iOS 5 Programming Cookbook

第十章
通讯录

版本 1.0

翻译时间：2012-07-16

DevDiv 翻译： kyelup cloudhsu 耐心摩卡
wangli2003j3 xiebaochun dymx101
Jimmylianf BeyondVincent

DevDiv 校对： laigb kyelup

DevDiv 编辑： BeyondVincent

写在前面

目前，移动开发被广大的开发者们看好，并大量的加入移动领域的开发。

鉴于以下原因：

- 国内的相关中文资料缺乏
- 许多开发者对 E 文很是感冒
- 电子版的文档利于技术传播和交流

DevDiv.com 移动开发论坛特此成立了翻译组，翻译组成员具有丰富的移动开发经验和英语翻译水平。组员们利用业余时间，把一些好的相关英文资料翻译成中文，为广大移动开发者尽一点绵薄之力，希望能对读者有些许作用，在此也感谢组员们的辛勤付出。

关于 DevDiv

DevDiv 已成长为国内最具人气的综合性移动开发社区

更多相关信息请访问 DevDiv 移动开发论坛。

技术支持

首先 DevDiv 翻译组对您能够阅读本文以及关注 DevDiv 表示由衷的感谢。

在您学习和开发过程中，或多或少会遇到一些问题。DevDiv 论坛集结了一流的移动专家，我们很乐意与您一起探讨移动开发。如果您有什么问题和技术需要支持的话，请访问 DevDiv 移动开发论坛 或者发送邮件到 BeyondVincent@DevDiv.com，我们将尽力所能及的帮助你。

关于本文的翻译

感谢 kyelup、cloudhsu、耐心摩卡、wangli2003j3、xiebaochun、dymx101、jimmylianf 和 BeyondVincent 对本文的翻译，同时非常感谢 laigb 和 kyelup 在百忙中抽出时间对翻译初稿的认真校验，指出了文章中的错误。才使本文与读者尽快见面。由于书稿内容多，我们的知识有限，尽管我们进行了细心的检查，但是还是会存在错误，这里恳请广大读者批评指正，并发送邮件至 BeyondVincent@devdiv.com，在此我们表示衷心的感谢。

读者查看下面的帖子可以持续关注章节翻译更新情况

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译各章节汇总

另外，我们也为大家准备了 iOS6 新特征的相关内容，请参考下面的帖子：

iOS6 新特征：参考资料和示例汇总------持续更新中

目录

前言

参考帖子

[DevDiv翻译]iOS 5 Programming Cookbook翻译_前言

第 1 章 基础入门

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第一章_基础入门

第 2 章 使用控制器和视图

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第二章_使用控制器和视图(上)

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第二章_使用控制器和视图(下)

第 3 章　构造和使用 Table View

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第三章_构造和使用 Table View

第 4 章 Storyboards

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第四章_Storyboards

第 5 章　并发

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 翻译_第五章_并发

第 6 章 定位核心与地图

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 翻译_第六章_核心定位与地图

第 7 章 实现手势识别的功能

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 翻译_第七章_实现手势识别

第 8 章 网络，JSON，XML 以及 Twitter

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 翻译_第八章_网络，JSON，XML 以及 Twitter

第 9 章 音频和视频

参考帖子

[DEVDIV]iOS 5 Programming Cookbook 中文翻译_第九章_音频和视频

第 10 章 通讯录

10.0. 介绍

iOS 设备上，通讯录允许用户添加、删除和修改他们的联系人信息。通讯录可以包含个人通讯录或群组通讯录的一个集合。每个联系人都有包括关于自己的一些信息，如姓名、电话号码和电子邮件等。每个信息项可以是单独的一个值也可以是多个值。例如，姓就是单独的一个值，而电话号码可以包含多个值（e.g.，假如用户有两个家庭号码）。

在 iOS SDK 中的 AddressBook.framework 框架允许我们与设备上的通讯录进行数据交互。我们可以取得用户的通讯录上的全部数据，插入和修改信息，等等。

在应用程序上使用通讯录上的相关功能，可以按照以下步骤先给你的程序添加 AddressBook.framework：

1、在 Xcode 上点击你的工程图标；
2、选择你要添加 Address Book 框架到目标文件夹；
3、在屏幕的顶部选择 Build Phases 标签栏；
4、在 Biuld Phase 标签里，使用 Libraries box 寻找可拓展 Link Binary 并按下“+”按钮；
5、在显示出来的列表中，选择 AddressBook 并按下“Add”按钮（见图 10-1）。

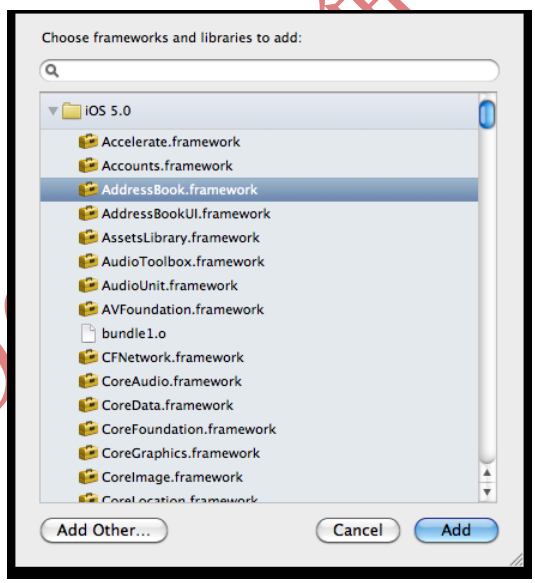

图 10-1

在给你的程序里添加了这个框架后，你想使用 address book 相关功能时，你必须在头文件或执行文件里包含这个框架的头文件，如下：

```
#import <UIKit/UIKit.h>
#import <AddressBook/AddressBook.h>
@interface RootViewController : UIViewController
@end
```

