

iPhone SDK: Accès au carnet d'adresses de l'iPhone

par [Morvan Mikaël \(Morvan Mikaël\)](#) (Blog)

Date de publication : 03 juin 2008

Dernière mise à jour :

Cet article est un tutoriel montrant la méthode d'accès au carnet d'adresses de l'iPhone. Les frameworks mis en oeuvre sont le framework AddressBook et AddressBookUI.

I - Introduction.....	3
II - Description de l'application.....	3
III - Nouveau projet dans XCode.....	4
IV - Modification de l'interface avec Interface Builder.....	4
V - Le Framework AddressBook.....	4
VI - Modification du code pour accéder au framework AddressBook.....	5
VII - Lien IHM et Code dans Interface Builder.....	9
VIII - C'est terminé.....	10
IX - Conclusion.....	10
X - Remerciements et liens.....	10

I - Introduction

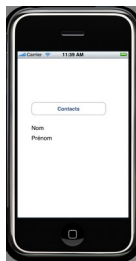
Ce tutoriel présente l'accès à la base de données des contacts de l'iPhone. Je décris pas à pas la réalisation d'une application simple permettant d'appeler le framework AddressBook et AddressBookUI.

Pour réaliser cette application, j'ai utilisé la version bêta 6 de l'iPhone SDK. Cette version corrige beaucoup de bugs présents dans la bêta 5 (notamment le "un peu moins bugué Interface Builder") et apporte plus de stabilité à l'ensemble. On sent bien que la version finale du framework est proche...

J'ai noté de nombreux changements dans le code généré lors de la création d'un nouveau projet (ViewController par défaut, nouveaux types d'application possible,...). Je vous conseillerai donc d'utiliser la dernière version de l'iPhone SDK pour suivre mon tutoriel.

II - Description de l'application

L'application réalisée dans le tutoriel n'est vraiment pas complexe. Elle consiste en un bouton et deux labels...



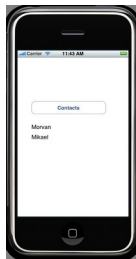
Application du tutoriel

Le clic sur le bouton affichera l'interface de sélection des contacts. Cette interface standard permet de parcourir la base de données des contacts et de sélectionner un contact voire une information d'un contact.



Liste des contacts

Une fois que la sélection du contact sera réalisée, le nom et le prénom de la personne seront affichés dans les labels.

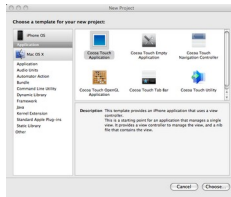


Sélection d'un contact

Bon, maintenant que je vous ai fait envie avec cette petite application, je vais vous montrer comment la réaliser.

III - Nouveau projet dans XCode

Ouvrez votre XCode et allez dans le menu "File" puis "New Project...". Vous arrivez sur la fenêtre suivante:



Choix d'un nouveau projet

Sélectionnez l'item "Cocoa Touch Application" et cliquez sur le bouton "Choose..." en bas à droite de la fenêtre.

Donnez un nom à votre projet. Pour ma part ce sera "tuto2".

Dans la liste des fichiers présents, on peut constater des changements par rapport à la version bêta 5: le ViewController. Les plus attentifs remarqueront que le ViewController a en plus son propre fichier ".XIB" permettant de modifier l'IHM du ViewController avec Interface Builder. Je ne vais pas m'étendre plus avant sur le ViewController mais le fait d'avoir plusieurs fichiers ".XIB" séparés permet de bien séparer le code entre les différentes fonctionnalités d'une application. Et accessoirement, ça permet le travail en équipe.

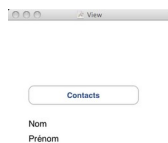
On va maintenant créer notre belle interface à l'aide d'Interface Builder en double-cliquant sur le fichier tuto2ViewController.xib

IV - Modification de l'interface avec Interface Builder

Ajoutez un "Round Rect Button" et deux "Label" sur la "View" en réalisant un drag drop depuis la librairie de composants.

