CS193P - Lecture 13

iPhone Application Development

Address Book - Putting People in Your App

Announcements

- Paparazzi 3 due tomorrow at 11:59PM
- Paparazzi 4 (last assignment!) due next Wednesday

Final Project Proposals

- **Due tomorrow night!**
  - Handout on website has all the info
- If you still need an idea for a project, let us know
- We will be responding with feedback & a thumbs-up

Today's Topics

- Address Book APIs
- CoreFoundation
- Merging from an external source of people
- Using contacts in your application

Putting Contacts in Your App

The Hello World of Address Book

The Hello World of Address Book

- Create a person and set some properties
- Create ABPersonViewController
- Push the view controller onto the navigation stack

CoreFoundation

CoreFoundation vs. Foundation

CoreFoundation vs. Foundation

- CoreFoundation is a framework written in C

CoreFoundation vs. Foundation

- CoreFoundation is a framework written in C
- Many **parallels** to Foundation
  - CFDictionaryRef, CFStringRef
  - CFRetain, CFRelease

CoreFoundation vs. Foundation

- CoreFoundation is a framework written in C
- Many **parallels** to Foundation
  - CFDictionaryRef, CFStringRef
  - CFRetain, CFRelease
- AddressBook framework is also C-based
  - Uses CoreFoundation data types and semantics

CoreFoundation vs. Foundation

- CoreFoundation is a framework written in C
- Many **parallels** to Foundation
  - CFDictionaryRef, CFStringRef
  - CFRetain, CFRelease
- AddressBook framework is also C-based
  - Uses CoreFoundation data types and semantics
- Addition to memory management naming conventions
  - Functions with **Create** in their title return a retained object
  - For example, ABAddressBookCreate( );

Toll-Free Bridging

- Supported for many types of objects
  - Strings, arrays, dictionaries, dates, numbers, data streams, more
- Use an NSString* where a CFStringRef is expected & vice versa
- Very convenient for **mixing & matching** C with Objective-C

Toll-Free Bridging

- Supported for many types of objects
  - Strings, arrays, dictionaries, dates, numbers, data streams, more
- Use an NSString* where a CFStringRef is expected & vice versa
- Very convenient for **mixing & matching** C with Objective-C

```
CFArrayRef array = ABAddressBookCopyPeopleWithName(...);

NSLog(@“%d”, [(NSArray *)array count]);

NSMutableArray *mutableArray = [(NSArray *)array mutableCopy];
[mutableArray release];

if (array) {
  CFRelease(array);
}
```

CoreFoundation and NULL

CoreFoundation and NULL

- Unlike Objective-C, must NULL-check CF type objects

CoreFoundation and NULL

- Unlike Objective-C, must NULL-check CF type objects
  - (Since nil is typed id, we use NULL for CF)

CoreFoundation and NULL

- Unlike Objective-C, must NULL-check CF type objects
  - (Since nil is typed id, we use NULL for CF)

```
CFStringRef string = CreateSomeCFString...;
if (string != NULL) {
   DoSomethingWith(string);
   CFRelease(string);
}
```

CoreFoundation and NULL

- Unlike Objective-C, must NULL-check CF type objects
  - (Since nil is typed id, we use NULL for CF)

```
CFStringRef string = CreateSomeCFString...;
if (string != NULL) {
   DoSomethingWith(string);
   CFRelease(string);
}
```

  - Toll-free bridging can make this easier

CoreFoundation and NULL

- Unlike Objective-C, must NULL-check CF type objects
  - (Since nil is typed id, we use NULL for CF)

```
CFStringRef string = CreateSomeCFString...;
if (string != NULL) {
   DoSomethingWith(string);
   CFRelease(string);
}
```

  - Toll-free bridging can make this easier

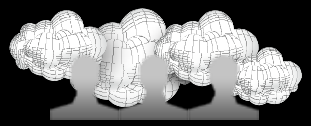
```
NSString *string = (NSString *)CreateSomeCFString...;
NSLog(@“%@”, [string lowercaseString]);
[string autorelease]; // Even use autorelease!
```

