l'indique, contenir des vues.
- Les conteneurs principaux sont :
 - TabBar Controller
 - Permet de créer plusieurs onglets en bas de l'écran (*Maximum 5*) et d'y rattacher des vues (*une par onglet*)
 - Navigation Controller
 - Permet de créer une « pile » d'écran. Cela donne la possibilité à l'utilisateur de revenir en arrière avec un bouton « Retour »

Vues & Conteneurs

41

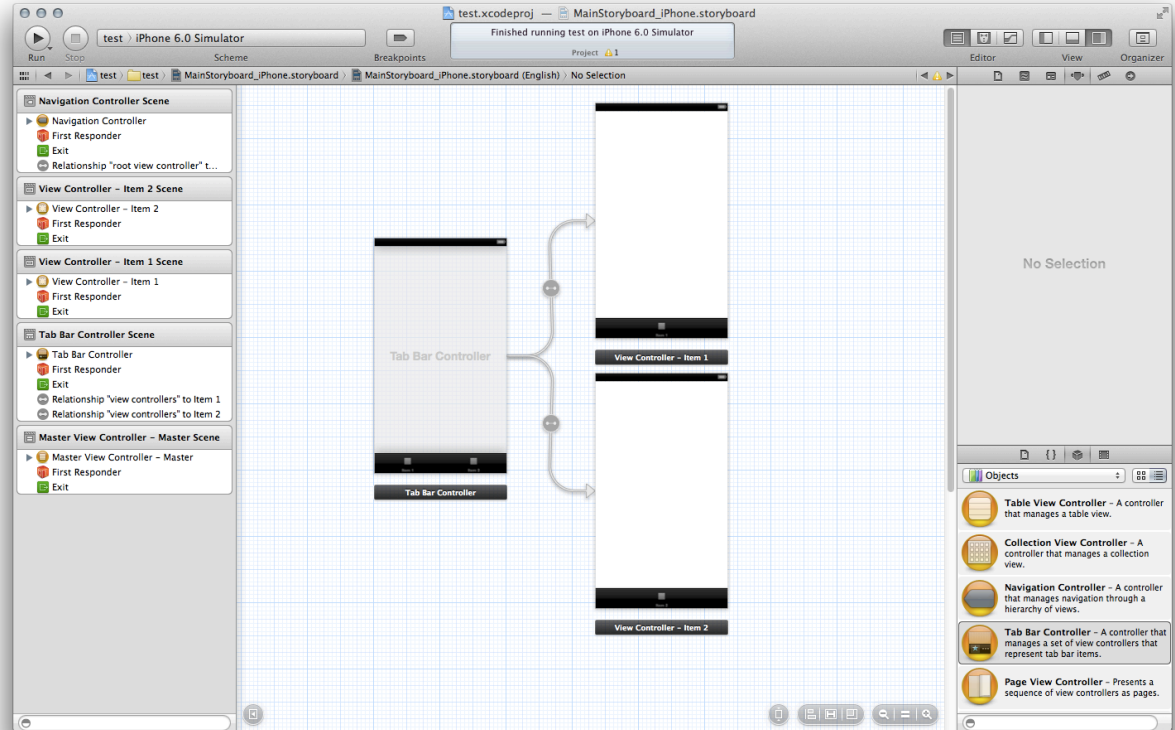


Vues & Conteneurs

42

Ajout d'un
TabBar
Controller dans