你可以在模拟器上使用 Address Book，但是模拟器上的通讯录里默认是空白的。假如你

想在模拟器上运行本章中的例子，那么你需要使用通讯录程序在电话里添加一些联系人信息。

我已经在我的 iOS 模拟器上的联系人里添加了 3 条信息，如图 10-2。

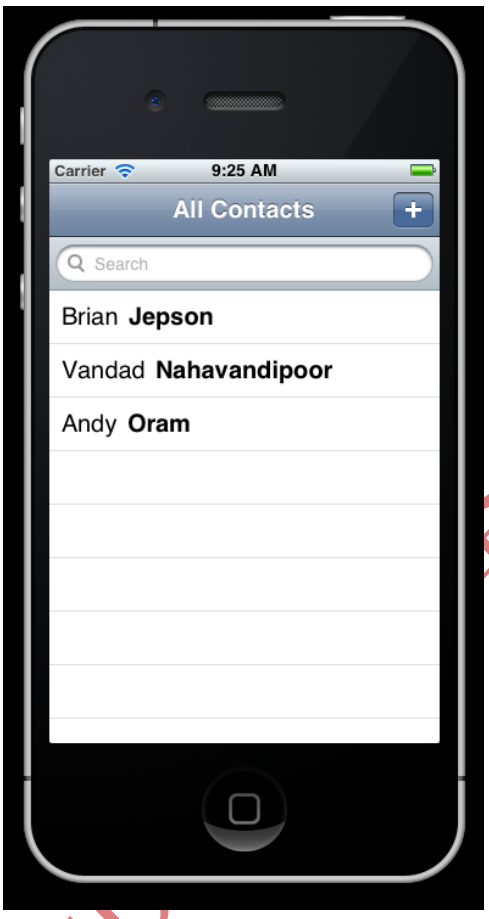

图 10-2

我建议在 iOS 模拟器上的通讯录里添加的内容应尽可能的多一些：多个工作于家庭电话号码，不同的地址，等等。只有这样的多样性，才能正确地测试通讯录框架的功能。

10.1. 取得 Address Book 一个引用

10.1.1. 问题

想要获得用户通讯录数据的一个引用。

10.1.2. 方案

在通讯录框架里使用 ABAddressBookCreate 功能：

```
- (BOOL) application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
ABAddressBookRef addressBook = ABAddressBookCreate();
if (addressBook != nil){
NSLog(@"Successfully accessed the address book.");
```

```
/* Work with the address book here */
/* Let's see if we have made any changes to the
address book or not, before attempting to save it */
if (ABAddressBookHasUnsavedChanges(addressBook)){
/* Now decide if you want to save the changes to
the address book */
NSLog(@"Changes were found in the address book.");
BOOL doYouWantToSaveChanges = YES;
/* We can make a decision to save or revert the
address book back to how it was before */
if (doYouWantToSaveChanges){
CFErrorRef saveError = NULL;
if (ABAddressBookSave(addressBook, &saveError)){
/* We successfully saved our changes to the
address book */
} else {
/* We failed to save the changes. You can now
access the [saveError] variable to find out
what the error is */
}
} else {
/* We did NOT want to save the changes to the address
book so let's revert it to how it was before */
ABAddressBookRevert(addressBook);
}
} else {
550 | Chapter 10: Address Book
/* We have not made any changes to the address book */
NSLog(@"No changes to the address book.");
}
CFRelease(addressBook);
} else {
NSLog(@"Could not access the address book.");
}
self.window = [[UIWindow alloc] initWithFrame:
[[UIScreen mainScreen] bounds]];
self.window.backgroundColor = [UIColor whiteColor];
[self.window makeKeyAndVisible];
return YES;
}
```

我创建了 doYouWantToSaveChange 本地变量并将它设置成 YES，这里仅仅为了演示所需，如果需要，我们可以取得通讯录中已经更改的内容（通过 ABAddressBookRevert 方法来取得）。例如，可以添加代码提示用户是否需要保存修改后的内容，假如用户选择否，则通讯录内容取得到原来的状态。关于 Address Book 框架导入到你的程序的更多信息，请参考本章的介绍部分。

10.1.3. 讨论

要得到用户通讯录数据的一个引用，我们必须使用 ABAddressBookCreate 函数。假如这个通讯录不能访问，这个函数将返回 ABAddressBookRef 数据类型的 nil 值。你必须在访问通讯录之前检查这个函数返回的 nil 值。尝试去修改一个返回 nil 值的通讯录将会因运行出错而终止你的程序。

再给用户通讯录检索了一个引用后，你可以开始更改联系方式，读取信息项，等等。假如你已经更改了通讯录的内容，ABAddressBookHasUnsavedChanges 函数可以返回 YES 告知用户修改成功。

ABAddressBookCreate 函数返回的数据库实例，在使用完后必须将它释放掉，可以使用 CFRelease Core Foundation 方法。如示例代码里那样。

当决定是否更改通讯录数据库后，你可以分别使用 AbAddressBookSave 或 ABAddressBookRevert 方式来保存或放弃更改。

10.2. 在通讯录取得所有的联系人

10.2.1. 问题

想要在用户通讯录里取得所有的联系方式。

10.2.2. 方案

使用 ABAddressBookCopyOfAllPeople 函数取得所有的联系数据：

```
- (BOOL) application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
ABAddressBookRef addressBook = ABAddressBookCreate();
if (addressBook != nil){
NSLog(@"Successfully accessed the address book.");
NSArray *arrayOfAllPeople = (__bridge_transfer NSArray *)
ABAddressBookCopyArrayOfAllPeople(addressBook);
NSUInteger peopleCounter = 0;
for (peopleCounter = 0;
peopleCounter < [arrayOfAllPeople count];
peopleCounter++){
ABRecordRef thisPerson =
(__bridge ABRecordRef)[arrayOfAllPeople objectAtIndex:peopleCounter];
NSLog(@"%@", thisPerson);
/* Use the [thisPerson] address book record */
}
CFRelease(addressBook);
}
self.window = [[UIWindow alloc] initWithFrame:
[[UIScreen mainScreen] bounds]];
self.window.backgroundColor = [UIColor whiteColor];
[self.window makeKeyAndVisible];
return YES;
}
```

10.2.3. 讨论

在访问用户通讯录数据库后，我们可以调用 ABAddressBookCopyArrayOfAllPeople 函数

取得通讯录里的所有联系人数据。这个函数返回的是 CFArrayRef 类型的数组值，但是这里有几个常使用的 CFArray 类型的数组方法：

VFArrayGetCount：

调用这个方法来获取 CFArrayRef 的一个实例的项目数目，这和取得一个 NSArray 的实例方法的数目相似。

CFArrayGetValueAtIndex：

调用这个方法来取得在 CFArrayRef 的一个实例的详细位置的条目，这和一个 NSArray 的 ObjiectAtIndex:实例方法类似。

CFArrayRef Core Foundation 对象是一个支持 Tool-Free Binary 相对应的 NSArray。就是说我们可以简地结合 Core Foundation 数组并将它实例化成一个 NSArray 的实例。ARC 使用_bridge_transfer 关键字（它会减少在 Core Foundation 对象的参考数目，自从本地变量默认是个强变量，不需要做额外的工作就能保留这些内容。

这些联系人以数组形式输入的项目（可以调用 ABAddressBookCopyArrayOfAllPeople 函数来取得这些项目）是 ABRecordRef 类型的。在 10.3 章节，你会看到怎样去访问词条的不同属性，比如在通讯录数据库里的个人的词条。

10.2.4. 参考

10.1 章节

10.3. 取得通讯录联系人的属性

10.3.1. 问题

你已经在通讯录里取得了一个项目的一个参考，例如个人词条，并且你想取得个人的属性，比如姓和名。

10.3.2. 方案

在个人通讯录记录中使用 ABRecordCopyValue 方法

10.3.3. 讨论

在通讯录数据库里的记录是 ABRecordRef 类型的。每条记录可以是一个群或一个人。我们先不考虑群，所以把重点放到个人上。每个人都有各种的数据类型，比如他的姓和名，电子邮箱，等等。但是记住：许多这样的值都是可选填的，再给通讯录添加新联系人时，可以省去比如电话号码，名字，电子邮件，个人主页等信息。