Beyond Hello World

Social Networking Website

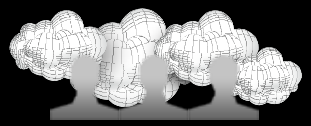
Social Networking Website

- People on the web



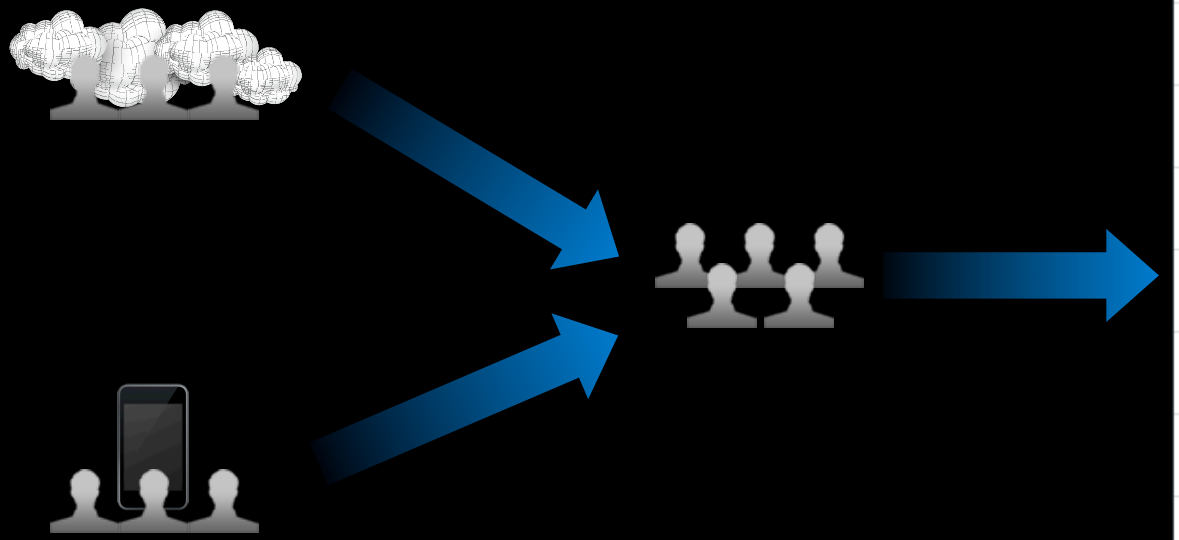
Social Networking Website

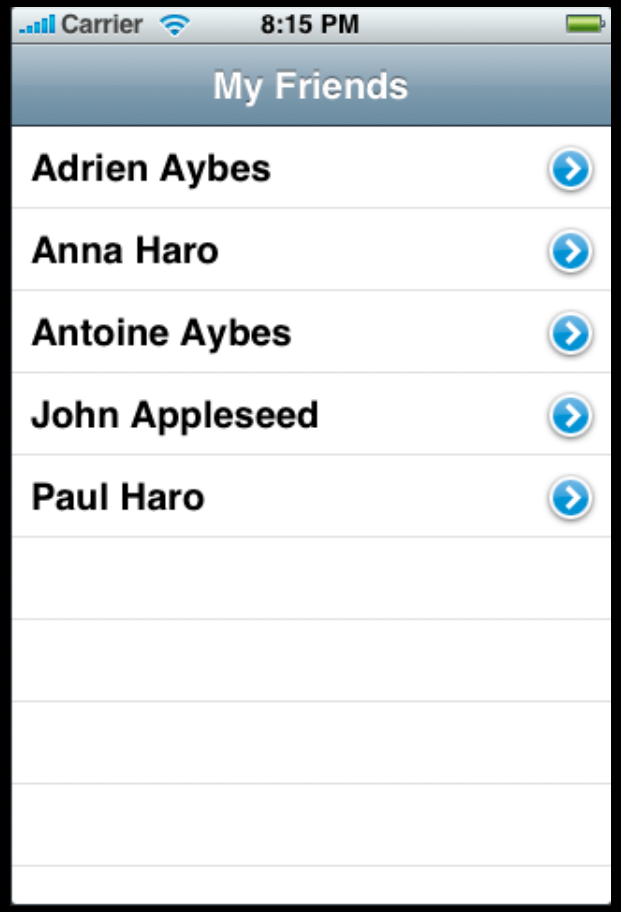
- People on the web
- People on the iPhone



Social Networking Website

- People on the web
- People on the iPhone
- Reconciling them





What Do We Need to Do?

What Do We Need to Do?

- Download

What Do We Need to Do?

- Download
- Search

What Do We Need to Do?

- Download
- Search
- Update

What Do We Need to Do?

- Download
- Search
- Update
- Display

Search

Step One

- Get the address book
- Search the people

Search

Step One

- Get the address book
- Search the people

```
ABAddressBookRef ab = ABAddressBookCreate();
CFArrayRef people = ABAddressBookCopyPeopleWithName(ab, name);
```