Donnez les propriétés du bouton, saisissez "Contacts" comme titre et faites de même pour les labels en saisissant "Nom" et "Prénom"

Vous devriez obtenir l'interface suivante:



View dans Interface Builder

Enregistrez et revenez à XCode

V - Le Framework AddressBook

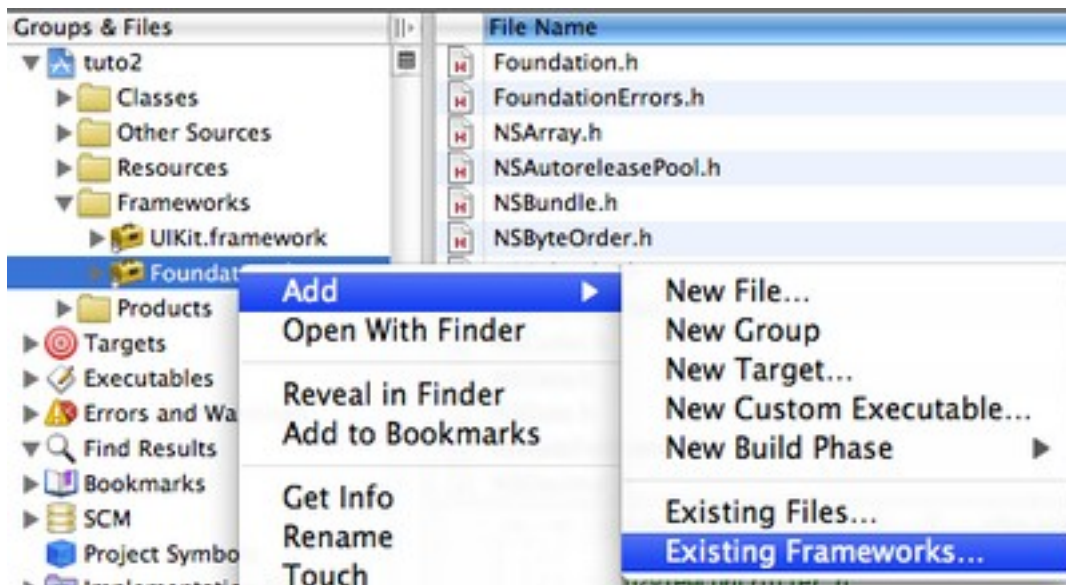
Un petit aparté sur les frameworks AddressBook et AddressBookUI avant de continuer notre application.

Le framework **AddressBook** permet d'accéder à la base de données de contacts de l'iPhone. Cette base de données stocke les contacts et leurs caractéristiques. Les applications telles que l'envoi de mail ou l'envoi de SMS utilisent cette base de données.

Le framework **AddressBookUI** fournit une IHM permettant d'afficher le contenu de la base de données des contacts. C'est ce framework que nous allons utiliser dans le tutoriel.

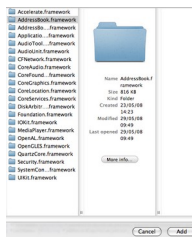
VI - Modification du code pour accéder au framework AddressBook

Avant de pouvoir utiliser les frameworks, il faut les déclarer dans le projet. Pour se faire, on clic droit (ou cmd clic) sur le répertoire "Frameworks" dans la liste des fichiers du projet.



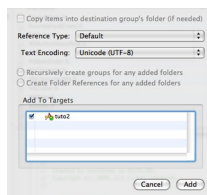
Import des frameworks

et on sélectionne le framework AddressBook dans la liste des frameworks proposés



Sélection du frameworks

et bien sûr on le rajoute au projet



Ajout du frameworks

Il faut refaire la même chose pour le framework AddressBookUI

Afin de pouvoir utiliser l'IHM d'accès à la base de données des contacts, il faut que notre ViewController implémente une Interface: ABPeoplePickerNavigationControllerDelegate

Cette implémentation permet au ViewController d'appeler le Framework en s'assurant qu'il contient bien toutes les méthodes demandées par l'interface. Les méthodes à implémenter sont les suivantes:

- peoplePickerNavigationController:shouldContinueAfterSelectingPerson
- peoplePickerNavigationController:shouldContinueAfterSelectingPerson:property:identifier:
- peoplePickerNavigationControllerDidCancel

Les deux premières méthodes permettent pour la première d'avoir l'utilisateur sélectionné et pour la seconde d'avoir la propriété sélectionnée. Pour la dernière méthode, elle permet de gérer l'événement d'annulation.

On va coder un peu maintenant !

Notre code source initial pour le tuto2ViewController.h

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

@interface tuto2ViewController : UIViewController {
}

@end
```

et le code modifié pour prendre en compte la délégation

```
#import <UIKit/UIKit.h>
#import <AddressBookUI/ABPersonViewController.h>
#import <AddressBookUI/ABPeoplePickerNavigationController.h>

@interface tuto2ViewController : UIViewController <UIActionSheetDelegate,
ABPeoplePickerNavigationControllerDelegate>{
}
```

Maintenant on va rajouter les liens entre l'interface et notre code par le biais des IBOutlet. On en profite pour ajouter un événement pour gérer le clic sur le bouton de l'IHM.

On modifie le code du tuto2ViewController.h comme suit