un Storyboard

*A noter ici,
l'ajout de 2
View Controller
en plus*



Vues & Conteneurs

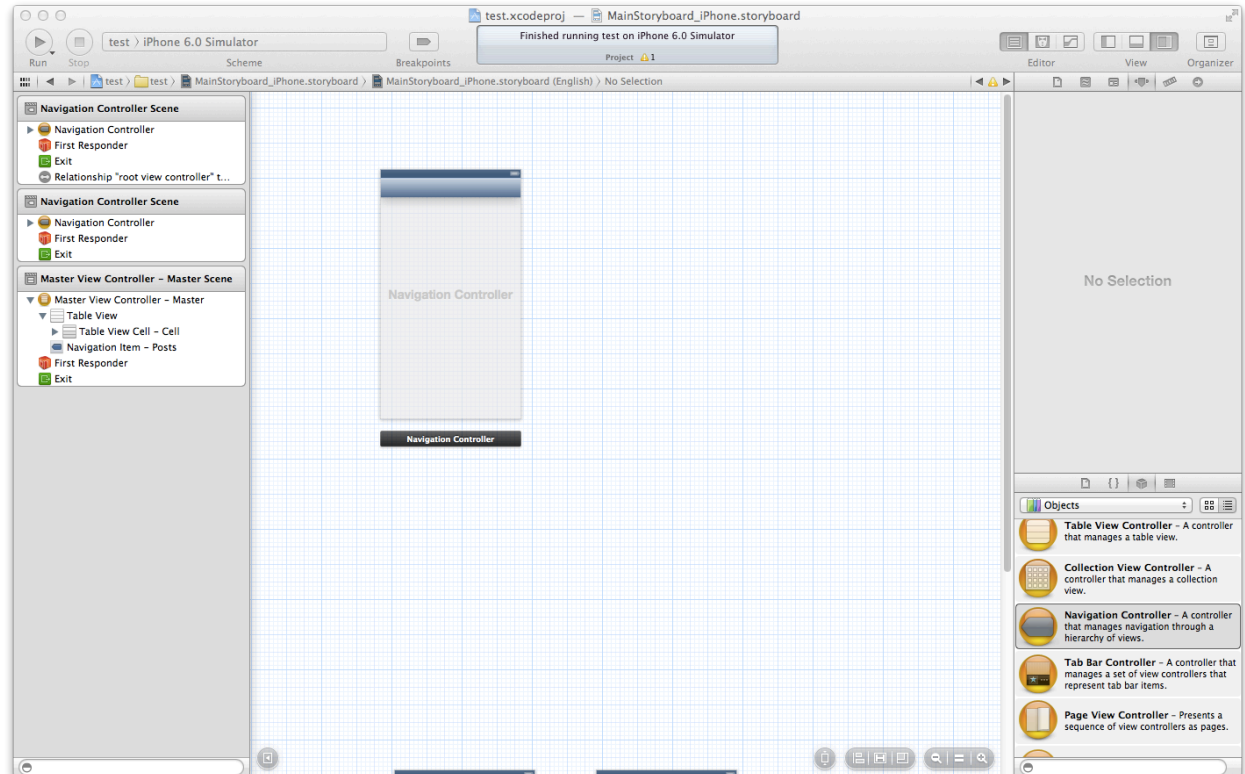
43



Vues & Conteneurs

44

Ajout d'un
Navigation
Controller dans
un Storyboard

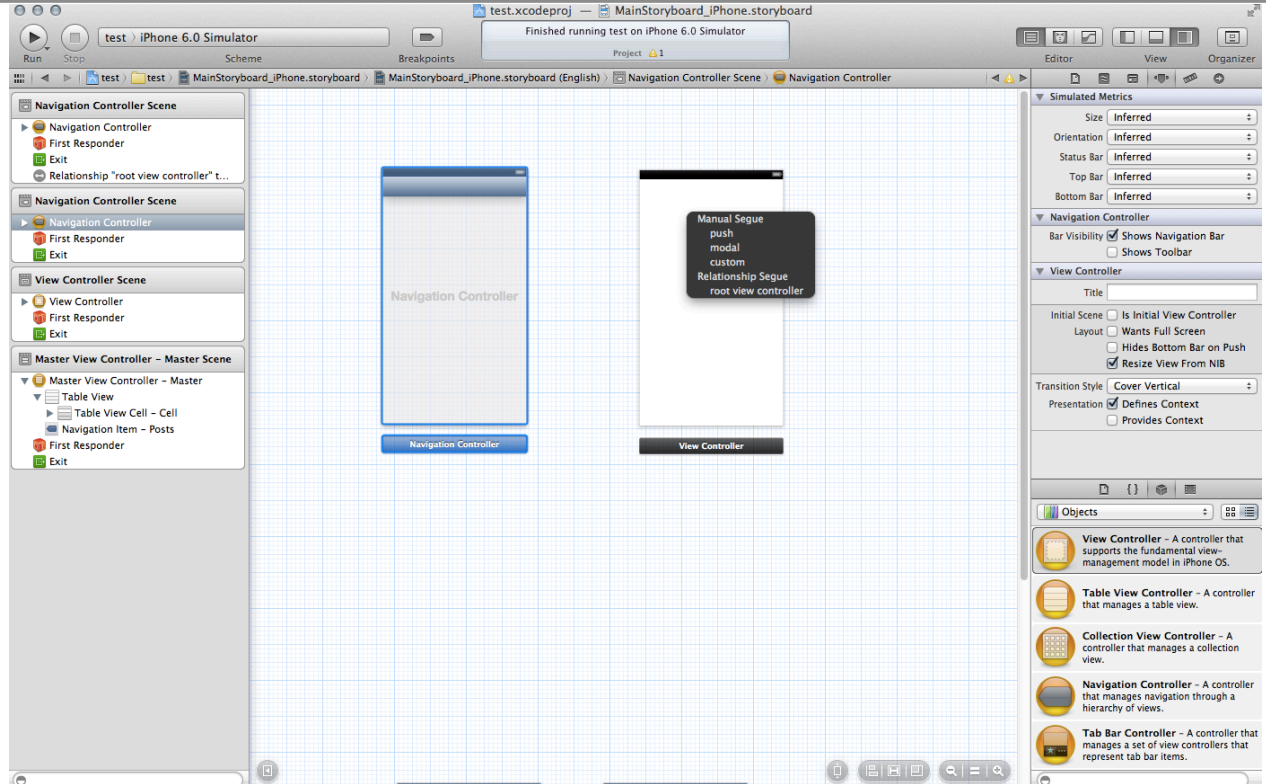


Vues & Conteneurs

45

Ajout d'un View Controller

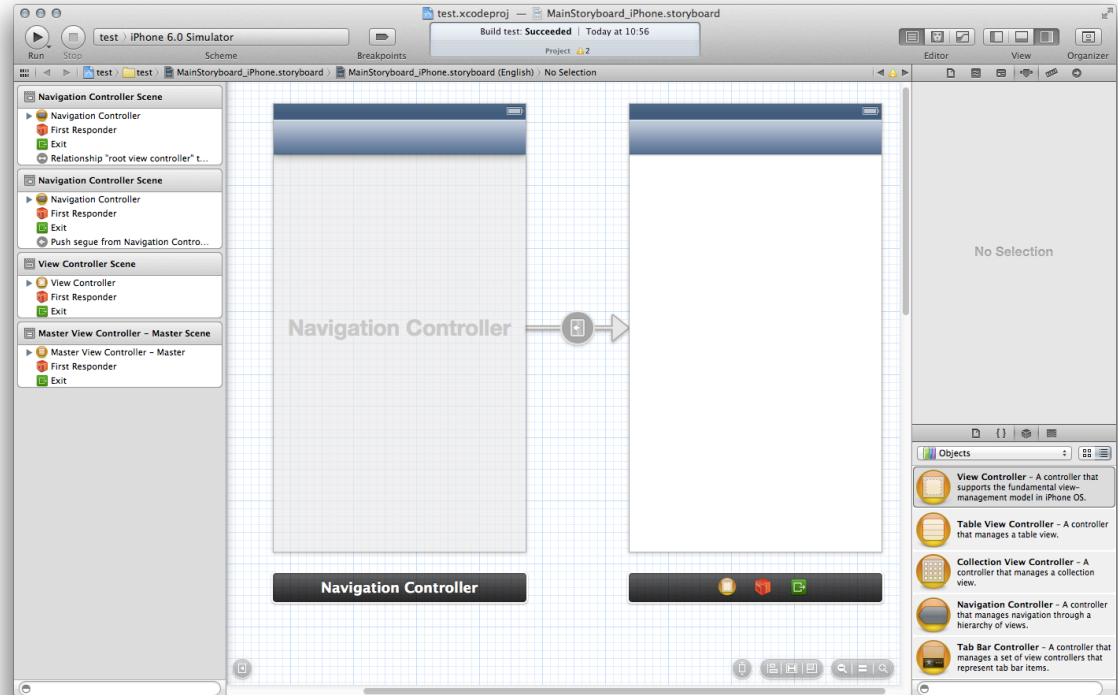
Choix de la **méthode** de transition : *Root view controller* ou *Push*



Vues & Conteneurs

46

Une fois la transition validée et activée.