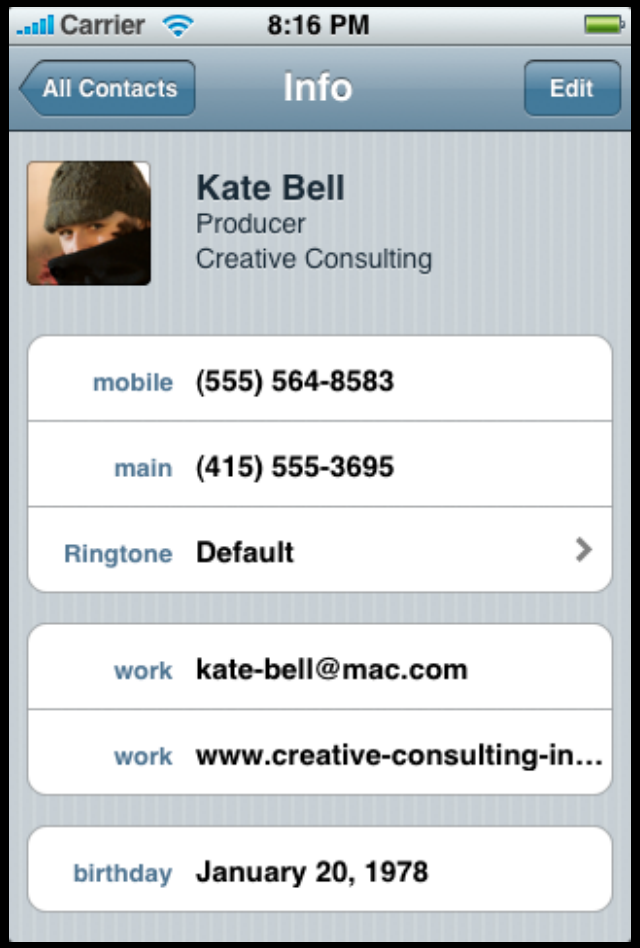
Address Book

- ABAddressBookRef
- Gives you access to the people
- Central point for all things address book
- Multiple instances, a single database

```
ABAddressBookRef ab = ABAddressBookCreate();
```

Person

- ABRecordRef
- A collection of properties
  - First and last name
  - Image
  - Phone numbers, emails, etc…



Properties

- Properties can have different types
  - String
  - Date
  - Dictionary, Data…
- Some properties may have multiple values
  - Telephone: home, work, mobile, fax…
- Person properties in ABPerson.h

Single Value Properties

- First Name, last name, birthday, etc…
- CoreFoundation types

Single Value Properties

- First Name, last name, birthday, etc…
- CoreFoundation types
- Retrieve values with ABRecordCopyValue(…)

Single Value Properties

- First Name, last name, birthday, etc…
- CoreFoundation types
- Retrieve values with ABRecordCopyValue(…)

```
CFStringRef first =
            ABRecordCopyValue(person, kABPersonFirstNameProperty);
```

Single Value Properties

- First Name, last name, birthday, etc…
- CoreFoundation types
- Retrieve values with ABRecordCopyValue(…)

```
CFStringRef first =
            ABRecordCopyValue(person, kABPersonFirstNameProperty);
```

- Set values with ABRecordSetValue(…)

Single Value Properties

- First Name, last name, birthday, etc…
- CoreFoundation types
- Retrieve values with ABRecordCopyValue(…)

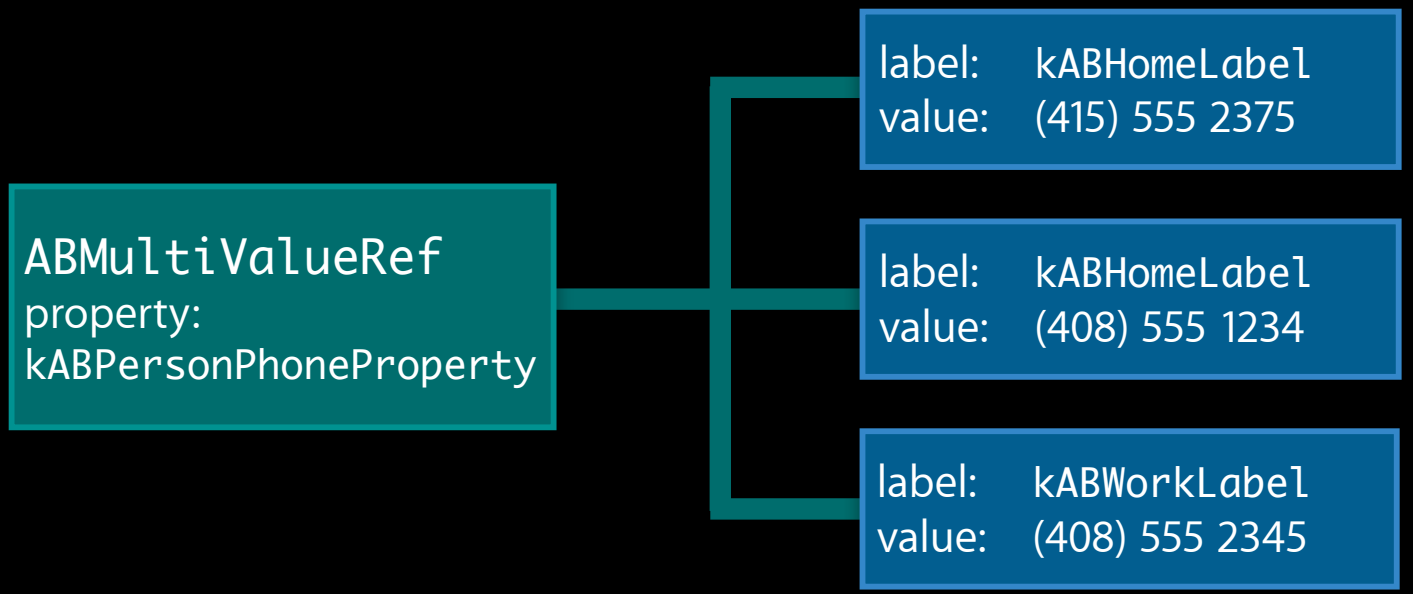
```
CFStringRef first =
            ABRecordCopyValue(person, kABPersonFirstNameProperty);
```