```
#import <UIKit/UIKit.h>
#import <AddressBookUI/ABPersonViewController.h>
#import <AddressBookUI/ABPeoplePickerNavigationController.h>

@interface tuto2ViewController : UIViewController <UIActionSheetDelegate,
ABPeoplePickerNavigationControllerDelegate>{
    IBOutlet UILabel *labelNom;
    IBOutlet UILabel *labelPrenom;
}

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

On va pouvoir modifier le .m

Le code initial du .m

```
#import "tuto2ViewController.h"

@implementation tuto2ViewController

/*
 Implement loadView if you want to create a view hierarchy programmatically
- (void)loadView {
}
*/

/*
 Implement viewDidLoad if you need to do additional setup after loading the view.
- (void)viewDidLoad {
    [super viewDidLoad];
}
*/

- (BOOL)shouldAutorotateToInterfaceOrientation:
(UIInterfaceOrientation)interfaceOrientation {
    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning]; // Releases the view if it doesn't have a superview
    // Release anything that's not essential, such as cached data
}

- (void)dealloc {
    [super dealloc];
}

@end
```

On va ajouter les méthodes demandées par le delegate. Le code final est le suivant:

```
#import "tuto2ViewController.h"

@implementation tuto2ViewController

- (IBAction)contactClicked:(id)sender {
    // Sur le clic du bouton, on initialise et on lance le controller

    ABPeoplePickerNavigationController *controller = [[ABPeoplePickerNavigationController
alloc] init];
    controller.peoplePickerDelegate= self;
    [self presentModalViewController:controller animated:YES];
}

- (BOOL)peoplePickerNavigationController:
(ABPeoplePickerNavigationController *)peoplePicker {ligne tronquée}
    shouldContinueAfterSelectingPerson:(ABRecordRef)person {
        NSString *firstName= (NSString *)ABRecordCopyValue(person,
kABPersonFirstNameProperty);
        NSString *lastName= (NSString *)ABRecordCopyValue(person, kABPersonLastNameProperty);
    }
}
```

```

// Je montre juste comment avoir accès à la personne sélectionnée
NSLog(@"Personne selectionnee: %@, %@", lastName, firstName);

// Si on souhaite ne pas accéder aux propriétés alors on retourne NO
// Avec la valeur YES, la fenêtre d'accès aux propriétés sera ouverte
return YES;
}

- (BOOL)peoplePickerNavigationController:
(ABPeoplePickerNavigationController *)peoplePicker {ligne tronquée}
    shouldContinueAfterSelectingPerson:(ABRecordRef)person property:
(ABPropertyID)property identifier:(ABMultiValueIdentifier)identifier{

// On ferme le controller
[peoplePicker dismissModalViewControllerAnimated:YES];
// et on le libère
[peoplePicker release];

// On récupère le nom et le prénom du contact sélectionné
NSString *firstName= (NSString *)ABRecordCopyValue(person,
kABPersonFirstNameProperty);
NSString *lastName= (NSString *)ABRecordCopyValue(person, kABPersonLastNameProperty);

// Et on les affiche dans les labels
[labelNom setText:lastName];
[labelPrenom setText:firstName];

return NO;
}

- (void)peoplePickerNavigationControllerDidCancel:
(ABPeoplePickerNavigationController *)peoplePicker{

// En cas d'annulation, on ferme le controller
[peoplePicker dismissModalViewControllerAnimated:YES];
// et on le libère
[peoplePicker release];

}

/*
Implement viewDidLoad if you want to create a view hierarchy programmatically
- (void)viewDidLoad {
}
*/

/*
Implement viewDidLoad if you need to do additional setup after loading the view.
- (void)viewDidLoad {
[super viewDidLoad];
}
*/

- (BOOL)shouldAutorotateToInterfaceOrientation:
(UIInterfaceOrientation)interfaceOrientation {
// Return YES for supported orientations
return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

- (void)didReceiveMemoryWarning {
[super didReceiveMemoryWarning]; // Releases the view if it doesn't have a superview
// Release anything that's not essential, such as cached data
}