ABRecordCopyValue 接受一条通讯录的记录和属性，并且将它看做是它的第二个参数。这第二个蚕食是我们想要取得的内容。下面是一些常用的属性（所有的属性都有是在 ABPerson.h 头文件里定义的常量值）。

KABPwesonFirstNameProperty

这个方法会取得名字的第一个字。返回的是 CFStringRef 类型的值，它可以很容易的被转化成 NSString 类型的值。

KABPersonLastNameProperty

这个值将取得姓名的最后一个字。想第一个字的属性那样，返回的是 CFString 类型的值，他可以使用 bridge cast 装换成 NSString 类型的值。

KABPersonMiddleNameProperty

这个值会返回名字的中间字。想第一个字那样，返回的是 CFString 类型的值，它可以使用 bridge cast 转成 NSString 类型。

KABPersonEmailProperty

这个值将取得个人的电子邮箱地址。这种情况下返回的是 ABMultiValueRef 类型的值。这种数据类型包含多个值在里面，但是不是真正的数组形式。这种数据类型将会在以后讨论。

我们从 ABRecordCopyValue 函数回复的值是明确的，属性类型，比如 CFStringRef。这个函数能返回复杂的值，比如联系电子邮件地址。这电子邮件又分为家庭邮件，工作邮件，等等。这些值在通讯录框架里被称为 multivalues。很多的函数函数都允许我们使用多个值（它们是属于 ABMultiValueRef）：

ABMultiValueGetCount

它将返回 multivalue 里的数值对。

ABMultiValueCopyLabelAtIndex

它将返回在一个确定的有序的关于多值的标签。例如，假如用户有三个邮箱，如工作，家庭，实验邮箱，工作用的邮箱应该是排在第一位的。这个函数将取得关于这个邮箱的标签（在这个例子里的工作邮箱）。请记住多值对不需要具有标签。确保你已检查了 NULL 值。

ABMultiValueCopyValueAtIndex

它将返回在一个有序的关于多值项的字符串值。我知道这很难理解，但是想象一下，用户有工作，家庭，实验邮箱。假如我们提供 0 给这个函数，它将会取得用户的联系工作邮箱。

现在我们先写一个简单的程序，这个程序可以取得通讯录里的所有联系方式并打印出这些联系人的名字，邮箱地址等对象。

```
- (BOOL) application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
ABAddressBookRef addressBook = ABAddressBookCreate();
if (addressBook != nil){
NSLog(@"Successfully accessed the address book.");
NSArray *allPeople = (__bridge_transfer NSArray *)
ABAddressBookCopyArrayOfAllPeople(addressBook);
NSUInteger peopleCounter = 0;
for (peopleCounter = 0;
peopleCounter < [allPeople count];
peopleCounter++){
ABRecordRef thisPerson = (__bridge ABRecordRef)
[allPeople objectAtIndex:peopleCounter];
NSString *firstName = (__bridge_transfer NSString *)
ABRecordCopyValue(thisPerson, kABPersonFirstNameProperty);
NSString *lastName = (__bridge_transfer NSString *)
ABRecordCopyValue(thisPerson, kABPersonLastNameProperty);
NSString *email = (__bridge_transfer NSString *)
ABRecordCopyValue(thisPerson, kABPersonEmailProperty);
```

```
NSLog(@"First Name = %@", firstName);
NSLog(@"Last Name = %@", lastName);
NSLog(@"Address = %@", email);
}
CFRelease(addressBook);
}
self.window = [[UIWindow alloc] initWithFrame:
[[UIScreen mainScreen] bounds]];
self.window.backgroundColor = [UIColor whiteColor];
[self.window makeKeyAndVisible];
return YES;
}
```

假如现在运行这个程序，里面有 3 个我已经添加到了通讯录里的联系人，我将会得到下面的控制台信息：

```
Successfully accessed the address book.
First Name = Vandad
Last Name = Nahavandipoor
Address = ABMultiValueRef 0x684ad50 with 2 value(s)
0: _$!<Home>!$_ (0x684af00) - vandad.np@gmail.com (0x684af20)
1: _$!<Work>!$_ (0x684aee0) - iosandosx@gmail.com (0x684af90)
First Name = Brian
Last Name = Jepson
Address = ABMultiValueRef 0x684ac00 with 1 value(s)
0: _$!<Home>!$_ (0x684adf0) - brian@oreilly.com (0x684ae10)
First Name = Andy
Last Name = Oram
Address = ABMultiValueRef 0x684a710 with 1 value(s)
0: _ $!<Home>!$_ (0x684ace0) - andy@oreilly.com (0x684ad00)
```

很明显，多值区，邮箱不能作为字符串对象来读取。所以我们将使用我刚刚刚学到的函数执行一个可以接受 ABRecordRef 类型的对象并能正确读取多值区记录并将它打印到控制台上的方法：

```
- (void) logPersonEmails:(ABRecordRef)paramPerson{
if (paramPerson == NULL){
NSLog(@"The given person is NULL.");
return;
}
ABMultiValueRef emails =
ABRecordCopyValue(paramPerson, kABPersonEmailProperty);
if (emails == NULL){
NSLog(@"This contact does not have any emails.");
return;
}
/* Go through all the emails */
NSUInteger emailCounter = 0;
for (emailCounter = 0;
emailCounter < ABMultiValueGetCount(emails);
emailCounter++){
/* Get the label of the email (if any) */
NSString *emailLabel = (__bridge_transfer NSString *)
ABMultiValueCopyLabelAtIndex(emails, emailCounter);
NSString *localizedEmailLabel = (__bridge_transfer NSString *)
ABAddressBookCopyLocalizedLabel((__bridge CFStringRef)emailLabel);
```

```
/* And then get the email address itself */
NSString *email = (__bridge_transfer NSString *)
ABNSLog(@"Label = %@, Localized Label = %@, Email = %@",
emailLabel,
localizedEmailLabel,
email);
}
CFRelease(emails);
}
- (BOOL) application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions{MultiValueCopyValueAtIndex(emails, emailCounter);
ABAddressBookRef addressBook = ABAddressBookCreate();
if (addressBook != nil){
NSLog(@"Successfully accessed the address book.");
NSArray *allPeople = (__bridge_transfer NSArray *)
ABAddressBookCopyArrayOfAllPeople(addressBook);
NSUInteger peopleCounter = 0;
for (peopleCounter = 0;
peopleCounter < [allPeople count];
peopleCounter++){
ABRecordRef thisPerson =
(__bridge ABRecordRef)[allPeople objectAtIndex:peopleCounter];
NSString *firstName = (__bridge_transfer NSString *)
ABRecordCopyValue(thisPerson, kABPersonFirstNameProperty);
NSString *lastName = (__bridge_transfer NSString *)
ABRecordCopyValue(thisPerson, kABPersonLastNameProperty);
NSLog(@"First Name = %@", firstName);
NSLog(@"Last Name = %@", lastName);
[self logPersonEmails:thisPerson];
}
CFRelease(addressBook);
}
self.window = [[UIWindow alloc] initWithFrame:
[[UIScreen mainScreen] bounds]];
self.window.backgroundColor = [UIColor whiteColor];
[self.window makeKeyAndVisible];
return YES;
}
```