- Set values with ABRecordSetValue(…)

```
CFDateRef date = CFDateCreate(…)
ABRecordSetValue(person, kABPersonBirthdayProperty, date, &error);
```

Multi Value Properties

- Phones, emails, URLs, etc…
- Access just like single value properties
- ABMultiValueRef
- Container for values and labels



ABMultiValueRef

ABMultiValueRef

- Count

ABMultiValueRef

- Count

```
CFIndex count = ABMultiValueGetCount(multiValue);
```

ABMultiValueRef

- Count

```
CFIndex count = ABMultiValueGetCount(multiValue);
```

- Value

ABMultiValueRef

- Count

```
CFIndex count = ABMultiValueGetCount(multiValue);
```

- Value

```
CFTypeRef value = ABMultiValueCopyValueAtIndex(mv, index);
```

ABMultiValueRef

- Count

```
CFIndex count = ABMultiValueGetCount(multiValue);
```

- Value

```
CFTypeRef value = ABMultiValueCopyValueAtIndex(mv, index);
```

- Label

ABMultiValueRef

- Count

```
CFIndex count = ABMultiValueGetCount(multiValue);
```

- Value

```
CFTypeRef value = ABMultiValueCopyValueAtIndex(mv, index);
```

- Label

```
CFStringRef label = ABMultiValueCopyLabelAtIndex(mv, index);
```

ABMultiValueRef

- Count

```
CFIndex count = ABMultiValueGetCount(multiValue);
```

- Value

```
CFTypeRef value = ABMultiValueCopyValueAtIndex(mv, index);
```

- Label

```
CFStringRef label = ABMultiValueCopyLabelAtIndex(mv, index);
```

- Identifier

ABMultiValueRef

- Count

```
CFIndex count = ABMultiValueGetCount(multiValue);
```

- Value

```
CFTypeRef value = ABMultiValueCopyValueAtIndex(mv, index);
```

- Label

```
CFStringRef label = ABMultiValueCopyLabelAtIndex(mv, index);
```

- Identifier

```
CFIndex identifier = ABMultiValueGetIdentifierAtIndex(mv, index);
```

Update

Step Two

- Mutate the multi value
- Add the value
- Set the value on the person
- Save the Address Book

Update

Step Two

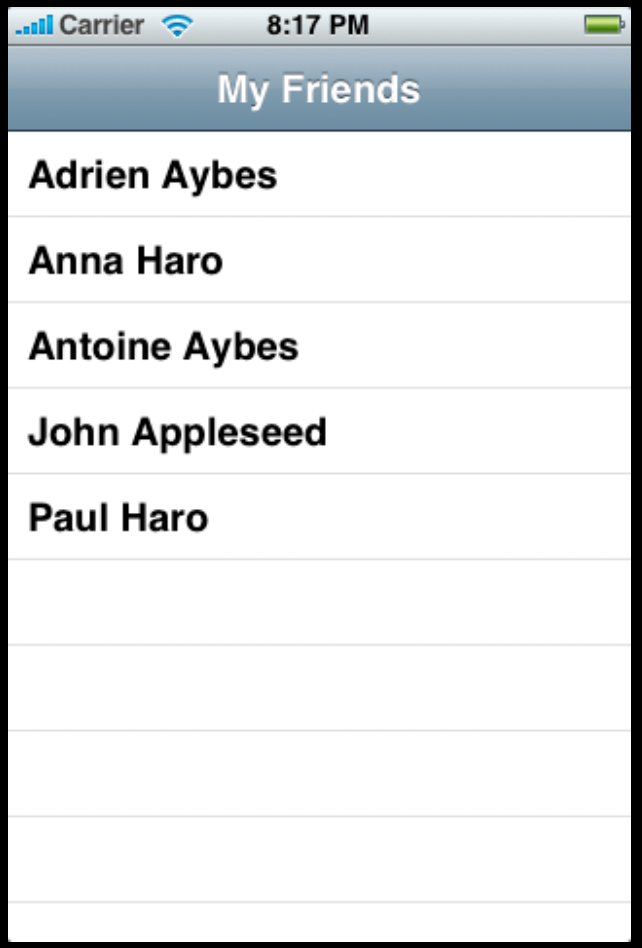
- Mutate the multi value
- Add the value
- Set the value on the person
- Save the Address Book

```
ABMultiValueRef urls = ABRecordCopyValue(person, kABPersonURLProperty);

ABMutableMultiValueRef urlCopy = ABMultiValueCreateMutableCopy(urls);
ABMultiValueAddValueAndLabel(urlCopy, "the url", "social", NULL);
ABRecordSetValue(person, urlCopy, kABPersonURLProperty);

ABAddressBookSave(ab, &err);
```

