Using The Address Book

or: How I Learned To Stop Worrying And Love The Address Book

Why use the Address Book?

- Using the address book gives you an easy way to access and use data your users already have.
- Make up for things Contacts can’t do.
- Extend what your existing application can do.

Two Frameworks

- **AddressBook** allows for direct manipulation of the address book.
- **AddressBookUI** Set of view controllers which allow picking, adding new, editing, or viewing contacts.

AddessBook

- Four basic data types: Address books, Records, Single-value properties, and Multivalue properties.
- C based, follows Core Foundation rules for memory management.

Address Books

- Stored as an ABAddressBookRef.
- Used to get information about and edit the address book directly.
- One per thread.
- Changes must be saved.

AddressBook Functions

- **ABAddressBookCreate**: Create a new address book.
- **ABAddressBookSave**: Commit any changes that you've made to the address book.
- **ABAddressBookRevert**: Throw away unsaved changes.
- **ABAddressBookAddRecord**: Add or remove a record from the address book. Both take an ABRecordRef.
- **ABAddressBookRegisterExternalCallBack**: Use to receive a call when the address book is changed by another app, or another thread in your app.

San Jose, CA
360|iDev slide to rock
March 2-4, 2009

Records

- Both people and groups are stored as ABRecordRef’s.
- Find out the type by using the function **ABRecordGetRecordType**, which returns an kABGroupType or kABPersonType.
- Do not pass a record address threads, ask for the records identifier and use that instead.
- Use **ABRecordGetRecordID** to get the unique ID from a group or person record.
- **ABRecordCopyValue**, **ABRecordSetValue**, **ABRecordRemoveValue** are used to copy, set, or remove values from records.

Group Records

- ABGroupCreate
- ABGroupCopyArrayOfAllMembers
- ABGroupCopyArrayOfAllMembersWithSortOrdering
- ABGroupAddRecord, and ABGroupRemoveRecord
- ABAddressBookCopyArrayOfAllMembers

Person Records

- ABPersonCreate
- ABPersonGetTypeOfProperty
- ABPersonCopyLocalizedPropertyName
- ABPersonHasImageData, ABPersonCopyImageData, ABPersonRemoveImageData, and ABPersonSetImageData
- ABAddressBookComparePeopleByName
- ABAddressBookCopyPeopleWithName

Person Properties

kABPersonFirstNameProperty
kABPersonLastNameProperty
kABPersonMiddleNameProperty
kABPersonPrefixProperty
kABPersonSuffixProperty
kABPersonNicknameProperty
kABPersonFirstNamePhoneticProperty
kABPersonLastNamePhoneticProperty
kABPersonMiddleNamePhoneticProperty
kABPersonOrganizationProperty
kABPersonJobTitleProperty
kABPersonDepartmentProperty
kABPersonEmailProperty
kABPersonBirthdayProperty
kABPersonNoteProperty
kABPersonCreationDateProperty
kABPersonModificationDateProperty
kABPersonAddressProperty

Displaying Names

- *ABRecordCopyCompositeName* will return a string for a group or person record with the correct formatting.
- *ABPersonCompositeNameFormat* can be used to determine first name or last name first name format.
- *ABPersonGetSortOrdering* tells if people records should be sorted by first or last name.

Getting Records

- **ABAddressBookGetGroupWithRecordID & ABAddressBookGetPersonWithRecordID** are used to get group or person records when you have the ID.
- You can ask the how many people or groups there are by using **ABAddressBookGetGroupCount** & **ABAddressBookGetPersonCount**
- **ABAddressBookCopyArrayOfAllGroups**
- **ABAddressBookCopyArrayOfAllPeople**