Vues & Conteneurs

47

- Chaque écran est contrôlé par un **Controller**. (*Parfois plus d'un*)
- Un contrôleur gère l'ensemble des messages des vues qu'il possède.
- Dans le cas d'un TabBar Controller ou Navigation Controller, le premier contrôleur ajouté à ceux-ci, sera considéré comme le **RootView Controller**.

Vues & Conteneurs

48

- Il existe plusieurs types de View Controller, pour les plus courants :
 - **View Controller** : Le contrôleur par défaut.
 - **Table View Controller** : Très utilisé, celui-ci permet de gérer l'affichage d'une liste de données.
 - **Collection View Controller** : Moins fréquemment utilisé, sert à présenter un ensemble de vues, sous forme de mosaïque.

Vues & Conteneurs

49

- ▣ Une vue représente la partie purement esthétique.
- ▣ Elle définit l'apparence, ainsi que les contrôles qui seront utilisables.
- ▣ Exemple de vue :
 - TableView, collectionView, ImageView, TextView, WebView, MapView, ScrollView...

Vues & Conteneurs

50

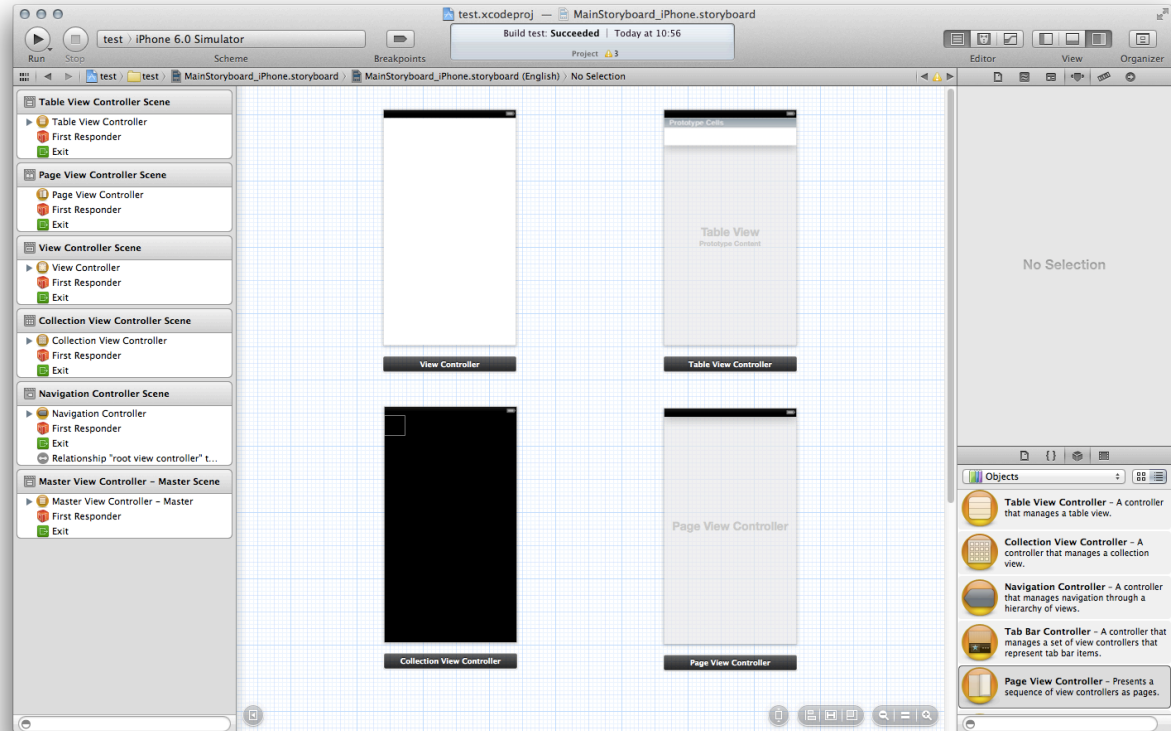
Ajout d'un
contrôleur

1. View

2. Table View

3. Collection

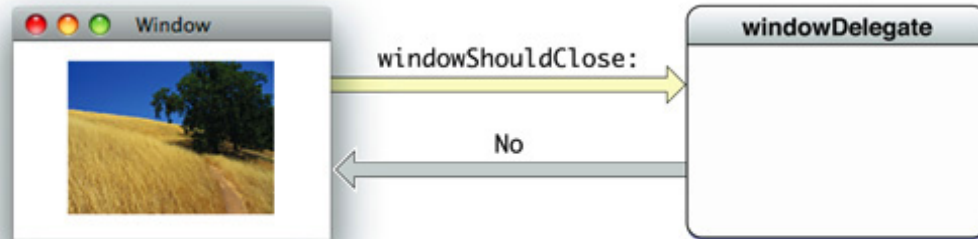
4. Page



Delegate

51

- Le *delegate* correspond généralement au contrôleur.
- Il est en charge de répondre aux événements déclenchés côté vue / utilisateur.



Composants & Événements

52

- Afin de définir l'apparence au sein de la vue, vous utiliserez différents composants, tels que :
 - Label
 - Button
 - Text Field
 - Slider
 - Switch...