在 NULL 值里调用 CFRelease 程序将会破坏你的程序。确保在调用 Core Foundation 程序之前检查 NULL 值。

函数 ABMutilValueCopyLabelAtIndex 返回的标签值是隐藏类型的并且很难读取。比如“_$!<Other>!$_”和“_$!<Home>!$_”，它们可以被设置成其他或家庭的电子邮件，你可以使用 ABMultiValueCopyLabelAtIndex 函数先复制这个标签并将这个函数返回的值粘贴到 ABAddressBookCopyLocalizedlabel 函数。

10.3.4. 参考

第 10.1 章节和 10.2 章节

10.4. 在通讯录里插入联系人信息

10.4.1. 问题

你想要新建一个联系人并将它插入到用户通讯录。

10.4.2. 方案

使用 ABPersonCreate 函数创建一个新的联系人。使用 ABRecordSetValue 函数设置联系人的属性并使用 ABAddressBookRecord 函数将联系人添加到通讯录里。

10.4.3. 讨论

在使用 ABAddressBookCreate 函数访问通讯录数据库后。你可以开始插入新群组和联系人记录到数据库了。在这章节里，我们会重点放在将个人记录添加到通讯录里。关于插入新的群组到通讯录里信息，请参考 10.5 章节。

创建个人记录我们必须使用到 ABPersonCreate 函数。但是只是用这个函数是不能将个人记录添加到通讯录里的。你必须在通讯录里保存你的记录到数据库里。

通过使用 ABPersonCreate 函数，你会取得一个关于 ABRecordRef 类型的 Core Foundation。现在你可以调用 ABRecordSetValue 函数来设置新个人信息的变化属性。一旦你完成这些工作，你必须添加一个新成员记录到这个数据库里。你可以使用 ABAddressBookAddRecord 函数来完成。完成之后，你必须保存任何改变后的信息以保证新的个人信息记录被真正地保存起来了。可以使用 ABAddressBookSave 函数来保存。

让我们结合这些内容去执行允许我们去插入一个新成员信息记录到通讯录里的方法：

```
- (ABRecordRef) newPersonWithFirstName:(NSString *)paramFirstName
lastName:(NSString *)paramLastName
inAddressBook:(ABAddressBookRef)paramAddressBook{
ABRecordRef result = NULL;
if (paramAddressBook == NULL){
NSLog(@"The address book is NULL.");
return NULL;
}
if ([paramFirstName length] == 0 &&
[paramLastName length] == 0){
NSLog(@"First name and last name are both empty.");
return NULL;
}
result = ABPersonCreate();
if (result == NULL){
NSLog(@"Failed to create a new person.");
return NULL;
}
BOOL couldSetFirstName = NO;
BOOL couldSetLastName = NO;
CFErrorRef setFirstNameError = NULL;
CFErrorRef setLastNameError = NULL;
couldSetFirstName = ABRecordSetValue(result,
```

```
kABPersonFirstNameProperty,
(__bridge CFTypeRef)paramFirstName,
&setFirstNameError);
couldSetLastName = ABRecordSetValue(result,
kABPersonLastNameProperty,
(__bridge CFTypeRef)paramLastName,
&setLastNameError);
CFErrorRef couldAddPersonError = NULL;
BOOL couldAddPerson = ABAddressBookAddRecord(paramAddressBook,
result,
&couldAddPersonError);
if (couldAddPerson){
NSLog(@"Successfully added the person.");
} else {
NSLog(@"Failed to add the person.");
CFRelease(result);
result = NULL;
return result;
}
if (ABAddressBookHasUnsavedChanges(paramAddressBook)){
CFErrorRef couldSaveAddressBookError = NULL;
BOOL couldSaveAddressBook = ABAddressBookSave(paramAddressBook,
&couldSaveAddressBookError);
if (couldSaveAddressBook){
NSLog(@"Successfully saved the address book.");
} else {
NSLog(@"Failed to save the address book.");
}
}
if (couldSetFirstName &&
couldSetLastName){
NSLog(@"Successfully set the first name and the last name of the person.");
} else {
NSLog(@"Failed to set the first name and/or last name of the person.");
}
return result;
}
```

And now in our app delegate, let's invoke this method and pass it an address book object:

```
- (BOOL) application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
ABAddressBookRef addressBook = ABAddressBookCreate();
if (addressBook != NULL){
ABRecordRef anthonyRobbins = [self newPersonWithFirstName:@"Anthony"
lastName:@"Robbins"
inAddressBook:addressBook];
if (anthonyRobbins != NULL){
NSLog(@"Anthony Robbins' record is inserted into the Address Book.");
CFRelease(anthonyRobbins);
}
CFRelease(addressBook);
}
self.window = [[UIWindow alloc] initWithFrame:
[[UIScreen mainScreen] bounds]];
// Override point for customization after application launch.
self.window.backgroundColor = [UIColor whiteColor];
[self.window makeKeyAndVisible];
return YES;
}
```

我们执行的 NewPersonWithFirstName：LastName：inAddressBook：这个方法在通讯录里创建了一个新的联系人。调用这个函数后，你会看到图 10-3 所示节结果。

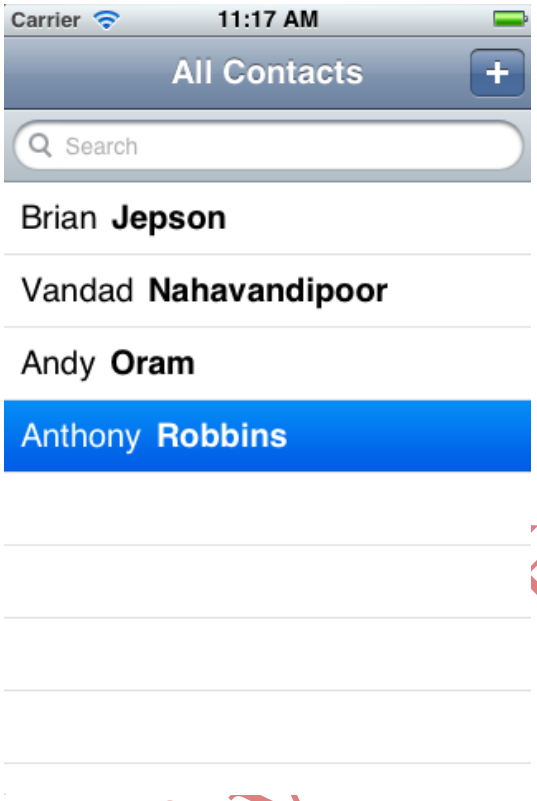

图 10-3

在 Core Foundation 上的内存管理和在 Cocoa Touch 里写程序可能有很大的不同。因为这个问题已经超出本书的内容，所以请先阅读苹果网站的“Memory Management Programming Guide for Core Foundation”这个文档。网址：http://developer.apple.com/iphone/library/documentation/corefoundation/Conceptual/CFMemoryMgmt/CFMemoryMgmt.html