Single Value Properties

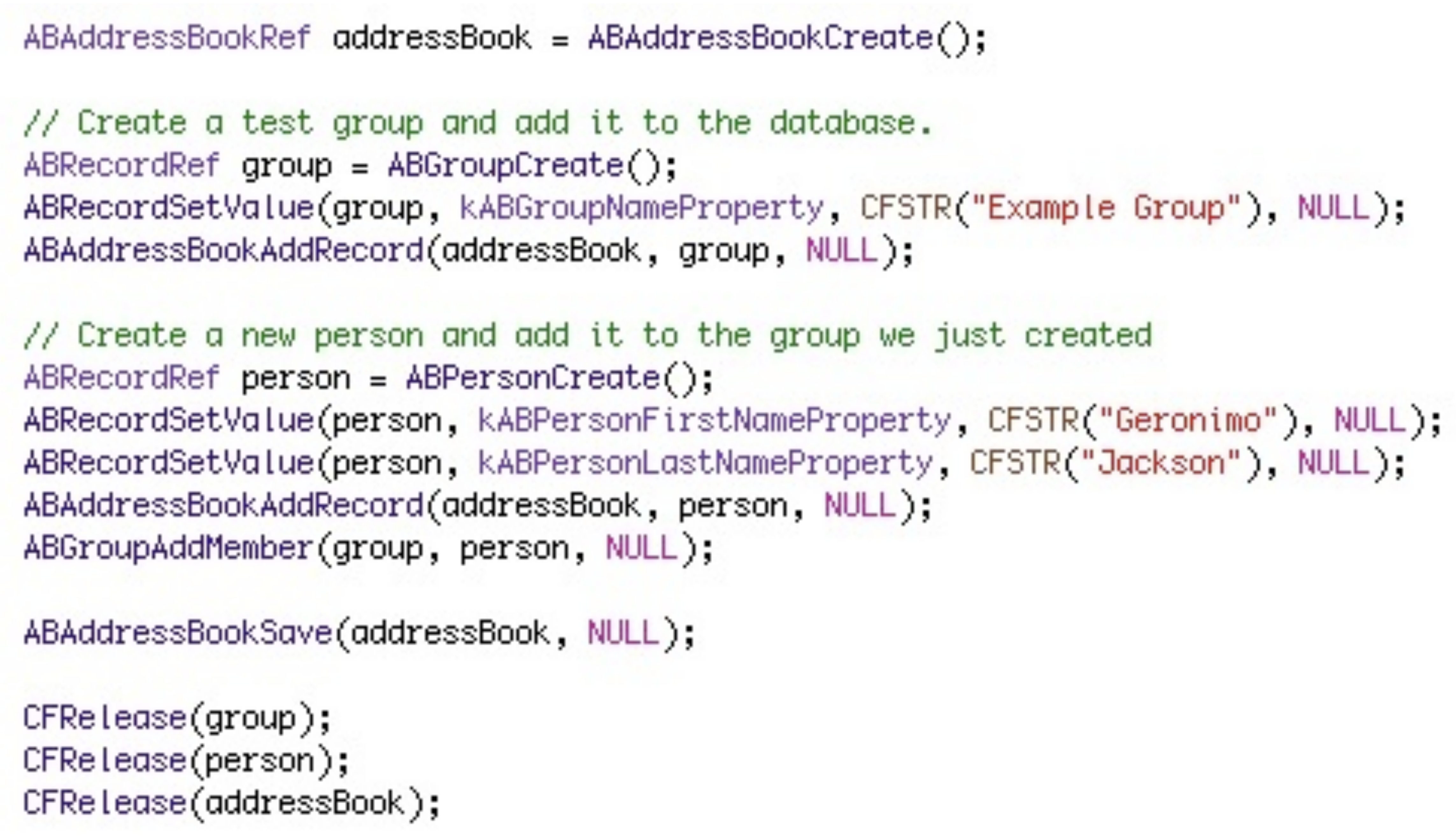
- Pass a property name and a value.
- *Example:*

```
ABRecordSetValue(person,
kABPersonFirstNameProperty,
CFSTR(“First Name”), &error);
```

San Jose, CA
360 iDev slide to rock
March 2-4, 2009

Example!