Display

Step Three

- Sort
- Get the name
- Display



Sorting

- We'll do it for you
- ABPersonGetSortOrdering
- ABPersonComparePeopleByName

Sorting

- We'll do it for you
- ABPersonGetSortOrdering
- ABPersonComparePeopleByName

```
CFMutableArrayRef people = // obtain an array of people
CFRange fullRange = CFRangeMake(0, CFArrayGetCount(people));

ABPersonSortOrdering sortOrdering = ABPersonGetSortOrdering();

CFArraySortValues(people, fullRange, ABPersonComparePeopleByName,
(void*)sortOrdering);
```

Sorting

- We'll do it for you
- ABPersonGetSortOrdering
- ABPersonComparePeopleByName

```
CFMutableArrayRef people = // obtain an array of people
CFRange fullRange = CFRangeMake(0, CFArrayGetCount(people));

ABPersonSortOrdering sortOrdering = ABPersonGetSortOrdering();

CFArraySortValues(people, fullRange, ABPersonComparePeopleByName,
(void*)sortOrdering);
```

```
// Objective-C alternative
[people sortUsingFunction:ABPersonComparePeopleByName context:
(void*)sortOrdering];
```

Getting the Name

- ABRecordCopyCompositeName

Getting the Name

- ABRecordCopyCompositeName

```
ABRecordRef person = // get a person
CFStringRef name = ABRecordCopyCompositeName(person);

// do something clever with that person's name
```

Getting the Name

- ABRecordCopyCompositeName

```
ABRecordRef person = // get a person
CFStringRef name = ABRecordCopyCompositeName(person);

// do something clever with that person's name
```

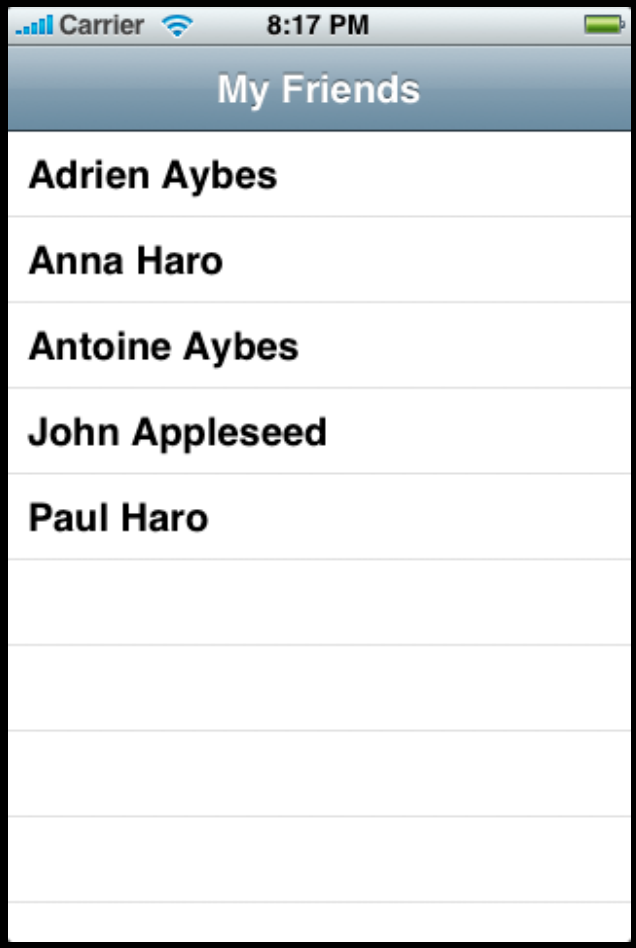
```
ABRecordRef person = // get a person
NSString *name = (NSString*)ABRecordCopyCompositeName(person);

// do something clever with that person's name
```