10.5. 插入群组到通讯录

10.5.1. 问题

你想把你的一些联系人归为一类。

10.5.2. 方案

使用 ABGroupCreate 函数。

记住，Core Foundation 的内存管理比 Xcode 的静态解析器可以处理的任务还要复杂。因此尝试使用 LLVM 编译器编译 Core Foundation 代码来解析将会出现很多的警告。你可以忽

视这些警告并使用 Instruments 测试这些代码防止你的测试中跳出。我建议你先熟悉一下“Memory Management Programming Guide for Core Foundation”这个文档。

10.5.3. 讨论

在你回复通讯录数据库参考后，你可以调用 ABGroupCreate 函数创建一个新的群组。但是，在你插入这个群组之前还要做一些其他的工作。首先使用 ABRecorsSetValue 函数的 KABGroupNameProperty 属性给群组取名，如代码所示。

然后使用 ABAddressBookAddRecord 函数给通讯录添加数据库，更多的信息请参考 10.4 章节。

在通讯录数据库里插入已存在的群组名时将会创建一个空白的群组。在接下来的章节将会学习到如何避免插入同名群组。

在添加群组到通讯录后，还要使用 ABAddressBookSave 函数给群组里的内容保存起来。

现在就执行一允许我们在通讯录里创建一个任何名字的群组的方法：

```
- (ABRecordRef) newGroupWithName:(NSString *)paramGroupName
inAddressBook:(ABAddressBookRef)paramAddressBook{
            ABRecordRef result = NULL;
if (paramAddressBook == NULL){
NSLog(@"The address book is nil.");
return NULL;
}
result = ABGroupCreate();
if (result == NULL){
NSLog(@"Failed to create a new group.");
return NULL;
}
BOOL couldSetGroupName = NO;
CFErrorRef error = NULL;
couldSetGroupName = ABRecordSetValue(result,
kABGroupNameProperty,
(__bridge CFTypeRef)paramGroupName,
&error);
if (couldSetGroupName){
BOOL couldAddRecord = NO;
CFErrorRef couldAddRecordError = NULL;
couldAddRecord = ABAddressBookAddRecord(paramAddressBook,
result,
&couldAddRecordError);
if (couldAddRecord){
NSLog(@"Successfully added the new group.");
if (ABAddressBookHasUnsavedChanges(paramAddressBook)){
BOOL couldSaveAddressBook = NO;
CFErrorRef couldSaveAddressBookError = NULL;
couldSaveAddressBook = ABAddressBookSave(paramAddressBook,
```

```
&couldSaveAddressBookError);
if (couldSaveAddressBook){
NSLog(@"Successfully saved the address book.");
} else {
CFRelease(result);
result = NULL;
NSLog(@"Failed to save the address book.");
}
} else {
CFRelease(result);
result = NULL;
NSLog(@"No unsaved changes.");
}
} else {
CFRelease(result);
result = NULL;
NSLog(@"Could not add a new group.");
}
} else {
CFRelease(result);
result = NULL;
NSLog(@"Failed to set the name of the group.");
}
return result;
}
```

现在就调用这个方法：

```
- (BOOL) application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
ABAddressBookRef addressBook = ABAddressBookCreate();
if (addressBook != nil){
NSLog(@"Successfully accessed the address book.");
ABRecordRef personalCoachesGroup =
[self newGroupWithName:@"Personal Coaches"
inAddressBook:addressBook];
if (personalCoachesGroup != NULL){
NSLog(@"Successfully created the group.");
CFRelease(personalCoachesGroup);
} else {
NSLog(@"Could not create the group.");
}
CFRelease(addressBook);
}
self.window = [[UIWindow alloc] initWithFrame:
[[UIScreen mainScreen] bounds]];
self.window.backgroundColor = [UIColor whiteColor];
[self.window makeKeyAndVisible];
return YES;
}
```

运行这段代码后，将会得到图 10-4 所示的结果。

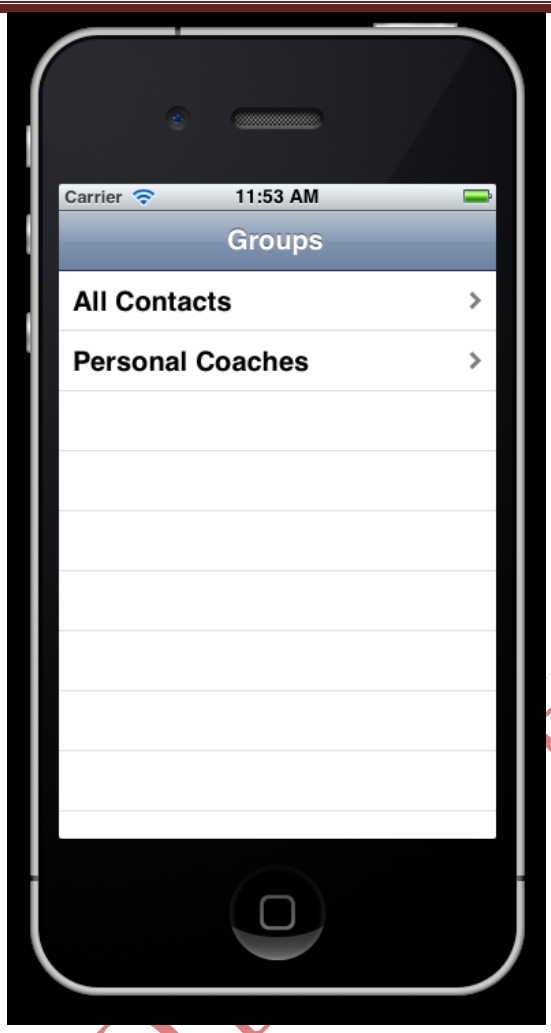

图 10-4 在通讯录数据库中创建一个新的群组

10.6. 给群组里添加联系人

10.6.1. 问题

想要给通讯录里的群组里添加联系人。

10.6.2. 方案

使用 ABGroupAddMenber 函数。

10.6.3. 讨论

学会了给通讯录里数据库里插入个人联系方式和群组联系方式。在前面的章节里我们执行了 newPersonWithFirstName:lastName:inAddressBook: 和
newGroupWithName:inAddressBook:这两个方法。现在要将我们存入数据库里的联系人添