#import <AddressBook/AddressBook.h>



```
ABAddressBookRef addressBook = ABAddressBookCreate();

// Create a test group and add it to the database.
ABRecordRef group = ABGroupCreate();
ABRecordSetValue(group, kABGroupNameProperty, CFSTR("Example Group"), NULL);
ABAddressBookAddRecord(addressBook, group, NULL);

// Create a new person and add it to the group we just created
ABRecordRef person = ABPersonCreate();
ABRecordSetValue(person, kABPersonFirstNameProperty, CFSTR("Geronimo"), NULL);
ABRecordSetValue(person, kABPersonLastNameProperty, CFSTR("Jackson"), NULL);
ABAddressBookAddRecord(addressBook, person, NULL);
ABGroupAddMember(group, person, NULL);

ABAddressBookSave(addressBook, NULL);

CFRelease(group);
CFRelease(person);
CFRelease(addressBook);
```

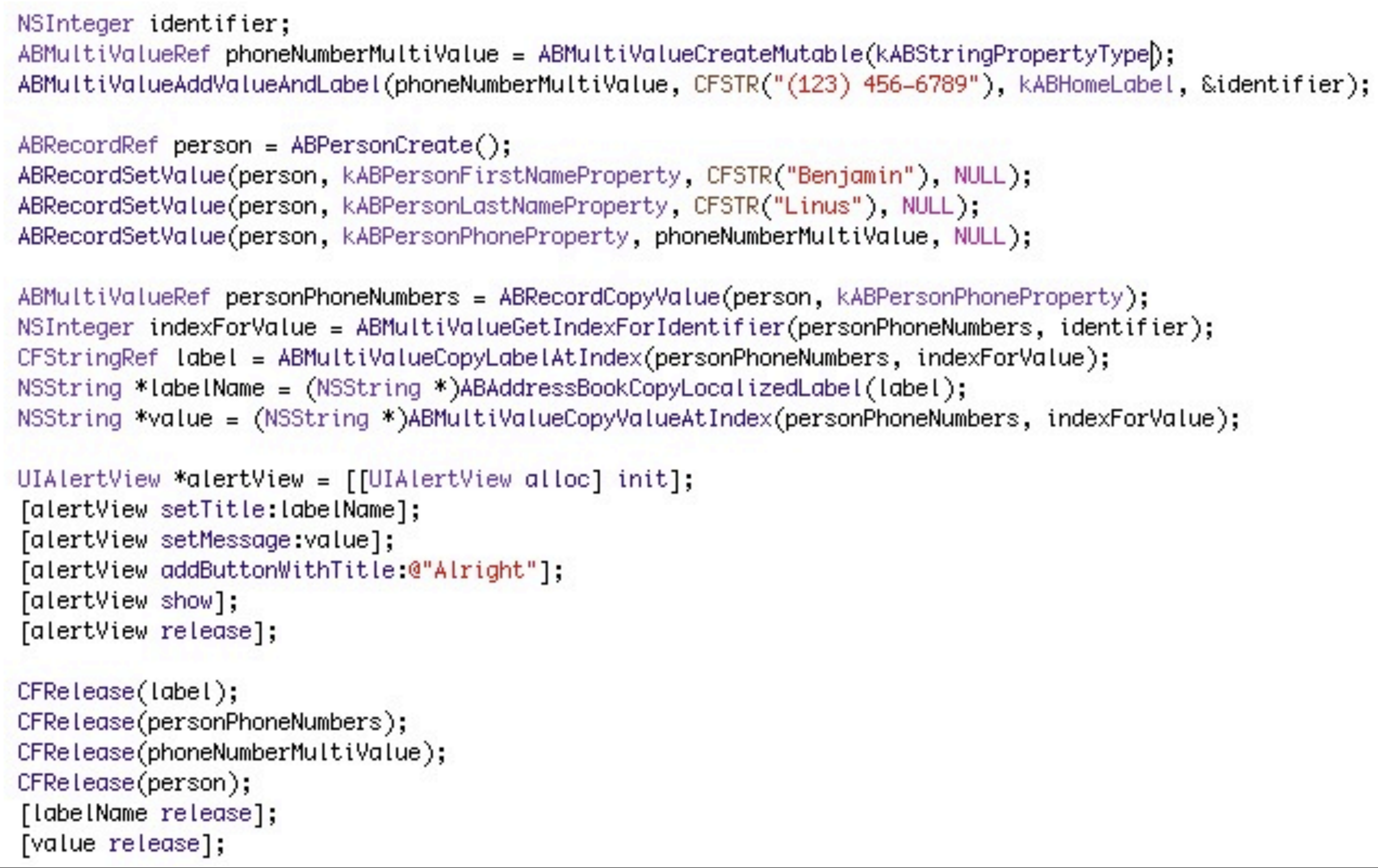
Multivalue properties

- Things like phone numbers, email addresses, where a single contact may have multiple.
- List of values, each of which has a text label and an identifier.
- Functions used to read are,
  ABMultiValueCopyLabelAtIndex
  ABMultiValueCopyValueAtIndex
  ABMultiValueCopyArrayOfAllValues
  ABMultiValueGetIndexForIdentifier
  ABMultiValueGetIdentifierAtIndex

