Hello World : Objective-C

TP1 : Développement mobile, environnement iOS

Partie 1 : Création du projet de base

1. Découverte de XCode

Comme vu pendant le cours nous allons créer notre première application iPhone OS. Pour cela lancer XCode et créer un projet selon vos préférences.

Dans un premier temps nous allons créer une single view application.

Nommer votre projet à votre guise de même vous devez définir le bundle identifier. Ce dernier peut être associé au packageName en Java.

2. Lancement de l'émulateur

Une fois le projet créé, nous allons le lancer une première fois dans l'émulateur. Nous pouvons ainsi voir que le code de base compile et s'exécute correctement.

Afin de pouvoir économiser les performances de la VM l'émulateur d'iPhone 5 est conseillé.

3. Let's hack !

Prenons le temps d'appréhender l'environnement de développement, retrouvons les bases que l'on peut connaitre dans Eclipse ou Intellij' Idea.

Partie 2 : Hello World console et lancement de l'application

Dans cette première partie nous nous intéresserons plus à la partie console d'exécution qu'à la partie graphique qui sera l'objet de la troisième partie du TP.

1. Création de la première classe

Le langage Objective-c est un langage orienté objets, il est donc composé de classes, d'attributs, de méthodes …

Nous allons donc créer un premier couple de fichier HelloWorld.h et HelloWorld.m.

Le but de la classe va être d'affecter une valeur à un attribut de la classe puis d'afficher celui-ci dans la console d'exécution

```
//
//  HelloWorld.h
//

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


@interface Helloworld : NSObject

@property (nonatomic, retain) NSString * text;

- (void) printHelloWorld;

@end
```

```
//
//  HelloWorld.m
//


#import "HelloWorld.h"


@implementation HelloWorld : NSObject
@synthesize text;

- (void) printHelloWorld
{
      self.text = @"Hello World";
      NSLog(@"Mon premier texte console : %@", self.text);
}

@end
```

Cette méthode affiche donc le texte au préalable assigné en dur dans le code.

Nous allons maintenant instancier cette classe dans le contrôleur principal de l'application puis lancer l'exécution de notre application sur l'émulateur.

Pour se faire se reporter au cours concernant la partie instanciation d'une classe.

2. Fonction NSLog

La fonction NSLog utilisé ci-dessus est l'équivalent du println en C. Elle permet d'afficher des informations dans la console. Attention pour afficher une variable il est impératif d'utiliser le bon type.

3. Aller plus loin

Question : Rendez le texte affiché paramétrable. Deux solutions doivent être envisagées :