加到群组里。结合这三小节的内容，可以使用如下代码来完成：

```
- (BOOL) addPerson:(ABRecordRef)paramPerson
toGroup:(ABRecordRef)paramGroup
saveToAddressBook:(ABAddressBookRef)paramAddressBook{
BOOL result = NO;
if (paramPerson == NULL ||
paramGroup == NULL ||
paramAddressBook == NULL){
NSLog(@"Invalid parameters are given.");
return NO;
}
CFErrorRef error = NULL;
/* Now attempt to add the person entry to the group */
result = ABGroupAddMember(paramGroup,
paramPerson,
&error);
if (result == NO){
NSLog(@"Could not add the person to the group.");
return result;
}
/* Make sure we save any unsaved changes */
if (ABAddressBookHasUnsavedChanges(paramAddressBook)){
BOOL couldSaveAddressBook = NO;
CFErrorRef couldSaveAddressBookError = NULL;
couldSaveAddressBook = ABAddressBookSave(paramAddressBook,
&couldSaveAddressBookError);
if (couldSaveAddressBook){
NSLog(@"Successfully added the person to the group.");
result = YES;
} else {
NSLog(@"Failed to save the address book.");
}
} else {
NSLog(@"No changes were saved.");
}
return result;
}
- (BOOL) application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
ABAddressBookRef addressBook = ABAddressBookCreate();
if (addressBook != nil){
ABRecordRef richardBranson = [self newPersonWithFirstName:@"Richard"
lastName:@"Branson"
inAddressBook:addressBook];
if (richardBranson != NULL){
ABRecordRef entrepreneursGroup = [self newGroupWithName:@"Entrepreneurs"
inAddressBook:addressBook];
if (entrepreneursGroup != NULL){
if ([self addPerson:richardBranson
toGroup:entrepreneursGroup
saveToAddressBook:addressBook]){
NSLog(@"Successfully added Richard Branson \
to the Entrepreneurs Group");
} else {
NSLog(@"Failed to add Richard Branson to the Entrepreneurs group.");
}
CFRelease(entrepreneursGroup);
```

```
} else {
NSLog(@"Failed to create the Entrepreneurs group.");
}
CFRelease(richardBranson);
} else {
NSLog(@"Failed to create an entity for Richard Branson.");
}
CFRelease(addressBook);
} else {
NSLog(@"Address book is nil.");
}
self.window = [[UIWindow alloc] initWithFrame:
[[UIScreen mainScreen] bounds]];
self.window.backgroundColor = [UIColor whiteColor];
[self.window makeKeyAndVisible];
return YES;
}
```

现在可以在“Entrepreneurs”群组里看到我们添加的联系人了，如图 10-5 所示。

图 10-5 在群组中添加一个联系人

10.6.4. 参考

章节 10.5

10.7. 查找通讯录联系人

10.7.1. 问题

想要在通讯录里查找确定的单个联系人或者群组。

10.7.2. 方案

使用 ABAddressBookCopyArrayOfAllPeople 和 ABAddressBookCopyArrayOfAllGroups 函数来查找群部的联系人和群组。遍历返回的数组寻找你需要找的联系人。你可以使用 ABAddress

BookCopyPeopleWithName 函数通过名字寻找联系人。

10.7.3. 讨论

到目前为止，我们已经在通讯录里插入了群组和联系人，但是并不知道是否有重名的。可以使用 ABAddressBookCopyArrayOfAllPeople 和 ABAddressBookCopyArrayOfAllGroups 函数取得通讯录里所有的联系人和群组的数组并查找是否有重名的。下面是这两个函数的几个方法：

```
- (BOOL) doesPersonExistWithFirstName:(NSString *)paramFirstName
lastName:(NSString *)paramLastName
inAddressBook:(ABRecordRef)paramAddressBook{
BOOL result = NO;
if (paramAddressBook == NULL){
NSLog(@"The address book is null.");
return NO;
}
NSArray *allPeople = (__bridge_transfer NSArray *)
ABAddressBookCopyArrayOfAllPeople(paramAddressBook);
NSUInteger peopleCounter = 0;
for (peopleCounter = 0;
peopleCounter < [allPeople count];
peopleCounter++){
ABRecordRef person = (__bridge ABRecordRef)
[allPeople objectAtIndex:peopleCounter];
NSString *firstName = (__bridge_transfer NSString *)
ABRecordCopyValue(person, kABPersonFirstNameProperty);
NSString *lastName = (__bridge_transfer NSString *)
ABRecordCopyValue(person, kABPersonLastNameProperty);
BOOL firstNameIsEqual = NO;
BOOL lastNameIsEqual = NO;
if ([firstName length] == 0 &&
[paramFirstName length] == 0){
firstNameIsEqual = YES;
}
else if ([firstName isEqualToString:paramFirstName]){
firstNameIsEqual = YES;
```

```
}
if ([lastName length] == 0 &&
[paramLastName length] == 0){
lastNameIsEqual = YES;
}
else if ([lastName isEqualToString:paramLastName]){
lastNameIsEqual = YES;
}
if (firstNameIsEqual &&
lastNameIsEqual){
return YES;
}
}
return result;
}
```

同样的，我们可以使用 ABAddressBookCopyArrayOfAllGroups 函数检查通讯录里的群组是否有重名：

```
- (BOOL) doesGroupExistWithGroupName:(NSString *)paramGroupName
inAddressBook:(ABAddressBookRef)paramAddressBook{
BOOL result = NO;
if (paramAddressBook == NULL){
NSLog(@"The address book is null.");
return NO;
}
NSArray *allGroups = (__bridge_transfer NSArray *)
ABAddressBookCopyArrayOfAllGroups(paramAddressBook);
NSUInteger groupCounter = 0;
for (groupCounter = 0;
groupCounter < [allGroups count];
groupCounter++){
ABRecordRef group = (__bridge ABRecordRef)
[allGroups objectAtIndex:groupCounter];
NSString *groupName = (__bridge_transfer NSString *)
ABRecordCopyValue(group, kABGroupNameProperty);
if ([groupName length] == 0 &&
[paramGroupName length] == 0){
return YES;
}
else if ([groupName isEqualToString:paramGroupName]){
return YES;
}
}
return result;
}
```

请尽量避免使用空字符或者 nil，Null 之类的来命名群组。

我们可以如下使用 doesGroupExistWithGroupName:inAddressBook:方法：

```
- (BOOL) application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
ABAddressBookRef addressBook = ABAddressBookCreate();
if (addressBook != NULL){
if ([self doesGroupExistWithGroupName:@"O'Reilly"
inAddressBook:addressBook]){
NSLog(@"The O'Reilly group already exists in the address book.");
} else {
ABRecordRef oreillyGroup = [self newGroupWithName:@"O'Reilly"
inAddressBook:addressBook];
if (oreillyGroup != NULL){
NSLog(@"Successfully created a group for O'Reilly.");
CFRelease(oreillyGroup);
} else {
NSLog(@"Failed to create a group for O'Reilly.");
}
}
CFRelease(addressBook);
}
self.window = [[UIWindow alloc] initWithFrame:
[[UIScreen mainScreen] bounds]];
self.window.backgroundColor = [UIColor whiteColor];
[self.window makeKeyAndVisible];
return YES;
}
```

请参考 10.5 章节执行 createNewGroupWithName:inAddressBook:方法。

就像之前看到的，我们有两个方法在通讯录数据库里查找联系人：

1、使用 ABAddressBookCopyArrayOfAllPeople 函数取得通讯录里所有联系人的列表，取得列表里所有的记录并将我们要查找的字符串与每个联系人的名字相比较。这个函数帮助我们找到与我们要查找的相匹配的联系人。你可以通过联系人的任何属性来查找联系人，比如名字，邮箱，电话号码，等等。