Demo

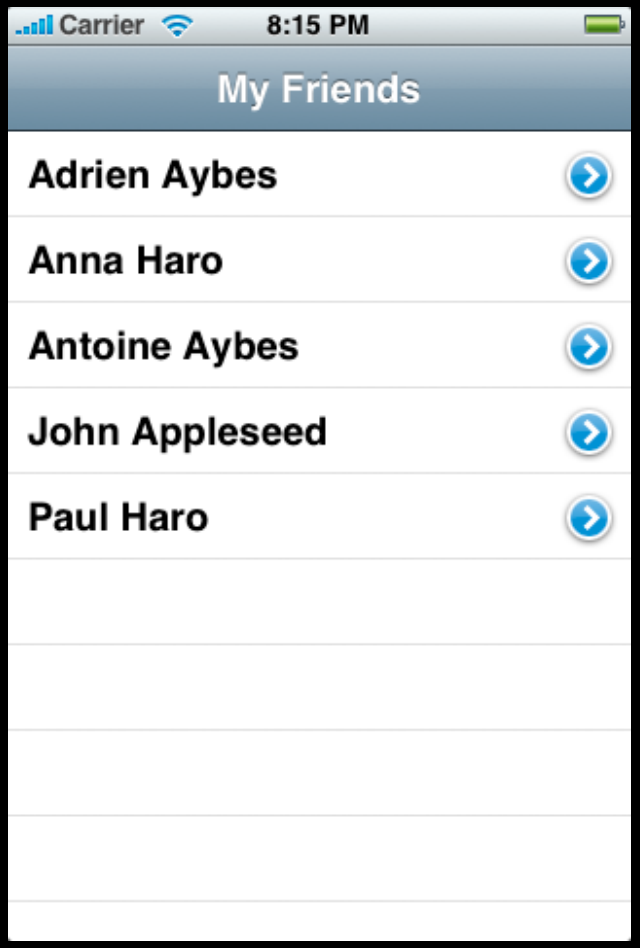
Bringing the people to the phone

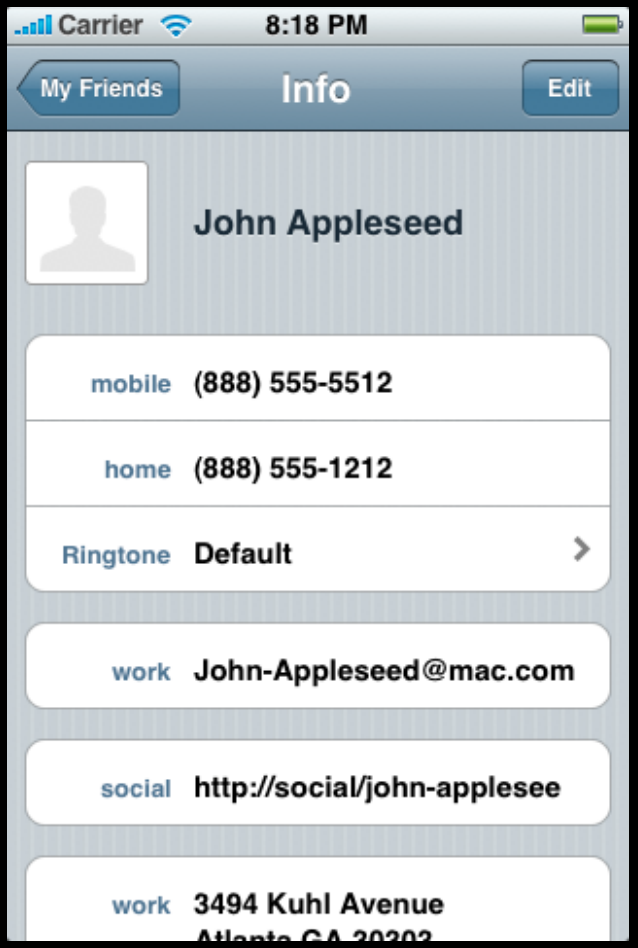
What We Just Saw

- Searching for people by name
- Using multi values
- Sorting and Displaying people