Mutable Multivalue Properties

- **ABMultiValueCreateMutableCopy** to create a mutable copy of a multivalue property.
- **ABMultiValueCreateMutable** to construct your own.
- Functions for editing mutable multivalue properties
  ABMultiValueAddValueAndLabel
  ABMultiValueInsertValueAndLabelAtIndex
  ABMultiValueReplaceValueAtIndex
  ABMultiValueRemoveValueAndLabelAtIndex

Another Example!



```
NSInteger identifier;
ABMultiValueRef phoneNumberMultiValue = ABMultiValueCreateMutable(kABStringPropertyType);
ABMultiValueAddValueAndLabel(phoneNumberMultiValue, CFSTR("(123) 456-6789"), kABHomeLabel, &identifier);

ABRecordRef person = ABPersonCreate();
ABRecordSetValue(person, kABPersonFirstNameProperty, CFSTR("Benjamin"), NULL);
ABRecordSetValue(person, kABPersonLastNameProperty, CFSTR("Linus"), NULL);
ABRecordSetValue(person, kABPersonPhoneProperty, phoneNumberMultiValue, NULL);

ABMultiValueRef personPhoneNumbers = ABRecordCopyValue(person, kABPersonPhoneProperty);
NSInteger indexForValue = ABMultiValueGetIndexForIdentifier(personPhoneNumbers, identifier);
CFStringRef label = ABMultiValueCopyLabelAtIndex(personPhoneNumbers, indexForValue);
NSString *labelName = (NSString *)ABAddressBookCopyLocalizedLabel(label);
NSString *value = (NSString *)ABMultiValueCopyValueAtIndex(personPhoneNumbers, indexForValue);

UIAlertView *alertView = [[UIAlertView alloc] init];
[alertView setTitle:labelName];
[alertView setMessage:value];
[alertView addButtonWithTitle:@"Alright"];
[alertView show];
[alertView release];

CFRelease(label);
CFRelease(personPhoneNumbers);
CFRelease(phoneNumberMultiValue);
CFRelease(person);
[labelName release];
[value release];
```

Street Addresses

- Multivalue of dictionaries.
- Each address component is a key/value pair.

San Jose, CA
360|iDev slide to rock
March 2-4, 2009

Street Address Example

```
NSDictionary *dictionary = [[NSDictionary alloc] initWithObjects:[NSArray arrayWithObjects:@"1234 J Street", @"Sacramento", @"CA", @"95820", nil]
                                                         forKeys:[NSArray arrayWithObjects:
                                                                  (NSString *)kABPersonAddressStreetKey,
                                                                  (NSString *)kABPersonAddressCityKey,
                                                                  (NSString *)kABPersonAddressStateKey,
                                                                  (NSString *)kABPersonAddressZIPKey, nil]];

NSInteger identifier;
ABMultiValueRef multiValue = ABMultiValueCreateMutable(kABDictionaryPropertyType);
ABMultiValueAddValueAndLabel(multiValue, (CFDictionaryRef *)dictionary, kABWorkLabel, &identifier);

ABRecordRef person = ABPersonCreate();
ABRecordSetValue(person, kABPersonFirstNameProperty, CFSTR("Saul"), NULL);
ABRecordSetValue(person, kABPersonLastNameProperty, CFSTR("Tigh"), NULL);
ABRecordSetValue(person, kABPersonAddressProperty, multiValue, NULL);
ABRecordCopyValue(person, kABPersonAddressProperty);

NSInteger addressIndex = ABMultiValueGetIndexForIdentifier(multiValue, identifier);
NSDictionary *addressDictionary = (NSDictionary *)ABMultiValueCopyValueAtIndex(multiValue, addressIndex);

for (id key in addressDictionary) {
    NSLog(@"%@ = %@\n", key, [addressDictionary objectForKey:key]);
}

[addressDictionary release];
[dictionary release];
CFRelease(person);
CFRelease(multiValue);
```

AddressBookUI

- View controllers for many common address book related functions.
- 4 controllers:
  - ABPeoplePickerNavigationController
  - ABPersonViewController
  - ABNewPersonViewController
  - ABUnknownPersonViewController

San Jose, CA
360 iDev slide to rock
March 2-4, 2009

Picking People

- ABPeoplePickerNavigationController presents a very similar view to Contacts.
- Present as a modal view controller and implement the delegate methods.