Composants & Événements

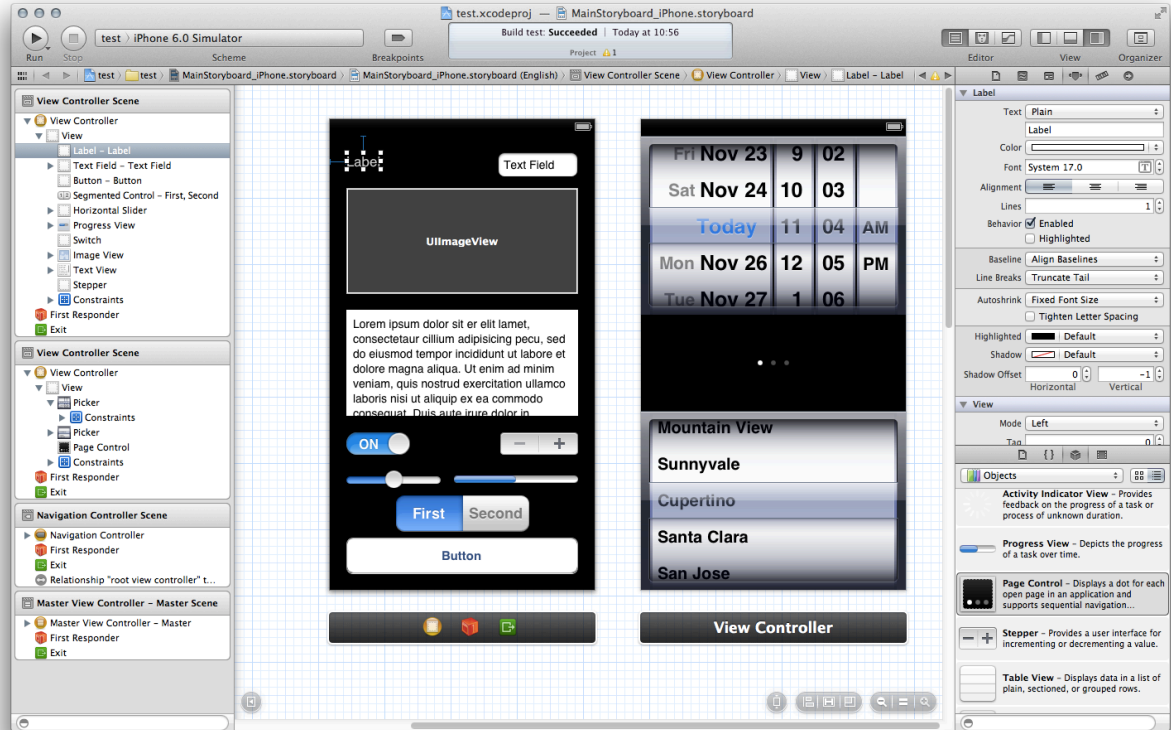
53

- Tous les composants possèdent un certain nombre d'action disponible. (**Events**)
- Pour traiter le clique sur un bouton par exemple :
 - Créer une méthode dans le contrôleur (Ex. : *btnSend*)
 - Lié l'événement désiré (Ex. : *Touch Up Inside*) à la méthode créée (*Graphiquement !*)
 - Et c'est tout !

Composants & Événements

54

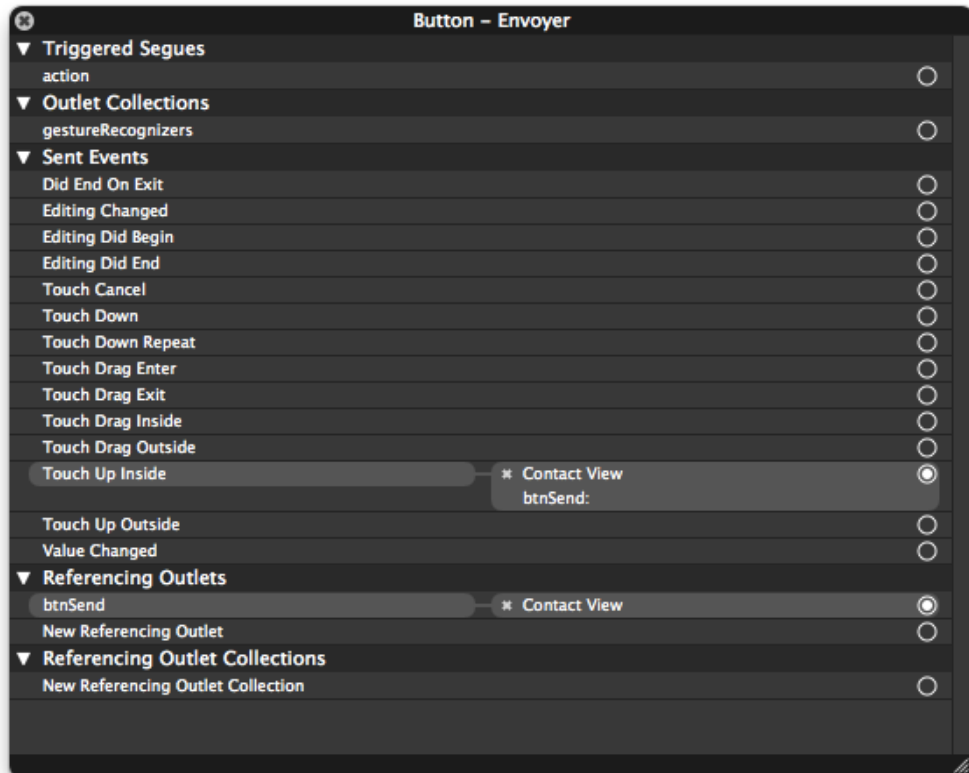
Ajout de
différents
composants



Composants & Événements

55

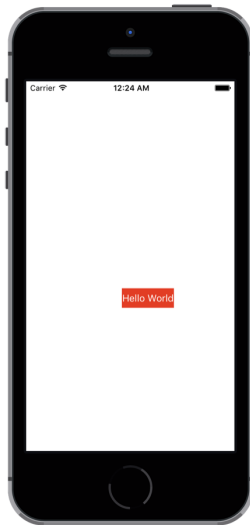
Liste des
événements
possible pour
un bouton



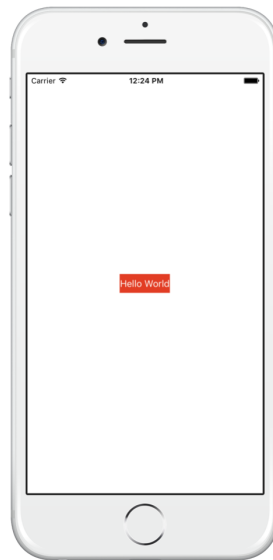
Auto Layout - Problématique

56

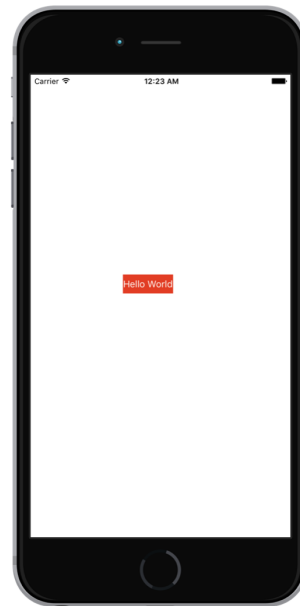
Nouveaux
iphones
entraînent de
nouvelles
résolutions.