Showing Detailed Information

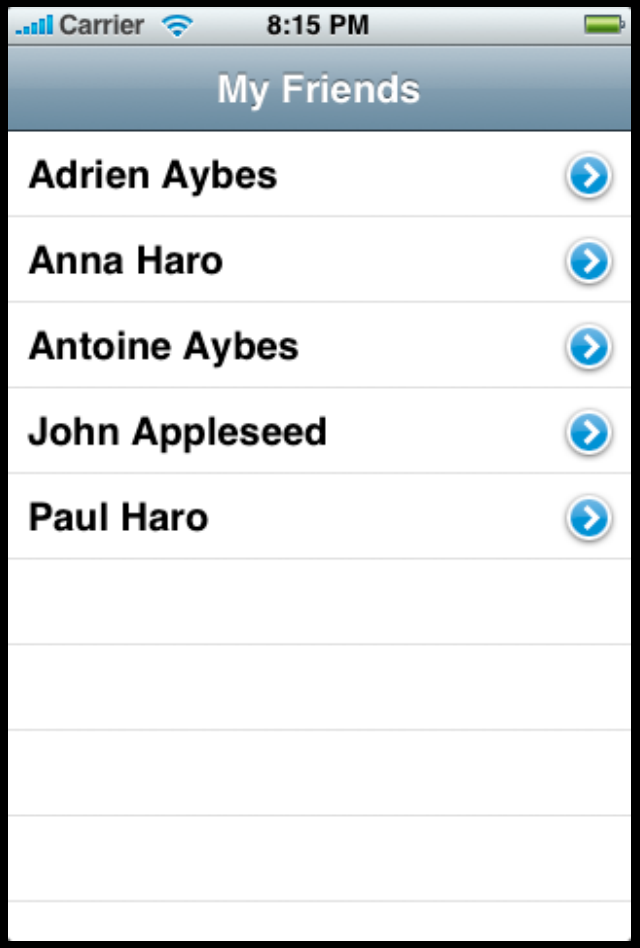


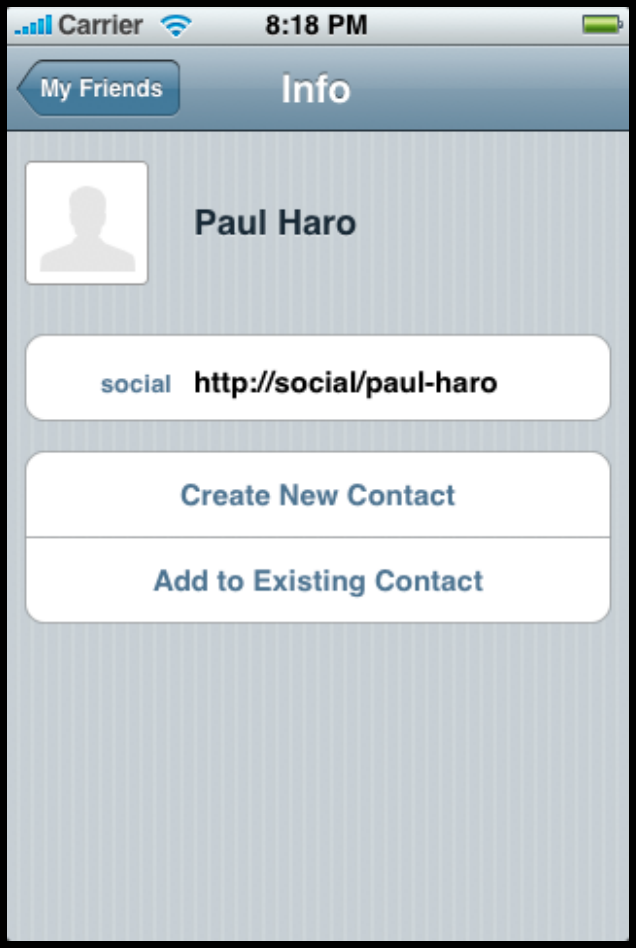


Person View Controller

- ABPersonViewController
  - displayedPerson
  - displayedProperties
  - allowsEditing

Adding Contacts to Address Book





Unknown Person View Controller

- ABUnknownPersonViewController
  - displayedPerson
  - allowsAddingToAddressBook
  - delegate

Unknown Person View Controller

- ABUnknownPersonViewController
  - displayedPerson
  - allowsAddingToAddressBook
  - delegate

```
- (void)unknownPersonViewController:(ABUnknownPersonViewController *)
unknownCardViewController didResolveToPerson:(ABRecordRef)person {

  // do something

}
```

Demo

Showing the people

What Did We Just See?

- Display contacts with ABPersonViewController
- Add to Address Book with ABUnknownPersonViewController

What If It Takes Too Long?

- Doing things in the background
  - NSThread
  - pthread

Threading Model

- Each thread needs its own ABAddressBookRef
- What you can pass between threads:
  - Values
  - ABRecordID