ABPeoplePickerNavigationControllerDelegate Methods
peoplePickerNavigationControllerDidCancel:
peoplePickerNavigationController:shouldContinueAfterSelectingPerson:
peoplePickerNavigationController:shouldContinueAfterSelectingPerson:property:identifier:

Viewing/Editing Person Records

- ABPersonViewController will display and allow editing of person records.
- Will call *personViewController:shouldPerformDefaultActionForPerson:property:identifier:*

  On its delegate when the user taps a property.

Creating New Person Records

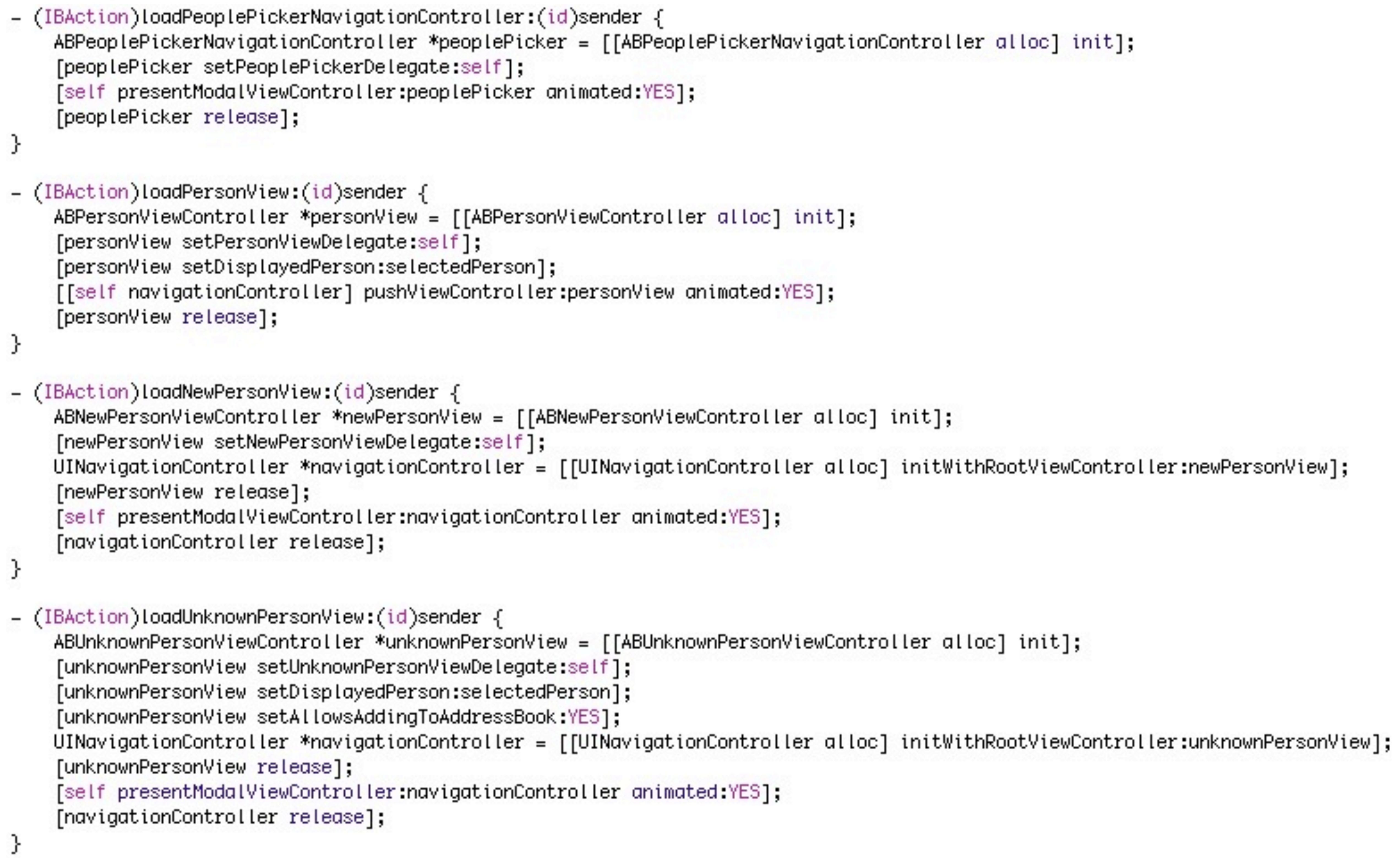
- *ABNewPersonViewController* allows users to create new people and add them to the database with almost no code.
- When finished, it will call *newPersonViewController:didCompleteWithNewPerson:*