iPhone 5s



iPhone 6

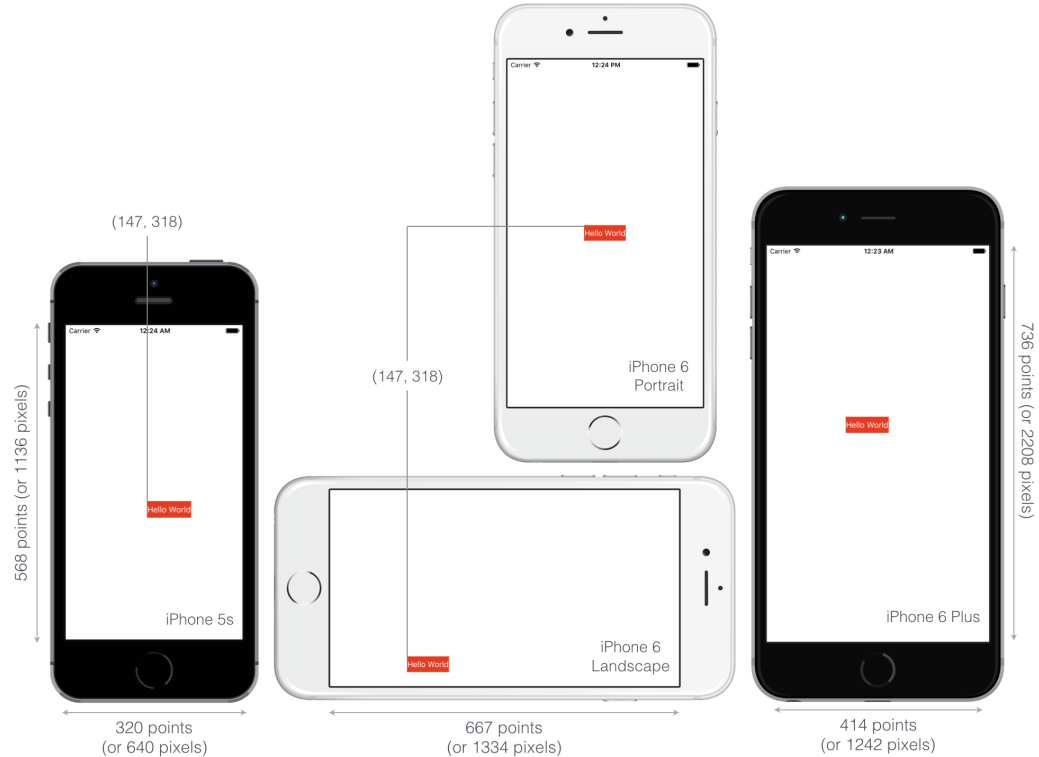


iPhone 6 Plus

Auto Layout - Problématique

57

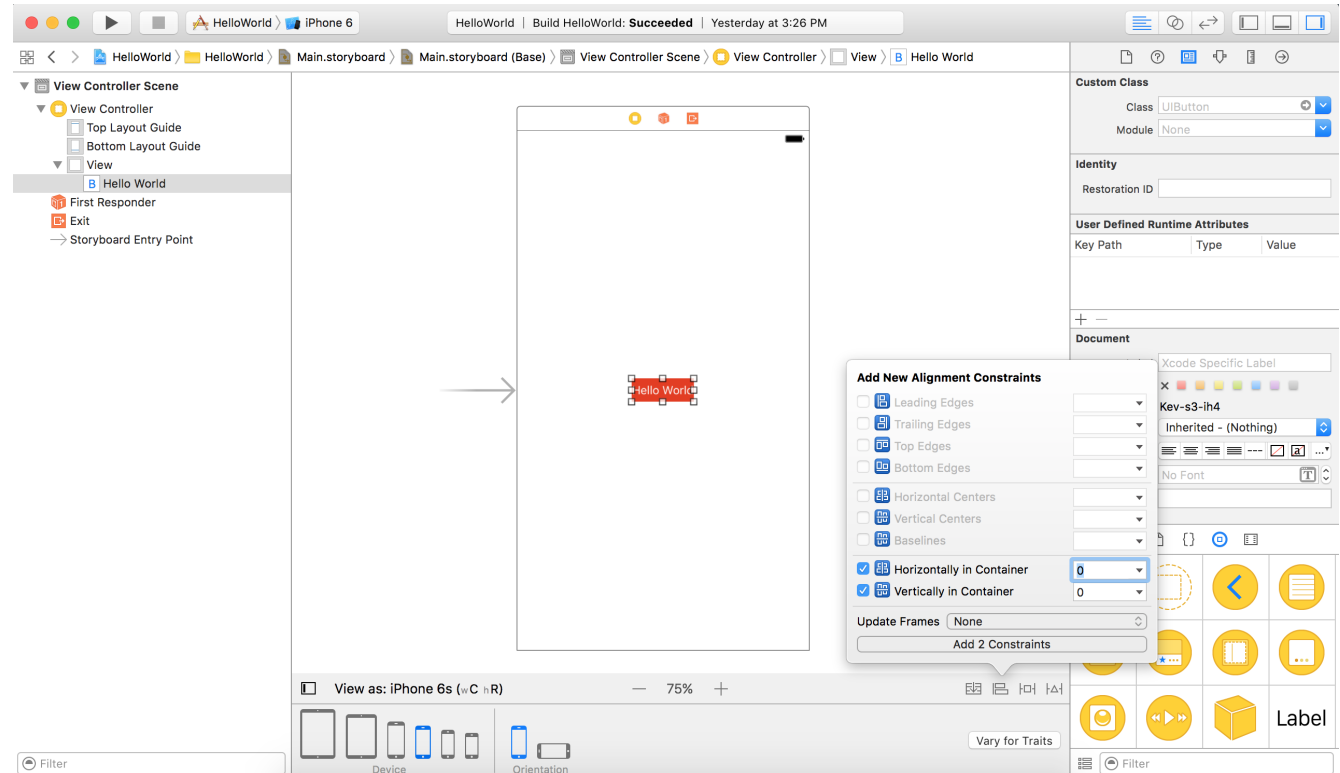
Gestion des
modes