2、请求 Address Book 框架基于综合名字来执行查找。使用 ABAddressBookCopyPeopleWithName 函数完成这项工作。

下面是一个使用 ABAddressBookCopyPeopleWithName 函数来查找一个确定的联系人的例子:

```
- (BOOL) doesPersonExistWithFullName:(NSString *)paramFullName
inAddressBook:(ABAddressBookRef)paramAddressBook{
BOOL result = NO;
if (paramAddressBook == NULL){
NSLog(@"Address book is null.");
return NO;
}
NSArray *allPeopleWithThisName = (__bridge_transfer NSArray *)
ABAddressBookCopyPeopleWithName(paramAddressBook,
(__bridge CFStringRef)paramFullName);
if ([allPeopleWithThisName count] > 0){
result = YES;
}
return result;
}
Here is how we can use this method that we just implemented:
- (BOOL) application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
```

```
ABAddressBookRef addressBook = ABAddressBookCreate();
if (addressBook != NULL){
if ([self doesPersonExistWithFullName:@"Anthony Robbins"
inAddressBook:addressBook]){
NSLog(@"Anthony Robbins exists in the address book.");
} else {
NSLog(@"Anthony Robbins does not exist in the address book.");
ABRecordRef anthonyRobbins = [self newPersonWithFirstName:@"Anthony"
lastName:@"Robbins"
inAddressBook:addressBook];
if (anthonyRobbins != NULL){
NSLog(@"Successfully created a record for Anthony Robbins");
CFRelease(anthonyRobbins);
} else {
NSLog(@"Failed to create a record for Anthony Robbins");
}
}
CFRelease(addressBook);
}
self.window = [[UIWindow alloc] initWithFrame:
[[UIScreen mainScreen] bounds]];
self.window.backgroundColor = [UIColor whiteColor];
[self.window makeKeyAndVisible];
return YES;
}
```

使用这个函数我们可以没必要知道联系人的全名就可以查找到他。我们只要输入名字的一部分就可以查找的这个联系人了。

使用 ABAddressBookCopyPeopleWithName 函数执行查找在某些情况下是不明智的。

10.8. 取得和设置联系人的通讯录图片

10.8.1. 问题

想要取得和设置通讯录联系人的图片。

10.8.2. 方案

使用以下的其中一个函数：
ABPersonHasImageData
使用这个函数来寻找通讯录里具有图片的联系人。
ABPersonCopyImageData
使用这个函数取得图片数据（假如有的话）。
ABPersonSetImageData
使用这个函数设置联系人的图片数据。

10.8.3. 讨论

如之前提到的，我们可以使用 ABPersonCopyImageData 函数取得与通讯录有关图片的一些数据。我们可以在一个比较方便的方法来使用这个函数：

```
- (UIImage *) getPersonImage:(ABRecordRef)paramPerson{
UIImage *result = nil;
if (paramPerson == NULL){
NSLog(@"The person is nil.");
return NULL;
}
NSData *imageData = (__bridge_transfer NSData *)
ABPersonCopyImageData(paramPerson);
if (imageData != nil){
UIImage *image = [UIImage imageWithData:imageData];
result = image;
}
return result;
}
```

函数 ABPersonSetImageData 是在通讯录里设置联系人图片的。这个函数使用的是数据而不是图片本身，所以我们要从 UIImage 取得 NSData。假如要将这些数据转换成 PNG 图片，可以使用 UIImagePNGRepresentation 函数取得 UIImage 类型的 PNG 格式的图片。可以使用 UIImageJPEGRepresentation 函数从 UIImage 的实例取得 JPG 格式的图片数据。下面黑丝允许我们设置一个联系人图片的方法：

```
- (BOOL) setPersonImage:(ABRecordRef)paramPerson
inAddressBook:(ABAddressBookRef)paramAddressBook
withImageData:(NSData *)paramImageData{
BOOL result = NO;
if (paramAddressBook == NULL){
NSLog(@"The address book is nil.");
return NO;
}
if (paramPerson == NULL){
NSLog(@"The person is nil.");
return NO;
}
CFErrorRef couldSetPersonImageError = NULL;
BOOL couldSetPersonImage =
ABPersonSetImageData(paramPerson,
(__bridge CFDataRef)paramImageData,
&couldSetPersonImageError);
if (couldSetPersonImage){
NSLog(@"Successfully set the person's image. Saving...");
if (ABAddressBookHasUnsavedChanges(paramAddressBook)){
BOOL couldSaveAddressBook = NO;
CFErrorRef couldSaveAddressBookError = NULL;
couldSaveAddressBook = ABAddressBookSave(paramAddressBook,
&couldSaveAddressBookError);
if (couldSaveAddressBook){
NSLog(@"Successfully saved the address book.");
result = YES;
} else {
NSLog(@"Failed to save the address book.");
```

```
}
} else {
NSLog(@"There are no changes to be saved!");
}
} else {
NSLog(@"Failed to set the person's image.");
}
return result;
}
```

现在可以使用这些方法写一个简单的程序。在这个例子代码里，我们想要完成以下功能：

1、创建一个含有两个标签和两个图片的控制器。

2、尝试从通讯录里取得名为 Anthony Robbins 的联系人，假如这个联系人不存在，我们可以创建它。

3、取得这联系人的当前图片并将它显示在第一个图片视图里。

4、给这个联系人设置一个新的图片并将它显示在第二个图片视图里。

现在开始吧，下面是视图控制器的.h 文件：

```
#import <UIKit/UIKit.h>
#import <AddressBook/AddressBook.h>
@interface Retrieving_and_Setting_a_Person_s_Address_Book_ImageViewController
: UIViewController
@property (nonatomic, strong) UILabel *labelOldImage;
@property (nonatomic, strong) UIImageView *imageViewOld;
@property (nonatomic, strong) UILabel *labelNewImage;
@property (nonatomic, strong) UIImageView *imageViewNew;
@end
```

继续吧，现在结合下面的属性：

```
#import "Retrieving_and_Setting_a_Person_s_Address_Book_ImageViewController.h"
@implementation
Retrieving_and_Setting_a_Person_s_Address_Book_ImageViewController
@synthesize labelOldImage;
@synthesize imageViewOld;
@synthesize labelNewImage;
@synthesize imageViewNew;
...
```