Unknown Person Controller

- Add to or create a new person with existing data.
- Create person with known information, use setDisplayedPerson:
- Will call *unknownPersonViewController:didResolveToPerson:* on it's delegate when finished.

San Jose, CA
360|iDev slide to rock
March 2-4, 2009

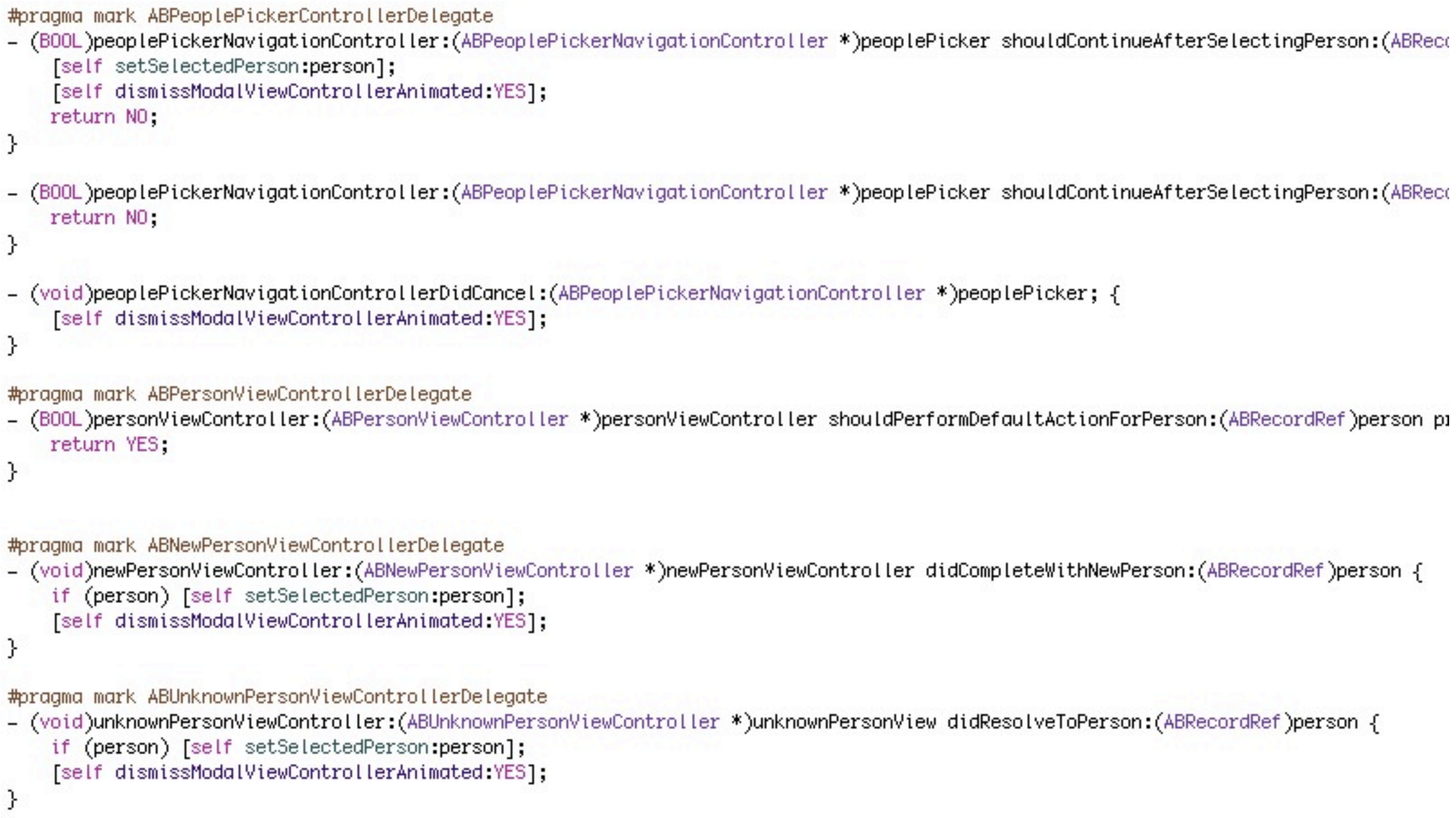
Loading...



```
- (IBAction)loadPeoplePickerNavigationController:(id)sender {
    ABPeoplePickerNavigationController *peoplePicker = [[ABPeoplePickerNavigationController alloc] init];
    [peoplePicker setPeoplePickerDelegate:self];
    [self presentModalViewController:peoplePicker animated:YES];
    [peoplePicker release];
}

- (IBAction)loadPersonView:(id)sender {
    ABPersonViewController *personView = [[ABPersonViewController alloc] init];
    [personView setPersonViewDelegate:self];
    [personView setDisplayedPerson:selectedPerson];
    [[self navigationController] pushViewController:personView animated:YES];
    [personView release];
}

- (IBAction)loadNewPersonView:(id)sender {
    ABNewPersonViewController *newPersonView = [[ABNewPersonViewController alloc] init];
    [newPersonView setNewPersonViewDelegate:self];
    UINavigationController *navigationController = [[UINavigationController alloc] initWithRootViewController:newPersonView];
    [newPersonView release];
    [self presentModalViewController:navigationController animated:YES];
    [navigationController release];
}

- (IBAction)loadUnknownPersonView:(id)sender {
    ABUnknownPersonViewController *unknownPersonView = [[ABUnknownPersonViewController alloc] init];
    [unknownPersonView setUnknownPersonViewDelegate:self];
    [unknownPersonView setDisplayedPerson:selectedPerson];
    [unknownPersonView setAllowsAddingToAddressBook:YES];
    UINavigationController *navigationController = [[UINavigationController alloc] initWithRootViewController:unknownPersonView];
    [unknownPersonView release];
    [self presentModalViewController:navigationController animated:YES];
    [navigationController release];
}
```