portrait/paysa
ge



Auto Layout - Contraintes

58

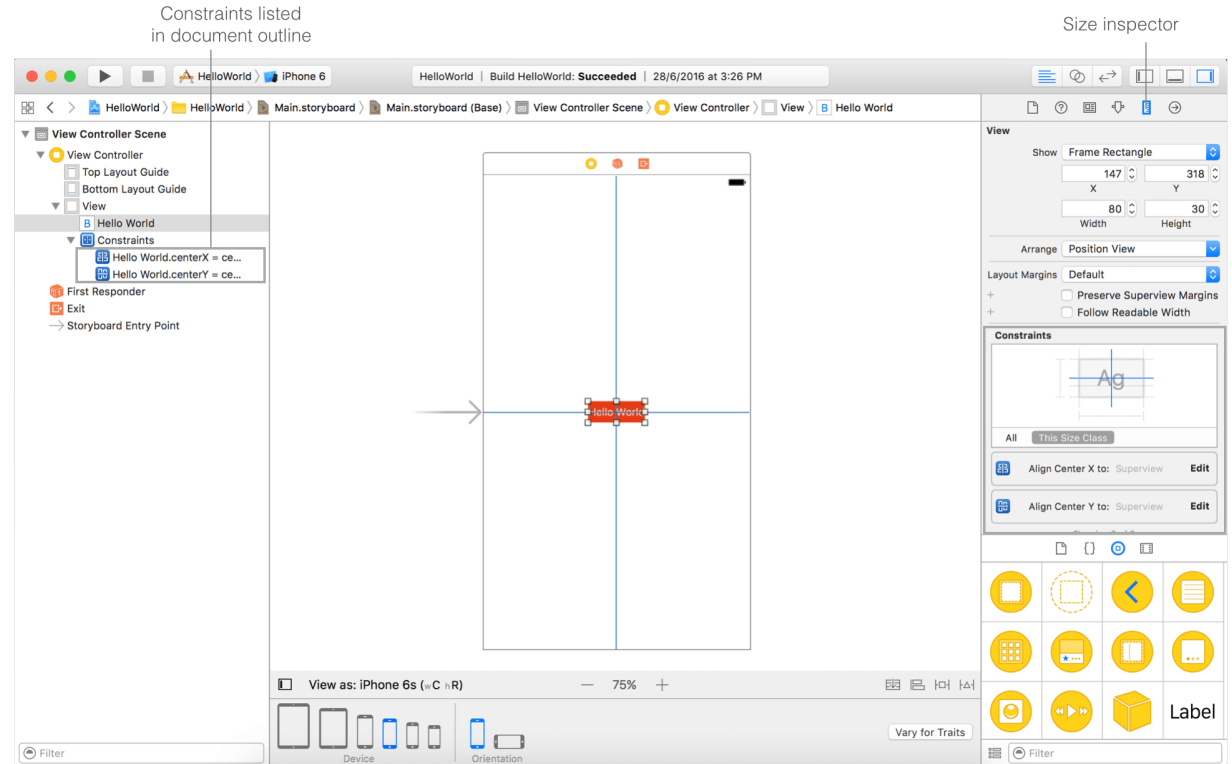
Ajout de
contrainte
permettant de
définir
l'emplacement
de manière
relative