接下来是我们控制器的 viewDidLoad 方法，我们会实例化我们的标签和图片视图并将它放置到我们的视图控制器里。

```
- (void) changeYPositionOfView:(UIView *)paramView
to:(CGFloat)paramY{
CGRect viewFrame = paramView.frame;
viewFrame.origin.y = paramY;
paramView.frame = viewFrame;
}
- (void) createLabelAndImageViewForOldImage{
self.labelOldImage = [[UILabel alloc] initWithFrame:CGRectZero];
self.labelOldImage.text = @"Old Image";
self.labelOldImage.font = [UIFont systemFontOfSize:16.0f];
[self.labelOldImage sizeToFit];
self.labelOldImage.center = self.view.center;
```

```
[self.view addSubview:self.labelOldImage];
[self changeYPositionOfView:self.labelOldImage
to:80.0f];
self.imageViewOld = [[UIImageView alloc] initWithFrame:CGRectMake(0.0f,
0.0f,
100.0f,
100.0f)];
self.imageViewOld.center = self.view.center;
self.imageViewOld.contentMode = UIViewContentModeScaleAspectFit;
[self.view addSubview:self.imageViewOld];
[self changeYPositionOfView:self.imageViewOld
to:105.0f];
}
- (void) createLabelAndImageViewForNewImage{
self.labelNewImage = [[UILabel alloc] initWithFrame:CGRectZero];
self.labelNewImage.text = @"New Image";
self.labelNewImage.font = [UIFont systemFontOfSize:16.0f];
[self.labelNewImage sizeToFit];
self.labelNewImage.center = self.view.center;
[self.view addSubview:self.labelNewImage];
[self changeYPositionOfView:self.labelNewImage
to:210.0f];
self.imageViewNew = [[UIImageView alloc] initWithFrame:CGRectMake(0.0f,
0.0f,
100.0f,
100.0f)];
self.imageViewNew.center = self.view.center;
self.imageViewNew.contentMode = UIViewContentModeScaleAspectFit;
[self.view addSubview:self.imageViewNew];
[self changeYPositionOfView:self.imageViewNew
to:235.0f];
}
- (void)viewDidLoad{
[super viewDidLoad];
self.view.backgroundColor = [UIColor whiteColor];
[self createLabelAndImageViewForOldImage];
[self createLabelAndImageViewForNewImage];
}
- (void)viewDidUnload{
[super viewDidUnload];
self.labelOldImage = nil;
self.imageViewOld = nil;
self.labelNewImage = nil;
self.imageViewNew = nil;
}
```

现在我们需要修改一下 viewDidLoad 一些功能，我们能从通讯录里读取联系人的图片然后设置图片并显示新的图片，可以使用在这张杰学到的函数或者其他章节学到的函数：

```
- (ABRecordRef) getPersonWithFirstName:(NSString *)paramFirstName
lastName:(NSString *)paramLastName
inAddressBook:(ABRecordRef)paramAddressBook{
ABRecordRef result = NULL;
if (paramAddressBook == NULL){
NSLog(@"The address book is null.");
return NULL;
}
```

```
NSArray *allPeople = (__bridge_transfer NSArray *)
ABAddressBookCopyArrayOfAllPeople(paramAddressBook);
NSUInteger peopleCounter = 0;
for (peopleCounter = 0;
peopleCounter < [allPeople count];
peopleCounter++){
ABRecordRef person = (__bridge ABRecordRef)
[allPeople objectAtIndex:peopleCounter];
NSString *firstName = (__bridge_transfer NSString *)
ABRecordCopyValue(person, kABPersonFirstNameProperty);
NSString *lastName = (__bridge_transfer NSString *)
ABRecordCopyValue(person, kABPersonLastNameProperty);
BOOL firstNameIsEqual = NO;
BOOL lastNameIsEqual = NO;
if ([firstName length] == 0 &&
[paramFirstName length] == 0){
firstNameIsEqual = YES;
}
else if ([firstName isEqualToString:paramFirstName]){
firstNameIsEqual = YES;
}
if ([lastName length] == 0 &&
[paramLastName length] == 0){
lastNameIsEqual = YES;
}
else if ([lastName isEqualToString:paramLastName]){
lastNameIsEqual = YES;
}
if (firstNameIsEqual &&
lastNameIsEqual){
return person;
}
}
return result;
}
- (void)viewDidLoad{
[super viewDidLoad];
self.view.backgroundColor = [UIColor whiteColor];
[self createLabelAndImageViewForOldImage];
[self createLabelAndImageViewForNewImage];
ABAddressBookRef addressBook = ABAddressBookCreate();
if (addressBook != NULL){
ABRecordRef anthonyRobbins = [self getPersonWithFirstName:@"Anthony"
lastName:@"Robbins"
inAddressBook:addressBook];
if (anthonyRobbins == NULL){
NSLog(@"Couldn't find record. Creating one...");
anthonyRobbins = [self newPersonWithFirstName:@"Anthony"
lastName:@"Robbins"
inAddressBook:addressBook];
if (anthonyRobbins == NULL){
NSLog(@"Failed to create a new record for this person.");
CFRelease(addressBook);
return;
}
}
self.imageViewOld.image = [self getPersonImage:anthonyRobbins];
NSString *newImageFilePath =
[[NSBundle mainBundle] pathForResource:@"Anthony Robbins"
```

```
ofType:@"jpg"];
UIImage *newImage = [[UIImage alloc]
initWithContentsOfFile:newImageFilePath];
NSData *newImageData = UIImagePNGRepresentation(newImage);
if ([self setPersonImage:anthonyRobbins
inAddressBook:addressBook
withImageData:newImageData]){
NSLog(@"Successfully set this person's new image.");
self.imageViewNew.image = [self getPersonImage:anthonyRobbins];
} else {
NSLog(@"Failed to set this person's new image.");
}
CFRelease(anthonyRobbins);
CFRelease(addressBook);
}
}
```

下面是显示的结果：

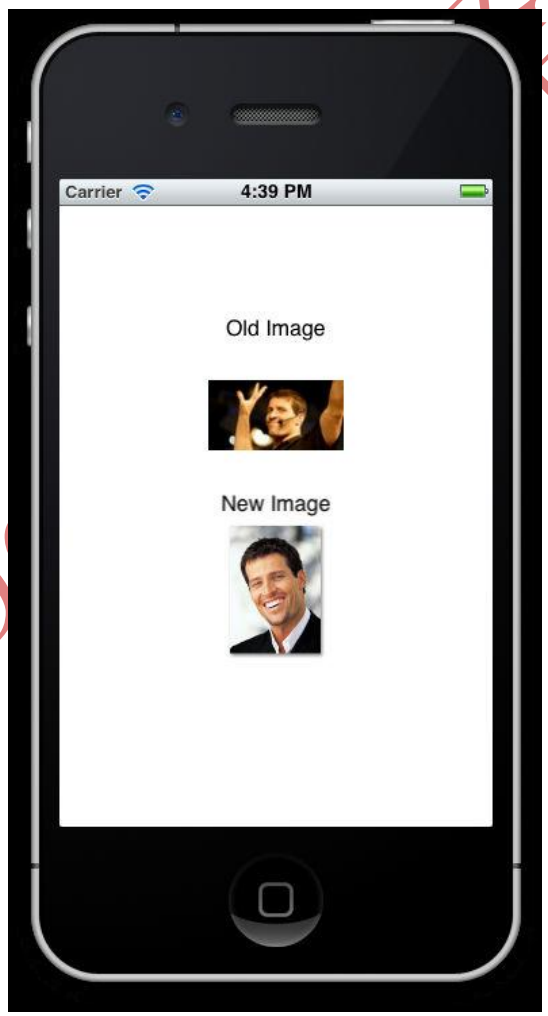

图 10-6 使用新的图片替换某条联系人中的图片

点击这里访问：DevDiv.com 移动开发论坛