Using the People

Adding people to an existing application

Pick an email address and send an email

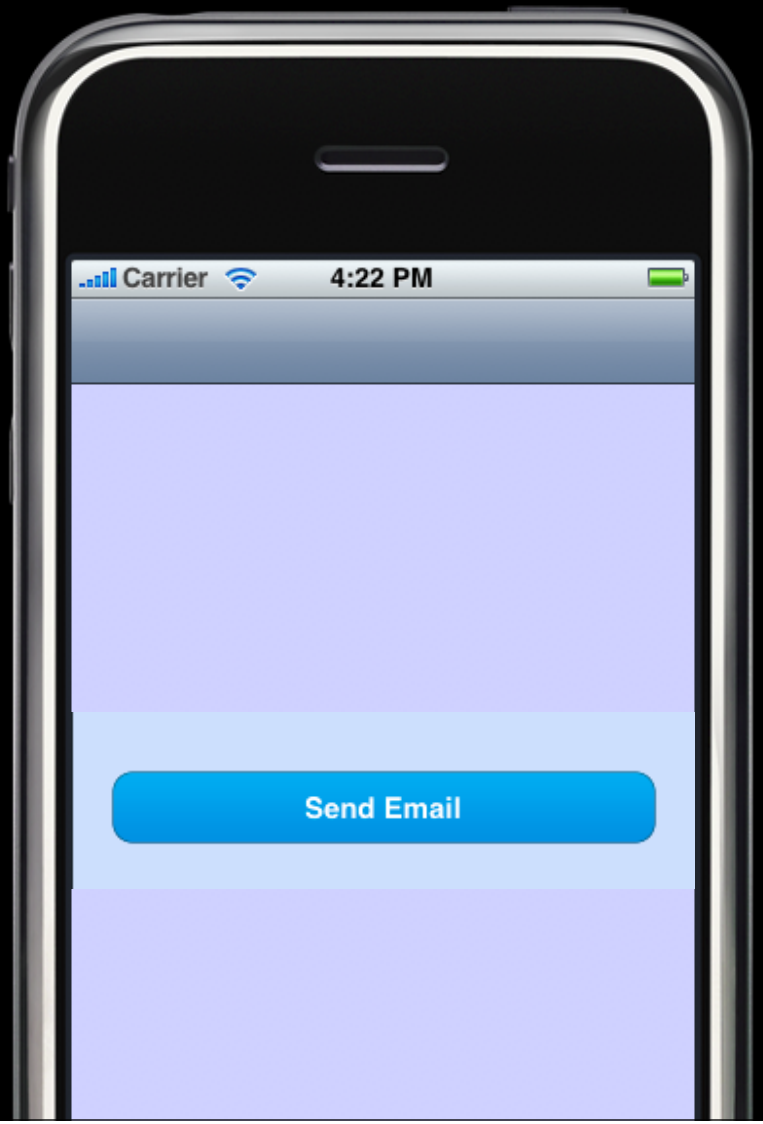
- Create an email button
- Pick a person
- Build an email URL and open it

Picking People

- ABPeoplePickerNavigationController

Picking People

- ABPeoplePickerNavigationController



Picking People

- ABPeoplePickerNavigationController

ABPeoplePickerNavigationController

- Present the Navigation Controller
- ABPeoplePickerNavigationControllerDelegate
  - Cancellation
  - Selection of a person
  - Selection of a value

Demo

Because my friends like to receive emails

What We Just Saw

- Present ABPeoplePickerNavigationController modally
- Delegate callbacks

People Storage

- Get the record identifier
- Serialize and deserialize it
- Look it up in Address Book

People Storage

- Get the record identifier
- Serialize and deserialize it
- Look it up in Address Book

```
ABRecordID personID = ABRecordGetRecordID(person);
NSNumber *personIDAsNumber = [NSNumber numberWithInt:personID];

// serialize the NSNumber
```

People Storage

- Get the record identifier
- Serialize and deserialize it
- Look it up in Address Book

```
ABRecordID personID = ABRecordGetRecordID(person);
NSNumber *personIDAsNumber = [NSNumber numberWithInt:personID];

// serialize the NSNumber
```

```
NSNumber *personIDAsNumber = // Deserialize the NSNumber
ABRecordID personID = [personIDAsNumber intValue];

ABRecordRef person = ABAddressBookGetPersonWithRecordID(ab, personID);
```

What if the Database Changed?

- Check before displaying
- Store extra information

What if the Database Changed?

- Check before displaying
- Store extra information

```
ABRecordID recordID = // get the record

ABRecordRef person = ABAddressBookGetPersonWithRecordID(ab, personID);

if (person != NULL) {
    // use the person
} else {
    // fallback to other data
}
```

Notifications

- Register callback
  - ABAddressBookRegisterExternalChangeCallback
  - C callback
- And then what?
  - Revert to get the changes
  - Update the user interface

Notifications

- Register callback
  - ABAddressBookRegisterExternalChangeCallback
  - C callback
- And then what?
  - Revert to get the changes
  - Update the user interface

```
// Recipe Detail View Controller
ABAddressBookRegisterExternalChangeCallback(ab, abChanged, self);
```

Notifications

- Register callback
  - ABAddressBookRegisterExternalChangeCallback
  - C callback
- And then what?
  - Revert to get the changes
  - Update the user interface

```
// Recipe Detail View Controller
ABAddressBookRegisterExternalChangeCallback(ab, abChanged, self);
```

```
void abChanged(ABAddressBookRef ab, CFDictionaryRef info, void
*context) {
    ABAddressBookRevert(ab);
    [(RecipeDetailViewController*)context reloadData];
}
```

More on Change Callbacks

- Revert or ignore the changes
- Threading and change callbacks

Summary

- Low level C API for dealing with people
- View controllers for presenting the people

Questions?