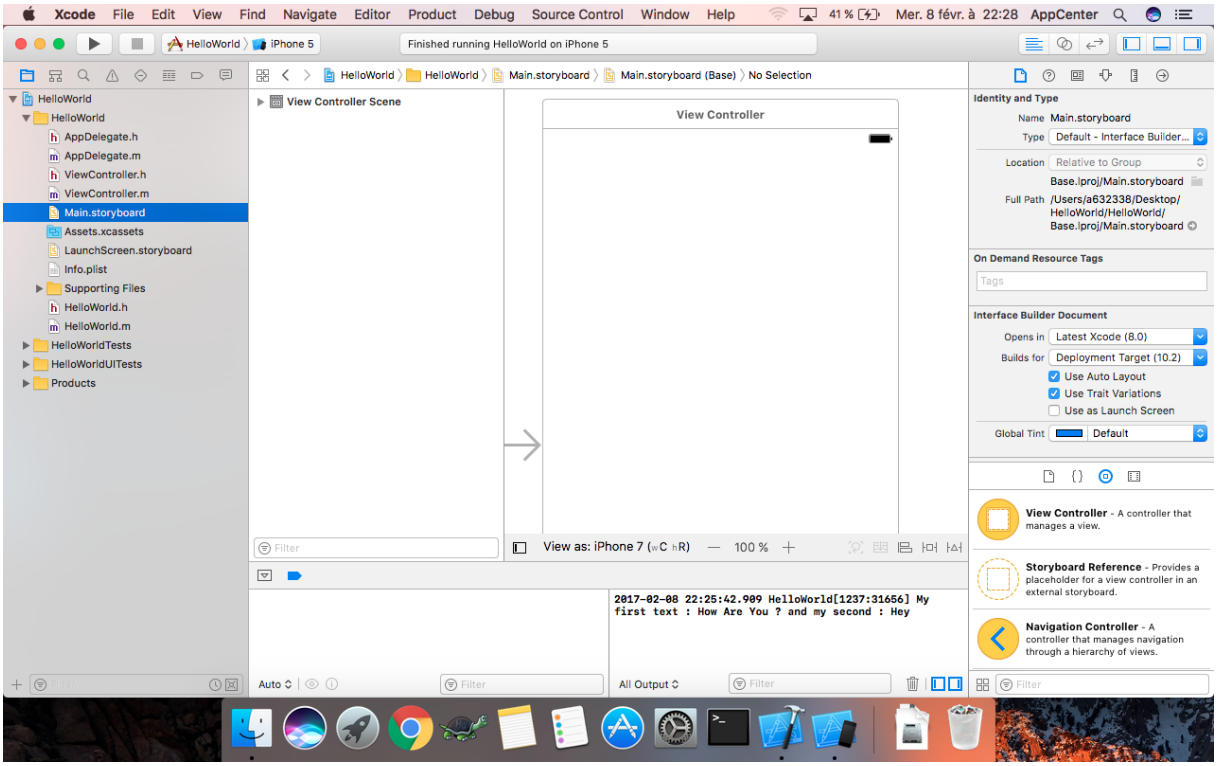
- Assignation de la valeur de l'attribut hors de la classe.
- Passage en paramètre de fonction de la valeur attendue.
- (Bonus : Un mix des deux solutions avec plusieurs chaines de caractères affichées sur la même ligne.)

Partie 3 : Manipulation d'éléments graphiques

Nous allons dans cette partie prendre en main l'aspect graphique du développement iOS, affichage de texte, de boutons et création d'interactions entre eux.

1. Affichage d'un texte via le storyboard

Afin de réaliser notre HelloWorld il va falloir ajouter un objet UILabel à la vue principale de notre storyboard.



Une fois ce label ajouté vous pouvez lancer l'application et admirer un parfait Hello World.

2. Affichage d'un texte via le code

Si nous prenons l'implémentation de notre vueController : viewController.m, nous allons y trouver toute l'implémentation de ce qui est fait via le storyboard.

Vous avez cette méthode que nous allons compléter afin d'afficher un texte supplémentaire.

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

Question : A l'aide de la documentation UILabel (https://developer.apple.com/reference/uikit/uilabel) ajouter un texte en haut à gauche de votre écran d'émulateur.

3. Affichage d'un bouton et changement de page.

Nous allons ici créer une deuxième vue et un enchainement de vue via un bouton.

Vous devez donc ajouter les éléments suivants :

- Une nouvelle vue (ViewControler)
- Un bouton
- Des informations sur la seconde vue pour les différencier.

Il faut donc ensuite lier tous ces éléments.

Indice : le clic droit est votre ami et la création de classe de contrôleur peuvent être nécessaire.

4. Modification du texte affiché dynamiquement

Nous allons à présente mettre en pratique les liens entre interface graphique et code.

Sur la seconde vue nous allons ajouter un champ texte et un bouton. Une fois ces éléments graphiques mis en place il va falloir les lier entre eux afin de pouvoir définir un texte via l'input et d'afficher la valeur de celui-ci lors de l'appui sur le bouton.

L'icône des doubles cercles permets de mettre en parallèle dans XCode la vue code et graphique.

Indice : Il va falloir ajouter des Outlets et des actions.