San Jose, CA
360 iDev slide to rock
March 2-4, 2009

Using



```
#pragma mark ABPeoplePickerControllerDelegate
- (BOOL)peoplePickerNavigationController:(ABPeoplePickerNavigationController *)peoplePicker shouldContinueAfterSelectingPerson:(ABRec
    [self setSelectedPerson:person];
    [self dismissModalViewControllerAnimated:YES];
    return NO;
}

- (BOOL)peoplePickerNavigationController:(ABPeoplePickerNavigationController *)peoplePicker shouldContinueAfterSelectingPerson:(ABRec
    return NO;
}

- (void)peoplePickerNavigationControllerDidCancel:(ABPeoplePickerNavigationController *)peoplePicker; {
    [self dismissModalViewControllerAnimated:YES];
}

#pragma mark ABPersonViewControllerDelegate
- (BOOL)personViewController:(ABPersonViewController *)personViewController shouldPerformDefaultActionForPerson:(ABRecordRef)person pı
    return YES;
}


#pragma mark ABNewPersonViewControllerDelegate
- (void)newPersonViewController:(ABNewPersonViewController *)newPersonViewController didCompleteWithNewPerson:(ABRecordRef)person {
    if (person) [self setSelectedPerson:person];
    [self dismissModalViewControllerAnimated:YES];
}

#pragma mark ABUnknownPersonViewControllerDelegate
- (void)unknownPersonViewController:(ABUnknownPersonViewController *)unknownPersonView didResolveToPerson:(ABRecordRef)person {
    if (person) [self setSelectedPerson:person];
    [self dismissModalViewControllerAnimated:YES];
}
```

Contact Me

- Collin Donnell (iPhone Developer)
- Email: collindonnell@gmail.com
- Twitter: collindonnell