Auto Layout - Contraintes

59

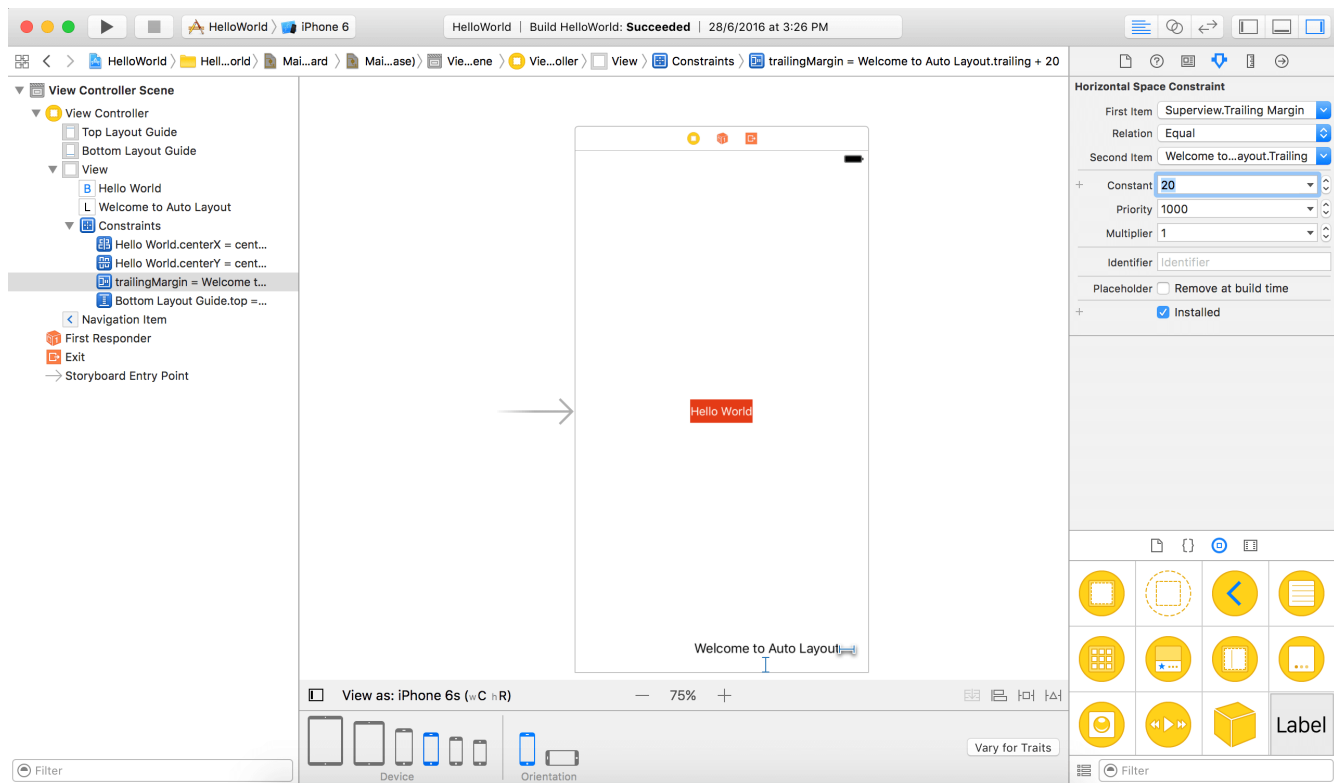
Exemple de
composant
centré



Auto Layout - Editor

60

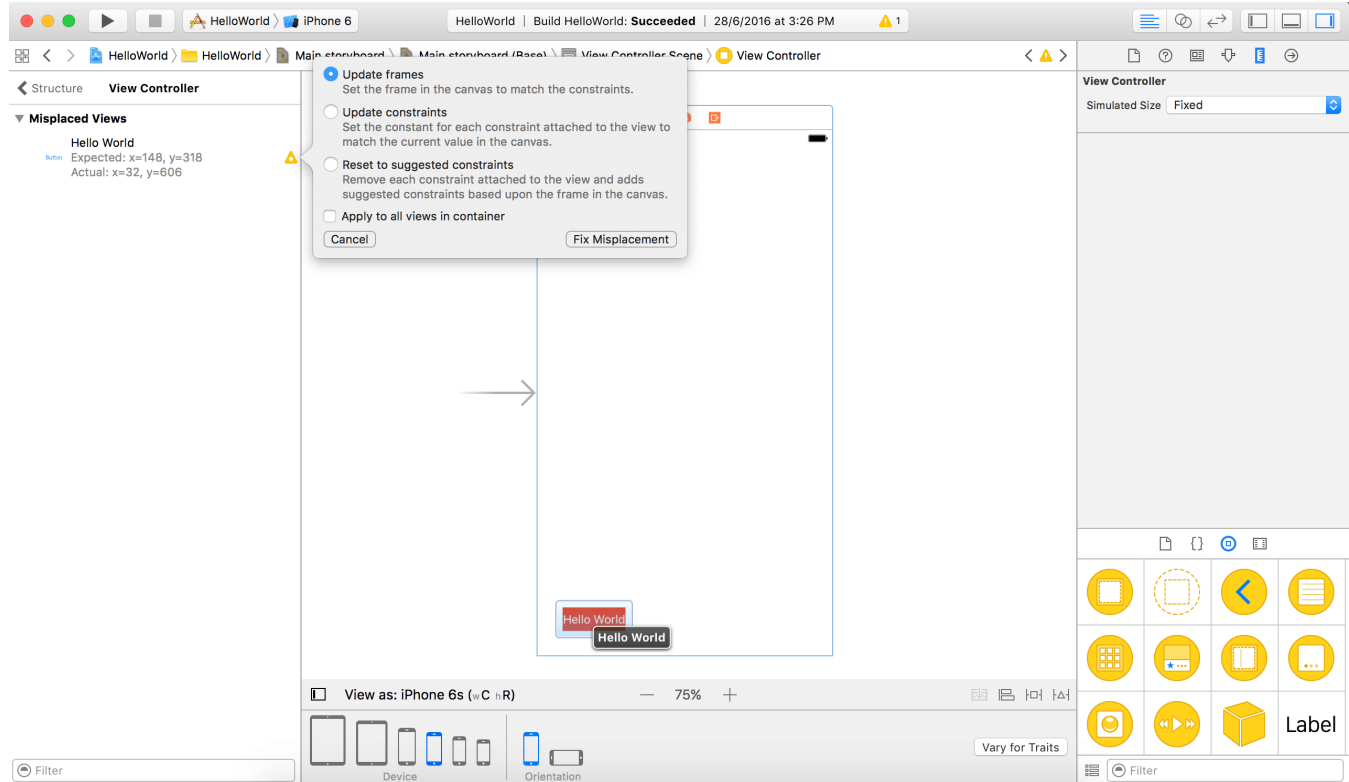
Avant-dernier onglet dans le panneau des propriétés à droite