```

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

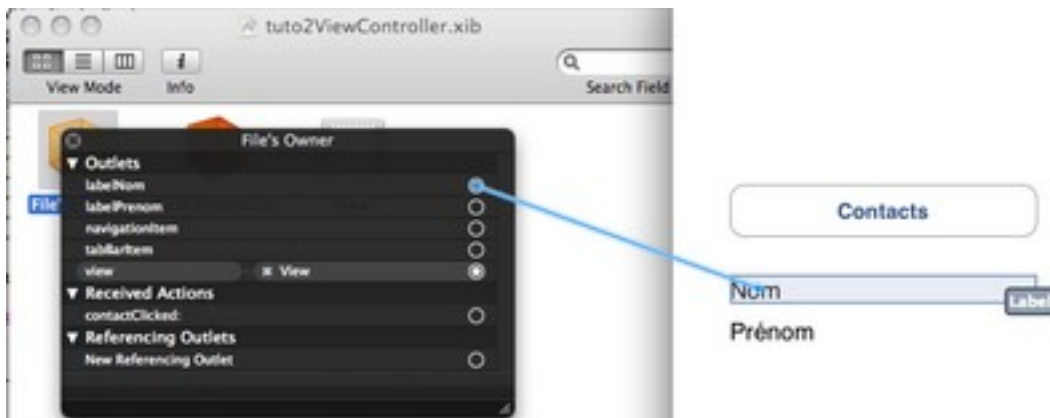
@end
```

Voilà, on a terminé de coder. Il reste à réaliser le lien entre le code et l'IHM à l'aide d'Interface Builder.

VII - Lien IHM et Code dans Interface Builder

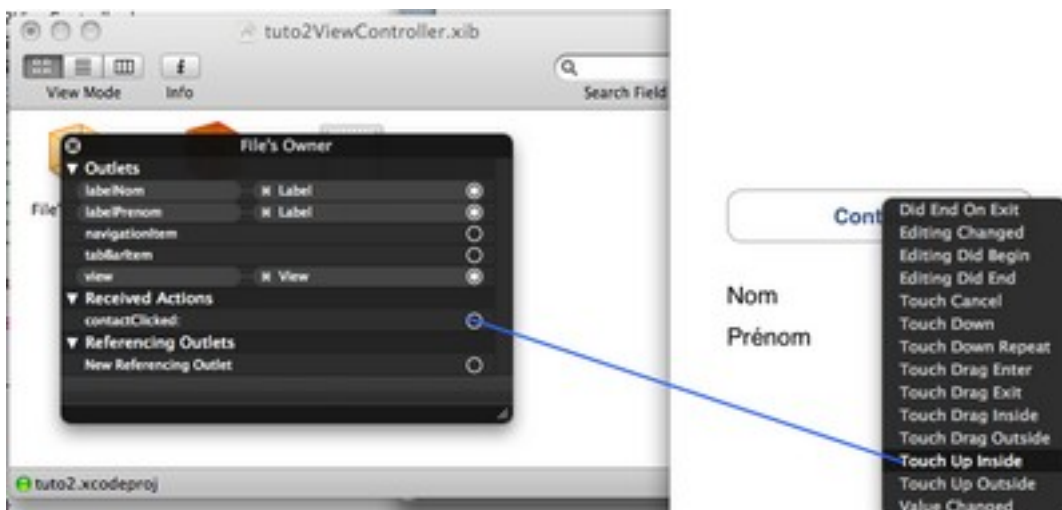
On double-clic sur le fichier tuto2ViewController.xib pour accéder à Interface Builder.

Un clic droit (ou cmd clic) sur le composant "File's Owner" donne accès aux Outlets et aux actions. On doit donc relier les Outlets vers les composants Label.



Liaison Outlet vers Label Nom

Et pour terminer, on relie l'événement "Touch Up Inside" du composant "Round Rect Button" à l'action "contactClick":



Liaison événement vers l'action contactClicked

On enregistre et on revient sous XCode.

VIII - C'est terminé

On a terminé.



Build and Go

On build et on lance. On doit obtenir l'application du début.

IX - Conclusion

Ce tutoriel a montré pas à pas comment tirer parti des IHM déjà toutes faites de l'iPhone. L'ajout de ces frameworks d'interface est un vrai plus pour le côté RAD du développement sur l'iPhone. Le gain de temps apporté est indéniable et le temps passé à appréhender le framework sera largement amorti.

Le framework d'accès à la base de données des contacts n'est pas le seul framework existant. Il existe par exemple un framework permettant de lire une vidéo dans un player vidéo et un autre framework permettant l'accès à la bibliothèque d'image de l'iPhone...

Mais ceci est pour de futurs tutoriels...

X - Remerciements et liens

Je voudrais remercier [diogene](#) et la rédaction Mac pour les corrections apportées à mon article.

Vous pourrez retrouver la documentation sur le site d'Apple concernant l'iPhone SDK. Il faudra bien entendu s'inscrire avant de pouvoir y accéder.

Lien vers l'iPhone Dev Center

Et un dernier lien vers mon article précédent concernant l'approche RAD sur l'iPhone

Approche RAD pour l'iPhone SDK avec Interface Builder