Auto Layout - Conflits

61

Identifier et résoudre les conflits de positionnement



Auto Layout - Icônes

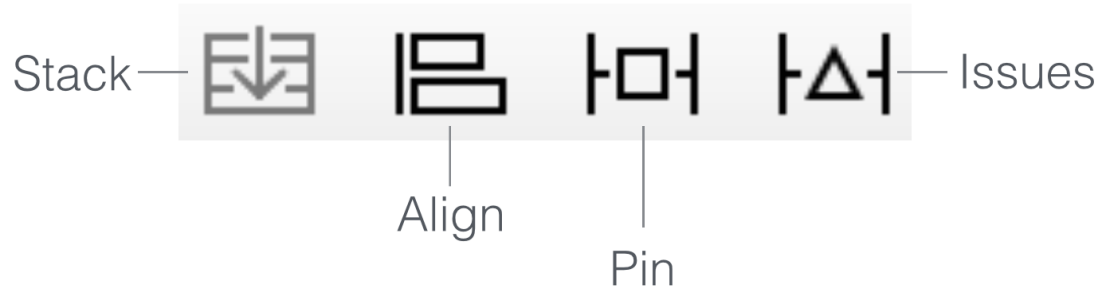
62

Stack – Regrouper les vues

Align – Contraintes d'alignements

Pin – Contraintes d'espacement

Issues – Résoudre les problèmes de positionnement



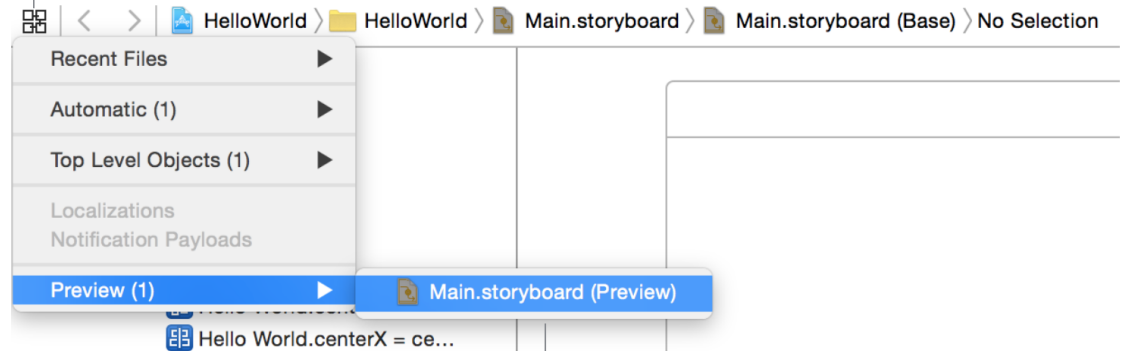
Auto Layout - Preview

63

Prévisualiser le
résultat selon
les iPhones

1

Open the assistant
pop-up menu



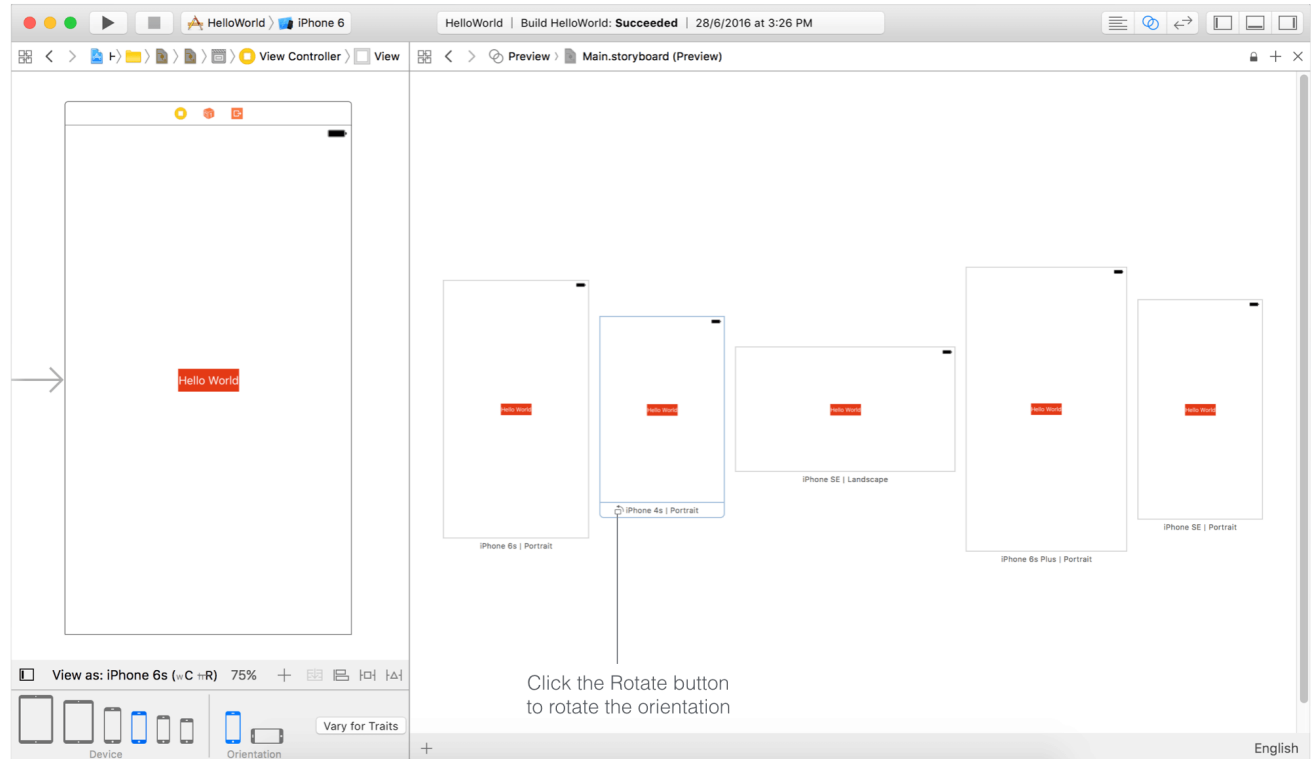
2

Hold option key and
click Main.storyboard (Preview)

Auto Layout - Preview

64

Prévisualiser le
résultat selon
les iPhones



Passer des paramètres

65

- Créer une liaison entre les 2 écrans, avec un nom de transition (*Storyboard segue* → *Identifier*)
- Editer la méthode `prepareForSegue` pour passer les paramètres comme suit :

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {  
    if ([[segue identifier] isEqualToString:@"showDetail"]) {  
        DetailViewController *controller = (DetailViewController *)[segue  
destinationViewController];  
        [controller setDetailItem:object] ;  
    }  
}
```

Table View – Alimentation

66

- ▣ Alimenter les données du NSMutableArray nommé self.objects dans la méthode viewDidLoad :

```
- (void) viewDidLoad
{
    [self.objects addObject:xxx] ;
    [self.objects addObject:xxx] ;
    ...
}
```

Table View – Taille de la liste

67

- Renseigner le nombre d'éléments de la liste dans les méthodes :

```
– (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {  
    return 1;  
}
```

```
– (NSInteger)tableView:(UITableView *)tableView  
numberOfRowsInSection:(NSInteger)section {  
    return self.objects.count;  
}
```

Table View – Affichage

68

- Compléter la méthode `cellForRowAtIndexPath` afin de définir l'affichage :

```
- (UITableViewCell *)tableView:(UITableView *)tableView  
cellForRowAtIndexPath:(NSIndexPath *)indexPath {  
    UITableViewCell *cell = [tableView  
dequeueReusableCellWithIdentifier:@"Cell"  
forIndexPath:indexPath];  
  
    NSDate *object = self.objects[indexPath.row];  
    cell.textLabel.text = [object description];  
    return cell;  
}
```


Table View – Transition

69

- Compléter la méthode `prepareForSegue` afin de transmettre le paramètre :

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue
sender:(id)sender {
    if ([[segue identifier] isEqualToString:@"showDetail"]) {
        NSIndexPath *indexPath = [self.tableView
indexPathForSelectedRow];
        DetailViewController *controller = (DetailViewController
*)[segue destinationViewController];
        [controller setDetailItem:self.objects[indexPath.row]];
    }
}
```